PERFORMANCE INDEX ANALYSIS OF CONTROL SYSTEMS SYNTHESIZED FOR A COUNTER-CURRENT PROCESS

This dissertation has been submitted to the Department of Electrical Engineering, University of Cape Town, in fulfilment of the requirements for a M.Sc. Degree in Electrical Engineering.

by

Anton de Waal

April 1990

ACKNOWLEDGMENTS

I wish to express my sincere gratitude to the following:

The Council for Mineral Technology (MINTEK) for providing the funding for this research and for my subsistence over the past two years.

Prof. Martin Braae for the supervision and guidance which he has afforded me for the duration of this project.

Dr. Ian Barker of MINTEK for his useful and relevant input and for the interest which he has shown in this project.

Mr. Heinz Geise of MINTEK for his assistance in the aquisition of the equipment for the heat exchanger.

Mr. David Kenyon and especially Mr. Peter Lewis of the Electrical Workshop at UCT for their assistance and for the use of their facilities.

Dr. Duncan Fraser of the Chemical Engineering Department at UCT for his guidance in the heat exchanger design phase.

Pauline, for her patience, support and encouragement.

ABSTRACT

Indices which quantitatively predict the ability of a particular MIMO control system design to meet certain performance objectives are discussed in this dissertation. These performance objectives include:

- Input and output rejection of noise and disturbances.
- Input and output insensitivity to changes in the process.
- Good output regulation and minimal control effort.

A counter-current heat exchanger process (simulating the carbon-in-pulp gold extraction process) has been designed and constructed. The quality of control systems synthesized for this process is gauged using these performance indices. The control of this large-scale process has been decentralized to reduce the design complexity. The strategy adopted is to regulate the process flows and levels using a cascade compensator, and then to use these setpoints as inputs to control the temperatures at various points on the heat exchange cascade.

Standard PI-type controllers were designed for the regulation of the flow and level outputs. The outputs were regulated accurately in the presence of process disturbances, and the relatively slow level dynamics were improved considerably.

Performance index analysis of control systems is achieved by observing frequency-dependent plots of the singular values of

certain characteristic matrices describing the system. These matrices are obtained as ratios of the plant and controller matrices.

A CAD software package has been developed to enable this type of analysis on state-space control systems with state observers. The package enables the synthesis of LQG optimal control systems and the quantitative assesment of the ability of such systems to meet their design objectives.

A package has been developed to run the heat exchanger and to implement control systems for the process.

Two LQG control systems were synthesized for the process in which the two flow inputs (setpoints) were used to regulate the two outlet temperature outputs. The performance of the systems were predicted quantitatively using performance indices. The systems were tested by digital simulation. Good correlation between the predicted and simulated performance was observed. One of the systems was successfully implemented on the heat exchanger process, and the controlled system revealed that the predictions made by the indices were indeed accurate.

This analysis technique is a powerful indicator of the general performance of a MIMO control system. It also provides a means of analysing the frequency-dependent performance of state-space control systems.

TABLE OF CONTENTS

LIST OF ILLUSTRATIONS

CHAPTER SIX

APPENDIX H

APPENDIX I

APPENDIX J

APPENDIX K

APPENDIX O

APPENDIX Q

APPENDIX R

LIST OF TABLES

APPENDIX C

APPENDIX D

APPENDIX E

APPENDIX F

APPENDIX G

APPENDIX H

APPENDIX O

APPENDIX Q

LIST OF PHOTOGRAPHS

NOMENCLATURE

Symbol	Meaning
ADC	Analog to Digital Converter
a..b	a through b
$A(s)^*$	Complex Conjugate of Matrix A(s)
BIM	Binary Interaction Matrix
$B(G_c(s))$	Binary Interaction Matrix of transfer function matrix $G_c(s)$
CACSD	Computer Aided Control System Design
CH	Cold-to-Hot
CIP	Carbon-In-Pulp
CL	Characteristic Loci
C/L	Closed Loop
C_n	Complex n-dimensional Linear Space
C^{nxn}	Mapping from a Complex n-dimensional Linear Space to another Complex n-dimensional Linear Space
C^n_Q	Given a positive definite matrix $Q \in C^{nxn}$, define a unitary space C^n_Q as the set of complex n-vectors $X(s) = (x_1(s),....,x_n(s))^T$ together with Euclidean norm $\|\cdot\|_Q$ defined in (4.34)
DAC	Digital to Analog Converter
Difft	Difference in Temperature
D(s)	Vector of Process Disturbances
E(s)	Vector of Errors

eig(**A(s)**)	Eigenvalues of Matrix **A(s)**
F(s)	Feedback Compensator Transfer Function Matrix
gr(s)	Gain Ratio
$G_c(s)$	Control Valve Input to Flow and Level Output Transfer Function Matrix
$G_{ft}(s)$	Flow Input to Temperature Output Transfer Function Matrix
gm	Gain Margin
G(s)	Process Transfer Function Matrix
$G_m(s)$	Modified Transfer Function Matrix (Including Cascade Compensator)
HC	Hot-to-Cold
HERIG	Heat Exchanger Rig Running and Analysis Package
INA	Inverse Nyquist Array
K(s)	Cascade Compensator Transfer Function Matrix
$K_c(s)$	Cascade Compensator Transfer Function Matrix (used in control of flows and levels)
K_{ij}	Open Loop Gain
LMTDiff	Log Mean Temperature Difference
LQ	Linear Quadratic
LQG	Linear Quadratic Gaussian
$L_1(s)$	Return Matrix at Output Node (= **M(s)G(s)**)
$I + L_1(s)$	Return Difference Matrix at Output Node
$I + L_1^{-1}(s)$	Inverse Return Difference Matrix at Output Node
$L_2(s)$	Return Matrix at Input Node (= **G(s)M(s)**)
$I + L_2(s)$	Return Difference Matrix at Input Node
$I + L_2^{-1}(s)$	Inverse Return Difference Matrix at Input Node

M_{av}	Average Magnitude
Mean Err	Mean Error
MIMO	Multi Input Multi Output (Multivariable)
Mp	Predicted Response Ratio Maxima
MR_{max}/A_{pert}	Actual Maximum Response Ratio
Mt	Tested response Ratio Maxima
M(s)	Feedback Loop Transfer Matrix (= **K(s)F(s)**)
N(s)	Vector of Noise of Output Sensors
OPTCAD	Optimal LQG Control System Synthesis and Performance Index Analysis Package
O/L	Open Loop
P(s)	Precompensator Transfer Function Matrix
$\mathbf{P_c(s)}$	Additional Cascade Precompensator Transfer Function Matrix used for LQG Control System
PI	Proportional-Integral
rad/sec	Radians per Second
RHP	Right Halfplane
R(s)	Vector of Process Setpoints
SISO	Single Input Single Output (Singlevariable)
$\mathbf{S_i(s)}$	Characteristic Matrices $(\mathbf{I} + \mathbf{L_i(s)})^{-1}$
T_{ij}	Dead-time
T_{ij}	Response-time
$\mathbf{T_i(s)}$	Characteristic Matrices $\mathbf{I - S_i(s)} = \mathbf{L_i(s)(I + L_i(s))^{-1}} = \mathbf{(I + L_i(s))^{-1}L_i(s)}$
UCT	University of Cape Town
U(s)	Vector of Process Inputs
Y(s)	Vector of Process Outputs
w_{osc}	Frequency of Oscillation

$\mathbf{W(s)}$	Characteristic Matrix $\mathbf{S_1(s)M(s) = M(s)S_2(s)}$
$\mathbf{Z(s)}$	Vector of Process Output Measurements
3φ	Three Phase
φ_M	Phase Margin

INTRODUCTION

The Heat Exchanger Counter-current Process

A heat exchanger counter-current process has been designed and constructed. This pilot plant is housed in the Control Laboratory at the University of Cape Town (Department of Electrical and Electronic Engineering).

The heat exchanger process was designed to simulate the carbon-in-pulp (CIP) counter-current process. In the CIP process, gold in pulp is absorbed onto activated charcoal. The gold-laden charcoal (product of the CIP process) is then eluted (stripped) and gold is finally electrowon from the strip liquor.

The sponsors of this M.Sc. project, the Council for Mineral Technology (MINTEK), in collaboration with the South African gold-mining industry, have undertaken a major research and development program over the past 15 years aimed at the development of the CIP process. In particular, the Measurement and Control Division at MINTEK has been concerned with (amongst other aspects) the engineering of suitable process-control strategies for implementation on such gold-extraction plants.

Before engineering a control structure for the CIP process, it would be instructive to gain some experience in the

control of counter-current processes. The heat exchanger process has been constructed with this objective in mind. Analysis and control of this process would give the designer experience in the control of counter-current processes. This experience could then be drawn from in the design of a controller structure for the CIP process (as well as for other counter-current processes).

The actual CIP and heat exchange processes are discussed and compared in chapter 1. The heat exchanger design and construction details are also described. The chapter ends with the comparison between the designed and observed temperature distributions across the process. The effects on this distribution of changes in the process stream flowrates are also observed.

Control Structure for the Heat Exchanger Process

The structure of a large-scale system, such as the heat exchanger system, needs to be considered carefully before deciding on an appropriate controller structure. Decentralized control schemes can be applied to the control of large-scale systems, in which the process model is divided up into subsystems and stable controllers are designed independently for each of these subsystems. This simplifies the design process, although some interaction between the subsystems generally does exist.

The structure of the heat exchanger process is analysed in chapter 2 using the binary interaction matrix. The analysis points to the implementation of the following control structure for the process:

(i) Cascade compensator for Flows and Levels

A cascade compensator is proposed for the closed loop control of the process subsystem relating the flow and level outputs to the control valve inputs.

ii) Full Controller for Temperatures

The flow and level setpoints for the cascade compensated system are treated as inputs to a modified heat exchanger process. A full controller structure is proposed for temperature outputs using these inputs as well as the remaining stirring rate inputs.

Design and Implementation of Flow and Level Controllers

The design of controllers for the flow and level outputs is described in chapter 3. The structure of the process sybsystem model relating these outputs to the control valve inputs is analysed further to decide on a suitable structure

for the cascade controller. The process model is determined and a control system implemented which was designed with the following objectives in mind:

(i) Regulation of Flow and Level Outputs

(ii) Improvement of Level Response Dynamics

(iii) Rejection of Disturbances between Loops in System

(iv) Closed Loop Stability

The closed loop performance of the resultant system is then analysed at the end of chapter 3.

Performance Index Analysis of MIMO Control Systems

A quantitative measure of the quality of each of the following feedback properties of multivariable control systems needs to be obtained:

(i) Sensor Noise Rejection by Inputs and Outputs

(ii) Process Disturbance Rejection by Inputs and Outputs

(iii) Insensitivity of Inputs and Outputs to Changes in the Process Model

(iv) Stability Margins for Closed Loop System

A quantitative representation is also required for the quality of the setpoint tracking by the outputs and for the effects that setpoint variations have on the inputs (control effort).

These frequency-dependent measures of the quality of a control system design are defined as performance indices. The performance of a multivariable control system can be predicted by considering performance index plots generated for the system.

Performance indices are developed in chapter 4 which gauge the performance of a control system in terms of the properties discussed. These indices are related to frequency-dependent plots of the singular values of system matrices derived from the process and controller models.

Performance index analysis can be applied to control systems derived using any multivariable control system design technique. The analysis of systems synthesized using LQG optimal control techniques is considered in this dissertation.

Chapter 4 also contains a discussion of LQG techniques, and the necessary relationships for the performance index analysis of such systems are developed.

CAD and Process Control Software Developed

The availability of computing power has made it possible to implement the computationally intensive frequency-domain performance index analysis technique in the form of a computer aided design (CAD) package.

A package for the LQG synthesis, performance index analysis and digital simulation of state-space control systems with observers has been developed. This was done in order to fulfill the requirements of the investigation into the application of performance index analysis techniques. A description of this package, referred to as OPTCAD, is given in chapter 5.

Chapter 5 also contains a description of the software package, named HERIG, developed for the running, control and analysis of the heat exchanger process.

Performance Index Analysis of Heat Exchanger Controllers

LQG control systems are to be synthesized to control the two outlet temperature outputs of the modified heat exchanger process using the two counter current process stream flow inputs. The performance of the systems can be predicted and compared by producing performance index analysis plots for each system.

The synthesis of trial control systems 1, 2 and 3 for the process is discussed in chapter 6. Systems 2 and 3 are analysed using performance indices, and quantitative predictions are made regarding the performance of these systems.

Systems 2 and 3 are implemented on a digital simulator, and system 2 on the heat exchanger process itself. Results, from tests done to determine the accuracy of predictions made by the performance index analysis for each system, are quoted and discussed in chapter 6.

CHAPTER ONE

THE HEAT EXCHANGER COUNTER-CURRENT PROCESS

1.1) Introduction

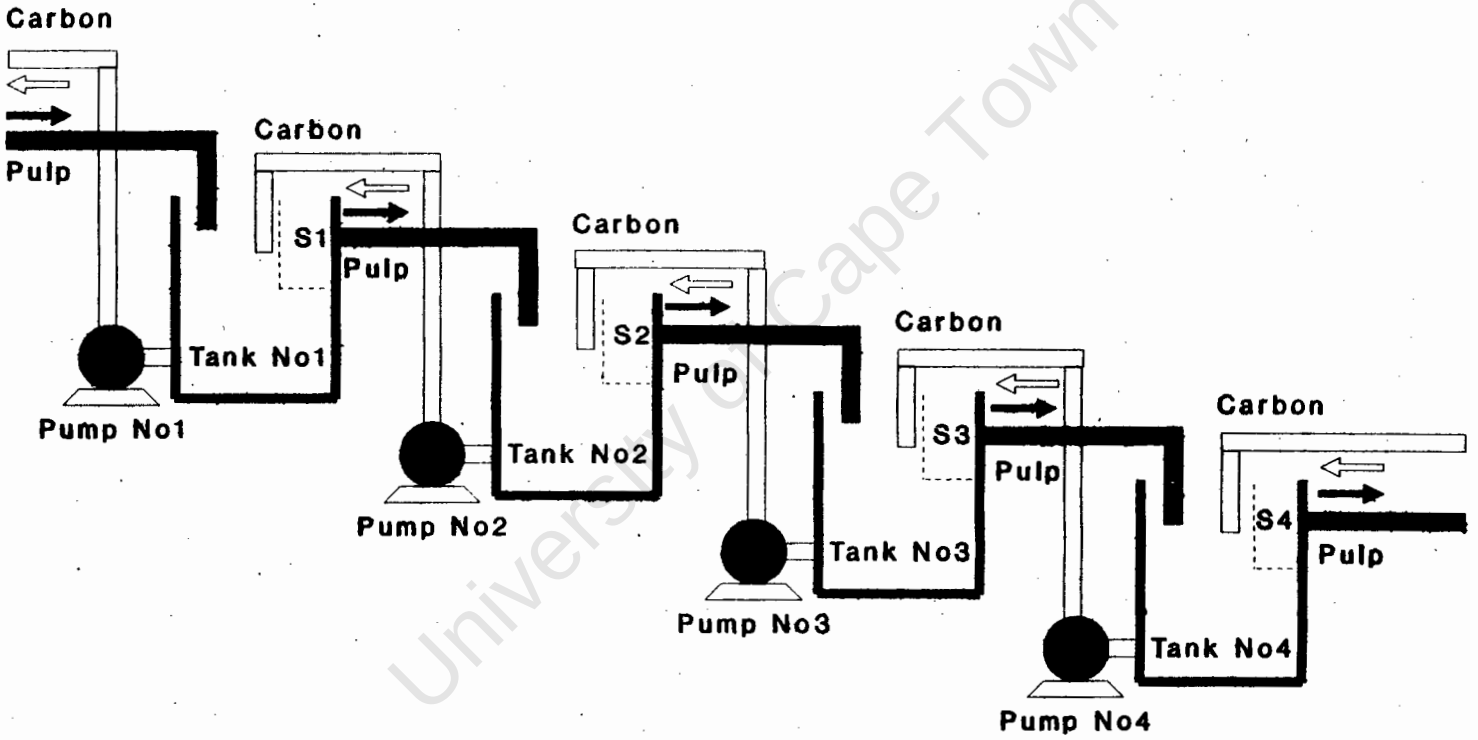
This chapter begins with a brief discussion of the actual CIP process, followed by a motivation for the choice of a heat exchanger to simulate the CIP process. A comparison of the two processes, in which their similarities and differences are outlined, is then presented. A section is then dedicated to the design and construction details of the heat exchanger. This section includes an input-output description of the process. This is followed by a section in which the observed temperature distributions across the pilot plant are compared to those assumed in the design phase. The effects of varying the process stream flowrates on the distributions are also observed and discussed. Concluding remarks are made at the end of the chapter.

1.2) Description of the Carbon-In-Pulp Process

The process of using carbon to recover gold in solution from the pulp is discussed in this section. Frequent reference is made to figure 1.1, which gives a schematic of the CIP process.

Figure 1.1

Schematic of Carbon-In-Pulp Cascade



In the CIP process, pulp is contacted in several fully mixed-state stages with carbon inside cascade absorption vessels (tanks No1 to No4). Gold, which is in solution in the pulp, is absorbed onto active carbon granules during each contacting stage. The granular carbon (typical diameter 1.2 to 2.4mm) is separated in the tanks from the finer pulp slurry (typical particle diameter < 0.65mm) by the screens S1 to S4. The pulp flows continuously from tank No1 to tank No4, while the carbon is pumped countercurrently from tank No4 to tank No1. The transfer of carbon up the cascade is intermittent and it is usually pumped across once a day (residence time is thus one day). The gold transfer process is enhanced by mechanical agitation of the vessels. The residence time of the pulp in the vessels is typically 60 to 80 minutes.

The concentration of gold in solution decreases as the pulp moves down the cascade, while the concentration of gold on the carbon increases as the carbon is pumped up the cascade. Typical concentrations of gold on the carbon pumped from tank No1 (product) vary between 4000 and 5000 grammes per ton (g/t). The concentration of gold in solution exhausted from tank No4 (barrens) is typically around 0.04 g/t.

The absorption characteristics of carbon in a CIP circuit, as well as the metallurgical performance of the plant are influenced by a number of factors. Some of these factors are listed below:

(i) Concentration of "free" cyanide in solution (sodium cyanide conc. optimally 100 to 150 g/t)

(ii) The pH of pulp (Optimal pH around 8)

(iii) Concentration of calcium ion in solution

(iv) Activity of carbon

(v) Distribution of carbon in the cascade

(vi) Carbon transfer strategies employed

Increasing amount of carbon in circuit or adding fresh high-active carbon to tank No4 lowers the concentration of gold on the barrens, but increases the carbon inventory and carbon retention time

The heat exchanger circuit has been chosen to simulate the CIP process. The next section gives a motivation for this choice, a discussion of the heat exchanger process, as well as a comparison of the two processes.

1.3) The Heat Exchanger System Chosen to Simulate the Carbon-in-Pulp Process

1.3.1) Counter-current processes Considered for the Simulation of the CIP Process

A suitable low-cost counter-current process is required to simulate the CIP process and its related control problems on a laboratory scale. A few processes considered for this purpose are listed and briefly commented on in the following paragraphs.

(i) Distillation Column

Although this is a counter-current process, it would not be possible to simulate the fully mixed-state stages found on the CIP process using a distillation column.

(ii) Rotary Kiln

Apart from being very impractical to implement on a laboratory scale, this counter-current process would also be unable to simulate the mixed-state stages found on the CIP process.

(iii) Ion Exchange Process

This process would simulate the CIP process well, but the costs of ion concentration measurement would be prohibitively high.

(iv) Convection along an Iron Bar

This counter-current process is simple, inexpensive and easy to implement on a laboratory scale, but also has no mixed-state stages.

(v) Water Heat Exchanger Process

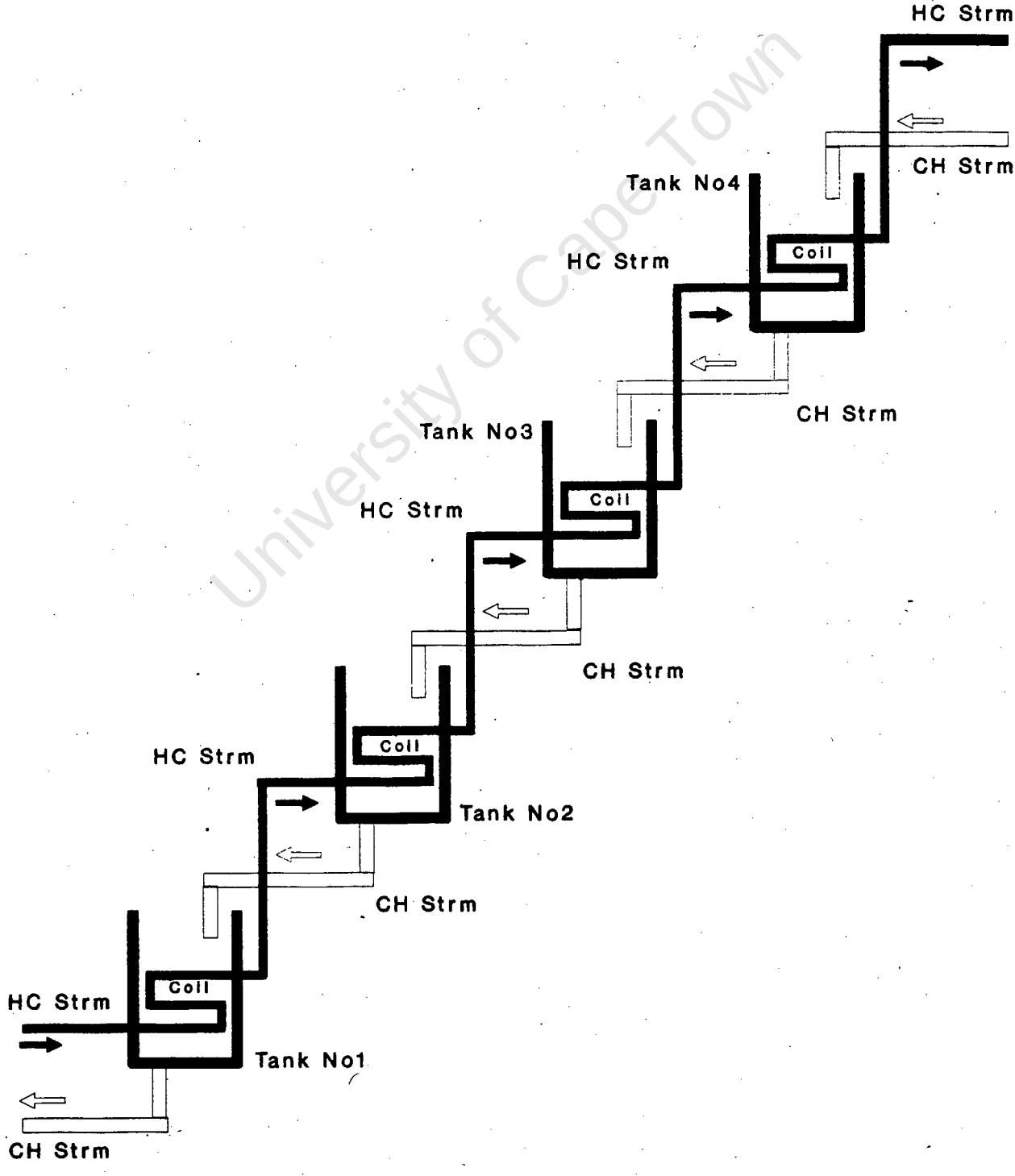
This is a counter-current process which is simple to design and implement on a laboratory scale, and can be designed with mixed-state stages as found on the CIP circuit.

The water heat exchanger process has been chosen to simulate the CIP process.

1.3.2) Description of the Heat Exchanger Process

Reference in the description of the heat exchanger is made to figure 1.2, which provides a schematic of the process.

Figure 1.2
Schematic of Heat Exchanger



In the heat exchanger process, the hot to cold (termed HC) stream is pumped continuously through the heating coils of tanks No1 to No4. The cold to hot (termed CH) stream moves countercurrently through tanks No4 to No1 under the force of gravity. The two streams are in thermal contact via the walls of the heating coils in each of the four exchange vessels. This contact in each vessel allows heat energy to move from the HC stream to the CH stream. The movement of energy takes place as a result of the temperature gradient between the two streams. The walls of the coil provide a physical (but not thermal) separation between the streams. Agitation of the water in each vessel ensures that a fully-mixed state occurs within the vessel. The residence time of CH stream in each vessel is dependent on its flowrate and on the volume of the vessel itself. The residence time of the HC stream in a particular vessel is dependent on its flowrate and on the volume of the coil through which it flows.

The temperature of the HC stream decreases as it ~~gains~~ *loses* thermal energy while moving from tank to tank, while the temperature of the CH stream increases as it moves in the opposite direction.

The amount of heat transferred between the streams is affected by (amongst others) the factors mentioned below:

(i) Thermal conductivity coil walls

(ii) Degree of agitation of water in the vessels

(iii) Temperature at the source of the CH stream

(iv) Routing and flowrate of the CH stream

(v) Points (iii) and (iv) for HC stream

1.3.3) Comparison of Heat Exchanger and CIP Processes

Figures 1.1 and 1.2 of the CIP and heat exchanger processes respectively will be referred to in the comparison of the two processes.

(i) Analogies between processes

The concentration of gold on the carbon is analogous to the temperature of the water in the CH stream. Similarly, the concentration of gold in solution in the pulp is analogous to the temperature of the water in the HC stream.

The increase in the gold concentration on the carbon as it moves through the reactors is analogous to the increase in the temperature along the CH stream. A similar analogy exists between the pulp and the HC stream.

A gold concentration gradient exists between the CIP streams, which causes the gold to move from the pulp to the carbon. This gradient is analogous to the temperature gradient existing between the heat exchanger streams, which causes a move of thermal energy.

The residence time of the CIP carbon stream (24 hours) is approximately 18 to 24 times longer than that of the pulp stream (60 to 80 minutes). On the heat exchanger, the residence time of the CH stream is proportional to the volume of a vessel. The residence time for the HC stream is proportional the interior volume of the submerged coil. Since the volume of a vessel is far greater than that of the interior of a coil, a similar relationship in residence times exists on the heat exchanger process as does on the CIP process (provided that the flowrates in the two heat exchanger streams are of the same order of magnitude).

Finally, an analogy can be drawn between the factors which affect the absorption characteristics and metallurgical performance of the CIP process and those which affect the amount of heat transferred on the heat exchanger process.

(ii) Differences between processes

The transfer of carbon on CIP plants usually takes place as a batch process once a day, and not continuously throughout the day. The heat exchanger shown in figure 1.2 has control valves C1 to C5 to regulate the flow of the CH stream between the vessels. Under normal operation, the CH stream flows continuously between the vessels, and is not transferred in batch as is done with the carbon. This continuous mode of operation is chosen in order to implement linear control and optimization techniques on the process.

A special case of the continuous mode of operation would be to open or shut the control valves completely. The transfer of the CH stream would become a batch process as for the transfer of carbon. Linear control theory can, however not be applied to the control of batch processes. The batch operation of the CIP and heat exchanger systems can thus not be optimized using these control techniques. Little would be learned about the use of control techniques in the optimization of counter-current processes. One of the motivations for the construction of the heat exchanger is to gain knowledge on the control of counter-current processes, which could be applied to the control of the CIP process.

The continuous mode of operation, although not currently the mode of operation on CIP processes, has thus been chosen because of this objective. An interesting topic for future research on the heat exchanger could be a comparison of the respective efficiencies obtained when running under batch or continuous modes of operation. This could assist in assessing whether the current batch mode of operation on CIP plants is indeed the most efficient. Such research could reveal that there are indeed significant advantages to transferring carbon continuously in CIP processes.

1.4) Design and Construction of Heat Exchanger Process

The design details of the heat exchanger process are chosen with the following design objectives in mind:

(i) Cost efficient design

(ii) Analogy with carbon-in-pulp process

(iii) Ease of practical laboratory implementation

1.4.1) Design Specifications

A block diagram of the heat exchanger process is given in figure 1.3, and is referenced in the discussion throughout this section. The temperatures at each point T1 to T10 on

the process are indicated, as well as the temperature change in a particular stream (flows F1 and F2) across each reactor (eg a positive 25°C change in the temperature of the CH stream across tank No4 is represented by "> +25 >").

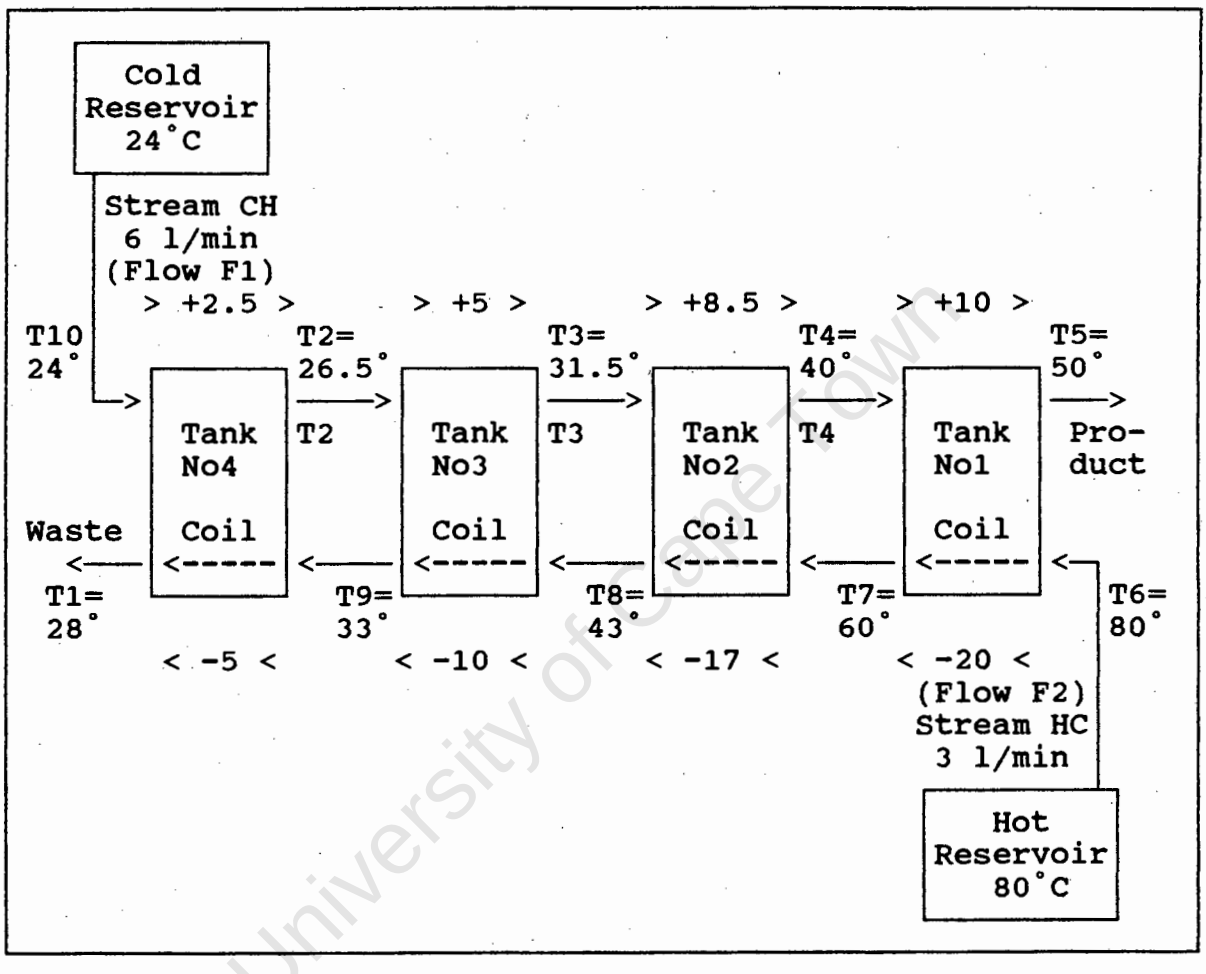


Figure 1.3: Block Diagram of Heat Exchanger Process

In the CIP process shown in figure 1.1, the concentration of gold at the end of the pulp stream (barrens) is to be kept at a minimum. This avoids excessive loss of gold still in solution in the barren pulp when it is discarded. The concentration of gold in the barrens is thus only slightly higher than the concentration on the incoming carbon.

To simulate this condition on the heat exchanger process, the HC stream is cooled more than the CH stream is heated. The specified design waste temperature can thus be chosen to be only slightly higher than the temperature of the incoming CH stream from the cold reservoir.

In order to cool the HC stream more than the CH stream is heated, the flowrate ratio is chosen as:

(HC Stream Flowrate) / (CH Stream Flowrate) = 1/2 (1.1)

Under these specified conditions, the temperature changes across each reactor for the HC stream is double that for the CH stream, as seen in figure 1.3.

The concentration gradient between the streams of the CIP process decreases from tank No1 to No4. As a result, less gold is transferred between the streams when moving from tank No1 to No4. The analogous situation occurs on the heat exchanger and is reflected in the lower temperature gradients observed moving from tank No1 to No4.

1.4.2) Summary of Heat Exchanger Design

Figure 1.4 shows the configuration of the heat exchanger plant, which has been designed bearing in mind the design objectives listed in 1.4. The actual design procedure is broken down and discussed in the paragraphs that follow.

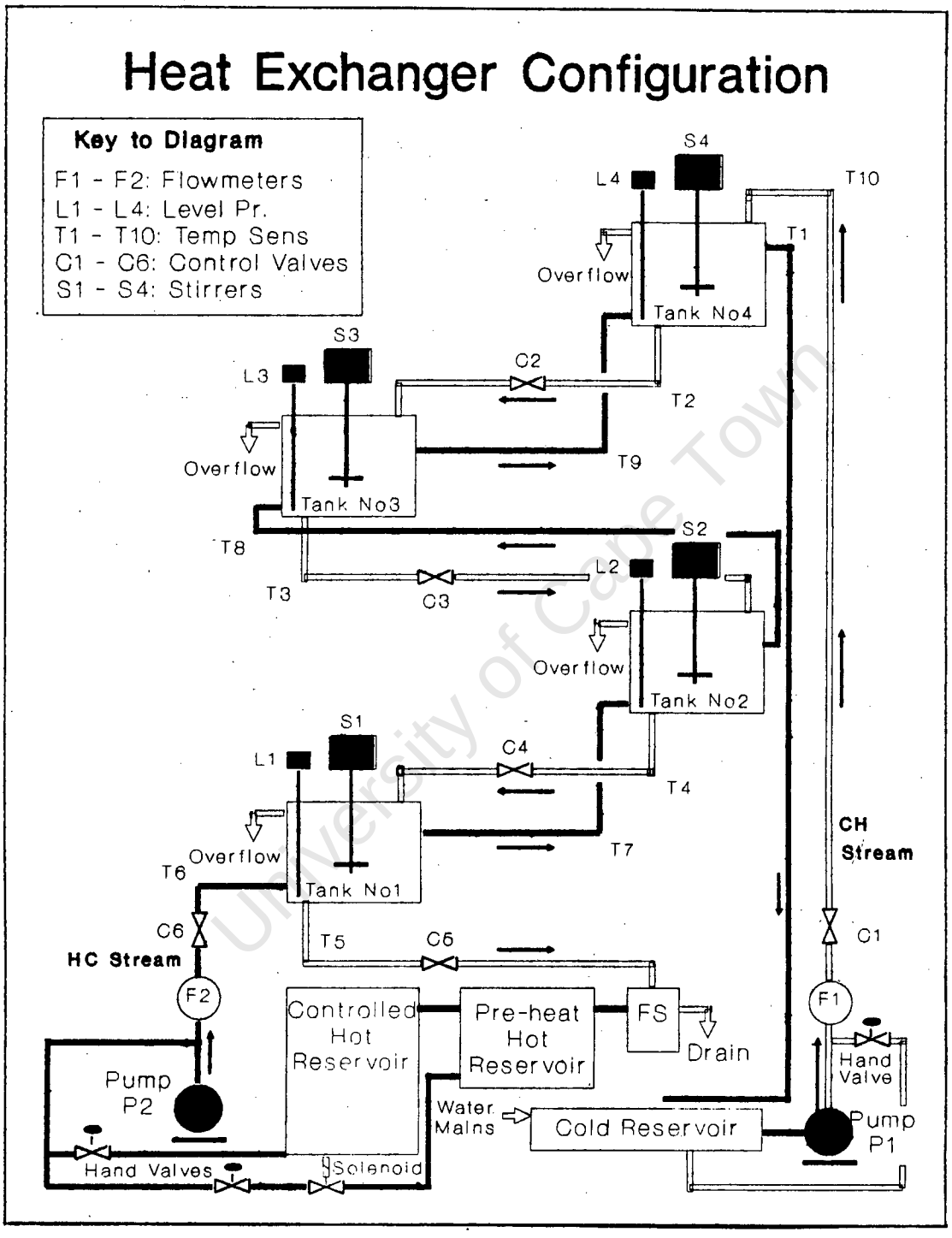


Figure 1.4: Configuration Adopted for Heat Exchanger

(i) Design of flowrates and reservoirs

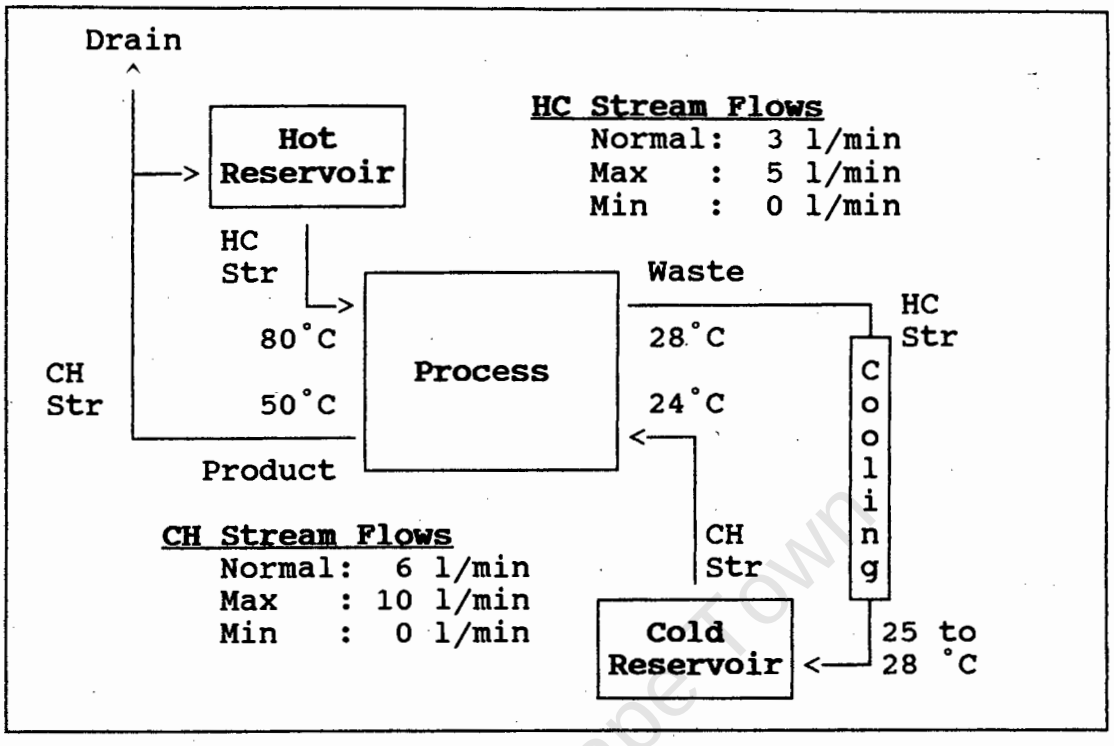


Figure 1.5: Thermal Characteristics of Process

The designed flowrates were chosen after careful consideration of the thermal energy balance for the system as a whole and of the cost of flow measurement. The schematic in figure 1.5 shows the thermal characteristics of the process and indicates the normal, maximum and minimum designed flowrates.

The cost of measuring flowrates below 1 l/min accurately exceeds R3000 per flow measurement set, a figure beyond the limited budget for this project. Higher flowrates were chosen for this reason. High flowrates of heated water, on the other hand, require excessive heating power (especially for the hot reservoir). The flowrates thus have to be chosen so that the electrical power

required for heating does not exceed 25 kW, the maximum at which the laboratory three-phase power supply is rated. If more power were required, the energy costs of running the plant would also become excessive.

The hot reservoir is required to supply hot water continuously at a temperature of 80°C. The inflow to the reservoir comes from the flow separating tank FS as shown in figure 1.4. This tank separates the recycled CH stream (product in figure 1.3), channeling some of it to top up the hot reservoir, while sending the remainder of the water down the drain. (CH stream flowrate twice that of HC stream)

Thermal energy is required to raise the recycled water at 50°C to 80°C in the hot reservoir. The maximum rate of energy required is when the HC stream flow drawn from the hot reservoir is at its maximum. Calculations done in Appendix A show that choosing a maximum flowrate of 5 l/min requires less than 13.5 kW of electrical power to heat the water.

The actual heating of the water is split into two stages, as can be seen in figure 1.4. The first stage, the pre-heating hot reservoir (PHR) is fitted with three 3 kW heating elements that are thermostat controlled. The PHR heats the recycled water from 50°C to 75°C. The second stage, the controlled hot reservoir (CHR), is fitted with three 1.5 kW elements and a sensitive

temperature controller. Water from the PHR is heated and controlled accurately to 80°C in the CHR.

The controlled cold reservoir (CCR on figure 1.4) is required to supply cold water continuously at a temperature of 24°C. The inflow to the reservoir originates from two sources, namely:

- water mains at a temperature of roughly 20°C
- recycled water from HC stream at temperature somewhere below 28°C

At steady-state, (CH flow double HC flow) the reservoir temperature will settle to a temperature somewhere below 24°C. Some energy is thus required to raise the temperature of the reservoir to exactly 24°C.

The maximum power required by the cold reservoir will occur when the HC stream flow is zero while the CH stream flow from the reservoir is at 10 l/min. The calculations done in Appendix A show that slightly less than 4.5 kW of power is required in this case. The CCR is therefore fitted with a sensitive temperature controller and three 1.5 kW heating elements. The temperature in the CCR is thus controlled accurately to 24°C

The maximum total power demanded by all reservoirs is thus 18 kW, which is within the specified 25 kW maximum.

(ii) Choosing configuration of exchange vessels

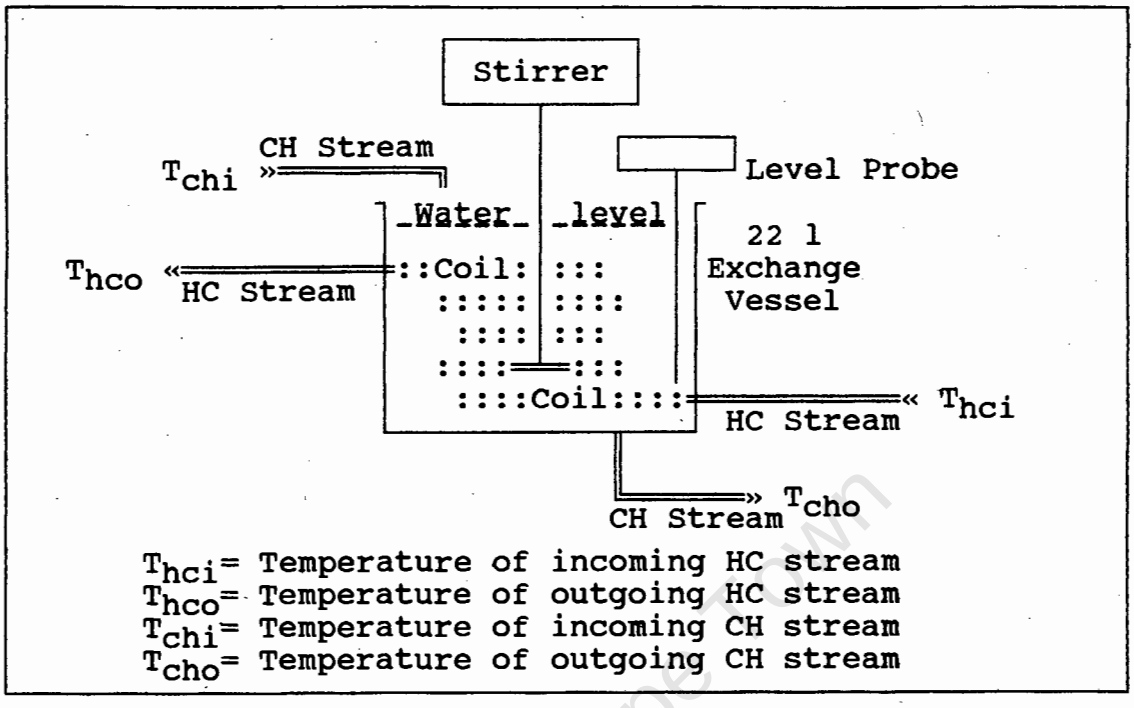


Figure 1.6: Typical exchange vessel of Heat Exchanger

Each exchange vessel, one of which is shown in figure 1.6, consists of the following:

a) **22 litre Covered Tank:** Vessel in which the exchange of heat occurs. Water from the CH stream enters the tank through its cover and exits through its base, residing in the tank for some time

b) **Copper Heating Coil:** Water of the HC stream passes through this coil. Heat is transferred through the walls of this coil to the water inside the reactor tanks.

c) **Level Probe:** Capacitive level probes source a 0 to 10 volt signal, indicating the water level.

d) **Variable Speed Stirrer:** The degree of agitation of the water in the tank is varied, thus varying the rate at which heat is transferred from the coils to the water in the tank.

The configuration of the heat exchange vessels and the associated piping has been chosen to realize the designed flowrates. The physical constraint of limited headroom in the laboratory had to be overcome in the design of the configuration, shown in figure 1.4.

The design process to determine the pipe diameters and tank heights is iterative and is shown in Appendix B. The plant was designed on a "per tank" basis, meaning that the configuration of each of the four tanks is identical.

A 0.75 inch diameter was chosen for all piping between the reactor vessels. A head of approximately 10 cm is necessary to realize the specified flowrate in the CH stream as it flows under gravity from tank No4 to tank No1. A 20 cm vertical spacing was chosen between the tanks, providing sufficient head to realize this flowrate.

(iv) Design of copper heating coils

The temperature profile across the process (indicated in figure 1.3) can be achieved by the careful design of the submerged heating coils. The design process, which is detailed in Appendix C, has taken the following points into account:

- Degree of agitation of water in vessel
- Material constituting the heating coils
- Coil dimensions (tube diameter, thickness,..)
- Flow velocity inside the coils

The designed coils are made up of numerous windings of copper tubing. A coil winding diameter of 25 cm diameter was chosen. Calculations of the length of 1/2 inch tubing required and the resultant number of coil windings required are shown in Appendix C. The winding specifications decided on are summarized in table 1.1 below:

Tank No	Length of Tubing (metres)	Number of Windings
1	3.3 to 3.7	5
2	5.7 to 6.4	8
3	9.2 to 10.2	13
4	4.4 to 4.9	6

Table 1.1: Submerged Lengths and Corresponding Number of Coil Windings for each tank

(v) Choosing stirrer speeds, impellers and motors

The design calculations in Appendix C assume a certain degree of agitation of the water inside the vessels. Stirrers are required to provide this agitation.

Variable speed stirrers are included in the vessels. Changing the stirring rate in a vessel, or the shape of the stirrer impeller, affects the heat transfer coefficient between the agitated water and the coil. This has a significant effect on the log mean temperature difference observed as a process stream passes through a vessel. Stirring speeds could thus also be used in the control of temperatures.

These effects are observed in Appendix D, where calculations are done to observe the effects on CH stream as it passes through tank No3.

Impellers of length 4 cm and width 2 cm have been chosen, rotating at speeds varying between 0 and 2000 rpm. Three-phase, 370 W induction motors with variable speed drives were chosen to drive the stirrers. These motors were chosen instead of D.C. motors, as the humidity from the vessels containing hot water could corrode the brushes of D.C. motors. There is also no cost advantage in choosing variable speed D.C. motors.

1.4.3) Input-Output Description of Heat Exchanger Process

Figure 1.7 summarizes the input/output characteristics of the heat exchanger process.

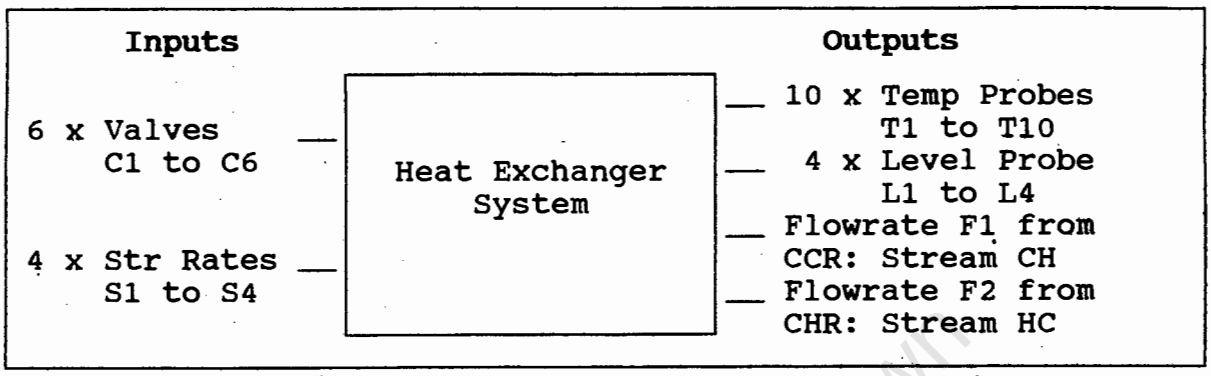


Figure 1.7: Input/Output schematic of Heat Exchanger

The system has 10 inputs and 16 outputs. This implies that all the outputs of the system cannot be controlled by its inputs as the number of outputs exceed the number of inputs. It may be possible to control a maximum of 10 outputs using the available inputs.

If variable speed stirrers were not included, there would only be 6 inputs to the system, meaning that it would be possible to control a maximum of 6 outputs. The inclusion of the stirrers thus makes the system more flexible in terms of its input/output control capabilities.

The variation of the stirring rate in the vessels of the heat exchanger cold be considered analogous to the addition of some catalyst to the vessels on the CIP process.

1.4.4) Instrumentation of Process and Interfacing to a Digital Computer

Figure 1.8 gives a block diagram of the wiring and interfacing of the heat exchanger system. The ensuing discussion, which summarizes the instrumentation and interfacing of the process, refers extensively to this diagram.

(i) Wiring of power to the plant

The power distribution box distributes power, along cables PD, to the following items of equipment and instrumentation:

a) **Equipment on the Process:** Compressor, reservoir pumps and the solenoid valve.

b) **Temperature Control Box:** Temperature control containing instrumentation controlling reservoir temperatures and three phase (abbreviated to 3φ) relays powering heating elements in the reservoirs.

c) **Instrumentation Box:** Instrumentation which conditions and interfaces plant input/output signals.

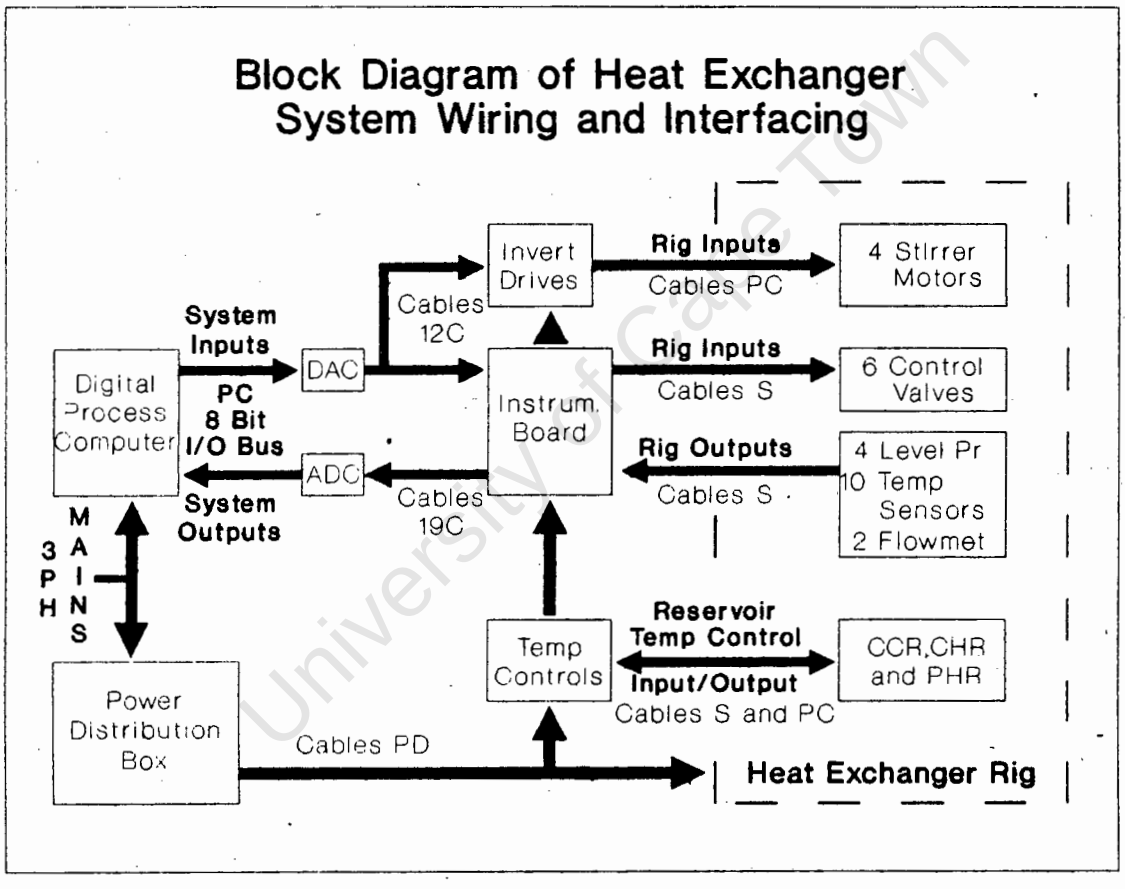


Figure 1.8: Block Diagram of Wiring and Interfacing

d) **Inverter Drive Box:** Inverters driving the stirrer motors.

A detailed description of the wiring of power to the plant is found in Appendix E

(ii) Wiring of inputs and outputs of the plant

The input/output signals of the plant can be split into two types, namely Standard Signals and Power Signals.

Standard signals (0 -> 5 or 10 V, 4 -> 20 mA) are low power sensing and control signals. Cables S carry signals between the instrumentation box and the following items of equipment on the plant:

a) **Control Valves:** (input) Current signals control the functioning of current to pressure (I/P) transducers which open and shut control valves pneumatically.

b) **Level Probes:** (output) Sensors source voltage signals indicating tank levels.

c) **Temperature Sensors:** (output) Sensors source current signals proportional to temperatures at various points on the plant.

d) **Flowmeters:** (output) Sensors source pulsed voltage signals of varying frequency indicating flows in the two process streams.

Power signals are high power sensing and control signals which actually drive items of equipment. Cables PC carry these signals, driving and originating from the following items of equipment:

a) **Stirrer Motors:** (input) Motors driven by 3φ variable frequency power signals, varying the speed of the motors.

b) **Reservoir Elements:** (input) 3φ power to the elements is switched on and off to control the water temperatures.

c) **PHR Thermostat:** (output) Signal from PHR thermostat switches relays, which drive elements, on and off.

Appendix F contains details of the wiring of plant inputs and outputs to signal driving and detection equipment.

(iii) Interfacing to a digital computer

A digital computer is used in the control of the heat exchanger. An interface between the digital environment of the computer and the analog environment of the plant is thus necessary.

Two 8-channel digital-to-analog converters (abbreviated to DACs) are used to provide input interfacing. Input signal conditioning instruments in the inverter drive and instrumentation boxes are connected to the DACs via cables 12C. DAC1 is programmed to source 0 - 5V signals required by the inverter drives to control the speeds of the stirrer motors. DAC2 is programmed to source 4 - 20 mA signals required by the I/P converter instrumentation to control the opening and shutting of control valves.

The outputs are interfaced to the computer using two 8-channel analog-to-digital converters (abbreviated to ADCs). The output signal conditioning instrumentation is connected to the ADCs via cables 19C. The instruments source 0 - 10V and 2 - 10V signals which are interfaced to the ADCs.

A detailed description of the interfacing with wiring diagrams is contained in Appendix G

1.4.5) Commissioning of Heat Exchanger

The heat exchanger was commissioned on 15 July 1989 after a construction, interfacing, calibration and testing period of almost 17 months. Photograph 1.1 shown below describes the commissioned heat exchanger process.

Photograph 1.1: The Heat Exchanger Process

1.5) Observed Steady-State Temperature Distributions

1.5.1) Comparison of Designed and Actual Distributions

The results from two independent tests (referred to as case 1 and case 2) are considered in which the process is run under conditions assumed for the design. These conditions are:

i) The levels in all reactor vessels at 90% full.

ii) All stirring rates at 90% of full (2000 rpm) speed.

iii) CH stream flow F1 and HC stream flow F2 kept at 50% of maximum. (10 and 5 l/min maximum respectively)

The resultant temperature distribution obtained across the process streams for each case is indicated in figure 1.9, together with the designed temperature distribution shown in figure 1.3. The trends in the observed results show good correlation with those for the design. The curves for the actual process are shifted downwards due to the fact that the temperature in the CCR was maintained at a lower temperature than in the design specifications.

It is instructive to consider the log mean temperature difference (LMTDiff) across each reactor when an assessment is to be made of the accuracy of the design. This negates effects of temperature offsets due to differences between

actual reservoir temperatures used and those assumed in the design.

A detailed comparison of the designed and observed temperature distributions is given in table 1.2. The designed and observed temperatures at points T1 to T10 are indicated in the table. The temperature difference Difft in each stream as it passes through a vessel and the LMTDiff across each vessel are also indicated. The differences between the DiffT obtained across each vessel for the designed and for the observed streams are indicated as DiffTerr. Differences in the designed and observed LMTDiff across each vessel are indicated as LMTDerr.

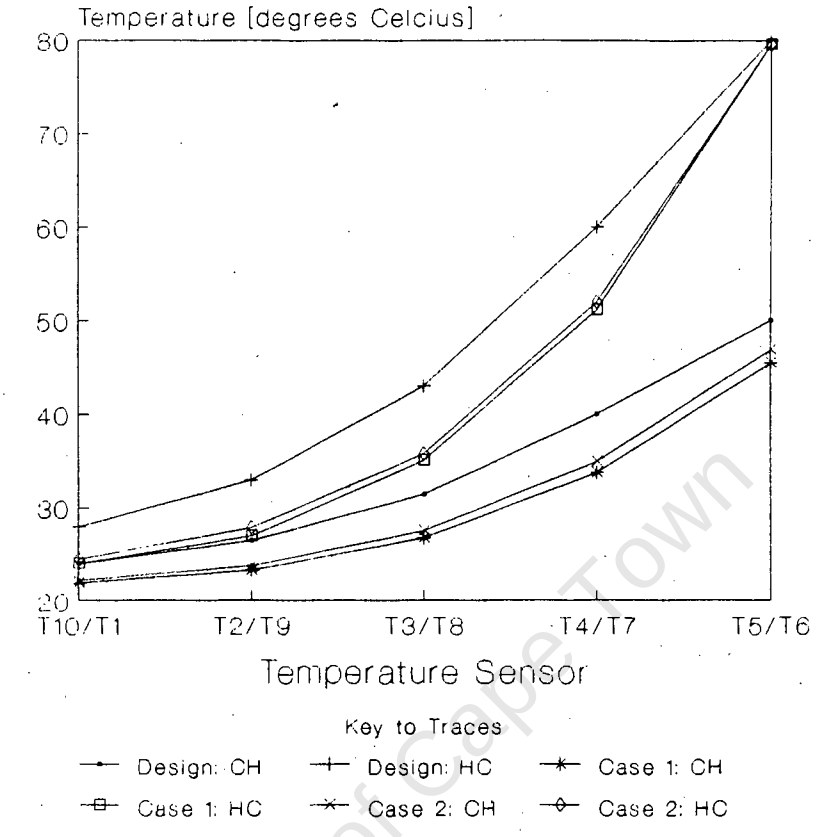
The mean error (Mean Err) between the designed and actual LMTDiff is shown in table 1.2 to be -2.42°C for case 1 and -2.76°C for case 2. The standard deviation (Stdev Err) in each of these errors is given as 0.479°C and 0.843°C respectively. The percentage error in the design is then calculated in each case from equations (1.2) and (1.3).

$$\%_{deserr} = (\text{Mean Err})/(\text{Max Design Temp Range}) \times 100\%$$
$$= (\text{Mean Err})/(80-24) \times 100\% \qquad (1.2)$$

with a percentage standard deviation of:

$$\%_{sdeverr} = (\text{Stdev Err})/(\text{Max Design Temp Range}) \times 100\%$$
$$= (\text{Stdev Err})/(80-24) \times 100\% \qquad (1.3)$$

Figure 1.9: Steady-State Distributions
Predicted and Actual Distributions



Flows = 50%, Levels = 90%, Stirs = 90%

Position	Design:CH	DiffT	Design:HC	DiffT	LMTDiff	Case 1:CH	DiffT	DiffTerr	Case 1:HC	DiffT	DiffTerr	LMTDiff	LMTDerr
T5/T6	50	-10	80	-20	18.20478	45.5	-11.7	-1.7	79.6	-28.6	-8.6	15.67510	-2.52968
T4/T7	40	-8.5	60	-17	8.960951	33.8	-7	1.5	51	-15.9	1.1	6.156717	-2.80423
T3/T8	31.5	-5	43	-10	4.909464	26.8	-3.5	1.5	35.1	-8.1	1.9	2.174091	-2.73537
T2/T9	26.5	-2.5	33	-5	3.409857	23.3	-1.4	1.1	27	-3	2	1.801793	-1.60806
T10/T1	24		28			21.9			24				
Mean Err								0.6			-0.9		-2.41933
StDev Err								1.337908			4.459260		0.479156

Position	Design:CH	DiffT	Design:HC	DiffT	LMTDiff	Case 2:CH	DiffT	DiffTerr	Case 2:CH	DiffT	DiffTerr	LMTDiff	LMTDerr
T5/T6	50	-10	80	-20	18.20478	46.9	-11.9	-1.9	79.6	-27.6	-7.6	14.85360	-3.35117
T4/T7	40	-8.5	60	-17	8.960951	35	-7.4	1.1	52	-16.2	0.8	5.300428	-3.66052
T3/T8	31.5	-5	43	-10	4.909464	27.6	-3.8	1.2	35.8	-7.9	2.1	2.388072	-2.52139
T2/T9	26.5	-2.5	33	-5	3.409857	23.8	-1.6	0.9	27.9	-3.4	1.6	1.923444	-1.48641
T10/T1	24		28			22.2			24.5				
Mean Err								0.325			-0.775		-2.75487
StDev Err								1.289137			3.967603		0.842509

Table 1.2 : Analysis of Predicted and Actual Steady-State Temperatures across the Process

The results obtained for case 1 are $\%_{deserr}$ = -4.32% with $\%_{sdeverr}$ = 0.86%, and for case 2 are $\%_{deserr}$ = -4.93% with $\%_{sdeverr}$ = 1.51%.

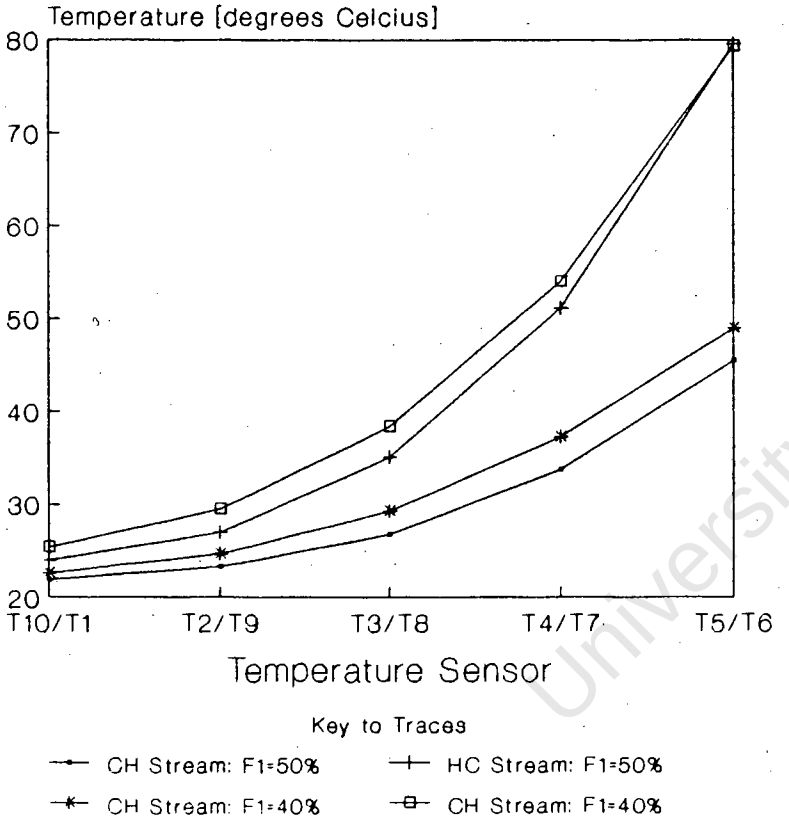
The traces for case 1 and case 2 are close together, indicating a good degree of repeatability of the conditions observed on the process.

1.5.2) Effects of Changing Flows on Distributions

The steady-state effects on the process temperature distributions of varying the flows in the process streams are seen in figures 1.10 and 1.11. Figure 1.10 shows a downward shift in the distribution curves as the flow F1 from the CCR is increased from 40% to 50% while keeping F2 at 50%. Figure 1.11 shows an upward shift in the curves as flow F2 from the CHR is increased from 40% to 50% while keeping F1 at 50% (levels and stirring rates kept at 90% in both cases).

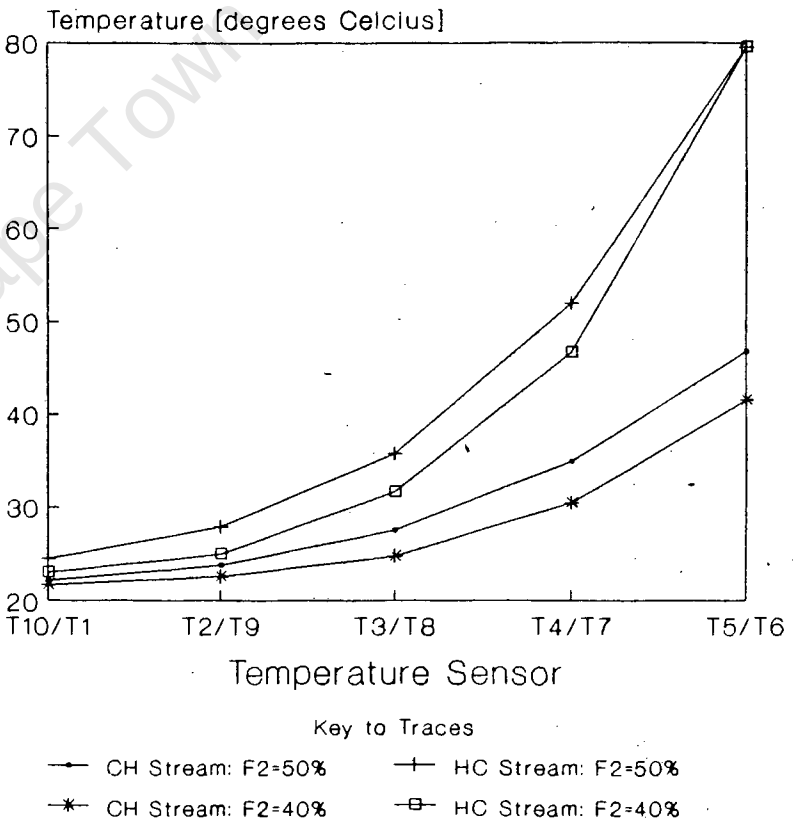
The shifts in the traces show that changing the flows in the process streams has a significant effect on the temperature distribution across the process, meaning that they could be used in the control of the process temperatures.

Figure 1.10: Steady-State Flow Effects
Changing F1 from 40% to 50%



F2 = 50%, Levels = 90%, Stirs = 90%

Figure 1.11: Steady-State Flow Effects
Changing F2 from 40% to 50%



F1 = 50%, Levels = 90%, Stirs = 90%

1.7) Concluding Remarks

A number of useful analogies exist between the heat exchanger design and the CIP processes. The fully commissioned process behaves as designed with less than 5% average design error in the LMTDiff across each heat exchange vessel. The process can thus be used to explore the existing analogies, in an attempt to gain experience which can be drawn from in the design of a control system for the CIP process.

The most significant difference between the processes is the fact that CIP processes are usually run as batch processes, while the heat exchanger is a continuous process. It is possible to run a continuous process in batch mode, and future research on the heat exchanger could involve a comparison of the respective efficiencies obtained when running under batch and continuous modes of operation. This could assist in determining whether the current batch mode of running CIP processes is indeed the most efficient one. Running a process under a continuous mode of operation makes it possible to apply the wealth of existing knowledge on linear control systems to the control of the process.

The significant changes observed in the temperature distribution across the process when the flows in each of the streams are changed indicate that these flows are important inputs to be considered in the control of the temperatures.

CHAPTER TWO

CONTROL STRUCTURE FOR THE HEAT EXCHANGER PROCESS

The heat exchanger process structure is analyzed in this chapter in order to decide on an appropriate control structure.

The chapter begins with a brief review of modern techniques available for the analysis large-scale process structures. The decentralization of control structures using cascade compensators is discussed in this review presented in section 2.1.

The techniques reviewed are then applied to the analysis of the heat exchanger process structure in section 2.2. A controller structure based on the process structure is decided on. The resultant control system structure is then discussed and analyzed further.

Concluding remarks are made in section 2.3, in which the control strategy for the heat exchanger process is discussed.

2.1) The Analysis and Control of Large-Scale Systems

Multivariable (MIMO) processes with many inputs and outputs (generally in excess of 5 inputs and 5 outputs) are classified as large-scale systems. Decentralized control

schemes, discussed by Gear (Gear, 1988) in his postgraduate dissertation, can be applied in the control of such processes.

Application of these control techniques require that the process be divided up into a number of subsystems. Controllers can then be designed for each subsystem independently, ignoring any interaction which may occur between the different subsystems. Independent compensators can thus be designed to regulate a subsection of process outputs without attempting to control any of the other outputs. Interactions between the subsystems will not generally be negligible, but choosing a suitable subsystem structure will ensure stability of the total system under distributive control. Analysis of the structure of the system is thus of importance.

2.1.1) Binary Interaction Matrix

Before deciding on a subsystem structure for a process, the structure of the process model needs to be considered carefully.

A powerful tool for the analysis of process structures is the binary interaction matrix (abbreviated to BIM), which is also discussed in Gear's (Gear, 1988) dissertation.

The system matrix of a process with m inputs and n outputs is given by:

$$\mathbf{G(s)} = \{g_{ij}(s)\} \; (i = 1..n, \; j = 1..m) \qquad (2.1)$$

When determining the binary interaction matrix (abbreviated to BIM) for a process, the physical relationship between each input and output needs to be considered. This is done in order to decide whether or not a specific input will affect an output.

If any physical connection occurs between a process input j and output i, the input will affect the output. The respective element $g_{ij}(s)$ relating them in the system matrix **G(s)**, will thus be nonzero. If no connection occurs, the element $g_{ij}(s)$ will be zero.

This analysis of the process makes no attempt to determine the actual value of the $g_{ij}(s)$ term, but only aims to determine whether such a term is zero or nonzero.

The BIM for the process under consideration is given by:

$$\mathbf{B(G(s))} = \{b_{ij}\} \; (i = 1..n, \; j = 1..m) \qquad (2.2)$$

The elements b_{ij} are defined as follows:

considering $g_{ij}(s)$ for i = 1..n, j = 1..m,

$$b_{ij} = \begin{cases} 1 & \text{if } g_{ij}(s) \neq 0 \\ 0 & \text{if } g_{ij}(s) = 0 \end{cases} \qquad (2.3)$$

Important Note:

The definition of **B** does not require the value of any $g_{ij}(s)$ term to be known, only whether such a term is zero or nonzero.

2.1.2) Regulating a Subsystem of Outputs using a Cascade Compensator

Assume that structural analysis of some 6x4 process yields a BIM:

$$\mathbf{B(G(s))} = \begin{array}{c} \\ y1 \\ y2 \\ y3 \\ y4 \\ y5 \\ y6 \end{array} \begin{array}{c} \begin{array}{cccc} u1 & u2 & u3 & u4 \end{array} \\ \left[\begin{array}{cccc} 1 & 0 & 0 & 0 \\ 0 & 1 & 0 & 0 \\ 1 & 1 & 0 & 0 \\ 0 & 1 & 1 & 1 \\ 1 & 0 & 1 & 1 \\ 1 & 0 & 1 & 1 \end{array}\right] \end{array} \qquad (2.4)$$

Assume that the process is split into four subsystems (2x2 blocking) as shown in (2.4), then G(s) can be expressed in terms of its constituent subsystems:

$$G(s) = \left[\begin{array}{c|c} G_{11}(s) & G_{12}(s) \\ \hline G_{21}(s) & G_{22}(s) \end{array}\right] \qquad (2.5)$$

where:

$G_{11}(s)$: 3x3 system submatrix relating inputs u1..u3 to outputs y1..y3

$G_{12}(s)$: 3x1 (zero) system submatrix relating input u4 to outputs y1..y3

$G_{12}(s)$: 3x3 system submatrix relating inputs u1..u3 to outputs y4..y6

$G_{22}(s)$: 1x3 system submatrix relating input u1 to outputs y4..y6

G(s) is typical of the class of systems with block (lower) triangular structures. ($G_{12}(s)$ zero matrix).

On condition that no unstable elements occur in subsystem matrix $G_{21}(s)$, it is possible to design a 3x3 cascade compensator $K_{c11}(s)$ for the 3x3 subsystem $G_{11}(s)$. This compensator will regulate outputs y1..y3 using inputs u1..u3. Gear (Gear, 1988) has shown that the compensator will not destabilize the system as a whole if the abovementioned condition is met.

It is often convenient to design compensators which regulate a subsystem of process outputs before attempting the control of the remaining process outputs.

2.2) Control Structure for the Heat Exchanger Process

The (large scale) structure of the heat exchanger process model is analyzed in this section and a suitable controller structure for the system is suggested based on this analysis.

The BIM for the system is shown in section 2.2.1, and a controller structure for the process is suggested in section 2.2.2. Section 2.2.3 discusses the suggested control structure, and analyses the control system further using the BIM.

2.2.1) Binary Interaction Matrix for Process

The heat exchanger process has 16 outputs and 10 inputs. The process thus has a 16x10 system matrix:

$$\mathbf{G(s)} = \{g_{ij}(s)\} \quad (i = 1..16,\ j = 1..10) \qquad (2.6)$$

G(s) relates the system inputs and outputs according to (2.7).

$$\begin{bmatrix} F1(s) \\ F2(s) \\ L4(s) \\ L3(s) \\ L2(s) \\ L1(s) \\ T1(s) \\ T2(s) \\ T3(s) \\ T4(s) \\ T5(s) \\ T6(s) \\ T7(s) \\ T8(s) \\ T9(s) \\ T10(s) \end{bmatrix} = \begin{bmatrix} y1(s) \\ y2(s) \\ y3(s) \\ y4(s) \\ y5(s) \\ y6(s) \\ y7(s) \\ y8(s) \\ y9(s) \\ y10(s) \\ y11(s) \\ y12(s) \\ y13(s) \\ y14(s) \\ y15(s) \\ y16(s) \end{bmatrix} = G(s) \begin{bmatrix} C1(s) \\ C2(s) \\ C3(s) \\ C4(s) \\ C5(s) \\ C6(s) \\ S1(s) \\ S2(s) \\ S3(s) \\ S4(s) \end{bmatrix} = G(s) \begin{bmatrix} u1(s) \\ u2(s) \\ u3(s) \\ u4(s) \\ u5(s) \\ u6(s) \\ u7(s) \\ u8(s) \\ u9(s) \\ u10(s) \end{bmatrix} \qquad (2.7)$$

where system outputs, y1(s)..y16(s) given by:

Flow outputs	F1(s)..F2(s)
Level outputs	L1(s)..L4(s)
Temperature outputs	T1(s)..T10(s)

and system inputs, u1(s)..u10(s) given by:

C-Valve inputs	C1(s)..C6(s)
Stirrer inputs	S1(s)..S4(s)

Figure 2.1 gives a block diagram representation of the process.

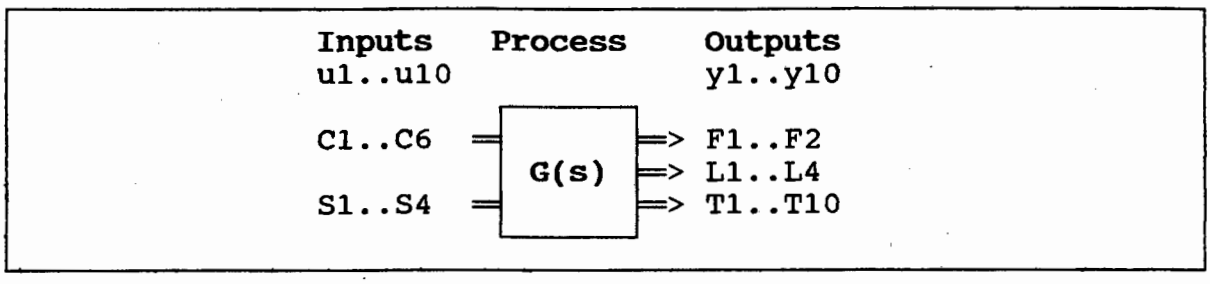


Figure 2.1: Block Diagram of Heat Exchanger System

The outputs which are affected by each input can be determined by observing the physical structure of the process. From these observations, the BIM for the process, **B(G(s))** can be constructed as detailed in the following paragraph.

If, for j = 1..10, i = 1..16:

(i) *Input j **affects** output i*

Then $g_{ij}(s) \neq 0$, thus $b_{ij} = 1$ (equ. (2.3))

(ii) *Input j **does not affect** output i*

Then $g_{ij}(s) = 0$, thus $b_{ij} = 0$ (equ. (2.3))

Observations of this nature made in Appendix H result in the following BIM being constructed for the heat exchanger process:

$$
B(G(s)) \quad = \quad
\begin{array}{c}
\\
F1 \\ F2 \\ L4 \\ L3 \\ L2 \\ L1 \\ T1 \\ T2 \\ T3 \\ T4 \\ T5 \\ T6 \\ T7 \\ T8 \\ T9 \\ T10
\end{array}
\begin{array}{c}
\begin{array}{cccccccccc} C1 & C2 & C3 & C4 & C5 & C6 & S1 & S2 & S3 & S4 \end{array} \\
\begin{bmatrix}
1 & 0 & 0 & 0 & 0 & 0 & 0 & 0 & 0 & 0 \\
0 & 0 & 0 & 0 & 0 & 1 & 0 & 0 & 0 & 0 \\
1 & 1 & 0 & 0 & 0 & 0 & 0 & 0 & 0 & 0 \\
1 & 1 & 1 & 0 & 0 & 0 & 0 & 0 & 0 & 0 \\
1 & 1 & 1 & 1 & 0 & 0 & 0 & 0 & 0 & 0 \\
1 & 1 & 1 & 1 & 1 & 0 & 0 & 0 & 0 & 0 \\
1 & 1 & 1 & 1 & 1 & 1 & 1 & 1 & 1 & 1 \\
1 & 1 & 1 & 1 & 1 & 1 & 1 & 1 & 1 & 1 \\
1 & 1 & 1 & 1 & 1 & 1 & 1 & 1 & 1 & 1 \\
1 & 1 & 1 & 1 & 1 & 1 & 1 & 1 & 1 & 1 \\
1 & 1 & 1 & 1 & 1 & 1 & 1 & 1 & 1 & 1 \\
1 & 1 & 1 & 1 & 1 & 1 & 1 & 1 & 1 & 1 \\
1 & 1 & 1 & 1 & 1 & 1 & 1 & 1 & 1 & 1 \\
1 & 1 & 1 & 1 & 1 & 1 & 1 & 1 & 1 & 1 \\
1 & 1 & 1 & 1 & 1 & 1 & 1 & 1 & 1 & 1 \\
1 & 1 & 1 & 1 & 1 & 1 & 1 & 1 & 1 & 1
\end{bmatrix}
\end{array}
\tag{2.8}
$$

2.2.2) Motivation for Inclusion of Cascade Compensator

Rearranging the BIM in (2.8) and partitioning (2x2 blocking) the resultant BIM into four subsystems realizes a process model with a block (lower) triangular structure, as can be seen in (2.9).

$$
B(G(s)) \quad = \quad
\begin{array}{c}
 \\ F1 \\ L4 \\ L3 \\ L2 \\ L1 \\ F2 \\ T1 \\ T2 \\ T3 \\ T4 \\ T5 \\ T6 \\ T7 \\ T8 \\ T9 \\ T10
\end{array}
\begin{array}{c}
\begin{array}{cccccccccc} C1 & C2 & C3 & C4 & C5 & C6 & S1 & S2 & S3 & S4 \end{array} \\
\left[\begin{array}{cccccc|cccc}
1 & 0 & 0 & 0 & 0 & 0 & 0 & 0 & 0 & 0 \\
1 & 1 & 0 & 0 & 0 & 0 & 0 & 0 & 0 & 0 \\
1 & 1 & 1 & 0 & 0 & 0 & 0 & 0 & 0 & 0 \\
1 & 1 & 1 & 1 & 0 & 0 & 0 & 0 & 0 & 0 \\
1 & 1 & 1 & 1 & 1 & 0 & 0 & 0 & 0 & 0 \\
0 & 0 & 0 & 0 & 0 & 1 & 0 & 0 & 0 & 0 \\
\hline
1 & 1 & 1 & 1 & 1 & 1 & 1 & 1 & 1 & 1 \\
1 & 1 & 1 & 1 & 1 & 1 & 1 & 1 & 1 & 1 \\
1 & 1 & 1 & 1 & 1 & 1 & 1 & 1 & 1 & 1 \\
1 & 1 & 1 & 1 & 1 & 1 & 1 & 1 & 1 & 1 \\
1 & 1 & 1 & 1 & 1 & 1 & 1 & 1 & 1 & 1 \\
1 & 1 & 1 & 1 & 1 & 1 & 1 & 1 & 1 & 1 \\
1 & 1 & 1 & 1 & 1 & 1 & 1 & 1 & 1 & 1 \\
1 & 1 & 1 & 1 & 1 & 1 & 1 & 1 & 1 & 1 \\
1 & 1 & 1 & 1 & 1 & 1 & 1 & 1 & 1 & 1 \\
1 & 1 & 1 & 1 & 1 & 1 & 1 & 1 & 1 & 1
\end{array}\right]
\end{array}
\qquad (2.9)
$$

G(s) can be expressed in terms of its subsystem matrices as follows:

$$
G(s) = \left[\begin{array}{c|c} G_{11}(s) & G_{12}(s) \\ \hline G_{21}(s) & G_{22}(s) \end{array}\right] \qquad (2.10)
$$

where:

$G_{11}(s)$: 6x6 system submatrix

$G_{12}(s)$: 6x4 system (zero) submatrix

$G_{12}(s)$: 10x6 system submatrix

$G_{22}(s)$: 10x4 system submatrix

The following observations were made from output responses to initial tests performed on control valve inputs (responses shown in Appendix H):

(i) The responses of level and flow outputs to changes in valve inputs are significantly faster than those of temperature outputs

(ii) No unstable elements occur anywhere in G(s)

Observation (i) shows that the process is a typical fast-slow type system. It would therefore be convenient to design a **fast control system to regulate the level and flow outputs** before attempting to control the slower temperature outputs. This would involve the design of a cascade compensator for the $G_{11}(s)$ system submatrix.

Observation (ii) implies that a fast level and flow compensator can be designed to control subsystem $G_{11}(s)$ **without destabilizing the process as a whole.**

Based on these observations, the design of a 6x6 cascade compensator $K_c(s)$ for the process is suggested. This compensator is to regulate the flow and level outputs before it is attempted to control the temperature outputs.

2.2.3) Discussion of Structure of Modified Process with Cascade Compensator

A block diagram for the modified system with a cascade compensator $K_c(s)$ for the flows and levels is given in figure 2.2

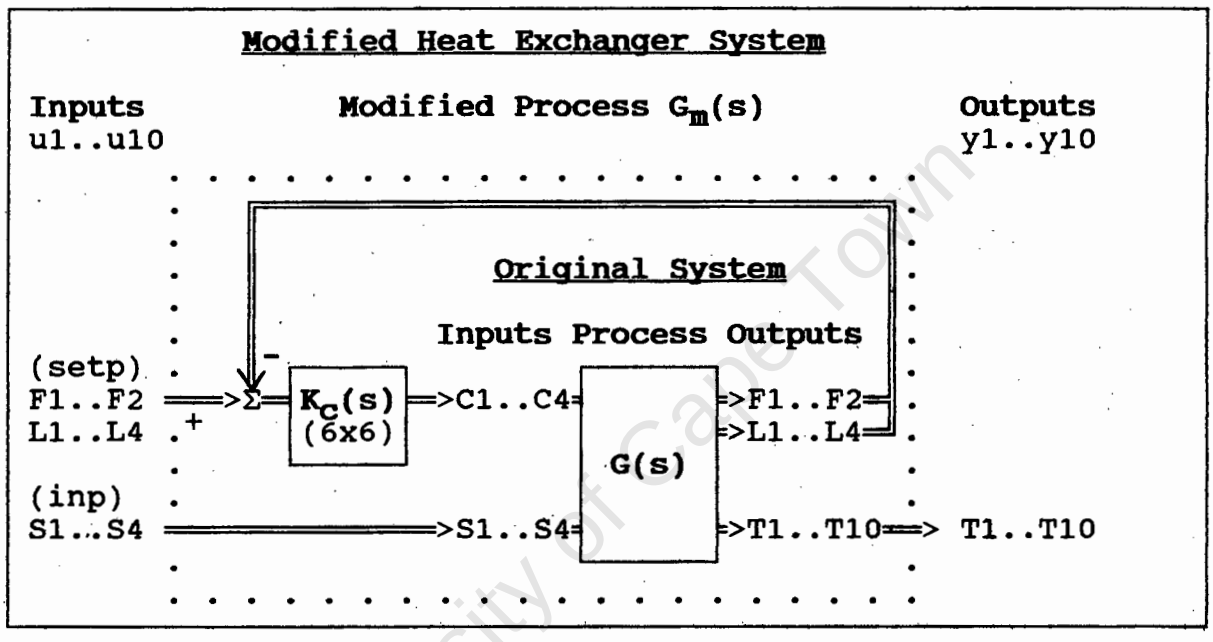


Figure 2.2: Block Diagram of Modified Heat Exchanger System With Cascade Compensator for Flows and Levels

Figure 2.2 shows that the original heat exchange system shown in figure 2.1 has been modified with the inclusion of a 6x6 cascade compensator $K_c(s)$. $K_c(s)$ acts on $G_{11}(s)$ (referred to as $G_c(s)$ for the rest of this report to avoid confusion), a 6x6 subsystem matrix of G(s), to regulate the outputs F1 to F2 and L1 to L4.

The level and flow setpoints can now be considered as inputs to the modified process, as shown in figure 2.2. This

modified system with 10 inputs and 10 outputs has a 10x10 system matrix:

$$G_m(s) = \{gm_{ij}(s)\} \quad (i = 1..10,\ j = 1..10) \quad (2.11)$$

$G_m(s)$ relates the modified system inputs and outputs according to (2.12) below:

$$\begin{bmatrix} T1(s) \\ T2(s) \\ T3(s) \\ T4(s) \\ T5(s) \\ T6(s) \\ T7(s) \\ T8(s) \\ T9(s) \\ T10(s) \end{bmatrix} = \begin{bmatrix} y1(s) \\ y2(s) \\ y3(s) \\ y4(s) \\ y5(s) \\ y6(s) \\ y7(s) \\ y8(s) \\ y9(s) \\ y10(s) \end{bmatrix} = G_m(s) \begin{bmatrix} F1(s) \\ F2(s) \\ L1(s) \\ L2(s) \\ L3(s) \\ L4(s) \\ S1(s) \\ S2(s) \\ S3(s) \\ S4(s) \end{bmatrix} = G_m(s) \begin{bmatrix} u1(s) \\ u2(s) \\ u3(s) \\ u4(s) \\ u5(s) \\ u6(s) \\ u7(s) \\ u8(s) \\ u9(s) \\ u10(s) \end{bmatrix} \quad (2.12)$$

where system outputs, y1(s)..y10(s) given by:

Temperature outputs T1(s)..T10(s)

and system inputs, u1(s)..u10(s) given by:

Flow setpoints	F1(s)..F2(s)
Level setpoints	L1(s)..L4(s)
Stirrer inputs	S1(s)..S4(s)

The BIM in (2.8) shows that any one control valve input affects all of the temperature outputs. Control valves C1 to C6 are used to regulate flow and level outputs. This means

that changes in the any one of the flow or level setpoints (inputs to the modified process) will affect all of the temperature outputs via regulating changes in the control valve inputs. A BIM for the modified system, $B(G_m(s))$ can thus be constructed taking these points into consideration.

$$B(G_m(s)) = \begin{array}{c} \\ T1 \\ T2 \\ T3 \\ T4 \\ T5 \\ T6 \\ T7 \\ T8 \\ T9 \\ T10 \end{array} \begin{array}{c} \begin{array}{cccccccccc} F1 & F2 & L1 & L2 & L3 & L4 & S1 & S2 & S3 & S4 \end{array} \\ \left[\begin{array}{cccccccccc} 1 & 1 & 1 & 1 & 1 & 1 & 1 & 1 & 1 & 1 \\ 1 & 1 & 1 & 1 & 1 & 1 & 1 & 1 & 1 & 1 \\ 1 & 1 & 1 & 1 & 1 & 1 & 1 & 1 & 1 & 1 \\ 1 & 1 & 1 & 1 & 1 & 1 & 1 & 1 & 1 & 1 \\ 1 & 1 & 1 & 1 & 1 & 1 & 1 & 1 & 1 & 1 \\ 1 & 1 & 1 & 1 & 1 & 1 & 1 & 1 & 1 & 1 \\ 1 & 1 & 1 & 1 & 1 & 1 & 1 & 1 & 1 & 1 \\ 1 & 1 & 1 & 1 & 1 & 1 & 1 & 1 & 1 & 1 \\ 1 & 1 & 1 & 1 & 1 & 1 & 1 & 1 & 1 & 1 \\ 1 & 1 & 1 & 1 & 1 & 1 & 1 & 1 & 1 & 1 \end{array}\right] \end{array} \tag{2.13}$$

Temperature outputs T1 to T10 can now be controlled using the flow and level setpoints as well as the stirrer inputs. Note that for this modified system, flow and level outputs cannot be controlled, as their setpoints are being used in the control of the temperature outputs.

2.3) Concluding Remarks

The heat exchanger process model has been analyzed using BIM techniques. This analysis suggests a definite control structure for the process, consisting of the two parts, as discussed in (i) and (ii).

(i) Cascade Compensator for Fast Flows and Levels

The flow and level outputs respond faster to changes in the inputs than do the temperature outputs. BIM analysis of the process shows that it is possible to control the faster flow and level outputs before attempting control of the temperature outputs. A cascade compensator $K_c(s)$ is thus proposed for the closed loop control of the system (lower triangular) submatrix $G_c(s)$. $G_c(s)$ relates the flow and level outputs to the control valve inputs. The design and implementation of a controller for this lower triangular subsystem is dealt with in chapter 3.

(ii) Full Controller for Slow Temperatures

The closed loop control system containing the compensator $K_c(s)$ is treated as a modified (full) process model $G_m(s)$. $G_m(s)$ relates the temperature outputs to the flow and level inputs (setpoints of the $G_c(s)$, $K_c(s)$ subsystem) and stirrer inputs. This full structure can now be controlled using multivariable control techniques. (dealt with in chapter 5)

CHAPTER THREE

DESIGN AND IMPLEMENTATION OF FLOW AND LEVEL CONTROLLERS

3.1) Introduction: Lower Triangular Subsection of Process

The BIM in (2.9) shows that the heat exchanger process model **G(s)** can be spilt into the four system submatrices, $G_c(s) = G_{11}(s)$, $G_{12}(s)$ (= 0), $G_{21}(s)$ and $G_{22}(s)$, defined in (2.10). No unstable elements occur in the off-diagonal submatrices, meaning that compensators can be designed for any of them separately without destabilizing the process as a whole.

$G_c(s)$ relates the 6 control valve inputs to the 2 flow and 4 level outputs. The elements $gc_{ij}(s)$ of the 6x6 system submatrix $G_c(s)$ are thus defined as:

$$G_c(s) = \{gc_{ij}(s)\} \quad (i = 1..6,\ j = 1..6) \qquad (3.1)$$

$G_c(s)$ relates the inputs and outputs as shown in (3.2). (Inputs and outputs as defined for (2.7)).

$$\begin{bmatrix} F1(s) \\ F2(s) \\ L4(s) \\ L3(s) \\ L2(s) \\ L1(s) \end{bmatrix} = \begin{bmatrix} y1(s) \\ y2(s) \\ y3(s) \\ y4(s) \\ y5(s) \\ y6(s) \end{bmatrix} = G_c(s) \begin{bmatrix} C1(s) \\ C2(s) \\ C3(s) \\ C4(s) \\ C5(s) \\ C6(s) \end{bmatrix} = G_c(s) \begin{bmatrix} u1(s) \\ u2(s) \\ u3(s) \\ u4(s) \\ u5(s) \\ u6(s) \end{bmatrix} \qquad (3.2)$$

The BIM for $G_c(s)$, called $B(G_c(s))$, is given in (3.3) and is essentially the upper left submatrix of the BIM in (2.9). The BIM for $G_c(s)$ is seen to be lower-triangular.

$$B(G_c(s)) = \begin{array}{c} \\ F1 \\ L4 \\ L3 \\ L2 \\ L1 \\ F2 \end{array} \begin{array}{c} \begin{array}{cccccc} C1 & C2 & C3 & C4 & C5 & C6 \end{array} \\ \begin{bmatrix} 1 & 0 & 0 & 0 & 0 & 0 \\ 1 & 1 & 0 & 0 & 0 & 0 \\ 1 & 1 & 1 & 0 & 0 & 0 \\ 1 & 1 & 1 & 1 & 0 & 0 \\ 1 & 1 & 1 & 1 & 1 & 0 \\ 0 & 0 & 0 & 0 & 0 & 1 \end{bmatrix} \end{array} \qquad (3.3)$$

The breakdown of this chapter is now discussed. Section 3.2 contains a review of modern theories for the control of processes with triangular structures (such as the structure in (3.3)). A suitable structure for a cascade compensator $K_c(s)$ for the closed loop control of the levels and flows is then decided on, based on the theory reviewed. Specific transfer function elements of the system submatrix $G_c(s)$ are determined in section 3.3. In section 3.4, a cascade compensator $K_c(s)$ is designed (based on the theory reviewed) for the closed loop regulation of the flow and level outputs. The performance objectives for the design are stated, and the design procedure undertaken to achieve the required performance is demonstrated. The design for $K_c(s)$ is implemented on the heat exchanger process, and the performance of the resultant control system is observed. Concluding remarks on chapter 3 are made in section 3.5.

3.2) The Control of Processes with a Triangular Structure

Assume that a process with system matrix **G(s)** is to be controlled using a cascade compensator **K(s)**. A block diagram of the resultant closed loop (abbreviate to C/L) control system including the cascade compensator is given in figure 3.1.

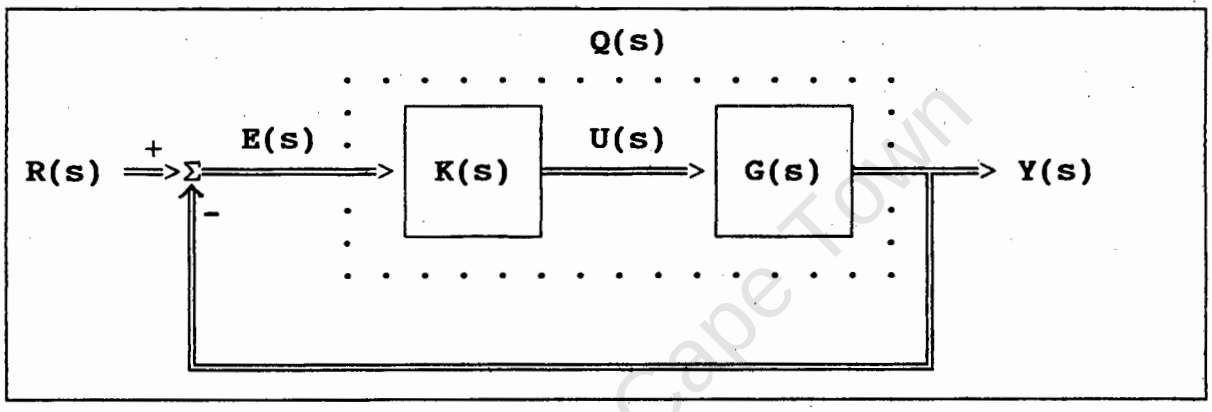


Figure 3.1: Block Diagram of Closed Loop System with Cascade Compensator

Key to symbolic representations in figure 3.1:

U(s) : Vector of Process Inputs

Y(s) : Vector of Process Outputs
R(s) : Vector of Setpoints for Outputs

E(s) : Vector of Errors in outputs (**R(s) - Y(s)**)

The open loop (abbreviated to O/L) system matrix **Q(s)**, including compensator, is given by:

$$Q(s) = G(s)\,K(s) \qquad (3.4)$$

The corresponding C/L system matrix **H(s)**, including compensator is then:

$$H(s) = (I + Q(s))^{-1} Q(s) \qquad (3.5)$$

Consider the case where **G(s)** has a (lower) block triangular structure as shown in (3.5) for a 2x2 blocking.

$$G(s) = \left[\begin{array}{c|c} G_{11}(s) & 0 \\ \hline G_{21}(s) & G_{22}(s) \end{array}\right] \qquad (3.6)$$

Any controller **K(s)** may be chosen for the process, provided that the controller (Gear 1988, pp63..64):

(a) Does not introduce any unstable poles into the off-diagonal submatrices of
Q(s) = G(s) K(s).

(b) Does not destroy the block triangular structure of the system. (ie **Q(s)** is also block triangular)

(c) Ensures that each on-diagonal subsystem of **Q(s)** is C/L stable.

The two structures for **K(s)** that fulfill requirement (b) are the block diagonal and block triangular structures. The former structure is discussed in section 3.2.1 and the latter in section 3.2.2. Extensive reference is made in these

sections to Gear's thesis (Gear, 1988 pp54..68). Section 3.2.3 discusses the reasons for adopting a diagonal compensator structure for the control of flows and levels on heat exchanger process.

3.2.1) Diagonal Controller Structures

The block diagonal structure for **K(s)**, partitioned in conformity with **G(s)**, is shown in (3.7) and realizes a fully decentralized controller for the process.

$$\mathbf{K(s)} = \left[\begin{array}{c|c} \mathbf{K_{11}(s)} & \mathbf{0} \\ \hline \mathbf{0} & \mathbf{K_{22}(s)} \end{array}\right] \qquad (3.7)$$

The O/L system matrix **Q(s) = G(s) K(s)** shown in (3.8) will consequently be block triangular.

$$\mathbf{Q(s)} = \left[\begin{array}{c|c} \mathbf{Q_{11}(s)} & \mathbf{0} \\ \hline \mathbf{Q_{21}(s)} & \mathbf{Q_{22}(s)} \end{array}\right] \qquad (3.8)$$

where:

$$\mathbf{Q_{11}(s)} = \mathbf{G_{11}(s)\ K_{11}(s)}$$
$$\mathbf{Q_{22}(s)} = \mathbf{G_{22}(s)\ K_{22}(s)}$$
$$\mathbf{Q_{21}(s)} = \mathbf{G_{21}(s)\ K_{11}(s)}$$

Provided that $K_{11}(s)$ has no unstable poles, no unstable poles will be introduced to the off-diagonal submatrices of $Q(s)$, thus fulfilling requirement (a).

The C/L system matrix $H(s) = (I + Q(s))^{-1} Q(s)$ shown in (3.9) is then also block triangular.

$$H(s) = \left[\begin{array}{c|c} H_{11}(s) & 0 \\ \hline H_{21}(s) & H_{22}(s) \end{array}\right] \qquad (3.9)$$

where:

$$H_{11}(s) = (I + Q_{11}(s))^{-1} Q_{11}(s)$$

$$H_{22}(s) = (I + Q_{22}(s))^{-1} Q_{22}(s)$$

$$H_{21}(s) = (I + Q_{22}(s))^{-1} Q_{21}(s) \{I - (I + Q_{11}(s))^{-1} Q_{11}(s)\}$$

The following conclusions can be drawn:

(i) Interaction between Loops:

The term $H_{21}(s)$ represents the degree to which the subsystem $H_{22}(s)$ is coupled to the subsystem $H_{11}(s)$. $H_{21}(s)$ can be reduced by 'large gains' in $Q_{22}(s)$ and $Q_{11}(s)$. This can be realized by ensuring that the 'gains' of $K_{22}(s)$ and $K_{11}(s)$ are 'large'. This is equivalent to designing controllers to reject disturbances from other loops.

(ii) Stability of Closed Loop System:

Designing $K_{22}(s)$ and $K_{11}(s)$ to stabilize $Q_{22}(s)$ and $Q_{11}(s)$ respectively will ensure that the entire system is C/L stable. The transfer function elements in system submatrices $G_{22}(s)$ and $G_{11}(s)$ are required for this design. Provided that it is known that the elements of $G_{21}(s)$ have no unstable poles, the actual transfer functions of the elements are not required to be known. This important conclusion implies that only the transfer function elements of the diagonal submatrices of $G(s)$ need to be determined to design a stable $K(s)$ with a diagonal structure for the process.

3.2.2) Triangular Controller Structures

The block triangular structure for $K(s)$, partitioned in conformity with $G(s)$, is shown in (3.10).

$$K(s) = \left[\begin{array}{c|c} K_{11}(s) & 0 \\ \hline K_{21}(s) & K_{22}(s) \end{array}\right] \qquad (3.10)$$

The O/L system matrix $Q(s) = G(s)\ K(s)$ shown in (3.11) will once again be block triangular.

$$Q(s) = \left[\begin{array}{c|c} Q_{11}(s) & 0 \\ \hline Q_{21}(s) & Q_{22}(s) \end{array}\right] \qquad (3.11)$$

where:

$$\begin{aligned} Q_{11}(s) &= G_{11}(s)\ K_{11}(s) \\ Q_{22}(s) &= G_{22}(s)\ K_{22}(s) \\ Q_{21}(s) &= G_{21}(s)\ K_{11}(s) + G_{22}(s)\ K_{21}(s) \end{aligned}$$

$G_{21}(s)$ must be stable to ensure that $Q_{21}(s)$ has no unstable poles. (In order to fulfill requirement (a)).

The controller submatrix $K_{21}(s)$ is a feedforward controller designed to reduce or eliminate disturbances caused by $G_{21}(s)$. The ideal case assumes:

$$K_{21}(s) = -\ G_{22}(s)^{-1}\ G_{21}(s)\ K_{11}(s) \qquad (3.12)$$

Choosing $K_{21}(s)$ to satisfy (3.12) will ensure that $Q_{21}(s) = 0$, thus eliminating any disturbances due to $G_{21}(s)$.

The C/L system matrix $H(s) = (I + Q(s))^{-1}\ Q(s)$ for the ideal case, as shown in (3.13), will be block diagonal.

$$H(s) = \left[\begin{array}{c|c} H_{11}(s) & 0 \\ \hline 0 & H_{22}(s) \end{array}\right] \qquad (3.13)$$

where:

$$H_{11}(s) = (I + Q_{11}(s))^{-1} Q_{11}(s)$$
$$H_{22}(s) = (I + Q_{22}(s))^{-1} Q_{22}(s)$$

The following conclusions can once again be drawn:

(i) Interaction between Loops vs Stability of Closed Loop System

The term $K_{21}(s)$ relies on the inversion of the system submatrix $G_{22}(s)$ for elimination of interaction in the ideal case. The elements in $G_{22}(s)$ may contain right halfplane (abbreviated to RHP) zeroes. Inversion of such a matrix would introduce unstable RHP poles into the elements of $K_{21}(s)$ to cancel the RHP zeroes in $G_{22}(s)$. This pole-zero cancellation technique often does not succeed due to changes in the process model and modelling inaccuracies. The nett effect of this is the introduction of unstable poles in the submatrix $Q_{21}(s)$, which could cause instability of the process as a whole. The introduction of unstable poles into $Q(s)$ is a violation of requirement (a) for the choice of controller matrix $K(s)$.

If no unstable zeroes occur in the elements of $G_{22}(s)$, this technique of feedforward control is very effective in eliminating interaction between the loops.

(ii) Stability of Closed Loop System

All three controller subsections $K_{22}(s)$, $K_{11}(s)$ and $K_{21}(s)$ can be designed separately. Designing $K_{22}(s)$ and $K_{11}(s)$ to stabilize $Q_{22}(s)$ and $Q_{11}(s)$ respectively will ensure that the entire system is C/L stable. Stability is also subject to the condition that no unstable poles are introduced to $Q_{21}(s)$ by the design of $K_{21}(s)$ and $K_{11}(s)$.

3.2.3) Structure Adopted for the Control of Flows and Levels

A decision has been made to adopt a diagonal structure for the cascade compensator $K_c(s)$ for the heat exchanger process subsystem $G_c(s)$. It is aimed to deal with the problem of interaction by ensuring that the loop gains realized by the elements of $K_c(s)$ are sufficiently high to reject disturbances from other loops.

The choice of a diagonal structure for $K_c(s)$ is motivated by the following points listing the inherent advantages of such a structure:

(i) Simplicity of Design

Repetitive partitioning of $B(G_c(s))$ in (3.3) yields a 6x6 blocking shown in (3.14).

$$B(G_c(s)) = \begin{array}{c} \\ F1 \\ L4 \\ L3 \\ L2 \\ L1 \\ F2 \end{array} \begin{array}{c} \begin{array}{cccccc} C1 & C2 & C3 & C4 & C5 & C6 \end{array} \\ \left[\begin{array}{c|c|c|c|c|c} 1 & 0 & 0 & 0 & 0 & 0 \\ \hline 1 & 1 & 0 & 0 & 0 & 0 \\ \hline 1 & 1 & 1 & 0 & 0 & 0 \\ \hline 1 & 1 & 1 & 1 & 0 & 0 \\ \hline 1 & 1 & 1 & 1 & 1 & 0 \\ \hline 0 & 0 & 0 & 0 & 0 & 1 \end{array}\right] \end{array} \qquad (3.14)$$

$G_c(s)$ can now be represented by a block or composite matrix as shown in (3.15).

$$G_c(s) = \left[\begin{array}{c|c|c|c|c|c} gc_{11} & 0 & 0 & 0 & 0 & 0 \\ \hline gc_{21} & gc_{22} & 0 & 0 & 0 & 0 \\ \hline gc_{31} & gc_{32} & gc_{33} & 0 & 0 & 0 \\ \hline gc_{41} & gc_{42} & gc_{43} & gc_{44} & 0 & 0 \\ \hline gc_{51} & gc_{52} & gc_{53} & gc_{54} & gc_{55} & 0 \\ \hline 0 & 0 & 0 & 0 & 0 & gc_{66} \end{array}\right] \qquad (3.15)$$

The compensator required will have the identical block structure as the process model. The off-diagonal elements will all be zero for the block diagonal controller structure adopted. The composite matrix for the compensator is given in (3.16).

$$K_c(s) = \left[\begin{array}{c|c|c|c|c|c} kc_{11} & 0 & 0 & 0 & 0 & 0 \\ \hline 0 & kc_{22} & 0 & 0 & 0 & 0 \\ \hline 0 & 0 & kc_{33} & 0 & 0 & 0 \\ \hline 0 & 0 & 0 & kc_{44} & 0 & 0 \\ \hline 0 & 0 & 0 & 0 & kc_{55} & 0 \\ \hline 0 & 0 & 0 & 0 & 0 & kc_{66} \end{array}\right] \quad (3.16)$$

The following is noted concerning (3.15) and (3.16):

(a) Elements $gc_{ij}(s)$ and $kc_{ij}(s)$, which are functions in the complex variable s, are represented by gc_{ij} and kc_{ij} respectively, omitting the identifier '(s)'.

(b) The blocks in the composite matrices in (3.15) and (3.16) are all transfer function entries, and no longer submatrix entries.

The elements $kc_{ij}(s)$ can thus be designed using singlevariable (referred to as SISO) techniques on the corresponding elements $gc_{ij}(s)$. The MIMO design problem has simplified into 6 SISO designs.

(ii) Reduced Modelling Requirements

Only the diagonal submatrices of $G_c(s)$ need to be determined to design a stable $K_c(s)$ with a diagonal structure for the process (section 3.2.1). Only 6

transfer function elements, $gc_{ij}(s)$ (i = j = 1..6) thus need to be determined for the controller design. If a triangular structure were adopted for $\mathbf{K_c(s)}$, all the elements in $\mathbf{G_c(s)}$ (16 elements) would have to be determined for the design.

3.3) Determining Diagonal Elements of Subsystem $G_c(s)$ use in the Design of Flow and Level Compensator $K_c(s)$

The design of $\mathbf{K_c(s)}$ requires only the diagonal transfer function elements of the system submatrix $\mathbf{G_c(s)}$ to be known. The elements required are thus:

$$gc_{ij}(s) \ (i = j = 1..6) \qquad (3.17)$$

The diagonal elements $gc_{ij}(s)$ of $\mathbf{G_c(s)}$ are determined by stepping each control valve input j and observing the response of the appropriate flow or level output i (i = j = 1..6). A transfer function model for $gc_{ij}(s)$ relating the input to the appropriate output is then fitted to the response data.

Table 3.1 shows which input is stepped and output observed in the determination of each element.

Element $gc_{ij}(s)$	Input Stepped	Output Observed
$gc_{11}(s)$	C1(s)	F1(s)
$gc_{22}(s)$	C2(s)	L4(s)
$gc_{33}(s)$	C3(s)	L3(s)
$gc_{44}(s)$	C4(s)	L2(s)
$gc_{55}(s)$	C5(s)	L1(s)
$gc_{66}(s)$	C6(s)	F2(s)

Table 3.1: Inputs and Outputs Characterizing $gc_{ij}(s)$

The responses of the appropriate outputs to step tests done on the each input are shown in Appendix I. A model is fitted to each of the observed responses. Details of the modelling procedure are in Appendix I. The average transfer function model determined for each $gc_{ij}(s)$ is quoted in table 3.2.

Element $gc_{ij}(s)$	Transfer Function [(ADC units) / (DAC units)]	Dead-Time [sec]
$gc_{11}(s)$	$\frac{-1.66}{1 + 6.6s}$	0.0
$gc_{22}(s)$	$\frac{8.5}{1 + 575s}$	0.0
$gc_{33}(s)$	$\frac{6.0}{1 + 455s}$	15.0
$gc_{44}(s)$	$\frac{6.7}{1 + 490s}$	10.0
$gc_{55}(s)$	$\frac{7.9}{1 + 311s}$	9.0
$gc_{66}(s)$	$\frac{-1.19}{1 + 6.3s}$	0.0

Table 3.2: Average Transfer Function Models determined for elements $gc_{ij}(s)$ $(i = j = 1..6)$

3.4) Design of Flow and Level Compensator $K_C(s)$ for Heat Exchanger

3.4.1) Design Objectives

The following objectives are considered when designing the cascade compensator $K_C(s)$:

(i) Regulation of Flow and Level Outputs

Under C/L conditions, the control system is required to regulate the process flow and level outputs to their desired setpoints.

(ii) Improvement of Dynamics

Table 3.2 shows that the O/L response times of the levels vary between 311 and 575 seconds (poles between -0.00321 and -0.00174 rad/sec). Those for the flows vary between 6.3 and 6.6 seconds (poles between -0.158 and -0.151 rad/sec). C/L control is required to increase the speed of response of the levels.

(iii) Rejection of Disturbances

The structure of the $G_C(s)$ is lower triangular, meaning that diagonal elements of the system matrix are coupled to other elements 'higher up' in the system matrix (not

the other way round). For example, level L1 is coupled to, and thus will be affected by, the control valves C5..C1. Controllers for the elements 'lower down' in the system matrix will thus have to be designed to reject disturbances from the coupling to loops 'further up'. A controller for level L1 using SISO techniques on element $gc_{55}(s)$, for example, will have to be designed to reject disturbances due to the effects of elements $gc_{51}(s)..gc_{54}(s)$.

(iv) Closed Loop Stability

The resultant compensator design $\mathbf{K_c(s)}$ should ensure that the system is stable under C/L conditions at all times.

3.4.2) Design Procedure

The design of the diagonal compensator $\mathbf{K_c(s)}$ essentially consists of the design of 6 SISO controllers for the diagonal elements of the subsystem matrix $\mathbf{G_c(s)}$.

A proportional integral (abbreviate to PI) type configuration has been chosen for the elements of $\mathbf{K_c(s)}$. The integrating action of such controllers has good regulation and disturbance rejection properties. A design analysis procedure based on the techniques of Nyquist (Raven, 1987) has been adopted for each of the SISO designs.

The procedure for the design of the element $kc_{33}(s)$ of $\mathbf{K_c(s)}$ is shown in this section. The procedure for each of the other elements $kc_{ij}(s)$ (i = j = 1,2, 4..6) is identical, and these designs are shown in Appendix I. The resultant design for $\mathbf{K_c(s)}$ is shown in section 3.4.3.

Element $kc_{33}(s)$ is designed using SISO techniques on the element $gc_{33}(s)$ of $\mathbf{G_c(s)}$. From table 3.2,

$$gc_{33}(s) = \frac{6.7\ e^{-15s}}{1 + 490s} \qquad (3.18)$$

The simulated O/L time response of the element is given in figure 3.2, showing the slow 490 second response time.

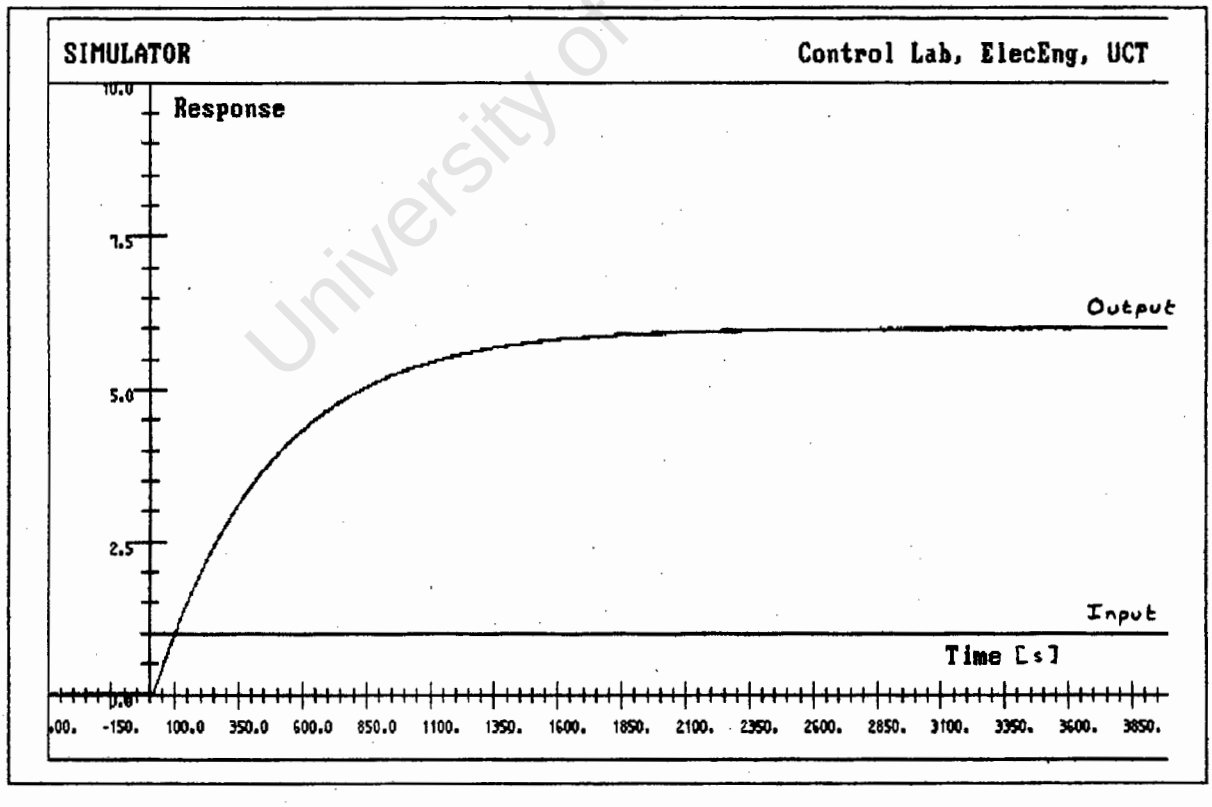


Figure 3.2: Simulated Open loop Time Response of Level L3 to a Unit Step in Control Valve C3

Frequency dependent analysis of this element is assisted by the polar Nyquist plot of $gc_{33}(jw)$ shown in figure 3.3. The analysis predicts the feedback properties of the subsystem that will result if the loop is closed on the uncompensated element $gc_{33}(s)$. The C/L transfer function $hc_{33}(s)$ of this uncompensated subsystem is defined in (3.19).

$$hc_{33}(s) = (1 - gc_{33}(s))^{-1} gc_{33}(s) \qquad (3.19)$$

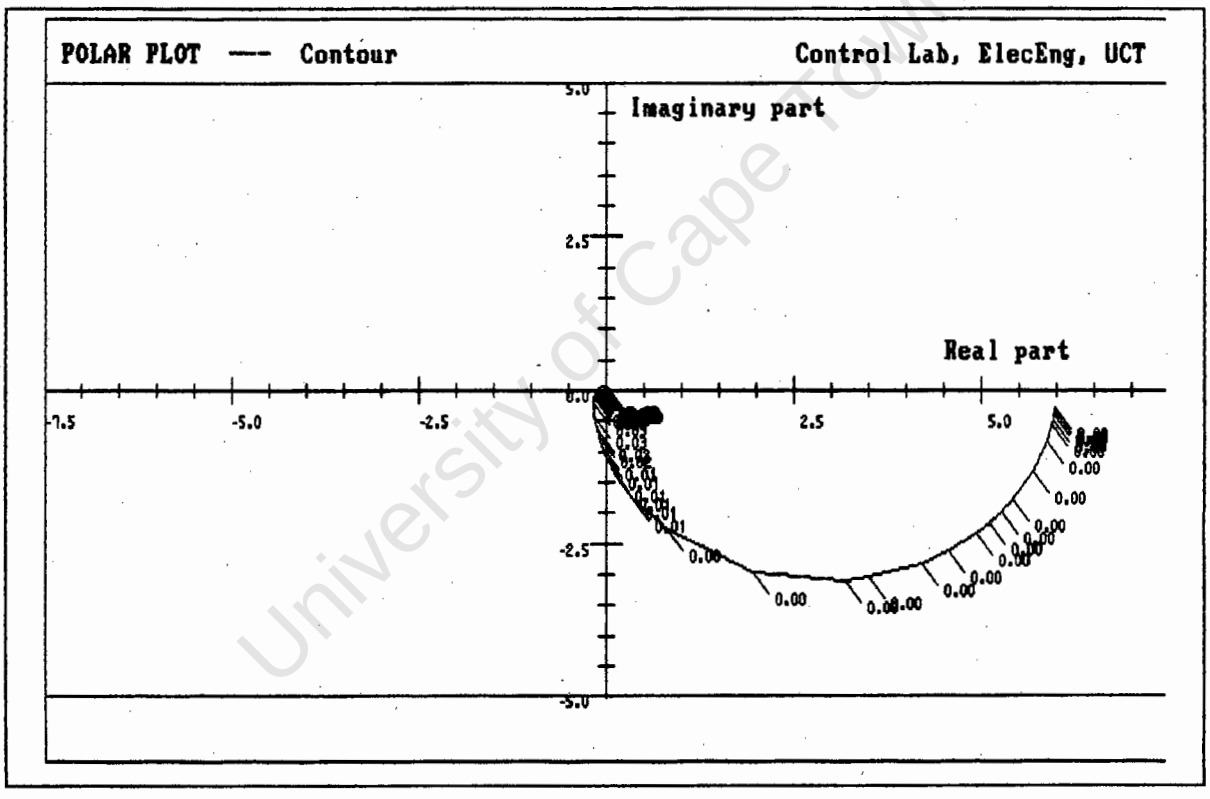


Figure 3.3: Nyquist Plot for element $gc_{33}(jw)$ of $G_c(s)$

The effect of the 15 second dead-time can be seen by the encirclement of the origin by the plot. No encirclements of the s = -1 point are present, and the gain and phase margins

exceed 5 and 60° respectively. The subsystem hc33(s) in (3.19) will thus be stable under unity feedback conditions.

The plot never enters the region between the M = 1.1 and M = 0.9 circles, meaning that the uncompensated C/L gain $|hc_{33}(jw)|$ will never be near unity. The output regulation will thus be poor if the loop is closed.

The rejection of disturbances by the output requires that the O/L gain of the process, $|gc_{33}(jw)|$ is large. Rejection of D.C disturbances by closing the loop requires infinite O/L gain at D.C.. The maximum O/L gain occurs at D.C., but is limited to 6.7 meaning that the transmission gain between disturbances and output under C/L conditions will equal 0.13. D.C. disturbances will thus have a significant effect on the output. The magnitude $|gc_{33}(jw)| = 1$ at a frequency as low as 0.01 rad/sec implies that the C/L transmission gain at this frequency and above will exceed 0.5. Rejection of disturbances by the output will thus be poor if the loop is closed.

The poor feedback properties of the subsystem are to be improved by inserting a cascade PI compensator in the loop. Iterative design has produced the compensator $kc_{33}(s)$ shown in (3.20).

$$kc_{33}(s) = \frac{4 + 320s}{80s} \qquad (3.20)$$

The compensated O/L relation $qc_{33}(s)$ is defined in (3.21), and the resultant C/L transfer function $hc_{33}(s)$ is shown in (3.22).

$$qc_{33}(s) = gc_{33}(s)\, kc_{33}(s) \qquad (3.21)$$

$$hc_{33}(s) = (1 - qc_{33}(s))^{-1}\, qc_{33}(s) \qquad (3.22)$$

The feedback properties of this compensated subsystem can now be determined from the Nyquist plot of $qc_{33}(jw)$ shown in figure 3.3.

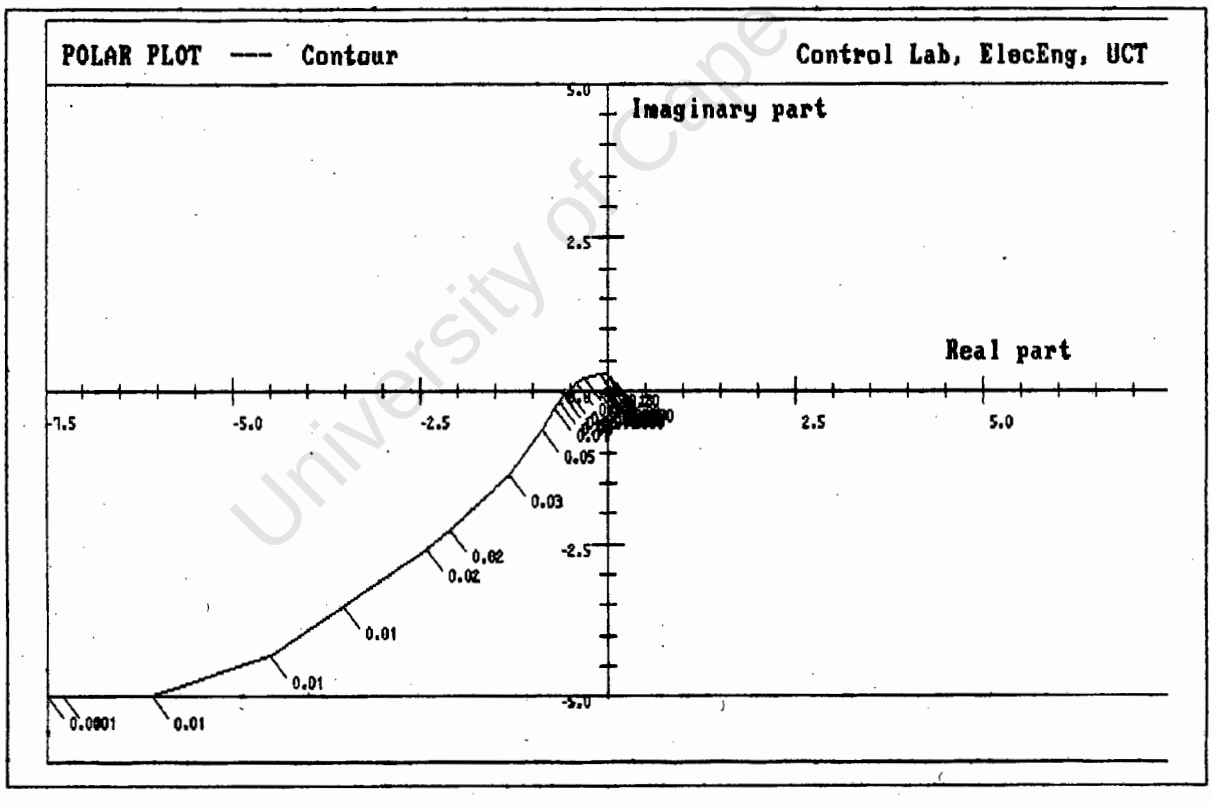


Figure 3.4: Nyquist Plot for Element $qc_{33}(jw)$

No encirclements of the s = -1 point are present, but a significant reduction in both gain and phase margin is

observed. These reductions are not however sufficient to make the compensated C/L subsystem go unstable.

The plot stays in the region between the M = 1.1 and M = 0.9 circles for all frequencies below 0.02 rad/sec, meaning that good output regulation will occur over this frequency band (best at D.C. due to integrating action) and the C/L response time should be in the order of 10 seconds.

The O/L gain is infinite at D.C., meaning that the no D.C. disturbances will affect the output if the loop is closed. The O/L gain is also large (greater than 5) for all frequencies less than 0.01 rad/sec, meaning that all disturbances at frequencies less than 0.01 rad/sec will be rejected with a C/L transmission gain of less than 0.16.

The plot of the simulated time response of the C/L subsystem to a unity step in the level setpoint (figure 3.3) shows that the output response time has been reduced from 490 seconds (O/L) to 12 seconds (C/L), as predicted. The output is also successfully regulated to its setpoint.

The price paid for this improvement in performance is large input excursions shown in the plot and a reduction in the stability of the system causing the observed oscillation (slight) and overshoot. A 55% percentage overshoot, M_p is observed in the output response.

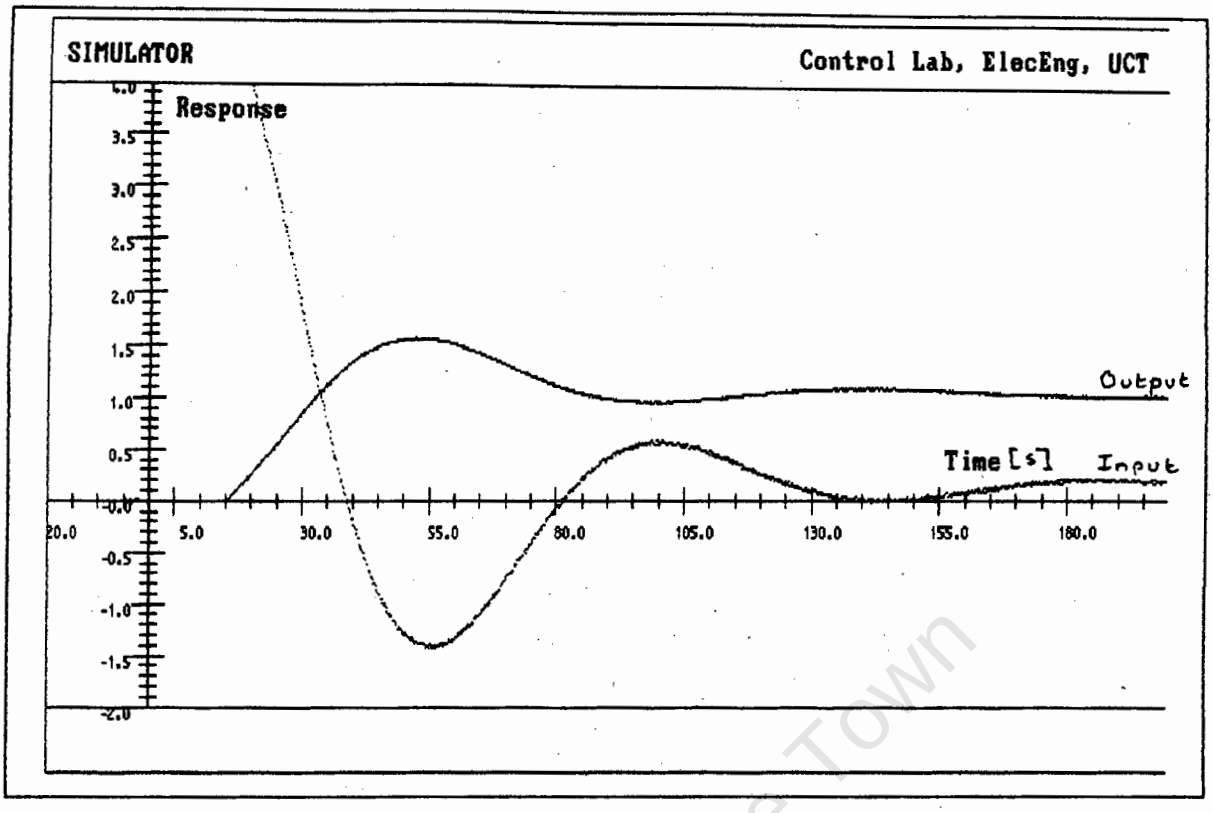


Figure 3.5: Simulated Closed Loop Time Response of Level L3 to a Unity Step in its Setpoint

3.4.3) Resultant Design for $K_C(s)$

The design procedure outlined in the previous section was implemented in the design of the remaining elements of $K_C(s)$. The design details are shown in Appendix J. The results of these designs for each element are quoted in table 3.3.. The resultant output response times and percentage overshoot obtained from closed loop simulations shown in the appendix are also quoted in the table for each flow or level output.

Element $kc_{ij}(s)$	Transfer Function [(ADC units) / (DAC units)]	Output F or L	Res Time [sec]	M_p [%]
$kc_{11}(s)$	$\frac{-0.6 - 3.96s}{6.6s}$	F1	6.6	0
$kc_{22}(s)$	$\frac{4 + 320s}{80s}$	L4	15.0	10
$kc_{33}(s)$	$\frac{4 + 320s}{80s}$	L3	12.0	55
$kc_{44}(s)$	$\frac{4 + 320s}{80s}$	L2	10.8	30
$kc_{55}(s)$	$\frac{4 + 320s}{80s}$	L1	7.2	55
$kc_{66}(s)$	$\frac{-0.84 - 5.29s}{6.3s}$	F2	9.3	0

Table 3.3: Cascade PI Controller Transfer Function Elements and Response Times of Outputs

3.4.4) Closed Loop Performance of System with Cascade Compensator $K_c(s)$

The diagonal cascade PI compensator $K_c(s)$ has been incorporated in the feedback control loop for the regulation of the flow and level outputs. The C/L response of level L3, and the effects on the other levels are observed in this section. The C/L responses of the other levels as well as the flows and their effects on other loops are observed in Appendix K. An analysis of the resultant responses is finally given in table 3.4 at the end of the section, showing

to what extent the design objectives stated in 3.4.1 have been met.

Figure 3.6 shows the C/L response of level L3 to a positive 10% (410 ADC units) change in setpoint. The response time of the output has improved from 455 seconds under O/L conditions (table 3.2) to 15.9 seconds under these C/L conditions (figure 3.6).

The effects of stepping L3 on the other tank levels are shown in figure 3.7. Only levels L2 and L1 ('lower down' in system matrix), as expected, are affected by changing L3 setpoint. These effects are compensated for by the high-gain controller elements $kc_{44}(s)$ and $kc_{55}(s)$ designed to reject disturbances from other loops. Both levels L2 and L1 suffer a transient perturbation only, which is less than 5% of the full scale level reading. This represents a maximum transient transmission gain of 0.5 and zero steady-state transmission gain between the setpoint L3 and the outputs L2 and L1.

No effects (obviously - no physical connection and $K_c(s)$ decentralized controller) on the flow outputs were observed as a result of this change in level setpoint.

The entire system remained stable under C/L conditions, and the observed overshoot of 15.2% indicates that the phase margin is in fact larger than predicted in section 3.4.3.

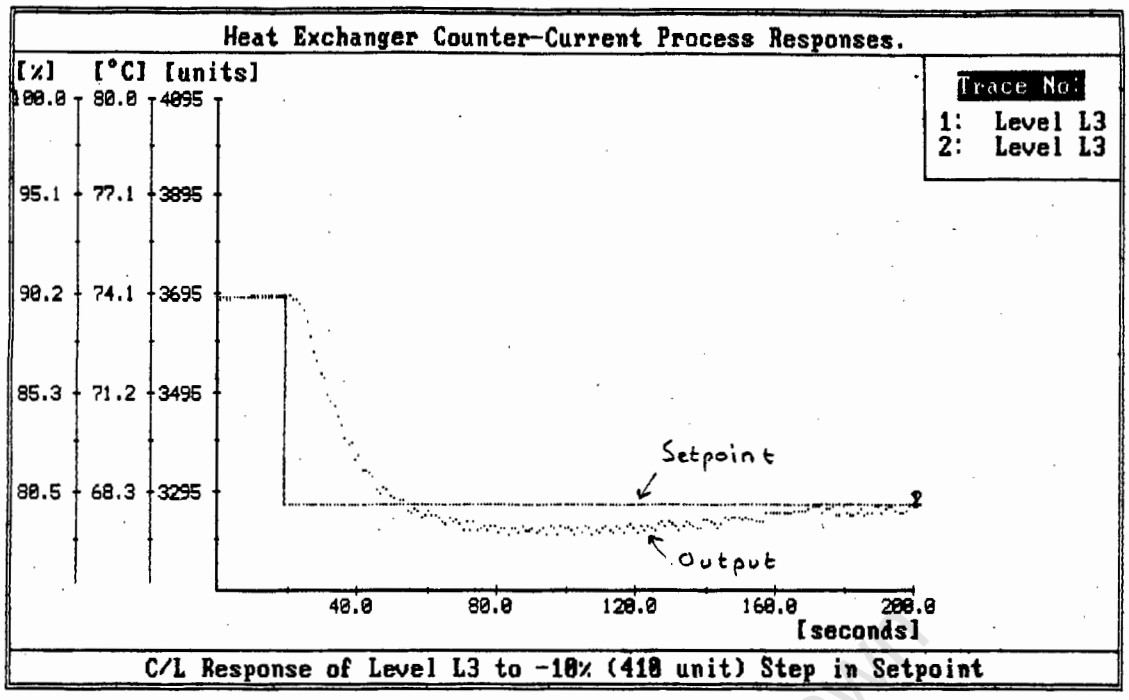


Figure 3.6: Closed Loop (with $K_c(s)$) Response of L3

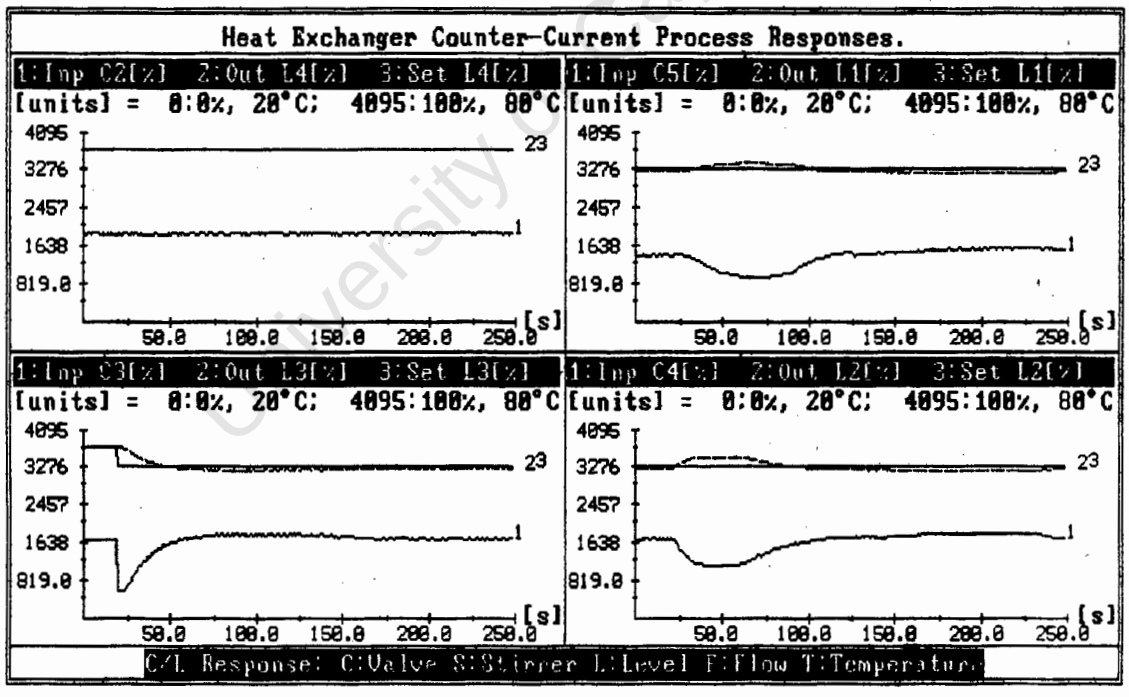


Figure 3.7: Interaction Caused by Stepping L3 Setpoint

The results from C/L step tests done on the remaining flow and level outputs are shown in Appendix K. Table 3.4 generated from these results summarizes how the design objectives stated in section 3.4.1 have been met. The process remained stable under all conditions while tests were being performed.

Setpoint Stepped	% Error in Output (% of Max)	Response Time (sec)	Loop Coupling to Other Outputs (Transmit. Gain)	Stability and Overshoot
F1	< 1 %	4.4	< 0.3 Transient Zero S-State to L4..L1	Stable, Oversht: 0.0 %
L4	< 1 %	11.5	< 0.7 Transient Zero S-State to L3..L1	Stable, Oversht: 0.5 %
L3	< 1 %	15.9	< 0.5 Transient Zero S-State to L2..L1	Stable, Oversht: 15.2 %
L2	< 5 %	19.7	< 0.5 Transient Zero S-State to L1 only	Stable, Oversht: 14.7 %
L1	< 5 %	12.2	No Coupling to Any of the Other Outputs	Stable, Oversht: 16.2 %
F2	< 1 %	2.7	No Coupling to Any of the Other Outputs	Stable, Oversht: 0.0 %

Table 3.4: Closed Loop Performance of Subsystem $G_c(s)$ with Flow and Level Precompensator $K_c(s)$

3.5) Concluding Remarks

The upper left submatrix $G_c(s) = G_{11}(s)$, of the process model G(s), relates the 6 control valve inputs to the 2 flow and 4 level outputs, and has a lower-triangular structure. A diagonal cascade PI compensator $K_c(s)$ has successfully been designed and implemented to control the flows and levels under C/L conditions.

First order models have been fitted to response data generated to obtain the diagonal elements of $G_c(s)$. The response times of the flows vary between 6.3 and 6.6 seconds, while those for the levels vary between 311 and 575 seconds.

The control system designed for the process succeeded in satisfying the design objectives, and was found to:

i) Regulate of Flow and Level Outputs to their respective setpoints with less than 1% (of full scale) steady-state error in the flow outputs and less than 5% steady-state error in the level outputs.

ii) Improve the dynamics of the output responses. The flow response times were reduced to between 2.7 and 4.4 seconds while those for the levels were reduced to between 11.5 and 19.7 seconds.

iii) Compensate to reduce interaction between the loops in the control system. Steady-state transmission between loops has been eliminated, and outputs are only transiently effected by changes in setpoints in other loops. Transient gains between setpoints and outputs of other loops were found to be less than 0.7 in all cases.

The price paid for the improvement in these feedback properties of the system were found to be:

i) Reduction in the stability of the system. Levels were found to overshoot their setpoints by between 0.5 and 16.2% and slight decaying oscillations were observed in the level outputs and control valve inputs.

ii) Large excursions in the control valve inputs.

CHAPTER FOUR

PERFORMANCE INDEX ANALYSIS OF MIMO CONTROL SYSTEMS

4.1) Feedback Properties and Performance Indices

The main objective of this dissertation, as stated in the introduction, is the analysis of the frequency dependent properties of MIMO control system designs, not the actual design procedure. A quantitative measure for the "goodness" of the properties of a MIMO feedback design therefore needs to be developed.

4.1.1) Definition of Control System Feedback Properties

The frequency dependent properties of a feedback control system design are referred to as its 'feedback properties'. The following feedback properties are considered in this dissertation:

(i) *Response of Inputs and Outputs to Process Disturbances and Noise in the Sensors as a Function of Frequency*

(ii) *Sensitivity of Inputs and Outputs to Changes in Process Model as a Function of Frequency*

(iii) *Closed Loop Stability of Process*

The properties listed are referred to as 'feedback properties' because they can only be improved (Safonov, 1981)

using C/L feedback. O/L control systems will not change these properties at all. A control system design with good feedback properties is referred to as a 'robust' system (Safonov, 1981).

Another property considered is that of setpoint tracking by the process outputs as a function of frequency. This is not strictly a feedback property (Safonov, 1981), although much theoretical discussion has centered around the improvement of this property using feedback (eg integral action in loop ensures unity C/L gain). Modern approaches to the problem of output regulation suggest a different route, namely, first ensuring that the system is robust before designing a dynamic precompensator to facilitate setpoint tracking. The property of setpoint tracking remains an important one however, so it is included in the discussion of feedback properties presented in this chapter. Coupled to this property of output regulation, is the property of control effort (input response to setpoint) as a function of frequency, which is also considered in this dissertation.

4.1.2) Definition of Performance Indices

The measure of the frequency dependent "goodness" of a particular feedback property is defined as the "performance index" for that property. Each index then quantitatively gauges the performance of the control system in terms of a particular feedback property as a function of frequency.

4.2) Feedback Properties of Generalized MIMO and SISO Control Systems

The feedback properties of a generalized MIMO control system are discussed in this section. All matrices referred to are assumed to be nxn linear mappings within the complex linear space C_n, and all vectors are assumed to be n-dimensional elements of C_n. A SISO control system is assumed to be a special 1x1 case of the generalized control system representation in which the matrices reduce to scalar ratios and the vectors reduce to scalars.

The discussion points to a method of analysis of the feedback properties of MIMO control systems which is based on some representation for the frequency dependent sizes of matrices discussed later on in this section.

The method of analysis of SISO control systems is treated as a special case of the method for generalized systems. The relationships between this method of analysis and the classical Nyquist-type (Raven, 1987) analysis method for SISO systems are pointed out.

4.2.1) Basic Control Loop Definitions

In this section, definitions are made for a generalized nxn MIMO control loop, and the special 1x1 SISO cases of these definitions are pointed out.

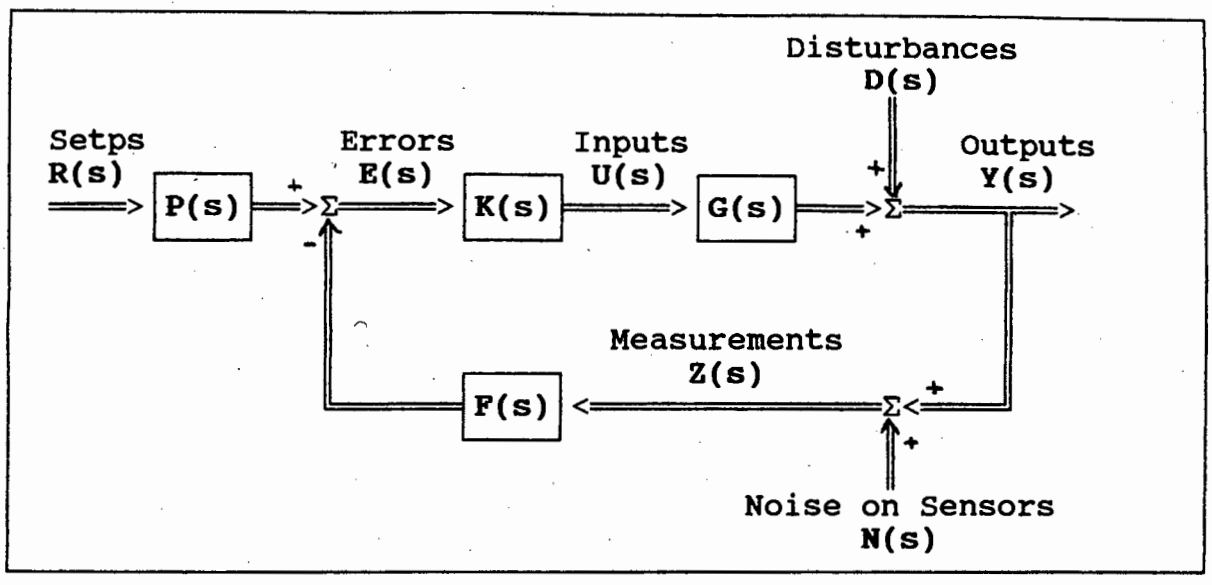


Figure 4.1: Block Diagram of Generalized MIMO Feedback Control System Including Process Disturbances and Noise on Sensors

Elements in the generalized MIMO control loop shown in figure 4.1 are:

G(s) : Process Transfer Function Matrix

K(s) : Cascade Compensator Transfer Function Matrix

F(s) : Feedback Compensator Transfer Function Matrix

P(s) : Precompensator Transfer Function Matrix

Considering the system shown in figure 4.1, the following relationships are deduced:

Plant

$$\mathbf{Y(s)} = \mathbf{G(s)U(s)} + \mathbf{D(s)} \qquad (4.1)$$

Controller

$$\mathbf{U(s)} = \mathbf{K(s)\{P(s)R(s) - F(s)Y(s)\}} \qquad (4.2)$$

Sensor

$$\mathbf{Z(s)} = \mathbf{Y(s)} + \mathbf{N(s)} \qquad (4.3)$$

where **Y(s)**, **U(s)**, **D(s)**, **R(s)**, **Z(s)** and **N(s)** are vectors defined in figure 4.1.

The following characteristic matrices are defined:

Feedback Loop Transfer Matrix

$$\mathbf{M(s)} = \mathbf{K(s)F(s)} \qquad (4.4)$$

Return Matrix at Input Node

$$\mathbf{L_2(s)} = \mathbf{G(s)M(s)} \qquad (4.5)$$

Return Matrix at Output Node

$$\mathbf{L_1(s)} = \mathbf{M(s)G(s)} \qquad (4.6)$$

Return Difference Matrix at Input Node

$$\mathbf{I} + \mathbf{L_2(s)}$$

Return Difference Matrix at Output Node

$$I + L_1(s)$$

Inverse Return Difference Matrix at Input Node

$$I + L_2^{-1}(s) \quad \text{(if the inverse, } L_2^{-1}(s) \text{ exists)}$$

Inverse Return Difference Matrix at Output Node

$$I + L_1^{-1}(s) \quad \text{(if the inverse, } L_1^{-1}(s) \text{ exists)}$$

Note that in the SISO case, $l_1(s) = l_2(s) = l(s)$. Here $l(s)$ is known as the *return ratio* while $(1 + l^{-1}(s))$ is known as the *return difference ratio*. These ratios are independent of the input or output nodes.

Additional characteristic matrices which are useful in the analysis of the feedback properties of a control system are also defined (for i = 1..2):

$$S_i(s) = (I + L_i(s))^{-1} \tag{4.7}$$

$$\begin{aligned} T_i(s) &= I - S_i(s) \\ &= L_i(s)(I + L_i(s))^{-1} \\ &= (I + L_i(s))^{-1}L_i(s) \end{aligned} \tag{4.8}$$

$$\begin{aligned} W(s) &= S_1(s)M(s) \\ &= M(s)S_2(s) \end{aligned} \tag{4.9}$$

Inspection of the definitions in (4.4) to (4.9) facilitates the derivation of the following identities:

$$T_2(s) = G(s)W(s) \tag{4.10}$$

$$T_1(s) = W(s)G(s) \tag{4.11}$$

Also for i = 1..2,

$$T_i(s) = (I + L_i^{-1}(s))^{-1} \quad (4.12)$$

$$W(s) = (I + L_2^{-1}(s))G(s) \quad (4.13)$$

$$W(s) = G(s)(I + L_1^{-1}(s)) \quad (4.14)$$

For the SISO case, $s_1(s) = s_2(s) = s(s)$, $t_1(s) = t_2(s) = t(s)$ and $w(s) = m(s)s(s) = t(s)g(s)$.

4.2.2) Transmission Matrices and Transmission Ratios

The responses of **U(s)** and **Y(s)** can be represented as the sums of three components generated by **R(s)**, **D(s)** and **N(s)** respectively, as shown in (4.15).

$$\begin{bmatrix} Y(s) \\ U(s) \end{bmatrix} = \begin{bmatrix} Y_R(s) \\ U_R(s) \end{bmatrix} + \begin{bmatrix} Y_D(s) \\ U_D(s) \end{bmatrix} + \begin{bmatrix} Y_N(s) \\ U_N(s) \end{bmatrix} \quad (4.15)$$

Simple block diagram manipulation of figure 4.1 and the application of equations (4.1) to (4.14) show that:

$$\begin{aligned} \begin{bmatrix} Y_R(s) \\ U_R(s) \end{bmatrix} &= \begin{bmatrix} G(s)S_1(s)K(s)P(s) \\ S_1(s)K(s)P(s) \end{bmatrix} R(s) \\ &= \begin{bmatrix} S_2(s)G(s)K(s)P(s) \\ S_1(s)K(s)P(s) \end{bmatrix} R(s) \\ &= \begin{bmatrix} T_{YR}(s) \\ T_{UR}(s) \end{bmatrix} R(s) \end{aligned} \quad (4.16)$$

$$\begin{bmatrix} Y_D(s) \\ U_D(s) \end{bmatrix} = \begin{bmatrix} S_2(s) \\ -\bar{W}(s) \end{bmatrix} D(s)$$

$$= \begin{bmatrix} T_{YD}(s) \\ T_{UD}(s) \end{bmatrix} D(s) \qquad (4.17)$$

$$\begin{bmatrix} Y_N(s) \\ U_N(s) \end{bmatrix} = \begin{bmatrix} -T_2(s) \\ -\bar{W}(s) \end{bmatrix} N(s)$$

$$= \begin{bmatrix} T_{YN}(s) \\ T_{UN}(s) \end{bmatrix} N(s) \qquad (4.18)$$

The matrices $T_{YR}(s)$, $T_{UR}(s)$, $T_{YD}(s)$, $T_{UD}(s)$, $T_{YN}(s)$ and $T_{UN}(s)$ defined in (4.16) to (4.18) are referred to as "transmission matrices".

(i) Performance of MIMO systems

The degree to which the inputs and outputs respond to perturbations in the setpoints, disturbances and noise on the sensors is related to the "sizes" of the appropriate transmission matrices. These "sizes" are in turn related to the "sizes" of the characteristic matrices $W(s)$, $T_i(s)$ and $S_i(s)$. A measure for the "size" of matrices is thus required to give a quantitative representation of the transmission feedback properties of a MIMO system. The measure for size will be dependent on the frequency at which it is being made.

(ii) Performance of SISO systems

For the SISO case, the transmission matrices reduce to the scalar "transmission ratios". The input and output responses are easily determined by measuring the magnitude and phase of these ratios as a function of frequency. These measures quantify the transmission feedback properties of a SISO system entirely.

Ideally, the magnitude of the ratio $t_{yr}(s)$ should be unity over all frequencies, meaning that the output component $y_r(s)$ will follow all setpoint perturbations r(s) (ignoring phase at this stage). This condition is equivalent to ensuring that the Nyquist plot (Raven, 1987) for g(jw)k(jw) remains near the M = 1 circle. The performance in terms of setpoint tracking can thus be determined by observing the proximity of the plot to the M = 1 circle. The phase of the ratio indicates the relative phase shift between the setpoint and the resultant output response of the system as a function of frequency.

The magnitude of the ratio $t_{ur}(s)$ should be kept as small as possible over all frequencies to ensure minimal control effort. The phase, as before, represents the phase shift between the setpoint and input.

The response of the output and input to disturbances is characterized by the frequency dependent scalar ratios $t_{yd}(s) = s(s)$ and $t_{ud}(s) = -w(s)$ respectively (equation (4.17)). The performance in terms of these transmission feedback properties can be determined quantitatively from the magnitude and phase of these ratios. Small magnitudes of s(s) and w(s) at a given frequency will ensure good rejection of disturbances by the output and input respectively at that frequency.

The response to noise is determined from the ratios $t_{yn}(s) = -t(s)$ and $t_{un}(s) = w(s)$. (equation (4.18)). The performance in terms of these feedback properties can be determined as before.

Response to disturbances and noise can also be analysed using the Nyquist plot. Large magnitudes of the open loop gain $|g(jw)k(jw)|$ at some frequency implies good rejection of disturbances by the output at that frequency. This also, unfortunately, means that sensor noise and disturbances at this frequency will be amplified at the input.

4.2.3) Sensitivity Matrices and Sensitivity Ratios

Assume that the controller matrices **P(s)**, **K(s)** and **F(s)** in figure 4.1 remain unchanged at all frequencies, as well as the setpoint signals **R(s)**. Assume also that all disturbances

D(s) and sensor noises N(s) are zero. The nominal model for the process is referred to as $G_0(s)$. The "sensitivity matrices" $SE_Y(s)$ and $SE_U(s)$ relate the changes in G(s) from its nominal model to the resultant changes in the closed loop transmission ratios $T_{YR}(s)$ and $T_{UR}(s)$. Theorems stated and proved in Appendix L derive the sensitivity relations given in (4.19) and (4.20).

$$\delta(T_{YR}(s)) = SE_Y(s)\ \delta(G(s)) = S_2(s)\ \delta(G(s)) \qquad (4.19)$$

$$\delta(T_{UR}(s)) = SE_U(s)\ \delta(G(s)) = T_1(s)\ \delta(G(s)) \qquad (4.20)$$

where:

$\delta(G(s))$ represents change in process model G(s)

$\delta(T_{YR}(s))$ represents resultant change in C/L setpoint to output transmission matrix $T_{YR}(s)$

$\delta(T_{UR}(s))$ represents resultant change in C/L setpoint to input transmission matrix $T_{UR}(s)$

Note that $S_2(s)$ and $T_1(s)$ are assumed to be the characteristic matrices obtained for the nominal model for G(s) (ie for $G(s) = G_0(s)$).

(i) Performance of MIMO systems

The transmission of phase and amplitude invariant setpoint signals at a given frequency to the inputs and outputs will change according to (4.19) and (4.20). The resultant phase and amplitude of the input and output signals at that frequency will thus change as the process model changes. The degree to which the signals will change is related to the "sizes" of the sensitivity matrices, and thus ultimately to the "sizes" of the characteristic matrices $S_2(s)$ and $T_2(s)$. The input and output sensitivity feedback properties can thus be quantitatively represented by some expression for the frequency dependent "sizes" of these matrices.

(ii) Performance of SISO systems

The sensitivity matrices reduce to the "sensitivity ratios" $se_y(s)$ and $se_u(s)$ for a SISO system. The magnitude and phase of these ratios as a function of frequency thus quantify the input and output sensitivity feedback properties for the SISO case.

Observation of the sensitivity ratio magnitude $|se_y(jw)|$ at some frequency gives a quantitative representation of the sensitivity of the output at that frequency to changes in g(s). This is equivalent to observing the

proximity of a Nyquist plot for q(jw) to sensitivity circles (Raven, 1987) on the axes at that frequency.

4.2.4) Stability Analysis of MIMO and SISO Control Systems

The stability margin (Safonov, 1981) of a system can be defined as the "size" of the "smallest" perturbation $\delta(L_i(s))$ (over all frequencies) in the return matrix, from its nominal value based on $G_0(s)$, such that the system matrix $S_i(s) = (I + L_i(s))$ goes unstable. Under this condition, the feedback system (4.1) to (4.3) is destabilized.

It is further shown that $S_i(s)$ is not asymptotically stable iff there exists some complex frequency s_0 with $Re(s_0) \geq 0$ and some nonzero $x_1 \in C_n$ such that

$$x_1 = -\delta(L_i(s_0))\, T_i(s)\, x_1 \qquad (4.21)$$

Note that the matrix $T_i(s)$ is once again assumed to be the matrix obtained for the nominal model, $G_0(s)$.

(i) Performance of MIMO systems

Equation 4.21 forms the basis from which the stability of a MIMO feedback system can be analysed. This equation shows that the "size" of the "smallest"

$\delta(L_i(s))$ for which the system is not asymptotically stable, is in some sense inversely proportional to the "size" of $T_i(s)$ for values of complex frequency s in the right-half of the complex s-plane.

(ii) Performance of SISO systems

The special 1x1 SISO case of 4.21 is once again considered, and it is seen that in this case, equation 4.21 reduces to:

$$\delta(1(s_0)) = -1/t(s_0) \qquad (4.22)$$

The classical SISO gain margin gm is computed by considering $\delta(1(s_0)) = gm - 1$, leading to the conclusion that the gain margin is the least constant $gm > 1$ for some $Re(s_0) \geq 0$ in equation (4.23).

$$gm = -1/t(s_0) + 1 \qquad (4.23)$$

The value for s_0 satisfying the condition in (4.23) is a -180° crossover frequency $s_0 = jw_{180}$ satisfying (4.24).

$$angle(1(jw_{180})) = -180^\circ \qquad (4.24)$$

This result is consistent with the definition of gain margin on a Nyquist plot. Considering the case where $f(s) = 1$, $p(s) = 1$ for simplicity, it turns out that

$l(s) = q(s) = g(s)k(s)$. The result is consistent as the gain margin is defined at the frequency point where the polar plot of $q(jw)$ has a phase angle of -180°.

The classical SISO phase margin φ_M is similarly computed for $\delta(l(s_0)) = \exp(-j\varphi_M) - 1$, leading to the conclusion that the phase margin is the smallest angle $\varphi_M > 0$ (in radians) for some $Re(s_0) \geq 0$ in equation (4.25).

$$j\varphi_M = \ln(-1/t(s_0) + 1) \qquad (4.25)$$

The value for s_0 satisfying the condition in (4.25) is a 0dB crossover frequency $s_0 = jw_c$ satisfying (4.26).

$$|l(jw_c)| = 1 \qquad (4.26)$$

This result is once again consistent with the definition of phase margin on a Nyquist plot as the phase margin is defined at the frequency point where the polar plot of $q(jw)$ has a magnitude of 1 (equivalent to 0dB).

4.2.5) Effects on MIMO and SISO Feedback Properties

Equations 4.15 to 4.26 show that the "sizes" of the characteristic matrices $W(s)$, $S_1(s)$, $S_2(s)$, $T_1(s)$, $T_2(s)$ as a function of frequency are directly related to the quality of the feedback properties of a MIMO control system.

In this subsection, the effects of the characteristic matrices becoming vanishingly small are discussed. The effects of the corresponding characteristic ratios vanishing are identical for SISO control systems. Certain fundamental design trade-offs are also discussed.

(i) Effects of Vanishing Characteristic Matrices

If, at some complex frequency s, **W(s)** vanishes, then $T_1(s)$ and $T_2(s)$ also vanish (equations (4.10) and (4.11)) and the following desirable feedback properties result:

a) $U_D(s)$, $Y_N(s)$ and $U_N(s)$ vanish, making inputs and outputs insensitive to sensor noise, as well as making inputs insensitive to disturbances. (Equations (4.17) and (4.18))

b) $SE_U(s)$ vanishes, making the inputs insensitive to changes in the process model. (Equation (4.20))

c) $\delta(L_i(s))$ approaches infinity, meaning that an arbitrarily large $\delta(L_i(s))$ cannot destabilize the system. (Equation (4.21))

If however, at this frequency, $S_1(s)$ and $S_2(s)$ vanish, the following desirable feedback properties result:

a) $Y_D(s)$ vanish, making outputs insensitive to disturbances. (Equation (4.17))

b) $SE_Y(s)$ vanishes, making the outputs insensitive to changes in the process model. (Equation (4.19))

(ii) Design Trade-Offs

The identity $S_i(s) + T_i(s) = I$ is obtained from equation (4.8), and implies that both $S_i(s)$ and $T_i(s)$ cannot be made to vanish at the same frequency. The design of a feedback control system then becomes trade-off between the "smallness" of $T_i(s)$ against the "smallness" of $S_i(s)$ (i = 1..2) over certain frequency ranges. It is impossible, for example, to design a system with good rejection of disturbances by the outputs and rejection of noise by the inputs at the same frequency. Common design practice is to arrange good output rejection of disturbances at frequencies near DC, while arranging good rejection of sensor noise by the inputs at high frequencies.

The need to meaningfully quantify the "size" of frequency dependent matrices in the analysis of such control systems is therefore apparent.

The quality of the feedback properties for the SISO case is adequately described by the frequency dependent magnitude and phase of the characteristic ratios. No further representation for the size of these ratios is necessary as in the MIMO case. The design of a SISO control system thus consists of trading off the magnitude and phase of the ratio t(s) against the magnitude and phase of the ratio s(s).

4.3) Quantitative Analysis of Feedback Properties of SISO and MIMO Control Systems

4.3.1) Inadequacies of Current MIMO Analysis Techniques

Two common MIMO control system analysis techniques are the Characteristic Loci (CL) (Fisher, 1988) and Inverse Nyquist Array (INA) (Venzke, 1988) techniques. Quantitative analysis of most of the feedback properties discussed in section 4.2 is not easily possible using these techniques. CL techniques are useful in the stability analysis of MIMO systems, whereas INA techniques are good at diagnosing interaction between loops in such systems (Fisher, 1988). Extracting information from INA and CL analysis plots regarding the transmission of

disturbances and sensor noise, as well as regarding sensitivity, is not a trivial task. Further limitations of these techniques are also pointed out (Doyle & Stein, 1981).

As a result of the inadequacies of existing MIMO analysis techniques, some technique of quantitatively representing the feedback properties needs to be developed. Representations based on the transmission, sensitivity and stability matrices have been developed in various papers (Doyle & Stein, 1981), (Safonov, 1981). Such representations are introduced in section 4.2.

As discussed in section 4.2, the quality of the feedback properties of a MIMO control system is related to the frequency dependent "sizes" of the characteristic matrices $\mathbf{W(s)}$, $\mathbf{S_i(s)}$ and $\mathbf{T_i(s)}$ (i = 1..2). A popular choice as a representation of the "size" of a matrix is that of the eigenvalues of the matrix. This representation, however, has some shortcomings, as can be seen in two examples presented in Safonov's paper (Safonov, 1981).

4.3.2) Singular Values as Expression of "Size" of Matrix

A meaningful measure of the "size" at some frequency of a system described by a transfer function matrix, in the control system context, could be one which bears some relation to the input-output gain ratio of the system at that

frequency. A number of definitions are made in this subsection as a build-up to such a representation of matrix "size".

(i) Definition of Euclidean Norm of a Vector

Given a positive definite matrix $Q \in C^{nxn}$, define a unitary space C^n_Q as the set of complex n-vectors $X(s) = (x_1(s),....,x_n(s))^T$ together with Euclidean norm $\|\cdot\|_Q$ defined in (4.27)

$$\|X(s)\|_Q = (X(s)^T Q X(s))^{1/2} \qquad (4.27)$$

(ii) Definition of System Gain Ratio

Consider a system matrix $A(s)$ relating input signals $U(s)$ to output signals $Y(s)$ so that $Y(s) = A(s)U(s)$. Assume that the sizes of the signals are represented by their norms $\|U(s)\|_{U(s)}$ and $\|Y(s)\|_{Y(s)}$. The gain ratio $gr(s)$ of the system as a function of complex frequency s would then be:

$$gr(s) = \|A(s)U(s)\|_{Y(s)} / \|U(s)\|_{U(s)} \qquad (4.28)$$

(iii) Definition of Matrix Maximum and Minimum Gain

The maximum gain of **A(s)** at a particular complex frequency s, denoted by $\sigma_{max}(\mathbf{A(s)};U(s),Y(s))$, is the subordinant matrix norm

$$\sigma_{max}(\mathbf{A(s)};U(s),Y(s)) = \|\mathbf{A(s)}\|_{U(s),Y(s)}$$

$$= \max_{\mathbf{U(s)} \neq 0} \frac{\|\mathbf{A(s)U(s)}\|_{Y(s)}}{\|\mathbf{Y(s)}\|_{U(s)}} \qquad (4.29)$$

Similarly, the minimum gain of **A(s)** is defined as

$$\sigma_{min}(\mathbf{A(s)};U(s),Y(s)) = \min_{\mathbf{U(s)} \neq 0} \frac{\|\mathbf{A(s)U(s)}\|_{Y(s)}}{\|\mathbf{Y(s)}\|_{U(s)}} \qquad (4.30)$$

(iv) Definition of Matrix Singular Values

The singular values of a matrix **A(s)**: $\mathbf{C^n_{Q1}} \rightarrow \mathbf{C^m_{Q2}}$, at some complex frequency s, are denoted as $\sigma_j(\mathbf{A(s)};Q_1,Q_2)$, (j = 1..n) and defined in (4.31).

$$\sigma_j(\mathbf{A(s)};Q_1,Q_2) = \sqrt{(\ eig(\mathbf{A(s)}^*\mathbf{A(s)})\ )} \qquad (4.31)$$

where

$$\mathbf{A}(s)^{*} = \mathbf{Q}_1^{-1}\mathbf{A}(s)^{T}\mathbf{Q}_2 \qquad (4.32)$$

(v) Relationship between Maximum Gains and Singular Values

An important mathematical result is that the maximum gain of a system matrix **A(s)**, at some complex frequency s, in fact equals the maximum singular value of that matrix at that frequency (Safonov, 1981). The same relationship holds for the minimum gain and the minimum singular value. These results are summarized by equations (4.33) and (4.34).

$$\sigma_{max}(\mathbf{A}(s);\mathbf{Q}_1,\mathbf{Q}_2) = \max_j\{\sigma_j(\mathbf{A}(s);\mathbf{Q}_1,\mathbf{Q}_2)\} \qquad (4.33)$$

$$\sigma_{min}(\mathbf{A}(s);\mathbf{Q}_1,\mathbf{Q}_2) = \min_j\{\sigma_j(\mathbf{A}(s);\mathbf{Q}_1,\mathbf{Q}_2)\} \qquad (4.34)$$

The results shown in (4.33) and (4.34) are very significant as they imply that a meaningful measure of the "size" of a system matrix at some complex frequency s can be obtained by observing the maximum and minimum singular values of the matrix at that frequency. These values will indicate the maximum and minimum gain that will exist between the inputs and outputs of the system matrix (at some s).

4.3.3) Assigning Quantitative Values to Performance Indices Gauging Feedback Properties

The maximum and minimum singular values (as a function of complex frequency s) of the characteristic matrices $\mathbf{W(s)}$, $\mathbf{T_i(s)}$ and $\mathbf{S_i(s)}$, as well as those for transmission matrices $\mathbf{T_{YR}(s)}$ and $\mathbf{T_{UR}(s)}$, are used in performance indices which quantitatively gauge the feedback properties of a MIMO control system.

(i) Response of Inputs and Outputs to Process Disturbances and Noise in the Sensors

The frequency dependent performance index, $\|\mathbf{T_{YD}(s)}\|_{max}$ gauges the maximum response of the output $\mathbf{Y(s)}$ to disturbances $\mathbf{D(s)}$, and is defined as:

$$\begin{aligned} \|\mathbf{T_{YD}(s)}\|_{max} &= \sigma_{max}(\mathbf{S_2(s)};\mathbf{Q_1},\mathbf{Q_2}) \\ &= \max_j\{\sigma_j(\mathbf{S_2(s)};\mathbf{Q_1},\mathbf{Q_2})\} \end{aligned} \qquad (4.35)$$

Similarly, to gauge the maximum response of the input to disturbances, the performance index $\|\mathbf{T_{UD}(s)}\|_{max}$ is defined as:

$$\begin{aligned} \|\mathbf{T_{UD}(s)}\|_{max} &= \sigma_{max}(\mathbf{W(s)};\mathbf{Q_1},\mathbf{Q_2}) \\ &= \max_j\{\sigma_j(\mathbf{W(s)};\mathbf{Q_1},\mathbf{Q_2})\} \end{aligned} \qquad (4.36)$$

The frequency dependent performance index, $\|T_{YN}(s)\|_{max}$ gauges the maximum response of the output $Y(s)$ to noise on the sensors $N(s)$, and is defined as:

$$\begin{aligned} \|T_{YN}(s)\|_{max} &= \sigma_{max}(T_2(s);Q_1,Q_2) \\ &= \max_j\{\sigma_j(T_2(s);Q_1,Q_2)\} \end{aligned} \qquad (4.37)$$

Similarly, to gauge the maximum response of the input to noise on sensors, the performance index $\|T_{UN}(s)\|_{max}$ is defined as:

$$\begin{aligned} \|T_{UN}(s)\|_{max} &= \sigma_{max}(W(s);Q_1,Q_2) \\ &= \max_j\{\sigma_j(W(s);Q_1,Q_2)\} \end{aligned} \qquad (4.38)$$

(ii) Sensitivity of Inputs and Outputs to Changes in Process Model

The frequency dependent performance index, $\|SE_Y(s)\|_{max}$ defined in (4.39) gauges the maximum change in setpoint to output transmission matrix $T_{YR}(s)$ (and thus maximum change in outputs $Y(s)$) which results from changes in the process model $G(s)$ from its nominal value $G_0(s)$.

$$\begin{aligned} \|SE_Y(s)\|_{max} &= \sigma_{max}(S_2(s);Q_1,Q_2) \\ &= \max_j\{\sigma_j(S_2(s);Q_1,Q_2)\} \end{aligned} \qquad (4.39)$$

Similarly, to gauge the maximum change in setpoint to output transmission matrix $T_{YR}(s)$ which results from changes in the process model, the performance index $\|SE_U(s)\|_{max}$ is defined as:

$$\begin{aligned} \|SE_U(s)\|_{max} &= \sigma_{max}(T_1(s);Q_1,Q_2) \\ &= \max_j\{\sigma_j(T_1(s);Q_1,Q_2)\} \end{aligned} \qquad (4.40)$$

(iii) Stability

Manipulation in Safonov's (Safonov, 1981) paper of equation (4.21), on which the stability analysis of a MIMO system is based yields the following stability results (defined at the control input channels - could have defined at output channels using matrix $T_2(s)$):

The system has a gain margin in each of the plant's control input channels of at least gm where:

$$\begin{aligned} gm &= \inf_w (20\log_{10}(1 + \sigma_{min}(I + L_1^{-1}(jw)))) \text{ dB} \\ &= \inf_w (20\log_{10}(1 + \sigma_{min}(T_1^{-1}(jw)))) \text{ dB} \end{aligned} \qquad (4.41)$$

The system has a phase margin in each control channel of at least φ_M where:

$$\varphi_M = 2 \inf_w \arcsin(\tfrac{1}{2} \sigma_{min}(I + L_1^{-1}(jw)))$$

$$= 2 \inf_w \arcsin(\tfrac{1}{2} \sigma_{min}(T_1^{-1}(jw))) \qquad (4.42)$$

The margins defined hold even when the gain or phase variations occur simultaneously in several or all input channels.

(iv) Setpoint Tracking and Control Effort

The setpoint tracking of a MIMO system is characterized by the "closeness" of the transmission matrix $T_{YR}(s)$ to the identity matrix. The "size" of the matrix ERR(s) defined in (4.43) thus gives a measure of "how far" the matrix $T_{YR}(s)$ is from the ideal case of the identity matrix.

$$ERR(s) = (I - T_{YR}(s)) \qquad (4.43)$$

The frequency dependent performance index, $\|ERR(s)\|_{max}$ gauges the quality of the tracking of the setpoint R(s) by the output Y(s), and is defined in (4.44).

$$\|ERR(s)\|_{max} = \sigma_{max}((I - S_2(s)G(s)K(s)P(s));Q_1,Q_2) = \max_j\{\sigma_j((I - S_2(s)G(s)K(s)P(s)) ;Q_1,Q_2)\} \quad (4.44)$$

Similarly, to gauge the maximum response of the input to perturbations in the setpoint, the performance index $\|T_{UR}(s)\|_{max}$ is defined as:

$$\|T_{UR}(s)\|_{max} = \sigma_{max}(S_1(s)K(s)P(s);Q_1,Q_2) = \max_j\{\sigma_j(S_1(s)K(s)P(s);Q_1,Q_2)\} \quad (4.45)$$

Plots of the performance indices defined in (i), (ii) and (iv) as functions of frequency characterize the performance of the system in terms of their respective feedback properties. The ideal case for each plot would be zero magnitude at all frequencies. Obviously this case is impossible to realize for all of the plots simultaneously, so the designed performance in terms of the various feedback properties will have to traded off over different frequency bands.

The stability feedback property is quantitatively gauged by the gain and phase margin expressions defined in equations (4.41) and (4.42).

Singular value decomposition of the characteristic matrices has thus provided a quantitative means of expressing the quality of all of the considered feedback properties of a MIMO control system.

4.4) Modern Control System Synthesis Techniques

Two important control system synthesis techniques which have developed in the past two decades are H^{∞} (Craig, 1989) and Linear Quadratic Gaussian (LQG) (Johnson & Grimble, 1978) techniques.

4.4.1) Brief Discussion of H^{∞} and LQG Techniques

(i) H^{∞} Techniques

Design specifications (robustness and performance) for H^{∞} control systems are expressed in terms of constraints on weighting functions. The process model is then augmented with these weighting functions.

H^{∞} synthesis problem is then solved which effectively "inverts" stable plant dynamics, and substitutes in their place the desirable dynamics prescribed by the weighting functions.

Finally it is verified if the design specifications are met and if control rate and magnitude constraints are adhered to. This is achieved by observing singular value plots of the sensitivity **S(s)** and complementary sensitivity **C(s)** matrices as functions of frequency. These matrices are essentially the characteristic matrices $S_2(s)$ and $T_2(s)$ respectively, as defined in section 4.1 for the case where **F(s) = I**.

Process repeated if control system is not satisfactory.

Many research papers have been produced on the synthesis of H^∞ control systems with desired feedback properties. (Zames & Bruce, 1983), (Zames, 1981).

(ii) Linear Quadratic Gaussian Techniques

The LQG technique synthesizes an optimal controller and state estimator for a system in state space format.

The design objectives are specified by assigning limits to excursions in the process inputs and states, as well as specifying the expected sensor noise and process disturbances. These limits are expressed by input and state weighting matrices and noise covariance matrices.

A cost function J_C is minimized trading off the desired limiting of large excursions in the inputs against those

in the states. This involves the solution of the Riccati equation to synthesize a full state feedback control law.

Another cost function J_f is minimized penalizing large errors in the estimation of the process states by a Kalman state estimator. This also involves the solution of the Riccati equation to synthesize a gain matrix for the Kalman observer.

The choice of the weighting and covariance matrices has been the topic of many theoretical papers. (Johnson & Grimble, 1987), (Broussard, 1982), (Safonov, 1981), (Gupta, 1980). Discussion on the analysis of LQG system feedback properties is also introduced in some papers. (Safonov, 1981), (Doyle & Stein, 1981).

4.4.2) Reasons for the Choice of LQG Technique for Performance Index Analysis

The LQG technique has been chosen as a controller synthesis technique (to synthesize control systems to be analysed using performance indices) for the reasons outlined in this subsection.

Research is being done at the University of Cape Town (UCT) Electrical Engineering department into modern state space and frequency domain design techniques for MIMO control systems.

Postgraduate theses have been produced which investigate a number of MIMO control system design techniques. Dissertations discussing Decentralized Control (Gear, 1988), Characteristic Loci (Fisher, 1988), Inverse Nyquist Array and State Space Pole Placement (Venzke, 1988), Variable State Space (Ginsberg, 1989) have been produced.

This thesis explores the use of performance indices in the analysis of control systems. These systems could be designed using any of the aforementioned MIMO technique, but it was decided to choose a synthesis technique to automate the design procedure as much as possible.

The analysis (or verification) of control systems synthesized using H^{∞} techniques forms an integral part of the H^{∞} design procedure (singular value plots of sensitivity and complementary sensitivity matrices). The analysis of H^{∞} systems is therefore a widely discussed topic. Detailed analysis of LQG systems is not as well explored. The topic is introduced in the papers by Safonov et al (Safonov, 1981) and Doyle & Stein (Doyle & Stein, 1981). For this reason, the LQG synthesis technique has been chosen for the application of performance index analysis.

4.5) LQG Control System Synthesis Techniques

A more detailed look at LQG techniques is taken in this section. Figure 4.2 shows a block diagram of a state feedback control system containing a Kalman filter state estimator.

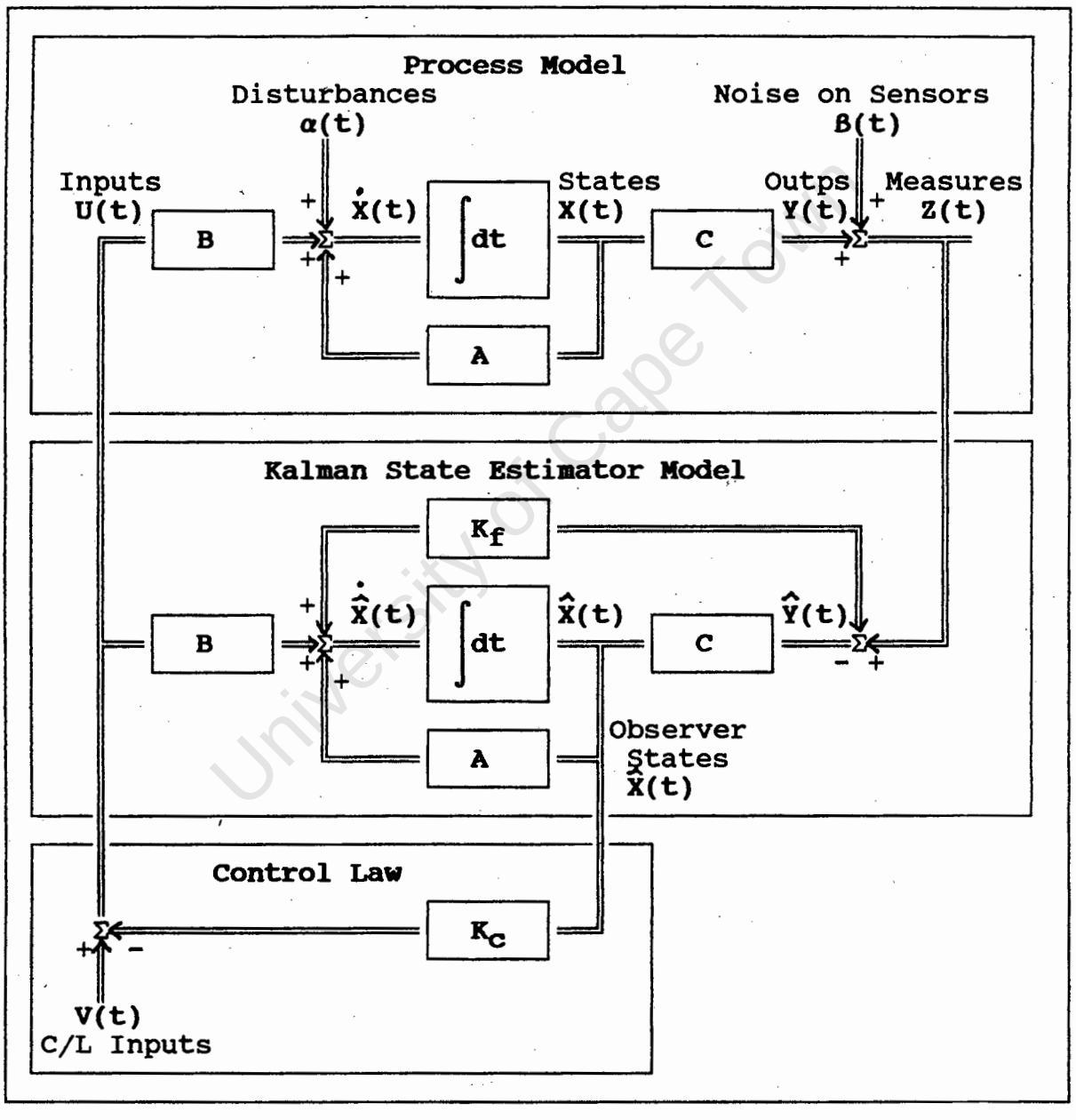


Figure 4.2: Block Diagram of State Feedback System including Kalman Filter State Estimator

4.5.1) State Space Description of Process Model

LQG optimal control techniques attempt to minimize the performance index J_C for the process, as given in (4.46).

$$J_C = \lim_{T\to\infty} E\{ {}^{1}/_{2T} \int_{-T}^{T} \{X^T(t)Q_1X(t) + U^T(t)R_1U(t)\}\, dt \quad (4.46)$$

where:

- $X(t)$: Vector of time dependent process states
- $U(t)$: Vector of time dependent process inputs
- Q_1 : Matrix penalizing large deviations in process states
- R_1 : Matrix penalizing large deviations in process inputs

Figure 4.2 shows that equation (4.46) is subject to the state-space system description:

$$d/dt(X(t)) = AX(t) + BU(t) \quad (4.47)$$

$$Y(t) = CX(t) \quad (4.48)$$

with measurement equation:

$$Z(t) = Y(t) + \beta(t)$$

$$= CX(t) + \beta(t) \quad (4.49)$$

where:

$\alpha(t)$: Plant disturbances

$Z(t)$: Vector of time dependent measurements of process outputs

$Y(t)$: Vector of time dependent process outputs

$\beta(t)$: Measurement noise

D : Considered to be identity matrix, I

4.5.2) Kalman Filter

The Kalman filter shown in figure 4.2 is now considered. The performance index J_f for the filter, as given in (4.50), is to be minimized in this case.

$$J_f = E\{[X(t) - \hat{X}(t)]^T[X(t) - \hat{X}(t)]\} \qquad (4.50)$$

where:

$X(t)$: Vector of time dependent process states

$\hat{X}(t)$: Vector of time dependent estimates of process states

4.5.3) LQG Problem Assumptions

i) Define $\theta(t) = (\alpha(t), \beta^T(t))^T$; where $\theta(t)$ is a zero-mean white Gaussian noise process with covariance:

$$\mathrm{Cov}[\theta(t),\theta(t)] = \begin{bmatrix} Q & 0 \\ 0 & R \end{bmatrix} \delta(t-T) \qquad (4.51)$$

ii) The matrices characterizing the noise covariances satisfy:

$$Q = Q^T Q \qquad (4.52)$$

$$R = R^T \qquad (4.53)$$

iii) Pair (A,QD) is completely controllable. Pair (A,C) is completely observable.

iv) The matrices characterizing the control properties satisfy:

$$Q_1 = {Q_1}^T Q_1 \qquad (4.54)$$

$$R_1 = {R_1}^T \qquad (4.55)$$

iii) Pair (A,B) is completely controllable. Pair (A,Q_1C) is completely observable.

4.5.4) LQG Problem Solution

(i) Controller

The optimal control law is given by:

$$U(t) = -K_C\hat{X}(t) \qquad (4.56)$$

where:

K_C = Control gain

$$= R^{-1}B^TP_\infty \qquad (4.57)$$

and P_∞ solves the control algebraic Riccati equation:

$$A^TP_\infty + P_\infty A - P_\infty BR_1^{-1}B^TP_\infty + C^TQ_1C = 0 \qquad (4.58)$$

and:

$$P^T_\infty = P_\infty > 0 \qquad (4.59)$$

(ii) Kalman Filter

The estimate of the state is that produced by the Kalman Filter equations:

$$d/dt(\hat{X}(t)) = A\hat{X}(t) + BU(t) + K_f(Z(t) - C\hat{X}(t)) \qquad (4.60)$$

where:

K_f = Kalman gain

$$= S_\infty C^T R^{-1} \qquad (4.61)$$

and S_∞ solves the filter algebraic Riccati equation:

$$AS_\infty + S_\infty A^T - S_\infty C^T R^{-1} C S_\infty + DQD^T = 0 \qquad (4.62)$$

and:

$$S^T_\infty = S_\infty > 0 \qquad (4.63)$$

4.5.5) LQG Controller Synthesis Procedure

The procedure adopted to obtain a controller and filter using LQG techniques can be summarized as follows:

i) Choose suitable control property weighting matrices Q_1 and R_1 for process.

ii) Solve the control algebraic Riccati equation to obtain a control gain matrix K_C for the process.

iii) Specify suitable noise covariance matrices Q and R for Kalman filter.

iv) Solve the filter algebraic Riccati equation to obtain suitable Kalman gain matrix K_f for the observer.

v) Implement K_c and K_f on the process as shown in figure 4.2.

4.5.6) Comment on Stability of LQG Control Systems

The separation theorem for optimal control systems with state observers (Kwakernaak & Sivan, 1972 pp 378..382) shows that:

i) The stability of the cascaded filter and controlled system is determined by the eigenvalues of the system matrices:

$$A - BK_c$$

$$A - K_fC$$

ii) The separation theorem implies that the dynamics of the control and filter systems can be treated separately. This implies that the stable design of a controller and filter can be done separately and still yield a stable overall system.

More recent papers on this subject dispute the claim that the overall system will be stable if stable designs for the K_c and K_f are done separately. (Doyle, 1978) showed that no stability margins can be guaranteed for a LQG system since

perturbations in the feedback loop cause the filter model to no longer mirror the process model. A further paper by Doyle & Stein (Doyle & Stein, 1978) addresses this problem by adjusting the noise covariance matrices **Q** and **R**. The paper shows that "the commonly suggested approach of making all roots of the error dynamics arbitrarily faster is generally the wrong thing to do".

The good robustness properties of LQ regulators (LQG systems without state observer) are recovered by an observer-adjustment procedure. This procedure redefines **Q** as shown in (4.64) while the definition for **R** remains unchanged.

$$\mathbf{Q} = \mathbf{Q_0} + q^2\mathbf{BVB}^T \qquad (4.64)$$

where:

V : Any Positive Definite Matrix

As $q \rightarrow \infty$, the robustness properties of a LQG system become identical to those of a LQ system. LQ systems have an infinite gain margin and a phase margin of at least 60°.

The improvement in robustness properties due to this "robustness recovery" technique occurs at the expense of the good dynamic properties of the optimal filter (such as fast error dynamics).

4.6) Performance Index Analysis Applied to Control Systems Synthesized using LQG Techniques

The frequency-domain performance index analysis of LQG control systems is dealt with in this section.

Comparison of figures 4.1 and 4.2 shows that the block diagram of a generalized control system differs from that of a LQG control system. The block diagram in figure 4.2 (LQG system) will thus have to be manipulated into the form of figure 4.1 (generalized system) to facilitate generalized performance index analysis of the LQG system. Expressions will be obtained from this transformation for the generalized system matrices **P(s)**, **K(s)**, **G(s)** and **F(s)** in terms of the state space system matrices **A**, **B**, **C**, $\mathbf{K_c}$, and $\mathbf{K_f}$.

Block diagram manipulation of figure 4.2 and algebraic manipulation of the state-space equations (4.47) to (4.60) transforms the LQG state space system into the format shown in figure 4.3. Note also that the system has been transformed to the s-domain by replacing the integrating blocks in figure 4.2 with the s-domain integrating operator 1/s (taking Laplace transforms). The matrix $\Phi(s)$ in figure 4.3 is defined in equation (4.65).

$$\Phi(s) = (sI - A + K_f C) \qquad (4.65)$$

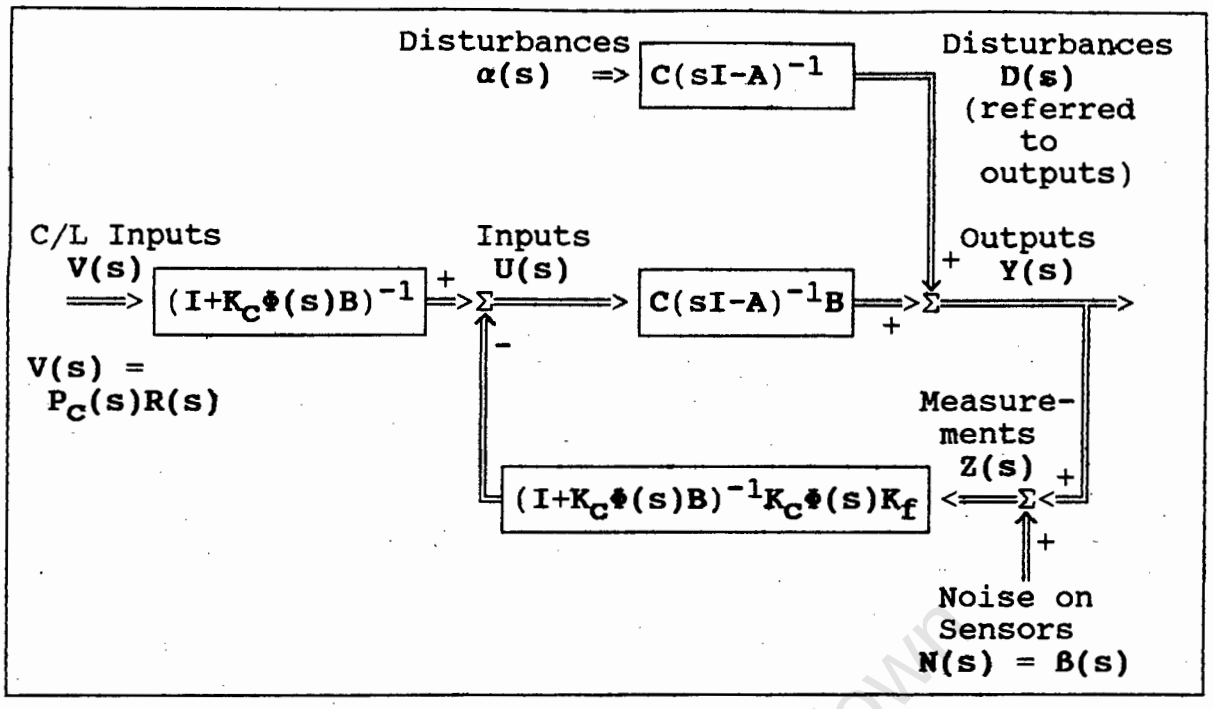


Figure 4.3: Block Diagram of the Transformed LQG State-space system in the s-domain

Note:

i) State-space systems are not designed to ensure tracking of the C/L inputs $\mathbf{V(s)} = L\{\mathbf{V(t)}\}$ by the outputs **Y(s)**. An additional cascade precompensator $\mathbf{P_C(s)}$ will have to be included with the C/L system (designed to be robust) if tracking of the setpoints **R(s)** by the outputs **Y(s)** is to be ensured.

ii) The disturbances $\boldsymbol{\alpha(s)} = L\{\boldsymbol{\alpha(t)}\}$ on the differential of the states is referred to the outputs by

multiplying the vector $\alpha(s)$ by the (non-square) matrix $C(sI-A)^{-1}$.

Comparing of figures 4.1 and 4.3, and considering note (i) above shows the following relationships existing between the generalized and LQG systems:

$$G(s) = C(sI - A)^{-1}B \tag{4.66}$$

$$K(s) = I \text{ (identity matrix)} \tag{4.67}$$

$$F(s) = (I + K_c\Phi(s)B)^{-1}K_c\Phi(s)K_f \tag{4.68}$$

$$P(s) = (I + K_c\Phi(s)B)^{-1}P_c(s) \tag{4.69}$$

The LQG system in figure 4.2 has successfully been transformed into the form of the generalized control system shown in figure 4.1. The characteristic, transmission, sensitivity and stability matrices can now be evaluated for the LQG system using the identities in (4.66) to (4.69). This will enable quantitative frequency-domain performance index analysis of the feedback properties of the system.

4.7) Concluding Remarks

The performance indices developed quantitatively represent the "goodness" of the frequency-dependent feedback properties of MIMO control systems, and are intimately related to the "sizes" of the characteristic matrices $S_i(s)$, $T_i(s)$ and $W(s)$ ($i = 1..2$). These matrices are functions of the generalized MIMO control system transfer function matrices $G(s)$, $K(s)$, $F(s)$ and $P(s)$.

The properties of noise and disturbance transmission are measured by the frequency dependent maximum singular values of the transmission matrices $T_{YN}(s) = -T_2(s)$, $T_{UN}(s) = -W(s)$, $T_{YD}(s) = S_2(s)$ and $T_{UD}(s) = -W(s)$.

The properties of input and output sensitivity to changes in the process are similarly related to the maximum singular values of matrices $SE_Y(s) = S_2(s)$ and $SE_U(s) = T_1(s)$.

The stability of the system is related to the minimum singular value of the matrix $T_1^{-1}(jw)$ over all frequencies by the relation $gm = \inf_w (20\log_{10}(1 + \sigma_{min}(T_1^{-1}(jw))))$ dB

and relation $pm = 2 \inf_w \arcsin(\frac{1}{2} \sigma_{min}(T_1^{-1}(jw)))$.

The performance in terms of the control effort and setpoint tracking is characterized by the maximum singular values of the transmission matrices $T_{YR}(s)$ and $T_{UR}(s)$, which are

functions of the characteristic matrices and transfer function matrices.

The performance indices defined for MIMO control systems reduce to the classical measures of performance defined for SISO systems, in the special 1x1 case of MIMO systems.

Performance index analysis can be applied to control systems designed using any MIMO technique. State feedback systems synthesized using LQG techniques have been chosen for the demonstration of performance index analysis techniques.

The matrices $\mathbf{G(s)}$, $\mathbf{F(s)}$ and $\mathbf{P(s)}$ in the generalized MIMO frequency-domain representation are related to the state feedback system matrices $\mathbf{A}$, $\mathbf{B}$, $\mathbf{C}$, $\mathbf{K_c}$ and $\mathbf{K_f}$ by the three relations $\mathbf{G(s) = C(sI-A)^{-1}B}$, $\mathbf{F(s) = (I+K_c\Phi(s)B)^{-1}K_c\Phi(s)K_f}$ and $\mathbf{P(s) = (I+K_c\Phi(s)B)^{-1}}$, where $\mathbf{\Phi(s) = (sI - A + K_fC)}$. For LQG systems, $\mathbf{K(s) = I}$. These relations make it possible to analyse the performance of LQG control systems using the performance indices developed for generalized frequency-domain MIMO control systems.

CHAPTER FIVE

OPTIMAL CONTROL CAD AND PROCESS CONTROL SOFTWARE DEVELOPED

5.1) Background to Development of CAD System

A computer-aided optimal control system design (CACSD) package, has been developed to:

i) Synthesize optimal LQG control systems.

ii) Enable the frequency-dependent performance index analysis of the feedback properties of state space systems with observers (of which LQG systems are a special case).

iii) Enable the digital simulation of such systems under open and closed loop conditions.

The reasons for the development of the CACSD package follow:

i) A tool for the frequency-domain analysis of the properties of state-space systems with observers is required.

ii) CACSD packages are being developed at UCT for the implementation of various control system design techniques, to gain experience in their use.

5.2) Procedure for the Synthesis and Analysis of MIMO LQG Control Systems

The synthesis and analysis procedure for LQG control systems is best described with the assistance of the flowchart of a typical design session given in figure 5.1.

5.2.1) Block 1: Start

A design session is started with the specification of a state-space model for the process. The number of process states and inputs, as well as the plant matrices **A**, **B**, and **C** are entered or loaded.

5.2.2) Block 2: Initial Design Specifications

The initial design specifications are entered or loaded in the form of the initial weighting ($\mathbf{Q}_1$, $\mathbf{R}_1$) and covariance (**Q**, **R**) matrices. The algorithm used for the solution of the Riccati equation (in the LQG synthesis) also requires an initial estimate of the control and Kalman gain matrices $\mathbf{K}_c$ and $\mathbf{K}_f$. Initial estimates for these matrices are also entered or loaded at this stage (estimates have to realize a C/L stable control system).

5.2.3) Block 3: Synthesis of Optimal Controller

An optimal state-feedback gain matrix K_c and Kalman state-observer gain matrix K_f are synthesized automatically using the procedure outlined in section 4.5.4.

5.2.4) Block 4: Transformation to Generalized Format

The state-space system representation for the process is transformed to the format for a generalized MIMO control system. The matrices **G(s)**, **K(s)**, **F(s)** and **P(s)**, as well as **W(s)**, $S_i(s)$ and $T_i(s)$ are then evaluated at frequency points over some frequency range $w_0 \leq w < w_f$ (where $s = jw$). The user specifies the range and number of frequency points.

5.2.5) Block 5: Performance Index Analysis

The performance indices gauging each feedback property are evaluated at each frequency point using values obtained for the matrices in block 4.

5.2.6) Block 6: Presentation of Performance Data

The performance index data is plotted against frequency and analysed further. The plots and analysis are presented concisely to the user on two graphics screens. The performance data presented provides a means by which the user can judge the quality of the design and decide if the

performance is satisfactory. Analysis of the design could also be assisted by doing a time simulation to observe the system responses to noise, disturbances, setpoints, etc.

5.2.7) Block 7: Changing Design to Improve Performance

If the performance of the design is unsatisfactory, the user can make changes to the weighting and covariance matrices. This is done in an attempt to improve the performance the system with a new controller synthesized using these matrices.

5.2.8) Block 8: Adopting Design for Implementation

If the performance of the design is satisfactory, the control system that was synthesized can be adopted for implementation. The analysed design can be saved (process, controller, weighting and covariance matrices, as well as performance index data) for later reference and for implementation.

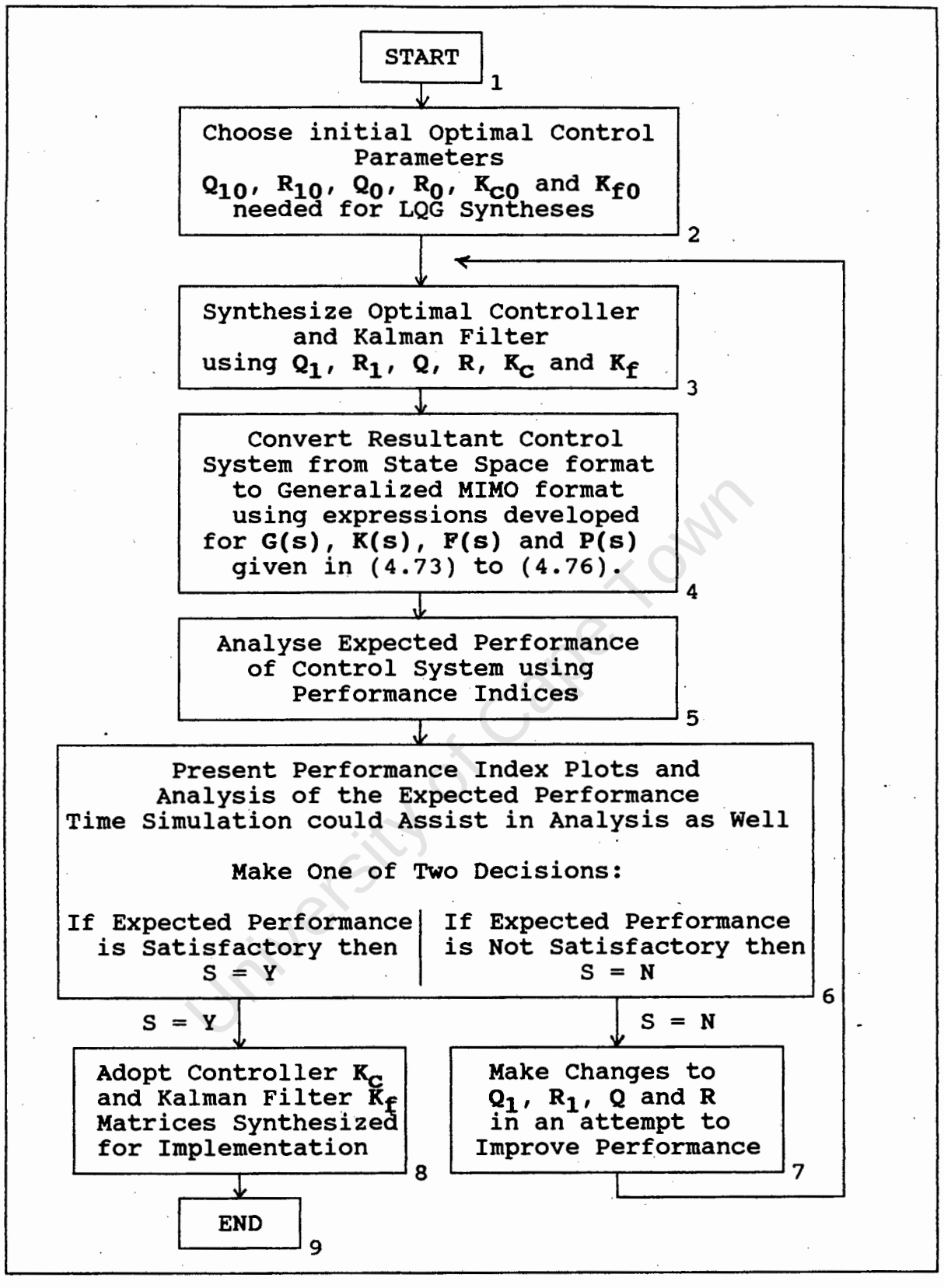


Figure 5.1: Algorithm for the design of a control system using LQG synthesis and performance index analysis techniques.

5.3) Description of CAD System

The optimal controller synthesis and analysis package (referred to as OPTCAD) is menu-driven (using menu routines developed by Fisher (Fisher, 1988)) to make it simple and quick to use and understand. The main menu of the system gives the user the choice to run any one of the four major parts of the package, which are discussed in sections 5.3.1 to 5.3.4. This discussion is followed by the specification of the hardware and software requirements for running the package in section 5.3.5. A key to the listings of the source code and software development information is then given in section 5.3.6.

5.3.1) Data Manipulation Facility

Selecting the "Data" option in the main menu (and when it occurs in menus lower down in menu structure) allows the user to perform the following operations on logical groupings of the matrices **A**, **B**, **C**, $\mathbf{K_c}$, $\mathbf{K_f}$, $\mathbf{Q_1}$, $\mathbf{R_1}$, **Q**, **R** and **Pc**:

(i) Save

Groupings of the matrices can be stored to a long term storage device (eg. to a file on disk).

(ii) Load

Groupings of the matrices can be loaded from a long term storage device.

iii) Edit

Any element of each matrix can be edited or modified by the user at any stage of the design procedure.

The "Data" option also allows groups of other constants, used in the running of the package and in the specification of the process, to be loaded, saved or modified by the user at any stage of the design procedure. Performance index data generated by the analysis procedure can be saved, or previously generated data can be loaded for viewing.

5.3.2) LQG Optimal Controller Synthesis Facility

Selecting the "Optcon" option in the main menu invokes the synthesis procedure discussed in section 4.5.4 to generate an optimal controller gain matrix $\mathbf{K_c}$ and Kalman observer gain matrix $\mathbf{K_f}$ automatically. The synthesis procedure minimizes the cost functions J_c and J_f indicated in equations (4.53) and (4.57) respectively and effectively consists of two calls to an algorithm which solves the algebraic Riccati equation. The weighting and covariance matrices $\mathbf{Q_1}$, $\mathbf{R_1}$, $\mathbf{Q}$ and $\mathbf{R}$ as well as the current (or initial) matrices $\mathbf{K_c}$ and $\mathbf{K_f}$ are used in the optimal synthesis.

5.3.3) Performance Index Analysis Facility

Selecting the "Analyse" option invokes the performance index analysis facility for the analysis of the system under consideration. The following operations can be performed by selecting this option:

(i) Create

Performance index data for plotting can be created by selecting this option. The user specifies the frequency range $w_0 \leq w < w_f$ over which the data is to be created. The state-space system is then transformed to the generalized system format before performance index data is created over the specified frequency range.

The transformation is implemented by setting s = jw, and applying the (complex) expressions for **G(s)**, **K(s)**, **F(s)** and **P(s)** in (4.66) to (4.69) at frequency points over the specified range. The characteristic matrices **W(s)**, $\mathbf{S_i(s)}$ and $\mathbf{T_i(s)}$ are then also evaluated at each frequency-point.

The performance indices gauging each feedback property of the control system are evaluated using the values obtained for the matrices in the transformation at each frequency-point. This is done by applying equations (4.42) to (4.52) (obtaining maximum singular values of

various matrices at each frequency point). The data points are stored during this analysis procedure.

(ii) Format

The performance index data calculated is grouped into four graphs, namely:

Graph 1: Performance indices $\|T_{YN}(s)\|_{max}$ and $\|T_{UN}(s)\|_{max}$ gauging the transmission of sensor noise to inputs and outputs respectively.

Graph 2: Performance indices $\|T_{YD}(s)\|_{max}$ and $\|T_{UD}(s)\|_{max}$ gauging the transmission of process disturbances to inputs and outputs respectively.

Graph 3: Performance indices $\|SE_Y(s)\|_{max}$ and $\|SE_U(s)\|_{max}$ gauging the sensitivity of inputs and outputs respectively G(s).

Graph 4: Performance indices $\|ERR(s)\|_{max}$ and $\|T_{UR}(s)\|_{max}$ respectively gauging the errors in the tracking of process setpoints by the outputs and the transmission of process setpoints to inputs.

The user can select all four graphs to be plotted on one screen, or any one of the graphs on its own. The vertical and horizontal axes of the plots can also be specified by the user.

(iii) View

Selecting this option triggers the plotting of the above performance index graphs as selected by the user on the Hercules graphics screen 1.

Graphics screen 0 contains an analysis of the performance index plots to assist the user in the interpretation of the plots. Comments from the plots on screen 1 (F3 key used to flip between screens 0 and 1) are generated which gauge the performance of the system over three frequency bands, namely:

Low Frequency	:	$w_0 \leq w < w_f/3$
Intermediate Frequency	:	$w_f/3 \leq w < 2w_f/3$
High Frequency	:	$2w_f/3 \leq w < w_f$

The comments are generated by observing the average of the performance index magnitudes over each range. The smaller each magnitude average is, the better the performance of the system over that range in terms of the particular feedback property.

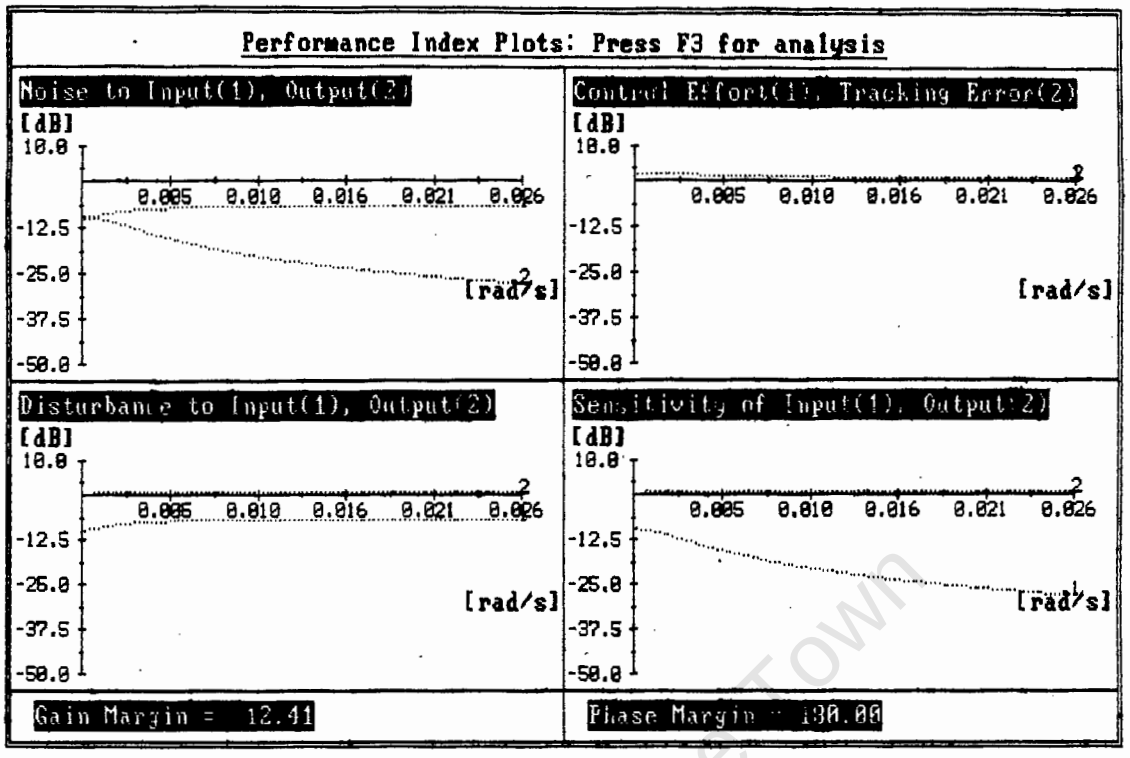


Figure 5.2: Example of a Performance Index Group Plot

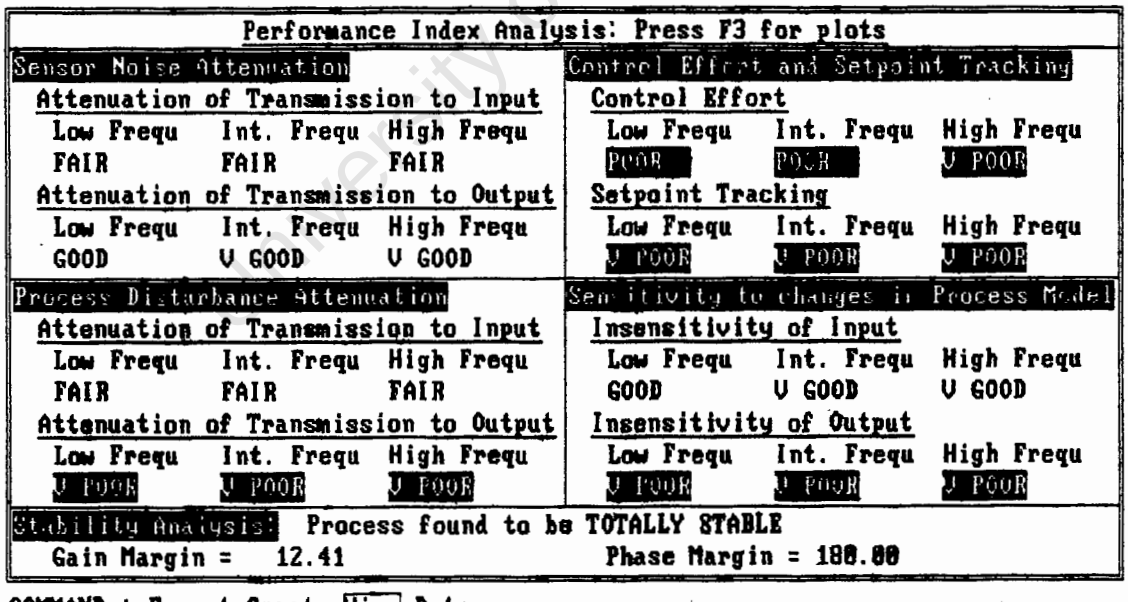


Figure 5.3: Example of Comments Generated on Performance Index Group Plot

The comments are classified according to the average magnitudes (termed M_{av}) as follows:

		M_{av}	>	1.0	: Very Poor
1.0	≥	M_{av}	>	0.5	: Poor
0.5	≥	M_{av}	>	0.25	: Fair
0.25	≥	M_{av}	>	0.1	: Good
0.1	≥	M_{av}			: Very Good

Figure 5.2 gives an example of a performance index group plot, and figure 5.3 an example of the comments generated from these plots. The calculated gain and phase margin for the system are also indicated on both screens.

5.3.4) State-space System Time Simulation Facility

Selecting the "Synthesize" option provides the user with a time simulation facility. A 4th order Runga-Kutta algorithm is used to perform the time simulation, meaning that the time step for the simulation is critical when considering accuracy.

In the time simulation, the system can be simulated either under O/L or C/L conditions, the controller and observer included or excluded in the loop. Sensor noise, disturbances and setpoint signals can be injected in the loop. These signals can be either sinusoidal (amplitude and frequency

chosen by user) or Gaussian (standard deviation chosen by user). The simulated response traces of the inputs, outputs and states (process and/or observer) can be plotted at the user's specification.

5.3.5) Hardware and Software Requirements for Running the CAD Package

OPTCAD can be run on any 8088/286/386 processor XT/AT/386 compatible microcomputer. An 8087/287/387 mathematics coprocessor is recommended as the performance index analysis routine requires a great deal of mathematical processing and would be very slow otherwise (calculating 100 frequency points on 10MHz 8088 machine without coprocessor takes approximately an hour while the equivalent time taken on a 25MHz 80386 machine with coprocessor is approximately 3 minutes). The machine must support Hercules graphics and contain at least one external disk drive. Concurrent running of the memory resident INT10, HGC and HARDCOPY graphics support software programs is required when running this program.

5.3.6) Appendix Listings for the OPTCAD Package

Appendix M contains all the specific information concerning the OPTCAD system in the sections listed below:

M1: OPTCAD System Subroutines

M2: OPTCAD Data Manipulation Subroutines

M3: Mathematics Utility Subroutines

M4: Graphics Utility Subroutines

M5: Modified Subroutines Originally Written by Ian Fisher

M6: OPTCAD Menu Definition Routine

M7: OPTCAD Include Files

M8: OPTCAD Development/Execution Information

Additional routines written by Fisher (Fisher, 1988) have been linked into the libraries used in the development of this package, and source code listings of these routines can be found in his M.Sc. dissertation.

5.4) Software Developed for the Running, Controlling and Analysis of the Heat Exchanger

The heat exchanger rig is run entirely by a digital computer, which is programmed to manipulate the inputs and read the outputs of the process. A software package called HERIG has been developed specifically for the running and analysis of

the rig. This software also serves as an interface between the user and the process. The package is once again menu driven using the same menu and graphics routines as those used in OPTCAD. The three major options presented to the user are discussed in sections 5.4.1 to 5.4.3, section 5.4.4 discusses the hardware and software requirements, while section 5.4.5 refers to appendix listings of development information.

5.4.1) Running, Controlling and User Interfacing

Invoking the "Run" option in the main menu of HERIG initializes routines which manipulate the plant inputs manually, as well as doing so automatically by applying some control law.

The control of the rig is implemented in two stages, namely:

i) The stabilizing control of the process flow and level outputs using the control valve inputs.

ii) The LQG control of temperature outputs T1 and T5 using the flow inputs (setpoints to stabilizing control algorithm) F1 and F2.

It is possible to step the plant inputs when in manual control, as well as the setpoints when running under stabilizing or LQG control. It is also possible to inject

sensor noise, disturbance and setpoint perturbations in all of the respective channels while running under LQG control.

HERIG has 8 basic modes for running the process under manual and automatic stabilizing control, as listed in table 6.1.

Mode	Name of Mode	Loop	Description of Mode
1	Manual Input Select	O/L	Select input to change or choose new mode
2	Manual Input Vary	O/L	Vary selected input or choose new mode
3	Manual Input Step Select	O/L	Select input to step or choose choose new mode
4	Manual Input Step Vary	O/L	Vary step size or perform step test
5	Auto Setpoint Select	C/L	Select setp. to change or choose new mode
6	Auto Setpoint Vary	C/L	Vary selected setp. or choose new mode
7	Auto Setpoint Step Select	C/L	Select setp. to step or choose new mode
8	Auto Setpoint Vary	C/L	Vary step size or perform step test
Q	Quit		Exit from rig running algorithm back to calling routine

Table 5.1: Modes of Operation of Rig Running Algorithm

If the rig is to be run under LQG control, it is first initialized to run under the stabilizing control mode 5. The user indicates via the keyboard when LQG control of the process is to commence. The LQG control law is then strapped

onto the stabilizing control algorithm to change the flow setpoints for the control of temperature outputs T1 and T5.

Figure 5.4 shows a flowchart of the rig running algorithm generalized for the process running under manual, stabilizing or LQG modes of control.

The interface between the user and the program is simple and logical to enable easy manipulation of the 10 plant inputs and 10 setpoints, as well as reading of the 16 plant outputs.

The two highlighted blocks 7 and 8 on the diagram indicate the sections of the rig running algorithm where direct interaction occurs with the user.

Block 8: The user can perform required functions on the rig while it is running. This is enabled by the program continually scanning the keyboard for user instructions.

Block 9: Changes in the state of the rig (changes in inputs, outputs, setp., mode, etc) are reported to the user by continually updating and making necessary changes to the interface screen.

Figures 5.5 and 5.6 show interface screens obtained with the rig running respectively under manual mode 2, and under stabilizing control mode 7 with LQG controller included.

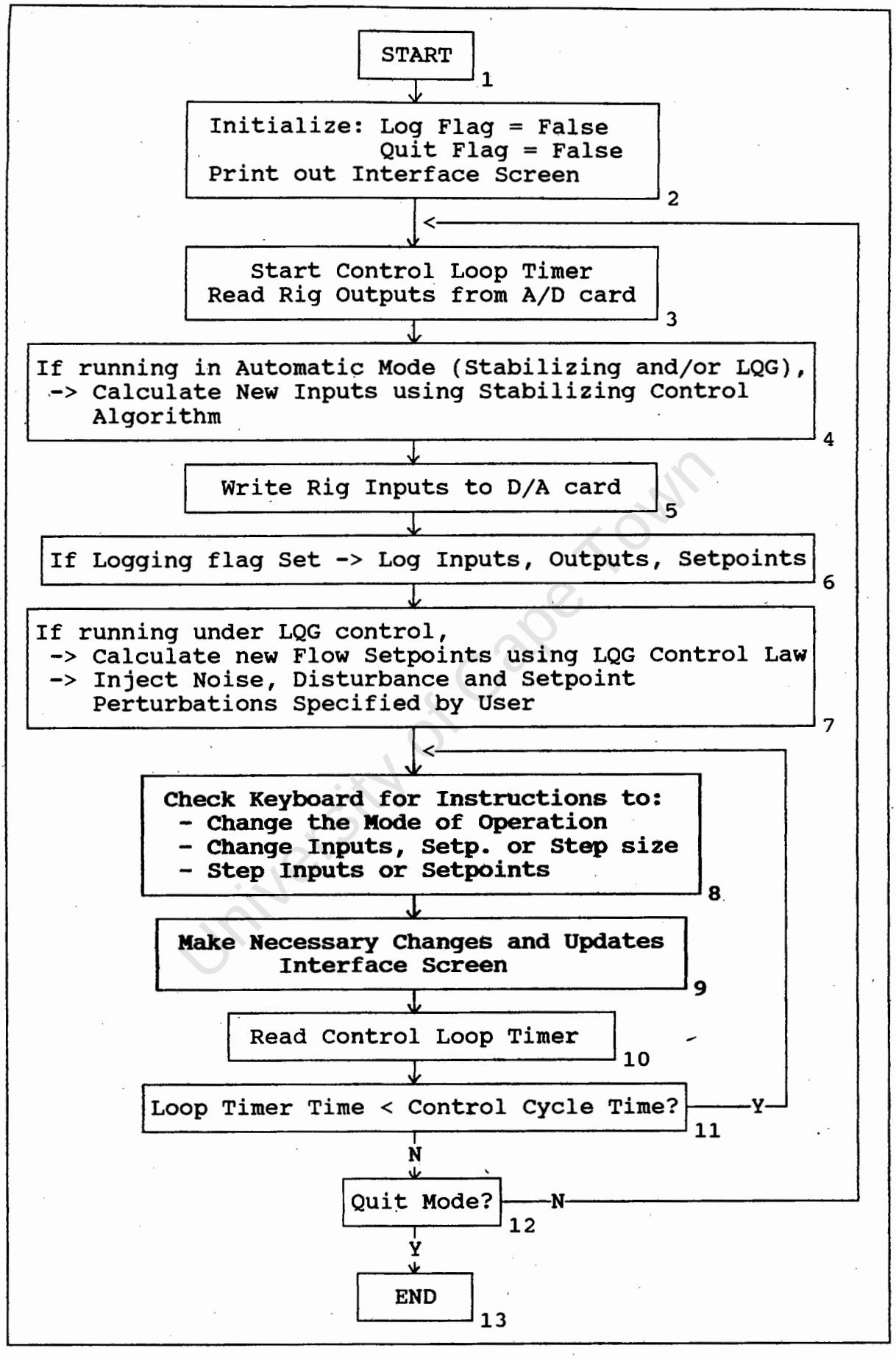


Figure 5.4: Flowchart of Rig Running and Control Algorithm

The reactor tanks (T1 to T4), the reservoirs (PHR, CHR, CCR) as well as the two counter-current flows (F1 and F2) are represented on the interface screens.

The values of the process inputs (Valves C1 to C6, Stirrers S1 to S4), outputs (Levels L1 to L4, Temperatures T1 to T10, Flows F1 to F2) and setpoints (L1 to L4, T1, T3, T5, T8 for automatic modes only) are also shown on the interface screens. These values are indicated on appropriate positions on the screens, giving the user a meaningful representation of what is occurring on the rig itself.

The options available to the user are given in the bottom section of the screens (change of mode, changing values, etc). These can be invoked by the user depressing the keys indicated.

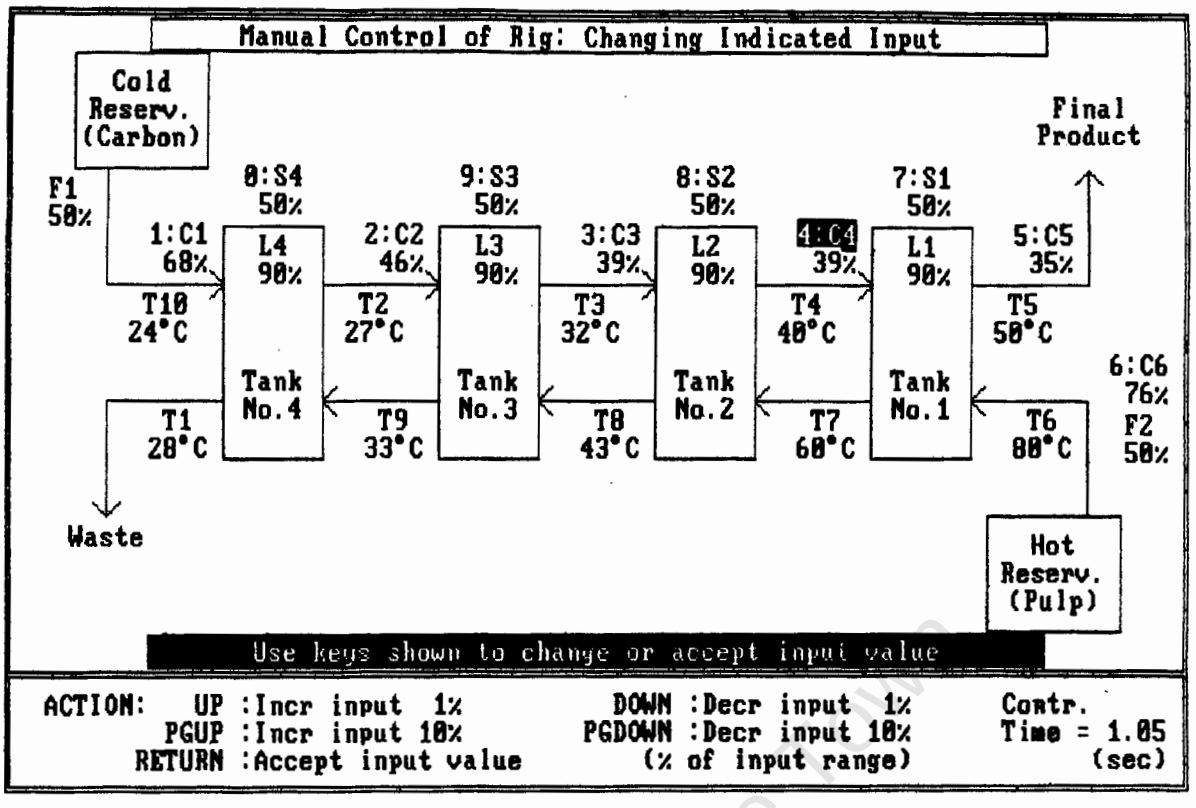


Figure 5.5: Interface Screen for Rig Running under Mode 2

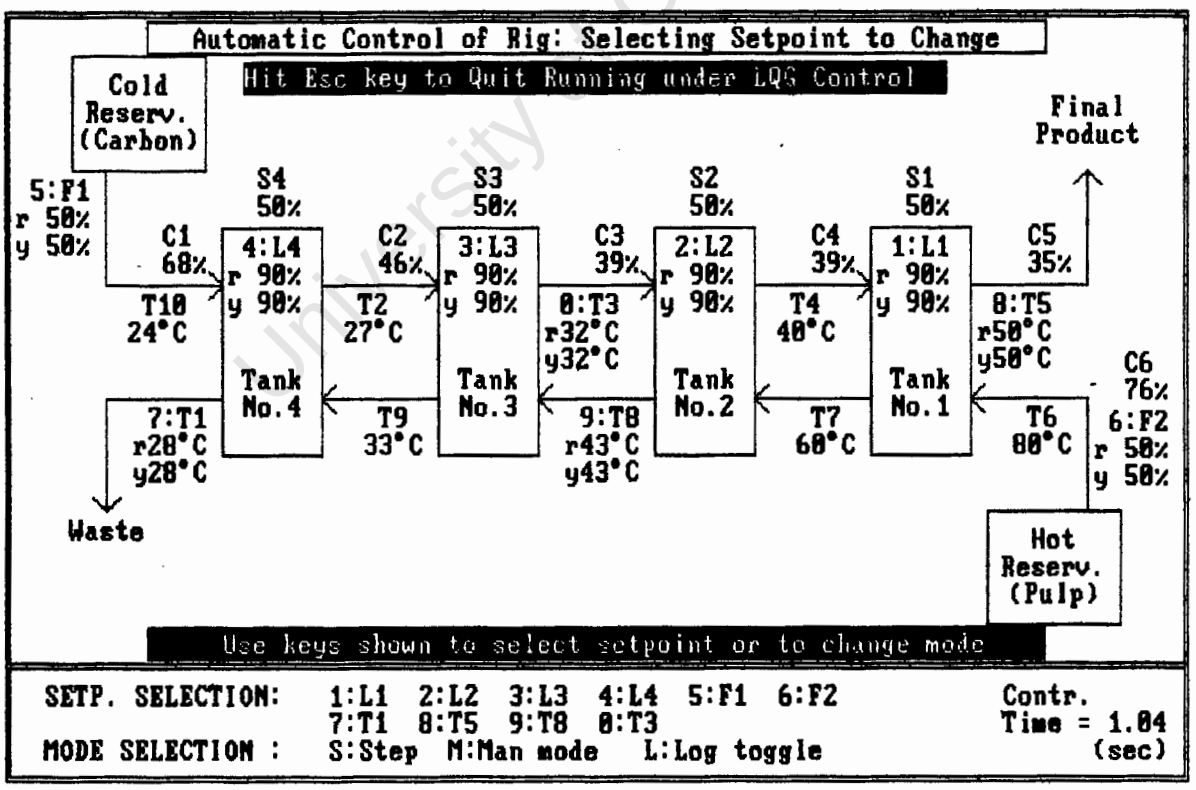


Figure 5.6: Interface Screen for Rig Running under Mode 7 With LQG Controller Strapped

5.4.2) Manipulation of Rig Running and Control Parameters

Selecting the "Param" option in the main menu allows the user to perform the following functions:

(i) Saving Parameters

The current process inputs and setpoints as well as the logged data can be stored to a long term storage device for later analysis. It is also possible to save rig running parameters such as the sample time, control time and other default settings. The stabilizing PI controller and the LQG controller and observer matrices (**A**, **B**, **C**, $\mathbf{K_c}$ and $\mathbf{K_f}$) can also be saved under this option. The data file format of the LQG controller is identical to the format used in the OPTCAD design package, ensuring easy implementation of controllers designed using this package.

(ii) Loading Parameters

Previously saved process inputs and setpoints, as well as response data, can be loaded from a long term storage device for analysis. Rig running parameters such as the sample time, control time and other default settings can also be loaded, as can the stabilizing PI controller matrix and the LQG controller and observer matrices.

(iii) Editing Parameters

It is useful to be able to modify certain parameters while the program is running instead of being limited merely to loading and saving these parameters. The parameter modifying option makes it possible to enter or edit on the screen values for the control and logging intervals, default settings and LQG controller matrices.

5.4.3) Analysis Plots of Heat Exchanger Response Data

When the "Heplot" option in the main menu is selected, the user has the choice of either:

i) Plotting any one of four sets of four graphs on the screen on predefined axes for quick reference. An example of such a plot is given in figure 5.7. The response data not shown in this plot is grouped in the other three remaining sets of graphs.

ii) Selecting any input, output or setpoint for plotting on user-defined axes for detailed analysis. An example of a plot in which the response traces and graph axes were chosen by the user is given in figure 5.8.

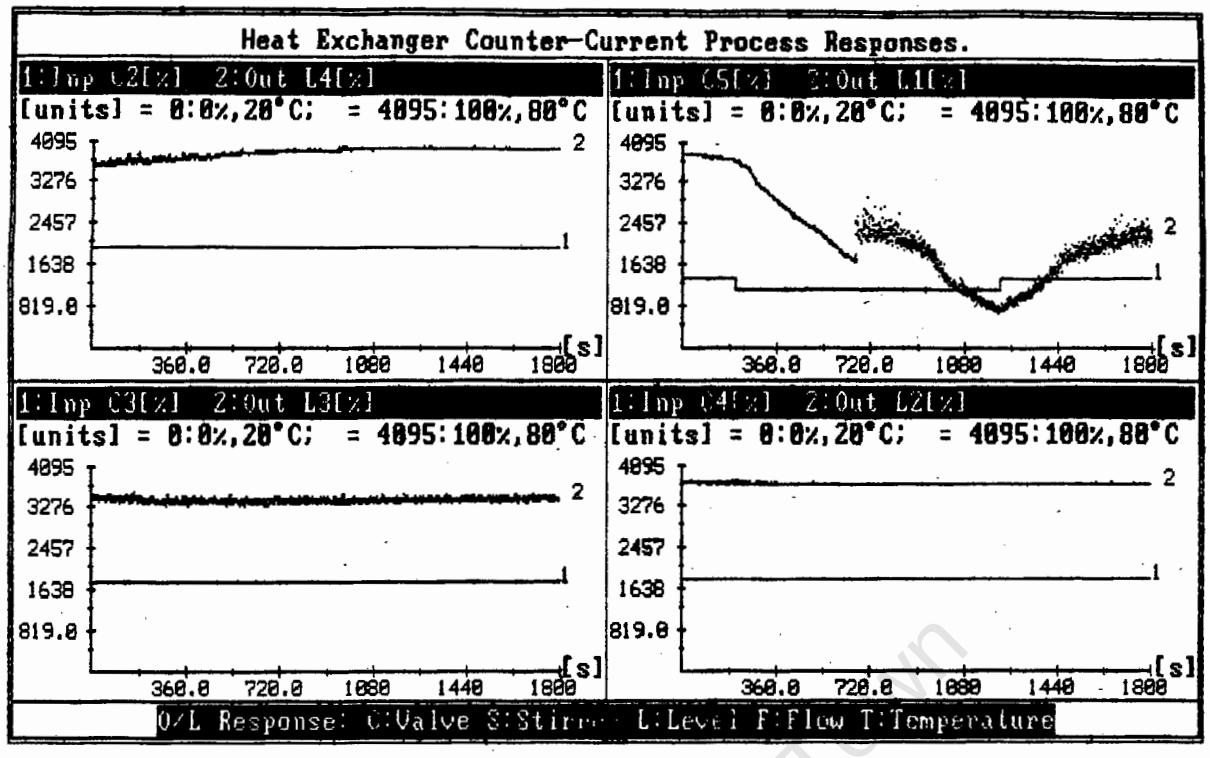


Figure 5.7: Set of Four Graphs of Response Data

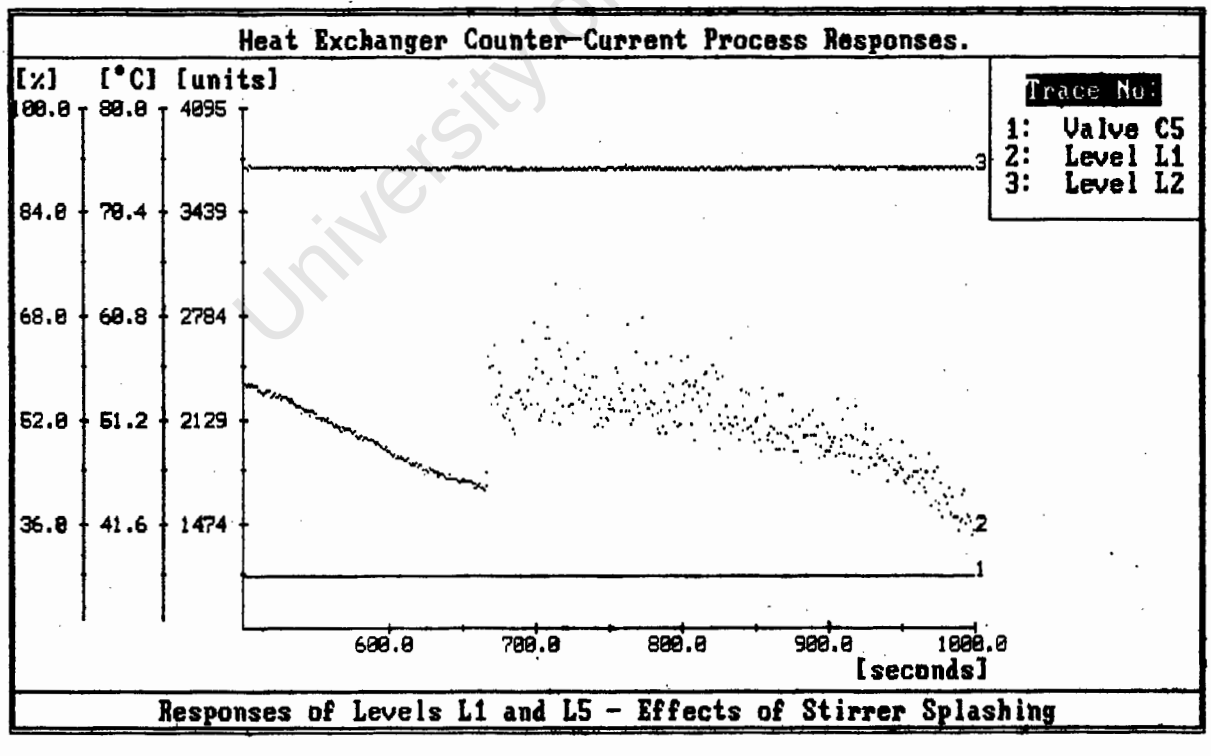


Figure 5.8: User Selected Graph of Response Data

5.4.4) Hardware and Software Requirements for Running the HERIG Package

HERIG can be run on any 8088/80286/80386 processor XT/AT/386 compatible microcomputer with or without a mathematics coprocessor. The machine must support Hercules graphics and contain at least one external disk drive. Interfacing with the process requires two DT2801 ADC cards and two DT2815 DAC cards. The program can be run without these cards for data analysis by changing the file "herig.sys" as specified in the file. Concurrent running of the memory resident INT10, HGC and HARDCOPY graphics support software programs is required when running this program.

5.4.5) Appendix Listings for the HERIG Package

Appendix N contains all the specific information concerning the HERIG system in the sections listed below:

- **N1:** HERIG System Subroutines
- **N2:** HERIG Data Manipulation Subroutines
- **N3:** Utility Routines for Analog Interface Cards
- **N4:** HERIG Menu Definition Routine and Modified Assembler Routines Originally Written by Ian Fisher
- **N5:** HERIG Include Files
- **N6:** HERIG Development/Execution Information

Additional routines written by Fisher (Fisher, 1988) have been linked into the libraries used in the development of this package, and source code listings of these routines can be found in his M.Sc. dissertation. Utility routines developed for the OPTCAD package have also been linked to these libraries.

5.5) Concluding Remarks

The menu structure of the OPTCAD CACSD package provides a user-friendly tool for the synthesis, analysis and simulation of state feedback control systems with state observers.

Process, controller, synthesis, analysis and simulation data is easily manipulated using the data load/save/edit options.

The LQG synthesis option automatically synthesizes a state feedback controller gain matrix K_c and Kalman state observer gain matrix K_f from the specified control weighting matrices (Q_1, R_1) and noise covariance matrices (Q, R).

The user is able to specify the frequency ranges over which a control system is to be analysed using performance indices. All of the created performance index data is presented to the user in a group of four plots, as well as comments derived from the plots regarding the quality of a particular design. Any one of the four plots can also be observed on their own.

The scales and formats for the time simulation option can be defined by the user for the open or closed loop digital simulation of a specific control system.

The HERIG package developed for the running, control and analysis of the heat exchanger process is also menu-driven. The rig running interface screen provides a logical representation of the process inputs, outputs and setpoints to the user, allowing easy observation of happenings on the process.

Process and controller parameters can be manipulated using the parameter load/save/modify options, and test data can be loaded or saved. The LQG controller data file format is identical to that used by the OPTCAD package, enabling easy implementation of control systems designed using the package.

The 8 modes of operation of the rig running algorithm enable the user to perform a variety of operations on the rig while it is running under manual, stabilizing or LQG control. Open and closed loop step tests can be performed, perturbations injected, and test data logged. These operations are invoked by selecting one of the options on the interface screen.

The test data captured can be plotted out on predefined or user-specified sets of axes. This plotting facility assists the user in determining certain process parameters (eg. process model, response times, noise/disturbance/setpoint response, etc).

CHAPTER SIX

PERFORMANCE INDEX ANALYSIS OF HEAT EXCHANGER CONTROLLERS

6.1) Introduction

All of the elements in the modified heat exchanger process model $G_m(s)$ are stable, so it is possible to control only two of the outputs, outlet temperatures T1 and T5, using the flow inputs F1 and F2 (chapter 2). The remaining inputs can be kept constant and the remaining outputs merely observed.

The theory of performance index analysis is put to test in this chapter. This is done as follows:

i) Create performance index analysis data for trial LQG control systems synthesized to control temperatures T1 and T5 using flows F1 and F2.

ii) Predict performance of the systems from this data in terms of their feedback properties.

iii) Implement the control systems on a digital simulator and on the actual process and do tests to verify whether the predictions made were correct.

6.2) Determination of Process Model

6.2.1) Frequency Domain Process Model

The determined model relating the flow inputs F1 and F2 to the temperature outputs T1 and T5, referred to as $G_{ft}(s)$, is a 2x2 matrix of polynomials in the frequency domain. Each element $gft_{ij}(s)$ of $G_{ft}(s)$ is thus defined as:

$$G_{ft}(s) = \{gft_{ij}(s)\} \; (i = 1..2, \; j = 1..2)$$

$$= \begin{bmatrix} gft_{11}(s) & gft_{12}(s) \\ gft_{21}(s) & gft_{22}(s) \end{bmatrix} \qquad (6.1)$$

$G_{ft}(s)$ relates the inputs and outputs as shown in (6.2).

$$\begin{bmatrix} T1(s) \\ T5(s) \end{bmatrix} = \begin{bmatrix} y1(s) \\ y2(s) \end{bmatrix} = G_{ft}(s) \begin{bmatrix} F1(s) \\ F2(s) \end{bmatrix} = G_{ft}(s) \begin{bmatrix} u1(s) \\ u2(s) \end{bmatrix} \qquad (6.2)$$

The elements $gft_{ij}(s)$ are determined by doing step tests on each input j (j = 1..2) and observing the response of both of the outputs i (i = 1..2) for each test. A transfer function model for $gft_{ij}(s)$ relating an input i to an output j is then fitted to the response data for each test.

Table 6.1 shows which input is stepped and output observed in the determination of each element.

Element $gc_{ij}(s)$	Input Stepped	Output Observed
$gft_{11}(s)$	F1(s)	T1(s)
$gft_{21}(s)$	F1(s)	T5(s)
$gft_{12}(s)$	F2(s)	T1(s)
$gft_{22}(s)$	F2(s)	T5(s)

Table 6.1: Inputs Stepped and Outputs Observed in the Determination of Transfer Function Elements

The responses of both outputs to three step tests done on each input are shown in Appendix O. A model is also fitted in the appendix to each of the observed responses using an algorithm called NELM (Astrom & Wittenmark, 1984).

The mean of the three transfer function models determined for each element $gc_{ij}(s)$ is quoted in table 6.2. The mean of each constant in each transfer function element is quoted, followed by (separated by ±) the standard deviation in the constant.

Element $gft_{ij}(s)$	Transfer Function [(ADC units) / (DAC units)]	Dead-Time [sec]
$gft_{11}(s)$	$\frac{-0.198 \pm 0.0426}{1 + (858.5 \pm 162.79)\ s}$	60.0 ± 8.17
$gft_{21}(s)$	$\frac{-0.604 \pm 0.0673}{1 + (379.8 \pm 104.45)\ s}$	66.7 ± 12.47
$gft_{12}(s)$	$\frac{0.298 \pm 0.0499}{1 + (928.3 \pm 158.53)\ s}$	8.3 ± 2.36
$gft_{22}(s)$	$\frac{0.754 \pm 0.0553}{1 + (509.9 \pm 33.37)\ s}$	0.0 ± 0.0

Table 6.2: Average Transfer Function Models determined for elements $gft_{ij}(s)$ (i = 1..2, j = 1..2)

6.2.2) Discussion of Determined Process Model

(i) First-order Responses and Unmodelled Dynamics

Table 6.2 shows that all of the elements in $G_{ft}(s)$ have been modelled by first order transfer functions with dead-times, and are in the form:

$$gft_{ij}(s) = \frac{K_{ij}\ \exp(-sT_{ij})}{1 + sT_{ij}} \qquad (6.3)$$

where:

K_{ij} : Open Loop Gain

T_{ij} : Response-time

$\mathcal{T}_{ij}$: Dead-time

Although the first order models are a good approximation to the actual model, some of the less significant higher-order dynamics in the responses have been neglected to reduce modelling complexities. Figure 6.1 shows one of the response plots analysed in Appendix O (Figure O12) which has been the affected by higher order dynamics. The actual and fitted responses are indicated on the plot, and the irregularity in the actual response trace at approximately time t = 300 seconds is clearly due to neglected higher order dynamics. Figure 6.2 shows the response of control valve C5, to the step in flow F1, as it compensates to stabilize the level L1. This causes a transient in the CH stream outlet flow from tank 1 which is the likely cause of the transient in the response of the CH outlet temperature T5. A more detailed analysis of the response is likely to indicate the presence of a faster system pole on the complex s-plane in the vicinity of s = $-2*\pi/200 = -0.031$ rad/sec.

(ii) System Dead-times

The dead-times of the elements in table 6.2 are generally short when compared to the corresponding response-times. The dead-time is less than 10% as long as the response-time for each element, except for $gtf_{21}(s)$, for which this value is 17.6%.

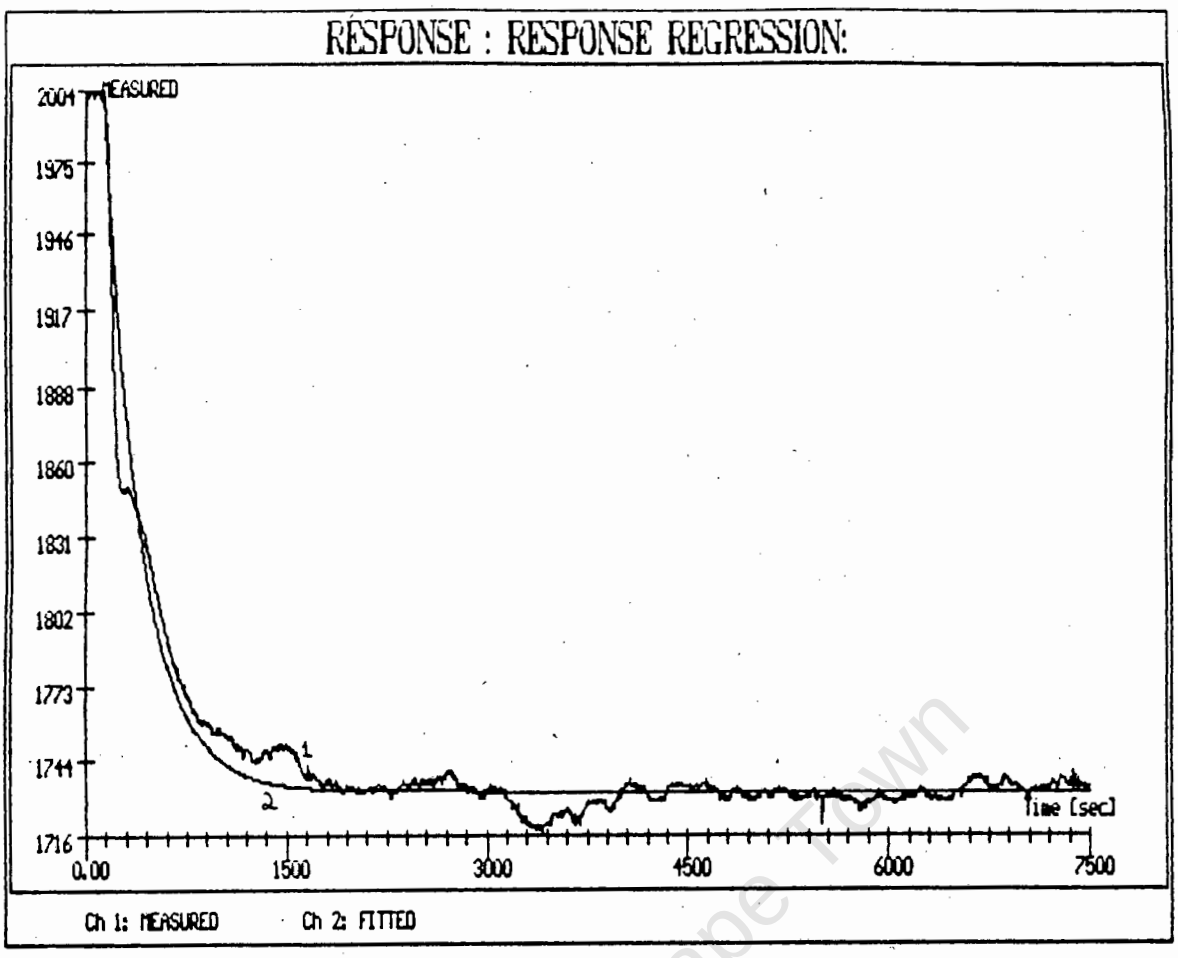


Figure 6.1: Response of T5 to a -410 unit Step in F1

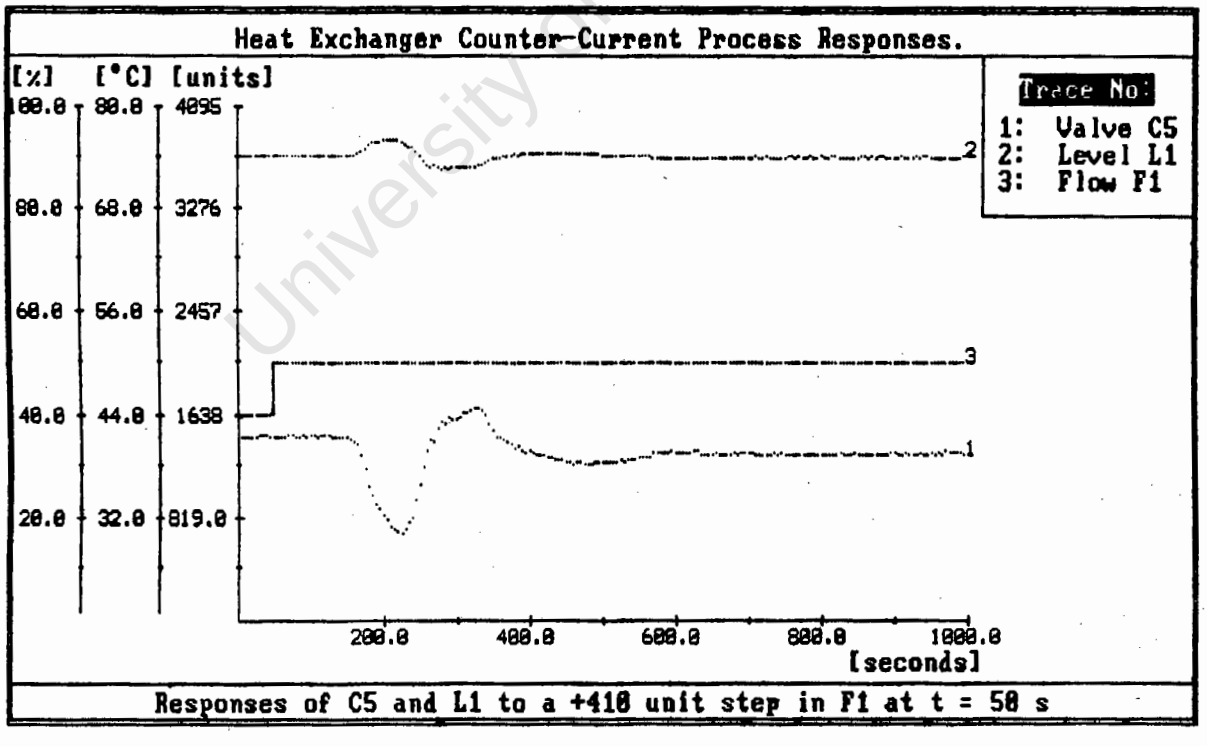


Figure 6.2: Response of C5 to a -410 unit Step in F1

(iii) Uncertainties and Nonlinearities

The large standard deviations obtained in determining the response-times (ranging between 6.5% to as high as 27.5% of mean), the open loop gains (7.3% to 21.5%) and the dead-times (0% to 28.4%) indicate that a significant degree of uncertainty exists in these values.

A control system designed for the process should provide adequate gain and phase margin to accommodate such uncertainty.

(iv) Low Gains in Process Model

The model $G_{ft}(s)$ relates the flow inputs to the temperature outputs according to (6.2). Any *dynamic changes* $\delta\{(F1,F2)^T\}$ in the flow inputs are thus similarly related to *dynamic changes* in the temperature outputs $\delta(\{T1,T5)^T\}$. The inverse of (6.2) will thus predict the changes in inputs required to realize a given change in the outputs, as shown in (6.4)

$$\begin{bmatrix} \delta\{F1(s)\} \\ \delta\{F2(s)\} \end{bmatrix} = G_{ft}^{-1}(s) \begin{bmatrix} \delta\{T1(s)\} \\ \delta\{T5(s)\} \end{bmatrix} \quad (6.4)$$

The changes in the steady-state inputs required to realize a specific change in the outputs can be

predicted by considering (6.4) when s = 0. At steady-state, thus,

$$G_{ft}^{-1}(0) = (G_{ft}(0))^{-1} = \begin{bmatrix} -0.198 & 0.298 \\ -0.604 & 0.754 \end{bmatrix}^{-1}$$

$$= \begin{bmatrix} 24.56 & -9.71 \\ 19.67 & -6.45 \end{bmatrix} \quad (6.5)$$

Consider the process running at steady-state with $(F1,F2)^T = (50\%,50\%)^T$ and $(T1,T5)^T = (28^\circ C,50^\circ C)$. Now assume that the steady-state temperature outputs are required to be changed to $(T1,T5)^T = (28^\circ C,52.5^\circ C)^T$. This represents a 2.5/60 change in the 20 to 80°C full scale temperature range, so the change in outputs is given as $\delta(\{T1,T5)^T\} = (0,0.042)^T$.

The change in the inputs required to realize this change in outputs, obtained from (6.4) at s = 0, will be $\delta\{(F1,F2)^T\} = (-0.403,-0.277)^T$. This means that F1 and F2 will change respectively by -0.403*100 and -0.277*100 of their 0 to 100% full scale ranges. The flow inputs required to realize the new temperature outputs will thus be $(F1,F2)^T = (9.7\%,22.3\%)^T$.

The drastic changes in the flow inputs required are due to the fact that the open loop gains in the elements of $G_{ft}(s)$ are very low. A great deal of control effort is

thus required to enable successful tracking of the setpoints by the outputs.

(v) Slow Response Times in Process Model

The response-times of the elements in $G_{ft}(s)$ are very slow, between 379.8 and 928.3 seconds. The gain of $G_{ft}(s)$ is thus reduced rapidly when passing through and beyond the frequency band 0.0010 to 0.0026 rad/sec.

6.2.3) Transformation Into State-Space Description

A LQG control system can only be synthesized for the process if the model is described as a state-space system.

The representation of dead-times in the state-space description of process models is cumbersome (Borrie, 1986), and since the dead-times in the elements of $G_{ft}(s)$ are generally short compared to their long response-times, the dead-times are neglected in the state-space description of the process.

The process model for $G_{ft}(s)$ shown in (6.6) is thus assumed in the description of the process, where the gains K_{ij} and response times T_{ij} are derived from table 6.2.

$$G_{ft}(s) = \begin{bmatrix} \frac{K_{11}}{1 + sT_{11}} & \frac{K_{12}}{1 + sT_{12}} \\ \frac{K_{21}}{1 + sT_{21}} & \frac{K_{22}}{1 + sT_{22}} \end{bmatrix} \qquad (6.6)$$

The system is transformed in Appendix P into the state-space system description (6.7) to (6.11).

$$d/dt(X(t)) = AX(t) + BU(t) \qquad (6.7)$$

$$Y(t) = CX(t) \qquad (6.8)$$

where:

$$A = \begin{bmatrix} K_{11} & K_{12} & 0.0 & 0.0 \\ 0.0 & -K_{12}/T_{12} & 0.0 & 0.0 \\ 0.0 & 0.0 & -1/T_{21} & 0.0 \\ 0.0 & 0.0 & K_{21} & K_{22} \end{bmatrix}$$

$$= \begin{bmatrix} -1.17E\text{-}03 & 1.73E\text{-}05 & 0.0 & 0.0 \\ 0.0 & -1.08E\text{-}03 & 0.0 & 0.0 \\ 0.0 & 0.0 & -2.63E\text{-}03 & 0.0 \\ 0.0 & 0.0 & 5.07E\text{-}04 & -1.96E\text{-}03 \end{bmatrix} \qquad (6.9)$$

$$B = \begin{bmatrix} K_{11}/T_{11} & K_{12}/T_{12} \\ 0.0 & -K_{12}/(K_{11}T_{12}) \\ -K_{21}/(K_{22}T_{21}) & 0.0 \\ K_{21}/T_{21} & K_{22}/T_{22} \end{bmatrix}$$

$$= \begin{bmatrix} -2.31E\text{-}04 & 3.21E\text{-}04 \\ 0.0 & 1.62E\text{-}03 \\ 2.11E\text{-}03 & 0.0 \\ -1.59E\text{-}03 & 1.48E\text{-}03 \end{bmatrix} \qquad (6.10)$$

$$\mathbf{C} = \begin{bmatrix} 1.0 & 0.0 & 0.0 & 0.0 \\ 0.0 & 0.0 & 0.0 & 1.0 \end{bmatrix} \qquad (6.11)$$

Note that the transformed system has been rearranged so that the matrix **C** maps two of the process states $x_1(t)$ and $x_4(t)$, of $\mathbf{X(t)} = (x_1(t), x_2(t), x_3(t), x_4(t))^T$, directly to the process outputs $\mathbf{Y(t)} = (y_1(t), y_2(t))^T$ in (6.8). This has been done to make the limiting of the states in the LQG procedure more meaningful, The limiting of states $x_1(t)$ and $x_4(t)$ with weighting matrix $\mathbf{Q_1}$ then effectively reduces to the limiting of the process outputs.

6.2.4) Controllability and Observability of System

The system (6.7)..(6.11) has been tested and found to be both controllable and observable, meaning that all of the modes can be controlled by the inputs and that all of the modes affect the outputs (Borrie, 1986).

6.3) Design Synthesis, Implementation and Analysis Procedure

Three trial control systems are synthesized for the process.

Trial control system 1 is synthesized for the process as a design starting point in section 6.4.

Trial control system 2 is synthesized and analysed in section 6.5. The analysis data is verified by implementation of the system on a digital simulator and on the process itself.

Trial control system 3 is synthesized and analysed in section 6.6. The analysis data is verified by implementation on a simulator, and reasons for instabilities encountered when implementation is attempted on the process are discussed.

The procedure adopted for the LQG synthesis and performance index analysis of control systems for the process is discussed in this section. The procedure for the verification of the analysis data is also discussed.

6.3.1) LQG Synthesis

Weighting and covariance matrices are chosen to specify the objectives of each trial control system, giving reasons for the choices. A LQG controller based on these matrices is synthesized in each case using the "Synthesize" option in the OPTCAD package, producing an appropriate state feedback controller matrix, K_c and Kalman gain matrix, K_f.

The eigenvalues of $A - BK_c$ indicate the pole positions of the state feedback system, whereas the eigenvalues of $A - K_fC$ indicate the poles of the Kalman filter. These C/L pole positions for each of the trial systems are stated and commented on.

6.3.2) Performance Index Analysis

Selecting the "Analyse" option in the main menu of OPTCAD initiates the performance index analysis procedure. Selecting the "Create" submenu allows the specification of the evaluation frequency range and the number of points to be analysed.

Performance index analysis data (100 data points per index) has been created for trial systems 2 and 3 over the frequency range 0 to 0.026 rad/sec. This upper frequency limit was chosen as it is the highest frequency modelled in $G_{ft}(s)$. No significant higher frequency effects were noticed in performance index plots over a wider frequency band (up to 10 rad/sec).

A group-plot of the data, as well as the comments produced by the package predicting the frequency-dependent performance, are presented for these systems (similar to figures 5.2 and 5.3).

6.3.3) Verification of Performance Index Data

Predictions made by the performance index analysis data for systems 2 and 3 are verified in appendices Q and R respectively. This is done by undertaking a more detailed analysis, over each of the three frequency bands, of each of the plots in the group plots. Quantitative predictions of the quality of the design feedback properties are made from

this analysis and presented in a tabular form in tables 6.3 and 6.4. These quantitative predictions are then verified by tests done on the control system, the results of which are presented in the same tables.

(i) Transmission of Noise, Disturbances and Setpoints

Consider, for example, the verification of data created predicting the input response to sensor noise perturbations, **N(s)**.

Two frequency points in each of the three analysis frequency bands (low, intermediate and high, as defined in section 5.3) are chosen. The values of the index $\|T_{UN}(s)\|_{max}$ at these two frequencies in each band are observed from detailed plots in appendices Q and R. The maximum of these two values is noted in tables 6.3 and 6.4 for systems 1 and 2 respectively. This magnitude predicts the maximum transmission, to the inputs, of noise perturbations synthesized from a mixture of the two frequencies under consideration.

Sinusoidal noise perturbations of equal amplitudes A_{pert} and at each of the two selected frequencies are injected into each of the two channels of the control loop. The response of the inputs to these perturbations are logged for all possible phase relationships between the two perturbing signals. The response plots for the

simulated and actual implementation of the system on the process are shown in appendices Q and R, and the maximum response magnitude observed in the two input channels, MR_{max} is determined. The ratio MR_{max}/A_{pert} indicates the maximum transmission, between the selected frequencies, of sensor noise to the inputs.

The theory in chapter 4 predicts that the actual maximum response ratio MR_{max}/A_{pert} should always be less than or equal to the predicted maximum response ratio $\|T_{UN}(s)\|_{max}$. The analysis data can thus be verified by comparing the predicted ratios to the actual response ratios.

The method of verification of the predictions made regarding the response of the inputs to sensor noise in a particular frequency band is generalized to the verification of all the performance indices predicting the input and output response to noise and disturbance perturbations over all frequency bands. The predicted input and error responses to setpoint perturbations is also verified in this manner.

The response ratios obtained from simulated response data, as well as from response data of the actual process, are tabulated next to the predicted ratios for comparison in tables 6.3 and 6.4. Instabilities encountered in the implementation of system 3 on the

process means that only the predicted and simulated results are compared for this system in table 6.4.

(ii) Sensitivity to Changes in Process Model

Discrepancies between the simulated and observed responses of the inputs, outputs and errors to perturbations may occur if the inputs and outputs are sensitive to changes in the process model. These effects are observed in the table 6.3 in which the simulated and observed results for system 2 are shown.

(iii) Stability and Stability Margins

The predicted stability margins are verified by observing input and output responses to closed loop step tests done on the setpoints of the simulated model (systems 2 and 3) and on those of the actual process (system 2).

6.4) Synthesis and Discussion of Trial System 1

6.4.1) LQG Synthesis

All of the weighting and covariance matrices are chosen as the identity matrix, as a synthesis starting-point.

$Q_1 = I$ and $R_1 = I$ implies that excursions in all of the states and inputs are penalized equally. In practice, however, excursions only in the more expensive inputs should be limited, and the effects of the states on the outputs should be considered before arbitrarily limiting them.

$Q = I$ and $R = I$ implies that the noise processes $\alpha(t)$ and $\beta(t)$ are independent white Gaussian noise processes with zero means and unity variances, which is not very realistic, but provides a starting point for the design.

A LQG control system synthesized using these matrices produces the gain matrices quoted in (6.14) and (6.15).

$$K_c = \begin{bmatrix} -7.00E\text{-}02 & 3.70E\text{-}02 & 3.13E\text{-}01 & -2.63E\text{-}01 \\ 9.10E\text{-}02 & 5.00E\text{-}01 & 3.80E\text{-}02 & 2.58E\text{-}01 \end{bmatrix} \tag{6.14}$$

$$K_f = \begin{bmatrix} 9.99E\text{-}01 & .00E+00 \\ -8.00E\text{-}03 & .00E+00 \\ .00E+00 & 9.50E\text{-}02 \\ .00E+00 & 9.98E\text{-}01 \end{bmatrix} \tag{6.15}$$

The poles of the state feedback system lie at $s = -0.0012$, $s = -0.0016$, $s = -0.0026$ and $s = -0.0039$. These poles are all stable and lie near the positions of the system open loop poles.

The poles of the Kalman filter system lie at s = -0.00108, s = -0.00286, s = -1.0 and s = -1.0. Although all of these poles are stable, the repeated pole at s = -1.0 is almost 500 times as fast as the fastest pole in the open loop system, which was pointed out in chapter 4 as being an undesirable property when considering the overall stability of the system. These poles in the observer could cause unmodelled and/or unknown faster modes in the process to be excited, driving the system unstable.

The design acts only as a starting point, so performance index analysis and verification for this system is omitted.

6.5) Synthesis, Implementation and Analysis of Trial System 2

6.5.1) LQG Synthesis

The concept of limiting the states of a system becomes clearer when rewriting the LQG cost function J_C as:

$$J_C = \lim_{T\to\infty} E\{ {}^1/_{2T} \int_{-T}^{T} \{Y^T(t)Q'_1Y(t) + U^T(t)R_1U(t)\}\, dt \qquad (6.16)$$

Note that:

$$\begin{aligned} Y^T(t)Q'_1Y(t) &= (CX(t))^T\, Q'_1\, CX(t) \\ &= X(t)^T\, (C^TQ'_1C)\, X(t) \\ &= X(t)^T\, Q_1\, X(t) \end{aligned} \qquad (6.17)$$

Comparing (6.16) with the expression for J_c quoted in section 4.5.1, and considering (6.17), shows that the state weighting matrix Q_1 can be constructed from a matrix Q'_1 which limits excursions in the outputs. The resultant expression for this matrix is given in (6.18).

$$Q_1 = C^T Q'_1 C \qquad (6.18)$$

$Q'_1 \in R_{2x2}$ is chosen as the identity matrix to limit both outputs equally, realizing the state weighting matrix Q_1 shown in equation (6.19). $R_1 = I$ is once again adopted for this trial system.

$$Q_1 = \begin{bmatrix} 1.0 & 0.0 & 0.0 & 0.0 \\ 0.0 & 0.0 & 0.0 & 0.0 \\ 0.0 & 0.0 & 0.0 & 0.0 \\ 0.0 & 0.0 & 0.0 & 1.0 \end{bmatrix} \qquad (6.19)$$

In an attempt to slow down the observer poles and specify a noise process $\alpha(t)$ only on the differential of states affecting the outputs, the matrix Q is modified and becomes $Q = 0.1*Q_1$, with $R = I$ as before.

A LQG controller based on these matrices is synthesized to produce the gain matrices quoted in (6.20) and (6.21).

$$K_c = \begin{bmatrix} -7.62E\text{-}02 & -4.92E\text{-}04 & -2.32E\text{-}02 & -2.88E\text{-}01 \\ 1.14E\text{-}01 & 7.90E\text{-}04 & 2.81E\text{-}02 & 3.05E\text{-}01 \end{bmatrix} \qquad (6.20)$$

$$K_f = \begin{bmatrix} 3.15E-01 & .00E+00 \\ 1.59E-06 & .00E+00 \\ .00E+00 & 2.02E-05 \\ .00E+00 & 3.14E-01 \end{bmatrix} \quad (6.21)$$

The state feedback system poles are located at s = -0.0033, s = -0.0011, s = -0.0012 and s = -0.0022, once again lying near the positions of the system open loop poles.

The Kalman filter system pole positions are evaluated to be at s = -0.00108, s = -0.00263, s = -0.32 and s = -0.32. All of these poles are stable, and this time the repeated pole at s = -0.32 is about 100 times as fast as the fastest pole in the open loop system.

6.5.2) Performance Index Analysis

Figure 6.3 shows a group plot of performance index data created for this trial system. The analysis of the data is presented in figure 6.4.

6.5.3) Verification of Performance Index Data

Appendix Q contains individual plots of each of the four the performance index graphs shown in figure 6.3, as required for the verification procedure described in section 6.4.3.

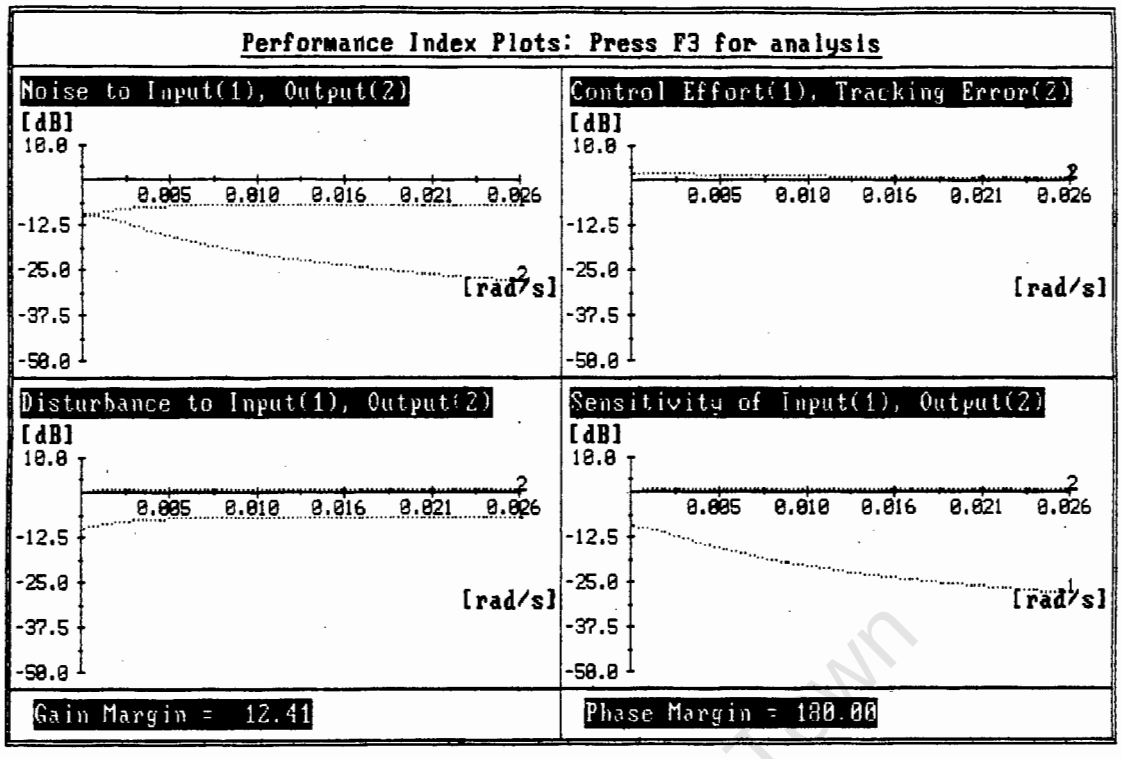
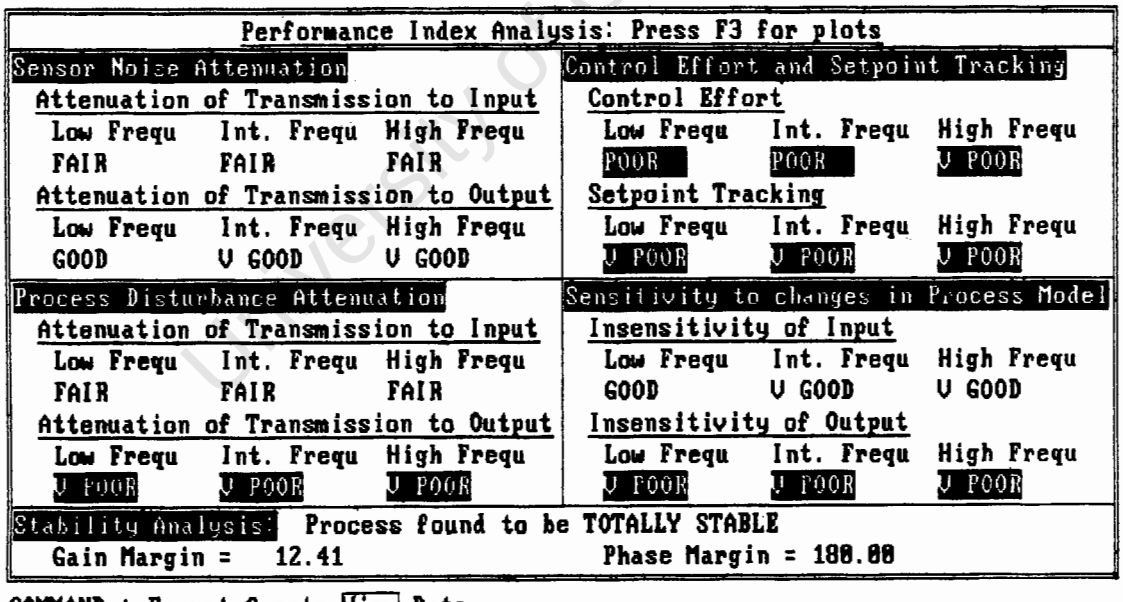


Figure 6.3: System 2 - Performance Index Group Plot



COMMAND : Format Create View Data

BRIEF : View Performance Index Data and Analysis.

Figure 6.4: System 2 - Comments on Performance Index Group Plot

(i) Transmission of Noise, Disturbances and Setpoints

Plots of the simulated response of the inputs and outputs to noise, disturbance and setpoint perturbations are shown in Appendix Q. The predicted and simulated maximum response ratios obtained from this appendix are tabulated together in table 6.3.

Finally, the design was implemented on the process itself. The process input and output response traces are also shown in Appendix Q. The observed maximum response ratios are quoted in table 6.3.

Table 6.3 also indicates percentage errors which occur between the predicted Mp and tested Mt response ratio maxima. Note that the ratio maxima are quoted in decibels to cope widely ranging numbers and errors which occur are calculated by considering the absolute maxima, Abs(Mp) and Abs(Mt), according to equation 6.22.

$$\% \text{ Error} = 100 * (\text{Abs}(Mt) - \text{Abs}(Mp)) / \text{Abs}(Mp) \qquad (6.22)$$

A negative percentage error indicates that a tested maximum is in fact less than its predicted maximum.

Good correlation exists between the predicted and simulated maxima for all indices except for $\|T_{YN}(s)\|_{max}$. The maxima obtained for remaining indices differ from their predicted values by less than 10% in all cases.

The negative errors in verifying $\|T_{UN}(s)\|_{max}$ and $\|T_{UD}(s)\|_{max}$ indicate that the tested response ratio maxima were all less than their predicted values.

Similar results are observed for the process response maxima, although the errors in $\|T_{YN}(s)\|_{max}$, $\|T_{YD}(s)\|_{max}$ and $\|ERR(s)\|_{max}$ are generally larger than those observed in the simulation results. The larger errors observed in these response ratio magnitudes can be explained when considering the output sensitivity performance index $\|SE_Y(s)\|_{max}$ (0dB in table 6.3 in all bands) and the fact that large uncertainties in the $G_{ft}(s)$ shown in section 6.2.2 suggest that large changes in the process model are to be expected. Any changes in the process model are transmitted directly to the outputs and will affect the output perturbation response magnitudes. This sensitivity performance index can thus also be used to predict inaccuracies which may occur in the output response performance indices, which are based on the nominal process model.

Numerical errors in the calculation of the matrix $T_2(s)$ in the OPTCAD package could mean that the predictions made by the index $\|T_{YN}(s)\|_{max}$ are incorrect. This could explain the large errors for the simulated and observed results in verifying $\|T_{YN}(s)\|_{max}$

			Frequency Band		
			Low	Intermed.	High
$\|T_{UN}(s)\|_{max} = \|W(s)\|_{max}$		(dB)	-7.7	-7.1	-7.1
MR_{max}/A_{pert}	Simulated	(dB)	-8.18	-7.74	-7.74
	% Error	(%)	-0.54	-0.71	-0.71
MR_{max}/A_{pert}	Observed	(dB)	-8.32	-7.42	-7.34
	% Error	(%)	-0.69	-0.36	-0.27
$\|T_{YN}(s)\|_{max} = \|T_2(s)\|_{max}$		(dB)	-15.0	-22.3	-26.3
MR_{max}/A_{pert}	Simulated	(dB)	-13.15	-18.42	-22.05
	% Error	(%)	23.74	56.32	63.12
MR_{max}/A_{pert}	Observed	(dB)	-11.55	-16.32	-22.14
	% Error	(%)	48.76	99.07	61.44
$\|T_{UD}(s)\|_{max} = \|W(s)\|_{max}$		(dB)	-7.7	-7.1	-7.1
MR_{max}/A_{pert}	Simulated	(dB)	-8.18	-7.96	-7.96
	% Error	(%)	-5.38	-9.43	-9.43
MR_{max}/A_{pert}	Observed	(dB)	-8.37	-7.52	-7.47
	% Error	(%)	-7.42	-4.72	-4.17
$\|T_{YD}(s)\|_{max} = \|S_2(s)\|_{max}$		(dB)	0.0	0.0	0.0
MR_{max}/A_{pert}	Simulated	(dB)	0.34	0.42	0.42
	% Error	(%)	3.99	4.95	4.95
MR_{max}/A_{pert}	Observed	(dB)	0.34	0.90	0.63
	% Error	(%)	3.99	10.92	7.52
$\|T_{UR}(s)\|_{max}$		(dB)	0.0	0.0	0.0
MR_{max}/A_{pert}	Simulated	(dB)	0.66	0.42	0.25
	% Error	(%)	7.90	4.95	2.92
MR_{max}/A_{pert}	Observed	(dB)	0.64	0.52	0.57
	% Error	(%)	7.65	6.17	6.78
$\|ERR(s)\|_{max}$		(dB)	1.13	0.54	0.36
MR_{max}/A_{pert}	Simulated	(dB)	1.58	0.82	0.42
	% Error	(%)	5.32	3.28	6.93
MR_{max}/A_{pert}	Observed	(dB)	1.38	1.63	1.38
	% Error	(%)	2.92	13.37	12.46
$\|SE_U(s)\|_{max} = \|T_1(s)\|_{max}$		(dB)	-15.0	-22.3	-26.8
$\|SE_Y(s)\|_{max} = \|S_2(s)\|_{max}$		(dB)	0.0	0.0	0.0

Table 6.3: **Predicted, Simulated and Observed Maximum Response Ratios**

(Obtained from tables Q.1 to Q.9)

(ii) Sensitivity to Changes in Process Model

The sensitivity indices quoted in table 6.3 predict that the inputs will be insensitive to changes in the process model and will become more insensitive as the frequency of the input signals increase. The predicted 0dB output sensitivity indices means that any changes in the process model will directly affect the outputs in all frequency bands.

These predictions explain the generally larger differences between the simulated and observed values for $\|\mathbf{ERR(s)}\|_{max}$ than for $\|\mathbf{T_{UR}(s)}\|_{max}$.

(iii) Stability and Stability Margins

Figure 6.5 shows the simulated C/L responses of the inputs and outputs to a step in each of the temperature setpoints. Traces 1, 2 and 3 on the left hand set of simulation axes represent setpoint T1, input F1 and output T1 respectively, while these traces represent setpoint T5, input F2 and output T5 respectively on the right hand set of axes. Setpoint T1 is stepped by +10 units at t = 100 sec, while T5 is stepped by +10 units at t = 5100 sec.

Figures 6.6 and 6.7 show the responses of both of the actual process outputs to steps in temperature setpoints

T1 and T5 respectively. In figure 6.6, setpoint T1 is stepped by +673 units at t = 100s, and then by -1000 units at t = 5000 sec. In figure 6.7, setpoint T5 is stepped by +1630 units at t = 100 sec and by -2048 units at t = 5010 sec.

The simulated responses are C/L stable, and the zero output overshoot verifies the predicted very large 180° phase margin. The process output responses are also C/L stable, meaning that the predicted gain margin of 12.41 is sufficient so as not to allow changes in the model to destabilize the system. The simulated output response times are 900 sec for T1 and 350 sec for T5. The process response times vary between 800 and 900 sec for output T1, and between 320 and 400 sec for output T5.

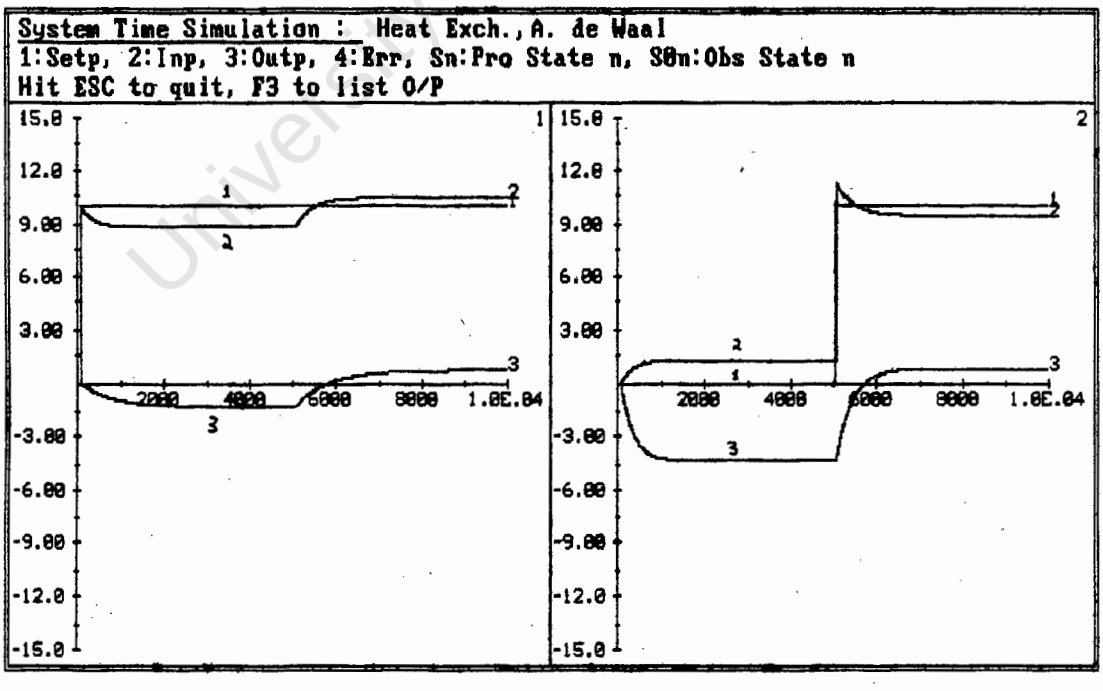


Figure 6.5: System 2 - Simulated C/L Input/Output Response

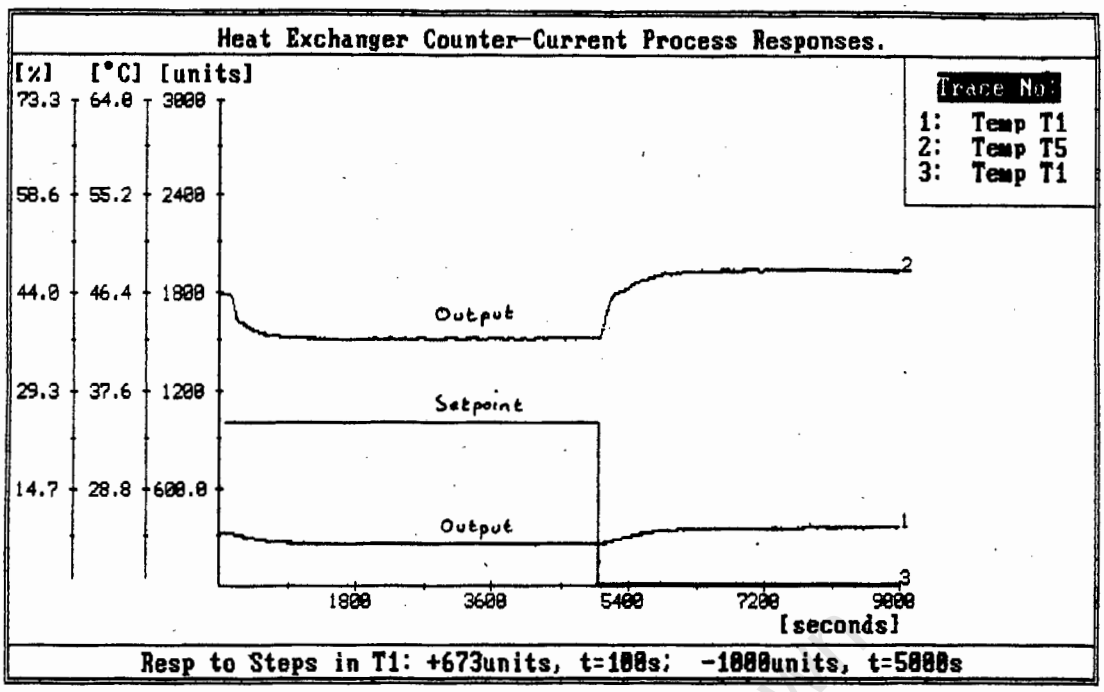


Figure 6.6: System 2 - Observed C/L Response of Outputs T1 and T5 to Steps in Setpoint T1

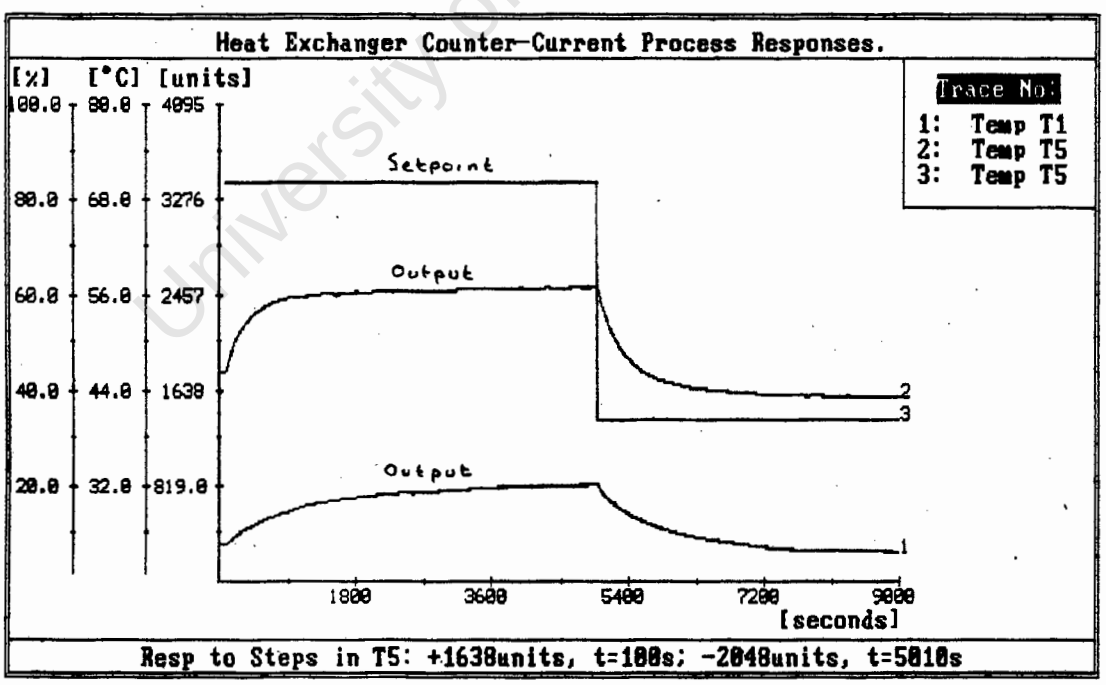


Figure 6.7: System 2 - Observed C/L Response of Outputs T1 and T5 to Steps in Setpoint T5

(iv) Control Effort and Setpoint Tracking

Chapter 4 indicates that these properties are not strictly "feedback properties". The LQG synthesis technique, furthermore, is not concerned with the tracking of the C/L inputs **V(s)** (= **R(s)** for $\mathbf{P_c(s)} = \mathbf{I}$ in this case) by the outputs. This lack of tracking can be seen in the C/L step test plots shown in figures 6.5 to 6.7.

A precompensator, indicated as $\mathbf{P_c(s)}$ in figure 4.3, can alter the tracking and control effort properties without affecting any of the other feedback properties of the system, which are independent of the precompensator matrix. A constant precompensator matrix $\mathbf{P_c}$ is thus determined for the process, to ensure steady-state tracking of the setpoints **R(s)**, by inverting the C/L process model at DC. This matrix is given in (6.23), and the resultant control effort and setpoint tracking performance index plot is given in figure 6.8.

$$\mathbf{P_c} = \begin{bmatrix} 2.41E+01 & -9.83E+00 \\ 2.04E+01 & -6.38E+00 \end{bmatrix} \qquad (6.23)$$

The performance index plot in figure 6.8 predicts zero (less than -30dB) steady-state tracking error, with the inevitable trade-off of large control effort (more than 30dB amplification of setpoint signals to inputs). The sharp increase of the tracking error index with

frequency indicates that the output will not track setpoints at frequencies much higher than DC. The tracking index is greater than 0dB for all frequencies above 0.001 rad/sec, meaning that the maximum time required for errors to decay will be in the order of 6000 seconds.

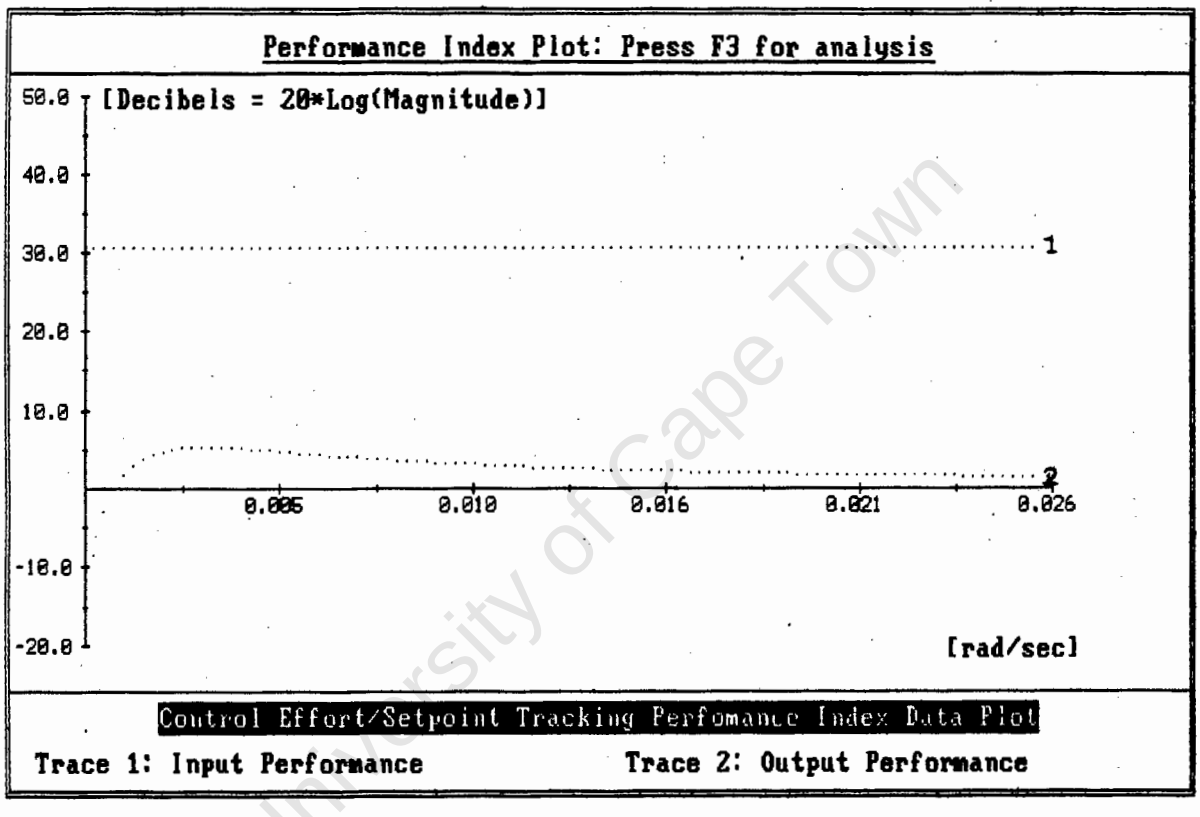


Figure 6.8: System 2 - Setpoint Tracking / Control Effort Performance Index Plot (Including P_C)

The predictions are verified in figures 6.9 and 6.10 showing the simulated input and output responses respectively of the system, including precompensator P_C, to a +10 unit step in setpoint T1 at t = 100 sec, and a +10 unit change in setpoint T5 at t = 5100 sec. The outputs track their respective setpoints at steady-

state. The errors in output T1 have not decayed entirely after 5100 seconds, verifying the predicted maximum error decay time. Very large changes are seen in the flow inputs, as required to realize the changes in the temperature outputs (25dB maximum transmission of setpoint to input F1 in figure 6.9). This phenomenon is predicted by the control effort performance index, and is also discussed in section 6.2.2.

The stepping of setpoint T1 affects the output T5 to a large extent, while the effect of stepping T5 on the output T1 is very slight. Performance index analysis has no means of predicting this interaction or in which of the loops it will occur.

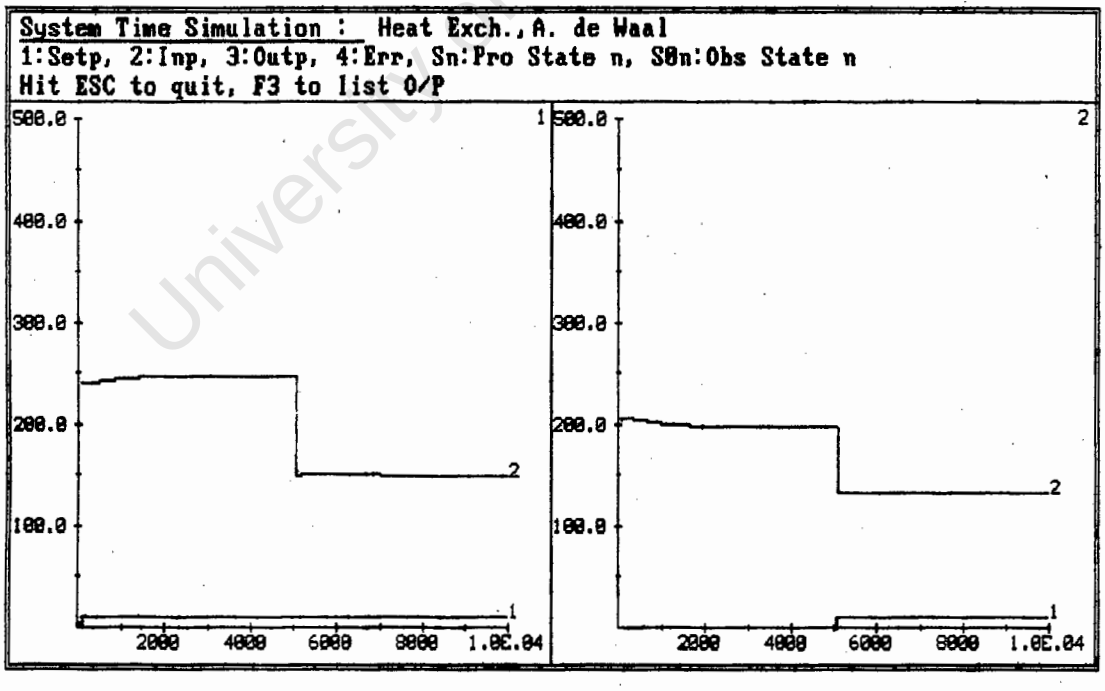


Figure 6.9: System 2 - Simulated C/L Responses of Inputs (Including P_C)

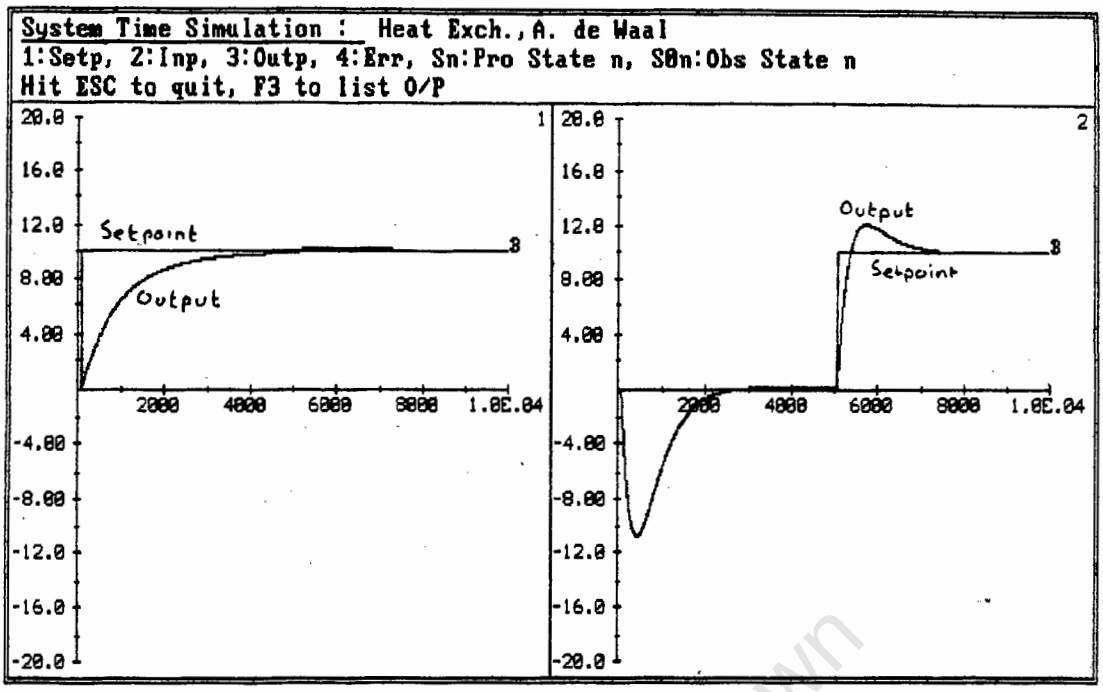


Figure 6.10: System 2 - Simulated C/L Responses of Outputs (Including P_C)

Figure 6.11 shows the response of the process inputs to the stepping of setpoint T1 by +34 units at t = 80 sec and by -68 units at t = 6070 seconds. Figures 6.12 and 6.13 show the response of outputs T1 and T5 respectively to these step tests. Figure 6.14 shows the response of the process inputs to the stepping of setpoint T5 by +136 units at t = 100 sec by -272 units at t = 4600 seconds. The responses of outputs T1 and T5 to these step tests are shown in figure 6.15

The same input trends as observed in the process responses as in the simulation. The steady-state maximum transmission of setpoints to the inputs in figure 6.11 (observed in F1) varies between 24.6dB and

27.6dB, while in figure 6.14 (also in F1) this maximum is 19.7dB. These maxima are all less than their predicted 30dB maximum. The failure of the outputs to track the setpoints exactly at steady-state is due effects of changes in the process model, as predicted by $\|SE_Y(s)\|_{max}$. Figure 6.12 shows a -5.1dB maximum transmission of setpoint to error for T1 and a -9.2dB maximum in the transmission for T5 is observed in figure 6.15.

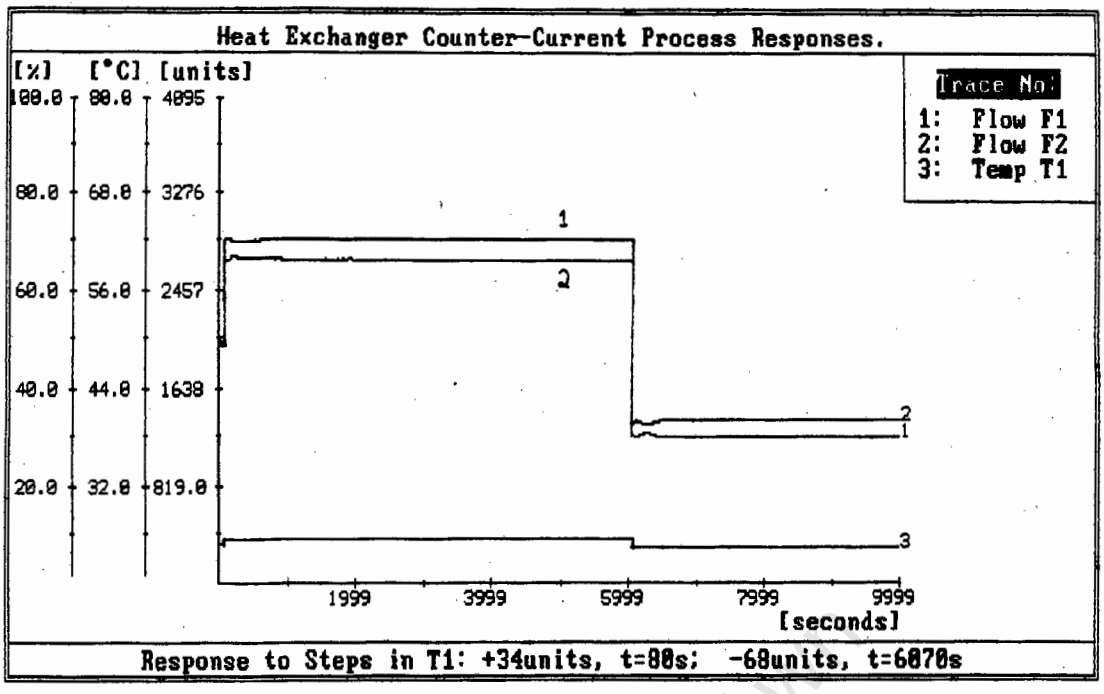


Figure 6.11: System 2 - Observed C/L Response of Inputs F1 and F2 to Steps in Setpoint T1 (Including P_c)

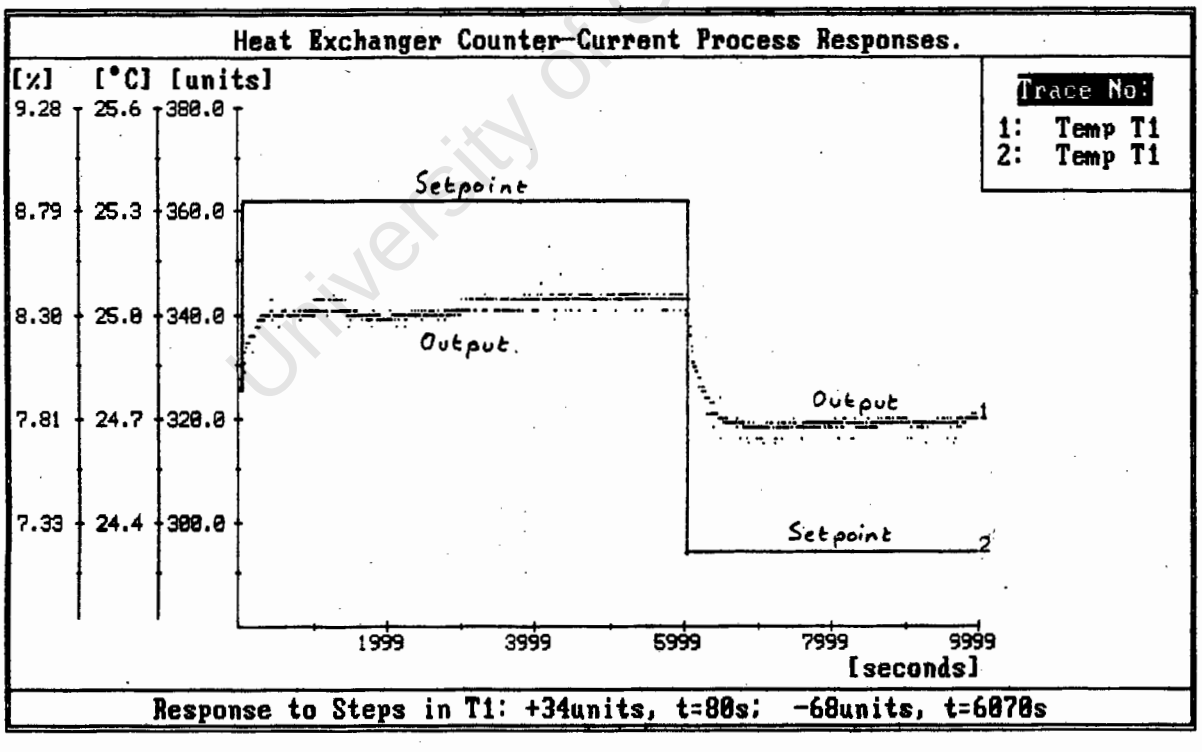


Figure 6.12: System 2 - Observed C/L Response of Output T1 to Steps in Setpoint T1 (Including P_c)

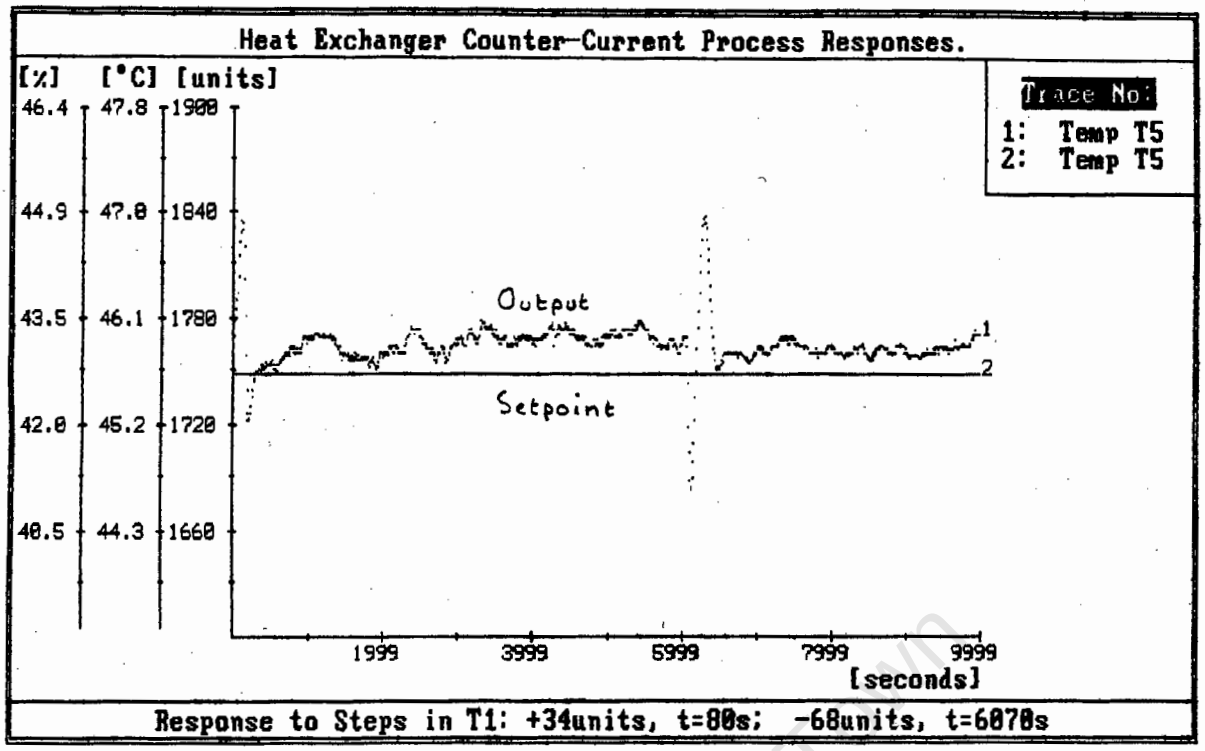


Figure 6.13: System 2 - Observed C/L Response of Output T5 to Steps in Setpoint T1 (Including P_c)

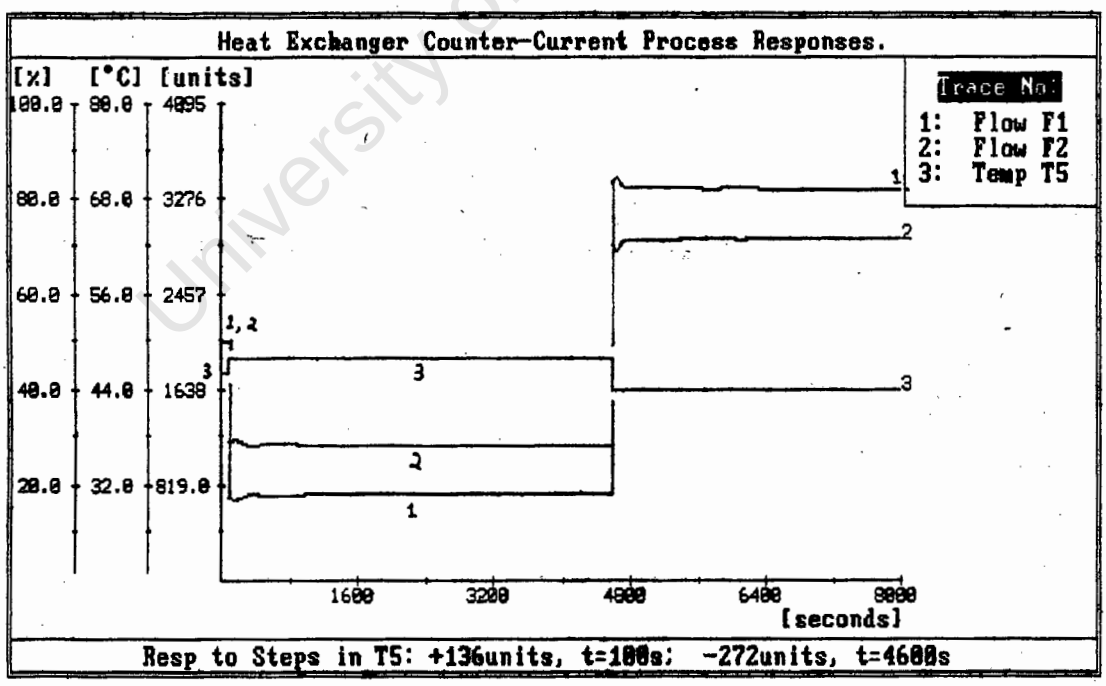


Figure 6.14: System 2 - Observed C/L Response of Inputs F1 and F2 to Steps in Setpoint T5 (Including P_c)

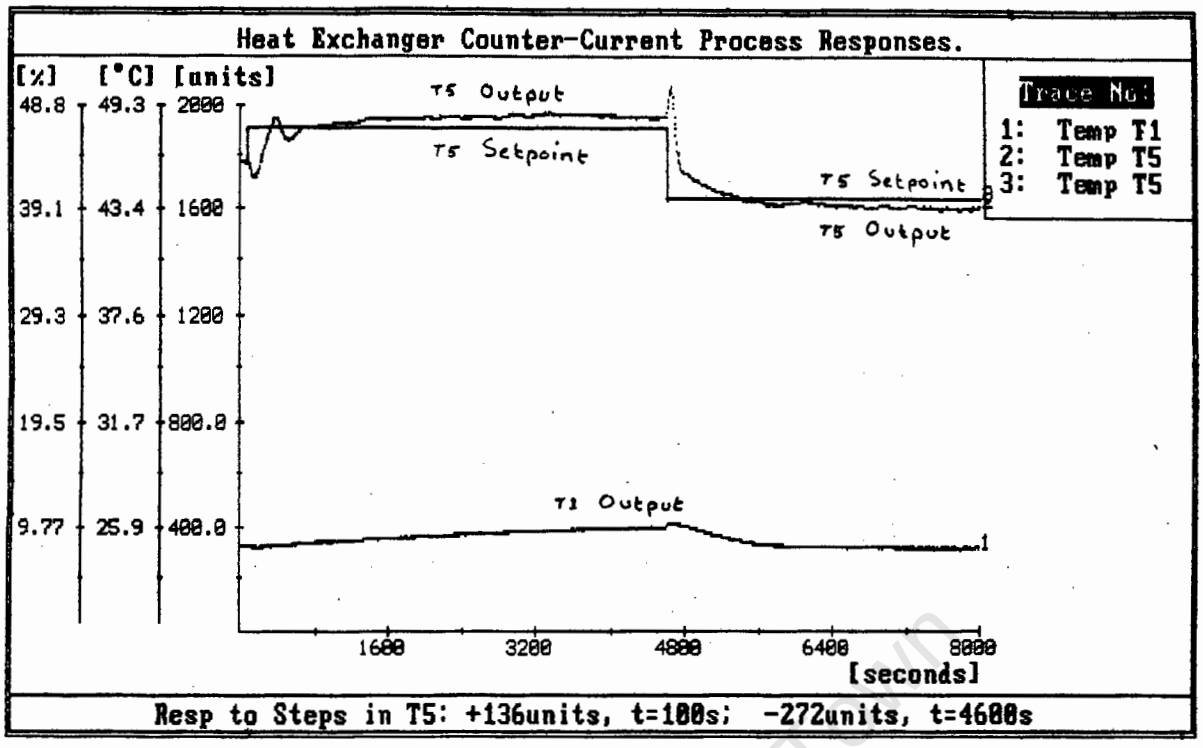


Figure 6.15: System 2 - Observed C/L Response of Outputs T1 and T5 to Steps in Setp. T5 (Including P_c)

6.5.4) Comments on Design

The small gains in $G_{ft}(s)$ cause the return matrices $L_i(s)$ to become very small (equations (4.5) and (4.6)), meaning that the characteristic matrices $S_i(s)$ tend towards the identity matrix (equation (4.9)) and the characteristic matrices $T_i(s)$ (equation (4.10)) tend to vanish. Equation (4.15) shows that $W(s)$ also tends to vanish as $G_{ft}(s)$ and $L_2(s)$ vanish.

The effects on the sizes of these matrices are observed in table 6.3, and are listed briefly:

i) As $S_i(s)$ -> I, $\|T_{YD}(s)\|_{max}$ -> 1, meaning that disturbances tend to be transmitted straight to the outputs. This is an undesirable effect.

ii) As $W(s)$ -> 0, $\|T_{UD}(s)\|_{max}$ -> 0 and $\|T_{UN}(s)\|_{max}$ -> 0, meaning that the inputs become unaffected by noise or disturbances. This is a desirable effect.

iii) As $T_2(s)$ -> 0, $\|T_{YN}(s)\|_{max}$ -> 0, meaning that the outputs become unaffected by noise. This is also a desirable effect.

iv) As $S_2(s)$ -> I, $\|SE_Y(s)\|_{max}$ -> 1, meaning that the outputs become affected by changes in the process model. This is an undesirable effect.

v) As $T_1(s)$ -> 0, $\|SE_U(s)\|_{max}$ -> 0, meaning that the inputs become insensitive to changes in the process model. This is a desirable effect.

vi) $T_1(s)$ -> 0 causes the stability margins to tend to their maximum values, as indicated by the large gain and phase margins evaluated for the system.

6.6) Synthesis, Implementation and Analysis of Trial System 3

6.6.1) LQG Synthesis

The inputs in system 2 are limited to the same degree as the outputs. The low gains in the process model, pointed out in section 6.2.6, make very large excursions in the inputs necessary to realize significant compensating changes in the outputs. Limiting of the outputs as opposed to the inputs allows more swing in the inputs to compensate for the effects of process disturbances and changes in the process model. No obvious approach however exists for the LQG synthesis of systems for disturbance rejection and output insensitivity.

The desired improvement in these feedback properties require large gains in **F(s)**, which will occur at the expense of the good properties discussed for system 2. Effects of noise and disturbances on the inputs will have to be tolerated.

The output weighting matrix Q'_1, after testing and analysing a number of matrix structures, is chosen bearing in mind the preceding discussion. Q'_1 for system 3 is shown in equation (6.24). The matrices $R_1 = I$ and $R = I$ used in system 2 are again adopted. The matrix $Q = 0.1*I$ is adopted, specifying noise processes on the differentials of all of the states.

$$Q'_1 = \begin{bmatrix} 50000.0 & 25000.0 \\ 25000.0 & 50000.0 \end{bmatrix} \qquad (6.24)$$

A LQG controller based on these matrices is synthesized to produce the gain matrices quoted in (6.25) and (6.26).

$$K_c = \begin{bmatrix} -4.86E+01 & 1.07E-01 & -6.71E-01 & -1.25E+02 \\ 2.09E+02 & 1.25E-01 & -4.33E-01 & 9.48E+01 \end{bmatrix} \quad (6.25)$$

$$K_f = \begin{bmatrix} 3.15E-01 & .00E+00 \\ 2.53E-03 & .00E+00 \\ .00E+00 & 2.99E-02 \\ .00E+00 & 3.14E-01 \end{bmatrix} \quad (6.26)$$

The state feedback system poles are located at s = -0.4095, s = -0.0107, s = -0.000985 ± j0.000460. The large swing allowed in the inputs relative to that in the outputs has caused large elements to occur in K_c. These elements are responsible for the fast C/L pole (> 100 times as fast as fastest O/L pole). This system pole could cause unmodelled or unknown faster modes in the process to be excited.

The Kalman filter system pole positions are evaluated to be at s = -0.00108, s = -0.00268, s = -0.32 and s = -0.32. All of these poles are stable, and the same repeated pole observed in system 2 at s = -0.32 is present.

6.6.2) Performance Index Analysis

Figures 6.16 and 6.17 show respectively a group plot of data created for this trial system, and the analysis of this data.

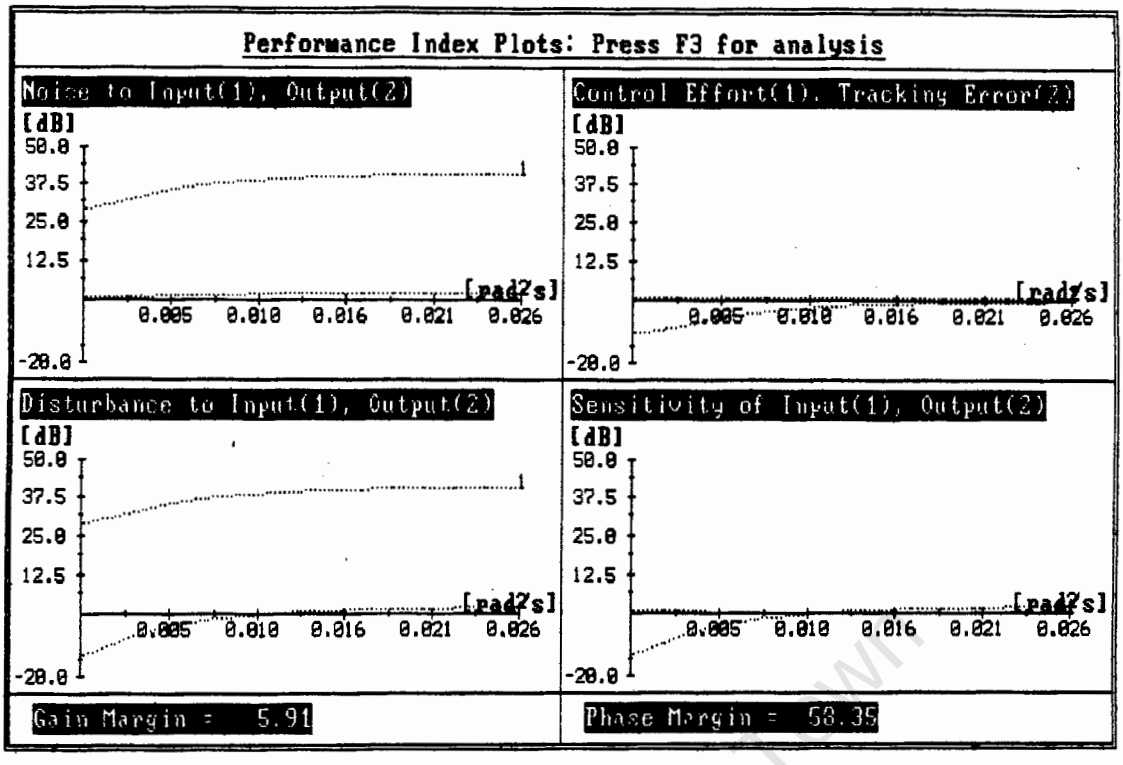
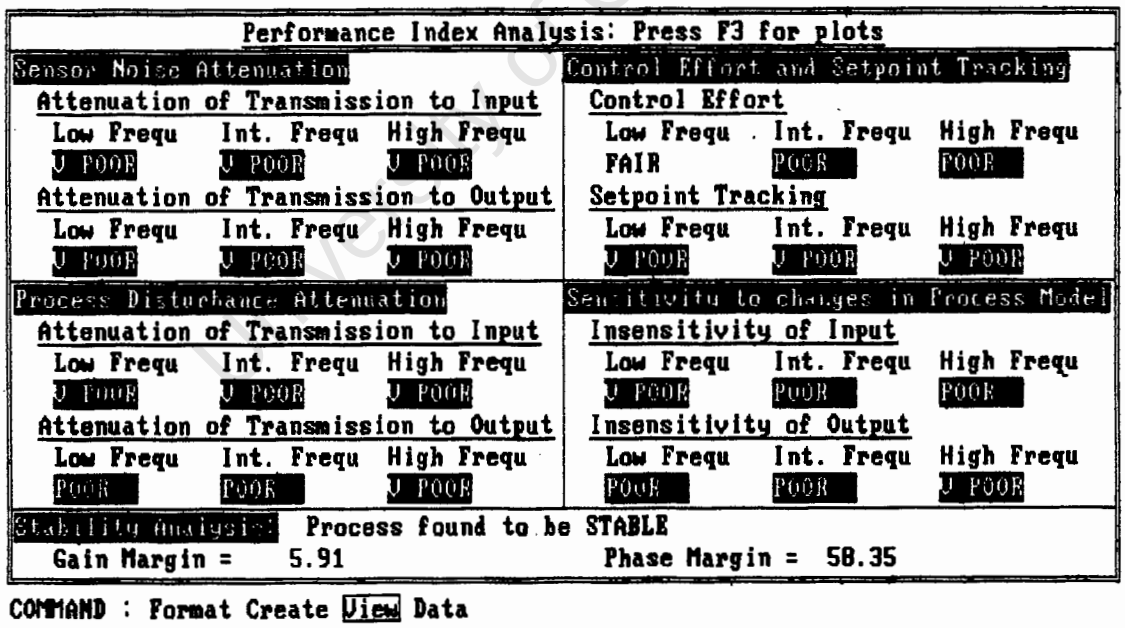


Figure 6.16: System 3 - Performance Index Group Plot



COMMAND : Format Create View Data

BRIEF : View Performance Index Data and Analysis.

Figure 6.17: System 3 - Comments on Performance Index Group Plot

6.6.3) Verification of Performance Index Data

Appendix R contains individual plots of each of the four performance index graphs shown in figure 6.16, as required for the verification procedure described in section 6.4.3.

(i) Transmission of Noise, Disturbances and Setpoints

Simulation response plots for the verification procedure are shown in Appendix R, and table 6.4 summarizes the predicted and simulated results obtained. No process response data is quoted in table 6.4 due to instabilities in the actual process implementation discussed under (iii).

The percentage errors, as defined in section 6.5.3, occurring between the predicted Mp and tested Mt response ratio maxima are presented in table 6.4.

The negative errors obtained in all of the results, except in those for the verification of $\|T_{YN}(s)\|_{max}$, indicates that the simulated maxima are less than the predicted maxima. The performance indices have thus correctly given a quantitative prediction for the maxima of these feedback properties. Errors smaller than -25% obtained in some cases indicate that the response maxima predicted, although correct, were rather conservative.

		Frequency Band		
		Low Band	Int Band	High Band
$\|T_{UN}(s)\|_{max} = \|W(s)\|_{max}$	(dB)	35.0	40.0	41.5
MR_{max}/A_{pert} Simulated	(dB)	32.77	37.50	38.79
% Error	(%)	-22.64	-25.01	-26.80
$\|T_{YN}(s)\|_{max} = \|T_2(s)\|_{max}$	(dB)	0.9	1.8	1.8
MR_{max}/A_{pert} Simulated	(dB)	2.28	4.08	4.56
% Error	(%)	17.22	30.02	37.40
$\|T_{UD}(s)\|_{max} = \|W(s)\|_{max}$	(dB)	35.0	40.0	41.5
MR_{max}/A_{pert} Simulated	(dB)	32.77	37.50	38.79
% Error	(%)	-22.64	-25.01	-26.80
$\|T_{YD}(s)\|_{max} = \|S_2(s)\|_{max}$	(dB)	-4.2	1.1	2.1
MR_{max}/A_{pert} Simulated	(dB)	-6.17	-0.72	0.82
% Error	(%)	-20.29	-18.90	-13.70
$\|T_{UR}(s)\|_{max}$ Predicted	(dB)	-6.0	-1.1	-0.2
MR_{max}/A_{pert} Simulated	(dB)	-6.38	-1.94	-0.91
% Error	(%)	-4.28	-9.22	-7.74
$\|ERR(s)\|_{max}$ Predicted	(dB)	0.0	0.0	0.0
MR_{max}/A_{pert} Simulated	(dB)	0.00	0.00	0.00
% Error	(%)	0.00	0.00	0.00
$\|SE_U(s)\|_{max} = \|T_1(s)\|_{max}$	(dB)	0.2	0.0	0.0
$\|SE_Y(s)\|_{max} = \|S_2(s)\|_{max}$	(dB)	-4.2	1.1	2.1

Table 6.4: Predicted and Simulated Maximum Response Ratios (Obtained from tables R.1 to R.6)

Inaccuracies in determining $T_2(s)$ are the most likely cause of the large errors in $\|T_{YN}(s)\|_{max}$.

(ii) Sensitivity to Changes in Process Model

The plots in figure 6.16 suggest that the inputs will be sensitive to changes in the process model in all bands. They also suggest that the outputs will be insensitive to low frequency variations in the process model, but that this sensitivity will increase as the frequency of the outputs increase.

(iii) Stability and Stability Margins

Figure 6.18 shows the simulated C/L responses to the stepping of the temperature setpoints. Setpoint T1 is stepped by +10 units at t = 100 sec, while T5 is stepped by +10 units at t = 1100 sec.

The simulated responses are C/L stable, and the zero output overshoot verifies the predicted 58° phase margin. The simulated response time is approximately 100 sec for both outputs T1 and T5.

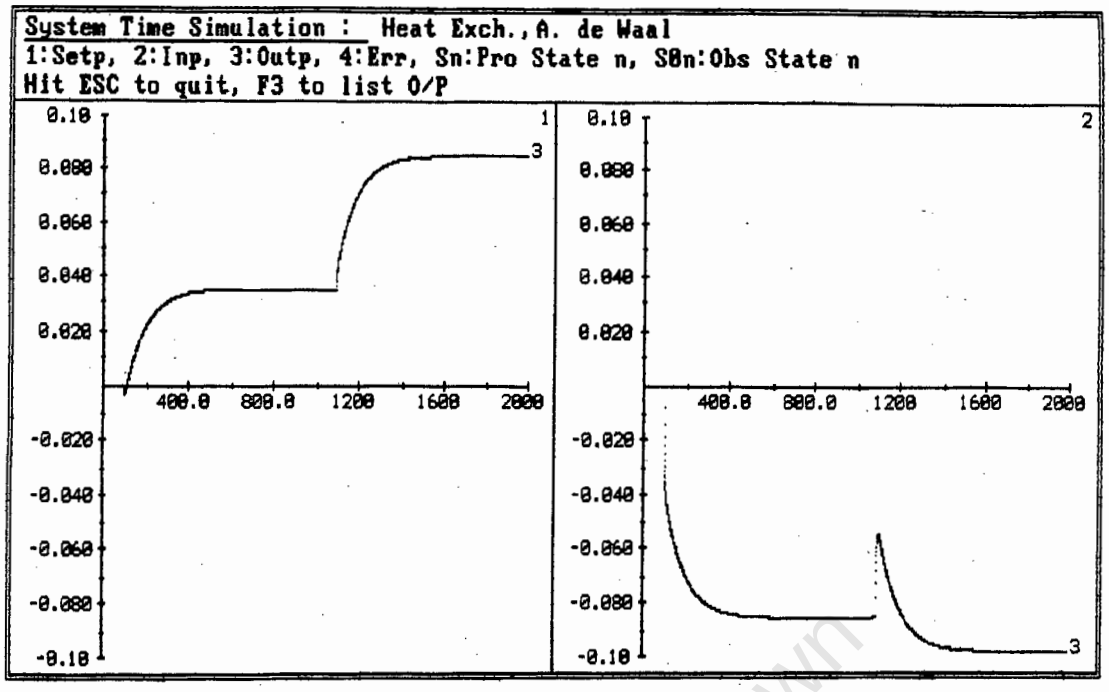


Figure 6.18: System 3 - Simulated C/L Response of Inputs and Outputs

Instabilities observed in figure 6.19 were encountered with the attempted implementation of the design on the process itself. The system is driven to oscillate at a w_{osc} = 0.026 rad/sec. The most likely reasons for the instabilities encountered are:

a) The possibility of an unmodelled system pole in the vicinity of s = -0.031 rad/sec, close to w_{osc}, is discussed in section 6.2.2. In its attempt to achieve the fast response indicated by the eigenvalue -0.4095, the control system could excite this pole, driving the system into oscillation.

b) Dead-times in order of 8.3 to 66.7 seconds could cause oscillation at frequencies between 0.757 and 0.094 rad/sec if the system is driven sufficiently.

c) Reduction in the gain and phase margins means that smaller variations of the process model could destabilize the system.

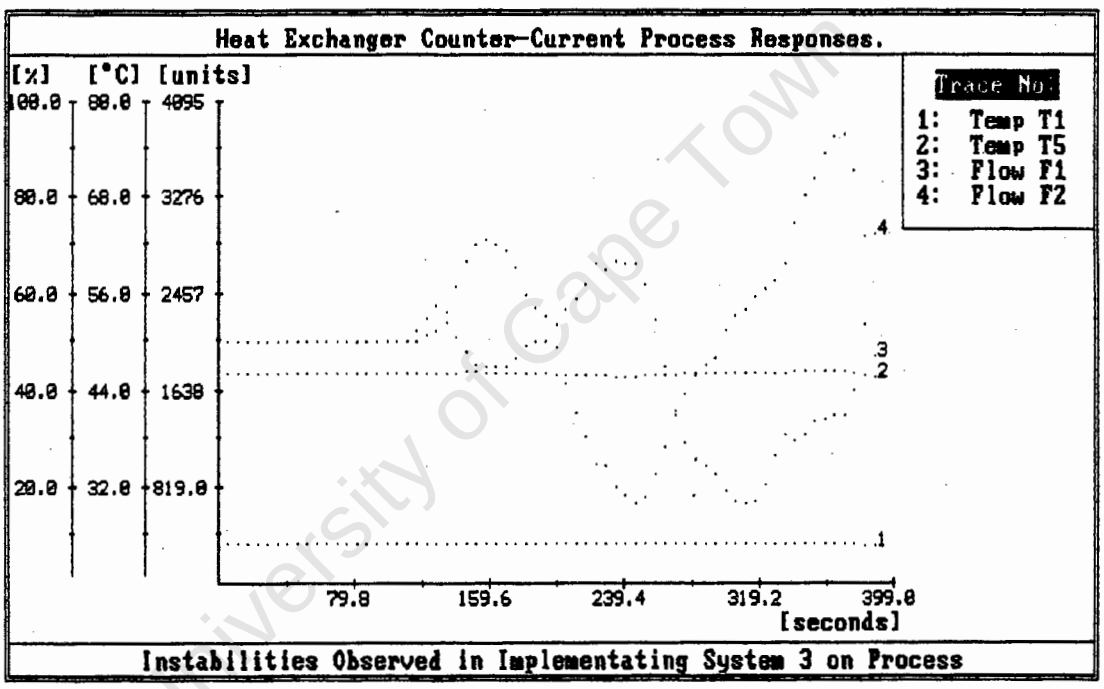


Figure 6.19: System 3 - Process Instabilities Encountered

Had the abovementioned effects been modelled, the LQG synthesis procedure could then have taken them into account. The stability performance index analysis would also pick up any such effects. The effects discussed are however not modelled in the state-space description

of the process, so it is impossible for the performance index analysis procedure to detect them.

(iv) Control Effort and Setpoint Tracking

A constant precompensator matrix P_C for steady-state tracking is determined and given in (6.27), and the resultant control effort and setpoint tracking performance index plot with P_C is given in figure 6.20.

$$P_C = \begin{bmatrix} 2.41E+01 & -9.83E+00 \\ 2.04E+01 & -6.38E+00 \end{bmatrix} \qquad (6.27)$$

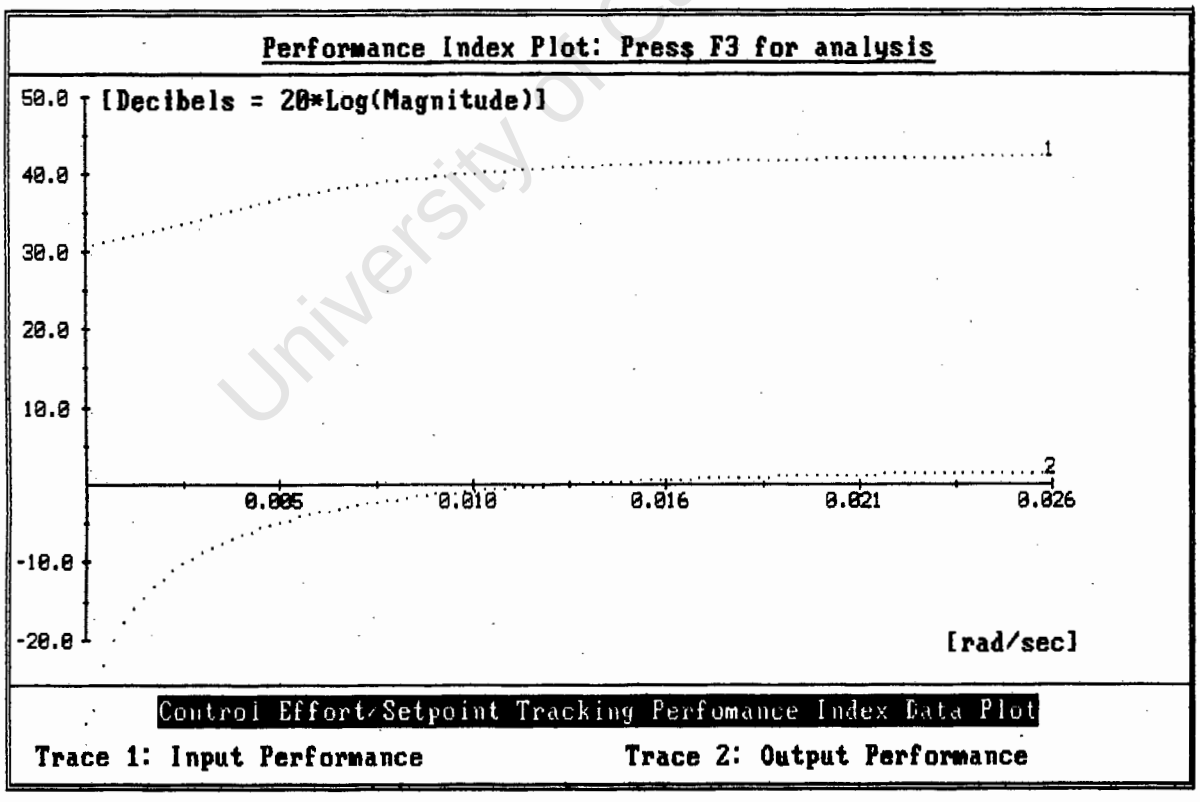


Figure 6.20: System 3 - Setpoint Tracking / Control Effort Performance Index Plot (Including P_C)

The performance index plot in figure 6.20 once again predicts zero (less than -30dB) steady-state tracking error, with the inevitable trade-off of large control effort (more than 30dB amplification of setpoint signals to inputs). A slower increase in the tracking error index with frequency indicates that the output will track higher frequency setpoint excursions than for system 2. The tracking index is greater than 0dB for all frequencies above 0.012 rad/sec, meaning that the maximum time required for errors to decay will be in the order of 500 seconds.

The predictions are verified in figures 6.21 and 6.22 showing the simulated C/L response of the system, including precompensator P_C, to a +10 unit step in setpoint T1 at t = 100 sec, and a +10 unit step in setpoint T5 at t = 1100 sec.

The outputs track their respective setpoints at steady-state. The errors in output T1 have decayed entirely after 600 seconds, verifying the predicted decrease in the maximum error decay time from that for system 2. Very large changes are predicted and seen in the flow inputs to realize the required changes in the temperature outputs (28dB maximum transmission of setpoint to input F1 in figure 6.21).

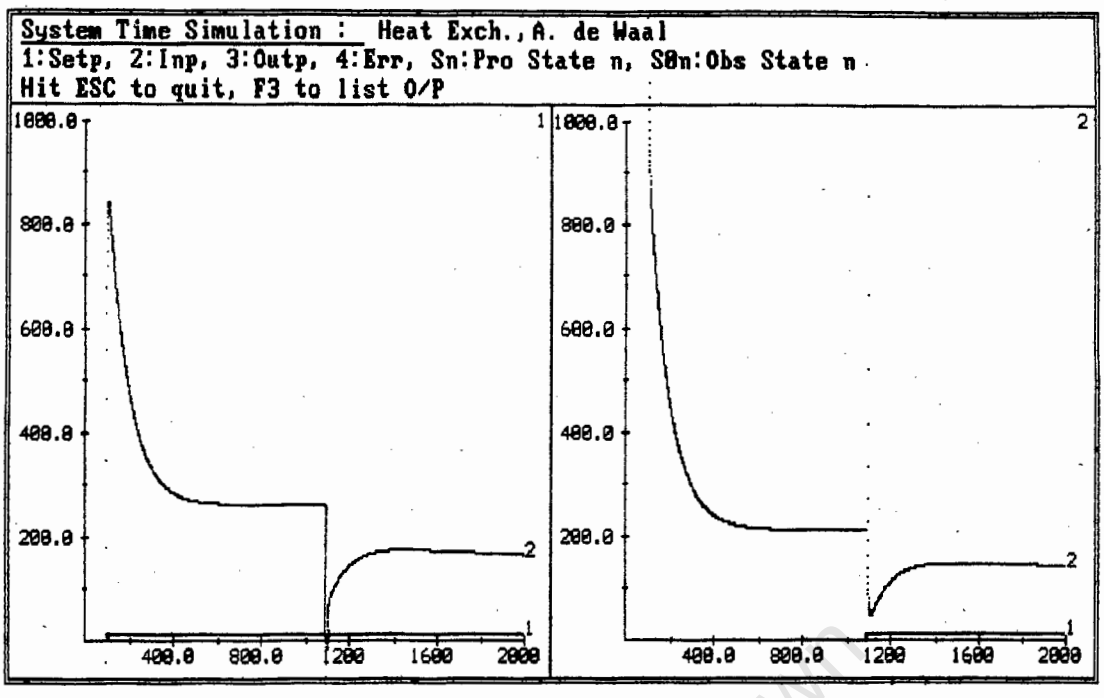


Figure 6.21: System 2 - Simulated C/L Responses of Inputs (Including P_c)

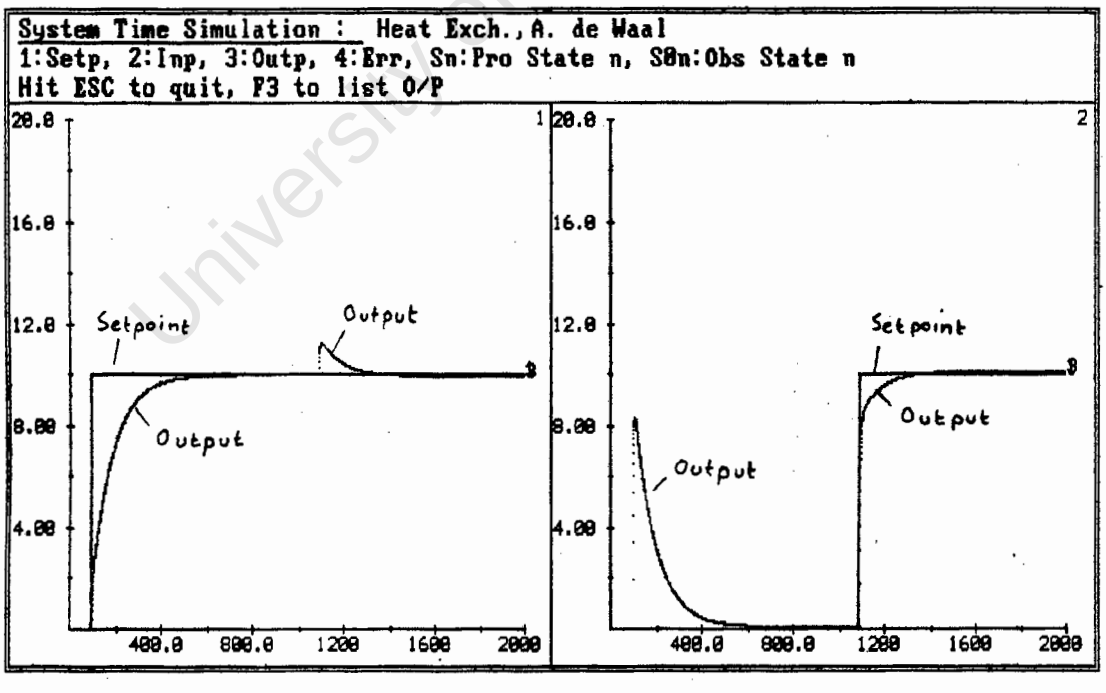


Figure 6.22: System 2 - Simulated C/L Responses of Outputs (Including P_c)

It is interesting to note that input F1 responds to a greater extent than does F2. The performance index analysis has picked up the maximum response, but has no means of predicting in which loop this response occurs. Interaction is once again observed between the loops.

6.6.4) Comments on Design

The large elements in K_c have effectively increased the gain of F(s) at low frequencies. Equations 4.5 and 4.5 show that this will cause $S_2(s)$ to become smaller at low frequencies, improving the output disturbance rejection and making the system insensitive to changes in the process model at low frequencies. Table 6.4 shows that this has occurred, but at the expense of the W(s) (predicted by equations (4.4) and (4.11)) becoming very large over all frequencies, the inputs becoming very sensitive to noise, disturbances and changes in the process model. The design trade-off between input and output sensitivities is demonstrated in the above discussion.

The speed of response has also been improved by this high-gain controller, as seen in the response plots in figures 6.18 and 6.22.

6.7) Concluding Remarks

The performance index analysis of LQG control systems and the implementation and testing of the designs on a digital simulator and on the actual process has led to the following conclusions regarding performance index analysis techniques:

(i) Powerful Control System Analysis Tool

The good correspondence observed between the predicted, simulated and process response maxima indicates that this technique provides a powerful tool for the quantitative analysis of control systems in terms of these feedback properties.

The slightly larger errors observed in the process output response maxima than in the simulated response maxima for system 2 were predicted by the output sensitivity performance index. This index indicated that changes in the process model (expected from large uncertainties in the model) would affect the outputs directly. These changes would thus also affect the output responses to perturbations. The input and output sensitivity indices can thus also be considered as built in indicators predicting errors that can be expected in the perturbation response predictions.

An exception to the good results generated for systems 2 and 3 was that of predicting output response maxima to sensor noise perturbations. A likely source of these errors is in the complex numerical computations required to obtain a value for the index in the OPTCAD package.

A reliable quantitative representation for the stability margins of a control system is also obtained by the performance index analysis. These margins are related to the classical gain and phase margin definitions.

(ii) Conservative Estimates of Feedback Properties

The large negative errors obtained in the estimation of some of the response maxima for system 3 means that the simulated response maxima fell well within their predicted limits. The prediction of the maxima in these cases were thus rather conservative.

(iv) Inability to Identify Source of Response Maxima

Although the indices predict the response maxima accurately, these maxima could occur in any one of the loops within the system, while the responses in the remaining loops could be relatively small. This phenomenon is observed in the larger response to changes in the setpoints of input F1 than of input F2. The

performance index analysis technique predicts large responses but has no means of identifying the specific loops in which such responses will occur.

(v) Inability to Predict Interaction Between Loops

Performance index analysis techniques have no means of predicting interaction which may occur between the loops in a control system.

(vi) Limits on Control System Feedback Properties

The effects of the fundamental limit imposed on the feedback properties of control systems by the relation $S_i(s) + T_i(s) = I$ are observed performance indices generated for systems 2 and 3. Attenuation of the same perturbations cannot be achieved for the inputs and outputs simultaneously. The performance of the inputs has to be traded off against that of the outputs.

CHAPTER 7

CONCLUSIONS

7.1) The Heat Exchanger Counter-current Process

The steady-state temperature distribution across the heat exchanger process behaved as designed. Errors in the observed log mean temperature difference across each exchange vessel were less than 5% in all cases (chapter 1).

The following analogies exist between the heat exchanger and CIP processes (chapter 1):

i) Heat exchanger CH stream is analogous to the carbon stream in the CIP process.

ii) Heat exchanger HC stream is analogous to the pulp stream in the CIP process.

iii) Longer residence time of CH stream is analogous to longer residence time in carbon stream.

iv) Decreasing temperature gradient across heat exchange vessels is analogous to decreasing concentration gradient across CIP reaction vessels when moving in the direction of any of the counter current streams.

v) Mixed states in the exchange vessels are analogous to the mixed states in the CIP reaction vessels.

A notable difference is that the CIP process is at present run as a batch process whereas the heat exchanger process is normally run as a continuous process. Future research on the heat exchanger could involve a comparison of the respective efficiencies obtained when running under batch and continuous modes of operation. This could assist in assessing whether the current batch mode of operation of the CIP process is indeed the most efficient one.

7.2) Control of the Heat Exchanger Flows and Levels

The diagonal PI cascade compensator $K_c(s)$ (chapter 3), designed and implemented to control the upper left subsystem $G_c(s)$ of the process model G(s)(chapter 2) , succeeded in:

i) Regulating the flow and level outputs to their respective setpoints with maximum steady-state errors in the flow and level outputs of less than 1% and 5% (of full scale) respectively

ii) Improving the speed of the level responses from between 311 and 575 sec under open loop conditions to between 11.5 and 19.7 sec under closed loop conditions.

iii) Eliminating steady-state interaction between the control loops, and reducing transient interaction such that the maximum transient gain observed between setpoints and outputs of other loops was found to be less than 0.7.

The price paid for the good output performance was a slight reduction in the stability of the system (level outputs overshooting by between 0.5 and 16.2%) and large excursions (control effort) in the control valve inputs.

7.3) Analysis of Control Systems using Performance Indices

The performance indices developed in chapter 4 provide a powerful tool with which the behavior of a multivariable control system can be predicted quantitatively.

The maximum singular values of the characteristic matrices **W(s)**, $S_i(s)$ and $T_i(s)$, which are functions of the process and controller system matrices **G(s)**, **K(s)** and **F(s)**, provide accurate and quantitative frequency-dependent indices by which the maximum input and output responses to sensor noise and process disturbances can be gauged, as well as by which the sensitivity of the process inputs and outputs to changes in the process model can be predicted. Additional indices, based on these characteristic matrices and on the process precompensator matrix **P(s)**, make it possible to quantitatively gauge the response of the inputs and outputs

to setpoint excursions. The sensitivity indices also provide a means of estimating the errors which may occur in the predictions due to changes in the process model (response prediction indices based on the nominal process model).

The system stability margins are related to the minimum singular value of the matrix $T_1^{-1}(jw)$ over all frequencies by the relation $gm = \inf_w (20\log_{10}(1 + \sigma_{min}(T_1^{-1}(jw))))$ dB

and relation $pm = 2 \inf_w \arcsin(\frac{1}{2} \sigma_{min}(T_1^{-1}(jw)))$.

These margins are multivariable generalizations of the classical single-variable stability margins.

LQG optimal control systems can be analysed using these performance indices by relating the matrices **G(s)**, **F(s)** and **P(s)** to the state feedback system matrices **A**, **B**, **C**, K_c and K_f using relationships developed in chapter 4.

The performance indices defined for MIMO control systems reduce to the classical measures of performance defined for SISO systems, in the special 1x1 case of MIMO systems.

The indices, although generally accurate in predicting the response maxima, sometimes give a rather conservative estimate of these maxima. This is seen in the conservative predictions for system 3 (section 6.6), reflected in the large negative prediction errors obtained.

The response indices, although able to identify the maximum possible response that will be observed in all of the loops of a control system, are unable to identify in which particular loops such responses will occur. In layman's terms, it tells us that there is a problem, but not where the problem is.

The indices are also unable to predict interaction that might occur. Large magnitudes in the tracking error index could be due to interaction between the loops, or due to the inability of a particular output to track its setpoint.

Designing multivariable controllers using these indices essentially involves trading off the sizes of the matrices $S_i(s)$, $T_i(s)$, and $W(s)$ against each other over different frequency bands. Good performance of in terms of one property is traded off against another in each frequency band. The performance of a system is limited by the relation $S_i(s) + T_i(s) = I$, which implies that $S_i(s)$ and $T_i(s)$ cannot be made to vanish at the same frequencies.

7.4) OPTCAD Controller Synthesis and Analysis Package

This package developed provides a user-friendly tool to enable the performing of the following functions:

i) Synthesis of LQG control systems.

ii) Frequency-domain performance index analysis of state feedback control systems with state observers (LQG system is a special case of this group of systems).

iii) Digital simulation of state feedback control systems with observers.

7.5) HERIG Package Developed for the Running, Control and Analysis of the Heat Exchanger Process

This package provides a simple and logical interface between the user and the heat exchanger process. Manipulation of the processs inputs and observation of the outputs is simplified by the meaningful interface screens. Process response data is easily logged, saved, loaded and plotted for analysis using this package. Control and other parameters are also be manipulated.

7.6) Implementation and Performance Index Analysis of LQG Control Systems for the Heat Exchanger Process

Trial control systems 2 and 3 synthesized for the heat exchanger process were analysed using performance indices.

The low gains in the process model, and the limiting of the inputs for system 2 caused the loop gain for the system to

become relatively small. This was reflected in the performance index analysis (figures 6.3 and 6.4, table 6.3) of the system, which showed that disturbances would be transmitted straight to the outputs, and also that the outputs would be sensitive to changes in the process model. The indices also revealed that sensor noise and disturbances would not be transmitted to the inputs, and that the inputs would be insensitive to changes in the process model. The low loop gains resulted in good stability margins being predicted for the system.

The implementation of system 2 on a digital simulator and on the process itself revealed good correspondence between the predicted and observed performance in all but one of the indices considered. Inaccuracies in this index, gauging transmission of sensor noise to outputs, are attributed to numerical errors occurring in the calculation of the index in the OPTCAD package.

The output sensitivity index correctly predicted the observed discrepancies between the simulated and process output responses, as well as errors in the output response indices, which are based on the nominal model for the process. These discrepancies are to be expected when considering the relatively large uncertainties in the process model discussed in section 6.2.2.

Inclusion of a precompensator enabled the outputs to track their setpoints in the simulated responses (figure 6.10).

The output sensitivity index, however, predicted the tracking errors observed in the process responses (figures 6.12 and 6.15).

The inputs were limited far less than the outputs for trial system 3, in an attempt to increase the loop gain sufficiently, so as to enable good rejection of disturbances by the outputs and output insensitivity to changes in the process model. No obvious approach is known for (LQG) synthesizing systems to reduce disturbance transmission to the outputs and output sensitivity to changes in the model.

These efforts met with success at low frequencies, the performance indices predicting a reduction in both the disturbance transmission to outputs and in the output sensitivity (table 6.4). The implementation of the system on a digital simulator revealed that the indices were accurate, although sometimes conservative, in predicting the response maxima. Errors were again encountered in the index predicting the transmission of sensor noise to the outputs.

The lifting of the constraints on the inputs and the resultant increase in loop gain caused the response times for system 3 to be shortened to approximately 10% of those observed for system 2 (as observed in the step tests shown in sections 6.5.3 and 6.6.3). Inclusion of a precompensator again enabled the outputs to track their setpoints in the simulation responses (figure 6.22). The increase in the speed of response is predicted by the tracking error

performance index for system 3 (figure 6.20) remaining below 0dB for frequencies more than 10 times as high as those for the corresponding index for system 2 (figure 6.8).

Instabilities observed in the implementation of the system on the heat exchanger process (figure 6.19) are attributed to unmodelled dynamics (due to the unmodelled faster system poles, dead times, etc discussed in section 6.2.2) being excited by the fast high-gain control system. These effects could not have been detected by the stability performance index analysis procedure, as the causes of the instabilities were not modelled in the system description used for the synthesis and analysis procedures.

Larger responses to changes in the setpoints are observed for input F1 than for input F2 in the implementation of control systems 1 and 2 (sections 6.5.3 and 6.6.3). As concluded earlier, the performance index analysis technique predicts large responses but has no means of identifying the specific loops in which such responses will occur.

The effects of the fundamental limit imposed on the feedback properties of control systems by $S_i(s) + T_i(s) = I$ are observed in performance indices generated for systems 2 and 3. Attenuation of the same perturbations cannot be achieved for the inputs and outputs simultaneously. The performance of the inputs has to be traded off against that of the outputs.

REFERENCES

1. Borrie J.A., Modern Control Systems - A Manual of Design Methods, Prentice/Hall International, London, 1986.

2. Broussard J.R. A Quadratic Weight Selection Algorithm, IEEE Trans. Aut. Contr., 1982, AC-27, (4), pp. 945-947.

3. Craig I.K. Sensitivity of H^{∞} controller designs to structured uncertainty, M.Sc. dissertation, Department of Aeronautics and Astronautics, Massachusetts Institute of Technology, May 1989.

4. Doyle J.C., Stein G. Multivariable Feedback Design: Concepts for a Classical/Modern Synthesis, IEEE Trans. Aut. Contr., 1981, AC-26, (1), pp. 4-16.

5. Doyle J.C., Stein G. Robustness with Observers, IEEE Trans. Aut. Contr., 1978, AC-24 ,(4), pp. 607-611.

6. Doyle J.C. Guaranteed Margins for LOG Regulators, IEEE Trans. Aut. Contr., 1978, AC-23 ,(54), pp. 756-757.

7. Fisher I.P., Multivariable Control of a Flotation Plant Simulator, M.Sc. Dissertation, UCT, Cape Town, December 1988.

8. Gear A.B.J., The Design of Decentralized Controllers for Large Scale Systems, M.Sc. Dissertation, UCT, Cape Town, February 1988.

9. Gupta N.K. Frequency-Shaped Cost Functionals: Extension of Linear-Quadratic-Gaussian Design Methods, J. Guid. Contr., 1980, Vol 3, (6), pp. 529-535.

10. Johnson M.A. and Grimble M.J. Recent trends in linear optimal quadratic multivariable control system design, IEE proceedings, Vol.134, Pt.D, No.1, Jan 1987.

11. Kwakernaak F., Sivan R. Linear Optimal Control Systems, Wiley-Interscience, New York, 1972.

12. Raven F.H. Automatic Control Engineering, 4th edition, McGraw Hill, 1987, Singapore

13. Safonov M.G. et al. Feedback Properties of multivariable systems: The role and use of the Return Difference Matrix, IEEE Trans. Aut. Contr., 1981, AC-26, pp47-65.

14. Venzke R.H.E., Comparison of INA Technique to the Pole Assignment Technique, M.Sc. Dissertation, UCT, Cape Town, 1988.

15. Zames G., Bruce A.F. Feedback, Minimax Sensitivity, and Optimal Robustness, IEEE Trans. Aut. Contr., 1983, AC-28, (5), pp. 585-601.

16. Zames G. Feedback and Optimal Sensitivity: Modal reference transformations, multiplicative seminorms and approximate inverses, IEEE Trans. Aut. Contr., 1981, AC-26, (2), pp. 301-320.

BIBLIOGRAPHY

1. Anderson B.D.O., Moore J.B. Linear Optimal Control, Prentice-Hall International Editions, Englewood Cliffs, New Jersey, USA, 1971.

2. Anderson B.D.O. Stability Results for Optimal Systems, Electronics Letters, 1969, Vol 5, (22), p. 545.

3. Anton, H. Elementary Linear Algebra, 5th Edition, John Wiley & Sons, USA, 1987

4. Astrom K.J., Wittenmark B. Adaptive Control, Addison-Wesley Publishing Company, USA, 1989.

5. Astrom K.J., Wittenmark B. Computer Controlled Systems, Prentice Hall, 1984.

6. Athans M. The Role and Use of the Stochastic Linear-Quadratic-Gaussian Problem in Control System Design, IEEE Trans. Aut. Contr., 1971, AC-16, (6), pp. 529-552.

7. Borrie J.A., Modern Control Systems - A Manual of Design Methods, Prentice/Hall International, London, 1986.

8. Broussard J.R. A Quadratic Weight Selection Algorithm, IEEE Trans. Aut. Contr., 1982, AC-27, (4), pp. 945-947.

9. Clarke D.W. et al. A generalized LQG approach to self-tuning control Part I, Aspects of design, Int. J. Control, 1985, Vol 41, (6), pp. 1509-1523.

10. Clarke D.W. et al. A generalized LQG approach to self-tuning control Part II, Implementation and simulation, Int. J. Control, 1985, Vol 41, (6), pp. 1524-1544.

11. Craig I.K. Sensitivity of H^{∞} controller designs to structured uncertainty, M.Sc. dissertation, Department of Aeronautics and Astronautics, Massachusetts Institute of Technology, May 1989.

12. Doyle J.C., Stein G. Multivariable Feedback Design: Concepts for a Classical/Modern Synthesis, IEEE Trans. Aut. Contr., 1981, AC-26, (1), pp. 4-16.

13. Doyle J.C., Stein G. Robustness with Observers, IEEE Trans. Aut. Contr., 1978, AC-24 ,(4), pp. 607-611.

14. Doyle J.C. Guaranteed Margins for LQG Regulators, IEEE Trans. Aut. Contr., 1978, AC-23 ,(54), pp. 756-757.

15. Fisher I.P., Multivariable Control of a Flotation Plant Simulator, M.Sc. Dissertation, UCT, Cape Town, December 1988.

16. Gear A.B.J., The Design of Decentralized Controllers for Large Scale Systems, M.Sc. Dissertation, UCT, Cape Town, February 1988.

17. Grimble M.J. Design of Optimal Stochastic Regulating Systems Including Integral Action, IEE Proc Contr. & Science, 1979, Vol 126, (9), pp. 841-848,

18. Gupta N.K. Frequency-Shaped Cost Functionals: Extension of Linear-Quadratic-Gaussian Design Methods, J. Guid. Contr., 1980, Vol 3, (6), pp. 529-535.

19. Harvey C.A., Stein G Quadratic Weights for Asymptotic Regulator Properties, IEEE Trans. Aut. Contr., 1978, AC-23, (3), pp. 378-387

20. Horowitz I. Synthesis of Feedback Systems, New York: Academic, 1963.

21. Johnson M.A. and Grimble M.J. Recent trends in linear optimal quadratic multivariable control system design, IEE proceedings, Vol.134, Pt.D, No.1, Jan 1987.

22. Kern D.Q. Process Heat Transfer, International Student Edition, 1950.

23. Kwakernaak F., Sivan R. Linear Optimal Control Systems, Wiley-Interscience, New York, 1972.

24. Lehtomake N. et al. Robustness Results in the Linear-Quadratic Gaussian Based Multivariable Control Designs, IEEE Trans. Aut. Contr., 1981, AC-26, (1), pp. 75-92.

25. Lord R.C., Minton P.E., Slusser R.P. Design of Heat Exchangers, Chemical Engineering, 26 January 1970.

26. Macfarlane A.G.J. An Expert System Approach to Computer-Aided Design of Multivariable Systems, Springer-Verlag, 1987.

27. Macfarlane G.J. Return-difference and Return-ratio Matrices and their Use in Analysis and Design of Multivariable Feedback Control Systems, Proc. IEE, 1970, Vol 117, (10), pp. 2037-2049.

28. Martin W.L. Handbook of Industrial Piping, Sir Isaac Pitman & Sons, Ltd, London, 1961.

29. Perry R.H., Green D. Perry's Chemical Engineer's Handbook, 50th Anniversary Edition - International Student Edition, 1985.

30. Porter B. Optimal Control of Multivariable Linear Systems Incorporating Integral Feedback, Electronics Letters, 1971, Vol 7, (8), pp. 170-172.

31. Raven F.H. Automatic Control Engineering, 4th edition, McGraw Hill, 1987, Singapore

32. Rosenbrock H.H., McMorran P.D. Good, Bad or Optimal?, IEEE Trans. Aut. Contr., 1971, AC-16, (6), pp. 552-554

33. Safonov M.G. et al. Feedback Properties of multivariable systems: The role and use of the Return Difference Matrix, IEEE Trans. Aut. Contr., 1981, AC-26, pp47-65.

34. Venzke R.H.E., Comparison of INA Technique to the Pole Assignment Technique, M.Sc. Dissertation, UCT, Cape Town, 1988.

35. Youla D.C. et al. Modern Wiener-Hopf design of optimal controller - Part II: The Multivariable case, IEEE Trans. Aut. Contr., 1976, AC-21, (3), pp. 319-338

36. Zames G., Bruce A.F. Feedback, Minimax Sensitivity, and Optimal Robustness, IEEE Trans. Aut. Contr., 1983, AC-28, (5), pp. 585-601.

37. Zames G. Feedback and Optimal Sensitivity: Modal reference transformations, multiplicative seminorms and approximate inverses, IEEE Trans. Aut. Contr., 1981, AC-26, (2), pp. 301-320.

APPENDIX A

THERMAL CALCULATIONS FOR RESERVOIRS

A.1) Calculations for Hot Reservoirs

Hot reservoir system required to heat water from 50°C to 80°C at a rate of 5 l/min. Split heating into two stages, namely the pre-heating hot reservoir (PHR) and the controlled hot reservoir (CHR). PHR heats water from 50°C and controls its temperature to 75°C, and the CHR heats it further and controls its temperature to 80°C.

A.1.1) Pre-heating Hot Reservoir

	Outgoing Stream Enthalpy:	H(75°C) = 313.9 kJ/kg
Less	Incoming Stream Enthalpy:	H(50°C) = 209.3 kJ/kg
Equals	Change in Enthalpy :	δH = 104.6 kJ/kg

So, Maximum Rate of Heating Energy Required,

$$Q = \delta H * \text{Maximum Mass Flowrate}$$
$$= 104.6 \text{ kJ/kg} * 5 \text{ l/min} * 1/60 \text{ min/sec} * 1 \text{ kg/l}$$
$$= 8.72 \text{ kJ/sec}$$
$$= 8.72 \text{ kW}$$

Specify 3 X 3 kW (9 kW maximum) heating elements for PHR with thermostat temperature control

A.1.2) Controlled Hot Reservoir

	Outgoing Stream Enthalpy:	H(80°C) = 334.9 kJ/kg
Less	Incoming Stream Enthalpy:	H(75°C) = 313.9 kJ/kg
Equals	Change in Enthalpy :	δH = 21.0 kJ/kg

So, Maximum Rate of Heating Energy Required,

$Q = \delta H$ * Maximum Mass Flowrate
= 21.0 kJ/kg * 5 l/min * 1/60 min/sec * 1 kg/l
= 1.75 kJ/sec
= 1.75 kW

Specify 3 X 1.5 kW (4.5 kW maximum) heating elements for PHR with sensitive RTD temperature control system. Deliberately overspecify in case extreme conditions cause PHR outlet temperature to fall below 75°C

A.2) Calculations for Cold Reservoir

Controlled cold reservoir (CCR) required to heat water from 20°C to 24°C at a rate of 10 l/min.

	Outgoing Stream Enthalpy:	H(24°C) = 100.6 kJ/kg
Less	Incoming Stream Enthalpy:	H(50°C) = 83.9 kJ/kg
Equals	Change in Enthalpy :	δH = 16.7 kJ/kg

So, Maximum Rate of Heating Energy Required,

$$Q = \delta H * \text{Maximum Mass Flowrate}$$
$$= 16.7 \text{ kJ/kg} * 10 \text{ l/min} * 1/60 \text{ min/sec} * 1 \text{ kg/l}$$
$$= 2.79 \text{ kJ/sec}$$
$$= 2.79 \text{ kW}$$

Specify 3 X 1.5 kW (4.5 kW maximum) heating elements for PHR with sensitive RTD temperature control system. Overspecify deliberately in case temperature of water mains drops.

A.3) Photograph of Reservoirs

Photograph PA.1 has been taken of the heat exchanger reservoirs, and shows the CHR and PHR in the foreground, with the open CCR in the background.

Photograph PA.1: Reservoirs of Heat Exchanger Process

APPENDIX B

PHYSICAL LAYOUT AND PIPING OF HEAT EXCHANGER

B.1) Physical Layout of Heat Exchanger

Figure B1 on the following page gives a detailed schematic of the physical layout of the heat exchanger. The head required by the outlet piping of the tanks to realize the designed flowrates for the process are calculated and discussed in section B.2 of this appendix.

B.2) Piping and Head Calculations

The flow velocity v of a liquid flowing at volumetric flowrate Q in a pipe of diameter D is given by the expression:

$$v = (4\,Q) / (\pi\,D^2) \qquad (B1)$$

Once the velocity v has been determined, the Reynold's number indicating the degree of turbulence in the pipe can be obtained from the expression:

$$N_{Re} = (v\,D\,p) / \mu \qquad (B2)$$

where:

p = density of liquid

μ = viscosity of liquid

The friction factor f for the liquid flowing in a particular type of piping can then be determined from existing tables. Once this has been determined, the head h drive the liquid through the piping under the force of gravity at this flowrate can be determined from the expression:

$$h = (2 f l_t v^2) / (D g) \qquad (B3)$$

where:

l_t = total lumped length of pipe

g = acceleration due to the force of gravity

Any effects due to flowmeters, valves, bends and other constrictions in the piping are considered as equivalent lengths of pipe, l_e. These equivalent lengths of pipe are added to the actual length of piping l_p to make up the total lumped length of pipe l_t. This gives the following expression for the total lumped length of pipe:

$$l_t = (l_{e1} + l_{e2} + \ldots) + l_p \qquad (B4)$$

The cold-to-hot (CH) stream flows downwards under gravity feed from tank T4 to tank T1. The outflow configuration for each tank is identical, meaning that the calculation of flow from each tank is also identical. Flow calculations are thus done on a "per tank" basis.

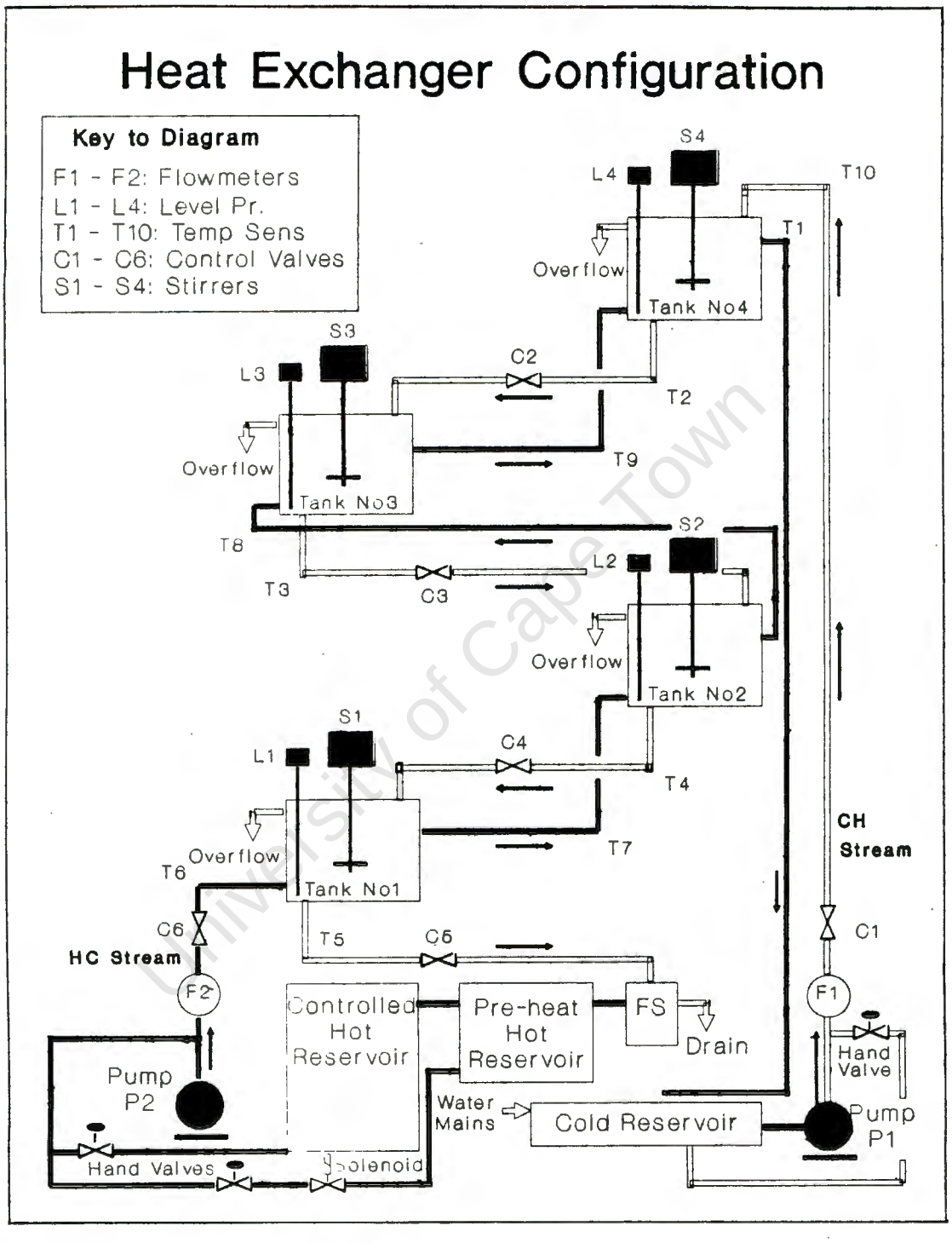


Figure B1: Layout of Heat Exchanger Process

Provision is to be made for two elbow bends (l_{eb}), a flowmeter (l_{ef}) and a control valve (l_{ev}) constricting the 0.6 m (l_p) length of polyethylene piping leading from the base of each tank. The maximum flowrate of 10 l/min must be achievable from each tank at all times (ie even when it is almost empty). Table B1 summarizes the results from calculations done to determine the head required to realize this flowrate for piping with two different inner diameter ratings.

Diam (in) D	Velocity (m/s) v	Reyn N_{Re}	Equivalent Length (m) l_{ev}	l_{eb}	l_{ef}	Tot Len (m) l_t	Head (cm) h
3/4	0.351	6687	3.23	0.57	3.81	8.22	9.8
7/8	0.258	5734	3.78	0.67	4.45	9.49	5.2

Table B1: Calculation of Head Required for Different Pipe Diameters

Figure B1 shows that even when the tanks are empty, the head in the outflow pipe is 20 cm, which is sufficient to sustain the maximum flowrate for both diameters of piping considered. The 3/4 inch piping is chosen as it matches the control valve and flowmeter inlet diameters more closely than the wider 7/8 inch piping.

APPENDIX C

HEATING COIL DESIGN TO REALIZE TEMPERATURE DISTRIBUTION

Figure C1 shows a block diagram of the heat exchanger system, while figure C2 gives a schematic of one of the exchange vessels found on the process. The submerged heating coil in each exchange vessel needs to be designed to achieve the temperature profile specified across the vessel. The coil characteristics will determine the thermal resistance that exists between the two streams.

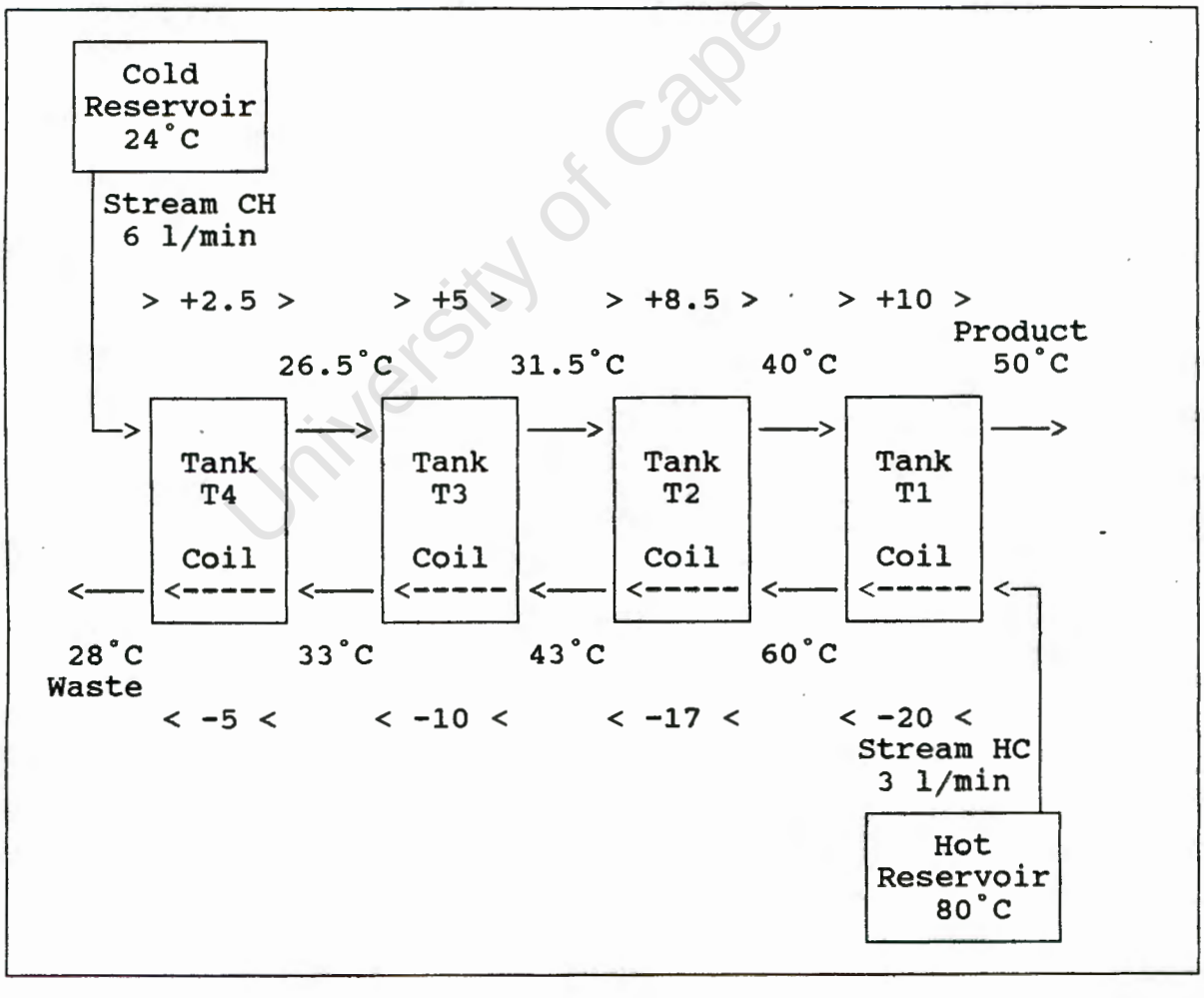


Figure C1: Block Diagram of Heat Exchanger

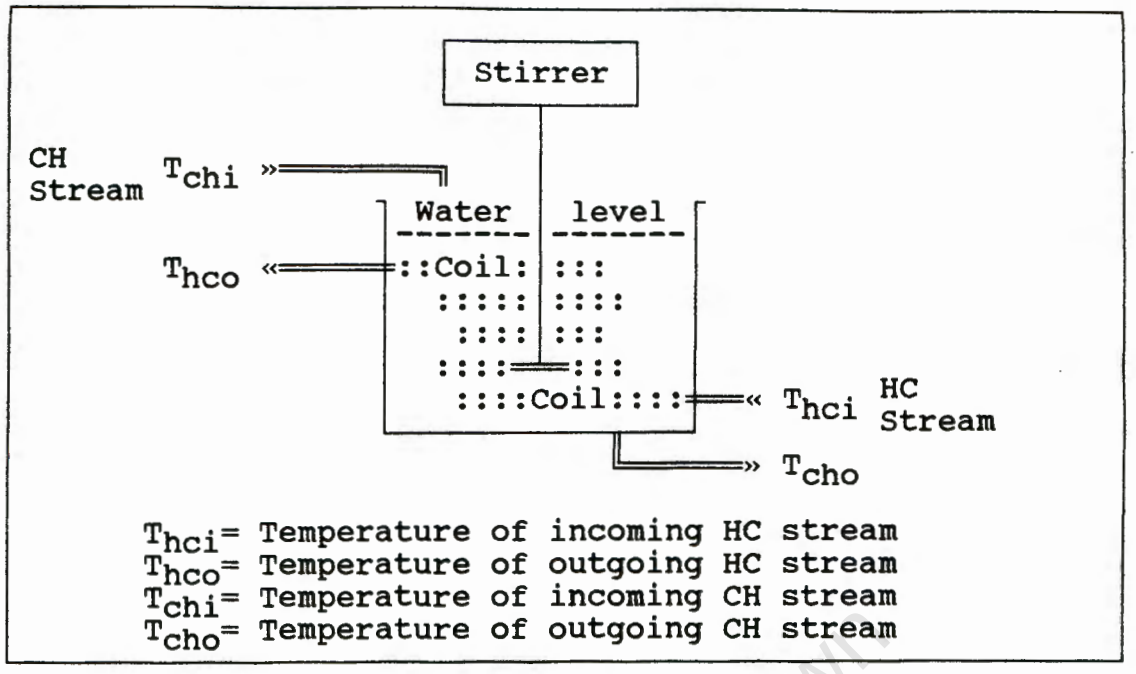


Figure C2: Typical Heat Exchange Vessel

The rate of heat transferred Q from the HC stream to the CH stream is given by the expression:

$$Q = U_o A_o \delta Tm \qquad (C1)$$

where:

U_o = Heat transfer correlation factor

A_o = Outer surface area of coil

δT_m = Log mean temperature difference

The area A_o of the coil can be expressed in terms of its outer diameter D_o and length l_t:

$$A_o = \pi D_o l_t \qquad (C2)$$

Substituting C2 into C1 and rearranging yields an expression for the length of tubing required for the heating coil in a heat exchange vessel:

$$l_t = Q / (\pi D U_o \delta T_m) \qquad (C3)$$

The heat transfer correlation factor for the vessel is given by:

$$1/U_o = 1/h_o + (x A_o)/(k A_w) + A_o/(h_i A_i)$$
$$= 1/h_o + (x D_o)/(k D_w) + D_o/(h_i D_i) \qquad (C4)$$

where:

x = Thickness of coil

k = Thermal conductivity of coil material

A_w = Mean surface area of coil

h_i = Heat transfer coefficient between inner wall of tube and water inside tube

h_o = Heat transfer coefficient between outer wall of tube and water inside vessel

A_i = Inner surface area of coil

For the fully-mixed case under consideration, the log mean temperature difference δT_m across the vessel is given by:

$$\delta T_m = \frac{(T_{hco} - T_{hci})}{\ln\{ (T_{hci} - T_{cho})/(T_{hco} - T_{cho}) \}} \quad (C5)$$

The outer heat transfer coefficient h_o can be determined for the vessel using equations C6 and C7 which follow.

For crossflow case:

$$h_o = 0.198\ C_p\ G\ (N_{Re})^{-0.4}\ (N_{Pr})^{-1/3} \quad (C6)$$

For parallel flow case:

$$h_o = 0.0299\ C_p\ G\ (N_{Re})^{-0.2}\ (N_{Pr})^{-2/3} \quad (C7)$$

where:

Cp = Specific heat of liquid in vessel

G = Mass flowrate of liquid per unit area

N_{Re} = Reynold's number of liquid in vessel

N_{Pr} = Prantyl number of liquid in vessel

Tables relate the flow velocity v inside the tube to the heat transfer coefficient h_i between the liquid in the tube and the tube walls.

So, for any vessel, if C_p, G, N_{Re} and N_{Pr} are known for the liquid in the vessel, h_o can be calculated using equation C6 or C7. If the velocity in v in the tube is known, h_i can be determined from tables (Kern p835). Knowing x, k, A_w, h_i, h_o

and A_i, U_o can be calculated using equation C4. From the specifications of the temperature profiles across the vessel, δT_m can be calculated using equation C5. Q can be calculated by considering the change in temperature across the vessel in any one of the streams. Substituting the values for U_o, δT_m and Q into equation C3 will yield l_t, the length of tubing required for the coils. The number of coil windings required in each case can also easily be determined from the specifications for the diameter of the windings, D_c.

C.1) Specifications of exchange vessels and coils

C.1.1) Specifications of tubing for coils

Copper heating coils are used with the following specifications:

$k = 398$ W/(m)(K) (Perry p3-261)

$D_o = 0.5$ inch

$x = 0.75$ mm

$= 0.03$ inch

So,

$D_i = 0.47$ inch

$D_w = 0.485$ inch

The tubing is wound into helical coils of 25 cm diameter inside the vessels, so,

$$D_t = 0.25 \text{ m}$$

C.1.2) Determining outer heat transfer coefficient, h_o

Assume that the stirrer agitates water such that the reynold's number inside the vessel is approximately:

$$N_{Re} = 5000$$

Assume also that the water moves over the coils at a velocity of:

$$v = 1 \text{ ft/sec}$$

Then, mass flowrate of water (density p) per unit area:

$$G = v\,p$$
$$= 62.5 \text{ lb/(sec ft}^2\text{)}$$

Prantyl number at 142.2°F and heat capacity at 138.6°F (average over operating temperature range for all vessels):

$$N_{Pr} = 3.7$$
$$C_p = 1 \text{ B.t.u./(lb)(°F)}$$

So, for crossflow case:

h_o = 623.5 (equation C6)

and for parallel flow case:

h_o = 511.96 (equation C7)

C.1.3) Determining inner heat transfer coefficient, h_i

The flow velocity v of the water flowing inside the tubing of inner diameter D_i at a rate of F = 3 l/min is given by:

$v = F / A_i$

= 1.47 ft/sec

Table in Kern p 835 shows the heat transfer coefficient between the water and the inner surface of the coil is:

h_i = 640

C.1.4) Determining heat transfer correlation factor, U_o

Equation C4 is applied using the constants determined for the exchange vessel in sections C.1.1 to C.1.3. The heat transfer correlation factor is thus determined to be:

Crossflow case:

$$U_o = 305.904 \text{ B.t.u./(hr ft}^2 \text{ }^\circ\text{F)}$$

Parallel flow case:

$$U_o = 276.392 \text{ B.t.u./(hr ft}^2 \text{ }^\circ\text{F)}$$

C.2) Coil Lengths and Corresponding Numbers of Windings

Table C1 shows results obtained for δT_m and Q for each vessel. The lengths of required tubing l_t obtained when δT_m, Q and U_o (U_o calculated in C.1.3) for each vessel are substituted into equation C3 are also quoted. The number of coil windings required to obtain the calculated tube length is also given. Results for cross and parallel flow quoted.

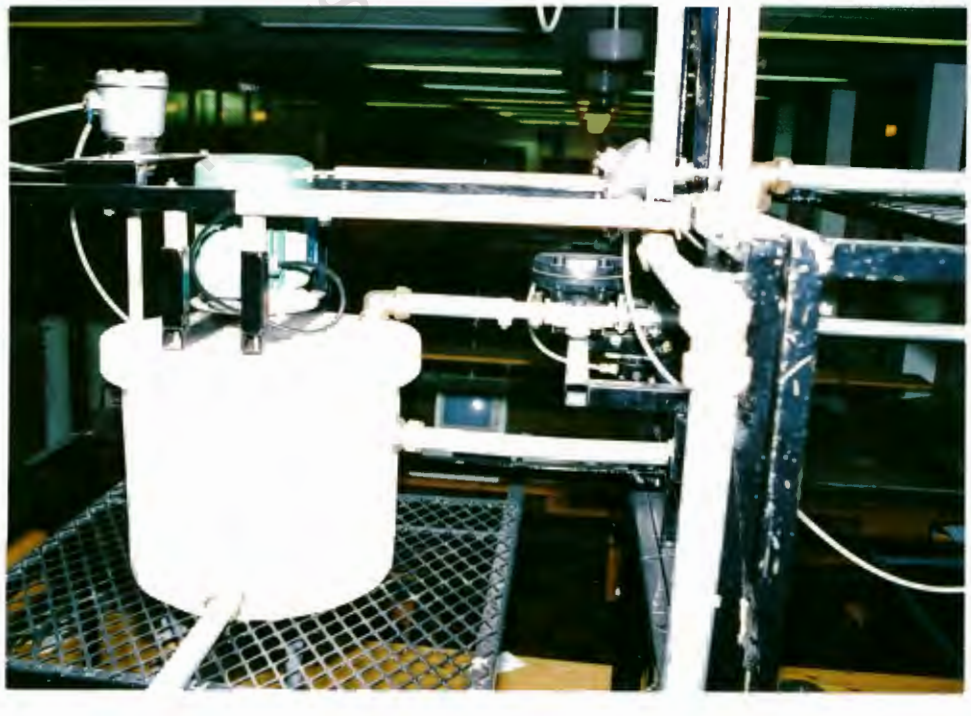
Tank	LMT Dif δT_m (°C)	Heat Tr Q (W)	Length Crs (m)	Length Par (m)	Windings Cross	Windings Parall
1	18.2	4190	3.3	3.7	4.3	4.7
2	8.96	3560	5.7	6.4	7.4	8.1
3	3.29	2090	9.2	10.2	11.8	13.0
4	3.41	1045	4.4	4.9	5.7	6.3

Table C1: Calculation Results Obtained For Each Vessel

C.3) Photographs of Heat Exchange Vessels

Photograph PC.1 shows a dissembled heat exchange vessel. The copper heating coils inside the vessel, the stirrer motor and impeller shaft, and the level probe can bee seen in the photograph. Photograph PC.2 shows a fully assembled heat exchange vessel, which is one of the four (Tank No. 2) on the heat exchanger process. The control valve regulating the flow into the vessel can also be seen, as well as a temperature sensor on one of the pipe bends.

Photograph PC.1: Dissembled Heat Exchange Vessel



Photograph PC.2: Fully Assembled Heat Exchange Vessel

APPENDIX D

EFFECTS OF STIRRING AND CHOICE OF STIRRER CONFIGURATION

Variable speed stirrers have been agitate the water in the exchange vessels of the heat exchanger. In this appendix, the thermal effects of stirring on the reactor tanks are discussed. The reasons for the choice of particular stirrer impellers, speeds and motors are then discussed.

Rearranging equation C3 in Appendix C shows that the log mean temperature difference across a vessel, δT_m, can be expressed as:

$$\delta Tm = Q / (Uo \pi D_o l_t) \qquad (D1)$$

D.1) Effects of varying Impellers Sizes and Speeds on Log Mean Temperature Difference δT_m

Assuming that the heat transferred between the two streams (Q) is to be kept constant, the log mean temperature difference across the reactor (δTm) will change as the correlation factor (U_o) is changed. (assuming all other factors remain unchanged) As U_o increases (thermal resistance decreasing), the temperature gradient required to drive the heat energy from the one stream to the other (δT_m) is expected to drop off.

The outer heat transfer coefficient, h_o is dependent on the degree of agitation of the water in the reactor vessel, and is thus dependent on the degree to which the water in the vessel is being stirred. Thus by changing the degree to which the water is being stirred, h_o can be varied, thus changing U_o (equation C4 in appendix C) and ultimately varying the temperature change across the reactor (equation D1).

An expression relating the speed of rotation of the stirrer, as well as the size of the impeller, to the external heat transfer coefficient h_o of the vessel, can be found in Kern's book "Process Heat Transfer" (p722), and is given as:

$$h_o D_j/k = 0.87 \{(L^2 N p)/\mu\}^{.667} \{(C_p \mu)/k\}^{.333} \{\mu/\mu_w\}^{0.14} \quad (D2)$$

where,

D_j = Diameter of reactor

k_w = Thermal conductivity of water

L = **Length of impeller**

N = **No. of revolutions per hour done by impeller**

p = Average density of water

μ = Viscosity of liquid in reactor

μ_w = Viscosity of water (so $\mu/\mu_w = 1$)

C_p = Specific heat of water

D.2) Effects of Size and Speed of Impeller on Driving Motor Power Required

Different impeller sizes and speeds present different loads to the driving motor. It must be ensured that the motor is capable of driving the specific impeller chosen over the range of operating speeds selected.

On page 719 of his book, Kern states the following relationship between the motor power required and the impeller size and speed:

$$hp = 1.29*10^{-4} D_j^{1.1} L^{2.72} N^{2.86} y^{0.3} z^{0.6} \mu^{0.14} p^{0.86} \quad (D3)$$

where,

hp = Motor power required to drive impeller

y = Width of agitator

z = Height of wetted portion of vessel

D_j, L, N, μ and p as defined for equation 4

D.3) Stirrer Specifications for Heat Exchange Vessels

A heat transfer coefficient of between 511 and 624 was assumed in the calculations in appendix C for the steady-state calculations of the required coil lengths. It can be seen from equations D1 to D3 that it is possible to calculate h_o, δT_m and hp as a function of the impeller shape (L and y) and the speed of revolution of the impeller (N).

Tables D1 and D2 give results obtained for two different impeller shapes for calculations done for exchange tank T3 (refer to Appendix C) to determine:

a) external heat transfer coefficient, ho (equation C2) and the log mean temperature difference across the reactor, δT_m (equations C2 and C1) as a function of rotational speed of impeller.

b) required driving motor power required as a function of rotational speed of impeller (equation C3).

Length of impeller, L = 4cm Width of impeller, y = 2cm			
Rotation speed (rpm)	**Transfer coeff** (h_o)	**L.M.Temp. Difference** (°C)	**Power Required** (kW)
200	161.07	7.26	0.11
700	371.31	4.02	4.27
1200	531.86	3.27	19.9
1500	617.17	3.03	37.8
1700	670.87	2.91	54.1
2200	796.69	2.69	113.1
2700	913.24	2.55	203.2

Table D1: Transfer coefficients, Log Mean Temperature Differences for Varying Impeller Speeds
Shape 1: L = 4cm, y = 2cm, Tank T3

Length of impeller, L = 5cm Width of impeller, y = 2cm			
Rotation speed (rpm)	Transfer coeff (h_o)	L.M.Temp. Difference (°C)	Power Required (kW)
200	216.89	5.79	0.22
700	499.98	3.38	7.85
1200	716.16	2.82	36.7
1500	831.03	2.65	69.4
1700	903.34	2.56	99.3
2200	1072.76	2.40	207.6
2700	1229.69	2.28	372.9

Table D2: Transfer coefficients, Log Mean Temperature Differences for Varying Impeller Speeds
Shape 2: L = 5cm, y = 2cm, Tank T3

(i) Choice of Impeller

It can clearly be seen from tables D1 and D2 that varying the rotational speed of the impeller has a significant effect on the external heat transfer coefficient, and ultimately on the temperature difference across the reactor. This suggests that stirring rates in the reactors could be treated as significant inputs to the Heat Exchanger system if temperatures at various points along the system are to be controlled.

It should be noted, however that when h_o is reduced to below $h_o \approx 10^{-3}$, no significant increase in the temperature difference is observed from the calculations. This can be explained by considering

equation C4 in appendix C. The inverse correlation factor ($1/U_o$) is made up of three terms, the first one only being contributed to by h_o. The second term represents the thermal resistance of the material of the tube, while the third term represents the thermal resistance between the water in the tube and the wall of the tube. Evaluating the first two terms in the expression yields:

$(x * D_o) / (k * D_w) = 2.82 * 10^{-6}$ (second term)

$D_o / (h_i * D_i) = 1.66 * 10^{-3}$ (third term)

The second term can clearly be ignored (very small relative to others). It can be seen, however that the third term will become dominant in the equation when $1/h_o$ is reduced much below 10^{-3}. A point is thus reached (as can be seen in tables D1 and D2) that additional stirring will no longer change $1/U_o$ significantly and the temperature difference will no longer be decreased noticeably. The 5cm impeller does decrease δT_m further than does the 4cm one, but the power required to do so increases sharply. (the cost of variable speed motors capable of developing above 300W of power is relatively high - see table D3) Increasing the flowrate in the coils would increase h_i, and this method could be used instead of additional stirring to change δT_m further.

Calculations done in Appendix C show that at steady state δTm, across tank T3 should be approximately equal to 3.3°C. This situation should occur in the middle of the range of operating speeds of the motor, which is 1500rpm for three phase induction motors. It would then be possible to increase and decrease the speed of the motor from steady state and observe the effects on the temperatures across the vessel. Observing the temperatures calculated in tables D1 and D2, it can be seen that the 4cm impeller comes closer to fulfilling this requirement than does the 5cm one. Furthermore, the range of temperatures over which δT_m can be varied is larger for the former than for the latter.

It appears, from the points raised above, that an impeller which is 4cm in length and 2cm in width is best suited to the requirements for this tank. Since the tank configurations were chosen to be identical, the same stirrer configuration is chosen for the other tanks.

It should be noted from the tables that the relationship between the stirring rates and temperature differences is highly nonlinear and that this is expected to present a challenging control problem.

(ii) Choice of Motor

The choice of the motor to be used can be made after considering:

a) Environmental conditions and reliability
b) Power requirements
c) Cost

The stirrer motors are likely to be running in a humid environment due to evaporation of the hot water in the reactor and reservoir tanks. For this reason, A.C. induction motors are preferred above D.C. motors with brushes which are likely to corrode in the environment invisaged. A.C. Motors are thus likely to be more reliable in this application. There is also no significant cost advantage in acquiring D.C. motors as variable speed D.C. motors are comparable in price to A.C. induction motors with variable speed drives

The power required to drive the 4cm impeller at 2700rpm is approximately 200W. It is thus advisable to choose a driving motor which is capable of producing at least 300W of power. (150% of rated requirement) A list of drives and motors with their respective prices available locally are given in table D3.

Power	Motor and Drive Type	Price
1.1kW	Motor with Danfoss drive	R3200
750W	Motor with Taiwanese drive	R2000
370W	Motor with Taiwanese drive	R1125

Table D3: A.C. Motors, Drives and Powers with their respective Prices

The 370W system was considered adequate and four such motors and drives have been acquired for the heat exchanger rig.

APPENDIX E

WIRING OF POWER TO HEAT EXCHANGER RIG AND INSTRUMENTATION

E.1) Physical Layout of Power Distribution Board

A schematic of the power distribution board assembled for the heat exchanger rig is given in figure E1.

Three phase (termed 3φ) power is supplied to the board from the laboratory 3φ supply (230 V line-to-neutral). Power is cabled to an 80 A/φ 3φ earth leakage unit (E.L.U.), followed by a 80 A/φ 3φ mains switch. This mains switch powers up the 3φ busbars from which power is tapped off for the various items of equipment used on the rig.

The switches SW1 to SW9 shown in figure A1 are the switches distributing single phase power to equipment via power distribution cables PD1 to PD9. Each cable taken off the switches also contains a neutral line, which is tapped off of the neutral busbar shown in the figure.

Switches SW10 to SW12 distribute 3φ power to the heating equipment of the rig via distribution cables PD10 to PD12. Each cable also taps a neutral line off of the neutral bars as the heating elements in the reservoirs are connected in star.

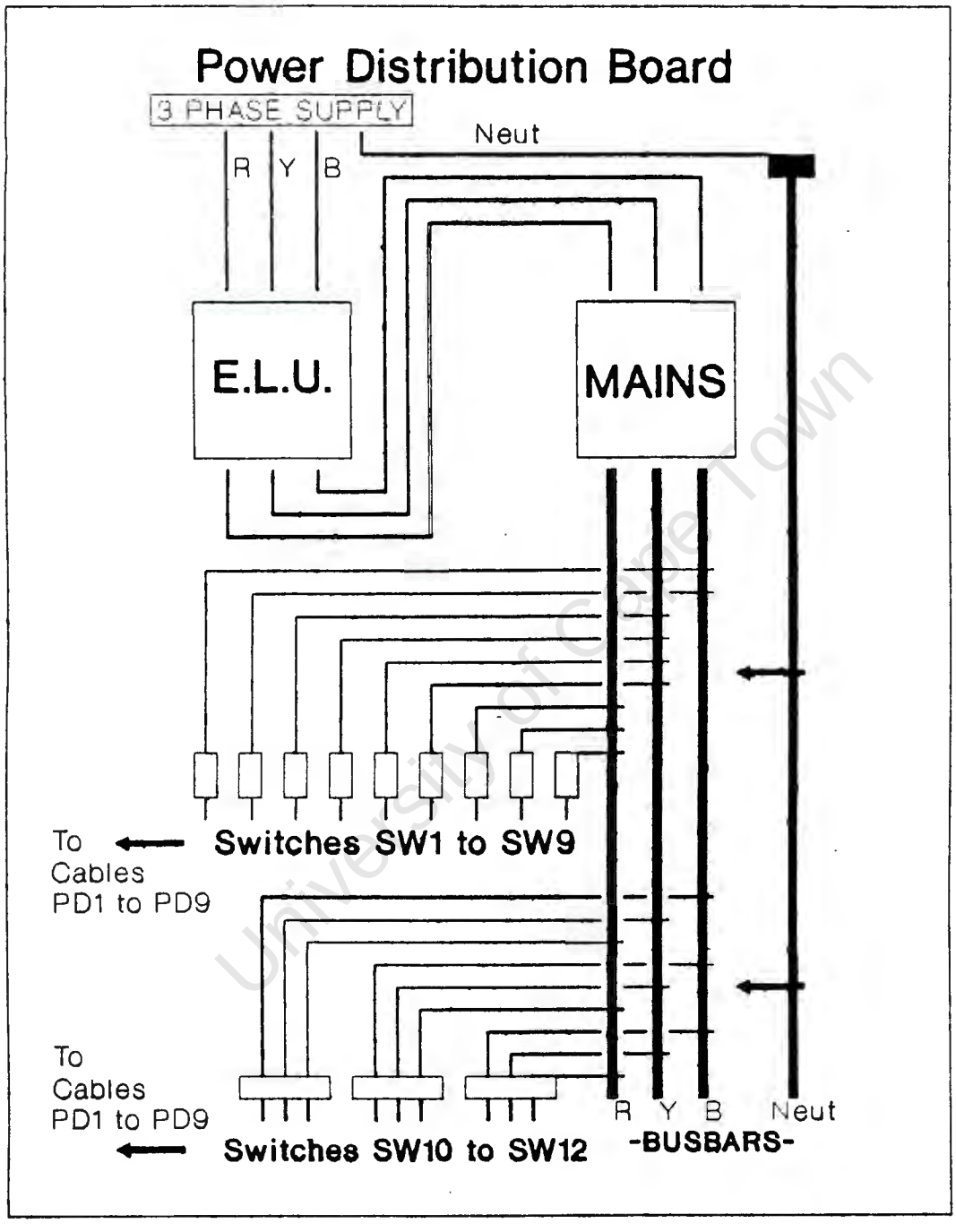


Figure E1: Physical Layout of Power Distribution Box

E.2) Power Distribution Wiring

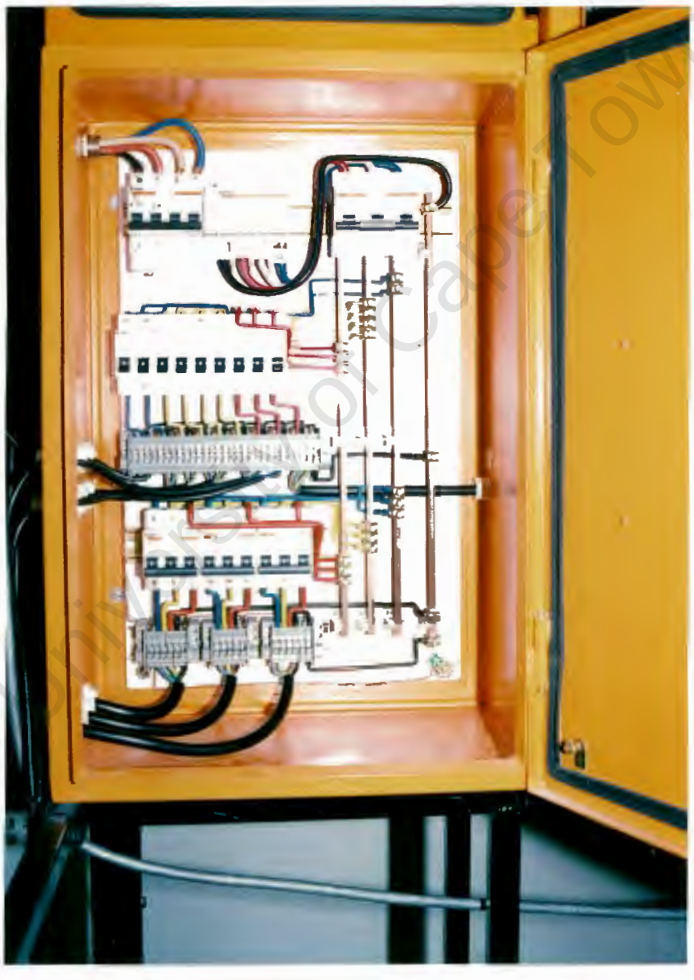
Table A1 shows the way in which the switches and cables are designated to various items of equipment on the rig. The ratings of the respective switches, as well as the codes used to describe the equipment are also given.

Switch No.	Switch Rating (A/φ)	Power Dist. Cable No.	Equip. Code	Item of Equipment Sing/Three Phase
SW1	15	PD1	SV	Solenoid Valve for Recycle of Hot Reservoir - Blue Ph
SW2	15	PD2	CP	Compress - Blue Ph
SW3	15	PD3	S1	Stir No. 1 - Yellow Ph
SW4	15	PD4	S2	Stir No. 2 - Yellow Ph
SW5	15	PD5	S3	Stir No. 3 - Yellow Ph
SW6	15	PD6	S4	Stir No. 4 - Yellow Ph
SW7	15	PD7	PCHR	Pump of Contr Hot Reservoir - Red Ph
SW8	15	PD8	PCCR	Pump of Contr Cold Reservoir - Red Ph
SW9	15	PD9	I	Instrument - Red Ph
SW10	25	PD10	PHR	PreH Hot Reservoir - Three Phase
SW11	15	PD11	CHR	Contr Hot Reservoir - Three Phase
SW12	15	PD12	CCR	Contr Cold Reservoir - Three Phase

Table E1: Designation of Switches and Power Distribution Cables to Equipment on the Rig

E.3) Photograph of Power Distribution Box

Photograph PE.1 shows the power distribution box constructed for the distribution of three phase power to the process. The elements indicated in figure E.1 are seen in the photograph.



Photograph PE.1: Power Distribution Box

APPENDIX F

WIRING OF PLANT INPUT AND OUTPUT SIGNALS

F.1) Standard Low Power Sensing and Control Signals

F.1.1) Physical Layout of Instrumentation Board

A simplified schematic of layout of the instrumentation box assembled for the heat exchanger rig is given in figure F1.

Digital to analog converters (abbreviated to DACs) in the process computer source 4-20 mA current signals proportional to the desired control pressure to be applied to control valves on the rig. In figure F1, C1 to C6 represent the 24 V power supply units which power the 4-20 mA current loops controlling I/P (current to pressure) converters. These I/P converters apply the desired pneumatic control pressure to open and shut the control valves.

DACs also provide 0-5 V signals to the inverter drives controlling the speeds of the stirrer motors.

L1 to L4 shown in the schematic indicate the units which condition the signals received from the capacitive level transmitters on the rig. These signals are converted to 0-10 V signals indicating the extent to which the respective tanks on the rig are filled. These conditioned signals are

then fed to analog to digital converters (ADCs) in the process computer.

Power supplies powering temperature transmitters on the rig are indicated by TS1 to TS10 on figure F1. The transmitters source 4-20 mA current signals, which are converted to 2-10 V voltage signals by series loop resistors in the instrumentation box. These voltage signals are proportional to the temperatures at different points on the rig, and are fed into ADCs.

The flowmeter signal conditioning units, F1 and F2 shown in the schematic power the flowmeters on the rig. They also condition the variable frequency pulsed signals received from these flowmeters, and convert them into 4-20 mA signals indicating the flows measured in the respective sections of piping on the rig. Series loop resistors in the instrumentation box convert these current signals into 2-10 V voltage signals for the ADCs.

Input signals to the plant (to I/P converters), and output signals from the plant (from temperature transmitters, capacitance levels probes, flowmeters) are fed into the instrumentation box along signal cables S1 to S7, as well as along S(i) and S(ii).

F.1.2) Sensing and Control Signal Wiring

Table F1 shows the designation of the signal cables to the signal conditioning items in the instrumentation box. The table also shows cable S8, designated to interconnect the temperature controllers (for the two controlled reservoirs) to their respective temperature probes. These controllers control the switching of relays for the CCR and CHR. Controllers and power relays for reservoirs are located in a separate temperature control box.

Signal Cable	Multicore Wire Pair: Green	Red	Blue	Single/ Multicore Cable
S1	C1	L2	L3	Multi
S2	C6	L1		Multi
S3	C5	TS5	TS6	Multi
S4	C4	TS4	TS7	Multi
S5	C3	TS3	TS8	Multi
S6	C2	TS2	TS9	Multi
S7	L4	TS1	TS10	Multi
S8	TSCCR	TSCHR		Multi
S(i)	F1 (Cold to Hot stream)			Single
S(ii)	F2 (Hot to Cold stream)			Single

Table F1: Designation of Signal Cables to Instrumentation on the Rig

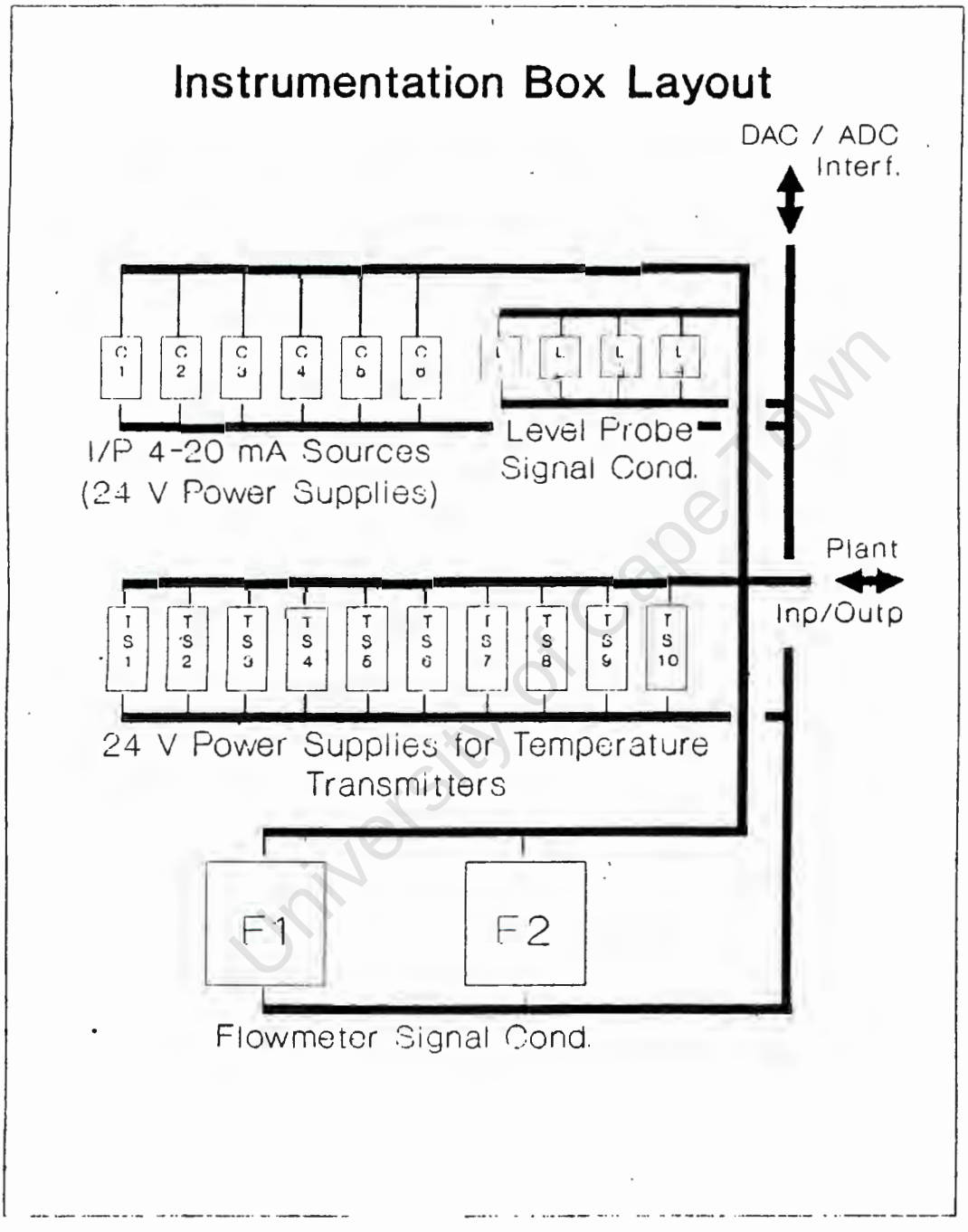


Figure F1: Physical Layout of Instrumentation Box

F.1.3) Photograph of Instrumentation Box

Photograph PF.1 has been taken of the instrumentation box, and shows the housing and interconnection of the equipment referred to in this section.

Photograph PF.1: Instrumentation Box

F.2) Power Signals

Cables conveying power signals to equipment on the rig are those that:

a) Convey 3φ variable frequency power signals from the inverter drives to the stirrer motors.

b) Convey 3φ switched power from the power relays to the elements in the reservoirs.

c) Convey a single phase switched power signal from the thermostat on the pre-heating hot reservoir to the three phase relay which powers the elements on the reservoir.

F.2.1) Power Signal Wiring

Table F2 below gives designation of the cables to the elements and motors on the rig.

Power Control Cable No.	Equip. Code	Item of Equipment
PC1	S1	Stir No. 1 - Three Phase
PC2	S2	Stir No. 2 - Three Phase
PC3	S3	Stir No. 3 - Three Phase
PC4	S4	Stir No. 4 - Three Phase
PC5	PHR	Pre-heating Hot Reservoir - Three Phase
PC6	CHR	Controlled Hot Reservoir - Three Phase
PC7	CCR	Controlled Cold Reservoir - Three Phase
PC8	TPHR	Thermostat of Pre-heating Hot Reserv - Single Phase

Table F2: Power Control Cables to Stirrers and Elements

APPENDIX G

INTERFACING TO DACs AND ADCs

G.1) Physical Layout of Interfacing Equipment

Figure G1 shows the way in which the DACs and ADCs are interfaced to the rig instrumentation.

G.2) Wiring for Interfacing to DACs and ADCs

G.2.1) Interfacing to DACs

The DACs used are DT2815 half-size interface cards which slot into the 8-bit input/output (I/O) bus of IBM XT/AT compatible microcomputer systems. The DACs are capable of delivering 0 - 5V or 4 - 20mA output signals on each of their 8 channels.

Standard DT757 screw termination panels for user connections are plugged into the DACs and housed in an interface box. Each DAC requires two DT757 termination panels. These panels have two sets of terminals each, sets TB1 and TB2, each containing 7 screw terminals for user connections.

Cable 12C1 interfaces the termination panels of DAC1 to the inverter drives of the stirrer motors via 12-pin connectors. These connectors are mounted on the walls of the interface and inverter boxes. DAC1 is programmed to source 0 - 5V

output signals on 4 of its 8 channels individually. These voltages are the control voltage used by the inverters to vary the speeds of the stirrer motors. Similarly, cable 12C2 carries the 4 - 20mA signals programmed on 6 of the channels of DAC2 from the termination panels to the instrumentation box. These signals instruct the I/P converters to open and shut control the control valves on the rig. The 12-pin connectors of 12C2 are mounted on the interface box and the instrumentation box.

Each channel of DAC1 is assigned to a pair of terminals on one of its two DT757s. Table G1(i) shows how the terminals are assigned to each channel of DAC1. The DT757s are connected to the 12-pin connectors of cable 12C1, and the pins on the connector are assigned to each channel as indicated in the table. It also indicates which item of instrumentation is assigned to each channel. Table G1(ii) does the same for the channels of DAC2.

DAC1 Chan No.	DT757 Term Panel 1 Terminal No	DT757 Term Panel 2 Terminal No	12C1 Connector Pin No.	Instr Code
0+	TB2,7		1	S1+
0-	TB1,6		3	S1-
1+	TB2,8		2	S2+
1-	TB1,7		4	S2-
2+	TB2,9		5	S3+
2-	TB1,8		9	S3-
3+	TB2,10		8	S4+
3-	TB1,9		12	S4-

Table G1(i): Assignment of Channels of DAC1

DAC2 Chan No.	DT757 Term Panel 1 Terminal No	DT757 Term Panel 2 Terminal No	12C2 Connector Pin No.	Instr Code
0+	TB2,1		1	C1+
0-	TB1,6		3	C1-
1+	TB2,2		2	C2+
1-	TB1,7		4	C2-
2+	TB2,3		5	C3+
2-	TB1,8		9	C3-
3+	TB2,4		8	C4+
3-	TB1,9		12	C4-
4+		TB2,1	7	C5+
4-		TB1,6	11	C5-
5+		TB2,2	6	C6+
5-		TB1,7	10	C6-

Table G1(ii): Assignment of Channels of DAC2

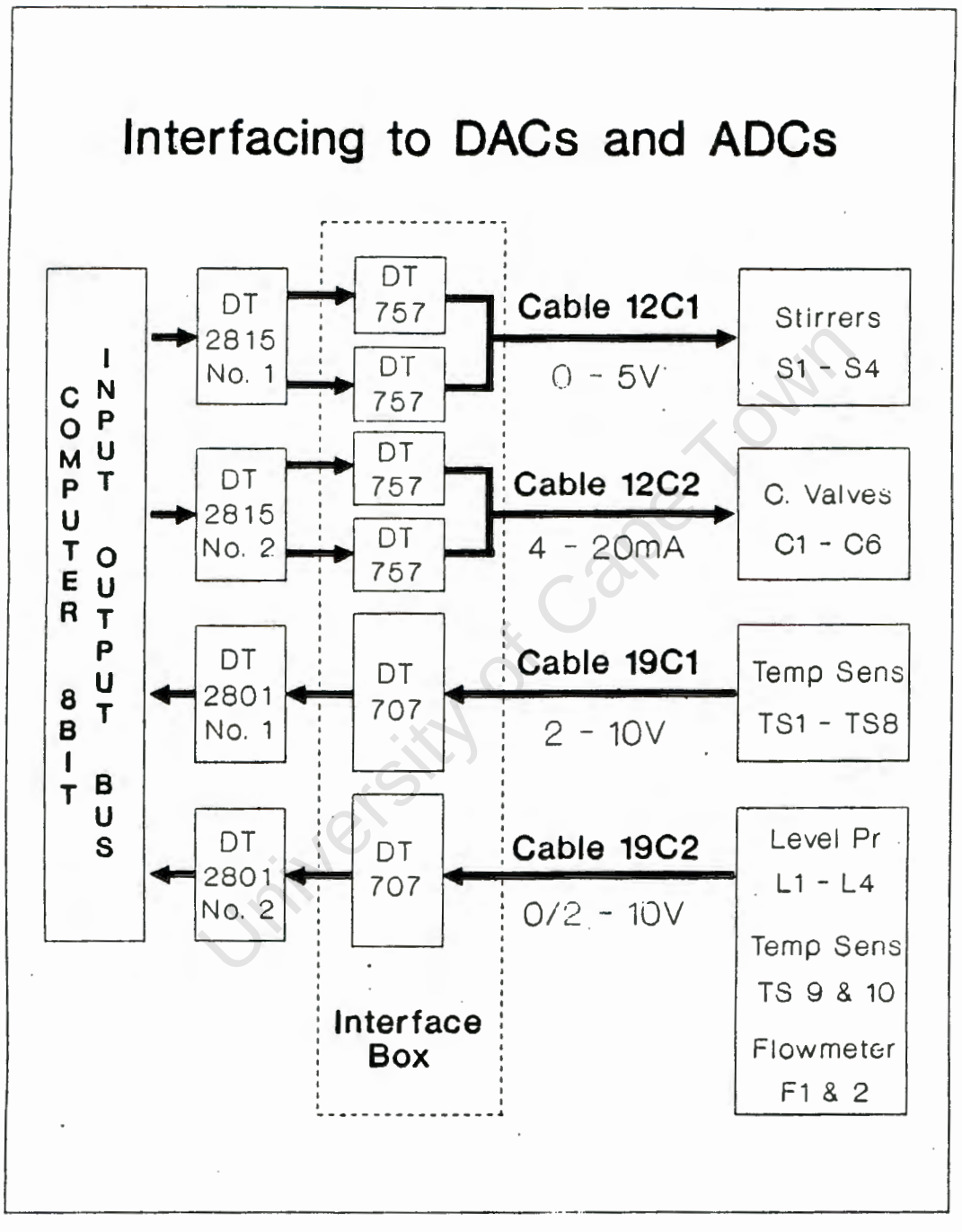


Figure G1: Layout of DAC and ADC Interfacing Equipment

G.2.2) Interfacing to ADCs

The ADCs used are DT2801 full-size interface cards which also slot into an 8-bit XT/AT I/O bus. These ADCs are set up to input 0 - 10V voltage signals on 8 of their analog data input channels.

The screw termination panel used for user connections are the DT707 screw termination panels (only 1 per ADC), and these are also housed in the interface box. Each channel of an ADC has an appropriately labelled pair of screw terminals on the DT707 panel assigned to it.

The interfacing between the termination panel of ADC1 and the instrumentation box is done via the 19-pin connectors on the cable 19C1. ADC1 inputs the 2 - 10V signals from temperature sensors (TS1 to TS8) which indicate temperatures at various locations on the rig.

Similarly, cable 19C2 carries the following signals from the instrument box to the interface box:

a) 2 - 10V signals from the remaining temperature sensor transmitters (TS9 & 10)
b) 0 - 10V signals from level probe transmitters (L1 to L4)
c) 2 - 10V signals from flow transmitters (F1 & 2)

As usual, the 19-pin connectors of cables 19C1 and 19C2 are mounted on the walls of the interface and instrumentation boxes.

The assignment of each channel of ADC1 to a pair of pins on the connector on 19C1 is given in table G2(i). The corresponding rig instrumentation connected to each channel of ADC1 is also given in the table. The same is done for ADC2 in table G2(ii). The terminals on the DT707 screw termination panels are marked with the names of the ADC channels.

ADC1 Chan No.	19C1 Connector Pin No.	Instr Code
0+	16	TS1+
0-	19	TS1-
1+	18	TS2+
1-	15	TS2-
2+	14	TS3+
2-	17	TS3-
3+	8	TS4+
3-	13	TS4-
4+	4	TS5+
4-	1	TS5-
5+	2	TS6+
5-	5	TS6-
6+	6	TS7+
6-	3	TS7-
7+	12	TS8+
7-	7	TS8-

Table G2(i): Assignment of Channels of ADC1

ADC2 Chan No.	19C2 Connector Pin No.	Instr Code
0+	16	F1+
0-	19	F1-
1+	18	F2+
1-	15	F2-
2+	14	L1+
2-	17	L1-
3+	8	L2+
3-	13	L2-
4+	4	L3+
4-	1	L3-
5+	2	L4+
5-	5	L4-
6+	6	TS9+
6-	3	TS9-
7+	12	TS10+
7-	7	TS10-

Table G2(ii): Assignment of Channels of ADC2

APPENDIX H

STRUCTURAL ANALYSIS OF HEAT EXCHANGER PROCESS

H.1) Obtaining BIM by Observing Physical Structure of Heat Exchanger

Tables H1 to H7 show which of the process inputs, valves C1..C6 and stirrers S1..S4, affect the process outputs, flows F1..F2, L1..L4, T1..T10. Brief explanations are also given for the entries in the table.

The information is obtained by considering the diagram of the configuration of the process shown in figure 1.4.

Output	Effect	Explanation
F1	Yes	C1 changes CH flow that F1 measures
F2	No	No physical fluid connection
L4	Yes	C1 changes flow into tank 4
L3	Yes	Disturbed level L4 affects level L3
L2	Yes	Disturbed level L3 affects level L2
L1	Yes	Disturbed level L2 affects level L1
T1	Yes	T1..T10 interact, disturbed by C1..C6
..	..	..
..	..	..
T10	Yes	T1..T10 interact, disturbed by C1..C6

Table H1: Effects of Input C1 on Outputs

Output	Effect	Explanation
F1	No	No physical fluid connection
F2	No	No physical fluid connection
L4	Yes	C2 changes flow out of tank 4
L3	Yes	Disturbed level L4 affects level L3
L2	Yes	Disturbed level L3 affects level L2
L1	Yes	Disturbed level L2 affects level L1
T1	Yes	T1..T10 interact, disturbed by C1..C6
..	..	..
..	..	..
T10	Yes	T1..T10 interact, disturbed by C1..C6

Table H2: Effects of Input C2 on Outputs

Output	Effect	Explanation
F1	No	No physical fluid connection
F2	No	No physical fluid connection
L4	No	No physical fluid connection
L3	Yes	C3 changes flow out of tank 3
L2	Yes	Disturbed level L3 affects level L2
L1	Yes	Disturbed level L2 affects level L1
T1	Yes	T1..T10 interact, disturbed by C1..C6
..	..	..
..	..	..
T10	Yes	T1..T10 interact, disturbed by C1..C6

Table H3: Effects of Input C3 on Outputs

Output	Effect	Explanation
F1	No	No physical fluid connection
F2	No	No physical fluid connection
L4	No	No physical fluid connection
L3	No	No physical fluid connection
L2	Yes	C4 changes flow out of tank 2
L1	Yes	Disturbed level L2 affects level L1
T1	Yes	T1..T10 interact, disturbed by C1..C6
..	..	..
..	..	..
T10	Yes	T1..T10 interact, disturbed by C1..C6

Table H4: Effects of Input C4 on Outputs

Output	Effect	Explanation
F1	No	No physical fluid connection
F2	No	No physical fluid connection
L4	No	No physical fluid connection
L3	No	No physical fluid connection
L2	No	No physical fluid connection
L1	Yes	C5 changes flow out of tank 1
T1	Yes	T1..T10 interact, disturbed by C1..C6
..	..	..
..	..	..
T10	Yes	T1..T10 interact, disturbed by C1..C6

Table H5: Effects of Input C5 on Outputs

Output	Effect	Explanation
F1	No	No physical fluid connection
F2	Yes	C6 changes HC flow that F2 measures
L4	No	No physical fluid connection
L3	No	No physical fluid connection
L2	No	No physical fluid connection
L1	No	No physical fluid connection
T1	Yes	T1..T10 interact, disturbed by C1..C6
..	..	..
..	..	..
T10	Yes	T1..T10 interact, disturbed by C1..C6

Table H6: Effects of Input C6 on Outputs

Output	Effect	Explanation
F1	No	No effect on volume of flow of liquid
F2	No	No effect on volume of flow of liquid
L4	No	No effect on volume of flow of liquid
L3	No	No effect on volume of flow of liquid
L2	No	No effect on volume of flow of liquid
L1	No	No effect on volume of flow of liquid
T1	Yes	T1..T10 interact, disturbed by S1..S4
..	..	..
..	..	..
T10	Yes	T1..T10 interact, disturbed by S1..S4

Table H7: Effects of Input S1..S4 on Outputs

The information presented in tables H1..H7 can be summarized in the form of a BIM for the process, B(G(s)), where:

B(G(s)) =

	C1	C2	C3	C4	C5	C6	S1	S2	S3	S4
F1	1	0	0	0	0	0	0	0	0	0
F2	0	0	0	0	0	1	0	0	0	0
L4	1	1	0	0	0	0	0	0	0	0
L3	1	1	1	0	0	0	0	0	0	0
L2	1	1	1	1	0	0	0	0	0	0
L1	1	1	1	1	1	0	0	0	0	0
T1	1	1	1	1	1	1	1	1	1	1
T2	1	1	1	1	1	1	1	1	1	1
T3	1	1	1	1	1	1	1	1	1	1
T4	1	1	1	1	1	1	1	1	1	1
T5	1	1	1	1	1	1	1	1	1	1
T6	1	1	1	1	1	1	1	1	1	1
T7	1	1	1	1	1	1	1	1	1	1
T8	1	1	1	1	1	1	1	1	1	1
T9	1	1	1	1	1	1	1	1	1	1
T10	1	1	1	1	1	1	1	1	1	1

(2.8)

H.2) Initial Responses of Process Ouputs to Changes in Control Valve Inputs

In this section, the observed responses to initial tests done on the control valve inputs are shown. These tests were done to obtain some indication of expected response times and possible instabilities of the process.

The relevant output responses to a step change in each input is considered in turn. Comments are made regarding the stability of the output responses and the response times observed. The title of each observed response is also given.

H.2.1) Input: Control Valve C1

Affected Outputs, Stability and Response Time:

Flow	F1	Stable	Fast (< 50 sec)
Levels	L1..L4	Stable	Fast (< 400 sec)
Temperatures	T1..T10	Stable	Slow (> 500 sec)

Response Plots:

Figure H1

Flow Response to Step in Control Valve C1

Figure H2

Level Response to Step in Control Valve C1

Figure H3

Temp. Response to Step in Control Valve C1

Figure H4

Temp. Response to Step in Control Valve C1

Figure H5

Temp. Response to Step in Control Valve C1

Figure H6

Temp. Response to Step in Control Valve C1

Figure H7

Temp. Response to Step in Control Valve C1

H.2.2) Input: Control Valve C2

Affected Outputs, Stability and Response Time:

Levels	L1..L4	Stable	Fast (< 400 sec)
Temperatures	T1..T10	Stable	Slow (> 500 sec)

Response Plots:

Figure H8

Level Response to Step in Control Valve C2

Figure H9

Temp. Response to Step in Control Valve C2

Figure H10

Temp. Response to Step in Control Valve C2

Figure H11

Temp. Response to Step in Control Valve C2

Figure H12

Temp. Response to Step in Control Valve C2

H.2.4) Input: Control Valve C4

Affected Outputs, Stability and Response Time:

Levels	L1..L2	Stable	Fast (< 400 sec)
Temperatures	T1..T10	Stable	Slow (> 500 sec)

Response Plots:

Figure H18

Level Response to Step in Control Valve C4

Figure H19

Temp. Response to Step in Control Valve C4

Figure H20

Temp. Response to Step in Control Valve C4

Figure H21

Temp. Response to Step in Control Valve C4

Figure H22

Temp. Response to Step in Control Valve C4

H.2.5) Input: Control Valve C5

Affected Outputs, Stability and Response Time:

Levels	L1	Stable	Fast (< 400 sec)
Temperatures	T1..T10	Stable	Slow (> 500 sec)

Response Plots:

Figure H23

Level Response to Step in Control Valve C5

Figure H24

Temp. Response to Step in Control Valve C5

Figure H25

Temp. Response to Step in Control Valve C5

Figure H26

Temp. Response to Step in Control Valve C5

Figure H27

Temp. Response to Step in Control Valve C5

H.2.6) Input: Control Valve C6

Affected Outputs, Stability and Response Time:

Flow	F2	Stable	Fast (< 400 sec)
Temperatures	T1..T10	Stable	Slow (> 500 sec)

Response Plots:

Figure H28

Flow Response to Step in Control Valve C6

Figure H29

Temp. Response to Step in Control Valve C6

Figure H30

Temp. Response to Step in Control Valve C6

Figure H31

Temp. Response to Step in Control Valve C6

Figure H32

Temp. Response to Step in Control Valve C6

Figure H33

Temp. Response to Step in Control Valve C6

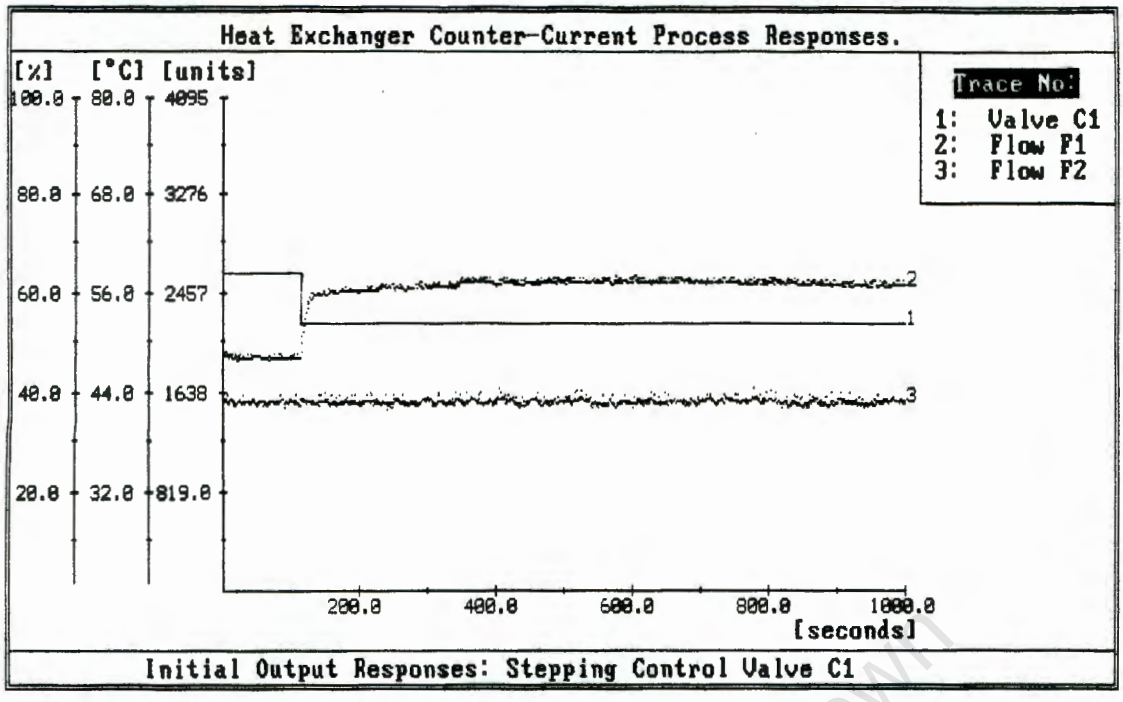


Figure H1: Flow Response to Step in Control Valve C1

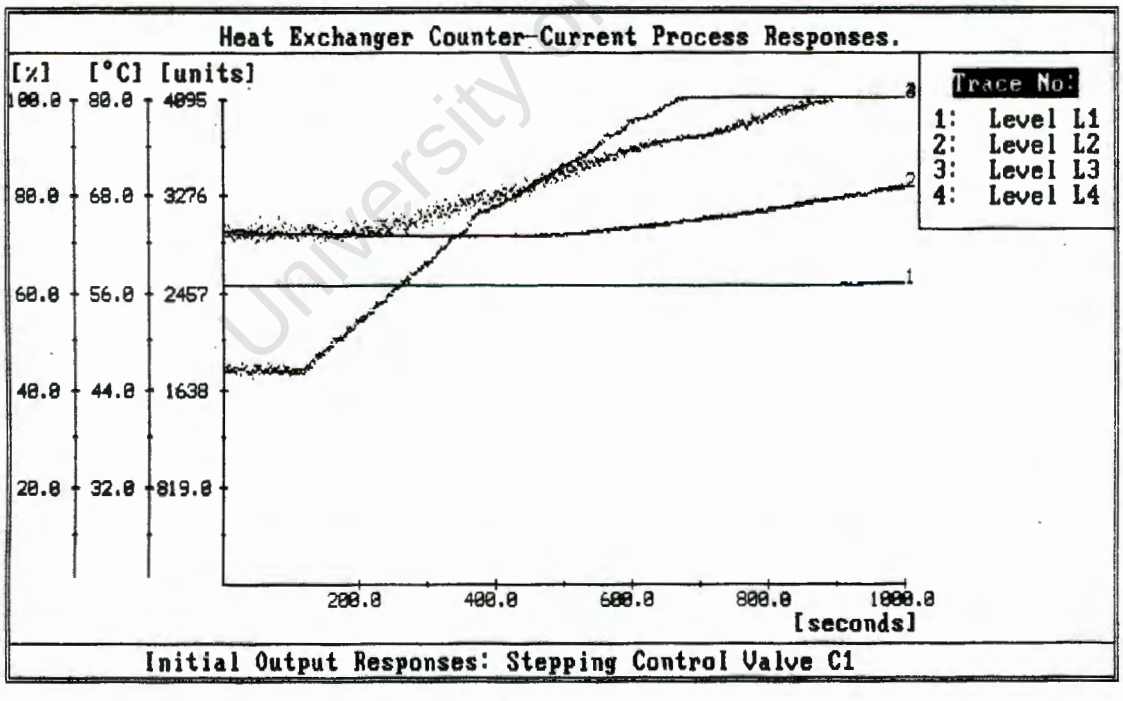


Figure H2: Level Response to Step in Control Valve C1

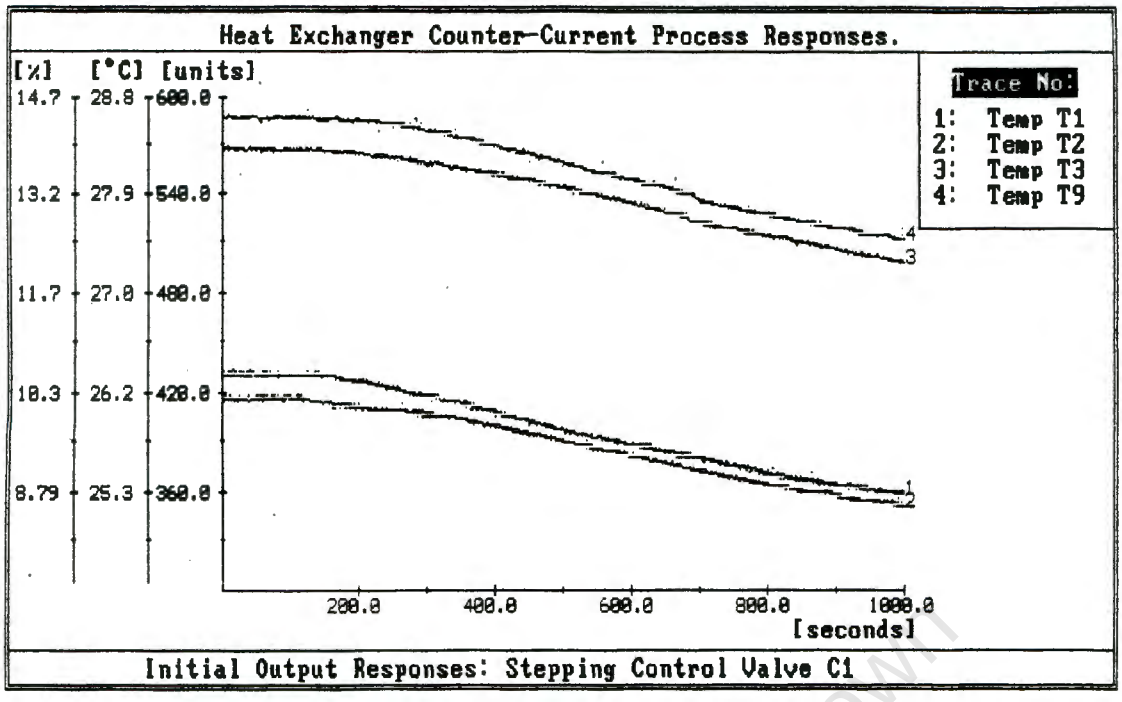


Figure H3: Temp. Response to Step in Control Valve C1

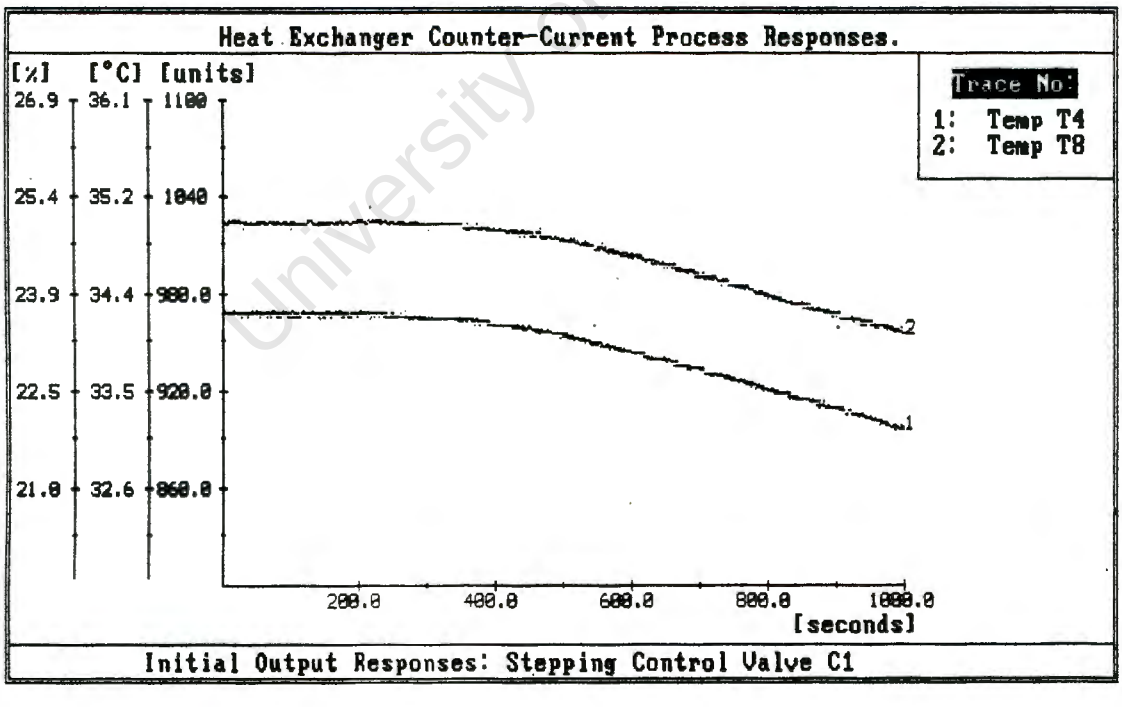


Figure H4: Temp. Response to Step in Control Valve C1

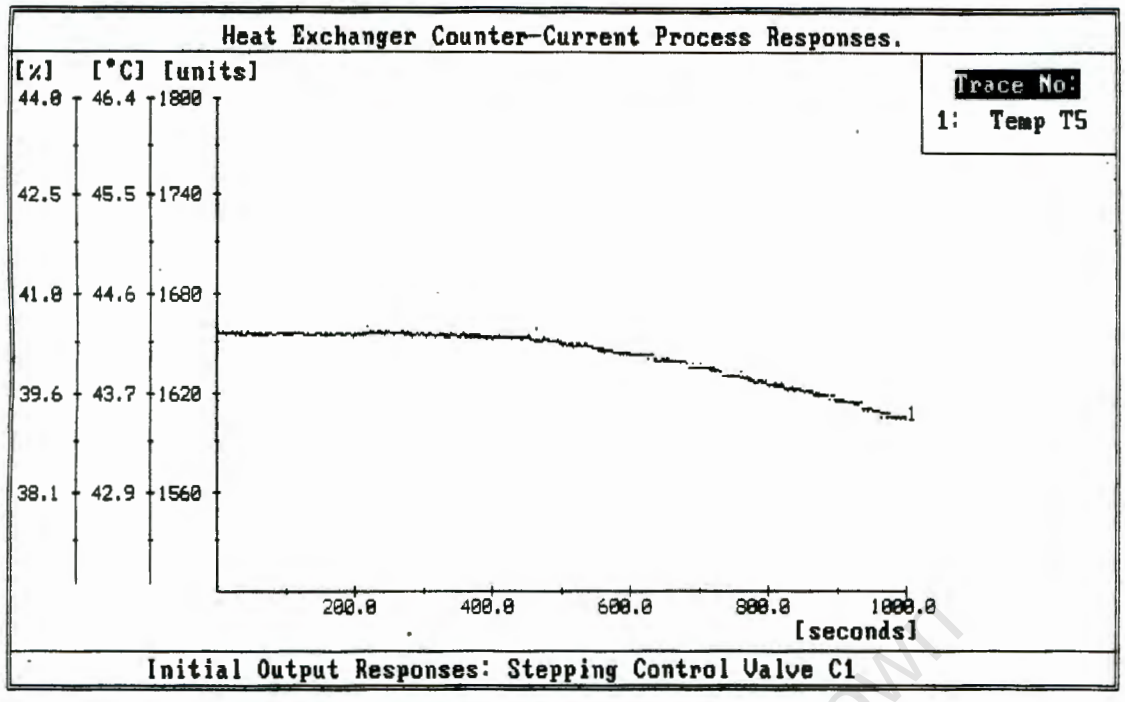


Figure H5: Temp. Response to Step in Control Valve C1

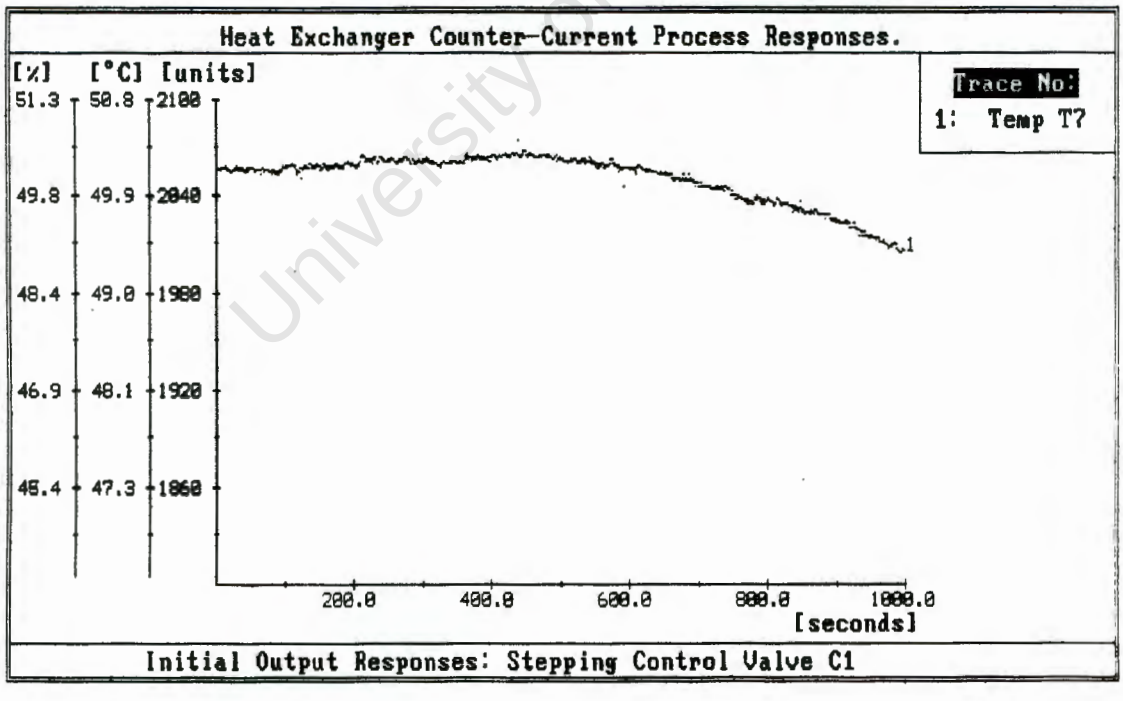


Figure H6: Temp. Response to Step in Control Valve C1

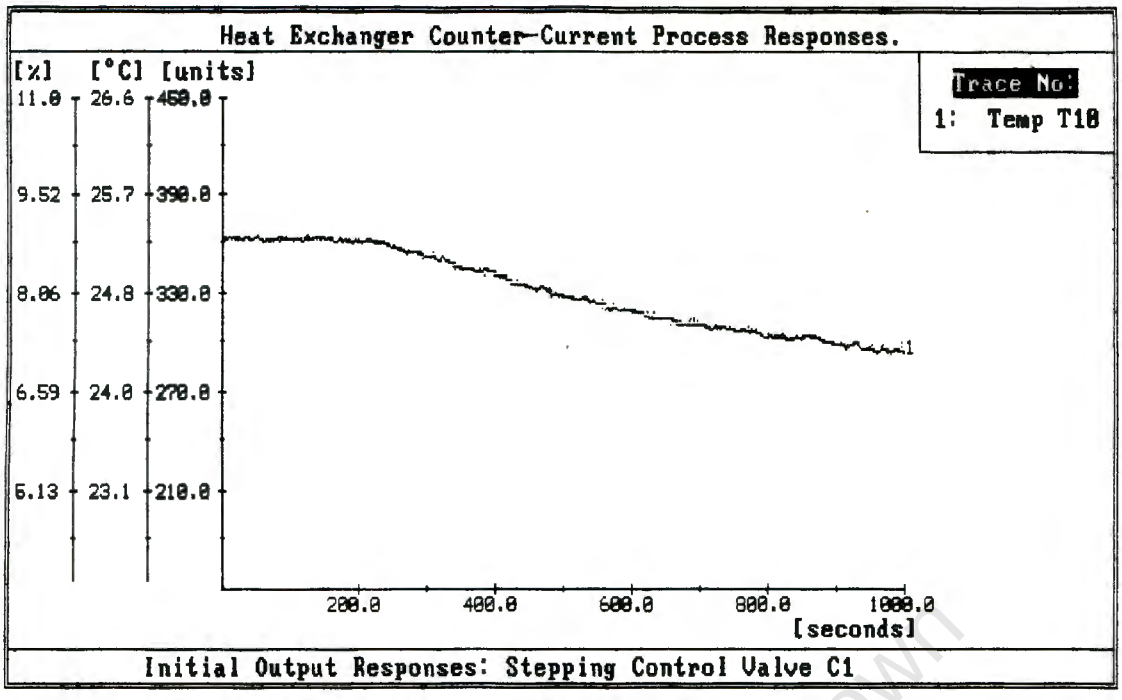


Figure H7: Temp. Response to Step in Control Valve C1

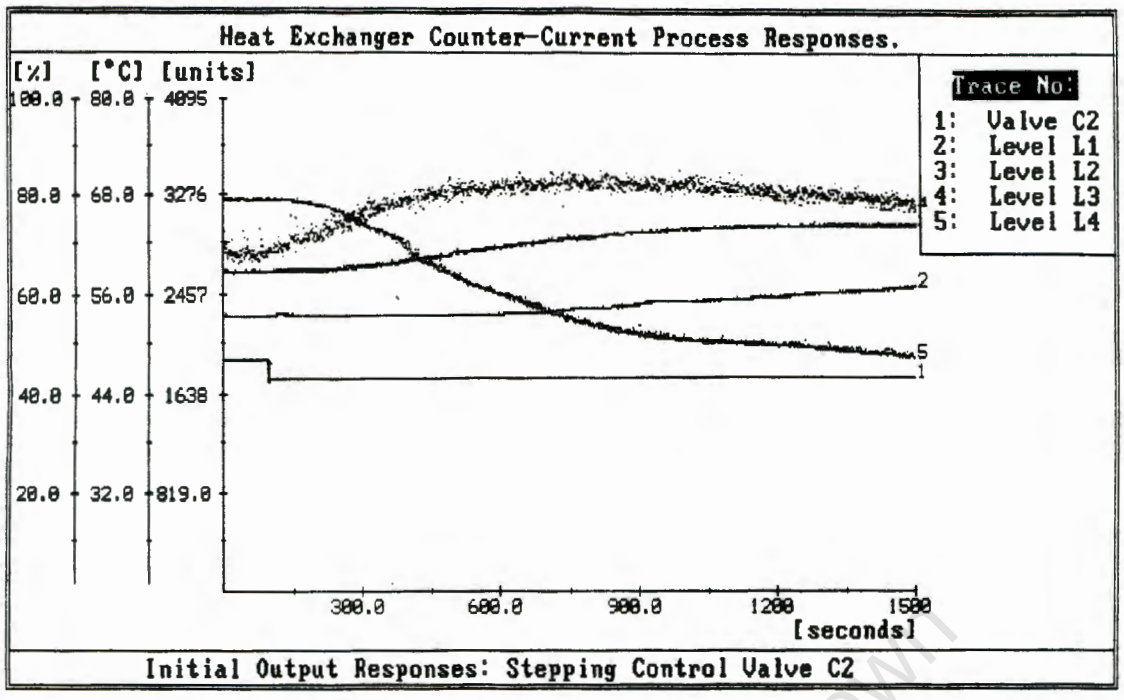


Figure H8: Level Response to Step in Control Valve C2

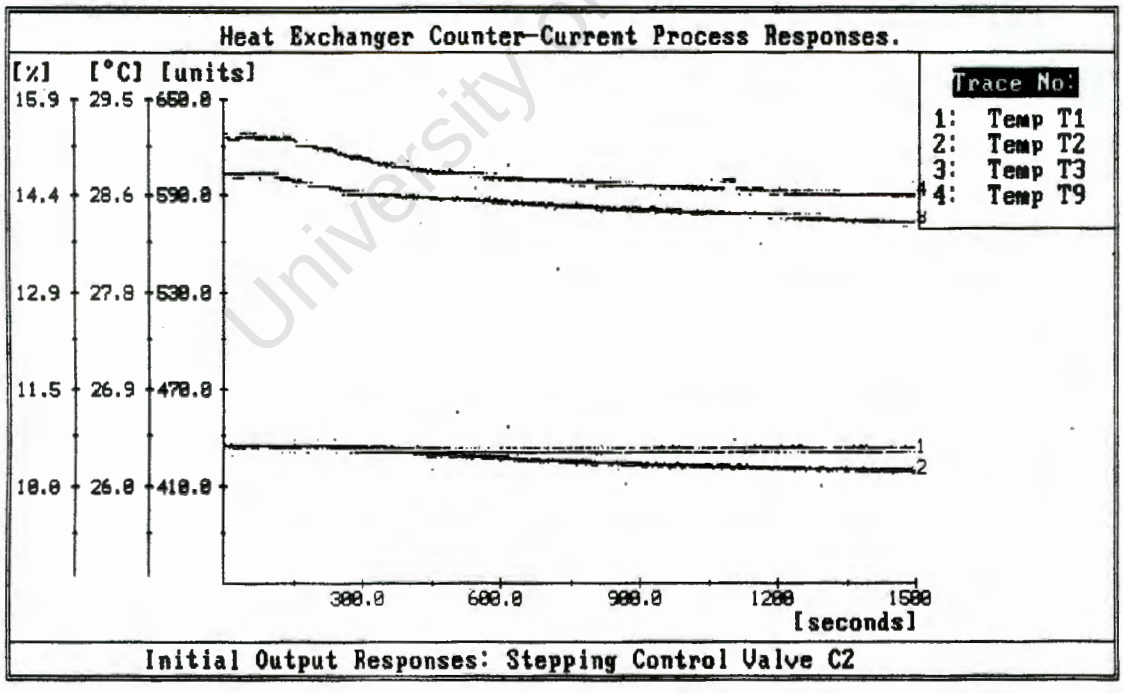


Figure H9: Temp. Response to Step in Control Valve C2

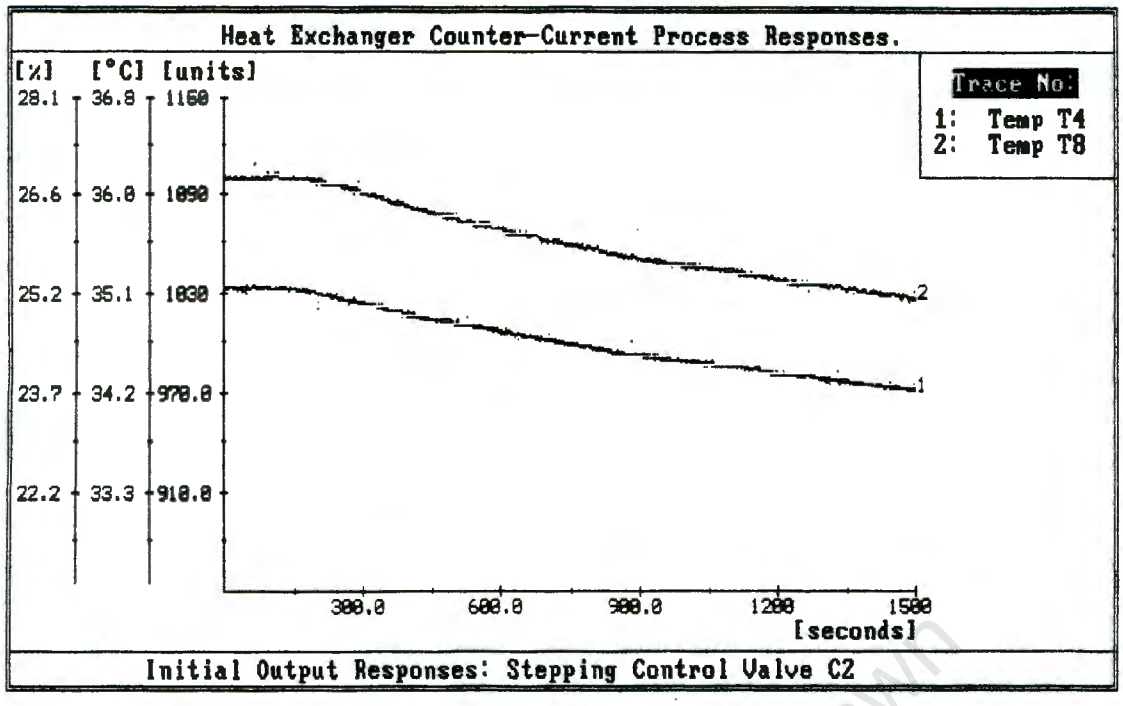


Figure H10: Temp. Response to Step in Control Valve C2

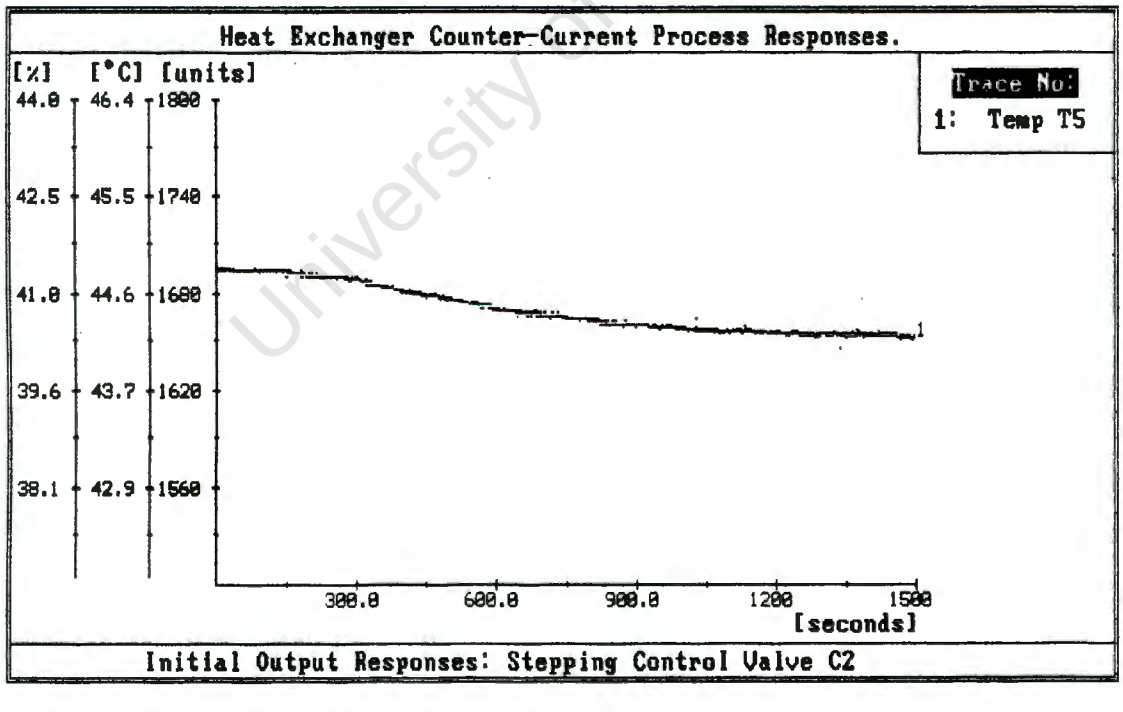


Figure H11: Temp. Response to Step in Control Valve C2

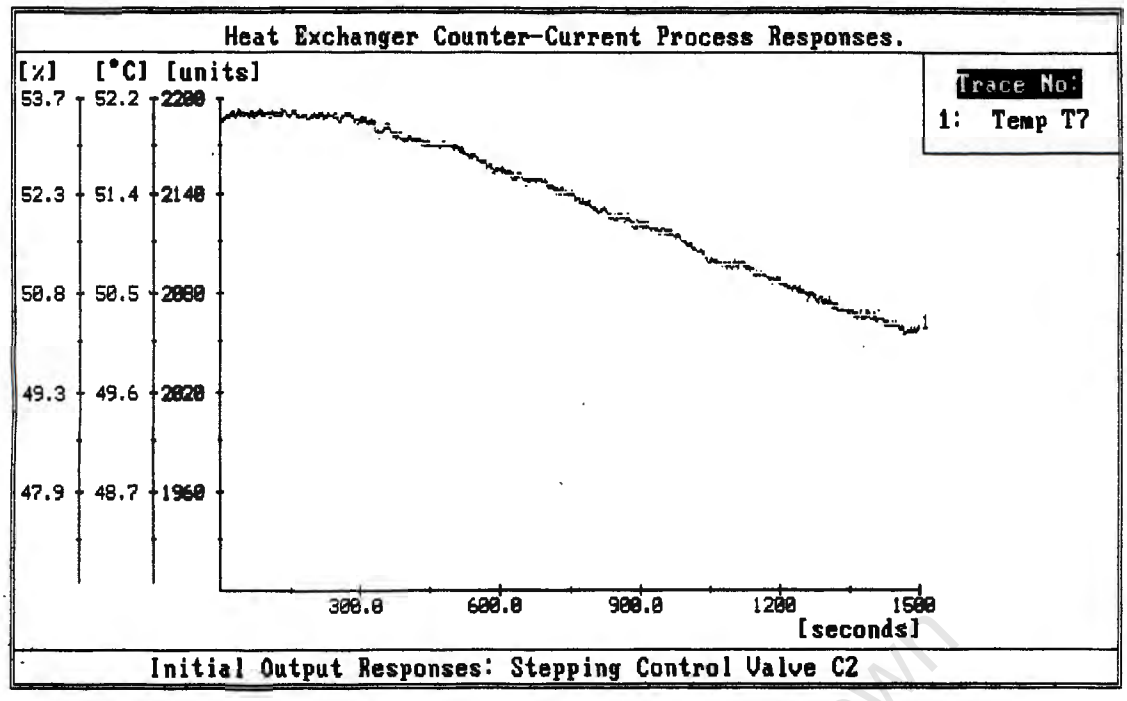


Figure H12: Temp. Response to Step in Control Valve C2

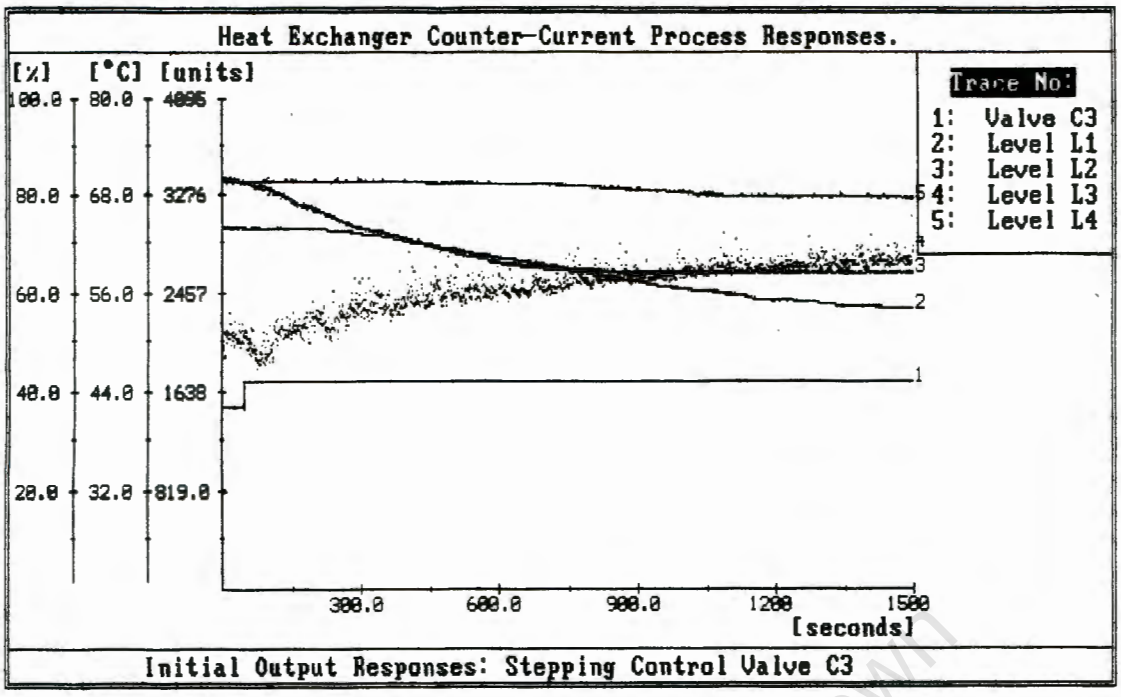


Figure H13: Level Response to Step in Control Valve C3

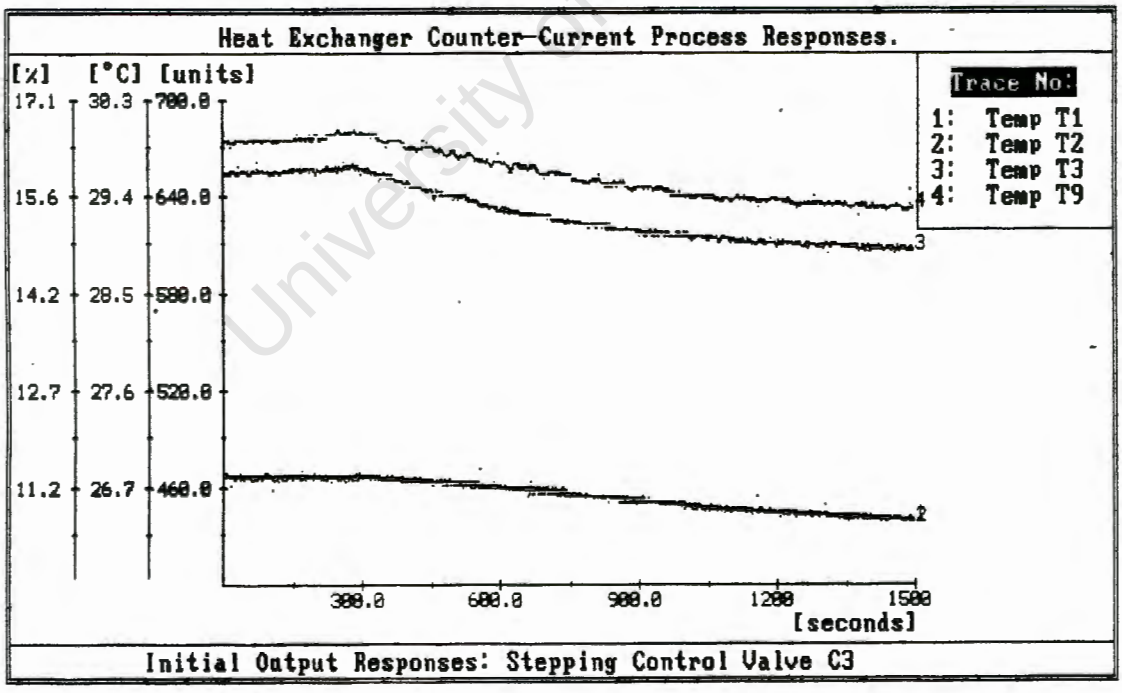


Figure H14: Temp. Response to Step in Control Valve C3

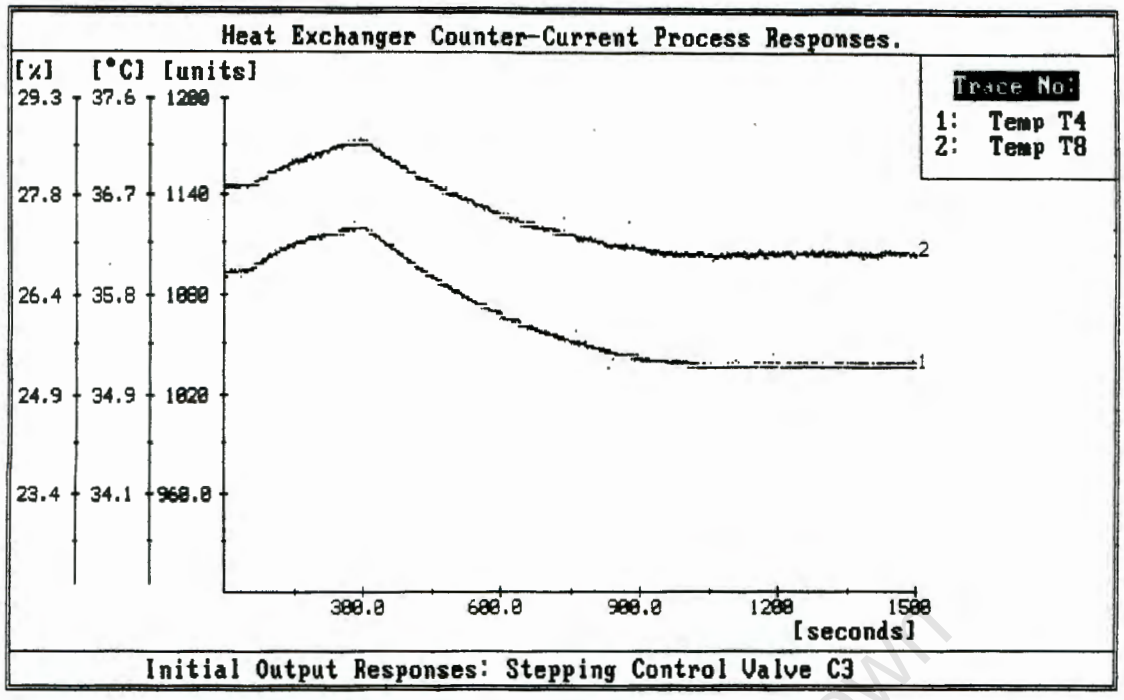


Figure H15: Temp. Response to Step in Control Valve C3

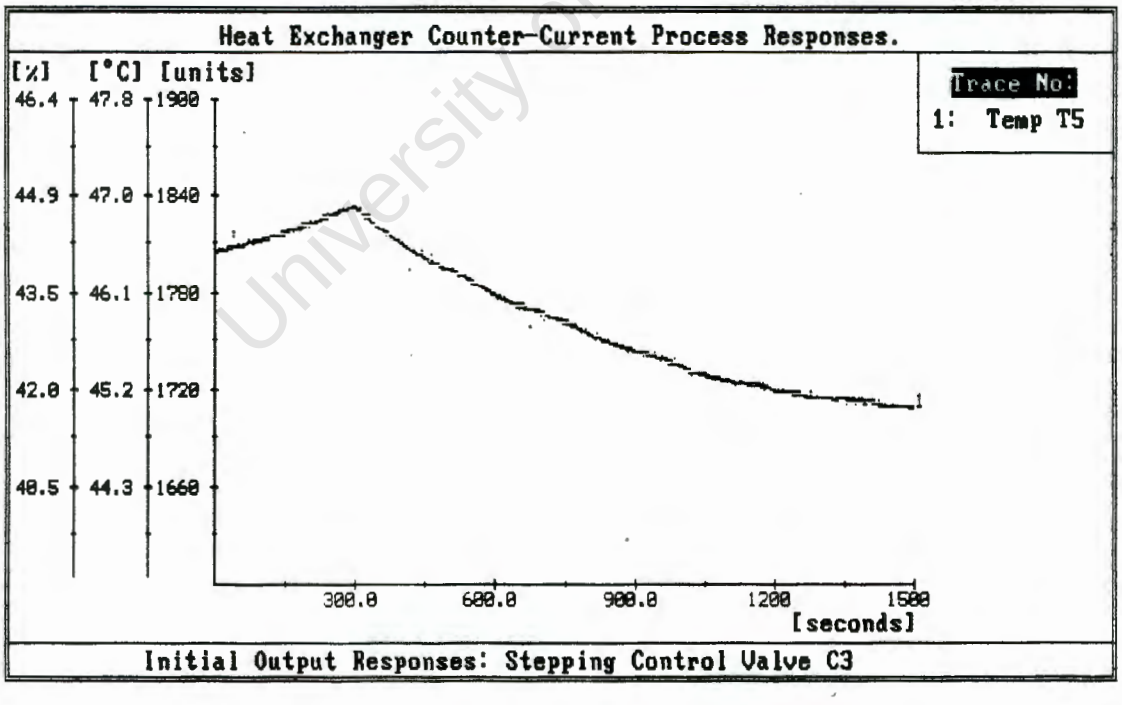


Figure H16: Temp. Response to Step in Control Valve C3

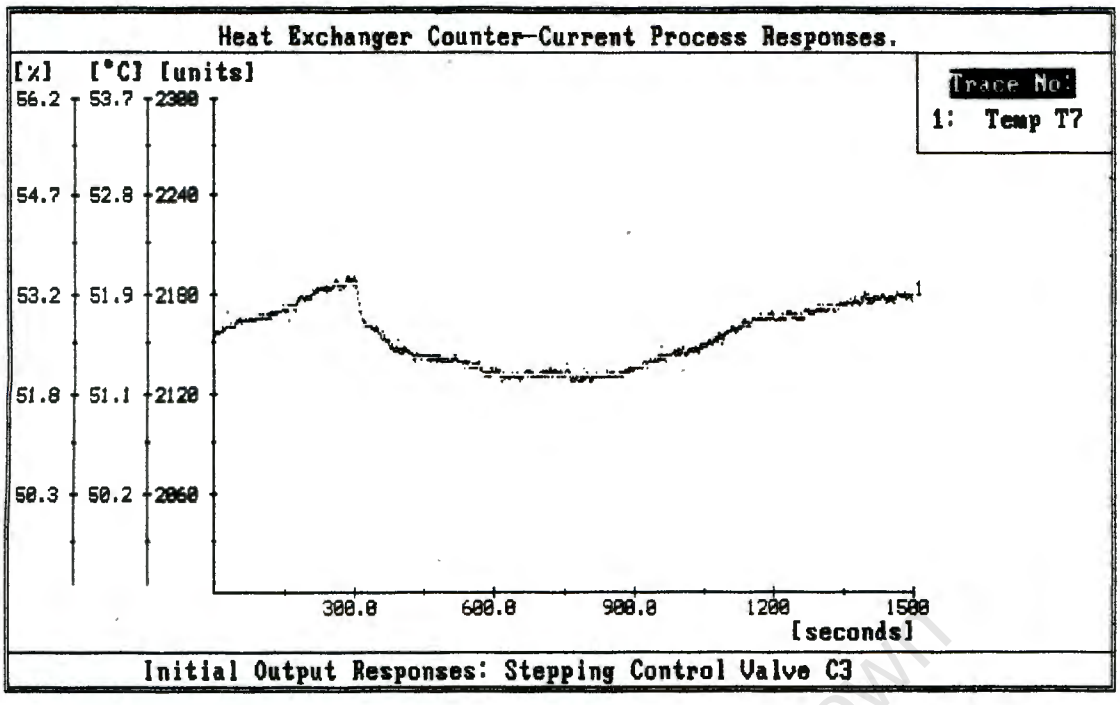


Figure H17: Temp. Response to Step in Control Valve C3

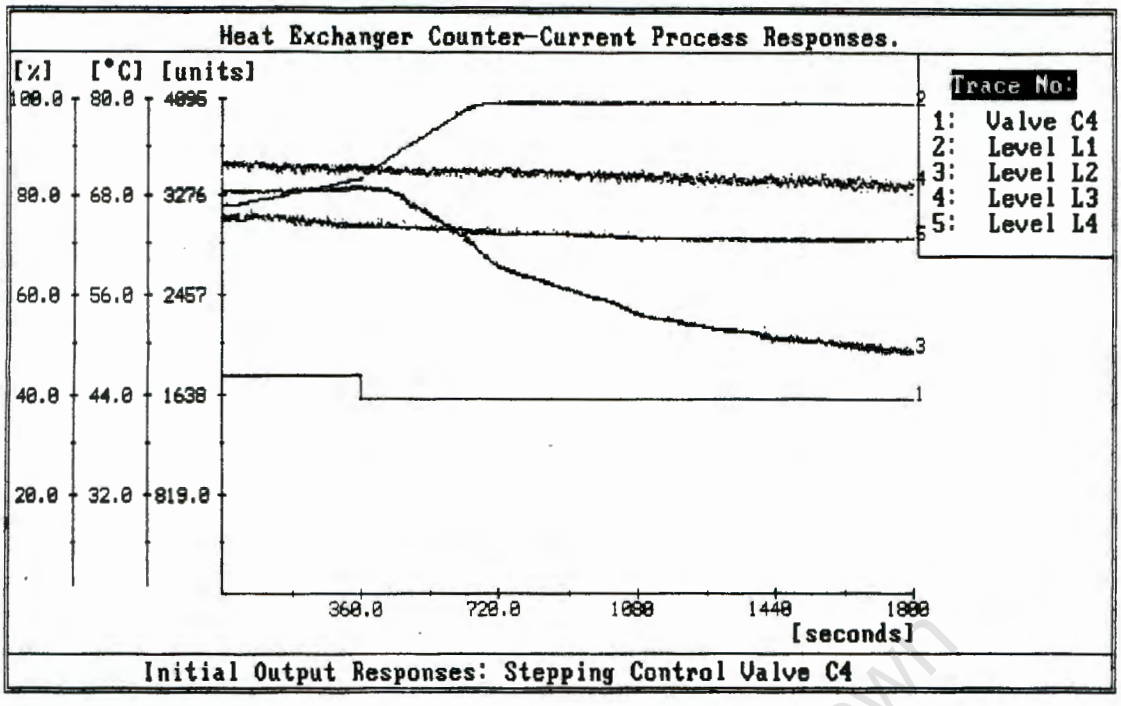


Figure H18: Level Response to Step in Control Valve C4

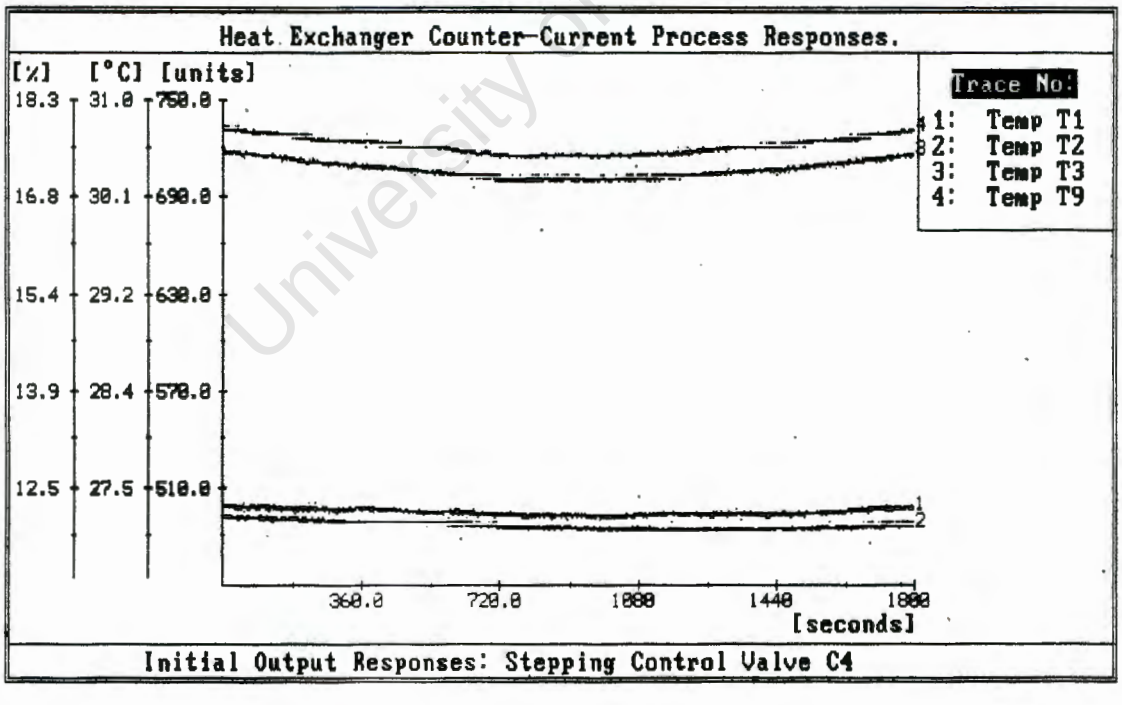


Figure H19: Temp. Response to Step in Control Valve C4

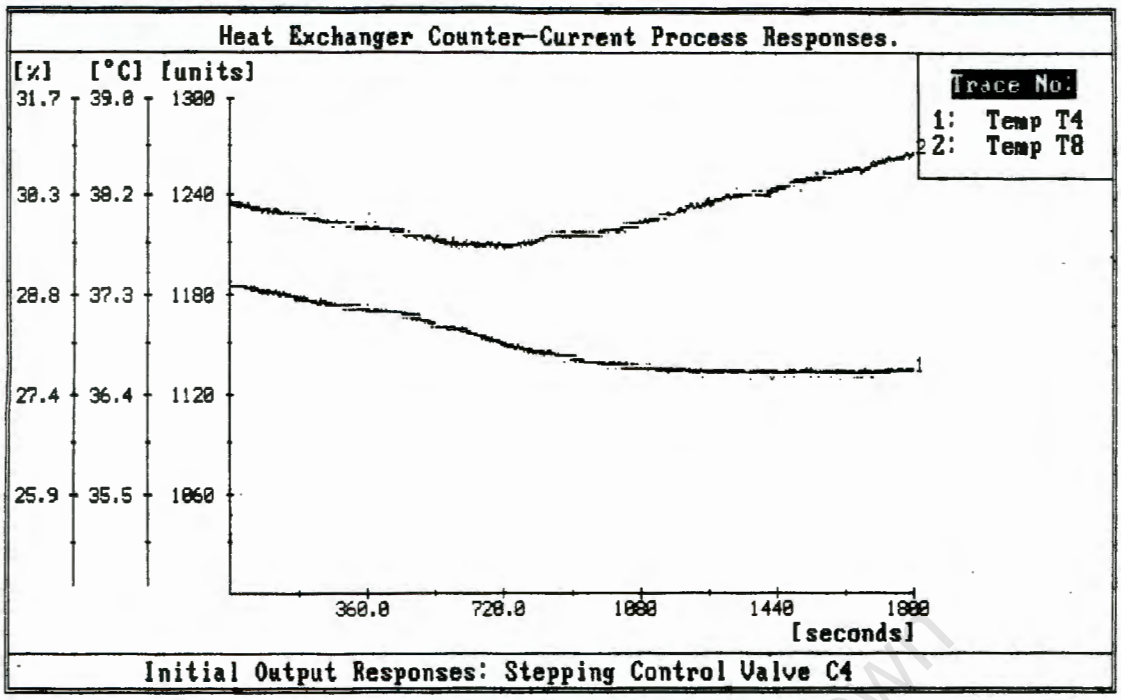


Figure H20: Temp. Response to Step in Control Valve C4

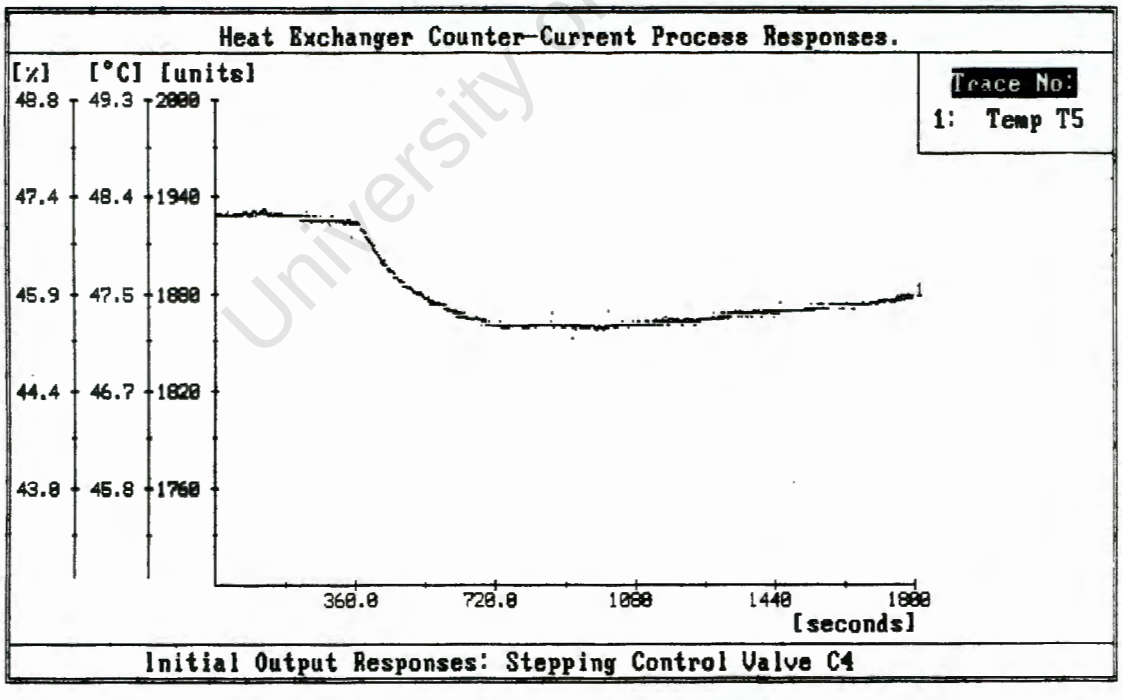


Figure H21: Temp. Response to Step in Control Valve C4

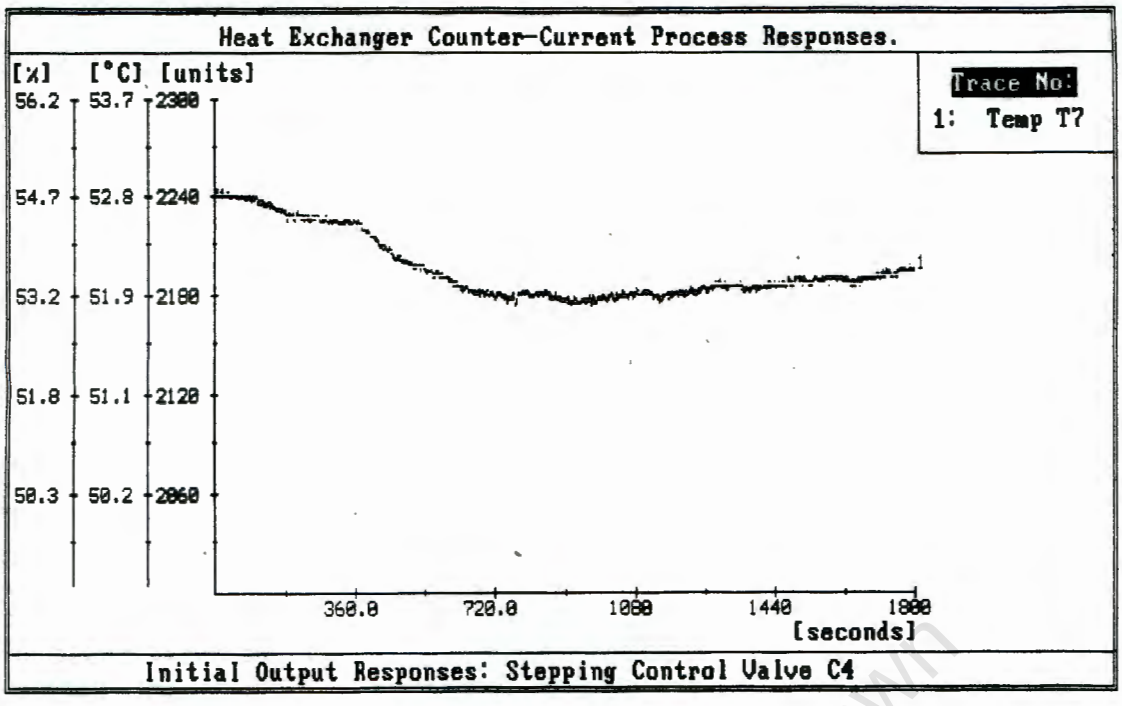


Figure H22: Temp. Response to Step in Control Valve C4

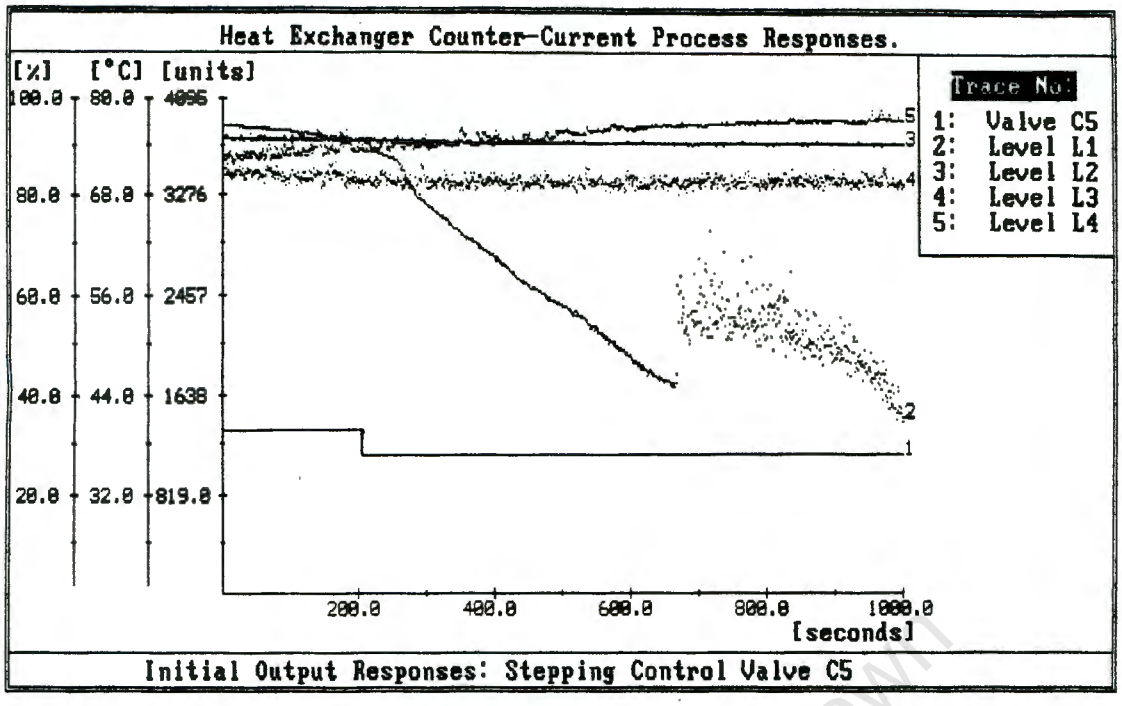


Figure H23: Level Response to Step in Control Valve C5

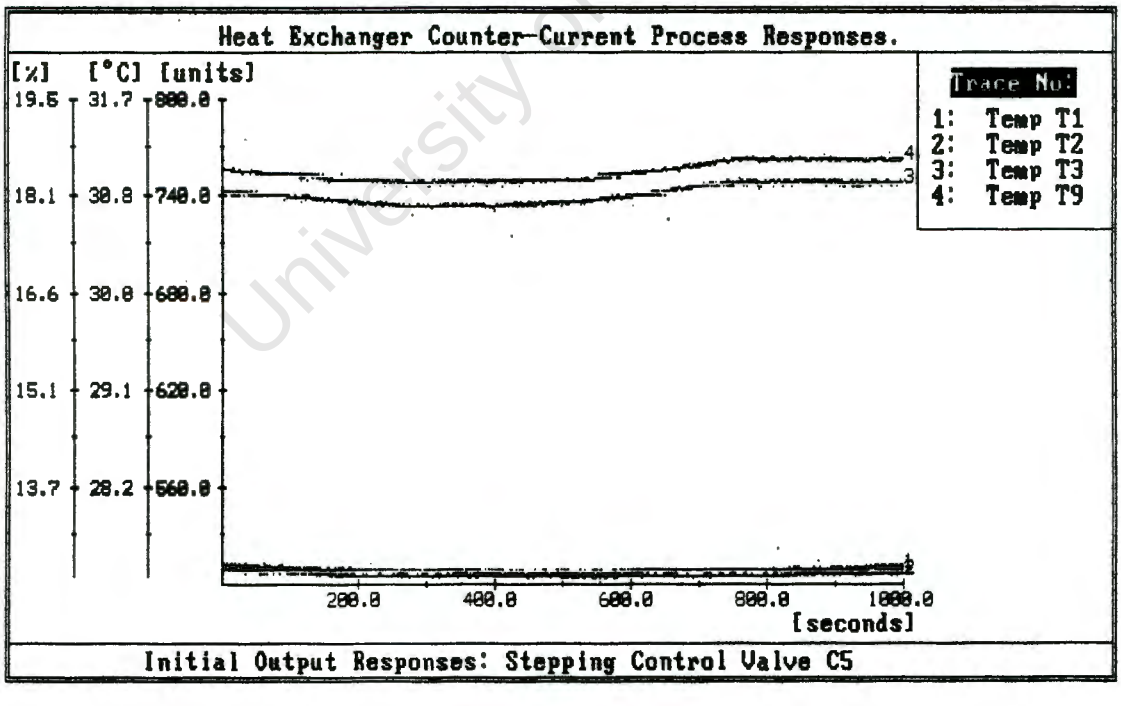


Figure H24: Temp. Response to Step in Control Valve C5

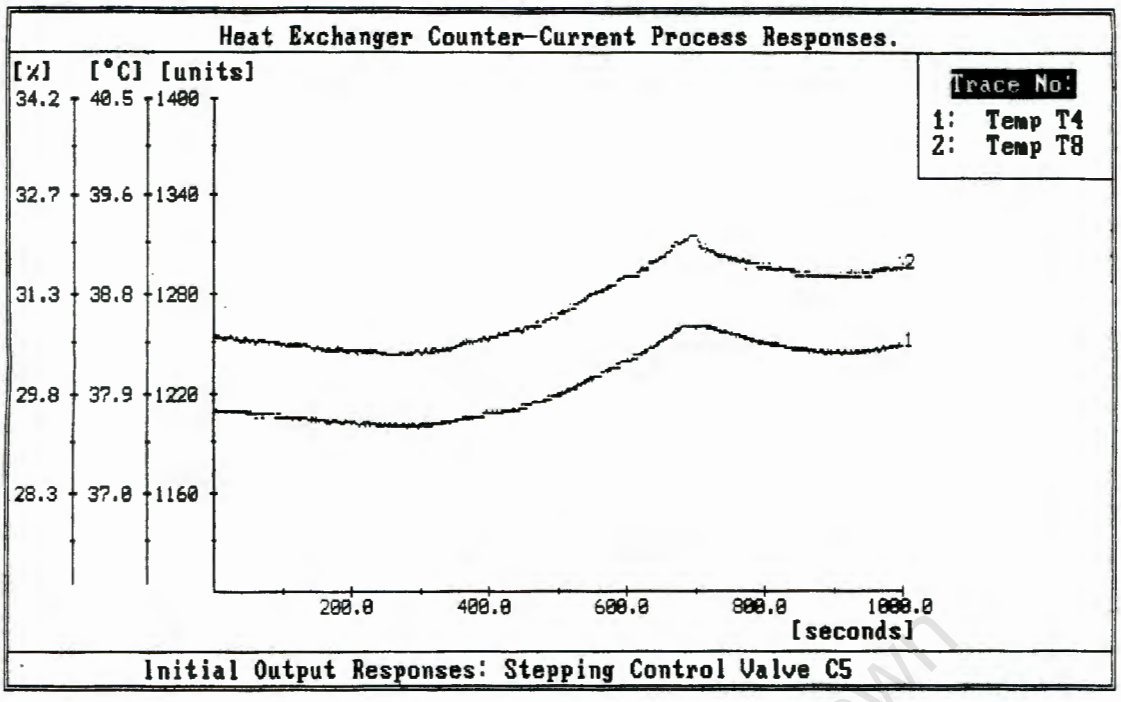


Figure H25: Temp. Response to Step in Control Valve C5

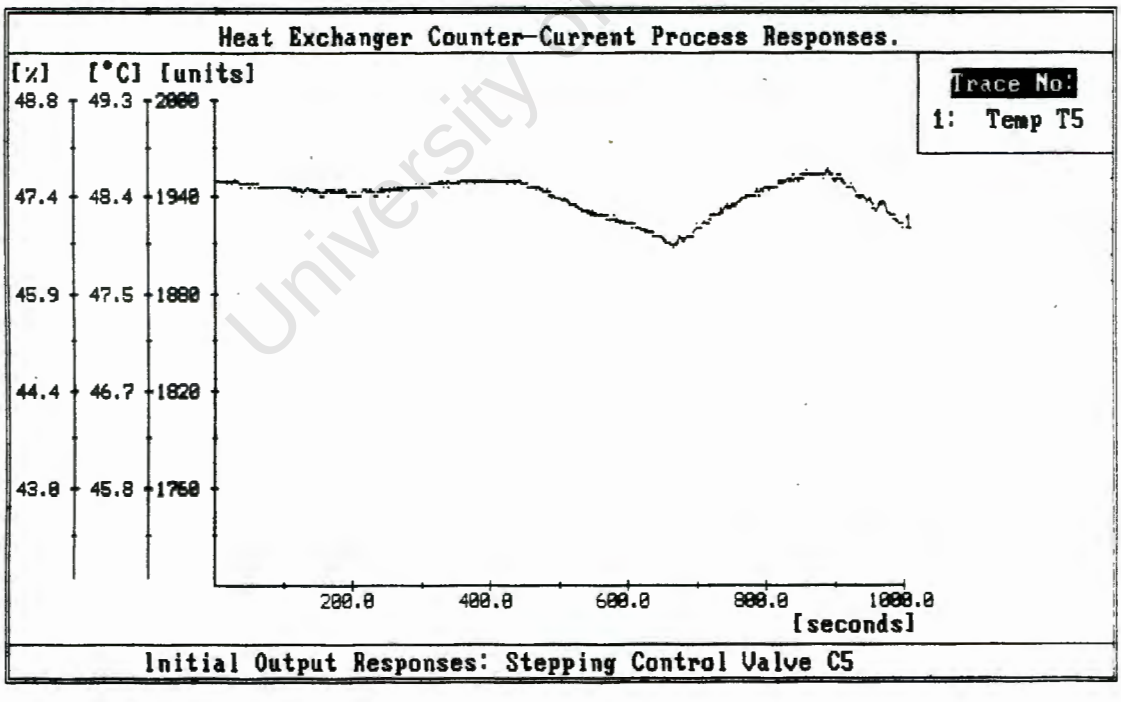


Figure H26: Temp. Response to Step in Control Valve C5

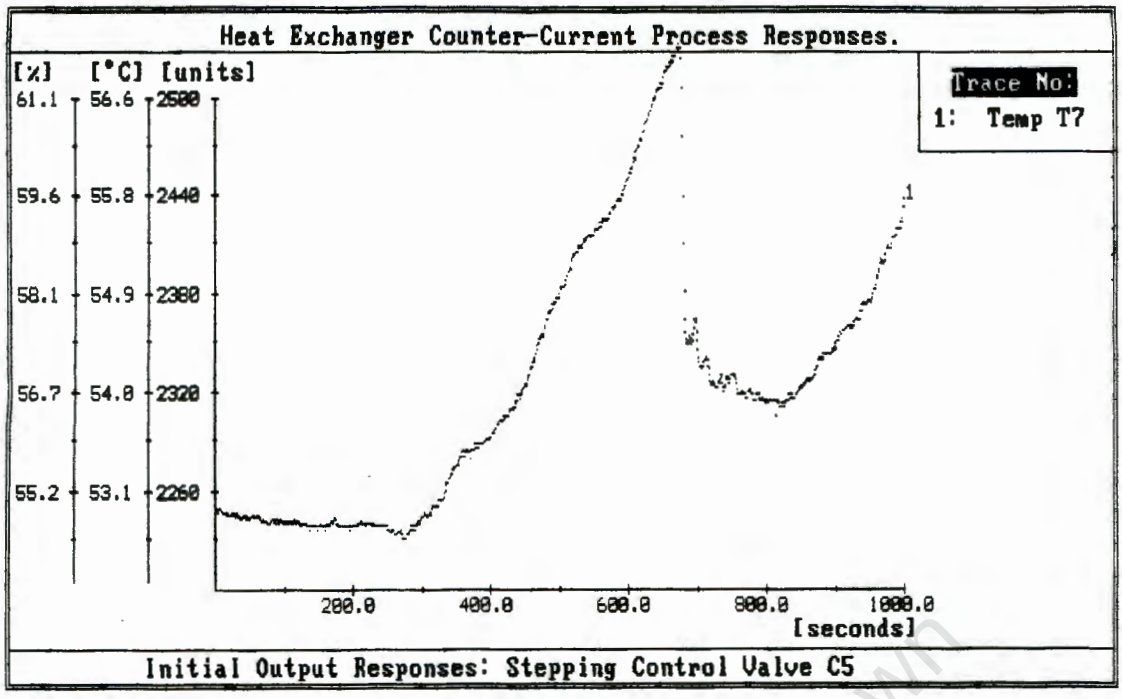


Figure H27: Temp. Response to Step in Control Valve C5

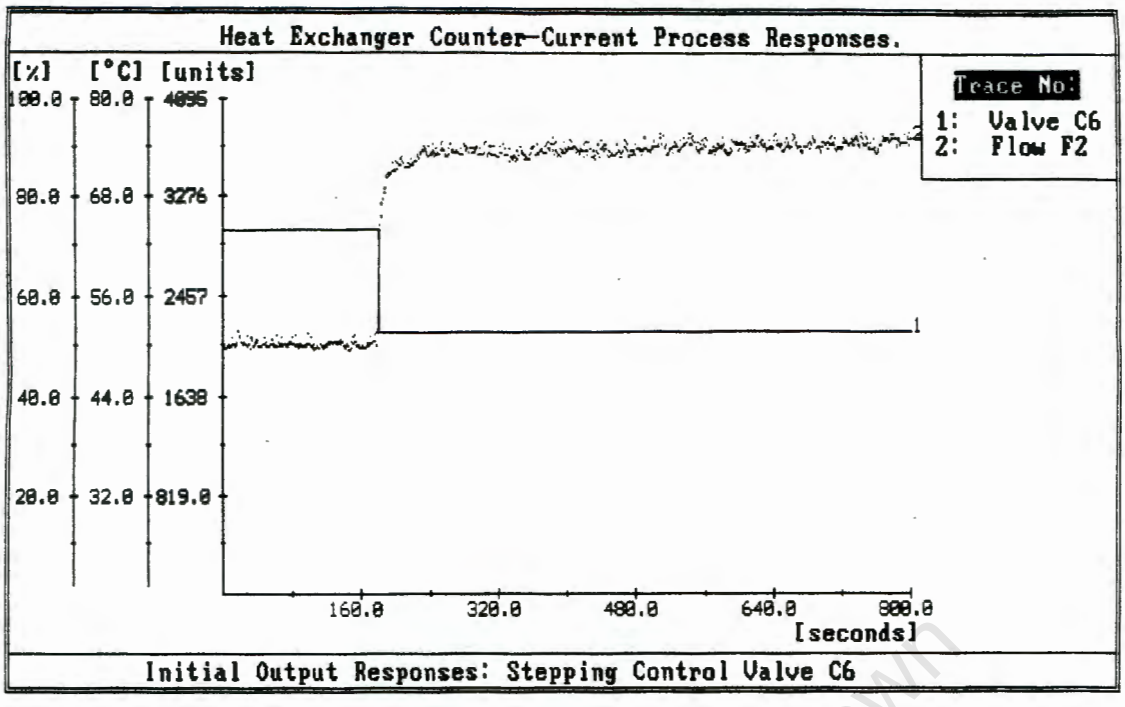


Figure H28: Flow Response to Step in Control Valve C6

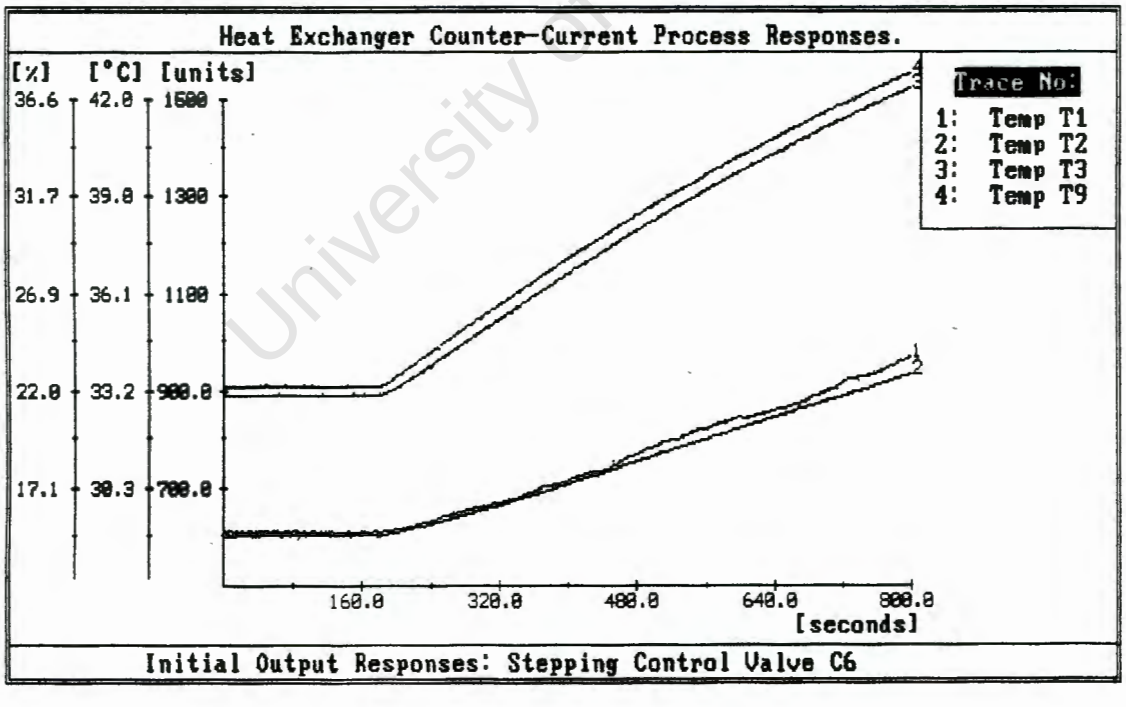


Figure H29: Temp. Response to Step in Control Valve C6

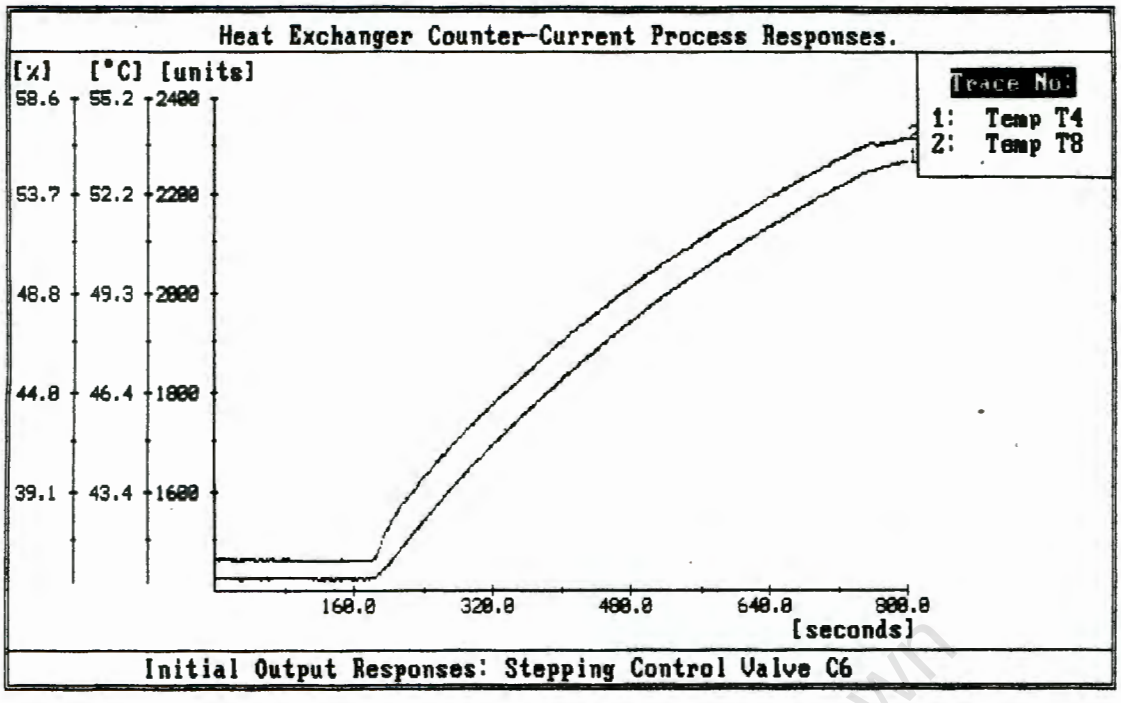


Figure H30: Temp. Response to Step in Control Valve C6

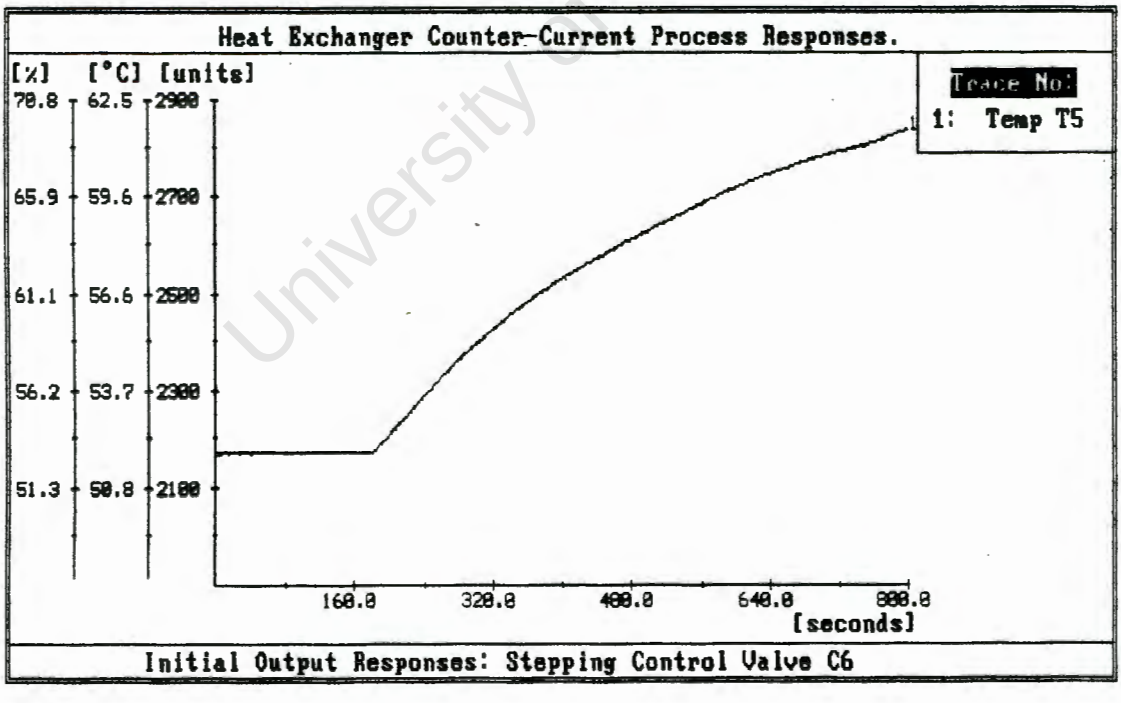


Figure H31: Temp. Response to Step in Control Valve C6

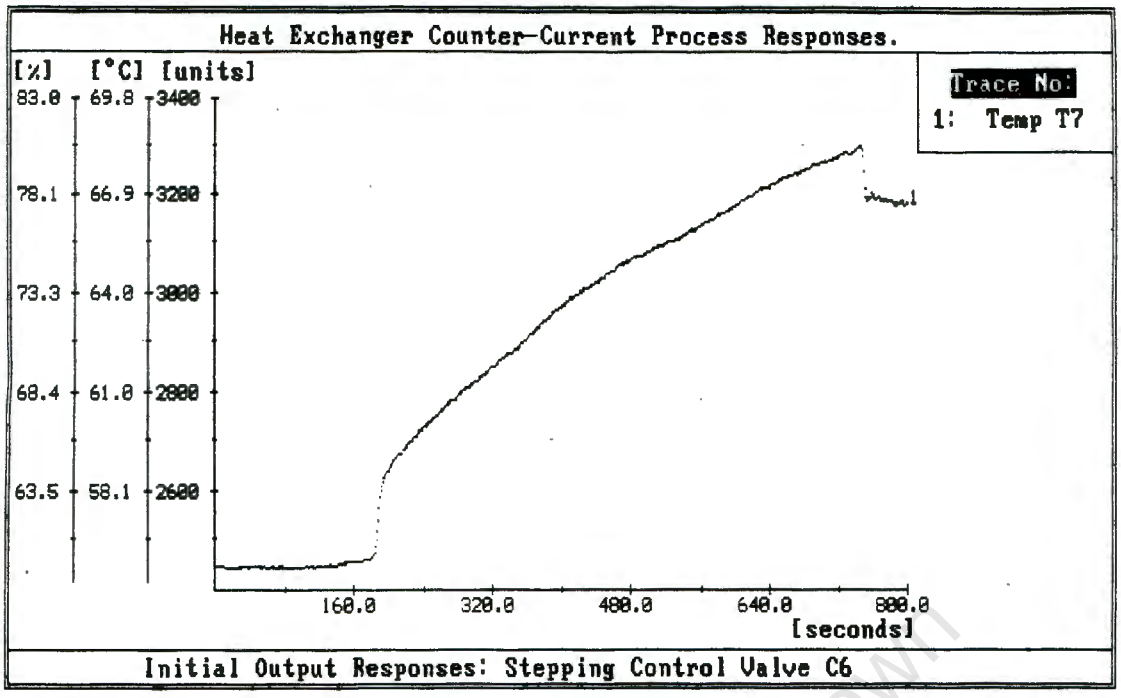


Figure H32: Temp. Response to Step in Control Valve C6

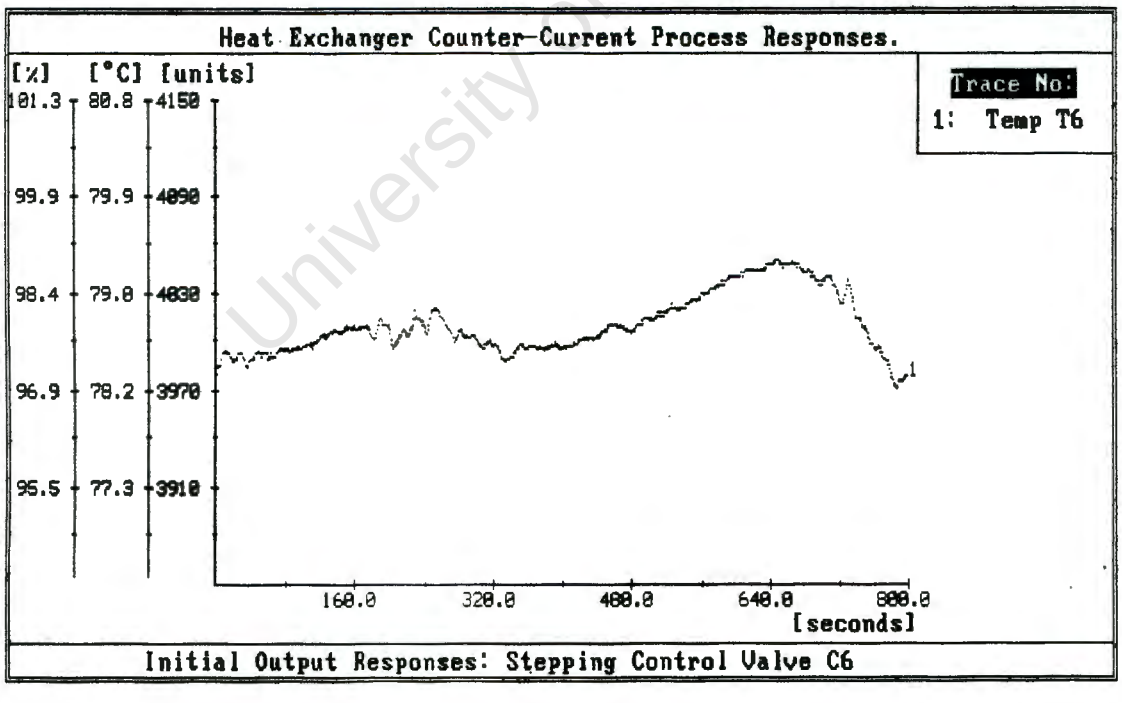


Figure H33: Temp. Response to Step in Control Valve C6

APPENDIX I

MODEL $G_c(s)$: VALVES INPUTS TO FLOW AND LEVEL OUTPUTS

In this appendix, the diagonal elements of $G_c(s)$ relating the control valve inputs to the appropriate flow and level outputs are determined.

Plots of the appropriate output responses to the stepping of each input are shown in figures I1 to I12. The transfer functions characterizing each response are also evaluated and quoted with each plot.

Each transfer function element to be determined is indicated in table I1, together with the input stepped and output response observed in determining the element. The table also assigns a label to each plot.

Element $gc_{ij}(s)$	Input Stepped	Output Observed	Figure Number
$gc_{11}(s)$	C1(s)	F1(s)	I1
$gc_{11}(s)$	C1(s)	F1(s)	I2
$gc_{22}(s)$	C2(s)	L4(s)	I3
$gc_{33}(s)$	C3(s)	L3(s)	I4
$gc_{33}(s)$	C3(s)	L3(s)	I5
$gc_{44}(s)$	C4(s)	L2(s)	I6
$gc_{55}(s)$	C5(s)	L1(s)	I7
$gc_{55}(s)$	C5(s)	L1(s)	I8
$gc_{55}(s)$	C5(s)	L1(s)	I9
$gc_{55}(s)$	C5(s)	L1(s)	I10
$gc_{66}(s)$	C6(s)	F2(s)	I11
$gc_{66}(s)$	C6(s)	F2(s)	I12

I.1) Comments on Test Data and Modelling Approach

I.1.1) Bad Data

The level response data presented in figures I7, I8 and I9 reveal an anomaly. The irregular data points occur when the level in tank T1 drops to below 40%. These points are due to splashing when the impeller of stirrer S1 is no longer submerged deeply enough beneath the surface of the water in the tank. The points are thus not a true reflection of the actual level in the tank, and are neglected as bad measurement data. Once the level has dropped to below 20%, the impeller emerges from the surface of the water, and no longer has an effect on the level measurement. Data points indicating a level of below 20% can once again be regarded a a true representation of the level in the tank.

The level in any tank is usually kept above 50%, so the effects of splashing will not normally present a problem. If, however it is required that the levels should be kept below 50%, the problem of stirrer splashing will have to be addressed. Possible solutions to this problem are listed below:

(a) Modify shape of stirrer impeller so as to reduce splashing effects.

(b) Turn stirrers off as soon as the level measurement drops to below 50%.

I.1.2) Neglecting of High Frequency Dynamics

Figures I1, I2, I11 and I12 show that the flow responses have high frequency sinusoidal components superimposed on their dominant first order response curves. These components are neglected in the modelling of the flow response as they are minor components of the overall response curve and it is aimed to keep the transfer function models as simple as possible.

I.2) Presentation and Analysis of Response Data

I.2.1) Stepping Valve C1, Observing Response of Flow F1

The open loop response of flow output F1 to a step change in the control valve input C1 is shown in figure I1

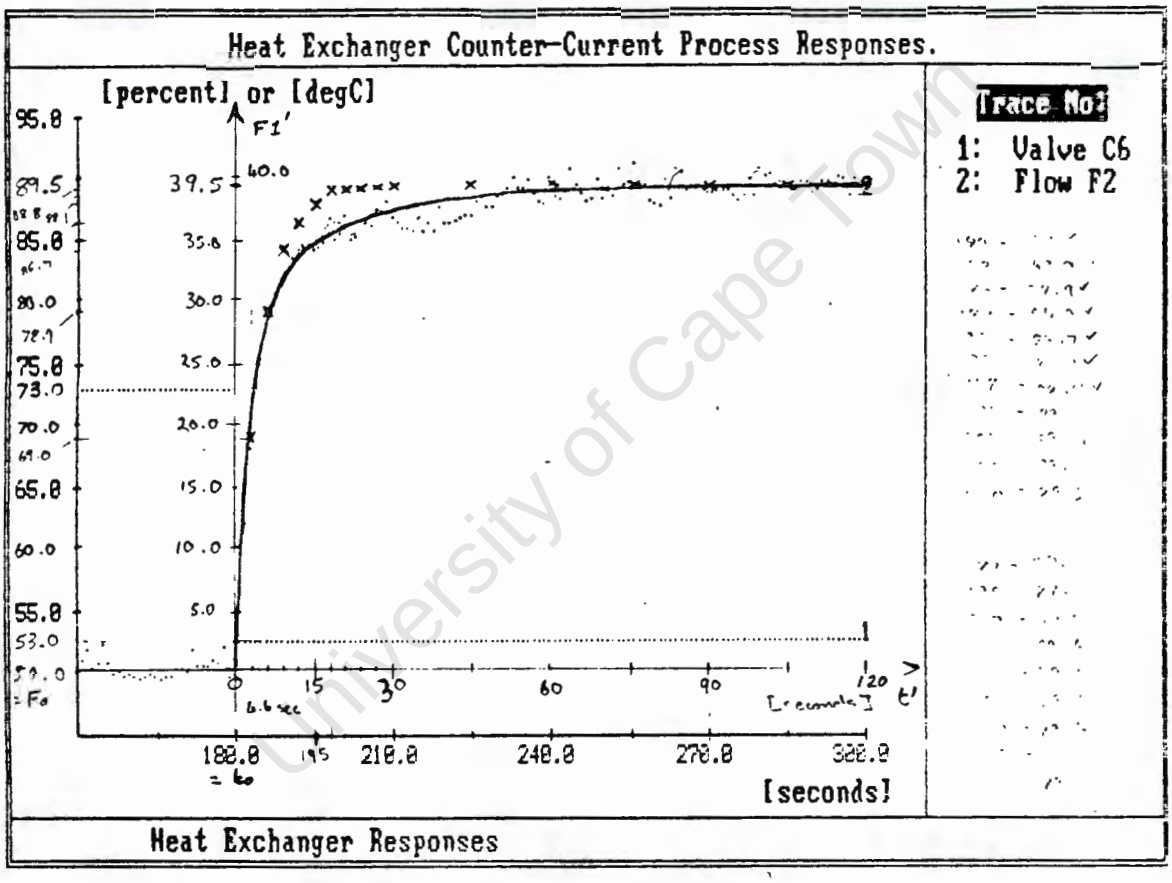


Figure I1: Open Loop Response and Fitted Transfer Function : $gc_{11}(s)$

Transfer Function:

$$gc_{11}(s) = \frac{-1.98}{1 + 4.6s} \qquad \text{Delay} = 0.0 \text{ sec}$$

I.2.1) Continued...

The open loop response of flow output F1 to a step change in the control valve input C1 is shown in figure I2

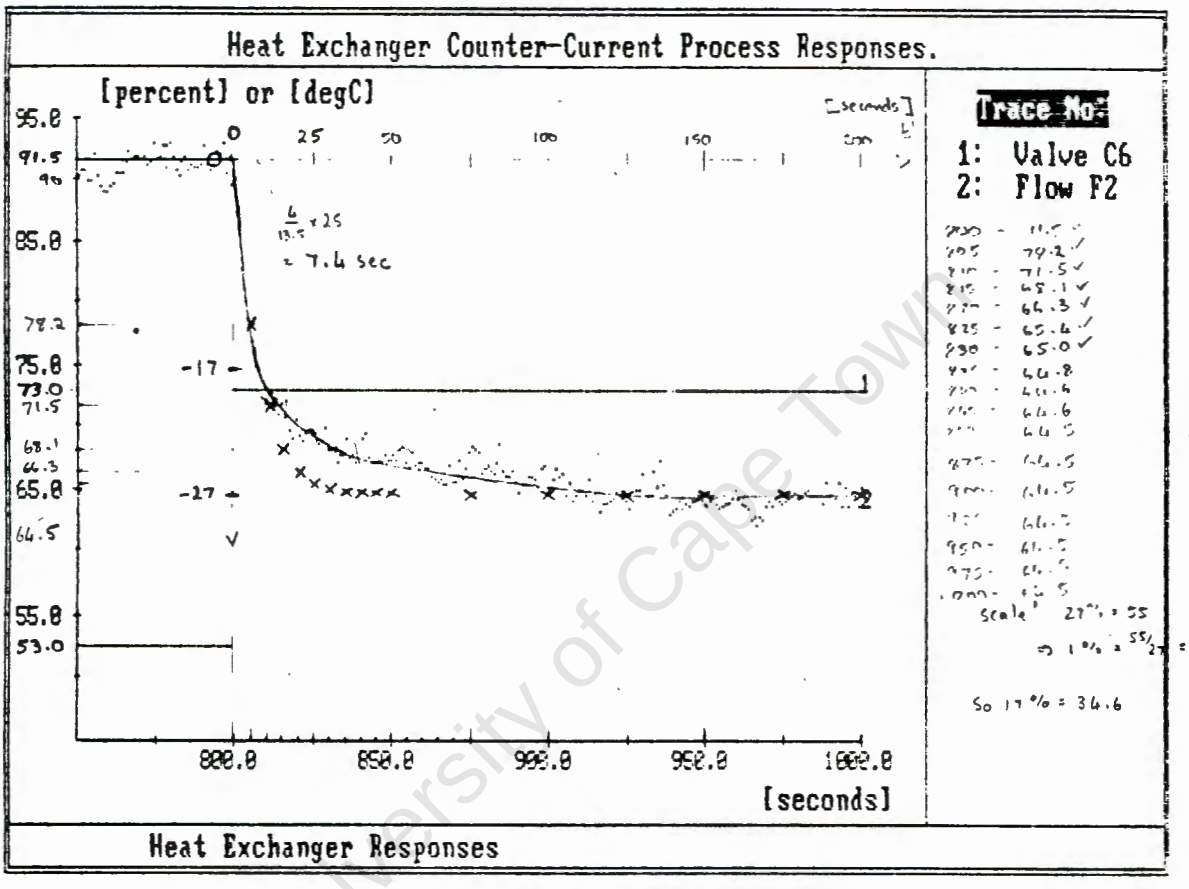


Figure I2: Open Loop Response and Fitted Transfer Function : $gc_{11}(s)$

Transfer Function:

$$gc_{11}(s) = \frac{-1.35}{1 + 7.4s} \qquad \text{Delay} = 0.0 \text{ sec}$$

I.2.2) Stepping Valve C2, Observing Response of Level L4

The open loop response of level output L4 to a step change in the control valve input C2 is shown in figure I3

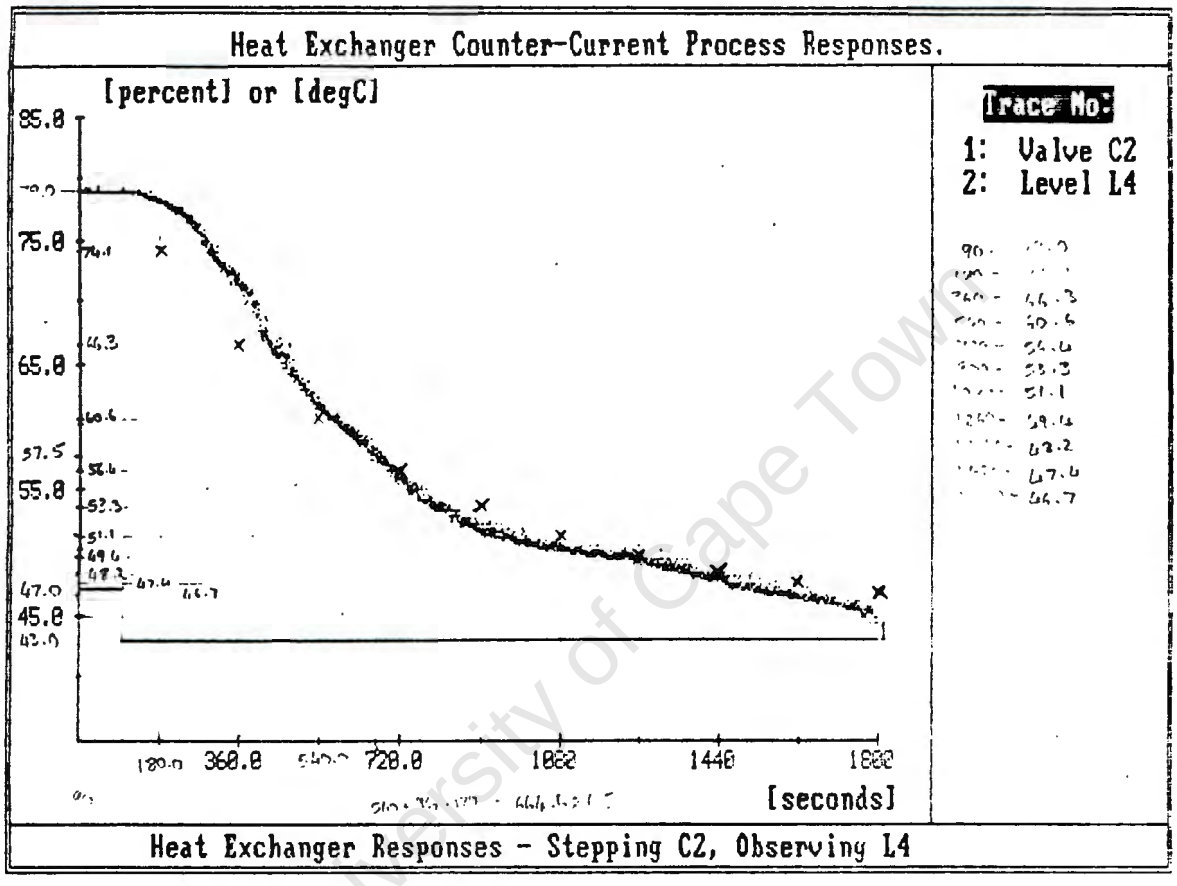


Figure I3: Open Loop Response and Fitted Transfer Function : $gc_{22}(s)$

Transfer Function:

$$gc_{22}(s) = \frac{8.5}{1 + 575s} \qquad \text{Delay} = 0.0 \text{ sec}$$

I.2.3) Stepping Valve C3, Observing Response of Level L3

The open loop response of level output L3 to a step change in the control valve input C3 is shown in figure I4

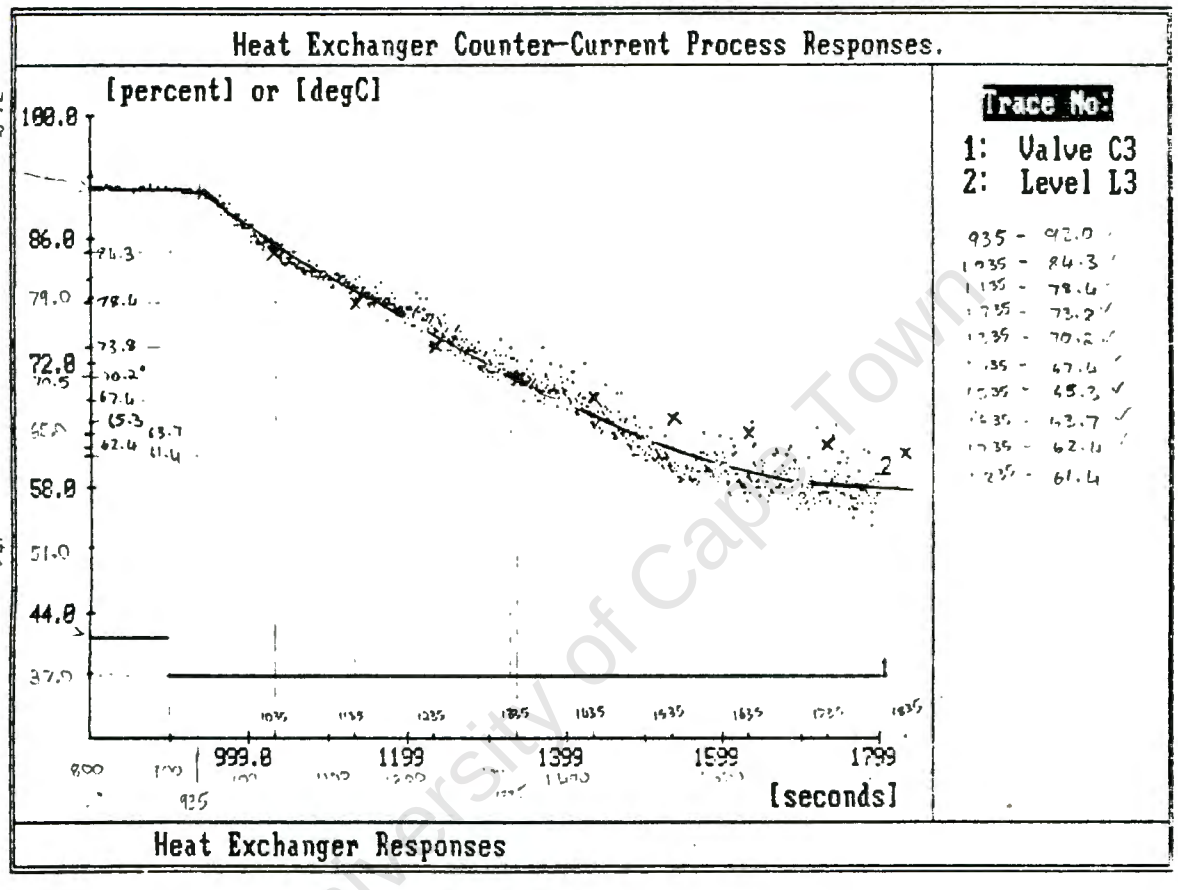


Figure I4: Open Loop Response and Fitted Transfer Function : $gc_{33}(s)$

Transfer Function:

$$gc_{33}(s) = \frac{8.5}{1 + 390s} \qquad \text{Delay} = 20.0 \text{ sec}$$

I.2.3) Continued...

The open loop response of level output L3 to a step change in the control valve input C3 is shown in figure I5

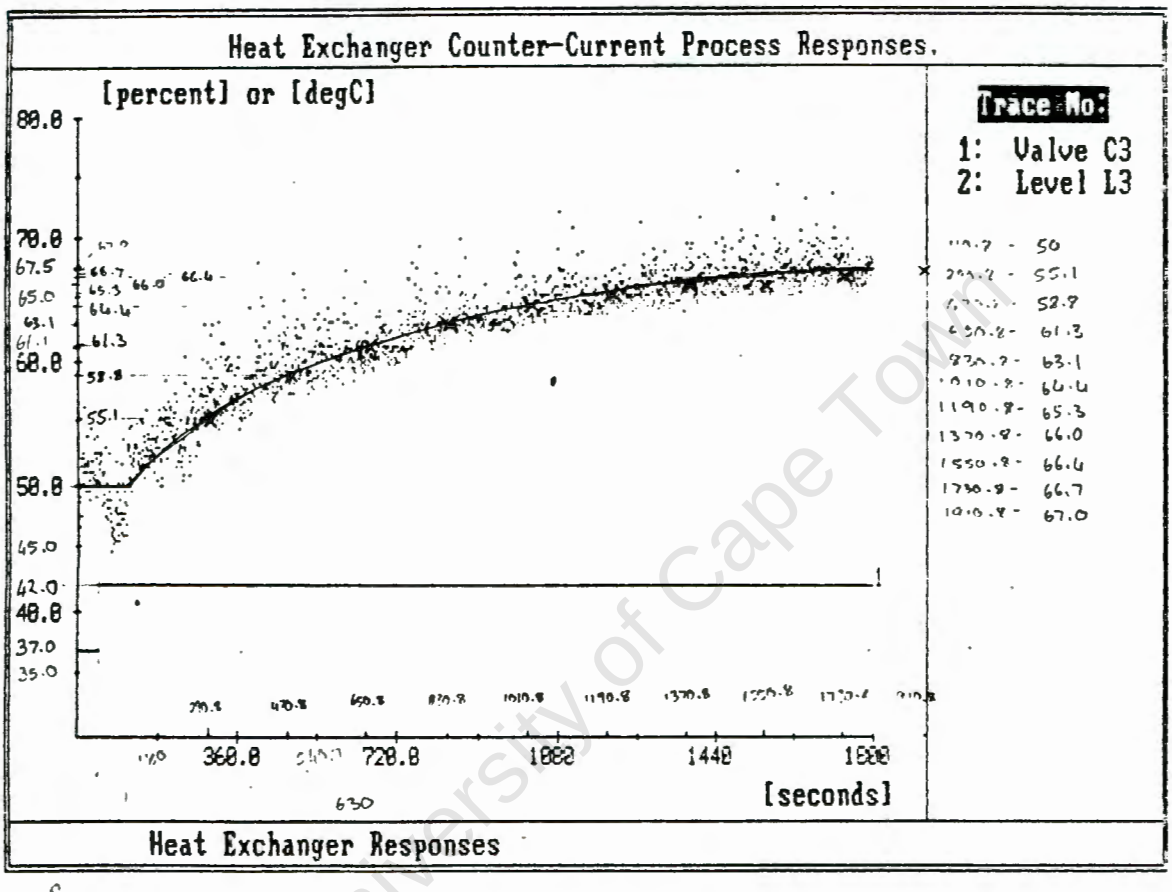


Figure I5: Open Loop Response and Fitted Transfer Function : $gc_{33}(s)$

Transfer Function:

$$gc_{33}(s) = \frac{3.5}{1 + 519s} \qquad \text{Delay} = 10.0 \text{ sec}$$

I.2.4) Stepping Valve C4, Observing Response of Flow L2

The open loop response of level output L2 to a step change in the control valve input C4 is shown in figure I6

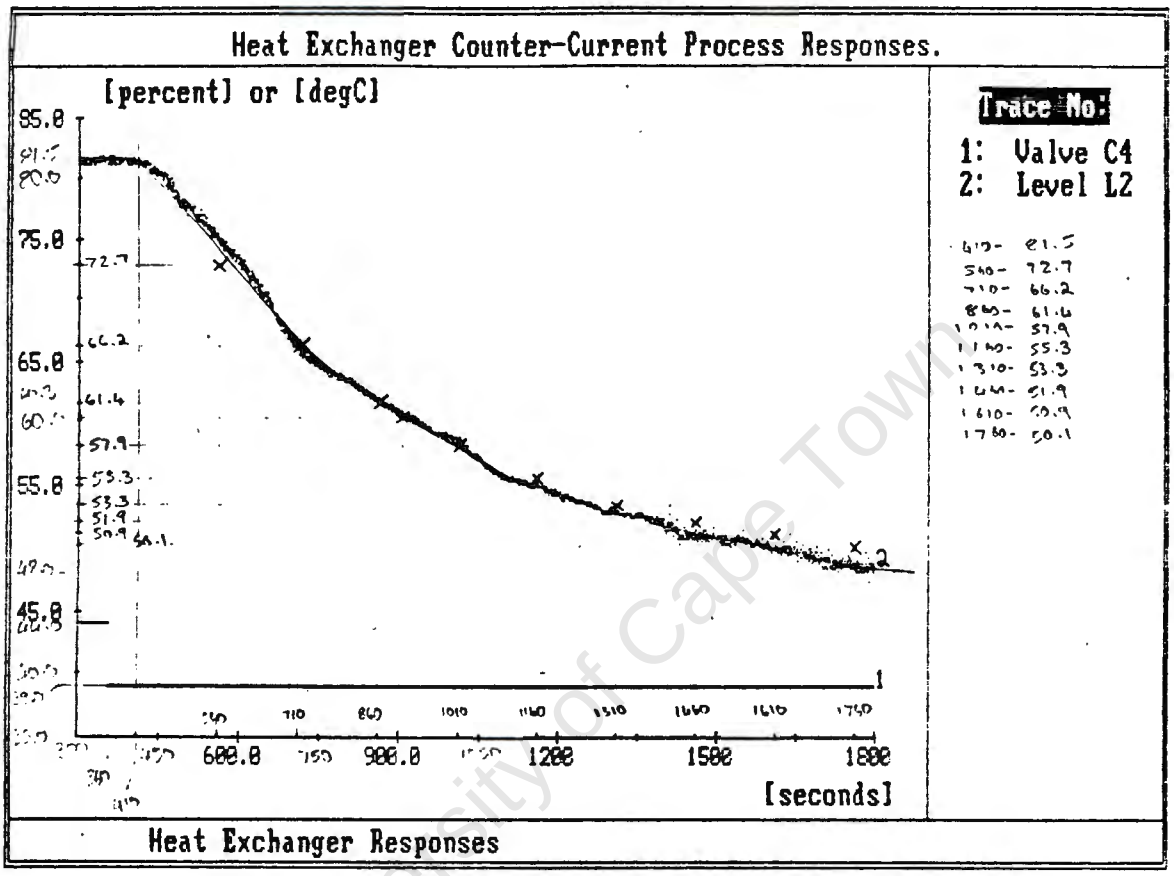


Figure I6: **Open Loop Response and Fitted Transfer Function : $gc_{44}(s)$**

Transfer Function:

$$gc_{44}(s) = \frac{6.7}{1 + 490s} \qquad \text{Delay} = 10.0 \text{ sec}$$

I.2.5) Continued...

The open loop response of level output F1 to a step change in the control valve input C5 is shown in figure I8

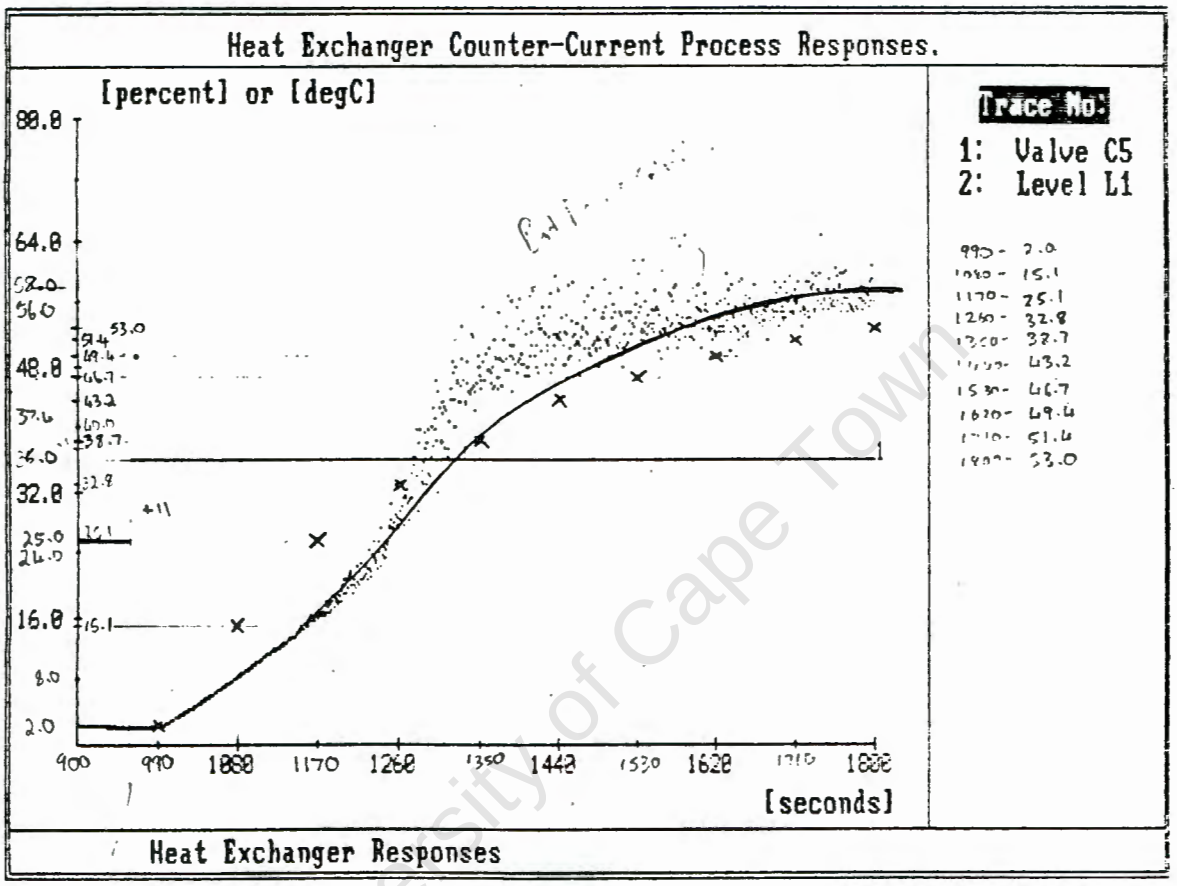


Figure I8: Open Loop Response and Fitted Transfer Function : $gc_{55}(s)$

Transfer Function:

$$gc_{55}(s) = \frac{5.1}{1 + 339s} \qquad \text{Delay} = 35.0 \text{ sec}$$

I.2.5) Continued...

The open loop response of level output F1 to a step change in the control valve input C5 is shown in figure I9

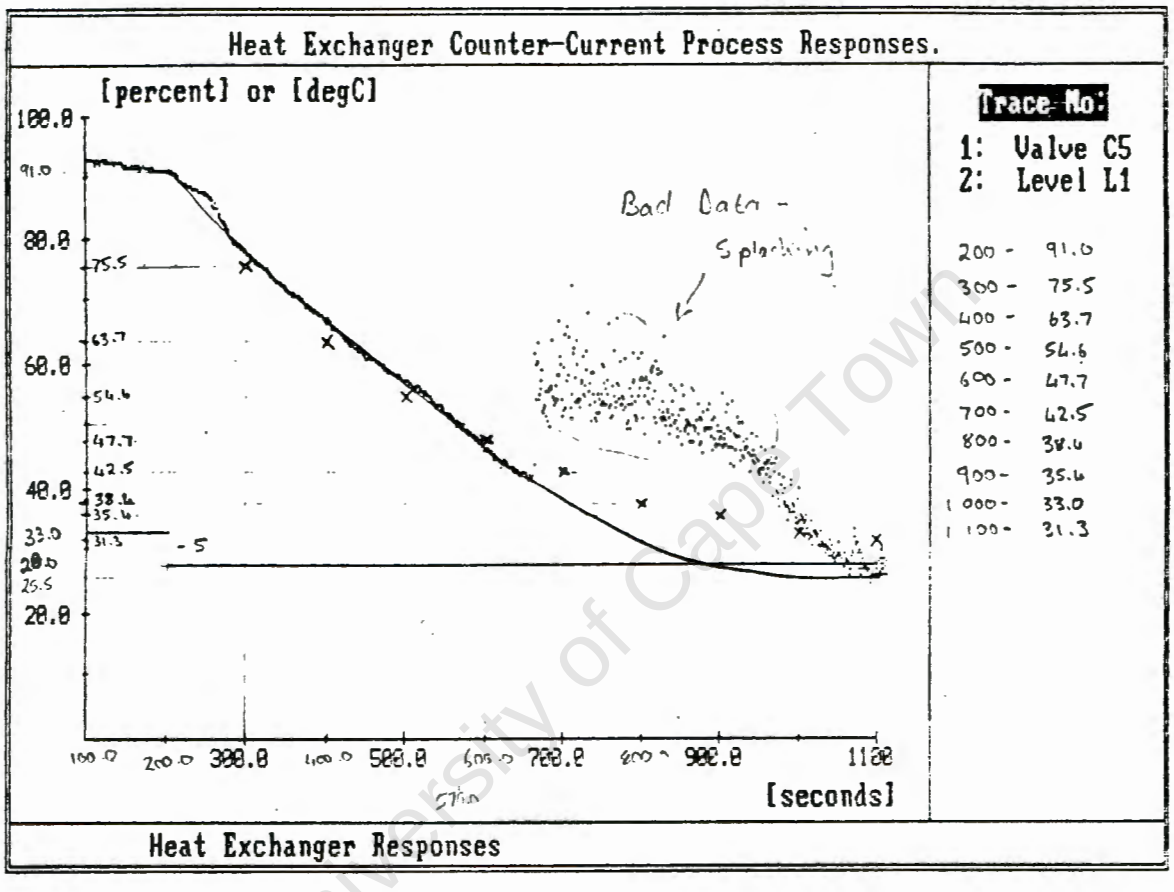


Figure I9: **Open Loop Response and Fitted Transfer Function : $gc_{55}(s)$**

Transfer Function:

$$gc_{55}(s) = \frac{13.1}{1 + 370s} \qquad \text{Delay} = 13.1 \text{ sec}$$

I.2.5) Continued

The open loop response of level output F1 to a step change in the control valve input C5 is shown in figure I10

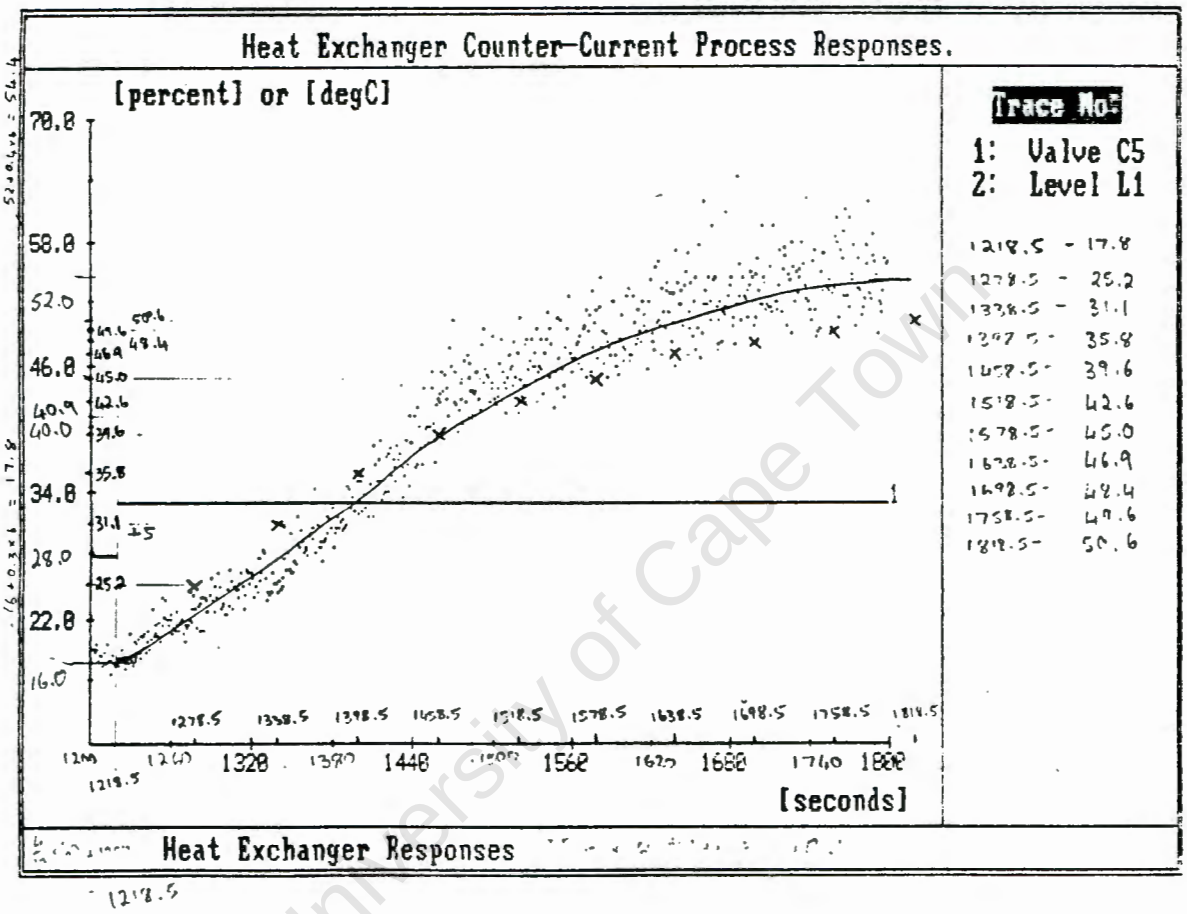


Figure I10: Open Loop Response and Fitted Transfer Function : $gc_{55}(s)$

Transfer Function:

$$gc_{55}(s) = \frac{7.32}{1 + 265s} \qquad \text{Delay} = 0.0 \text{ sec}$$

I.2.6) Stepping Valve C6, Observing Response of Flow F2

The open loop response of flow output F2 to a step change in the control valve input C6 is shown in figure I11.

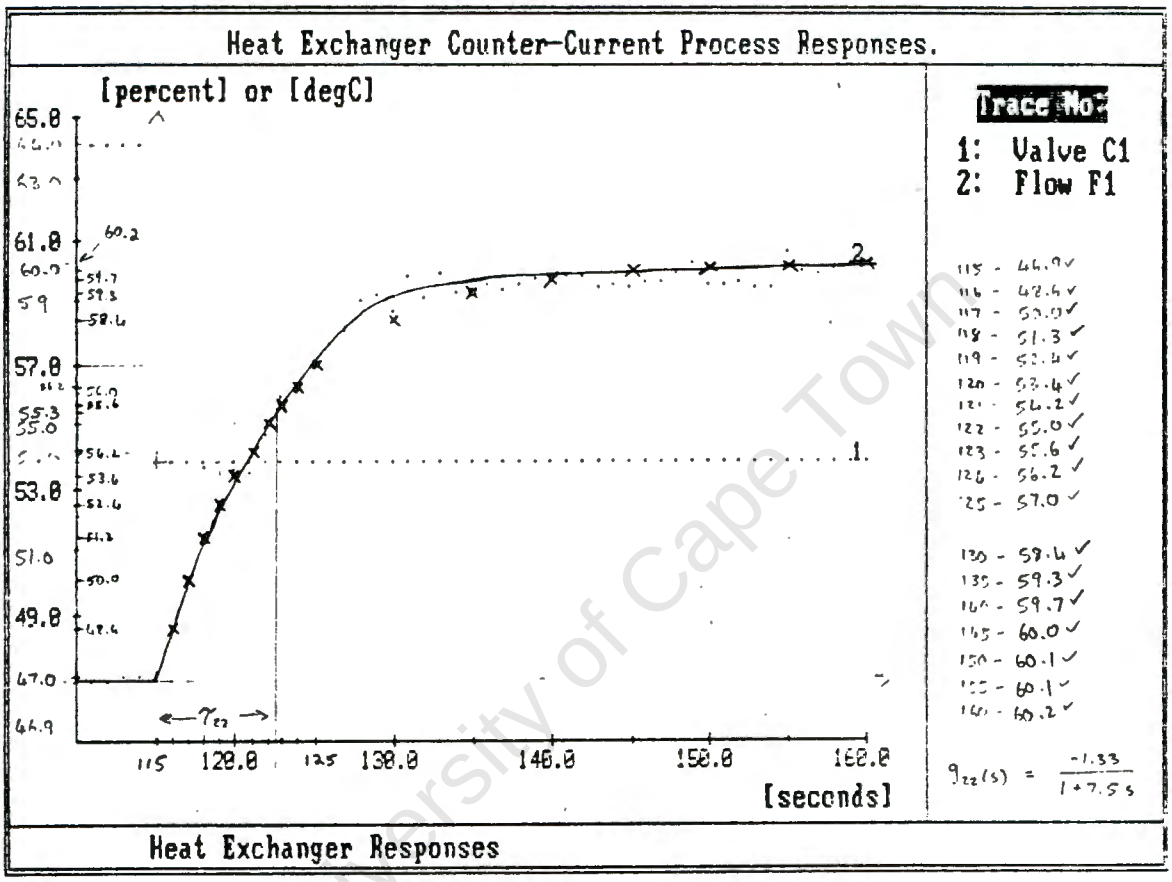


Figure I11: Open Loop Response and Fitted Transfer Function : $gc_{66}(s)$

Transfer Function:

$$gc_{66}(s) = \frac{-1.33}{1 + 7.5s} \qquad \text{Delay} = 0.0 \text{ sec}$$

I.2.6) Continued...

The open loop response of flow output F2 to a step change in the control valve input C6 is shown in figure I12.

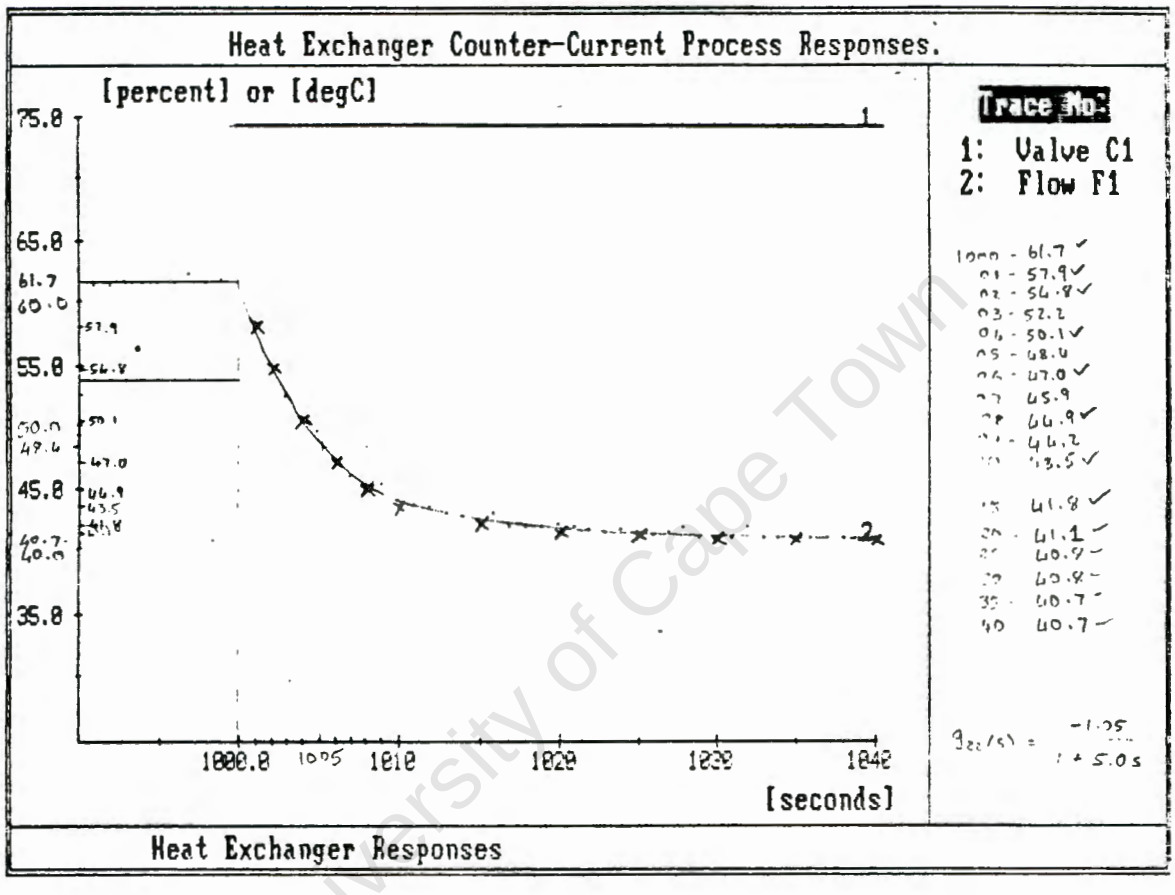


Figure I12: Open Loop Response and Fitted Transfer Function : $gc_{66}(s)$

Transfer Function:

$$gc_{66}(s) = \frac{-1.05}{1 + 5.0s} \qquad \text{Delay} = 0.0 \text{ sec}$$

APPENDIX J

DESIGN PLOTS OBTAINED FOR THE ADOPTED ELEMENTS OF $K_c(s)$

The design of PI compensator element $kc_{33}(s)$ by applying Nyquist analysis techniques on element $gc_{33}(s)$ is shown in section 3.4.2 of this dissertation. The technique for the design of the remaining elements $kc_{ij}(s)$ ($i = j = 1..6$) of $K_c(s)$ is identical. The following steps are taken in the design of each element (for $i = j = 1..6$):

(i) Simulation of Uncompensated Open Loop System

The simulated O/L output response to a unit step in the input for the appropriate transfer function element $gc_{ij}(s)$ is observed. The O/L response time and steady-state gain is observed from this plot.

(ii) Nyquist Plot for Uncompensated System

The polar nyquist plot for the uncompensated element $gc_{ij}(s)$ is then produced, from which the feedback properties for the uncompensated system are predicted.

(iii) Specification of Suitable PI Controller

An appropriate PI compensator $kc_{ij}(s)$ is decided on for $gc_{ij}(s)$, based on the nyquist plot for $gc_{ij}(s)$.

(iv) Nyquist Plot for Compensated System

The nyquist plot for the compensated element $qc_{ij}(s) = gc_{ij}(s)\ kc_{ij}(s)$ is produced. The feedback properties of the compensated process are observed from this plot.

(v) Simulation of Compensated Closed Loop System

The simulated C/L output response to a unit step in the setpoint for the appropriate transfer function element $hc_{ij}(s) = qc_{ij}(s) / (1 + qc_{ij}(s))$ is observed. The C/L response time and steady-state gain is observed from this plot.

It would be tedious to repeat the design discussion presented in section 3.4.2 for each element, as the design procedure is identical. For this reason, only the appropriate design plots are shown in this appendix for the design of each element. These plots are shown in figures J1 to J24.

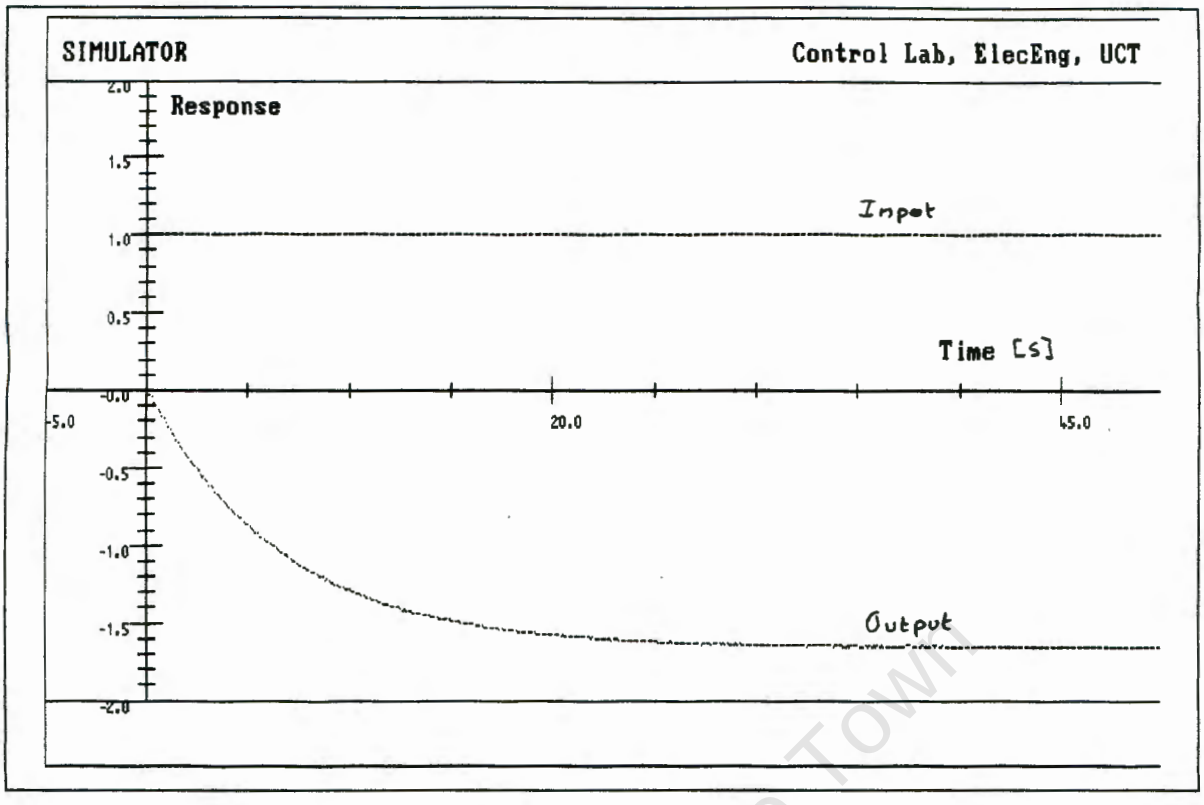


Figure J1: Simulated Open loop Time Response of Flow F1 to a Unit Step in Control Valve C1

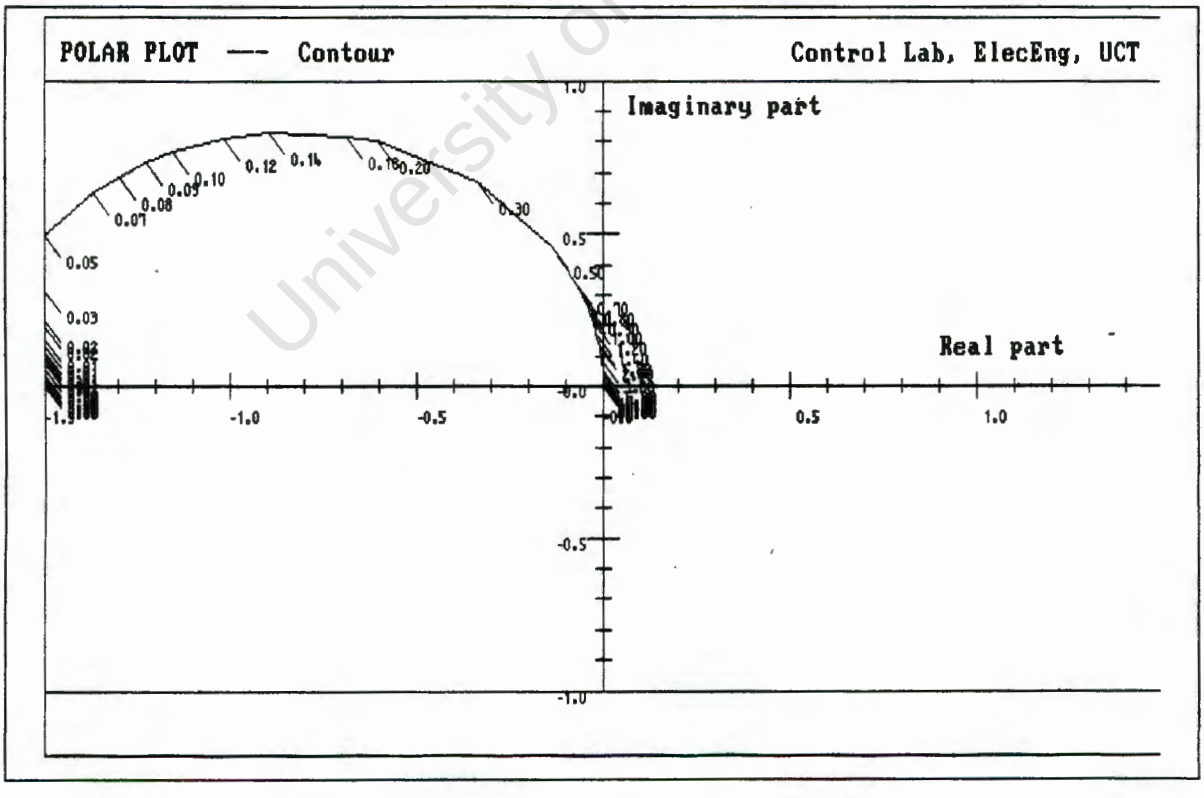


Figure J2: Nyquist Plot for element $gc_{11}(jw)$ of $G_c(s)$

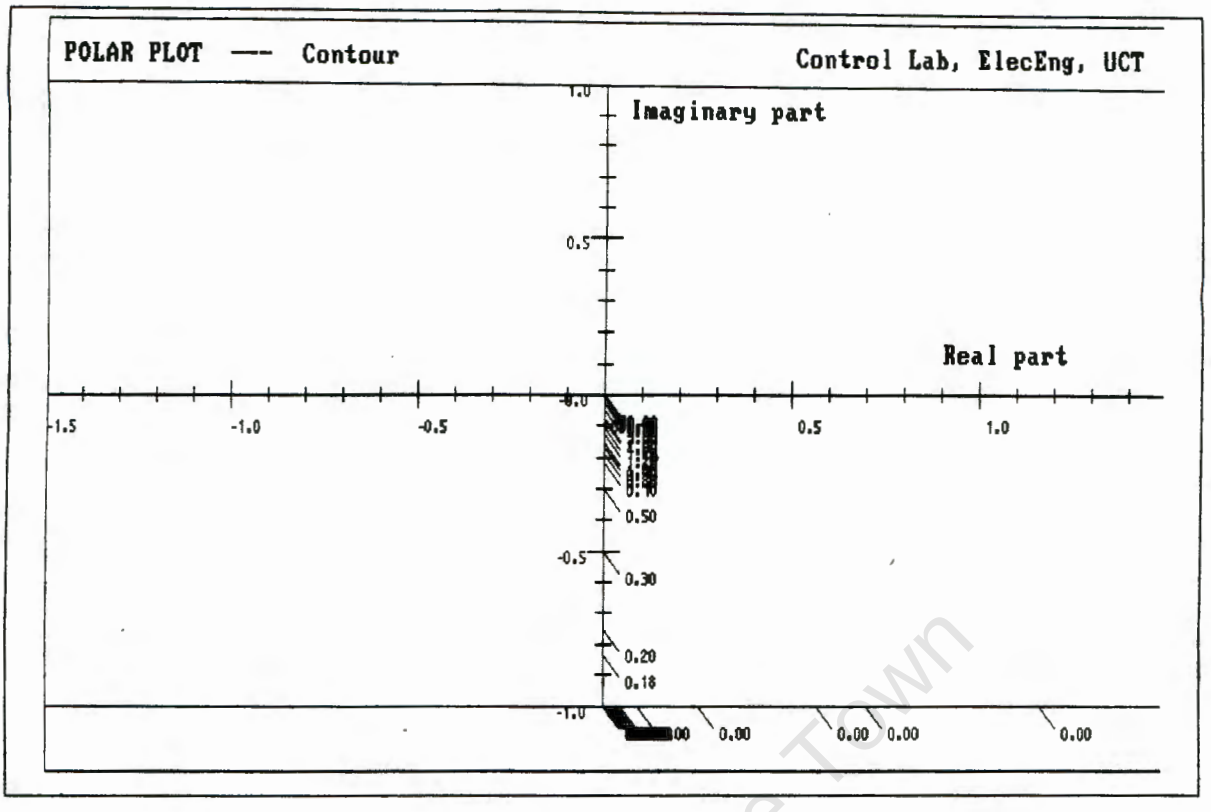


Figure J3: Nyquist Plot for Element $qc_{11}(jw)$

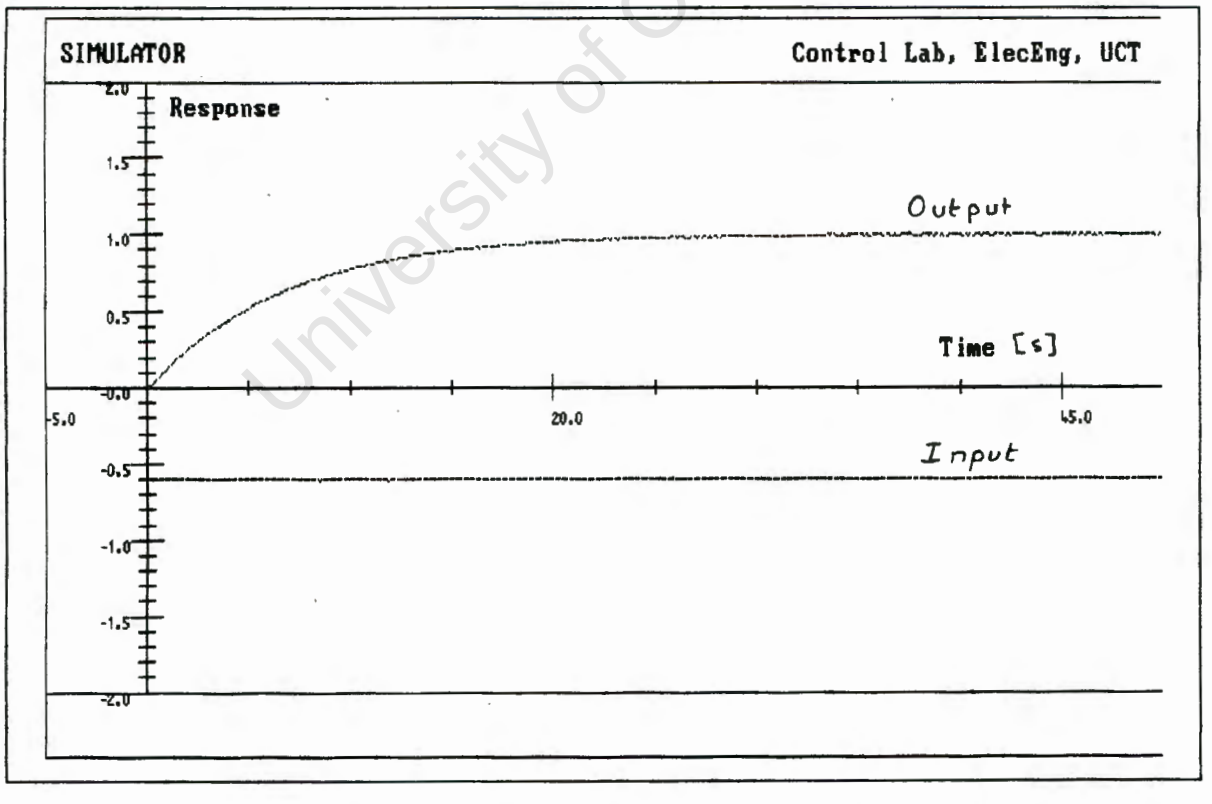


Figure J4: Simulated Closed Loop Time Response of Flow F1 to a Unity Step in its Setpoint

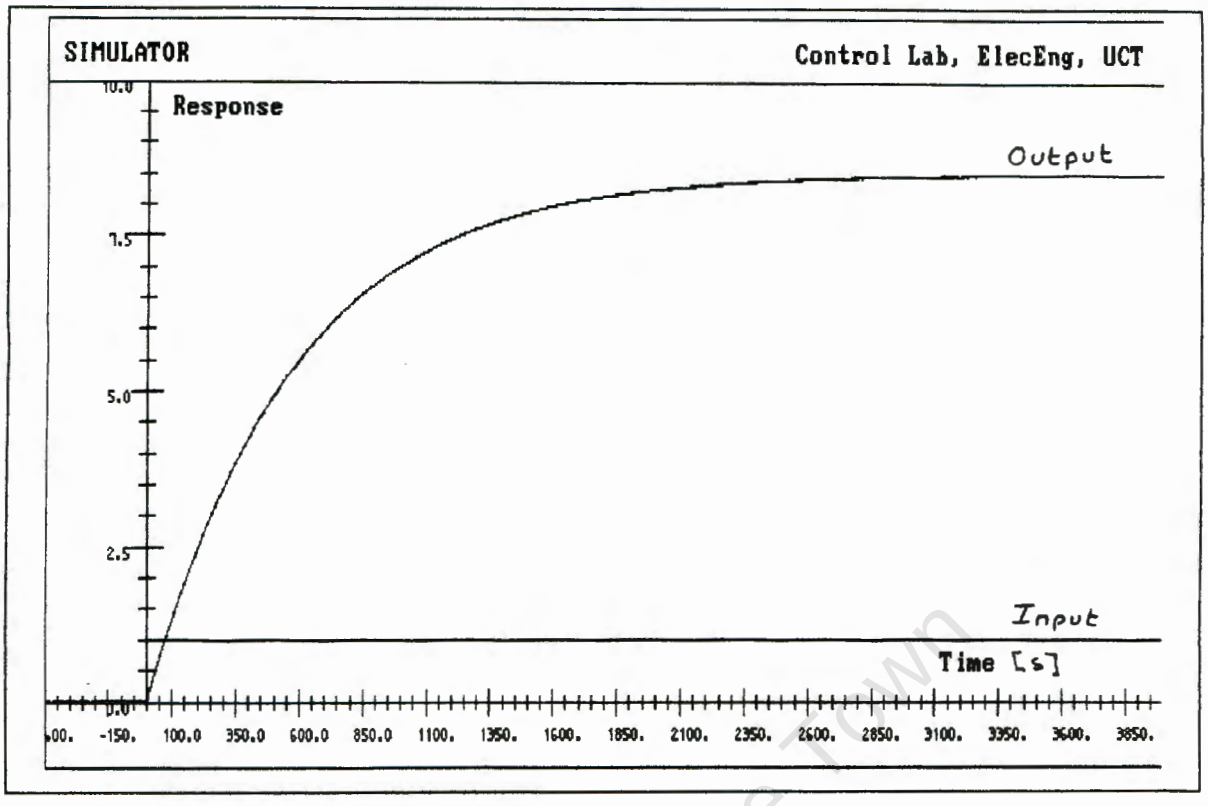


Figure J5: Simulated Open loop Time Response of Level L4 to a Unit Step in Control Valve C2

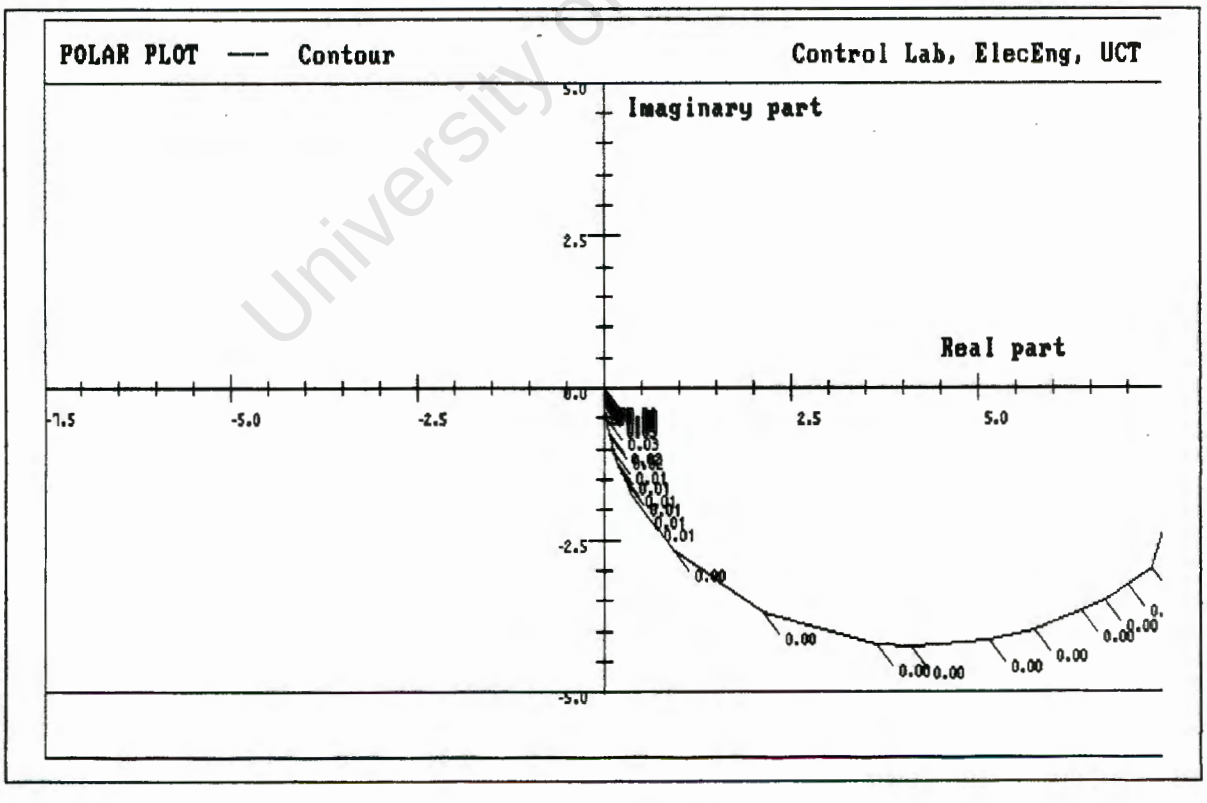


Figure J6: Nyquist Plot for element $gc_{22}(jw)$ of $G_c(s)$

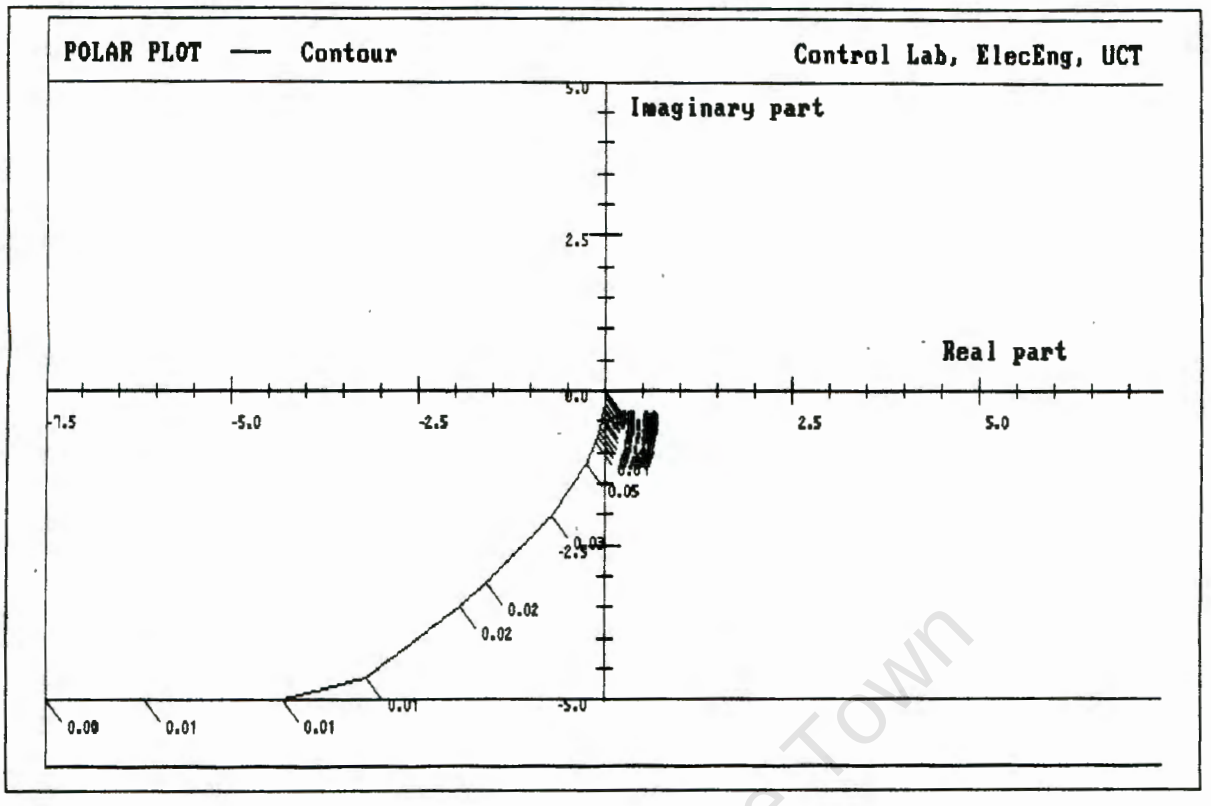


Figure J7: Nyquist Plot for Element $qc_{22}(jw)$

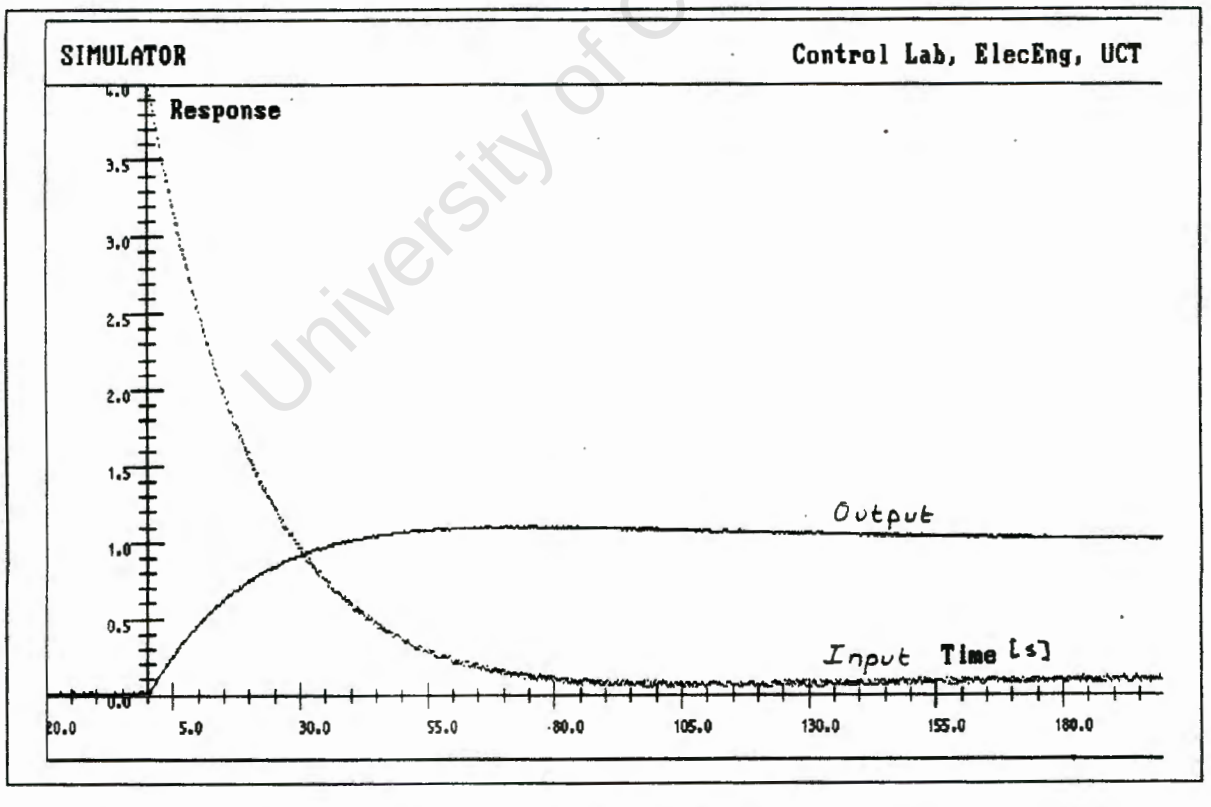


Figure J8: Simulated Closed Loop Time Response of Level L4 to a Unity Step in its Setpoint

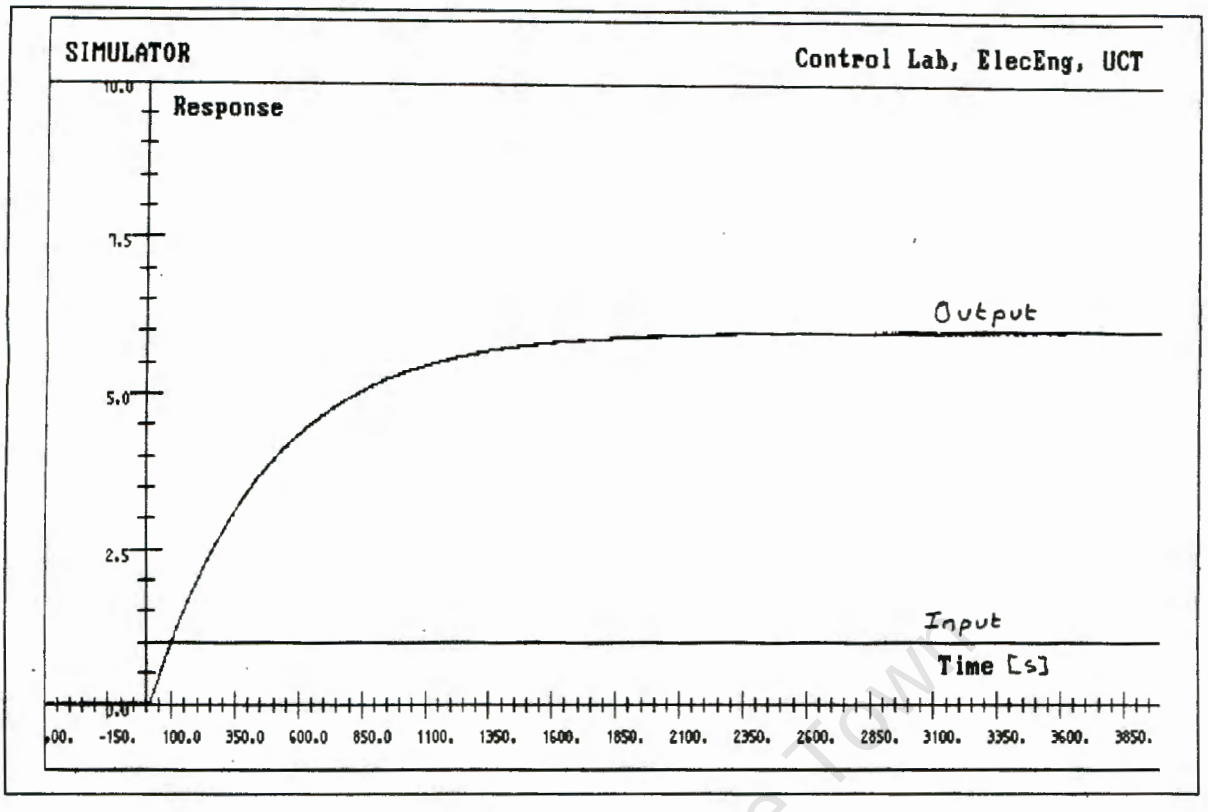


Figure J9: Simulated Open loop Time Response of Level L3 to a Unit Step in Control Valve C3

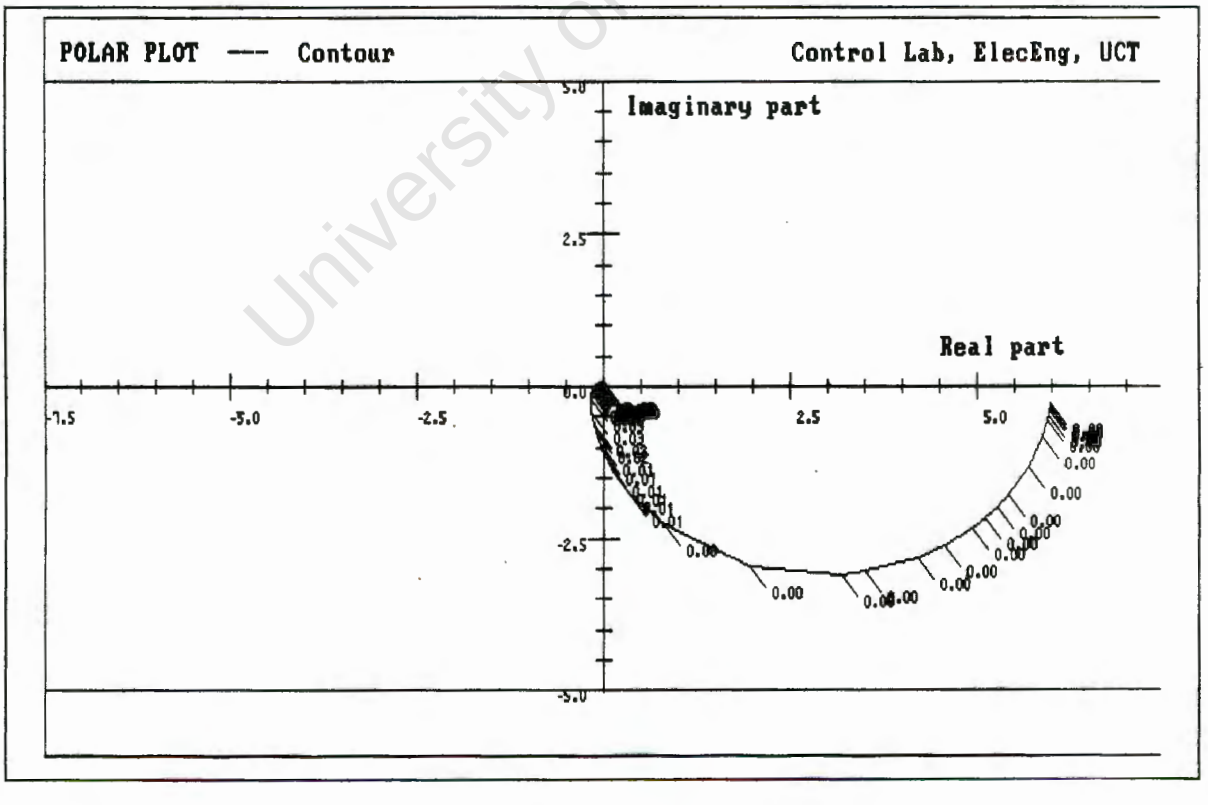


Figure J10: Nyquist Plot for element $gc_{33}(jw)$ of $G_c(s)$

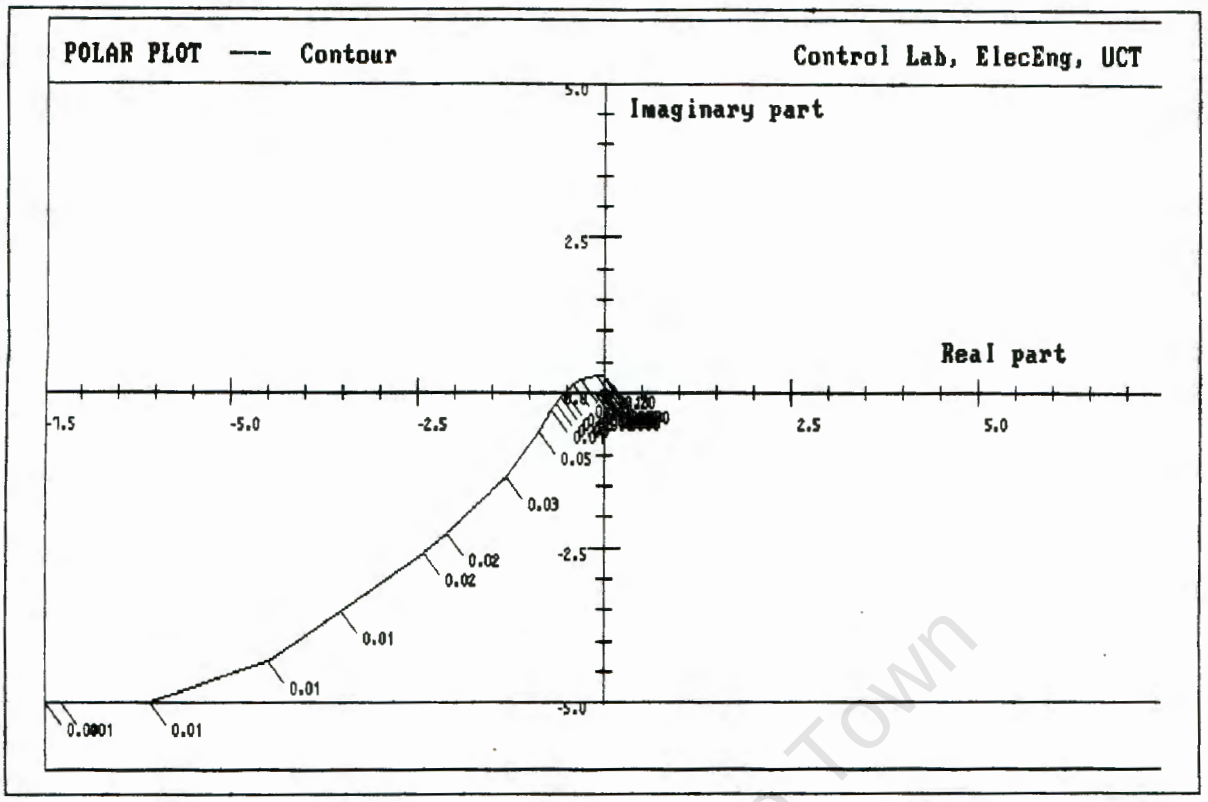


Figure J11: Nyquist Plot for Element $qc_{33}(jw)$

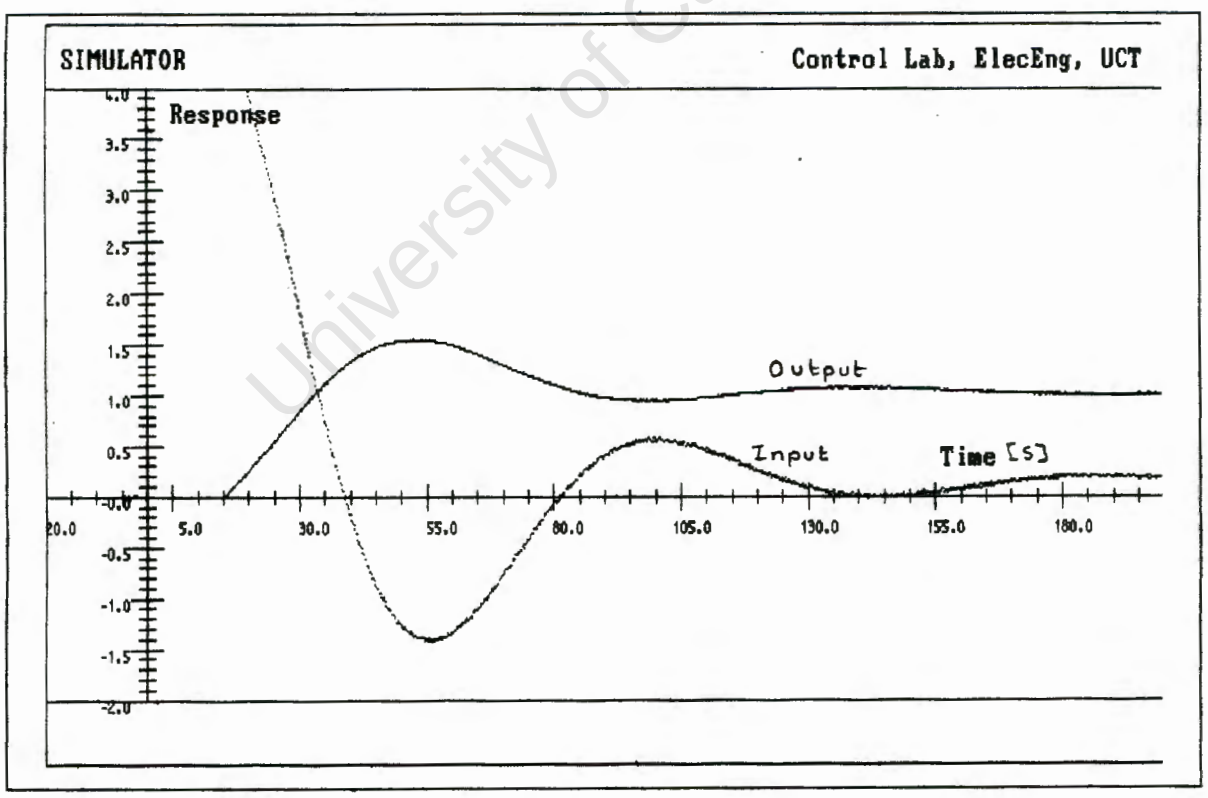


Figure J12: Simulated Closed Loop Time Response of Level L3 to a Unity Step in its Setpoint

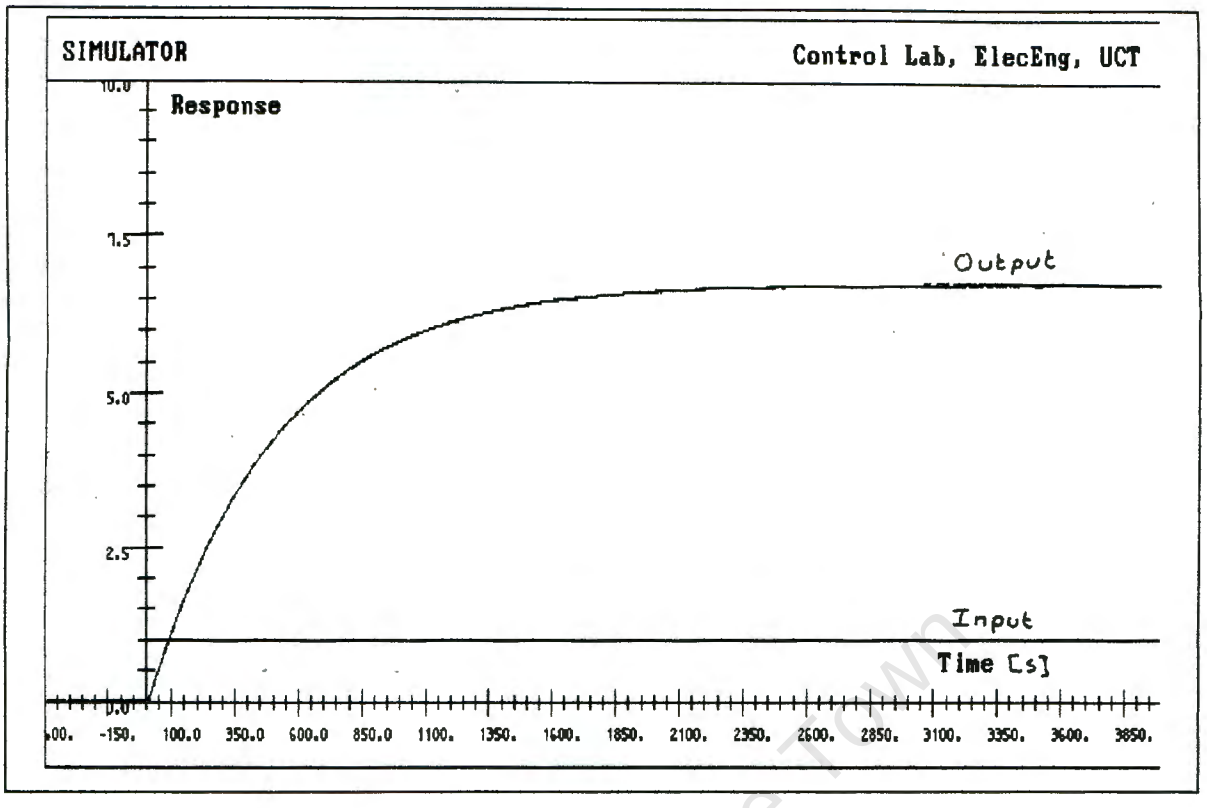


Figure J13: Simulated Open loop Time Response of Level L2 to a Unit Step in Control Valve C4

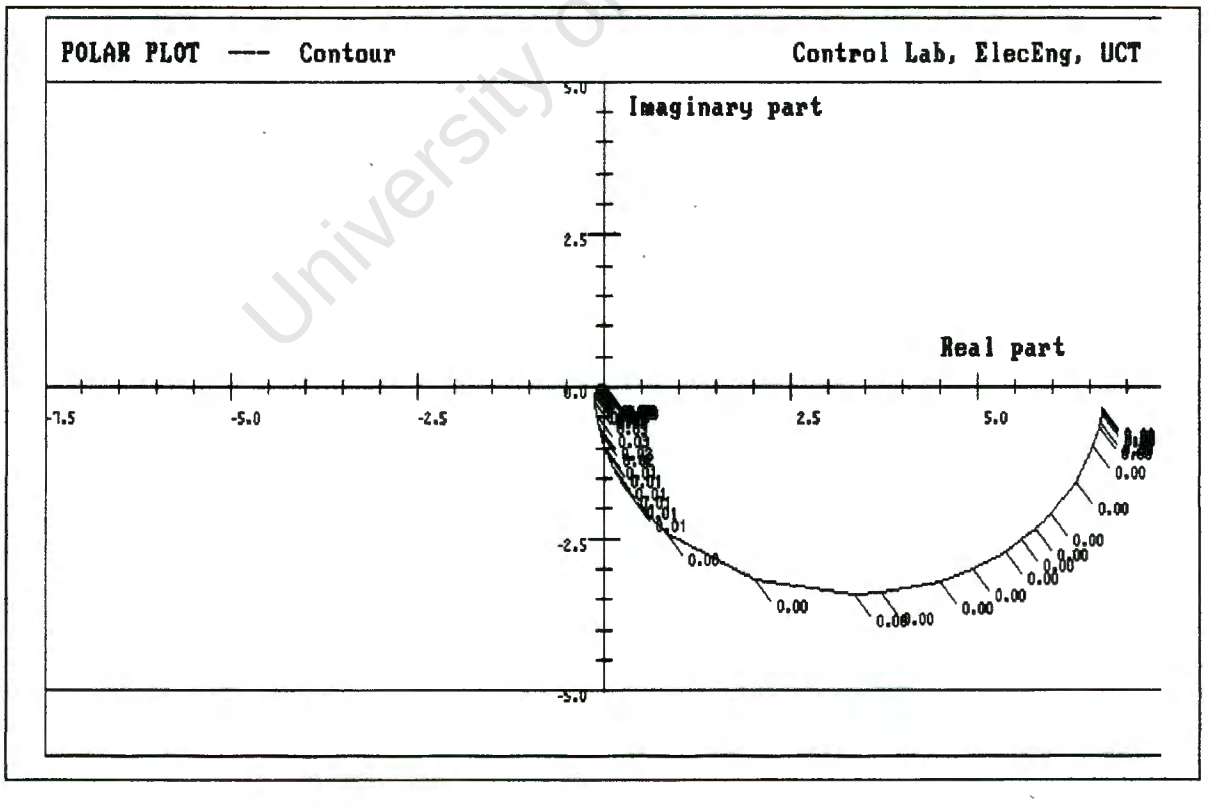


Figure J14: Nyquist Plot for element $gc_{44}(jw)$ of $G_C(s)$

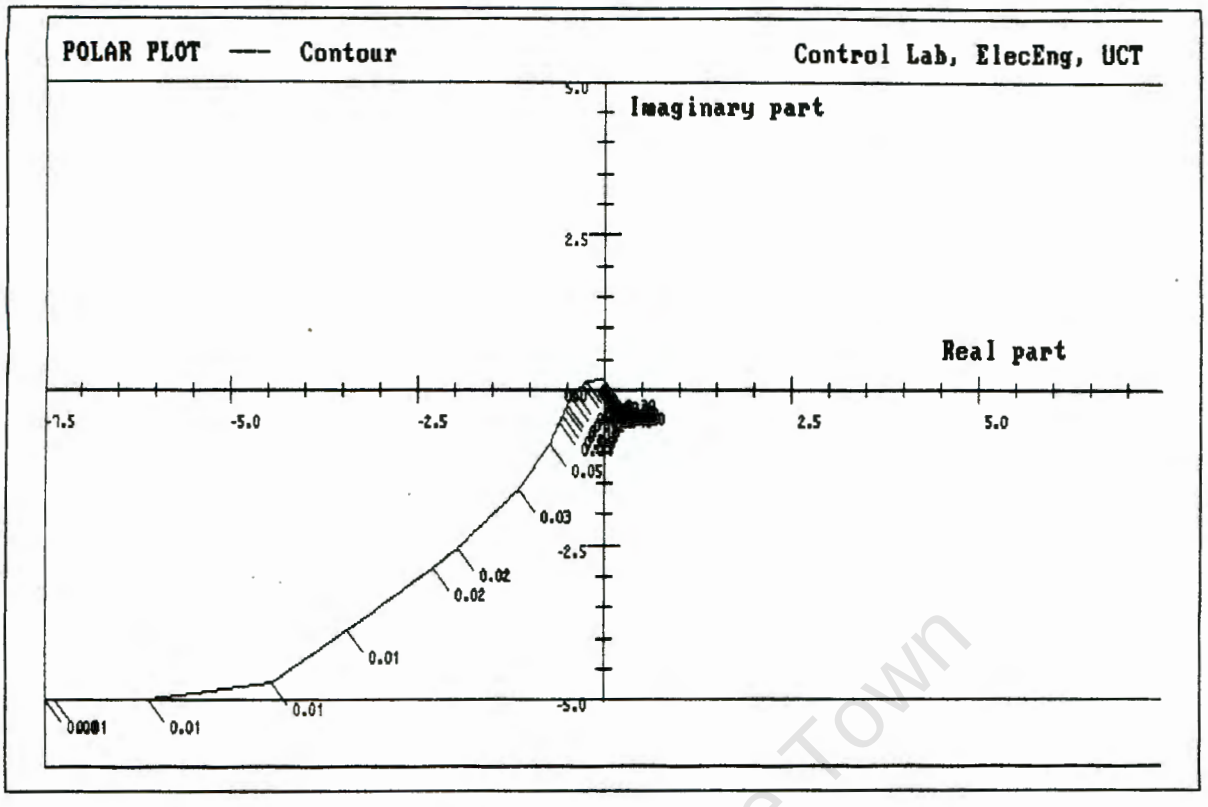


Figure J15: Nyquist Plot for Element $qc_{44}(jw)$

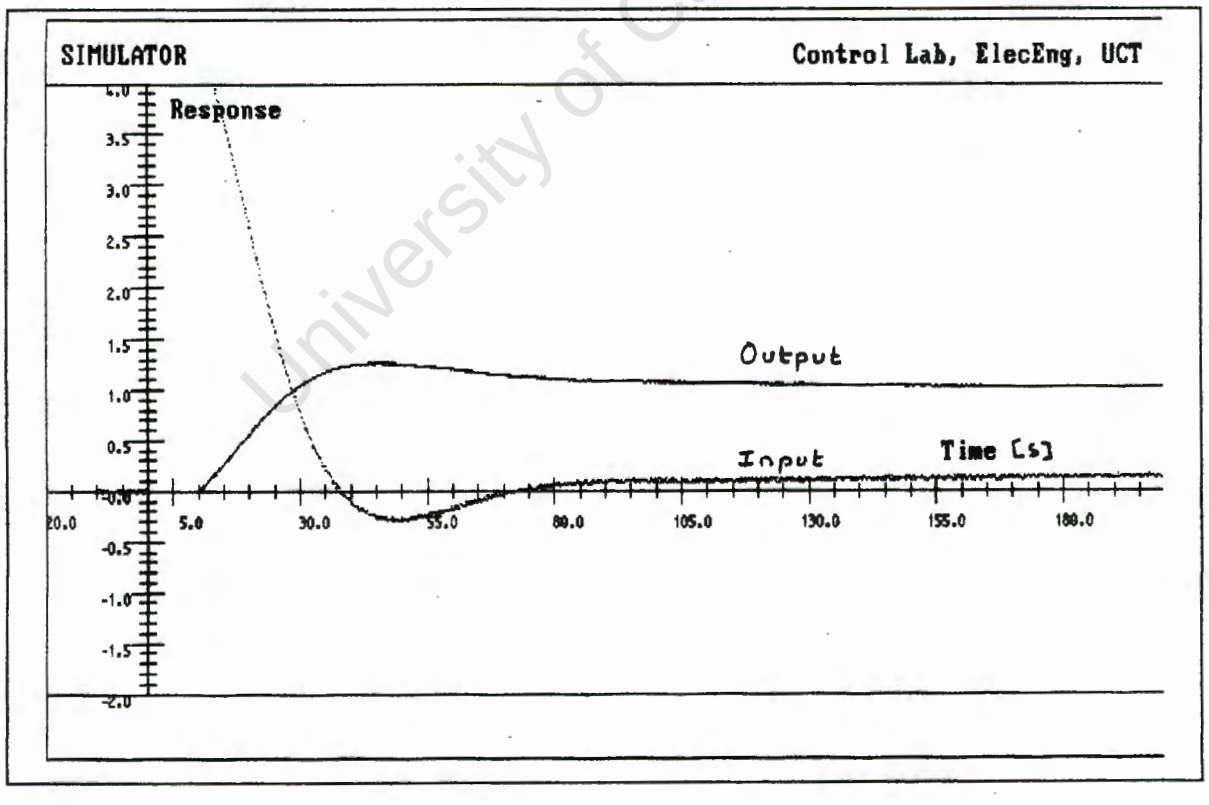


Figure J16: Simulated Closed Loop Time Response of Level L2 to a Unity Step in its Setpoint

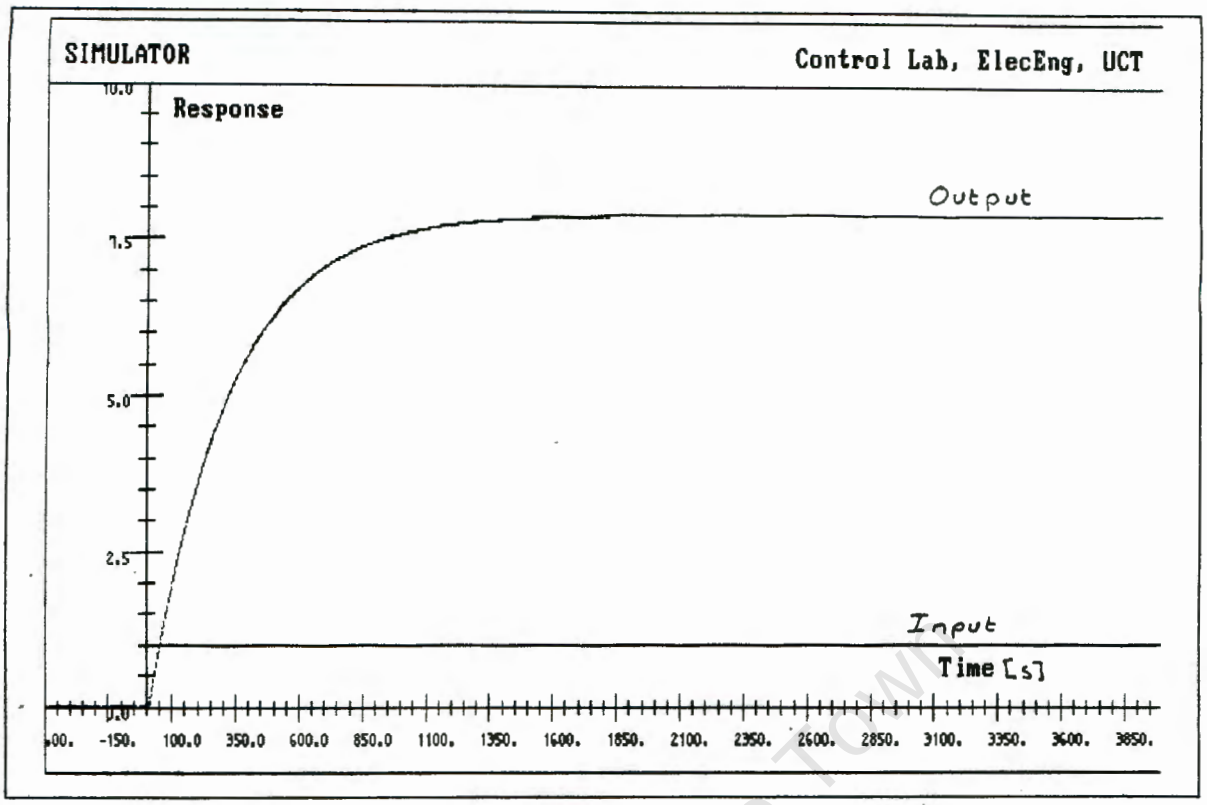


Figure J17: Simulated Open loop Time Response of Level L1 to a Unit Step in Control Valve C5

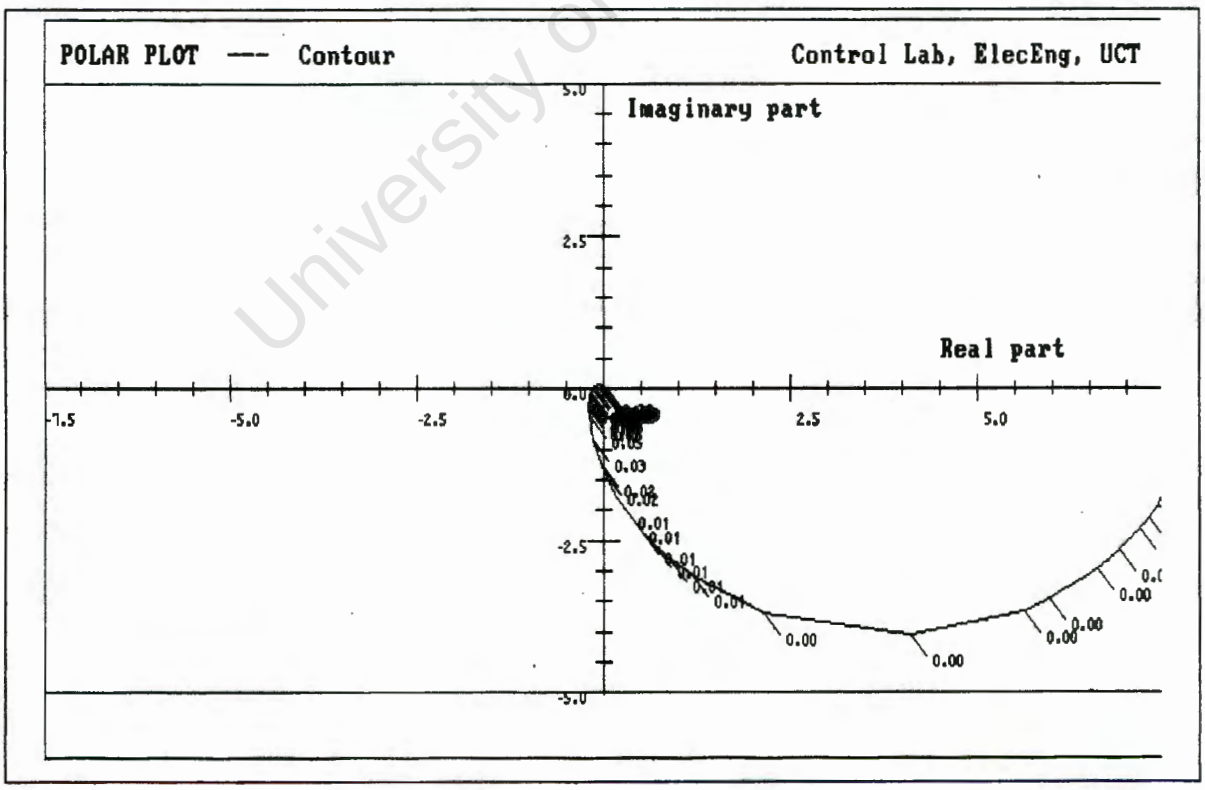


Figure J18: Nyquist Plot for element $gc_{55}(jw)$ of $G_C(s)$

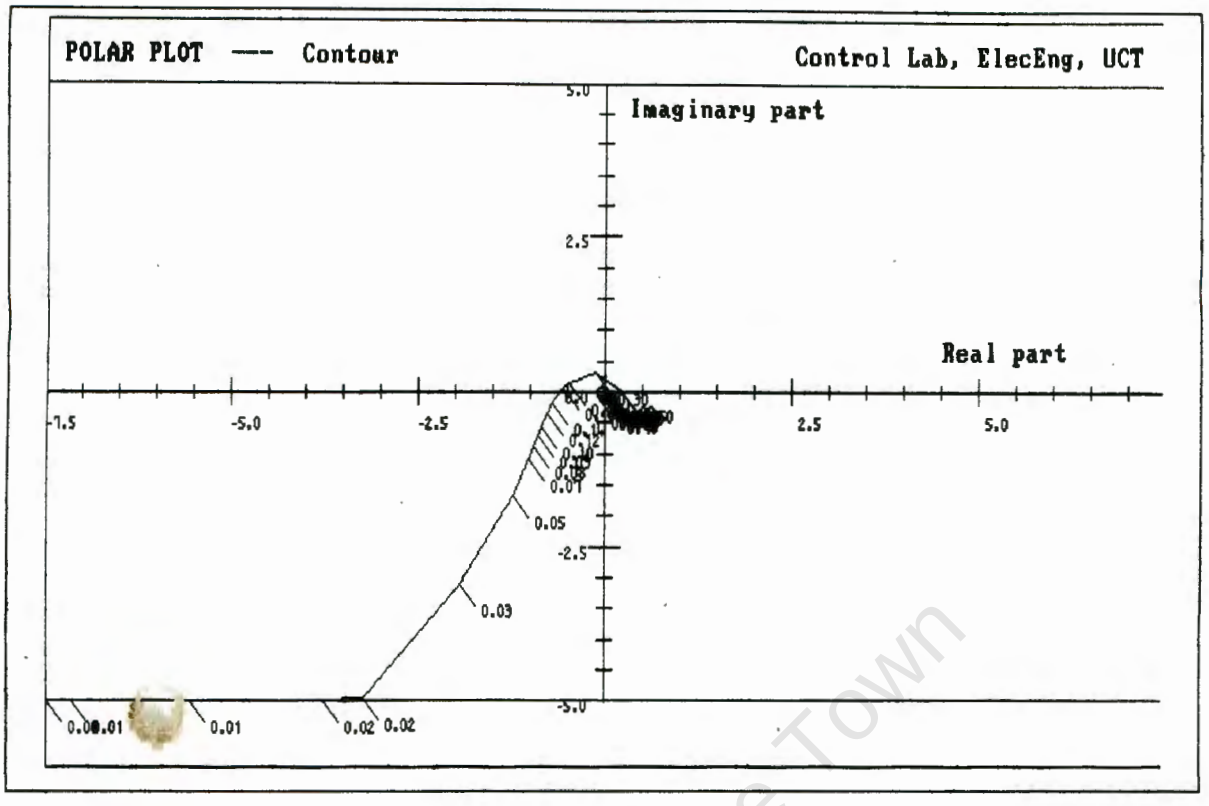


Figure J19: Nyquist Plot for Element $qc_{55}(jw)$

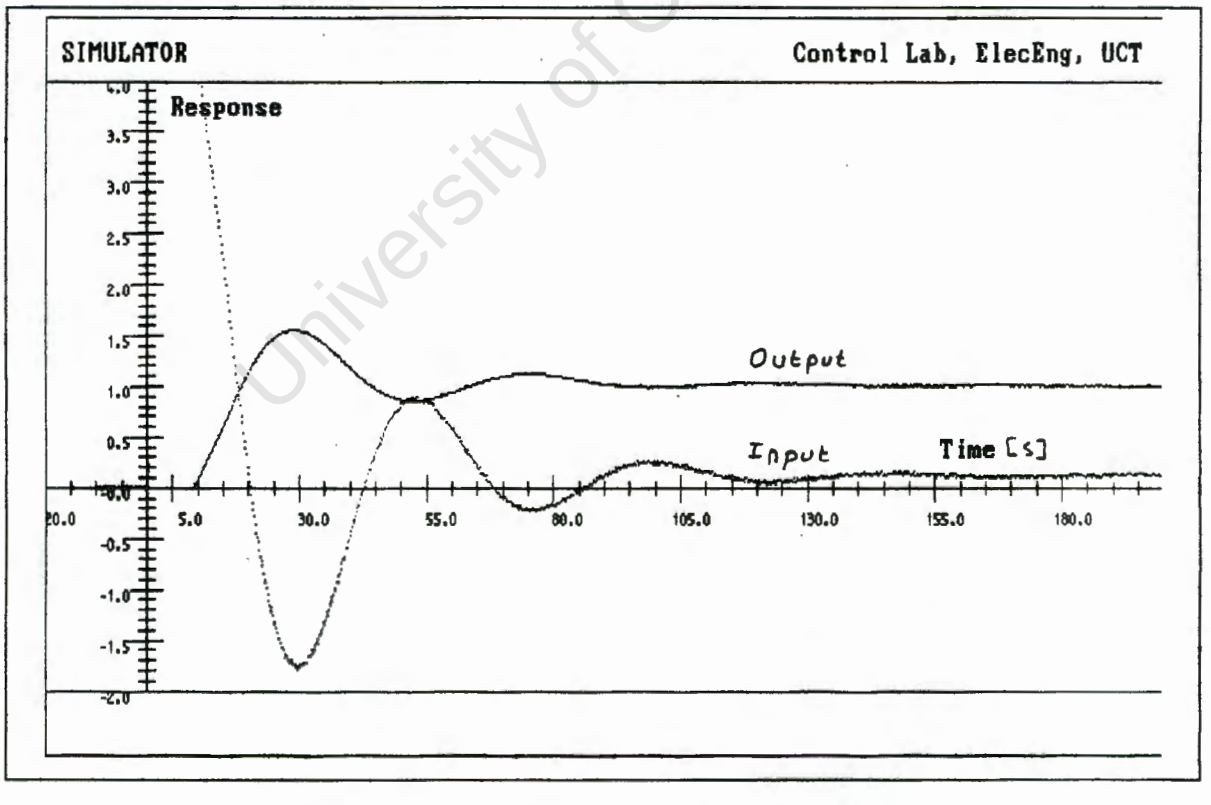


Figure J20: Simulated Closed Loop Time Response of Level L1 to a Unity Step in its Setpoint

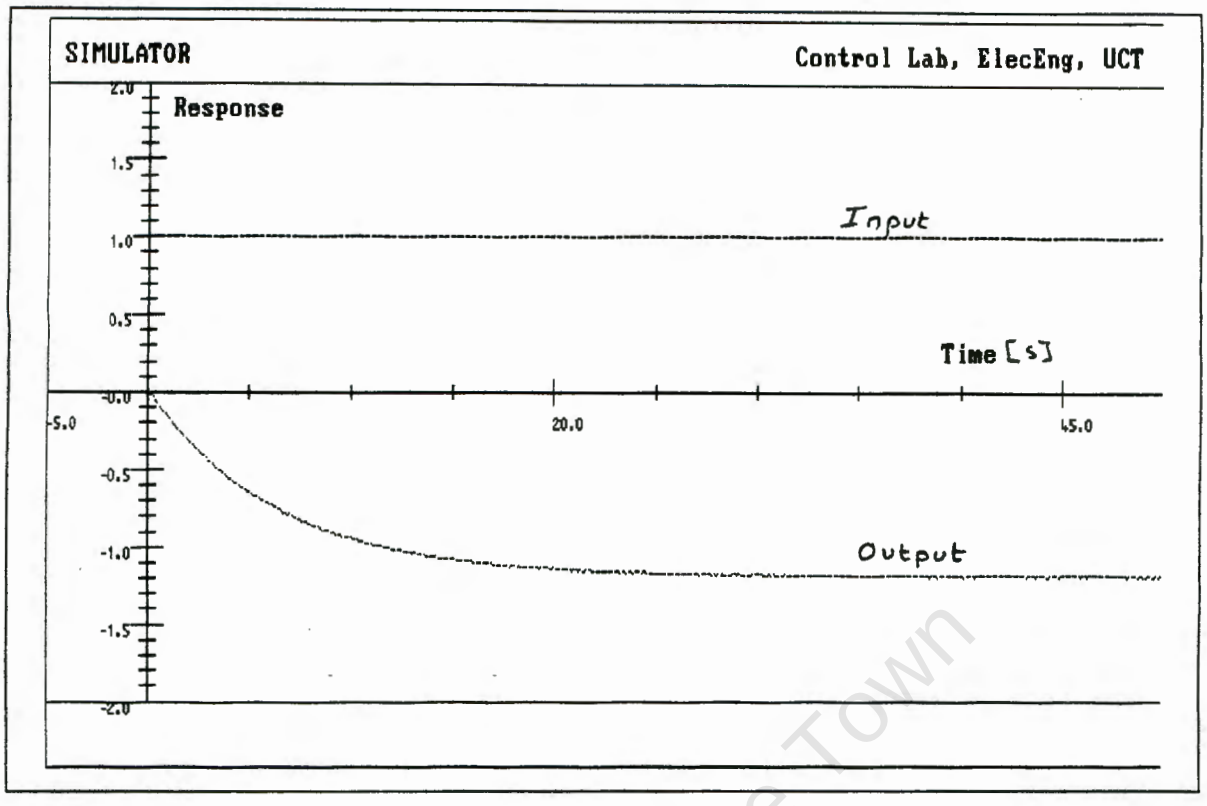


Figure J21: Simulated Open loop Time Response of Flow F2 to a Unit Step in Control Valve C6

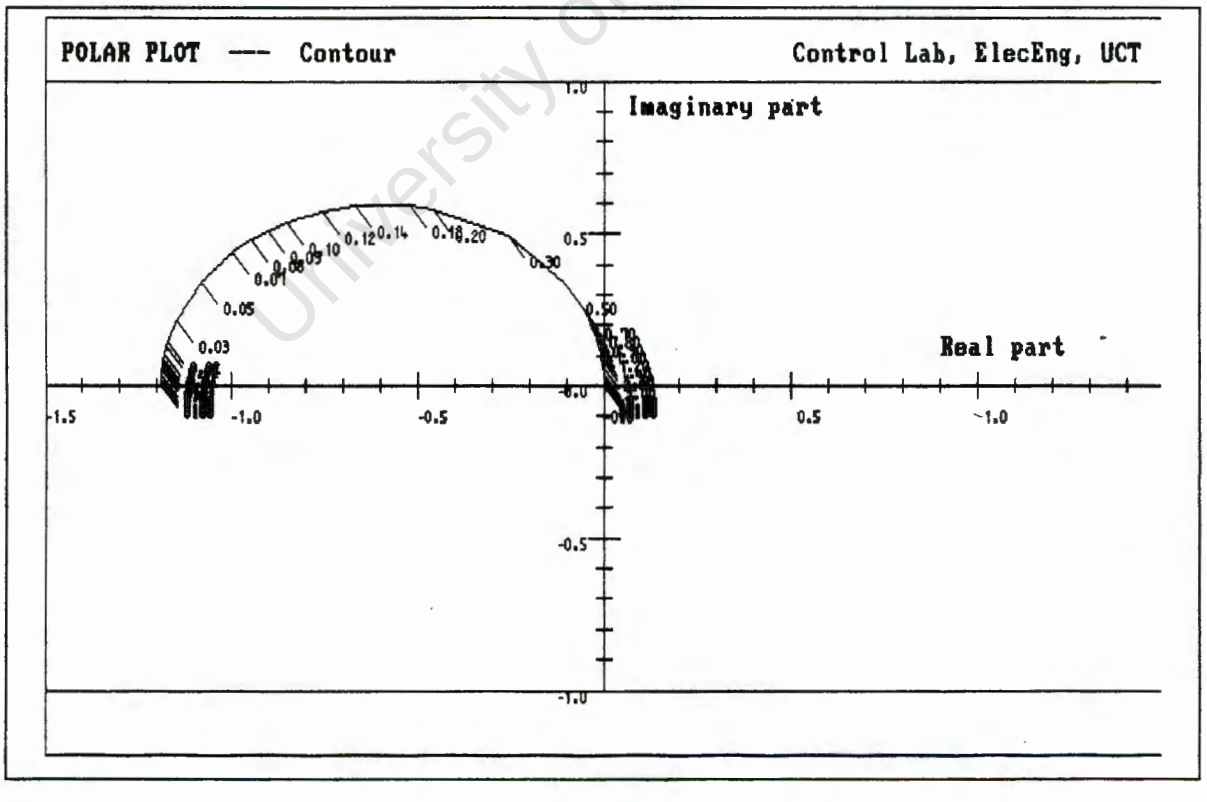


Figure J22: Nyquist Plot for element $gc_{66}(jw)$ of $G_c(s)$

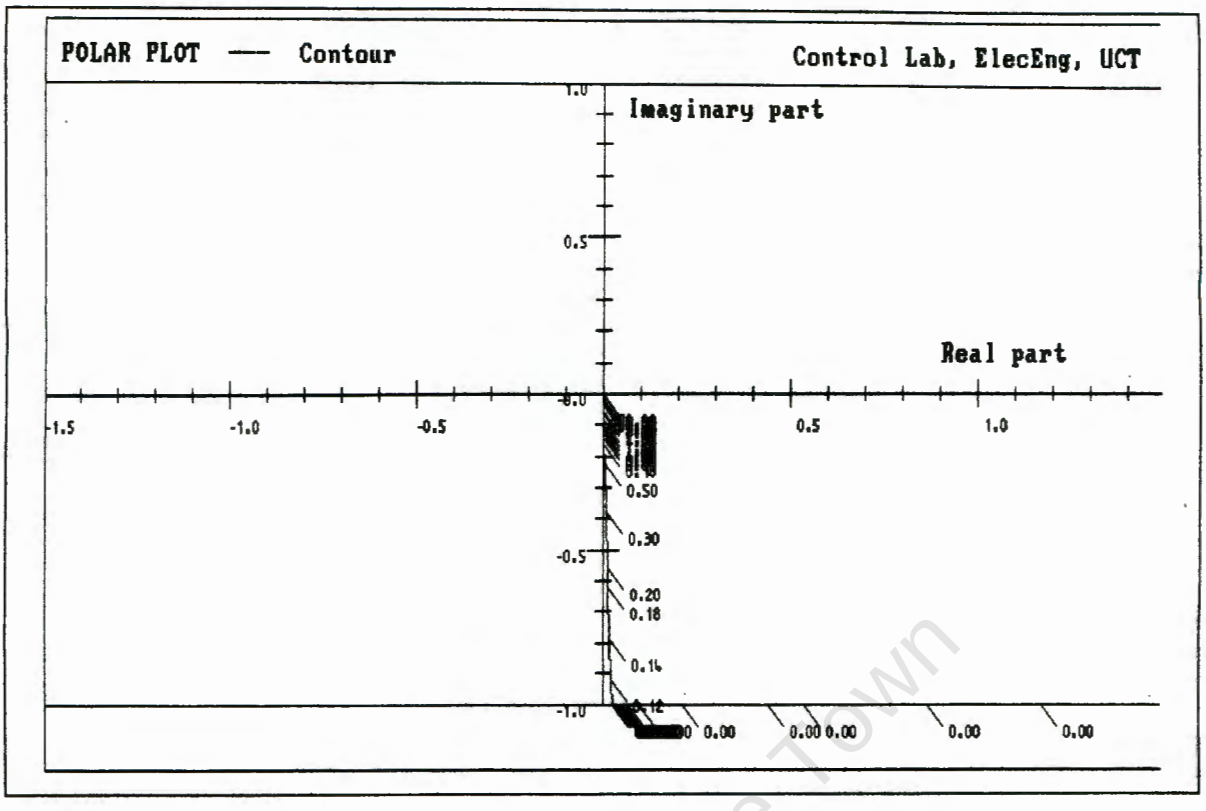


Figure J23: Nyquist Plot for Element $qc_{66}(jw)$

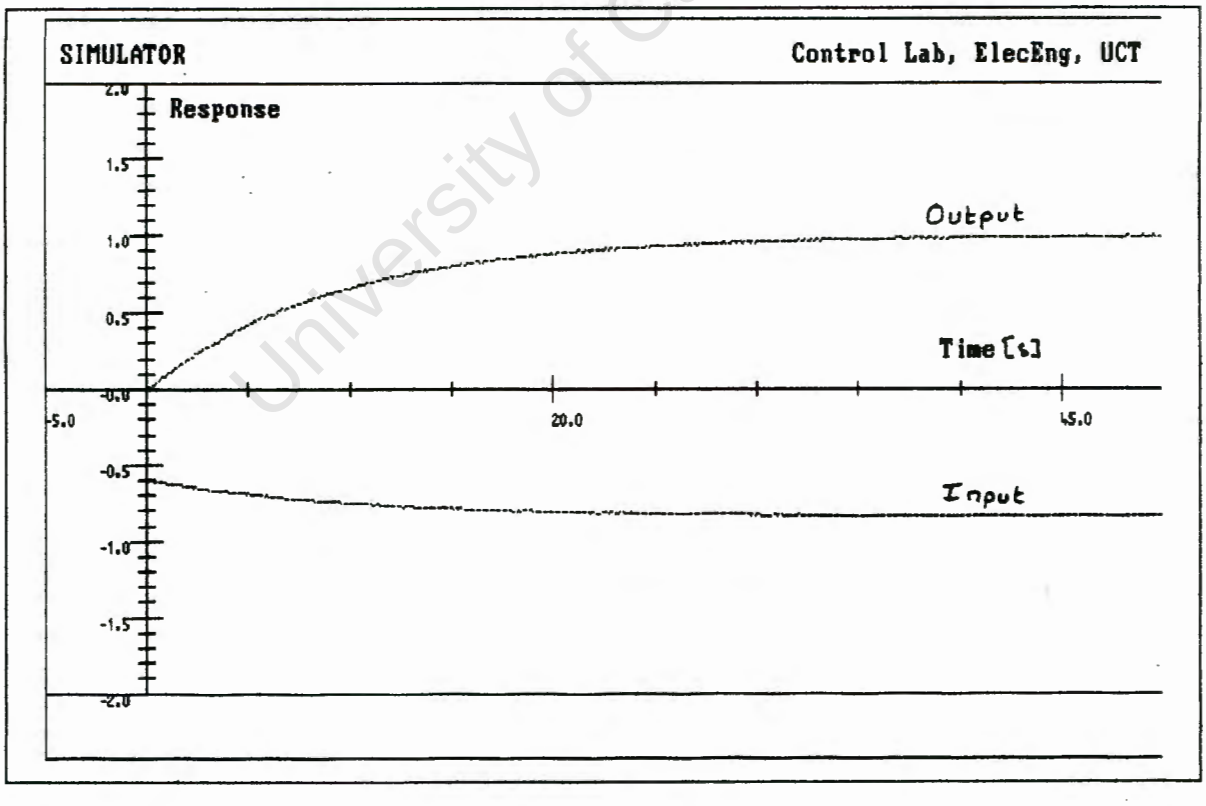


Figure J24: Simulated Closed Loop Time Response of Flow F2 to a Unity Step in its Setpoint

APPENDIX K

FLOW AND LEVEL RESPONSES WITH CASCADE COMPENSATOR $K_C(s)$

Figures K1 to K12 show the closed loop process responses to the stepping of the flow and level setpoints, with the cascade compensator $K_C(s)$ implemented in the control loop.

Each even numbered plot shows the response of a particular output to the indicated step in its setpoint.

Each odd numbered plot shows the response of the process outputs (levels) affected by a step change in this particular setpoint. The interactive coupling between this setpoint and the other loops in the system can be determined from these responses.

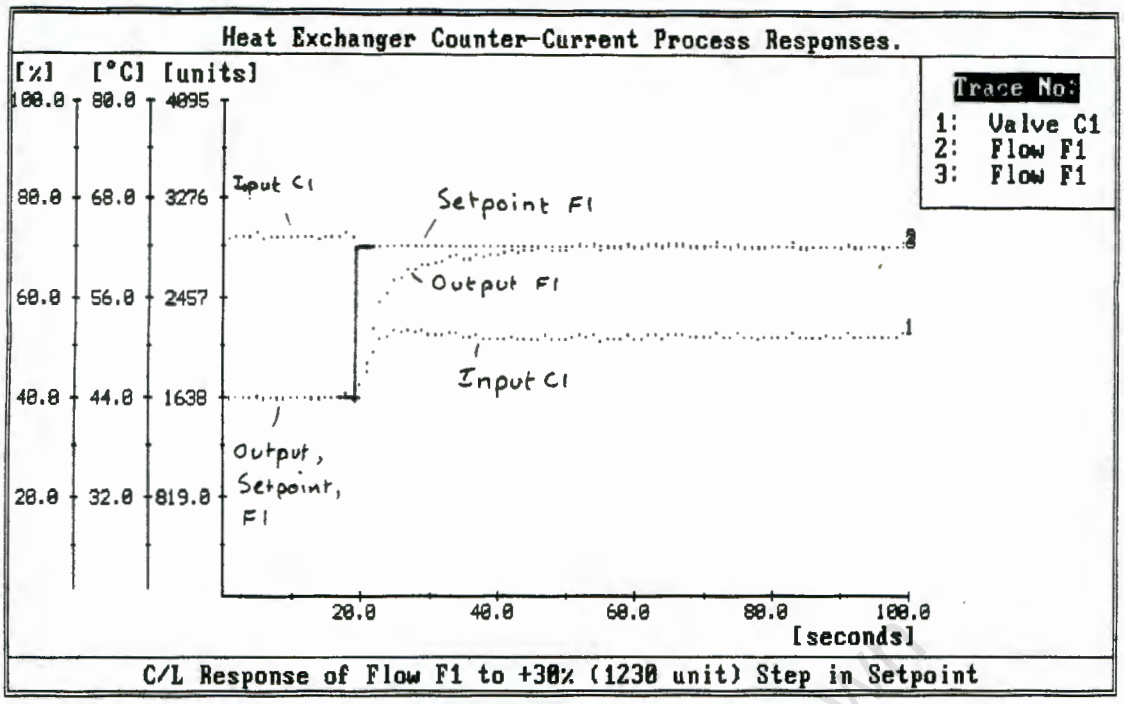


Figure K1: Closed Loop (with $K_c(s)$) Response of F1

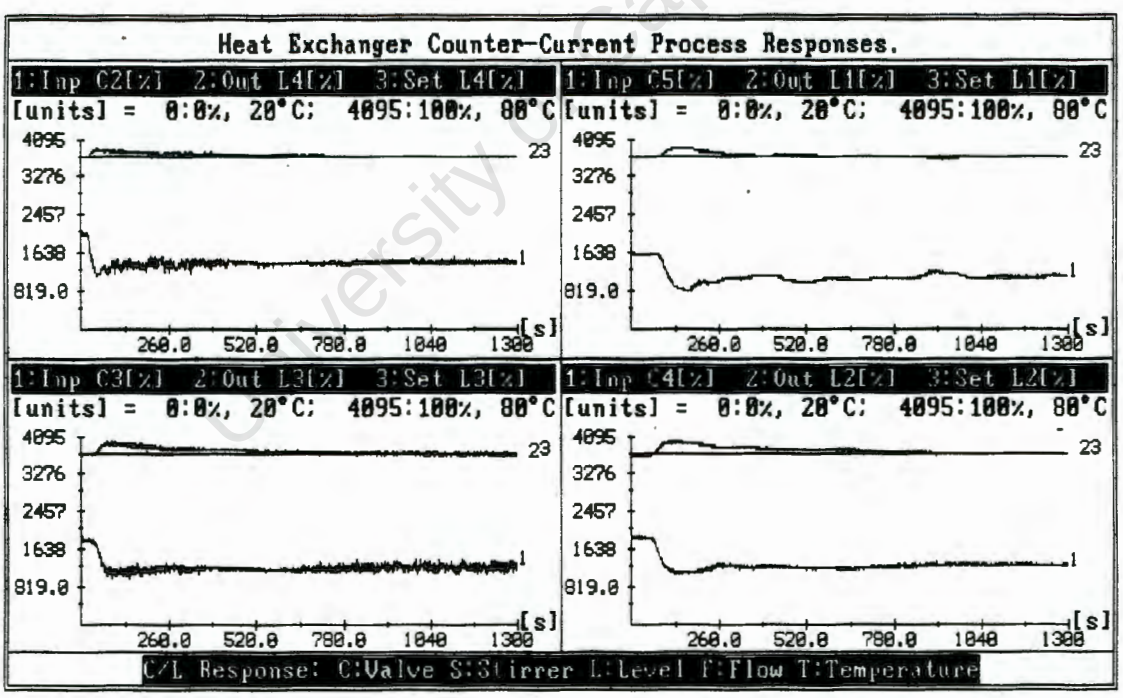


Figure K2: Interaction Caused by Stepping F1 Setpoint

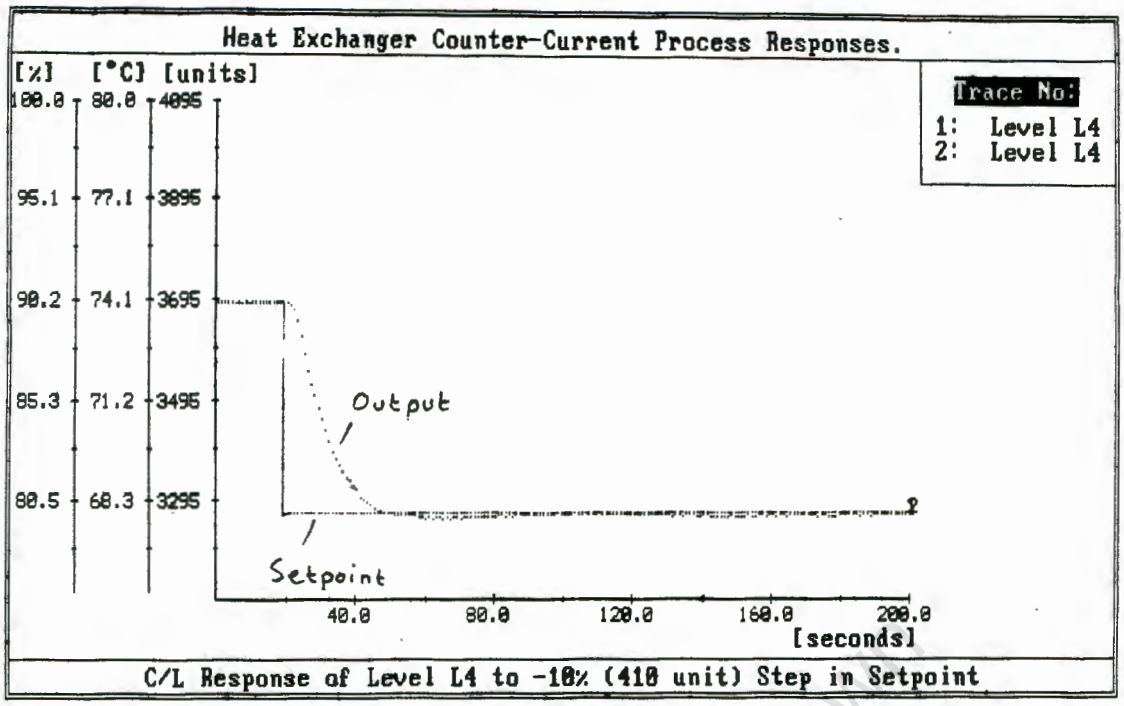


Figure K3: Closed Loop (with $K_C(s)$) Response of L4

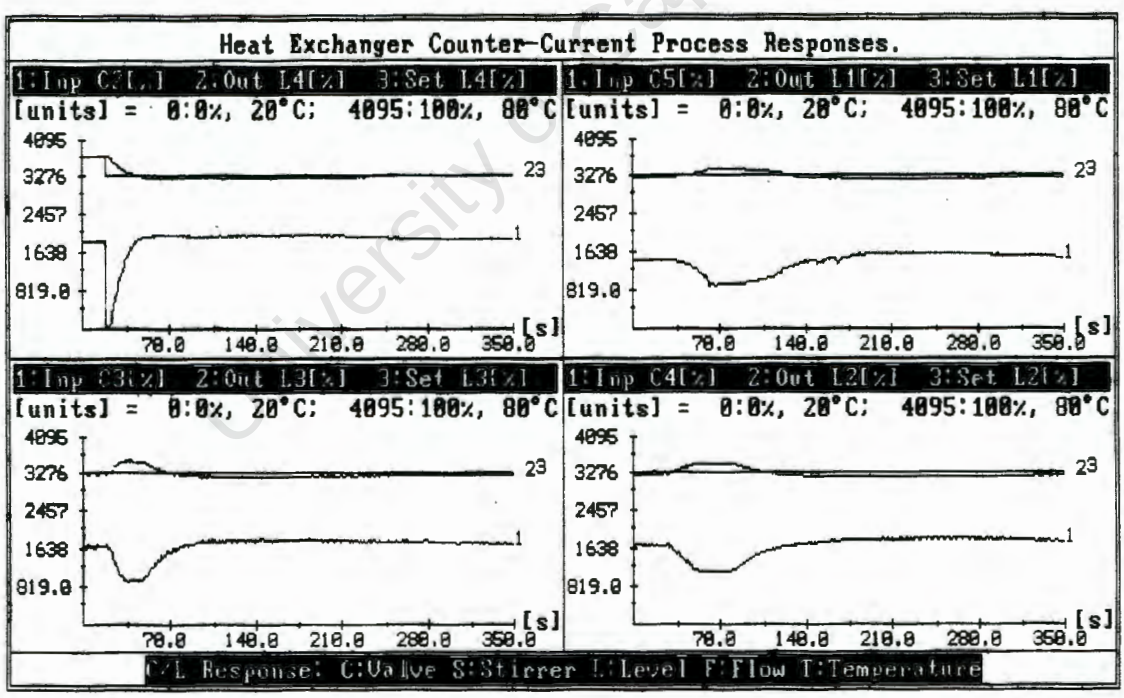


Figure K4: Interaction Caused by Stepping L4 Setpoint

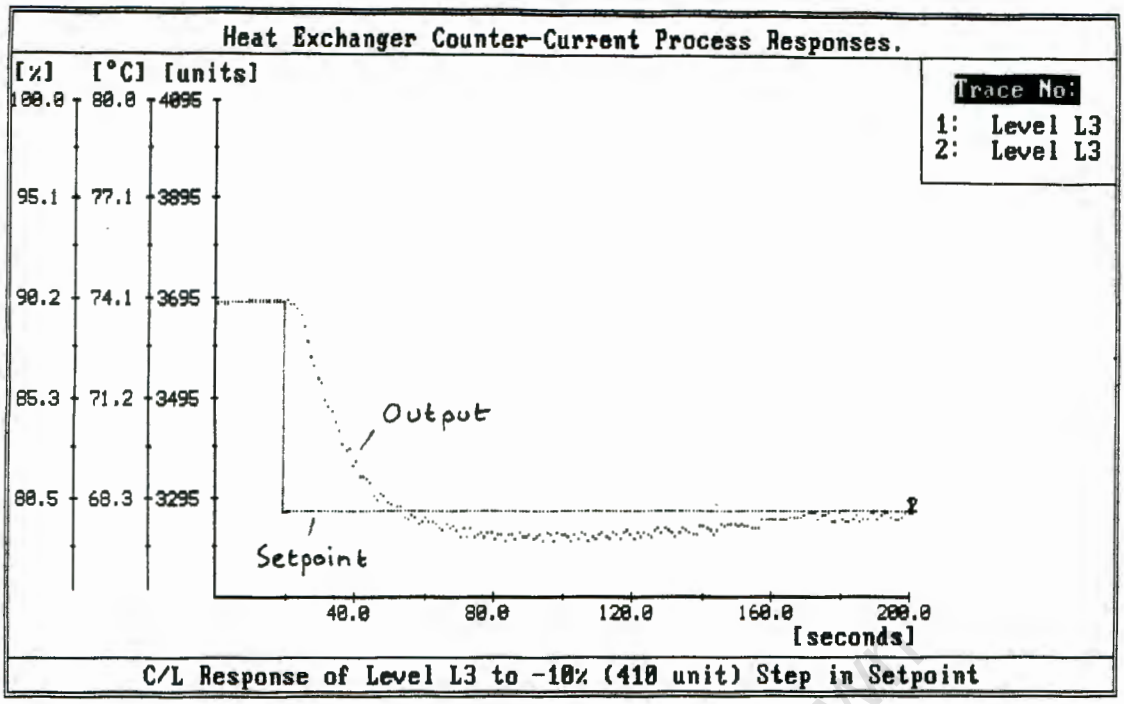


Figure K5: Closed Loop (with $K_c(s)$) Response of L3

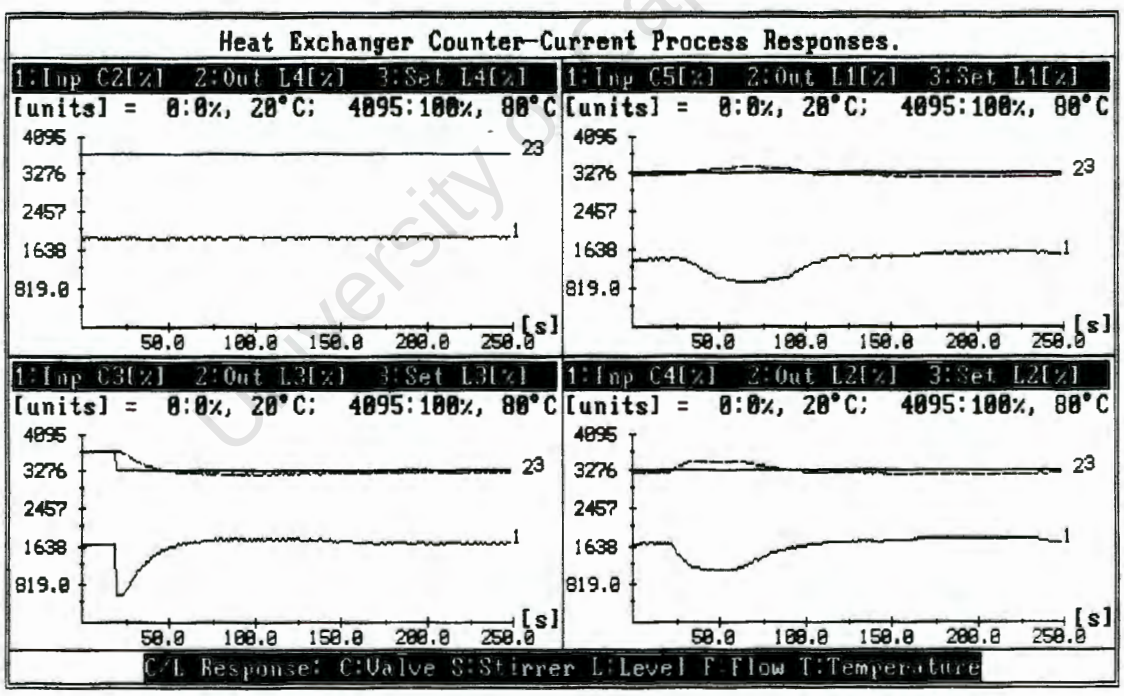


Figure K6: Interaction Caused by Stepping L3 Setpoint

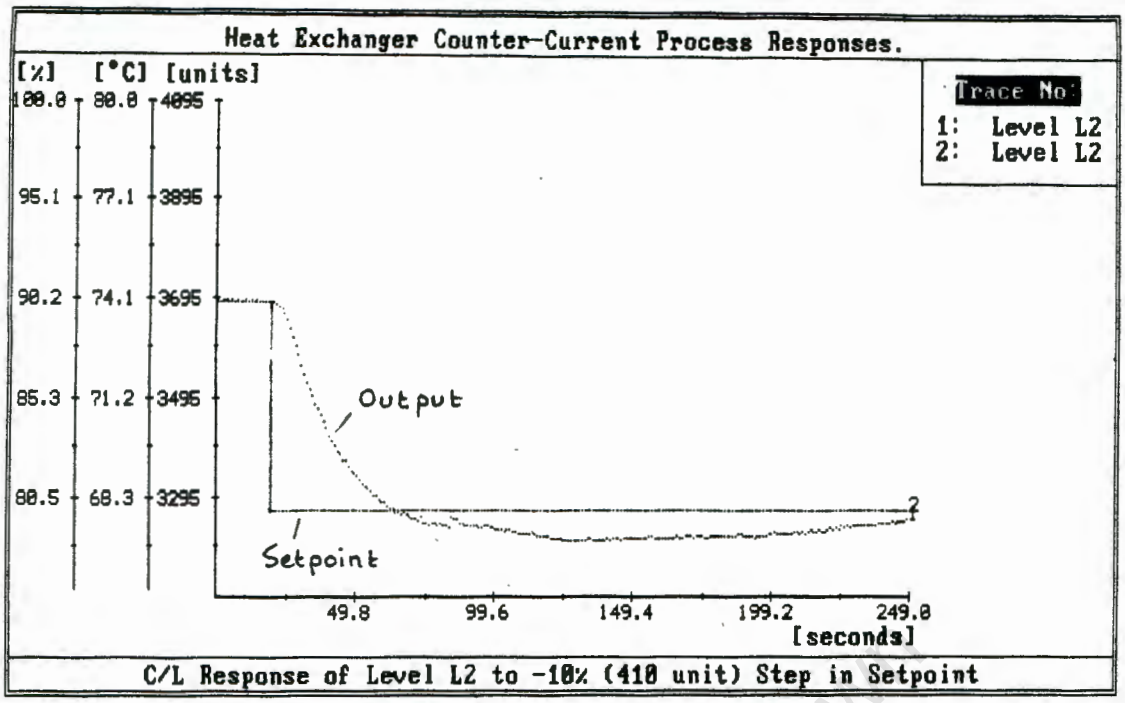


Figure K7: Closed Loop (with $K_c(s)$) Response of L2

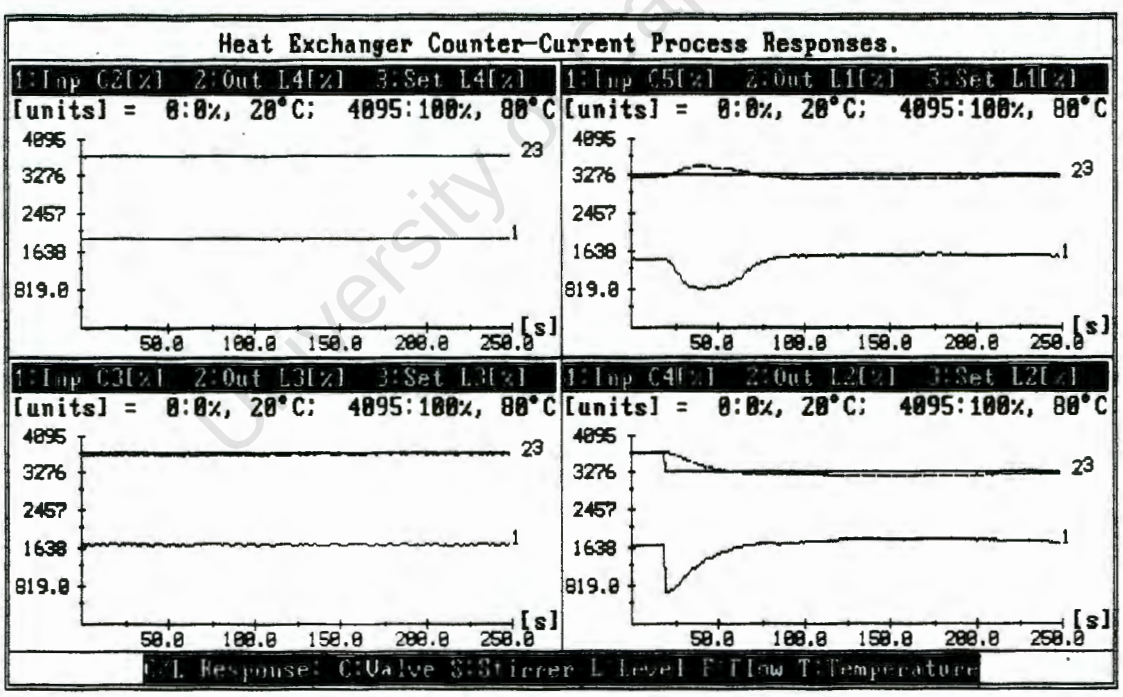


Figure K8: Interaction Caused by Stepping L2 Setpoint

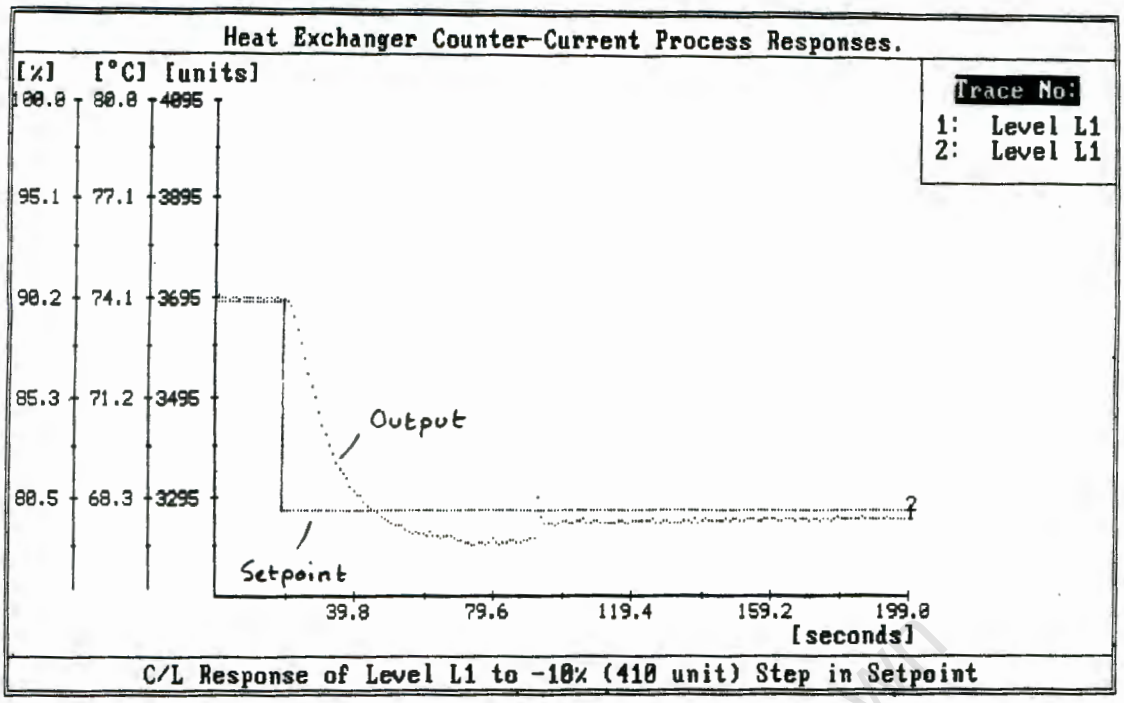


Figure K9: Closed Loop (with $K_c(s)$) Response of L1

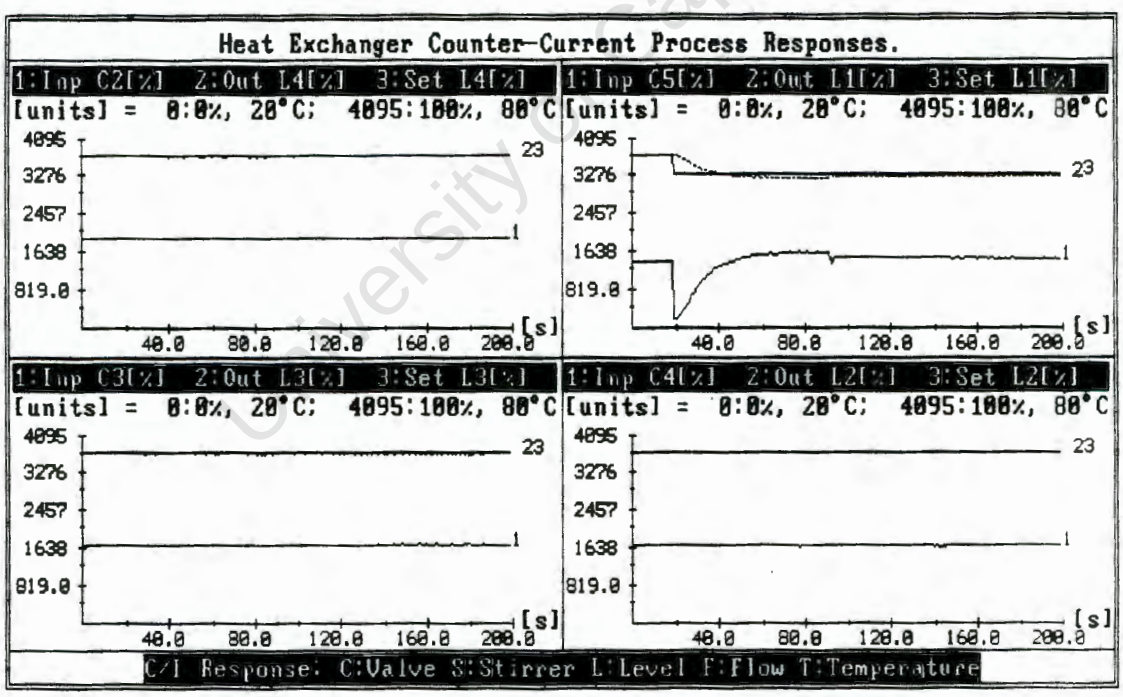


Figure K10: Interaction Caused by Stepping L1 Setpoint

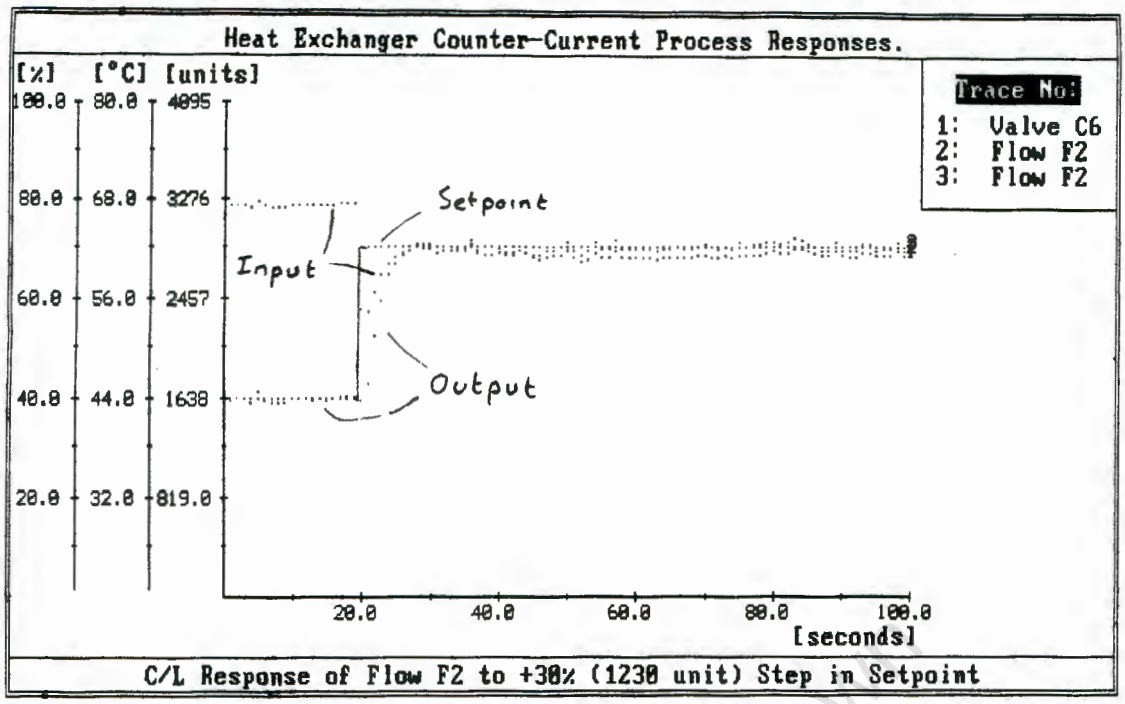


Figure K11: Closed Loop (with $K_C(s)$) Response of F2

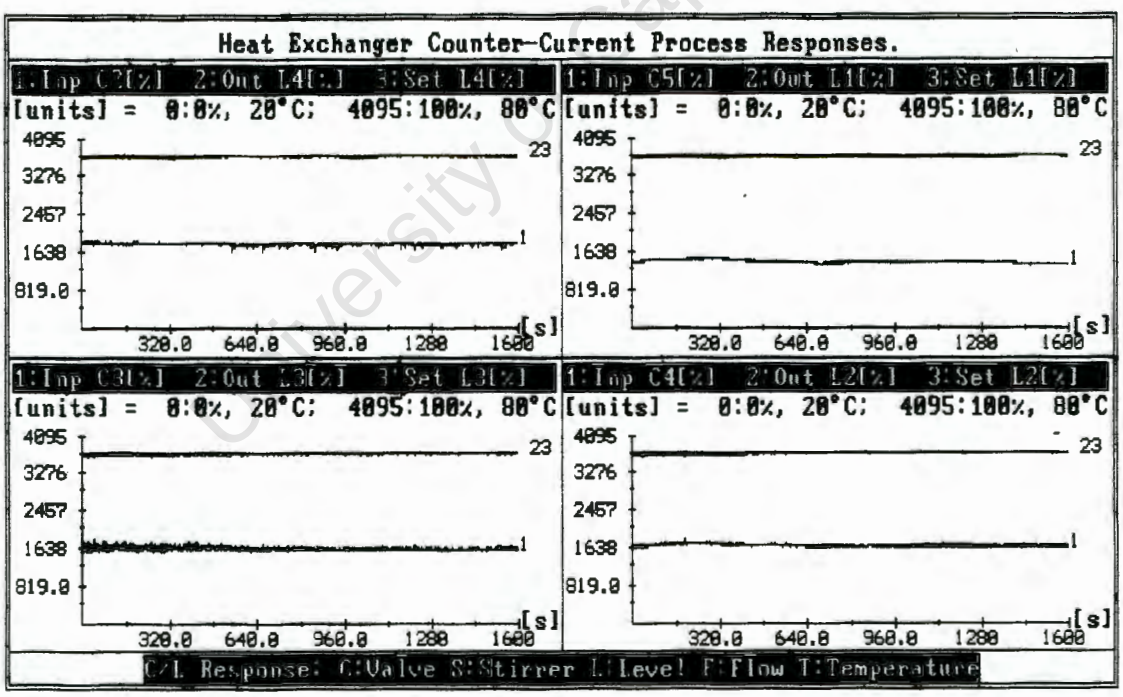


Figure K12: Interaction Caused by Stepping F2 Setpoint

APPENDIX L

SENSITIVITY MATRICES FOR GENERALIZED CONTROL SYSTEM

L1) Plant Output Sensitivity

Suppose that the nominal model for a process, represented by $G_0(s)$, differs from the true model $G(s)$ due to changes $\delta(G(s))$ in the nominal model as described in equation (L1).

$$G(s) = (I + \delta(G(s)))^{-1}G_0(s) \qquad (L1)$$

Define:

$$S_{02}(s) = (I + L_{02}(s))^{-1} \qquad (L2)$$

$$T_{02}(s) = I - S_{02}(s)$$

$$= (I + L_{02}(s))^{-1}L_{02}(s) \qquad (L3)$$

where:

$$L_{02}(s) = G_0(s)K(s)F(s) \qquad (L4)$$

Theorem:

If the nominal transmission matrix $T_{0YR}(s)$ characterizing the response of the process outputs to the process setpoints, is defined as:

$$T_{0YR}(s) = S_{02}(s)G_0(s)K(s)P(s) \qquad (L5)$$

Then,

$$T_{YR}(s) = \{I + \delta(T_{YR}(s))\}^{-1}T_{0YR}(s) \qquad (L6)$$

where:

$$\delta(T_{YR}(s)) = S_{02}(s)\delta(G(s)) \qquad (L7)$$

Proof:

The component of the response of the outputs resulting from perturbations in the setpoints, $Y_R(s)$, is given by:

$$Y_R(s) = T_{YR}(s)R(s) \qquad (L8)$$

then,

$$Y_R(s) = (I + L_2(s))^{-1}G(s)K(s)P(s)R(s)$$

so, dropping the function identifier (s) and rearranging,

$$(I + L_2)Y_R = GKPR$$

$$(I + GKF)Y_R = GKPR$$

applying (L1):

$$\{I + (I + \delta(G))^{-1}G_0KF\}Y_R = (I + \delta(G))^{-1}G_0KPR$$

applying (L4):

$$\{I + (I + \delta(G))^{-1}L_{02}\}Y_R = (I + \delta(G))^{-1}G_0KPR$$

Multiplying both sides by $(I + L_{02})^{-1}(I + \delta(G))$, we get:

$$\{(I + L_{02})^{-1}(I + \delta(G)) + (I + L_{02})^{-1}L_{02}\}Y_R = (I + L_{02})^{-1}G_0KPR$$

applying (L2) and (L3):

$$\{S_{02}(I + \delta(G)) + T_{02}\}Y_R = S_{02}G_0KPR$$

$$\{S_{02}(I + \delta(G)) + (I - S_{02})\}Y_R = S_{02}G_0KPR$$

applying (L5) and rearranging,

$$\{I + S_{02}\delta(G)\}Y_R = T_{0YR}R$$

so, restoring identifier (s) and rearranging,

$$Y_R(s) = \{I + S_{02}(s)\delta(G(s)\}^{-1}T_{0YR}(s)\ R(s) \quad (L9)$$

Comparing (L9) to (L8), it can be seen that:

$$T_{YR}(s) = \{I + S_{02}(s)\delta(G(s)\}^{-1}T_{0YR}(s)$$

$$= \{I + \delta(T_{YR}(s))\}^{-1}T_{0YR}(s) \quad (L10)$$

where:

$$\delta(T_{YR}(s)) = S_{02}(s)\delta(G(s)) \quad (L11)$$

L2) Plant Input (Control) Sensitivity

Suppose that the nominal model for a process, represented by $G_0(s)$, differs from the true model $G(s)$ due to changes $\delta(G(s))$ in the nominal model as described in equation (L12).

$$G(s) = G_0(s)(I + \delta(G(s))) \qquad \text{(L12)}$$

Define:

$$S_{01}(s) = (I + L_{01}(s))^{-1} \qquad \text{(L13)}$$

$$T_{01}(s) = I - S_{01}(s)$$

$$= (I + L_{01}(s))^{-1}L_{01}(s) \qquad \text{(L14)}$$

where:

$$L_{01}(s) = K(s)F(s)G_0(s) \qquad \text{(L15)}$$

Theorem:

If the nominal transmission matrix $T_{OUR}(s)$ characterizing the response of the process inputs to the process setpoints, is defined as:

$$T_{OUR}(s) = S_{01}(s)K(s)P(s) \qquad \text{(L16)}$$

Then,

$$T_{UR}(s) = \{I + \delta(T_{UR}(s))\}^{-1}T_{OUR}(s) \qquad \text{(L17)}$$

where:

$$\delta(T_{UR}(s)) = T_{01}(s)\delta(G(s)) \qquad \text{(L18)}$$

Proof:

The component of the response of the inputs resulting from perturbations in the setpoints, $U_R(s)$, is given by:

$$U_R(s) = T_{UR}(s)R(s) \qquad \text{(L19)}$$

then,

$$U_R(s) = (I + L_1(s))^{-1}K(s)P(s)R(s)$$

so, dropping the function identifier (s) and rearranging,

$$(I + L_1)U_R = KPR$$

$$(I + KFG)U_R = KPR$$

applying (L12):

$$\{I + KFG_0(I + \delta(G))\}Y_R = KPR$$

applying (L15):

$$\{I + L_{01}(I + \delta(G))\}U_R = KPR$$

Multiplying both sides by $(I + L_{01})^{-1}$, we get:

$$\{(I + L_{01})^{-1} + (I + L_{01})^{-1}L_{01}(I + \delta(G))\}U_R = (I + L_{01})^{-1}KPR$$

applying (L13) and (L14):

$$\{S_{01} + T_{01}(I + \delta(G))\}U_R = S_{01}KPR$$

$$\{S_{01} + (I - S_{01})(I + \delta(G))\}U_R = S_{01}KPR$$

applying (L16) and rearranging,

$$\{I + (I - S_{01})\delta(G)\}U_R = T_{0UR}R$$

applying (L14),

$$\{I + T_{01}\delta(G)\}U_R = T_{0UR}R$$

so, restoring identifier (s) and rearranging,

$$U_R(s) = \{I + T_{01}(s)\delta(G(s))\}^{-1}T_{0UR}(s)\ R(s) \quad (L20)$$

Comparing (L20) to (L19), it can be seen that:

$$T_{UR}(s) = \{I + T_{01}(s)\delta(G(s))\}^{-1}T_{0UR}(s)$$

$$= \{I + \delta(T_{UR}(s))\}^{-1}T_{0UR}(s) \quad (L21)$$

where:

$$\delta(T_{UR}(s)) = T_{01}(s)\delta(G(s)) \quad (L22)$$

APPENDIX M

OPTCAD SYSTEM INFORMATION

M1) OPTCAD System Subroutines

A table of the OPTCAD system subroutine names follows on the next page. The name of each subroutine, together with the file in which it is located and the page in the appendix on which it is listed, is given in the table.

OPTCAD System Subroutines			
Name	File	Description	Page
anal	ANAL.FOR	Performance Index Analysis Menu Driver	344
concal	CONCAL.FOR	State Feedback Control Law Calculation	344
create	CREATE.FOR	Drive Performance Index Creation Menu	345
dist	DIST.FOR	Inject Disturbances into Plant	345
disvop	DISVOP.FOR	Display Which Perf. Index Plot to Observe	346
excuse	EXCUSE.FOR	Utility for Program Development	347
filovw	FILOVW.FOR	Confirm Overwriting of File	347
gauin	GAUIN.FOR	Obtain Specifications of Gaussian Noise/Dist	347
getfil	GETFIL.FOR	Obtain Filename from User	348
getset	GETSET.FOR	Get Specs of Perturbations on Each Channel	349
inicad	INICAD.FOR	Initialize Program Parameters	350
inidat	INIDAT.FOR	Block Data Section Initializing Package Data	350
intper	INTPER.FOR	Integrate Perf. Indices over Frequency Range	351
jwmat	JWMAT.FOR	Create a Complex Matrix jwI	353
keydat	KEYDAT.FOR	Assign Values of Keys on Keyboard to Varibles	354
lhs	LHS.FOR	Utility Routine for Riccati Equation Solution	354
maknam	MAKNAM.FOR	Construct Filename, Including Path Name	355
nindin	NINDIN.FOR	Drive Gauss/Sinusoidal Perturb Selection Menu	356
noidis	NOIDIS.FOR	Specify Sinusoidal Noise/Disturbance Options	356
noise	NOISE.FOR	Generate Gaussian or Sinusoidal Noise	357
obssta	OBSSTA.FOR	Calculate Observer States	357
optcad	OPTCAD.FOR	Initialize Program and Present Main Menu	358
optkc	OPTKC.FOR	Synthesize Optimal LQ Controller Gain Matrix	359
optkf	OPTKF.FOR	Synthesize Optimal Kalman Gain Matrix	360
percal	PERCAL.FOR	Calculate Performance Index Data	362
peropt	PEROPT.FOR	Obtain Performance Index Plotting Options	366
perplt	PERPLT.FOR	View Specific Perf Index Data Plot and Analysis	368

OPTCAD System Subroutines			
Name	**File**	**Description**	**Page**
perset	PERSET.FOR	Perturb Setpoint Gaussianly or Sinusoidally	369
piplot	PIPLOT.FOR	View Group of Perf Index Data Plots and Analysis	370
pldat	PLDAT.FOR	Block Data containing Plotting Data	371
precal	PRECAL.FOR	Calculates Precompensator for S-state Tracking	372
prosta	PROSTA.FOR	Calculate Process States	373
setper	SETPER.FOR	Specify Sinus. Setp. Perturb. Options	374
sinin	SININ.FOR	Drive Sinusoid Perturbation Menu	375
synth	SYNTH.FOR	Call LQG Synthesis Procedures	375
togdty	TOGDTY.FOR	Toggle Output of State Selection	376
togopt	TOGOPT.FOR	User Toggle Yes/No Option Flag	376
togvop	TOGVOP.FOR	Toggle Performance Index Plot View Option	377

16. Zames G. Feedback and Optimal Sensitivity: Modal reference transformations, multiplicative seminorms and approximate inverses, IEEE Trans. Aut. Contr., 1981, AC-26, (2), pp. 301-320.

```
C
C     FILE                : ANAL.FOR
C ***************************************************************************
CN    MODULE NAME         : anal
CA    FUNCTION            : Performance Index Analysis Menu Driver
CS    CALL SEQUENCE       : call anal()
CI    INPUT PARAMETERS    : None
CO    OUTPUT PARAMETERS   : None
CG    GLOBAL VARIABLES    : Include files:  runvar.inc
C                           Common blocks:  /view/ vop
CM    MODULES CALLED      : Fortran   :  inmenu, wrhead,peropt,create,togopt,
C                                        piplot,data
C                           Assembler: DOMENU,WRTSTR,ERTONE,LEVEL,INKEY
CE    ERROR CONDITIONS    : None
CC    COMMENTS            : Initializes menu screens and drives performance
C                           index analysis menu.
C ***************************************************************************
      subroutine anal()
      implicit integer*2 (D)
$include: 'runvar.inc'
      integer vop,iopt,ixpos,ikey
      integer*2 opt13
      common /view/ vop

99    continue

      call inmenu()

C Write heading. (wrhead.for - iglib.lib)
      call wrhead(0,(720-45*9)/2,35,45,
     +   'Performance Index Data Creation and Analysis.')

C Get chosen option. (domenu.asm - iglib.lib)
      opt13 = DOMENU(13)

C Blank over heading. (util.asm - iglib.lib)
      call WRTSTR(0,(720-45*9)/2,35,45,
     +   '                                                ')

      if (opt13.eq.1) then
          call peropt()

      elseif (opt13.eq.2) then
          if ((.not.lqafl).and.(.not.deafl)) then
              call create()
              if (lqgfl) then
                  lqafl = .true.
              else
                  deafl = .true.
              endif
          else
              call ERTONE()
              call LEVEL(0)
              call WRTSTR(0,((720 - 41*9)/2),200,41,
     +             'Analysis Data Already Present: Recreate?:')
              call LEVEL(1)
              iopt = 1
              ikey = 1
              ixpos = ((720 - 41*9)/2) + 42*9
              call togopt(iopt,ixpos,200,ikey)
              if (iopt.eq.0) call create()
              call WRTSTR(0,((720 - 37*9)/2),200,37,
     +                    '                                    ')
          endif


      elseif (opt13.eq.3) then
          if (deafl.or.lqafl) then
              call intper()
              if (vop.eq.0) then
                  call piplot()
              else
                  call perplt(vop)
              endif
          else
              call ERTONE()
              call LEVEL(0)
              call WRTSTR(0,((720 - 37*9)/2),200,37,
     +             'No Analysis Data Present for Plotting')
              call LEVEL(1)
              key = INKEY(1)
              call WRTSTR(0,((720 - 37*9)/2),200,37,
     +                    '                                    ')
          endif

      elseif (opt13.eq.4) then
          call data()


      elseif (opt13.ne.0) then
          call ERTONE()

      endif

      if (opt13.ne.0) then
          goto 99
      endif
      return
      end
C ***************************************************************************
CH    REVISION HISTORY  :
C     VERSION       BY                  DATE        COMMENT
C      1.1     Anton de Waal          05/01/90  Final Commenting
C     FILEND            :
C ***************************************************************************


C
C     FILE                : : CONCAL.FOR
C ***************************************************************************
CN    MODULE NAME         : concal
CA    FUNCTION            : State Feedback Control Law Calculation
CS    CALL SEQUENCE       : call concal()
CI    INPUT PARAMETERS    : None
CO    OUTPUT PARAMETERS   : None
CG    GLOBAL VARIABLES    : Include files:  syst.inc,time.inc
```

```
CM     MODULES CALLED     : None
CE     ERROR CONDITIONS   : None
CC     COMMENTS           : Takes actual (if obsinc = 1) or observed states (if
C                           obsinc = 0), multiplies the states by feedback gain
C                           matrix kc, and subtracts the results from the closed
C                           loop inputs (vn) to realize the state feedback
C                           control law un = vn - kc*xn or un = vn - kc*xnobs
C ***************************************************************************
       subroutine concal()
$include: 'syst.inc'
$include: 'time.inc'
       integer order,dima
       real temp(15)

       order = int(syst(4))
       dima = int(syst(5))

       do 100 i = 1, dima
           if (obsinc.eq.1) then
               xo(i) = xn(i)
           else
               xo(i) = xnobs(i)
           endif
           if (i.le.order) temp(i) = 0.0
100    continue

       do 200 i = 1, order
           do 210 j = 1, dima
               temp(i) = temp(i) + kc(i,j)*xo(j)
210        continue
       un(i) = vn(i) - temp(i)
200    continue

       return
       end
C ***************************************************************************
CH     REVISION HISTORY   :
C      VERSION       BY                  DATE        COMMENT
C        1.1      A. de Waal           06/01/90  Finally Commented
C      FILEND             :
C ***************************************************************************


C
C      FILE               : CREATE.FOR
C ***************************************************************************
CM     MODULE NAME        : create
CA     FUNCTION           : Drive Performance Index Creation Menu
CS     CALL SEQUENCE      : call create()
CI     INPUT PARAMETERS   : None
CO     OUTPUT PARAMETERS  : None
CG     GLOBAL VARIABLES   : Include files:  runvar.inc
CM     MODULES CALLED     : Fortran  :  inmenu,wrhead,edpopt,percal,intper,piplot
C                           Assembler:  DOMENU,WRTSTR,ERTONE
CE     ERROR CONDITIONS   : None
CC     COMMENTS           : Initializes menu screen and drives performance
C                           index creation menu
C ***************************************************************************
       subroutine create()
       implicit integer*2 (D)
$include: 'runvar.inc'
       integer*2 opt131

99     continue

       call inmenu()

C Write heading. (wrhead.for - iglib.lib)
       call wrhead(0,(720-46*9)/2,35,46,
      +   'Creating and Analysing Performance Index Data.')

C Get chosen option. (domenu.asm - iglib.lib)
       opt131 = DOMENU(131)

C Blank over heading. (util.asm - iglib.lib)
       call WRTSTR(0,(720-46*9)/2,35,46,
      +   '                                                ')

       if (opt131.eq.1) then
           call edpopt()

       elseif (opt131.eq.2) then
           call percal()
           call intper()
           call piplot()
           if (lqgfl) then
               lqafl = .true.
           else
               deafl = .true.
           endif

       elseif (opt131.ne.0) then
           call ERTONE()

       endif

       if (opt131.ne.0) then
           goto 99
       endif

       return
       end
C ***************************************************************************
CH     REVISION HISTORY   :
C      VERSION       BY                  DATE        COMMENT
C        1.0      A. de Waal           06/01/90   Finally Commented
C      FILEND             :
C ***************************************************************************


C
C      FILE               : DIST.FOR
C ***************************************************************************
CM     MODULE NAME        : dist
CA     FUNCTION           : Inject Disturbances into Plant
CS     CALL SEQUENCE      : call dist()
```

```
CI     INPUT PARAMETERS   : None
CO     OUTPUT PARAMETERS  : None
CG     GLOBAL VARIABLES   : Include files:  syst.inc,time.inc
CM     MODULES CALLED     : None
CE     ERROR CONDITIONS   : None
CC     COMMENTS           : Disturbances Injected as Follows:
C                             If dtyp = 1:     Inject Sinusoidal disturbances on
C                                              Outputs
C                             If dtyp = 2:     Refer Gaussian Disturbances on States
C                                              to Outputs by multiplying Disturbance
C                                              Vector by C*inv(sI - A)
C                             If dtyp = 3:     Inject Gaussian disturbances on
C                                              Outputs
C ***************************************************************************
      subroutine dist()
      implicit double precision (r)
$include: 'syst.inc'
$include: 'time.inc'
      integer dima,order
      real zzn(15)

      order = int(syst(4))
      dima = int(syst(5))

      if (dtyp.eq.1) then
          do 30 i = 1, order
              if (dchset(i).eq.0)
     +            zn(i) = dchdat(1,i)*sin(dchdat(2,i)*ti)
30        continue
          do 40 i = 1, dima
              zzn(i) = 0.0
40        continue

      elseif (dtyp.eq.2) then
          do 10 i = 1, dima
              zzn(i) = sngl(rndnrm(dble(0.0),dble(sqrt(dampl)),irand))
10        continue


C Calculate zn(i) = C*inv(sI - A)*zzn(i)

      do 100 i = 1, dima
          swdxdt(i) = 0.0
          xo(i) = xndis(i)
          if (i.le.order) zn(i) = 0.0
100   continue
      do 200 n = 1, 4

C Work out k(n) = f( xndis(ndt)+c2(n)*k(n-1)*dt, ndt+c2(n)*ndt )
C               = a*( xndis(ndt) + c2(n)*k(n-1)*dt )
C               = dxdt(i), i=1, dima
          do 210 i = 1, dima
              dxdt(i) = 0.0
              do 220 j = 1, dima
                  dxdt(i) = dxdt(i) + a(i,j)*xndis(j)
220           continue
              dxdt(i) = dxdt(i) + zzn(i)
210       continue

C Work out swdxdt = k(1) + 2*k(2) + 2*k(3) + k(4)
C                 = c1(n)*k(n)
C                 = c1(n)*dxdt(i), i=1, dima
C                                  c1(i)=1, c1(2)=2, c1(3)=2, c1(4)=1
          do 230 i = 1, dima
              swdxdt(i) = swdxdt(i) + c1(n)*dxdt(i)

C Work out xndis(ndt)+c2(n)*k(n-1)*dt to be used in calculation of next k(n)
C      NOTE: only for n = 1, 2, 3: c2(1)=0.5, c2(2)=0.5, c2(3)=1.0
              if (n.lt.4) then
                  xndis(i) = xo(i) + c2(n)*dxdt(i)*dt

C Work out xndis( (n+1)*dt ) = xndis(n*dt) + ( k(1) + 2*k(2) + 2*k(3) + k(4) )*dt/6
C                            = xndis(n*dt) + swdxdt*dt/6
              else
                  xndis(i) = xo(i) + swdxdt(i)*dt/6
              endif
230       continue
200   continue
      do 300 i = 1, order
          do 310 j = 1, dima
              zn(i) = zn(i) + c(i,j)*xndis(j)
310       continue
300   continue

      else
          do 101 i = 1, order
              zn(i) = sngl(rndnrm(dble(0.0),dble(sqrt(dampl)),irand))
101       continue
          do 401 i = 1, dima
              zzn(i) = 0.0
401       continue
      endif

      return
      end
C ***************************************************************************
CH     REVISION HISTORY   :
C      VERSION       BY                  DATE         COMMENT
C        1.1      A. de Waal           06/01/90    Finally Commented
C      FILEND             :
C ***************************************************************************


C
C      FILE               : DISVOP.FOR
C ***************************************************************************
CN     MODULE NAME        : disvop
CA     FUNCTION           : Display Which Perf. Index Plot to Observe
CS     CALL SEQUENCE      : call disvop(iopt,xpos,ypos)
CI     INPUT PARAMETERS   : iopt:   View option identifier
C                           xpos,ypos:  x and y co-ords where brief to be printed
CO     OUTPUT PARAMETERS  : None
CG     GLOBAL VARIABLES   : None
CM     MODULES CALLED     : Assembler:  WRTSTR
CE     ERROR CONDITIONS   : Returns without action if iopt greater than 4
CC     COMMENTS           : Displays brief on graphics screen 0 regarding which
```

```
C                       performance index data plot selected for plotting
C ***************************************************************************
      subroutine disvop(iopt,xpos,ypos)
      integer iopt,xpos,ypos
      if (iopt.eq.0) call WRTSTR(0,xpos,ypos,42,
     +   'Group Displaying All Perfomance Index Data')
      if (iopt.eq.1) call WRTSTR(0,xpos,ypos,42,
     +   'Display Noise Atten Perfomance Index Data ')
      if (iopt.eq.2) call WRTSTR(0,xpos,ypos,42,
     +   'Display Dist. Atten Perfomance Index Data ')
      if (iopt.eq.3) call WRTSTR(0,xpos,ypos,42,
     +   'Display Sensitivity Perfomance Index Data ')
      if (iopt.eq.4) call WRTSTR(0,xpos,ypos,42,
     +   'Display Effort/Track Perfomance Index Data')
      return
      end
C ***************************************************************************
CH    REVISION HISTORY  :
C     VERSION       BY                  DATE        COMMENT
C       1.1     A. de Waal            06/01/90   Finally Commented
C     FILEND            :
C ***************************************************************************

C
C     FILE              : EXCUSE.FOR
C ***************************************************************************
CN    MODULE NAME       : excuse
CA    FUNCTION          : Utility for Program Development
CS    CALL SEQUENCE     : call excuse()
CI    INPUT PARAMETERS  : None
CO    OUTPUT PARAMETERS : None
CG    GLOBAL VARIABLES  : None
CM    MODULES CALLED    : Assembler:  WRTSTR,INKEY
CE    ERROR CONDITIONS  : None
CC    COMMENTS          : Utility subroutine used as dummy subroutine in
C                         program development
C ***************************************************************************
      subroutine excuse()
      call WRTSTR(0,250,55,25,'ROUTINE NOT YET DEVELOPED')
      key = INKEY(1)
      call WRTSTR(0,250,55,25,'                          ')
      return
      end
C ***************************************************************************
CH    REVISION HISTORY  :
C     VERSION       BY                  DATE        COMMENT
C       1.1     A. de Waal            06/01/90   Finally Commented
C     FILEND            :
C ***************************************************************************

C
C     FILE              : FILOVW.FOR
C ***************************************************************************
CN    MODULE NAME       : filovw
CA    FUNCTION          : Confirm Overwriting of File
CS    CALL SEQUENCE     : call filovw(filnam,yesno)
CI    INPUT PARAMETERS  : filnam: name of file
CO    OUTPUT PARAMETERS : yesno:  overwrite confirmation flag returned
CG    GLOBAL VARIABLES  : Include files:  keys.inc
CM    MODULES CALLED    : Assembler:  LENSTR,ERTONE,WRTSTR,LEVEL,BLKFIL,STRIN
CE    ERROR CONDITIONS  : None
CC    COMMENTS          : Interrogates user on graphics page 0 as to whether or
C                         not the file specified by filnam should be
C                         overwritten - Returns yesno = 'y' if file to be
C                                       overwritten and yesno = 'n' if not
C ***************************************************************************
      subroutine filovw(filnam,yesno)
      implicit integer*2 (S)
$include: 'keys.inc'
      integer*2 key,len
      character*1 yesno
      character*30 filnam
      len = LENSTR(30,filnam)
      call ERTONE()
      call WRTSTR(0,40,110,6,'File :')
      call LEVEL(2)
      call BLKFIL(115,113,(len*9),14)
      call WRTSTR(0,115,110,len,filnam)
      call LEVEL(1)
      call WRTSTR(0,(115+(len*9)),110,16,' already exists.')
      call WRTSTR(0,40,140,26,'Overwrite the file (y/n) ?')
      yesno = 'n'
98    continue
      key = STRIN(0,280,140,1,yesno)
      if (key.ne.retk) goto 98
      return
      end
C ***************************************************************************
CH    REVISION HISTORY  :
C     VERSION       BY                  DATE        COMMENT
C       1.1     A. de Waal            06/01/90   Finally Commented
C     FILEND            :
C ***************************************************************************

C
C     FILE              : GAUIN.FOR
C ***************************************************************************
CN    MODULE NAME       : gauin
CA    FUNCTION          : Obtain Specifications of Gaussian Noise/Dist
CS    CALL SEQUENCE     : call gauln()
CI    INPUT PARAMETERS  : None
CO    OUTPUT PARAMETERS : None
CG    GLOBAL VARIABLES  : Include files:  keys.inc,time.inc
CM    MODULES CALLED    : Fortran   :  wrhead,prtr4,getr4,togopt,togdty
C                         Assembler:  WRTSTR,WIPSCR
CE    ERROR CONDITIONS  : None
CC    COMMENTS          : Interrogates user regarding specifications of
C                         Gaussian noise on outputs and
C                         Gaussian disturbances to process
C ***************************************************************************
      subroutine gauin()
      implicit integer*2 (I,g)
$include: 'keys.inc'
$include: 'time.inc'
```

BIBLIOGRAPHY

1. Anderson B.D.O., Moore J.B. Linear Optimal Control, Prentice-Hall International Editions, Englewood Cliffs, New Jersey, USA, 1971.

2. Anderson B.D.O. Stability Results for Optimal Systems, Electronics Letters, 1969, Vol 5, (22), p. 545.

3. Anton, H. Elementary Linear Algebra, 5th Edition, John Wiley & Sons, USA, 1987

4. Astrom K.J., Wittenmark B. Adaptive Control, Addison-Wesley Publishing Company, USA, 1989.

5. Astrom K.J., Wittenmark B. Computer Controlled Systems, Prentice Hall, 1984.

6. Athans M. The Role and Use of the Stochastic Linear-Quadratic-Gaussian Problem in Control System Design, IEEE Trans. Aut. Contr., 1971, AC-16, (6), pp. 529-552.

7. Borrie J.A., Modern Control Systems - A Manual of Design Methods, Prentice/Hall International, London, 1986.

8. Broussard J.R. A Quadratic Weight Selection Algorithm, IEEE Trans. Aut. Contr., 1982, AC-27, (4), pp. 945-947.

```
      integer*2 npos,key

      call WIPSCR(0)
C Write heading. (wrhead.for - iglib.lib)
      call wrhead(0,(720-46*9)/2,25,43,
     +   'Gaussian Noise Perturbation Specifications.')

      call wrhead(0,30,50,36,
     +   'Specifications for Noise on Outputs.')
      call WRTSTR(0,50,70,33,'Include Noise on Outputs?   : No ')
      call WRTSTR(0,50,85,29,'Covariance of Noise         :')

      call wrhead(0,30,105,38,
     +   'Specifications for Plant Disturbances.')
      call WRTSTR(0,50,125,33,'Include Disturbances?       : No ')
      call WRTSTR(0,50,140,33,'Outputs or Diff of States?  : Out')
      call WRTSTR(0,50,155,29,'Covariance of Disturbances  :')

      call wrhead(0,30,175,42,
     +   'Specifications for Setpoint Perturbations.')
      call WRTSTR(0,50,195,33,'Include Perturbations?      : No ')
      call WRTSTR(0,50,210,29,'Covariance of Perturbations :')

      if (ninc.eq.0) call WRTSTR(0,50+30*9,70,3,'Yes')
      call prtr4(0,(50+30*9),85,7,'(g10.4)',10,nampl)
      if (dinc.eq.0) call WRTSTR(0,50+30*9,125,3,'Yes')
      if (rinc.eq.0) call WRTSTR(0,50+30*9,195,3,'Yes')
      if (dtyp.eq.2) call WRTSTR(0,50,140,33,
     +                           'Outputs or Diff of States?  : Sta')
      call prtr4(0,(50+30*9),155,7,'(g10.4)',10,dampl)
      call prtr4(0,(50+30*9),210,7,'(g10.4)',10,rampl)

      call WRTSTR(0,40,235,36,'RETURN, TAB and cursor keys to move.')
      call WRTSTR(0,40,250,16,'ESC key to exit.')

      npos = 1
199   continue

      if (npos.eq.1) then
          call togopt(ninc,50+30*9,70,key)
          if (ninc.eq.0) ntyp = 0
      elseif (npos.eq.2) then
          if (dinc.eq.0) then
              key = getr4(0,(50+30*9),85,7,'(g10.4)',10,nampl)
              call prtr4(0,(50+30*9),85,7,'(g10.4)',10,nampl)
          else
              key = retk
          endif
      elseif (npos.eq.3) then
          call togopt(dinc,50+30*9,125,key)
          if (dinc.eq.0) dtyp = 0
      elseif (npos.eq.4) then
          if (dinc.eq.0) then
              call togdty(dtyp,50+30*9,140,key)
          else
              key = retk
          endif
      elseif (npos.eq.5) then
          if (dinc.eq.0) then
              key = getr4(0,(50+30*9),155,7,'(g10.4)',10,dampl)
              call prtr4(0,(50+30*9),155,7,'(g10.4)',10,dampl)
          else
              key = retk
          endif
      elseif (npos.eq.6) then
          call togopt(rinc,50+30*9,195,key)
          if (ninc.eq.0) rtyp = 0
      elseif (npos.eq.7) then
          if (rinc.eq.0) then
              key = getr4(0,(50+30*9),210,7,'(g10.4)',10,rampl)
              call prtr4(0,(50+30*9),210,7,'(g10.4)',10,rampl)
          else
              key = retk
          endif
      endif

      if ((key.eq.downk).or.(key.eq.retk).or.(key.eq.tabk)) then
         if (npos.eq.7) then
            npos = 1
         else
            npos = npos + 1
         endif
      elseif ((key.eq.upk).or.(key.eq.rtabk)) then
         if (npos.eq.1) then
            npos = 7
         else
            npos = npos - 1
         endif
      endif
      if (key.ne.esck) goto 199

      call WIPSCR(0)
      return
      end
C ***************************************************************************
CH    REVISION HISTORY  :
C     VERSION       BY                DATE         COMMENT
C       1.1     A. de Waal           06/01/90   Finally Commented
C     FILEND            :
C ***************************************************************************


C
C     FILE              : GETFIL.FOR
C ***************************************************************************
CN    MODULE NAME       : getfil
CA    FUNCTION          : Obtain Filename from User
CS    CALL SEQUENCE     : call getfil(typnam,search,key,usenam)
CI    INPUT PARAMETERS  : typnam: filename used as prompt
C                         search: search string
CO    OUTPUT PARAMETERS : key:    key pressed in exiting from STRIN function
C                         usenam: filename obtained from user
CG    GLOBAL VARIABLES  : Include files:  keys.inc
CM    MODULES CALLED    : Fortran  :  wrhead,dodir
C                         Assembler:  WRTSTR,STRIN
```

```
     +  0.0, 1.0/
      data ((q1(j,i),i=1,4),j=1,4)
     +/ 1.0, 0.0, 0.0, 0.0,
     +  0.0, 1.0, 0.0, 0.0,
     +  0.0, 0.0, 1.0, 0.0,
     +  0.0, 0.0, 0.0, 1.0/
      data ((r1(j,i),i=1,2),j=1,2)
     +/ 1.0, 0.0,
     +  0.0, 1.0/
      data ((kc(j,i),i=1,4),j=1,2)
     +/ 1.0, 0.0, 0.0, 0.0,
     +  0.0, 0.0, 0.0, 1.0/
      data ((q(j,i),i=1,4),j=1,4)
     +/ 1.0, 0.0, 0.0, 0.0,
     +  0.0, 1.0, 0.0, 0.0,
     +  0.0, 0.0, 1.0, 0.0,
     +  0.0, 0.0, 0.0, 1.0/
      data ((r(j,i),i=1,2),j=1,2)
     +/ 1.0, 0.0,
     +  0.0, 1.0/
      data ((kf(j,i),i=1,2),j=1,4)
     +/ 0.0, 0.0,
     +  0.0, 0.0,
     +  0.0, 0.0,
     +  0.0, 0.0/
      data ((ptrsst(j,i),i=1,2),j=1,2) /1.0,0.0,0.0,1.0/

      data deffl /.false./,profl /.false./,grfl /.false./,
     +     lggfl /.false./,desfl /.false./,lgafl /.false./,
     +     deafl /.false./,kfl /.false./,simfl /.false./
      data (((pival(k,j,i),i=1,5),j=1,2),k=1,4) /40*0.0/
      data ((stval(j,i),i=1,5),j=1,2) /10*0.0/
      data (((picom(k,j,i),i=1,5),j=1,2),k=1,4) /40*'UNDEF '/
      data (((piclev(k,j,i),i=1,5),j=1,2),k=1,4) /40*0/
      data (stcom(i),i=1,2) /2*'ENTIRELY UNSTABLE   '/
      data (stclev(i),i=1,2) /2*0/
      data (((pidat(k,j,i),i=1,100),j=1,2),k=1,4) /800*0.0/
      data (simspc(i),i=1,10) /10*0.0/
      data (nspec(i),i=1,5) /5*0.0/
      data (dspec(i),i=1,5) /5*0.0/
      data (sspec(i),i=1,5) /5*0.0/
      data inpath /'c:\progs\optcad\data\    '/
      data outpth /'c:\progs\optcad\data\    '/
      data defdir /'     '/
      data infnam /'                          '/
      data fname /'                          '/
      data mname /'                          '/
      data pernam /'                           '/
      data usenam /'                                                  '/
      data (inpnms(i),i=1,10)
     +            /'Input 1   ','Input 2   ','Input 3   ','Input 4   ',
     +             'Input 5   ','Input 6   ','Input 7   ','Input 8   ',
     +             'Input 9   ','Input 10  '/
      data (outnms(i),i=1,10)
     +            /'Output 1  ','Output 2  ','Output 3  ','Output 4  ',
     +             'Output 5  ','Output 6  ','Output 7  ','Output 8  ',
     +             'Output 9  ','Output 10 '/
      data prjnm /'Heat Exch.'/
      data engnms /'A. de Waal'/
      data (xn(i),i=1,15) /15*0.0/
      data (un(i),i=1,15) /15*0.0/
      data (rn(i),i=1,15) /15*0.0/
      data (vn(i),i=1,15) /15*0.0/
      data (yn(i),i=1,15) /15*0.0/
      data (ynact(i),i=1,15) /15*0.0/
      data (zn(i),i=1,15) /15*0.0/
      data (nn(i),i=1,15) /15*0.0/
      data (xnobs(i),i=1,15) /15*0.0/
      data (ynobs(i),i=1,15) /15*0.0/
      data ti/0.0/,dt/0.5/,tend/2000.0/
      data (c1(i),i=1,4) /1,2,2,1/
      data (c2(i),i=1,3) /0.5,0.5,1.0/
      data obsinc/1/,coninc/1/
      data setplt/1/,inpplt/1/,outplt/1/,ostplt/1/,pstplt/1/
      data (stpinp(i),i=1,5) /5*0/
      data (stpdat(i),i=1,5) /5*0.0/
      data (stpinc(i),i=1,5) /5*0.0/
      data twmax/711/,twstrt/4/,thmax/270/,thstrt/316/,twidth/1/
     +     theigh/1/,txdel/711/,tydel/200/
      data (xtlim(i),i=1,3) /2000.0,0.0,0.0/
      data ((ytlim(i,j),i=1,3),j=1,10) /10.0,-10.0,0.0,
     +                                  10.0,-10.0,0.0,
     +                                  10.0,-10.0,0.0,
     +                                  10.0,-10.0,0.0,
     +                                  10.0,-10.0,0.0,
     +                                  10.0,-10.0,0.0,
     +                                  10.0,-10.0,0.0,
     +                                  10.0,-10.0,0.0,
     +                                  10.0,-10.0,0.0/
      data (tgraph(i),i=1,4) /10,172,340,110/
      data ninc/1/,dinc/1/,rinc/1/,ntyp/0/,dtyp/0/,rtyp/0/
      data nchset/15*1/,dchset/15*1/,rchset/15*1/
      data irand/10/,nampl/1.0/,dampl/1.0/,rampl/1.0/
      data nfre/0.0/,dfre/0.0/,rfre/0.0/
      data nchdat/30*10.0/,dchdat/30*10.0/,rchdat/30*10.0/
      end
C ***************************************************************************
CH    REVISION HISTORY  :
C     VERSION      BY                 DATE        COMMENT
C       1.1     A. de Waal           06/01/90   Finally Commented
C     FILEND            :
C ***************************************************************************



C
C     FILE              : INTPER.FOR
C ***************************************************************************
CN    MODULE NAME       : intper
CA    FUNCTION          : Integrate Perf. Indices over Frequency Range
CS    CALL SEQUENCE     : call intper()
CI    INPUT PARAMETERS  : None
CO    OUTPUT PARAMETERS : None
CG    GLOBAL VARIABLES  : Include files:  syst.inc,perdat.inc
CM    MODULES CALLED    : None
```

```
CE    ERROR CONDITIONS  : None
CC    COMMENTS          : Uses Simpson's Rule for numerical integration to
C                         integrate performance index plots over three
C                         frequency regions, low (w0 to wf/3), intermediate
C                         (wf/3 to 2wf/3) and high (2wf/3 to wf).  The results
C                         are scaled to achieve an indication of the relative
C                         performance over each region.
C ***************************************************************************
      subroutine intper()
$include: 'syst.inc'
$include: 'perdat.inc'
      integer pts,maxpts,grno,chno,regno
      integer maxp(3),intend(3)
      real sumod(4,2,3),sumev(4,2,3),inival(4,2),finval(4,2,3)
      real finfre(3),intc(3),integr(4,2,5)
      real inifre,w,w0,wf,winc

      open(4,file='intper.tes',status='unknown')

      w0 = syst(1)
      wf = syst(2)
      winc = syst(3)
      maxpts = int( (wf-w0)/winc )

C Defining the end pionts for each of the three integration regions
C     - intend(regno) must always be an odd number as
C       integral = K*(inival + 4*sumod + 2*sumev + finval),
C                    (inival at first (therefore odd) point,
C                    finval at final (therefore also odd) point intend.)
      do 50 regno = 1, 3
          maxp(regno) = int(regno*maxpts/3)
          if (mod(maxp(regno),2).eq.0) then
              intend(regno) = maxp(regno) - 1
          else
              intend(regno) = maxp(regno)
          endif
50    continue

C Initializing variables and even and odd sums to zero
      inifre = w0
      w = w0
      do 100 grno = 1, 4
          do 110 chno = 1, 2
              do 120 regno = 1, 3
                  sumod(grno,chno,regno) = 0.0
                  sumev(grno,chno,regno) = 0.0
120           continue
110       continue
100   continue

      do 200 pts = 1, intend(3)
          write(4,'(a,i5)')' Point no:',pts

C If pts even - xi odd  - add to sumod,
C    pts odd  - xi even - add to sumev

          if (mod(pts,2).eq.0) then

              do 300 grno = 1, 4
                  do 310 chno = 1, 2
                      do 320 regno = 1, 3
                          if (pts.lt.intend(regno)) then
                              sumod(grno,chno,regno) =
     +                        sumod(grno,chno,regno) +
     +                        pidat(grno,chno,pts)
                          endif
320                   continue
310               continue
300           continue

          else

              do 400 grno = 1, 4
                  do 410 chno = 1, 2
                      if (pts.eq.1) inival(grno,chno) =
     +                                pidat(grno,chno,pts)
                      do 420 regno = 1, 3
                          if (pts.eq.intend(regno)) then
                              finval(grno,chno,regno) =
     +                        pidat(grno,chno,pts)
                              finfre(regno) = w
                          elseif
     +                    ((pts.lt.intend(regno)).and.(pts.ne.1)) then
                              sumev(grno,chno,regno) =
     +                        sumev(grno,chno,regno) +
     +                        pidat(grno,chno,pts)
                          endif
420                   continue
410               continue
400           continue

          endif

          w = w + winc

200   continue

      do 500 regno = 1, 3
          intc(regno) = (finfre(regno) - inifre)/
     +                  (3*(intend(regno) - 1))
          do 510 grno = 1, 4
              do 520 chno = 1, 2
                  integr(grno,chno,regno) = intc(regno)*
     +            ( inival(grno,chno) + finval(grno,chno,regno) +
     +            4*sumod(grno,chno,regno) + 2*sumev(grno,chno,regno) )
520           continue
510       continue
500   continue


      write(4,*)' Calculating Integrals over extended regions'
      do 444 grno = 1, 4
       write(4,'(1x,a,i5)')'Graph no:',grno
       do 555 chno = 1, 2
          write(4,'(3x,(4g10.4))') (integr(grno,chno,i), i = 1, 4)
555    continue
```

```
444   continue

      write(4,*)' Calculating (a) Integrals,'
      write(4,*)'             (b) Perf Indices over individual regions'
      do 600 grno = 1, 4
          write(4,'(1x,a,i5)')'Graph no:',grno
          do 610 chno = 1, 2
              integr(grno,chno,4) = integr(grno,chno,3)
              pival(grno,chno,4) = integr(grno,chno,4)/
     +                             (finfre(3) - inifre)
              integr(grno,chno,3) = integr(grno,chno,3) -
     +                            integr(grno,chno,2)
              pival(grno,chno,3) = integr(grno,chno,3)/
     +                            (finfre(3) - finfre(2))
              integr(grno,chno,2) = integr(grno,chno,2) -
     +                            integr(grno,chno,1)
              pival(grno,chno,2) = integr(grno,chno,2)/
     +                            (finfre(2) - finfre(1))
              pival(grno,chno,1) = integr(grno,chno,1)/
     +                            (finfre(1) - inifre)
              do 345 regno = 1, 4
                  if (pival(grno,chno,regno).gt.1.0) then
                      picom(grno,chno,regno) = 'V POOR'
                      piclev(grno,chno,regno) = 0
                  elseif (pival(grno,chno,regno).gt.0.5) then
                      picom(grno,chno,regno) = 'POOR  '
                      piclev(grno,chno,regno) = 0
                  elseif (pival(grno,chno,regno).gt.0.25) then
                      picom(grno,chno,regno) = 'FAIR  '
                      piclev(grno,chno,regno) = 1
                  elseif (pival(grno,chno,regno).gt.0.1) then
                      picom(grno,chno,regno) = 'GOOD  '
                      piclev(grno,chno,regno) = 1
                  else
                      picom(grno,chno,regno) = 'V GOOD'
                      piclev(grno,chno,regno) = 1
                  endif
345           continue
              write(4,'(3x,(4g10.4))') (integr(grno,chno,i), i = 1, 4)
              write(4,'(3x,(4a6))') (picom(grno,chno,i), i = 1, 4)
              write(4,'(3x,(4g10.4))') (pival(grno,chno,i), i = 1, 4)
610       continue
600   continue

      if ((stval(1,1).lt.1).or.(stval(2,1).lt.30)) then
          stcom(1) = 'TOTALLY UNSTABLE     '
          stclev(1) = 0
      elseif ((stval(1,1).lt.3).or.(stval(2,1).lt.45)) then
          stcom(1) = 'QUESTIONABLY STABLE '
          stclev(1) = 0
      elseif ((stval(1,1).lt.5).or.(stval(2,1).lt.60)) then
          stcom(1) = 'STABLE               '
          stclev(1) = 1
      else
          stcom(1) = 'TOTALLY STABLE       '
          stclev(1) = 1
      endif

      close(4)
      return
      end
C *************************************************************************
CH    REVISION HISTORY  :
C     VERSION       BY                  DATE        COMMENT
C       1.1     A. de Waal            06/01/90   Finally Commented
C     FILEND            :
C *************************************************************************


C
C     FILE              : JWMAT.FOR
C *************************************************************************
CN    MODULE NAME       : jwmat
CA    FUNCTION          : Create a Complex Matrix jwI
CS    CALL SEQUENCE     : call jwmat(w,dim,jwmatr,jwmati)
CI    INPUT PARAMETERS  : w:  Frequency at which matrix to be created
C                         dim:    Actual dimension of square matrix
CO    OUTPUT PARAMETERS : jwmatr(id,id):  Real zero matrix of maximum dimension
C                                         (id,id)
C                                         and actual dimension (dim,dim)
C                         jwmati(id,id):  Imaginary matrix with value of w down
C                                         diagonal of matrix of maximum
C                                         dimension (id,id)
C                                         and actual dimension (dim,dim)
CG    GLOBAL VARIABLES  : None
CM    MODULES CALLED    : None
CE    ERROR CONDITIONS  : None
CC    COMMENTS          : Returns a matrix jwI, with the value of the frequency
C                         w as the diagonal entries of the matrix with actual
C                         dimension (dim,dim).  The other entries in the matrix
C                         of maximum dimension (id,id) are all zero.  This
C                         matrix is used in the evaluation of the plant
C                         matrix G(s) = C(jwI - A)B at a number of frequency
C                         points.
C *************************************************************************
      subroutine jwmat(w,dim,jwmatr,jwmati)
      integer id
      parameter (id = 15)
      integer dim
      real w
      real jwmatr,jwmati
      dimension jwmatr(id,id),jwmati(id,id)
      do 900 i = 1, id
          do 800 j = 1, id
              jwmatr(i,j) = 0.0
              jwmati(i,j) = 0.0
              if ((i.eq.j).and.(i.le.dim)) jwmati(i,j) = w
800       continue
900   continue
      return
      end
C *************************************************************************
CH    REVISION HISTORY  :
C     VERSION       BY                  DATE        COMMENT
C       1.1     A. de Waal            06/01/90   Finally Commented
C     FILEND            :
```

```
C ********************************************************************************

C
C     FILE              : KEYDAT.FOR
C ********************************************************************************
CN    MODULE NAME       : block data keydat
CA    FUNCTION          : Assign Values of Keys on Keyboard to Varibles
CS    CALL SEQUENCE     : not a subroutine
CI    INPUT PARAMETERS  : n/a
CO    OUTPUT PARAMETERS : n/a
CG    GLOBAL VARIABLES  : Include files:  keys.inc
CM    MODULES CALLED    : n/a
CE    ERROR CONDITIONS  : n/a
CC    COMMENTS          : Assigns the ASCII values of the keys on the keyboard
C                         to common variables that can be used in program units
C ********************************************************************************
      block data keydat
$include: 'keys.inc'
      data        upk    /72/,
     +            downk  /0080/,
     +            leftk  /0075/,
     +            rightk /0077/,
     +            homek  /0071/,
     +            endk   /0079/,
     +            pgupk  /0073/,
     +            pgdnk  /0081/,
     +            esck   /6912/,
     +            retk   /3328/,
     +            tabk   /2304/,
     +            rtabk  /0015/,
     +            sma /24832/,
     +            bga /16640/,
     +            sml /27648/,
     +            bgl /19456/,
     +            smq /28928/,
     +            bgq /20736/,
     +            smm /27904/,
     +            bgm /19712/,
     +            sms /29440/,
     +            bgs /21248/
      end

C ********************************************************************************
CH    REVISION HISTORY  :
C     VERSION       BY                  DATE         COMMENT
C       1.1     A. de Waal            06/01/90   Finally Commented
C     FILEND            :
C ********************************************************************************


C
C     FILE              : LHS.FOR
C ********************************************************************************
CN    NODULE NAME       : lhs
CA    FUNCTION          : Utility Routine for Riccati Equation Solution
CS    CALL SEQUENCE     : call lhs()
CI    INPUT PARAMETERS  : None
CO    OUTPUT PARAMETERS : None
CG    GLOBAL VARIABLES  : Common blocks:  /dims/ in,ir,im
C                                         /ricc/ amod,z
CM    MODULES CALLED    : None
CE    ERROR CONDITIONS  : None
CC    COMMENTS          : Utility routine called in the solution of the
C                         algebraic Riccati equation
C ********************************************************************************
      SUBROUTINE LHS
      PARAMETER (ID=15)
      COMMON /DIMS/ IN,IR,IM /RICC/ AMOD,Z
      DIMENSION AMOD(ID,ID),Z(ID,ID)


      DO 2 I=1,IM
      DO 2 J=1,IM
      Z(I,J)=0.
2     CONTINUE
C
      DO 25 I1=1,(IN-1)
            I2=1
            I3=1
5         IF (I3.LT.I1) THEN
               I2=I2+IN-I3+1
               I3=I3+1
        ENDIF
        IF (I3.LT.I1) GOTO 5

            DO 20 I4=(I1+1),IN
                  I5=0
                  DO 10 I6=1,(I4-1)
                        I5=I5+IN-I6+1
10                CONTINUE
                  I7=I4-1
                  I2=I2+1

                  DO 15 I8=(I5+1),(I5+IN-I4+1)
                     I9=I7+I8-I5
                     Z(I8,I2)=AMOD(I1,I9)
                     Z(I2,I8)=AMOD(I9,I1)
15                CONTINUE

20          CONTINUE
25    CONTINUE
C
      DO 55 I1=1,(IN-2)
            I2=1
            I3=1
30          IF (I3.LT.I1) THEN
               I2=I2+IN-I3+1
               I3=I3+1
            ENDIF
            IF (I3.LT.I1) GOTO 30
            DO 50 I4=(I1+1),(IN-1)
                  I5=1
                  DO 35 I6=1,(I4-1)
                        I5=I5+IN-I6+1
35                CONTINUE
```

```
                        I2=I2+1
                        IF (Z(I5,I2).NE.0.) THEN
                                DO 40 I7=1,(IN-I4)
                                        Z((I5+I7),(I2+I7))=Z(I5,I2)
40                              CONTINUE
                        END IF
                        IF (Z(I2,I5).NE.0.) THEN
                                DO 45 I7=1,(IN-I4)
                                        Z((I2+I7),(I5+I7))=Z(I2,I5)
45                              CONTINUE
                        END IF
50              CONTINUE
55      CONTINUE
C
        I1=0
        DO 70 I2=1,(IN-1)
                I1=I1+IN-I2+1
                DO 65 I3=(I2+1),IN
                        I4=1
                        DO 60 I5=1,(I3-1)
                                I4=I4+IN-I5+1
60                      CONTINUE
                        I6=I1-IN+I3

                        Z(I4,I6)=Z(I4,I6)+AMOD(I2,I3)

65              CONTINUE
70      CONTINUE
C
        I1=1
        DO 85 I2=1,IN
                DO 80 I3=I2,IN
                        I5=I1+I3-I2
                        DO 75 I4=I2,IN
                                I6=I1+I4-I2

                                IF(I3.EQ.I2) THEN
                                        Z(I5,I6)=2*AMOD(I4,I3)
                                ELSE
                                        Z(I5,I6)=AMOD(I4,I3)
                                END IF

                                IF ((I3.EQ.I4).AND.(I3.NE.I2)) THEN
                                        Z(I5,I6)=Z(I5,I6)+AMOD(I2,I2)
                                END IF

75                              CONTINUE
80              CONTINUE
                I1=I1+IN-I2+1
85      CONTINUE
C
        RETURN
        END
C *************************************************************************************
CH      REVISION HISTORY  :
C       VERSION         BY                      DATE            COMMENT
C         0.0       R. Henning                  ?           Creation
C         1.1       A. de Waal                06/01/90      Finally Commented
C       FILEND            :
C *************************************************************************************
C
C       FILE              : MAKNAM.FOR
C *************************************************************************************
CN      MODULE NAME       : maknam
CA      FUNCTION          : Construct Filename, Including Path Name
CS      CALL SEQUENCE     : call maknam(fname,pname,usenam)
CI      INPUT PARAMETERS  : fname - (character*25) The file name.
C                           pname - (character*25) The path name.
CO      OUTPUT PARAMETERS: usenam - (character*50) The final filename.
CG      GLOBAL VARIABLES  : None.
CM      MODULES CALLED    : APPEND (asm), CMPSTR (asm), COPYST (asm), RJUST (asm).
CE      ERROR CONDITIONS  : ?
C
CC      COMMENTS          : Right justifies  both file and path names, then checks
C                           the filename  for  drive and paths. If any found, then
C                           the name is copied  to  'usenam'.  If none found, then
C                           the pathname is check for drive and path. If not found
C                           then default  drive  and  path  is used, otherwise the
C                           specified drive  and  path  is  copied  into 'usenam',
C                           followed by the filename.
C *************************************************************************************
      subroutine maknam(fname,pname,usenam)
      implicit integer*2 (C)
      character*25 fname,pname
      character*50 usenam
      integer*2 pos
      character*1 drv
      usenam = '
      call RJUST(25,fname)
      call RJUST(25,pname)
      pos = CMPSTR(25,fname,1,':')
      if (pos.eq.0) then
         pos = CMPSTR(25,fname,1,'\')
         if (pos.eq.0) then
            pos = CMPSTR(25,pname,1,':')
            if (pos.eq.0) then
               call COPYST(25,pname,0,2,'a:',0,2)
            endif
            call APPEND(50,usenam,25,pname)
            call APPEND(50,usenam,25,fname)
         else
c             call COPYST(50,usenam,0,2,'a:',0,2)
            call APPEND(50,usenam,25,fname)
         endif
      else
         call APPEND(50,usenam,25,fname)
      endif
      return
      end

C *************************************************************************************
CH      REVISION HISTORY :
C       VERSION         BY                      DATE            COMMENT
C         1.00      Ian Fisher                23/06/88      Creation.
C         1.1       A. de Waal                06/01/90      Finally Commented
C       FILEND            :
```

```
C ********************************************************************************

C
C     FILE              : NINDIN.FOR
C ********************************************************************************
CN    MODULE NAME       : nindin
CA    FUNCTION          : Drive Gauss/Sinusoidal Perturb Selection Menu
CS    CALL SEQUENCE     : call nindin()
CI    INPUT PARAMETERS  : None
CO    OUTPUT PARAMETERS : None
CG    GLOBAL VARIABLES  : None
CM    MODULES CALLED    : Fortran  :  inmenu,wrhead,gauin,sinin
C                         Assembler:  DOMENU,WRTSTR,ERTONE
CE    ERROR CONDITIONS  : None
CC    COMMENTS          : Initializes menu screen and drives menu for the
C                         selection of Gaussian or sinusoidal perturbations
C ********************************************************************************
      subroutine nindin()
      implicit integer*2 (D)
      integer*2 opt141

99    continue

      call inmenu()

C Write heading. (wrhead.for - iglib.lib)
      call wrhead(0,(720-62*9)/2,35,62,
     +'Select Noise Type for which Specifications to be Made/Changed.')


C Get chosen option. (domenu.asm - iglib.lib)
      opt141 = DOMENU(141)

C Blank over heading. (util.asm - iglib.lib)
      call WRTSTR(0,(720-62*9)/2,35,62,
     +'                                                              ')

      if (opt141.eq.1) then
          call gauin()

      elseif (opt141.eq.2) then
          call sinin()

      elseif (opt141.ne.0) then
          call ERTONE()

      endif

      if (opt141.ne.0) then
          goto 99
      endif

      return
      end
C ********************************************************************************
CH    REVISION HISTORY  :
C     VERSION       BY                DATE         COMMENT
C       1.1     A. de Waal          06/01/90   Finally Commented
C     FILEND            :
C ********************************************************************************
C
C     FILE              : NOIDIS.FOR
C ********************************************************************************
CN    MODULE NAME       : noidis
CA    FUNCTION          : Specify Sinusoidal Noise/Disturbance Options
CS    CALL SEQUENCE     : call noidis()
CI    INPUT PARAMETERS  : None
CO    OUTPUT PARAMETERS : None
CG    GLOBAL VARIABLES  : Include files:  keys.inc,time.inc,syst.inc
CM    MODULES CALLED    : Fortran  :  wrhead,togopt,getset
C                         Assembler:  WRTSTR,WIPSCR
CE    ERROR CONDITIONS  : None
CC    COMMENTS          : Interrogates user about options for the inclusion
C                         of sinusoidal noise/disturbances.  User specifies
C                         whether noise/disturbances to be included for each
C                         of the channels.  Once these specifications have been
C                         made, the routine calls subroutine getset in which
C                         amplitude and frequency information for each channel
C                         is obtained from the user.
C ********************************************************************************
      subroutine noidis()
      implicit integer*2 (I,g)
$include: 'keys.inc'
$include: 'time.inc'
$include: 'syst.inc'

      integer npos,key
      integer dima,order

      order = syst(4)
      dima = syst(5)

      call WIPSCR(0)
C Write heading. (wrhead.for - iglib.lib)
      call wrhead(0,(720-48*9)/2,25,48,
     +   'Sinusoidal Noise and Disturbance Specifications.')

      call wrhead(0,30,45,36,
     +   'Specifications for Noise on Outputs.')
      call WRTSTR(0,50,70,33,'Include Noise on Outputs?   : No ')

      call wrhead(0,30,135,32,
     +   'Specifications for Disturbances.')
      call WRTSTR(0,50,160,33,'Include Distrubances?       : No ')

      if (ninc.eq.0) call WRTSTR(0,50+30*9,70,3,'Yes')
      if (dinc.eq.0) call WRTSTR(0,50+30*9,160,3,'Yes')

      call WRTSTR(0,40,235,36,'RETURN, TAB and cursor keys to move.')
      call WRTSTR(0,40,250,16,'ESC key to exit.')

      npos = 1
199   continue

      if (npos.eq.1) then
          call togopt(ninc,50+30*9,70,key)
```

```
          if ((ninc.eq.0).and.(key.ne.esck)) then
              ntyp = 1
              call WRTSTR(0,40,235,36,
     +                    '                                   ')
              call WRTSTR(0,40,250,16,'                       ')
              call getset(nchset,nchdat,order,50,90)
              call WRTSTR(0,40,235,36,
     +             'RETURN, TAB and cursor keys to move.')
              call WRTSTR(0,40,250,16,'ESC key to exit.')
          endif
       elseif (npos.eq.2) then
          call togopt(dinc,50+30*9,160,key)
          if ((dinc.eq.0).and.(key.ne.esck)) then
              dtyp = 1
              call WRTSTR(0,40,235,36,
     +                    '                                   ')
              call WRTSTR(0,40,250,16,'                       ')
              call getset(dchset,dchdat,order,50,180)
              call WRTSTR(0,40,235,36,
     +             'RETURN, TAB and cursor keys to move.')
              call WRTSTR(0,40,250,16,'ESC key to exit.')
          endif
       endif

       if ((key.eq.downk).or.(key.eq.retk).or.(key.eq.tabk)) then
          if (npos.eq.2) then
            npos = 1
          else
            npos = npos + 1
          endif
       elseif ((key.eq.upk).or.(key.eq.rtabk)) then
          if (npos.eq.1) then
            npos = 2
          else
            npos = npos - 1
          endif
       endif
       if (key.ne.esck) goto 199

       call WIPSCR(0)
       return
       end
C ********************************************************************************
CH     REVISION HISTORY  :
C      VERSION       BY                    DATE        COMMENT
C        1.1      A. de Waal            06/01/90   Finally Commented
C      FILEND          :
C ********************************************************************************

C
C      FILE              : NOISE.FOR
C ********************************************************************************
CN     MODULE NAME       : noise
CA     FUNCTION          : Generate Gaussian or Sinusoidal Noise
CS     CALL SEQUENCE     : call noise()
CI     INPUT PARAMETERS  : None
CO     OUTPUT PARAMETERS : None
CG     GLOBAL VARIABLES  : Include files:  time.inc,syst.inc
CM     MODULES CALLED    : Fortran  :  rndnrm
CE     ERROR CONDITIONS  : None
CC     COMMENTS          : Generate either:
C                               - A random number each time routine called
C                                 with a Gaussian PDF
C                               - A number which varies sinusoidally with global
C                                 variable ti
C ********************************************************************************
       subroutine noise()
       implicit double precision (r)
$include: 'time.inc'
$include: 'syst.inc'
       integer order

       order = int(syst(4))

       if (ntyp.eq.0) then
            do 100 i = 1, order
               nn(i) = angl(rndnrm(dble(0.0),dble(sqrt(nampl)),irand))
100        continue
       else
            do 300 i = 1, order
              if (nchset(i).eq.0)
     +           nn(i) = nchdat(1,i)*sin(nchdat(2,i)*ti)
300        continue
       endif

       return
       end
C ********************************************************************************
CH     REVISION HISTORY  :
C      VERSION       BY                    DATE        COMMENT
C        1.1      A. de Waal            06/01/90   Finally Commented
C      FILEND          :
C ********************************************************************************


C
C      FILE              : OBSSTA.FOR
C ********************************************************************************
CN     MODULE NAME       : obssta
CA     FUNCTION          : Calculate Observer States
CS     CALL SEQUENCE     : call obssta()
CI     INPUT PARAMETERS  : None
CO     OUTPUT PARAMETERS : None
CG     GLOBAL VARIABLES  : Include files:  syst.inc,time.inc
CM     MODULES CALLED    : None
CE     ERROR CONDITIONS  : None
CC     COMMENTS          : Subroutine uses a fourth order Runge-Kutta algorithm
C                          to perform the first order integrations necessary to
C                          calculate the new estimate of the process states
C                          (xnobs) from the previous estimates, the inputs (un)
C                          and the difference between the observer (ynobs) and
C                          process (yn) outputs.
C ********************************************************************************
       subroutine obssta()
$include: 'syst.inc'
$include: 'time.inc'
```

```
      integer order,dima
      real ynobso(15)

      order = int(syst(4))
      dima = int(syst(5))

      do 100 i = 1, dima
          swdxdt(i) = 0.0
          xo(i) = xnobs(i)
          if (i.le.order) then
              ynobso(i) = ynobs(i)
              ynobs(i) = 0.0
          endif
100   continue
      do 200 n = 1, 4

C Work out k(n) = f( xnobs(ndt)+c2(n)*k(n-1)*dt, ndt+c2(n)*ndt )
C               = a*( xnobs(ndt) + c2(n)*k(n-1)*dt )
C               = dxdt(i), i=1, dima
C
C          dxdt = Af(x) + Bu + Kf(y - yobs)
C
          do 210 i = 1, dima
              dxdt(i) = 0.0
              do 220 j = 1, dima
                  dxdt(i) = dxdt(i) + a(i,j)*xnobs(j)
                  if (j.le.order) dxdt(i) = dxdt(i) + b(i,j)*un(j) +
     +                                      kf(i,j)*(yn(j) - ynobso(j))
220           continue
210       continue

C Work out swdxdt = k(1) + 2*k(2) + 2*k(3) + k(4)
C                 = c1(n)*k(n)
C                 = c1(n)*dxdt(i), i=1, dima
C                                  c1(1)=1, c1(2)=2, c1(3)=2, c1(4)=1
          do 230 i = 1, dima
              swdxdt(i) = swdxdt(i) + c1(n)*dxdt(i)

C Work out xnobs(ndt)+c2(n)*k(n-1)*dt to be used in calculation of next k(n)
C       NOTE: only for n = 1, 2, 3: c2(1)=0.5, c2(2)=0.5, c2(3)=1.0
C
C     f(x) = x + c2(i)*dxdt*dt
C
              if (n.lt.4) then
                  xnobs(i) = xo(i) + c2(n)*dxdt(i)*dt

C Work out xnobs( (n+1)*dt ) = xnobs(n*dt) + ( k(1) + 2*k(2) + 2*k(3) + k(4) )*dt/6
C                            = xnobs(n*dt) + swdxdt*dt/6
              else
                  xnobs(i) = xo(i) + swdxdt(i)*dt/6
              endif
230       continue
200   continue
      do 300 i = 1, order
          do 310 j = 1, dima
              ynobs(i) = ynobs(i) + c(i,j)*xnobs(j)
310       continue
300   continue
      return
      end
C ***************************************************************************
CH    REVISION HISTORY  :
C     VERSION       BY                DATE          COMMENT
C       1.1     A. de Waal          06/01/90   Finally Commented
C     FILEND            :
C ***************************************************************************
C
C     FILE              : OPTCAD.FOR
C ***************************************************************************
CN    MODULE NAME       : optcad
CA    FUNCTION          : Initialize Program and Present Main Menu
CS    CALL SEQUENCE     : Main Program
CI    INPUT PARAMETERS  : None
CO    OUTPUT PARAMETERS : None
CG    GLOBAL VARIABLES  : Include files:  runvar.inc
CM    MODULES CALLED    : Fortran   :  inicad,inmenu,wrhead,data,synth,anal,
C                                      simul
C                         Assembler:  INKEY,INTONE,DOMENU,RETONE,RSTSCR,WIPSCR
C                                     WRTSTR,REDERR
CE    ERROR CONDITIONS  : INT10 GRAPHIX package not resident in the system,
C                         HERCULES card not installed.
CC    COMMENTS          : This program performs the following functions:-
C                           i) Initialises the 2 graphics pages.
C                          ii) Initialises variables.
C                         iii) Initialises the system timer interrupt for
C                              error tone operation.
C                          iv) Drives main program menu.
C
C                         This package requires the following in order to run:
C                           i) IBM AT/80386 (compatable, preferably a 80386)
C                                preferably with math coprocessor
C                          ii) AT to contain at least 512 K of RAM.
C                         iii) AT to contain a HERCULES graphics card.
C                          iv) AT to have access to at least one disk drive.
C                           v) INT10 GRAPHIX package is also required by the
C                              CAD package.
C ***************************************************************************
      program optcad
      implicit integer*2 (D)

$include: 'runvar.inc'
      integer*2 opt1

C Initialize variables used in subroutines. (inicad.for)
      call inicad()

C Initialize tone interrupt. (util.asm)
      call INTONE()

99    continue

C Initialize menu screens and menus. (inmenu.for)
      call inmenu()

C Write heading. (wrhead.for)
      call wrhead(0,(720-54*9)/2,35,54,
```

```
     +   'Analysis of Control Systems using Performance Indices.')


C Get chosen option. (domenu.asm)
      opt1 = DOMENU(1)
C Blank over heading. (util.asm)
      call WRTSTR(0,(720-54*9)/2,35,54,
     +   '                                                  ')

      if (opt1.eq.1) then
          call data()

      elseif (opt1.eq.2) then
          call synth()
          lqgfl = .true.
          qrfl = .false.
          kfl = .true.
          lqafl = .false.
          deafl = .false.

      elseif (opt1.eq.3) then
C Determine Performance Index Data
          call anal()

      elseif (opt1.eq.4) then
          call simul()

      endif

      if (opt1.ne.5) goto 99

C Reset INT24H vector. (derror.asm)
      call REDERR()

C Reset screen to text mode. (screen.asm)
      call RSTSCR()

C Reset tone interrupt vector. (util.asm)
      call RETONE()
      stop
      end
C *************************************************************************
CH    REVISION HISTORY  :
C     VERSION       BY                DATE        COMMENT
C       1.1     A. de Waal           06/01/90   Finally Commented
C     FILEND            :
C *************************************************************************
C
C     FILE              : OPTKC.FOR
C *************************************************************************
CN    MODULE NAME       : optkc
CA    FUNCTION          : Synthesize Optimal LQ Controller Gain Matrix
CS    CALL SEQUENCE     : call optkc(dimerr,conver)
CI    INPUT PARAMETERS  : none
CO    OUTPUT PARAMETERS : dimerr: integer     dimerr = 1  - error in matrix
C                                                dimensioning
C                                             dimerr = 0  - matricies correctly
C                                                dimensioned
C                         conver: integer     conver = n  - no convergence
C                                                after n iterations
C                                             conver = 0  - routine converged
CG    GLOBAL VARIABLES  : Include files:  syst.inc
C                         Common blocks:  /dims/ in,ir,im
C                                         /ricc/ amod,z
C                                         /optmat/ rinv,k,kl,kt,krk,rhs,inf,krq
C                                         /spmats/ rbcr,btct,bkkXc,rkkr
CM    MODULES CALLED    : Fortran  :  invert,trans,mult,lhs,simeq
CE    ERROR CONDITIONS  : Dimension error     dimerr = 1  - error in matrix
C                                                dimensioning
C                                             dimerr = 0  - matricies correctly
C                                                dimensioned
C                         Convergence error   conver = n  - routine not converged
C                                                after n iterations
C                                             conver = 0  - routine converged
CC    COMMENTS          : This routine solves the algebraic Riccati equation
C                         to compute an optimal LQ controller gain matrix (kc)
C                         for the S-space process (a,b,c) with weighting
C                         matrices (q,r).
C *************************************************************************
      subroutine optkc(dimerr,conver)
      parameter (iter = 500)
$include: 'syst.inc'
      integer ir,in,im,i,itno,iter,rows,cols,conver,dimerr
      real sum
      real z,amod
      real rinv,k,kl,kt,krk,rhs,inf,krq
      real rbcr,btct,bkkXc,rkkr
      dimension rinv(id,id),k(id,id),kl(id,id),rbcr(id,id),btct(id,id),
     +  bkkXc(id,id),z(id,id),rkkr(id,id),kt(id,id),krk(id,id),
     +  rhs(id),amod(id,id),inf(id,id),krq(id,id)
      common /dims/in,ir,im /ricc/ amod,z
      common /optmat/ rinv,k,kl,kt,krk,rhs,inf,krq
      common /spmats/ rbcr,btct,bkkXc,rkkr

      dimerr = 0
      conver = 0
C in = no of state variable    = dima = syst(5)
C ir = no of control variables = mord = syst(4)
      in = int(syst(5))
      ir = int(syst(4))

      im = 0
      do 900 i = 1, in
          im = im+in-i+1
900   continue

      do 890 rows = 1, ir
          do 880 cols = 1, in
              k(rows,cols) = kc(rows,cols)
              kl(rows,cols) = k(rows,cols)
880       continue
890   continue

C Calculating:
C       rbcr = inv(rl)*trans(b)
```

```
C This is done in order to later on evaluate:
C       kc = inv(r1)*trans(b)*inf
C          = rbcr*inf
      call invert(r1,rinv,ir,id,2*id)
      call trans(b,btct,in,ir)
      call mult(rinv,btct,rbcr,ir,ir,ir,in,dimerr)

      if (dimerr.eq.1) then
         return
      endif

      do 800 itno = 1, iter

C Calculating:
C     amod = (a - bkkXc)
           call mult (b,k,bkkXc,in,ir,ir,in,dimerr)
                ifl=1
                call add (a,bkkXc,amod,in,in,in,in,ifl,dimerr)
                call lhs()

C Calculating :
C      krq = -(trans(kc)*rl*kc) - ql
                call mult (rl,k,rkkr,ir,ir,ir,in,dimerr)
                call trans (k,kt,ir,in)
                call mult (kt,rkkr,krk,in,ir,ir,in,dimerr)
                ifl=0
                call add (krk,ql,krq,in,in,in,in,ifl,dimerr)


                il=1
                do 20 i2=1,in
                        do 10 i3=i2,in
                                rhs(il)=-1*krq(i2,i3)
                                il=il+1
10                      continue
20              continue

                call simeq(z,rhs,im)

C Calculating inf
                il=1
                do 40 i2=1,in
                        do 30 i3=i2,in
                                inf(i2,i3)=rhs(il)
                                inf(i3,i2)=rhs(il)
                                il=il+1
30                      continue
40              continue

C Calculating:
C       kc = inv(r1)*trans(b)*inf
C          = rbcr*inf
                call mult (rbcr,inf,k,ir,in,in,in,dimerr)

           if (dimerr.eq.1) then
              return
           endif

C Calculating sum of errors to see if algorithm has converged
           sum = 0
           do 790 rows = 1, ir
               do 780 cols = 1, in
                   sum = sum + (k1(rows,cols)-k(rows,cols))**2
780            continue
790        continue
           sum = sum/(ir*in)

C If sum of errors less than 0.0001, consider algorithm to have converged
C thus return assigning calculated value to kc
           if (sum.lt.1.0E-4) then
               do 770 rows = 1, ir
                   do 760 cols = 1, in
                       kc(rows,cols) = k(rows,cols)
760                continue
770            continue

               open(4,file='p.ric')
               do 43 rows = 1, in
                   write(4,*)(inf(rows,cols),cols = 1,in)
43             continue
               close(4)
               return
           endif

C Algorithm has not yet converged, so iterate again
           do 750 rows = 1, ir
               do 740 cols = 1, in
                   k1(rows,cols) = k(rows,cols)
740            continue
750        continue
800    continue

C Sum of errors (sum.ge.1.0E-4) still after (itno=iter) iterations
C so algorithm did not converge on a soution
C thus return without assigning a new value to kc
       conver = itno
       return
       end
C ***************************************************************************
CH     REVISION HISTORY  :
C      VERSION       BY                  DATE         COMMENT
C        0.0     R. Henning              ?
C        1.1     A. de Waal           06/01/90   Finally Commented
C      FILEND          :
C ***************************************************************************
C
C      FILE            : OPTKF.FOR
C ***************************************************************************
CN     NODULE NAME     : optkf
CA     FUNCTION        : Synthesize Optimal Kalman Gain Matrix
CS     CALL SEQUENCE   : call optkf(dimerr,conver)
CI     INPUT PARAMETERS  : none
CO     OUTPUT PARAMETERS : dimerr: integer     dimerr = 1  - error in matrix
C                                                  dimensioning
C                                               dimerr = 0  - matricies correctly
C                                                  dimensioned
```

```
C                          conver: integer    conver = n  - no convergence
C                                                          after n iterations
C                                             conver = 0  - routine converged
CG    GLOBAL VARIABLES   : Include files:  syst.inc
C                          Common blocks:  /dims/ in,ir,im
C                                          /ricc/ amod,z
C                                          /optmat/ rinv,k,k1,kt,krk,rhs,inf,krq
C                                          /spmats/ rbcr,btct,bkkXc,rkkr
CM    MODULES CALLED     : Fortran  :  invert,trans,mult,lhs,simeq
CE    ERROR CONDITIONS   : Dimension error     dimerr = 1  - error in matrix
C                                                          dimensioning
C                                             dimerr = 0  - matricies correctly
C                                                          dimensioned
C                          Convergence error   conver = n  - routine not converged
C                                                          after n iterations
C                                             conver = 0  - routine converged
CC    COMMENTS           : This routine solves the algebraic Riccati equation
C                          to compute an optimal Kalman gain matrix (kf)
C                          for the S-feedback Kalman observer (a,b,c) with
C                          covariance matrices (q1,r1).
C ***************************************************************************
      subroutine optkf(dimerr,conver)
      parameter (iter = 500)
$include: 'syst.inc'
      integer ir,in,im,i,itno,iter,rows,cols,conver,dimerr
      real sum
      real z,amod
      real rinv,k,k1,kt,krk,rhs,inf,krq
      real rbcr,btct,bkkXc,rkkr
      real temp
      dimension rinv(id,id),k(id,id),k1(id,id),rbcr(id,id),btct(id,id),
     +   bkkXc(id,id),z(id,id),rkkr(id,id),kt(id,id),krk(id,id),
     +   rhs(id),amod(id,id),inf(id,id),krq(id,id)
      dimension temp(id,id)
      common /dims/in,ir,im /ricc/ amod,z
      common /optmat/ rinv,k,k1,kt,krk,rhs,inf,krq
      common /spmats/ rbcr,btct,bkkXc,rkkr


      dimerr = 0
      conver = 0
C in = no of state variable    = dima = syst(5)
C ir = no of control variables = mord = syst(4)
      in = int(syst(5))
      ir = int(syst(4))

      im = 0
      do 900 i = 1, in
          im = im+in-i+1
900   continue

      do 890 rows = 1, in
          do 880 cols = 1, ir
              k(rows,cols) = kf(rows,cols)
              k1(rows,cols) = k(rows,cols)
880       continue
890   continue

C Calculating:
C       rbcr = trans(c)*inv(r)
C This is done in order to later on evaluate:
C       kf = inf*trans(c)*inv(r)
C          = inf*rbcr
      call invert(r,rinv,ir,id,2*id)
      call trans(c,btct,ir,in)
      call mult(btct,rinv,rbcr,in,ir,ir,ir,dimerr)

      if (dimerr.eq.1) then
          return
      endif

      do 800 itno = 1, iter

C Calculating:
C     amod = trans(a - bkkXc)
            call mult (k,c,bkkXc,in,ir,ir,in,dimerr)
                ifl=1
                call add (a,bkkXc,amod,in,in,in,in,ifl,dimerr)

           call trans(amod,temp,in,in)

           do 223 i = 1, in
               do 334 j = 1, in
                   amod(i,j) = temp(i,j)
334            continue
223        continue

                call lhs()

C Calculating :
C      rhs = -kf*r*trans(kf)) - q
                call mult (k,r,rkkr,in,ir,ir,ir,dimerr)
                call trans (k,kt,in,ir)
                call mult (rkkr,kt,krk,in,ir,ir,in,dimerr)
                ifl=0
                call add (krk,q,krq,in,in,in,in,ifl,dimerr)

                i1=1
                do 20 i2=1,in
                        do 10 i3=i2,in
                                rhs(i1)=-1*krq(i2,i3)
                                i1=i1+1
10                      continue
20              continue

                call simeq(z,rhs,im)

C Calculating inf
                i1=1
                do 40 i2=1,in
                        do 30 i3=i2,in
                                inf(i2,i3)=rhs(i1)
                                inf(i3,i2)=rhs(i1)
                                i1=i1+1
30                      continue
40              continue
```

```
C Calculating:
C       kf = inf*trans(c)*inv(r)
C          = inf*rbcr
                call mult (inf,rbcr,k,in,in,in,ir,dimerr)

          if (dimerr.eq.1) then
              return
          endif

C Calculating sum of errors to see if algorithm has converged
          sum = 0
          do 790 rows = 1, in
              do 780 cols = 1, ir
                  sum = sum + (k1(rows,cols)-k(rows,cols))**2
780           continue
790       continue
          sum = sum/(in*ir)

C If sum of errors less than 0.0001, consider algorithm to have converged
C thus return assigning calculated value to kf
          if (sum.lt.1.0E-4) then
              do 770 rows = 1, in
                  do 760 cols = 1, ir
                      kf(rows,cols) = k(rows,cols)
760               continue
770           continue

              open(4,file='s.ric')
              do 43 rows = 1, in
                  write(4,*)(inf(rows,cols),cols = 1,in)
43            continue
              close(4)
              return
          endif

C Algorithm has not yet converged, so iterate again
          do 750 rows = 1, in
              do 740 cols = 1, ir
                  k1(rows,cols) = k(rows,cols)
740           continue
750       continue
800   continue

C Sum of errors (sum.ge.1.0E-4) still after (itno=iter) iterations
C so algorithm did not converge on a soution
C thus return without assigning a new value to kf
      conver = itno
      return
      end
C ***************************************************************************
CH    REVISION HISTORY  :
C     VERSION       BY                  DATE        COMMENT
C       0.0     R. Henning              ?
C       1.1     A. de Waal           06/01/90   Finally Commented
C     FILEND            :
C ***************************************************************************

C
C     FILE              : PERCAL.FOR
C ***************************************************************************
CN    MODULE NAME       : percal
CA    FUNCTION          : Calculate Performance Index Data
CS    CALL SEQUENCE     : call percal()
CI    INPUT PARAMETERS  : None
CO    OUTPUT PARAMETERS : None
CG    GLOBAL VARIABLES  : Include files:  syst.inc,permat.inc,perdat.inc
CM    MODULES CALLED    : Fortran   :  togopt,jwmat,add,cinv,cmult,mult,savmat,
C                                      svmax,svmin,qzveca,prtr4,prti2
C                         Assembler:  WRTSTR,ERTONE,LEVEL,INKEY
CE    ERROR CONDITIONS  :
CC    COMMENTS          : Converts LQG control system to format of generalized
C                         MIMO control system by calculating G(jw), K(jw),
C                         F(jw) and P(jw) at each frequency point on the
C                         specified range.  The perfromance index data points
C                         are then also calculated at these frequency points
C                         by determining the maximum and minimum singular
C                         of the matrices indicated in the comments.  These
C                         data points are stored in the global variable pidat
C                         for later plotting and analysis.  The stability
C                         margins are also calculated and stored.  The
C                         poles of the closed loop system are also evaluated
C                         at the end of the algorithm.
C ***************************************************************************
      subroutine percal()
      implicit integer (c,I,s)
$include: 'syst.inc'
$include: 'permat.inc'
$include: 'perdat.inc'
      integer page,key,flush
      integer mord,dima,maxpts,pts,merr,rows,cols,err
      integer merr,ierr
      real w0,wf,winc
      real gmtemp,pmtemp,gm,pm,marg
      real tevecr(ip),teveci(ip)
c      common /vecs/ tevecr,teveci

      open(4,file='percal.tes',status='unknown')

      gmtemp = 1000.0
      pmtemp = 1000.0
      w0 = syst(1)
      wf = syst(2)
      winc = syst(3)
      mord = int(syst(4))
      dima = int(syst(5))
      page = 0

      do 950 rows = 1, ip
          do 940 cols = 1, ip
              zermat(rows,cols) = 0.0
              eyemat(rows,cols) = 0.0
              ptrsst(rows,cols) = 0.0
              tevecr(rows) = 0.0
              teveci(rows) = 0.0
              if (rows.eq.cols) then
```

```
                eyemat(rows,cols) = 1.0
                ptrsst(rows,cols) = 1.0
              endif
940       continue
950   continue

      precom = 1
      call WRTSTR(0,int((720-49*9)/2),150,49,
     +    'Calculate for Zero Steady-State Error? (Y/N):    ')
      call togopt(precom,int((720-49*9)/2)+47*9,150,key)


      w = 0.0
      write(4,*) ' Calculating Precomp. for Zero S-State Error'
      write(4,*) ' w ='
      write(4,'(1x,g10.2)')w

C Calculating (jwI - A)
      call jwmat(w,dima,jwmatr,jwmati)
      call add(jwmatr,a,jamatr,dima,dima,dima,dima,1,merr)
      if (merr.ne.0) goto 101
      call add(jwmati,zermat,jamati,dima,dima,dima,dima,1,merr)
      if (merr.ne.0) goto 101

C Calculating G = C*inv(jwI - A)*B
      call cinv(jamatr,jamati,tematr,temati,dima,ip,2*ip)
      call cmult(c,zermat,tematr,temati,tema2r,tema2i,
     +           mord,dima,dima,dima,merr)
      if (merr.ne.0) goto 101
      call cmult(tema2r,tema2i,b,zermat,gmatr,gmati,
     +           mord,dima,dima,mord,merr)
      if (merr.ne.0) goto 101

C Calculating phi = inv(jwI - A + Kf*C)
      call mult(kf,c,kfcmat,dima,mord,mord,dima,merr)
      if (merr.ne.0) goto 101
      call add(jamatr,kfcmat,tematr,dima,dima,dima,dima,0,merr)
      if (merr.ne.0) goto 101
      call cinv(tematr,jamati,fimatr,fimati,dima,ip,2*ip)

C Calculating P = inv(I + Kc*phi*B)
      call cmult(kc,zermat,fimatr,fimati,tematr,temati,
     +           mord,dima,dima,dima,merr)
      if (merr.ne.0) goto 101
      call cmult(tematr,temati,b,zermat,tema2r,tema2i,
     +           mord,dima,dima,mord,merr)
      if (merr.ne.0) goto 101
      call add(eyemat,tema2r,tematr,mord,mord,mord,mord,0,merr)
      if (merr.ne.0) goto 101
      call add(zermat,tema2i,temati,mord,mord,mord,mord,0,merr)
      if (merr.ne.0) goto 101
      call cinv(tematr,temati,pmatr,pmati,mord,ip,2*ip)

C Calculating M = F = inv(I + Kc*phi*B)*Kc*phi*Kf
C                   = P*Kc*phi*Kf
      call cmult(pmatr,pmati,kc,zermat,tematr,temati,
     +           mord,dima,dima,dima,merr)
      if (merr.ne.0) goto 101
      call cmult(tematr,temati,fimatr,fimati,tema2r,tema2i,
     +           mord,dima,dima,dima,merr)
      if (merr.ne.0) goto 101
      call cmult(tema2r,tema2i,kf,zermat,mmatr,mmati,
     +           mord,dima,dima,mord,merr)
      if (merr.ne.0) goto 101

C Calculating L2 = G*M
      call cmult(gmatr,gmati,mmatr,mmati,l2matr,l2mati,
     +           mord,mord,mord,mord,merr)
      if (merr.ne.0) goto 101

C Calculating S2 = inv(I + L2)
      call add(eyemat,l2matr,tematr,mord,mord,mord,mord,0,merr)
      if (merr.ne.0) goto 101
      call add(zermat,l2mati,temati,mord,mord,mord,mord,0,merr)
      if (merr.ne.0) goto 101
      call cinv(tematr,temati,s2matr,s2mati,mord,ip,2*ip)

C Calculating Ptr = inv(S2*G*P)   : s = 0
      call cmult(s2matr,s2mati,gmatr,gmati,tematr,temati,
     +           mord,mord,mord,mord,merr)
      if (merr.ne.0) goto 101
      call cmult(tematr,temati,pmatr,pmati,tema2r,tema2i,
     +           mord,mord,mord,mord,merr)
      if (merr.ne.0) goto 101
      call cinv(tema2r,tema2i,ptrsst,tema2i,mord,ip,2*ip)

C Saving Precompensator Matrix to File
      err = savmat(mord,mord,ptrsst,
     +     'Re(Precomp Matrix)       ',4)
      err = savmat(mord,mord,tema2i,
     +     'Im(Precomp Matrix)       ',4)
      if (err.ne.0) then
          call ERTONE()
          call LEVEL(0)
          call WRTSTR(0,50,100,59,
     +    '***ERROR: Could not write Precomp matrix to file percal.tes')
          call LEVEL(1)
          key = INKEY(1)
          call WRTSTR(0,50,100,59,
     +    '                                                          ')
      endif


      w = w0
      maxpts = int( (wf-w0)/winc )

      call WRTSTR(0,int((720-32*9)/2),240,31,
     +    'Hit ESC key to Stop Calculating')
      flush = INKEY(2)
      do 900 pts = 1, maxpts
          call LEVEL(2)
          call WRTSTR(0,int((720-19*9)/2),200,19,'BUSY....Please Wait')
          call LEVEL(1)
          write(4,*) ' w, pts ='
          write(4,'(1x,g10.2,i5)')w,pts
```

```
C Calculating (jwI - A)
          call jwmat(w,dima,jwmatr,jwmati)
          call add(jwmatr,a,jamatr,dima,dima,dima,dima,1,merr)
          call add(jwmati,zermat,jamati,dima,dima,dima,dima,1,merr)
          if (merr.ne.0) goto 101

C Calculating G = C*inv(jwI - A)*B
          call cinv(jamatr,jamati,tematr,temati,dima,ip,2*ip)
          call cmult(c,zermat,tematr,temati,tema2r,tema2i,
     +               mord,dima,dima,merr)
          if (merr.ne.0) goto 101
          call cmult(tema2r,tema2i,b,zermat,gmatr,gmati,
     +               mord,dima,dima,mord,merr)
          if (merr.ne.0) goto 101

C Calculating phi = inv(jwI - A + Kf*C)
          call mult(kf,c,kfcmat,dima,mord,mord,dima,merr)
          if (merr.ne.0) goto 101
          call add(jamatr,kfcmat,tematr,dima,dima,dima,dima,0,merr)
          if (merr.ne.0) goto 101
          call cinv(tematr,jamati,fimatr,fimati,dima,ip,2*ip)

C Calculating P = inv(I + Kc*phi*B)
          call cmult(kc,zermat,fimatr,fimati,tematr,temati,
     +               mord,dima,dima,dima,merr)
          if (merr.ne.0) goto 101
          call cmult(tematr,temati,b,zermat,tema2r,tema2i,
     +               mord,dima,dima,mord,merr)
          if (merr.ne.0) goto 101
          call add(eyemat,tema2r,tematr,mord,mord,mord,mord,0,merr)
          if (merr.ne.0) goto 101
          call add(zermat,tema2i,temati,mord,mord,mord,mord,0,merr)
          if (merr.ne.0) goto 101
          call cinv(tematr,temati,pmatr,pmati,mord,ip,2*ip)

C Calculating M = P = inv(I + Kc*phi*B)*Kc*phi*Kf
C                   = P*Kc*phi*Kf
          call cmult(pmatr,pmati,kc,zermat,tematr,temati,
     +               mord,dima,dima,dima,merr)
          if (merr.ne.0) goto 101
          call cmult(tematr,temati,fimatr,fimati,tema2r,tema2i,
     +               mord,dima,dima,dima,merr)
          if (merr.ne.0) goto 101
          call cmult(tema2r,tema2i,kf,zermat,mmatr,mmati,
     +               mord,dima,dima,mord,merr)
          if (merr.ne.0) goto 101

C Calculating L2 = G*M
          call cmult(gmatr,gmati,mmatr,mmati,l2matr,l2mati,
     +               mord,mord,mord,mord,merr)
          if (merr.ne.0) goto 101

C Calculating L1 = M*G
          call cmult(mmatr,mmati,gmatr,gmati,l1matr,l1mati,
     +               mord,mord,mord,mord,merr)
          if (merr.ne.0) goto 101

C Calculating S2 = inv(I + L2)
          call add(eyemat,l2matr,tematr,mord,mord,mord,mord,0,merr)
          if (merr.ne.0) goto 101
          call add(zermat,l2mati,temati,mord,mord,mord,mord,0,merr)
          if (merr.ne.0) goto 101
          call cinv(tematr,temati,s2matr,s2mati,mord,ip,2*ip)

C Calculating S1 = inv(I + L1)
          call add(eyemat,l1matr,tematr,mord,mord,mord,mord,0,merr)
          if (merr.ne.0) goto 101
          call add(zermat,l1mati,temati,mord,mord,mord,mord,0,merr)
          if (merr.ne.0) goto 101
          call cinv(tematr,temati,s1matr,s1mati,mord,ip,2*ip)

C Calculating T2 = I - S2
          call add(eyemat,s2matr,t2matr,mord,mord,mord,mord,1,merr)
          if (merr.ne.0) goto 101
          call add(zermat,s2mati,t2mati,mord,mord,mord,mord,1,merr)
          if (merr.ne.0) goto 101

C Calculating T1 = I - S1
          call add(eyemat,s1matr,t1matr,mord,mord,mord,mord,1,merr)
          if (merr.ne.0) goto 101
          call add(zermat,s1mati,t1mati,mord,mord,mord,mord,1,merr)
          if (merr.ne.0) goto 101

C Calculating W = S1*M
          call cmult(s1matr,s1mati,mmatr,mmati,wmatr,wmati,
     +               mord,mord,mord,mord,merr)
          if (merr.ne.0) goto 101

          if (precom.eq.1) then
C Calculating Effort = S1*P
              call cmult(s1matr,s1mati,pmatr,pmati,efmatr,efmati,
     +                   mord,mord,mord,mord,merr)
              if (merr.ne.0) goto 101
          else
C Calculating Effort = S1*P*Ptr
              call cmult(s1matr,s1mati,pmatr,pmati,tematr,temati,
     +                   mord,mord,mord,mord,merr)
              if (merr.ne.0) goto 101
              call cmult(tematr,temati,ptrsst,zermat,efmatr,efmati,
     +                   mord,mord,mord,mord,merr)
              if (merr.ne.0) goto 101
          endif

          if (precom.eq.1) then
C Calculating Error = (I - S2*G*P)
              call cmult(s2matr,s2mati,gmatr,gmati,tematr,temati,
     +                   mord,mord,mord,mord,merr)
              if (merr.ne.0) goto 101
              call cmult(tematr,temati,pmatr,pmati,tema2r,tema2i,
     +                   mord,mord,mord,mord,merr)
              if (merr.ne.0) goto 101
              call add(eyemat,tema2r,ermatr,mord,mord,mord,mord,1,merr)
              if (merr.ne.0) goto 101
              call add(zermat,tema2i,ermati,mord,mord,mord,mord,1,merr)
              if (merr.ne.0) goto 101
          else
```

```
C Calculating Error = (I - S2*G*P*Ptr)
            call cmult(s2matr,s2mati,gmatr,gmati,tematr,temati,
     +                 mord,mord,mord,mord,merr)
            if (merr.ne.0) goto 101
            call cmult(tematr,temati,pmatr,pmati,tema2r,tema2i,
     +                 mord,mord,mord,mord,merr)
            if (merr.ne.0) goto 101
            call cmult(tema2r,tema2i,ptrsst,zermat,tematr,temati,
     +                 mord,mord,mord,mord,merr)
            if (merr.ne.0) goto 101
            call add(eyemat,tematr,ermatr,mord,mord,mord,mord,1,merr)
            if (merr.ne.0) goto 101
            call add(zermat,temati,ermati,mord,mord,mord,mord,1,merr)
            if (merr.ne.0) goto 101
          endif

C Assigning Values to Performance Index Data
C
C      - Sensor Noise Rejection
            call svmax(wmatr,wmati,eyemat,eyemat,eyemat,eyemat,mord,
     +                 pidat(1,1,pts),ierr)
            if (ierr.ne.0) goto 102

            write(4,*)' pidat(1,1,pts) ='
            write(4,*)pidat(1,1,pts)
            call svmax(t2matr,t2mati,eyemat,eyemat,eyemat,eyemat,mord,
     +                 pidat(1,2,pts),ierr)
            if (ierr.ne.0) goto 102
            write(4,*)' pidat(1,2,pts) ='
            write(4,*)pidat(1,2,pts)

C      - Distrubance Attenuation
            pidat(2,1,pts) = pidat(1,1,pts)
            write(4,*)' pidat(2,1,pts) ='
            write(4,*)pidat(2,1,pts)
            call svmax(s2matr,s2mati,eyemat,eyemat,eyemat,eyemat,mord,
     +                 pidat(2,2,pts),ierr)
            if (ierr.ne.0) goto 102
            write(4,*)' pidat(2,2,pts) ='
            write(4,*)pidat(2,2,pts)

C      - Sensitivity to Changes in Process Model
            call svmax(t1matr,t1mati,eyemat,eyemat,eyemat,eyemat,mord,
     +                 pidat(3,1,pts),ierr)
            if (ierr.ne.0) goto 102
c            pidat(3,1,pts) = pidat(1,2,pts)
            write(4,*)' pidat(3,1,pts) ='
            write(4,*)pidat(3,1,pts)
            pidat(3,2,pts) = pidat(2,2,pts)
            write(4,*)' pidat(3,2,pts) ='
            write(4,*)pidat(3,2,pts)

C      - Control Effort and Tracking Error
            call svmax(efmatr,efmati,eyemat,eyemat,eyemat,eyemat,mord,
     +                 pidat(4,1,pts),ierr)
            if (ierr.ne.0) goto 102
            write(4,*)' pidat(4,1,pts) ='
            write(4,*)pidat(4,1,pts)
            call svmax(ermatr,ermati,eyemat,eyemat,eyemat,eyemat,mord,
     +                 pidat(4,2,pts),ierr)
            if (ierr.ne.0) goto 102
            write(4,*)' pidat(4,2,pts) ='
            write(4,*)pidat(4,2,pts)

c            pidat(4,1,pts) = 1/svmin(efmatr,efmati,mord)
c            pidat(4,2,pts) = 1/svmin(ermatr,ermati,mord)

C      - Stability Margins (Gain and Phase Margins)
C           Calculate marg = svmin(I + inv(L1))
            call cinv(l1matr,l1mati,tematr,temati,mord,ip,2*ip)
            call add(eyemat,tematr,tema2r,mord,mord,mord,mord,0,merr)
            if (merr.ne.0) goto 101
            call add(zermat,temati,tema2i,mord,mord,mord,mord,0,merr)
            if (merr.ne.0) goto 101
            call svmin(tema2r,tema2i,eyemat,eyemat,eyemat,eyemat,mord,
     +                 marg,ierr)
            if (ierr.ne.0) goto 102
            write(4,'(3X,a,g10.4)')' Marg = ',marg
            gm = 20*log10(1+marg)
            write(4,*)' Gain margin'
            write(4,*)gm
            if (gm.lt.gmtemp) then
                gmtemp = gm
            endif
            if (marg.gt.2.0) marg = 2.0
            if (marg.lt.-2.0) marg = -2.0
            pm = 2*asin(0.5*marg)*360/(2*3.141592654)
            write(4,*)' Phase margin'
            write(4,*)pm
            if (pm.lt.pmtemp) then
                pmtemp = pm
            endif

            w = w + winc

            key = INKEY(4)
            if (key.eq.6912) then
                iout = 1
                call WRTSTR(0,int((720-19*9)/2),220,19,
     +                'Stop Calculating? :')
                call togopt(iout,int((720-19*9)/2)+20*9,220,flush)
                flush = INKEY(2)
                if (iout.eq.0) goto 901
            endif

900     continue

901     continue

        gm = gmtemp
        write(4,*)' Minimum Gain margin'
        write(4,*)gm
        stval(1,1) = gm
        pm = pmtemp
        write(4,*)' Minimum Phase margin'
        write(4,*)pm
```

```
      stval(2,1) = pm
      close(4)

C Calculate Pole Positions of State Feedback Controller
C (excluding Kalman Filter)
C Calculating eigenvalues( Ac = (A - B*Kc) )
      call mult(b,kc,tema2r,dima,mord,mord,dima,merr)
      if (merr.ne.0) goto 101
      call add(a,tema2r,tematr,dima,dima,dima,dima,1,merr)
      if (merr.ne.0) goto 101
      call qzveca(ip,tematr,zermat,tevecr,teveci,tema2r,tema2i,ierr)
      if (ierr.ne.0) goto 101

      call WIPSCR(0)
      call DISP(0)
      call wrhead(0,40,50,32,
     +   'Poles of F/B Controller Dynamics')
      do 200 i = 1, dima
          call prtr4(0,50,55+i*20,7,'(g10.4)',10,tevecr(i))
          call WRTSTR(0,50+11*9,55+i*20,3,' +j')
          call prtr4(0,50+14*9,55+i*20,7,'(g10.4)',10,teveci(i))
200   continue


C Calculate Pole Positions of Kalman Filter
C (excluding State Feedback Controller)
C Calculating eigenvalues( Af = (A - Kf*C) )
      call mult(kf,c,tema2r,dima,mord,mord,dima,merr)
      if (merr.ne.0) goto 101
      call add(a,tema2r,tematr,dima,dima,dima,dima,1,merr)
      if (merr.ne.0) goto 101
      call qzveca(ip,tematr,zermat,tevecr,teveci,tema2r,tema2i,ierr)
      if (ierr.ne.0) goto 101

      call wrhead(0,400,50,28,
     +   'Poles Kalman Filter Dynamics')
      do 300 i = 1, dima
          call prtr4(0,410,55+i*20,7,'(g10.4)',10,tevecr(i))
          call WRTSTR(0,410+11*9,55+i*20,3,' +j')
          call prtr4(0,410+14*9,55+i*20,7,'(g10.4)',10,teveci(i))
300   continue

      key = INKEY(1)


100   goto 103

101   continue
      call ERTONE()
      call LEVEL(0)
      call WRTSTR(page,100,280,30,'Matrix Dimension Error Ocurred')
      call LEVEL(1)
      key = INKEY(1)
102   continue
      call ERTONE()
      call LEVEL(0)
      call WRTSTR(page,100,280,44,
     +   'No Convergence - No of Iterations Performed:')
      call prti2(page,(100+44*9),280,4,'(15)',5,ierr)
      call LEVEL(1)
      key = INKEY(1)

103   continue



      return
      end
C ***************************************************************************
CH    REVISION HISTORY  :
C     VERSION       BY                DATE        COMMENT
C       1.1     A. de Waal          06/01/90   Finally Commented
C     FILEND            :
C ***************************************************************************
C
C     FILE              : PEROPT.FOR
C ***************************************************************************
CN    MODULE NAME       : peropt
CA    FUNCTION          : Obtain Performance Index Plotting Options
CS    CALL SEQUENCE     : call peropt()
CI    INPUT PARAMETERS  : None
CO    OUTPUT PARAMETERS : None
CG    GLOBAL VARIABLES  : Include files:  syst.inc,pldat.inc,keys.inc
CM    MODULES CALLED    : Fortran  :  wrhead,prtr8,getr8,disvop,togvop
C                         Assembler:  WRTSTR,WIPSCR,ERTONE,WIPSCR
CE    ERROR CONDITIONS  : None
CC    COMMENTS          : Interrogate user on graphics page 0 to obtain
C                         performance index plotting options ,ie:
C                         required group of performance index plots
C                         and scales of axes.
C ***************************************************************************
      subroutine peropt()
      implicit integer (g)
$include: 'syst.inc'
$include: 'pldat.inc'
$include: 'keys.inc'
      integer stpt,enpt,key,npos,maxpts,vop
      common /plpts/ stpt,enpt
      common /view/ vop

      call WIPSCR(0)
      call wrhead(0,(720-35*9)/2,35,35,
     +   'Performance Index Plotting Options.')

      call wrhead(0,40,60,37,
     +   'Vertical Axes: Performance Index Data')
      call WRTSTR(0,50,80,28,
     +   'Maximum Value:            dB')
      call WRTSTR(0,50,95,28,
     +   'Minimum Value:            dB')

      call wrhead(0,40,120,33,
     +   'Horizontal Axes: Frequency Points')
      call WRTSTR(0,50,140,37,
     +   'Maximum Frequency:             rad/sec')
```

```
      call WRTSTR(0,50,155,37,
     +   'Minimum Frequency:          rad/sec')
      call WRTSTR(0,50,170,12,'View Option:')

      w0 = syst(1)
      wf = syst(2)
      winc = syst(3)
      maxpts = int((wf - w0)/winc)

      do 100 i = 1, 4
          xaxdat(i,1) = dble(wf-(maxpts-enpt)*winc)
          if (xaxdat(i,1).gt.dble(wf)) xaxdat(i,1) = dble(wf)
          xaxdat(i,2) = dble(w0+(stpt-1)*winc)
          if (xaxdat(i,2).lt.dble(w0)) xaxdat(i,2) = dble(w0)
          xaxdat(i,3) = xaxdat(i,2)
100   continue

      npos = 1

199   continue

      call prtr8(0,50+15*9,80,7,'(g10.4)',10,yaxdat(1,1))
      call prtr8(0,50+15*9,95,7,'(g10.4)',10,yaxdat(1,2))
      call prtr8(0,50+19*9,140,7,'(g10.4)',10,xaxdat(1,1))
      call prtr8(0,50+19*9,155,7,'(g10.4)',10,xaxdat(1,2))
      call disvop(vop,int(50+13*9),170)
      call WRTSTR(0,40,235,36,'RETURN, TAB and cursor keys to move.')
      call WRTSTR(0,40,250,16,'ESC key to exit.')

      if (npos.eq.1) then
          key = getr8(0,50+15*9,80,7,'(g10.4)',10,yaxdat(1,1))
          do 200 i = 2, 4
              yaxdat(i,1) = yaxdat(1,1)
200       continue
      elseif (npos.eq.2) then
198       continue
          key = getr8(0,50+15*9,95,7,'(g10.4)',10,yaxdat(1,2))
          if (yaxdat(1,2).ge.yaxdat(1,1)) then
              call WRTSTR(0,150,220,45,
     +             '*** Error : Minimum must be Less than Maximum')
              call ERTONE()
          else
              call WRTSTR(0,150,220,45,
     +             '                                              ')
          endif
          if (yaxdat(1,2).ge.yaxdat(1,1)) goto 198
          do 300 i = 2, 4
              yaxdat(i,2) = yaxdat(1,2)
300       continue
      elseif (npos.eq.3) then

C Editing the Maximum Plot Frequency

196       continue
          key = getr8(0,50+19*9,140,7,'(g10.4)',10,xaxdat(1,1))
          if (sngl(xaxdat(1,1)).gt.wf) then
              call WRTSTR(0,150,220,50,
     +             '*** Error : Maximum Plot must be < Maximum Created')
              xaxdat(1,1) = dble(wf)
              call ERTONE()
          else
              call WRTSTR(0,150,220,50,
     +             '                                                  ')
          endif
          if (sngl(xaxdat(1,1)).gt.wf) goto 196
          do 400 i = 2, 4
              xaxdat(i,1) = xaxdat(1,1)
400       continue
          enpt = maxpts - int( (wf-xaxdat(1,1))/winc )
      elseif (npos.eq.4) then
195       continue
          key = getr8(0,50+19*9,155,7,'(g10.4)',10,xaxdat(1,2))
          if (xaxdat(1,2).ge.xaxdat(1,1)) then
              call WRTSTR(0,150,220,45,
     +             '*** Error : Minimum must be Less than Maximum')
              call ERTONE()
          elseif (sngl(xaxdat(1,2)).lt.w0) then
              call WRTSTR(0,150,220,50,
     +             '*** Error : Minimum Plot must be > Minimum Created')
              xaxdat(1,2) = dble(w0)
              call ERTONE()
          else
              call WRTSTR(0,150,220,50,
     +             '                                                  ')
          endif
          if (xaxdat(1,2).ge.xaxdat(1,1)) goto 195
          if (sngl(xaxdat(1,2)).lt.w0) goto 195
          xaxdat(1,3) = xaxdat(1,2)
          do 500 i = 2, 4
              xaxdat(i,2) = xaxdat(1,2)
              xaxdat(i,3) = xaxdat(1,2)
500       continue
          stpt = int( (xaxdat(1,2)-w0)/winc ) + 1
      elseif (npos.eq.5) then
         call togvop(vop,int(50+13*9),170,key)
      endif


      if ((key.eq.downk).or.(key.eq.retk).or.(key.eq.tabk)) then
         if (npos.eq.5) then
            npos = 1
         else
            npos = npos + 1
         endif
      elseif ((key.eq.upk).or.(key.eq.rtabk)) then
         if (npos.eq.1) then
            npos = 5
         else
            npos = npos - 1
         endif
      endif
      if (key.ne.esck) goto 199

      call WIPSCR(0)
      return
      end
```

```
C ***************************************************************************
CH    REVISION HISTORY  :
C     VERSION       BY                DATE        COMMENT
C       1.1     A. de Waal          06/01/90   Finally Commented
C     FILEND            :
C ***************************************************************************


C
C     FILE              : PERPLT.FOR
C ***************************************************************************
CN    MODULE NAME       : perplt
CA    FUNCTION          : View Specific Perf Index Data Plot and Analysis
CS    CALL SEQUENCE     : call perplt(vop)
CI    INPUT PARAMETERS  : vop:    integer - Indicates which group of
C                                           performance index data to be
C                                           plotted
C                                   vop = 1: Noise attenuation
C                                       = 2: Disturbance attenuation
C                                       = 3: Sensitivity
C                                       = 4: Control Effort and Setp. Tracking
C
CO    OUTPUT PARAMETERS : None
CG    GLOBAL VARIABLES  : Include files:  syst.inc,perdat.inc,pldat.inc
CM    MODULES CALLED    : Fortran  :  wrhead, dwaxes, drawpt, drwlet
C                         Assembler:  WIPSCR, MOVE, DLINE, WRTSTR, LEVEL
CE    ERROR CONDITIONS  : None
CC    COMMENTS          : User-specified performance index data plots are
C                         plotted on graphics page 1, and an analysis of all
C                         performance indices is presented on graphics page 0.
C ***************************************************************************
      subroutine perplt(vop)
      implicit integer*2 (D,I)
$include: 'syst.inc'
$include: 'perdat.inc'
$include: 'pldat.inc'
      integer vop
      integer stpt,enpt
      integer*2 i,j,pos,key,xpos,ypos,pts,com
      integer*2 page,xtype/1/,dots/1/,join/2/,nums/1/
      integer*2 graph(4)
      real xax(100), w0, wf, winc
      real*8 xp(3),yp(3)
      real*8 scale(2),centre(4)
      common /plpts/ stpt,enpt

      page = 1
      call GPAGE(page)
      call CLRSCR()
      call wrhead(page,((720-45*9)/2),22,45,
     +   'Performance Index Plot: Press F3 for analysis')
      call BOX(page,2,346,716,344)
      call BOX(page,4,344,712,340)
      call MOVE(4,30)
      call DLINE(716,30)
      call MOVE(4,300)
      call DLINE(716,300)
      call WRTSTR(page,20,335,26,
     +   'Trace 1: Input Performance')
      call WRTSTR(page,376,335,27,
     +   'Trace 2: Output Performance')
      call LEVEL(0)
      if (vop.eq.1) call WRTSTR(page,((720-44*9)/2),315,44,
     +   'Noise Attenuation Perfomance Index Data Plot')
      if (vop.eq.2) call WRTSTR(page,((720-50*9)/2),315,50,
     +   'Disturbance Attenuation Perfomance Index Data Plot')
      if (vop.eq.3) call WRTSTR(page,((720-38*9)/2),315,38,
     +   'Sensitivity Perfomance Index Data Plot')
      if (vop.eq.4) call WRTSTR(page,((720-59*9)/2),315,59,
     +   'Control Effort/Setpoint Tracking Perfomance Index Data Plot')
      call LEVEL(1)

      w0 = syst(1)
      wf = syst(2)
      winc = syst(3)
      npts = int((wf - w0)/winc)
      w = w0

      do 901 pts = 1, npts
          xax(pts) = w
          w = w + winc
901   continue


      nums = 1
      xtype = 1

      grno = vop
c      do 900 grno = 1,4
          do 800 axinf = 1,3
c              if (axinf.eq.1) xaxdat(grno,axinf) = DBLE(wf)
c              if (axinf.eq.2) xaxdat(grno,axinf) = DBLE(w0)
c              if (axinf.eq.3) xaxdat(grno,axinf) = DBLE(w0)
              xp(axinf) = xaxdat(grno,axinf)
              yp(axinf) = yaxdat(grno,axinf)
800       continue
          graph(1) = 10
          graph(2) = 285
          graph(3) = 640
          graph(4) = 250
          call WRTSTR(page,(650-9*9),285,9,'[rad/sec]')
          call WRTSTR(page,60,45,30,'[Decibels = 20*Log(Magnitude)]')
          call dwaxes(page,nums,xtype,xp,yp,graph,scale,centre)

          do 790 chan = 1, 2
              do 780 pts = stpt, enpt

                  call plotpt(page,xtype,xax(pts),
     +                        (20*ALOG10(pidat(grno,chan,pts))),
     +                        centre,scale)
780           continue
              xtrid = xax(enpt)
              if (chan.eq.1) then
                  call drwlet(page,xtrid,
     +                        (20*ALOG10(pidat(grno,chan,enpt))),
     +                        centre,scale,'1')
              else
```

```
                call drwlet(page,xtrid,
     +                      (20*ALOG10(pidat(grno,chan,enpt))),
     +                      centre,scale,'2')
            endif
790       continue


c900   continue

       page = 0

       call WIPSCR(page)
       call DISP(page)
       call wrhead(page,(720-46*9)/2,15,46,
      +   'Performance Index Analysis: Press F3 for plots')
       call MOVE(3,20)
       call DLINE(716,20)
       call MOVE(3,128)
       call DLINE(716,128)
       call MOVE(3,236)
       call DLINE(716,236)
       call MOVE(360,20)
       call DLINE(360,236)

       do 990 boxno = 1, 4
           xpos = boxcor(boxno,1) + 2
           ypos = boxcor(boxno,2) + 12
           call LEVEL(0)
           call WRTSTR(page,xpos,ypos,inflen(boxno,1),boxinf(boxno,1))
           call LEVEL(1)
           xpos = xpos + 15
           ypos = ypos + 15
           call wrhead(page,xpos,ypos,inflen(boxno,2),boxinf(boxno,2))
           xpos = xpos + 10
           ypos = ypos + 15
           call WRTSTR(page,xpos,ypos,inflen(boxno,3),boxinf(boxno,3))
           ypos = ypos + 15
           do 980 com = 1,3
               call LEVEL(piclev(boxno,1,com))
               call WRTSTR(page,(xpos + 9 + int( (com-1)*12*9 )),
     +                     ypos,6,picom(boxno,1,com))
               call LEVEL(1)
980        continue
           ypos = ypos + 15
           xpos = xpos - 10
           call wrhead(page,xpos,ypos,inflen(boxno,4),boxinf(boxno,4))
           xpos = xpos + 10
           ypos = ypos + 15
           call WRTSTR(page,xpos,ypos,inflen(boxno,5),boxinf(boxno,5))
           ypos = ypos + 15
           do 970 com = 1,3
               call LEVEL(piclev(boxno,2,com))
               call WRTSTR(page,(xpos + 9 + int( (com-1)*12*9 )),
     +                     ypos,6,picom(boxno,2,com))
               call LEVEL(1)
970        continue
990    continue
       call LEVEL(0)
       call WRTSTR(page,5,248,19,'Stability Analysis:')
       call LEVEL(1)
       call WRTSTR(page,(5+21*9),248,19,'Process found to be')
       call LEVEL(stclev)
       call WRTSTR(page,(5+25*9+17*9),248,20,stcom(1))
       call LEVEL(1)
       call WRTSTR(page,30,263,14,'Gain Margin = ')
       call prtr4(page,(30+15*9),263,6,'(f6.2)',6,stval(1,1))
       call WRTSTR(page,386,263,15,'Phase Margin = ')
       call prtr4(page,(386+15*9),263,6,'(f6.2)',6,stval(2,1))


       key = INKEY(1)
       return
       end
C ***************************************************************************
CH     REVISION HISTORY  :
C      VERSION       BY                  DATE        COMMENT
C        1.1      A. de Waal           06/01/90   Finally Commented
C      FILEND            :
C ***************************************************************************


C
C      FILE              : PERSET.FOR
C ***************************************************************************
CN     MODULE NAME       : perset
CA     FUNCTION          : Perturb Setpoint Gaussianly or Sinusoidally
CS     CALL SEQUENCE     : call perset()
CI     INPUT PARAMETERS  : None
CO     OUTPUT PARAMETERS : None
CG     GLOBAL VARIABLES  : Include files:   time.inc,syst.inc
CM     MODULES CALLED    : Fortran  :  rndnrm
CE     ERROR CONDITIONS  : None
CC     COMMENTS          : Perturb setpoint either:
C                               - Randomly with variable generated with a
C                                 Gaussian PDF
C                               - Sinusoidally
C ***************************************************************************
       subroutine perset()
       implicit double precision (r)
$include: 'time.inc'
$include: 'syst.inc'
       integer order

       order = int(syst(4))

       if (rtyp.eq.0) then
          do 100 i = 1, order
             rn(i) = sngl(rndnrm(dble(0.0),dble(sqrt(rampl)),irand))
100        continue
       else
           do 300 i = 1, order
               if (rchset(i).eq.0)
     +            rn(i) = rchdat(1,i)*sin(rchdat(2,i)*ti)
300        continue
```

```
      endif

      return
      end
C ********************************************************************************
CH    REVISION HISTORY  :
C     VERSION       BY                DATE        COMMENT
C       1.1     A. de Waal          06/01/90   Finally Commented
C     FILEND          :
C ********************************************************************************

C
C     FILE            : PIPLOT.FOR
C ********************************************************************************
CN    MODULE NAME     : piplot
CA    FUNCTION        : View Group of Perf Index Data Plots and Analysis
CS    CALL SEQUENCE   : call piplot()
CI    INPUT PARAMETERS  : None
CO    OUTPUT PARAMETERS : None
CG    GLOBAL VARIABLES  : Include files:  syst.inc,perdat.inc,pldat.inc
CM    MODULES CALLED    : Fortran  :  wrhead, dwaxes, drawpt, drwlet
C                         Assembler:  WIPSCR, MOVE, DLINE, WRTSTR, LEVEL
CE    ERROR CONDITIONS  : None
CC    COMMENTS          : Group of all performance index data plots are
C                         plotted on graphics page 1, and an analysis of all
C                         performance indices is presented on graphics page 0.
C ********************************************************************************
      subroutine piplot()
      implicit integer*2 (D,I)
$include: 'syst.inc'
$include: 'perdat.inc'
$include: 'pldat.inc'
      integer stpt,enpt
      integer*2 i,j,pos,key,xpos,ypos,pts,com
      integer*2 page,xtype/1/,dots/1/,join/2/,nums/1/
      integer*2 graph(4)
      real xax(100), w0, wf, winc
      real*8 xp(3),yp(3)
      real*8 scale(2),centre(4)
      common /plpts/ stpt,enpt

      page = 1
      call GPAGE(page)
      call CLRSCR()
      call wrhead(page,((720-46*9)/2),22,46,
     +   'Performance Index Plots: Press F3 for analysis')
      call BOX(page,2,346,716,344)
      call BOX(page,4,344,712,340)
      call MOVE(4,30)
      call DLINE(716,30)
      call MOVE(4,175)
      call DLINE(716,175)
      call MOVE(4,320)
      call DLINE(716,320)
      call MOVE(360,30)
      call DLINE(360,344)

      call LEVEL(0)
      call WRTSTR(page,10,45,28,
     +   'Noise to Input(1), Output(2)')
      call WRTSTR(page,10,190,34,
     +   'Disturbance to Input(1), Output(2)')
      call WRTSTR(page,366,190,34,
     +   'Sensitivity of Input(1), Output(2)')
      call WRTSTR(page,366,45,36,
     +   'Control Effort(1), Tracking Error(2)')
      call WRTSTR(page,20,335,15,
     +   'Gain Margin =  ')
      call prtr4(page,(20+14*9),335,6,'(f6.2)',6,stval(1,1))
      call WRTSTR(page,376,335,16,
     +   'Phase Margin =  ')
      call prtr4(page,(376+15*9),335,6,'(f6.2)',6,stval(2,1))
      call LEVEL(1)


      w0 = syst(1)
      wf = syst(2)
      winc = syst(3)
      npts = int((wf - w0)/winc)
      w = w0

      do 901 pts = 1, npts
          xax(pts) = w
          w = w + winc
901   continue


      nums = 1
      xtype = 1

      do 900 grno = 1,4
          do 800 axinf = 1,3
c               if (axinf.eq.1) xaxdat(grno,axinf) = DBLE(wf)
c               if (axinf.eq.2) xaxdat(grno,axinf) = DBLE(w0)
c               if (axinf.eq.3) xaxdat(grno,axinf) = DBLE(w0)
              xp(axinf) = xaxdat(grno,axinf)
              yp(axinf) = yaxdat(grno,axinf)
800       continue
          do 700 posinf = 1,4
              graph(posinf) = grpos(grno,posinf)
700       continue
          xpos = graph(1)
          ypos = graph(2) - graph(4) - 2
          call WRTSTR(page,xpos,ypos,4,'[dB]')
          xpos = graph(1) + graph(3) - 54
          ypos = graph(2) - graph(4)/2 + 20
          call WRTSTR(page,xpos,ypos,7,'[rad/s]')
          call dwaxes(page,nums,xtype,xp,yp,graph,scale,centre)

          do 790 chan = 1, 2
              do 780 pts = stpt, enpt

                  call plotpt(page,xtype,xax(pts),
     +                        (20*ALOG10(pidat(grno,chan,pts))),
     +                        centre,scale)
780           continue
```

```
            xtrid = xax(enpt)
            if (chan.eq.1) then
                call drwlet(page,xtrid,
     +                      (20*ALOG10(pidat(grno,chan,enpt))),
     +                      centre,scale,'1')
            else
                call drwlet(page,xtrid,
     +                      (20*ALOG10(pidat(grno,chan,enpt))),
     +                      centre,scale,'2')
            endif
790       continue

900   continue

      page = 0

      call WIPSCR(page)
      call DISP(page)
      call wrhead(page,(720-46*9)/2,15,46,
     +   'Performance Index Analysis: Press F3 for plots')
      call MOVE(3,20)
      call DLINE(716,20)
      call MOVE(3,128)
      call DLINE(716,128)
      call MOVE(3,236)
      call DLINE(716,236)
      call MOVE(360,20)
      call DLINE(360,236)

      do 990 boxno = 1, 4
          xpos = boxcor(boxno,1) + 2
          ypos = boxcor(boxno,2) + 12
          call LEVEL(0)
          call WRTSTR(page,xpos,ypos,inflen(boxno,1),boxinf(boxno,1))
          call LEVEL(1)
          xpos = xpos + 15
          ypos = ypos + 15
          call wrhead(page,xpos,ypos,inflen(boxno,2),boxinf(boxno,2))
          xpos = xpos + 10
          ypos = ypos + 15
          call WRTSTR(page,xpos,ypos,inflen(boxno,3),boxinf(boxno,3))
          ypos = ypos + 15
          do 980 com = 1,3
              call LEVEL(piclev(boxno,1,com))
              call WRTSTR(page,(xpos + int( (com-1)*12*9 )),
     +                    ypos,6,picom(boxno,1,com))
              call LEVEL(1)
980       continue
          ypos = ypos + 15
          xpos = xpos - 10
          call wrhead(page,xpos,ypos,inflen(boxno,4),boxinf(boxno,4))
          xpos = xpos + 10
          ypos = ypos + 15
          call WRTSTR(page,xpos,ypos,inflen(boxno,5),boxinf(boxno,5))
          ypos = ypos + 15
          do 970 com = 1,3
              call LEVEL(piclev(boxno,2,com))
              call WRTSTR(page,(xpos + int( (com-1)*12*9 )),
     +                    ypos,6,picom(boxno,2,com))
              call LEVEL(1)
970       continue
990   continue
      call LEVEL(0)
      call WRTSTR(page,5,248,19,'Stability Analysis:')
      call LEVEL(1)
      call WRTSTR(page,(5+21*9),248,19,'Process found to be')
      call LEVEL(stclev)
      call WRTSTR(page,(5+25*9+16*9),248,20,stcom(1))
      call LEVEL(1)
      call WRTSTR(page,30,263,14,'Gain Margin = ')
      call prtr4(page,(30+15*9),263,6,'(f6.2)',6,stval(1,1))
      call WRTSTR(page,386,263,15,'Phase Margin = ')
      call prtr4(page,(386+15*9),263,6,'(f6.2)',6,stval(2,1))


      key = INKEY(1)
      return
      end
C ***************************************************************************
CH    REVISION HISTORY  :
C     VERSION       BY                    DATE         COMMENT
C       1.1     A. de Waal            06/01/90    Finally Commented
C     FILEND            :
C ***************************************************************************

C
C     FILE              : PLDAT.FOR
C ***************************************************************************
CN    MODULE NAME       : block data pldat
CA    FUNCTION          : Block Data containing Plotting Data
CS    CALL SEQUENCE     : not a subroutine
CI    INPUT PARAMETERS  : n/a
CO    OUTPUT PARAMETERS : n/a
CG    GLOBAL VARIABLES  : Include files:  pldat.inc
C                         Common blocks:  /plpts/ stpt,enpt
C                                         /view/ vop
CM    MODULES CALLED    : n/a
CE    ERROR CONDITIONS  : n/a
CC    COMMENTS          : This block data section defines graph positions,
C                         default plotting data values and graph comments
C ***************************************************************************
      block data pldat
$include: 'pldat.inc'
      integer stpt,enpt
      integer vop
      common /plpts/ stpt,enpt
      common /view/ vop
      data stpt/1/, enpt/100/, vop/0/
      data ((grpos(j,i),i=1,4),j=1,4)/10,172,340,110,
     +                                10,317,340,110,
     +                                365,317,340,110,
     +                                365,172,340,110/
      data((xaxdat(j,i),i=1,3),j=1,4)/100.0,0.0,0.0,
```

```
     +                                100.0,0.0,0.0,
     +                                100.0,0.0,0.0,
     +                                100.0,0.0,0.0/
      data((yaxdat(j,i),i=1,3),j=1,4)/10.0,-50.0,0.0,
     +                                10.0,-50.0,0.0,
     +                                10.0,-50.0,0.0,
     +                                10.0,-50.0,0.0/
      data((boxcor(j,i),i=1,2),j=1,4)/4,20,
     +                                4,128,
     +                                360,128,
     +                                360,20/
      data((inflen(j,i),i=1,5),j=1,4)/24,36,35,37,35,
     +                                31,36,35,37,35,
     +                                39,22,35,23,35,
     +                                36,14,35,17,35/
      data((boxinf(j,i),i=1,5),j=1,4)
     +  /'Sensor Noise Attenuation                ',
     +   'Attenuation of Transmission to Input    ',
     +   'Low Frequ   Int. Frequ  High Frequ      ',
     +   'Attenuation of Transmission to Output   ',
     +   'Low Frequ   Int. Frequ  High Frequ      ',
     +   'Process Disturbance Attenuation         ',
     +   'Attenuation of Transmission to Input    ',
     +   'Low Frequ   Int. Frequ  High Frequ      ',
     +   'Attenuation of Transmission to Output   ',
     +   'Low Frequ   Int. Frequ  High Frequ      ',
     +   'Sensitivity to changes in Process Model ',
     +   'Insensitivity of Input                  ',
     +   'Low Frequ   Int. Frequ  High Frequ      ',
     +   'Insensitivity of Output                 ',
     +   'Low Frequ   Int. Frequ  High Frequ      ',
     +   'Control Effort and Setpoint Tracking    ',
     +   'Control Effort                          ',
     +   'Low Frequ   Int. Frequ  High Frequ      ',
     +   'Setpoint Tracking                       ',
     +   'Low Frequ   Int. Frequ  High Frequ      '/
      end
C ********************************************************************************
CH    REVISION HISTORY  :
C     VERSION        BY                DATE        COMMENT
C       1.1      A. de Waal           06/01/90   Finally Commented
C     FILEND           :
C ********************************************************************************


C
C     FILE               : PRECAL.FOR
C ********************************************************************************
CN    MODULE NAME        : precal
CA    FUNCTION           : Calculates Precompensator for S-state Tracking
CS    CALL SEQUENCE      : call precal()
CI    INPUT PARAMETERS   : None
CO    OUTPUT PARAMETERS  : None
CG    GLOBAL VARIABLES   : Include files:  syst.inc,permat.inc,perdat.inc
CM    MODULES CALLED     : Fortran   :  jwmat,add,cmult,cinv,savmat,prti2
C                          Assembler:  ERTONE,WRTSTR,LEVEL,INKEY
CE    ERROR CONDITIONS   : Matrix dimensioning and disk errors as printed to
C                          graphics page 0
CC    COMMENTS           : This routine effectively inverts the closed loop
C                          state feedback model and uses this matrix as a
C                          precompensator matrix to ensure zero steady-state
C                          error between the setpoints and the outputs.  This
C                          constant matrix is stored in the variable ptrsst.
C ********************************************************************************
      subroutine precal()
      implicit integer (c,s,I)
$include: 'syst.inc'
$include: 'permat.inc'
$include: 'perdat.inc'
      integer mord,dima,merr,err
      real w
      do 115 i = 1, ip
          do 125 j = 1, ip
              ptrsst(i,j) = 0.0
              if (i.eq.j) then
                  ptrsst(i,j) = 1.0
              endif
125       continue
115   continue
      mord = int(syst(4))
      dima = int(syst(5))

      w = 0.0
      write(4,*) ' Calculating Precomp. for Zero S-State Error'
      write(4,*) ' w ='
      write(4,'(1x,g10.2,i5)')w

C Calculating (jwI - A)
      call jwmat(w,dima,jwmatr,jwmati)
      call add(jwmatr,a,jamatr,dima,dima,dima,dima,1,merr)
      call add(jwmati,zermat,jamati,dima,dima,dima,dima,1,merr)
      if (merr.ne.0) goto 101

C Calculating G = C*inv(jwI - A)*B
      call cinv(jamatr,jamati,tematr,temati,dima,ip,2*ip)
      call cmult(c,zermat,tematr,temati,tema2r,tema2i,
     +           mord,dima,dima,dima,merr)
      if (merr.ne.0) goto 101
      call cmult(tema2r,tema2i,b,zermat,gmatr,gmati,
     +           mord,dima,dima,mord,merr)
      if (merr.ne.0) goto 101

C Calculating phi = inv(jwI - A + Kf*C)
      call mult(kf,c,kfcmat,dima,mord,mord,dima,merr)
      if (merr.ne.0) goto 101
      call add(jamatr,kfcmat,tematr,dima,dima,dima,dima,0,merr)
      if (merr.ne.0) goto 101
      call cinv(tematr,jamati,fimatr,fimati,dima,ip,2*ip)

C Calculating P = inv(I + Kc*phi*B)
      call cmult(kc,zermat,fimatr,fimati,tematr,temati,
     +           mord,dima,dima,dima,merr)
      if (merr.ne.0) goto 101
      call cmult(tematr,temati,b,zermat,tema2r,tema2i,
     +           mord,dima,dima,mord,merr)
      if (merr.ne.0) goto 101
```

```
      call add(eyemat,tema2r,tematr,mord,mord,mord,mord,0,merr)
      if (merr.ne.0) goto 101
      call add(zermat,tema2i,temati,mord,mord,mord,mord,0,merr)
      if (merr.ne.0) goto 101
      call cinv(tematr,temati,pmatr,pmati,mord,ip,2*ip)

C Calculating M = P = inv(I + Kc*phi*B)*Kc*phi*Kf
C                   = P*Kc*phi*Kf
      call cmult(pmatr,pmati,kc,zermat,tematr,temati,
     +           mord,dima,dima,dima,merr)
      if (merr.ne.0) goto 101
      call cmult(tematr,temati,fimatr,fimati,tema2r,tema2i,
     +           mord,dima,dima,dima,merr)
      if (merr.ne.0) goto 101
      call cmult(tema2r,tema2i,kf,zermat,mmatr,mmati,
     +           mord,dima,dima,mord,merr)
      if (merr.ne.0) goto 101

C Calculating L2 = G*M
      call cmult(gmatr,gmati,mmatr,mmati,l2matr,l2mati,
     +           mord,mord,mord,mord,merr)
      if (merr.ne.0) goto 101

C Calculating S2 = inv(I + L2)
      call add(eyemat,l2matr,tematr,mord,mord,mord,mord,0,merr)
      if (merr.ne.0) goto 101
      call add(zermat,l2mati,temati,mord,mord,mord,mord,0,merr)
      if (merr.ne.0) goto 101
      call cinv(tematr,temati,s2matr,s2mati,mord,ip,2*ip)

C Calculating Ptr = inv(S2*G*P)   : s = 0
      call cmult(s2matr,s2mati,gmatr,gmati,tematr,temati,
     +           mord,mord,mord,mord,merr)
      if (merr.ne.0) goto 101
      call cmult(tematr,temati,pmatr,pmati,tema2r,tema2i,
     +           mord,mord,mord,mord,merr)
      if (merr.ne.0) goto 101
      call cinv(tema2r,tema2i,ptrsst,tema2i,mord,ip,2*ip)

C Saving Precompensator Matrix to File
      err = savmat(mord,mord,ptrsst,
     +     'Re(Precomp Matrix)       ',4)
      err = savmat(mord,mord,tema2i,
     +     'Im(Precomp Matrix)       ',4)
      if (err.ne.0) then
          call ERTONE()
          call LEVEL(0)
          call WRTSTR(0,50,100,59,
     +   '***ERROR: Could not write Precomp matrix to file percal.tes')
          call LEVEL(1)
          key = INKEY(1)
          call WRTSTR(0,50,100,59,
     +   '                                                          ')
      endif

100   goto 103

101   continue
      call ERTONE()
      call LEVEL(0)
      call WRTSTR(0,100,280,30,'Matrix Dimension Error Ocurred')
      call LEVEL(1)
      key = INKEY(1)
102   continue
      call ERTONE()
      call LEVEL(0)
      call WRTSTR(0,100,280,44,
     +   'No Convergence - No of Iterations Performed:')
      call prti2(page,(100+44*9),280,4,'(i5)',5,ierr)
      call LEVEL(1)
      key = INKEY(1)

103   continue

      return
      end
C ***************************************************************************
CH    REVISION HISTORY :
C     VERSION       BY              DATE        COMMENT
C       1.1     A. de Waal          06/01/90   Finally Commented
C     FILEND        :
C ***************************************************************************


C
C     FILE              : PROSTA.FOR
C ***************************************************************************
CN    MODULE NAME       : prosta
CA    FUNCTION          : Calculate Process States
CS    CALL SEQUENCE     : call prosta()
CI    INPUT PARAMETERS  : None
CO    OUTPUT PARAMETERS : None
CG    GLOBAL VARIABLES  : Include files:  syst.inc,time.inc
CM    MODULES CALLED    : None
CE    ERROR CONDITIONS  : None
CC    COMMENTS          : Subroutine uses a fourth order Runge-Kutta algorithm
C                         to perform the first order integrations necessary to
C                         calculate the new process states (xn) from the
C                         previous states and the inputs (un).  The actual
C                         outputs (ynact) are also calculated from the states
C                         and possible disturbances (zn).  The actual outputs
C                         differ from the measured outputs (yn) if noise (nn)
C                         is present on the output sensors, and these values
C                         are calculated.
C ***************************************************************************
      subroutine prosta()
$include: 'syst.inc'
$include: 'time.inc'
      integer order,dima,plnoi

      order = int(syst(4))
      dima = int(syst(5))

      if ((dinc.eq.0).and.(plnoi.eq.0)) call pltsta(order,order,ti,zn)
      if ((ninc.eq.0).and.(plnoi.eq.0)) call pltsta(order,order,ti,nn)
```

```
      do 100 i = 1, dima
          swdxdt(i) = 0.0
          xo(i) = xn(i)
          if (i.le.order) yn(i) = 0.0
100   continue
      do 200 n = 1, 4

C Work out k(n) = f( xn(ndt)+c2(n)*k(n-1)*dt, ndt+c2(n)*ndt )
C               = a*( xn(ndt) + c2(n)*k(n-1)*dt )
C               = dxdt(i), i=1, dima
          do 210 i = 1, dima
              dxdt(i) = 0.0
              do 220 j = 1, dima
                  dxdt(i) = dxdt(i) + a(i,j)*xn(j)
                  if (j.le.order) dxdt(i) = dxdt(i) + b(i,j)*un(j)
220           continue
c              if ((dinc.eq.0).and.(dtyp.eq.0)) dxdt(i) = dxdt(i) + zn(i)
210       continue

C Work out swdxdt = k(1) + 2*k(2) + 2*k(3) + k(4)
C                 = c1(n)*k(n)
C                 = c1(n)*dxdt(i), i=1, dima
C                                  c1(i)=1, c1(2)=2, c1(3)=2, c1(4)=1
          do 230 i = 1, dima
              swdxdt(i) = swdxdt(i) + c1(n)*dxdt(i)

C Work out xn(ndt)+c2(n)*k(n-1)*dt to be used in calculation of next k(n)
C      NOTE: only for n = 1, 2, 3: c2(1)=0.5, c2(2)=0.5, c2(3)=1.0
              if (n.lt.4) then
                  xn(i) = xo(i) + c2(n)*dxdt(i)*dt

C Work out xn( (n+1)*dt ) = xn(n*dt) + ( k(1) + 2*k(2) + 2*k(3) + k(4) )*dt/6
C                         = xn(n*dt) + swdxdt*dt/6
              else
                  xn(i) = xo(i) + swdxdt(i)*dt/6
              endif
230       continue
200   continue
      do 300 i = 1, order
          do 310 j = 1, dima
              yn(i) = yn(i) + c(i,j)*xn(j)
310       continue
c          if ((dinc.eq.0).and.(dtyp.eq.1)) yn(i) = yn(i) + zn(i)
          if (dinc.eq.0) yn(i) = yn(i) + zn(i)
          ynact(i) = yn(i)
          if (ninc.eq.0) yn(i) = yn(i) + nn(i)
300   continue
      return
      end
C *****************************************************************************
CH    REVISION HISTORY  :
C     VERSION       BY                  DATE        COMMENT
C       1.1     A. de Waal            06/01/90   Finally Commented
C     FILEND            :
C *****************************************************************************
C
C     FILE              : SETPER.FOR
C *****************************************************************************
CN    MODULE NAME       : setper
CA    FUNCTION          : Specify Sinus. Setp. Perturb. Options
CS    CALL SEQUENCE     : call setper()
CI    INPUT PARAMETERS  : None
CO    OUTPUT PARAMETERS : None
CG    GLOBAL VARIABLES  : Include files:  keys.inc,time.inc,syst.inc
CM    MODULES CALLED    : Fortran  :  wrhead,togopt,getset
C                         Assembler:  WRTSTR,WIPSCR
CE    ERROR CONDITIONS  : None
CC    COMMENTS          : Interrogates user about options for the inclusion
C                         of sinusoidal setpoint perturbations.  User specifies
C                         whether perturbations to be included for each
C                         of the channels.  Once these specifications have been
C                         made, the routine calls subroutine getset in which
C                         amplitude and frequency information for each channel
C                         is obtained from the user.
C *****************************************************************************
      subroutine setper()
      implicit integer*2 (I,g)
$include: 'keys.inc'
$include: 'time.inc'
$include: 'syst.inc'

      integer npos,key
      integer dima,order

      order = syst(4)
      dima = syst(5)

      call WIPSCR(0)
C Write heading. (wrhead.for - iglib.lib)
      call wrhead(0,(720-48*9)/2,25,47,
     +   'Sinusoidal Setpoint Perturbation Specification.')

      call WRTSTR(0,50,70,33,'Include Perturbations?      : No ')

      if (rinc.eq.0) call WRTSTR(0,50+30*9,70,3,'Yes')

      call WRTSTR(0,40,250,16,'ESC key to exit.')

199   continue

      call togopt(rinc,50+30*9,70,key)
      if ((rinc.eq.0).and.(key.ne.esck)) then
          rtyp = 1
          call WRTSTR(0,40,235,36,
     +                '                                        ')
          call WRTSTR(0,40,250,16,'                             ')
          call getset(rchset,rchdat,order,50,90)
          call WRTSTR(0,40,235,36,
     +        'RETURN, TAB and cursor keys to move.')
          call WRTSTR(0,40,250,16,'ESC key to exit.')
      endif

      if (key.ne.esck) goto 199

      call WIPSCR(0)
      return
```

```
      end
C ********************************************************************************
CH    REVISION HISTORY  :
C     VERSION       BY                  DATE        COMMENT
C       1.1     A. de Waal            06/01/90   Finally Commented
C     FILEND        :
C ********************************************************************************


C
C     FILE              : SININ.FOR
C ********************************************************************************
CN    MODULE NAME       : sinin
CA    FUNCTION          : Drive Sinusoid Perturbation Menu
CS    CALL SEQUENCE     : call sinin()
CI    INPUT PARAMETERS  : None
CO    OUTPUT PARAMETERS : None
CG    GLOBAL VARIABLES  : None
CM    MODULES CALLED    : Fortran  :  inmenu,wrhead,noidis,setper
C                         Assembler:  DOMENU,WRTSTR,ERROTNE
CE    ERROR CONDITIONS  : None
CC    COMMENTS          : Initializes menu screen and drives menu for the
C                         choice whether sinusoidal perturbations are to be
C                         included on setpoints or as noise/disturbances.
C ********************************************************************************
      subroutine sinin()
      implicit integer*2 (D)
      integer*2 opt1411

99    continue

      call inmenu()

C Write heading. (wrhead.for - iglib.lib)
      call wrhead(0,(720-51*9)/2,35,51,
     +   'Specification of Sinusoidal Setpoint Perturbations.')

C Get chosen option. (domenu.asm - iglib.lib)
      opt1411 = DOMENU(1411)

C Blank over heading. (util.asm - iglib.lib)
      call WRTSTR(0,(720-51*9)/2,35,51,
     +   '                                                  ')

      if (opt1411.eq.1) then
          call noidis()

      elseif (opt1411.eq.2) then
          call setper()

      elseif (opt1411.ne.0) then
          call ERTONE()

      endif

      if (opt1411.ne.0) then
          goto 99
      endif

      return
      end
C ********************************************************************************
CH    REVISION HISTORY  :
C     VERSION       BY                  DATE        COMMENT
C       1.1     A. de Waal            06/01/90   Finally Commented
C     FILEND        :
C ********************************************************************************


C     FILE              : SYNTH.FOR
C ********************************************************************************
CN    MODULE NAME       : synth
CA    FUNCTION          : Call LQG Synthesis Procedures
CS    CALL SEQUENCE     : call synth()
CI    INPUT PARAMETERS  : None
CO    OUTPUT PARAMETERS : None
CG    GLOBAL VARIABLES  : None
CM    MODULES CALLED    : Fortran  :  optkc,optkf
C                         Assembler:  WRTSTR,ERTONE,INKEY
CE    ERROR CONDITIONS  : Matrix dimensioning of eigenvalue determination error
CC    COMMENTS          : Calls subroutines optkc and optkf to synthesize
C                         optimal controller gain kc and optimal Kalman filter
C                         gain kf respectively.  Prints out to graphics page 0
C                         if any errors ocurred in synthesis procedure
C ********************************************************************************
      subroutine synth()
      integer dimerr,conver

      call optkc(dimerr,conver)
      call optkf(dimerr,conver)

      if ((dimerr.ne.0).or.(conver.ne.0)) goto 101


100   continue
      goto 102

101   continue
      call ERTONE()
      if (dimerr.ne.0) then
          call WRTSTR(0,50,100,37,
     +          '****ERROR - Matrix Dimensioning Error')
      endif
      if (conver.ne.0) then
          call WRTSTR(0,50,100,56,
     +    '****ERROR - No convergence in Determining Eigenstructure')
      endif
      key = INKEY(1)

102   continue

      return
      end
C ********************************************************************************
CH    REVISION HISTORY  :
```

```
C      VERSION        BY                  DATE         COMMENT
C        1.1      A. de Waal            06/01/90    Finally Commented
C      FILEND           :
C ***************************************************************************


C
C      FILE             : TOGDTY.FOR
C ***************************************************************************
CN     MODULE NAME      : togdty
CA     FUNCTION         : Toggle Output of State Selection
CS     CALL SEQUENCE    : call togdty(iopt,xpos,ypos,key)
CI     INPUT PARAMETERS : iopt: integer - 0 shows outputs selected at present
C                                         2 shows states selected at present
C                         xpos,ypos: integer - Co-ords on screen where editing
C                                              to take place
CO     OUTPUT PARAMETERS : iopt: indicates edited selection
C                         key : integer - returned key code
CG     GLOBAL VARIABLES  : Include files:  keys.inc
CM     MODULES CALLED    : Assembler:  WRTSTR,LEVEL,INKEY
CE     ERROR CONDITIONS  : None
CC     COMMENTS          : Enables user to toggle selection of outputs or states
C                          by editing flag iopt on graphics screen 0.
C ***************************************************************************
       subroutine togdty(iopt,xpos,ypos,key)
       implicit integer (I)
$include: 'keys.inc'
       integer iopt,xpos,ypos,key

       call WRTSTR(0,40,265,59,
      +    'SPACE to toggle yes/no. RETURN and TAB keys to move option.')

       call LEVEL(0)
       if (iopt.eq.0) call WRTSTR(0,xpos,ypos,3,'Out')
       if (iopt.eq.2) call WRTSTR(0,xpos,ypos,3,'Sta')
       call LEVEL(1)

198    continue
       key = INKEY(1)
       if (key.eq.8192) then
           if (iopt.eq.0) then
               iopt = 2
               call LEVEL(0)
               call WRTSTR(0,xpos,ypos,3,'Sta')
               call LEVEL(1)
           else
               iopt = 0
               call LEVEL(0)
               call WRTSTR(0,xpos,ypos,3,'Out')
               call LEVEL(1)
           endif

       endif
       if ((key.ne.tabk).and.(key.ne.retk).and.(key.ne.rtabk).and.
      +    (key.ne.esck).and.(key.ne.upk).and.(key.ne.downk)) goto 198
       if (iopt.eq.0) call WRTSTR(0,xpos,ypos,3,'Out')
       if (iopt.eq.2) call WRTSTR(0,xpos,ypos,3,'Sta')

       call WRTSTR(0,40,265,59,
      +    '                                                              ')

       return
       end
C ***************************************************************************
CH     REVISION HISTORY  :
C      VERSION        BY                  DATE         COMMENT
C        1.1      A. de Waal            06/01/90    Finally Commented
C      FILEND           :
C ***************************************************************************
C
C      FILE             : TOGOPT.FOR
C ***************************************************************************
CN     MODULE NAME      : togopt
CA     FUNCTION         : User Toggle Yes/No Option Flag
CS     CALL SEQUENCE    : call togopt(iopt,xpos,ypos,key)
CI     INPUT PARAMETERS : iopt: integer - option flag 0 = yes, 1 = no
C                         xpos,ypos: integer - position on graphics screen 0
C                                              where editing to take place
CO     OUTPUT PARAMETERS : iopt: returned selection
C                          key:  returned key code
CG     GLOBAL VARIABLES  : Include files:  keys.inc
CM     MODULES CALLED    : Assembler:  WRTSTR,LEVEL,INKEY
CE     ERROR CONDITIONS  : None
CC     COMMENTS          : Enables user to toggle yes/no flag by editing flag
C                          iopt on graphics screen 0.
C ***************************************************************************
       subroutine togopt(iopt,xpos,ypos,key)
       implicit integer (I)
$include: 'keys.inc'
       integer iopt,xpos,ypos,key

       call WRTSTR(0,40,265,59,
      +    'SPACE to toggle yes/no. RETURN and TAB keys to move option.')

       call LEVEL(0)
       if (iopt.eq.0) call WRTSTR(0,xpos,ypos,3,'Yes')
       if (iopt.eq.1) call WRTSTR(0,xpos,ypos,3,'No ')
       call LEVEL(1)

198    continue
       key = INKEY(1)
       if (key.eq.8192) then
           if (iopt.eq.0) then
               iopt = 1
               call LEVEL(0)
               call WRTSTR(0,xpos,ypos,3,'No ')
               call LEVEL(1)
           else
               iopt = 0
               call LEVEL(0)
               call WRTSTR(0,xpos,ypos,3,'Yes')
               call LEVEL(1)
           endif

       endif
       if ((key.ne.tabk).and.(key.ne.retk).and.(key.ne.rtabk).and.
```

```
     +    (key.ne.esck).and.(key.ne.upk).and.(key.ne.downk)) goto 198
      if (iopt.eq.0) call WRTSTR(0,xpos,ypos,3,'Yes')
      if (iopt.eq.1) call WRTSTR(0,xpos,ypos,3,'No ')

      call WRTSTR(0,40,265,59,
     +   '                                                              ')

      return
      end
C ********************************************************************************
CH    REVISION HISTORY  :
C     VERSION       BY                  DATE        COMMENT
C       1.1     A. de Waal            06/01/90   Finally Commented
C     FILEND          :
C ********************************************************************************

C
C     FILE              : TOGVOP.FOR
C ********************************************************************************
CN    MODULE NAME       : togvop
CA    FUNCTION          : Toggle Performance Index Plot View Option
CS    CALL SEQUENCE     : call togdty(iopt,xpos,ypos,key)
CI    INPUT PARAMETERS  : iopt: integer - indicates which view group option
C                                         presently selected
C                         xpos,ypos: integer - Co-ords on screen where editing
C                                              to take place
CO    OUTPUT PARAMETERS : iopt: indicates edited selection
C                         key : integer - returned key code
CG    GLOBAL VARIABLES  : Include files:  keys.inc
CM    MODULES CALLED    : Fortran  :  disvop
C                         Assembler:  WRTSTR,LEVEL,INKEY
CE    ERROR CONDITIONS  : None
CC    COMMENTS          : Enables user to change selection of group of
C                         performance index plots to be displayed
C                         by editing flag iopt on graphics screen 0.
C ********************************************************************************
      subroutine togvop(iopt,xpos,ypos,key)
      implicit integer (I)
$include: 'keys.inc'
      integer iopt,xpos,ypos,key

      call WRTSTR(0,40,235,59,
     +   'SPACE to change option. RETURN and TAB keys to move option.')

      call LEVEL(0)
      call disvop(iopt,xpos,ypos)
      call LEVEL(1)

198   continue
      key = INKEY(1)
      if (key.eq.8192) then
          iopt = iopt + 1
          if (iopt.eq.5) iopt = 0
          call LEVEL(0)
          call disvop(iopt,xpos,ypos)
          call LEVEL(1)
      endif
      if ((key.ne.tabk).and.(key.ne.retk).and.(key.ne.rtabk).and.
     +    (key.ne.esck)) goto 198

      call disvop(iopt,xpos,ypos)

      call WRTSTR(0,40,235,59,
     +   '                                                              ')

      return
      end
C ********************************************************************************
CH    REVISION HISTORY  :
C     VERSION       BY                  DATE        COMMENT
C       1.1     A. de Waal            06/01/90   Finally Commented
C     FILEND          :
C ********************************************************************************

-
```

OPTCAD Data Manipulation Subroutines			
Name	**File**	**Description**	**Page**
data	DATA.FOR	Drive Data Loading/Saving/Editing Menu	381
eddat	EDDAT.FOR	Drive Data Editing Menu	381
eddef	EDDEF.FOR	Edit Package Default Settings	382
edgenm	EDGENM.FOR	Edit a Real Matrix on Graphics Screen	382
edk	EDK.FOR	Drive Menu for Edits to Contr/Obser Matrices	383
edpopt	EDPOPT.FOR	Edit Options for Performance Index Plots	384
edprec	EDPREC.FOR	Edit Precompensator Gain Matrix	385
edpro	EDPRO.FOR	Drive Menu for Editing of Process Description	385
edqr	EDQR.FOR	Drive Menu for Editing of Wt/Cov Matrices	386
edqr0	EDQR0.FOR	Drive Menu for Editing of Covariance Matrices	387
edqr1	EDQR1.FOR	Drive Menu for Editing of Weighting Matrices	387
edsim	EDSIM.FOR	Unused	388
edsys	EDSYS.FOR	Edit S-space System Constants	388
loddat	LODDAT.FOR	Drive Program Parameter Loading Menu	389
loddea	LODDEA.FOR	Load Non-LQG Design and Perf Index Analysis Data	390
loddes	LODDES.FOR	Load Non-LQG Control System Design	392
lodk	LODK.FOR	Load Controller and Observer Matrices	393
lodlqa	LODLQA.FOR	Load LQG Design and Perf Index Analysis Data	394
lodlqg	LODLQG.FOR	Load LQG Design	396
lodmat	LODMAT.FOR	Load Matrix Data from Disk	397
lodper	LODPER.FOR	Load Performance Index Data from Disk	398
lodpi	LODPI.FOR	Control Reading of MATLAB Perf Index Data	399
lodpro	LODPRO.FOR	Load Process Data from Disk	399
lodlqa	LODQR.FOR	Load LQG Weighting and Covariance Matrices	400
matdat	MATDAT.FOR	Load Performance Index Data from MATLAB Files	401
matst	MATST.FOR	Load Stability Data from MATLAB Files	402
rdmtld	RDMTLD.FOR	Read in a One-dimensional Array of MATLAB Data	402

OPTCAD Data Manipulation Subroutines			
Name	**File**	**Description**	**Page**
savdat	SAVDAT.FOR	Drive Program Parameter Saving Menu	403
savdea	SAVDEA.FOR	Save Non-LQG Design and Perf Index Analysis Data	404
savdes	SAVDES.FOR	Save Non-LQG Control System Design	408
savk	SAVK.FOR	Save Controller and Observer Matrices	410
savlqa	SAVLQA.FOR	Save LQG Design and Perf Index Analysis Data	412
savlqg	SAVLQG.FOR	Save LQG Design	416
savmat	SAVMAT.FOR	Save Matrix Data to Disk.	418
savper	SAVPER.FOR	Save Performance Index Data to Disk	418
savpro	SAVPRO.FOR	Save Process Data from Disk	419
savlqa	SAVQR.FOR	Save LQG Weighting and Covariance Matrices	421
wrcmat	WRCMAT.FOR	Write Complex Matrix to File	423

```
C
C     FILE                : DATA.FOR
C ***************************************************************************
CN    MODULE NAME         : data
CA    FUNCTION            : Drive Data Loading/Saving/Editing Menu
CS    CALL SEQUENCE       : call data()
CI    INPUT PARAMETERS    : None
CO    OUTPUT PARAMETERS   : None
CG    GLOBAL VARIABLES    : None
CM    MODULES CALLED      : Fortran   :  inmenu,wrhead,loddat,savdat,eddat
C                           Assembler:  DOMENU,WRTSTR,ERTONE
CE    ERROR CONDITIONS    : None
CC    COMMENTS            : Initialize menu screen and drive data
C                           loading/saving/editing menu
C ***************************************************************************
      subroutine data()
      implicit integer*2 (D)
      integer*2 opt11

99    continue

      call inmenu()

C Write heading. (wrhead.for - iglib.lib)
      call wrhead(0,(720-40*9)/2,35,40,
     +   'Load, Save or Edit Data Used in Package.')

C Get chosen opt11ion. (domenu.asm - iglib.lib)
      opt11 = DOMENU(11)

C Blank over heading. (util.asm - iglib.lib)
      call WRTSTR(0,(720-40*9)/2,35,40,
     +   '                                        ')

      if (opt11.eq.1) then
          call loddat()

      elseif (opt11.eq.2) then
          call savdat()

      elseif (opt11.eq.3) then
          call eddat()

      elseif (opt11.ne.0) then
          call ERTONE()

      endif

      if (opt11.ne.0) then
          goto 99
      endif

      return
      end
C ***************************************************************************
CH    REVISION HISTORY  :
C     VERSION      BY                  DATE         COMMENT
C       1.1     A. de Waal           06/01/90   Finally Commented
C     FILEND            :
C ***************************************************************************
C
C     FILE                : EDDAT.FOR
C ***************************************************************************
CN    MODULE NAME         : eddat
CA    FUNCTION            : Drive Data Editing Menu
CS    CALL SEQUENCE       : call eddat()
CI    INPUT PARAMETERS    : None
CO    OUTPUT PARAMETERS   : None
CG    GLOBAL VARIABLES    : Include files:  runvar.inc
CM    MODULES CALLED      : Fortran   :  inmenu,wrhead,eddef,edpro,edqr,edprec
C                           Assembler:  DOMENU,WRTSTR,ERTONE
CE    ERROR CONDITIONS    : None
CC    COMMENTS            : Initializes menu screen and drives data editing menu
C                           setting relevant flags
C ***************************************************************************
      subroutine eddat()
      implicit integer*2 (D)
$include: 'runvar.inc'
      integer*2 opt113

99    continue

      call inmenu()

C Write heading. (wrhead.for - iglib.lib)
      call wrhead(0,(720-54*9)/2,35,54,
     +   'Editing Package Default Data, Process and Design Data.')

C Get chosen option. (domenu.asm - iglib.lib)
      opt113 = DOMENU(113)

C Blank over heading. (util.asm - iglib.lib)
      call WRTSTR(0,(720-54*9)/2,35,54,
     +   '                                                      ')

      if (opt113.eq.1) then
          call eddef()
          deffl = .true.

      elseif (opt113.eq.2) then
          call edpro()
          profl = .true.

      elseif (opt113.eq.3) then
          call edqr()
          qrfl = .true.

      elseif (opt113.eq.4) then
          call edk()
          kfl = .true.

      elseif (opt113.eq.5) then
          call edprec()

      elseif (opt113.ne.0) then
          call ERTONE()
```

```
      endif

      if (optl13.ne.0) then
         goto 99
      endif

      return
      end
C ***************************************************************************
CH    REVISION HISTORY  :
C     VERSION      BY                  DATE        COMMENT
C       1.1     A. de Waal           06/01/90   Finally Commented
C     FILEND           :
C ***************************************************************************

C
C     FILE             : EDDEF.FOR
C ***************************************************************************
CN    MODULE NAME      : eddef
CA    FUNCTION         : Edit Package Default Settings
CS    CALL SEQUENCE    : call eddef()
CI    INPUT PARAMETERS : None
CO    OUTPUT PARAMETERS: None
CG    GLOBAL VARIABLES : Include files:  defnam.inc,keys.inc
CM    MODULES CALLED   : Fortran  :  wrhead,togopt
C                        Assembler:  WIPSCR,WRTSTR,STRIN
CE    ERROR CONDITIONS : None
CC    COMMENTS         : Edits package default settings on graphics screen 0
C                        and allows saving of settings to file optcad.sys
C ***************************************************************************
      subroutine eddef()
      implicit integer (S)
$include: 'defnam.inc'
$include: 'keys.inc'
      integer key,yesno

      yesno = 1

      call WIPSCR(0)
C Write heading. (wrhead.for - iglib.lib)
      call wrhead(0,(720-33*9)/2,35,33,
     +    'Editing Default Package Settings.')

      call WRTSTR(0,50,85,10,'Save Path:')
      call WRTSTR(0,50,105,10,'Load Path:')
      call WRTSTR(0,50,125,37,'Save New Default Settings? (Y/N): No ')
      call WRTSTR(0,175,85,25,outpth)
      call WRTSTR(0,175,105,25,inpath)

      npos = 1
199   continue

      if (npos.eq.1) then
          call WRTSTR(0,40,200,30,'Hit RETURN to accept pathname.')
          call WRTSTR(0,40,215,16,'Hit ESC to exit.')
99        continue
          key = STRIN(0,175,85,25,outpth)
          if ((key.ne.tabk).and.(key.ne.retk).and.(key.ne.rtabk).and.
     +        (key.ne.esck)) goto 99
          call WRTSTR(0,40,200,30,'                                   ')
          call WRTSTR(0,40,215,16,'                 ')
      elseif (npos.eq.2) then
          call WRTSTR(0,40,200,30,'Hit RETURN to accept pathname.')
          call WRTSTR(0,40,215,16,'Hit ESC to exit.')
97        continue
          key = STRIN(0,175,105,25,inpath)
          if ((key.ne.tabk).and.(key.ne.retk).and.(key.ne.rtabk).and.
     +        (key.ne.esck)) goto 97
          call WRTSTR(0,40,200,30,'                                   ')
          call WRTSTR(0,40,215,16,'                 ')
      elseif (npos.eq.3) then
          call togopt(yesno,50+34*9,125,key)
          if (yesno.eq.0) then
              open(3,file='optcad.sys',status='unknown')
              write(3,'(a)')outpth
              write(3,'(a)')inpath
              close(3)
          endif
      endif

      if ((key.eq.downk).or.(key.eq.retk).or.(key.eq.tabk)) then
         if (npos.eq.3) then
            npos = 1
         else
            npos = npos + 1
         endif
      elseif ((key.eq.upk).or.(key.eq.rtabk)) then
         if (npos.eq.1) then
            npos = 3
         else
            npos = npos - 1
         endif
      endif
      if (key.ne.esck) goto 199

      call WIPSCR(0)

      return
      end
C ***************************************************************************
CH    REVISION HISTORY  :
C     VERSION      BY                  DATE        COMMENT
C       1.1     A. de Waal           06/01/90   Finally Commented
C     FILEND           :
C ***************************************************************************



C
C     FILE             : EDGENM.FOR
C ***************************************************************************
CN    MODULE NAME      : edgenm
CA    FUNCTION         : Edit a Real Matrix on Graphics Screen
CS    CALL SEQUENCE    : call edgenm(edmatr,rows,cols,mname)
CI    INPUT PARAMETERS : edmatr: Matrix of maximum dimension (15,15)
```

```
C                         rows,cols:  Actual dimensions of matrix
C                         mname:  Name of matrix
CO    OUTPUT PARAMETERS : edmatr,rows,cols
CG    GLOBAL VARIABLES  : Common blocks:  /edmat/ mat
CM    MODULES CALLED    : Fortran  :  wrhead,editm
C                         Assembler:  WIPSCR,WRTSTR
CE    ERROR CONDITIONS  : None
CC    COMMENTS          : Types out matrix edmatr and then conditions matrix
C                         into format for editing using editm subroutine.
C                         Types out edited matrix after editing using editm.
C ************************************************************************************
      subroutine edgenm(edmatr,rows,cols,mname)
      integer rows,cols
      integer*2 n
      real edmatr(15,15)
      real*4 mat(15,15,2,13)
      character*25 mname
      common /edmat/ mat

      do 900 i = 1, 15
          do 800 j = 1, 15
              do 700 k = 1, 2
                  do 600 l = 1, 13
                      mat(i,j,k,l) = 0.0
600               continue
700           continue
800       continue
900   continue

      call WIPSCR(0)
      call wrhead(0,25,30,7,'Matrix:')
      call WRTSTR(0,30,50,25,mname)
      do 100 i = 1, rows
          do 200 j = 1, cols
              mat(i,j,i,1) = edmatr(i,j)
              call prtr4(0,(20+80*j),(80+i*15),6,'(e8.2)',
     +                   8,edmatr(i,j))
200       continue
100   continue

      if (rows.gt.cols) then
          n = rows
      else
          n = cols
      endif

      call WRTSTR(0,350,250,28,'Press Any Key to Edit Matrix')
      key = INKEY(1)

C Call matrix editing routine in file edtmat.for
      call editm(n,mname)

      call GPAGE(0)
      call WIPSCR(0)
      call DISP(0)
      call wrhead(0,25,30,7,'Matrix:')
      call WRTSTR(0,30,50,25,mname)
      do 150 i = 1, 15
          do 250 j = 1, 15
              edmatr(i,j) = mat(i,j,1,1)
              if ((i.le.rows).and.(j.le.cols))
     +            call prtr4(0,(20+80*j),(80+i*15),6,'(e8.2)',
     +                       8,edmatr(i,j))
250       continue
150   continue

      call WRTSTR(0,350,250,25,'Press Any Key to Continue')

      key = INKEY(1)

C Blank over heading. (util.asm - iglib.lib)
      call WRTSTR(0,(720-63*9)/2,35,63,
     +'                                                               ')

      return
      end
C ************************************************************************************
CH    REVISION HISTORY  :
C     VERSION       BY                  DATE        COMMENT
C       1.1     A. de Waal            06/01/90   Finally Commented
C     FILEND            :
C ************************************************************************************
C
C     FILE              : EDK.FOR
C ************************************************************************************
CN    MODULE NAME       : edk
CA    FUNCTION          : Drive Menu for Edits to Contr/Obser Matrices
CS    CALL SEQUENCE     : call edk()
CI    INPUT PARAMETERS  : None
CO    OUTPUT PARAMETERS : None
CG    GLOBAL VARIABLES  : Inclued files:  syst.inc
CM    MODULES CALLED    : Fortran  :  inmenu,wrhead,edgenm
C                         Assembler:  DOMENU,WRTSTR,ERTONE
CE    ERROR CONDITIONS  : None
CC    COMMENTS          : Initializes menu screen and drives menu for the
C                         editing of controller (kc) and observer (kf) gain
C                         matrices.
C ************************************************************************************
      subroutine edk()
      implicit integer*2 (D)
$include: 'syst.inc'
      integer dima,mord
      integer*2 opt1134
      character*25 mname
99    continue

      call inmenu()

      call WIPSCR(0)
C Write heading. (wrhead.for - iglib.lib)
      call wrhead(0,(720-63*9)/2,35,63,
     +'Editing Contr./Observ. Matrices Instead of Using LQG Synthesis.')


C Get chosen option. (domenu.asm - iglib.lib)
      opt1134 = DOMENU(1134)
```

```
            endif
            if (w0.gt.wf) goto 194
            syst(1) = w0
         elseif (npos.eq.3) then
192         continue
            key = geti2(0,50+27*9,170,4,'(i5)',5,maxpts)
            if (maxpts.lt.10) then
                call WRTSTR(0,150,220,43,
     +              '*** Error : Must Create More than 10 Points')
                call ERTONE()
            elseif (maxpts.gt.100) then
                call WRTSTR(0,150,220,44,
     +              '*** Error : Must Create Less than 100 Points')
                call ERTONE()
            else
                call WRTSTR(0,150,220,44,
     +              '                                                ')
            endif
            if ((maxpts.lt.10).or.(maxpts.gt.100)) goto 192
            winc = (wf - w0)/maxpts
            syst(3) = winc
         endif


         if ((key.eq.downk).or.(key.eq.retk).or.(key.eq.tabk)) then
            if (npos.eq.3) then
               npos = 1
            else
               npos = npos + 1
            endif
         elseif ((key.eq.upk).or.(key.eq.rtabk)) then
            if (npos.eq.1) then
               npos = 3
            else
               npos = npos - 1
            endif
         endif
         if (key.ne.esck) goto 199


         do 100 i = 1, 4
             xaxdat(i,1) = wf
             xaxdat(i,2) = w0
             xaxdat(i,3) = w0
100      continue

         stpt = 1
         enpt = maxpts

         call WIPSCR(0)
         return
         end
C ***************************************************************************
CH       REVISION HISTORY  :
C        VERSION       BY                  DATE         COMMENT
C          1.1      A. de Waal            06/01/90    Finally Commented
C        FILEND            :
C ***************************************************************************
C
C        FILE              : EDPREC.FOR
C ***************************************************************************
CN       MODULE NAME       : edprec
CA       FUNCTION          : Edit Precompensator Gain Matrix
CS       CALL SEQUENCE     : call edprec()
CI       INPUT PARAMETERS  : None
CO       OUTPUT PARAMETERS : None
CG       GLOBAL VARIABLES  : Include files:  syst.inc
CM       MODULES CALLED    : Fortran   :  wrhead,edgenm
C                            Assembler:  WIPSCR
CE       ERROR CONDITIONS  : None
CC       COMMENTS          : Edits precompensator matrix ptrsst on
C                            graphics screen 0.
C ***************************************************************************
         subroutine edprec()
$include: 'syst.inc'
         integer dima,mord
         character*25 mname

         call WIPSCR(0)
C Write heading. (wrhead.for - iglib.lib)
         call wrhead(0,(720-48*9)/2,35,48,
     +    'Editing Steady-State Precompensator Gain Matrix.')

         dima = int(syst(5))
         mord = int(syst(4))

         mname = 'Precomp. Gain Matrix     '
         call edgenm(ptrsst,mord,mord,mname)

         return
         end
C ***************************************************************************
CH       REVISION HISTORY  :
C        VERSION       BY                  DATE         COMMENT
C          1.1      A. de Waal            06/01/90    Finally Commented
C        FILEND            :
C ***************************************************************************



C
C        FILE              : EDPRO.FOR
C ***************************************************************************
CN       MODULE NAME       : edpro
CA       FUNCTION          : Drive Menu for Editing of Process Description
CS       CALL SEQUENCE     : call edpro()
CI       INPUT PARAMETERS  : None
CO       OUTPUT PARAMETERS : None
CG       GLOBAL VARIABLES  : Include files:  syst.inc
CM       MODULES CALLED    : Fortran   :  inmenu,wrhead
C                            Assembler:  DOMENU,WRTSTR,ERTONE
CE       ERROR CONDITIONS  : None
CC       COMMENTS          : Initialize menu screen and drive menu for the editing
```

```
C                          of the State-space description of the process.
C ***************************************************************************
      subroutine edpro()
      implicit integer*2 (D)
$include: 'syst.inc'
      integer dima,mord
      integer*2 opt1132
      character*25 mname
99    continue

      call inmenu()

      call WIPSCR(0)
C Write heading. (wrhead.for - iglib.lib)
      call wrhead(0,(720-41*9)/2,35,41,
     +   'Editing State Space Process Description.')


C Get chosen option. (domenu.asm - iglib.lib)
      opt1132 = DOMENU(1132)

C Blank over heading. (util.asm - iglib.lib)
      call WRTSTR(0,(720-41*9)/2,35,41,
     +   '                                         ')

      dima = int(syst(5))
      mord = int(syst(4))

      if (opt1132.eq.1) then
          call edsys()

      elseif (opt1132.eq.2) then
          mname = 'Process S-Space A-Matrix '
          call edgenm(a,dima,dima,mname)

      elseif (opt1132.eq.3) then
          mname = 'Process S-Space B-Matrix '
          call edgenm(b,dima,mord,mname)

      elseif (opt1132.eq.4) then
          mname = 'Process S-Space C-Matrix '
          call edgenm(c,mord,dima,mname)

      elseif (opt1132.ne.0) then
          call ERTONE()

      endif

      if (opt1132.ne.0) then
          goto 99
      endif

      return
      end
C ***************************************************************************
CH    REVISION HISTORY  :
C     VERSION       BY                  DATE        COMMENT
C       1.1     A. de Waal            06/01/90   Finally Commented
C     FILEND            :
C ***************************************************************************



C
C     FILE              : EDQR.FOR
C ***************************************************************************
CN    MODULE NAME       : edqr
CA    FUNCTION          : Drive Menu for Editing of Wt/Cov Matrices
CS    CALL SEQUENCE     : call edqr()
CI    INPUT PARAMETERS  : None
CO    OUTPUT PARAMETERS : None
CG    GLOBAL VARIABLES  : None
CM    MODULES CALLED    : Fortran  :  inmenu,wrhead,edqr1,edqr0
C                         Assembler:  DOMENU,WRTSTR,ERTONE
CE    ERROR CONDITIONS  : None
CC    COMMENTS          : Initializes menu screen and drives menu for decision
C                         for editing either weighting matrices
C                         or covariance matrices
C ***************************************************************************
      subroutine edqr()
      implicit integer*2 (D)
      integer*2 n
      integer*2 opt1133

99    continue

      call inmenu()
C Write heading. (wrhead.for - iglib.lib)
      call wrhead(0,(720-48*9)/2,35,48,
     +   'Editing Control Weighting and Noise Covariances.')


C Get chosen option. (domenu.asm - iglib.lib)
      opt1133 = DOMENU(1133)

C Blank over heading. (util.asm - iglib.lib)
      call WRTSTR(0,(720-48*9)/2,35,48,
     +   '                                            ')

      if (opt1133.eq.1) then
          call edqr1()

      elseif (opt1133.eq.2) then
          call edqr0()

      elseif (opt1133.ne.0) then
          call ERTONE()

      endif

      if (opt1133.ne.0) then
          goto 99
      endif

      return
```

```
      end
C *****************************************************************************
CH    REVISION HISTORY  :
C     VERSION       BY                DATE        COMMENT
C       1.1     A. de Waal           06/01/90   Finally Commented
C     FILEND          :
C *****************************************************************************



C
C     FILE              : EDQR0.FOR
C *****************************************************************************
CN    MODULE NAME       : edqr0
CA    FUNCTION          : Drive Menu for Editing of Covariance Matrices
CS    CALL SEQUENCE     : call edqr0()
CI    INPUT PARAMETERS  : None
CO    OUTPUT PARAMETERS : None
CG    GLOBAL VARIABLES  : Include files:  syst.inc
CM    MODULES CALLED    : Fortran  :  inmenu,wrhead,edgenm
C                         Assembler:  DOMENU,WRTSTR,ERTONE
CE    ERROR CONDITIONS  : None
CC    COMMENTS          : Initialize menu screen and drive menu for the editing
C                         of the observer covariance matrices q and r
C *****************************************************************************
      subroutine edqr0()
      implicit integer*2 (D)
$include: 'syst.inc'
      integer dima,mord
      integer*2 opt11332
      character*25 mname
99    continue

      call inmenu()

      call WIPSCR(0)
C Write heading. (wrhead.for - iglib.lib)
      call wrhead(0,(720-49*9)/2,35,49,
     +   'Editing Observer Noise Covariance Matrices Q & R.')


C Get chosen option. (domenu.asm - iglib.lib)
      opt11332 = DOMENU(11332)

C Blank over heading. (util.asm - iglib.lib)
      call WRTSTR(0,(720-49*9)/2,35,49,
     +'                                                 ')

      dima = int(syst(5))
      mord = int(syst(4))

      if (opt11332.eq.1) then
          mname = 'Observer Covar.  Matrix Q'
          call edgenm(q,dima,dima,mname)

      elseif (opt11332.eq.2) then
          mname = 'Observer Covar.  Matrix R'
          call edgenm(r,mord,mord,mname)

      elseif (opt11332.ne.0) then
          call ERTONE()

      endif

      if (opt11332.ne.0) then
          goto 99
      endif

      return
      end
C *****************************************************************************
CH    REVISION HISTORY  :
C     VERSION       BY                DATE        COMMENT
C       1.1     A. de Waal           06/01/90   Finally Commented
C     FILEND          :
C *****************************************************************************



C
C     FILE              : EDQR1.FOR
C *****************************************************************************
CN    MODULE NAME       : edqr1
CA    FUNCTION          : Drive Menu for Editing of Weighting Matrices
CS    CALL SEQUENCE     : call edqr1()
CI    INPUT PARAMETERS  : None
CO    OUTPUT PARAMETERS : None
CG    GLOBAL VARIABLES  : Include files:  syst.inc
CM    MODULES CALLED    : Fortran  :  inmenu,wrhead,edgenm
C                         Assembler:  DOMENU,WRTSTR,ERTONE
CE    ERROR CONDITIONS  : None
CC    COMMENTS          : Initialize menu screen and drive menu for the editing
C                         of the control weithting matrices q1 and r1
C *****************************************************************************
      subroutine edqr1()
      implicit integer*2 (D)
$include: 'syst.inc'
      integer dima,mord
      integer*2 opt11331
      character*25 mname
99    continue

      call inmenu()

      call WIPSCR(0)
C Write heading. (wrhead.for - iglib.lib)
      call wrhead(0,(720-49*9)/2,35,49,
     +   'Editing Input and State Weighting Matrices Q & R.')


C Get chosen option. (domenu.asm - iglib.lib)
```

```
      opt11331 = DOMENU(11331)

C Blank over heading. (util.asm - iglib.lib)
      call WRTSTR(0,(720-49*9)/2,35,49,
     +'                                                  ')

      dima = int(syst(5))
      mord = int(syst(4))

      if (opt11331.eq.1) then
          mname = 'Input Weighting Matrix R1'
          call edgenm(r1,mord,mord,mname)

      elseif (opt11331.eq.2) then
          mname = 'State Weighting Matrix Q1'
          call edgenm(q1,dima,dima,mname)

      elseif (opt11331.ne.0) then
          call ERTONE()

      endif

      if (opt11331.ne.0) then
          goto 99
      endif

      return
      end
C ******************************************************************************
CH    REVISION HISTORY  :
C     VERSION       BY                  DATE        COMMENT
C       1.I     A. de Waal            06/01/90   Finally Commented
C     FILEND          :
C ******************************************************************************




C
C     FILE              : EDSIM.FOR
C ******************************************************************************
CN    MODULE NAME       : edsim
CA    FUNCTION          : Unused
CS    CALL SEQUENCE     : call edsim()
CI    INPUT PARAMETERS  : None
CO    OUTPUT PARAMETERS : None
CG    GLOBAL VARIABLES  : None
CM    MODULES CALLED    : Fortran  :  inmenu,wrhead
C                         Assembler:  DOMENU,WRTSTR,ERTONE,INKEY
CE    ERROR CONDITIONS  : None
CC    COMMENTS          : Unused
C ******************************************************************************
      subroutine edsim()
      call WIPSCR(0)
C Write heading. (wrhead.for - iglib.lib)
      call wrhead(0,(720-44*9)/2,35,44,
     +   'Editing Data for Time Simulation of Process.')

      call excuse()

      key = INKEY(1)

C Blank over heading. (util.asm - iglib.lib)
      call WRTSTR(0,(720-44*9)/2,35,44,
     +   '                                            ')

      return
      end
C ******************************************************************************
CH    REVISION HISTORY  :
C     VERSION       BY                  DATE        COMMENT
C     FILEND          :
C ******************************************************************************


C
C     FILE              : EDSYS.FOR
C ******************************************************************************
CN    MODULE NAME       : edsys
CA    FUNCTION          : Edit S-space System Constants
CS    CALL SEQUENCE     : call edsys()
CI    INPUT PARAMETERS  : None
CO    OUTPUT PARAMETERS : None
CG    GLOBAL VARIABLES  : Include files:  keys.inc,syst.inc
CM    MODULES CALLED    : Fortran  :  wrhead,prtr4,prti2,getr4,geti2
C                         Assembler:  WRTSTR,ERTONE
CE    ERROR CONDITIONS  : None
CC    COMMENTS          : Edit constants specifying details of State-space
C                         process on graphics screen 0.
C ******************************************************************************
      subroutine edsys()
      implicit integer (g,I)
$include: 'keys.inc'
$include: 'syst.inc'

      integer npos,key,ninp,nsta

      call WIPSCR(0)
C Write heading. (wrhead.for - iglib.lib)
      call wrhead(0,(720-41*9)/2,35,41,
     +   'Editing State Space System Constants.')

      call wrhead(0,230,65,39,
     +   'Process Analysis Frequencies [rad/sec].')
      call WRTSTR(0,250,90,29,'Analysis Starting Frequency :')
      call WRTSTR(0,250,110,29,'Analysis End Frequency      :')
      call WRTSTR(0,250,130,29,'Analysis Frequency Increment:')
      call wrhead(0,230,155,33,'Dimensioning of Process Matrices.')
      call WRTSTR(0,250,180,29,'Number of Inputs to System  :')
      call WRTSTR(0,250,200,29,'Order of State-Space System :')

      npos = 1
199   continue
```

```
c      call WRTSTR(0,40,235,57,
c     +'                                                    ')
c      call WRTSTR(0,40,250,57,
c     +'                                                    ')
      call WRTSTR(0,40,235,36,'RETURN, TAB and cursor keys to move.')
      call WRTSTR(0,40,250,16,'ESC key to exit.')

      call prtr4(0,(250+30*9),90,7,'(g10.4)',10,syst(1))
      call prtr4(0,(250+30*9),110,7,'(g10.4)',10,syst(2))
      call prtr4(0,(250+30*9),130,7,'(g10.4)',10,syst(3))
      call prti2(0,(250+30*9),180,4,'(i3)',3,int(syst(4)))
      call prti2(0,(250+30*9),200,4,'(i3)',3,int(syst(5)))

      if (npos.eq.1) then
197       continue
          key = getr4(0,(250+30*9),90,7,'(g10.4)',10,syst(1))
          if (syst(1).lt.0.0) then
              call WRTSTR(0,300,220,31,
     +         '*** Error : Start Freq. < 0.0  ')
              call ERTONE()
          else
              call WRTSTR(0,300,220,31,
     +                                        ')
          endif
          if (syst(1).lt.0.0) goto 197
      elseif (npos.eq.2) then
195       continue
          key = getr4(0,(250+30*9),110,7,'(g10.4)',10,syst(2))
          if (syst(2).lt.syst(1)) then
              call WRTSTR(0,300,220,35,
     +             '*** Error : End Freq. < Start Freq.')
              call ERTONE()
          else
              call WRTSTR(0,300,220,35,
     +                                              ')
          endif
          if (syst(2).lt.syst(1)) goto 195
      elseif (npos.eq.3) then
193       continue
          key = getr4(0,(250+30*9),130,7,'(g10.4)',10,syst(3))
          if ( syst(3).gt.((syst(2)-syst(1))/10.0) ) then
              call WRTSTR(0,300,220,49,
     +            '*** Error : Freq. Incr > (End Freq-Start Freq)/10')
              call ERTONE()
          else
              call WRTSTR(0,300,220,49,
     +                                                    ')
          endif
          if ( syst(3).gt.((syst(2)-syst(1))/10.0) ) goto 193
      elseif (npos.eq.4) then
191       continue
          ninp = int(syst(4))
          key = geti2(0,(250+30*9),180,4,'(i3)',3,ninp)
          syst(4) = real(ninp)
          if (syst(4).lt.1.0) then
              call WRTSTR(0,300,220,28,
     +            '*** Error : No of inputs < 1')
              call ERTONE()
          else
              call WRTSTR(0,300,220,28,
     +                                        ')
          endif
          if (syst(4).lt.1.0) goto 191
      elseif (npos.eq.5) then
189       continue
          nsta = int(syst(5))
          key = geti2(0,(250+30*9),200,4,'(i3)',3,nsta)
          syst(5) = real(nsta)
          if (syst(5).lt.syst(4)) then
              call WRTSTR(0,300,220,39,
     +            '*** Error : No of States < No of inputs')
              call ERTONE()
          else
              call WRTSTR(0,300,220,39,
     +                                                  ')
          endif
          if (syst(5).lt.syst(4)) goto 189
      endif

      if ((key.eq.downk).or.(key.eq.retk).or.(key.eq.tabk)) then
         if (npos.eq.5) then
            npos = 1
         else
            npos = npos + 1
         endif
      elseif ((key.eq.upk).or.(key.eq.rtabk)) then
         if (npos.eq.1) then
            npos = 5
         else
            npos = npos - 1
         endif
      endif
      if (key.ne.esck) goto 199

C Blank over heading. (util.asm - iglib.lib)
      call WRTSTR(0,(720-41*9)/2,35,41,
     +'                                        ')
      return
      end
C ***************************************************************************
CH    REVISION HISTORY  :
C     VERSION       BY                 DATE        COMMENT
C       1.1     A. de Waal           06/01/90   Finally Commented
C     FILEND        :
C ***************************************************************************


C
C     FILE              : LODDAT.FOR
C ***************************************************************************
CN    MODULE NAME       : loddat
CA    FUNCTION          : Drive Program Parameter Loading Menu
CS    CALL SEQUENCE     : call loddat()
CI    INPUT PARAMETERS  : None
CO    OUTPUT PARAMETERS : None
CG    GLOBAL VARIABLES  : Include files:  runvar.inc
```

```
CM     MODULES CALLED     : Fortran  :  inmenu,wrhead,lodpro,lodqr,lodlqg
C                                       loddes,lodlqa,loddea,lodk,lodpi
C                           Assembler:  DOMENU,WRTSTR,ERTONE
CE     ERROR CONDITIONS   : None
CC     COMMENTS           : Initializes menu screen and drives menu for the
C                           loading of program parameters.  Sets appropriate
C                           flags when selections are made.
C ***************************************************************************
       subroutine loddat
       implicit integer*2 (D)
$include: 'runvar.inc'
       integer*2 opt111

99     continue

       call inmenu()

C Write heading. (wrhead.for - iglib.lib)
       call wrhead(0,(720-54*9)/2,35,54,
      +   'Loading Package Default Data, Process and Design Data.')

C Get chosen option. (domenu.asm - iglib.lib)
       opt111 = DOMENU(111)

C Blank over heading. (util.asm - iglib.lib)
       call WRTSTR(0,(720-54*9)/2,35,54,
      +   '                                                    ')

       if (opt111.eq.1) then
           call lodpro()
           profl = .true.
           lqgfl = .false.
           desfl = .false.
           lqafl = .false.
           deafl = .false.

       elseif (opt111.eq.2) then
           call lodqr()
           qrfl = .true.
           lqgfl = .false.
           desfl = .false.
           lqafl = .false.
           deafl = .false.

       elseif (opt111.eq.3) then
           call lodlqg()
           lqgfl = .true.
           qrfl = .false.
           kfl = .false.
           lqafl = .false.
           deafl = .false.

       elseif (opt111.eq.4) then
           call loddes()
           desfl = .true.
           lqgfl = .false.
           lqafl = .false.
           deafl = .false.
           kfl = .true.

       elseif (opt111.eq.5) then
           call lodlqa()
           lqafl = .true.
           desfl = .false.
           deafl = .false.
           profl = .false.
           qrfl = .false.
           lqgfl = .true.
           kfl = .false.

       elseif (opt111.eq.6) then
           call loddea()
           deafl = .true.
           lqgfl = .false.
           lqafl = .false.
           profl = .false.
           desfl = .false.
           kfl = .false.

       elseif (opt111.eq.7) then
           call lodk()
           kfl = .true.
           lqgfl = .false.
           desfl = .false.
           lqafl = .false.
           deafl = .false.

       elseif (opt111.eq.8) then
           call lodpi()
           deafl = .true.
           lqgfl = .false.
           lqafl = .false.

       elseif (opt111.ne.0) then
           call ERTONE()

       endif

       if (opt111.ne.0) then
           goto 99
       endif
       return
       end
C ***************************************************************************
CH     REVISION HISTORY  :
C      VERSION        BY                 DATE         COMMENT
C        1.1      A. de Waal           06/01/90    Finally Commented
C      FILEND           :
C ***************************************************************************


C
C      FILE             : LODDEA.FOR
C ***************************************************************************
CN     MODULE NAME      : loddea
CA     FUNCTION         : Load Non-LQG Design and Perf Index Analysis Data
```

```
CS     CALL SEQUENCE     : call loddea()
CI     INPUT PARAMETERS  : None
CO     OUTPUT PARAMETERS : None
CG     GLOBAL VARIABLES  : Include files:  keys.inc,syst.inc,defnam.inc,
C                                          perdat.inc
CM     MODULES CALLED    : Fortran   :  wrhead,maknam,dodir,wrtitl,lodmat,lodper,
C                                       prderr
C                          Assembler:  WIPSCR,WRTSTR,STRIN,RSTERR,ERTONE,INKEY,
C                                       GETERR
CE     ERROR CONDITIONS  : Disk Errors as Indicated on Screen
CC     COMMENTS          : User is interrogated to determine from which file
C                          data is to be loaded, and the following data
C                          for a non-LQG control system is loaded:
C                              - Process Data (Model and System Specifications)
C                              - Non-LQG Controller (Controller (kc) and
C                                                    Observer (kf) Matrices)
C                              - Non-LQG system performance Index Analysis Data
C ******************************************************************************
      subroutine loddea()
      implicit integer*2 (c,G,l,S)
$include: 'keys.inc'
$include: 'syst.inc'
$include: 'defnam.inc'
$include: 'perdat.inc'
      integer*2 dima,mord,npts,nchan,grno,nanal,nchan,rows,cols
      integer*2 funit
      integer*2 key,ferr,derr,lerr,len,chkio
      integer stpt,enpt
      character*1 yesno
      common /plpts/ stpt,enpt

      call WIPSCR(0)
cC Write heading. (wrhead.for - iglib.lib)
c      call wrhead(0,(720-59*9)/2,35,59,
c     +   'Loading Perf. Index Analysis Data for Design not using LQG.')

C Information that would have been passed to savef subroutine
      infnam = ';Analysed LQG Design     '
      fname = 'des0.dea                  '
      funit = 4

C Information that would have been in defnam.inc file
c     inpath = 'c:\dewaal\progs\optcad\ '
      defdir = '*.dea'

C Information that would have been in syst.inc file

C Make a filename usenam(50) = inpath(25)+fname(25)
      call maknam(fname,inpath,usenam)

C Print out default directory page 0, yupperlim 180, length 4, defdir .dea,
C                          inpath 'c:\dewaal\progs\optcad   '
      call dodir(0,180,5,defdir,inpath)

      call wrtitl(0,40,60,30,'Loading Information from File.')
      call wrtitl(0,40,85,19,'Loading from file :')
      call WRTSTR(0,40,110,30,'Hit RETURN to accept filename.')
      call WRTSTR(0,40,125,16,'Hit ESC to exit.')
99    continue
      key = STRIN(0,220,85,50,usenam)
      if ((key.ne.retk).and.(key.ne.esck)) goto 99
      call WRTSTR(0,40,110,30,'                                   ')
      call WRTSTR(0,40,125,16,'                                   ')
      if (key.eq.retk) then
         call RSTERR()
         open(funit,FILE=usenam,STATUS='OLD',IOSTAT=ferr)
         if (ferr.eq.0) then
            call WRTSTR(0,40,110,24,'Loading Information ....')

                  read(funit,'(a)',iostat=chkio,err=203)
                  read(funit,'(5g10.4)',iostat=chkio,err=203)
     +                 (syst(i),i=1,5)
                  dima = int(syst(5))
                  mord = int(syst(4))
                  stpt = 1
                  enpt = int( (syst(2)-syst(1))/syst(3) )
                  read(funit,'(a)',iostat=chkio,err=203)

                  call lodmat(rows,cols,a,mname,funit,lerr)
                  if (lerr.ne.0) goto 203
                  call lodmat(rows,cols,b,mname,funit,lerr)
                  if (lerr.ne.0) goto 203
                  call lodmat(rows,cols,c,mname,funit,lerr)
                  if (lerr.ne.0) goto 203
                  call lodmat(rows,cols,kci,mname,funit,lerr)
                  if (lerr.ne.0) goto 203
                  call lodmat(rows,cols,kfi,mname,funit,lerr)
                  if (lerr.ne.0) goto 203
                  call lodmat(rows,cols,kc,mname,funit,lerr)
                  if (lerr.ne.0) goto 203
                  call lodmat(rows,cols,kf,mname,funit,lerr)
                  if (lerr.ne.0) goto 203
                  call lodmat(rows,cols,ptrsst,mname,funit,lerr)
                  if (lerr.ne.0) goto 203

                  npts = 100
                  nanal = 5
                  nchan = 2

                  grno = 1
                  call lodper(grno,nanal,nchan,pival,pernam,funit,lerr)
                  if (lerr.ne.0) goto 203
                  grno = 2
                  call lodper(grno,nanal,nchan,pival,pernam,funit,lerr)
                  if (lerr.ne.0) goto 203
                  grno = 3
                  call lodper(grno,nanal,nchan,pival,pernam,funit,lerr)
                  if (lerr.ne.0) goto 203
                  grno = 4
                  call lodper(grno,nanal,nchan,pival,pernam,funit,lerr)
                  if (lerr.ne.0) goto 203

                  call lodmat(2,5,stval,mname,funit,lerr)
                  if (lerr.ne.0) goto 203

                  read(funit,'(a)',iostat=chkio,err=203)
```

```
                do 111 i=1,4
                    read(funit,'(a)',iostat=chkio,err=203)
                    do 112 j=1,2
                        read(funit,'(a)',iostat=chkio,err=203)
                        do 113 k = 1, 5
                            read(funit,'(a)',iostat=chkio,err=203)
     +                          picom(i,j,k)
113                     continue
112                 continue
111             continue
                read(funit,'(a)',iostat=chkio,err=203)

                read(funit,'(a)',iostat=chkio,err=203)
                read(funit,'(a)',iostat=chkio,err=203)
     +              (stcom(i),i=1,2)
                read(funit,'(a)',iostat=chkio,err=203)

                grno = 1
                call lodper(grno,npts,nchan,pidat,pernam,funit,lerr)
                if (lerr.ne.0) goto 203
                grno = 2
                call lodper(grno,npts,nchan,pidat,pernam,funit,lerr)
                if (lerr.ne.0) goto 203
                grno = 3
                call lodper(grno,npts,nchan,pidat,pernam,funit,lerr)
                if (lerr.ne.0) goto 203
                grno = 4
                call lodper(grno,npts,nchan,pidat,pernam,funit,lerr)
                if (lerr.ne.0) goto 203

            if (chkio.eq.0) goto 201
203         call ERTONE()
            lerr = chkio
201         continue

            close(funit)
            if (lerr.eq.0) then
               call WRTSTR(0,215,110,20,'Data load completed.')
            else
               call WRTSTR(0,40,130,36,
     +         '*** Error : Data load not completed.')
               key = INKEY(1)
            endif
         else
            derr = GETERR()
            if (derr.eq.-1) then
               close(funit)
               call ERTONE()
               call WRTSTR(0,40,110,34,
     +              '*** Error : Cannot load from file.')
            else
               call prderr(0,40,110,derr)
            endif
         endif
       else
          call WRTSTR(0,40,140,32,'Data load operation cancelled.')
       endif


cC Blank over heading. (util.asm - iglib.lib)
c       call WRTSTR(0,(720-59*9)/2,35,59,
c      +   '                                                          ')

       return
       end
C ***************************************************************************
CH     REVISION HISTORY  :
C      VERSION       BY                DATE          COMMENT
C        1.1      A. de Waal          06/01/90    Finally Commented
C      FILEND            :
C ***************************************************************************



C
C      FILE              : LODDES.FOR
C ***************************************************************************
CN     MODULE NAME       : loddes
CA     FUNCTION          : Load Non-LQG Control System Design
CS     CALL SEQUENCE     : call loddes()
CI     INPUT PARAMETERS  : None
CO     OUTPUT PARAMETERS : None
CG     GLOBAL VARIABLES  : Include files:  keys.inc,syst.inc,defnam.inc,
C                                          perdat.inc
CM     MODULES CALLED    : Fortran   :  wrhead,maknam,dodir,wrtitl,lodmat,
C                                       prderr
C                          Assembler:  WIPSCR,WRTSTR,STRIN,RSTERR,ERTONE,INKEY,
C                                       GETERR,
CE     ERROR CONDITIONS  : Disk Errors as Indicated on Screen
CC     COMMENTS          : User is interrogated to determine from which file
C                          data is to be loaded, and the following data
C                          for a non-LQG control system is loaded:
C                              - Process Data (Model and System Specifications)
C                              - Non-LQG Controller (Controller (kc) and
C                                                    Observer (kf) Matrices)
C ***************************************************************************
       subroutine loddes()
       implicit integer*2 (c,G,l,S)
$include: 'keys.inc'
$include: 'syst.inc'
$include: 'defnam.inc'
$include: 'perdat.inc'
       integer*2 dima,mord,npts,nchan,grno,nanal,nchan,rows,cols
       integer*2 funit
       integer*2 key,ferr,derr,lerr,len,chkio
       character*1 yesno

       call WIPSCR(0)
cC Write heading. (wrhead.for - iglib.lib)
c       call wrhead(0,(720-60*9)/2,35,60,
c      +'Loading Load System with Contr./Observ. Designed Not Using LQG.')

C Information that would have been passed to savef subroutine
       infnam = ';Analysed LQG Design      '
       fname = 'des0.des                   '
       funit = 4
```

```
C Information that would have been in defnam.inc file
c      inpath = 'c:\dewaal\progs\optcad\ '
       defdir = '*.des'

C Information that would have been in syst.inc file

C Make a filename usenam(50) = inpath(25)+fname(25)
       call maknam(fname,inpath,usenam)

C Print out default directory page 0, yupperlim 180, length 4, defdir .des,
C                              inpath 'c:\dewaal\progs\optcad  '
       call dodir(0,180,5,defdir,inpath)

       call wrtitl(0,40,60,30,'Loading Information from File.')
       call wrtitl(0,40,85,19,'Loading from file :')
       call WRTSTR(0,40,110,30,'Hit RETURN to accept filename.')
       call WRTSTR(0,40,125,16,'Hit ESC to exit.')
99     continue
       key = STRIN(0,220,85,50,usenam)
       if ((key.ne.retk).and.(key.ne.esck)) goto 99
       call WRTSTR(0,40,110,30,'                                   ')
       call WRTSTR(0,40,125,16,'                  ')
       if (key.eq.retk) then
          call RSTERR()
          open(funit,FILE=usenam,STATUS='OLD',IOSTAT=ferr)
          if (ferr.eq.0) then
             call WRTSTR(0,40,110,24,'Loading Information ....')

                   read(funit,'(a)',iostat=chkio,err=203)
                   read(funit,'(5g10.4)',iostat=chkio,err=203)
     +                  (syst(i),i=1,5)
                   dima = int(syst(5))
                   mord = int(syst(4))
                   read(funit,'(a)',iostat=chkio,err=203)

                   call lodmat(rows,cols,a,mname,funit,lerr)
                   if (lerr.ne.0) goto 203
                   call lodmat(rows,cols,b,mname,funit,lerr)
                   if (lerr.ne.0) goto 203
                   call lodmat(rows,cols,c,mname,funit,lerr)
                   if (lerr.ne.0) goto 203
                   call lodmat(rows,cols,kci,mname,funit,lerr)
                   if (lerr.ne.0) goto 203
                   call lodmat(rows,cols,kfi,mname,funit,lerr)
                   if (lerr.ne.0) goto 203
                   call lodmat(rows,cols,kc,mname,funit,lerr)
                   if (lerr.ne.0) goto 203
                   call lodmat(rows,cols,kf,mname,funit,lerr)
                   if (lerr.ne.0) goto 203
                   call lodmat(rows,cols,ptrsst,mname,funit,lerr)
                   if (lerr.ne.0) goto 203


             if (chkio.eq.0) goto 201
203          call ERTONE()
             lerr = chkio
201          continue

             close(funit)
             if (lerr.eq.0) then
                call WRTSTR(0,215,110,20,'Data load completed.')
             else
                call WRTSTR(0,40,130,36,
     +          '*** Error : Data load not completed.')
                key = INKEY(1)
             endif
          else
             derr = GETERR()
             if (derr.eq.-1) then
                close(funit)
                call ERTONE()
                call WRTSTR(0,40,110,34,
     +               '*** Error : Cannot load from file.')
             else
                call prderr(0,40,110,derr)
             endif
          endif
       else
          call WRTSTR(0,40,140,30,'Data load operation cancelled.')
       endif


cC Blank over heading. (util.asm - iqlib.lib)
c       call WRTSTR(0,(720-60*9)/2,35,60,
c      +'                                                          ')

       return
       end
C ******************************************************************************
CH     REVISION HISTORY  :
C      VERSION        BY                  DATE         COMMENT
C        1.1      A. de Waal            06/01/90    Finally Commented
C      FILEND           :
C ******************************************************************************


C
C      FILE              : LODK.FOR
C ******************************************************************************
CN     MODULE NAME       : lodk
CA     FUNCTION          : Load Controller and Observer Matrices
CS     CALL SEQUENCE     : call lodk()
CI     INPUT PARAMETERS  : None
CO     OUTPUT PARAMETERS : None
CG     GLOBAL VARIABLES  : Include files:  keys.inc,syst.inc,defnam.inc,
C                                          perdat.inc
CM     MODULES CALLED    : Fortran   :  wrhead,maknam,dodir,wrtitl,lodmat,
C                                       prderr
C                          Assembler:  WIPSCR,WRTSTR,STRIN,RSTERR,ERTONE,INKEY,
C                                       GETERR
CE     ERROR CONDITIONS  : Disk Errors as Indicated on Screen
CC     COMMENTS          : User is interrogated to determine from which file
C                          data is to be loaded, and the following data
C                          is loaded:
C                              - Controller (Controller (kc) and
```

```
C                                                  Observer (kf) Matrices)
C ********************************************************************************
      subroutine lodk()
      implicit integer*2 (c,G,l,S)
$include: 'keys.inc'
$include: 'syst.inc'
$include: 'defnam.inc'
$include: 'perdat.inc'
      integer*2 dima,mord,npts,nchan,grno,nanal,nchan,rows,cols
      integer*2 funit
      integer*2 key,ferr,derr,lerr,len,chkio
      character*1 yesno

      call WIPSCR(0)
cC Write heading. (wrhead.for - iglib.lib)
c      call wrhead(0,(720-53*9)/2,35,53,
c     +   'Loading Controller and Kalman Observer Gain Matrices.')

C Information that would have been passed to savef subroutine
      infnam = ';Analysed LQG Design      '
      fname = 'des0.k                     '
      funit = 4

C Information that would have been in defnam.inc file
c     inpath = 'c:\dewaal\progs\optcad\ '
      defdir = '*.k  '

C Information that would have been in syst.inc file

C Make a filename usenam(50) = inpath(25)+fname(25)
      call maknam(fname,inpath,usenam)

C Print out default directory page 0, yupperlim 180, length 4, defdir .k,
C                          inpath 'c:\dewaal\progs\optcad  '
      call dodir(0,180,5,defdir,inpath)

      call wrtitl(0,40,60,30,'Loading Information from File.')
      call wrtitl(0,40,85,19,'Loading from file :')
      call WRTSTR(0,40,110,30,'Hit RETURN to accept filename.')
      call WRTSTR(0,40,125,16,'Hit ESC to exit.')
99    continue
      key = STRIN(0,220,85,50,usenam)
      if ((key.ne.retk).and.(key.ne.esck)) goto 99
      call WRTSTR(0,40,110,30,'                                        ')
      call WRTSTR(0,40,125,16,'                    ')
      if (key.eq.retk) then
         call RSTERR()
         open(funit,FILE=usenam,STATUS='OLD',IOSTAT=ferr)
         if (ferr.eq.0) then
            call WRTSTR(0,40,110,24,'Loading Information ....')

                  call lodmat(rows,cols,kc,mname,funit,lerr)
                  if (lerr.ne.0) goto 203
                  call lodmat(rows,cols,kf,mname,funit,lerr)
                  if (lerr.ne.0) goto 203
                  call lodmat(rows,cols,ptrsst,mname,funit,lerr)
                  if (lerr.ne.0) goto 203

            if (chkio.eq.0) goto 201
203         call ERTONE()
            lerr = chkio
201         continue

            close(funit)
            if (lerr.eq.0) then
               call WRTSTR(0,215,110,20,'Data load completed.')
            else
               call WRTSTR(0,40,130,36,
     +         '*** Error : Data load not completed.')
               key = INKEY(1)
            endif
         else
            derr = GETERR()
            if (derr.eq.-1) then
               close(funit)
               call ERTONE()
               call WRTSTR(0,40,110,34,
     +               '*** Error : Cannot load from file.')
            else
               call prderr(0,40,110,derr)
            endif
         endif
      else
         call WRTSTR(0,40,140,30,'Data load operation cancelled.')
      endif


cC Blank over heading. (util.asm - iglib.lib)
c      call WRTSTR(0,(720-53*9)/2,35,53,
c     +   '                                                     ')

      return
      end
C ********************************************************************************
CH    REVISION HISTORY  :
C     VERSION      BY                 DATE        COMMENT
C       1.1     A. de Waal           06/01/90   Finally Commented
C     FILEND            :
C ********************************************************************************



C
C     FILE              : LODLQA.FOR
C ********************************************************************************
CN    MODULE NAME       : lodlqa
CA    FUNCTION          : Load LQG Design and Perf Index Analysis Data
CS    CALL SEQUENCE     : call lodlqa()
CI    INPUT PARAMETERS  : None
CO    OUTPUT PARAMETERS : None
CG    GLOBAL VARIABLES  : Include files:  keys.inc,syst.inc,defnam.inc,
C                                         perdat.inc
CM    MODULES CALLED    : Fortran  :  wrhead,maknam,dodir,wrtitl,lodmat,lodper,
C                                     prderr
```

```
C                          Assembler:  WIPSCR,WRTSTR,STRIN,RSTERR,ERTONE,INKEY,
C                                      GETERR
CE    ERROR CONDITIONS   : Disk Errors as Indicated on Screen
CC    COMMENTS           : User is interrogated to determine from which file
C                          data is to be loaded, and the following data
C                          for a LQG control system is loaded:
C                              - Process Data (Model and System Specifications)
C                              - LQG Weighting and Covariance Matrices
C                              - LQG Controller (Controller (kc) and
C                                                Observer (kf) Matrices)
C                              - LQG system performance Index Analysis Data
C ********************************************************************************
      subroutine lodlqa()
      implicit integer*2 (c,G,l,S)
$include: 'keys.inc'
$include: 'syst.inc'
$include: 'defnam.inc'
$include: 'perdat.inc'
      integer*2 dima,mord,npts,nchan,grno,nanal,nchan,rows,cols
      integer*2 funit
      integer*2 key,ferr,derr,lerr,len,chkio
      integer stpt,enpt
      character*1 yesno
      common /plpts/ stpt,enpt

      call WIPSCR(0)

cC Write heading. (wrhead.for - iglib.lib)
c      call wrhead(0,(720-63*9)/2,35,63,
c     +'Loading Load Perf. Index Analysis Data for Previous LQG Design.')

C Information that would have been passed to savef subroutine
      infnam = ';Analysed LQG Design      '
      fname = 'des0.lqa                   '
      funit = 4

C Information that would have been in defnam.inc file
c     inpath = 'c:\dewaal\progs\optcad\ '
      defdir = '*.lqa'

C Information that would have been in syst.inc file

C Make a filename usenam(50) = inpath(25)+fname(25)
      call maknam(fname,inpath,usenam)

C Print out default directory page 0, yupperlim 180, length 4, defdir .lqa,
C                          inpath 'c:\dewaal\progs\optcad  '
      call dodir(0,180,5,defdir,inpath)

      call wrtitl(0,40,60,30,'Loading Information from File.')
      call wrtitl(0,40,85,19,'Loading from file :')
      call WRTSTR(0,40,110,30,'Hit RETURN to accept filename.')
      call WRTSTR(0,40,125,16,'Hit ESC to exit.')
99    continue
      key = STRIN(0,220,85,50,usenam)
      if ((key.ne.retk).and.(key.ne.esck)) goto 99
      call WRTSTR(0,40,110,30,'                                   ')
      call WRTSTR(0,40,125,16,'                    ')
      if (key.eq.retk) then
         call RSTERR()
         open(funit,FILE=usenam,STATUS='OLD',IOSTAT=ferr)
         if (ferr.eq.0) then
            call WRTSTR(0,40,110,24,'Loading Information ....')

                  read(funit,'(a)',iostat=chkio,err=203)
                  read(funit,'(5g10.4)',iostat=chkio,err=203)
     +                 (syst(i),i=1,5)
                  dima = int(syst(5))
                  mord = int(syst(4))
                  stpt = 1
                  enpt = int( (syst(2)-syst(1))/syst(3) )
                  read(funit,'(a)',iostat=chkio,err=203)

                  call lodmat(rows,cols,a,mname,funit,lerr)
                  if (lerr.ne.0) goto 203
                  call lodmat(rows,cols,b,mname,funit,lerr)
                  if (lerr.ne.0) goto 203
                  call lodmat(rows,cols,c,mname,funit,lerr)
                  if (lerr.ne.0) goto 203
                  call lodmat(rows,cols,kci,mname,funit,lerr)
                  if (lerr.ne.0) goto 203
                  call lodmat(rows,cols,kfi,mname,funit,lerr)
                  if (lerr.ne.0) goto 203
                  call lodmat(rows,cols,ql,mname,funit,lerr)
                  if (lerr.ne.0) goto 203
                  call lodmat(rows,cols,rl,mname,funit,lerr)
                  if (lerr.ne.0) goto 203
                  call lodmat(rows,cols,kc,mname,funit,lerr)
                  if (lerr.ne.0) goto 203
                  call lodmat(rows,cols,q,mname,funit,lerr)
                  if (lerr.ne.0) goto 203
                  call lodmat(rows,cols,r,mname,funit,lerr)
                  if (lerr.ne.0) goto 203
                  call lodmat(rows,cols,kf,mname,funit,lerr)
                  if (lerr.ne.0) goto 203
                  call lodmat(rows,cols,ptrsst,mname,funit,lerr)
                  if (lerr.ne.0) goto 203

                  npts = 100
                  nanal = 5
                  nchan = 2

                  grno = 1
                  call lodper(grno,nanal,nchan,pival,pernam,funit,lerr)
                  if (lerr.ne.0) goto 203
                  grno = 2
                  call lodper(grno,nanal,nchan,pival,pernam,funit,lerr)
                  if (lerr.ne.0) goto 203
                  grno = 3
                  call lodper(grno,nanal,nchan,pival,pernam,funit,lerr)
                  if (lerr.ne.0) goto 203
                  grno = 4
                  call lodper(grno,nanal,nchan,pival,pernam,funit,lerr)
                  if (lerr.ne.0) goto 203

                  call lodmat(2,5,stval,mname,funit,lerr)
```

```
                if (lerr.ne.0) goto 203

                read(funit,'(a)',iostat=chkio,err=203)
                do 111 i=1,4
                    read(funit,'(a)',iostat=chkio,err=203)
                    do 112 j=1,2
                        read(funit,'(a)',iostat=chkio,err=203)
                        do 113 k = 1, 5
                            read(funit,'(a)',iostat=chkio,err=203)
     +                          picom(i,j,k)
113                     continue
112                 continue
111             continue
                read(funit,'(a)',iostat=chkio,err=203)

                read(funit,'(a)',iostat=chkio,err=203)
                read(funit,'(a)',iostat=chkio,err=203)
     +              (stcom(i),i=1,2)
                read(funit,'(a)',iostat=chkio,err=203)

                grno = 1
                call lodper(grno,npts,nchan,pidat,pernam,funit,lerr)
                if (lerr.ne.0) goto 203
                grno = 2
                call lodper(grno,npts,nchan,pidat,pernam,funit,lerr)
                if (lerr.ne.0) goto 203
                grno = 3
                call lodper(grno,npts,nchan,pidat,pernam,funit,lerr)
                if (lerr.ne.0) goto 203
                grno = 4
                call lodper(grno,npts,nchan,pidat,pernam,funit,lerr)
                if (lerr.ne.0) goto 203

            if (chkio.eq.0) goto 201
203         call ERTONE()
            lerr = chkio
201         continue

            close(funit)
            if (lerr.eq.0) then
               call WRTSTR(0,215,110,22,'Matrix load completed.')
            else
               call WRTSTR(0,40,130,38,
     +         '*** Error : Matrix load not completed.')
               key = INKEY(1)
            endif
         else
            derr = GETERR()
            if (derr.eq.-1) then
               close(funit)
               call ERTONE()
               call WRTSTR(0,40,110,34,
     +             '*** Error : Cannot load from file.')
            else
               call prderr(0,40,110,derr)
            endif
         endif
      else
         call WRTSTR(0,40,140,32,'Matrix load operation cancelled.')
      endif

cC Blank over heading. (util.asm - iglib.lib)
c      call WRTSTR(0,(720-63*9)/2,35,63,
c     +'                                                               ')

      return
      end
C ****************************************************************************
CH    REVISION HISTORY  :
C     VERSION       BY                DATE          COMMENT
C       1.1     A. de Waal            06/01/90    Finally Commented
C     FILEND            :
C ****************************************************************************
C
C     FILE              : LODLQG.FOR
C ****************************************************************************
CN    MODULE NAME       : lodlqg
CA    FUNCTION          : Load LQG Design
CS    CALL SEQUENCE     : call lodlqg()
CI    INPUT PARAMETERS  : None
CO    OUTPUT PARAMETERS : None
CG    GLOBAL VARIABLES  : Include files:  keys.inc,syst.inc,defnam.inc,
C                                         perdat.inc
CM    MODULES CALLED    : Fortran   :  wrhead,maknam,dodir,wrtitl,lodmat,
C                                      prderr
C                         Assembler:  WIPSCR,WRTSTR,STRIN,RSTERR,ERTONE,INKEY,
C                                      GETERR
CE    ERROR CONDITIONS  : Disk Errors as Indicated on Screen
CC    COMMENTS          : User is interrogated to determine from which file
C                         data is to be loaded, and the following data
C                         for a LQG control system is loaded:
C                             - Process Data (Model and System Specifications)
C                             - LQG Weighting and Covariance Matrices
C                             - LQG Controller (Controller (kc) and
C                                               Observer (kf) Matrices)
C ****************************************************************************
      subroutine lodlqg()
      implicit integer*2 (c,G,l,S)
$include: 'keys.inc'
$include: 'syst.inc'
$include: 'defnam.inc'
$include: 'perdat.inc'
      integer*2 dima,mord,npts,nchan,grno,nanal,nchan,rows,cols
      integer*2 funit
      integer*2 key,ferr,derr,lerr,len,chkio
      character*1 yesno

      call WIPSCR(0)
cC Write heading. (wrhead.for - iglib.lib)
c      call wrhead(0,(720-54*9)/2,35,54,
c     +   'Loading System with LQG Controller Design Information.')


C Information that would have been passed to savef subroutine
      infnam = ';LQG Design (Unanalysed) '
```

```
      fname = 'des0.lqg                '
      funit = 4

C Information that would have been in defnam.inc file
c     inpath = 'c:\dewaal\progs\optcad\ '
      defdir = '*.lqg'

C Information that would have been in syst.inc file

C Make a filename usenam(50) = inpath(25)+fname(25)
      call maknam(fname,inpath,usenam)


C Print out default directory page 0, yupperlim 180, length 4, defdir .lqa,
C                          inpath 'c:\dewaal\progs\optcad   '
      call dodir(0,180,5,defdir,inpath)


      call wrtitl(0,40,60,25,'Loading Matrix from File.')
      call wrtitl(0,40,85,19,'Loading from file :')
      call WRTSTR(0,40,110,30,'Hit RETURN to accept filename.')
      call WRTSTR(0,40,125,16,'Hit ESC to exit.')
99    continue
      key = STRIN(0,220,85,50,usenam)
      if ((key.ne.retk).and.(key.ne.esck)) goto 99
      call WRTSTR(0,40,110,30,'                                ')
      call WRTSTR(0,40,125,16,'                    ')
      if (key.eq.retk) then
         call RSTERR()
         open(funit,FILE=usenam,STATUS='OLD',IOSTAT=ferr)
         if (ferr.eq.0) then
            call WRTSTR(0,40,110,19,'Loading Matrix ....')

                  read(funit,'(a)',iostat=chkio,err=203)
                  read(funit,'(5g10.4)',iostat=chkio,err=203)
     +                 (syst(i),i=1,5)
                  dima = int(syst(5))
                  mord = int(syst(4))
                  read(funit,'(a)',iostat=chkio,err=203)


                  call lodmat(rows,cols,a,mname,funit,lerr)
                  if (lerr.ne.0) goto 203
                  call lodmat(rows,cols,b,mname,funit,lerr)
                  if (lerr.ne.0) goto 203
                  call lodmat(rows,cols,c,mname,funit,lerr)
                  if (lerr.ne.0) goto 203
                  call lodmat(rows,cols,kci,mname,funit,lerr)
                  if (lerr.ne.0) goto 203
                  call lodmat(rows,cols,kfi,mname,funit,lerr)
                  if (lerr.ne.0) goto 203
                  call lodmat(rows,cols,ql,mname,funit,lerr)
                  if (lerr.ne.0) goto 203
                  call lodmat(rows,cols,rl,mname,funit,lerr)
                  if (lerr.ne.0) goto 203
                  call lodmat(rows,cols,kc,mname,funit,lerr)
                  if (lerr.ne.0) goto 203
                  call lodmat(rows,cols,q,mname,funit,lerr)
                  if (lerr.ne.0) goto 203
                  call lodmat(rows,cols,r,mname,funit,lerr)
                  if (lerr.ne.0) goto 203
                  call lodmat(rows,cols,kf,mname,funit,lerr)
                  if (lerr.ne.0) goto 203
                  call lodmat(rows,cols,ptrsst,mname,funit,lerr)
                  if (lerr.ne.0) goto 203


            if (chkio.eq.0) goto 201
203         call ERTONE()
            lerr = chkio
201         continue

            close(funit)
            if (lerr.eq.0) then
               call WRTSTR(0,215,110,22,'Matrix load completed.')
            else
               call WRTSTR(0,40,130,38,
     +         '*** Error : Matrix load not completed.')
               key = INKEY(1)
            endif
         else
            derr = GETERR()
            if (derr.eq.-1) then
               close(funit)
               call ERTONE()
               call WRTSTR(0,40,110,34,
     +              '*** Error : Cannot load from file.')
            else
               call prderr(0,40,110,derr)
            endif
         endif
      else
         call WRTSTR(0,40,140,32,'Matrix load operation cancelled.')
      endif


cC Blank over heading. (util.asm -- iglib.lib)
c      call WRTSTR(0,(720-54*9)/2,35,54,
c     +   '                                                  ')

      return
      end
C ***************************************************************************
CH    REVISION HISTORY  :
C     VERSION       BY                DATE        COMMENT
C       1.1     A. de Waal          06/01/90   Finally Commented
C     FILEND          :
C ***************************************************************************
C
C         FILE NAME          : LODMAT.FOR
C ***************************************************************************
CN    MODULE NAME       : lodmat
CA    FUNCTION          : Load Matrix Data from Disk
CS    CALL SEQUENCE     : error = lodmat(n,mat,mname,funit)
CI    INPUT PARAMETERS :            rows - (integer*2) Number of Rows.
```

```
C                                cols - (integer*2) Number of Cols.
C                             mat(15,15) - (real*4) The matrix to be loaded.
C                                  mname - (character*25) The matrix title.
C                                  funit - (integer*2) The drive unit number.
CO    OUTPUT PARAMETERS: error - (integer*2) The error return code.
CG    GLOBAL VARIABLES : None.
CM    MODULES CALLED   : ERTONE (asm).
CE    ERROR CONDITIONS : ?
C
CC    COMMENTS         : Loads the matrix  information from the unit number
C                        passed from calling routine. If any errors occurs,
C                        then an error tone is sounded and the code passed
C                        back to the calling routine.
C ***************************************************************************
      subroutine lodmat(rows,cols,mat,mname,funit,lerr)
      integer*2 rows,cols,lerr
      real mat(15,15)
      character*25 mname
      integer*2 funit, chkio
      character*1 dummy

      read(funit,'(a)',iostat=chkio,err=300) mname
      read(funit,'(2i6)',iostat=chkio,err=300) rows,cols

      do 250 i = 1,rows
         do 251 j = 1,cols
              mat(i,j) = 0.0
251      continue
250   continue

      if (chkio.eq.0) then
         do 897 i = 1,rows
              read(funit,'(15g10.4)',iostat=chkio,err=300)
     +             (mat(i,j), j=1,cols)
897      continue
      endif
      read(funit,'(a)',iostat=chkio,err=300)

      if (chkio.eq.0) goto 301
300   call ERTONE()
301   continue
      lerr = chkio
      return
      end
C ***************************************************************************
CH    REVISION HISTORY  :
C     VERSION       BY                  DATE        COMMENT
C       1.1     A. de Waal            06/01/90   Finally Commented
C     FILEND          :
C ***************************************************************************



C
C     FILE              : LODPER.FOR
C ***************************************************************************
CN    MODULE NAME       : lodper
CA    FUNCTION          : Load Performance Index Data from Disk
CS    CALL SEQUENCE     : call lodper(grno,rows,cols,mat,pernam,funit,lerr)
CI    INPUT PARAMETERS  : grno:   Performance index graph number
C                         rows:   Number of rows to load
C                                 (no of frequency points)
C                         cols:   Number of columns to load
C                                 (no of regions of analysis)
C                         funit:  File unit specifier
CO    OUTPUT PARAMETERS : mat(4,2,*): Matrix containing
C                                 perfromance index plot data
C                         pernam:     Name of matrix containing
C                                     performance index plot data
C                         lerr:       Flag set if error ocurred in reading
C                                     from file
CG    GLOBAL VARIABLES  : None
CM    MODULES CALLED    : Assembler:  ERTONE
CE    ERROR CONDITIONS  : lerr set to nonzero if error ocurred in opening or
C                         reading from file
CC    COMMENTS          : Reads performance index data from file indicated by
C                         unit specifier into variable mat(4,2,*), and returns
C                         variable.  Errors in reading indicated on the screen.
C ***************************************************************************
      subroutine lodper(grno,rows,cols,mat,pernam,funit,lerr)
      integer*2 grno,rows,cols,funit,lerr,chkio
      real mat(4,2,*)
      character*25 pernam

      read(funit,'(a)',iostat=chkio,err=300) pernam
      read(funit,'(2i6)',iostat=chkio,err=300) rows,cols

      do 100 i = 1, cols
          do 200 j = 1, rows
              mat(grno,i,j) = 0.0
200       continue
100   continue

      if (chkio.eq.0) then
          do 900 i = 1,rows
              read(funit,'(2g10.4)',iostat=chkio,err=300)
     +             (mat(grno,j,i), j=1,cols)
900       continue
      endif
      read (funit,'(a)',iostat=chkio,err=300)

      if (chkio.eq.0) goto 301
300   call ERTONE()
301   continue
      lerr = chkio
      return
      end
C ***************************************************************************
CH    REVISION HISTORY  :
C     VERSION       BY                  DATE        COMMENT
C       1.1     A. de Waal            06/01/90   Finally Commented
C     FILEND          :
C ***************************************************************************
C
C     FILE              : LODPI.FOR
C ***************************************************************************
```

```
CN     MODULE NAME        : lodpi
CA     FUNCTION           : Control Reading of MATLAB Perf Index Data
CS     CALL SEQUENCE      : call lodpi()
CI     INPUT PARAMETERS   : None
CO     OUTPUT PARAMETERS  : None
CG     GLOBAL VARIABLES   : None
CM     MODULES CALLED     : Fortran  :  matdat,matst
CE     ERROR CONDITIONS   : none
CC     COMMENTS           : Control loading of performance index data from
C                           MATLAB data files
C ***************************************************************************
       subroutine lodpi()
       character*25 filnam

       filnam = 'tnupr.dat                '
       call matdat(filnam,1,1)
       filnam = 'tnypr.dat                '
       call matdat(filnam,1,2)
       filnam = 'tdupr.dat                '
       call matdat(filnam,2,1)
       filnam = 'tdypr.dat                '
       call matdat(filnam,2,2)
       filnam = 'sugpr.dat                '
       call matdat(filnam,3,1)
       filnam = 'sygpr.dat                '
       call matdat(filnam,3,2)
       filnam = 'trupr.dat                '
       call matdat(filnam,4,1)
       filnam = 'trepr.dat                '
       call matdat(filnam,4,2)

       filnam = 'gmarpr.dat               '
       call matst(filnam,1)
       filnam = 'pmarpr.dat               '
       call matst(filnam,2)

       return
       end
C ***************************************************************************
CH     REVISION HISTORY   :
C      VERSION       BY                  DATE         COMMENT
C        1.1      A. de Waal           06/01/90    Finally Commented
C      FILEND             :
C ***************************************************************************


C
C      FILE               : LODPRO.FOR
C ***************************************************************************
CN     MODULE NAME        : lodpro
CA     FUNCTION           : Load Process Data from Disk
CS     CALL SEQUENCE      : call lodpro()
CI     INPUT PARAMETERS   : None
CO     OUTPUT PARAMETERS  : None
CG     GLOBAL VARIABLES   : Include files:  keys.inc,syst.inc,defnam.inc,
C                                           perdat.inc
CM     MODULES CALLED     : Fortran  :  wrhead,maknam,dodir,wrtitl,lodmat,
C                                       prderr
C                           Assembler:  WIPSCR,WRTSTR,STRIN,RSTERR,ERTONE,INKEY,
C                                       GETERR
CE     ERROR CONDITIONS   : Disk Errors as Indicated on Screen
CC     COMMENTS           : User is interrogated to determine from which file
C                           data is to be loaded, and the following data
C                           is loaded:
C                             - Process Data (Model and System Specifications)
C ***************************************************************************
       subroutine lodpro()
       implicit integer*2 (c,G,l,S)
$include: 'keys.inc'
$include: 'syst.inc'
$include: 'defnam.inc'
$include: 'perdat.inc'
       integer*2 dima,mord,npts,nchan,grno,nanal,nchan,rows,cols
       integer*2 funit
       integer*2 key,ferr,derr,lerr,len,chkio
       character*1 yesno

c      call WIPSCR(0)
cC Write heading. (wrhead.for - iglib.lib)
c      call wrhead(0,(720-21*9)/2,35,21,
c     +   'Loading Process Data.')

C Information that would have been passed to savef subroutine
       infnam = ';Analysed LQG Design     '
       fname = 'des0.pro                  '
       funit = 4

C Information that would have been in defnam.inc file
c      inpath = 'c:\dewaal\progs\optcad\ '
       defdir = '*.pro'

C Information that would have been in syst.inc file

C Make a filename usenam(50) = inpath(25)+fname(25)
       call maknam(fname,inpath,usenam)

C Print out default directory page 0, yupperlim 180, length 4, defdir .pro,
C                           inpath 'c:\dewaal\progs\optcad  '
       call dodir(0,180,5,defdir,inpath)

       call wrtitl(0,40,60,30,'Loading Information from File.')
       call wrtitl(0,40,85,19,'Loading from file :')
       call WRTSTR(0,40,110,30,'Hit RETURN to accept filename.')
       call WRTSTR(0,40,125,16,'Hit ESC to exit.')
99     continue
       key = STRIN(0,220,85,50,usenam)
       if ((key.ne.retk).and.(key.ne.esck)) goto 99
       call WRTSTR(0,40,110,30,'                              ')
       call WRTSTR(0,40,125,16,'                 ')
       if (key.eq.retk) then
          call RSTERR()
          open(funit,FILE=usenam,STATUS='OLD',IOSTAT=ferr)
          if (ferr.eq.0) then
             call WRTSTR(0,40,110,24,'Loading Information ....')
```

```
                read(funit,'(a)',iostat=chkio,err=203)
                read(funit,'(5g10.4)',iostat=chkio,err=203)
     +              (syst(i),i=1,5)
                dima = int(syst(5))
                mord = int(syst(4))
                read(funit,'(a)',iostat=chkio,err=203)

                call lodmat(rows,cols,a,mname,funit,lerr)
                if (lerr.ne.0) goto 203
                call lodmat(rows,cols,b,mname,funit,lerr)
                if (lerr.ne.0) goto 203
                call lodmat(rows,cols,c,mname,funit,lerr)
                if (lerr.ne.0) goto 203
                call lodmat(rows,cols,kci,mname,funit,lerr)
                if (lerr.ne.0) goto 203
                call lodmat(rows,cols,kfi,mname,funit,lerr)
                if (lerr.ne.0) goto 203
                call lodmat(rows,cols,ptrsst,mname,funit,lerr)
                if (lerr.ne.0) goto 203


             if (chkio.eq.0) goto 201
203          call ERTONE()
             lerr = chkio
201          continue

             close(funit)
             if (lerr.eq.0) then
                call WRTSTR(0,215,110,20,'Data load completed.')
             else
                call WRTSTR(0,40,130,36,
     +          '*** Error : Data load not completed.')
                key = INKEY(1)
             endif
          else
             derr = GETERR()
             if (derr.eq.-1) then
                close(funit)
                call ERTONE()
                call WRTSTR(0,40,110,34,
     +              '*** Error : Cannot load from file.')
             else
                call prderr(0,40,110,derr)
             endif
          endif
       else
          call WRTSTR(0,40,140,30,'Data load operation cancelled.')
       endif


cC Blank over heading. (util.asm - iglib.lib)
c       call WRTSTR(0,(720-21*9)/2,35,21,
c      +   '                     ')

       return
       end
C ***************************************************************************
CH     REVISION HISTORY  :
C      VERSION        BY                DATE        COMMENT
C        1.1      A. de Waal          06/01/90   Finally Commented
C      FILEND           :
C ***************************************************************************
C
C      FILE               : LODQR.FOR
C ***************************************************************************
CN     MODULE NAME        : lodlqa
CA     FUNCTION           : Load LQG Weighting and Covariance Matrices
CS     CALL SEQUENCE      : call lodqr()
CI     INPUT PARAMETERS   : None
CO     OUTPUT PARAMETERS  : None
CG     GLOBAL VARIABLES   : Include files:  keys.inc,syst.inc,defnam.inc,
C                                            perdat.inc
CM     MODULES CALLED     : Fortran :   wrhead,maknam,dodir,wrtitl,lodmat,
C                                       prderr
C                            Assembler:  WIPSCR,WRTSTR,STRIN,RSTERR,ERTONE,INKEY,
C                                       GETERR
CE     ERROR CONDITIONS   : Disk Errors as Indicated on Screen
CC     COMMENTS           : User is interrogated to determine from which file
C                           data is to be loaded, and the following data
C                           for a LQG control system is loaded:
C                            - LQG Weighting and Covariance Matrices
C ***************************************************************************
       subroutine lodqr()
       implicit integer*2 (c,G,l,S)
$include: 'keys.inc'
$include: 'syst.inc'
$include: 'defnam.inc'
$include: 'perdat.inc'
       integer*2 dima,mord,npts,nchan,grno,nanal,nchan,rows,cols
       integer*2 funit
       integer*2 key,ferr,derr,lerr,len,chkio
       character*1 yesno

       call WIPSCR(0)
cC Write heading. (wrhead.for - iglib.lib)
c       call wrhead(0,(720-52*9)/2,35,52,
c      +   'Loading Control Weighting and Noise Covariance Data.')

C Information that would have been passed to savef subroutine
       infnam = ';Analysed LQG Design      '
       fname = 'des0.qr                    '
       funit = 4

C Information that would have been in defnam.inc file
c      inpath = 'c:\dewaal\progs\optcad\ '
       defdir = '*.qr'

C Information that would have been in syst.inc file

C Make a filename usenam(50) = inpath(25)+fname(25)
       call maknam(fname,inpath,usenam)

C Print out default directory page 0, yupperlim 180, length 4, defdir .qr,
C                           inpath 'c:\dewaal\progs\optcad  '
       call dodir(0,180,4,defdir,inpath)
```

```
      call wrtitl(0,40,60,30,'Loading Information from File.')
      call wrtitl(0,40,85,19,'Loading from file :')
      call WRTSTR(0,40,110,30,'Hit RETURN to accept filename.')
      call WRTSTR(0,40,125,16,'Hit ESC to exit.')
99    continue
      key = STRIN(0,220,85,50,usenam)
      if ((key.ne.retk).and.(key.ne.esck)) goto 99
      call WRTSTR(0,40,110,30,'                                   ')
      call WRTSTR(0,40,125,16,'                   ')
      if (key.eq.retk) then
         call RSTERR()
         open(funit,FILE=usenam,STATUS='OLD',IOSTAT=ferr)
         if (ferr.eq.0) then
            call WRTSTR(0,40,110,24,'Loading Information ....')

                  call lodmat(rows,cols,q1,mname,funit,lerr)
                  if (lerr.ne.0) goto 203
                  call lodmat(rows,cols,r1,mname,funit,lerr)
                  if (lerr.ne.0) goto 203
                  call lodmat(rows,cols,q,mname,funit,lerr)
                  if (lerr.ne.0) goto 203
                  call lodmat(rows,cols,r,mname,funit,lerr)
                  if (lerr.ne.0) goto 203

            if (chkio.eq.0) goto 201
203         call ERTONE()
            lerr = chkio
201         continue

            close(funit)
            if (lerr.eq.0) then
               call WRTSTR(0,215,110,20,'Data load completed.')
            else
               call WRTSTR(0,40,130,36,
     +         '*** Error : Data load not completed.')
               key = INKEY(1)
            endif
         else
            derr = GETERR()
            if (derr.eq.-1) then
               close(funit)
               call ERTONE()
               call WRTSTR(0,40,110,34,
     +              '*** Error : Cannot load from file.')
            else
               call prderr(0,40,110,derr)
            endif
         endif
      else
         call WRTSTR(0,40,140,30,'Data load operation cancelled.')
      endif

cC Blank over heading. (util.asm - iglib.lib)
c      call WRTSTR(0,(720-52*9)/2,35,52,
c     +   '                                                  ')

      return
      end
C *****************************************************************************
CH    REVISION HISTORY  :
C     VERSION       BY                  DATE          COMMENT
C       1.1     A. de Waal            06/01/90   Finally Commented
C     FILEND            :
C *****************************************************************************
C
C     FILE              : MATDAT.FOR
C *****************************************************************************
CN    NODULE NAME       : matdat
CA    FUNCTION          : Load Performance Index Data from MATLAB Files
CS    CALL SEQUENCE     : call matdat(filnam,group,chan)
CI    INPUT PARAMETERS  : filnam: character*20  - name of file to be loaded
C                                                 from
C                         group:  integer       - group in variable pidat to
C                                                 which data is to be assigned
C                         chan:   integer       - channel in group into which
C                                                 data is to be read
CO    OUTPUT PARAMETERS : none
CG    GLOBAL VARIABLES  : Include files:  perdat.inc
CM    MODULES CALLED    : Fortran  :  rdmtld
CE    ERROR CONDITIONS  : File opening or reading error
CC    COMMENTS          : Loads MATLAB performance index data from file
C                         specified by variable filnam.  Indicates reading
C                         error on graphics page 0.
C *****************************************************************************
      subroutine matdat(filnam,group,chan)
$include: 'perdat.inc'
      integer npts,erflg,group,chan
      real temp(100)
      character*25 filnam

      npts = 100

      do 99 i = 1, npts
         temp(i) = 0.0
99    continue

      call WRTSTR(0,100,100,24,'About to read from file:')
      call WRTSTR(0,(100 + 25*9),100,25,filnam)

      call rdmtld(filnam,npts,temp,erflg)

      if (erflg.eq.0) then
         call WRTSTR(0,100,150,20,'Just read from file:')
         call WRTSTR(0,(100 + 21*9),150,25,filnam)
         do 900 i = 1, npts
c           call prtr4(0,150,(20+i*12),7,'(E10.5)',10,temp(i))
            pidat(group,chan,i) = temp(i)
c           call prtr4(0,300,(20+i*12),7,'(E10.5)',10,
c     +                pidat(group,chan,i))
900      continue
      elseif (erflg.eq.1) then
         call LEVEL(0)
         call ERTONE()
         call WRTSTR(0,100,200,27,'ERROR: Could not open file:')
         call WRTSTR(0,(100+28*9),200,25,filnam)
         call LEVEL(1)
```

```
      else
          call LEVEL(0)
          call ERTONE()
          call WRTSTR(0,100,200,27,'ERROR: Could not read file:')
          call WRTSTR(0,(100+28*9),200,25,filnam)
          call LEVEL(1)
      endif

      return
      end

C ***************************************************************************
CH    REVISION HISTORY  :
C     VERSION       BY                  DATE        COMMENT
C       1.1     A. de Waal            06/01/90   Finally Commented
C     FILEND          :
C ***************************************************************************


C
C     FILE              : MATST.FOR
C ***************************************************************************
CN    MODULE NAME       : matst
CA    FUNCTION          : Load Stability Data from MATLAB Files
CS    CALL SEQUENCE     : call matst(filnam,nmar)
CI    INPUT PARAMETERS  : filnam: character*20  - name of file to be loaded
C                                                 from
C                         nmar:   integer       - stability margin
C                                                 (gain or phase)
CO    OUTPUT PARAMETERS : none
CG    GLOBAL VARIABLES  : Include files:  perdat.inc
CM    MODULES CALLED    : Fortran  :  rdmtld
CE    ERROR CONDITIONS  : File opening or reading error
CC    COMMENTS          : Loads MATLAB stability data from file
C                         specified by variable filnam.  Indicates reading
C                         error on graphics page 0.
C ***************************************************************************
      subroutine matst(filnam,nmar)
$include: 'perdat.inc'
      integer npts,erflg,group,chan
      character*25 filnam

      npts = 1

      call WRTSTR(0,100,100,24,'About to read from file:')
      call WRTSTR(0,(100 + 25*9),100,25,filnam)

      call rdmtld(filnam,npts,stval(nmar,1),erflg)

      if (erflg.eq.0) then
          call WRTSTR(0,100,150,20,'Just read from file:')
          call WRTSTR(0,(100 + 21*9),150,25,filnam)
c          call prtr4(0,300,125,7,'(E10.5)',10,
c     +                stval(nmar,1))
900       continue
      elseif (erflg.eq.1) then
          call LEVEL(0)
          call ERTONE()
          call WRTSTR(0,100,200,27,'ERROR: Could not open file:')
          call WRTSTR(0,(100+28*9),200,25,filnam)
          call LEVEL(1)
      else
          call LEVEL(0)
          call ERTONE()
          call WRTSTR(0,100,200,27,'ERROR: Could not read file:')
          call WRTSTR(0,(100+28*9),200,25,filnam)
          call LEVEL(1)
      endif

      return
      end

C ***************************************************************************
CH    REVISION HISTORY  :
C     VERSION       BY                  DATE        COMMENT
C       1.1     A. de Waal            06/01/90   Finally Commented
C     FILEND          :
C ***************************************************************************


C
C     FILE              : RDMTLD.FOR
C ***************************************************************************
CN    MODULE NAME       : rdmtld
CA    FUNCTION          : Read in a One-dimensional Array of MATLAB Data
CS    CALL SEQUENCE     : call rdmtld(filnam,npts,varble,erflg)
CI    INPUT PARAMETERS  : filnam: character - name of MATLAB data file
C                                             to be read from
C                         npts:   integer   - number of data points in
C                                             data file to be read from
CO    OUTPUT PARAMETERS : varble: real      - variable into which data
C                                             points are to be read
C                         erflg:  integer   - 0:  no error ocurred while
C                                               reading data
C                                             1:  error ocurred while
C                                               opening file
C                                             2:  error ocurred while
C                                               reading data
CG    GLOBAL VARIABLES  : Include files:  defnam.inc
CM    MODULES CALLED    : None
CE    ERROR CONDITIONS  : File opening or reading error
CC    COMMENTS          : This subroutine reads a one-dimensional array of
C                         data from a MATLAB data file
C ***************************************************************************
      subroutine rdmtld(filnam,npts,varble,erflg)
$include: 'defnam.inc'
      integer npts,erflg,funit,rderr
      real varble(*)
      character*25 filnam

20    format(BZ,E25.15)

C Make a filename usenam(50) = inpath(25)+filnam(25)
      call maknam(filnam,inpath,usenam)

      do 999 i = 1, npts
```

```
          varble(i) = 0.0
999   continue

      erflg = 0
      funit = 4
      open(funit,FILE=usenam,STATUS='UNKNOWN',IOSTAT=erflg)
      if (erflg.eq.0) then
          do 900 i = 1, 3
              read (funit,'(a)',IOSTAT=rderr,ERR=500)
900       continue
          do 890 i = 1, npts
              read (funit,20,IOSTAT=rderr,ERR=500) varble(i)
890       continue
500       if (rderr.ne.0) then
              erflg = 2
          endif
          close(funit)
      else
          erflg = 1
      endif
      return
      end
C *****************************************************************************
CH    REVISION HISTORY  :
C     VERSION       BY                DATE         COMMENT
C       0.0     A. de Waal          27/09/89       Creation
C       1.1     A. de Waal          06/01/90    Finally Commented
C     FILEND            :
C *****************************************************************************

C
C     FILE              : SAVDAT.FOR
C *****************************************************************************
CN    MODULE NAME       : savdat
CA    FUNCTION          : Drive Program Parameter Saving Menu
CS    CALL SEQUENCE     : call savdat()
CI    INPUT PARAMETERS  : None
CO    OUTPUT PARAMETERS : None
CG    GLOBAL VARIABLES  : Include files:  runvar.inc
CM    MODULES CALLED    : Fortran  :  inmenu,wrhead,savpro,savqr,savlqg
C                                     savdes,savlqa,savdea,savk
C                         Assembler:  DOMENU,WRTSTR,ERTONE
CE    ERROR CONDITIONS  : None
CC    COMMENTS          : Initializes menu screen and drives menu for the
C                         saving of program parameters.  Sets appropriate
C                         flags when selections are made.
C *****************************************************************************
      subroutine savdat
      implicit integer*2 (D)
$include: 'runvar.inc'
      integer*2 optl12

99    continue

      call inmenu()

C Write heading. (wrhead.for - iglib.lib)
      call wrhead(0,(720-53*9)/2,35,53,
     +   'Saving Package Default Data, Process and Design Data.')

C Get chosen option. (domenu.asm - iglib.lib)
      optl12 = DOMENU(112)

C Blank over heading. (util.asm - iglib.lib)
      call WRTSTR(0,(720-53*9)/2,35,53,
     +   '                                                     ')

      if (optl12.eq.1) then
          if (profl) then
              call savpro()
          else
              call ERTONE()
              call LEVEL(0)
              call WRTSTR(0,((720 - 36*9)/2),200,36,
     +            'Nothing Specified or Loaded to Save.')
              call LEVEL(1)
              key = INKEY(1)
              call WRTSTR(0,((720 - 36*9)/2),200,36,
     +                    '                                    ')
          endif

      elseif (optl12.eq.2) then
          if (qrfl) then
              call savqr()
          else
              call ERTONE()
              call LEVEL(0)
              call WRTSTR(0,((720 - 36*9)/2),200,36,
     +            'Nothing Specified or Loaded to Save.')
              call LEVEL(1)
              key = INKEY(1)
              call WRTSTR(0,((720 - 36*9)/2),200,36,
     +                    '                                    ')
          endif

      elseif (optl12.eq.3) then
          if (lqgfl) then
              call savlqg()
          else
              call ERTONE()
              call LEVEL(0)
              call WRTSTR(0,((720 - 35*9)/2),200,35,
     +            'Not a true and complete LQG design.')
              call LEVEL(1)
              key = INKEY(1)
              call WRTSTR(0,((720 - 35*9)/2),200,35,
     +                    '                                   ')
          endif

      elseif (optl12.eq.4) then
          if (deafl) then
              call savdes()
          else
              call ERTONE()
              call LEVEL(0)
              call WRTSTR(0,((720 - 38*9)/2),200,38,
```

```
     +            'Not a true and complete manual design.')
                  call LEVEL(1)
                  key = INKEY(1)
                  call WRTSTR(0,((720 - 38*9)/2),200,38,
     +                '                                      ')
              endif

             elseif (opt112.eq.5) then
                  if (lqafl) then
                      call savlqa()
                  else
                      call ERTONE()
                      call LEVEL(0)
                      call WRTSTR(0,((720 - 45*9)/2),200,45,
     +                    'Not a true, complete and analysed LQG design.')
                      call LEVEL(1)
                      key = INKEY(1)
                      call WRTSTR(0,((720 - 45*9)/2),200,45,
     +                    '                                            ')
                  endif

             elseif (opt112.eq.6) then
                  if (deafl) then
                      call savdea()
                  else
                      call ERTONE()
                      call LEVEL(0)
                      call WRTSTR(0,((720 - 48*9)/2),200,48,
     +                    'Not a true, complete and analysed manual design.')
                      call LEVEL(1)
                      key = INKEY(1)
                      call WRTSTR(0,((720 - 48*9)/2),200,48,
     +                    '                                              ')
                  endif

             elseif (opt112.eq.7) then
                  if (kfl) then
                      call savk()
                  else
                      call ERTONE()
                      call LEVEL(0)
                      call WRTSTR(0,((720 - 36*9)/2),200,36,
     +                    'Nothing Specified or Loaded to Save.')
                      call LEVEL(1)
                      key = INKEY(1)
                      call WRTSTR(0,((720 - 36*9)/2),200,36,
     +                    '                                    ')
                  endif

             elseif (opt112.ne.0) then
                  call ERTONE()

             endif

             if (opt112.ne.0) then
                 goto 99
             endif

             return
             end
C ***************************************************************************
CH    REVISION HISTORY  :
C     VERSION       BY                DATE       COMMENT
C       1.1     A. de Waal          06/01/90   Finally Commented
C     FILEND          :
C ***************************************************************************

C
C     FILE              : SAVDEA.FOR
C ***************************************************************************
CM    MODULE NAME       : savdea
CA    FUNCTION          : Save Non-LQG Design and Perf Index Analysis Data
CS    CALL SEQUENCE     : call savdea()
CI    INPUT PARAMETERS  : None
CO    OUTPUT PARAMETERS : None
CG    GLOBAL VARIABLES  : Include files:  keys.inc,syst.inc,defnam.inc,
C                                         perdat.inc
CM    MODULES CALLED    : Portran  :  wrhead,maknam,dodir,wrtitl,savmat,savper,
C                                     prderr
C                         Assembler:  WIPSCR,WRTSTR,STRIN,RSTERR,ERTONE,INKEY,
C                                     GETERR
CE    ERROR CONDITIONS  : Disk Errors as Indicated on Screen
CC    COMMENTS          : User is interrogated to determine to which file
C                         data is to be saved, and the following data
C                         for a non-LQG control system is saved:
C                             - Process Data (Model and System Specifications)
C                             - Non-LQG Controller (Controller (kc) and
C                                                   Observer (kf) Matrices)
C                             - Non-LQG system performance Index Analysis Data
C ***************************************************************************
      subroutine savdea()
      implicit integer*2 (c,G,L,S)
$include: 'keys.inc'
$include: 'syst.inc'
$include: 'defnam.inc'
$include: 'perdat.inc'
      integer*2 dima,mord,npts,nchan,grno,nanal,nchan
      integer*2 funit
      integer*2 key,ferr,derr,serr,len,chkio
      character*1 yesno

100   format('; ')
105   format(';WO, wf, winc, mord, dima =')

      call WIPSCR(0)

cC Write heading. (wrhead.for - iglib.lib)
c      call wrhead(0,(720-58*9)/2,35,58,
c     +   'Saving Perf. Index Analysis Data for Design not using LQG.')


C Information that would have been passed to savef subroutine
      infnam = ';Analysed Non-LQG Design '
      fname = 'des0.dea                  '
      funit = 4
```

```
C Information that would have been in defnam.inc file
c      outpth = 'c:\dewaal\progs\optcad\ '
       defdir = '*.dea'

C Information that would have been in syst.inc file

C Make a filename usenam(50) = outpth(25)+fname(25)
       call maknam(fname,outpth,usenam)


C Print out default directory page 0, yupperlim 180, length 4, defdir .dea,
C                          outpth 'c:\dewaal\progs\optcad  '
       call dodir(0,180,5,defdir,outpth)


       call wrhead(0,40,60,27,'Saving Information to File.')
       call wrhead(0,40,85,14,'Save to file :')
       call WRTSTR(0,40,110,30,'Hit RETURN to accept filename.')
       call WRTSTR(0,40,125,16,'Hit ESC to exit.')
99     continue
       key = STRIN(0,175,85,50,usenam)
       if ((key.ne.retk).and.(key.ne.esck)) goto 99
       call WRTSTR(0,40,110,30,'                                  ')
       call WRTSTR(0,40,125,16,'                    ')
       if (key.eq.retk) then
          call RSTERR()
          open(funit,FILE=usenam,STATUS='OLD',IOSTAT=ferr)
          if (ferr.gt.0) then
             derr = GETERR()
             if (derr.eq.-1) then
                close(funit)
                call WRTSTR(0,40,110,19,'Opening new file : ')
                call WRTSTR(0,215,110,50,usenam)
                open(funit,FILE=usenam,STATUS='NEW',IOSTAT=ferr)
                if (ferr.eq.0) then

                   write(funit,105,iostat=chkio,err=203)
                   write(funit,'(5g10.4)',iostat=chkio,err=203)
     +                  (syst(i),i=1,5)
                   dima = int(syst(5))
                   mord = int(syst(4))
                   write(funit,100,iostat=chkio,err=203)

                   mname = ';A =                                  '
                   serr = savmat(dima,dima,a,mname,funit)
                   if (serr.ne.0) goto 203
                   mname = ';B =                                  '
                   serr = savmat(dima,mord,b,mname,funit)
                   if (serr.ne.0) goto 203
                   mname = ';C =                                  '
                   serr = savmat(mord,dima,c,mname,funit)
                   if (serr.ne.0) goto 203
                   mname = ';KCI =                                '
                   serr = savmat(mord,dima,kci,mname,funit)
                   if (serr.ne.0) goto 203
                   mname = ';KFI =                                '
                   serr = savmat(dima,mord,kfi,mname,funit)
                   if (serr.ne.0) goto 203
                   mname = ';KC =                                 '
                   serr = savmat(mord,dima,kc,mname,funit)
                   if (serr.ne.0) goto 203
                   mname = ';KF =                                 '
                   serr = savmat(dima,mord,kf,mname,funit)
                   if (serr.ne.0) goto 203
                   mname = ';Ptr =                                '
                   serr = savmat(mord,mord,ptrsst,mname,funit)
                   if (serr.ne.0) goto 203

                   npts = 100
                   nanal = 5
                   nchan = 2

                   grno = 1
                   pernam = ';PERVAL: SENSOR NOISE REJ'
                   serr = savper(grno,nanal,nchan,pival,pernam,funit)
                   if (serr.ne.0) goto 203

                   grno = 2
                   pernam = ';PERVAL: DISTURBANCE REJ '
                   serr = savper(grno,nanal,nchan,pival,pernam,funit)
                   if (serr.ne.0) goto 203

                   grno = 3
                   pernam = ';PERVAL: SENSITIVITY     '
                   serr = savper(grno,nanal,nchan,pival,pernam,funit)
                   if (serr.ne.0) goto 203

                   grno = 4
                   pernam = ';PERVAL: EFFORT & TRACK  '
                   serr = savper(grno,nanal,nchan,pival,pernam,funit)
                   if (serr.ne.0) goto 203
cccc
                   mname = ';STABVAL: GAIN & PHASE M '
                   serr = savmat(2,5,stval,mname,funit)
                   if (serr.ne.0) goto 203
cccc
                   write(funit,'(a)',iostat=chkio,err=203)
     +                  ';PERFORMANCE INDEX COMMENTS'
                   do 111 i=1,4
                      if (i.eq.1)
     +                   write(funit,'(a)',iostat=chkio,err=203)
     +                        ';Input(1) and Output(2) Sens to Noise'
                      if (i.eq.2)
     +                   write(funit,'(a)',iostat=chkio,err=203)
     +                   ';Input(1) and Output(2) Sens to Disturbances'
                      if (i.eq.3)
     +                   write(funit,'(a)',iostat=chkio,err=203)
     +                   ';Input(1) and Output(2) Sens to Model Change'
                      if (i.eq.4)
     +                   write(funit,'(a)',iostat=chkio,err=203)
     +                   ';Control Effort(1) and Setpoint Tracking(2)'
                      do 112 j=1,2
                         if (j.eq.1)
     +                      write(funit,'(a)',iostat=chkio,err=203)
     +                           ';Channel 1'
                         if (j.eq.2)
```

```
     +                    write(funit,'(a)',iostat=chkio,err=203)
     +                        ';Channel 2'
                       do 113 k = 1, 5
                          write(funit,'(a)',iostat=chkio,err=203)
     +                        picom(i,j,k)
113                    continue
112                 continue
111                 continue
                    write(funit,100,iostat=chkio,err=203)

                    write(funit,'(a)',iostat=chkio,err=203)
     +                   ';GAIN AND PHASE MARGIN COMMENTS'
                    write(funit,'(a)',iostat=chkio,err=203)
     +                   (stcom(i),i=1,2)
                    write(funit,100,iostat=chkio,err=203)


                    grno = 1
                    pernam = ';PERDAT: SENSOR NOISE REJ'
                    serr = savper(grno,npts,nchan,pidat,pernam,funit)
                    if (serr.ne.0) goto 203
                    grno = 2
                    pernam = ';PERDAT: DISTURBANCE REJ '
                    serr = savper(grno,npts,nchan,pidat,pernam,funit)
                    if (serr.ne.0) goto 203
                    grno = 3
                    pernam = ';PERDAT: SENSITIVITY     '
                    serr = savper(grno,npts,nchan,pidat,pernam,funit)
                    if (serr.ne.0) goto 203
                    grno = 4
                    pernam = ';PERDAT: EFFORT & TRACK  '
                    serr = savper(grno,npts,nchan,pidat,pernam,funit)
                    if (serr.ne.0) goto 203


                    write(funit,100,iostat=chkio,err=203)
                    write(funit,'(a)',iostat=chkio,
     +                   err=203) '; Group of Data :'
                    write(funit,'(a)',iostat=chkio,
     +                   err=203) '; ****************'
                    write(funit,'(a)',iostat=chkio,err=203) infnam
                    if (chkio.eq.0) goto 201
203                 call ERTONE()
201                 continue

                    close(funit)
                    if (serr.eq.0) then
                       call WRTSTR(0,40,130,20,'Data save completed.')
                    else
                       call WRTSTR(0,40,130,36,
     +                     '*** Error : Data save not completed.')
                       call WRTSTR(0,40,145,26,
     +                         'Error ocurred while saving')
                       call WRTSTR(0,40,160,25,mname)
                    endif
                 else
                    call ERTONE()
                    call WRTSTR(0,40,130,33,
     +                   '*** Error : Cannot open new file.')
                    close(funit)
                 endif
              else
                 call prderr(0,40,130,derr)
              endif
           else
              len = LENSTR(50,usenam)
              call ERTONE()
              call WRTSTR(0,40,110,6,'File :')
              call LEVEL(2)
              call BLKFIL(115,113,(len*9),14)
              call WRTSTR(0,115,110,len,usenam)
              call LEVEL(1)
              call WRTSTR(0,(115+(len*9)),110,16,' already exists.')
              call WRTSTR(0,40,140,26,'Overwrite the file (y/n) ?')
              yesno = 'n'
98            continue
              key = STRIN(0,280,140,1,yesno)
              if (key.ne.retk) goto 98
              if ((yesno.eq.'y').or.(yesno.eq.'Y')) then
                 close(funit)
                 call RSTERR()
                 open(funit,FILE=usenam,STATUS='OLD',IOSTAT=ferr)
                 if (ferr.eq.0) then

                    write(funit,105,iostat=chkio,err=223)
                    write(funit,'(5g10.4)',iostat=chkio,err=223)
     +                   (syst(i),i=1,5)
                    dima = int(syst(5))
                    mord = int(syst(4))
                    write(funit,100,iostat=chkio,err=223)


                    mname = ';A =                    '
                    serr = savmat(dima,dima,a,mname,funit)
                    if (serr.ne.0) goto 223
                    mname = ';B =                    '
                    serr = savmat(dima,mord,b,mname,funit)
                    if (serr.ne.0) goto 223
                    mname = ';C =                    '
                    serr = savmat(mord,dima,c,mname,funit)
                    if (serr.ne.0) goto 223
                    mname = ';KCI =                  '
                    serr = savmat(mord,dima,kci,mname,funit)
                    if (serr.ne.0) goto 223
                    mname = ';KFI =                  '
                    serr = savmat(dima,mord,kfi,mname,funit)
                    if (serr.ne.0) goto 223
                    mname = ';KC =                   '
                    serr = savmat(mord,dima,kc,mname,funit)
                    if (serr.ne.0) goto 223
                    mname = ';KF =                   '
                    serr = savmat(dima,mord,kf,mname,funit)
                    if (serr.ne.0) goto 223
                    mname = ';Ptr =                  '
                    serr = savmat(mord,mord,ptrsst,mname,funit)
                    if (serr.ne.0) goto 223
```

```
                npts = 100
                nanal = 5
                nchan = 2

                grno = 1
                pernam = ';PERVAL: SENSOR NOISE REJ'
                serr = savper(grno,nanal,nchan,pival,pernam,funit)
                if (serr.ne.0) goto 223

                grno = 2
                pernam = ';PERVAL: DISTURBANCE REJ '
                serr = savper(grno,nanal,nchan,pival,pernam,funit)
                if (serr.ne.0) goto 223

                grno = 3
                pernam = ';PERVAL: SENSITIVITY     '
                serr = savper(grno,nanal,nchan,pival,pernam,funit)
                if (serr.ne.0) goto 223

                grno = 4
                pernam = ';PERVAL: EFFORT & TRACK  '
                serr = savper(grno,nanal,nchan,pival,pernam,funit)
                if (serr.ne.0) goto 223
cccc
                mname = ';STABVAL: GAIN & PHASE M '
                serr = savmat(2,5,stval,mname,funit)
                if (serr.ne.0) goto 223
cccc
                write(funit,'(a)',iostat=chkio,err=223)
     +               ';PERFORMANCE INDEX COMMENTS'
                do 881 i=1,4
                    if (i.eq.1)
     +                  write(funit,'(a)',iostat=chkio,err=223)
     +                      ';Input(1) and Output(2) Sens to Noise'
                    if (i.eq.2)
     +                  write(funit,'(a)',iostat=chkio,err=223)
     +                  ';Input(1) and Output(2) Sens to Disturbances'
                    if (i.eq.3)
     +                  write(funit,'(a)',iostat=chkio,err=223)
     +                  ';Input(1) and Output(2) Sens to Model Change'
                    if (i.eq.4)
     +                  write(funit,'(a)',iostat=chkio,err=223)
     +                  ';Control Effort(1) and Setpoint Tracking(2)'
                    do 882 j=1,2
                        if (j.eq.1)
     +                      write(funit,'(a)',iostat=chkio,err=223)
     +                          ';Channel 1'
                        if (j.eq.2)
     +                      write(funit,'(a)',iostat=chkio,err=223)
     +                          ';Channel 2'
                        do 883 k = 1, 5
                            write(funit,'(a)',iostat=chkio,err=223)
     +                          picom(i,j,k)
883                     continue
882                 continue
881             continue
                write(funit,100,iostat=chkio,err=223)

                write(funit,'(a)',iostat=chkio,err=223)
     +              ';GAIN AND PHASE MARGIN COMMENTS'
                write(funit,'(a)',iostat=chkio,err=223)
     +              (stcom(i),i=1,2)
                write(funit,100,iostat=chkio,err=223)


                grno = 1
                pernam = ';PERDAT: SENSOR NOISE REJ'
                serr = savper(grno,npts,nchan,pidat,pernam,funit)
                if (serr.ne.0) goto 223
                grno = 2
                pernam = ';PERDAT: DISTURBANCE REJ '
                serr = savper(grno,npts,nchan,pidat,pernam,funit)
                if (serr.ne.0) goto 223
                grno = 3
                pernam = ';PERDAT: SENSITIVITY     '
                serr = savper(grno,npts,nchan,pidat,pernam,funit)
                if (serr.ne.0) goto 223
                grno = 4
                pernam = ';PERDAT: EFFORT & TRACK  '
                serr = savper(grno,npts,nchan,pidat,pernam,funit)
                if (serr.ne.0) goto 223


                write(funit,100,iostat=chkio,err=223)
                write(funit,'(a)',iostat=chkio,
     +                err=223) '; Group of Data :'
                write(funit,'(a)',iostat=chkio,
     +                err=223) '; ****************'
                write(funit,'(a)',iostat=chkio,err=223) infnam
                if (chkio.eq.0) goto 221
223             call ERTONE()
221             continue

                close(funit)
                if (serr.eq.0) then
                   call WRTSTR(0,40,170,20,
     +                 'Data save completed.')
                else
                   call WRTSTR(0,40,170,36,
     +                 '*** Error : Data save not completed.')
                endif
             else
                derr = GETERR()
                if (derr.eq.-1) then
                   call ERTONE()
                   call WRTSTR(0,40,140,38,
     +                 '*** Error : Cannot overwrite old file.')
                   close(funit)
                else
                   call prderr(0,40,170,derr)
                endif
             endif
          else
             call WRTSTR(0,40,170,30,
     +            'Data save operation cancelled.')
```

```
            endif
         endif
      else
         call WRTSTR(0,40,170,30,
     +              'Data save operation cancelled.')
      endif
      close(funit)

      key = INKEY(1)


cC Blank over heading. (util.asm - iglib.lib)
c      call WRTSTR(0,(720-58*9)/2,35,58,
c     +   '                                                          ')

      return
      end
C ********************************************************************************
CH    REVISION HISTORY  :
C     VERSION       BY                DATE        COMMENT
C       1.1     A. de Waal           06/01/90   Finally Commented
C     FILEND          :
C ********************************************************************************
C
C     FILE              : SAVDES.FOR
C ********************************************************************************
CN    MODULE NAME       : savdes
CA    FUNCTION          : Save Non-LQG Control System Design
CS    CALL SEQUENCE     : call savdes()
CI    INPUT PARAMETERS  : None
CO    OUTPUT PARAMETERS : None
CG    GLOBAL VARIABLES  : Include files:  keys.inc,syst.inc,defnam.inc,
C                                         perdat.inc
CM    MODULES CALLED    : Fortran   :  wrhead,maknam,dodir,wrtitl,savmat,
C                                      prderr
C                         Assembler:   WIPSCR,WRTSTR,STRIN,RSTERR,ERTONE,INKEY,
C                                      GETERR,
CE    ERROR CONDITIONS  : Disk Errors as Indicated on Screen
CC    COMMENTS          : User is interrogated to determine to which file
C                         data is to be saved, and the following data
C                         for a non-LQG control system is saved:
C                             - Process Data (Model and System Specifications)
C                             - Non-LQG Controller (Controller (kc) and
C                                                   Observer (kf) Matrices)
C ********************************************************************************
      subroutine savdes()
      implicit integer*2 (c,G,L,S)
$include: 'keys.inc'
$include: 'syst.inc'
$include: 'defnam.inc'
$include: 'perdat.inc'
      integer*2 dima,mord,npts,nchan,grno,nanal,nchan
      integer*2 funit
      integer*2 key,ferr,derr,serr,len,chkio
      character*1 yesno

100   format('; ')
105   format(';w0, wf, winc, mord, dima =')

      call WIPSCR(0)


C Information that would have been passed to savef subroutine
      infnam = ';Non-LQG Design (Unan)'
      fname = 'des0.des'
      funit = 4

C Information that would have been in defnam.inc file
c     outpth = 'c:\dewaal\progs\optcad\ '
      defdir = '*.des'

C Information that would have been in syst.inc file

C Make a filename usenam(50) = outpth(25)+fname(25)
      call maknam(fname,outpth,usenam)


C Print out default directory page 0, yupperlim 180, length 4, defdir .des,
C                            outpth 'c:\dewaal\progs\optcad  '
      call dodir(0,180,5,defdir,outpth)


      call wrhead(0,40,60,27,'Saving Information to File.')
      call wrhead(0,40,85,14,'Save to file :')
      call WRTSTR(0,40,110,30,'Hit RETURN to accept filename.')
      call WRTSTR(0,40,125,16,'Hit ESC to exit.')
99    continue
      key = STRIN(0,175,85,50,usenam)
      if ((key.ne.retk).and.(key.ne.esck)) goto 99
      call WRTSTR(0,40,110,30,'                                  ')
      call WRTSTR(0,40,125,16,'                 ')
      if (key.eq.retk) then
         call RSTERR()
         open(funit,FILE=usenam,STATUS='OLD',IOSTAT=ferr)
         if (ferr.gt.0) then
            derr = GETERR()
            if (derr.eq.-1) then
               close(funit)
               call WRTSTR(0,40,110,19,'Opening new file : ')
               call WRTSTR(0,215,110,50,usenam)
               open(funit,FILE=usenam,STATUS='NEW',IOSTAT=ferr)
               if (ferr.eq.0) then

                  write(funit,105,iostat=chkio,err=203)
                  write(funit,'(5g10.4)',iostat=chkio,err=203)
     +                 (syst(i),i=1,5)
                  dima = int(syst(5))
                  mord = int(syst(4))
                  write(funit,100,iostat=chkio,err=203)

                  mname = ';A =                    '
                  serr = savmat(dima,dima,a,mname,funit)
                  if (serr.ne.0) goto 203
                  mname = ';B =                    '
                  serr = savmat(dima,mord,b,mname,funit)
```

```
                if (serr.ne.0) goto 203
                mname = ';C =                    ,
                serr = savmat(mord,dima,c,mname,funit)
                if (serr.ne.0) goto 203
                mname = ';KCI =                  ,
                serr = savmat(mord,dima,kci,mname,funit)
                if (serr.ne.0) goto 203
                mname = ';KFI =                  ,
                serr = savmat(dima,mord,kfi,mname,funit)
                if (serr.ne.0) goto 203
                mname = ';KC =                   ,
                serr = savmat(mord,dima,kc,mname,funit)
                if (serr.ne.0) goto 203
                mname = ';KF =                   ,
                serr = savmat(dima,mord,kf,mname,funit)
                if (serr.ne.0) goto 203
                mname = ';Ptr =                  ,
                serr = savmat(mord,mord,ptrsst,mname,funit)
                if (serr.ne.0) goto 203


                write(funit,100,iostat=chkio,err=203)
                write(funit,'(a)',iostat=chkio,
     +               err=203) '; Group of Data :'
                write(funit,'(a)',iostat=chkio,
     +               err=203) '; ***************'
                write(funit,'(a)',iostat=chkio,err=203) infnam
                if (chkio.eq.0) goto 201
203             call ERTONE()
201             continue

                close(funit)
                if (serr.eq.0) then
                   call WRTSTR(0,40,130,20,'Data save completed.')
                else
                   call WRTSTR(0,40,130,36,
     +                 '*** Error : Data save not completed.')
                   call WRTSTR(0,40,145,26,
     +                     'Error ocurred while saving')
                   call WRTSTR(0,40,160,25,mname)
                endif
             else
                call ERTONE()
                call WRTSTR(0,40,130,33,
     +               '*** Error : Cannot open new file.')
                close(funit)
             endif
          else
             call prderr(0,40,130,derr)
          endif
       else
          len = LENSTR(50,usenam)
          call ERTONE()
          call WRTSTR(0,40,110,6,'File :')
          call LEVEL(2)
          call BLKFIL(115,113,(len*9),14)
          call WRTSTR(0,115,110,len,usenam)
          call LEVEL(1)
          call WRTSTR(0,(115+(len*9)),110,16,' already exists.')
          call WRTSTR(0,40,140,26,'Overwrite the file (y/n) ?')
          yesno = 'n'
98        continue
          key = STRIN(0,280,140,1,yesno)
          if (key.ne.retk) goto 98
          if ((yesno.eq.'y').or.(yesno.eq.'Y')) then
             close(funit)
             call RSTERR()
             open(funit,FILE=usenam,STATUS='OLD',IOSTAT=ferr)
             if (ferr.eq.0) then

                write(funit,105,iostat=chkio,err=223)
                write(funit,'(5g10.4)',iostat=chkio,err=223)
     +               (syst(i),i=1,5)
                dima = int(syst(5))
                mord = int(syst(4))
                write(funit,100,iostat=chkio,err=223)


                mname = ';A =                    ,
                serr = savmat(dima,dima,a,mname,funit)
                if (serr.ne.0) goto 223
                mname = ';B =                    ,
                serr = savmat(dima,mord,b,mname,funit)
                if (serr.ne.0) goto 223
                mname = ';C =                    ,
                serr = savmat(mord,dima,c,mname,funit)
                if (serr.ne.0) goto 223
                mname = ';KCI =                  ,
                serr = savmat(mord,dima,kci,mname,funit)
                if (serr.ne.0) goto 223
                mname = ';KFI =                  ,
                serr = savmat(dima,mord,kfi,mname,funit)
                if (serr.ne.0) goto 223
                mname = ';KC =                   ,
                serr = savmat(mord,dima,kc,mname,funit)
                if (serr.ne.0) goto 223
                mname = ';KF =                   ,
                serr = savmat(dima,mord,kf,mname,funit)
                if (serr.ne.0) goto 223
                mname = ';Ptr =                  ,
                serr = savmat(mord,mord,ptrsst,mname,funit)
                if (serr.ne.0) goto 223


                write(funit,100,iostat=chkio,err=223)
                write(funit,'(a)',iostat=chkio,
     +               err=223) '; Group of Data :'
                write(funit,'(a)',iostat=chkio,
     +               err=223) '; ***************'
                write(funit,'(a)',iostat=chkio,err=223) infnam
                if (chkio.eq.0) goto 221
223             call ERTONE()
221             continue

                close(funit)
                if (serr.eq.0) then
```

```
                  call WRTSTR(0,40,170,20,
     +                 'Data save completed.')
               else
                  call WRTSTR(0,40,170,36,
     +                 '*** Error : Data save not completed.')
               endif
            else
               derr = GETERR()
               if (derr.eq.-1) then
                  call ERTONE()
                  call WRTSTR(0,40,140,38,
     +                 '*** Error : Cannot overwrite old file.')
                  close(funit)
               else
                  call prderr(0,40,170,derr)
               endif
            endif
         else
            call WRTSTR(0,40,170,30,
     +              'Data save operation cancelled.')
         endif
      endif
      else
         call WRTSTR(0,40,170,30,
     +           'Data save operation cancelled.')
      endif
      close(funit)

      key = INKEY(1)

cC Write heading. (wrhead.for - iglib.lib)
c      call wrhead(0,(720-57*9)/2,35,57,
c     +   'Saving System with Contr./Observ. Designed Not Using LQG.')


cC Blank over heading. (util.asm - iglib.lib)
c      call WRTSTR(0,(720-57*9)/2,35,57,
c     +   '                                                          ')

      return
      end
C ***************************************************************************
CH    REVISION HISTORY  :
C     VERSION       BY                DATE        COMMENT
C       1.1     A. de Waal          06/01/90   Finally Commented
C     FILEND          :
C ***************************************************************************



C
C     FILE              : SAVK.FOR
C ***************************************************************************
CN    MODULE NAME       : savk
CA    FUNCTION          : Save Controller and Observer Matrices
CS    CALL SEQUENCE     : call savk()
CI    INPUT PARAMETERS  : None
CO    OUTPUT PARAMETERS : None
CG    GLOBAL VARIABLES  : Include files:  keys.inc,syst.inc,defnam.inc,
C                                         perdat.inc
CM    MODULES CALLED    : Fortran  :  wrhead,maknam,dodir,wrtitl,savmat,
C                                     prderr
C                         Assembler:  WIPSCR,WRTSTR,STRIN,RSTERR,ERTONE,INKEY,
C                                     GETERR
CE    ERROR CONDITIONS  : Disk Errors as Indicated on Screen
CC    COMMENTS          : User is interrogated to determine to which file
C                         data is to be saved, and the following data
C                         is saved:
C                             - Controller (Controller (kc) and
C                                           Observer (kf) Matrices)
C ***************************************************************************
      subroutine savk()
      implicit integer*2 (c,G,L,S)
$include: 'keys.inc'
$include: 'syst.inc'
$include: 'defnam.inc'
$include: 'perdat.inc'
      integer*2 dima,mord,npts,nchan,grno,nanal,nchan
      integer*2 funit
      integer*2 key,ferr,derr,serr,len,chkio
      character*1 yesno

100   format('; ')
105   format(';w0, wf, winc, mord, dima =')

      call WIPSCR(0)

cC Write heading. (wrhead.for - iglib.lib)
c      call wrhead(0,(720-52*9)/2,35,52,
c     +   'Saving Controller and Kalman Observer Gain Matrices.')


C Information that would have been passed to savef subroutine
      infnam = ';Contr & Kalman Gain Mxs '
      fname = 'des0.k
      funit = 4

C Information that would have been in defnam.inc file
c     outpth = 'c:\dewaal\progs\optcad\ '
      defdir = '*.k  '

C Information that would have been in syst.inc file

C Make a filename usenam(50) = outpth(25)+fname(25)
      call maknam(fname,outpth,usenam)


C Print out default directory page 0, yupperlim 180, length 4, defdir .k  ,
C                               outpth 'c:\dewaal\progs\optcad  '
      call dodir(0,180,5,defdir,outpth)


      call wrhead(0,40,60,27,'Saving Information to File.')
      call wrhead(0,40,85,14,'Save to file :')
      call WRTSTR(0,40,110,30,'Hit RETURN to accept filename.')
      call WRTSTR(0,40,125,16,'Hit ESC to exit.')
```

```
99    continue
      key = STRIN(0,175,85,50,usenam)
      if ((key.ne.retk).and.(key.ne.esck)) goto 99
      call WRTSTR(0,40,110,30,'                                    ')
      call WRTSTR(0,40,125,16,'                  ')
      if (key.eq.retk) then
         call RSTERR()
         open(funit,FILE=usenam,STATUS='OLD',IOSTAT=ferr)
         if (ferr.gt.0) then
            derr = GETERR()
            if (derr.eq.-1) then
               close(funit)
               call WRTSTR(0,40,110,19,'Opening new file : ')
               call WRTSTR(0,215,110,50,usenam)
               open(funit,FILE=usenam,STATUS='NEW',IOSTAT=ferr)
               if (ferr.eq.0) then

                  mname = ';KC =                    '
                  serr = savmat(mord,dima,kc,mname,funit)
                  if (serr.ne.0) goto 203
                  mname = ';KF =                    '
                  serr = savmat(dima,mord,kf,mname,funit)
                  if (serr.ne.0) goto 203
                  mname = ';Ptr =                   '
                  serr = savmat(mord,mord,ptrsst,mname,funit)
                  if (serr.ne.0) goto 203


                  write(funit,100,iostat=chkio,err=203)
                  write(funit,'(a)',iostat=chkio,
     +                 err=203) '; Group of Data :'
                  write(funit,'(a)',iostat=chkio,
     +                 err=203) '; ****************'
                  write(funit,'(a)',iostat=chkio,err=203) infnam
                  if (chkio.eq.0) goto 201
203               call ERTONE()
201               continue

                  close(funit)
                  if (serr.eq.0) then
                     call WRTSTR(0,40,130,20,'Data save completed.')
                  else
                     call WRTSTR(0,40,130,36,
     +                    '*** Error : Data save not completed.')
                     call WRTSTR(0,40,145,26,
     +                       'Error ocurred while saving')
                     call WRTSTR(0,40,160,25,mname)
                  endif
               else
                  call ERTONE()
                  call WRTSTR(0,40,130,33,
     +                 '*** Error : Cannot open new file.')
                  close(funit)
               endif
            else
               call prderr(0,40,130,derr)
            endif
         else
            len = LENSTR(50,usenam)
            call ERTONE()
            call WRTSTR(0,40,110,6,'File :')
            call LEVEL(2)
            call BLKFIL(115,113,(len*9),14)
            call WRTSTR(0,115,110,len,usenam)
            call LEVEL(1)
            call WRTSTR(0,(115+(len*9)),110,16,' already exists.')
            call WRTSTR(0,40,140,26,'Overwrite the file (y/n) ?')
            yesno = 'n'
98          continue
            key = STRIN(0,280,140,1,yesno)
            if (key.ne.retk) goto 98
            if ((yesno.eq.'y').or.(yesno.eq.'Y')) then
               close(funit)
               call RSTERR()
               open(funit,FILE=usenam,STATUS='OLD',IOSTAT=ferr)
               if (ferr.eq.0) then
                  mname = ';KC =                    '
                  serr = savmat(mord,dima,kc,mname,funit)
                  if (serr.ne.0) goto 223
                  mname = ';KF =                    '
                  serr = savmat(dima,mord,kf,mname,funit)
                  if (serr.ne.0) goto 223
                  mname = ';Ptr =                   '
                  serr = savmat(mord,mord,ptrsst,mname,funit)
                  if (serr.ne.0) goto 223


                  write(funit,100,iostat=chkio,err=223)
                  write(funit,'(a)',iostat=chkio,
     +                 err=223) '; Group of Data :'
                  write(funit,'(a)',iostat=chkio,
     +                 err=223) '; ****************'
                  write(funit,'(a)',iostat=chkio,err=223) infnam
                  if (chkio.eq.0) goto 221
223               call ERTONE()
221               continue

                  close(funit)
                  if (serr.eq.0) then
                     call WRTSTR(0,40,170,20,
     +                    'Data save completed.')
                  else
                     call WRTSTR(0,40,170,36,
     +                    '*** Error : Data save not completed.')
                  endif
               else
                  derr = GETERR()
                  if (derr.eq.-1) then
                     call ERTONE()
                     call WRTSTR(0,40,140,38,
     +                    '*** Error : Cannot overwrite old file.')
                     close(funit)
                  else
                     call prderr(0,40,170,derr)
                  endif
               endif
```

```
            else
               call WRTSTR(0,40,170,30,
     +              'Data save operation cancelled.')
            endif
         endif
       else
          call WRTSTR(0,40,170,30,
     +            'Data save operation cancelled.')
       endif
       close(funit)

       key = INKEY(1)

cC Blank over heading. (util.asm - iglib.lib)
c       call WRTSTR(0,(720-52*9)/2,35,52,
c      +    '                                                    ')

       return
       end
C ********************************************************************************
CH     REVISION HISTORY  :
C      VERSION       BY                 DATE        COMMENT
C        1.1      A. de Waal           06/01/90   Finally Commented
C      FILEND            :
C ********************************************************************************
C
C      FILE              : SAVLQA.FOR
C ********************************************************************************
CN     MODULE NAME       : savlqa
CA     FUNCTION          : Save LQG Design and Perf Index Analysis Data
CS     CALL SEQUENCE     : call savlqa()
CI     INPUT PARAMETERS  : None
CO     OUTPUT PARAMETERS : None
CG     GLOBAL VARIABLES  : Include files:  keys.inc,syst.inc,defnam.inc,
C                                          perdat.inc
CM     MODULES CALLED    : Fortran   :  wrhead,maknam,dodir,wrtitl,savmat,savper,
C                                       prderr
C                          Assembler:  WIPSCR,WRTSTR,STRIN,RSTERR,ERTONE,INKEY,
C                                       GETERR
CE     ERROR CONDITIONS  : Disk Errors as Indicated on Screen
CC     COMMENTS          : User is interrogated to determine to which file
C                          data is to be saved, and the following data
C                          for a LQG control system is saved:
C                              - Process Data (Model and System Specifications)
C                              - LQG Weighting and Covariance Matrices
C                              - LQG Controller (Controller (kc) and
C                                                Observer (kf) Matrices)
C                              - LQG system performance Index Analysis Data
C ********************************************************************************
       subroutine savlqa()
       implicit integer*2 (c,G,L,S)
$include: 'keys.inc'
$include: 'syst.inc'
$include: 'defnam.inc'
$include: 'perdat.inc'
       integer*2 dima,mord,npts,nchan,grno,nanal,nchan
       integer*2 funit
       integer*2 key,ferr,derr,serr,len,chkio
       character*1 yesno

100    format('; ')
105    format(';w0, wf, winc, mord, dima =')
       call WIPSCR(0)

cC Write heading. (wrhead.for - iglib.lib)
c       call wrhead(0,(720-56*9)/2,35,56,
c      +    'Saving Perf. Index Analysis Data for Current LQG Design.')

C Information that would have been passed to savef subroutine
       infnam = ';Analysed LQG Design    '
       fname = 'des0.lqa                 '
       funit = 4

C Information that would have been in defnam.inc file
c       outpth = 'c:\progs\optcad\data\    '
       defdir = '*.lqa'

C Information that would have been in syst.inc file

C Make a filename usenam(50) = outpth(25)+fname(25)
       call maknam(fname,outpth,usenam)


C Print out default directory page 0, yupperlim 180, length 4, defdir .lqa,
C                           outpth 'c:\dewaal\progs\optcad  '
       call dodir(0,180,5,defdir,outpth)


       call wrhead(0,40,60,27,'Saving Information to File.')
       call wrhead(0,40,85,14,'Save to file :')
       call WRTSTR(0,40,110,30,'Hit RETURN to accept filename.')
       call WRTSTR(0,40,125,16,'Hit ESC to exit.')
99     continue
       key = STRIN(0,175,85,50,usenam)
       if ((key.ne.retk).and.(key.ne.esck)) goto 99
       call WRTSTR(0,40,110,30,'                                    ')
       call WRTSTR(0,40,125,16,'                  ')
       if (key.eq.retk) then
          call RSTERR()
          open(funit,FILE=usenam,STATUS='OLD',IOSTAT=ferr)
          if (ferr.gt.0) then
             derr = GETERR()
             if (derr.eq.-1) then
                close(funit)
                call WRTSTR(0,40,110,19,'Opening new file : ')
                call WRTSTR(0,215,110,50,usenam)
                open(funit,FILE=usenam,STATUS='NEW',IOSTAT=ferr)
                if (ferr.eq.0) then

                   write(funit,105,iostat=chkio,err=203)
                   write(funit,'(5g10.4)',iostat=chkio,err=203)
     +                  (syst(i),i=1,5)
                   dima = int(syst(5))
                   mord = int(syst(4))
                   write(funit,100,iostat=chkio,err=203)
```

```
      mname = ';A =                    '
      serr = savmat(dima,dima,a,mname,funit)
      if (serr.ne.0) goto 203
      mname = ';B =                    '
      serr = savmat(dima,mord,b,mname,funit)
      if (serr.ne.0) goto 203
      mname = ';C =                    '
      serr = savmat(mord,dima,c,mname,funit)
      if (serr.ne.0) goto 203
      mname = ';KCI =                  '
      serr = savmat(mord,dima,kci,mname,funit)
      if (serr.ne.0) goto 203
      mname = ';KFI =                  '
      serr = savmat(dima,mord,kfi,mname,funit)
      if (serr.ne.0) goto 203
      mname = ';Q1 =                   '
      serr = savmat(dima,dima,q1,mname,funit)
      if (serr.ne.0) goto 203
      mname = ';R1 =                   '
      serr = savmat(mord,mord,r1,mname,funit)
      if (serr.ne.0) goto 203
      mname = ';KC =                   '
      serr = savmat(mord,dima,kc,mname,funit)
      if (serr.ne.0) goto 203
      mname = ';Q =                    '
      serr = savmat(dima,dima,q,mname,funit)
      if (serr.ne.0) goto 203
      mname = ';R =                    '
      serr = savmat(mord,mord,r,mname,funit)
      if (serr.ne.0) goto 203
      mname = ';KF =                   '
      serr = savmat(dima,mord,kf,mname,funit)
      if (serr.ne.0) goto 203
      mname = ';Ptr =                  '
      serr = savmat(mord,mord,ptrsst,mname,funit)
      if (serr.ne.0) goto 203

      npts = 100
      nanal = 5
      nchan = 2

      grno = 1
      pernam = ';PERVAL: SENSOR NOISE REJ'
      serr = savper(grno,nanal,nchan,pival,pernam,funit)
      if (serr.ne.0) goto 203

      grno = 2
      pernam = ';PERVAL: DISTURBANCE REJ '
      serr = savper(grno,nanal,nchan,pival,pernam,funit)
      if (serr.ne.0) goto 203

      grno = 3
      pernam = ';PERVAL: SENSITIVITY     '
      serr = savper(grno,nanal,nchan,pival,pernam,funit)
      if (serr.ne.0) goto 203

      grno = 4
      pernam = ';PERVAL: EFFORT & TRACK  '
      serr = savper(grno,nanal,nchan,pival,pernam,funit)
      if (serr.ne.0) goto 203
cccc
      mname = ';STABVAL: GAIN & PHASE M '
      serr = savmat(2,5,stval,mname,funit)
      if (serr.ne.0) goto 203
cccc
      write(funit,'(a)',iostat=chkio,err=203)
     +     ';PERFORMANCE INDEX COMMENTS'
      do 111 i=1,4
          if (i.eq.1)
     +        write(funit,'(a)',iostat=chkio,err=203)
     +             ';Input(1) and Output(2) Sens to Noise'
          if (i.eq.2)
     +        write(funit,'(a)',iostat=chkio,err=203)
     +        ';Input(1) and Output(2) Sens to Disturbances'
          if (i.eq.3)
     +        write(funit,'(a)',iostat=chkio,err=203)
     +        ';Input(1) and Output(2) Sens to Model Change'
          if (i.eq.4)
     +        write(funit,'(a)',iostat=chkio,err=203)
     +        ';Control Effort(i) and Setpoint Tracking(2)'
          do 112 j=1,2
              if (j.eq.1)
     +            write(funit,'(a)',iostat=chkio,err=203)
     +                 ';Channel 1'
              if (j.eq.2)
     +            write(funit,'(a)',iostat=chkio,err=203)
     +                 ';Channel 2'
              do 113 k = 1, 5
                  write(funit,'(a)',iostat=chkio,err=203)
     +                picom(i,j,k)
113           continue
112       continue
111   continue
      write(funit,100,iostat=chkio,err=203)

      write(funit,'(a)',iostat=chkio,err=203)
     +     ';GAIN AND PHASE MARGIN COMMENTS'
      write(funit,'(a)',iostat=chkio,err=203)
     +     (stcom(i),i=1,2)
      write(funit,100,iostat=chkio,err=203)


      grno = 1
      pernam = ';PERDAT: SENSOR NOISE REJ'
      serr = savper(grno,npts,nchan,pidat,pernam,funit)
      if (serr.ne.0) goto 203
      grno = 2
      pernam = ';PERDAT: DISTURBANCE REJ '
      serr = savper(grno,npts,nchan,pidat,pernam,funit)
      if (serr.ne.0) goto 203
      grno = 3
      pernam = ';PERDAT: SENSITIVITY     '
      serr = savper(grno,npts,nchan,pidat,pernam,funit)
      if (serr.ne.0) goto 203
      grno = 4
```

```
                pernam = ';PERDAT: EFFORT & TRACK  '
                serr = savper(grno,npts,nchan,pidat,pernam,funit)
                if (serr.ne.0) goto 203


                write(funit,100,iostat=chkio,err=203)
                write(funit,'(a)',iostat=chkio,
     +               err=203) '; Group of Data :'
                write(funit,'(a)',iostat=chkio,
     +               err=203) '; ***************'
                write(funit,'(a)',iostat=chkio,err=203) infnam
                if (chkio.eq.0) goto 201
203             call ERTONE()
201             continue

                close(funit)
                if (serr.eq.0) then
                   call WRTSTR(0,40,130,20,'Data save completed.')
                else
                   call WRTSTR(0,40,130,36,
     +                  '*** Error : Data save not completed.')
                   call WRTSTR(0,40,145,26,
     +                     'Error ocurred while saving')
                   call WRTSTR(0,40,160,25,mname)
                endif
             else
                call ERTONE()
                call WRTSTR(0,40,130,33,
     +               '*** Error : Cannot open new file.')
                close(funit)
             endif
          else
             call prderr(0,40,130,derr)
          endif
       else
          len = LENSTR(50,usenam)
          call ERTONE()
          call WRTSTR(0,40,110,6,'File :')
          call LEVEL(2)
          call BLKFIL(115,113,(len*9),14)
          call WRTSTR(0,115,110,len,usenam)
          call LEVEL(1)
          call WRTSTR(0,(115+(len*9)),110,16,' already exists.')
          call WRTSTR(0,40,140,26,'Overwrite the file (y/n) ?')
          yesno = 'n'
98        continue
          key = STRIN(0,280,140,1,yesno)
          if (key.ne.retk) goto 98
          if ((yesno.eq.'y').or.(yesno.eq.'Y')) then
             close(funit)
             call RSTERR()
             open(funit,FILE=usenam,STATUS='OLD',IOSTAT=ferr)
             if (ferr.eq.0) then

                write(funit,105,iostat=chkio,err=223)
                write(funit,'(5g10.4)',iostat=chkio,err=223)
     +               (syst(i),i=1,5)
                dima = int(syst(5))
                mord = int(syst(4))
                write(funit,100,iostat=chkio,err=223)


                mname = ';A =                    '
                serr = savmat(dima,dima,a,mname,funit)
                if (serr.ne.0) goto 223
                mname = ';B =                    '
                serr = savmat(dima,mord,b,mname,funit)
                if (serr.ne.0) goto 223
                mname = ';C =                    '
                serr = savmat(mord,dima,c,mname,funit)
                if (serr.ne.0) goto 223
                mname = ';KCI =                  '
                serr = savmat(mord,dima,kci,mname,funit)
                if (serr.ne.0) goto 223
                mname = ';KFI =                  '
                serr = savmat(dima,mord,kfi,mname,funit)
                if (serr.ne.0) goto 223
                mname = ';Q1 =                   '
                serr = savmat(dima,dima,q1,mname,funit)
                if (serr.ne.0) goto 223
                mname = ';R1 =                   '
                serr = savmat(mord,mord,r1,mname,funit)
                if (serr.ne.0) goto 223
                mname = ';KC =                   '
                serr = savmat(mord,dima,kc,mname,funit)
                if (serr.ne.0) goto 223
                mname = ';Q =                    '
                serr = savmat(dima,dima,q,mname,funit)
                if (serr.ne.0) goto 223
                mname = ';R =                    '
                serr = savmat(mord,mord,r,mname,funit)
                if (serr.ne.0) goto 223
                mname = ';KF =                   '
                serr = savmat(dima,mord,kf,mname,funit)
                if (serr.ne.0) goto 223
                mname = ';Ptr =                  '
                serr = savmat(mord,mord,ptrsst,mname,funit)
                if (serr.ne.0) goto 223

                npts = 100
                nanal = 5
                nchan = 2

                grno = 1
                pernam = ';PERVAL: SENSOR NOISE REJ'
                serr = savper(grno,nanal,nchan,pival,pernam,funit)
                if (serr.ne.0) goto 223

                grno = 2
                pernam = ';PERVAL: DISTURBANCE REJ '
                serr = savper(grno,nanal,nchan,pival,pernam,funit)
                if (serr.ne.0) goto 223

                grno = 3
                pernam = ';PERVAL: SENSITIVITY     '
                serr = savper(grno,nanal,nchan,pival,pernam,funit)
```

```
                  if (serr.ne.0) goto 223

                  grno = 4
                  pernam = ';PERVAL: EFFORT & TRACK  '
                  serr = savper(grno,nanal,nchan,pival,pernam,funit)
                  if (serr.ne.0) goto 223
cccc
                  mname = ';STABVAL: GAIN & PHASE M '
                  serr = savmat(2,5,stval,mname,funit)
                  if (serr.ne.0) goto 223
cccc
                  write(funit,'(a)',iostat=chkio,err=223)
     +                 ';PERFORMANCE INDEX COMMENTS'
                  do 881 i=1,4
                      if (i.eq.1)
     +                    write(funit,'(a)',iostat=chkio,err=223)
     +                        ';Input(1) and Output(2) Sens to Noise'
                      if (i.eq.2)
     +                    write(funit,'(a)',iostat=chkio,err=223)
     +                    ';Input(1) and Output(2) Sens to Disturbances'
                      if (i.eq.3)
     +                    write(funit,'(a)',iostat=chkio,err=223)
     +                    ';Input(1) and Output(2) Sens to Model Change'
                      if (i.eq.4)
     +                    write(funit,'(a)',iostat=chkio,err=223)
     +                    ';Control Effort(1) and Setpoint Tracking(2)'
                      do 882 j=1,2
                          if (j.eq.1)
     +                        write(funit,'(a)',iostat=chkio,err=223)
     +                            ';Channel 1'
                          if (j.eq.2)
     +                        write(funit,'(a)',iostat=chkio,err=223)
     +                            ';Channel 2'
                          do 883 k = 1, 5
                              write(funit,'(a)',iostat=chkio,err=223)
     +                            picom(i,j,k)
883                       continue
882                   continue
881               continue
                  write(funit,100,iostat=chkio,err=223)

                  write(funit,'(a)',iostat=chkio,err=223)
     +                 ';GAIN AND PHASE MARGIN COMMENTS'
                  write(funit,'(a)',iostat=chkio,err=223)
     +                 (stcom(i),i=1,2)
                  write(funit,100,iostat=chkio,err=223)


                  grno = 1
                  pernam = ';PERDAT: SENSOR NOISE REJ'
                  serr = savper(grno,npts,nchan,pidat,pernam,funit)
                  if (serr.ne.0) goto 223
                  grno = 2
                  pernam = ';PERDAT: DISTURBANCE REJ '
                  serr = savper(grno,npts,nchan,pidat,pernam,funit)
                  if (serr.ne.0) goto 223
                  grno = 3
                  pernam = ';PERDAT: SENSITIVITY     '
                  serr = savper(grno,npts,nchan,pidat,pernam,funit)
                  if (serr.ne.0) goto 223
                  grno = 4
                  pernam = ';PERDAT: EFFORT & TRACK  '
                  serr = savper(grno,npts,nchan,pidat,pernam,funit)
                  if (serr.ne.0) goto 223


                  write(funit,100,iostat=chkio,err=223)
                  write(funit,'(a)',iostat=chkio,
     +                 err=223) '; Group of Data :'
                  write(funit,'(a)',iostat=chkio,
     +                 err=223) '; ***************'
                  write(funit,'(a)',iostat=chkio,err=223) infnam
                  if (chkio.eq.0) goto 221
223               call ERTONE()
221               continue

                  close(funit)
                  if (serr.eq.0) then
                     call WRTSTR(0,40,170,20,
     +                    'Data save completed.')
                  else
                     call WRTSTR(0,40,170,36,
     +                    '*** Error : Data save not completed.')
                  endif
               else
                  derr = GETERR()
                  if (derr.eq.-1) then
                     call ERTONE()
                     call WRTSTR(0,40,140,38,
     +                    '*** Error : Cannot overwrite old file.')
                     close(funit)
                  else
                     call prderr(0,40,170,derr)
                  endif
               endif
            else
               call WRTSTR(0,40,170,30,
     +              'Data save operation cancelled.')
            endif
         endif
      else
         call WRTSTR(0,40,170,30,
     +        'Data save operation cancelled.')
      endif
      close(funit)

      key = INKEY(1)

cC Blank over heading. (util.asm - iglib.lib)
c      call WRTSTR(0,(720-56*9)/2,35,56,
c     +   '                                                        ')

      return
      end
C *****************************************************************************
CH    REVISION HISTORY  :
```

```
C      VERSION       BY                  DATE        COMMENT
C        1.1      A. de Waal            06/01/90   Finally Commented
C      FILEND          :
C ***************************************************************************
C
C      FILE              : SAVLQG.FOR
C ***************************************************************************
CN     MODULE NAME       : savlqg
CA     FUNCTION          : Save LQG Design
CS     CALL SEQUENCE     : call savlqg()
CI     INPUT PARAMETERS  : None
CO     OUTPUT PARAMETERS : None
CG     GLOBAL VARIABLES  : Include files:  keys.inc,syst.inc,defnam.inc,
C                                          perdat.inc
CM     MODULES CALLED    : Fortran   :  wrhead,maknam,dodir,wrtitl,savmat,
C                                       prderr
C                          Assembler:  WIPSCR,WRTSTR,STRIN,RSTERR,ERTONE,INKEY,
C                                       GETERR
CE     ERROR CONDITIONS  : Disk Errors as Indicated on Screen
CC     COMMENTS          : User is interrogated to determine to which file
C                          data is to be saved, and the following data
C                          for a LQG control system is saved:
C                              - Process Data (Model and System Specifications)
C                              - LQG Weighting and Covariance Matrices
C                              - LQG Controller (Controller (kc) and
C                                                Observer (kf) Matrices)
C ***************************************************************************
      subroutine savlqg()
      implicit integer*2 (c,G,L,S)
$include: 'keys.inc'
$include: 'syst.inc'
$include: 'defnam.inc'
$include: 'perdat.inc'
      integer*2 dima,mord,npts,nchan,grno,nanal,nchan
      integer*2 funit
      integer*2 key,ferr,derr,serr,len,chkio
      character*1 yesno

100   format('; ')
105   format(';w0, wf, winc, mord, dima =')
      call WIPSCR(0)

cC Write heading. (wrhead.for - iqlib.lib)
c      call wrhead(0,(720-53*9)/2,35,53,
c     +   'Saving System with LQG Controller Design Information.')

C Information that would have been passed to savef subroutine
      infnam = ';LQG Design (Unanalysed) '
      fname = 'des0.lqg
      funit = 4

C Information that would have been in defnam.inc file
c     outpth = 'c:\dewaal\progs\optcad\ '
      defdir = '*.lqg'

C Information that would have been in syst.inc file

C Make a filename usenam(50) = outpth(25)+fname(25)
      call maknam(fname,outpth,usenam)


C Print out default directory page 0, yupperlim 180, length 4, defdir .lqg,
C                          outpth 'c:\dewaal\progs\optcad  '
      call dodir(0,180,5,defdir,outpth)


      call wrhead(0,40,60,27,'Saving Information to File.')
      call wrhead(0,40,85,14,'Save to file :')
      call WRTSTR(0,40,110,30,'Hit RETURN to accept filename.')
      call WRTSTR(0,40,125,16,'Hit ESC to exit.')
99    continue
      key = STRIN(0,175,85,50,usenam)
      if ((key.ne.retk).and.(key.ne.esck)) goto 99
      call WRTSTR(0,40,110,30,'                                   ')
      call WRTSTR(0,40,125,16,'                          ')
      if (key.eq.retk) then
         call RSTERR()
         open(funit,FILE=usenam,STATUS='OLD',IOSTAT=ferr)
         if (ferr.gt.0) then
            derr = GETERR()
            if (derr.eq.-1) then
               close(funit)
               call WRTSTR(0,40,110,19,'Opening new file : ')
               call WRTSTR(0,215,110,50,usenam)
               open(funit,FILE=usenam,STATUS='NEW',IOSTAT=ferr)
               if (ferr.eq.0) then

                  write(funit,105,iostat=chkio,err=203)
                  write(funit,'(5g10.4)',iostat=chkio,err=203)
     +                 (syst(i),i=1,5)
                  dima = int(syst(5))
                  mord = int(syst(4))
                  write(funit,100,iostat=chkio,err=203)

                  mname = ';A =                          '
                  serr = savmat(dima,dima,a,mname,funit)
                  if (serr.ne.0) goto 203
                  mname = ';B =                          '
                  serr = savmat(dima,mord,b,mname,funit)
                  if (serr.ne.0) goto 203
                  mname = ';C =                          '
                  serr = savmat(mord,dima,c,mname,funit)
                  if (serr.ne.0) goto 203
                  mname = ';KCI =                        '
                  serr = savmat(mord,dima,kci,mname,funit)
                  if (serr.ne.0) goto 203
                  mname = ';KFI =                        '
                  serr = savmat(dima,mord,kfi,mname,funit)
                  if (serr.ne.0) goto 203
                  mname = ';Q1 =                         '
                  serr = savmat(dima,dima,q1,mname,funit)
                  if (serr.ne.0) goto 203
                  mname = ';R1 =                         '
                  serr = savmat(mord,mord,r1,mname,funit)
                  if (serr.ne.0) goto 203
                  mname = ';KC =                         '
```

```
                serr = savmat(mord,dima,kc,mname,funit)
                if (serr.ne.0) goto 203
                mname = ';Q =                    ,
                serr = savmat(dima,dima,q,mname,funit)
                if (serr.ne.0) goto 203
                mname = ';R =                    ,
                serr = savmat(mord,mord,r,mname,funit)
                if (serr.ne.0) goto 203
                mname = ';KF =                   ,
                serr = savmat(dima,mord,kf,mname,funit)
                if (serr.ne.0) goto 203
                mname = ';Ptr =                  ,
                serr = savmat(mord,mord,ptrsst,mname,funit)
                if (serr.ne.0) goto 203

                write(funit,100,iostat=chkio,err=203)
                write(funit,'(a)',iostat=chkio,
     +               err=203) '; Group of Data :'
                write(funit,'(a)',iostat=chkio,
     +               err=203) '; ***************'
                write(funit,'(a)',iostat=chkio,err=203) infnam
                if (chkio.eq.0) goto 201
203             call ERTONE()
201             continue

                close(funit)
                if (serr.eq.0) then
                   call WRTSTR(0,40,130,20,'Data save completed.')
                else
                   call WRTSTR(0,40,130,36,
     +                 '*** Error : Data save not completed.')
                   call WRTSTR(0,40,145,26,
     +                    'Error ocurred while saving')
                   call WRTSTR(0,40,160,25,mname)
                endif
             else
                call ERTONE()
                call WRTSTR(0,40,130,33,
     +               '*** Error : Cannot open new file.')
                close(funit)
             endif
          else
             call prderr(0,40,130,derr)
          endif
       else
          len = LENSTR(50,usenam)
          call ERTONE()
          call WRTSTR(0,40,110,6,'File :')
          call LEVEL(2)
          call BLKFIL(115,113,(len*9),14)
          call WRTSTR(0,115,110,len,usenam)
          call LEVEL(1)
          call WRTSTR(0,(115+(len*9)),110,16,' already exists.')
          call WRTSTR(0,40,140,26,'Overwrite the file (y/n) ?')
          yesno = 'n'
98        continue
          key = STRIN(0,280,140,1,yesno)
          if (key.ne.retk) goto 98
          if ((yesno.eq.'y').or.(yesno.eq.'Y')) then
             close(funit)
             call RSTERR()
             open(funit,FILE=usenam,STATUS='OLD',IOSTAT=ferr)
             if (ferr.eq.0) then

                write(funit,105,iostat=chkio,err=223)
                write(funit,'(5g10.4)',iostat=chkio,err=223)
     +               (syst(i),i=1,5)
                dima = int(syst(5))
                mord = int(syst(4))
                write(funit,100,iostat=chkio,err=223)


                mname = ';A =                    ,
                serr = savmat(dima,dima,a,mname,funit)
                if (serr.ne.0) goto 223
                mname = ';B =                    ,
                serr = savmat(dima,mord,b,mname,funit)
                if (serr.ne.0) goto 223
                mname = ';C =                    ,
                serr = savmat(mord,dima,c,mname,funit)
                if (serr.ne.0) goto 223
                mname = ';KCI =                  ,
                serr = savmat(mord,dima,kci,mname,funit)
                if (serr.ne.0) goto 223
                mname = ';KFI =                  ,
                serr = savmat(dima,mord,kfi,mname,funit)
                if (serr.ne.0) goto 223
                mname = ';Q1 =                   ,
                serr = savmat(dima,dima,q1,mname,funit)
                if (serr.ne.0) goto 223
                mname = ';R1 =                   ,
                serr = savmat(mord,mord,r1,mname,funit)
                if (serr.ne.0) goto 223
                mname = ';KC =                   ,
                serr = savmat(mord,dima,kc,mname,funit)
                if (serr.ne.0) goto 223
                mname = ';Q =                    ,
                serr = savmat(dima,dima,q,mname,funit)
                if (serr.ne.0) goto 223
                mname = ';R =                    ,
                serr = savmat(mord,mord,r,mname,funit)
                if (serr.ne.0) goto 223
                mname = ';KF =                   ,
                serr = savmat(dima,mord,kf,mname,funit)
                if (serr.ne.0) goto 223
                mname = ';Ptr =                  ,
                serr = savmat(mord,mord,ptrsst,mname,funit)
                if (serr.ne.0) goto 223


                write(funit,100,iostat=chkio,err=223)
                write(funit,'(a)',iostat=chkio,
     +               err=223) '; Group of Data :'
                write(funit,'(a)',iostat=chkio,
     +               err=223) '; ***************'
                write(funit,'(a)',iostat=chkio,err=223) infnam
```

```
CO     OUTPUT PARAMETERS: error - (integer*2) The error return code.
CG     GLOBAL VARIABLES : None.
CM     MODULES CALLED   : Assembler:   ERTONE
CE     ERROR CONDITIONS : As indicated by error return code
CC     COMMENTS         : Stores the matrix  information on the unit number
C                         passed from calling routine. If any errors occurs,
C                         then an error tone is sounded and the code passed
C                         back to the calling routine.
C ***************************************************************************
       integer*2 function savper(grno,rows,cols,mat,pernam,funit)
       integer*2 grno,rows,cols
       real mat(4,2,100)
       character*25 pernam
       integer*2 funit, chkio
100    format('; ')

       write(funit,'(a)',iostat=chkio,err=200) pernam
       write(funit,'(2i6)',iostat=chkio,err=200) rows,cols
       if (chkio.eq.0) then
          do 997 i = 1,rows
               write(funit,'(2g10.4)',iostat=chkio,err=200)
     +              (mat(grno,j,i), j=1,cols)
997       continue
       endif
       write (funit,100,iostat=chkio,err=200)

       if (chkio.eq.0) goto 201
200    call ERTONE()
201    continue
       savper = chkio
       return
       end
C ***************************************************************************
CH     REVISION HISTORY  :
C      VERSION        BY                  DATE         COMMENT
C        1.1      A. de Waal            06/01/90    Finally Commented
C      FILEND           :
C ***************************************************************************
C
C      FILE             : SAVPRO.FOR
C ***************************************************************************
CN     MODULE NAME      : savpro
CA     FUNCTION         : Save Process Data from Disk
CS     CALL SEQUENCE    : call savpro()
CI     INPUT PARAMETERS : None
CO     OUTPUT PARAMETERS : None
CG     GLOBAL VARIABLES : Include files:  keys.inc,syst.inc,defnam.inc,
C                                         perdat.inc
CM     MODULES CALLED   : Fortran  :  wrhead,maknam,dodir,wrtitl,savmat,
C                                     prderr
C                         Assembler:  WIPSCR,WRTSTR,STRIN,RSTERR,ERTONE,INKEY,
C                                     GETERR
CE     ERROR CONDITIONS : Disk Errors as Indicated on Screen
CC     COMMENTS         : User is interrogated to determine to which file
C                         data is to be saved, and the following data
C                         is saved:
C                             - Process Data (Model and System Specifications)
C ***************************************************************************
       subroutine savpro()
       implicit integer*2 (c,G,L,S)
$include: 'keys.inc'
$include: 'syst.inc'
$include: 'defnam.inc'
$include: 'perdat.inc'
       integer*2 dima,mord,npts,nchan,grno,nanal,nchan
       integer*2 funit
       integer*2 key,ferr,derr,serr,len,chkio
       character*1 yesno

100    format('; ')
105    format(';w0, wf, winc, mord, dima =')

       call WIPSCR(0)
cC Write heading. (wrhead.for - iglib.lib)
c       call wrhead(0,(720-39*9)/2,35,39,
c      +   'Saving State Space Process Description.')



C Information that would have been passed to savef subroutine
       infnam = ';State Space Process Desc'
       fname = 'des0.pro                '
       funit = 4

C Information that would have been in defnam.inc file
c      outpth = 'c:\dewaal\progs\optcad\ '
       defdir = '*.pro'

C Information that would have been in syst.inc file

C Make a filename usenam(50) = outpth(25)+fname(25)
       call maknam(fname,outpth,usenam)


C Print out default directory page 0, yupperlim 180, length 4, defdir .lqa,
C                               outpth 'c:\dewaal\progs\optcad  '
       call dodir(0,180,5,defdir,outpth)


       call wrhead(0,40,60,27,'Saving Information to File.')
       call wrhead(0,40,85,14,'Save to file :')
       call WRTSTR(0,40,110,30,'Hit RETURN to accept filename.')
       call WRTSTR(0,40,125,16,'Hit ESC to exit.')
99     continue
       key = STRIN(0,175,85,50,usenam)
       if ((key.ne.retk).and.(key.ne.esck)) goto 99
       call WRTSTR(0,40,110,30,'                                  ')
       call WRTSTR(0,40,125,16,'                  ')
       if (key.eq.retk) then
          call RSTERR()
          open(funit,FILE=usenam,STATUS='OLD',IOSTAT=ferr)
          if (ferr.gt.0) then
             derr = GETERR()
             if (derr.eq.-1) then
                close(funit)
```

```
            call WRTSTR(0,40,110,19,'Opening new file : ')
            call WRTSTR(0,215,110,50,usenam)
            open(funit,FILE=usenam,STATUS='NEW',IOSTAT=ferr)
            if (ferr.eq.0) then

               write(funit,105,iostat=chkio,err=203)
               write(funit,'(5g10.4)',iostat=chkio,err=203)
     +              (syst(i),i=1,5)
               dima = int(syst(5))
               mord = int(syst(4))
               write(funit,100,iostat=chkio,err=203)

               mname = ';A =                          ,
               serr = savmat(dima,dima,a,mname,funit)
               if (serr.ne.0) goto 203
               mname = ';B =                          ,
               serr = savmat(dima,mord,b,mname,funit)
               if (serr.ne.0) goto 203
               mname = ';C =                          ,
               serr = savmat(mord,dima,c,mname,funit)
               if (serr.ne.0) goto 203
               mname = ';KCI =                        ,
               serr = savmat(mord,dima,kci,mname,funit)
               if (serr.ne.0) goto 203
               mname = ';KFI =                        ,
               serr = savmat(dima,mord,kfi,mname,funit)
               if (serr.ne.0) goto 203
               mname = ';Ptr =                        ,
               serr = savmat(mord,mord,ptrsst,mname,funit)
               if (serr.ne.0) goto 203

               write(funit,100,iostat=chkio,err=203)
               write(funit,'(a)',iostat=chkio,
     +              err=203) '; Group of Data :'
               write(funit,'(a)',iostat=chkio,
     +              err=203) '; ***************'
               write(funit,'(a)',iostat=chkio,err=203) infnam
               if (chkio.eq.0) goto 201
203            call ERTONE()
201            continue

               close(funit)
               if (serr.eq.0) then
                  call WRTSTR(0,40,130,20,'Data save completed.')
               else
                  call WRTSTR(0,40,130,36,
     +                 '*** Error : Data save not completed.')
                  call WRTSTR(0,40,145,26,
     +                    'Error ocurred while saving')
                  call WRTSTR(0,40,160,25,mname)
               endif
            else
               call ERTONE()
               call WRTSTR(0,40,130,33,
     +              '*** Error : Cannot open new file.')
               close(funit)
            endif
         else
            call prderr(0,40,130,derr)
         endif
      else
         len = LENSTR(50,usenam)
         call ERTONE()
         call WRTSTR(0,40,110,6,'File :')
         call LEVEL(2)
         call BLKFIL(115,113,(len*9),14)
         call WRTSTR(0,115,110,len,usenam)
         call LEVEL(1)
         call WRTSTR(0,(115+(len*9)),110,16,' already exists.')
         call WRTSTR(0,40,140,26,'Overwrite the file (y/n) ?')
         yesno = 'n'
98       continue
         key = STRIN(0,280,140,1,yesno)
         if (key.ne.retk) goto 98
         if ((yesno.eq.'y').or.(yesno.eq.'Y')) then
            close(funit)
            call RSTERR()
            open(funit,FILE=usenam,STATUS='OLD',IOSTAT=ferr)
            if (ferr.eq.0) then

               write(funit,105,iostat=chkio,err=223)
               write(funit,'(5g10.4)',iostat=chkio,err=223)
     +              (syst(i),i=1,5)
               dima = int(syst(5))
               mord = int(syst(4))
               write(funit,100,iostat=chkio,err=223)


               mname = ';A =                          ,
               serr = savmat(dima,dima,a,mname,funit)
               if (serr.ne.0) goto 223
               mname = ';B =                          ,
               serr = savmat(dima,mord,b,mname,funit)
               if (serr.ne.0) goto 223
               mname = ';C =                          ,
               serr = savmat(mord,dima,c,mname,funit)
               if (serr.ne.0) goto 223
               mname = ';KCI =                        ,
               serr = savmat(mord,dima,kci,mname,funit)
               if (serr.ne.0) goto 223
               mname = ';KFI =                        ,
               serr = savmat(dima,mord,kfi,mname,funit)
               if (serr.ne.0) goto 223
               mname = ';Ptr =                        ,
               serr = savmat(mord,mord,ptrsst,mname,funit)
               if (serr.ne.0) goto 223

               write(funit,100,iostat=chkio,err=223)
               write(funit,'(a)',iostat=chkio,
     +                err=223) '; Group of Data :'
               write(funit,'(a)',iostat=chkio,
     +                err=223) '; ***************'
               write(funit,'(a)',iostat=chkio,err=223) infnam
               if (chkio.eq.0) goto 221
223            call ERTONE()
221            continue
```

```
                  close(funit)
                  if (serr.eq.0) then
                     call WRTSTR(0,40,170,20,
     +                    'Data save completed.')
                  else
                     call WRTSTR(0,40,170,36,
     +                    '*** Error : Data save not completed.')
                  endif
               else
                  derr = GETERR()
                  if (derr.eq.-1) then
                     call ERTONE()
                     call WRTSTR(0,40,140,38,
     +                    '*** Error : Cannot overwrite old file.')
                     close(funit)
                  else
                     call prderr(0,40,170,derr)
                  endif
               endif
            else
               call WRTSTR(0,40,170,30,
     +                 'Data save operation cancelled.')
            endif
         endif
      else
         call WRTSTR(0,40,170,30,
     +             'Data save operation cancelled.')
      endif
      close(funit)

      key = INKEY(1)

cC Blank over heading. (util.asm - iglib.lib)
c      call WRTSTR(0,(720-39*9)/2,35,39,
c     +   '                                       ')

      return
      end
C *****************************************************************************
CH    REVISION HISTORY  :
C     VERSION      BY                 DATE        COMMENT
C       1.1      A. de Waal          06/01/90   Finally Commented
C     FILEND            :
C *****************************************************************************
C
C     FILE              : SAVQR.FOR
C *****************************************************************************
CN    MODULE NAME       : savlqa
CA    FUNCTION          : Save LQG Weighting and Covariance Matrices
CS    CALL SEQUENCE     : call savqr()
CI    INPUT PARAMETERS  : None
CO    OUTPUT PARAMETERS : None
CG    GLOBAL VARIABLES  : Include files:  keys.inc,syst.inc,defnam.inc,
C                                         perdat.inc
CM    MODULES CALLED    : Fortran  :  wrhead,maknam,dodir,wrtitl,savmat,
C                                     prderr
C                         Assembler:  WIPSCR,WRTSTR,STRIN,RSTERR,ERTONE,INKEY,
C                                     GETERR
CE    ERROR CONDITIONS  : Disk Errors as Indicated on Screen
CC    COMMENTS          : User is interrogated to determine to which file
C                         data is to be saved, and the following data
C                         for a LQG control system is saved:
C                             - LQG Weighting and Covariance Matrices
C *****************************************************************************
      subroutine savqr()
      implicit integer*2 (c,G,L,S)
$include: 'keys.inc'
$include: 'syst.inc'
$include: 'defnam.inc'
$include: 'perdat.inc'
      integer*2 dima,mord,npts,nchan,grno,nanal,nchan
      integer*2 funit
      integer*2 key,ferr,derr,serr,len,chkio
      character*1 yesno

100   format('; ')
105   format(';w0, wf, winc, mord, dima =')

      call WIPSCR(0)

cC Write heading. (wrhead.for - iglib.lib)
c      call wrhead(0,(720-55*9)/2,35,55,
c     +   'Saving Control Weighting and Noise Covariance Matrices.')


C Information that would have been passed to savef subroutine
      infnam = ';Weighting & Covar Matric'
      fname = 'des0.qr                '
      funit = 4

C Information that would have been in defnam.inc file
c     outpth = 'c:\dewaal\progs\optcad\ '
      defdir = '*.qr'

C Information that would have been in syst.inc file

C Make a filename usenam(50) = outpth(25)+fname(25)
      call maknam(fname,outpth,usenam)


C Print out default directory page 0, yupperlim 180, length 4, defdir .qr ,
C                             outpth 'c:\dewaal\progs\optcad  '
      call dodir(0,180,4,defdir,outpth)


      call wrhead(0,40,60,27,'Saving Information to File.')
      call wrhead(0,40,85,14,'Save to file :')
      call WRTSTR(0,40,110,30,'Hit RETURN to accept filename.')
      call WRTSTR(0,40,125,16,'Hit ESC to exit.')
99    continue
      key = STRIN(0,175,85,50,usenam)
      if ((key.ne.retk).and.(key.ne.esck)) goto 99
      call WRTSTR(0,40,110,30,'                              ')
      call WRTSTR(0,40,125,16,'                  ')
```

```
      if (key.eq.retk) then
         call RSTERR()
         open(funit,FILE=usenam,STATUS='OLD',IOSTAT=ferr)
         if (ferr.gt.0) then
            derr = GETERR()
            if (derr.eq.-1) then
               close(funit)
               call WRTSTR(0,40,110,19,'Opening new file : ')
               call WRTSTR(0,215,110,50,usenam)
               open(funit,FILE=usenam,STATUS='NEW',IOSTAT=ferr)
               if (ferr.eq.0) then

                  mname = ';Q1 =                    ,
                  serr = savmat(dima,dima,q1,mname,funit)
                  if (serr.ne.0) goto 203
                  mname = ';R1 =                    ,
                  serr = savmat(mord,mord,r1,mname,funit)
                  if (serr.ne.0) goto 203
                  mname = ';Q =                     ,
                  serr = savmat(dima,dima,q,mname,funit)
                  if (serr.ne.0) goto 203
                  mname = ';R =                     ,


                  write(funit,100,iostat=chkio,err=203)
                  write(funit,'(a)',iostat=chkio,
     +                 err=203) '; Group of Data :'
                  write(funit,'(a)',iostat=chkio,
     +                 err=203) '; ***************'
                  write(funit,'(a)',iostat=chkio,err=203) infnam
                  if (chkio.eq.0) goto 201
203               call ERTONE()
201               continue

                  close(funit)
                  if (serr.eq.0) then
                     call WRTSTR(0,40,130,20,'Data save completed.')
                  else
                     call WRTSTR(0,40,130,36,
     +                    '*** Error : Data save not completed.')
                     call WRTSTR(0,40,145,26,
     +                       'Error ocurred while saving')
                     call WRTSTR(0,40,160,25,mname)
                  endif
               else
                  call ERTONE()
                  call WRTSTR(0,40,130,33,
     +                 '*** Error : Cannot open new file.')
                  close(funit)
               endif
            else
               call prderr(0,40,130,derr)
            endif
         else
            len = LENSTR(50,usenam)
            call ERTONE()
            call WRTSTR(0,40,110,6,'File :')
            call LEVEL(2)
            call BLKFIL(115,113,(len*9),14)
            call WRTSTR(0,115,110,len,usenam)
            call LEVEL(1)
            call WRTSTR(0,(115+(len*9)),110,16,' already exists.')
            call WRTSTR(0,40,140,26,'Overwrite the file (y/n) ?')
            yesno = 'n'
98          continue
            key = STRIN(0,280,140,1,yesno)
            if (key.ne.retk) goto 98
            if ((yesno.eq.'y').or.(yesno.eq.'Y')) then
               close(funit)
               call RSTERR()
               open(funit,FILE=usenam,STATUS='OLD',IOSTAT=ferr)
               if (ferr.eq.0) then

                  mname = ';Q1 =                    ,
                  serr = savmat(dima,dima,q1,mname,funit)
                  if (serr.ne.0) goto 223
                  mname = ';R1 =                    ,
                  serr = savmat(mord,dima,kc,mname,funit)
                  if (serr.ne.0) goto 223
                  mname = ';Q =                     ,
                  serr = savmat(dima,dima,q,mname,funit)
                  if (serr.ne.0) goto 223
                  mname = ';R =                     ,
                  serr = savmat(mord,mord,r,mname,funit)
                  if (serr.ne.0) goto 223

                  write(funit,100,iostat=chkio,err=223)
                  write(funit,'(a)',iostat=chkio,
     +                 err=223) '; Group of Data :'
                  write(funit,'(a)',iostat=chkio,
     +                 err=223) '; ***************'
                  write(funit,'(a)',iostat=chkio,err=223) infnam
                  if (chkio.eq.0) goto 221
223               call ERTONE()
221               continue

                  close(funit)
                  if (serr.eq.0) then
                     call WRTSTR(0,40,170,20,
     +                    'Data save completed.')
                  else
                     call WRTSTR(0,40,170,36,
     +                    '*** Error : Data save not completed.')
                  endif
               else
                  derr = GETERR()
                  if (derr.eq.-1) then
                     call ERTONE()
                     call WRTSTR(0,40,140,38,
     +                    '*** Error : Cannot overwrite old file.')
                     close(funit)
                  else
                     call prderr(0,40,170,derr)
                  endif
               endif
            else
```

```
            call WRTSTR(0,40,170,30,
     +              'Data save operation cancelled.')
         endif
      endif
      else
         call WRTSTR(0,40,170,30,
     +              'Data save operation cancelled.')
      endif
      close(funit)

      key = INKEY(1)

cC Blank over heading. (util.asm - iglib.lib)
c       call WRTSTR(0,(720-55*9)/2,35,55,
c     + '                                                       ')

      return
      end
C ***************************************************************************
CH    REVISION HISTORY   :
C     VERSION        BY                DATE        COMMENT
C       1.1      A. de Waal           06/01/90    Finally Commented
C     FILEND             :
C ***************************************************************************
C
C     FILE               : WRCMAT.FOR
C ***************************************************************************
CN    MODULE NAME        : wrcmat
CA    FUNCTION           : Write Complex Matrix to File
CS    CALL SEQUENCE      : call wrcmat(rows,cols,matr,mati,funit,mname)
CI    INPUT PARAMETERS   : rows,cols:   integer - Actual dimensions of matrix
C                          matr(15,15):    real - Real part of matrix of
C                                                 dimension (15,15) to be written
C                          mati(15,15):    real - Imaginary part of matrix of
C                                                 dimension (15,15) to be written
C                          funit:       integer - File unit specifier
C                          mname:     character - Name of matrix
CO    OUTPUT PARAMETERS  : None
CG    GLOBAL VARIABLES   : None
CM    MODULES CALLED     : None
CE    ERROR CONDITIONS   : Disk error - no timeout
CC    COMMENTS           : Writes complex matrix (matr,mati) to disk file
C                          indicated by file unit specifief (funit)
C ***************************************************************************
      subroutine wrcmat(rows,cols,matr,mati,funit,mname)
      integer rows,cols,funit
      real matr(15,15),mati(15,15)
      character*10 mname
      write(funit,'(a)') mname
      write(funit,'(a)') ' Real:'
      do 999 i = 1, rows
          write(funit,'(3X,15g10.4)') (matr(i,j),j = 1,cols)
999   continue
      write(funit,'(a)') ' Imag:'
      do 899 i = 1, rows
          write(funit,'(3X,15g10.4)') (mati(i,j),j = 1,cols)
899   continue
      return
      end

C ***************************************************************************
CH    REVISION HISTORY   :
C     VERSION        BY                DATE        COMMENT
C       1.1      A. de Waal           06/01/90    Finally Commented
C     FILEND             :
C ***************************************************************************
→
```

M3) OPTCAD Mathematics Utility Subroutines

A table of the mathematical subroutine names follows on the next page. The name of each subroutine, together with the file in which it is located and the page in the appendix on which it is listed, is given in the table.

OPTCAD Mathematics Utility Subroutines			
Name	**File**	**Description**	**Page**
add	ADD.FOR	Adds Two Matrices	426
cinv	CINV.FOR	Invert a Complex Matrix	426
cmult	CMULT.FOR	Multiply two complex matrices	427
invert	INVERT.FOR	Invert a Real Matrix	427
mult	MULT.FOR	Multiply Two Matrices	428
rndnrm	RNDNRM.FOR	Generate Random Numbers with Gaussian PDF	428
simeq	SIMEQ.FOR	Solve System of Simultaneous Equations	429
svd	SVD.FOR	Singular Value Decomposition	429
svmax	SVMAX.FOR	Obtain Maximum Singular Value of Matrix	430
svmin	SVMIN.FOR	Obtain Minimum Singular Value of Matrix	431
trans	TRANS.FOR	Return the Transpose of a Matrix	431
urand	URAND.FOR	Uniform Random Number Generator	432

```
C
C     FILE              : ADD.FOR
C ***************************************************************************
CN    MODULE NAME       : add
CA    FUNCTION          : Adds Two Matrices
CS    CALL SEQUENCE     : CALL ADD(A,B,C,I,J,K,L,IFL)
CI    INPUT PARAMETERS  : A:MATRIX OF DIMENSION (I,J)
C                                B:MATRIX OF DIMENSION (K,L)
C                                I,J,K,L:DIMENSIONS OF INPUT MATRICES
C                                IFL:FLAG TO INDICATE ADDITION OR SUBTRACTION
CO    OUTPUT PARAMETERS : C:MATRIX OF DIMENSION(I,L)
C                         ERR: ERR = 0 correct dimensions for marix mult
C                         ERR = 1 incorrect dimensions for matrix mult
CG    GLOBAL VARIABLES  : NONE
CM    MODULES CALLED    : NONE
CE    ERROR CONDITIONS  : ERROR IF DIMENSIONS DO NOT MATCH.
CC    COMMENTS          : THIS ROUTINE ADDS OR SUBTRACTS TWO MATRICES. IF THE
C                                FLAG IFL=0, C=A+B IS CALCULATED; IF IFL=1, C=A-B.
C                         THE MATRICES ARE ALL DIMENSIONED BY THE PARAMETER ID.
C                               THIS IS MERELY TO CONFORM TO THE SWAT DEBUGGER, WHICH
C                               DOES NOT LIKE ARRAYS WITH VARIABLE DIMENSIONS.
C ***************************************************************************
      SUBROUTINE ADD(A,B,C,I,J,K,L,IFL,ERR)
      INTEGER ERR
      PARAMETER (ID=15)
      DIMENSION A(ID,ID),B(ID,ID),C(ID,ID)
C
C     IF IFL=0 THE ROUTINE CALCULATES C=A+B.
C     IF IFL=1 THE ROUTINE CALCULATES C=A-B
C
      IF ((I.NE.K).AND.(J.NE.L)) THEN
        ERR = 1
        RETURN
      END IF
      IF (IFL.EQ.0) THEN
              DO 20 I1=1,I
                      DO 10 I2=1,J
                              C(I1,I2)=A(I1,I2)+B(I1,I2)
 10                   CONTINUE
 20           CONTINUE
      ELSE
              DO 40 I1=1,I
                      DO 30 I2=1,J
                              C(I1,I2)=A(I1,I2)-B(I1,I2)
 30                   CONTINUE
 40           CONTINUE
      END IF
      RETURN
      END
C ***************************************************************************
CH    REVISION HISTORY  :
C     VERSION       BY                  DATE          COMMENT
C       1.00      R.HENNING               7/4/87
C       1.10      A.DE WAAL               5/1/90        REVISED
C     FILEND            :
C ***************************************************************************
C
C     FILE              : CINV.FOR
C ***************************************************************************
CN    MODULE NAME       : cinv
CA    FUNCTION          : Invert a Complex Matrix
CS    CALL SEQUENCE     : call cinv(ar,ai,br,bi,n,max,max2)
CI    INPUT PARAMETERS  : ar: Real part of matrix of dimension (max,max)
C                         ai: Imaginary part of matrix of dimension (max,max)
C                         n : Order of matrix (n less than or equal to max)
C                         max:    Dimension of matrix
C                         max2:   2xDimension of matrix
CO    OUTPUT PARAMETERS : br: Real part of matrix of dimension (max,max)
C                         bi: Imaginary part of matrix of dimension (max,max)
CG    GLOBAL VARIABLES  : None
CM    MODULES CALLED    : None
CE    ERROR CONDITIONS  : None
CC    COMMENTS          : Inverts complex matrix (ar,ai) of dimension max and
C                         order n and returns result in (br,bi)
C ***************************************************************************
c +++ Program to invert a complex matrix.
c +++ -----------------------------------
      subroutine cinv(ar,ai,br,bi,n,max,max2)
      parameter (id=15,id2=id*2)
      real ar(max,max),ai(max,max)
      real br(max,max),bi(max,max)
      real gr(id,id2),gi(id,id2)
      complex piv,gipj,tarco,gitj

      do 98 i=1,n
         do 97 j=1,n
            if (i.eq.j) then
               br(i,j) = 1.0
               bi(i,j) = 0.0
            else
               br(i,j) = 0.0
               bi(i,j) = 0.0
            endif
97       continue
98    continue
      do 10 i=1,n
         do 20 j=1,n
            gr(i,j) = ar(i,j)
            gi(i,j) = ai(i,j)
20       continue
10    continue

      do 30 i=1,n
         do 35 j=1,n
            gr(i,n+j) = br(i,j)
            gi(i,n+j) = bi(i,j)
35       continue
30    continue

      do 40 ip=1,n
         piv = cmplx(gr(ip,ip),gi(ip,ip))
         do 50 j=1,2*n
            gipj = cmplx(gr(ip,j),gi(ip,j))
            gipj = gipj/piv
            gr(ip,j) = real(gipj)
            gi(ip,j) = imag(gipj)
```

```
50       continue
         do 60 it=1,n
            if (it.ne.ip) then
               tarco = cmplx(gr(it,ip),gi(it,ip))
               do 70 j=i,2*n
                  gitj = cmplx(gr(it,j),gi(it,j))
                  gipj = cmplx(gr(ip,j),gi(ip,j))
                  gitj = gitj - gipj*tarco
                  gr(it,j) = real(gitj)
                  gi(it,j) = imag(gitj)
70             continue
            endif
60       continue
40     continue
       DO 100 I=1,N
            DO 110 J=1,N
                 Br(I,J)=Gr(I,N+J)
                 Bi(I,J)=Gi(I,N+J)
110         CONTINUE
100    CONTINUE
       return
       end
C ***************************************************************************
CH     REVISION HISTORY  :
C      VERSION       BY                    DATE         COMMENT
C        1.1      A. de Waal           06/01/90  Finally Commented
C      FILEND            :
C ***************************************************************************
C
C      FILE              : CMULT.FOR
C ***************************************************************************
CN     MODULE NAME       : cmult
CA     FUNCTION          : Multiply two complex matrices
CS     CALL SEQUENCE     : call cmult(ar,ai,br,bi,cr,ci,i,j,k,l,arr)
CI     INPUT PARAMETERS  :         ar: Real part of matrix of dimension (i,j)
C                                      ai: Imaginary part of matrix of dimension (i,j)
C                                      ai: Real part of matrix of dimension (k,l)
C                                      ai: Imaginary part of matrix of dimension (k,l)
C                          i,j,k,l:    Matrix dimensions
CO     OUTPUT PARAMETERS :          cr: Real part of matrix (cr,ci) = (ar,ai)*(br,bi)
C                             of dimension (i,l)
C                          ci: Imaginary part of matrix
C                             (cr,ci) = (ar,ai)*(br,bi) of dimension (i,l)
C                          err: err = 0 correct dimensions for marix mult
C                               err = 1 incorrect dimensions for matrix mult
CG     GLOBAL VARIABLES  : None
CM     MODULES CALLED    : None
CE     ERROR CONDITIONS  :          stops if j.ne.k ie. if dimensions do not match
CC     COMMENTS          : Multiplies complex matrix (ar,ai) by complex matrix
C                          (br,bi) and returns result in complex matrix (cr,ci)
C ***************************************************************************
       SUBROUTINE CMULT(AR,AI,BR,BI,CR,CI,I,J,K,L,ERR)
       INTEGER ERR
         PARAMETER (ID=15)
       COMPLEX AT,BT,CT
         DIMENSION AR(ID,ID),BR(ID,ID),CR(ID,ID)
         DIMENSION AI(ID,ID),BI(ID,ID),CI(ID,ID)
100    format(A30)
       ERR = 0
         IF (J.NE.K) THEN
           ERR = 1
           RETURN
         END IF
         DO 20 I1=1,I
                 DO 10 I2=1,L
                         CR(I1,I2)=0.
                         CI(I1,I2)=0.
10               CONTINUE
20       CONTINUE
         DO 50 I1=1,I
                 DO 40 I2=1,L
                          DO 30 I3=1,J
                   AT = CMPLX(AR(I1,I3),AI(I1,I3))
                   BT = CMPLX(BR(I3,I2),BI(I3,I2))
                   CT = CMPLX(CR(I1,I2),CI(I1,I2))
                   CT = CT + AT*BT
C                                  C(I1,I2)=C(I1,I2)+A(I1,I3)*B(I3,I2)
                   CR(I1,I2) = REAL(CT)
                   CI(I1,I2) = IMAG(CT)
30                   CONTINUE
40               CONTINUE
50       CONTINUE
         RETURN
         END
C ***************************************************************************
CH     REVISION HISTORY  :
C      VERSION       BY                    DATE         COMMENT
C         0.00      R.HENNING             07/04/87
C        1.1      A. de Waal           06/01/90  Extended to complex and finally
C                                                    commented
C      FILEND            :
C ***************************************************************************
C
C      FILE              : INVERT.FOR
C ***************************************************************************
CN     MODULE NAME       : invert
CA     FUNCTION          : Invert a Real Matrix
CS     CALL SEQUENCE     : call invert(a,b,n,max,max2)
CI     INPUT PARAMETERS  : a(max,max): Matrix of maximum dimension (max,max)
C                          n:  Actual dimension of matrix
C                          max:    Maximum dimension of matrix
C                          max2:   Unused
CO     OUTPUT PARAMETERS : b(max,max): Matrix of maximum dimension (max,max)
CG     GLOBAL VARIABLES  : None
CM     MODULES CALLED    : None
CE     ERROR CONDITIONS  : None
CC     COMMENTS          : Inverts matrix a of maximum dimension (max,max) and
C                          actual dimension (n,n) and returns the result in an
C                          identically dimensioned matrix b
C ***************************************************************************
       subroutine INVERT(a,b,n,max,max2)
       parameter (id=15,id2=id*2)
       real a(max,max)
       real b(max,max)
       real q(id,id2)
```

```
      do 98 i=1,n
         do 97 j=1,n
            if (i.eq.j) then
               b(i,j) = 1.0
            else
               b(i,j) = 0.0
            endif
97       continue
98    continue
      do 10 i=1,n
         do 20 j=1,n
            g(i,j) = a(i,j)
20       continue
10    continue

      do 30 i=1,n
         do 35 j=1,n
            g(i,n+j) = b(i,j)
35       continue
30    continue

      do 40 ip=1,n
         piv = g(ip,ip)
         do 50 j=1,2*n
            g(ip,j) = g(ip,j)/piv
50       continue
         do 60 it=1,n
            if (it.ne.ip) then
               tarco = g(it,ip)
               do 70 j=1,2*n
                  g(it,j) = g(it,j)-g(ip,j)*tarco
70             continue
            endif
60       continue
40    continue
      DO 100 I=1,N
            DO 110 J=1,N
                  B(I,J)=G(I,N+J)
110         CONTINUE
100   CONTINUE
      return
      end
C *************************************************************************
CH    REVISION HISTORY  :
C     VERSION       BY                  DATE         COMMENT
C       1.1     A. de Waal            06/01/90   Finally Commented
C     FILEND            :
C *************************************************************************
C
C     FILE              : MULT.FOR
C *************************************************************************
CN    MODULE NAME       : mult
CA    FUNCTION          : Multiply Two Matrices
CS    CALLING SEQUENCE  : CALL MULT(A,B,C,I,J,K,L)
CI    INPUT PARAMETERS  : A:MATRIX OF DIMENSION (I,J)
C                                 B:MATRIX OF DIMENSION (N,L)
CO    OUTPUT PARAMETERS : C=AB:MATRIX OF DIMENSION I,L
C                         ERR: ERR = 0 correct dimensions for marix mult
C                              ERR = 1 incorrect dimensions for matrix mult
CG    GLOBAL VARIABLES  : None
CM    MODULES CALLED    : None
CE    ERROR CONDITIONS  : STOPS IF J.NE.K ie.DIMENSIONS DON'T MATCH
CC    COMMENTS          : Multiplies matrix A by matrix B and returns result
C                         in matrix C
C *************************************************************************
      SUBROUTINE MULT(A,B,C,I,J,K,L,ERR)
      INTEGER ERR
        PARAMETER (ID=15)
        DIMENSION A(ID,ID),B(ID,ID),C(ID,ID)
100   format(A30)
      ERR = 0
        IF (J.NE.K) THEN
          ERR = 1
          RETURN
        END IF
        DO 20 I1=1,I
                DO 10 I2=1,L
                        C(I1,I2)=0.
10              CONTINUE
20      CONTINUE
        DO 50 I1=1,I
                DO 40 I2=1,L
                        DO 30 I3=1,J
                                C(I1,I2)=C(I1,I2)+A(I1,I3)*B(I3,I2)
30                      CONTINUE
40              CONTINUE
50      CONTINUE
        RETURN
        END
C *************************************************************************
CH    REVISION HISTORY  :
C     VERSION       BY                  DATE         COMMENT
C       0.00    R.HENNING             07/04/87
C       1.1     A. de Waal            06/01/90   Finally Commented
C     FILEND            :
C *************************************************************************
C
C     FILE              : RNDNRM.FOR
C *************************************************************************
CN    MODULE NAME       : rndnrm
CA    FUNCTION          : Generate Random Numbers with Gaussian PDF
CS    CALL SEQUENCE     : rand = rndnrm(mean,stdev,irand)
CI    INPUT PARAMETERS  : mean    - double precision : Mean of normal distrib
C                         stdev   - double precision : Standard deviation of
C                                                      normal distribution
CO    OUTPUT PARAMETERS : rndnrm - double precision : Random number
CG    GLOBAL VARIABLES  : None
CM    MODULES CALLED    : Fortran  :  urand
CE    ERROR CONDITIONS  : None
CC    COMMENTS          : Calculates a random normal deviate.  This is a number
C                         randomly drawn from a normal distribution.  The
C                         apecific normal distribution can be identified by
C                         giving the mean and the standard deviation
C                         Taken from Rugg, Tom and Feldman, Phil
C                             "Turbo Pascal Program Library"
C                             Que Corporation, Indianapolis, Copyright 1986
```

```
C ***************************************************************************
      double precision function rndnrm(mean,stdev,irand)
      implicit double precision (u)
      double precision randa,randb,rad2,dev,mean,stdev
      integer*4 irand

100   continue
          randa = 2.0*urand(irand) - 1.0
          randb = 2.0*urand(irand) - 1.0
          rad2 = randa*randa + randb*randb
      if ( rad2.ge.(dble(1.0)) ) goto 100
      dev = randa*dsqrt( (-2.0*dlog(rad2))/rad2 )
      rndnrm = mean + dev*stdev
      return
      end
C ***************************************************************************
CH    REVISION HISTORY  :
C     VERSION       BY                DATE        COMMENT
C       1.1     A. de Waal           06/01/90   Finally Commented
C     FILEND            :
C ***************************************************************************


C
C     FILE              : SIMEQ.FOR
C ***************************************************************************
CN    MODULE NAME       : simeq
CA    FUNCTION          : Solve System of Simultaneous Equations
CS    CALL SEQUENCE     : call simeq(a,b,n)
CI    INPUT PARAMETERS  : a(max,max): real - Matrix of maximum
C                                            dimension (max,max)
C                         b(max)     : real - Vector of maximum dimension (max)
C                         n          : integer - Actual dimension of
C                                                matrix/vector
CO    OUTPUT PARAMETERS : a,b : Solution to simultaneous equation
CG    GLOBAL VARIABLES  : None
CM    MODULES CALLED    : None
CE    ERROR CONDITIONS  : None
CC    COMMENTS          : Subroutine calculates system of linear simultaneous
C                         equations a*b = 0
C ***************************************************************************
      SUBROUTINE SIMEQ(a,b,n)
      parameter (max = 15)
      parameter (max2 = max+1)
      real a(max,max)
      real b(max)
      real g(max,max2)

      do 10 i=1,n
         do 20 j=1,n
            g(i,j) = a(i,j)
20       continue
10    continue

      do 30 i=1,n
         g(i,n+1) = b(i)
30    continue

      do 40 ip=1,n
         piv = g(ip,ip)
         do 50 j=1,n+1
            g(ip,j) = g(ip,j)/piv
50       continue
         do 60 it=1,n
            if (it.ne.ip) then
               tarco = g(it,ip)
               do 70 j=1,n+1
                  g(it,j) = g(it,j)-g(ip,j)*tarco
70             continue
            endif
60       continue
40    continue

      do 80 i=1,n
         b(i) = g(i,n+1)
80    continue
      return
      end
C ***************************************************************************
CH    REVISION HISTORY  :
C     VERSION       BY                DATE        COMMENT
C       1.1     A. de Waal           06/01/90   Finally Commented
C     FILEND            :
C ***************************************************************************
C
C     FILE              : SVD.FOR
C ***************************************************************************
CN    MODULE NAME       : svd
CA    FUNCTION          : Singular Value Decomposition
CS    CALL SEQUENCE     : call svd(matr,mati,q1r,q1i,q2r,q2i,mord,svvec,ierr)
CI    INPUT PARAMETERS  : matr(ip,ip): real - Real part of matrix, dimension
C                                             (ip,ip) on which singular value
C                                             decomposition to be performed
C                         : mati(ip,ip): real - Imaginary part of matrix, dimen
C                                             (ip,ip) on which singular value
C                                             decomposition to be performed
C                         : q1r(ip,ip):  real - Real part of matrix, dimension
C                                             (ip,ip) used for scaling in
C                                             singular value decomposition
C                                             procedure
C                         : q1i(ip,ip):  real - Imaginary part of matrix, dimen
C                                             (ip,ip) used for scaling in
C                                             singular value decomposition
C                                             procedure
C                         : q2r(ip,ip):  real - Real part of matrix, dimension
C                                             (ip,ip) used for scaling in
C                                             singular value decomposition
C                                             procedure
C                         : q2i(ip,ip):  real - Imaginary part of matrix, dimen
C                                             (ip,ip) used for scaling in
C                                             singular value decomposition
C                                             procedure
C                         : mord:      integer - Actual order of square matrix
C                                                (matr,mati)
CO    OUTPUT PARAMETERS : svvec:  real - Vector of dimension (ip) containing
C                                        singular values of matrix (matr,mati)
```

```
C                         ierr:integer - Flag indicating (if nonzero) that
C                                        error ocurred in singular value
C                                        decomposition procedure
CG    GLOBAL VARIABLES  : Common blocks:  /utilmt/ zermat,eyemat
C                                         /vecs/ tevecr,teveci
CM    MODULES CALLED    : Fortran  :  add,trans,cinv,cmult,qzveca
CE    ERROR CONDITIONS  : Error in determining eigenvalues indicated by setting
C                         flag ierr to nonzero value
CC    COMMENTS          : This routine determines the vectior svvec(ip) with
C                         elements
C                         svvec(i) = sqare root(eigenvalues(a*astar))
C                         where
C                  a            = original complex matrix (matr,mati)
C                  (astr,asti) = inv(q1r,q1i)*trans(conj(matr,mati))*(q2r,q2i)
C                         according to the definition of the singular values of
C                         a matrix being the sqare roots of the eigenvalues of
C                         a matrix multiplied by its complex conjugate
C ***************************************************************************
      subroutine svd(matr,mati,q1r,q1i,q2r,q2i,mord,svvec,ierr)
      integer ip
      parameter (ip = 15)
      integer mord,ierr
      real matr,mati,qr1,qi1,q2r,q2i,
     +     tematr,temati,tema2r,tema2i,zermat,eyemat,
     +     ctmatr,ctmati,astr,asti
      real svvec,
     +     tevecr,teveci
      dimension matr(ip,ip),mati(ip,ip),
     +          qr1(ip,ip),qi1(ip,ip),q2r(ip,ip),q2i(ip,ip),
     +          tematr(ip,ip),temati(ip,ip),tema2r(ip,ip),tema2i(ip,ip),
     +          zermat(ip,ip),eyemat(ip,ip),
     +          ctmatr(ip,ip),ctmati(ip,ip),astr(ip,ip),asti(ip,ip)
      dimension svvec(ip),tevecr(ip),teveci(ip)
C      common /utilmt/ tematr,temati,tema2r,tema2i,
C     +                zermat,eyemat
      common /utilmt/ zermat,eyemat
      common /vecs/   tevecr,teveci

      do 100 i = 1, ip
          do 110 j = 1, ip
              zermat(i,j) = 0.0
110       continue
          svvec(i) = 0.0
100   continue

C Calculating complex conjugate transpose of matrix (matr,mati)
      call add(zermat,mati,temati,mord,mord,mord,mord,1,merr)
      call trans(matr,ctmatr,mord,mord)
      call trans(temati,ctmati,mord,mord)

C Calculating (astr,asti) = inv(q1r,q1i)*trans(conj(matr,mati))*(q2r,q2i)
      call cinv(q1r,q1i,tematr,temati,mord,ip,2*ip)
      call cmult(tematr,temati,ctmatr,ctmati,tema2r,tema2i,
     +           mord,mord,mord,mord,merr)
      call cmult(tema2r,tema2i,q2r,q2i,astr,asti,
     +           mord,mord,mord,mord,merr)

C Calculating (tematr,temati) = (matr,mati)*(astr,asti)
      call cmult(matr,mati,astr,asti,tematr,temati,
     +           mord,mord,mord,mord,merr)

C Calculating eigenvalues of (matr,mati)*(astr,asti)
      call qzveca(ip,tematr,temati,tevecr,teveci,tema2r,tema2i,ierr)
      if (ierr.ne.0) return

C Calculating the singular values of matrix
C     (svvecr,svveci) = sqrt( eig((matr,mati)*(astr,asti)) )
C                     = sqrt( positive real )
      do 200 i = 1, ip
          svvec(i) = sqrt(tevecr(i))
200   continue

      return
      end

C ***************************************************************************
CH    REVISION HISTORY  :
C     VERSION      BY                  DATE        COMMENT
C       1.1     A. de Waal           06/01/90   Finally Commented
C     FILEND            :
C ***************************************************************************
C
C     FILE              : SVMAX.FOR
C ***************************************************************************
CN    MODULE NAME       : svmax
CA    FUNCTION          : Obtain Maximum Singular Value of Matrix
CS    CALL SEQUENCE     : call svmax(matr,mati,q1r,q1i,q2r,q2i,mord,value,ierr)
CI    INPUT PARAMETERS  : matr(ip,ip): real - Real part of matrix, dimension
C                                             (ip,ip) on which singular value
C                                             decomposition to be performed
C                         : mati(ip,ip): real - Imaginary part of matrix, dimen
C                                             (ip,ip) on which singular value
C                                             decomposition to be performed
C                         : q1r(ip,ip):  real - Real part of matrix, dimension
C                                             (ip,ip) used for scaling in
C                                             singular value decomposition
C                                             procedure
C                         : q1i(ip,ip):  real - Imaginary part of matrix, dimen
C                                             (ip,ip) used for scaling in
C                                             singular value decomposition
C                                             procedure
C                         : q2r(ip,ip):  real - Real part of matrix, dimension
C                                             (ip,ip) used for scaling in
C                                             singular value decomposition
C                                             procedure
C                         : q2i(ip,ip):  real - Imaginary part of matrix, dimen
C                                             (ip,ip) used for scaling in
C                                             singular value decomposition
C                                             procedure
C                         : mord:      integer - Actual order of square matrix
C                                             (matr,mati)
CO    OUTPUT PARAMETERS : value:     integer - Maximum singular value
C                         ierr :     integer - Flag set to nonzero if svd
C                                             algorithm fails
CG    GLOBAL VARIABLES  : None
CM    MODULES CALLED    : Fortran  :  svd
```

```
CE     ERROR CONDITIONS   : Flag ierr set to nonzero if svd algorithm fails
CC     COMMENTS           : Subroutine calls subroutine svd to calculate the
C                           singular values of matrix (matr,mati) scaled by
C                           indicated matrices.  The maximum entry in the vector
C                           (svvec) of singular values is then identified and
C                           returned to the calling procedure.
C ***************************************************************************
      subroutine svmax(matr,mati,q1r,q1i,q2r,q2i,mord,value,ierr)
      integer id
      parameter (id = 15)
      integer mord,ierr
      real matr,mati,q1r,q1i,q2r,q2i
      real svvec
      real value
      dimension matr(id,id),mati(id,id),
     +          q1r(id,id),q1i(id,id),q2r(id,id),q2i(id,id)
      dimension svvec(id)

      call svd(matr,mati,q1r,q1i,q2r,q2i,mord,svvec,ierr)
      if (ierr.ne.0) goto 101
      value = svvec(1)
      do 100 i = 2, mord
          if (svvec(i).gt.value) value = svvec(i)
100   continue
101   continue
      return
      end


C ***************************************************************************
CH     REVISION HISTORY   :
C      VERSION        BY                  DATE         COMMENT
C        1.1      A. de Waal            06/01/90    Finally Commented
C      FILEND             :
C ***************************************************************************


C
C      FILE               : SVMIN.FOR
C ***************************************************************************
CN     MODULE NAME        : svmin
CA     FUNCTION           : Obtain Minimum Singular Value of Matrix
CS     CALL SEQUENCE      : call svmin(matr,mati,q1r,q1i,q2r,q2i,mord,value,ierr)
CI     INPUT PARAMETERS   : matr(ip,ip): real - Real part of matrix, dimension
C                                               (ip,ip) on which singular value
C                                               decomposition to be performed
C                           : mati(ip,ip): real - Imaginary part of matrix, dimen
C                                               (ip,ip) on which singular value
C                                               decomposition to be performed
C                           : q1r(ip,ip):  real - Real part of matrix, dimension
C                                               (ip,ip) used for scaling in
C                                               singular value decomposition
C                                               procedure
C                           : q1i(ip,ip):  real - Imaginary part of matrix, dimen
C                                               (ip,ip) used for scaling in
C                                               singular value decomposition
C                                               procedure
C                           : q2r(ip,ip):  real - Real part of matrix, dimension
C                                               (ip,ip) used for scaling in
C                                               singular value decomposition
C                                               procedure
C                           : q2i(ip,ip):  real - Imaginary part of matrix, dimen
C                                               (ip,ip) used for scaling in
C                                               singular value decomposition
C                                               procedure
C                           : mord:     integer - Actual order of square matrix
C                                               (matr,mati)
CO     OUTPUT PARAMETERS  : value:    integer - Minimum singular value
C                             ierr :    integer - Flag set to nonzero if svd
C                                               algorithm fails
CG     GLOBAL VARIABLES   : None
CM     MODULES CALLED     : Fortran  :  svd
CE     ERROR CONDITIONS   : Flag ierr set to nonzero if svd algorithm fails
CC     COMMENTS           : Subroutine calls subroutine svd to calculate the
C                           singular values of matrix (matr,mati) scaled by
C                           indicated matrices.  The minimum entry in the vector
C                           (svvec) of singular values is then identified and
C                           returned to the calling procedure.
C ***************************************************************************
      subroutine svmin(matr,mati,q1r,q1i,q2r,q2i,mord,value,ierr)
      integer id
      parameter (id = 15)
      integer mord,ierr
      real matr,mati,q1r,q1i,q2r,q2i
      real svvec
      real value
      dimension matr(id,id),mati(id,id),
     +          q1r(id,id),q1i(id,id),q2r(id,id),q2i(id,id)
      dimension svvec(id)

      call svd(matr,mati,q1r,q1i,q2r,q2i,mord,svvec,ierr)
      if (ierr.ne.0) goto 101
      value = svvec(1)
      do 100 i = 2, mord
          if (svvec(i).lt.value) value = svvec(i)
100   continue
101   continue
      return
      end


C ***************************************************************************
CH     REVISION HISTORY   :
C      VERSION        BY                  DATE         COMMENT
C        1.1      A. de Waal            06/01/90    Finally Commented
C      FILEND             :
C ***************************************************************************
C
C      FILE               : TRANS.FOR
C ***************************************************************************
CN     MODULE NAME        : trans
CA     FUNCTION           : Return the Transpose of a Matrix
CS     CALLING SEQUENCE   : CALL TRANS(AI,BI,I,J)
CI     INPUT PARAMETERS   : AI:MATRIX OF DIMENSION (I,J)
CO     OUTPUT PARAMETERS  : BI:MATRIX OF DIMENSION (J,I)
CG     GLOBAL VARIABLES   : None
```

```
CM     MODULES CALLED     : None
CE     ERROR CONDITIONS   : None
CC     COMMENTS           : THE MATRIX DIMENSIONS ARE SET VIA A PARAMETER
C                           STATEMENT THIS IS FOR CONVENIENCE IN USING THE SWAT
C                           DEBUGGER ON THE MV10000
C ***************************************************************************
        SUBROUTINE TRANS(AI,BI,I,J)
        PARAMETER (ID=15)
        DIMENSION AI(ID,ID),BI(ID,ID)
      do 100 i1 = 1,i
         do 200 i2 = 1,j
            bi(i2,i1) = 0.0
200      continue
100   continue
        DO 20 I1=1,I
                DO 10 I2=1,J
                        BI(I2,I1)=AI(I1,I2)
10              CONTINUE
20      CONTINUE
        RETURN
        END
C ***************************************************************************
CH     REVISION HISTORY   :
C      VERSION       BY                    DATE         COMMENT
C         1.00     R.HENNING               07/04/87
C       1.1      A. de Waal              06/01/90    Finally Commented
C      FILEND             :
C ***************************************************************************
C
C      FILE               : URAND.FOR
C ***************************************************************************
CN     MODULE NAME        : urand
CA     FUNCTION           : Uniform Random Number Generator
CS     CALL SEQUENCE      : call urand(iy)
CI     INPUT PARAMETERS   : iy: integer*4 - used by urand subroutine - must be
C                                           initialized by user only before
C                                           first calling of urand and never
C                                           changed again.
CO     OUTPUT PARAMETERS  : iy: value to be stored and not altered
CG     GLOBAL VARIABLES   : None
CM     MODULES CALLED     : None
CE     ERROR CONDITIONS   : None
CC     COMMENTS           :
C       URAND IS A UNIFORM RANDOM NUMBER GENERATOR BASED  ON  THEORY  AND
C  SUGGESTIONS  GIVEN  IN  D.E. KNUTH (1969),  VOL  2.   THE integer*4  IY
C  SHOULD BE INITIALIZED TO AN ARBITRARY integer*4 PRIOR TO THE FIRST CALL
C  TO URAND.  THE CALLING PROGRAM SHOULD  NOT  ALTER  THE  VALUE  OF  IY
C  BETWEEN  SUBSEQUENT CALLS TO URAND.  VALUES OF URAND WILL BE RETURNED
C  IN THE INTERVAL (0,1).
C       SEE FORSYTHE, MALCOLM AND MOLER (1977).
C ***************************************************************************
      DOUBLE PRECISION FUNCTION URAND(IY)
      integer*4 IY
C
C
      integer*4 IA,IC
      DOUBLE PRECISION HALFM,FULLM,DA,DC,S
      DATA FULLM/0.0D0/
C
C     FOR MS-FORTRAN VERSION V3.3
C
      IF (FULLM .GT. 0.0D0) GOTO 20
      HALFM = 1000000000.0D0
      FULLM = 2000000000.0D0
C
C  COMPUTE MULTIPLIER AND INCREMENT FOR LINEAR CONGRUENTIAL METHOD
C
      IA = int4(8*int4(HALFM*ATAN(1.D0)/8.D0) + int4(5))
      IC = int4(2*int4(HALFM*(0.5D0-SQRT(3.D0)/6.D0)) + int4(1))
C
      DA = DBLE(IA)
      DC = DBLE(IC)
C
C  S IS THE SCALE FACTOR FOR CONVERTING TO FLOATING POINT
C
      S = 0.5D0/HALFM
C
C  COMPUTE NEXT RANDOM NUMBER
C
20    IY    = int4(MOD(DA*DBLE(IY)+DC,FULLM))
      URAND = DBLE(IY)*S
      RETURN
      END
C ***************************************************************************
CH     REVISION HISTORY   :
C      VERSION       BY                    DATE         COMMENT
C       1.1      A. de Waal              06/01/90    Finally Commented
C      FILEND             :
C ***************************************************************************
```

M4) Graphics Utility Subroutines

A table of the graphics utility subroutine names follows on the next page. The name of each subroutine, together with the file in which it is located and the page in the appendix on which it is listed, is given in the table.

Graphics Utility Subroutines			
Name	**File**	**Description**	**Page**
arhead	ARHEAD.FOR	Draw Arrowhead on Present Graphics Screen	435
drwlet	DRWLET.FOR	Draw Number on Graph on Specified Graphics Page	435
geti2	GETI2.FOR	Input or Edit Integer(*2) on Graphics Page	436
geti4	GETI4.FOR	Input or Edit Integer(*4) on Graphics Page	436
getr4	GETR4.FOR	Input or Edit Real(*4) on Graphics Page	436
getr8	GETR8.FOR	Input or Edit Real(*8) on Graphics Page	437
prti2	PRTI2.FOR	Print Integer(*2) Number to Graphics page.	437
prti4	PRTI4.FOR	Print Integer(*4) Number to Graphics page.	437
prtr4	PRTR4.FOR	Print Real(*4) Number to Graphics page.	438
prtr8	PRTR8.FOR	Print Real(*8) Number to Graphics page.	438
wrhead	WRHEAD.FOR	Write Heading on Graphics Page	438

```
C
C     FILE              : ARHEAD.FOR
C ***************************************************************************
CN    MODULE NAME       : arhead
CA    FUNCTION          : Draw Arrowhead on Present Graphics Screen
CS    CALL SEQUENCE     : call arhead(artype,size)
CI    INPUT PARAMETERS  : artype: integer - Arrow type specifier
C                                           1:  left to right
C                                           2:  right to left
C                                           3:  top to bottom
C                                           4:  bottom to top
C                         size  : integer - cross sectional size in dots
CO    OUTPUT PARAMETERS : None
CG    GLOBAL VARIABLES  : None
CM    MODULES CALLED    : Assembler:  MOVE,DLINE
CE    ERROR CONDITIONS  : None
CC    COMMENTS          : Just draws specified arrowhead on graphics page
C                         active at the time of calling routine in the position
C                         of the cursor at present.
C ***************************************************************************
      subroutine arhead(artype,size)
$include: 'heblock.inc'
      integer*2 artype,size

      external MOVE
      external DLINE

C Arrowhead from left to right
      if (artype.eq.1) then
          xpos = x - size
          ypos = y - (3*size/4)
C Arrowhead from right to left
      elseif (artype.eq.2) then
          xpos = x + size
          ypos = y - (3*size/4)
C Arrowhead from top to bottom
      elseif (artype.eq.3) then
          ypos = y - (3*size/4)
          xpos = x - size
C Arrowhead from bottom to top
       else
          ypos = y + (3*size/4)
          xpos = x - size
       endif

       call MOVE(x,y)
       call DLINE(xpos,ypos)

C Arrowhead from left to right
C Arrowhead from right to left
      if (artype.le.2) then
          ypos = y + (3*size/4)
C Arrowhead from top to bottom
C Arrowhead from bottom to top
       else
          xpos = x + size
       endif

       call MOVE(x,y)
       call DLINE(xpos,ypos)
       call MOVE(x,y)

       return
       end
C ***************************************************************************
CH    REVISION HISTORY  :
C     VERSION       BY                  DATE        COMMENT
C       1.1     A. de Waal            06/01/90   Finally Commented
C     FILEND            :
C ***************************************************************************

C
C     FILE              : DRWLET.FOR
C ***************************************************************************
CN    MODULE NAME       : drwlet
CA    FUNCTION          : Draw Number on Graph on Specified Graphics Page
CS    CALL SEQUENCE     : call drwlet(pg,x,y,cent,scal,let)
CI    INPUT PARAMETERS  : pg: integer - Graphics page on which to draw
C                         x,y : integer - Co-ords on page where to draw
C                 scal(2)   - (real*8) The   axes    to   page
C                             scaling   factors   (  dots/axis
C                             unit).
C                             scale(1) - x axes scale.
C                             scale(2) - y axes scale.
C               cent(4)     - (real*8) The origin of  the axes
C                             on the actual  graphics page and
C                             the  axes  origin  (as given  in
C                             x(3) and y(3)).
C                             centre(1) - x position of origin
C                                         specified in dots.
C                             centre(2) - y position of origin
C                                         specified in dots.
C                             centre(3) - x co-ord  of  origin
C                                         specified   in  axes
C                                         units.
C                             centre(4) - y co-ord  of  origin
C                                         specified  in   axes
C                                         units.
C                         let: character*1 - Letter/number to be drawn
CO    OUTPUT PARAMETERS : None
CG    GLOBAL VARIABLES  : None
CM    MODULES CALLED    : Assembler:  GPAGE,LEVEL
C                         Fortran   :  gnum
CE    ERROR CONDITIONS  : None
CC    COMMENTS          : Draws graphics number on a set of axes drawn on
C                         the indicated graphics screen.  The size and position
C                         of the number are scaled appropriately.
C ***************************************************************************
      subroutine drwlet(pg,x,y,cent,scal,let)
      integer*2 pg
      real*4 x,y
      real*8 cent(4),scal(2)
      character*1 let
      integer*2 xp,yp

      xp = int((x-cent(3))*scal(1)+cent(1))+2
```

```
      yp = int(cent(2)-(y-cent(4))*scal(2))
      call GPAGE(pg)
      call LEVEL(1)
      call gnum(pg,xp,yp,let)
      return
      end
C ********************************************************************************
CH    REVISION HISTORY  :
C     VERSION       BY                  DATE         COMMENT
C       1.1     A. de Waal            06/01/90   Finally Commented
C     FILEND            :
C ********************************************************************************

C
C     FILE              : GETI2.FOR
C ********************************************************************************
CN    MODULE NAME       : geti2
CA    FUNCTION          : Input or Edit Integer(*2) on Graphics Page
CS    CALL SEQUENCE     : key = geti2(pg,x,y,flen,form,width,ni2)
CI    INPUT PARAMETERS  : pg:   integer - Page on which number is to be edited
C                                         (0 or 1)
C                         x,y:  integer - Co-ordinates of number on screen
C                         flen: integer - Length of the format string:format
C                         form: character*flen - Format string (eg.'(f10.4)')
C                         width:integer - Width of field to be printed
C                         ni2 : integer*2 - The number to be printed
CO    OUTPUT PARAMETERS : geti2 : integer - Returned key from routine
CG    GLOBAL VARIABLES  : None
CM    MODULES CALLED    : getnum
CE    ERROR CONDITIONS  : None
CC    COMMENTS          : Utility routine calling getnum routine for the
C                         editing of integer*2 numbers on the graphics page
C                         specified by user.
C ********************************************************************************
      integer*2 function geti2(pg,x,y,flen,form,width,ni2)
      implicit integer*2 (G)
      integer*2 type /1/, ni2
      integer*4 ni4 /0/
      real*4 nr4 /0/
      real*8 nr8 /0/
      geti2 = getnum(pg,x,y,type,flen,form,width,ni2,ni4,nr4,nr8)
      return
      end
C ********************************************************************************
CH    REVISION HISTORY  :
C     VERSION       BY                  DATE         COMMENT
C       1.1     A. de Waal            06/01/90   Finally Commented
C     FILEND            :
C ********************************************************************************

C
C     FILE              : GETI4.FOR
C ********************************************************************************
CN    MODULE NAME       : geti4
CA    FUNCTION          : Input or Edit Integer(*4) on Graphics Page
CS    CALL SEQUENCE     : key = geti2(pg,x,y,flen,form,width,ni4)
CI    INPUT PARAMETERS  : pg:   integer - Page on which number is to be edited
C                                         (0 or 1)
C                         x,y:  integer - Co-ordinates of number on screen
C                         flen: integer - Length of the format string:format
C                         form: character*flen - Format string (eg.'(f10.4)')
C                         width:integer - Width of field to be printed
C                         ni4 : integer*4 - The number to be printed
CO    OUTPUT PARAMETERS : geti4 : integer - Returned key from routine
CG    GLOBAL VARIABLES  : None
CM    MODULES CALLED    : getnum
CE    ERROR CONDITIONS  : None
CC    COMMENTS          : Utility routine calling getnum routine for the
C                         editing of integer*4 numbers on the graphics page
C                         specified by user.
C ********************************************************************************
      integer*2 function geti4(pg,x,y,flen,form,width,ni4)
      implicit integer*2 (G)
      integer*2 type /2/, ni2 /0/
      integer*4 ni4
      real*4 nr4 /0/
      real*8 nr8 /0/
      geti4 = getnum(pg,x,y,type,flen,form,width,ni2,ni4,nr4,nr8)
      return
      end
C ********************************************************************************
CH    REVISION HISTORY  :
C     VERSION       BY                  DATE         COMMENT
C       1.1     A. de Waal            06/01/90   Finally Commented
C     FILEND            :
C ********************************************************************************

C
C     FILE              : GETR4.FOR
C ********************************************************************************
CN    MODULE NAME       : getr4
CA    FUNCTION          : Input or Edit Real(*4) on Graphics Page
CS    CALL SEQUENCE     : key = getr4(pg,x,y,flen,form,width,nr4)
CI    INPUT PARAMETERS  : pg:   integer - Page on which number is to be edited
C                                         (0 or 1)
C                         x,y:  integer - Co-ordinates of number on screen
C                         flen: integer - Length of the format string:format
C                         form: character*flen - Format string (eg.'(f10.4)')
C                         width:integer - Width of field to be printed
C                         ni2 : real*4 - The number to be printed
CO    OUTPUT PARAMETERS : getr4 : integer - Returned key from routine
CG    GLOBAL VARIABLES  : None
CM    MODULES CALLED    : getnum
CE    ERROR CONDITIONS  : None
CC    COMMENTS          : Utility routine calling getnum routine for the
C                         editing of real*4 numbers on the graphics page
C                         specified by user.
C ********************************************************************************
      integer*2 function getr4(pg,x,y,flen,form,width,nr4)
      implicit integer*2 (G)
      integer*2 type /3/, ni2 /0/
      integer*4 ni4 /0/
      real*4 nr4
      real*8 nr8 /0/
      getr4 = getnum(pg,x,y,type,flen,form,width,ni2,ni4,nr4,nr8)
      return
```

```
      end
C ********************************************************************************
CH    REVISION HISTORY  :
C     VERSION       BY                    DATE         COMMENT
C       1.1     A. de Waal            06/01/90   Finally Commented
C     FILEND            :
C ********************************************************************************

C
C     FILE              : GETR8.FOR
C ********************************************************************************
CN    MODULE NAME       : getr8
CA    FUNCTION          : Input or Edit Real(*8) on Graphics Page
CS    CALL SEQUENCE     : key = getr8(pg,x,y,flen,form,width,nr8)
CI    INPUT PARAMETERS  : pg:   integer - Page on which number is to be edited
C                                         (0 or 1)
C                         x,y:  integer - Co-ordinates of number on screen
C                         flen: integer - Length of the format string:format
C                         form: character*flen - Format string (eg.'(f10.4)')
C                         width:integer - Width of field to be printed
C                         ni2 : real*8 - The number to be printed
CO    OUTPUT PARAMETERS : getr8 : integer - Returned key from routine
CG    GLOBAL VARIABLES  : None
CM    MODULES CALLED    : getnum
CE    ERROR CONDITIONS  : None
CC    COMMENTS          : Utility routine calling getnum routine for the
C                         editing of real*8 numbers on the graphics page
C                         specified by user.
C ********************************************************************************
      integer*2 function getr8(pg,x,y,flen,form,width,nr8)
      implicit integer*2 (G)
      integer*2 type /4/, ni2 /0/
      integer*4 ni4 /0/
      real*4 nr4 /0/
      real*8 nr8
      getr8 = getnum(pg,x,y,type,flen,form,width,ni2,ni4,nr4,nr8)
      return
      end
C ********************************************************************************
CH    REVISION HISTORY  :
C     VERSION       BY                    DATE         COMMENT
C       1.1     A. de Waal            06/01/90   Finally Commented
C     FILEND            :
C ********************************************************************************

C
C     FILE              : PRTI2.FOR
C ********************************************************************************
CN    MODULE NAME       : prti2
CA    FUNCTION          : Print Integer(*2) Number to Graphics page.
CS    CALL SEQUENCE     : call prti2(page,x,y,flen,format,width,ni2)
CI    INPUT PARAMETERS  : page: integer*2 - The page on which the number is to
C                                           be printed. (range : 0 or 1)
C                         x,y : integer*2 - The co-ordinates of the printed
C                                           number.
C                         flen: integer*2 - Length of the format string:format.
C                         format: character*flen - The number format string.
C                                                  (eg. '(f10.4)' flen = 7)
C                         width: integer*2 - The width of the field to be
C                                            printed.
C                         ni2  : integer*2 - The number to be printed.
C
CO    OUTPUT PARAMETERS: None.
CG    GLOBAL VARIABLES : None.
CM    MODULES CALLED   : Fortran  :  numstr
C                        Assembler:  WRTSTR
CE    ERROR CONDITIONS : The format string must not be greater than 40 chars.
CC    COMMENTS         : Utility routine calling prtnum routine for the
C                        printing of integer*2 numbers on the graphics page
C                        specified by user.
C ********************************************************************************
      subroutine prti2(pg,x,y,flen,form,width,ni2)
      integer*2 pg,x,y,type /1/,flen,width,ni2
      integer*4 ni4 /0/
      real*4 nr4 /0/
      real*8 nr8 /0/
      character*40 form

      character*40 prstr
      call numstr(type,flen,form,ni2,ni4,nr4,nr8,width,prstr)
      call WRTSTR(pg,x,y,width,prstr)
      return
      end
C ********************************************************************************
CH    REVISION HISTORY  :
C     VERSION       BY                    DATE         COMMENT
C       1.1     A. de Waal            06/01/90   Finally Commented
C     FILEND            :
C ********************************************************************************

C
C     FILE              : PRTI4.FOR
C ********************************************************************************
CN    MODULE NAME       : prti4
CA    FUNCTION          : Print Integer(*4) Number to Graphics page.
CS    CALL SEQUENCE     : call prti4(page,x,y,flen,format,width,ni4)
CI    INPUT PARAMETERS  : page: integer*2 - The page on which the number is to
C                                           be printed. (range : 0 or 1)
C                         x,y : integer*2 - The co-ordinates of the printed
C                                           number.
C                         flen: integer*2 - Length of the format string:format.
C                         format: character*flen - The number format string.
C                                                  (eg. '(f10.4)' flen = 7)
C                         width: integer*2 - The width of the field to be
C                                            printed.
C                         ni4  : integer*4 - The number to be printed.
C
CO    OUTPUT PARAMETERS: None.
CG    GLOBAL VARIABLES : None.
CM    MODULES CALLED   : Fortran  :  numstr
C                        Assembler:  WRTSTR
CE    ERROR CONDITIONS : The format string must not be greater than 40 chars.
CC    COMMENTS         : Utility routine calling prtnum routine for the
C                        printing of integer*4 numbers on the graphics page
C                        specified by user.
C ********************************************************************************
      subroutine prti4(pg,x,y,flen,form,width,ni4)
```

```
      integer*2 pg,x,y,type /2/,flen,width,ni2 /0/
      integer*4 ni4
      real*4 nr4 /0/
      real*8 nr8 /0/
      character*40 form

      character*40 prstr
      call numstr(type,flen,form,ni2,ni4,nr4,nr8,width,prstr)
      call WRTSTR(pg,x,y,width,prstr)
      return
      end
C *************************************************************************
CH    REVISION HISTORY  :
C     VERSION       BY                  DATE        COMMENT
C       1.1     A. de Waal            06/01/90   Finally Commented
C     FILEND          :
C *************************************************************************
C
C     FILE            : PRTR4.FOR
C *************************************************************************
CN    MODULE NAME     : prtr4
CA    FUNCTION        : Print Real(*4) Number to Graphics page.
CS    CALL SEQUENCE   : call prtr4(page,x,y,flen,format,width,nr4)
CI    INPUT PARAMETERS : page: integer*2 - The page on which the number is to
C                                          be printed. (range : 0 or 1)
C                        x,y : integer*2 - The co-ordinates of the printed
C                                          number.
C                        flen: integer*2 - Length of the format string:format.
C                        format: character*flen - The number format string.
C                                          (eg. '(f10.4)' flen = 7)
C                        width: integer*2 - The width of the field to be
C                                          printed.
C                        nr4  : real*4 - The number to be printed.
C
CO    OUTPUT PARAMETERS: None.
CG    GLOBAL VARIABLES : None.
CM    MODULES CALLED   : Fortran  :  numstr
C                        Assembler:  WRTSTR
CE    ERROR CONDITIONS : The format string must not be greater than 40 chars.
CC    COMMENTS         : Utility routine calling prtnum routine for the
C                        printing of real*4 numbers on the graphics page
C                        specified by user.
C *************************************************************************
      subroutine prtr4(pg,x,y,flen,form,width,nr4)
      integer*2 pg,x,y,type /3/,flen,width,ni2 /0/
      integer*4 ni4 /0/
      real*4 nr4
      real*8 nr8 /0/
      character*40 form

      character*40 prstr
      call numstr(type,flen,form,ni2,ni4,nr4,nr8,width,prstr)
      call WRTSTR(pg,x,y,width,prstr)
      return
      end
C *************************************************************************
CH    REVISION HISTORY  :
C     VERSION       BY                  DATE        COMMENT
C       1.1     A. de Waal            06/01/90   Finally Commented
C     FILEND          :
C *************************************************************************
C
C     FILE            : PRTR8.FOR
C *************************************************************************
CN    MODULE NAME     : prtr8
CA    FUNCTION        : Print Real(*8) Number to Graphics page.
CS    CALL SEQUENCE   : call prtr8(page,x,y,flen,format,width,nr8)
CI    INPUT PARAMETERS : page: integer*2 - The page on which the number is to
C                                          be printed. (range : 0 or 1)
C                        x,y : integer*2 - The co-ordinates of the printed
C                                          number.
C                        flen: integer*2 - Length of the format string:format.
C                        format: character*flen - The number format string.
C                                          (eg. '(f10.4)' flen = 7)
C                        width: integer*2 - The width of the field to be
C                                          printed.
C                        nr8  : real*8 - The number to be printed.
C
CO    OUTPUT PARAMETERS: None.
CG    GLOBAL VARIABLES : None.
CM    MODULES CALLED   : Fortran  :  numstr
C                        Assembler:  WRTSTR
CE    ERROR CONDITIONS : The format string must not be greater than 40 chars.
CC    COMMENTS         : Utility routine calling prtnum routine for the
C                        printing of real*8 numbers on the graphics page
C                        specified by user.
C *************************************************************************
      subroutine prtr8(pg,x,y,flen,form,width,nr8)
      integer*2 pg,x,y,type /4/,flen,width,ni2 /0/
      integer*4 ni4 /0/
      real*4 nr4 /0/
      real*8 nr8
      character*40 form

      character*40 prstr
      call numstr(type,flen,form,ni2,ni4,nr4,nr8,width,prstr)
      call WRTSTR(pg,x,y,width,prstr)
      return
      end
C *************************************************************************
CH    REVISION HISTORY  :
C     VERSION       BY                  DATE        COMMENT
C       1.1     A. de Waal            06/01/90   Finally Commented
C     FILEND          :
C *************************************************************************
C
C     FILE            : WRHEAD.FOR
C *************************************************************************
CN    MODULE NAME     : wrhead
CA    FUNCTION        : Write Heading on Graphics Page
CS    CALL SEQUENCE   : call wrhead(page,x,y,length,string)
CI    INPUT PARAMETERS : page - (integer*2) The page to which the
C                                          heading is to be written.
C                          x,y - (integer*2) The x,y co-ords of the title.
C                        length - (integer*2) The length of the title.
C                        string - (character*length) The title string.
```

```
CO     OUTPUT PARAMETERS: None.
CG     GLOBAL VARIABLES : None.
CM     MODULES CALLED   : Assembler:   DLINE,MOVE,WRTSTR
CE     ERROR CONDITIONS : None
CC     COMMENTS         : Writes out the string and the draws a line underneath
C                         the string for the length of the string.
C ************************************************************************************
      subroutine wrhead(pg,x,y,len,str)
      integer*2 pg,x,y,len
      character*80 str
      call WRTSTR(pg,x,y,len,str)
      call MOVE(x,y+3)
      call DLINE((len*9+x),y+3)
      return
      end
C ************************************************************************************
CH     REVISION HISTORY  :
C      VERSION        BY                  DATE         COMMENT
C        1.1      A. de Waal            06/01/90    Finally Commented
C      FILEND             :
C ************************************************************************************
+C
```

M5) Modified Subroutines Originally Written by Ian Fisher

A table of modified Fisher subroutine names follows on the next page. The name of each subroutine, together with the file in which it is located and the page in the appendix on which it is listed, is given in the table.

Modified Subroutines Originally Written by Ian Fisher			
Name	**File**	**Description**	**Page**
dodir	DODIR	To print out a directory list.	442
editm	EDTMAT.FOR	To edit a matrix of polynomials.	442
movcrs	EDTMAT.FOR	To move the matrix edit cursor.	443
dwsqrs	EDTMAT.FOR	To draw the matrix grid.	444
dwshap	EDTMAT.FOR	To draw a number of shapes on the matrix grid.	444
wrtitl	EDTMAT.FOR	To write a title on a page.	445
setblk	EDTMAT.FOR	To scan through the matrix and find all non-zero	445
upblk	EDTMAT.FOR	To update the blocks on the grid once a polynomial has	445
GETNUM	GETNUM.FOR	To input or edit a number on a graphics page.	446
gnum	GNUMS.FOR	To print graphic numbers.	447
QZVECA	MATH.FOR	Subroutine to calculate the eigen values and vectors	449
COMHES	MATH.FOR	This subroutine is a translation of the ALGOL	449
COMLR2	MATH.FOR	This subroutine is a translation of the ALGOL	450
NUMSTR	NUMSTR.FOR	To convert a number to a string.	454
prderr	PRDERR.FOR	To print an appropriate disk error message.	454
simopt	SIMOPT.FOR	Edit Time Simulation Parameters	455
simscl	SIMSCL.FOR	Edit Time Simulation Axes and Scales.	457
prscal	SIMSCL.FOR	Print Out Set of Scales and I/O Names.	457
edscal	SIMSCL.FOR	Edit the Axes Scales.	458
simul	SIMUL.FOR	Drive the Time Simulation Menu.	459
dosim	SIMUL.FOR	Perform the Actual Simulation.	459
simgrf	SIMUL.FOR	Plot All Time Simulation Axes	461
inisim	SIMUL.FOR	Initialise States and Arrays for Simulation	462
pltsim	SIMUL.FOR	Plot Outputs from Simulation.	463
pltsta	SIMUL.FOR	Plot States from Simulation	463
chkqt	SIMUL.FOR	Check if User wishes to Quit Simulation.	463

```
C
C     FILE              : DODIR
C ***************************************************************************
CN    MODULE NAME       : dodir
CA    FUNCTION          : To print out a directory list.
CS    CALL SEQUENCE     : call dodir(pg,ylimit,len,str)
CI    INPUT PARAMETERS  :     pg - (INTEGER*2) The page to which the directory
C                                              will be written.
C                         ylimit - (INTEGER*2) The upper Y limit on the screen.
C                            len - (INTEGER*2) Length of default search string.
C                            str - (CHARACTER*len) The search string.
CO    OUTPUT PARAMETERS: None.
CG    GLOBAL VARIABLES  : None.
CM    MODULES CALLED    : FFIRST (asm),  FNEXT (asm), WRTSTR (asm), GPAGE (asm),
C                         MOVE (asm),  DLINE (asm),  wrhead (for), LENSTR (asm),
C                         COPYST (asm), RSTERR (asm), GETERR (asm), prti2 (for)
C                         prderr (for).
CE    ERROR CONDITIONS  : ?
C
CC    COMMENTS          : Just prints out as many directory entries that can
C                         be found or fitted onto the screen.
C ***************************************************************************
      subroutine dodir(pg,ylimit,dirlen,str,path)
      implicit integer*2 (F,G,L)
      integer*2 pg,ylimit,dirlen
      character*25 str,path
      integer*2 len,len2,find,xpos,ypos,fcount,count2,pos,derr
      character*15 retstr
      character*25 namstr
      character*50 fstr

      call GPAGE(pg)
      call MOVE(4,ylimit)
      call DLINE(714,ylimit)
      call wrhead(pg,20,ylimit+15,15,'Search string :')
      call RJUST(dirlen,str)
      len = LENSTR(dirlen,str)
      namstr = '                         '
      call COPYST(25,namstr,0,len,str,0,len)
      call maknam(namstr,path,fstr)
      len = LENSTR(50,fstr)
      if (len.gt.36) then
         len2 = 36
      else
         len2 = len
      endif
      call WRTSTR(0,165,ylimit+15,len2,fstr)
      fcount = 0
      xpos = 20
      ypos = ylimit+32
      call RSTERR()
      find = FFIRST(len,fstr,len2,retstr)
      if (find.eq.0) then
102      continue
         fcount = fcount + 1
         call WRTSTR(pg,xpos,ypos,len2,retstr)
         xpos = xpos + 140
         if (xpos.gt.680) then
            xpos = 20
            ypos = ypos + 15
         endif
         if (ypos.gt.270) goto 104
         find = FNEXT(len2,retstr)
         if (find.eq.0) goto 102
104      continue
         count2 = fcount
106      continue
         find = FNEXT(len2,retstr)
         if (find.eq.0) count2 = count2 + 1
         if (find.eq.0) goto 106
         call prti2(0,495,ylimit+15,1,4,'(i2)',2,fcount)
         call WRTSTR(0,525,ylimit+15,10,'printed / ')
         call prti2(0,615,ylimit+15,1,4,'(i3)',3,count2)
         call WRTSTR(0,650,ylimit+15,6,'found.')
      else
         derr = GETERR()
         if (derr.ne.-1) then
            call prderr(pg,40,ylimit+40,derr)
         else
            call WRTSTR(pg,40,ylimit+40,15,'No files found.')
         endif
      endif
      return
      end

C ***************************************************************************
CH    REVISION HISTORY :
C     VERSION       BY                  DATE         COMMENT
C       1.00     Ian Fisher            23/06/88    Creation.
C       1.1      A. de Waal            06/01/90   Finally Commented
C     FILEND            :
C ***************************************************************************
C
C     FILE              : EDTMAT.FOR
C ***************************************************************************
CN    MODULE NAME       : editm
CA    FUNCTION          : To edit a matrix of polynomials.
CS    CALL SEQUENCE     : call editm(n,mat,mname,fname)
CI    INPUT PARAMETERS  : n - (integer*2)
C                         mat(15,15,2,13) - (real*4) The matrix to be edited.
C                         mname - (character*25) The matrix title.
C                         fname - (character*25) The matrix filename.
CO    OUTPUT PARAMETERS: None
CG    GLOBAL VARIABLES  : keys.inc
CM    MODULES CALLED    : dwshap (for), dwsqrs (for), ERTONE (asm), getnum (for)
C                         INKEY (asm), movcrs (for),  poly (for),  prtnum (for),
C                         setblk (for), STRIN (asm), upblk (for),  WIPSCR (asm),
C                         wrtitl (for), WRTSTR (asm).
CE    ERROR CONDITIONS  : ?
C
CC    COMMENTS          : Allows the user to edit  a mtrix of polynomials. Both
C                         the graphics screens are used for this function.
C                         On the first screen the matrix is shown as grid, each
C                         block representing  an  element (a polynomial) of the
C                         matrix. If the any item of the polynomial is non-zero,
C                         then a solid block is drawn in the grid square.
```

```
C                       A cursor is  also  present  on the grid : the user can
C                       move around  the  grid  using the cursor keys. The ESC
C                       key permits the user to  exit the facility. The RETURN
C                       key enables the  user to edit  the  element  where the
C                       cursor is at that time.
C                       The  polynomial  is  then  displayed  on  the   second
C                       graphics  page  and  permits  the user to edit all the
C                       relevant  information.  The ESC  key  returns the user
C                       to the matrix grid.
C
C ***************************************************************************
      subroutine editm(n,mname)
      implicit integer*2 (I,S,g)
      integer*2 n
      real*4 mat(15,15,2,13)
      character*25 mname
$include: 'keys.inc'
      integer*2 i,j,pos,key,nold
      common /edmat/ mat
      call GPAGE(1)
      call WIPSCR(1)
      call DISP(1)
      call wrtitl(1,480,65,7,'Title :')
      call WRTSTR(1,480,85,25,mname)
      if ((n.gt.0).and.(n.lt.16)) then
         call dwsqrs(n,1)
         call setblk(n,mat,1)
      else
         call WRTSTR(1,100,100,24,'Matrix too Large to Edit')
         call WRTSTR(1,100,100,24,'                        ')
         key = INKEY(1)
         return
      endif
      i = 1
      j = 1
70    continue
      if ((n.gt.0).and.(n.lt.21)) then
            call wrtitl(1,480,130,18,'Current position :')
            call dwshap(n,i,j,2,2)

102         continue
            call WRTSTR(1,485,150,1,'(')
            call prti2(1,500,150,4,'(i2)',2,i)
            call WRTSTR(1,525,150,1,',')
            call prti2(1,540,150,4,'(i2)',2,j)
            call WRTSTR(1,565,150,1,')')

            call wrtitl(1,480,200,22,'Matrix Element Value :')
            call prtr4(1,500,220,7,'(e15.3)',15,mat(i,j,1,1))

            call WRTSTR(1,480,270,24,'Use cursor keys to move.')
            call WRTSTR(1,480,285,21,'RETURN key to select.')
            call WRTSTR(1,480,305,16,'ESC key to exit.')
            key = INKEY(1)
            if (key.eq.retk) then
               call getr4(1,500,220,7,'(e15.3)',15,mat(i,j,1,1))
               call upblk(n,mat,i,j)
            elseif (key.eq.upk) then
               call movcrs(n,i,-1,j,0)
            elseif (key.eq.downk) then
               call movcrs(n,i,+1,j,0)
            elseif (key.eq.leftk) then
               call movcrs(n,i,0,j,-1)
            elseif (key.eq.rightk) then
               call movcrs(n,i,0,j,+1)
            elseif (key.eq.homek) then
               call movcrs(n,i,-1,j,-1)
            elseif (key.eq.endk) then
               call movcrs(n,i,+1,j,-1)
            elseif (key.eq.pgupk) then
               call movcrs(n,i,-1,j,+1)
            elseif (key.eq.pgdnk) then
               call movcrs(n,i,+1,j,+1)
            endif
            if ((key.ne.esck).and.(key.ne.tabk).and.(key.ne.rtabk))
     +         goto 102

            call dwshap(n,i,j,2,2)
            call WRTSTR(1,480,100,19,'                        ')
            call WRTSTR(1,480,120,19,'                        ')
            call WRTSTR(1,480,270,24,'Use tab keys and RETURN ')
            call WRTSTR(1,480,285,21,'key to select.          ')
            call WRTSTR(1,480,305,16,'ESC key to exit.')
      else
            call WRTSTR(1,100,100,24,'Matrix too Large to Edit')
            call WRTSTR(1,100,100,24,'                        ')
            key = INKEY(1)
            return
      endif
      if (key.ne.esck) goto 70
      call WIPSCR(1)
      return
      end

C ***************************************************************************
CN    MODULE NAME       : movcrs
CA    FUNCTION          : To move the matrix edit cursor.
CS    CALL SEQUENCE     : call movcrs(n,row,rowinc,col,colinc)
CI    INPUT PARAMETERS  :      n - (integer*2) Order of the system.
C                          row - (integer*2) Current row position.
C                       rowinc - (integer*2) Row increment.
C                          col - (integer*2) Current column position.
C                       colinc - (integer*2) Column increment.
CO    OUTPUT PARAMETERS: row,col - new values of cursor position.
CG    GLOBAL VARIABLES  : None.
CM    MODULES CALLED    : dwshap (for).
CE    ERROR CONDITIONS  : ?
C
CC    COMMENTS          : Blanks out the cursor in the old position and then
C                         adds the increments to the old position  pointers.
C                         The cursor is then drawn in the new position.
C
C ***************************************************************************
      subroutine movcrs(n,row,rowinc,col,colinc)
      integer*2 n,row,rowinc,col,colinc
      if (n.gt.1) then
```

```
         call dwshap(n,row,col,2,2)
         if (rowinc.ne.0) then
            row = row + rowinc
            if (row.eq.0) row = n
            if (row.gt.n) row = 1
         endif
         if (colinc.ne.0) then
            col = col + colinc
            if (col.eq.0) col = n
            if (col.gt.n) col = 1
         endif
         call dwshap(n,row,col,2,2)
      endif
      return
      end

C ***************************************************************************
CN    MODULE NAME       : dwsqrs
CA    FUNCTION          : To draw the matrix grid.
CS    CALL SEQUENCE     : call dwsqrs(n,level)
CI    INPUT PARAMETERS  :     n - (integer*2) The order of the system.
C                         level - (integer*2) The write intensity level.
CO    OUTPUT PARAMETERS: None.
CG    GLOBAL VARIABLES  : None.
CM    MODULES CALLED    : DLINE (asm), LEVEL (asm), MOVE (asm).
CE    ERROR CONDITIONS  : ?
C
CC    COMMENTS          : Sets the write intensity level, calculates the
C                         dimensions of the grid and then draws the grid.
C
C ***************************************************************************
      subroutine dwsqrs(n,lev)
      integer*2 n,lev
      integer*2 xcent,ycent,xblk,yblk,xcalc,ycalc,scale
      call LEVEL(lev)
      if (n.gt.9) then
         scale = 1
      else
         scale = 2
      endif
      xcent = 250
      ycent = 165
      xblk  = scale*22
      yblk  = scale*14

      xcalc = xcent - int (real(xblk)*real(n)/2.0)
      ycalc = ycent - int (real(yblk)*real(n)/2.0)
      do 99 i = 1,(n+1)
         call MOVE(xcalc,((i-1)*yblk+ycalc))
         call DLINE((xcalc+n*xblk),((i-1)*yblk+ycalc))
99    continue

      do 98 i = 1,(n+1)
         call MOVE(((i-1)*xblk+xcalc),ycalc)
         call DLINE(((i-1)*xblk+xcalc),(ycalc+n*yblk))
98    continue
      call LEVEL(1)
      return
      end

C ***************************************************************************
CN    MODULE NAME       : dwshap
CA    FUNCTION          : To draw a number of shapes on the matrix grid.
CS    CALL SEQUENCE     : call dwshap(n,row,col,shape,level)
CI    INPUT PARAMETERS  :     n - (integer*2) The order of the system.
C                       row,col - (integer*2) The row,col position of the shape.
C                         shape - (integer*2) The shape to be drawn on the grid:
C                                             1 - block
C                                             2 - cross (cursor)
C                                             3 - circle
C                         level - (integer*2) The intensity level at which the
C                                             shape is to be drawn.
CO    OUTPUT PARAMETERS: None.
CG    GLOBAL VARIABLES  : None.
CM    MODULES CALLED    : BLKFIL (asm), CIRC (asm), DLINE (asm), LEVEL (asm),
C                         MOVE (asm).
CE    ERROR CONDITIONS  : ?
C
CC    COMMENTS          : Draws a shape at the position and level specified
C                         on the matrix grid.
C
C ***************************************************************************
      subroutine dwshap(n,row,col,shape,lev)
      integer*2 n,row,col,shape,lev
      integer*2 xcent,ycent,xblk,yblk,xcalc,ycalc,scale
      if (n.gt.9) then
         scale = 1
      else
         scale = 2
      endif
      xcent = 250
      ycent = 165
      xblk  = scale*22
      yblk  = scale*14

      xcalc = xcent - int (real(xblk)*real(n)/2.0) + (col-1)*xblk
      ycalc = ycent - int (real(yblk)*real(n)/2.0) + row*yblk

      if (shape.eq.1) then
         call LEVEL(lev)
         call BLKFIL((xcalc+6),(ycalc-3),(xblk-12),(yblk-6))
         call LEVEL(1)
      elseif (shape.eq.2) then
         if (n.gt.1) then
            call LEVEL(lev)
            call MOVE((xcalc+int(real(xblk)/2.0)),ycalc)
            call DLINE((xcalc+int(real(xblk)/2.0)),(ycalc-yblk))
            call MOVE(xcalc,(ycalc-int(real(yblk)/2.0)))
            call DLINE((xcalc+xblk),(ycalc-int(real(yblk)/2.0)))
            call LEVEL(1)
         endif
      elseif (shape.eq.3) then
         call LEVEL(lev)
         call CIRC((xcalc+int(real(xblk)/2.0)),
     +             (ycalc+int(real(yblk)/2.0)),(int(real(yblk)/2.0)-1))
         call LEVEL(1)
```

```
      endif
      return
      end

C *************************************************************************
CN    MODULE NAME       : wrtitl
CA    FUNCTION          : To write a title on a page.
CS    CALL SEQUENCE     : call wrtitl(page,x,y,length,string)
CI    INPUT PARAMETERS  : page - (integer*2) The page to which the title is to
C                                            be written.
C                            x,y - (integer*2) The x,y co-ords of the title.
C                         length - (integer*2) The length of the title.
C                         string - (character*length) The title string.
CO    OUTPUT PARAMETERS: None.
CG    GLOBAL VARIABLES  : None.
CM    MODULES CALLED    : DLINE (asm), MOVE (asm), WRTSTR (asm).
CE    ERROR CONDITIONS  : ?
C
CC    COMMENTS          : Writes out the string and the draws a line underneath
C                         the string for the length of the string.
C
C *************************************************************************
      subroutine wrtitl(pg,x,y,len,str)
      integer*2 pg,x,y,len
      character*80 str
      call WRTSTR(pg,x,y,len,str)
      call MOVE(x,y+3)
      call DLINE((len*9+x),y+3)
      return
      end

C *************************************************************************
CN    MODULE NAME       : setblk
CA    FUNCTION          : To scan through the matrix and find all non-zero
C                         polynomial elements.
CS    CALL SEQUENCE     : call setblk(n,mat,level)
CI    INPUT PARAMETERS  : n - (integer*2) The order of the system.
C                         mat(15,15,2,13) - (real*4) The matrix being edited.
C                                   level - (integer*2) The level at which the
C                                                       blocks are to be drawn
C                                                       on the grid.
CO    OUTPUT PARAMETERS: None.
CG    GLOBAL VARIABLES  : None.
CM    MODULES CALLED    : dwshap (for).
CE    ERROR CONDITIONS  : ?
C
CC    COMMENTS          : Scans  through the  whole matrix, checking all the
C                         polynomial elements for non-zero items. If a poly-
C                         nomial has a non-zero item a block is drwan at the
C                         corresponding position on the grid.
C
C *************************************************************************
      subroutine setblk(n,mat,lev)
      integer*2 n
      real*4 mat(15,15,2,13)
      integer*2 lev
      integer*2 i,j,k,l,nonzer,shap
      shap = 1
      if ((n.gt.0).and.(n.lt.16)) then
         do 50 i = 1,n
            do 51 j = 1,n
               nonzer = 0
               k = int(mat(i,j,1,13))
               do 53 l = 0,k
                  if (abs(mat(i,j,1,(l+1))).gt.(1.0e-8)) then
                    nonzer = 1
                  endif
53             continue
               k = int(mat(i,j,2,13))
               do 54 l = 0,k
                  if (abs(mat(i,j,2,(l+1))).gt.(1.0e-8)) then
                    nonzer = 1
                  endif
54             continue
               if (nonzer.eq.1) then
                  call dwshap(n,i,j,shap,lev)
               endif
51          continue
50       continue
      endif
      return
      end

C *************************************************************************
CN    MODULE NAME       : upblk
CA    FUNCTION          : To update the blocks on the grid once a polynomial has
C                         been edited.
CS    CALL SEQUENCE     : call upblk(n,mat,row,col)
CI    INPUT PARAMETERS  : n - (integer*2) The order of the system.
C                         mat(15,15,2,13) - (real*4) The matrix being edited.
C                                 row,col - (integer*2) The polynomial position
C                                                       in the matrix.
CO    OUTPUT PARAMETERS: None.
CG    GLOBAL VARIABLES  : None.
CM    MODULES CALLED    : dwshap (for).
CE    ERROR CONDITIONS  : ?
C
CC    COMMENTS          : Checks  the  elements of the polynomial just edited,
C                         checking  for  non-zero  elements. If  any  non-zero
C                         elements are found then a block is drawn on the grid.
C
C *************************************************************************
      subroutine upblk(n,mat,row,col)
      integer*2 n
      real*4 mat(15,15,2,13)
      integer*2 row,col
      integer*2 k,l,nonzer,shap,cros,lev0,lev1,lev2
      shap = 1
      cros = 2
      lev0 = 0
      lev1 = 1
      lev2 = 2
      if ((n.gt.0).and.(n.lt.16)) then
         nonzer = 0
         k = int(mat(row,col,1,13))
         do 53 l = 0,k
```

```
            if (abs(mat(row,col,1,(l+1))).gt.(1.0e-8)) then
               nonzer = 1
            endif
53          continue
            k = int(mat(row,col,2,13))
            do 54 l = 0,k
               if (abs(mat(row,col,2,(l+1))).gt.(1.0e-8)) then
                  nonzer = 1
               endif
54          continue
            if (nonzer.eq.1) then
               call dwshap(n,row,col,cros,lev2)
               call dwshap(n,row,col,shap,lev1)
               call dwshap(n,row,col,cros,lev2)
            else
               call dwshap(n,row,col,cros,lev2)
               call dwshap(n,row,col,shap,lev0)
               call dwshap(n,row,col,cros,lev2)
            endif
         endif
         return
         end

C ***************************************************************************
CH       REVISION HISTORY :
C        VERSION        BY                  DATE         COMMENT
C        1.00           Ian Fisher          23/06/88     Creation.
C        FILEND         :
C ***************************************************************************
C
C        FILE           : GETNUM.FOR
C ***************************************************************************
CN       MODULE NAME    : GETNUM
CA       FUNCTION       : To input or edit a number on a graphics page.
CS       CALL SEQUENCE  : key = getnum(page,x,y,type,flen,format,width,
C                                    ni2,ni4,nr4,nr8)
CI       INPUT PARAMETERS : key  - (integer*2) The returned key from the routine.
C                           page - (integer*2) The page on which the number is to
C                                              be printed. (range : 0 or 1)
C                           x,y  - (integer*2) The co-ordinates of the printed
C                                              number.
C                           type - (integer*2) The type of number to be output:
C                                              type = 1 : integer*2
C                                              type = 2 : integer*4
C                                              type = 3 : real*4
C                                              type = 4 : real*8
C                           flen - (integer*2) Length of the format string:format.
C                           format - (character*flen) The number format string.
C                                                     (eg. '(f10.4)' flen = 7)
C                           width - (integer*2) The width of the field to be
C                                               printed.
C                           ni2 - (integer*2) The number to be printed.
C                           ni4 - (integer*4) The number to be printed.
C                           nr4 - (real*4) The number to be printed.
C                           nr8 - (real*8) The number to be printed.
C
CO       OUTPUT PARAMETERS: Edited number : number
C                           Returned key from the editor : key.
CG       GLOBAL VARIABLES : None.
CM       MODULES CALLED   : BLKFIL (asm), INKEY (asm), LEVEL (asm),  NOBLNK (asm),
C                           STRIN (asm), STRCPY (asm), WRTSTR (asm), numstr (for),
C                           prtnum (for), ERTONE (asm).
CE       ERROR CONDITIONS : The format string must not be greater than 40 chars.
C                           The number must not result in a string greater than
C                           40 chars.
CC       COMMENTS         : Inputs a number  from  the  page  and  the  co-ords
C                           specified. The number is treated as a string  while
C                           being edited. The string length being specified  by
C                           the given number width.
C                           The format of the number is available to  the  user
C                           through the use of the F2 key. When the F2  key  is
C                           hit a second time, the user can resume editing  the
C                           string.
C                           If the ESC key is hit during  editing, the  routine
C                           resets the number to the initial  value and returns
C                           control to the calling routine.
C ***************************************************************************
      integer*2 function getnum(pg,x,y,type,flen,form,width,
     +                          ni2,ni4,nr4,nr8)
      implicit integer*2 (I,S)

      integer*2 pg,x,y,type,flen
      character*40 form
      integer*2 width

      character*40 edtstr,edtst2,form2
      real*8 tempr8, nr8
      real*4 tempr4, nr4
      integer*4 tempi4, ni4
      integer*2 tempi2,lstrt, ni2
      integer*2 key,ctrla,f2key,repflg,twid,chkio

      ctrla = 0256
      f2key = 60
      chkio = 0
      escflg = 0

      call numstr(type,flen,form,ni2,ni4,nr4,nr8,width,edtstr)
      edtst2 = edtstr
101   continue
      repflg = 0
      call NOBLNK(width,edtstr)
      call RJUST(width,edtstr)
102   key = STRIN(pg,x,y,width,edtstr)
      call NOBLNK(width,edtstr)
      lstrt = 40
      call STRCPY(lstrt,form2,flen,form)
      if (key.eq.ctrla) then
         edtstr = edtst2
         repflg = 1
      elseif (key.eq.f2key) then
         if (width.ge.flen) then
            call WRTSTR(pg,x,y,width,form2)
            twid = width*9
            call LEVEL(2)
            call BLKFIL(x,y+2,twid,13)
```

```
            call LEVEL(1)
100         continue
            if (INKEY(1).ne.f2key) goto 100
         endif
         repflg = 1
      else
         escflg = 0
         if (type.eq.1) then
            read(edtstr,form2,iostat=chkio,end=103,err=103) ni2
            call prti2(pg,x,y,flen,form,width,ni2)
         elseif (type.eq.2) then
            read(edtstr,form2,iostat=chkio,end=103,err=103) ni4
            call prti4(pg,x,y,flen,form,width,ni4)
         elseif (type.eq.3) then
            read(edtstr,form2,iostat=chkio,end=103,err=103) nr4
            call prtr4(pg,x,y,flen,form,width,nr4)
         elseif (type.eq.4) then
            read(edtstr,form2,iostat=chkio,end=103,err=103) nr8
            call prtr8(pg,x,y,flen,form,width,nr8)
         endif
      endif
      if (repflg.eq.1) goto 101
      getnum = key
      if (chkio.eq.0) then
         return
      endif
103   call ERTONE()
      goto 102
      return
      end
C ***************************************************************************
CH    REVISION HISTORY  :
C     VERSION       BY                  DATE        COMMENT
C       1.0     Ian Fisher              ?       Creation and Commenting
C       1.1     A. de Waal           06/01/90   Modification and Commenting
C     FILEND            :
C ***************************************************************************
C$include: 'lst.inc'
C
C     FILE              : GNUMS.FOR
C ***************************************************************************
CN    MODULE NAME       : gnum
CA    FUNCTION          : To print graphic numbers.
CS    CALL SEQUENCE     : call gnum(page,x,y,num)
CI    INPUT PARAMETERS  :
C
C                         page - (integer*2) Page to which the numbers are to
C                                            be printed.
C                          x,y - (integer*2) X,y co-ords of number.
C                          num - (character*1) The number to be printed.
CO    OUTPUT PARAMETERS: None.
CG    GLOBAL VARIABLES  : None.
CM    MODULES CALLED    : GPAGE (asm), MOVE (asm), DLINE (asm), PLOT (asm).
CE    ERROR CONDITIONS  : ?
C
CC    COMMENTS          : The routine draws the number or letter on the graphics
C                         screen. The advatages of this is smaller characters
C                         and the characters do not erase a character block
C                         while being written.
C                         The following characters can be written :-
C                         0,1,2,3,4,5,6,7,8,9,.,-,s,S,char(233)
C
C ***************************************************************************
      subroutine gnum(pg,x,y,num)
      integer*2 pg,x,y
      character*1 num
      integer*2 xw,yh,xtemp,ytemp

      xw = 5
      yh = 6
      call GPAGE(pg)
      if (num.eq.' ')then
      elseif (num.eq.'0')then
         call MOVE(x+1,y)
         call DLINE(x,y-1)
         call DLINE(x,y-yh+1)
         call DLINE(x+1,y-yh)
         call DLINE(x+xw-1,y-yh)
         call DLINE(x+xw,y-yh+1)
         call DLINE(x+xw,y-1)
         call DLINE(x+xw-1,y)
         call DLINE(x+1,y)
         call MOVE(x,y-1)
         call DLINE(x+xw,y-yh+1)
      elseif (num.eq.'1')then
         xtemp = int(real(xw)/2.0)
         call MOVE(x+xtemp-1,y)
         call DLINE(x+xtemp+1,y)
         call MOVE(xtemp+x,y)
         call DLINE(xtemp+x,y-yh)
         call DLINE(xtemp+x-1,y-yh+1)
      elseif (num.eq.'2')then
         call MOVE(x+xw,y)
         call DLINE(x,y)
         call DLINE(x+xw,y-yh+2)
         call DLINE(x+xw,y-yh+1)
         call DLINE(x+xw-1,y-yh)
         call DLINE(x+1,y-yh)
         call DLINE(x,y-yh+1)
      elseif (num.eq.'3')then
         call MOVE(x,y-1)
         call DLINE(x+1,y)
         call DLINE(x+xw-1,y)
         call DLINE(x+xw,y-1)
         ytemp = int(real(yh)/2.0)
         call DLINE(x+xw,y-ytemp+1)
         call DLINE(x+xw-1,y-ytemp)
         call MOVE(x+2,y-ytemp)
         call DLINE(x+xw-1,y-ytemp)
         call DLINE(x+xw,y-ytemp-1)
         call DLINE(x+xw,y-yh+1)
         call DLINE(x+xw-1,y-yh)
         call DLINE(x+1,y-yh)
         call DLINE(x,y-yh+1)
      elseif (num.eq.'4')then
         xtemp = int(real(xw)/2.0)+2
```

```
      ytemp = int(real(yh)/2.0)
      call MOVE(x+xtemp,y)
      call DLINE(x+xtemp,y-yh)
      call DLINE(x,y-ytemp+1)
      call DLINE(x+xw,y-ytemp+1)
   elseif (num.eq.'5')then
      ytemp = int(real(yh)/2.0)
      call MOVE(x,y-1)
      call DLINE(x+1,y)
      call DLINE(x+xw-1,y)
      call DLINE(x+xw,y-1)
      call DLINE(x+xw,y-ytemp+1)
      call DLINE(x+xw-1,y-ytemp)
      call DLINE(x,y-ytemp)
      call DLINE(x,y-yh)
      call DLINE(x+xw,y-yh)
   elseif (num.eq.'6')then
      xtemp = int(real(xw)/2.0)
      ytemp = int(real(yh)/2.0)
      call MOVE(x+xtemp+1,y-yh)
      call DLINE(x+xtemp,y-yh)
      call DLINE(x,y-ytemp-1)
      call DLINE(x,y-1)
      call DLINE(x+1,y)
      call DLINE(x+xw-1,y)
      call DLINE(x+xw,y-1)
      call DLINE(x+xw,y-ytemp+1)
      call DLINE(x+xw-1,y-ytemp)
      call DLINE(x,y-ytemp)
   elseif (num.eq.'7')then
      xtemp = int(real(xw)/2.0)
      ytemp = int(real(yh)/2.0)
      call MOVE(x,y-yh+1)
      call DLINE(x,y-yh)
      call DLINE(x+xw,y-yh)
      call DLINE(x+xw,y-yh+2)
      call DLINE(x+xtemp,y-ytemp+1)
      call DLINE(x+xtemp,y)
   elseif (num.eq.'8')then
      ytemp = int(real(yh)/2.0)
      call MOVE(x+1,y)
      call DLINE(x+xw-1,y)
      call DLINE(x+xw,y-1)
      call DLINE(x+xw,y-ytemp+1)
      call DLINE(x+xw-1,y-ytemp)
      call DLINE(x+1,y-ytemp)
      call DLINE(x,y-ytemp-1)
      call DLINE(x,y-yh+1)
      call DLINE(x+1,y-yh)
      call DLINE(x+xw-1,y-yh)
      call DLINE(x+xw,y-yh+1)
      call DLINE(x+xw,y-ytemp-1)
      call DLINE(x+xw-1,y-ytemp)
      call MOVE(x+1,y-ytemp)
      call DLINE(x,y-ytemp+1)
      call DLINE(x,y-1)
      call DLINE(x+1,y)
   elseif (num.eq.'9')then
      ytemp = int(real(yh)/2.0)
      call MOVE(x+1,y)
      call DLINE(x+2,y)
      call DLINE(x+xw,y-ytemp+2)
      call DLINE(x+xw,y-yh+1)
      call DLINE(x+xw-1,y-yh)
      call DLINE(x+1,y-yh)
      call DLINE(x,y-yh+1)
      call DLINE(x+1,y-ytemp)
      call DLINE(x+xw,y-ytemp)
   elseif (num.eq.'-') then
      ytemp = int(real(yh)/2.0)
      call MOVE(x+1,y-ytemp)
      call DLINE(x+xw-1,y-ytemp)
   elseif ((num.eq.'s').or.(num.eq.'S')) then
      call MOVE(x,y-1)
      call DLINE(x+1,y)
      call DLINE(x+xw-1,y)
      call DLINE(x+xw,y-1)
      call DLINE(x,y-yh+1)
      call DLINE(x+1,y-yh)
      call DLINE(x+xw-1,y-yh)
      call DLINE(x+xw,y-yh+1)
   elseif ((num.eq.'e').or.(num.eq.'E')) then
      xtemp = int(real(xw)/2.0)
      ytemp = int(real(yh)/2.0)
      call MOVE(x+xw,y-1)
      call DLINE(x+xw,y)
      call DLINE(x,y)
      call DLINE(x,y-yh)
      call DLINE(x+xw,y-yh)
      call DLINE(x+xw,y-yh+1)
      call MOVE(x,y-ytemp)
      call DLINE(x+xtemp,y-ytemp)
   elseif (num.eq.char(233)) then
      call MOVE(x+2,y)
      call DLINE(x,y-2)
      call DLINE(x,y-yh+2)
      call DLINE(x+2,y-yh)
      call DLINE(x+xw-2,y-yh)
      call DLINE(x+xw,y-yh+2)
      call DLINE(x+xw,y-2)
      call DLINE(x+xw-2,y)
      call DLINE(x+2,y)
      ytemp = int(real(yh)/2.0)
      call MOVE(x,y-ytemp)
      call DLINE(x+xw,y-ytemp)
   else
      xtemp = int(real(xw)/2.0)
      call PLOT(x+xtemp,y)
      call PLOT(x+xtemp+1,y)
      call PLOT(x+xtemp,y-1)
      call PLOT(x+xtemp+1,y-1)
   endif
   return
   end
```

```
C ***************************************************************************
CH    REVISION HISTORY :
C     VERSION        BY                    DATE        COMMENT
C       1.00     Ian Fisher             23/06/88    Creation.
C       1.1      A. de Waal             06/01/90   No Changes.
C     FILEND           :
C ***************************************************************************
c  $include: 'lst.inc'
C
C     FILE             : MATH.FOR
C ***************************************************************************
CN    MODULE NAME      : QZVECA
CA    FUNCTION         : Subroutine to calculate the eigen values and vectors
C                        of the problem :-     A*x = lambda*x
C                        Where A is a square matrix of complex numbers.
C
CS    CALL SEQUENCE    : call qzveca(n,ar,ai,alfr,alfi,zr,zi,ierr)
CI    INPUT PARAMETERS :       n - (integer*2) Order of the matices.
C                        ar(n,n) - (real*4) The real part of the matrix : A.
C                        ai(n,n) - (real*4) The complex part of the matrix : A.
C
CO    OUTPUT PARAMETERS: alfr(n) - (real*4) The real part of the eigenvalues.
C                        alfi(n) - (real*4) The  imaginary part of the eigen-
C                                           values.
C                        zr(n,n) - (real*4) The real part of the eigenvectors.
C                        zi(n,n) - (real*4) The imaginary part of the eigen-
C                                           vectors.
C                           ierr - (integer*2) The   number   of   iterations
C                                              performed if the algorithm did
C                                              NOT converge.
C                                              Else ierr = 0, if the algorithm
C                                              did converge.
C
CG    GLOBAL VARIABLES : None.
CM    MODULES CALLED   : COMHES (for), COMLR2 (for).
CE    ERROR CONDITIONS : If n > 15 : max. size of arrays expected.
CC    COMMENTS         : The  routine  uses  the  routines for the generalized
C                        eigen-value and -vector problem to  solve  the
C                        problem :  A*x = lambda*x
C
C                  NOTE:  The original matrices, 'ar' and 'ai' are destroyed.
C
C ***************************************************************************
      subroutine qzveca(n,ar,ai,alfr,alfi,zr,zi,ierr)
      integer*2 n
      real*4 ar(n,n),ai(n,n),alfr(n),alfi(n),zr(n,n),zi(n,n)
      integer*2 ierr

      integer*2 swaps(30)

      call COMHES(n,1,n,ar,ai,swaps)
      call COMLR2(n,1,n,swaps,ar,ai,alfr,alfi,zr,zi,ierr,1)

      return
      end

C ***************************************************************************
CN    MODULE NAME      : COMHES
CA    FUNCTION         : This subroutine is a translation of the ALGOL
C                        procedure COMHES, Num. Math. 12, 349-368(1968) by
C                        Martin and Wilkinson. Handbook for Auto. Comp.,
C                        Vol.ii-linear algebra, 339-358(1971).
CS    CALL SEQUENCE    : call comhes(n,low,igh,ar,ai,int)
CI    INPUT PARAMETERS :
C                     n  - (integer*2) Order of the matrix.
C                  low
C                  igh  - (integer*2) Integers determined by the balancing
C                          subroutine  CBAL. if CBAL has not been used, set
C                          low=1, igh=n.
C              ar(n,n)
C              ai(n,n) - (real*4) Contain the real and imaginary parts,
C                        respectively, of the complex input matrix.
C
CO    OUTPUT PARAMETERS:
C              ar(n,n)
C              ai(n,n) - (real*4) Contain the  real  and imaginary parts,
C                        of the hessenberg matrix. The  multipliers which
C                        were used in the reduction  are  stored  in  the
C                        remaining triangles under the Hessenberg matrix.
C                  int - (integer*2) Contains information on the rows and
C                        columns  interchanged  in  the  reduction.  Only
C                        elements low through igh are used.
C
CG    GLOBAL VARIABLES : None.
CM    MODULES CALLED   :
CE    ERROR CONDITIONS : ?
C
CC    COMMENTS         : Given a complex general matrix, this subroutine
C                        reduces a submatrix situated in rows and columns
C                        low through igh to upper Hessenberg form by
C                        stabilized elementary similarity transformations.
C
C                        Arithmetic is real except for the replacement of
C                        the ALGOL procedure CDIV by complex division.
C
C                        Questions and comments should be directed to :-
C                        B. S. GARBOW,
C                        APPLIED MATHEMATICS DIVISION,
C                        ARGONNE NATIONAL LABORATORY.
C ***************************************************************************
C
      SUBROUTINE COMHES(N,LOW,IGH,AR,AI,INT)
      INTEGER*2 I,J,M,N,LA,IGH,KP1,LOW,MM1,MP1
      REAL*4 AR(N,N),AI(N,N)
      REAL*4 XR,XI,YR,YI
C     REAL*4 ABS
      INTEGER*2 INT(IGH)
      COMPLEX X,Y
      REAL*4 T1(2),T2(2)
      EQUIVALENCE (X,T1(1),XR),(T1(2),XI),(Y,T2(1),YR),(T2(2),YI)
C
      LA = IGH - 1
      KP1 = LOW + 1
      IF(LA .LT. KP1) GO TO 200
C
      IF(KP1.GT.LA) GO TO 10000
```

```
      DO 180 M = KP1, LA
      MM1 = M - 1
      XR = 0.0
      XI = 0.0
      I = M
C
      IF(M.GT.IGH) GO TO 10010
      DO 100 J = M, IGH
      IF (ABS(AR(J,MM1)) + ABS(AI(J,MM1))
     C.LE. ABS(XR) + ABS(XI)) GO TO 100
      XR = AR(J,MM1)
      XI = AI(J,MM1)
      I = J
  100 CONTINUE
10010 CONTINUE
C
      INT(M) = I
      IF (I .EQ. M) GO TO 130
C     ********** INTERCHANGE ROWS AND COLUMNS OF AR AND AI **********
      IF(MM1.GT.N) GO TO 10020
      DO 110 J = MM1, N
      YR = AR(I,J)
      AR(I,J) = AR(M,J)
      AR(M,J) = YR
      YI = AI(I,J)
      AI(I,J) = AI(M,J)
      AI(M,J) = YI
  110 CONTINUE
10020 CONTINUE
C
      IF(1.GT.IGH) GO TO 10030
      DO 120 J = 1, IGH
      YR = AR(J,I)
      AR(J,I) = AR(J,M)
      AR(J,M) = YR
      YI = AI(J,I)
      AI(J,I) = AI(J,M)
      AI(J,M) = YI
  120 CONTINUE
10030 CONTINUE
C     **********END INTERCHANGE **********
  130 IF (XR .EQ. 0.0 .AND. XI .EQ. 0.0) GO TO 180
      MP1 = M + 1
C
      IF(MP1.GT.IGH) GO TO 10040
      DO 160 I = MP1, IGH
      YR = AR(I,MM1)
      YI = AI(I,MM1)
      IF (YR .EQ. 0.0 .AND. YI .EQ. 0.0) GO TO 160
      Y = Y / X
      AR(I,MM1) = YR
      AI(I,MM1) = YI
C
      IF(M.GT.N) GO TO 10050
      DO 140 J = M, N
      AR(I,J) = AR(I,J) - YR * AR(M,J) + YI * AI(M,J)
      AI(I,J) = AI(I,J) - YR* AI(M,J) - YI* AR(M,J)
  140 CONTINUE
10050 CONTINUE
C
      IF(1.GT.IGH) GO TO 10060
      DO 150 J = 1, IGH
      AR(J,M) = AR(J,M) + YR * AR(J,I) - YI * AI(J,I)
      AI(J,M) = AI(J,M) + YR * AI(J,I) + YI * AR(J,I)
  150 CONTINUE
10060 CONTINUE
  160 CONTINUE
10040 CONTINUE
  180 CONTINUE
10000 CONTINUE
  200 RETURN
      END

C ******************************************************************************
CN    MODULE NAME       : COMLR2
CA    FUNCTION          : This subroutine is a translation of the ALGOL
C                         procedure COMLR2,  Num. Math. 16, 181-204(1970) by
C                         Peters and Wilkinson. Handbook for Auto. Comp.,
C                         Vol.ii-linear algebra, 372-395(1971).
CS    CALL SEQUENCE     : call comlr2(n,low,igh,int,hr,hi,wr,wi,zr,zi,
C                                     ierr,nobak)
CI    INPUT PARAMETERS :
C                   n - (integer*2) Order of the matrix.
C               nobak - (integer*2) Extra  parameter which  tells if  the user
C                       wants the eigenvectors (=1) or just the transformation
C                       matrices (=0).
C                 low
C                 igh - (integer*2)  Integers  detrmined  by  the  balancing
C                       subroutine  CBAL.  If  CBAL  has  not  been  used, set
C                       low=1, igh=n.
C                 int - (integer*2)  Contains  information  on  the  rows  and
C                       columns  interchanged  in  the  reduction  by  COMHES,
C                       if  performed.  Only  elements  low  through  igh  are
C                       used.  If  the  eigenvectors  of  the Hessenberg matrix
C                       are desired, set int(j)=j for these elements.
C             hr(n,n)
C             hi(n,n) -  (real*4)  Contains  the  real  and  imaginary  parts,
C                        respectively, of the complex upper Hessenberg matrix.
C                        Their  lower triangles below the subdiagonal contains
C                        the  multipliers  which were used in the reduction by
C                        COMHES,if performed.
C                        If the eigenvectors  of  the  Hessenberg  matrix  are
C                        desired, these elements must be set to zero.
C
CO    OUTPUT PARAMETERS:
C                       The  upper  Hessenberg  portions  of hr and hi have been
C                       destroyed.
C               wr(n)
C               wi(n) - (real*4)  Contains   the  real  and  imaginary  parts,
C                       respectively,  of   the eigenvalues.  If  an  error
C                       exit  is  made,  the  eigenvalues  should  be  correct
C                       for indices ierr+1,...,n.
C             zr(n,n)
C             zi(n,n) - (real*4)  Contains  the  real  and  imaginary  parts,
C                       respectively,  of  the  eigenvectors.  The eigenvectors
C                       are unnormalized. If  an  error  exit is made, none of
```

```
C                       the eigenvectors has been found.
C                ierr - (integer*2) Set to : 0 - for normal return,
C                                            J - if  the  J-th  eigenvalue  has
C                                                not been determined  after  30
C                                                iterations.
CG    GLOBAL VARIABLES : None.
CM    MODULES CALLED   :
CE    ERROR CONDITIONS : ?
C
CC    COMMENTS         : This subroutine finds the eigenvalues and eigenvectors
C                        of a complex upper Hessenberg matrix  by  the modified
C                        LR  method.  The  eigenvectors  of  a  complex general
C                        matrix can  also be  found  if COMHES has been used to
C                        reduce this general matrix to hessenberg form.
C
C                        Arithmetic is real  except  for the replacement of the
C                        ALGOL procedure  CDIV  by  complex division and use of
C                        the subroutines CSQRT  and  CMPLX in computing complex
C                        square roots.
C
C                        Questions and comments should be directed to :-
C                        B. S. GARBOW,
C                        APPLIED MATHEMATICS DIVISION,
C                        ARGONNE NATIONAL LABORATORY
C ********************************************************************************
C
      SUBROUTINE  COMLR2 (N,LOW,IGH,INT,HR,HI,WR,WI,ZR,ZI,IERR,NOBAK)
C
      INTEGER*2 I,J,K,L,M,N,EN,II,JJ,LL,MM,NN,IGH,IM1,IP1,
     +          ITS,LOW,MP1,ENM1,IEND,IERR
      INTEGER*2  NOBAK
      REAL*4 HR(N,N),HI(N,N),WR(N),WI(N),ZR(N,N),ZI(N,N)
      REAL*4 SI,SR,TI,TR,XI,XR,YI,YR,ZZI,ZZR,NORM,MACHEP
C     REAL*4 ABS
      INTEGER*2 INT(IGH)
C     INTEGER*2 MINO
      COMPLEX X,Y,Z
C     COMPLEX CSQRT,CMPLX
      REAL*4 T1(2),T2(2),T3(2)
      EQUIVALENCE (X,T1(1),XR),(T1(2),XI),(Y,T2(1),YR),(T2(2),YI),
     +            (Z,T3(1),ZZR),(T3(2),ZZI)
C     ------------------------------------------------------------------
C     ************ MACHEP IS A MACHINE DEPENDENT PARAMETER SPECIFYING
C                  THE RELATIVE PRECISION OF FLOATING POINT ARITHMETIC.
      MACHEP=2.0**(-15)
      IERR = 0
C     ********** INITIALIZE EIGENVECTOR MATRIX **********
      IF(1.GT.N) GO TO 10000
      DO 100 I = 1,N
C
      IF(1.GT.N) GO TO 10000
      DO 100 J = 1, N
      ZR(I,J) = 0.0
      ZI(I,J) = 0.0
      IF (I .EQ. J) ZR(I,J) = 1.0
  100 CONTINUE
10000 CONTINUE
C     ********** FORM THE MATRIX OF ACCUMULATED TRANSFORMATIONS
C                FROM THE INFORMATION LEFT BY COMHES **********
      IEND = IGH - LOW - 1
      IF (IEND .LE. 0) GO TO 180
C     ********** FOR I=IGH-1 STEP -1 UNTIL LOW+1 DO -- **********
      IF(1.GT.IEND) GO TO 10010
      DO 160 II = 1, IEND
      I = IGH - II
      IP1 = I + 1
C
      IF(IP1.GT.IGH) GO TO 10020
      DO 120 K = IP1, IGH
      ZR(K,I) = HR(K,I-1)
      ZI(K,I) = HI(K,I-1)
  120 CONTINUE
10020 CONTINUE
C
      J=INT(I)
      IF (I .EQ. J) GO TO 160
C
      IF(I.GT.IGH) GO TO 10030
      DO 140 K = I, IGH
      ZR(I,K) = ZR(J,K)
      ZI(I,K) = ZI(J,K)
      ZR(J,K) = 0.0
      ZI(J,K) = 0.0
  140 CONTINUE
10030 CONTINUE
C
      ZR(J,I) = 1.0
  160 CONTINUE
10010 CONTINUE
C     ********** STORE ROOTS ISOLATED BY CBAL **********
  180 IF(1.GT.N) GO TO 10040
      DO 200 I = 1, N
      IF (I .GE. LOW .AND. I .LE. IGH) GO TO 200
      WR(I) = HR(I,I)
      WI(I) = HI(I,I)
  200 CONTINUE
10040 CONTINUE
C
      EN = IGH
      TR = 0.0
      TI = 0.0
C     ********** SEARCH FOR NEXT EIGENVALUE **********
  220 IF (EN .LT. LOW) GO TO 680
      ITS = 0
      ENM1 = EN - 1
C     ********** LOOK FOR SINGLE SMALL SUB-DIAGONAL ELEMENT
C                FOR L=EN STEP -1 UNTIL LOW DO -- **********
  240 IF(LOW.GT.EN) GO TO 10050
      DO 260 LL = LOW, EN
      L = EN + LOW - LL
      IF (L .EQ. LOW) GO TO 300
      IF (ABS(HR(L,L-1)) + ABS(HI(L,L-1)) .LE.
     C    MACHEP * (ABS(HR(L-1,L-1)) + ABS(HI(L-1,L-1))
     C          + ABS(HR(L,L)) + ABS(HI(L,L)))) GO TO 300
  260 CONTINUE
10050 CONTINUE
```

```
C     ********** FORM SHIFT ************
  300 IF (L .EQ. EN) GO TO 660
      IF (ITS .EQ. 45) GO TO 1000
C      IF (ITS .EQ. 30) GO TO 1000
      IF (ITS .EQ. 15 .OR. ITS .EQ. 30) GO TO 320
C      IF (ITS .EQ. 10 .OR. ITS .EQ. 20) GO TO 320
      SR = HR(EN,EN)
      SI = HI(EN,EN)
      XR = HR(ENM1,EN) * HR(EN,ENM1) - HI(ENM1,EN) * HI(EN,ENM1)
      XI = HR(ENM1,EN) * HI(EN,ENM1) + HI(ENM1,EN) * HR(EN,ENM1)
      IF (XR .EQ. 0.0 .AND. XI .EQ. 0.0) GO TO 340
      YR = (HR(ENM1,ENM1) - SR) / 2.0
      YI = (HI(ENM1,ENM1) - SI) / 2.0
      Z = CSQRT(CMPLX(YR**2-YI**2+XR,2.0*YR*YI+XI))
      IF (YR * ZZR + YI * ZZI .LT. 0.0) Z = -Z
      X = X / (Y + Z)
      SR = SR - XR
      SI = SI - XI
      GO TO 340
C     ********** FORM EXCEPTIONAL SHIFT **********
  320 SR = ABS(HR(EN,ENM1)) + ABS(HR(ENM1,EN-2))
      SI = ABS(HI(EN,ENM1)) + ABS(HI(ENM1,EN-2))
C
  340 IF(LOW.GT.EN) GO TO 10060
      DO 360 I = LOW, EN
      HR(I,I) = HR(I,I) - SR
      HI(I,I) = HI(I,I) - SI
  360 CONTINUE
10060 CONTINUE
C
      TR = TR + SR
      TI = TI + SI
      ITS = ITS + 1
C     ********** LOOK FOR TWO CONSECUTIVE SMALL
C                SUB-DIAGONAL ELEMENTS **********
      XR = ABS(HR(ENM1,ENM1)) + ABS(HI(ENM1,ENM1))
      YR = ABS(HR(EN,ENM1)) + ABS(HI(EN,ENM1))
      ZZR = ABS(HR(EN,EN)) + ABS(HI(EN,EN))
C     ********** FOR M=EN-1 STEP -1 UNTIL L DO -- **********
      IF(L.GT.ENM1) GO TO 10070
      DO 380 MM = L, ENM1
      M = ENM1 + L - MM
      IF (M .EQ. L) GO TO 420
      YI = YR
      YR = ABS(HR(M,M-1)) + ABS(HI(M,M-1))
      XI = ZZR
      ZZR = XR
      XR = ABS(HR(M-1,M-1)) + ABS(HI(M-1,M-1))
      IF(YR .LE. MACHEP*ZZR/YI*(ZZR +XR +XI)) GO TO 420
  380 CONTINUE
10070 CONTINUE
C     ********** TRIANGULAR DECOMPOSITION H=L* R **********
  420 MP1 = M + 1
C
      IF(MP1.GT.EN) GO TO 10080
      DO 520 I = MP1, EN
      IM1 = I - 1
      XR = HR(IM1,IM1)
      XI = HI(IM1,IM1)
      YR = HR(I,IM1)
      YI = HI(I,IM1)
      IF (ABS(XR) + ABS(XI) .GE. ABS(YR) + ABS(YI)) GO TO 460
C     ********** INTERCHANGE ROWS OF HR AND HI **********
      IF(IM1.GT.N) GO TO 10090
      DO 440 J = IM1, N
      ZZR = HR(IM1,J)
      HR(IM1,J) = HR(I,J)
      HR(I,J) = ZZR
      ZZI= HI(IM1,J)
      HI(IM1,J) = HI(I,J)
      HI(I,J) = ZZI
  440 CONTINUE
10090 CONTINUE
C
      Z = X / Y
      WR(I) = 1.0
      GO TO 480
  460 Z = Y / X
      WR(I) = -1.0
  480 HR(I,IM1) = ZZR
      HI(I,IM1) = ZZI
C
      IF(I.GT.N) GO TO 10100
      DO 500 J = I, N
      HR(I,J) = HR(I,J) - ZZR * HR(IM1,J) + ZZI * HI(IM1,J)
      HI(I,J) = HI(I,J) - ZZR * HI(IM1,J) - ZZI * HR(IM1,J)
  500 CONTINUE
10100 CONTINUE
C
  520 CONTINUE
10080 CONTINUE
C     ********** COMPOSITION R*L=H **********
      IF(MP1.GT.EN) GO TO 10110
      DO 640 J = MP1, EN
      XR = HR(J,J-1)
      XI = HI(J,J-1)
      HR(J,J-1) = 0.0
      HI(J,J-1) = 0.0
C     ********** INTERCHANGE COLUMNS OF HR, HI, ZR, AND ZI,
C                IF NECESSARY **********
      IF (WR(J) .LE. 0.0) GO TO 580
C
      IF(1.GT.J) GO TO 10120
      DO 540 I = 1, J
      ZZR = HR(I,J-1)
      HR(I,J-1) = HR(I,J)
      HR(I,J) = ZZR
      ZZI = HI(I,J-1)
      HI(I,J-1)=HI(I,J)
      HI(I,J) = ZZI
  540 CONTINUE
10120 CONTINUE
C
      IF(LOW.GT.IGH) GO TO 10130
      DO 560 I = LOW, IGH
      ZZR = ZR(I,J-1)
```

```
      ZR(I,J-1) = ZR(I,J)
      ZR(I,J) = ZZR
      ZZI = ZI(I,J-1)
      ZI(I,J-1) = ZI(I,J)
      ZI(I,J) = ZZI
  560 CONTINUE
10130 CONTINUE
C
  580 IF(1.GT.J) GO TO 10140
      DO 600 I = 1, J
      HR(I,J-1) = HR(I,J-1) + XR * HR(I,J) - XI * HI(I,J)
      HI(I,J-1) = HI(I,J-1) + XR * HI(I,J) + XI * HR(I,J)
  600 CONTINUE
10140 CONTINUE
C     ********** ACCUMULATE TRANSFORMATIONS **********
      IF(LOW.GT.IGH) GO TO 10150
      DO 620 I = LOW, IGH
      ZR(I,J-1) = ZR(I,J-1) + XR * ZR(I,J) - XI * ZI(I,J)
      ZI(I,J-1) = ZI(I,J-1) + XR * ZI(I,J) + XI * ZR(I,J)
  620 CONTINUE
10150 CONTINUE
C
  640 CONTINUE
10110 CONTINUE
C
      GO TO 240
C     ********** A ROOT FOUND **********
  660 HR(EN,EN) = HR(EN,EN) + TR
      WR(EN) = HR(EN,EN)
      HI(EN,EN) = HI(EN,EN) + TI
      WI(EN) = HI(EN,EN)
      EN = ENM1
      GO TO 220
C     ********** ALL ROOTS FOUND. BACKSUBSTITUTE TO FIND
C                VECTORS OF UPPER TRIANGULAR FORM **********
  680 IF (N .EQ. 1) GO TO 1001
      IF(NOBAK.EQ.0) GO TO 1001
      NORM = 0.0
C
      IF(1.GT.N) GO TO 10160
      DO 720 I = 1, N
C
      IF(I.GT.N) GO TO 10160
      DO 720 J = I, N
      NORM = NORM + ABS(HR(I,J)) + ABS(HI(I,J))
  720 CONTINUE
10160 CONTINUE
C     ********** FOR EN=N STEP -1 UNTIL 2 DO -- **********
      IF(2.GT.N) GO TO 10170
      DO 800 NN = 2, N
      EN = N + 2 - NN
      XR = WR(EN)
      XI = WI(EN)
      ENM1 = EN - 1
C     ********** FOR I=EN-1 STEP -1 UNTIL 1 DO -- **********
      IF(1.GT.ENM1) GO TO 10180
      DO 780 II = 1, ENM1
      I = EN - II
      ZZR = HR(I,EN)
      ZZI = HI(I,EN)
      IF (I .EQ. ENM1) GO TO 760
      IP1 = I + 1
C
      IF(IP1.GT.ENM1) GO TO 10190
      DO 740 J = IP1, ENM1
      ZZR = ZZR + HR(I,J) * HR(J,EN) - HI(I,J) * HI(J,EN)
      ZZI = ZZI + HR(I,J) * HI(J,EN) + HI(I,J) * HR(J,EN)
  740 CONTINUE
10190 CONTINUE
C
  760 YR = XR - WR(I)
      YI = XI - WI(I)
      IF (YR .EQ. 0.0 .AND. YI .EQ. 0.0) YR = MACHEP * NORM
      Z = Z / Y
      HR(I,EN)= T3(1)
      HI(I,EN)= T3(2)
  780 CONTINUE
10180 CONTINUE
C
  800 CONTINUE
10170 CONTINUE
C     ********** END BACKSUBSTITUTION **********
      ENM1 = N - 1
C     ********** VECTORS OF ISOLATED ROOTS **********
      IF(1.GT.ENM1) GO TO 10200
      DO 840 I = 1, ENM1
      IF (I .GE. LOW .AND. I .LE. IGH) GO TO 840
      IP1 = I + 1
C
      IF(IP1.GT.N) GO TO 10210
      DO 820 J = IP1, N
      ZR(I,J) = HR(I,J)
      ZI(I,J) = HI(I,J)
  820 CONTINUE
10210 CONTINUE
C
840   CONTINUE
10200 CONTINUE
C     ********** MULTIPLY BY TRANSFORMATION MATRIX TO GIVE
C                VECTORS OF ORIGINAL FULL MATRIX.
C                FOR J=N STEP -1 UNTIL LOW+1 DO -- **********
      IF(LOW.GT.ENM1) GO TO 10220
      DO 880 JJ = LOW, ENM1
      J = N + LOW - JJ
      M = MINO(J-1,IGH)
C
      IF(LOW.GT.IGH) GO TO 10220
      DO 880 I = LOW, IGH
      ZZR = ZR(I,J)
      ZZI = ZI(I,J)
C
      IF(LOW.GT.M) GO TO 10230
      DO 860 K = LOW, M
      ZZR = ZZR + ZR(I,K) * HR(K,J) - ZI(I,K) * HI(K,J)
      ZZI = ZZI + ZR(I,K) * HI(K,J) + ZI(I,K) * HR(K,J)
  860 CONTINUE
```

```
10230 CONTINUE
C
      ZR(I,J) = ZZR
      ZI(I,J) = ZZI
  880 CONTINUE
10220 CONTINUE
      GO TO 1001
C     ********** SET ERROR -- NO CONVERGENCE TO AN
C                EIGENVALUE AFTER 30 ITERATIONS **********
1000  IERR = EN
1001  RETURN
      END
C ***************************************************************************
CH    REVISION HISTORY :
C     VERSION       BY                  DATE        COMMENT
C       1.00    Ian Fisher           23/06/88    Creation.
C       1.1     A. de Waal           06/01/90   Finally Commented
C     FILEND          :
C ***************************************************************************
C
C     FILE            : NUMSTR.FOR
C ***************************************************************************
CN    MODULE NAME     : NUMSTR
CA    FUNCTION        : To convert a number to a string.
CS    CALL SEQUENCE   : call numstr(type,lform,format,ni2,ni4,nr4,nr8,
C                                   length,string)
CI    INPUT PARAMETERS : type - (integer*2) The type of number to be output:
C                                           type = 1 : integer*2
C                                           type = 2 : integer*4
C                                           type = 3 : real*4
C                                           type = 4 : real*8
C                        lform - (integer*2) Length of the format string:format
C                        format - (character*flen) The number format string.
C                                                   (eg. '(f10.4)' flen = 7)
C                        ni2 - (integer*2) Number to be converted to string.
C                        ni4 - (integer*4) Number to be converted to string.
C                        nr4 - (real*4) Number to be converted to string.
C                        nr8 - (real*8) Number to be converted to string.
C                        length - (integer*2) Length of the string.
C                        string - (character*length) Destination string.
CO    OUTPUT PARAMETERS: Number in string format.
CG    GLOBAL VARIABLES : None.
CM    MODULES CALLED   : STRCPY (asm), ERTONE (asm)
CE    ERROR CONDITIONS : The format string must not be greater than 40 chars.
C                        The number must not result in a string greater than
C                        40 chars.
CC    COMMENTS         : The routine uses a device directed write statement
C                        to convert the number to a string according to the
C                        given format.
C ***************************************************************************
      subroutine numstr(ntype,lform,form,ni2,ni4,nr4,nr8,lstr,outstr)
      integer*2 ntype
      integer*2 lform
      character*40 form
      integer*2 lstr,lstrt,lformt
      character*40 outstr

      character*40 form2,out2
      real*8 temp
      integer*2 chkio

      integer*2 ni2
      integer*4 ni4
      real*4 nr4
      real*8 nr8

      chkio = 0

      lstrt = 40
      call STRCPY(lstrt,form2,lform,form)
      if (ntype.eq.1) then
         write(out2,form2,iostat=chkio,err=200) ni2
      elseif (ntype.eq.2) then
         write(out2,form2,iostat=chkio,err=200) ni4
      elseif (ntype.eq.3) then
         write(out2,form2,iostat=chkio,err=200) nr4
      else
         write(out2,form2,iostat=chkio,err=200) nr8
      endif
200   if (chkio.ne.0) then
         call ERTONE()
      endif
      lformt = 40
      call STRCPY(lstr,outstr,lformt,out2)
      return
      end
C ***************************************************************************
CH    REVISION HISTORY  :
C     VERSION       BY                  DATE        COMMENT
C       1.0     Ian Fisher             ?          Creation
C       1.1     A. de Waal             ?          Modified
C       1.1     A. de Waal           06/01/90   Finally Commented
C     FILEND          :
C ***************************************************************************
C
C     FILE                  : PRDERR.FOR
C ***************************************************************************
CN    MODULE NAME     : prderr
CA    FUNCTION        : To print an appropriate disk error message.
CS    CALL SEQUENCE   : call prderr(page,x,y,derr)
CI    INPUT PARAMETERS : page - (integer*2) The page to which the message is
C                                           to be written.
C                         x,y - (integer*2) The x,y co-ords of the message.
C                        derr - (integer*2) The disk error number.
CO    OUTPUT PARAMETERS: None.
CG    GLOBAL VARIABLES : None.
CM    MODULES CALLED   : ERTONE (asm), WRTSTR (asm).
CE    ERROR CONDITIONS : ?
C
CC    COMMENTS         : The routine interprets the disk drive error and then
C                        prints an  appropriate message at the user specified
C                        location.
C
C ***************************************************************************
      subroutine prderr(pg,x,y,derr)
      integer*2 pg,x,y,derr
```

```
      call ERTONE()
      if (derr.eq.0) then
         call WRTSTR(pg,x,y,33,'*** Error : Disk write protected.')
      elseif (derr.eq.1) then
         call WRTSTR(pg,x,y,35,'*** Error : Unknown unit specified.')
      elseif (derr.eq.2) then
         call WRTSTR(pg,x,y,33,'*** Error : Disk drive not ready.')
      elseif (derr.eq.3) then
         call WRTSTR(pg,x,y,38,'*** Error : Unknown command for drive.')
      elseif (derr.eq.4) then
         call WRTSTR(pg,x,y,35,'*** Error : Data CRC error on disk.')
      elseif (derr.eq.5) then
         call WRTSTR(pg,x,y,45,
     +              '*** Error : Bad request for structure length.')
      elseif (derr.eq.6) then
         call WRTSTR(pg,x,y,29,'*** Error : Drive seek error.')
      elseif (derr.eq.7) then
         call WRTSTR(pg,x,y,31,'*** Error : Unknown media type.')
      elseif (derr.eq.8) then
         call WRTSTR(pg,x,y,29,'*** Error : Sector not found.')
      elseif (derr.eq.10) then
         call WRTSTR(pg,x,y,29,'*** Error : Disk write fault.')
      elseif (derr.eq.11) then
         call WRTSTR(pg,x,y,28,'*** Error : Disk read fault.')
      elseif (derr.eq.12) then
         call WRTSTR(pg,x,y,33,'*** Error : General disk failure.')
      endif
      return
      end
C *****************************************************************************
CH    REVISION HISTORY :
C     VERSION       BY                DATE        COMMENT
C       1.00    Ian Fisher          23/06/88    Creation.
C       1.1     A. de Waal          06/01/90    Finally Commented
C     FILEND          :
C *****************************************************************************

C
C     FILE            : SIMOPT.FOR
C *****************************************************************************
CN    MODULE NAME     : simopt
CA    FUNCTION        : Edit Time Simulation Parameters
CS    CALL SEQUENCE   : call simopt()
CI    INPUT PARAMETERS : None.
CO    OUTPUT PARAMETERS: None.
CG    GLOBAL VARIABLES : Include files:time.inc,keys.inc,syst.inc
CM    MODULES CALLED  : Fortran  :   wrhead,prti2,prtr4,togopt,geti2,getr4
C                       Assembler:   WIPSCR,WRTSTR,INKEY,ERTONE
CE    ERROR CONDITIONS : As indicated on grahpics screen 0
CC    COMMENTS        : Prints  all   the  parameters  concerning  the  time
C                       simulation and enables the user to edit them. Bounds
C                       checking is  done  as  far as possible (ie. negative
C                       time, etc.).
C                       The ESC key returns control to the calling routine.
C *****************************************************************************
      subroutine simopt()
      implicit integer*2 (I,g,S)
$include: 'time.inc'
$include: 'keys.inc'
$include: 'syst.inc'
      integer*2 pos,pos2,key,ferr,i,key2,order
      real*4 temp1

      order = int(syst(4))
      pos = 1
      call WIPSCR(0)
      call wrhead(0,250,25,27,'Time Simulation Parameters.')
      call WRTSTR(0,40,40,25,'Kalman State Observer: No ')
      call WRTSTR(0,350,40,25,'State F/B Controller : No ')
      call WRTSTR(0,40,55,13,'Plotting Out:')
      call WRTSTR(0,200,55,41,
     +   'Setpoints: No   Inputs: No   Outputs: No ')
      call WRTSTR(0,200,70,41,
     +   'Process States: No  Observer States: No ')

      call WRTSTR(0,40,90,22,'Time duration [secs] :')
      call WRTSTR(0,40,110,22,'Time step     [secs] :')
      call wrhead(0,20,135,8,'Step No.')
      call wrhead(0,110,135,9,'Input No.')
      call wrhead(0,210,135,9,'Increment')
      call wrhead(0,320,135,11,'Time [secs]')
      call WRTSTR(0,430,135,13,'(after start)')
      if (obsinc.eq.0) call WRTSTR(0,40+23*9,40,3,'Yes')
      if (coninc.eq.0) call WRTSTR(0,350+23*9,40,3,'Yes')
      if (setplt.eq.0) call WRTSTR(0,200+11*9,55,3,'Yes')
      if (inpplt.eq.0) call WRTSTR(0,200+24*9,55,3,'Yes')
      if (outplt.eq.0) call WRTSTR(0,200+38*9,55,3,'Yes')
      if (ostplt.eq.0) call WRTSTR(0,200+38*9,70,3,'Yes')
      if (pstplt.eq.0) call WRTSTR(0,200+16*9,70,3,'Yes')
      call prtr4(0,240,90,7,'(g10.4)',10,tend)
      call prtr4(0,240,110,7,'(g10.4)',10,dt)
      do 888 i = 1,5
         call prti2(0,40,(140+i*15),4,'(i3)',3,i)
         call prti2(0,140,(140+i*15),4,'(i2)',2,stpinp(i))
         call prtr4(0,210,(140+i*15),7,'(g10.4)',10,stpinc(i))
         call prtr4(0,320,(140+i*15),7,'(g10.4)',10,stpdat(i))
888   continue

199   continue
      call WRTSTR(0,40,235,59,
     +'                                                                ')
      call WRTSTR(0,40,250,59,
     +'                                                                ')
      call WRTSTR(0,40,235,36,'RETURN, TAB and cursor keys to move.')
      call WRTSTR(0,40,250,16,'ESC key to exit.')
      if (pos.eq.1) then
          call togopt(obsinc,40+23*9,40,key)
      elseif (pos.eq.2) then
          call togopt(coninc,350+23*9,40,key)
      elseif (pos.eq.3) then
          call togopt(setplt,200+11*9,55,key)
      elseif (pos.eq.4) then
          call togopt(inpplt,200+24*9,55,key)
      elseif (pos.eq.5) then
          call togopt(outplt,200+38*9,55,key)
      elseif (pos.eq.6) then
```

```
         call togopt(pstplt,200+16*9,70,key)
      elseif (pos.eq.7) then
         call togopt(ostplt,200+38*9,70,key)
      elseif (pos.eq.8) then
197      continue
         key = getr4(0,240,90,7,'(g10.4)',10,tend)
         if (tend.lt.(0.0)) then
            call WRTSTR(0,350,90,31,'*** Error : Time duration < 0.0')
            call ERTONE()
         else
            call WRTSTR(0,350,90,31,'                                  ')
         endif
         if (tend.lt.(0.0)) goto 197
      elseif (pos.eq.9) then
195      continue
         key = getr4(0,240,110,7,'(g10.4)',10,dt)
         if (dt.lt.(0.0)) then
            call WRTSTR(0,350,110,27,'*** Error : Time step < 0.0')
            call ERTONE()
         else
            call WRTSTR(0,350,110,27,'                                 ')
         endif
         if (dt.lt.(0.0)) goto 195
      elseif (pos.ge.10) then
         pos2 = 1
         do 180 i = 10,22,3
            if ((i.ne.pos).and.(pos2.lt.100)) then
               pos2 = pos2 + 1
            elseif (i.eq.pos) then
               pos2 = pos2 + 100
            endif
180      continue
         if (pos2.gt.100) then
            pos2 = pos2 - 100
179         continue
            key = geti2(0,140,(140+pos2*15),4,'(i3)',3,stpinp(pos2))
            if ((stpinp(pos2).lt.0).or.(stpinp(pos2).gt.order)) then
               call WRTSTR(0,430,(140+pos2*15),27,
     +              '*** Error : 0 ≤ input no. <')
               call prti2(0,673,(140+pos2*15),4,'(i2)',2,order)
               call ERTONE()
            else
               call WRTSTR(0,430,(140+pos2*15),30,
     +              '                                ')
            endif
            if ((stpinp(pos2).lt.0).or.(stpinp(pos2).gt.order))
     +         goto 179
         endif
         pos2 = 1
         do 178 i = 11,23,3
            if ((i.ne.pos).and.(pos2.lt.100)) then
               pos2 = pos2 + 1
            elseif (i.eq.pos) then
               pos2 = pos2 + 100
            endif
178      continue
         if (pos2.gt.100) then
            pos2 = pos2 - 100
            key = getr4(0,210,(140+pos2*15),7,'(g10.4)',10,
     +                  stpinc(pos2))
         endif
         pos2 = 1
         do 177 i = 12,24,3
            if ((i.ne.pos).and.(pos2.lt.100)) then
               pos2 = pos2 + 1
            elseif (i.eq.pos) then
               pos2 = pos2 + 100
            endif
177      continue
         if (pos2.gt.100) then
            pos2 = pos2 - 100
175         continue
            key = getr4(0,320,(140+pos2*15),7,'(g10.4)',10,
     +                  stpdat(pos2))
            if (stpdat(pos2).gt.tend) then
               call WRTSTR(0,430,(140+pos2*15),27,
     +              '*** Error : Time > Duration')
               call ERTONE()
            elseif (stpdat(pos2).lt.(0.0)) then
               call WRTSTR(0,430,(140+pos2*15),22,
     +              '*** Error : Time < 0.0')
               call ERTONE()
            else
               call WRTSTR(0,430,(140+pos2*15),27,
     +              '                           ')
            endif
            if ((stpdat(pos2).gt.tend).or.(stpdat(pos2).lt.(0.0)))
     +         goto 175
         endif
      endif
      if (key.eq.downk) then
         if (pos.gt.9) then
            pos = pos + 3
         else
            pos = pos + 1
         endif
         if (pos.gt.24) pos = 1
      elseif ((key.eq.retk).or.(key.eq.tabk)) then
         pos = pos + 1
         if (pos.gt.24) pos = 1
      elseif (key.eq.upk) then
         if (pos.gt.11) then
            pos = pos - 3
         elseif (pos.eq.11) then
            pos = pos - 2
         else
            pos = pos - 1
         endif
         if (pos.lt.1) pos = 24
      elseif (key.eq.rtabk) then
         pos = pos - 1
         if (pos.lt.1) pos = 24
      endif
      if (key.ne.esck) goto 199
      call WIPSCR(0)
      return
```

```
      end
C ********************************************************************************
CH    REVISION HISTORY :
C     VERSION      BY                  DATE        COMMENT
C       1.00    A. de Waal           30/10/89    Creation.
C       1.1     A. de Waal           06/01/90   Modified & Finally Commented
C     FILEND           :
C ********************************************************************************
C
C     FILE             : SIMSCL.FOR
C ********************************************************************************
CN    MODULE NAME      : simscl
CA    FUNCTION         : Edit Time Simulation Axes and Scales.
CS    CALL SEQUENCE    : call simscl()
CI    INPUT PARAMETERS : None.
CO    OUTPUT PARAMETERS: None.
CG    GLOBAL VARIABLES : Include files:   time.inc,syst.inc,keys.inc
CM    MODULES CALLED   : Fortran  :   prti2,geti2,wrhead,prscal,edscal
C                        Assembler:   WIPSCR,DISP,LEVEL,MOVE,DLINE,WRTSTR,INKEY
CE    ERROR CONDITIONS : None
CC    COMMENTS         : Prints out all the axes limits for the time simulation
C                        plots. The user can then edit the scales accordingly.
C                        The ESC key returns control to the calling routine.
C ********************************************************************************
      subroutine simscl()
      implicit integer*2 (I)
$include: 'time.inc'
$include: 'syst.inc'
$include: 'keys.inc'
      integer*2 pos,key,elem,lastel,xpos,ypos,order

      order = int(syst(4))
      call WIPSCR(1)
      call DISP(1)
      call LEVEL(1)
      call wrhead(1,250,17,32,'Time Simulation Axes Definition.')
      call MOVE(5,25)
      call DLINE(714,25)
      call MOVE(185,25)
      call DLINE(185,316)
      call wrhead(1,10,39,9,'Element :')
      if (order.gt.0) then
         xpos = 105
         ypos = 39
         do 99 i = 1,order
            call WRTSTR(1,xpos,ypos,1,'(')
            call prti2(1,xpos+10,ypos,4,'(i2)',2,i)
            call WRTSTR(1,xpos+30,ypos,1,',')
            call prti2(1,xpos+40,ypos,4,'(i2)',2,i)
            call WRTSTR(1,xpos+60,ypos,1,')')
            ypos = ypos + 14
99       continue
         call wrhead(1,400,40,3,'In ')
         call WRTSTR(1,400,54,3,'Out')
         call WRTSTR(1,436,47,1,'=')
         call wrhead(1,200,95,14,'Output Scale :')
         call wrhead(1,400,95,9,'Maximum :')
         call wrhead(1,400,110,9,'Minimum :')
         pos = 1
         elem = 1
         lastel = 0
         call WRTSTR(1,200,250,20,'Cursor keys to move.')
         call WRTSTR(1,200,265,35,
     +                'RETURN and TAB keys to edit scales.')
         call WRTSTR(1,200,280,16,'ESC key to exit.')
         call prscal(lastel,elem)
98       continue
         key = INKEY(1)
         if (key.eq.downk) then
            lastel = elem
            elem = elem + 1
            if (elem.gt.order) elem = 1
            call prscal(lastel,elem)
         elseif (key.eq.upk) then
            lastel = elem
            elem = elem - 1
            if (elem.lt.1) elem = order
            call prscal(lastel,elem)
         elseif ((key.eq.tabk).or.(key.eq.rtabk).or.(key.eq.retk)) then
            call WRTSTR(1,200,250,27,'RETURN, TAB and cursor keys')
            call WRTSTR(1,200,265,35,
     +                  'to move.                                  ')
            call edscal(elem)
            call WRTSTR(1,200,250,27,'Cursor keys to move.       ')
            call WRTSTR(1,200,265,35,
     +                 'RETURN and TAB keys to edit scales.')
            lastel = elem
         endif
         if (key.ne.esck) goto 98
      endif
      call WIPSCR(1)
      return
      end

C ********************************************************************************
CN    MODULE NAME      : prscal
CA    FUNCTION         : Print Out Set of Scales and I/O Names.
CS    CALL SEQUENCE    : call prscal(last,element)
CI    INPUT PARAMETERS :    last - (integer*2) The previous element printed.
C                        element - (integer*2) The current element to be
C                                               printed.
CO    OUTPUT PARAMETERS: None.
CG    GLOBAL VARIABLES : Include files:time.inc,sysnms.inc
CM    MODULES CALLED   : Fortran  :   prtr8,wrhead
C                        Assembler:   BLKFIL,LENSTR,LEVEL,WRTSTR
CE    ERROR CONDITIONS : ?
C
CC    COMMENTS         : The routine  blanks out the previous elements
C                        parameters from the screen and then prints the
C                        current element parameters to the screen. The
C                        parameters include all the scales and that elements
C                        I/O names.
C ********************************************************************************
      subroutine prscal(lastel,elem)
      implicit integer*2 (L)
      integer*2 lastel,elem
```

```
$include: 'time.inc'
$include: 'sysnms.inc'
      integer*2 xpos,ypos,len

      call WRTSTR(1,455,40,25,'                              ')
      call WRTSTR(1,455,54,25,'                              ')
      len = LENSTR(25,inpnms(elem))
      if (LENSTR(25,outnms(elem)).gt.len) then
         len = LENSTR(25,outnms(elem))
      endif
      call wrhead(1,455,40,len,inpnms(elem))
      call WRTSTR(1,455,54,len,outnms(elem))
      xpos = 105
      ypos = 39 + (elem-1)*14
      call LEVEL(2)
      if ((lastel.ne.0).and.(lastel.ne.elem)) then
         call BLKFIL(xpos,(41+(lastel-1)*14),69,14)
      endif
      if (lastel.ne.elem) then
         call BLKFIL(xpos,ypos+2,69,14)
      endif
      call LEVEL(1)
      call prtr8(1,490,95,7,'(g10.4)',10,ytlim(1,elem))
      call prtr8(1,490,110,7,'(g10.4)',10,ytlim(2,elem))
      return
      end

C ******************************************************************************
CM    MODULE NAME       : edscal
CA    FUNCTION          : Edit the Axes Scales.
CS    CALL SEQUENCE     : call edscal(element)
CI    INPUT PARAMETERS  : element - (integer*2) The element to be edited.
CO    OUTPUT PARAMETERS: None.
CG    GLOBAL VARIABLES  : Include files:time.inc,keys.inc
CM    MODULES CALLED    : Fortran  :box,getr8
C                         Assembler:LEVEL,BLKFIL,ERTONE, WRTSTR
CE    ERROR CONDITIONS  : None
CC    COMMENTS          : Enables the user to edit the axes scales. The routine
C                         does do bound checking.
C                         The ESC key returns control to the calling routine.
C ******************************************************************************
      subroutine edscal(elem)
      implicit integer*2 (g)
      integer*2 elem
$include: 'time.inc'
$include: 'keys.inc'
      integer*2 key,pos,xpos,ypos,len

      xpos = 105
      ypos = 39 + (elem-1)*14
      call LEVEL(2)
      call BLKFIL(xpos,ypos+2,69,14)
      call box(xpos,ypos+2,69,12)
      call LEVEL(1)

      pos = 1
199   continue
      if (pos.eq.1) then
198      continue
         key = getr8(1,490,95,7,'(g10.4)',10,ytlim(1,elem))
         if (ytlim(1,elem).lt.(0.0)) then
            call WRTSTR(1,200,200,25,
     +                  '*** Error : Maximum < 0.0')
            call ERTONE()
         else
            call WRTSTR(1,200,200,25,
     +                  '                          ')
         endif
         if (ytlim(1,elem).lt.(0.0)) goto 198
      elseif (pos.eq.2) then
197      continue
         key = getr8(1,490,110,7,'(g10.4)',10,ytlim(2,elem))
         if (ytlim(2,elem).gt.(0.0)) then
            call WRTSTR(1,200,200,25,
     +                  '*** Error : Minimum > 0.0')
            call ERTONE()
         else
            call WRTSTR(1,200,200,25,
     +                  '                          ')
         endif
         if (ytlim(2,elem).gt.(0.0)) goto 197
      endif
      if (key.eq.downk) then
         pos = pos + 1
         if (pos.gt.2) pos = 1
      elseif (key.eq.upk) then
         pos = pos - 1
         if (pos.eq.0) pos = 2
      elseif ((key.eq.retk).or.(key.eq.tabk)) then
         pos = pos + 1
         if (pos.gt.2) pos = 1
      elseif (key.eq.rtabk) then
         pos = pos - 1
         if (pos.lt.1) pos = 2
      endif
      if (key.ne.esck) goto 199
      ytlim(3,elem) = 0.0
      xpos = 105
      ypos = 39 + (elem-1)*14
      call LEVEL(2)
      call box(xpos,ypos+2,69,12)
      call BLKFIL(xpos,ypos+2,69,14)
      call LEVEL(1)
      return
      end

C ******************************************************************************
CH    REVISION HISTORY :
C     VERSION      BY                DATE        COMMENT
C       1.00     Ian Fisher         23/06/88    Creation.
C       1.1      A. de Waal         06/01/90   Modified & Finally Commented
C     FILEND             :
C ******************************************************************************
cc$include: 'lst.inc'
C
C     FILE               : SIMUL.FOR
```

```
C ********************************************************************************
CN     MODULE NAME       : simul
CA     FUNCTION          : Drive the Time Simulation Menu.
CS     CALL SEQUENCE     : call simul()
CI     INPUT PARAMETERS  : None.
CO     OUTPUT PARAMETERS: None.
CG     GLOBAL VARIABLES  : None.
CM     MODULES CALLED    : Fortran  :   wrhead,stpopt,simscl,dosim,newmat
C                          Assembler:   DOMENU,WIPSCR
CE     ERROR CONDITIONS  : None
CC     COMMENTS          : Drives  the  time  simulation  menu,  calling  the
C                          appropriate routine upon the users request.
C ********************************************************************************
       subroutine simul()
       implicit integer*2 (D)
       integer*2 opt14
99     continue
       call wrhead(0,250,35,23,'System Time Simulation.')
       opt14 = DOMENU(14)
       if (opt14.eq.1) then
           call simopt()
       elseif (opt14.eq.2) then
          call simscl()
       elseif (opt14.eq.3) then
          call nindin()
       elseif (opt14.eq.4) then
          call dosim()
       elseif (opt14.eq.5) then
          call data()
       endif
       if (opt14.ne.0) goto 99
       call WIPSCR(0)
       return
       end

C ********************************************************************************
CN     MODULE NAME       : dosim
CA     FUNCTION          : Perform the Actual Simulation.
CS     CALL SEQUENCE     : call dosim()
CI     INPUT PARAMETERS  : None.
CO     OUTPUT PARAMETERS: None.
CG     GLOBAL VARIABLES  : Include files:   time.inc,syst.inc,keys.inc,defnam.inc
CM     MODULES CALLED    : Fortran  :   inisim,simgrf,simat,prtnum,pltsim,chkqt,
C                                       dwinps (for)
C                          Assembler:   WRTSTR,INKEY,WIPSCR
CE     ERROR CONDITIONS  : None
CC     COMMENTS          : Co-ordinates the actual time simulation : performs the
C                          necessary operations if the system is closed  loop  or
C                          open loop, controller excluded or included.
C                          The routine also checks if the  user wants to quit the
C                          simulation before running to  completion. This is done
C                          by polling the keyboard after each  step  and checking
C                          if any keys have been entered by the user.
C ********************************************************************************
       subroutine dosim()
       implicit integer*2 (I,g)
$include: 'time.inc'
$include: 'syst.inc'
$include: 'keys.inc'
$include: 'defnam.inc'

       integer*2 key,flush,order,dima,ferr
       integer finc
       real prevo(15) /15*0.0/
       real wfilt,tfilt,afilt,bfilt
       real er(15)
       real*4 ti2,xtrid
       real*8 scale(2), centre(4)
       character*1 trid

       order = int(syst(4))
       dima = int(syst(5))

       wfilt = 0.2
       tfilt = 1/wfilt
       afilt = exp(-dt/tfilt)
       bfilt = 1.0 - afilt
       do 25 i = 1, order
           prevo(i) = 0.0
25     continue

       call inisim()
       call WIPSCR(0)

55     continue
       plnoi = 1
       call WRTSTR(0,50,100,27,'Plot Perturbation Traces? :')
       call togopt(plnoi,50+28*9,100,ikey)

       precom = 1
       call WRTSTR(0,50,120,62,
      +'Include Precompensator for Zero Steady-State Error? (Y/N):    ')
       call togopt(precom,50+60*9,120,key)

       finc = 1
       call WRTSTR(0,50,140,40,
      +    'Include Lowpass Filter on Outputs (Y/N):')
       call togopt(finc,50+41*9,140,key)

       if (finc.eq.0) then
           call WRTSTR(0,50,160,31,
      +    'Filter 3dB Cut-Off Frequency = ')
           key = getr4(0,50+32*9,160,7,'(g10.4)',10,wfilt)
       endif

       call WRTSTR(0,50,180,40,
      +    'Hit RETURN to Start or ESC to re-specify')
       if (key.eq.esck) goto 55

       call simgrf()
       flush = INKEY(2)
       ti2 = -1.0

c      fname = 'noise.noi'
c      call maknam(fname,inpath,usenam)
c      open (1,file=usenam,status='unknown',iostat=ferr)
```

```
c       fname = 'dist.dis                '
c       call maknam(fname,inpath,usenam)
c       open (2,file=usenam,status='unknown',iostat=ferr)
c       fname = 'input.inp               '
c       call maknam(fname,inpath,usenam)
c       open (3,file=usenam,status='unknown',iostat=ferr)
c       fname = 'output.out              '
c       call maknam(fname,inpath,usenam)
c       open (4,file=usenam,status='unknown',iostat=ferr)
cc       fname = 'sdis.dis               '
cc       call maknam(fname,inpath,usenam)
cc       open (9,file=usenam,status='unknown',iostat=ferr)
c       iwrite = 0
c
c       if (ferr.ne.0) then
c           call ERTONE()
c           call LEVEL(0)
c           call WRTSTR(0,50,250,34,
c      +             'ERROR: Could Not Open File for FFT')
c           call LEVEL(1)
c       endif

        do 798 i = 1, int(tend/dt)
            call prtr4(1,75,(thstrt-thmax-18),7,'(g10.4)',10,ti)

            do 701 j = 1,5
                if ((ti.ge.stpdat(j)).and.(ti2.lt.stpdat(j))) then
                    if ((stpinp(j).gt.0).and.(stpinp(j).le.order)) then
                        rn(stpinp(j)) = rn(stpinp(j)) + stpinc(j)
                    endif
                endif
701         continue

            if (rinc.eq.0) call perset()

            if (precom.eq.0) then
                do 111 j = 1, order
                    vn(j) = 0.0
                    do 112 k = 1, order
                        vn(j) = vn(j) + ptrsst(j,k)*rn(k)
112                 continue
111             continue
            else
                do 115 j = 1, order
                    vn(j) = rn(j)
115             continue
            endif

            if (coninc.eq.1) then
                do 110 k = 1, order
                    un(k) = vn(k)
110             continue
            endif

            if (ninc.eq.0) call noise()
            if (dinc.eq.0) call dist()

C Calculate process states, xn and process outputs, yn
            call prosta()

C Lowpass filtering outputs to get rid of High Frequency Noise
C HOROWITZ & HILL : THE ART OF ELECRONICS, PAGE447
            if (finc.eq.0) then
                do 23 n = 1, order
                    ynact(n) = afilt*prevo(n) + bfilt*ynact(n)
                    prevo(n) = ynact(n)
23              continue
            endif

C If observer to be included, calculate observed states, xnobs
C                                 and obserer outputs, ynobs

            if (obsinc.eq.0) call obssta()

C If controller to be included, calculate new input, un
            if (coninc.eq.0) call concal()

            if (setplt.eq.0) call pltsim(order,ti,rn)
            if (inpplt.eq.0) call pltsim(order,ti,un)
            if (outplt.eq.0) call pltsim(order,ti,ynact)
            if (ostplt.eq.0) call pltsta(order,dima,ti,xnobs)
            if (pstplt.eq.0) call pltsta(order,dima,ti,xn)

            do 876 n = 1, order
                er(n) = rn(n) - yn(n)
876         continue
            if ((rinc.eq.0).and.(plnoi.eq.0))
     +          call pltsim(order,ti,er)


c           iwrite = iwrite + 1
c           if (iwrite.le.1024) then
c               write(1,'(2f14.4)',err=203)(nn(k),k=1,order)
c               write(1,'(2f14.4)',err=203)0.0,0.0
c               write(2,'(2f14.4)',err=203)(zn(k),k=1,order)
c               write(2,'(2f14.4)',err=203)0.0,0.0
c               write(3,'(2f14.4)',err=203)(un(k),k=1,order)
c               write(3,'(2f14.4)',err=203)0.0,0.0
c               write(4,'(2f14.4)',err=203)(ynact(k),k=1,order)
c               write(4,'(2f14.4)',err=203)0.0,0.0
c           endif


202         continue
            goto 204
203         continue
            call ERTONE()
            call LEVEL(0)
            call WRTSTR(0,50,270,38,
     +             'ERROR: Could Not Write to File for FFT')
            call LEVEL(1)
204         continue

            ti2 = ti
            ti = ti + dt
            key = INKEY(4)
```

```
            if (key.ne.0) then
               call chkgt(key)
               call WRTSTR(1,10,(thstrt-thmax-18),50,
     +              '                                                ')
               call WRTSTR(1,10,(thstrt-thmax-18),7,'Time =  ')
               call WRTSTR(1,10,(thstrt-thmax-4),31,
     +             'Hit ESC to quit the simulation.')
            endif
            if (key.ne.0) goto 799
798      continue
799      continue

c         close(1)
c         close(2)
c         close(3)
c         close(4)
c         close(9)

         xtrid = ti
         do 666 igrph = 1, order
            do 555 j = 1, 4
               if (j.le.2) scale(j) = tscale(j,igrph)
               centre(j) = tcentr(j,igrph)
555         continue
            trid = '1'
            if (setplt.eq.0)
     +         call drwlet(1,xtrid,rn(igrph),centre,scale,trid)
            trid = '2'
            if (inpplt.eq.0)
     +         call drwlet(1,xtrid,un(igrph),centre,scale,trid)
            trid = '3'
            if (outplt.eq.0)
     +         call drwlet(1,xtrid,ynact(igrph),centre,scale,trid)
            trid = '4'
            if ((rinc.eq.0).and.(plnoi.eq.0))
     +         call drwlet(1,xtrid,er(igrph),centre,scale,trid)
            if (ostplt.eq.0) then
               do 333 k = 1, dima
                  trid = 'S'
                  call drwlet(1,xtrid,xnobs(k),centre,scale,trid)
                  trid = 'O'
                  call drwlet(1,(xtrid+7/sngl(scale(1))),xnobs(k),
     +                        centre,scale,trid)
                  trid = char(48+k)
                  call drwlet(1,real(xtrid+14/sngl(scale(1))),xnobs(k),
     +                        centre,scale,trid)
c                  call drwlet(1,real(xtrid+14/sngl(scale(1))),er(k),
c     +                        centre,scale,trid)
333            continue
            endif
            if (pstplt.eq.0) then
               do 444 k = 1, dima
                  trid = 'S'
                  call drwlet(1,xtrid,xn(k),centre,scale,trid)
                  trid = char(48+k)
                  call drwlet(1,real(xtrid+7/sngl(scale(1))),xn(k),
     +                        centre,scale,trid)
444            continue
            endif
666      continue

         call WRTSTR(1,10,(thstrt-thmax-18),61,
     +'1:Setp, 2:Inp, 3:Outp, 4:Err, Sn:Pro State n, SOn:Obs State n')
         call WRTSTR(1,10,(thstrt-thmax-4),31,
     +           'Hit ESC to quit, F3 to list O/P')
710      continue
         key = INKEY(1)
         if (key.ne.esck) goto 710
         call WIPSCR(1)
         call WIPSCR(0)
         return
         end

C ***************************************************************************
CN    MODULE NAME       : simgrf
CA    FUNCTION          : Plot All Time Simulation Axes
CS    CALL SEQUENCE     : call simgrf()
CI    INPUT PARAMETERS  : None.
CO    OUTPUT PARAMETERS: None.
CG    GLOBAL VARIABLES  : Include files:   syst.inc,time.inc,sysnms.inc
CM    MODULES CALLED    : Fortran  :   wrhead,prtnum,dwaxes,dwnumb
C                         Assembler:   WIPSCR,LENSCR,LEVEL,MOVE,DLINE,WRTSTR
CE    ERROR CONDITIONS  : None
CC    COMMENTS          : The routine first  calculates  how  the page is to be
C                         split up. Each  set  of axes is then drawn in equally
C                         sized block on the page. Each  set  of  axes has it's
C                         parameters (scales, centres, etc.) stored in an array
C                         for use at a later stage  when  points are plotted on
C                         the set of axes.
C ***************************************************************************
         subroutine simgrf()
         implicit integer*2 (L)
$include: 'syst.inc'
$include: 'time.inc'
$include: 'sysnms.inc'
         integer*2 i,j,k,len,chkio,xpos,ypos,order
c         real*8 xt(3), yt(3), scal(2), cent(4)
         character*10 tstr

         order = int(syst(4))
         xpos = 40
         ypos = 50
         call WIPSCR(0)
         call wrhead(0,200,17,17,'System Parameters')
         call wrhead(0,40,35,3,'No.')
         call wrhead(0,100,35,5,'Input')
         call wrhead(0,350,35,6,'Output')
         do 800 i = 1,order
            call prti2(0,xpos,ypos,4,'(i2)',2,i)
            call WRTSTR(0,(xpos+60),ypos,25,inpnms(i))
            call WRTSTR(0,(xpos+310),ypos,25,outnms(i))
            ypos = ypos + 15
800      continue

         call DISP(1)
         call WIPSCR(1)
```

```
      twmax   = 711
      twstrt = 4
      thmax   = 270
      thstrt = 316
      call wrhead(1,10,(thstrt-thmax-32),25,
     +            'System Time Simulation : ')
      call WRTSTR(1,245,(thstrt-thmax-32),25,prjnm)
      len = LENSTR(25,prjnm)
      call WRTSTR(1,(245+len*9),(thstrt-thmax-32),2,', ')
      call WRTSTR(1,(245+(len+2)*9-7),(thstrt-thmax-32),25,engnms)
      call WRTSTR(1,10,(thstrt-thmax-18),7,'Time = ')
          call WRTSTR(1,10,(thstrt-thmax-4),31,
     +          'Hit ESC to quit the simulation.')
      if (order.le.1) then
         twidth = 1
      elseif (order.le.4) then
         twidth = 2
      elseif (order.le.9) then
         twidth = 3
      elseif (order.le.16) then
         twidth = 4
      else
         twidth = 6
      endif
      if ((order.gt.0).and.(order.lt.11)) then
         theigh = int (order/twidth)
         if ((real(order)/real(twidth)).gt.(real(int(order/twidth))))
     +                                    theigh = theigh + 1
         txdel = int(twmax/twidth)
         tydel = int(thmax/theigh)
         call LEVEL(1)
         do 199 i = 1,(twidth-1)
            call MOVE((i*txdel+twstrt),(thstrt-theigh*tydel))
            call DLINE((i*txdel+twstrt),thstrt)
199      continue
         do 198 i = 1,theigh
            call MOVE(twstrt,(thstrt-i*tydel))
            call DLINE((twstrt+twmax-1),(thstrt-i*tydel))
198      continue
         j = 1
         k = 1
         do 197 i = 1,order
            tgraph(1) = twstrt + 4 + (j-1)*txdel
            tgraph(2) = thstrt - thmax - 2 + k*tydel
            tgraph(3) = txdel - 8
            tgraph(4) = tydel - 4
c            do 246 j = 1, 3
c              xt(j) = xtlim(j)
c              yt(j) = ytlim(j,i)
c246         continue
c              call dwaxes(1,1,1,xt,yt,tgraph,
c     +                    scal,cent)
            call dwaxes(1,1,1,xtlim(1),ytlim(1,i),tgraph,
     +                  tscale(1,i),tcentr(1,i))
c            do 244 j = 1, 4
c              if (j.le.2) tscale(j,i) = scal(j)
c              tcentr(j,i) = cent(j)
c244         continue
            tstr = '          '
            write(tstr,'(i2)',iostat=chkio,err=299) i
            call dwnumb(1,(twstrt+j*txdel-16),
     +           (thstrt-thmax+(k-1)*tydel+10),tstr,2)
            j = j + 1
            if (j.gt.twidth) then
               j = 1
               k = k + 1
            endif
299         continue
197      continue
      endif

      return
      end

C ***************************************************************************
CN    MODULE NAME      : inisim
CA    FUNCTION         : Initialise States and Arrays for Simulation
CS    CALL SEQUENCE    : call inisim()
CI    INPUT PARAMETERS : None
CO    OUTPUT PARAMETERS: None
CG    GLOBAL VARIABLES : Include files:   time.inc,syst.inc
CM    MODULES CALLED   : None
CE    ERROR CONDITIONS : None
CC    COMMENTS         : Just initialises all the states to zero.
C                        The arrays :C1 and C2, are initialised for the set of
C                        Runga-Kutta equations.
C ***************************************************************************
      subroutine inisim()
$include: 'time.inc'
$include: 'syst.inc'
      integer*2 i,j,k,order,dima

      order = int(syst(4))
      dima = int(syst(5))

      c1(1) = 1.0
      c1(2) = 2.0
      c1(3) = 2.0
      c1(4) = 1.0

      c2(1) = 0.5
      c2(2) = 0.5
      c2(3) = 1.0

      xtlim(1) = tend
      xtlim(2) = 0.0
      xtlim(3) = 0.0

      do 88 i = 1,order
         rn(i) = 0.0
         vn(i) = 0.0
         un(i) = 0.0
         yn(i) = 0.0
         ynobs(i) = 0.0
         nn(i) = 0.0
```

```
88    continue

      do 99 i = 1, dima
          xn(i) = 0.0
          xnobs(i) = 0.0
          zn(i) = 0.0
99    continue

c      do 87 i = 1,12000
c         usim(i) = 0.0
c         ysim(i) = 0.0
c87    continue

      ti = 0.0
      return
      end

C ***************************************************************************
CN    MODULE NAME       : pltsim
CA    FUNCTION          : Plot Outputs from Simulation.
CS    CALL SEQUENCE     : call pltsim(n,t,y)
CI    INPUT PARAMETERS :
C                    n - (integer*2) Order of the system.
C                    t - (real*4) The current time of the simulation (X axis).
C                y(15) - (real*4) The system outputs.
C
CO    OUTPUT PARAMETERS: None
CG    GLOBAL VARIABLES : Include files:   time.inc
CM    MODULES CALLED   : Fortran  :   plotpt
CE    ERROR CONDITIONS : None
CC    COMMENTS         : Plots each set of points on the resepective set of
C                        axes.
C ***************************************************************************
      subroutine pltsim(n,t,pt)
      integer*2 n
      real*4 t,pt(15)
$include: 'time.inc'
      real*8 cent(4),scal(2)
      integer*2 xtype,i,j
      xtype = 1
      do 399 i = 1,n
         do 398 j = 1,4
            cent(j) = tcentr(j,i)
398      continue
         scal(1) = tscale(1,i)
         scal(2) = tscale(2,i)
         call plotpt(1,xtype,t,pt(i),cent,scal)
399   continue
      return
      end
C ***************************************************************************
CN    MODULE NAME       : pltsta
CA    FUNCTION          : Plot States from Simulation
CS    CALL SEQUENCE     : call pltsim(n,t,y)
CI    INPUT PARAMETERS :
C                    n - (integer*2) Number of states of the system.
C                    t - (real*4) The current time of the simulation (X axis).
C                y(15) - (real*4) The system states.
C
CO    OUTPUT PARAMETERS: None
CG    GLOBAL VARIABLES : Include files:   time.inc
CM    MODULES CALLED   : Fortran  :   plotpt
CE    ERROR CONDITIONS : None
CC    COMMENTS         : Plots each set of points on the resepective set of
C                        axes.
C ***************************************************************************
      subroutine pltsta(n,m,t,pt)
      integer*2 n,m
      real*4 t,pt(15)
$include: 'time.inc'
      real*8 cent(4),scal(2)
      integer*2 xtype,i,j
      xtype = 1
      do 399 i = 1,n
         do 398 j = 1,4
            cent(j) = tcentr(j,i)
398      continue
         scal(1) = tscale(1,i)
         scal(2) = tscale(2,i)
         do 222 k = 1, m
             call plotpt(1,xtype,t,pt(k),cent,scal)
222      continue
399   continue
      return
      end
C ***************************************************************************
CN    MODULE NAME       : chkqt
CA    FUNCTION          : Check if User wishes to Quit Simulation.
CS    CALL SEQUENCE     : call chkqt(key)
CI    INPUT PARAMETERS : None.
CO    OUTPUT PARAMETERS:
C                    key - (integer*2) The returned user option :
C                          0 : user does not wish to quit.
C                          1 : user wishes to quit.
CG    GLOBAL VARIABLES : Include files:   time.inc,keys.inc
CM    MODULES CALLED   : Assembler:   INKEY,WRTSTR,STRIN
CE    ERROR CONDITIONS : None
CC    COMMENTS         : Just confirms if the user wishes to quit the
C                        simulation before it has run to completion.
C ***************************************************************************
      subroutine chkqt(key)
      implicit integer*2 (G,I,S)
      integer*2 key
      integer*2 flush,pg
      character*1 yesno
$include: 'keys.inc'
$include: 'time.inc'
      flush = INKEY(2)
      if (key.eq.esck) then
         call WRTSTR(1,10,(thstrt-thmax-4),50,
     +        '                                                  ')
         call WRTSTR(1,10,(thstrt-thmax-4),39,
     +        'Do you want quit the simulation (y/n) ?')
         yesno = 'n'
111      continue
         key = STRIN(1,361,(thstrt-thmax-4),1,yesno)
```

```
         if (key.ne.retk) goto 111
         if ((yesno.eq.'y').or.(yesno.eq.'Y')) then
            key = 1
         else
            key = 0
         endif
      else
         key = 0
      endif
      call WRTSTR(1,10,(thstrt-thmax-4),50,
     +   '                                            ')
      return
      end
C ******************************************************************************
CH    REVISION HISTORY :
C     VERSION      BY                DATE        COMMENT
C       1.00    Ian Fisher         23/06/88    Creation
C       1.1     A. de Waal         06/01/90   Modified & Finally Commented
C     FILEND         :
C ******************************************************************************
```

M6) OPTCAD Menu Definition Routine

The menu structure for the OPTCAD package is entered into the menu text file, MTEXT.ASM. The data in this file is accesed by the assembler a routine called DOMENU.ASM (Fisher, 1988), which presents specific options to the user on the hercules graphics page 0.

A listing of the file MTEXT.ASM for the OPTCAD package is given on the following page.

```
;File:MTEXT.ASM
;Name:mtext
;Function:CAD Package Menu Text
; MODULE : Menu macro routines.
;          *********************

TITLE Sets up the menu data segments.
;     *******************************

        PUBLIC   FHELP

; ****************************************************************************
; REVISION HISTORY :
; VERSION       BY                DATE        COMMENT
;    1.00       Ian Fisher        23/06/88    Creation.
;
; ****************************************************************************

; Termination characters.
; ***********************
EOM     EQU     02      ; End of menu.
EOS     EQU     00      ; End of string.
EOT     EQU     04      ; End of table.
ESC     EQU     1BH     ; ESC character.
HLP     EQU     03      ; Start of help.

; Macro definitions.
; ******************
; Set up menu data area.
; **********************

STARTMENU       MACRO   MENU_NUM
        MENU_DATA       SEGMENT PUBLIC 'FAR_DATA'
MENU&MENU_NUM   LABEL BYTE
        MENU_DATA       ENDS
        TABLE_DATA      SEGMENT PUBLIC 'FAR_DATA'
        DW      MENU_NUM
        DW      MENU&MENU_NUM
        TABLE_DATA      ENDS
        MENU_DATA       SEGMENT PUBLIC 'FAR_DATA'
        ENDM

; End menu data area.
; *******************
ENDMENU MACRO
        MENU_DATA       ENDS
        ENDM

; ****************************************************************************
; Menu data areas.
; ****************

TABLE_DATA      SEGMENT WORD PUBLIC 'FAR_DATA'
TABLE_DATA      ENDS

MENU_DATA       SEGMENT WORD PUBLIC 'FAR_DATA'
MENU_DATA       ENDS

MENU_DATA       SEGMENT WORD PUBLIC 'FAR_DATA'
FHELP   DB      'LOCI.HLP       ',0
MENU_DATA       ENDS

; ****************************************************************************
; The structure of a menu is described :-
;
;     STARTMENU   number
;                 DB    'option1',HLP,'Description of option1',0
;                 DB    'option2',HLP,'Description of option2',0
;                 .
;                 .
;                 DB    'option(n-1)',HLP,'Description of option(n-1)',0
;                 DB    'option(n)',HLP,'Description of option(n)',0
;                 DB    EOM
;     ENDMENU
;
; STARTMENU and ENDMENU are macros defined above, 'number' is parameter of
; the macro : STARTMENU.
; HLP and EOM have been defined in the equates above.
;
; 'number' defines the menu to be used when calling DOMENU from FORTRAN.
;
; The 'number' can be any value except 0, this is used by the system to
; define the end of the menu data area and tables.
;
; For a description of how the menu is displayed, see the routine : DOMENU
; which resides in the module DOMENU.ASM.
;
; ****************************************************************************
;
; Menu definition.
; ****************
STARTMENU       1
        DB      'Data',HLP,'Load/Save/Edit Package Data.',0
        DB      'Optcon',HLP,'Synthesize Optimal LQG Controller.',0
        DB      'Analyse',HLP,'Analyse and Plot Performance Indices.',0
        DB      'Simulate',HLP,'Simulate Open or Closed Loop Process.',0
        DB      'Quit',HLP,'Exit from program.',0
        DB      EOM
ENDMENU

STARTMENU       11
        DB      'Load',HLP,'Load Package Data.',0
        DB      'Save',HLP,'Save Package Data.',0
        DB      'Edit',HLP,'Edit Package Data.',0
        DB      EOM
ENDMENU

STARTMENU       111
        DB      'Proc',HLP,'Load State Space Process Description.',0
        DB      'Wt/Cov',HLP,'Load Control Weighting and Noise Covariance Matrices.',0
        DB      'LQGDes',HLP,'Load System with LQG Controller Design Information.',0
        DB      'ManDes',HLP,'Load System with Contr./Observ. Designed Not Using LQG.',0
        DB      'AnalLQG',HLP,'Load Perf. Index Analysis Data for Previous LQG Design.',0
        DB      'DesManal',HLP,'Load Perf. Index Analysis Data for Design not using LQG.',0
        DB      'KfKc',HLP,'Load Controller and Kalman Observer Gain Matrices.',0
```

```
        DB      'Exdat',HLP,'Load External MATLAB Generated Performance Index Data.',0
        DB      EOM
ENDMENU

STARTMENU       112
        DB      'Proc',HLP,'Save State Space Process Description.',0
        DB      'Wt/Cov',HLP,'Save Control Weighting and Noise Covariance Matrices.',0
        DB      'LQGDes',HLP,'Save System with LQG Controller Design Information.',0
        DB      'ManDes',HLP,'Save System with Contr./Observ. Designed Not Using LQG.',0
        DB      'AnalLQG',HLP,'Save Perf. Index Analysis Data for Current LQG Design.',0
        DB      'DesMAnal',HLP,'Save Perf. Index Analysis Data for Design not using LQG.',0
        DB      'KfKc',HLP,'Save Controller and Kalman Observer Gain Matrices.',0
        DB      EOM
ENDMENU

STARTMENU       113
        DB      'SDef',HLP,'Edit Default Package Settings.',0
        DB      'Proc',HLP,'Edit State Space Process Description.',0
        DB      'Wt/Cov',HLP,'Edit Control Weighting and Noise Covariances.',0
        DB      'KfKc',HLP,'Edit Contr./Observ. Matrices Instead of Using LQG Synthesis.',0
        DB      'TrPrec',HLP,'Edit Steady-state Precompensator Gain Matrix.',0
        DB      EOM
ENDMENU

STARTMENU       1131
        DB      EOM
ENDMENU

STARTMENU       1132
        DB      'Syst',HLP,'Edit System Constants.',0
        DB      'AMat',HLP,'Edit Process State Space A-Matrix.',0
        DB      'BMat',HLP,'Edit Process State Space B-Matrix.',0
        DB      'CMat',HLP,'Edit Process State Space C-Matrix.',0
        DB      EOM
ENDMENU

STARTMENU       1133
        DB      'Weight',HLP,'Edit Control Weighting Matrices.',0
        DB      'Covar',HLP,'Edit Noise Covariance Matrices.',0
        DB      EOM
ENDMENU

STARTMENU       11331
        DB      'Input',HLP,'Edit Input Weighting Matrix.',0
        DB      'State',HLP,'Edit State Weighting Matrix.',0
        DB      EOM
ENDMENU

STARTMENU       11332
        DB      'Q',HLP,'Edit Observer Q-Matrix.',0
        DB      'R',HLP,'Edit Observer R-Matrix.',0
        DB      EOM
ENDMENU

STARTMENU       1134
        DB      'Contr',HLP,'Edit Controller Gain Matrix.',0
        DB      'Obser',HLP,'Edit Observer Gain Matrix.',0
        DB      EOM
ENDMENU

STARTMENU       13
        DB      'Format',HLP,'Edit Format of Perfomance Index Analysis Plots.',0
        DB      'Create',HLP,'Create Performance Index Data and Analysis.',0
        DB      'View',HLP,'View Performance Index Data and Analysis.',0
        DB      'Data',HLP,'Load/Save/Edit Package Data.',0
        DB      EOM
ENDMENU

STARTMENU       131
        DB      'Format',HLP,'Edit Format of Perfomance Index Analysis.',0
        DB      'Go',HLP,'Start Creation and Analysis Procedure.',0
        DB      EOM
ENDMENU


STARTMENU       14
        DB      'Format',HLP,'Edit Format of Simulation',0
        DB      'Scales',HLP,'Edit Scales of Simulator Plots',0
        DB      'Perturb',HLP,'Edit Specifications of Perturbations to Process',0
        DB      'Go',HLP,'Start Simulation',0
        DB      'Data',HLP,'Load/Save/Edit Package Data.',0
        DB      EOM
ENDMENU

STARTMENU       141
        DB      'Gaussian',HLP,'Gaussian Perturbations.',0
        DB      'Sinusoid',HLP,'Sinusoidal Perturbations.',0
        DB      EOM
ENDMENU

STARTMENU       1411
        DB      'NoiseDist',HLP,'Specify Sinusoidal Noise/Disturbances.',0
        DB      'Setpoints',HLP,'Specify Sinusoidal Setpoint Perturbations.',0
        DB      EOM
ENDMENU


STARTMENU       0
ENDMENU

; ************************************************************************************
        END
```

M7) OPTCAD System Include Files

A table of the names of the global variable groupings in the OPTCAD system include files follows on the next page. The name of each group of global variables is given in the table, together with the include file in which it is located and the page in the appendix on which it is listed.

Grouping of Global Variables in System Include Files			
Common Block	Include File	Description	Page
defnam	DEFNAM.INC	Filename and pathnames.	470
fnames	FNAMES.INC	Filename and pathnames.	470
keys	KEYS.INC	Keyboard constants.	470
perdat	PERDAT.INC	Performance Index Data	470
permat	PERMAT.INC	Matrices used in Performance Index Calculation	470
pldat	PLDAT.INC	Data for Group Plotting of Performance Indices	471
runvar	RUNVAR.INC	Flags Used in the Running of CAD System	471
sysnms	SYSNMS.INC	System I/O names, matrix names and system titles.	471
syst	SYST.INC	State-space System Matrices	471
time1	TIME.INC	Part of the state variables for the time simulation.	471
time2	TIME.INC	Part of the state variables for the time simulation.	471
time3	TIME.INC	Arrays for time simulation axes settings.	472
time4	TIME.INC	Arrays for time simulation axes settings.	472

```
C
C     INCLUDE FILE : DEFNAM.INC
C ***************************************************************************
CN     COMMON NAME   : defnam
CA     DESCRIPTION   : Filename and pathnames.
CP     PARAMETERS    :
C
C            outpth - Save pathname.
C            inpath - Load pathname.
C            lqanam - Filename for Analysed LQG Design Information.
C            lqgnam - Filename for LQG Design Information (Not Analysed).
C            deanam - Filename for Analysed Non-LQG Design Information.
C            desnam - Filename for Non-LQG Design Information (Not Analysed).
C            pronam - Filename for Process Data.
C            qrnam  - Filename for Q, R, Q1 and R1 Matrix Information.
C            knam   - Filename for Kc and Kf Matrix Information.
C            defnam - Filename for Package Default Setting Information.
C ***************************************************************************

      character*25 outpth,inpath
      character*5 defdir
      character*25 infnam,fname,mname,pernam
      character*50 usenam
      common /defnam/ outpth,inpath
      common /filnam/ defdir,infnam,fname,mname,pernam,usenam
C ***************************************************************************
C
C     INCLUDE FILE : FNAMES.INC
C ***************************************************************************
CN     COMMON NAME   : fnames
CA     DESCRIPTION   : Filename and pathnames.
CP     PARAMETERS    :
C
C            outpth - Save pathname.
C            inpath - Load pathname.
C            gfnam  - Filename for G(s) matrix.
C            kfnam  - Filename for K(s) matrix.
C            prjfil - Filename for project file.
C ***************************************************************************

      character*25 outpth,inpath,gfnam,kfnam
      character*5 prport
      character*50 prjfil
      common /fnames/ outpth,inpath,gfnam,kfnam,prport,prjfil

C ***************************************************************************
C
C     INCLUDE FILE : KEYS.INC
C ***************************************************************************
CN     COMMON NAME   : keys
CA     DESCRIPTION   : Keyboard constants.
CP     PARAMETERS    :
C

C ***************************************************************************
      integer*2 upk,downk,leftk,rightk,homek,endk,pgupk,pgdnk,
     +          esck,retk,tabk,rtabk
      integer*2 sma,bga,sml,bgl,smq,bgq,smm,bgm,sms,bgs
      common /key/ upk,downk,leftk,rightk,homek,endk,pgupk,pgdnk,
     +             esck,retk,tabk,rtabk,
     +             sma,bga,sml,bgl,smq,bgq,smm,bgm,sms,bgs

C ***************************************************************************
C
C     INCLUDE FILE : PERDAT.INC
C ***************************************************************************
CN     COMMON NAME   : perdat
CA     DESCRIPTION   : Performance Index Data
CP     PARAMETERS    :
C
C ***************************************************************************
      real pidat,pival,stval
      integer piclev,stclev
      character*6 picom
      character*20 stcom
      dimension pidat(4,2,100),pival(4,2,5),stval(2,5)
      dimension picom(4,2,5),stcom(2)
      dimension piclev(4,2,5),stclev(2)
      common /perdat/ pidat,pival,stval
      common /percom/ stcom,picom
      common /perlev/ piclev,stclev
C ***************************************************************************

C     INCLUDE FILE : PERMAT.INC
C ***************************************************************************
CN     COMMON NAME   : permat
CA     DESCRIPTION   : Matrices used in Performance Index Calculation
CP     PARAMETERS    :
C
C ***************************************************************************
      integer ip
      parameter (ip = 15)
      real gmatr,gmati,fimatr,fimati,
     +     pmatr,pmati,fmatr,fmati,
     +     mmatr,mmati,l2matr,l2mati,
     +     l1matr,l1mati,s2matr,s2mati,
     +     s1matr,s1mati,t2matr,t2mati,
     +     t1matr,t1mati,
     +     wmatr,wmati,efmatr,efmati,
     +     ermatr,ermati,jwmatr,jwmati,
     +     tematr,temati,tema2r,tema2i,
     +     zermat,eyemat,
     +     jamatr,jamati,
     +     kfcmat
      dimension gmatr(ip,ip),gmati(ip,ip),fimatr(ip,ip),fimati(ip,ip),
     +          pmatr(ip,ip),pmati(ip,ip),fmatr(ip,ip),fmati(ip,ip),
     +          mmatr(ip,ip),mmati(ip,ip),l2matr(ip,ip),l2mati(ip,ip),
     +          l1matr(ip,ip),l1mati(ip,ip),s2matr(ip,ip),s2mati(ip,ip),
     +          s1matr(ip,ip),s1mati(ip,ip),t2matr(ip,ip),t2mati(ip,ip),
     +          t1matr(ip,ip),t1mati(ip,ip),
     +          wmatr(ip,ip),wmati(ip,ip),efmatr(ip,ip),efmati(ip,ip),
     +          ermatr(ip,ip),ermati(ip,ip),jwmatr(ip,ip),jwmati(ip,ip),
     +          tematr(ip,ip),temati(ip,ip),tema2r(ip,ip),tema2i(ip,ip),
     +          zermat(ip,ip),eyemat(ip,ip),
     +          jamatr(ip,ip),jamati(ip,ip),
     +          kfcmat(ip,ip)
```

```
      common /mats/ gmatr,gmati,fmatr,fmati,
     +     pmatr,pmati,fmatr,fmati,
     +     mmatr,mmati,l2matr,l2mati,
     +     l1matr,l1mati,s2matr,s2mati,
     +     s1matr,s1mati,t2matr,t2mati,
     +     t1matr,t1mati,
     +     wmatr,wmati,efmatr,efmati,
     +     ermatr,ermati,jwmatr,jwmati,
     +     jamatr,jamati,
     +     kfcmat
      common /utilmt/ zermat,eyemat
c      common /utilmt/ tematr,temati,tema2r,tema2i,
c     +                zermat,eyemat
C ***************************************************************************
C
C     INCLUDE FILE : PLDAT.INC
C ***************************************************************************
CN    COMMON NAME  : pldat
CA    DESCRIPTION  : Data for Group Plotting of Performance Indices
CP    PARAMETERS   :
C
C ***************************************************************************
      integer*2 grpos(4,4),boxcor(4,2),inflen(4,5)
      real*8 xaxdat(4,3),yaxdat(4,3)
      character*40 boxinf(4,5)
      common /plpos/ grpos,xaxdat,yaxdat,boxcor,inflen
      common /lnam/ boxlnf
C ***************************************************************************
C
C     INCLUDE FILE : RUNVAR.INC
C ***************************************************************************
CN    COMMON NAME  : runvar
CA    DESCRIPTION  : Flags Used in the Running of CAD System
CP    PARAMETERS   :
C
C ***************************************************************************
      logical deffl,profl,qrfl,lqgfl,desfl,lqafl,deafl,kfl,simfl
      common /runfl/ deffl,profl,qrfl,lqgfl,desfl,lqafl,deafl,kfl,simfl
C ***************************************************************************
C
C     INCLUDE FILE : SYSNMS.INC
C ***************************************************************************
CN    COMMON NAME  : sysnms
CA    DESCRIPTION  : System I/O names, matrix names and system titles.
CP    PARAMETERS   :
C           inpnms - System input names.
C           outnms - System output names.
C            prjnm - Project title.
C           engnms - Engineer or design team name.
C ***************************************************************************
      dimension inpnms(10),outnms(10)
      character*25 inpnms,outnms,prjnm,engnms
      common /sysnms/ inpnms,outnms,prjnm,engnms

C ***************************************************************************
C
C     INCLUDE FILE : SYST.INC
C ***************************************************************************
CN    COMMON NAME  : syst
CA    DESCRIPTION  : State-space System Matrices
CP    PARAMETERS   :
C
C ***************************************************************************
      integer id,nsyst
      parameter (id    = 15,
     +           nsyst = 5)
      integer precom
      real a,b,c,kci,kfi,q1,r1,kc,q,r,kf,syst,ptrsst
      dimension a(id,id),b(id,id),c(id,id),kci(id,id),kfi(id,id),
     +          q1(id,id),r1(id,id),kc(id,id),
     +          q(id,id),r(id,id),kf(id,id),ptrsst(id,id),
     +          syst(nsyst)
      common /sssdat/ a,b,c,kci,kfi,q1,r1,kc,q,r,kf,ptrsst
      common /syst/ syst
      common /precom/ precom
C ***************************************************************************
C
C     INCLUDE FILE : TIME.INC
C ***************************************************************************
CN    COMMON NAME  : time1
CA    DESCRIPTION  : Part of the state variables for the time simulation.
CP    PARAMETERS   :
C ***************************************************************************
      real swdxdt(15),xo(15),dxdt(15)
      real xn(15),un(15),rn(15),vn(15),yn(15),ynact(15),zn(15),nn(15)
      real xnobs(15),ynobs(15)
      real xndis(15)
      common /simutl/ swdxdt,xo,dxdt
      common /time1/ xn,un,rn,vn,yn,ynact,zn,nn,xnobs,ynobs,xndis

C ***************************************************************************
CN    COMMON NAME  : time2
CA    DESCRIPTION  : Part of the state variables for the time simulation.
CP    PARAMETERS   :
C                      ti - Current time.
C                      dt - Time increment.
C                    tend - Time limit.
C                   c1(i) - Runga-Kutta dxdt summing weights.
C                   c2(i) - Runga-Kutta xn calculation time increments.
C                  obsinc - Observer included(0) or excluded(1).
C                  coninc - Controller included(0) or excluded(1).
C                  setplt - Plot(0) or Not Plot(1) setpoints
C                  inpplt - Plot(0) or Not Plot(1) inputs
C                  outplt - Plot(0) or Not Plot(1) outputs
C                  ostplt - Plot(0) or Not Plot(1) observer states
C                  pstplt - Plot(0) or Not Plot(1) process states
C              stpinp(i) - Number of input to be stepped.
C              stpdat(i) - Time at which input is to be stepped.
C              stpinc(i) - Step size.
C                  twmax - Time : Maximum width (dots).
C                 twstrt - Time : X start point.
C                  thmax - Time : Maximum height (dots).
C                 thstrt - Time : Y start point.
C                 twidth - Time : No. of plots across the page.
C                 theigh - Time : No. of plots down the page.
C                  txdel - Time : Width of one plot (dots).
```

```
C                  tydel - Time : Height of one plot (dots).
C ************************************************************************
      real*4 ti,dt,tend,c1(4),c2(3)
      integer*2 obsinc,coninc,setplt,inpplt,outplt,ostplt,pstplt
      integer*2 stpinp(5)
      real*4 stpdat(5),stpinc(5)
      integer*2 twmax,twstrt,thmax,thstrt,twidth,theigh,txdel,tydel
      common /time2/ ti,dt,tend,c1,c2,
     +               obsinc,coninc,setplt,inpplt,outplt,ostplt,pstplt,
     +               stpinp,stpdat,stpinc,
     +               twmax,twstrt,thmax,thstrt,twidth,theigh,
     +               txdel,tydel

C ************************************************************************
CN    COMMON NAME  : time3
CA    DESCRIPTION  : Arrays for time simulation axes settings.
CP    PARAMETERS   :
C            xtlim - X axes bounds.
C            ytlim - Y axes bounds.
C           tscale - Axes scales (dots/unit).
C          tcentre - Axes centre (dots and units).
C           tgraph - Temporary array used to draw axes.
C ************************************************************************

      real*8 xtlim(3),ytlim(3,10),tscale(2,10),tcentr(4,10)
      integer*2 tgraph(4)
      common /time3/ xtlim,ytlim,tscale,tcentr,tgraph

C ************************************************************************
CN    COMMON NAME  : time4
CA    DESCRIPTION  : Arrays for time simulation axes settings.
CP    PARAMETERS   :
C ************************************************************************
      integer ninc,dinc,rinc,ntyp,dtyp,rtyp,plnoi
      integer nchset(15),dchset(15),rchset(15)
      integer*4 irand
      real nampl,dampl,rampl,nfre,dfre,rfre
      real nchdat(2,15),dchdat(2,15),rchdat(2,15)
      common /time4/ ninc,dinc,rinc,ntyp,dtyp,rtyp,nchset,dchset,rchset,
     +               irand,nampl,dampl,rampl,nfre,dfre,rfre,
     +               nchdat,dchdat,rchdat,plnoi

C ************************************************************************
```

M8) OPTCAD Development/Execution Information

The required subroutines and other program modules for the development of the OPTCAD package are grouped into libraries of object code. These libraries (and some other object files) are then linked to the object file OPTCAD.OBJ compiled from the main program to form the executable file, OPTCAD.EXE. The following instrucution shown in bold type is given to the Microsoft Fortran LINKer:

link
optcad+inicad+inidat+keydat+anal+piplot+pldat+excuse,,,iglib+
math+iolib+edlib+lqg+perlib+simlib/SE:360/E;

The object modules optcad.obj, inicad.obj, inidat.obj, keydat.obj, anal.obj, piplot.obj, pldat.obj and excuse.obj are linked, together with the modules grouped in the libraries iglib.lib, math.lib, iolib.lib, lqg.lib, perlib.lib, and simlib.lib, by the Microsoft LINKer. The tables that follow show which data and object modules are grouped together in the indicated libraries. The modules are indicated in capital letters, while the file in which the modules are located are indicated by lowercase type.

Module	File	Module	File
DEFNAM...	....eddef	DWSHAP...	....edtmat
DWSQRS...	....edtmat	EDDAT....	....eddat
EDDEF....	....eddef	EDGENM...	....edgenm
EDITM....	....edtmat	EDK.....	....edk
EDMAT....	....edtmat	EDPREC...	....edprec
EDPRO....	....edpro	EDQR.....	....edqr
EDQR0....	....edqr0	EDQR1....	....edqr1
EDSYS....	....edsys	FILNAM...	....eddef
KEY.....	....eddef	MOVCRS...	....edtmat
PRECOM...	....edk	RUNFL....	....eddat
SETBLK...	....edtmat	SSSDAT...	....edk
SYST.....	....edk	UPBLK....	....edtmat
WRTITL...	....edtmat		

Table M8.1: Modules in the library edlib.lib

Module	File	Module	File
APPEND...	....strops	ARC.....	....util
ASPACE...	....axes	ATTPG....	....util
BLKFIL...	....util	BLKSEC...	....dohelp
BOX.....	....util	CALAXE...	....axes
CIRC.....	....util	CLRSCR...	....util
CMPSTR...	....strops	COPYST...	....strops
DEFDRV...	....util	DERROR...	....derror
DISP.....	....util	DISPAGE..	....util
DLINE....	....util	DOAX.....	....axes
DOGNUM...	....axes	DOHELP...	....dohelp
DOMENU...	....domenu	DRAWPT...	....plot
DRVHLP...	....dohelp	DRWLET...	....drwlet
DWAXES...	....axes	DWNUMB...	....axes
DXTICK...	....axes	DYTICK...	....axes
ERTONE...	....util	FFIRST...	....util
FHELP....	....mtext	FILL.....	....util
FINDPG...	....dohelp	FLIPG1...	....help
FNEXT....	....util	GETATT...	....util
GETDEC...	....axes	GETERR...	....derror
GETHEL...	....help	GETI2....	....geti2
GETI4....	....geti4	GETKEY...	....getkey
GETNUM...	....getnum	GETOCT...	....axes
GETPG....	....util	GETR4....	....getr4
GETR8....	....getr8	GETWPG...	....util
GMODE....	....util	GNUM.....	....gnums
GPAGE....	....util	INDERR...	....derror
INKEY....	....getkey	INKEY2...	....help
INMENU...	....inmenu	INT10....	....util
INTONE...	....util	INTSCR...	....screen
LENSTR...	....strops	LEVEL....	....util
MOVE.....	....util	NOBLNK...	....strin
NUMSTR...	....numstr	PLOT.....	....util
PLOTPT...	....plot	PNUMS....	....axes
PRHELP...	....dohelp	PRINTCH..	....util
PRPAGE...	....dohelp	PRSCRS...	....pscr
PRSTR....	....util	PRTI2....	....prti2
PRTI4....	....prti4	PRTR4....	....prtr4
PRTR8....	....prtr8	REDERR...	....derror
RETONE...	....util	RJUST....	....strops
RSTERR...	....derror	RSTSCR...	....screen
SCANIN...	....strin	STRCPY...	....strops
STRIN....	....strin	TMODE....	....util
TO_UPPER.	....util	WIPSCR...	....util
WRHEAD...	....wrhead	WRITPG...	....util
WRTSTR...	....util	XDECAD...	....axes
XLIN.....	....axes	XOCTAV...	....axes
XPOS.....	....util	YDATA....	....axes
YPOS.....	....util		

Table M8.2: Modules in the library iglib.lib

Module	File	Module	File
DATA.....	....data	DEFNAM...	....rdmtld
DODIR....	....dodir	FILNAM...	....rdmtld
FILOVW...	....filovw	GETFIL...	....getfil
KEY.....	....lodlqg	LODDAT...	....loddat
LODDEA...	....loddea	LODDES...	....loddes
LODK.....	....lodk	LODLQA...	....lodlqa
LODLQG...	....lodlqg	LODMAT...	....lodmat
LODPER...	....lodper	LODPI....	....lodpi
LODPRO...	....lodpro	LODQR....	....lodqr
MAKNAM...	....maknam	MATDAT...	....matdat
MATST....	....matst	PERCOM...	....matdat
PERDAT...	....matdat	PERLEV...	....matdat
PLPTS....	....lodlqa	PRDERR...	....prderr
PRECOM...	....lodlqg	RDMTLD...	....rdmtld
RUNFL....	....loddat	SAVDAT...	....savdat
SAVDEA...	....savdea	SAVDES...	....savdes
SAVK.....	....savk	SAVLQA...	....savlqa
SAVLQG...	....savlqg	SAVMAT...	....savmat
SAVPER...	....savper	SAVPRO...	....savpro
SAVQR....	....savqr	SSSDAT...	....lodlqg
SYST.....	....lodlqg		

Table M8.3: Modules in the library iolib.lib

Module	File	Module	File
DIMS.....	....optkc	LHS.....	....lhs
OPTKC....	....optkc	OPTKF....	....optkf
OPTMAT...	....optkc	PRECOM...	....optkc
RICC.....	....optkc	SPMATS...	....optkc
SSSDAT...	....optkc	SYNTH....	....synth
SYST.....	....optkc		

Table M8.4: Modules in the library lqg.lib

Module	File	Module	File
ADD.....	....add	CINV.....	....cinv
CMULT....	....cmult	COMHES...	....math
COMLR2...	....math	INVERT...	....invert
MULT.....	....mult	QZVECA...	....math
RNDNRM...	....rndnrm	SIMEQ....	....simeq
SVD.....	....svd	SVMAX....	....svmax
SVMIN....	....svmin	TRANS....	....trans
URAND....	....urand	UTILMT...	....svd
VECS.....	....svd		

Table M8.5: Modules in the library math.lib

Module	File	Module	File
CREATE...	....create	DISVOP...	....disvop
EDPOPT...	....edpopt	INTPER...	....intper
JWMAT....	....jwmat	KEY.....	....peropt
LNAM.....	....peropt	MATS.....	....percal
PERCAL...	....percal	PERCOM...	....percal
PERDAT...	....percal	PERLEV...	....percal
PEROPT...	....peropt	PERPLT...	....perplt
PLPOS....	....peropt	PLPTS....	....peropt
PRECOM...	....percal	RUNFL....	....create
SSSDAT...	....percal	SYST.....	....percal
TOGVOP...	....togvop	UTILMT...	....percal
VIEW.....	....peropt	WRCMAT...	....wrcmat

Table M8.6: Modules in the library perlib.lib

Module	File	Module	File
CHKQT....	....simul	CONCAL...	....concal
DEFNAM...	....simul	DIST.....	....dist
DOSIM....	....simul	EDSCAL...	....simscl
FILNAM...	....simul	GAUIN....	....gauin
GETSET...	....getset	INISIM...	....simul
KEY.....	....gauin	NINDIN...	....nindin
NOIDIS...	....noidis	NOISE....	....noise
OBSSTA...	....obssta	PERSET...	....perset
PLTSIM...	....simul	PLTSTA...	....simul
PRECOM...	....noise	PROSTA...	....prosta
PRSCAL...	....simscl	SETPER...	....setper
SIMGRF...	....simul	SIMOPT...	....simopt
SIMSCL...	....simscl	SIMUL....	....simul
SIMUTL...	....gauin	SININ....	....sinin
SSSDAT...	....noise	SYSNMS...	....simul
SYST.....	....noise	TIME1....	....gauin
TIME2....	....gauin	TIME3....	....gauin
TIME4....	....gauin	TOGDTY...	....togdty
TOGOPT...	....togopt		

Table M8.7: Modules in the library simlib.lib

APPENDIX N

HERIG SYSTEM INFORMATION

N1) HERIG System Subroutines

A table of the HERIG system subroutine names follows on the next page. The name of each subroutine, together with the file in which it is located and the page in the appendix on which it is listed, is given in the table.

HERIG System Subroutines			
Name	File	Description	Page
blmap	BLMAP.FOR	Maps Comment/Diagram Positions on Run Screen	482
conalg	CONALG.FOR	Stabilizing PI Control Algorithm	483
fixdat	FIXDAT.FOR	Convert Input and Output Data for Group Plotting	483
fixout	FIXOUT.FOR	Convert Temperature Output from ADC	484
fixset	FIXSET.FOR	Convert Setpoint Data for Group Plotting	484
fixstp	FIXSTP.FOR	Convert Stepped Setpoint for Group Plotting	485
formpl	FORMPL.FOR	Obtain Format of User Response Data Plots	485
grinit	GRINIT.FOR	Initialize Set of Graph Axes	486
grldat	GRLDAT.FOR	Block Data Containing Group Data Plot Labels	487
hebdat	HEBDAT.FOR	Block Data of HE Block Diag Labels/Dimen	488
heblock	HEBLOCK.FORPrint	Heat Exchanger Block Diagram Shell	488
heplot	HEPLOT.FOR	Drive User/Group Plot Menu and Plot Group Data	489
herig	HERIG.FOR	Initialize Program and Present Main Menu	491
herun	HERUN.FOR	Runs Rig for Modes of Stabilizing Control	492
iaddeg	IADDEG.FOR	Convert Real Degrees Celcius to Integer Number	493
iadper	IADPER.FOR	Convert Real Percentage to Integer Number	493
idegrc	IDEGRC.FOR	Convert Integer Degrees Celcius to Integer Number	493
imxmn	IMXMN.FOR	Check and Clip Max/Min Values to DAC	494
imxmn	IMXMN.FOR	Check and Clip Max/Min Setpoint Values	494
imxmnt	IMXMNT.FOR	Check and Clip Max/Min Temperature	494
inicon	INICON.FOR	Initialize Stabilizing PI Controller Parameters	494
inilog	INILOG.FOR	Initialize Input/Output/Setpoint Logged Values	495
iniovr	INIOVR.FOR	Initialize Input/Output/Setpoint Values	495
init	INIT.FOR	Call and Include Initializing Proc/Block Data	496
inrunv	INRUNV.FOR	Block Data Initializing Run and LQG Variables	496
iperc	IPERC.FOR	Convert Integer Number to Integer Percentage	498
itmxmn	ITMXMN.FOR	Check and Clip Max/Min Values to DAC	498

OPTCAD System Subroutines			
Name	**File**	**Description**	**Page**
keyerr	KEYERR.FOR	Indicate Incorrect Key Pressed	498
logval	LOGVAL.FOR	Log Input/Output/Setpoint Values	498
lqgrun	LQGRUN.FOR	Runs Rig for Modes of LQG control	499
lqgtyp	LQGTYP.FOR	Drive Menu for Options Running under LQG Control	502
maxmin	MAXMIN.FOR	Check and Clip Max/Min Values to DAC	503
mlmvi2	MLMVI2.FOR	Multiply Integer Matrix and Vector	503
mlmvr8	MLMVR8.FOR	Multiply Real*8 Matrix and Vector	503
onepl	ONEPL.FOR	Plot Response Data Chosen by User	504
pervvn	PERVVN.FOR	Perturb Rig Setpoint Gaussianly of Sinuloidally	505
qrldat	GRLDAT.FOR	Block Data Containing User Plot Data and Labels	506
prsel	PRSEL.FOR	Utility for Selecting Data for Plotting	507
rblock	RBLOCK.FOR	Draw Reactor Block on Heat Exch Block Diag Shell	507
rdegrc	RDEGRC.FOR	Convert Integer Number to Real Degrees Celcius	507
rperc	RPERC.FOR	Convert Integer Number to Real Percentage	508
runini	RUNINI.FOR	Initialize Screens/Flags for Running of Rig	508
runkey	RUNKEY.FOR	Service Keyboard Run Mode	508
runrig	RUNRIG.FOR	Drive Rig Running Menu	511
hebdat	HEBDAT.FOR	Block Data of HE Block Diag Run Screen Labels	512
selpl	SELPL.FOR	Controls Selection of Data for User Plots	512
servop	SERVOP.FOR	Service Keyb. while Select Resp Data for Plots	513
typscr	TYPSCR.FOR	Type Headings on Heat Exchanger Block Diagram	514
updscr	UPDSCR.FOR	Update Values of Inp/Outp/Setp on Run Block Diag	516
usplot	USPLOT.FOR	Drive User Plot Option Menu	517

```
C
C     FILE                : BLMAP.FOR
C ***************************************************************************
CN    MODULE NAME         : blmap
CA    FUNCTION            : Maps Comment/Diagram Positions on Run Screen
CS    CALL SEQUENCE       : call blmap()
CI    INPUT PARAMETERS    : None
CO    OUTPUT PARAMETERS   : None
CG    GLOBAL VARIABLES    : Following variables contained in include file:
C                           heblock.inc
C                           resw,resh,vline,hline,tw,th,chw,chh,
C                           x,y,x0,y0,xpos,ypos,
C                           posts(2,10),posfs(2,2),posls(2,4),poscv(2,6),
C                           posst(2,4),post(2,4),optxin,optyin,
C                           tsnam(2,10),fsnam(2,2),lsnam(2,4),cvnam(2,6),
C                           stnam(2,4),tnam(4),
C                           gap,blh,bltop,stptl,stptbl
CM    MODULES CALLED      : None!
CE    ERROR CONDITIONS    : None!
CC    COMMENTS            : Maps the input and output variable names to positions
C                           on the screen.  These names are printed onto the
C                           heat exchanger block diagram in the subroutine
C                           prtscr()
C ***************************************************************************
      subroutine blmap()
$include: 'heblock.inc'

      integer*2 mult,inc,dec,tdec,count
      integer*2 xin,yin,tsxin,tsyin,fsxin,fsyin,lsxin,lsyin,
     +          cvxin,cvyin,stxin,styin,txin,tyin
      integer*2 gap,blh,bltop,stptl,stptbl
      common /scrcon/ gap,blh,bltop,stptl,stptbl

      mult = 0
      inc = 1
      dec = 4
      tdec = 9
      count = 1

C Initial points on screen defined
      x0 = (720 - ( 2*(resw + (hline - 3*resw/4)) + 4*tw + 3*hline) )/2
      y0 = resh + 2*chh

      xin = x0 + resw/4 + hline
      yin = y0 + vline

C Initial positions of name sets defined
      tsxin = xin + tw + 1.5*chw
      tsyin = yin + chh
      fsxin = x0 + resw/4 - 5*chw
      fsyin = y0 + chh
      lsxin = xin + (tw - 4*chw)/2
      lsyin = yin - th/4 + chh
      cvxin = xin + tw + hline - 5*chw
      cvyin = yin - 1.5*chh
      stxin = xin + (tw - 4*chw)/2
      styin = yin - th/4 - 1.5*chh
      txin = xin + (tw - 4*chw)/2
      tyin = yin + 3*th/4 - 1.5*chh

C Initial points of option menu defined
      optxin = 10
      optyin = y0 + 2*vline + th/2 + resh + ((y0 - resh) - 7) + 2 + chh

C Useful dimensions defined
      gap = (y0 - resh) - 7
      blh = 2*resh + 2*vline + th/2
      bltop = y0 - resh

C Defining positions of names on the screen
      do 900 count = 1, 4

C Temperature sensors T2 to T5
         posts(1,(inc+1)) = tsxin + mult*(tw + hline)
         posts(2,(inc+1)) = tsyin
C Temperature sensors T9 to T6
         posts(1,tdec) = cvxin + mult*(tw + hline)
         posts(2,tdec) = tsyin + th/2
C Level sensors L1 to L4
         posls(1,dec) = lsxin + mult*(tw + hline)
         posls(2,dec) = lsyin
C Control valves C2 to C5
         poscv(1,(inc+1)) = cvxin + mult*(tw + hline)
         poscv(2,(inc+1)) = cvyin
C Stirrers S1 to S4
         posst(1,dec) = stxin + mult*(tw + hline)
         posst(2,dec) = styin
C Tank labels No4 to No1
         post(1,dec) = txin + mult*(tw + hline)
         post(2,dec) = tyin

         inc = inc + 1
         dec = dec - 1
         tdec = tdec - 1
         mult = mult + 1

900   continue

C Temperature sensors T1 and T10
      posts(1,1) = xin - 5*chw
      posts(2,1) = yin + th/2 + chh
      posts(1,10) = x0 + resw/4 + 1.5*chw
      posts(2,10) = tsyin
C Control valves C1 and C6
      poscv(1,1) = xin - 5*chw
      poscv(2,1) = cvyin
      poscv(1,6) = xin + 4*(tw + hline) + 1.5*chw
      poscv(2,6) = yin + th/2 - chh
C Flow sensors F1 and F2
      posfs(1,1) = fsxin
      posfs(2,1) = fsyin
      posfs(1,2) = xin + 4*(tw + hline) + 1.5*chw
      posfs(2,2) = yin + th/2 + 1.2*chh

      return
      end
```

```
C                    thus (5-grph) ranges from 4 to 1)
               setper(i) = real(setval(i,(5-grph)))
900         continue
        elseif (opt.eq.2) then
            do 850 i = 1, sampl
C                   Flowmeters F1 to F2 correspond to
C                     setpoints No5 to No6
C                   (grph ranges from 1 to 4:
C                         thus (grph+4) ranges from 5 to 8)
               if (grph.le.2) setper(i) = real(setval(i,(grph+4)))
 850        continue
        else
            do 800 i = 1, sampl
C                   Temp probe T5: setpoint No8
C                   Temp probe T8: setpoint No9
C                   Temp probe T3: setpoint No10
C                   Temp probe T1: setpoint No7
               if (grph.eq.1) setper(i) = real(setval(i,8))
               if (grph.eq.2) setper(i) = real(setval(i,9))
               if (grph.eq.3) setper(i) = real(setval(i,10))
               if (grph.eq.4) setper(i) = real(setval(i,7))
800         continue
        endif
        return
        end
C ***************************************************************************
CH      REVISION HISTORY  :
C       VERSION       BY                  DATE        COMMENT
C         1.1      A. de Waal           06/01/90   Finally Commented
C       FILEND            :
C ***************************************************************************

C
C       FILE              : FIXSTP.FOR
C ***************************************************************************
CN      MODULE NAME       : fixstp
CA      FUNCTION          : Convert Stepped Setpoint for Group Plotting
CS      CALL SEQUENCE     : call fixstp()
CI      INPUT PARAMETERS  : None
CO      OUTPUT PARAMETERS : None
CG      GLOBAL VARIABLES  : Include files:  logval.inc
C                           Common blocks:  /stpper/ stpper
C                                           /runmod/ mode
CM      MODULES CALLED    : None
CE      ERROR CONDITIONS  : None
CC      COMMENTS          : Does the following:
C                                If running in manual mode (mode.le.4),then
C                                   Convert integer value for stepped input
C                                       (only) to real percentage
C                                Elseif running in automatic mode (mode.gt.4),
C                                 then
C                                   Convert integer value for stepped setpoint
C                                       (only) to real percentage
C                                That's It, if anyone has bothered to read this!
C ***************************************************************************
        subroutine fixstp()
$include: 'logval.inc'
        integer*2 mode,stepid
        real*4 stpper(1800)
        common /stpper/ stpper
        common /runmod/ mode
        if (mode.le.4) then
            do 900 i = 1, sampl
               stpper(i) = rperc(inpval(i,stepid))
900         continue
        else
            do 850 i = 1, sampl
               if (stepid.le.6) then
                  stpper(i) = rperc(setval(i,stepid))
               else
                  stpper(i) = rdegrc(setval(i,stepid))
               endif
850         continue
        endif
        return
        end
C ***************************************************************************
CH      REVISION HISTORY  :
C       VERSION       BY                  DATE        COMMENT
C         1.1      A. de Waal           06/01/90   Finally Commented
C       FILEND            :
C ***************************************************************************
C
C       FILE              : FORMPL.FOR
C ***************************************************************************
CN      MODULE NAME       : formpl
CA      FUNCTION          : Obtain Format of User Response Data Plots
CS      CALL SEQUENCE     : call formpl()
CI      INPUT PARAMETERS  : None
CO      OUTPUT PARAMETERS : None
CG      GLOBAL VARIABLES  : Include files:  keys.inc,logval.inc
C                           Common blocks:  /scalop/ scalop,forsel,forno
C                                           /plcomm/ plcomm
CM      MODULES CALLED    : Fortran   :  wrhead,prti2,geti2
C                           Assembler:  WIPSCR,WRTSTR,DISP,INKEY,STRIN,ERTONE
CE      ERROR CONDITIONS  : None
CC      COMMENTS          : Interrogate user on graphics page 0 to obtain format
C                           of rig response data.  Specifications for the
C                           vertical and horizontal axes, as well as comments for
C                           the plots are obtained.
C ***************************************************************************
        subroutine formpl()
        implicit integer*2 (g,S,I)
$include: 'keys.inc'
$include: 'logval.inc'
        integer*2 scalop(2,2,3),forsel,forno,page,key
        character*60 plcomm
        common /scalop/ scalop,forsel,forno
        common /plcomm/ plcomm

        forsel = 1
        forno = 1
        page = 0
        call WIPSCR(0)
        call DISP(0)
        call wrhead(page,((720-9*31)/2),20,31,
```

```
     +             'Choosing format for User Plots.')
      call wrhead(page,(scalop(1,1,1)-120),(scalop(1,1,2)-20),34,
     +'Scale of Horizontal Axis (seconds).')
      call WRTSTR(page,(scalop(1,1,1)-100),scalop(1,1,2),8,
     +'Maximum:')
      call WRTSTR(page,(scalop(1,2,1)-100),scalop(1,2,2),8,
     +'Minimum:')
      call wrhead(page,(scalop(2,1,1)-120),(scalop(2,1,2)-20),31,
     +   'Scale of Vertical Axis (units).')
      call WRTSTR(page,(scalop(2,1,1)-100),scalop(2,1,2),8,
     +'Maximum:')
      call WRTSTR(page,(scalop(2,2,1)-100),scalop(2,2,2),8,
     +'Minimum:')
      call wrhead(page,90,145,35,'Enter comment for chosen user plot.')

      do 45 k = 1, 2
          do 46 j = 1,2
              call prti2(page,scalop(k,j,1),scalop(k,j,2),4,'(I4)',
     +                   4,scalop(k,j,3))
46        continue
45    continue
      call WRTSTR(page,90,165,60,plcomm)

50    continue

      if (forsel.ne.3) then
100       continue
          key = geti2(page,scalop(forsel,forno,1),
     +    scalop(forsel,forno,2),4,'(I4)',4,scalop(forsel,forno,3))
          if ((key.ne.esck).and.(key.ne.upk).and.(key.ne.downk))
     +        goto 100

          call WRTSTR(page,200,240,32,
     +         '                                  ')
          call WRTSTR(page,200,260,32,
     +         '                                  ')
          if ((forno.eq.1).and.(forsel.eq.1)) then
              if (scalop(forsel,forno,3).gt.((sampl-1)/hertz)) then
                  call ERTONE()
                  call LEVEL(0)
                  call WRTSTR(page,200,240,27,
     +                        'Maximum time cannot exceed ')
                  call prti2(page,(200+27*9),240,4,('I4'),4,
     +                       int((sampl-1)/hertz))
                  call LEVEL(1)
                  call WRTSTR(page,200,260,27,
     +                        'Correct value and continue.')
                  key = INKEY(2)
              endif
          endif
      else
200       continue
          key = STRIN(page,90,165,60,plcomm)
          if ((key.ne.esck).and.(key.ne.upk).and.(key.ne.downk))
     +        goto 200
      endif


      if (key.eq.upk) then
          forno = forno - 1
          if (forsel.eq.3) then
              forsel = 2
              forno = 2
          elseif (forno.eq.0) then
              forsel = forsel - 1
              forno = 2
              if (forsel.eq.0) then
                  forsel = 3
                  forno = 1
              endif
          endif
      elseif (key.eq.downk) then
          forno = forno + 1
          if (forsel.eq.3) then
              forsel = 1
              forno = 1
          elseif (forno.eq.3) then
              forsel = forsel + 1
              forno = 1
              if (forsel.eq.4) forsel = 1
          endif
      else
          if (key.ne.esck) call ERTONE()
      endif
      if (key.ne.esck) goto 50
      return
      end
C ******************************************************************************
CH    REVISION HISTORY  :
C     VERSION       BY                DATE        COMMENT
C       1.1     A. de Waal          06/01/90   Finally Commented
C     FILEND            :
C ******************************************************************************


C
C     FILE              : GRINIT.FOR
C ******************************************************************************
CN    MODULE NAME       : grinit
CA    FUNCTION          : Initialize Set of Graph Axes
CS    CALL SEQUENCE     : call grinit(page,nums,xtype,x,y,dim1,dim2,dim3,dim4,
C                                     scale,centre)
CI    INPUT PARAMETERS  : page:  integer - Page number on which to be drawn
C                         nums:  integer - Axes routine function number
C                                          0 : No numbers on axes
C                                          1 : Numbers on the axes
C                                          2 : No axes or numbers to be drawn
C                         xtype: integer - Type of axes to be drawn
C                            1 : linear
C                            2 : decade
C                            3 : octave
C                         x(3) : real*8  - X axes dimensions :
C                                    x(1): Xmaximum.
C                                    x(2): Xminimum.
C                                    x(3): X axes or origin.
C                         y(3) : real*8  - Y axes dimensions :
```

```
c                                   y(1): Ymaximum.
c                                   y(2): Yminimum.
c                                   y(3): Y axes or origin.
c                   dim1..dim4 : integer*2 - Dimensions  of   the
c                                area on  the  graphics  page  in
c                                which  the set of axes are to be
c                                drawn.
c                                dim1: X co-ord  of  the area
c                                      : lower left corner.
c                                dim2: Y co-ord of  the  area
c                                      : lower left corner.
c                                dim3: Width of  area  speci-
c                                      fied in dots.
c                                dim4: Height of area  speci-
c                                      fied in dots.
c
CO    OUTPUT PARAMETERS:
c                 scale(2) - (real*8) The   axes    to   page
c                                scaling   factors   (  dots/axis
c                                unit).
c                                scale(1) - x axes scale.
c                                scale(2) - y axes scale.
c                centre(4) - (real*8) The origin of  the axes
c                                on the actual  graphics page and
c                                the  axes  origin  (as given  in
c                                x(3) and y(3)).
c                                centre(1) - x position of origin
c                                      specified in dots.
c                                centre(2) - y position of origin
c                                      specified in dots.
c                                centre(3) - x co-ord  of  origin
c                                      specified   in  axes
c                                      units.
c                                centre(4) - y co-ord  of  origin
c                                      specified  in   axes
c                                      units.
CG    GLOBAL VARIABLES  : None
CO    OUTPUT PARAMETERS :
CM    MODULES CALLED    : Fortran  :  dwaxes
CE    ERROR CONDITIONS  : None
CC    COMMENTS          : Utility routine to interface to graph axes drawing
c                         routine dwaxes.
c ***************************************************************************
      subroutine grinit(page,nums,xtype,x,y,dim1,dim2,dim3,dim4,
     +                  scale,centre)
      integer*2 page,nums,xtype,dim1,dim2,dim3,dim4,graph(4)
      real*8 x,y,scale,centre
      graph(1) = dim1
      graph(2) = dim2
      graph(3) = dim3
      graph(4) = dim4
      call dwaxes(page,nums,xtype,x,y,graph,scale,centre)
      return
      end
c ***************************************************************************
CH    REVISION HISTORY  :
c     VERSION     BY                  DATE         COMMENT
c       1.1     A. de Waal          06/01/90   Finally Commented
c     FILEND            :
c ***************************************************************************

c
c     FILE              : GRLDAT.FOR
c ***************************************************************************
CN    MODULE NAME       : grldat
CA    FUNCTION          : Block Data Containing Group Data Plot Labels
CS    CALL SEQUENCE     : Not a subroutine
CI    INPUT PARAMETERS  : n/a
CO    OUTPUT PARAMETERS : n/a
CG    GLOBAL VARIABLES  : Include files:  grlab.inc
CM    MODULES CALLED    : n/a
CE    ERROR CONDITIONS  : n/a
CC    COMMENTS          : Block data section containing label data for all the
c                         group response data plots.
c ***************************************************************************
      block data grldat
$include: 'grlab.inc'
      data (xlpos(i),i=1,4)/6,6,362,362/
      data (ylpos(i),i=1,4)/35,175,175,35/
      data lablen/39/,stepll/60/,stinll/2/,grbrfl/60/
      data (((label(k,j,i),i=1,4),j=1,4),k=1,2)
C AUTMAN k = 1  (manual mode)
C  option j = 1
     +           /'1:Inp C2[%]  2:Out L4[%]                ',
     +            '1:Inp C3[%]  2:Out L3[%]                ',
     +            '1:Inp C4[%]  2:Out L2[%]                ',
     +            '1:Inp C5[%]  2:Out L1[%]                ',
C  option j = 2
     +            '1:Inp C1[%]  2:Out F1[%]                ',
     +            '1:Inp C6[%]  2:Out F2[%]                ',
     +            '1:Inp C1[%]  2:Out T10[°C]              ',
     +            '1:Inp C6[%]  2:Out T6[°C]               ',
C  option j = 3
     +            '1:Inp S1[%]  2:Out T5[°C]               ',
     +            '1:Inp S2[%]  2:Out T8[°C]               ',
     +            '1:Inp S3[%]  2:Out T3[°C]               ',
     +            '1:Inp S4[%]  2:Out T1[°C]               ',
C  option j = 4
     +            '1:Inp S1[%]  2:Out T7[°C]               ',
     +            '1:Inp S2[%]  2:Out T4[°C]               ',
     +            '1:Inp S3[%]  2:Out T9[°C]               ',
     +            '1:Inp S4[%]  2:Out T2[°C]               ',
C AUTMAN k = 2  (automatic mode)
C  option j = 1
     +            '1:Inp C2[%]  2:Out L4[%]  3:Set L4[%]   ',
     +            '1:Inp C3[%]  2:Out L3[%]  3:Set L3[%]   ',
     +            '1:Inp C4[%]  2:Out L2[%]  3:Set L2[%]   ',
     +            '1:Inp C5[%]  2:Out L1[%]  3:Set L1[%]   ',
C  option j = 2
     +            '1:Inp C1[%]  2:Out F1[%]  3:Set F1[%]   ',
     +            '1:Inp C6[%]  2:Out F2[%]  3:Set F2[%]   ',
     +            '1:Inp C1[%]  2:Out T10[°C]              ',
     +            '1:Inp C6[%]  2:Out T6[°C]               ',
C  option j = 3
     +            '1:Inp S1[%]  2:Out T5[°C]  3:Set T5[°C]',
     +            '1:Inp S2[%]  2:Out T8[°C]  3:Set T8[°C]',
```

```
     +            '1:Inp S3[%]  2:Out T3[°C]  3:Set T3[°C]',
     +            '1:Inp S4[%]  2:Out T1[°C]  3:Set T1[°C]',
C  option j = 4
     +            '1:Inp S1[%]  2:Out T7[°C]               ',
     +            '1:Inp S2[%]  2:Out T4[°C]               ',
     +            '1:Inp S3[%]  2:Out T9[°C]               ',
     +            '1:Inp S4[%]  2:Out T2[°C]               '/
      data (stplab(i),i=1,2)
     +/'Open Loop Step Tests:     Trace 4 represents Stepped Input: ',
     + 'Closed Loop Step Tests: Trace 4 represents Stepped Setpoint:'/
      data ((stpins(j,i),i=1,10),j=1,2)
     +    /'C1','C2','C3','C4','C5','C6',
     +     'S1','S2','S3','S4',
     +     'L1','L2','L3','L4','F1','F2',
     +     'T1','T5','T8','T3'/
      data (grbrf(i),i=1,2)
     +/'O/L Response: C:Valve S:Stirrer L:Level F:Flow T:Temperature',
     + 'C/L Response: C:Valve S:Stirrer L:Level F:Flow T:Temperature'/
      data bl60
     +/'                                                            '/
      data bl2/'  '/
      end
C ***************************************************************************
CH    REVISION HISTORY  :
C     VERSION       BY                  DATE        COMMENT
C       1.1      A. de Waal           06/01/90   Finally Commented
C     FILEND          :
C ***************************************************************************
C
C     FILE              : HEBDAT.FOR
C ***************************************************************************
CN    MODULE NAME       : hebdat
CA    FUNCTION          : Block Data of HE Block Diag Labels/Dimen
CS    CALL SEQUENCE     : Not a subroutine
CI    INPUT PARAMETERS  : n/a
CO    OUTPUT PARAMETERS : n/a
CG    GLOBAL VARIABLES  : Common blocks:  /blvals/ resw,resh,vline,hline,tw,
C                                                  th,chw,chh
C                                         /blnam/  tsnam,fsnam,lsnam,cvnam,
C                                                  stnam,tnam
CM    MODULES CALLED    : n/a
CE    ERROR CONDITIONS  : n/a
CC    COMMENTS          : Block data section containing label and dimensioning
C                         data for heat exchanger block diagram.
C ***************************************************************************
      block data hebdat
      integer*2 resw,resh,vline,hline,tw,th,chw,chh
      character*4 tsnam(2,10),fsnam(2,2),lsnam(2,4),cvnam(2,6),
     +            stnam(2,4),tnam(4)

      common /blvals/ resw,resh,vline,hline,tw,th,
     +               chw,chh
      common /blnam/ tsnam,fsnam,lsnam,cvnam,stnam,tnam

      data resw/80/,resh/50/,vline/50/,hline/70/,tw/60/,th/100/,
     +     chw/9/,chh/12/
      data ((tsnam(j,i),i=1,10),j=1,2)/' T1 ',' T2 ',' T3 ',' T4 ',
     +                                 ' T5 ',' T6 ',' T7 ',' T8 ',
     +                                 ' T9 ',' T10',
     +                                 '7:T1',' T2 ','0:T3',' T4 ',
     +                                 '8:T5',' T6 ',' T7 ','9:T8',
     +                                 ' T9 ',' T10'/
      data ((fsnam(j,i),i=1,2),j=1,2)/' F1 ',' F2 ','5:F1','6:F2'/
      data ((lsnam(j,i),i=1,4),j=1,2)/' L1 ',' L2 ',' L3 ',' L4 ',
     +                                '1:L1','2:L2','3:L3','4:L4'/
      data ((cvnam(j,i),i=1,6),j=1,2)/'1:C1','2:C2','3:C3','4:C4',
     +                                '5:C5','6:C6',
     +                                ' C1 ',' C2 ',' C3 ',' C4 ',
     +                                ' C5 ',' C6 '/
      data ((stnam(j,i),i=1,4),j=1,2)/'7:S1','8:S2','9:S3','0:S4',
     +                                ' S1 ',' S2 ',' S3 ',' S4 '/
      data (tnam(i),i=1,4)/'No.1','No.2','No.3','No.4'/

      end
C ***************************************************************************
CH    REVISION HISTORY  :
C     VERSION       BY                  DATE        COMMENT
C       1.1      A. de Waal           06/01/90   Finally Commented
C     FILEND          :
C ***************************************************************************
C
C     FILE              : HEBLOCK.FOR
C ***************************************************************************
CN    MODULE NAME       : heblock
CA    FUNCTION          : Print Heat Exchanger Block Diagram Shell
CS    CALL SEQUENCE     : call heblock()
CI    INPUT PARAMETERS  : None
CO    OUTPUT PARAMETERS : None
CG    GLOBAL VARIABLES  : Include files:  heblock.inc
CM    MODULES CALLED    : Fortran   :  rblock(reac), arhead(type,size)
C                         Assembler:  BOX(x,y,width,height), MOVE(x,y),
C                                     DLINE(x,y), GPAGE(page),
C                                     WRTSTR(page,x,y,length,string),
CE    ERROR CONDITIONS  : None
CC    COMMENTS          : Draws heat exchanger block diagram shell on page 0.
C                         Symbols for the tanks, piping and reservoirs are
C                         drawn to give a physically meaningful diagram of the
C                         system on the screen for easy understanding by the
C                         user.  Instrumentation and readings can be printed
C                         in an appropriate format on this diagram shell while
C                         the system is being run.
C ***************************************************************************
      subroutine heblock()
$include: 'heblock.inc'
      integer*2 reacno,gap
      integer*2 reac,borw,borh,key,page

      external DISP
      external BOX
      external MOVE
      external DLINE
      external WRTSTR
      external rblock
      external arhead

      reac = 4
```

```
      borw = 710
      borh = 338
      key = 0
      page = 0

      x = 5
      y = 343
      call BOX(x,y,borw,borh)

      x = 7
      gap = (y0 - resh) - 7
      y = 7 + gap + 2*(resh + vline) + th/2 + gap
      borw = 706
      borh = gap + 2*(resh + vline) + th/2 + gap
      call BOX(x,y,borw,borh)

      borh = 341 - (y + 2)
      y = 341
      call BOX(x,y,borw,borh)

      call BOX(x0,y0,resw,resh)
      xpos = x0 + (resw - 4*chw)/2
      ypos = y0 - resh + 4*chh/3
      call WRTSTR(page,xpos,ypos,4,'Cold')
      xpos = x0 + (resw - 7*chw)/2
      ypos = ypos + chh
      call WRTSTR(page,xpos,ypos,7,'Reserv.')
      xpos = x0 + (resw - 8*chw)/2
      ypos = ypos + chh
      call WRTSTR(page,xpos,ypos,8,'(Carbon)')
      x = x0 + resw/4
      y = y0

      call MOVE(x,y)
      y = y + vline
      call DLINE(x,y)
      x = x + hline
      call DLINE(x,y)

      call arhead(1,8)

      y = y + th/2
      call MOVE(x,y)
      x = x - hline
      call DLINE(x,y)
      y = y + vline
      call DLINE(x,y)
      call arhead(3,8)
      xpos = x - (5*chw/2)
      ypos = y + chh
      call WRTSTR(page,xpos,ypos,5,'Waste')
      x = x + hline
      y = y - vline - th/2

      do 100, reacno = 1, 4
          call rblock(reac)
          reac = reac - 1
100   continue

      call MOVE(x,y)
      y = y - vline
      call DLINE(x,y)
      call arhead(4,8)
      xpos = x - (5*chw/2)
      ypos = y - 2*chh
      call WRTSTR(page,xpos,ypos,5,'Final')
      xpos = x - (7*chw/2)
      ypos = ypos + chh
      call WRTSTR(page,xpos,ypos,7,'Product')

      y = y + vline + th/2
      call MOVE(x,y)
      y = y + vline
      call DLINE(x,y)
      x = x - (3*resw/4)
      y = y + resh
      call BOX(x,y,resw,resh)
      xpos = x + (resw - 3*chw)/2
      ypos = y - resh + 4*chh/3
      call WRTSTR(page,xpos,ypos,3,'Hot')
      xpos = x + (resw - 7*chw)/2
      ypos = ypos + chh
      call WRTSTR(page,xpos,ypos,7,'Reserv.')
      xpos = x + (resw - 6*chw)/2
      ypos = ypos + chh
      call WRTSTR(page,xpos,ypos,6,'(Pulp)')

      return
      end
C *****************************************************************************
CH    REVISION HISTORY  :
C     VERSION        BY                 DATE         COMMENT
C       1.0      A. de Waal           17/05/89      Creation
C       1.1      A. de Waal           06/01/90   Finally Commented
C     FILEND           :
C *****************************************************************************

C
C     FILE             : HEPLOT.FOR
C *****************************************************************************
CN    MODULE NAME      : heplot
CA    FUNCTION         : Drive User/Group Plot Menu and Plot Group Data
CS    CALL SEQUENCE    : call heplot()
CI    INPUT PARAMETERS : None
CO    OUTPUT PARAMETERS : None
CG    GLOBAL VARIABLES : Include files:  logval.inc,grlab.inc
CM    MODULES CALLED   : Fortran  :  wrhead,prtr4,prtr8,fixstp,dwaxes,fixdat,
C                                    drawpt,drwlet,fixset,usplot
C                        Assembler:  WIPSCR,DISP,WRTSTR,DOMENU,MOVE,DLINE,
C                                    LEVEL,INKEY
CE    ERROR CONDITIONS : None
CC    COMMENTS         : Does the following:
C                            - Displays plot choice menu as well as plot axes
C                              and sample data information
C                            - Drives menu for choice of one of group plots or
C                              user-specified plot
```

```
C                               - If one of group-plots selected (option 4..1),
C                                 plots out chosen group of plots of response
C                                 data on appropriate axes with appropriate
C                                 labels
C                               - If user plot selected, calls user plot
C                                 subroutine
C ***************************************************************************
      subroutine heplot()
      implicit integer*2 (D,I)
$include: 'logval.inc'
$include: 'grlab.inc'
      integer*2 i,j,pos,mode,key,autman
      integer*2 graph,opt13,page,xscale/1/,dots/1/,join/2/,nums/1/
      integer*2 grfdim(4)
      real*8 pmax,pmin,xp(3),yp(3)
      real*4 time(1800)
      real*8 scale(2),centre(4)
      real*4 inper(1800),outper(1800),setper(1800),stpper(1800)
      common /runmod/ mode
      common /manper/ inper,outper
      common /autper/ setper
      common /stpper/ stpper

      if (mode.le.4) then
         autman = 1
      else
         autman = 2
      endif

100   continue
      page = 0
      nums = 1
      call WIPSCR(0)
      call DISP(0)
      call wrhead(0,50,20,18,'Plotting Rig Data.')
      call wrhead(0,20,40,30,'Current rig data parameters :-')
      call WRTSTR(0,20,55,22,'Data from rig sampled ')
      pos = 20 + 22*9
      call prti2(0,pos,55,4,'(i4)',4,sampl)
      call WRTSTR(0,(pos+5*9),55,25,' times at a frequency of ')
      call prtr4(0,(pos+30*9),55,7,'(f7.2)',7,hertz)
      call WRTSTR(0,(pos+38*9),55,6,'Hertz.')
      call wrhead(0,20,90,19,'Axes Information :-')
      call WRTSTR(0,20,105,30,'Horizontal Axes : Time [secs] :')
      totime = real(sampl-1)/hertz
      call prtr4(0,300,105,7,'(f10.4)',10,totime)
      call WRTSTR(0,20,120,43,
     +'Vertical Axes   : Percentage of Full Scale.')
      pmax = 4095.0
      pmin = 0.0
      call WRTSTR(0,210,135,24,'Max :              Min : ')
      call prtr8(0,260,135,7,'(g10.4)',10,pmax)
      call prtr8(0,430,135,7,'(g10.4)',10,pmin)
      call WRTSTR(0,20,150,43,
     +'                : Degrees Celcius (Temp).  ')
      call WRTSTR(0,210,165,24,'Max :              Min : ')
      call prtr8(0,260,165,7,'(g10.4)',10,pmax)
      call prtr8(0,430,165,7,'(g10.4)',10,pmin)

C
      opt13 = DOMENU(13)
C
      if ((opt13.le.4).and.(opt13.ne.0)) then
          page = 1
          call WIPSCR(page)
          call DISP(page)

          xp(1) = DBLE(INT2((sampl-1)/(5*hertz)))*5.0
          if ( (MOD( ((sampl-1)/hertz),5) ).gt.0 ) xp(1) = xp(1) + 5.0
          xp(2) = 0.0
          xp(3) = 0.0
          yp(1) = pmax
          yp(2) = pmin
          yp(3) = pmin

          do 199 i = 0,(sampl-1)
              time(i+1) = REAL(i)/(hertz)
199       continue

          call MOVE(4,301)
          call DLINE(715,301)
          call MOVE(4,162)
          call DLINE(715,162)
          call MOVE(360,301)
          call DLINE(360,22)
          call WRTSTR(page,((720-49*9)/2),18,49,
     +            'Heat Exchanger Counter-Current Process Responses.')
          call MOVE(4,22)
          call DLINE(715,22)
          if ((mode.eq.3).or.(mode.eq.4).or.
     +        (mode.eq.7).or.(mode.eq.8)) then
              call fixstp()
              call WRTSTR(page,((720-65*9)/2),313,
     +                    stepll,bl60)
              call WRTSTR(page,((720-65*9)/2+60*9),313,
     +                    stinll,bl2)
              call LEVEL(0)
              call WRTSTR(page,((720-65*9)/2),313,
     +                    stepll,stplab(autman))
              call WRTSTR(page,((720-65*9)/2+60*9),313,
     +                    stinll,stpins(autman,stepid))
              call LEVEL(1)
          else
              call WRTSTR(page,((720-65*9)/2),313,
     +                    stepll,bl60)
              call WRTSTR(page,((720-65*9)/2+60*9),313,
     +                    stinll,bl2)
              call LEVEL(0)
              call WRTSTR(page,((720-60*9)/2),313,
     +                    grbrfl,grbrf(autman))
              call LEVEL(1)
          endif

          do 900 graph = 1, 4
              if (graph.eq.1) then
C Lower Left Corner X co-ord
```

```
                grfdim(1) = 10
C Lower Left Corner Y co-ord
                grfdim(2) = 163
C Width
                grfdim(3) = 335
C Height
                grfdim(4) = 110
                call WRTSTR(page,6,(163-110)-4,39,
     +            '[units] =  0:0%, 20°C;  4095:100%, 80°C')
                call WRTSTR(page,333,151,3,'[s]')
            elseif (graph.eq.2) then
                grfdim(1) = 10
                grfdim(2) = 303
                grfdim(3) = 335
                grfdim(4) = 110
                call WRTSTR(page,6,(303-110)-4,39,
     +            '[units] =  0:0%, 20°C;  4095:100%, 80°C')
                call WRTSTR(page,333,290,3,'[s]')
            elseif (graph.eq.3) then
                grfdim(1) = 365
                grfdim(2) = 303
                grfdim(3) = 335
                grfdim(4) = 110
                call WRTSTR(page,362,(303-110)-4,39,
     +            '[units] =  0:0%, 20°C;  4095:100%, 80°C')
                call WRTSTR(page,689,290,3,'[s]')
            else
                grfdim(1) = 365
                grfdim(2) = 163
                grfdim(3) = 335
                grfdim(4) = 110
                call WRTSTR(page,362,(163-110)-4,39,
     +            '[units] =  0:0%, 20°C;  4095:100%, 80°C')
                call WRTSTR(page,689,151,3,'[s]')
            endif
            call LEVEL(0)
            call WRTSTR(page,xlpos(graph),ylpos(graph),
     +                  lablen,label(autman,opt13,graph))
            call LEVEL(1)
            call dwaxes(page,nums,xscale,xp,yp,grfdim,scale,centre)
            call fixdat(opt13,graph)
            call drawpt(page,xscale,time,inper,sampl,centre,
     +                  scale,join)
            xtrid = time(sampl)
            call drwlet(page,xtrid,inper(sampl),
     +                  centre,scale,'1')
            call drawpt(page,xscale,time,outper,sampl,centre,
     +                  scale,dots)
            xtrid = time(sampl) + (1.0/hertz)*7.0/(SNGL(scale(1)))
            call drwlet(page,xtrid,outper(sampl),
     +                  centre,scale,'2')
            if ((mode.ge.5).and.(opt13.le.3)) then
                if(.not.( (opt.eq.3).and.
     +                    ((graph.eq.3).or.(graph.eq.4)) ) ) then
                    call fixset(opt13,graph)
                    call drawpt(page,xscale,time,setper,sampl,
     +                          centre,scale,join)
                    xtrid = time(sampl) +
     +                            2.0*7.0/(SNGL(scale(1)))
                    call drwlet(page,xtrid,setper(sampl),
     +                        centre,scale,'3')
                endif
            endif
            if ((mode.eq.3).or.(mode.eq.4).or.
     +          (mode.eq.7).or.(mode.eq.8)) then
                call drawpt(page,xscale,time,stpper,sampl,centre,
     +                      scale,join)
                xtrid = time(sampl) +
     +                      3.0*7.0/(SNGL(scale(1)))
                call drwlet(page,xtrid,stpper(sampl),
     +                      centre,scale,'4')
            endif
900         continue
            key = INKEY(1)
        endif
        if (opt13.eq.5) then
            call usplot()
        endif
        if (opt13.ne.0) goto 100
        return
        end
C ***************************************************************************
CH      REVISION HISTORY  :
C       VERSION       BY                  DATE         COMMENT
C         1.1      A. de Waal           06/01/90    Finally Commented
C       FILEND            :
C ***************************************************************************

C
C       FILE              : HERIG.FOR
C ***************************************************************************
CN      MODULE NAME       : herig
CA      FUNCTION          : Initialize Program and Present Main Menu
CS      CALL SEQUENCE     : Main Program
CI      INPUT PARAMETERS  : None
CO      OUTPUT PARAMETERS : None
CG      GLOBAL VARIABLES  : None
CM      MODULES CALLED    : Fortran  :  init,inmenu,wrhead,hepar,runrig,heplot
C                           Assembler:  INTONE,DOMENU,RETONE,RSTSCR,WIPSCR,
C                                       WRTSTR,REDERR
CE      ERROR CONDITIONS  : INT10 GRAPHIX package not resident in the system,
C                           HERCULES card not installed.
C
CC      COMMENTS          : This program performs the following functions :-
C                             i) Initialises the 2 graphics pages.
C                            ii) Initialises variables.
C                           iii) Initialises the system timer interrupt for
C                                error tone operation.
C                            iv) Drives main program menu.
C
C                           This package requires the following in order to run:
C                             i) IBM PC - XT/AT (compatable, preferably an AT)
C                            ii) PC to contain at least 512 K of RAM.
C                           iii) PC to contain a HERCULES graphics card.
C                            iv) PC to have access to at least one disk drive.
C                             v) INT10 GRAPHIX package is also rquired by the CAD
```

```
C                               package.
C                           vi) DT2815 Digital-to-Analog Card and DT 2801
C                               Analog-to-Digital Card (program can run
C                               without cards if specify "test" instead of
C                               "runs" in herig.sys file)
C ********************************************************************************
      program herig
      implicit integer*2 (D)

      integer*2 opt1

C Initialize variables used in subroutines. (init.for)
      call init()

C Initialize tone interrupt. (util.asm)
      call INTONE()

99    continue

C Initialize menu screens and menus. (inmenu.for)
      call inmenu()

C Write heading. (wrhead.for)
      call wrhead(0,180,35,40,
     +            'Heat Exchanger Rig Analysis and Control.')

C Get chosen option. (domenu.asm)
      opt1 = DOMENU(1)
C Blank over heading. (util.asm)
      call WRTSTR(0,180,35,41,
     +                                                    ')
      if (opt1.eq.1) then
          call hepar()
      elseif (opt1.eq.2) then
          call runrig()
      elseif (opt1.eq.3) then
          call hepiot()
      endif
      if (opt1.ne.4) goto 99

C Reset INT24H vector. (derror.asm)
      call REDERR()

C Reset screen to text mode. (screen.asm)
      call RSTSCR()

C Reset tone interrupt vector. (util.asm)
      call RETONE()
      stop
      end
C ********************************************************************************
CH    REVISION HISTORY  :
C     VERSION       BY                DATE        COMMENT
C       1.1     A. de Waal          06/01/90   Finally Commented
C     FILEND            :
C ********************************************************************************
C
C     FILE              : HERUN.FOR
C ********************************************************************************
CN    MODULE NAME       : herun
CA    FUNCTION          : Runs Rig for Modes of Stabilizing Control
CS    CALL SEQUENCE     : call herun()
CI    INPUT PARAMETERS  : None
CO    OUTPUT PARAMETERS : None
CG    GLOBAL VARIABLES  : Include files:  heblock.inc,time.inc
C                         Common blocks:  /runmod/ mode
C                                         /runflg/ chscr,log,quit
C                                         /adstat/ adstat
CM    MODULES CALLED    : Fortran  :  runini,readop,conalg,wrinp,logval,runkey,
C                                     typscr,updscr,prtr4
C                         Assembler:  GETTIM,CLRSCR
CE    ERROR CONDITIONS  : None
CC    COMMENTS          : This routine first initializes the mode of operation
C                         and the run screen, and then enters the stabilizing
C                         control mode main running loop.  The rig is run by
C                         this loop in the various stabilizing control
C                         modes defined in the subroutine runkey().
C ********************************************************************************
      subroutine herun()
$include: 'heblock.inc'
$include: 'time.inc'
      real*4 ctime/0.0/,difft/0.0/
      integer*2 runcyc,timewt,ihr,imin,isec,i100th
      integer*2 mode
      logical*2 chscr,log,quit
      character*4 adstat
      common /runmod/ mode
      common /runflg/ chscr,log,quit
      common /adstat/ adstat
      assign 900 to runcyc
      assign 800 to timewt

      call runini()

900   continue

C Printing out the sample time
      xpos = optxin + 2*chw + 64*chw + 7*chw
      ypos = optyin + 4*chh/3
      call prtr4(page,xpos,ypos,6,'(F4.2)',4,difft)

C Getting the time at the beginning of the loop
      call GETTIM(ihr,imin,isec,i100th)
      ctime = real(isec) + (real(i100th))/100.0

C Performing the reading of rig outputs, the implementation of the
C control algorithm, the writing of the rig inputs and the logging of
C inputs, outputs and setpoints.
      if ((adstat.eq.'runs').or.(adstat.eq.'RUNS')) call readop()
      if ((mode.ge.5).and.(.not.quit)) call conalg()
      if ((adstat.eq.'runs').or.(adstat.eq.'RUNS')) call wrinp()
      if (log) call logval()

800   continue

C Checking the keyboard for instructions to:
```

```
C         - change the mode of operation
C         - change inputs or setpoints
C         - step inputs or setpoints
      call runkey()

C Change the run screen headings if necessary (eg at a change of operating
C                                                    mode)
      if (chscr) call typscr()

C Update ,on the run screen, the values of
C         - inputs
C         - setpoints
C         - outputs
      call updscr()

C Getting the time at the end of the loop and calculating the time elapsed
C since the beginning of the loop
      call GETTIM(ihr,imin,isec,i100th)
      difft = real(isec) + (real(i100th)/100.0)
      if (difft.lt.ctime) difft = difft + 60.0
      difft = abs(difft - ctime)

C If the elapsed time is less than the desired sample time (1 sec), scan the
C keyboard and update the screen until the desired time has elapsed
      if (difft.le.dt) goto timewt

C Restart the cycle if not quit mode
      if (.not.quit) goto runcyc

      call CLRSCR()
      return
      end
C ***************************************************************************
CH    REVISION HISTORY  :
C     VERSION      BY                 DATE        COMMENT
C       1.0     A. de Waal          26-05-89    Commented
C       1.1     A. de Waal          06/01/90    Finally Commented
C     FILEND            :
C ***************************************************************************

C
C     FILE              : IADDEG.FOR
C ***************************************************************************
CN    MODULE NAME       : iaddeg
CA    FUNCTION          : Convert Real Degrees Celcius to Integer Number
CS    CALL SEQUENCE     : ival = iaddeg(degrc)
CI    INPUT PARAMETERS  : degrc : real    - Real degree celcius number in the
C                                           range 20 to 80 degrees to be
C                                           converted
CO    OUTPUT PARAMETERS : iaddeg: integer - Converted integer value in the
C                                           range 0 to 4095
CG    GLOBAL VARIABLES  : None
CM    MODULES CALLED    : None
CE    ERROR CONDITIONS  : None
CC    COMMENTS          : Converts real expression for degrees celcius in the
C                         range 20 to 80 degrees to an integer expression for
C                         temperature in range 0 to 4095
C ***************************************************************************
      integer*2 function iaddeg(degrc)
      real*4 degrc
      iaddeg = INT(ANINT(
     +                (4095.0/60.0)*(degrc-20.0) ))
      return
      end
C ***************************************************************************
CH    REVISION HISTORY  :
C     VERSION      BY                 DATE        COMMENT
C       1.1     A. de Waal          06/01/90    Finally Commented
C     FILEND            :
C ***************************************************************************
C
C     FILE              : IADPER.FOR
C ***************************************************************************
CN    MODULE NAME       : iadper
CA    FUNCTION          : Convert Real Percentage to Integer Number
CS    CALL SEQUENCE     : ival = iadper(perc)
CI    INPUT PARAMETERS  : perc  : real    - Real percentage in the
C                                           range 0 to 100 percent to be
C                                           converted
CO    OUTPUT PARAMETERS : iadper: integer - Converted integer value in the
C                                           range 0 to 4095
CG    GLOBAL VARIABLES  : None
CM    MODULES CALLED    : None
CE    ERROR CONDITIONS  : None
CC    COMMENTS          : Converts real expression for percentage in the
C                         range 0 to 100 percent to an integer expression for
C                         value in range 0 to 4095
C ***************************************************************************
      integer*2 function iadper(perc)
      real*4 perc
      iadper = INT(ANINT( 40.95*perc ))
      return
      end
C ***************************************************************************
CH    REVISION HISTORY  :
C     VERSION      BY                 DATE        COMMENT
C       1.1     A. de Waal          06/01/90    Finally Commented
C     FILEND            :
C ***************************************************************************
C
C     FILE              : IDEGRC.FOR
C ***************************************************************************
CN    MODULE NAME       : idegrc
CA    FUNCTION          : Convert Integer Degrees Celcius to Integer Number
CS    CALL SEQUENCE     : ival = idegrc(ivalue)
CI    INPUT PARAMETERS  : ivalue: integer - Integer degree celcius number in
C                                           the range 20 to 80 degrees to be
C                                           converted
CO    OUTPUT PARAMETERS : iaddeg: integer - Converted integer value in the
C                                           range 0 to 4095
CG    GLOBAL VARIABLES  : None
CM    MODULES CALLED    : None
CE    ERROR CONDITIONS  : None
CC    COMMENTS          : Converts integer expression for degrees celcius in
C                         the range 20 to 80 degrees to an integer expression
C                         for temperature in range 0 to 4095
```

```
C ********************************************************************************
      integer*2 function idegrc(ivalue)
      integer*2 ivalue
      idegrc = INT(ANINT(REAL(ivalue)*60.0/4095.0 + 20.0))
      return
      end
C ********************************************************************************
CH    REVISION HISTORY  :
C     VERSION       BY                  DATE        COMMENT
C       1.1     A. de Waal            06/01/90   Finally Commented
C     FILEND            :
C ********************************************************************************
C
C     FILE              : IMXMN.FOR
C ********************************************************************************
CN    MODULE NAME       : imxmn
CA    FUNCTION          : Check and Clip Max/Min Values to DAC
CS    CALL SEQUENCE     : ivalue = imxmn(ivalue)
CI    INPUT PARAMETERS  : ivalue: integer - value to be checked and clipped
CO    OUTPUT PARAMETERS : imxmn : integer - value, clipped if necessary
CG    GLOBAL VARIABLES  : None
CM    MODULES CALLED    : None
CE    ERROR CONDITIONS  : None
CC    COMMENTS          : Checks values to be written to DAC before they are
C                         written and ensures that they lie between the range
C                         0 to 4095 by clipping them if necessary
C ********************************************************************************
      integer*2 function imxmn(ivalue)
      integer*2 ivalue
      if (ivalue.gt.4095) then
          imxmn = 4095
      elseif (ivalue.lt.0) then
          imxmn = 0
      else
          imxmn = ivalue
      endif
      return
      end
C ********************************************************************************
CH    REVISION HISTORY  :
C     VERSION       BY                  DATE        COMMENT
C       1.1     A. de Waal            06/01/90   Finally Commented
C     FILEND            :
C ********************************************************************************

C
C     FILE              : IMXMN.FOR
C ********************************************************************************
CN    MODULE NAME       : imxmn
CA    FUNCTION          : Check and Clip Max/Min Setpoint Values
CS    CALL SEQUENCE     : ivalue = imxmns(ivalue)
CI    INPUT PARAMETERS  : ivalue: integer - value to be checked and clipped
CO    OUTPUT PARAMETERS : imxmn : integer - value, clipped if necessary
CG    GLOBAL VARIABLES  : None
CM    MODULES CALLED    : None
CE    ERROR CONDITIONS  : None
CC    COMMENTS          : Checks setpoint values to be used and ensures that
C                         they lie in the range -4095 to 4095 by clipping if
C                         necessary.
C ********************************************************************************
      integer*2 function imxmns(ivalue)
      integer*2 ivalue
      if (ivalue.gt.4095) then
          imxmns = 4095
      elseif (ivalue.lt.(-4095)) then
          imxmns = -4095
      else
          imxmns = ivalue
      endif
      return
      end
C ********************************************************************************
CH    REVISION HISTORY  :
C     VERSION       BY                  DATE        COMMENT
C       1.1     A. de Waal            06/01/90   Finally Commented
C     FILEND            :
C ********************************************************************************
C
C     FILE              : IMXMNT.FOR
C ********************************************************************************
CN    MODULE NAME       : imxmnt
CA    FUNCTION          : Check and Clip Max/Min Temperature
CS    CALL SEQUENCE     : value = imxmn(value)
CI    INPUT PARAMETERS  : value : real*8 - value to be checked and clipped
CO    OUTPUT PARAMETERS : imxmnt: real*8 - value, clipped if necessary
CG    GLOBAL VARIABLES  : None
CM    MODULES CALLED    : None
CE    ERROR CONDITIONS  : None
CC    COMMENTS          : Checks temperature values and ensures that they lie
C                         between the range 0 to 4095 by clipping them if
C                         necessary
C ********************************************************************************
      real*8 function mxmnt(value)
      real*8 value
      if (value.gt.80.0) then
          mxmnt = 80.0
      elseif (value.lt.20.0) then
          mxmnt = 20.0
      else
          mxmnt = value
      endif
      return
      ehd
C ********************************************************************************
CH    REVISION HISTORY  :
C     VERSION       BY                  DATE        COMMENT
C       1.1     A. de Waal            06/01/90   Finally Commented
C     FILEND            :
C ********************************************************************************

C
C     FILE              : INICON.FOR
C ********************************************************************************
CN    MODULE NAME       : inicon
CA    FUNCTION          : Initialize Stabilizing PI Controller Parameters
CS    CALL SEQUENCE     : call inicon()
```

```
CI    INPUT PARAMETERS  : None
CO    OUTPUT PARAMETERS : None
CG    GLOBAL VARIABLES  : Include files:  convar.inc
CM    MODULES CALLED    : None
CE    ERROR CONDITIONS  : None
CC    COMMENTS          : Initializes arrays used in the stabilizing PI
C                         controller algorithm conalg()
C ***************************************************************************
      subroutine inicon()
$include: 'convar.inc'
      do 900 i = 1, 10
          do 800 j = 1, 10
              gptr(i,j) = 1
              pptr(i,j) = 1
              if (i.eq.j) then
                  inpmap(i,j) = 1
                  outmap(i,j) = 1
                  setmap(i,j) = 1
              else
                  inpmap(i,j) = 0
                  outmap(i,j) = 0
                  setmap(i,j) = 0
              endif
              prop(i,j) = 0.0000001
              intg(i,j) = 10000000
              do 700 k = 1, 100
                  dlyerr(i,j,k) = 0
700           continue
800       continue
900   continue
      conlod = .false.
      return
      end
C ***************************************************************************
CH    REVISION HISTORY  :
C     VERSION       BY                  DATE        COMMENT
C       1.1     A. de Waal            06/01/90   Finally Commented
C     FILEND            :
C ***************************************************************************


C
C     FILE              : INILOG.FOR
C ***************************************************************************
CN    MODULE NAME       : inilog
CA    FUNCTION          : Initialize Input/Output/Setpoint Logged Values
CS    CALL SEQUENCE     : call inilog()
CI    INPUT PARAMETERS  : None
CO    OUTPUT PARAMETERS : None
CG    GLOBAL VARIABLES  : Include files:  logval.inc
C                         Common blocks:  /modfr/ modfr
CM    MODULES CALLED    : None
CE    ERROR CONDITIONS  : None
CC    COMMENTS          : Initializes arrays containing current logged values
C                         of inputs, outputs and setpoints to zero.  Also
C                         initializes log variables and flags.
C ***************************************************************************
      subroutine inilog()
$include: 'logval.inc'
      character*3 autman
      logical*2 modfr
      common /modfr/ modfr
      sampl = 10
      stepid = 1
      logged = .false.
      modfr = .false.
      do 900 i = 1, 10
          do 800 j = 1, 16
              if (j.le.10) then
                  inpval(i,j) = 0
                  setval(i,j) = 0
              else
                  outval(i,j) = 0
              endif
800       continue
900   continue
      return
      end
C ***************************************************************************
CH    REVISION HISTORY  :
C     VERSION       BY                  DATE        COMMENT
C       1.1     A. de Waal            06/01/90   Finally Commented
C     FILEND            :
C ***************************************************************************


C
C     FILE              : INIOVR.FOR
C ***************************************************************************
CN    MODULE NAME       : iniovr
CA    FUNCTION          : Initialize Input/Output/Setpoint Values
CS    CALL SEQUENCE     : call iniovr()
CI    INPUT PARAMETERS  : None
CO    OUTPUT PARAMETERS : None
CG    GLOBAL VARIABLES  : Include files:  iovar.inc,defnam.inc,logval.inc
CM    MODULES CALLED    : None
CE    ERROR CONDITIONS  : None
CC    COMMENTS          : Initializes vectors containing input, output and
C                         setpoint variables for the running of the rig by
C                         reading them from the file herig.sys (can be
C                         changed by user.
C ***************************************************************************
      subroutine iniovr()
$include: 'iovar.inc'
$include: 'defnam.inc'
$include: 'logval.inc'
      integer*2 iofile
      integer logint
      real*4 rinp(10),rout(16),rset(10)
      character*4 adstat
      common /adstat/ adstat
60    format(16I5)
70    format(10I5)
80    format(16F6.1)
90    format(10F6.1)
100   format(I5)
```

```
      iofile = 3

      open(iofile,file='herig.sys')
      read(iofile,90)rinp
      read(iofile,80)rout
      read(iofile,90)rset
      steplv = 0.0
      do 900 i = 1, 16
          if (i.le.10) then
              input(i) = INT(ANINT( 40.95*rinp(i) ))
              if (i.le.6) then
                  setp(i) = INT(ANINT( 40.95*rset(i) ))
              else
                  setp(i) = INT(ANINT(
     +                (4095.0/60.0)*(rset(i)-20.0) ))
              endif
          endif
          if ((i.le.8).or.(i.ge.15)) then
              output(i) = INT(ANINT(
     +                (4095.0/60.0)*(rout(i)-20.0) ))
          else
              output(i) = INT(ANINT( 40.95*rout(i) ))
          endif
900   continue
      read(iofile,100)logint
      hertz = 1/real(logint)
c      write(*,*)' Logging interval (seconds):'
c      write(*,100)logint
      read(iofile,'(a)',iostat=ichk,err=200)outpth
      read(iofile,'(a)',iostat=ichk,err=200)inpath
      read(iofile,'(a)',iostat=ichk,err=200)adstat
199   goto 201
200   write(*,*)'Error Loading from file herig.sys'
201   continue
c      pause
c      write(*,70)input
c      write(*,60)output
c      write(*,70)setp
c      pause
      return
      end
C ***************************************************************************
CH    REVISION HISTORY  :
C     VERSION       BY                DATE        COMMENT
C       1.1     A. de Waal           06/01/90   Finally Commented
C     FILEND            :
C ***************************************************************************
C
C     FILE              : INIT.FOR
C ***************************************************************************
CN    MODULE NAME       : init
CA    FUNCTION          : Call and Include Initializing Proc/Block Data
CS    CALL SEQUENCE     : call init()
CI    INPUT PARAMETERS  : None
CO    OUTPUT PARAMETERS : None
CG    GLOBAL VARIABLES  : Include files:  (Block data files) hebdat.for,
C                                         inrunv.for,scrdat.for,grldat.for,
C                                         pldat.for,keydat.for
C                         Common blocks:  /adstat/ adstat
CM    MODULES CALLED    : Fortran   :  blmap,iniovr,inicon,inilog,setadp,setdap
C                                      wrinp
CE    ERROR CONDITIONS  : None
CC    COMMENTS          : Includes block data files initializing all data to
C                         be used in program and calls initializing procedures
C ***************************************************************************
$include: 'hebdat.for'
$include: 'inrunv.for'
$include: 'scrdat.for'
$include: 'grldat.for'
$include: 'pldat.for'
$include: 'keydat.for'
      subroutine init()
      character*4 adstat
      common /adstat/ adstat

      call blmap()
      call iniovr()
      call inicon()
      call inilog()
      if ((adstat.eq.'runs').or.(adstat.eq.'RUNS')) then
          call setadp()
          call setdap()
          call wrinp()
      else
          write(*,*)' Configured for Testing Without Interface Cards'
          write(*,*)' To change, modify herig.sys file'
          write(*,*)' Replace <test> with <runs>'
          pause
      endif
      return
      end
C ***************************************************************************
CH    REVISION HISTORY  :
C     VERSION       BY                DATE        COMMENT
C       1.1     A. de Waal           06/01/90   Finally Commented
C     FILEND            :
C ***************************************************************************

C
C     FILE              : INRUNV.FOR
C ***************************************************************************
CN    MODULE NAME       : inrunv
CA    FUNCTION          : Block Data Initializing Run and LQG Variables
CS    CALL SEQUENCE     : Not a subroutine
CI    INPUT PARAMETERS  : n/a
CO    OUTPUT PARAMETERS : n/a
CG    GLOBAL VARIABLES  : Include files:  runvar.inc,syst.inc,defnam.inc,
C                                         sysnms.inc,time.inc
CM    MODULES CALLED    : n/a
CE    ERROR CONDITIONS  : n/a
CC    COMMENTS          : Initializes variables and flags for running of
C                         program, as well as variables for the running the
C                         rig under LQG control
C ***************************************************************************
      block data inrunv
$include: 'runvar.inc'
```

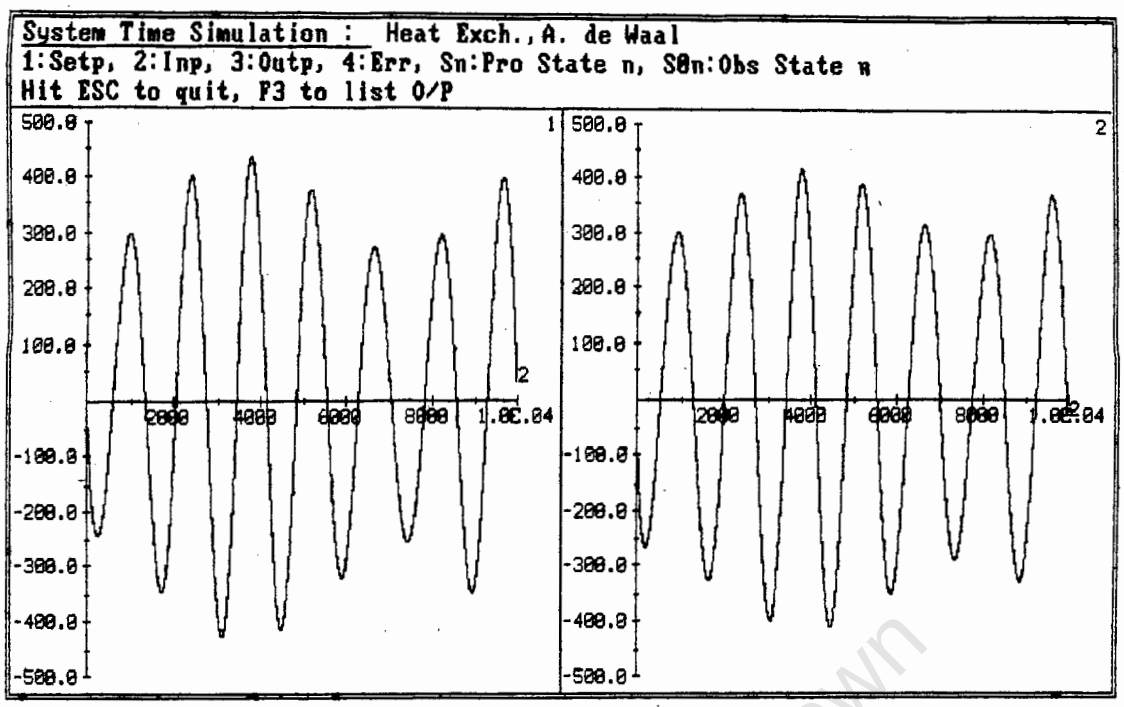


Figure R.11: System 3 - Simulated Response of Inputs to Low Frequency Disturbances

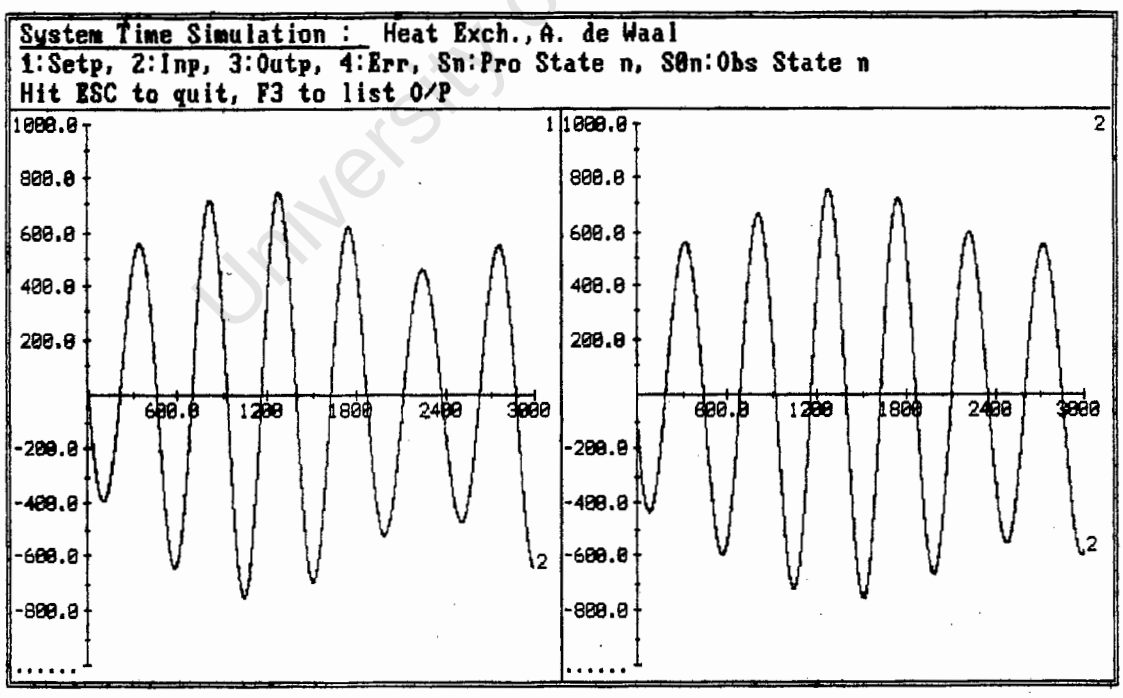


Figure R.12: System 3 - Simulated Response of Inputs to Interm. Frequency Disturbances

```
$include: 'syst.inc'
$include: 'defnam.inc'
$include: 'sysnms.inc'
$include: 'time.inc'
      data mode/1/
      data chscr/.false./,log/.false./,quit/.false./
      data key/0/
      data precom/1/
      data (syst(i),i=1,5)
     + / .0000E+00, 2.6198E-02, 2.6198E-04, 2.0000E+00, 4.0000E+00/
      data ((a(j,i),i=1,4),j=1,4)
     +/-1.1648E-03,   1.7342E-05,    .0000E+00,    .0000E+00,
     +   .0000E+00,  -1.0772E-03,    .0000E+00,    .0000E+00,
     +   .0000E+00,    .0000E+00,  -2.6330E-03,    .0000E+00,
     +   .0000E+00,    .0000E+00,   5.0653E-04,  -1.9612E-03/
      data ((b(j,i),i=1,2),j=1,4)
     +/-2.3063E-04,   3.2102E-04,
     +   .0000E+00,   1.6213E-03,
     +  2.1092E-03,    .0000E+00,
     + -1.5903E-03,   1.4787E-03/
      data ((c(j,i),i=1,4),j=1,2)
     +/ 1.0, 0.0, 0.0, 0.0,
     +  0.0, 0.0, 0.0, 1.0/
      data ((kci(j,i),i=1,4),j=1,2)
     +/ 1.0, 0.0, 0.0, 0.0,
     +  0.0, 0.0, 0.0, 1.0/
      data ((kfi(j,i),i=1,2),j=1,4)
     +/ 1.0, 0.0,
     +  0.0, 0.0,
     +  0.0, 0.0,
     +  0.0, 1.0/
      data ((ql(j,i),i=1,4),j=1,4)
     +/ 1.0, 0.0, 0.0, 0.0,
     +  0.0, 1.0, 0.0, 0.0,
     +  0.0, 0.0, 1.0, 0.0,
     +  0.0, 0.0, 0.0, 1.0/
      data ((rl(j,i),i=1,2),j=1,2)
     +/ 1.0, 0.0,
     +  0.0, 1.0/
      data ((kc(j,i),i=1,4),j=1,2)
     +/ 1.0, 0.0, 0.0, 0.0,
     +  0.0, 0.0, 0.0, 1.0/
      data ((q(j,i),i=1,4),j=1,4)
     +/ 1.0, 0.0, 0.0, 0.0,
     +  0.0, 1.0, 0.0, 0.0,
     +  0.0, 0.0, 1.0, 0.0,
     +  0.0, 0.0, 0.0, 1.0/
      data ((r(j,i),i=1,2),j=1,2)
     +/ 1.0, 0.0,
     +  0.0, 1.0/
      data ((kf(j,i),i=1,2),j=1,4)
     +/ 0.0, 0.0,
     +  0.0, 0.0,
     +  0.0, 0.0,
     +  0.0, 0.0/
      data ((ptrsst(j,i),i=1,2),j=1,2) /1.0,0.0,0.0,1.0/
      data inpath /'c:\progs\rig\data\       '/
      data outpth /'c:\progs\rig\data\       '/
      data defdir /'      '/
      data infnam /'                        '/
      data fname /'                         '/
      data mname /'                         '/
      data pernam /'                        '/
      data usenam /'                                                  '/
      data (inpnms(i),i=1,10)
     +            /'Input 1   ','Input 2   ','Input 3   ','Input 4   ',
     +             'Input 5   ','Input 6   ','Input 7   ','Input 8   ',
     +             'Input 9   ','Input 10  '/
      data (outnms(i),i=1,10)
     +            /'Output 1  ','Output 2  ','Output 3  ','Output 4  ',
     +             'Output 5  ','Output 6  ','Output 7  ','Output 8  ',
     +             'Output 9  ','Output 10 '/
      data prjnm /'Heat Exch.'/
      data engnms /'A. de Waal'/
      data (xn(i),i=1,15) /15*0.0/
      data (un(i),i=1,15) /15*0.0/
      data (rn(i),i=1,15) /15*0.0/
      data (vn(i),i=1,15) /15*0.0/
      data (yn(i),i=1,15) /15*0.0/
      data (ynact(i),i=1,15) /15*0.0/
      data (zn(i),i=1,15) /15*0.0/
      data (nn(i),i=1,15) /15*0.0/
      data (xnobs(i),i=1,15) /15*0.0/
      data (ynobs(i),i=1,15) /15*0.0/
      data ti/0.0/,dt/1.0/,tend/3000.0/
      data (c1(i),i=1,4) /1,2,2,1/
      data (c2(i),i=1,3) /0.5,0.5,1.0/
      data obsinc/1/,coninc/1/
      data setplt/1/,inpplt/1/,outplt/1/,ostplt/1/,pstplt/1/
      data (stpinp(i),i=1,5) /5*0/
      data (stpdat(i),i=1,5) /5*0.0/
      data (stpinc(i),i=1,5) /5*0.0/
      data twmax/711/,twstrt/4/,thmax/270/,thstrt/316/,twidth/1/
     +     theigh/1/,txdel/711/,tydel/200/
      data (xtlim(i),i=1,3) /3000.0,0.0,0.0/
      data ((ytlim(i,j),i=1,3),j=1,10) /4095.0,0.0,0.0,
     +                                  4095.0,0.0,0.0,
     +                                  4095.0,0.0,0.0,
     +                                  4095.0,0.0,0.0,
     +                                  4095.0,0.0,0.0,
     +                                  4095.0,0.0,0.0,
     +                                  4095.0,0.0,0.0,
     +                                  4095.0,0.0,0.0,
     +                                  4095.0,0.0,0.0/
      data (tgraph(i),i=1,4) /10,172,340,110/
      data ninc/1/,dinc/1/,rinc/1/,ntyp/0/,dtyp/0/,rtyp/0/
      data nchset/15*1/,dchset/15*1/,rchset/15*1/
      data irand/10/,nampl/1.0/,dampl/1.0/,rampl/1.0/
      data nfre/0.0/,dfre/0.0/,rfre/0.0/
      data nchdat/30*10.0/,dchdat/30*10.0/,rchdat/30*10.0/
      end
C ***************************************************************************
CH    REVISION HISTORY  :
C     VERSION        BY                 DATE         COMMENT
C       1.1      A. de Waal           06/01/90    Finally Commented
```

```
C     FILEND            :
C ***************************************************************************
C
C     FILE              : IPERC.FOR
C ***************************************************************************
CN    MODULE NAME       : iperc
CA    FUNCTION          : Convert Integer Number to Integer Percentage
CS    CALL SEQUENCE     : ipercent = iperc(ivalue)
CI    INPUT PARAMETERS  : ivalue : integer - Integer value in the range
C                                            0 to 4095 to be converted
CO    OUTPUT PARAMETERS : iperc  : integer - Converted integer percentage in
C                                            the range 0 to 100 percent
CG    GLOBAL VARIABLES  : None
CM    MODULES CALLED    : None
CE    ERROR CONDITIONS  : None
CC    COMMENTS          : Converts integer number in range 0 to 4095 to an int
C                         expression of percentage in range 0 to 100 percent
C ***************************************************************************
      integer*2 function iperc(ivalue)
      integer*2 ivalue
      iperc = INT(ANINT(REAL(ivalue)/40.95))
      return
      end
C ***************************************************************************
CH    REVISION HISTORY  :
C     VERSION       BY                  DATE        COMMENT
C       1.1      A. de Waal           06/01/90   Finally Commented
C     FILEND            :
C ***************************************************************************
C
C     FILE              : ITMXMN.FOR
C ***************************************************************************
CN    MODULE NAME       : itmxmn
CA    FUNCTION          : Check and Clip Max/Min Values to DAC
CS    CALL SEQUENCE     : ivalue = itmxmn(ivalue)
CI    INPUT PARAMETERS  : ivalue: integer - value to be checked and clipped
CO    OUTPUT PARAMETERS : itmxmn : integer - value, clipped if necessary
CG    GLOBAL VARIABLES  : None
CM    MODULES CALLED    : None
CE    ERROR CONDITIONS  : None
CC    COMMENTS          : Checks values to be written to DAC before they are
C                         written and ensures that they lie between the range
C                         0 to 4095 by clipping them if necessary
C ***************************************************************************
      integer*2 function itmxmn(ivalue)
      integer*2 ivalue,itmxmn
      if (ivalue.gt.4095) then
          itmxmn = 4095
      elseif (ivalue.lt.0) then
          itmxmn = 0
      else
          itmxmn = ivalue
      endif
      return
      end
C ***************************************************************************
CH    REVISION HISTORY  :
C     VERSION       BY                  DATE        COMMENT
C       1.1      A. de Waal           06/01/90   Finally Commented
C     FILEND            :
C ***************************************************************************
C
C     FILE              : KEYERR.FOR
C ***************************************************************************
CN    MODULE NAME       : keyerr.for
CA    FUNCTION          : Indicate Incorrect Key Pressed
CS    CALL SEQUENCE     : call keyerr()
CI    INPUT PARAMETERS  : None
CO    OUTPUT PARAMETERS : None
CG    GLOBAL VARIABLES  : None
CM    MODULES CALLED    : Assembler:  ERTONE
CE    ERROR CONDITIONS  : None
CC    COMMENTS          : Beeps to indicate that incorrect key pressed
C ***************************************************************************
      subroutine keyerr()
      call ERTONE()
      return
      end
C ***************************************************************************
CH    REVISION HISTORY  :
C     VERSION       BY                  DATE        COMMENT
C       1.1      A. de Waal           06/01/90   Finally Commented
C     FILEND            :
C ***************************************************************************
C
C     FILE              : LOGVAL.FOR
C ***************************************************************************
CN    MODULE NAME       : logval
CA    FUNCTION          : Log Input/Output/Setpoint Values
CS    CALL SEQUENCE     : call logval()
CI    INPUT PARAMETERS  : None
CO    OUTPUT PARAMETERS : None
CG    GLOBAL VARIABLES  : Include files:  logval.inc,iovar.inc,runvar.inc,
C                                         time.inc
C                         Common blocks:  /ldsvmd/ autman
CM    MODULES CALLED    : None
CE    ERROR CONDITIONS  : None
CC    COMMENTS          : Logs values of process inputs, outputs and setpoints
C                         at time intervals equal to the specified logging
C                         interval.
C ***************************************************************************
      subroutine logval()
$include: 'logval.inc'
$include: 'iovar.inc'
$include: 'runvar.inc'
$include: 'time.inc'
      character*3 autman
      integer*2 i
      real tlog/0.0/
      common /ldsvmd/ autman
      tlog = tlog + dt
      if (tlog.ge.1/hertz) then
          tlog = 0.0
          if (sampl.eq.0) then
              stepid = -1
          endif
```

```
            sampl = sampl + 1
            if (mode.ge.5) autman = 'aut'
            if (sampl.gt.1800) then
                log = .false.
            endif
            do 900 i = 1,16
                outval(sampl,i) = output(i)
                if (i.le.10) then
                    inpval(sampl,i) = input(i)
                    setval(sampl,i) = setp(i)
                endif
900         continue
        endif
        return
        end
C ********************************************************************************
CH      REVISION HISTORY  :
C       VERSION        BY                    DATE         COMMENT
C         1.1      A. de Waal              06/01/90    Finally Commented
C       FILEND            :
C ********************************************************************************
C
C       FILE              : LQGRUN.FOR
C ********************************************************************************
CN      MODULE NAME       : lqgrun
CA      FUNCTION          : Runs Rig for Modes of LQG control
CS      CALL SEQUENCE     : call lqgrun()
CI      INPUT PARAMETERS  : None
CO      OUTPUT PARAMETERS : None
CG      GLOBAL VARIABLES  : Include files:  heblock.inc,screan.inc,iovar.inc,
C                                           time.inc,syst.inc,keys.inc
C                           Common blocks:  /runmod/ mode
C                                           /runflg/ chscr,log,quit
C                                           /adstat/ adstat
CM      MODULES CALLED    : Fortran  :  inisim,togopt,getr4,simgrf,wrhead,
C                                       parvvn,noise,dist,prosta,pltsta,obssta
C                                       runini,readop,conalg,wrinp,logval,runkey,
C                                       typscr,updscr,prtr4
C                           Assembler:  WIPSCR,WRTSTR,GETTIM,CLRSCR
CE      ERROR CONDITIONS  : None
CC      COMMENTS          : This routine starts off by running in the automatic
C                           mode as for herun (mode 5, settle = .true.).
C                           When the esc key is pressed, the mode switches to the
C                           mode running the rig under LQG control (mode = 5,
C                           settle = .false.) of temperatures T1 and T5 using
C                           flow setpoints F1 and F2 as inputs to this modified
C                           system.  Under this mode, the traces of the
C                           temperature setpoints and outputs for T1 and T5
C                           as well as flow setpoints (used as inputs) F1 and F2
C                           are plotted on the screen.  Noise, disturbance and
C                           setpoint perturbations are also injected as specified
C                           by user in the simopt procedure.  Rig inputs, outputs
C                           and setpoints can, as usual, be logged.  When the esc
C                           key is pressed again, logging is terminated and
C                           LQG control is suspended (stabilising control only
C                           again).
C ********************************************************************************
        subroutine lqgrun()
        implicit integer*2 (I,g)
$include: 'heblock.inc'
$include: 'screen.inc'
$include: 'iovar.inc'
$include: 'time.inc'
$include: 'syst.inc'
$include: 'keys.inc'

        integer*2 key,flush,order,dima,ferr
        integer*2 setp5s,setp6s,outp1s,outp5s
        integer finc,prevo1,prevo5
        real wfilt,tfilt,afilt,bfilt
        real er(15),plt(15)
        real vvn(15)/15*0.0/
        real*4 ti2,xtrid
        real*8 scale(2), centre(4)
        character*1 trid
        logical settle
        character*4 adstat

        real*4 ctime/0.0/,difft/0.0/
        integer*2 runcyc,timewt,ihr,imin,isec,i100th
        integer*2 mode,page/0/
        logical*2 chscr,log,quit
        common /runmod/ mode
        common /runflg/ chscr,log,quit
        common /adstat/ adstat
        assign 900 to runcyc
        assign 800 to timewt

        order = int(syst(4))
        dima = int(syst(5))
        settle = .true.
        pstplt = 1
        obsinc = coninc
        mode = 5
        chscr = .true.

        wfilt = 0.2
        tfilt = 1/wfilt
        afilt = exp(-dt/tfilt)
        bfilt = 1.0 - afilt

        call inisim()
        call WIPSCR(0)

55      continue
        plnoi = 1
        call WRTSTR(0,50,100,27,'Plot Perturbation Traces? :')
        call togopt(plnoi,50+28*9,100,ikey)

        precom = 1
        call WRTSTR(0,50,120,62,
       +'Include Precompensator for Zero Steady-State Error? (Y/N):    ')
        call togopt(precom,50+60*9,120,key)

        finc = 1
        call WRTSTR(0,50,140,40,
```

```
     +   'Include Lowpass Filter on Outputs (Y/N):')
         call togopt(finc,50+41*9,140,key)

         if (finc.eq.0) then
             call WRTSTR(0,50,160,31,
     +   'Filter 3dB Cut-Off Frequency = ')
             key = getr4(0,50+32*9,160,7,'(g10.4)',10,wfilt)
         endif

         call WRTSTR(0,50,180,40,
     +   'Hit RETURN to Start or ESC to re-specify')
         if (key.eq.esck) goto 55

         call simgrf()
         call wrhead(1,10,(thstrt-thmax-32),25,
     +              'Heat Exhang. Responses : ')
         call WRTSTR(1,10,(thstrt-thmax-4),31,
     +          'Hit ESC to start.                ')
         ti2 = -1.0

         call runini()

 900     continue

C Printing out the sample time
         xpos = optxin + 2*chw + 64*chw + 7*chw
         ypos = optyin + 4*chh/3
         call prtr4(page,xpos,ypos,6,'(F4.2)',4,difft)

C Getting the time at the beginning of the loop
         call GETTIM(ihr,imin,isec,i100th)
         ctime = real(isec) + (real(i100th))/100.0

C Performing the reading of rig outputs, the implementation of the
C control algorithm, the writing of the rig inputs and the logging of
C inputs, outputs and setpoints.
         if ((adstat.eq.'runs').or.(adstat.eq.'RUNS')) call readop()
         if ((mode.ge.5).and.(.not.quit)) call conalg()
         if ((adstat.eq.'runs').or.(adstat.eq.'RUNS')) call wrinp()
         if (log) call logval()

         if (.not.settle) then
             call prtr4(1,75,(thstrt-thmax-18),7,'(g10.4)',10,ti)

             rn(1) = real(setp(7) - outp1s)
             rn(2) = real(setp(8) - outp5s)

             un(1) = real(setp(5) - setp5s)
             un(2) = real(setp(6) - setp6s)

             if (coninc.eq.0) then
                 do 701 j = 1,5
                     if ((ti.ge.stpdat(j)).and.(ti2.lt.stpdat(j))) then
                         if ((stpinp(j).gt.0).and.
     +                       (stpinp(j).le.order)) then
                             rn(stpinp(j)) = rn(stpinp(j)) + stpinc(j)
                             if (stpinp(j).eq.1) then
                                 setp(7) = int(rn(1)) + outp1s
                             elseif (stpinp(j).eq.2) then
                                 setp(8) = int(rn(2)) + outp5s
                             endif
                         endif
                     endif
 701             continue
             else
                 do 702 j = 1,5
                     if ((ti.ge.stpdat(j)).and.(ti2.lt.stpdat(j))) then
                         if ((stpinp(j).gt.0).and.
     +                       (stpinp(j).le.order)) then
                             un(stpinp(j)) = un(stpinp(j)) + stpinc(j)
                             if (stpinp(j).eq.1) then
                                 setp(5) = int(un(1)) + setp5s
                             elseif (stpinp(j).eq.2) then
                                 setp(6) = int(un(2)) + setp6s
                             endif
                         endif
                     endif
 702             continue
             endif

             if (rinc.eq.0) then
                 call pervvn(vvn)
                 setp(7) = outp1s + vvn(1)
                 setp(8) = outp5s + vvn(2)
                 rn(1) = vvn(1)
                 rn(2) = vvn(2)
             endif

             if (precom.eq.0) then
                 do 111 j = 1, order
                     vn(j) = 0.0
                     do 112 k = 1, order
                         vn(j) = vn(j) + ptrsst(j,k)*rn(k)
 112                 continue
 111             continue
             else
                 do 115 j = 1, order
                     vn(j) = rn(j)
 115             continue
             endif

CCC (Only in "test" mode)
C Calculate process outputs output(i), and measured outputs, yn(i)
C In simulator, equivalent to call prosta() (do this in "test" mode)

             if ((adstat.ne.'runs').and.(adstat.ne.'RUNS')) then
                 if (ninc.eq.0) call noise()
                 if (dinc.eq.0) call dist()
                 call prosta()
                 output(1) = yn(1) + outp1s
                 output(5) = yn(2) + outp5s
                 if (ninc.eq.0) then
                     output(1) = output(1) - int(nn(1))
                     output(5) = output(5) - int(nn(2))
                 endif
                 if (dinc.eq.0) then
```

```
                output(1) = output(1) - int(zn(1))
                output(5) = output(5) - int(zn(2))
            endif
        endif

CCC (End Only in "test" mode)

        if (dinc.eq.0) then
            call dist()
            output(1) = output(1) + int(zn(1))
            output(5) = output(5) + int(zn(2))
            if (plnoi.eq.0)
     +          call pltsta(order,order,ti,zn)
        endif

C Lowpass filtering outputs to get rid of High Frequency Noise
C HOROWITZ & HILL : THE ART OF ELECRONICS, PAGE447
        if (finc.eq.0) then
            output(1) = afilt*prevo1 + bfilt*output(1)
            output(5) = afilt*prevo5 + bfilt*output(5)
            prevo1 = output(1)
            prevo5 = output(5)
        endif

        yn(1) = real(output(1) - outp1s)
        yn(2) = real(output(5) - outp5s)

        if (ninc.eq.0) then
            call noise()
            do 332 i = 1, order
                yn(i) = yn(i) + nn(i)
332         continue
            if (plnoi.eq.0)
     +          call pltsta(order,order,ti,nn)
        endif

C If observer to be included (always the case if coninc = 0 for LQG control),
C calculate observed states, xnobs and obserer outputs, ynobs

        if (obsinc.eq.0) call obssta()

C If controller to be included, calculate new input, un
        if (coninc.eq.0) call concal()

C Inputs to Rig Correspond to Flow Setpoints (Setpoints 5 and 6)
        setp(5) = int(un(1)) + setp5s
        setp(6) = int(un(2)) + setp6s

C Setpoints to Rig Correspond to Temperature Setpoints (Setpoints 7 and 8)
        if (setplt.eq.0) then
            plt(1) = real(setp(7))
            plt(2) = real(setp(8))
            call pltsim(order,ti,plt)
        endif

C Inputs to Rig correspond to Flow Setpoints (Setpoints 5 and 6)
        if (inpplt.eq.0) then
            plt(1) = real(setp(5))
            plt(2) = real(setp(6))
            call pltsim(order,ti,plt)
        endif

C Outputs from Rig correspond to Temperature Outputs (Outputs 1 and 5)
        if (outplt.eq.0) then
            plt(1) = real(output(1))
            plt(2) = real(output(5))
            call pltsim(order,ti,plt)
c            call pltsim(order,ti,ynact)
        endif

C Plot Observer States if Required
        if (ostplt.eq.0) call pltsta(order,dima,ti,xnobs)

C Calculating Error between Outputs (1 and 5) and Setpoints (7 and 8)
        er(1) = real(output(1) - setp(7))
        er(2) = real(output(5) - setp(8))
        if ((rinc.eq.0).and.(plnoi.eq.0))
     +      call pltsim(order,ti,er)

        ti2 = ti
        ti = ti + dt
      endif

800   continue

C Checking the keyboard for instructions to:
C       - change the mode of operation
C       - change inputs or setpoints
C       - step inputs or setpoints
      call runkey()
      if (mode.le.4) settle = .true.

C Change the run screen headings if necessary (eg at a change of operating
C                                                 mode)
      if (chscr) then
          call typscr()
          ypos = y0 - resh - 0.3*chh + 17
          if (settle) then
              call WRTSTR(0,(720-47*9)/2,ypos,47,
     +                  '                                              ')
              call LEVEL(0)
              call WRTSTR(0,(720-47*9)/2,ypos,47,
     +                  'Hit Esc to Switch from Settling to LQG Control')
              call LEVEL(1)
          else
              call WRTSTR(0,(720-47*9)/2,ypos,47,
     +                  '                                              ')
              call LEVEL(0)
              call WRTSTR(0,(720-47*9)/2,ypos,47,
     +                  'Hit Esc key to Quit Running under LQG Control ')
              call LEVEL(1)
          endif
      endif

C Update ,on the run screen, the values of
```

```
C         - inputs
C         - setpoints
C         - outputs
      call updscr()

C Getting the time at the end of the loop and calculating the time elapsed
C since the beginning of the loop
      call GETTIM(ihr,imin,isec,i100th)
      difft = real(isec) + (real(i100th)/100.0)
      if (difft.lt.ctime) difft = difft + 60.0
      difft = abs(difft - ctime)

C If the elapsed time is less than the desired sample time (1 sec), scan the
C keyboard and update the screen until the desired time has elapsed
      if (difft.le.dt) goto timewt

C Restart the cycle if not quit mode
      if (.not.quit) goto runcyc

C Restart the cycle under LQG control if quitting from settling mode
      if (settle) then
          quit = .false.
          settle = .false.
          mode = 5
          chscr = .true.
          setp5s = setp(5)
          setp6s = setp(6)
          outp1s = output(1)
          outp5s = output(5)
          setp(7) = outp1s
          setp(8) = outp5s
          prevo1 = outp1s
          prevo5 = outp5s
          goto runcyc
      endif

      xtrid = ti
      do 666 igrph = 1, order
          do 555 j = 1, 4
              if (j.le.2) scale(j) = tscale(j,igrph)
              centre(j) = tcentr(j,igrph)
555       continue
          trid = '1'
          if (setplt.eq.0) then
              if (igrph.eq.1) then
                  call drwlet(1,xtrid,real(setp(7)),centre,scale,trid)
              else
                  call drwlet(1,xtrid,real(setp(8)),centre,scale,trid)
              endif
          endif
          trid = '2'
          if (inpplt.eq.0) then
              if (igrph.eq.1) then
                  call drwlet(1,xtrid,real(setp(5)),centre,scale,trid)
              else
                  call drwlet(1,xtrid,real(setp(6)),centre,scale,trid)
              endif
          endif
          trid = '3'
          if (outplt.eq.0) then
              if (igrph.eq.1) then
                  call drwlet(1,xtrid,real(output(1)),centre,scale,trid)
              else
                  call drwlet(1,xtrid,real(output(5)),centre,scale,trid)
              endif
          endif
          trid = '4'
          if ((rinc.eq.0).and.(plnoi.eq.0))
     +        call drwlet(1,xtrid,er(igrph),centre,scale,trid)
          if (ostplt.eq.0) then
              do 333 k = 1, dima
                  trid = 'S'
                  call drwlet(1,xtrid,xnobs(k),centre,scale,trid)
                  trid = 'O'
                  call drwlet(1,(xtrid+7/sngl(scale(1))),xnobs(k),
     +                        centre,scale,trid)
                  trid = char(48+k)
                  call drwlet(1,real(xtrid+14/sngl(scale(1))),xnobs(k),
     +                        centre,scale,trid)
333           continue
          endif
666   continue

      call WRTSTR(1,10,(thstrt-thmax-18),44,
     +   '1:Setp, 2:Inp, 3:Outp, 4:Err, Sn:Pro State n')
      call WRTSTR(1,10,(thstrt-thmax-4),29,
     +         'Hit ESC to quit, F3 for Block')
710   continue
      call DISP(1)
      key = INKEY(1)
      if (key.ne.esck) goto 710
      call GPAGE(0)
      call CLRSCR()
      call GPAGE(1)
      call CLRSCR()
      call inmenu()
      return
      end
C ***************************************************************************
CH    REVISION HISTORY  :
C     VERSION       BY                  DATE        COMMENT
C       1.0     A. de Waal           17-11-89    Commented
C       1.1     A. de Waal           06/01/90   Finally Commented
C     FILEND            :
C ***************************************************************************
C
C     FILE              : LQGTYP.FOR
C ***************************************************************************
CN    MODULE NAME       : lqgtyp
CA    FUNCTION          : Drive Menu for Options Running under LQG Control
CS    CALL SEQUENCE     : call lqgtyp()
CI    INPUT PARAMETERS  : None
CO    OUTPUT PARAMETERS : None
CG    GLOBAL VARIABLES  : None
CM    MODULES CALLED    : Fortran  :  inmenu,wrhead,simopt,simscl,nindin,lqgrun,
C                                     hepar
```

```
C                       Assembler:  DOMENU,WRTSTR,ERTONE
CE    ERROR CONDITIONS  : None
CC    COMMENTS          : Initializes menu screen and drives menu for options
C                         for running of rig under LQG control. (similar to
C                         subroutine simul in for OPTCAD package)
C ************************************************************************************
      subroutine lqgtyp()
      implicit integer*2 (D)
      integer*2 opt122
99    continue
      call wrhead(0,int(720-51*9)/2,35,51,
     +   'Running Rig under Frequency Domain and LQG Control.')
      opt122 = DOMENU(122)
      if (opt122.eq.1) then
          call simopt()
      elseif (opt122.eq.2) then
          call simscl()
      elseif (opt122.eq.3) then
          call nindin()
      elseif (opt122.eq.4) then
c         call dosim()
          call lqgrun()
      elseif (opt122.eq.5) then
c          call data()
          call hepar()
      endif
      if (opt122.ne.0) goto 99
      call WIPSCR(0)
      return
      end
C ************************************************************************************
CH    REVISION HISTORY  :
C     VERSION       BY                  DATE        COMMENT
C       1.1     A. de Waal            06/01/90   Finally Commented
C     FILEND            :
C ************************************************************************************


C
C     FILE              : MAXMIN.FOR
C ************************************************************************************
CN    MODULE NAME       : maxmin
CA    FUNCTION          : Check and Clip Max/Min Values to DAC
CS    CALL SEQUENCE     : call maxmin(oldval,newval)
CI    INPUT PARAMETERS  : oldval: integer - value to be checked and clipped
CO    OUTPUT PARAMETERS : newval: integer - value, clipped if necessary
CG    GLOBAL VARIABLES  : None
CM    MODULES CALLED    : None
CE    ERROR CONDITIONS  : None
CC    COMMENTS          : Checks values to be written to DAC before they are
C                         written and ensures that they lie between the range
C                         0 to 4095 by clipping them if necessary
C ************************************************************************************
      subroutine maxmin(oldval,newval)
      integer*2 oldval,newval
      if (oldval.gt.4095) then
          newval = 4095
      elseif (oldval.lt.0) then
          newval = 0
      else
          nowval = oldval
      endif
      return
      end
C ************************************************************************************
CH    REVISION HISTORY  :
C     VERSION       BY                  DATE        COMMENT
C       1.1     A. de Waal            06/01/90   Finally Commented
C     FILEND            :
C ************************************************************************************
C
C     FILE              : MLMVI2.FOR
C ************************************************************************************
CN    MODULE NAME       : mlmvi2
CA    FUNCTION          : Multiply Integer Matrix and Vector
CS    CALL SEQUENCE     : call mlmvi2(matrix,rows,cols,vector,result)
CI    INPUT PARAMETERS  : matrix: integer - matrix of dimension (rows,cols)
C                         rows,cols: integer - dimensions of matrix
C                         vector: integer - vector of dimension (cols)
CO    OUTPUT PARAMETERS : result: integer - result of multiplication of
C                                           matrix by vector
CG    GLOBAL VARIABLES  : None
CM    MODULES CALLED    : None
CE    ERROR CONDITIONS  : None
CC    COMMENTS          : Multiplies integer matrix of dimension (rows,cols)
C                         by vector of dimension (rows) and returns the
C                         resultant vection of dimension (rows) to calling
C                         procedure.
C ************************************************************************************
      subroutine mlmvi2(matrix,rows,cols,vector,result)
      integer*2 matrix,vector,result
      integer*2 rows,cols,row,col
      dimension matrix(rows,cols),vector(cols),result(rows)
      do 900 row = 1, rows
          result(row) = 0
          do 800 col = 1, cols
              result(row) = result(row) + matrix(row,col)*vector(col)
800       continue
900   continue
      return
      end
C ************************************************************************************
CH    REVISION HISTORY  :
C     VERSION       BY                  DATE        COMMENT
C       1.1     A. de Waal            06/01/90   Finally Commented
C     FILEND            :
C ************************************************************************************
C
C     FILE              : MLMVR8.FOR
C ************************************************************************************
CN    MODULE NAME       : mlmvr8
CA    FUNCTION          : Multiply Real*8 Matrix and Vector
CS    CALL SEQUENCE     : call mlmvr8(matrix,rows,cols,vector,result)
CI    INPUT PARAMETERS  : matrix: real*8 - matrix of dimension (rows,cols)
C                         rows,cols: integer - dimensions of matrix
C                         vector: real*8 - vector of dimension (cols)
```

```
     +             '[°C]')
      grfdim(1) = 5+58
      grfdim(2) = 288
      grfdim(3) = 40
      grfdim(4) = 250
      yp(1) = DBLE(rdegrc(scalop(2,1,3)))
      yp(2) = DBLE(rdegrc(scalop(2,2,3)))
      yp(3) = DBLE(rdegrc(scalop(2,2,3)))
      call dwaxes(page,nums,xscale,xp,yp,grfdim,scale,centre)

      call WRTSTR(page,10,280,12,'            ')

      grfdim(1) = 15+43+43
      grfdim(2) = 288
      grfdim(3) = 500
      grfdim(4) = 250
      call WRTSTR(page,(500+45+45-9*9),295,9,'[seconds]')
      call WRTSTR(page,15+43+43,35,7,
     +             '[units]')

      xp(1) = DBLE(scalop(1,1,3))
C     xp(1) = DBLE(INT2(scalopt(1,1,3)/5))*5.0
C     if ( (MOD( ((sampl-1)/hertz),5) ).gt.0 ) xp(1) = xp(1) + 5.0
      xp(2) = DBLE(scalop(1,2,3))
      xp(3) = DBLE(scalop(1,2,3))
      yp(1) = DBLE(scalop(2,1,3))
      yp(2) = DBLE(scalop(2,2,3))
      yp(3) = DBLE(scalop(2,2,3))
      sttime = INT2(xp(2)*hertz)
      entime = INT2(xp(1)*hertz)

      call dwaxes(page,nums,xscale,xp,yp,grfdim,scale,centre)

      trace = 0

C Do for inputs, outputs and setpoints
      do 900 k = 1, 3

C Do for numbers 1 to 16
         do 800 j = 1, 16

C Only plot this data if flag set
            if (plotop(k,j,3).eq.1) then
               trace = trace + 1
               call numstr(1,4,'(I1)',trace,0,0,0,1,trid)
               call PRTI2(page,590,(45+trace*12),4,'(I2)',2,trace)
               call WRTSTR(page,(590+2*9),(45+trace*12),2,': ')
               call WRTSTR(page,(590+4*9),(45+trace*12),10,
     +                     insnam(k,j))

C If plotting an input
               if (k.eq.1) then
                  do 700 i = 1, 1800
                     inper(i) = real(inpval(i,j))
700               continue
                  do 650 i = sttime, (entime-1)
                     call plotpt(page,xscale,time(i+1),
     +                     inper(i+1),centre,scale)
650               continue
                  call drwlet(page,time(entime),inper(entime),
     +                     centre,scale,trid)

C Else if plotting an output
               elseif (k.eq.2) then
                  do 600 i = 1, 1800
                     if ((j.le.8).or.(j.ge.15)) then
                        outper(i) = real(outval(i,j))
                     else
                        outper(i) = real(outval(i,j))
                     endif
600               continue
                  do 550 i = sttime, (entime-1)
                     call plotpt(page,xscale,time(i+1),
     +                     outper(i+1),centre,scale)
550               continue
                  call drwlet(page,time(entime),outper(entime),
     +                     centre,scale,trid)

C Else plotting for setpoints
               else
                  do 500 i = 1, 1800
                     if (j.ge.6) then
                        setper(i) = real(setval(i,j))
                     else
                        setper(i) = real(setval(i,j))
                     endif
500               continue
                  do 450 i = sttime, (entime-1)
                     call plotpt(page,xscale,time(i+1),
     +                     setper(i+1),centre,scale)
450               continue
                  call drwlet(page,time(entime),setper(entime),
     +                     centre,scale,trid)
               endif
            endif
800      continue
900   continue

      call MOVE(590,(45+trace*12)+12)
      call DLINE(590,22)
      call MOVE(590,(45+trace*12)+12)
      call DLINE(715,(45+trace*12)+12)

      key = INKEY(1)
      return
      end
C ***************************************************************************
CH    REVISION HISTORY  :
C     VERSION       BY                DATE        COMMENT
C       1.1     A. de Waal          06/01/90   Finally Commented
C     FILEND          :
C ***************************************************************************
C
C     FILE            : PERVVN.FOR
C ***************************************************************************
CN    MODULE NAME     : pervvn
```

```
CA     FUNCTION           : Perturb Rig Setpoint Gaussianly of Sinuloidally
CS     CALL SEQUENCE      : call perset(vvn)
CI     INPUT PARAMETERS   : None
CO     OUTPUT PARAMETERS  : vvn: real - returned setpoint perturbing vector
CG     GLOBAL VARIABLES   : Include files:  time.inc,syst.inc
CM     MODULES CALLED     : Fortran  : rndnrm
CE     ERROR CONDITIONS   : None
CC     COMMENTS           : Perturb rig setpoint while running under LQG control
C                           by either:
C                               - Perturbing randomly with variable generated
C                                  with a Gaussian PDF
C                               - Perturbing sinusoidally
C ***************************************************************************
       subroutine pervvn(vvn)
       implicit double precision (r)
$include: 'time.inc'
$include: 'syst.inc'
       integer order
       real vvn(15)

       order = int(syst(4))

       if (rtyp.eq.0) then
            do 100 i = 1, order
               vvn(i) = sngl(rndnrm(dble(0.0),dble(sqrt(rampl)),irand))
100         continue
       else
           do 300 i = 1, order
               if (rchset(i).eq.0)
     +            vvn(i) = rchdat(1,i)*sin(rchdat(2,i)*ti)
300         continue
       endif

       return
       end
C ***************************************************************************
CH     REVISION HISTORY   :
C      VERSION       BY                  DATE         COMMENT
C        1.1      A. de Waal            06/01/90    Finally Commented
C      FILEND             :
C ***************************************************************************
C
C      FILE               : GRLDAT.FOR
C ***************************************************************************
CN     MODULE NAME        : grldat
CA     FUNCTION           : Block Data Containing User Plot Data and Labels
CS     CALL SEQUENCE      : Not a subroutine
CI     INPUT PARAMETERS   : n/a
CO     OUTPUT PARAMETERS  : n/a
CG     GLOBAL VARIABLES   : Common blocks:  /plotop/ plotop,typsel,nosel
C                                           /scalop/ scalop,forsel,forno
C                                           /insnam/ insnam
C                                           /plcomm/ plcomm
CM     MODULES CALLED     : n/a
CE     ERROR CONDITIONS   : n/a
CC     COMMENTS           : Block data section containing plot and label data for
C                           user selected data plots.
C ***************************************************************************
       block data pldat
       integer*2 plotop(3,16,3),scalop(2,2,3),typsel,nosel,forsel,forno
       character*10 insnam(3,16)
       character*60 plcomm
       common /plotop/ plotop,typsel,nosel
       common /scalop/ scalop,forsel,forno
       common /insnam/ insnam
       common /plcomm/ plcomm

       data nosel/1/,typsel/1/,forsel/1/,forno/1/
       data (((plotop(k,j,i),i=1,3),j=1,16),k=1,3)
C      k=1: Inputs
     +/180,50,0,180,62,0,180,74,0,180,86,0,
     + 180,98,0,180,110,0,180,122,0,180,134,0,
     + 180,146,0,180,158,0,180,170,0,180,182,0,
     + 180,194,0,180,206,0,180,218,0,180,230,0,
C      k=2: Outputs
     + 360,50,0,360,62,0,360,74,0,360,86,0,
     + 360,98,0,360,110,0,360,122,0,360,134,0,
     + 360,146,0,360,158,0,360,170,0,360,182,0,
     + 360,194,0,360,206,0,360,218,0,360,230,0,
C      k=3: Setpoints
     + 540,50,0,540,62,0,540,74,0,540,86,0,
     + 540,98,0,540,110,0,540,122,0,540,134,0,
     + 540,146,0,540,158,0,540,170,0,540,182,0,
     + 540,194,0,540,206,0,540,218,0,540,230,0/
       data (((scalop(k,j,i),i=1,3),j=1,2),k=1,2)
C      k=1: Horizontal axis
     +/150,80,1800,150,95,0,
C      k=2: Vertical axis
     + 510,80,4095,510,95,0/
       data ((insnam(k,j),j=1,16),k=1,3)
C      k=1: Inputs
     +/' Valve C1 ',' Valve C2 ',' Valve C3 ',' Valve C4 ',
     + ' Valve C5 ',' Valve C6 ',
     + ' Stir S1  ',' Stir S2  ',' Stir S3  ',' Stir S4  ',
     + '          ','          ','          ','          ',
     + '          ','          ',
C      k=2: Outputs
     + ' Temp T1  ',' Temp T2  ',' Temp T3  ',' Temp T4  ',
     + ' Temp T5  ',' Temp T6  ',' Temp T7  ',' Temp T8  ',
     + ' Flow F1  ',' Flow F2  ',
     + ' Level L1 ',' Level L2 ',' Level L3 ',' Level L4 ',
     + ' Temp T9  ',' Temp T10 ',
C      k=3: Setpoints
     + ' Level L1 ',' Level L2 ',' Level L3 ',' Level L4 ',
     + ' Flow F1  ',' Flow F2  ',
     + ' Temp T1  ',' Temp T5  ',' Temp T8  ',' Temp T3  ',
     + '          ','          ','          ','          ',
     + '          ','          ',
       data plcomm
     +/'Heat Exchanger Responses                                          '/
       end
C ***************************************************************************
CH     REVISION HISTORY   :
C      VERSION       BY                  DATE         COMMENT
C        1.1      A. de Waal            06/01/90    Finally Commented
C      FILEND             :
```

```
C ***************************************************************************
C
C     FILE              : PRSEL.FOR
C ***************************************************************************
CN    MODULE NAME       : prsel
CA    FUNCTION          : Utility for Selecting Data for Plotting
CS    CALL SEQUENCE     : call prsel(typsel,nosel,xpos,ypos,opt,lev)
CI    INPUT PARAMETERS  : typsel: integer - Indicates type of data
C                                           1: Setpoints, Inputs
C                                           2: Outputs
C                         nosel : integer - Indicates input, output
C                                           setpoint number
C                         xpos,ypos: integer - Co-ords on screen where
C                                              editing to take place
C                         opt: integer - Current option
C                                        0: don't plot
C                                        1: plot
C                         lev: integer - background level
C                                        0: white
C                                        1: black
C                                        2: XOR with current background
CO    OUTPUT PARAMETERS : opt: integer - Returned option
CG    GLOBAL VARIABLES  : None
CM    MODULES CALLED    : None
CE    ERROR CONDITIONS  : None
CC    COMMENTS          : Allows user to choose inputs, outputs and/or
C                         setpoints to be plotted by editing options on
C                         graphics page 0.
C ***************************************************************************
      subroutine prsel(typsel,nosel,xpos,ypos,opt,lev)
      integer*2 typsel,nosel,xpos,ypos,opt,lev
      character*3 flg
      call LEVEL(lev)
      if (opt.eq.0) then
          flg = 'OFF'
      else
          flg = 'ON '
      endif
      if (typsel.eq.2) then
          call WRTSTR(0,xpos,ypos,3,flg)
      else
          if (nosel.le.10) call WRTSTR(0,xpos,ypos,3,flg)
      endif
      call LEVEL(1)
      return
      end
C ***************************************************************************
CH    REVISION HISTORY  :
C     VERSION       BY                DATE         COMMENT
C       1.1     A. de Waal          06/01/90    Finally Commented
C     FILEND            :
C ***************************************************************************
C
C     FILE              : RBLOCK.FOR
C ***************************************************************************
CN    MODULE NAME       : rblock
CA    FUNCTION          : Draw Reactor Block on Heat Exch Block Diag Shell
CS    CALL SEQUENCE     : call rblock(reac)
CI    INPUT PARAMETERS  : reac : integer - reactor block number
CO    OUTPUT PARAMETERS : None
CG    GLOBAL VARIABLES  : Include files: heblock.inc
CM    MODULES CALLED    : Fortran  :  arhead
C                         Assembler:  MOVE,DLINE,BOX,WRTSTR
CE    ERROR CONDITIONS  : None
CC    COMMENTS          : Called from subroutine heblock(for) to draw
C                         reactor block shell units.  Draws a reactor block
C                         shell unit writing 'Tank' in the block.
C ***************************************************************************
      subroutine rblock(reac)
$include: 'heblock.inc'
      integer*2 reac

      external MOVE
      external DLINE
      external BOX
      external WRTSTR
      external arhead

      y = y + (3*th/4)
      call BOX(x,y,tw,th)

      xpos = x + (tw - 4*chw)/2
      ypos = y - 2.5*chh
      call WRTSTR(page,xpos,ypos,4,'Tank')

      x = x + tw
      y = y - (th/4)
      call MOVE(x,y)
      call arhead(2,8)
      x = x + hline
      call DLINE(x,y)

      x = x - hline
      y = y - th/2
      call MOVE(x,y)
      x = x + hline
      call DLINE(x,y)

      if (reac.ne.1) then
          call arhead(1,8)
      endif

      return
      end
C ***************************************************************************
CH    REVISION HISTORY  :
C     VERSION       BY                DATE         COMMENT
C       1.0     A. de Waal          17/05/89      Creation
C       1.1     A. de Waal          06/01/90    Finally Commented
C     FILEND            :
C ***************************************************************************
C
C     FILE              : RDEGRC.FOR
C ***************************************************************************
CN    MODULE NAME       : rdegrc
CA    FUNCTION          : Convert Integer Number to Real Degrees Celcius
```

```
CS    CALL SEQUENCE     : value = rdegrc(ivalue)
CI    INPUT PARAMETERS  : ivalue: integer - Integer number in range 0 to 4095
C                                           to be converted
CO    OUTPUT PARAMETERS : rdegrc: real    - Converted real degree celcius in
C                                           20 to 80
CG    GLOBAL VARIABLES  : None
CM    MODULES CALLED    : None
CE    ERROR CONDITIONS  : None
CC    COMMENTS          : Converts integer expression for temperature in
C                         the range 0 to 4095 to real degrees celcius in
C                         range 20 to 80 degrees
C ***************************************************************************
      real*4 function rdegrc(ivalue)
      integer*2 ivalue
      rdegrc = REAL(ivalue)*60.0/4095.0 + 20.0
      return
      end
C ***************************************************************************
CH    REVISION HISTORY  :
C     VERSION       BY                  DATE        COMMENT
C       1.1     A. de Waal            06/01/90   Finally Commented
C     FILEND            :
C ***************************************************************************
C
C     FILE              : RPERC.FOR
C ***************************************************************************
CN    MODULE NAME       : rperc
CA    FUNCTION          : Convert Integer Number to Real Percentage
CS    CALL SEQUENCE     : rpercent = rperc(ivalue)
CI    INPUT PARAMETERS  : ivalue : integer - Integer value in the range
C                                            0 to 4095 to be converted
CO    OUTPUT PARAMETERS : rperc  : real - Converted real percentage in
C                                         the range 0 to 100 percent
CG    GLOBAL VARIABLES  : None
CM    MODULES CALLED    : None
CE    ERROR CONDITIONS  : None
CC    COMMENTS          : Converts integer number in range 0 to 4095 to a real
C                         expression of percentage in range 0 to 100 percent
C ***************************************************************************
      real*4 function rperc(ivalue)
      integer*2 ivalue
      rperc = REAL(ivalue)/40.95
      return
      end
C ***************************************************************************
CH    REVISION HISTORY  :
C     VERSION       BY                  DATE        COMMENT
C       1.1     A. de Waal            06/01/90   Finally Commented
C     FILEND            :
C ***************************************************************************
C
C     FILE              : RUNINI.FOR
C ***************************************************************************
CN    MODULE NAME       : runini
CA    FUNCTION          : Initialize Screens/Flags for Running of Rig
CS    CALL SEQUENCE     : call runini()
CI    INPUT PARAMETERS  : None
CO    OUTPUT PARAMETERS : None
CG    GLOBAL VARIABLES  : Include files:  runvar.inc
CM    MODULES CALLED    : Fortran  :  heblock
C                         Assembler:  GPAGE,CLRSCR,DISP
CE    ERROR CONDITIONS  : None
CC    COMMENTS          : Initializes flags and run screen to enable the
C                         of the rig - called from rig running subroutines
C                         herun() and lqgrun().
C ***************************************************************************
      subroutine runini()
$include: 'runvar.inc'
      log = .false.
      chscr = .true.
      quit = .false.
      call GPAGE(0)
      call CLRSCR()
      call DISP(0)
      call heblock()
      return
      end
C ***************************************************************************
CH    REVISION HISTORY  :
C     VERSION       BY                  DATE        COMMENT
C     FILEND            :
C ***************************************************************************
C
C     FILE              : RUNKEY.FOR
C ***************************************************************************
CN    MODULE NAME       : runkey
CA    FUNCTION          : Service Keyboard Run Mode
CS    CALL SEQUENCE     : call runkey()
CI    INPUT PARAMETERS  : None
CO    OUTPUT PARAMETERS : None
CG    GLOBAL VARIABLES  : Include files:  runvar.inc,iovar.inc,keys.inc
C                         Common blocks:  /sampl/ sampl
C                                         /stepid/ stepid
C                                         /modfr/ modfr
CM    MODULES CALLED    : Fortran  :  keyerr,imxmn,imxmns
C                         Assembler:  INKEY,ERTONE,LEVEL,WRTSTR
CE    ERROR CONDITIONS  : None
CC    COMMENTS          : Services keyboard while rig is being run to determine
C                         whether the user requires a change in the running
C                         mode.
C ***************************************************************************
      subroutine runkey()
      implicit integer*2 (I)
      implicit real*8 (m)
$include: 'runvar.inc'
$include: 'iovar.inc'
$include: 'keys.inc'

      integer*2 numkey/-1/,oldnum/-1/,sampl,stepid
      logical*2 modfr
      common /sampl/ sampl
      common /stepid/ stepid
      common /modfr/ modfr

      key = INKEY(4)
```

```
if (key.ne.0) then

    if (mode.eq.1) then
        if ((key.eq.bgs).or.(key.eq.sms)) then
            mode = 3
            chscr = .true.
        elseif ((key.eq.bga).or.(key.eq.sma)) then
            if (.not.modfr) then
                mode = 5
                chscr = .true.
            else
                call ERTONE()
                call LEVEL(0)
                call WRTSTR(0,((720-9*42)/2),320,42,
+                   'CANNOT CHANGE MODE WHILE LOGGING STEP TEST')
                call LEVEL(1)
                chscr = .true.
            endif
        elseif ((key.eq.bgl).or.(key.eq.sml)) then
            if (.not.log) then
                sampl = 0
                log = .true.
            else
                log = .false.
                chscr = .true.
            endif
            chscr = .true.
        elseif (key.eq.esck) then
            quit = .true.
        endif
        oldnum = numkey
        numkey = (key - 12288)/256
        if ((numkey.le.9).and.(numkey.ge.0)) then
            if ((.not.modfr).or.(numkey.eq.oldnum)) then
                mode = 2
                inpno = numkey
                if (numkey.eq.0) inpno = inpno + 10
                chscr = .true.
            else
                call ERTONE()
                call LEVEL(0)
                call WRTSTR(0,((720-9*42)/2),320,42,
+                   'CANNOT CHANGE MODE WHILE LOGGING STEP TEST')
                call LEVEL(1)
                chscr = .true.
            endif
        else
            call keyerr()
        endif

    elseif (mode.eq.2) then
        if (key.eq.upk) then
            input(inpno) = imxmn(input(inpno) + 41)
        elseif (key.eq.pgupk) then
            input(inpno) = imxmn(input(inpno) + 410)
        elseif (key.eq.downk) then
            input(inpno) = imxmn(input(inpno) - 41)
        elseif (key.eq.pgdnk) then
            input(inpno) = imxmn(input(inpno) - 410)
        elseif (key.eq.retk) then
            mode = 1
            chscr = .true.
        else
            call keyerr()
        endif

    elseif (mode.eq.3) then
        if ((key.eq.bgm).or.(key.eq.smm)) then
            if (.not.modfr) then
                mode = 1
                chscr = .true.
            else
                call ERTONE()
                call LEVEL(0)
                call WRTSTR(0,((720-9*42)/2),320,42,
+                   'CANNOT CHANGE MODE WHILE LOGGING STEP TEST')
                call LEVEL(1)
                chscr = .true.
            endif
        elseif ((key.eq.bgl).or.(key.eq.sml)) then
            if (.not.log) then
                sampl = 0
                log = .true.
            else
                log = .false.
                chscr = .true.
            endif
            chscr = .true.
        elseif (key.eq.esck) then
            quit = .true.
        endif
        oldnum = numkey
        numkey = (key - 12288)/256
        if ((numkey.le.9).and.(numkey.ge.0)) then
            if ((.not.modfr).or.(numkey.eq.oldnum)) then
                mode = 4
                inpno = numkey
                if (numkey.eq.0) inpno = inpno + 10
                chscr = .true.
            else
                call ERTONE()
                call LEVEL(0)
                call WRTSTR(0,((720-9*42)/2),320,42,
+                   'CANNOT CHANGE MODE WHILE LOGGING STEP TEST')
                call LEVEL(1)
                chscr = .true.
            endif
        else
            call keyerr()
        endif

    elseif (mode.eq.4) then
        if (key.eq.upk) then
            steplv = imxmns(steplv + 41)
        elseif (key.eq.pgupk) then
            steplv = imxmns(steplv + 410)
```

```
        elseif (key.eq.downk) then
            steplv = imxmns(steplv - 41)
        elseif (key.eq.pgdnk) then
            steplv = imxmns(steplv -  410)
        elseif (key.eq.retk) then
            input(inpno) = imxmn( input(inpno) +
     +          INT(ANINT(REAL(steplv)*40.95/40.10)) )
            stepid = inpno
            mode = 3
            chscr = .true.
        else
            call keyerr()
        endif

    elseif (mode.eq.5) then
        if ((key.eq.bgs).or.(key.eq.sms)) then
            mode = 7
            chscr = .true.
        elseif ((key.eq.bgm).or.(key.eq.smm)) then
            if (.not.modfr) then
                mode = 1
                chscr = .true.
            else
                call ERTONE()
                call LEVEL(0)
                call WRTSTR(0,((720-9*42)/2),320,42,
     +              'CANNOT CHANGE MODE WHILE LOGGING STEP TEST')
                call LEVEL(1)
                chscr = .true.
            endif
        elseif ((key.eq.bgl).or.(key.eq.sml)) then
            if (.not.log) then
                sampl = 0
                log = .true.
            else
                log = .false.
                chscr = .true.
            endif
            chscr = .true.
        elseif (key.eq.esck) then
            quit = .true.
        endif
        oldnum = numkey
        numkey = (key - 12288)/256
        if ((numkey.le.9).and.(numkey.ge.0)) then
            if ((.not.modfr).or.(numkey.eq.oldnum)) then
                mode = 6
                setpno = numkey
                if (numkey.eq.0) setpno = setpno + 10
                chscr = .true.
            else
                call ERTONE()
                call LEVEL(0)
                call WRTSTR(0,((720-9*42)/2),320,42,
     +              'CANNOT CHANGE MODE WHILE LOGGING STEP TEST')
                call LEVEL(1)
                chscr = .true.
            endif
        else
            call keyerr()
        endif

    elseif (mode.eq.6) then
        if (key.eq.upk) then
            if (setpno.ge.7) then
                setp(setpno) = imxmn(setp(setpno) + 68)
            else
                setp(setpno) = imxmn(setp(setpno) + 41)
            endif
        elseif (key.eq.pgupk) then
            if (setpno.ge.7) then
                setp(setpno) = imxmn(setp(setpno) +  683)
            else
                setp(setpno) = imxmn(setp(setpno) + 410)
            endif
        elseif (key.eq.downk) then
            if (setpno.ge.7) then
                setp(setpno) = imxmn(setp(setpno) - 68)
            else
                setp(setpno) = imxmn(setp(setpno) - 41)
            endif
        elseif (key.eq.pgdnk) then
            if (setpno.ge.7) then
                setp(setpno) = imxmn(setp(setpno) -  683)
            else
                setp(setpno) = imxmn(setp(setpno) - 410)
            endif
        elseif (key.eq.retk) then
            mode = 5
            chscr = .true.
        else
            call keyerr()
        endif

    elseif (mode.eq.7) then
        if ((key.eq.bga).or.(key.eq.sma)) then
            if (.not.modfr) then
                mode = 5
                chscr = .true.
            else
                call ERTONE()
                call LEVEL(0)
                call WRTSTR(0,((720-9*42)/2),320,42,
     +              'CANNOT CHANGE MODE WHILE LOGGING STEP TEST')
                call LEVEL(1)
                chscr = .true.
            endif
        elseif ((key.eq.bgl).or.(key.eq.sml)) then
            if (.not.log) then
                sampl = 0
                log = .true.
            else
                log = .false.
                chscr = .true.
            endif
            chscr = .true.
```

```
          elseif (key.eq.esck) then
              quit = .true.
          endif
          oldnum = numkey
          numkey = (key - 12288)/256
          if ((numkey.le.9).and.(numkey.ge.0)) then
              if ((.not.modfr).or.(numkey.eq.oldnum)) then
                  mode = 8
                  setpno = numkey
                  if (numkey.eq.0) setpno = setpno + 10
                  chscr = .true.
              else
                  call ERTONE()
                  call LEVEL(0)
                  call WRTSTR(0,((720-9*42)/2),320,42,
     +                'CANNOT CHANGE MODE WHILE LOGGING STEP TEST')
                  call LEVEL(1)
                  chscr = .true.
              endif
          else
              call keyerr()
          endif

      elseif (mode.eq.8) then
          if (key.eq.upk) then
              steplv = imxmns(steplv + 41)
          elseif (key.eq.pgupk) then
              steplv = imxmns(steplv +  410)
          elseif (key.eq.downk) then
              steplv = imxmns(steplv - 41)
          elseif (key.eq.pgdnk) then
              steplv = imxmns(steplv -  410)
          elseif (key.eq.retk) then
              if (setpno.ge.7) then
                  setp(setpno) = imxmn( setp(setpno) +
     +                    INT(ANINT(REAL(steplv)*68.25/40.95)) )
              else
                  setp(setpno) = imxmn( setp(setpno) +
     +            INT(ANINT(REAL(steplv)*40.95/40.10)) )
              endif
              stepid = setpno
              mode = 7
              chscr = .true.
          else
              call keyerr()
          endif
      endif

      endif
      if ((log).and.(stepid.ne.-1)) then
          modfr = .true.
      else
          modfr = .false.
      endif
      key = INKEY(2)

      return
      end

C ***************************************************************************
CH    REVISION HISTORY   :
C     VERSION       BY                DATE        COMMENT
C       1.1      A. de Waal         06/01/90    Finally Commented
C     FILEND                 :
C ***************************************************************************
C
C     FILE                   : RUNRIG.FOR
C ***************************************************************************
CN    MODULE NAME            : runrig
CA    FUNCTION               : Drive Rig Running Menu
CS    CALL SEQUENCE          : call runrig()
CI    INPUT PARAMETERS       : None
CO    OUTPUT PARAMETERS      : None
CG    GLOBAL VARIABLES       : None
CM    MODULES CALLED         : Fortran   :  inmenu,wrhead,herun,lqgtyp
CE    ERROR CONDITIONS       : None
CC    COMMENTS               : Initializes menu screen and drives menu for selection
C                              of control conditions (LQG or stabilizing) under
C                              which rig is to be run.
C ***************************************************************************
      subroutine runrig()
      implicit integer*2 (D)
      integer*2 opt12

99    continue

      call inmenu()

C Write heading. (wrhead.for - iglib.lib)
      call wrhead(0,(720-40*9)/2,35,40,
     +  'Heat Exchanger Rig Running and Analysis.')

C Get chosen option. (domenu.asm - iglib.lib)
      opt12 = DOMENU(12)

C Blank over heading. (util.asm - iglib.lib)
      call WRTSTR(0,(720-40*9)/2,35,40,
     +  '                                        ')

      if (opt12.eq.1) then
          call herun()

      elseif (opt12.eq.2) then
          call lqgtyp()

      elseif (opt12.ne.0) then
          call ERTONE()

      endif

      if (opt12.ne.0) then
          goto 99
      endif

      return
```

```
      end
C ********************************************************************************
CH    REVISION HISTORY  :
C     VERSION       BY                  DATE         COMMENT
C       1.1     A. de Waal            06/01/90   Finally Commented
C     FILEND          :
C ********************************************************************************
C
C     FILE              : HEBDAT.FOR
C ********************************************************************************
CN    MODULE NAME       : hebdat
CA    FUNCTION          : Block Data of HE Block Diag Run Screen Labels
CS    CALL SEQUENCE     : Not a subroutine
CI    INPUT PARAMETERS  : n/a
CO    OUTPUT PARAMETERS : n/a
CG    GLOBAL VARIABLES  : Common blocks:  /scrlen/ titlen,brflen,optlen
C                                         /scrnam/ optitl,optbrf,opt
CM    MODULES CALLED    : n/a
CE    ERROR CONDITIONS  : n/a
CC    COMMENTS          : Block data section containing heat exchanger run
C                         screen labels and briefs to be printed on the block
C                         diagram while running under the various modes under
C                         manual or automatic control.
C ********************************************************************************
      block data scrdat
      integer*2 titlen,brflen,optlen
      character*60 optitl(8),optbrf(8),opt(8,3)

      common /scrlen/ titlen,brflen,optlen
      common /scrnam/ optitl,optbrf,opt

      data titlen/60/,brflen/60/,optlen/60/
      data (optitl(i),i=1,8)
     +/'      Manual Control of Rig: Selecting Input to Change       ',
     + '      Manual Control of Rig: Changing Indicated Input        ',
     + '      O/L Step Tests on Rig: Selecting Input to Step         ',
     + '      O/L Step Tests on Rig: Stepping Indicated Input        ',
     + '   Automatic Control of Rig: Selecting Setpoint to Change    ',
     + '   Automatic Control of Rig: Changing Indicated Setpoint     ',
     + '     C/L Step Tests on Rig: Selecting Setpoint to Step       ',
     + '     C/L Step Tests on Rig: Stepping Indicated Setpoint      '/
      data (optbrf(i),i=1,8)
     +/'      Use keys shown to select input or to change mode       ',
     + '       Use keys shown to change or accept input value        ',
     + '  Use keys shown to select input to step or to change mode   ',
     + '  Use keys shown to change or accept input step size value   ',
     + '     Use keys shown to select setpoint or to change mode     ',
     + '      Use keys shown to change or accept setpoint value      ',
     + ' Use keys shown to select setpoint to step or to change mode',
     + ' Use keys shown to change or accept setpoint step size value'/
      data ((opt(j,i),i=1,3),j=1,8)
     +/'INPUT SELECTION:   1:C1  2:C2  3:C3  4:C4  5:C5  6:C6       ',
     + '                   7:S1  8:S2  9:S3  0:S4                   ',
     + 'MODE SELECTION :   S:Step  A:Auto mode  L:Log toggle        ',
     + 'ACTION:   UP :Incr input  1%          DOWN :Decr input  1%  ',
     + '        PGUP :Incr input 10%        PGDOWN :Decr input 10%  ',
     + '      RETURN :Accept input value        (% of input range)  ',
     + 'STEP SELECTION :   1:C1  2:C2  3:C3  4:C4  5:C5  6:C6       ',
     + '                   7:S1  8:S2  9:S3  0:S4                   ',
     + 'MODE SELECTION :           M:Man mode  L:Log toggle         ',
     + 'ACTION:    UP :Incr step  1%          DOWN :Decr step  1%   ',
     + '         PGUP :Incr step 10%        PGDOWN :Decr step 10%   ',
     + '       RETURN :Accept step size        (% of input range)   ',
     + 'SETP. SELECTION:   1:L1  2:L2  3:L3  4:L4  5:F1  6:F2       ',
     + '                   7:T1  8:T5  9:T8  0:T3                   ',
     + 'MODE SELECTION :   S:Step  M:Man mode   L:Log toggle        ',
     + 'ACTION: UP :Incr setp  1%,  1°C        DOWN :Decr setp  1%,  1°C',
     + '      PGUP :Incr setp 10%, 10°C      PGDOWN :Decr setp 10%, 10°C',
     + '    RETURN :Accept setp value    (% of setp range,  °C temp)',
     + 'SETP. SELECTION:   1:L1  2:L2  3:L3  4:L4  5:F1  6:F2       ',
     + '                   7:T1  8:T5  9:T8  0:T3                   ',
     + 'MODE SELECTION :           A:Auto mode  L:Log toggle        ',
     + 'ACTION: UP :Incr step  1%,  1°C     DOWN :Decr step  1%,  1°C',
     + '      PGUP :Incr step 10%, 10°C   PGDOWN :Decr step 10%, 10°C',
     + '    RETURN :Accept step size     (% of setp range,  °C temp)'/
      end
C ********************************************************************************
CH    REVISION HISTORY  :
C     VERSION       BY                  DATE         COMMENT
C       1.1     A. de Waal            06/01/90   Finally Commented
C     FILEND          :
C ********************************************************************************
C
C     FILE              : SELPL.FOR
C ********************************************************************************
CN    MODULE NAME       : selpl
CA    FUNCTION          : Controls Selection of Data for User Plots
CS    CALL SEQUENCE     : call selpl()
CI    INPUT PARAMETERS  : None
CO    OUTPUT PARAMETERS : None
CG    GLOBAL VARIABLES  : Include files:  keys.inc
C                         Common blocks:  /runmod/ mode
C                                         /plotop/ plotop,typsel,nosel
C                                         /insnam/ insnam
CM    MODULES CALLED    : Fortran  :  wrhead,prsel,servop
C                         Assembler:  WIPSCR,DISP,LEVEL,INKEY
CE    ERROR CONDITIONS  : None
CC    COMMENTS          : Controls the selection by the user of various input
C                         output and setpoint response data for plotting.
C ********************************************************************************
      subroutine selpl()
      implicit integer*2 (I)
$include: 'keys.inc'
      integer*2 plotop(3,16,3),typsel,nosel
      integer*2 i,j,k,key,mode
      character*3 flg
      character*10 insnam(3,16)
      common /runmod/ mode
      common /plotop/ plotop,typsel,nosel
      common /insnam/ insnam
      call WIPSCR(0)
      call DISP(0)
      call wrhead(0,((720-9*54)/2),20,54,
     +   'Selecting Inputs, Outputs and Setpoints to be plotted.')
      call WRTSTR(0,20,248,58,
     +   'Spacebar to toggle, Arrow keys to move between selections.')
```

```
      call WRTSTR(0,20,262,58,
     +   'Esc key to accept selection and return to previous menu.  ')
      call LEVEL(0)
      call WRTSTR(0,(plotop(1,1,1)-9*11),(plotop(1,1,2)-15),7,
     +            'Inputs:')
      call WRTSTR(0,(plotop(2,1,1)-9*11),(plotop(2,1,2)-15),8,
     +            'Outputs:')
      if (mode.ge.5) then
          call WRTSTR(0,(plotop(3,1,1)-9*11),(plotop(3,1,2)-15),10,
     +            'Setpoints:')
      endif
      call LEVEL(1)
      typsel = 1
      nosel = 1
      do 900 k = 1,3
          do 800 j = 1,16
              if (mode.le.4) then
                  if (k.le.2) then
                      call WRTSTR(0,(plotop(k,j,1)-9*11),plotop(k,j,2),
     +                            10,insnam(k,j))
                      call prsel(k,j,plotop(k,j,1),plotop(k,j,2),
     +                           plotop(k,j,3),1)
                  endif
              else
                  call WRTSTR(0,(plotop(k,j,1)-9*11),plotop(k,j,2),
     +                        10,insnam(k,j))
                  call prsel(k,j,plotop(k,j,1),plotop(k,j,2),
     +                       plotop(k,j,3),1)
              endif
800       continue
900   continue
      call prsel(typsel,nosel,plotopt(1,1,1),plotopt(1,1,2),
     +           plotop(1,1,3),0)
910   continue
      key = INKEY(1)
      call servop(key)
      if (key.ne.esck) goto 910
      return
      end
C ********************************************************************************
CH    REVISION HISTORY  :
C     VERSION       BY                    DATE        COMMENT
C       1.1     A. de Waal            06/01/90   Finally Commented
C     FILEND            :
C ********************************************************************************
C
C     FILE              : SERVOP.FOR
C ********************************************************************************
CN    MODULE NAME       : servop
CA    FUNCTION          : Service Keyb. while Select Resp Data for Plots
CS    CALL SEQUENCE     : call servop(key)
CI    INPUT PARAMETERS  : key: integer - key pressed
CO    OUTPUT PARAMETERS : None
CG    GLOBAL VARIABLES  : Include files:  keys.inc
C                         Common blocks:  /runmod/ mode
C                                         /plotop/ plotop,typsel,nosel
CM    MODULES CALLED    : Fortran  :  prsel
CE    ERROR CONDITIONS  : None
CC    COMMENTS          : Prints out current selection status of all
C                         inputs/outputs/setpoints using subroutine prsel(...)
C                         Then interrogates keyboard and acts as following in
C                         response to the indicated keys being pressed:
C                             - Arrow keys: Set flag to highlight different
C                                           input/output/setpoint
C                             - Spacebar  : Toggle selection status of
C                                           presently highlighted
C                                           input/output/setpoint
C                          Once this has been done, prints out current
C                          selection status of all inputs/outputs/setpoints
C                          using subroutine prsel(...).
C ********************************************************************************
      subroutine servop(key)
$include: 'keys.inc'
      integer*2 plotop(3,16,3),mode,typsel,nosel
      common /runmod/ mode
      common /plotop/ plotop,typsel,nosel
      call prsel(typsel,nosel,plotop(typsel,nosel,1),
     +           plotop(typsel,nosel,2),plotop(typsel,nosel,3),1)
      if (key.eq.upk) then
          nosel = nosel - 1
          if (typsel.ne.2) then
              if (nosel.eq.0) nosel = 10
          else
              if (nosel.eq.0) nosel = 16
          endif
      endif
      if (key.eq.downk) then
          nosel = nosel + 1
          if (typsel.ne.2) then
              if (nosel.eq.11) nosel = 1
          else
              if (nosel.eq.17) nosel = 1
          endif
      endif
      if (key.eq.leftk) then
          typsel = typsel - 1
          if (mode.le.4) then
              if (typsel.eq.0) typsel = 2
          else
              if (typsel.eq.0) typsel = 3
          endif
          if (nosel.gt.10) nosel = 10
      endif
      if (key.eq.rightk) then
          typsel = typsel + 1
          if (mode.le.4) then
              if (typsel.eq.3) typsel = 1
          else
              if (typsel.eq.4) typsel = 1
          endif
          if (nosel.gt.10) nosel = 10
      endif
C If spacebar pressed, toggle flag
      if (key.eq.8192) then
          if (plotop(typsel,nosel,3).eq.0) then
              plotop(typsel,nosel,3) = 1
```

```
          else
              plotop(typsel,nosel,3) = 0
          endif
      endif
      call prsel(typsel,nosel,plotop(typsel,nosel,1),
     +          plotop(typsel,nosel,2),plotop(typsel,nosel,3),0)
      return
      end
C ********************************************************************************
CH    REVISION HISTORY  :
C     VERSION      BY                  DATE         COMMENT
C       1.1     A. de Waal           06/01/90   Finally Commented
C     FILEND            :
C ********************************************************************************


C
C     FILE              : TYPSCR.FOR
C ********************************************************************************
CN    MODULE NAME       : typscr
CA    FUNCTION          : Type Headings on Heat Exchanger Block Diagram
CS    CALL SEQUENCE     : call typscr()
CI    INPUT PARAMETERS  : None
CO    OUTPUT PARAMETERS : None
CG    GLOBAL VARIABLES  : Include files:  heblock.inc,screen.inc,runvar.inc
C                         Common blocks:  /scrcon/ gap,blh,bltop,stptl,stptbl
CM    MODULES CALLED    : Assembler:  WRTSTR,BLKFIL,LEVEL,BOX
CE    ERROR CONDITIONS  : None
CC    COMMENTS          : To type out, to the block diagram, the names of the
C                         measurement and control points on the rig for the
C                         different modes under which the rig is running.
C ********************************************************************************
      subroutine typscr()
$include: 'heblock.inc'
$include: 'screen.inc'
$include: 'runvar.inc'
      integer*2 naml,overwl,i,page,typbl,gap,blh,bltop
      integer*2 indl,bxh,xtpos,samtl,stptl,stptbl,len
      common /scrcon/ gap,blh,bltop,stptl,stptbl

      page = 0
      naml = 4
      overwl = 5
      samtl = 6
      stptbl = 17
      stptl = 11

C Typing out the names of the measurement and control points on the rig
C for operating modes: manual control(1), o/l step tests(3)
C                      automatic control(5), c/l step tests(7)
c      if ((mode.eq.1).or.(mode.eq.3).or.
c     +    (mode.eq.5).or.(mode.eq.7)) then
          if (mode.le.4) then
              typbl = 1
          else
              typbl = 2
          endif
          do 900 i = 1, 10
              call WRTSTR(page,posts(1,i),posts(2,i),naml,
     +                    tsnam(typbl,i))
              if (mode.le.4) then
                  xpos = posts(1,i) - chw
                  ypos = posts(2,i) + chh
                  call WRTSTR(page,xpos,ypos,overwl,'   °C')
                  ypos = ypos + chh
                  call WRTSTR(page,xpos,ypos,overwl,'     ')
              endif
              if (mode.ge.5) then
                  xpos = posts(1,i) - chw
                  ypos = posts(2,i) + chh
                  if ((i.eq.1).or.(i.eq.5).or.
     +                (i.eq.8).or.(i.eq.3)) then
                      call WRTSTR(page,xpos,ypos,overwl,'r  °C')
                      ypos = ypos + chh
                      call WRTSTR(page,xpos,ypos,overwl,'y  °C')
                  else
                      call WRTSTR(page,xpos,ypos,overwl,'   °C')
                      ypos = ypos + chh
                      call WRTSTR(page,xpos,ypos,overwl,'     ')
                  endif
              endif
              if (i.le.6) then
                  call WRTSTR(page,poscv(1,i),poscv(2,i),naml,
     +                        cvnam(typbl,i))
                  call WRTSTR(page,(poscv(1,i)-chw),(poscv(2,i)+chh),
     +                        overwl,'    %')
                  if (i.le.4) then
                      call WRTSTR(page,posls(1,i),posls(2,i),naml,
     +                            lsnam(typbl,i))
                      if (mode.le.4) then
                          xpos = posls(1,i) - chw
                          ypos = posls(2,i) + chh
                          call WRTSTR(page,xpos,ypos,overwl,'    %')
                          ypos = ypos + chh
                          call WRTSTR(page,xpos,ypos,overwl,'     ')
                      endif
                      if (mode.ge.5) then
                          xpos = posls(1,i) - chw
                          ypos = posls(2,i) + chh
                          call WRTSTR(page,xpos,ypos,overwl,'r   %')
                          ypos = ypos + chh
                          call WRTSTR(page,xpos,ypos,overwl,'y   %')
                      endif
                      call WRTSTR(page,posst(1,i),posst(2,i),naml,
     +                            stnam(typbl,i))
                      call WRTSTR(page,(posst(1,i)-chw),
     +                            (posst(2,i)+chh),overwl,'    %')
                      call WRTSTR(page,post(1,i),post(2,i),naml,
     +                            tnam(1))
                      if (i.le.2) then
                          call WRTSTR(page,posfs(1,i),
     +                                posfs(2,i),naml,fsnam(typbl,i))
                          if (mode.le.4) then
                              xpos = posfs(1,i) - chw
                              ypos = posfs(2,i) + chh
                              call WRTSTR(page,xpos,ypos,overwl,
```

```
     +                                        '    %')
                              ypos = ypos + chh
                              call WRTSTR(page,xpos,ypos,overwl,
     +                                        '    ')
                          endif
                          if (mode.ge.5) then
                              xpos = posfs(1,i) - chw
                              ypos = posfs(2,i) + chh
                              call WRTSTR(page,xpos,ypos,overwl,
     +                                        'r   %')
                              ypos = ypos + chh
                              call WRTSTR(page,xpos,ypos,overwl,
     +                                        'y   %')
                          endif
                      endif
                  endif
              endif
900           continue
c         endif

C Highlighting an input on the diagram if it has been selected
C to be changed or stepped
          if ((mode.eq.2).or.(mode.eq.4)) then
              typbl = 1
              if (inpno.le.6) then
                  xpos = poscv(1,inpno)
                  ypos = poscv(2,inpno)
                  call LEVEL(0)
                  call WRTSTR(page,xpos,ypos,naml,cvnam(typbl,inpno))
                  call LEVEL(1)
              elseif (inpno.gt.6) then
                  xpos = posst(1,(inpno-6))
                  ypos = posst(2,(inpno-6))
                  call LEVEL(0)
                  call WRTSTR(page,xpos,ypos,naml,stnam(typbl,(inpno-6)))
                  call LEVEL(1)
              endif
          endif

C Highlighting a setpoint on the diagram if it has been selected
C to be changed or stepped
          if ((mode.eq.6).or.(mode.eq.8)) then
              typbl = 2
              if (setpno.le.4) then
                  xpos = posls(1,setpno)
                  ypos = posls(2,setpno)
                  call LEVEL(0)
                  call WRTSTR(page,xpos,ypos,naml,lsnam(typbl,setpno))
                  call LEVEL(1)
              endif
              if ((setpno.gt.4).and.(setpno.le.6)) then
                  xpos = posfs(1,(setpno-4))
                  ypos = posfs(2,(setpno-4))
                  call LEVEL(0)
                  call WRTSTR(page,xpos,ypos,naml,fsnam(typbl,(setpno-4)))
                  call LEVEL(1)
              endif
              if (setpno.eq.7) then
                  xpos = posts(1,1)
                  ypos = posts(2,1)
                  call LEVEL(0)
                  call WRTSTR(page,xpos,ypos,naml,tsnam(typbl,1))
                  call LEVEL(1)
              endif
              if (setpno.eq.8) then
                  xpos = posts(1,5)
                  ypos = posts(2,5)
                  call LEVEL(0)
                  call WRTSTR(page,xpos,ypos,naml,tsnam(typbl,5))
                  call LEVEL(1)
              endif
              if (setpno.eq.9) then
                  xpos = posts(1,8)
                  ypos = posts(2,8)
                  call LEVEL(0)
                  call WRTSTR(page,xpos,ypos,naml,tsnam(typbl,8))
                  call LEVEL(1)
              endif
              if (setpno.eq.10) then
                  xpos = posts(1,3)
                  ypos = posts(2,3)
                  call LEVEL(0)
                  call WRTSTR(page,xpos,ypos,naml,tsnam(typbl,3))
                  call LEVEL(1)
              endif
          endif

C For each of the respective modes of operation, typing out the:
C         Title of the mode of operation
C         Brief instructing the user about option selection
C         Options available to user
C
C Typing out title
          xpos = (720 - titlen*chw)/2
          ypos = y0 - resh - 0.3*chh
          call WRTSTR(page,xpos,ypos,titlen,optitl(mode))
          ypos = ypos + 0.25*chh
          bxh = 7*chh/6
          call BOX(xpos,ypos,(titlen*chw),bxh)

C Typing out brief
          xpos = (720 - brflen*chw)/2
          ypos = bltop + blh + chh
          len = brflen*chw
          call BLKFIL(xpos,ypos,len,chh)
          call LEVEL(0)
          call WRTSTR(page,xpos,ypos,brflen,optbrf(mode))
          call LEVEL(1)

C Typing out options (also typing headings indicating the sample time)
          ypos = optyin + chh/3
          xpos = optxin + 2*chw
          xtpos = xpos + 64*chw
          do 800 i = 1,3
              call WRTSTR(page,xpos,ypos,optlen,opt(mode,i))
              if (i.eq.1) call WRTSTR(page,xtpos,ypos,samtl,'Contr.')
```

```
            if (i.eq.2) call WRTSTR(page,xtpos,ypos,samtl,'Time =')
            if (i.eq.3) then
                xtpos = xtpos + 6*chw
                call WRTSTR(page,xtpos,ypos,samtl,'(sec) ')
            endif
            ypos = ypos + chh
800     continue

C Typing out headings indicating the step size
C   (for manual and automatic modes)
        xpos = (720 - stptbl*chw)/2
        ypos = bltop + blh - chh
        len = stptbl*chw
        if ((mode.eq.4).or.(mode.eq.8)) then
            call BLKFIL(xpos,(ypos+chh/4),len,(5*chh/4))
            call LEVEL(0)
            call WRTSTR(page,xpos,ypos,stptbl,'Step Size =     % ')
            call LEVEL(1)
        else
            call LEVEL(0)
            call BLKFIL(xpos,(ypos+chh/4),len,(5*chh/4))
            call LEVEL(1)
        endif
C Typing out headings indicating the number of the last logged sample
C   (for manual and automatic modes)
        xpos = (720 - stptbl*chw)/2
        ypos = bltop + blh - 2.5*chh
        len = stptbl*chw
        if (log) then
            call BLKFIL(xpos,(ypos+chh/4),len,(5*chh/4))
            call LEVEL(0)
            call WRTSTR(page,xpos,ypos,stptbl,'Logging No:       ')
            call LEVEL(1)
        else
            call LEVEL(0)
            call BLKFIL(xpos,(ypos+chh/4),len,(5*chh/4))
            call LEVEL(1)
        endif

        chscr = .false.

        return
        end
C ***************************************************************************
CH      REVISION HISTORY  :
C       VERSION       BY                 DATE        COMMENT
C         1.0        adw                26-05-89    finally commented
C       FILEND          :
C ***************************************************************************
C
C       FILE              : UPDSCR.FOR
C ***************************************************************************
CN      MODULE NAME       : updscr
CA      FUNCTION          : Update Values of Inp/Outp/Setp on Run Block Diag
CS      CALL SEQUENCE     : call updscr()
CI      INPUT PARAMETERS  : None
CO      OUTPUT PARAMETERS : None
CG      GLOBAL VARIABLES  : Include files:  iovar.inc,heblock.inc,runvar.inc
C                           Common blocks:  /scrcon/ gap,blh,bltop,stptl,stptbl
C                                           /sampl/ sampl
CM      MODULES CALLED    : Fortran  :  prti4
C                           Assembler:  LEVEL,BLKFIL
CE      ERROR CONDITIONS  : Hercules graphics card not installed correctly
CC      COMMENTS          : This routine updates the values of rig inputs,
C                           outputs and setpoints on the run screen as required
C                           for the different operating modes.  It also updates
C                           the values of the step size when it is changed.
C ***************************************************************************
        subroutine updscr()
        implicit integer*2 (i)
$include: 'iovar.inc'
$include: 'heblock.inc'
$include: 'runvar.inc'
        integer*2 mode,page,sampl
        integer*2 gap,blh,bltop,stptl,stptbl
        integer*2 iset,iinp,iout
        common /scrcon/ gap,blh,bltop,stptl,stptbl
        common /sampl/ sampl
        page = 0

C Manual Mode -
C Printing out: output values for F1 to F2, L1 to L4, T1 to T10
C             : input values for C1 to C6, S1 to S4
        if ((mode.le.4).and.(.not.quit)) then
            do 900 i = 1, 16
                if (i.le.10) iinp = iperc(input(i))
C Outputs:
                if (i.le.8) then
                    iout = idegrc(output(i))
                    call prti2(page,posts(1,i),
     +                  (posts(2,i)+chh),4,'(I2)',2,iout)
                endif
                if ((i.gt.8).and.(i.le.10)) then
                    iout = iperc(output(i))
                    call prti2(page,posfs(1,(i-8)),
     +                  (posfs(2,(i-8))+chh),4,'(I3)',3,iout)
                endif
                if ((i.gt.10).and.(i.le.14)) then
                    iout = iperc(output(i))
                    call prti2(page,posls(1,(i-10)),
     +                  (posls(2,(i-10))+chh),4,'(I3)',3,iout)
                endif
                if (i.gt.14) then
                    iout = idegrc(output(i))
                    call prti2(page,posts(1,(i-6)),
     +                  (posts(2,(i-6))+chh),4,'(I2)',2,iout)
                endif
C Inputs:
                if (i.le.6) then
                    call prti2(page,poscv(1,i),
     +                  (poscv(2,i)+chh),4,'(I3)',3,iinp)
                endif
                if ((i.gt.6).and.(i.le.10)) then
                    call prti2(page,posst(1,(i-6)),
     +                  (posst(2,(i-6))+chh),4,'(I3)',3,iinp)
                endif
```

```
900       continue
      endif

C Automatic mode -
C Printing out: setpoint values for F1 to F2, L1 to L4
C             : output values for F1 to F2, L1 to L4, T1 to T10
C             : input values for C1 to C6, S1 to S4
      if ((mode.gt.4).and.(.not.quit)) then
          do 800 i = 1, 16
C Setpoints:
              if (i.le.4) then
                  iset = iperc(setp(i))
                  call prti2(page,posls(1,i),
     +                (posls(2,i)+chh),4,'(I3)',3,iset)
              endif
              if ((i.gt.4).and.(i.le.6)) then
                  iset = iperc(setp(i))
                  call prti2(page,posfs(1,(i-4)),
     +                (posfs(2,(i-4))+chh),4,'(I3)',3,iset)
              endif
C Outputs:
              if (i.le.8) then
                  iout = idegrc(output(i))
                  if ((i.eq.1).or.(i.eq.5).or.
     +                (i.eq.8).or.(i.eq.3)) then
                      call prti2(page,posts(1,i),
     +                 (posts(2,i)+(2*chh)),4,'(I2)',2,iout)
                  else
                      call prti2(page,posts(1,i),
     +                 (posts(2,i)+chh),4,'(I2)',2,iout)
                  endif
              endif
              if ((i.gt.8).and.(i.le.10)) then
                  iout = iperc(output(i))
                  call prti2(page,posfs(1,(i-8)),
     +                (posfs(2,(i-8))+(2*chh)),4,'(I3)',3,iout)
              endif
              if ((i.gt.10).and.(i.le.14)) then
                  iout = iperc(output(i))
                  call prti2(page,posls(1,(i-10)),
     +                (posls(2,(i-10))+(2*chh)),4,'(I3)',3,iout)
              endif
              if (i.gt.14) then
                  iout = idegrc(output(i))
                  call prti2(page,posts(1,(i-6)),
     +                (posts(2,(i-6))+chh),4,'(I2)',2,iout)
              endif
C Inputs:
              if (i.le.10) iinp = iperc(input(i))
              if (i.le.6)
     +            call prti2(page,poscv(1,i),
     +                (poscv(2,i)+chh),4,'(I3)',3,iinp)
              if ((i.gt.6).and.(i.le.10))
     +            call prti2(page,posst(1,(i-6)),
     +                (posst(2,(i-6))+chh),4,'(I3)',3,iinp)

800       continue

C Printing out temperature setpoint values for
C sensors T1, T3, T5 and T8 stored as setp(7) to setp(10)
C First T1 and T5 (setp(7) and setp(8))
          j = 7
          do 700, i = 1, 5, 4
              iset = idegrc(setp(j))
              call prti2(page,posts(1,i),(posts(2,i)+chh),
     +                   4,'(I2)',2,iset)
          j = j + 1
700       continue
C Then T8 and T3 (setp(9) and setp(10))
          iset = idegrc(setp(9))
          call prti2(page,posts(1,8),(posts(2,8)+chh),
     +                   4,'(I2)',2,iset)
          iset = idegrc(setp(10))
          call prti2(page,posts(1,3),(posts(2,3)+chh),
     +                   4,'(I2)',2,iset)
      endif

C Printing Step Size for step test modes
      if ((mode.eq.4).or.(mode.eq.8)) then
          xpos = (720 - stptbl*chw)/2
          xpos = xpos + (stptl*chw) + chw
          ypos = bltop + blh - chh
          call LEVEL(0)
          call prti2(page,xpos,ypos,4,'(I4)',4,iperc(steplv))
          call LEVEL(1)
      endif
C Typing out value or last logged sample
C   (for manual and automatic modes)
      if (log) then
          xpos = (720 - stptbl*chw)/2
          xpos = xpos + (stptl*chw) + chw
          ypos = bltop + blh - 2.5*chh
          call LEVEL(0)
          if (sampl.ne.0) call prti2(page,xpos,ypos,4,'(I4)',4,sampl)
          call LEVEL(1)
      endif
      return
      end

C ***************************************************************************
CH    REVISION HISTORY  :
C     VERSION       BY                  DATE        COMMENT
C       1.0     A. de Waal            26-05-89    Commented
C       1.1     A. de Waal            06/01/90    Finally Commented
C     FILEND            :
C ***************************************************************************
C
C     FILE              : USPLOT.FOR
C ***************************************************************************
CN    MODULE NAME       : usplot
CA    FUNCTION          : Drive User Plot Option Menu
CS    CALL SEQUENCE     : call usplot()
CI    INPUT PARAMETERS  : None
CO    OUTPUT PARAMETERS : None
CG    GLOBAL VARIABLES  : Include files:  logval.inc
C                         Common blocks:  /scalop/ scalop,forsel,forno
```

```
CM    MODULES CALLED    : Fortran  :  (this is so damn boring!) wrhead,selpl,
C                                      formpl,onepl
C                         Assembler:  WIPSCR,DISP,DOMENU,ERTONE
CE    ERROR CONDITIONS  : None
CC    COMMENTS          : Initializes menu screen and horizontal plot axis and
C                         drives menu for the user to choose whether he/she
C                         wishes to:
C                             - Select inputs/outputs/setpoints for plotting
C                             - Edit plot axes and comments
C                             - Plot presently selected data on presently
C                                 selected axes with presently selected
C                                 comments (sorry - Merry Christmas!)
C ***************************************************************************
      subroutine usplot()
      implicit integer*2 (D)
$include: 'logval.inc'
      integer*2 page,opt131
      integer*2 scalop(2,2,3),forsel,nosel
      integer*2 tmax
      common /scalop/ scalop,forsel,forno

      scalop(1,1,3) = (sampl-1)/hertz
100   continue
      page = 0
      call WIPSCR(page)
      call DISP(page)
      call wrhead(0,((720-9*34)/2),20,34,
     +            'Plotting responses chosen by user.')
      opt131 = DOMENU(131)
      if (opt131.eq.1) then
          call selpl()
          goto 100
      elseif (opt131.eq.2) then
          call formpl()
          goto 100
      elseif (opt131.eq.3) then
          call onepl()
          goto 100
      endif
      if (opt131.ne.0) then
          call ERTONE()
          goto 100
      endif
      return
      end
C ***************************************************************************
CH    REVISION HISTORY  :
C     VERSION       BY                 DATE        COMMENT
C       1.1     A. de Waal           06/01/90   Finally Commented
C     FILEND          :
C ***************************************************************************
```

N2) HERIG Data Manipulation Subroutines

The HERIG package utilizes the same data load/save/editing routines listed in section M2 for the OPTCAD package.

N3) Utility Routines for Analog Interface Cards

A table of the analog interfacing utility subroutine names follows on the next page. The name of each subroutine, together with the file in which it is located and the page in the appendix on which it is listed, is given in the table.

Utility Routines for Analog Interface Cards			
Name	**File**	**Description**	**Page**
dainit	DAINIT.FOR	Resets and Initializes DT2815 DAC Card	521
daout	DAOUT.FOR	Output Data to Specified Channel and DAC	521
datest	DATEST.FOR	DT2815 Calibration and Testing Program	521
readop	READOP.FOR	Read Outputs from DT2801 ADC Card	523
setadp	SETADP.FOR	Set DT2801 ADC Card Initialization Parameters	523
setdap	SETDAP.FOR	Sets DT2815 DAC Card Initialization Parameters	524
wrinp	WRINP.FOR	Write Inputs to DT2815 DAC Card	524

```
C
C         FILE                  : DAINIT.FOR
C ***************************************************************************
CN     MODULE NAME           : dainit
CA     FUNCTION              : Resets and Initializes DT2815 DAC Card
CS     CALL SEQUENCE         : call dainit(dacno,daerr)
CI     INPUT PARAMETERS      : dacno:  integer - DAC specifier
CO     OUTPUT PARAMETERS     : daerr:  integer - Set not equal to 4 if error occurs
C                                                in initialization procedure
CG     GLOBAL VARIABLES      : None
CM     MODULES CALLED        : Assembler:  RSTDA(),RSTDAI(),
C                                          OUTP(davaddr,davinidat)
CE     ERROR CONDITIONS      : daerr =  04 - no errors, ready to receive
C                                           initialization byte
C                              daerr <> 04 - error, not ready for byte
CC     COMMENTS              : Initializes indicated DAC card by
C                                  resetting card indicated (error flag defined)
C                                  outputting initialization byte to appropriate
C                                      address in I/O map
C ***************************************************************************
       subroutine dainit(dacno,daerr)
       implicit integer*2 (R,O)
       integer*2 daerr,addrv,davinidat,addri,daiinidat,dacno
       addrv = 548
       inidatv = 11
       addri = 550
       inidati = 13
       daerr = 0
10     continue
       if (dacno.eq.1) then
         daerr = RSTDA()
         write(*,*)' Reset DAC#1 daerr should = 4 - ready for init'
       else
         daerr = RSTDAI()
         write(*,*)' Reset DAC#2 daerr should = 4 - ready for init'
       endif
       write(*,*)' In dainit subr, daerr =',daerr
       if (daerr.eq.04) then
           if (dacno.eq.1) then
             call OUTP(addrv,inidatv)
           else
             call OUTP(addri,inidati)
           endif
       endif
       return
       end
C ***************************************************************************
CH     REVISION HISTORY  :
C      VERSION       BY                  DATE         COMMENT
C        1.0     A. de Waal            11-04-89     Creation
C        1.1     A. de Waal            06/01/90   Finally Commented
C      FILEND            :
C ***************************************************************************
C
C         FILE NAME             : DAOUT.FOR
C ***************************************************************************
CN     MODULE NAME           : daout
CA     FUNCTION              : Output Data to Specified Channel and DAC
CS     CALL SEQUENCE         : call daout(dacno,channel,value,daerr)
CI     INPUT PARAMETERS      : dacno    : integer - DAC Specifier
C                              channel  : integer - Channel Specifier
C                              value    : integer - Value between 0 and 4096
C                                                   to be output
CO     OUTPUT PARAMETERS     : daerr    : integer - Error flag set to nonzero if no
C                                                   errors occur in writing
CG     GLOBAL VARIABLES      : None
CM     MODULES CALLED        : Assembler:  OUTDA(channel,value),
C                                          OUTDAI(channel,value)
CE     ERROR CONDITIONS      : daerr = 0 - No errors in write
C                                    = 1 - Error in writing low data byte
C                                    = 2 - Error in writing high data byte
CC     COMMENTS              : To output data to a specified channel on one of the
C                              DT2815 D/A cards.  Cards have to have been
C                              initialized using dainit(..) subroutine.  Data either
C                              output to specified channel on voltage sourcing DAC
C                              using routine OUTDA(..), or to current sourcing DAC
C                              using routine OUTDAI(..).  Error flag set if problems
C                              occur in writing
C ***************************************************************************
       subroutine daout(dacno,channel,value,daerr)
       implicit integer*2 (O)
       integer*2 dacno,channel,value,daerr
       if (dacno.eq.1) then
           daerr = OUTDA(channel,value)
       else
           daerr = OUTDAI(channel,value)
       endif
       return
       end
C ***************************************************************************
CH     REVISION HISTORY  :
C      VERSION       BY                  DATE         COMMENT
C        1.0     A. de Waal            11-04-89     Creation
C        1.1     A. de Waal            06/01/90   Finally Commented
C      FILEND            :
C ***************************************************************************

C
C      FILE                  : DATEST.FOR
C ***************************************************************************
CN     MODULE NAME           : datest
CA     FUNCTION              : DT2815 Calibration and Testing Program
CS     CALL SEQUENCE         : Main Program
CI     INPUT PARAMETERS      : None
CO     OUTPUT PARAMETERS     : None
CG     GLOBAL VARIABLES      : None
CM     MODULES CALLED        : Fortran  :  dainit(dacno,daerr),
C                                          daout(dacno,channel,value,daerr)
CE     ERROR CONDITIONS      : DT2815 card errors
CC     COMMENTS              : Initializes card and then writes indicated values to
C                              cards for calibration.
C ***************************************************************************
       program datest
       integer*2 daerr,chno
       character ans
C
```

```
C   Resetting voltage output card
C
10    continue
      call dainit(1,daerr)
      if (daerr.eq.04) then
          write(*,*)' Initialized DAC#1'
          write(*,*)' Check that all outputs equal 0 volts'
          pause
      else
          write(*,*)' Card was not ready for initialization byte'
          pause
          goto 10
      endif
C
C   Resetting current output card
C
20    continue
      call dainit(2,daerr)
      if (daerr.eq.04) then
          write(*,*)' Initialized DAC#2'
          write(*,*)' Check that all outputs equal 4 milliamperes'
          pause
      else
          write(*,*)' Card was not ready for initialization byte'
          pause
          goto 20
      endif
C
C   Outputting minimum voltages to DT2815 card
C
30    continue
      do 40 chno = 0,3
          call daout(1,chno,0,daerr)
          if (daerr.eq.0) then
              write(*,*)' Outputting 0 units to channel',chno
              write(*,*)' Check that 0 volts appears on termination
     +                    panel'
              if (chno.eq.0) then
                  write(*,*)' Adjust pot until 0 volts appears'
                  pause
              endif
              pause
          elseif (daerr.eq.1) then
              write(*,*)' Error ocurred in writing low byte'
              goto 10
          elseif (daerr.eq.2) then
              write(*,*)' Error ocurred in writing high byte'
              goto 10
          endif
40    continue
C
C   Outputting minimum currents to DT2815 card
C
50    continue
      do 60 chno = 0,5
          call daout(2,chno,0,daerr)
          if (daerr.eq.0) then
              write(*,*)' Outputting 0 units to channel',chno
              write(*,*)' Check that 4 mA appears on termination panel'
              if (chno.eq.0) then
                  write(*,*)' Adjust pot until 4 mA appears'
                  pause
              endif
              pause
          elseif (daerr.eq.1) then
              write(*,*)' Error ocurred in writing low byte'
              goto 20
          elseif (daerr.eq.2) then
              write(*,*)' Error ocurred in writing high byte'
              goto 20
          endif
60    continue
C
C   Outputting maximum voltages to DT2815 card
C
70    continue
      do 80 chno = 0,3
          call daout(1,chno,4095,daerr)
          if (daerr.eq.0) then
              write(*,*)' Outputting 4095 units to channel',chno
              write(*,*)' Check that 5 volts appears on termination
     +                    panel'
              if (chno.eq.0) then
                  write(*,*)' Adjust pot until 5 volts appears'
                  pause
              endif
              pause
          elseif (daerr.eq.1) then
              write(*,*)' Error ocurred in writing low byte'
              goto 10
          elseif (daerr.eq.2) then
              write(*,*)' Error ocurred in writing high byte'
              goto 10
          endif
80    continue
C
C   Outputting maximum currents to DT2815 card
C
90    continue
      do 100 chno = 0,5
          call daout(2,chno,4095,daerr)
          if (daerr.eq.0) then
              write(*,*)' Outputting 4095 units to channel',chno
              write(*,*)' Check that 20 mA appears on termination panel'
              if (chno.eq.0) then
                  write(*,*)' Adjust pot until 20 mA appears'
                  pause
              endif
              pause
          elseif (daerr.eq.1) then
              write(*,*)' Error ocurred in writing low byte'
              goto 20
          elseif (daerr.eq.2) then
              write(*,*)' Error ocurred in writing high byte'
              goto 20
          endif
```

```
100   continue
      write(*,*)' Do you wish to repeat the test? (Y/N)'
      read(*,'(A1)')ans
      if (ans.eq.'Y') ans = 'y'
      if (ans.eq.'y') goto 30
      stop
      end
C ***************************************************************************
CH    REVISION HISTORY  :
C     VERSION       BY                  DATE        COMMENT
C       1.0     A. de Waal            11-04-89     Creation
C       1.1     A. de Waal            06/01/90   Finally Commented
C     FILEND            :
C ***************************************************************************

C
C     FILE              : READOP.FOR
C ***************************************************************************
CN    MODULE NAME       : readop
CA    FUNCTION          : Read Outputs from DT2801 ADC Card
CS    CALL SEQUENCE     : call readop()
CI    INPUT PARAMETERS  : None
CO    OUTPUT PARAMETERS : None
CG    GLOBAL VARIABLES  : Include files:  adpar.inc,iovar.inc
CM    MODULES CALLED    : Assembler:  WAITDT,OUTP,INP
CE    ERROR CONDITIONS  : Card error
CC    COMMENTS          : Instructs DT2801 card to read outputs by writing to
C                         the command register, waits for card to be ready, and
C                         inputs output values from card.
C ***************************************************************************
      subroutine readop()
      implicit integer*2 (I,O,f)
$include: 'adpar.inc'
$include: 'iovar.inc'
      integer*2 outpno,i,j,iout

      outpno = 1
      do 99 i = 1,2
          call WAITDT(stareg(i),writwt,0)
          call WAITDT(stareg(i),waitcm,1)
          call OUTP(comreg(i),crdrd)
          do 88 j = 1,8
              call WAITDT(stareg(i),readwt,1)
              iout = (int (INP(datreg(i))))
              call WAITDT(stareg(i),readwt,1)
              iout = (int(INP(datreg(i))*256)) + iout
              if ((outpno.le.10).or.(outpno.ge.15)) then
                  output(outpno) = fixout(iout)
              else
                  output(outpno) = iout
              endif
              outpno = outpno + 1
88        continue
99    continue
      return
      end
C ***************************************************************************
CH    REVISION HISTORY  :
C     VERSION       BY                  DATE        COMMENT
C       1.1     A. de Waal            06/01/90   Finally Commented
C     FILEND            :
C ***************************************************************************
C
C     FILE              : SETADP.FOR
C ***************************************************************************
CN    MODULE NAME       : setadp
CA    FUNCTION          : Set DT2801 ADC Card Initialization Parameters
CS    CALL SEQUENCE     : call setadp()
CI    INPUT PARAMETERS  : None
CO    OUTPUT PARAMETERS : None
CG    GLOBAL VARIABLES  : Include files:  adpar.inc
CM    MODULES CALLED    : Assembler:  OUTP,WAITDT,CONVRT
CE    ERROR CONDITIONS  : Card error
CC    COMMENTS          : Defines ADC card parameters (base addresses, wait
C                         cycle times and code words) for both of the DT2801
C                         cards in use.  Then initializes both cards by writing
C                         to the data and command register addresses and
C                         polling the status register at the appropriate
C                         addresses in the I/O map, waiting for the card
C                         to be ready before the next byte of information
C                         is output to the card.
C ***************************************************************************
      subroutine setadp()
      implicit integer*2 (I,O,R)
$include: 'adpar.inc'
      integer*4 temper,sams
      integer*2 basadr(2)
      integer*2 i,j,err,temp,clklo,clkhi,samlo,samhi
      integer*2 crdclr,crdclk,crdpar,crdadi
      integer*2 crdstp,crderr

      waitcm = 4
      writwt = 2
      readwt = 5
      crdclr = 1
      crdclk = 3
      crdpar = 13
      crdrd  = 14
      crdadi = 12
      crdstp = 15
      crderr = 2

      basadr(1) = 748
      basadr(2) = 750
      do 88 i = 1,2
          comreg(i) = basadr(i) + 1
          stareg(i) = basadr(i) + 1
          datreg(i) = basadr(i)

          call OUTP(comreg(i),crdstp)
          temp = INP(datreg(i))
          call WAITDT(stareg(i),writwt,0)
          call WAITDT(stareg(i),waitcm,1)
          call OUTP(comreg(i),crdclr)
          call WAITDT(stareg(i),writwt,0)
          call WAITDT(stareg(i),waitcm,1)
```

```
          call OUTP(comreg(i),crdclk)
          call WAITDT(stareg(i),writwt,0)
          temper = (int (1.0/(1600.0*1.5*0.0000025))) + 1
          call CONVRT(clklo,clkhi,temper)
          call OUTP(datreg(i),clklo)
          call WAITDT(stareg(i),writwt,0)
          call OUTP(datreg(i),clkhi)
          call WAITDT(stareg(i),writwt,0)
          call WAITDT(stareg(i),waitcm,1)
          call OUTP(comreg(i),crdpar)
          call WAITDT(stareg(i),writwt,0)
          call OUTP(datreg(i),0)
          call WAITDT(stareg(i),writwt,0)
          call OUTP(datreg(i),0)
          call WAITDT(stareg(i),writwt,0)
          call OUTP(datreg(i),7)
          call WAITDT(stareg(i),writwt,0)
          sams = 8
          call CONVRT(samlo,samhi,sams)
          call OUTP(datreg(i),samlo)
          call WAITDT(stareg(i),writwt,0)
          call OUTP(datreg(i),samhi)
88    continue
      return
      end

C ***************************************************************************
CH    REVISION HISTORY  :
C     VERSION       BY                  DATE        COMMENT
C       1.1     A. de Waal            06/01/90   Finally Commented
C     FILEND            :
C ***************************************************************************
C
C        FILE                 : SETDAP.FOR
C ***************************************************************************
CN    MODULE NAME       : setdap
CA    FUNCTION          : Sets DT2815 DAC Card Initialization Parameters
CS    CALL SEQUENCE     : call dainit(dacno,daerr)
CI    INPUT PARAMETERS  : dacno: integer - DAC Number (not used)
CO    OUTPUT PARAMETERS : daerr: integer - Set to 04 if no errors occurred
C                                          in initialization of card
CG    GLOBAL VARIABLES  : None
CM    MODULES CALLED    : Assembler:  RSTDA,RSTDAI,OUTP
CE    ERROR CONDITIONS  : daerr =  04 - no errors, ready to receive
C                                    initialization byte
C                         daerr <> 04 - error, not ready for byte
CC    COMMENTS          : Resets both DT2815 DAC cards, and then writes
C                         initialization bytes to the appropriate locations
C                         of the cards in the I/O space.
C ***************************************************************************
      subroutine setdap()
      implicit integer*2 (R,O)
      integer*2 daerr,addrv,inidatv,addri,inidati,dacno
      addrv = 548
      inidatv = 11
      addri = 550
      inidati = 13
      daerr = 0
10    continue
      daerr = RSTDA()
      daerr = RSTDAI()
      if (daerr.eq.04) then
          call OUTP(addrv,inidatv)
          call OUTP(addri,inidati)
      endif
      return
      end
C ***************************************************************************
CH    REVISION HISTORY  :
C     VERSION       BY                  DATE        COMMENT
C       1.0     A. de Waal            11-04-89     Creation
C       1.1     A. de Waal            06/01/90   Finally Commented
C     FILEND            :
C ***************************************************************************
C
C     FILE              : WRINP.FOR
C ***************************************************************************
CN    MODULE NAME       : wrinp
CA    FUNCTION          : Write Inputs to DT2815 DAC Card
CS    CALL SEQUENCE     : call wrinp()
CI    INPUT PARAMETERS  : None
CO    OUTPUT PARAMETERS : None
CG    GLOBAL VARIABLES  : None
CM    MODULES CALLED    : Assembler:  OUTDA(channel,value),
C                                     OUTDAI(channel,value)
CE    ERROR CONDITIONS  : daerr = 0 - No errors in write
C                               = 1 - Error in writing low data byte
C                               = 2 - Error in writing high data byte
C                         Should pass daerr value back to calling procedure
C                         Naughty, naughty!
CC    COMMENTS          : To output data to a specified channel on one of the
C                         DT2815 D/A cards using assembler routines OUTDA and
C                         OUTDAI.
C ***************************************************************************
      subroutine wrinp()
      implicit integer*2 (O)
$include: 'iovar.inc'
      integer*2 i,inpno,daerr
      do 900 inpno = 1, 10
          if (inpno.le.6) then
              daerr = OUTDAI((inpno-1),input(inpno))
          else
              daerr = OUTDA((inpno-7),input(inpno))
          endif
900   continue
      return
      end
C ***************************************************************************
CH    REVISION HISTORY  :
C     VERSION       BY                  DATE        COMMENT
C       1.0     A. de Waal            11-04-89     Creation
C       1.1     A. de Waal            06/01/90   Finally Commented
C     FILEND            :
C ***************************************************************************
```

N4) Assembler Modules: Menu Definition and Modified Fisher Routines

A table of the names of assembler routines for the menu definition of the HERIG package (MTEXT.ASM), and modified assembler routines written by Ian Fisher follows on the next page. The name of each subroutine and the page in the appendix on which it is listed, is given in the table. The modified routines of Fisher are all contained in the file ADUTIL.ASM, and the menu definition routine is found in the file MTEXT.ASM.

```
File: ADUTIL.ASM

; MODULE : ADIN
; *************

TITLE To read in data from a DT2801 A/D/A card.

DGROUP  GROUP   CONST, _BSS, _DATA
        ASSUME  DS: DGROUP
        ASSUME  SS: DGROUP
        ASSUME  ES: DGROUP

;-------------------------------------------------------------------------
;                   EXTERNAL DECLARATIONS.                              :
;-------------------------------------------------------------------------

;-----------   Subroutines.
        PUBLIC  RSTDA
        PUBLIC  RSTDAI
        PUBLIC  OUTDA
        PUBLIC  OUTDAI
        PUBLIC  WAITDT
        PUBLIC  INP
        PUBLIC  OUTP
        PUBLIC  DELAY
        PUBLIC  CONVRT
;-------------------------------------------------------------------------
;                       EQUATES                                         :
;-------------------------------------------------------------------------

;-----------   Card addresses.
BASE_ADR EQU    02ECH   ; Card base address.
COMMAND_REG     EQU     BASE_ADR+1      ; Command register address.
STATUS_REG      EQU     BASE_ADR+1      ; Status register address.
DATA_REG EQU    BASE_ADR ; Data register address.

;-----------   Card commands.
WAIT_CMD EQU    04H     ; Command wait.
WRITE_WAIT      EQU     02H     ; Write wait.
READ_WAIT       EQU     05H     ; Read wait.
CARD_CLEAR      EQU     01H     ; Clear the card.
CARD_CLOCK      EQU     03H     ;
CARD_PARAM      EQU     0DH     ;
CARD_READ       EQU     0EH     ;
CARD_ADIN       EQU     0CH     ; Read AD in.
CARD_STOP       EQU     0FH     ; Stop the card.
CARD_ERROR      EQU     02H     ;
ADIN_GAIN       EQU     00H     ; Zero gain on the card.
CHANO   EQU     00H     ; Channel 0.

;-------------------------------------------------------------------------
_BSS    SEGMENT WORD PUBLIC 'BSS'

_BSS    ENDS

CONST   SEGMENT WORD PUBLIC 'CONST'

CONST   ENDS

_DATA   SEGMENT WORD PUBLIC  'DATA'

_DATA   ENDS


_TEXT   SEGMENT BYTE PUBLIC 'CODE'
        ASSUME  CS: _TEXT
        ASSUME  DS: DGROUP
        ASSUME  SS: DGROUP
        ASSUME  ES: DGROUP

;-------------------------------------------------------------------------
;
; Function: RSTDA
; Inputs:       None.
; Outputs:      To D/A channels.
;           AX - status of reset.
; Calls: WAITCRD
; Destroys: Flags.
;
; Description: Resets the voltage D/A card.
;
;-------------------------------------------------------------------------
RSTDA   PROC FAR

        PUSH    BX              ;
        PUSH    CX              ;
        PUSH    DX              ;
        MOV     BX,0            ;
NEXT_RESET:                             ;
        CMP     BX,5            ;
        JE      RESET_FAIL              ;
        MOV     CX,0            ;
        MOV     DX,0225H        ;
        MOV     AL,0            ;
        OUT     DX,AL           ;
        CALL    FAR PTR DELAY   ;
NEXT_CHK:       CMP     CX,5000         ;
        JE      NEXT_RESET              ;
        INC     CX              ;
        MOV     DX,0225H        ;
        IN      AL,DX           ;
        CMP     AL,04           ;
        JE      DA_RESET        ;
        JMP     NEXT_CHK        ;
RESET_FAIL:     XOR     AL,AL           ;
DA_RESET:       XOR     AH,AH           ;
        POP     DX              ;
        POP     CX              ;
        POP     BX              ;
        RET
RSTDA   ENDP
;-------------------------------------------------------------------------
;
; Function: RSTDAI
; Inputs:       None.
; Outputs:      To D/A channels.
```

```
;           AX - status of reset.
; Calls: WAITCRD
; Destroys: Flags.
;
; Description: Resets the current (milliamperes) D/A card.
;
;-------------------------------------------------------------------------------
RSTDAI   PROC FAR

         PUSH    BX                ;
         PUSH    CX                ;
         PUSH    DX                ;
         MOV     BX,0              ;
NEXTRESET:                                 ;
         CMP     BX,5              ;
         JE      RESETFAIL                 ;
         MOV     CX,0              ;
         MOV     DX,0225H          ;
         MOV     AL,0              ;
         OUT     DX,AL             ;
         CALL    FAR PTR DELAY     ;
NEXTCHK: CMP     CX,5000           ;
         JE      NEXTRESET                 ;
         INC     CX                ;
         MOV     DX,0226H          ;
         IN      AL,DX             ;
         CMP     AL,04             ;
         JE      DARESET           ;
         JMP     NEXTCHK           ;
RESETFAIL:       XOR     AL,AL              ;
DARESET: XOR     AH,AH             ;
         POP     DX                ;
         POP     CX                ;
         POP     BX                ;
         RET
RSTDAI   ENDP
;-------------------------------------------------------------------------------
;
; Function: Output a value to voltage D/A channel
; Inputs:        err = OUTDA(channel,value)
; Outputs:       err = 0 - no errors
;                 1 - low byte error
;                 2 - high byte error
; Calls:
; Destroys: Flags.
;
; Description:  Outputs a Value to the Voltage DAC
;
;-------------------------------------------------------------------------------
OUTDA    PROC    FAR
         PUSH    BP                ;
         PUSH    ES                ;
         PUSH    BX                ;
         PUSH    CX                ;
         PUSH    DX                ;
         MOV     BP,SP             ;
         LES     BX,DWORD PTR [BP+14]      ;
         MOV     AX,WORD PTR ES:[BX]       ;
         MOV     CL,4              ;
         SHL     AX,CL             ;
         LES     BX,DWORD PTR [BP+18]      ;
         MOV     DX,WORD PTR ES:[BX]       ;
         SHL     DX,1              ;
         AND     DX,000EH          ;
         OR      DX,0001H          ;
         ADD     AX,DX             ;
         MOV     BX,AX             ;
         MOV     DX,0225H          ;
                                   ;
         MOV     CX,5000           ;
CHKDA1:                            ;
         DEC     CX                ;
         CMP     CX,0              ;
         JG      NOT_TIMEOUT1      ;
         MOV     AX,1              ;
         JMP     EXIT_OUTDA                ;
NOT_TIMEOUT1:                      ;
         IN      AL,DX             ;
         CMP     AL,0              ;
         JNE     CHKDA1            ;

         MOV     DX,0224H          ;
         MOV     AL,BL             ;
         CALL    FAR PTR DELAY     ;
         OUT     DX,AL             ;

         MOV     DX,0225H          ;
         MOV     CX,5000           ;
CHKDA2:                            ;
         DEC     CX                ;
         CMP     CX,0              ;
         JG      NOT_TIMEOUT2      ;
         MOV     AX,2              ;
         JMP     EXIT_OUTDA                ;
NOT_TIMEOUT2:                      ;
         IN      AL,DX             ;
         CMP     AL,010H           ;
         JNE     CHKDA2            ;
         MOV     DX,0224H          ;
         MOV     AL,BH             ;
         CALL    FAR PTR DELAY     ;
         OUT     DX,AL             ;
         XOR     AX,AX             ;
EXIT_OUTDA:                                ;
         POP     DX                ;
         POP     CX                ;
         POP     BX                ;
         POP     ES                ;
         POP     BP                ;
         RET     08
OUTDA    ENDP

;-------------------------------------------------------------------------------
;
; Function: Output a value to current (milliamperes) D/A channel
; Inputs:        err = OUTDAI(channel,value)
```

```
; Outputs:       err = 0 - no errors
;                 1 - low byte error
;                 2 - high byte error
; Calls:
; Destroys: Flags.
;
; Description:  Outputs a Value to the Current DAC
;
;-------------------------------------------------------------------------------
OUTDAI   PROC     FAR
         PUSH     BP                ;
         PUSH     ES                ;
         PUSH     BX                ;
         PUSH     CX                ;
         PUSH     DX                ;
         MOV      BP,SP             ;
         LES      BX,DWORD PTR [BP+14]      ;
         MOV      AX,WORD PTR ES:[BX]       ;
         MOV      CL,4              ;
         SHL      AX,CL             ;
         LES      BX,DWORD PTR [BP+18]      ;
         MOV      DX,WORD PTR ES:[BX]       ;
         SHL      DX,1              ;
         AND      DX,000EH          ;
         OR       DX,0001H          ;
         ADD      AX,DX             ;
         MOV      BX,AX             ;
         MOV      DX,0227H          ;
                                    ;
         MOV      CX,5000           ;
CHKDA3:                             ;
         DEC      CX                ;
         CMP      CX,0              ;
         JG       NOT_TIMEOUT3      ;
         MOV      AX,1              ;
         JMP      EXIT_OUTDAI               ;
NOT_TIMEOUT3:                       ;
         IN       AL,DX             ;
         CMP      AL,0              ;
         JNE      CHKDA3            ;

         MOV      DX,0226H          ;
         MOV      AL,BL             ;
         CALL     FAR PTR DELAY     ;
         OUT      DX,AL             ;

         MOV      DX,0227H          ;
         MOV      CX,5000           ;
CHKDA4:                             ;
         DEC      CX                ;
         CMP      CX,0              ;
         JG       NOT_TIMEOUT4      ;
         MOV      AX,2              ;
         JMP      EXIT_OUTDAI               ;
NOT_TIMEOUT4:                       ;
         IN       AL,DX             ;
         CMP      AL,010H           ;
         JNE      CHKDA4            ;
         MOV      DX,0226H          ;
         MOV      AL,BH             ;
         CALL     FAR PTR DELAY     ;
         OUT      DX,AL             ;
         XOR      AX,AX             ;
EXIT_OUTDAI:                        ;
         POP      DX                ;
         POP      CX                ;
         POP      BX                ;
         POP      ES                ;
         POP      BP                ;
         RET      08
OUTDAI   ENDP
;-------------------------------------------------------------------------------
;
; Function: Given a set of conditions ,wait for the conditions to be
;           met.
; Inputs:        call WAITDT(address,mask,case)
;             address - address of port
;                mask - condition flag (ANDed with input)
;            function - Function: 00 - FLAG TO BE CLEAR
;                                 01 - FLAG TO BE SET.
; Outputs:       None.
; Calls: None.
; Destroys: Flags.
;
; Description:  Waits for Set of Conditions are Met
;
;-------------------------------------------------------------------------------

WAITDT   PROC     FAR

         PUSH     BP                ;
         PUSH     ES                ;
         PUSH     AX                ;
         PUSH     BX                ;
         PUSH     CX                ;
         PUSH     DX                ;
         MOV      BP,SP             ;
         LES      BX,DWORD PTR [BP+24]      ;
         MOV      DX,WORD PTR ES:[BX]       ;
         LES      BX,DWORD PTR [BP+20]      ;
         MOV      CX,WORD PTR ES:[BX]       ;
         LES      BX,DWORD PTR [BP+16]      ;
         MOV      BX,WORD PTR ES:[BX]       ;
         MOV      BH,BL             ;
         MOV      BL,CL             ;
WAITNEXT:                           ;
         IN       AL,DX             ;
         AND      AL,BL             ;
         CMP      BH,0              ;
         JE       CHK_CLR           ;
         CMP      AL,0              ;
         JE       WAITNEXT          ;
         JMP      EXITWAIT          ;
CHK_CLR: CMP      AL,0              ;
         JE       EXITWAIT          ;
         JMP      WAITNEXT          ;
```

```
EXITWAIT:                                ;
        POP     DX              ;
        POP     CX              ;
        POP     BX              ;
        POP     AX              ;
        POP     ES              ;
        POP     BP              ;
        RET     0CH             ;
WAITDT  ENDP

;-------------------------------------------------------------------------------
;
; Function: To input data from a port in the I/O map.
; Inputs:       data = INP(address)
;           data - (integer*2) Data from port.
;           adress - (integer*2) adress of port.
; Outputs:
; Calls: None.
; Destroys: Flags.
;
; Description:  Inputs Data from Address in I/O Space
;
;-------------------------------------------------------------------------------
INP     PROC    FAR
        PUSH    BP              ;
        PUSH    ES              ;
        PUSH    BX              ;
        PUSH    DX              ;
        MOV     BP,SP           ;
        LES     BX,DWORD PTR [BP+12]    ;
        MOV     DX,WORD PTR ES:[BX]     ;
        IN      AL,DX           ;
        XOR     AH,AH           ;
        POP     DX              ;
        POP     BX              ;
        POP     ES              ;
        POP     BP              ;
        RET     04              ;
INP     ENDP

;-------------------------------------------------------------------------------
;
; Function: To output a byte to a port in the I/O map.
; Inputs:       call OUTP(address,data)
;           address - (integer*2) Address of port.
;              data - (integer*2) Data : truncated to a byte.
; Outputs:
; Calls:    None.
; Destroys: Flags.
;
; Description:  Outputs Data to Address in I/O Space
;
;-------------------------------------------------------------------------------
OUTP    PROC    FAR
        PUSH    BP              ;
        PUSH    ES              ;
        PUSH    AX              ;
        PUSH    BX              ;
        PUSH    DX              ;
        MOV     BP,SP           ;
        LES     BX,DWORD PTR [BP+14]    ;
        MOV     AX,WORD PTR ES:[BX]     ;
        LES     BX,DWORD PTR [BP+18]    ;
        MOV     DX,WORD PTR ES:[BX]     ;
        OUT     DX,AL           ;
        POP     DX              ;
        POP     BX              ;
        POP     AX              ;
        POP     ES              ;
        POP     BP              ;
        RET     08H             ;
OUTP    ENDP

;-------------------------------------------------------------------------------
DELAY   PROC    FAR

        PUSH    AX              ;
        MOV     AX,02710H               ;
CHKDELAY:       DEC     AX              ;
        CMP     AX,0            ;
        JNE     CHKDELAY        ;
        POP     AX              ;
        RET                     ;
DELAY   ENDP
;-------------------------------------------------------------------------------

CONVRT  PROC    FAR

        PUSH    BP              ;
        PUSH    ES              ;
        PUSH    AX              ;
        PUSH    BX              ;
        PUSH    CX              ;
        MOV     BP,SP           ;
        LES     BX,DWORD PTR [BP+14]    ;
        MOV     CX,WORD PTR ES:[BX]     ;
        LES     BX,DWORD PTR [BP+18]    ;
        MOV     AL,CH           ;
        XOR     AH,AH           ;
        MOV     WORD PTR ES:[BX],AX     ;
        LES     BX,DWORD PTR [BP+22]    ;
        MOV     AL,CL           ;
        XOR     AH,AH           ;
        MOV     WORD PTR ES:[BX],AX     ;
        POP     CX              ;
        POP     BX              ;
        POP     AX              ;
        POP     ES              ;
        POP     BP              ;
        RET     0CH             ;
CONVRT  ENDP

_TEXT   ENDS                    ;
                                ;
        END                     ;
```

```
;-------------------------------------------------------------------------------
; MODULE : Menu macro routines.
;          *********************

TITLE Sets up the menu data segments.
;     *********************************

        PUBLIC   FHELP

; ******************************************************************************
; REVISION HISTORY :
; VERSION       BY                  DATE        COMMENT
;    1.00       Ian Fisher          23/06/88    Creation.
;
; ******************************************************************************


;File: MTEXT.ASM
;Name: mtext
;Function: Rig Running Program Menu Text
; Termination characters.
; ***********************
EOM     EQU     02      ; End of menu.
EOS     EQU     00      ; End of string.
EOT     EQU     04      ; End of table.
ESC     EQU     1BH     ; ESC character.
HLP     EQU     03      ; Start of help.

; Macro definitions.
; ******************
; Set up menu data area.
; **********************

STARTMENU       MACRO   MENU_NUM
        MENU_DATA       SEGMENT PUBLIC 'FAR_DATA'
MENU&MENU_NUM   LABEL BYTE
        MENU_DATA       ENDS
        TABLE_DATA      SEGMENT PUBLIC 'FAR_DATA'
        DW      MENU_NUM
        DW      MENU&MENU_NUM
        TABLE_DATA      ENDS
        MENU_DATA       SEGMENT PUBLIC 'FAR_DATA'
        ENDM

; End menu data area.
; *******************
ENDMENU MACRO
        MENU_DATA       ENDS
        ENDM

; ******************************************************************************
; Menu data areas.
; ****************

TABLE_DATA      SEGMENT WORD PUBLIC 'FAR_DATA'
TABLE_DATA      ENDS

MENU_DATA       SEGMENT WORD PUBLIC 'FAR_DATA'
MENU_DATA       ENDS

MENU_DATA       SEGMENT WORD PUBLIC 'FAR_DATA'
FHELP   DB      'LOCI.HLP       ',0
MENU_DATA       ENDS

; ******************************************************************************
; The structure of a menu is described :-
;
;     STARTMENU   number
;                 DB    'option1',HLP,'Description of option1',0
;                 DB    'option2',HLP,'Description of option2',0
;                 .
;                 .
;                 DB    'option(n-1)',HLP,'Description of option(n-1)',0
;                 DB    'option(n)',HLP,'Description of option(n)',0
;                 DB    EOM
;     ENDMENU
;
; STARTMENU and ENDMENU are macros defined above, 'number' is parameter of
; the macro : STARTMENU.
; HLP and EOM have been defined in the equates above.
;
; 'number' defines the menu to be used when calling DOMENU from FORTRAN.
;
; The 'number' can be any value except 0, this is used by the system to
; define the end of the menu data area and tables.
;
; For a description of how the menu is displayed, see the routine : DOMENU
; which resides in the module DOMENU.ASM.
;
; ******************************************************************************
;
; Menu definition.
; ****************
STARTMENU       1
        DB      'Param',HLP,'Load/Save/Modify rig parameters.',0
        DB      'Run',HLP,'Run heat exchanger rig.',0
        DB      'Heplot',HLP,'Plot heat exchanger test data.',0
        DB      'Quit',HLP,'Stop plant and exit from program.',0
        DB      EOM
ENDMENU

STARTMENU       11
        DB      'Load',HLP,'Load rig parameters.',0
        DB      'Save',HLP,'Save rig parameters.',0
        DB      'Modify',HLP,'Modify rig parameters.',0
        DB      EOM
ENDMENU

STARTMENU       111
        DB      'Inputs',HLP,'Load values for input settings.',0
        DB      'Setpoints',HLP,'Load values for setpoint settings.',0
        DB      'Test',HLP,'Load step test data for plotting.',0
        DB      'Frcon',HLP,'Load Frequency Domain Controller.',0
        DB      'LQGcon',HLP,'Load LQG Controller.',0
        DB      'Runpar',HLP,'Load rig running parameters.',0
```

```
        DB      EOM
ENDMENU

STARTMENU       112
        DB      'Inputs',HLP,'Save current input settings.',0
        DB      'Setpoints',HLP,'Save current setpoint settings.',0
        DB      'Data',HLP,'Save step test data.',0
        DB      'Frcon',HLP,'Save Frequency Domain Controller.',0
        DB      'LQGcon',HLP,'Save LQG Controller.',0
        DB      'Runpar',HLP,'Save rig running parameters.',0
        DB      EOM
ENDMENU

STARTMENU       113
        DB      'Runpar',HLP,'Edit rig running parameters.',0
        DB      'LQGcon',HLP,'Edit LQG controller.',0
        DB      'TrPrec',HLP,'Edit Steady-state Precompensator Gain Matrix.',0
        DB      EOM
ENDMENU

STARTMENU       1134
        DB      'Contr',HLP,'Edit Controller Gain Matrix.',0
        DB      'Obser',HLP,'Edit Observer Gain Matrix.',0
        DB      EOM
ENDMENU

STARTMENU       12
        DB      'Fcon',HLP,'Allow Rig to Run using Frequency Domain Control Only.',0
        DB      'F+LQGcon',HLP,'Allow Rig to Run using Frequency Domain and LQG Control.',0
        DB      EOM
ENDMENU

STARTMENU       122
        DB      'Format',HLP,'Edit Format of Real-Time Rig Data Plots.',0
        DB      'Scales',HLP,'Edit Scales of Real-Time Rig Data Plots.',0
        DB      'Perturb',HLP,'Edit Specifications of Perturbations to Process.',0
        DB      'Go',HLP,'Start Running Process.',0
        DB      'Param',HLP,'Load/Save/Edit Rig Parameters.',0
        DB      EOM
ENDMENU

STARTMENU       13
        DB      'ASet',HLP,'Plot responses of L1 To L4.',0
        DB      'BSet',HLP,'Plot responses of F1, F2, T10 and T6.',0
        DB      'CSet',HLP,'Plot responses of T5, T8, T3 and T1.',0
        DB      'DSet',HLP,'Plot responses of T7, T4, T9 and T2.',0
        DB      'User',HLP,'Plot responses chosen by user.',0
        DB      EOM
ENDMENU

STARTMENU       131
        DB      'Select',HLP,'Select logged data to be plotted.',0
        DB      'Format',HLP,'Choose plotting format.',0
        DB      'Plot',HLP,'Plot selected data.',0
        DB      EOM
ENDMENU

STARTMENU       141
        DB      'Gaussian',HLP,'Gaussian Perturbations.',0
        DB      'Sinusoid',HLP,'Sinusoidal Perturbations.',0
        DB      EOM
ENDMENU

STARTMENU       1411
        DB      'NoiseDist',HLP,'Specify Sinusoidal Noise/Disturbances.',0
        DB      'Setpoints',HLP,'Specify Sinusoidal Setpoint Perturbations.',0
        DB      EOM
ENDMENU

STARTMENU       0
ENDMENU

; ***************************************************************************
        END
```

N5) HERIG Include Files

A table of the names of the global variable groupings in the HERIG system include files follows on the next page. The name of each group of global variables is given in the table, together with the include file in which it is located and the page in the appendix on which it is listed.

adpar	ADPAR.INC	Specification Data for Analog Interface Cards	535
convar	CONVAR.INC	Stabilizing PI Controller Variables	535
defnam	DEFNAM.INC	Filename and pathnames.	535
grlab	GRLAB.INC	Group Response Data Plot Labels and Specs	535
heblock	HEBLOCK.INC	Heat Exchanger Block Diagram Dimensions/Specs	535
iovar	IOVAR.INC	Current Input/Output/Setpoint Values	535
keys	KEYS.INC	Keyboard constants.	535
logval	LOGVAL.INC	Arrays of Logged Response Data and Details	536
runvar	RUNVAR.INC	Rig Running Mode and Other Run Variables	536
screen	SCREEN.INC	Bloc Diagram Run Screen Titles/Specs	536
sysnms	SYSNMS.INC	System I/O names, matrix names and system titles.	536
syst	SYST.INC	State-space System Matrices	536
time1	TIME.INC	Part of the state variables for the time simulation.	536
time2	TIME.INC	Part of the state variables for the time simulation.	536
time3	TIME.INC	Arrays for time simulation axes settings.	537
time4	TIME.INC	Arrays for time simulation axes settings.	537

```
C
C     INCLUDE FILE : ADPAR.INC
C ************************************************************************
CN    COMMON NAME  : adpar
CA    DESCRIPTION  : Specification Data for Analog Interface Cards
CP    PARAMETERS   :
C
C ************************************************************************
      integer*2 comreg(2),stareg(2),datreg(2)
      integer*2 readwt,writwt,waitcm,crdrd
      common /advar/ comreg,stareg,datreg
      common /adcom/ readwt,writwt,waitcm,crdrd

C ************************************************************************
C
C     INCLUDE FILE : CONVAR.INC
C ************************************************************************
CN    COMMON NAME  : convar
CA    DESCRIPTION  : Stabilizing PI Controller Variables
CP    PARAMETERS   :
C
C ************************************************************************
      integer*2 pptr(10,10),gptr(10,10),error(10),dlyerr(10,10,100)
      integer*2 inpmap(10,10),outmap(10,16),setmap(10,10)
      real*4 hertz
      real*8 prop(10,10),intg(10,10)
      logical*2 conlod
      common /convar/ pptr,gptr,inpmap,outmap,setmap,
     + prop,intg,error,dlyerr,conlod
      common /sfrequ/ hertz
C ************************************************************************
C
C     INCLUDE FILE : DEFNAM.INC
C ************************************************************************
CN    COMMON NAME  : defnam
CA    DESCRIPTION  : Filename and pathnames.
CP    PARAMETERS   :
C
C           outpth - Save pathname.
C           inpath - Load pathname.
C           lqanam - Filename for Analysed LQG Design Information.
C           lqgnam - Filename for LQG Design Information (Not Analysed).
C           deanam - Filename for Analysed Non-LQG Design Information.
C           desnam - Filename for Non-LQG Design Information (Not Analysed).
C           pronam - Filename for Process Data.
C           qrnam  - Filename for Q, R, Q1 and R1 Matrix Information.
C           knam   - Filename for Kc and Kf Matrix Information.
C           defnam - Filename for Package Default Setting Information.
C ************************************************************************

      character*25 outpth,inpath
      character*5 defdir
      character*25 infnam,fname,mname,pernam
      character*50 usenam
      common /defnam/ outpth,inpath
      common /filnam/ defdir,infnam,fname,mname,pernam,usenam

C ************************************************************************
C
C     INCLUDE FILE : GRLAB.INC
C ************************************************************************
CN    COMMON NAME  : grlab
CA    DESCRIPTION  : Group Response Data Plot Labels and Specs
CP    PARAMETERS   :
C ************************************************************************
      integer*2 xlpos(4),ylpos(4),lablen,stepll,stinll,grbrfl
      character*39 label(2,4,4)
      character*60 stplab(2),grbrf(2),bl60
      character*2 stpins(2,10),bl2
      common /labcon/ xlpos,ylpos,lablen,stepll,stinll,grbrfl
      common /labstr/ label,stplab,grbrf,stpins
C ************************************************************************
C
C     INCLUDE FILE : HEBLOCK.INC
C ************************************************************************
CN    COMMON NAME  : heblock
CA    DESCRIPTION  : Heat Exchanger Block Diagram Dimensions/Specs
CP    PARAMETERS   :
C
C ************************************************************************
      integer*2 resw,resh,vline,hline,tw,th,chw,chh
      integer*2 x,y,x0,y0,xpos,ypos,optxin,optyin
      integer*2 posts(2,10),posfs(2,2),posls(2,4),poscv(2,6),
     +          posst(2,4),post(2,4)
      character*4 tsnam(2,10),fsnam(2,2),lsnam(2,4),cvnam(2,6),
     +            stnam(2,4),tnam(4)

      common /blvals/ resw,resh,vline,hline,tw,th,chw,chh
      common /blnam/ tsnam,fsnam,lsnam,cvnam,stnam,tnam
      common /blcmap/ posts,posfs,posls,poscv,posst,post,optxin,optyin
      common x,y,x0,y0,xpos,ypos
C ************************************************************************
C
C     INCLUDE FILE : IOVAR.INC
C ************************************************************************
CN    COMMON NAME  : iovar
CA    DESCRIPTION  : Current Input/Output/Setpoint Values
CP    PARAMETERS   :
C
C ************************************************************************
      integer*2 input(10),setp(10),output(16)
      common /iovar/ input,setp,output
C ************************************************************************
C
C     INCLUDE FILE : KEYS.INC
C ************************************************************************
CN    COMMON NAME  : keys
CA    DESCRIPTION  : Keyboard constants.
CP    PARAMETERS   :
C

C ************************************************************************
      integer*2 upk,downk,leftk,rightk,homek,endk,pgupk,pgdnk,
     +          esck,retk,tabk,rtabk
      integer*2 sma,bga,sml,bgl,smq,bgq,smm,bgm,sms,bgs
      common /key/ upk,downk,leftk,rightk,homek,endk,pgupk,pgdnk,
```

```
     +             esck,retk,tabk,rtabk,
     +             sma,bga,sml,bgl,smq,bgq,smm,bgm,sms,bgs

C ************************************************************************
C
C     INCLUDE FILE : LOGVAL.INC
C ************************************************************************
CN    COMMON NAME  : logval
CA    DESCRIPTION  : Arrays of Logged Response Data and Details
CP    PARAMETERS   :
C
C ************************************************************************
      integer*2 inpval(1800,10),outval(1800,16),setval(1800,10)
      integer*2 stepid,sampl
      real*4 hertz
      logical*2 logged
      common /values/ inpval,outval,setval
      common /stepid/ stepid
      common /sampl/ sampl
      common /sfrequ/ hertz
      common /logprs/ logged
C ************************************************************************
C
C     INCLUDE FILE : RUNVAR.INC
C ************************************************************************
CN    COMMON NAME  : runvar
CA    DESCRIPTION  : Rig Running Mode and Other Run Variables
CP    PARAMETERS   :
C
C ************************************************************************
      integer*2 mode
      integer*2 inpno,setpno,outpno,steplv
      logical*2 chscr,log,quit
      common /runmod/ mode
      common /runflg/ chscr,log,quit
      common /runval/ inpno,setpno,outpno,steplv
C ************************************************************************
C
C     INCLUDE FILE : SCREEN.INC
C ************************************************************************
CN    COMMON NAME  : screen
CA    DESCRIPTION  : Bloc Diagram Run Screen Titles/Specs
CP    PARAMETERS   :
C
C ************************************************************************
      integer*2 titlen,brflen,optlen
      character*60 optitl(8),optbrf(8),opt(8,3)

      common /scrlen/ titlen,brflen,optlen
      common /scrnam/ optitl,optbrf,opt

C
C     INCLUDE FILE : SYSMNS.INC
C ************************************************************************
CN    COMMON NAME  : sysnms
CA    DESCRIPTION  : System I/O names, matrix names and system titles.
CP    PARAMETERS   :
C           inpnms - System input names.
C           outnms - System output names.
C            prjnm - Project title.
C           engnms - Engineer or design team name.
C ************************************************************************
      dimension inpnms(10),outnms(10)
      character*25 inpnms,outnms,prjnm,engnms
      common /sysnms/ inpnms,outnms,prjnm,engnms

C ************************************************************************
C
C     INCLUDE FILE : SYST.INC
C ************************************************************************
CN    COMMON NAME  : syst
CA    DESCRIPTION  : State-space System Matrices
CP    PARAMETERS   :
C
C ************************************************************************
      integer id,nsyst
      parameter (id    = 15,
     +           nsyst = 5)
      integer precom
      real a,b,c,kci,kfi,q1,r1,kc,q,r,kf,syst,ptrsst
      dimension a(id,id),b(id,id),c(id,id),kci(id,id),kfi(id,id),
     +          q1(id,id),r1(id,id),kc(id,id),
     +          q(id,id),r(id,id),kf(id,id),ptrsst(id,id),
     +          syst(nsyst)
      common /ssdat/ a,b,c,kci,kfi,q1,r1,kc,q,r,kf,ptrsst
      common /syst/ syst
      common /precom/ precom
C
C     INCLUDE FILE : TIME.INC
C ************************************************************************
C ************************************************************************
CN    COMMON NAME  : time1
CA    DESCRIPTION  : Part of the state variables for the time simulation.
CP    PARAMETERS   :
C ************************************************************************
      real swdxdt(15),xo(15),dxdt(15)
      real xn(15),un(15),rn(15),vn(15),yn(15),ynact(15),zn(15),nn(15)
      real xnobs(15),ynobs(15)
      real xndis(15)
      common /simutl/ swdxdt,xo,dxdt
      common /time1/ xn,un,rn,vn,yn,ynact,zn,nn,xnobs,ynobs,xndis

C ************************************************************************
CN    COMMON NAME  : time2
CA    DESCRIPTION  : Part of the state variables for the time simulation.
CP    PARAMETERS   :
C                     ti - Current time.
C                     dt - Time increment.
C                   tend - Time limit.
C                  c1(i) - Runga-Kutta dxdt summing weights.
C                  c2(i) - Runga-Kutta xn calculation time increments.
C                 obsinc - Observer included(0) or excluded(1).
C                 coninc - Controller included(0) or excluded(1).
C                 setplt - Plot(0) or Not Plot(1) setpoints
C                 inpplt - Plot(0) or Not Plot(1) inputs
C                 outplt - Plot(0) or Not Plot(1) outputs
```

```
C                ostplt - Plot(0) or Not Plot(1) observer states
C                pstplt - Plot(0) or Not Plot(1) process states
C             stpinp(i) - Number of input to be stepped.
C             stpdat(i) - Time at which input is to be stepped.
C             stpinc(i) - Step size.
C                 twmax - Time : Maximum width (dots).
C                twstrt - Time : X start point.
C                 thmax - Time : Maximum height (dots).
C                thstrt - Time : Y start point.
C                twidth - Time : No. of plots across the page.
C                theigh - Time : No. of plots down the page.
C                 txdel - Time : Width of one plot (dots).
C                 tydel - Time : Height of one plot (dots).
C ***************************************************************************
      real*4 ti,dt,tend,c1(4),c2(3)
      integer*2 obsinc,coninc,setplt,inpplt,outplt,ostplt,pstplt
      integer*2 stpinp(5)
      real*4 stpdat(5),stpinc(5)
      integer*2 twmax,twstrt,thmax,thstrt,twidth,theigh,txdel,tydel
      common /time2/ ti,dt,tend,c1,c2,
     +               obsinc,coninc,setplt,inpplt,outplt,ostplt,pstplt,
     +               stpinp,stpdat,stpinc,
     +               twmax,twstrt,thmax,thstrt,twidth,theigh,
     +               txdel,tydel

C ***************************************************************************
CN    COMMON NAME  : time3
CA    DESCRIPTION  : Arrays for time simulation axes settings.
CP    PARAMETERS   :
C            xtlim - X axes bounds.
C            ytlim - Y axes bounds.
C           tscale - Axes scales (dots/unit).
C          tcentre - Axes centre (dots and units).
C           tgraph - Temporary array used to draw axes.
C ***************************************************************************

      real*8 xtlim(3),ytlim(3,10),tscale(2,10),tcentr(4,10)
      integer*2 tgraph(4)
      common /time3/ xtlim,ytlim,tscale,tcentr,tgraph

C ***************************************************************************
CN    COMMON NAME  : time4
CA    DESCRIPTION  : Arrays for time simulation axes settings.
CP    PARAMETERS   :
C ***************************************************************************
      integer ninc,dinc,rinc,ntyp,dtyp,rtyp,plnoi
      integer nchset(15),dchset(15),rchset(15)
      integer*4 irand
      real nampl,dampl,rampl,nfre,dfre,rfre
      real nchdat(2,15),dchdat(2,15),rchdat(2,15)
      common /time4/ ninc,dinc,rinc,ntyp,dtyp,rtyp,nchset,dchset,rchset,
     +               irand,nampl,dampl,rampl,nfre,dfre,rfre,
     +               nchdat,dchdat,rchdat,plnoi

C ***************************************************************************
```

N6) HERIG Development/Execution Information

The required subroutines and other program modules for the development of the HERIG package are grouped into libraries of object code. These libraries (and some other object files) are then linked to the object file HERIG.OBJ compiled from the main program to form the executable file, HERIG.EXE. The following instrucution shown in bold type is given to the Microsoft Fortran LINKer:

link
herig+runrig+herun+lqgtyp+lqgrun+logval+lodlqg,,,iglib+helib+pltlib+adlib+inlib+iolib+tmplib+simlib+edlib/E/SE:256;

The object modules herig.obj, runrig.obj, herun.obj, lqgtyp.obj, lqgrun.obj, logval.obj and lodlqg.obj are linked, together with the modules grouped in the libraries iglib.lib, helib.lib, pltlib.lib, adlib.lib, inlib.lib, iolib.lib, tmplib.lib, simlib.lib and edlib.lib, by the Microsoft LINKer. The tables that follow show which data and object modules are grouped together in the indicated libraries. Tables have already been produced in Appendix M (M8) for the libraries iglib.lib, iolib.lib simlib and edlib.lib, which are common to both packages developed, and these are not repeated. The modules are indicated in capital letters, while the file in which the modules are located are indicated by lowercase type.

Module	File	Module	File
ADCOM....	....readop	ADVAR....	....readop
CONVRT...	....adutil	DELAY....	....adutil
INP.....	....adutil	IOVAR....	....wrinp
OUTDA....	....adutil	OUTDAI...	....adutil
OUTP.....	....adutil	READOP...	....readop
RSTDA....	....adutil	RSTDAI...	....adutil
SETADP...	....setadp	SETDAP...	....setdap
WAITDT...	....adutil	WRINP....	....wrinp

Table N8.1: Modules in the library adlib.lib

Module	File	Module	File
ARHEAD...	....arhead	AUTPER...	....fixset
BLCMAP...	....rblock	BLMAP....	....blmap
BLNAM....	....rblock	BLVALS...	....rblock
COMMQQ...	....rblock	CONALG...	....conalg
CONVAR...	....conalg	FHELP....	....mtext
FIXDAT...	....fixdat	FIXOUT...	....fixout
FIXSET...	....fixset	FIXSTP...	....fixstp
HEBLOC...	....heblock	IOVAR....	....updscr
KEY.....	....runkey	LABCON...	....grldat
LABSTR...	....grldat	LOGPRS...	....fixdat
MANPER...	....fixdat	MODFR....	....runkey
RBLOCK...	....rblock	RUNFLG...	....typscr
RUNKEY...	....runkey	RUNMOD...	....typscr
RUNVAL...	....typscr	SAMPL....	....updscr
SCRCON...	....blmap	SCRLEN...	....typscr
SCRNAM...	....typscr	SFREQU...	....fixdat
STEPID...	....runkey	STPPER...	....fixstp
TYPSCR...	....typscr	UPDSCR...	....updscr
VALUES...	....fixdat		

Table N8.2: Modules in the library helib.lib

Module	File	Module	File
ADSTAT...	....init	BLNAM....	....init
BLVALS...	....init	CONVAR...	....inicon
DEFNAM...	....init	FILNAM...	....init
INICON...	....inicon	INILOG...	....inilog
INIOVR...	....iniovr	INIT.....	....init
INSNAM...	....init	IOVAR....	....iniovr
KEY.....	....init	LABCON...	....init
LABSTR...	....init	LOGPRS...	....inilog
MODFR....	....inilog	PLCOMM...	....init
PLOTOP...	....init	PRECOM...	....init
RUNFLG...	....init	RUNINI...	....runini
RUNMOD...	....init	RUNVAL...	....init
SAMPL....	....inilog	SCALOP...	....init
SCRLEN...	....init	SCRNAM...	....init
SFREQU...	....inilog	SIMUTL...	....init
SSSDAT...	....init	STEPID...	....inilog
SYSNMS...	....init	SYST.....	....init
TIME1....	....init	TIME2....	....init
TIME3....	....init	TIME4....	....init
VALUES...	....inilog		

Table N8.3: Modules in the library inlib.lib

Module	File	Module	File
AUTPER...	....heplot	FORMPL...	....formpl
HEPLOT...	....heplot	INSNAM...	....selpl
KEY.....	....servop	LABCON...	....heplot
LABSTR...	....heplot	LOGPRS...	....heplot
MANPER...	....heplot	ONEPL....	....onepl
PLCOMM...	....formpl	PLOTOP...	....selpl
PRSEL....	....prsel	RUNMOD...	....heplot
SAMPL....	....heplot	SCALOP...	....usplot
SELPL....	....selpl	SERVOP...	....servop
SFREQU...	....heplot	STEPID...	....heplot
STPPER...	....heplot	USPLOT...	....usplot
VALUES...	....heplot		

Table N8.4: Modules in the library pltlib.lib

Module	File	Module	File
DATA.....	....data	EXCUSE...	....excuse
HEPAR....	....hepar	IADDEG...	....iaddeg
IADPER...	....iadper	IDEGRC...	....idegrc
IMXMN....	....imxmn	IMXMNS...	....imxmns
IPERC....	....iperc	KEYERR...	....keyerr
MLMVI2...	....mlmvi2	MLMVR8...	....mlmvr8
MXMN.....	....mxmn	MXMNS....	....mxmns
MXMNT....	....mxmnt	PERVVN...	....pervvn
PRECOM...	....pervvn	RDEGRC...	....rdegrc
RNDNRM...	....rndnrm	RPERC....	....rperc
SIMUTL...	....pervvn	SSSDAT...	....pervvn
SYST.....	....pervvn	TIME1....	....pervvn
TIME2....	....pervvn	TIME3....	....pervvn
TIME4....	....pervvn	URAND....	....urand

Table N8.5: Modules in the library tmplib.lib

APPENDIX O

MODEL $G_{ft}(s)$: FLOW INPUTS TO TEMPERATURE OUTPUTS

In this appendix, the elements of $G_{ft}(s)$ relating the flow inputs F1 and F2 to the temperature outputs T1 and T5 are determined.

Plots of the appropriate output responses to the stepping of each input are shown in figures O1 to O12. The transfer functions characterizing each response are also evaluated and quoted with each plot. These transfer functions were derived with the assistance of the NELM algorithm (Astrom & Wittenmark, 1984), and the fitted response data is plotted on the same axes as the observed step test response data.

Each transfer function element determined is indicated in table O1, together with the input stepped and output response observed in determining the element and the plot labels.

Element $gft_{ij}(s)$	Input Stepped	Output Observed	Figure Number
$gft_{11}(s)$	F1(s)	T1(s)	O1
$gft_{21}(s)$	F1(s)	T5(s)	O2
$gft_{12}(s)$	F2(s)	T1(s)	O3
$gft_{22}(s)$	F2(s)	T5(s)	O4
$gft_{12}(s)$	F2(s)	T1(s)	O5
$gft_{22}(s)$	F2(s)	T5(s)	O6
$gft_{12}(s)$	F2(s)	T1(s)	O7
$gft_{22}(s)$	F2(s)	T5(s)	O8
$gft_{11}(s)$	F1(s)	T1(s)	O9
$gft_{21}(s)$	F1(s)	T5(s)	O10
$gft_{11}(s)$	F1(s)	T1(s)	O11
$gft_{21}(s)$	F1(s)	T5(s)	O12

Table O.1: Key to Flow to Temp Model Identification Plots

O.1) Presentation and Analysis of Response Data

O.1.1) Stepping Flow F1, Observing Response of Temp T1

The O/L response of temp output T1 to a +820 unit step change in the flow input F1 is shown in figure O1 as channel 1, while the fitted response (using NELM) is shown as channel 2.

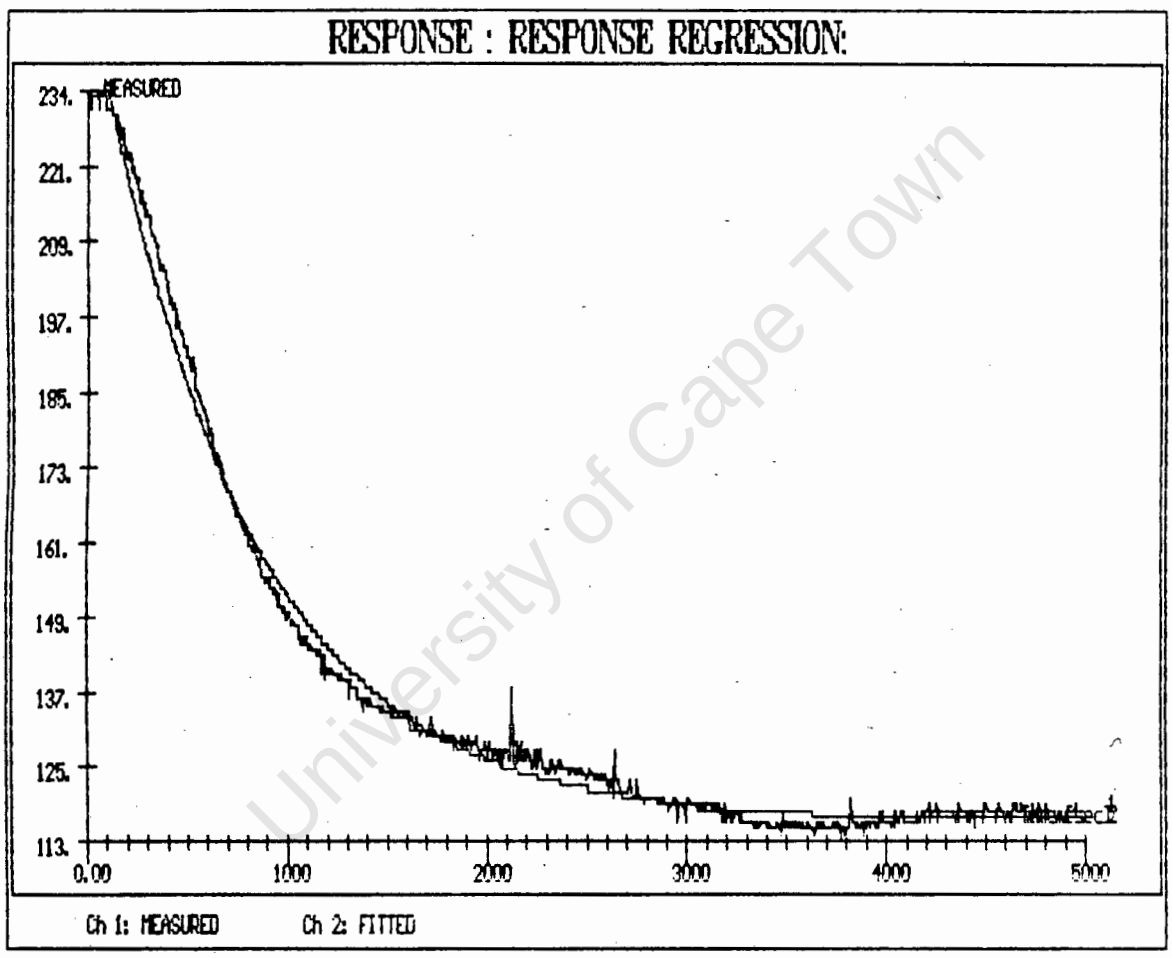


Figure O1: Open Loop Response and Fitted Transfer Function : $gft_{11}(s)$

Transfer Function:

$$gft_{11}(s) = \frac{-0.144}{1 + 769.2s} \qquad \text{Delay} = 50.0 \text{ sec}$$

O.1.2) Stepping Flow F1, Observing Response of Temp T5

The O/L response of temp output T5 to a +820 unit step change in the flow input F1 is shown in figure O2 as channel 1, while the fitted response (using NELM) is shown as channel 2.

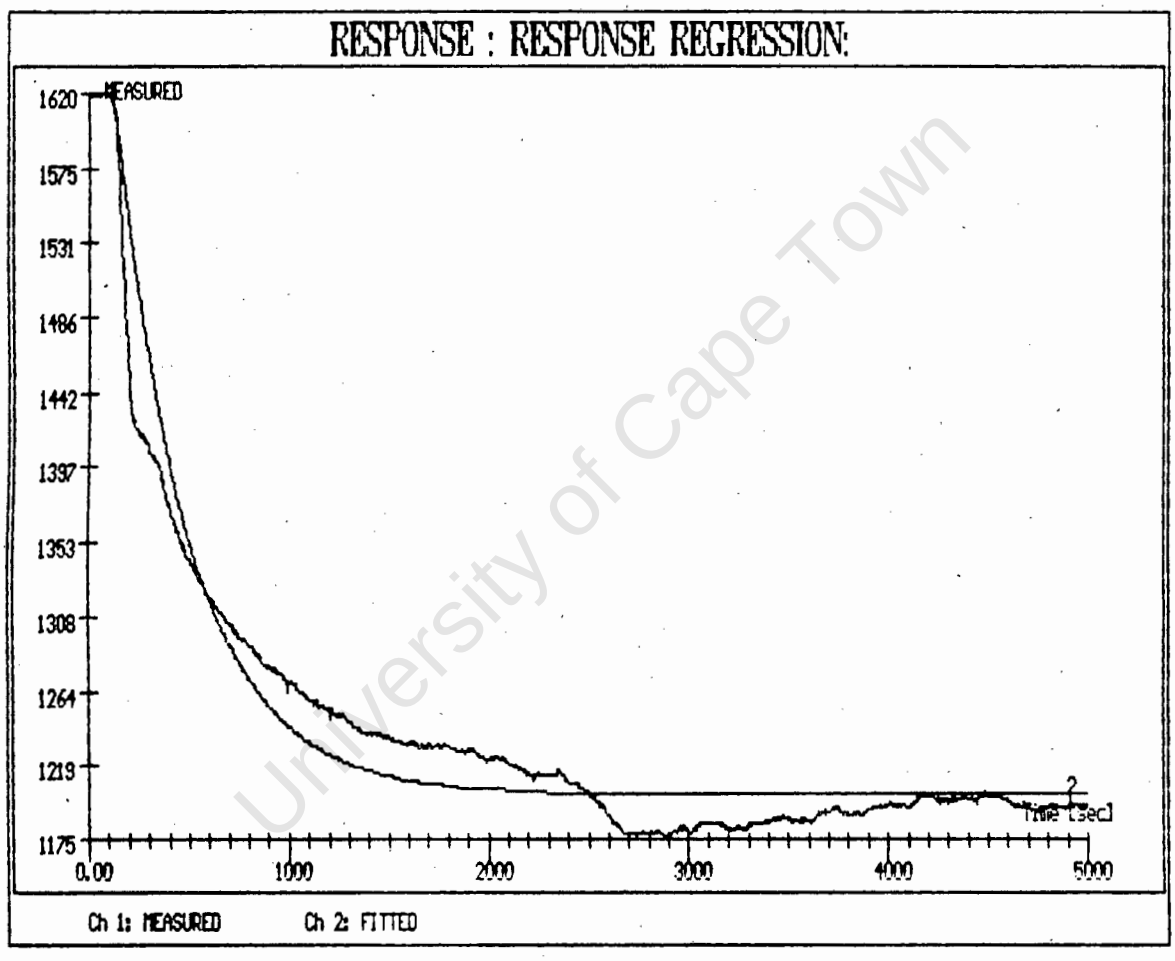


Figure O2: **Open Loop Response and Fitted Transfer Function : $gft_{21}(s)$**

Transfer Function:

$$gft_{21}(s) = \frac{-0.509}{1 + 337.4s} \qquad \text{Delay} = 70.0 \text{ sec}$$

O.1.3) Stepping Flow F2, Observing Response of Temp T1

The O/L response of temp output T1 to a +820 unit step change in the flow input F2 is shown in figure O3 as channel 1, while the fitted response (using NELM) is shown as channel 2.

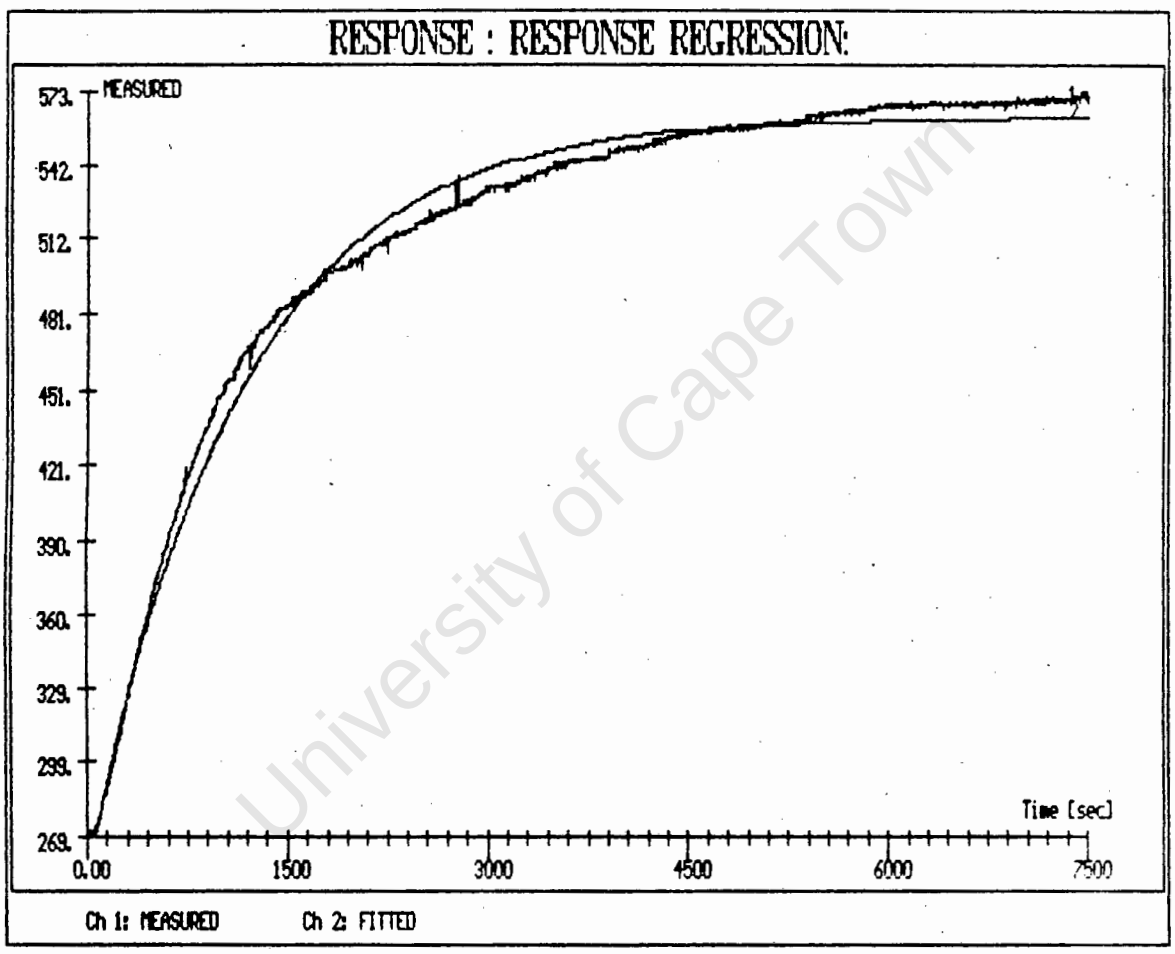


Figure O3: **Open Loop Response and Fitted Transfer Function : $gft_{12}(s)$**

Transfer Function:

$$gft_{12}(s) = \frac{0.358}{1 + 1123.6s} \qquad \text{Delay} = 10.0 \text{ sec}$$

O.1.4) Stepping Flow F2, Observing Response of Temp T5

The O/L response of temp output T5 to a +820 unit step change in the flow input F2 is shown in figure O4 as channel 1, while the fitted response (using NELM) is shown as channel 2.

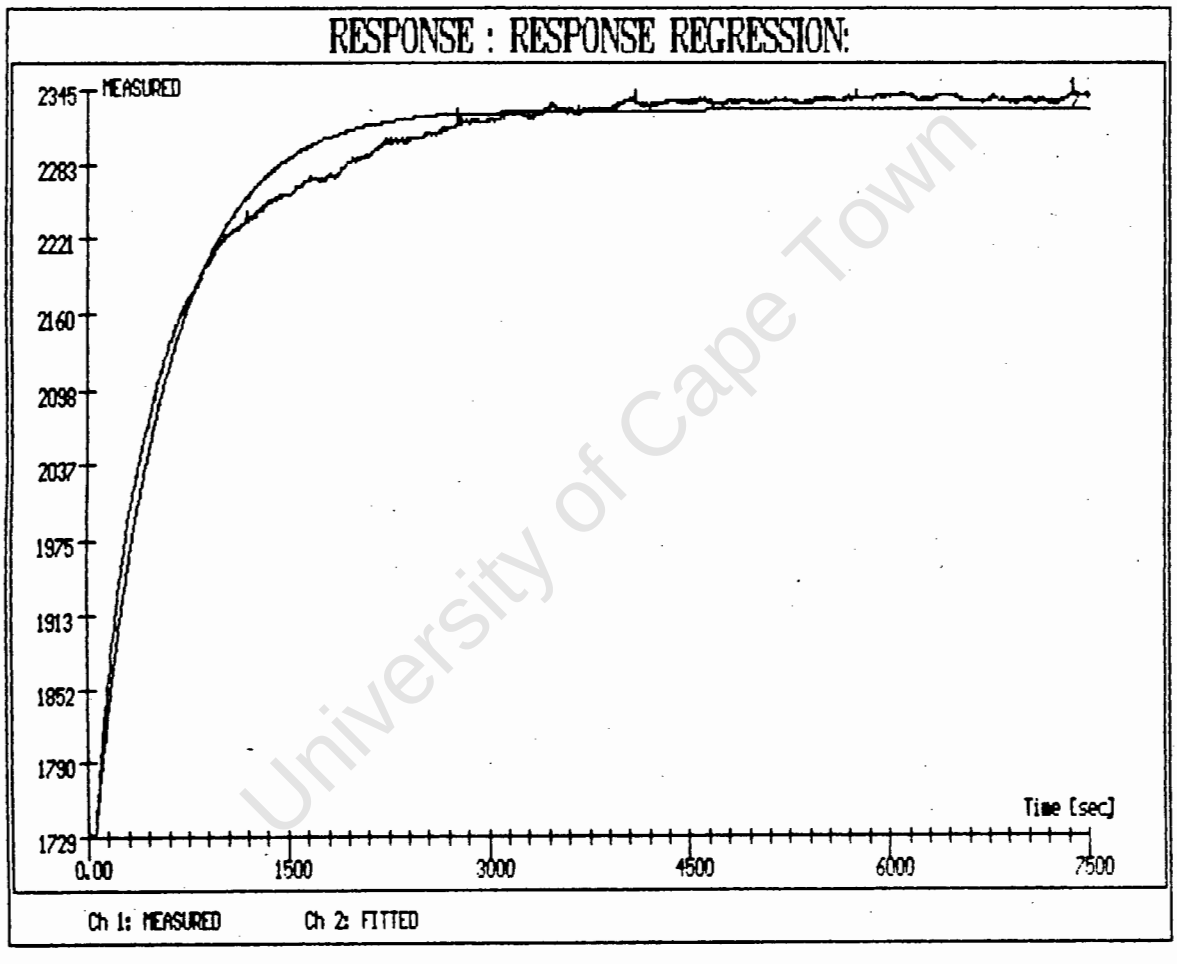


Figure O4: Open Loop Response and Fitted Transfer Function : $gft_{22}(s)$

Transfer Function:

$$gft_{22}(s) = \frac{0.731}{1 + 537.6s} \qquad \text{Delay} = 0.0 \text{ sec}$$

O.1.5) Stepping Flow F2, Observing Response of Temp T1

The O/L resp. of temp output T1 to a -1299 unit step change in the flow input F2 is shown in figure O5 as channel 1, while the fitted response (using NELM) is shown as channel 2.

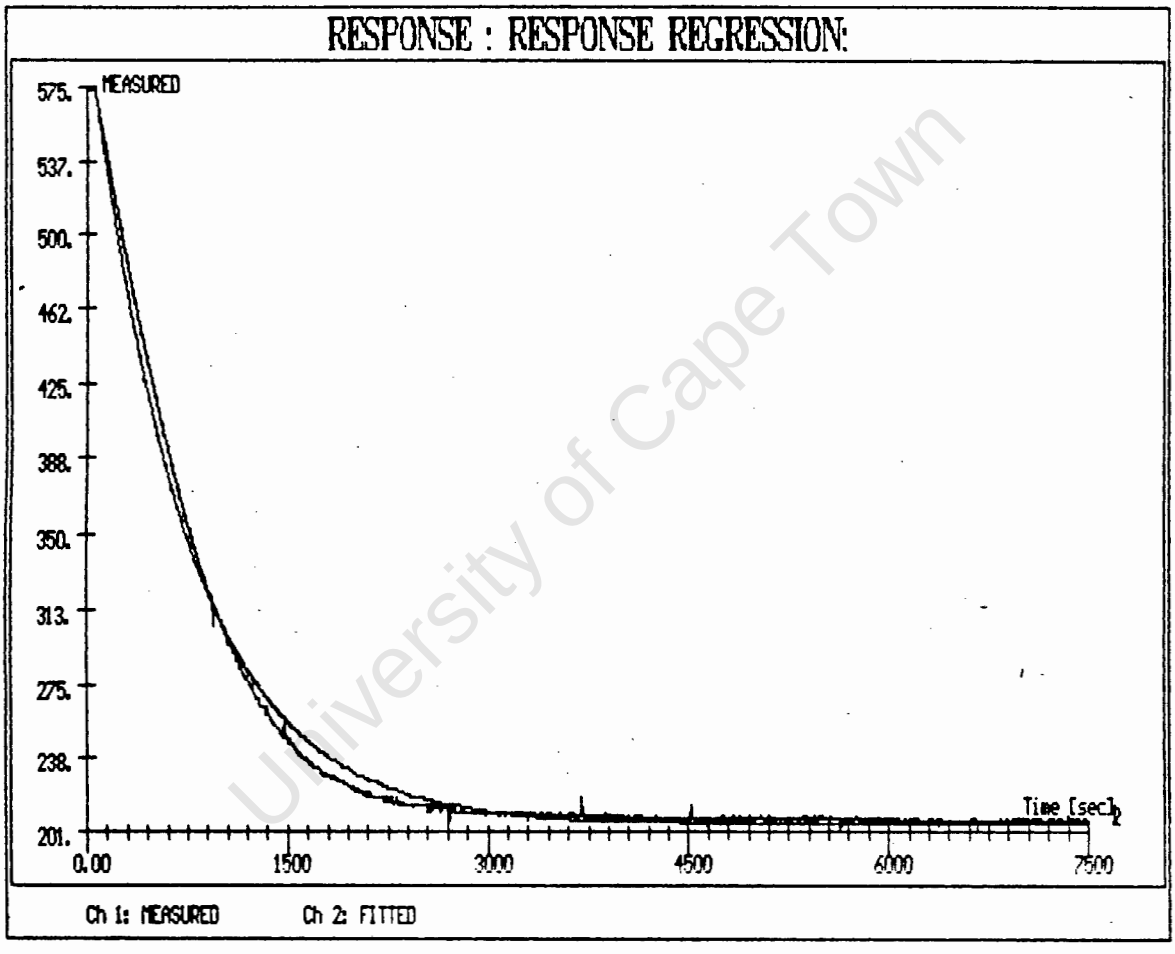


Figure O5: **Open Loop Response and Fitted Transfer Function : $gft_{12}(s)$**

Transfer Function:

$$gft_{12}(s) = \frac{0.301}{1 + 735.3s} \qquad \text{Delay} = 5.0 \text{ sec}$$

O.1.6) Stepping Flow F2, Observing Response of Temp T5

The O/L resp. of temp output T5 to a -1229 unit step change in the flow input F2 is shown in figure O6 as channel 1, while the fitted response (using NELM) is shown as channel 2.

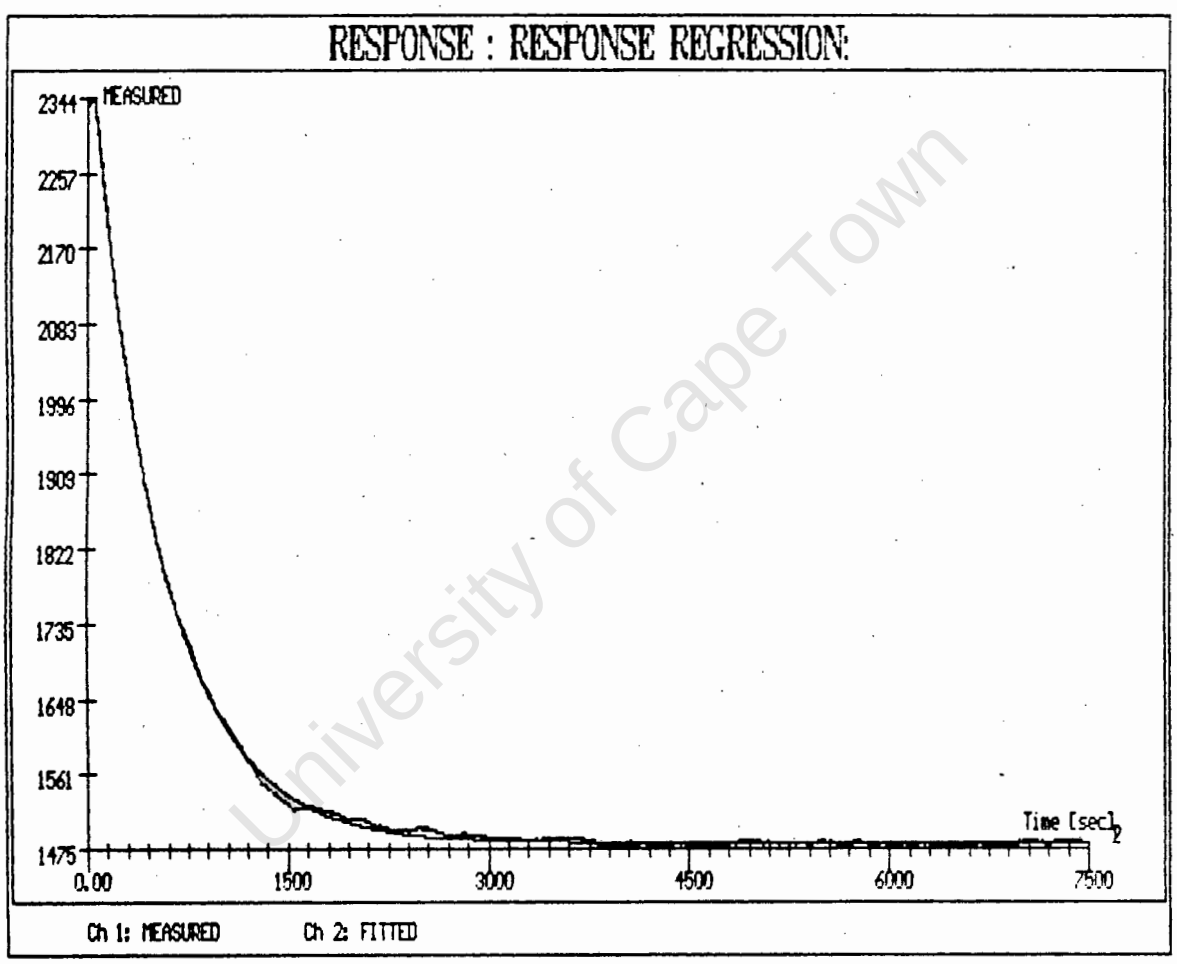


Figure O6: **Open Loop Response and Fitted Transfer Function : $gft_{22}(s)$**

Transfer Function:

$$gft_{22}(s) = \frac{0.701}{1 + 529.1s} \qquad \text{Delay} = 0.0 \text{ sec}$$

O.1.7) Stepping Flow F2, Observing Response of Temp T1

The O/L response of temp output T1 to a +410 unit step change in the flow input F2 is shown in figure O7 as channel 1, while the fitted response (using NELM) is shown as channel 2.

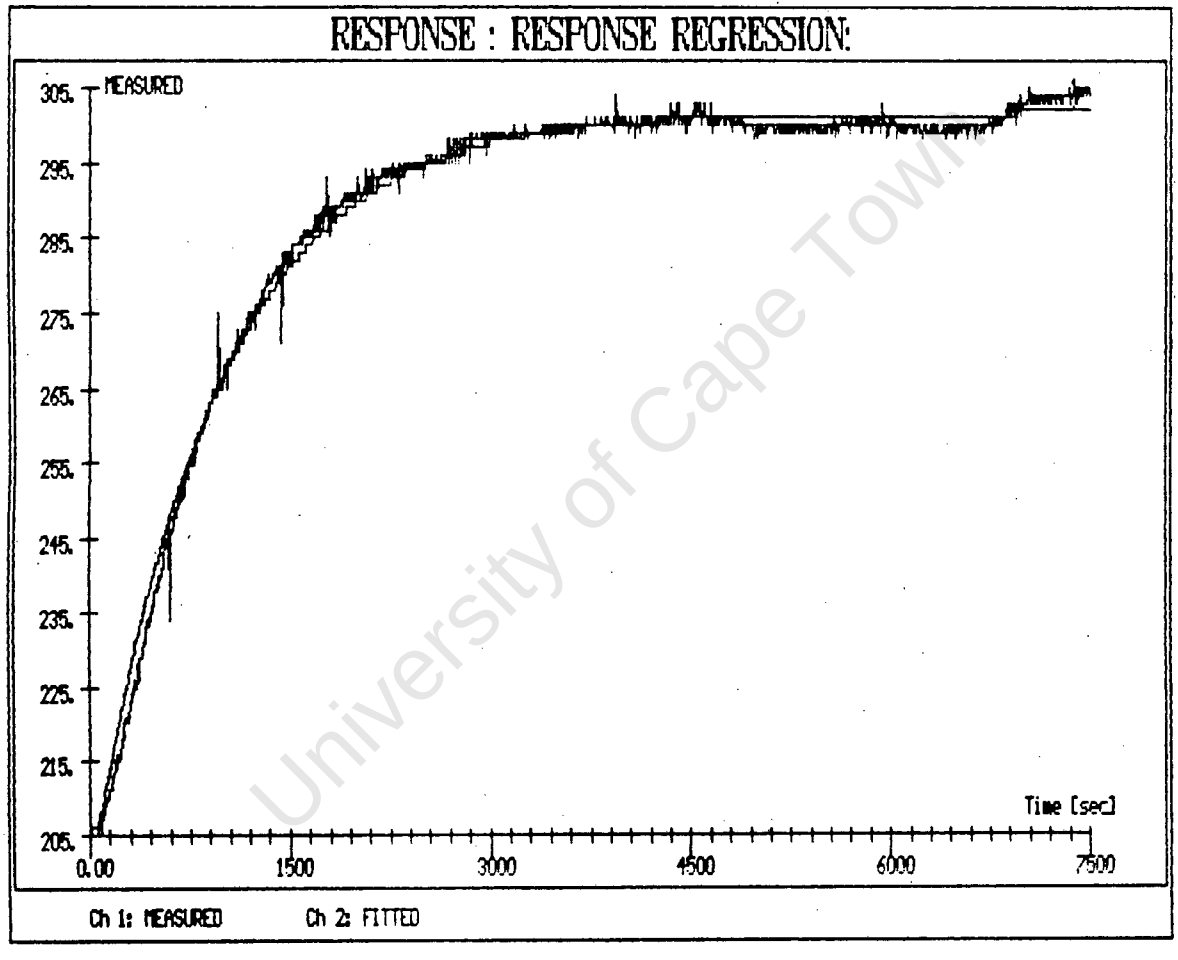


Figure O7: **Open Loop Response and Fitted Transfer Function : $gft_{12}(s)$**

Transfer Function:

$$gft_{12}(s) = \frac{0.236}{1 + 925.9s} \qquad \text{Delay} = 10.0 \text{ sec}$$

O.1.8) Stepping Flow F2, Observing Response of Temp T5

The O/L response of temp output T5 to a +410 unit step change in the flow input F2 is shown in figure O8 as channel 1, while the fitted response (using NELM) is shown as channel 2.

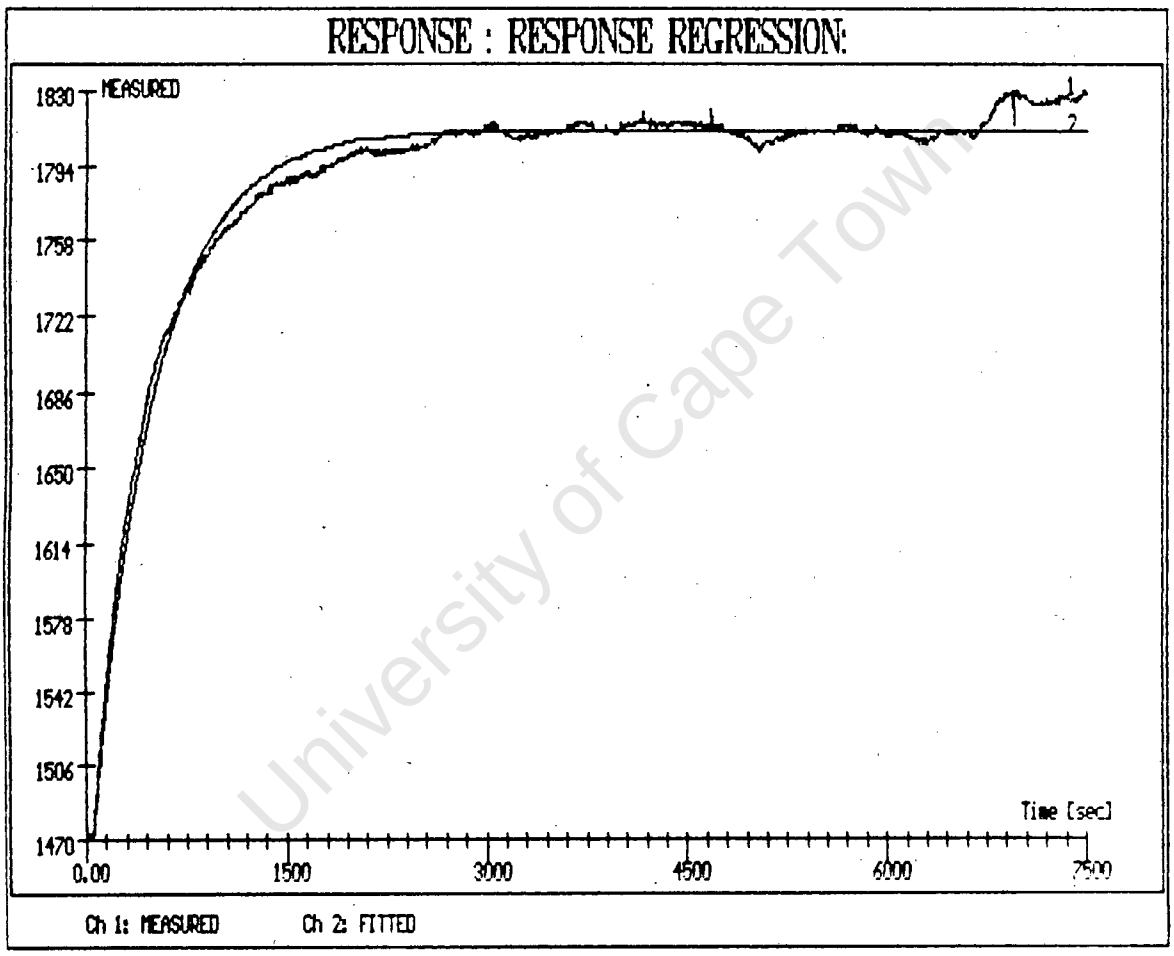


Figure O8: **Open Loop Response and Fitted Transfer Function : $gft_{22}(s)$**

Transfer Function:

$$gft_{22}(s) = \frac{0.831}{1 + 463.0s} \qquad \text{Delay} = 0.0 \text{ sec}$$

O.1.9) Stepping Flow F1, Observing Response of Temp T1

The O/L resp. of temp output T1 to a -1229 unit step change in the flow input F1 is shown in figure O9 as channel 1, while the fitted response (using NELM) is shown as channel 2.

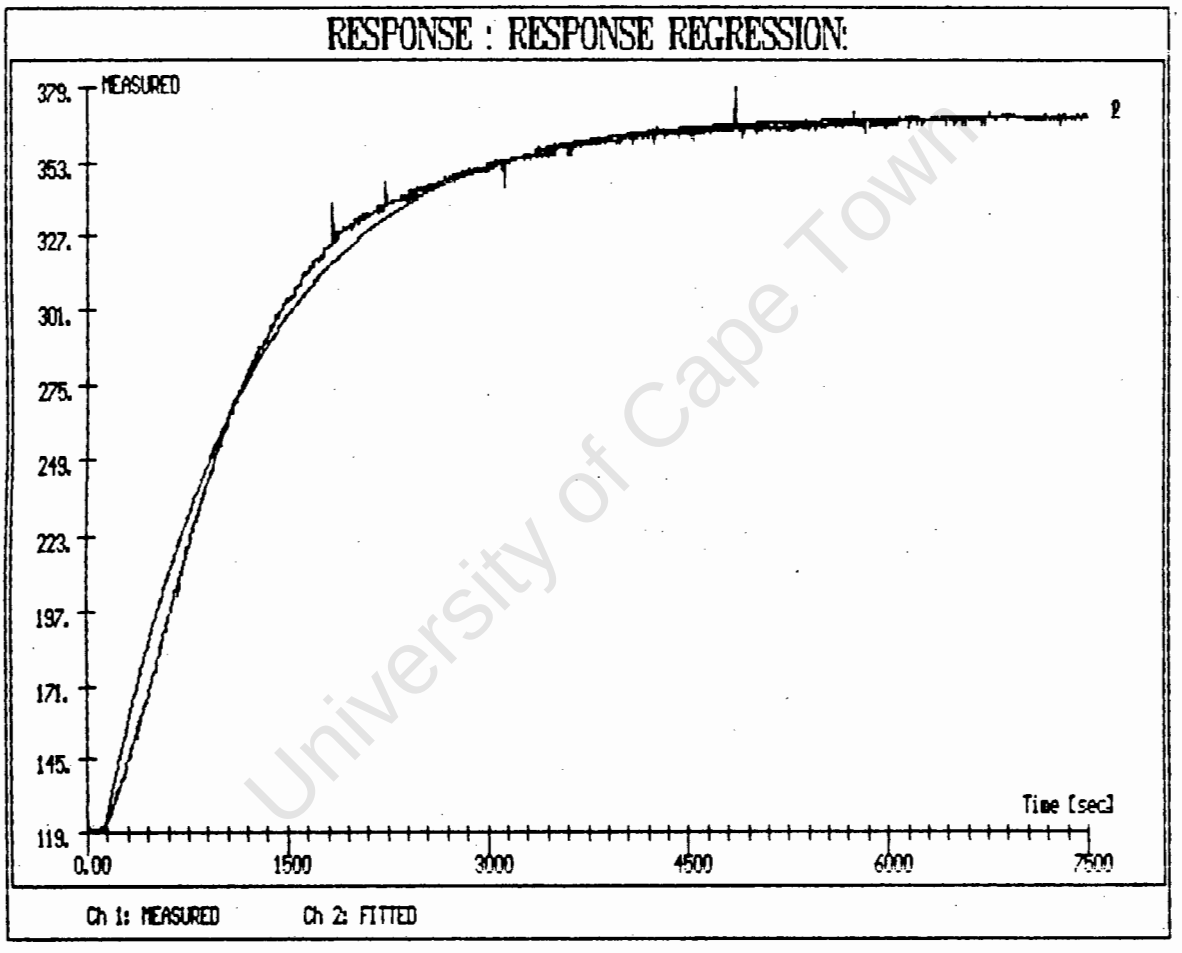


Figure O9: **Open Loop Response and Fitted Transfer Function : $gft_{11}(s)$**

Transfer Function:

$$gft_{11}(s) = \frac{-0.204}{1 + 1087.0s} \qquad \text{Delay} = 70.0 \text{ sec}$$

O.1.10) Stepping Flow F1, Observing Response of Temp T5

The O/L resp. of temp output T5 to a -1229 unit step change in the flow input F1 is shown in figure O10 as channel 1, while the fitted response (using NELM) is shown as channel 2.

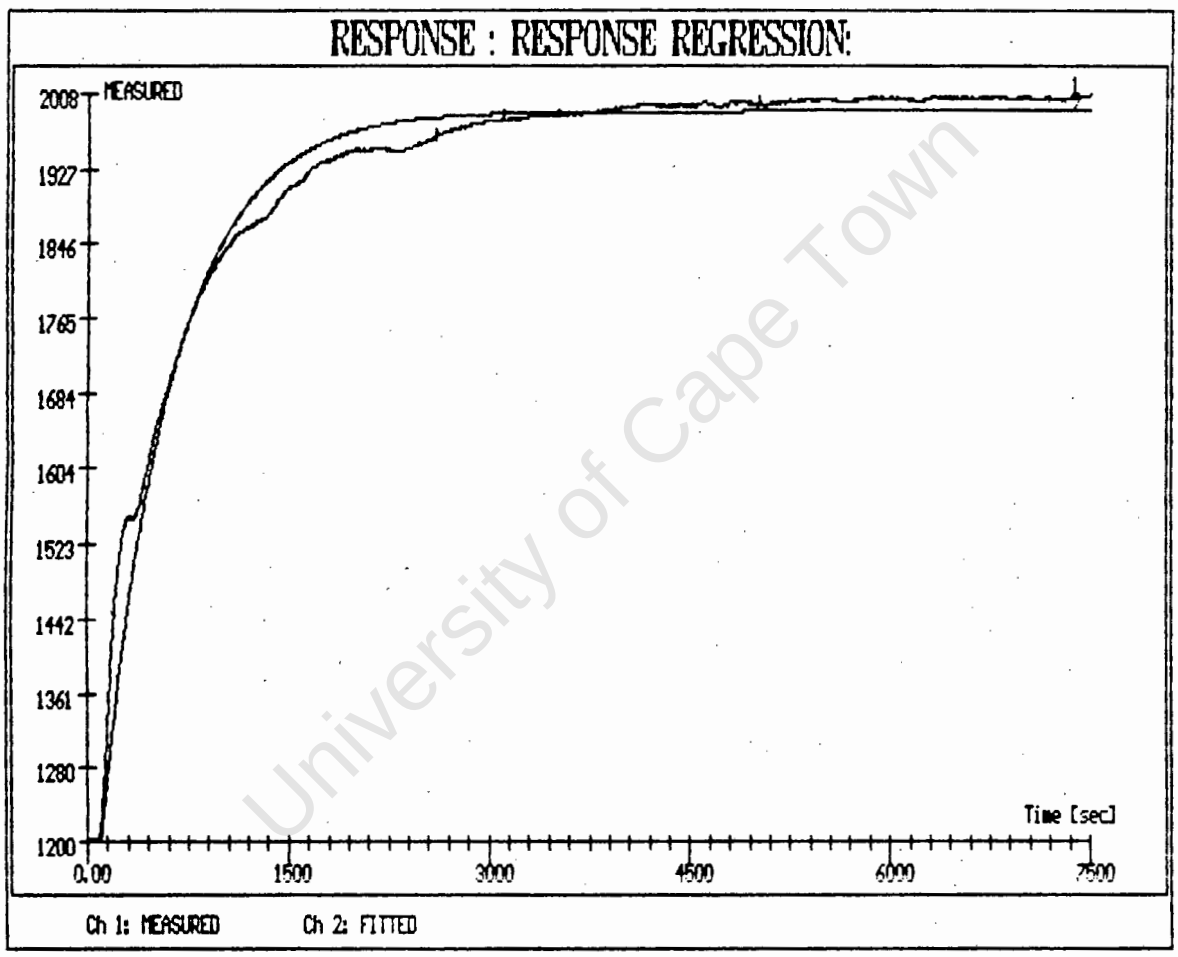


Figure O10: **Open Loop Response and Fitted Transfer Function : $gft_{21}(s)$**

Transfer Function:

$$gft_{21}(s) = \frac{-0.642}{1 + 523.6s} \qquad \text{Delay} = 50.0 \text{ sec}$$

O.1.11) Stepping Flow F1, Observing Response of Temp T1

The O/L response of temp output T1 to a +410 unit step change in the flow input F1 is shown in figure O1 as channel 1, while the fitted response (using NELM) is shown as channel 2.

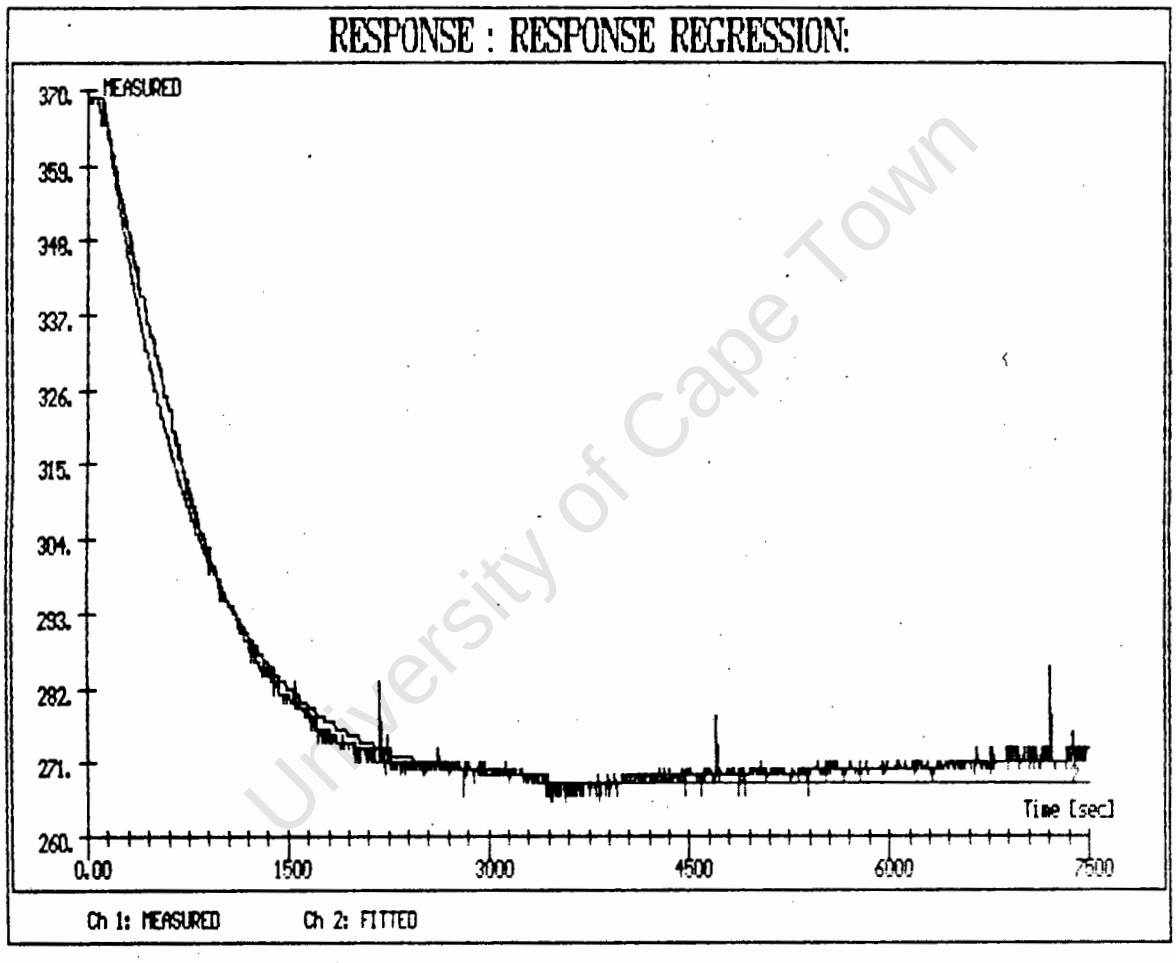


Figure O11: **Open Loop Response and Fitted Transfer Function : $gft_{11}(s)$**

Transfer Function:

$$gft_{11}(s) = \frac{-0.248}{1 + 719.4s} \qquad \text{Delay} = 60.0 \text{ sec}$$

O.1.12) Stepping Flow F1, Observing Response of Temp T5

The O/L response of temp output T5 to a +410 unit step change in the flow input F1 is shown in figure O12 as channel 1, while the fitted response (using NELM) is shown as channel 2.

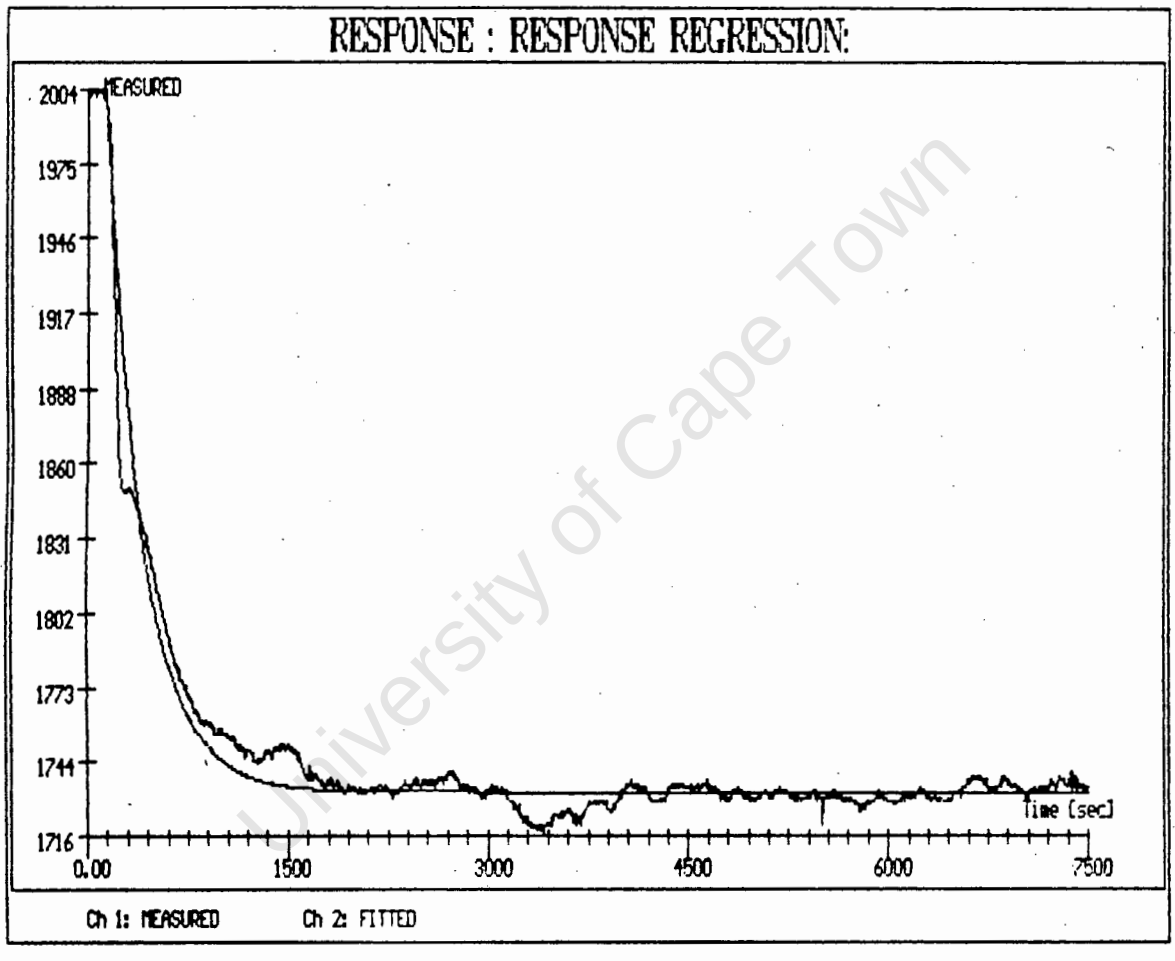


Figure O12: Open Loop Response and Fitted Transfer Function : $gft_{21}(s)$

Transfer Function:

$$gft_{21}(s) = \frac{-0.660}{1 + 278.6s} \qquad \text{Delay} = 80.0 \text{ sec}$$

APPENDIX P

TRANSFORMATION OF PROCESS INTO STATE SPACE DESCRIPTION

The frequency-domain description of the heat exchanger process model, relating the flow inputs to the outlet temperature outputs, is assumed to be in the form shown in (P.1).

$$\mathbf{G_{ft}(s)} = \begin{bmatrix} \dfrac{K_{11}}{1 + sT_{11}} & \dfrac{K_{12}}{1 + sT_{12}} \\ \dfrac{K_{21}}{1 + sT_{21}} & \dfrac{K_{22}}{1 + sT_{22}} \end{bmatrix} \qquad \text{(P.1)}$$

The description (P.1) can be transformed, by direct programming, to the state space system description (P.2)..(P.6)

$$d/dt(\mathbf{X(t)}) = \mathbf{AX(t)} + \mathbf{BU(t)} \qquad \text{(P.2)}$$

$$\mathbf{Y(t)} = \mathbf{CX(t)} \qquad \text{(P.3)}$$

where:

$$\mathbf{A} = \begin{bmatrix} -1/T_{11} & 0.0 & 0.0 & 0.0 \\ 0.0 & -1/T_{12} & 0.0 & 0.0 \\ 0.0 & 0.0 & -1/T_{21} & 0.0 \\ 0.0 & 0.0 & 0.0 & -1/T_{22} \end{bmatrix} \quad \text{(P.4)}$$

$$\mathbf{B} = \begin{bmatrix} 1/T_{11} & 0.0 \\ 0.0 & 1/T_{12} \\ 1/T_{21} & 0.0 \\ 0.0 & 1/T_{22} \end{bmatrix} \quad \text{(P.5)}$$

$$\mathbf{C} = \begin{bmatrix} K_{11} & K_{11} & 0.0 & 0.0 \\ 0.0 & 0.0 & K_{11} & K_{11} \end{bmatrix} \quad \text{(P.6)}$$

The matrices **A**, **B** and **C** obtained are checked by performing the multiplication $\mathbf{G(s)} = \mathbf{C}(s\mathbf{I} - \mathbf{A})^{-1}\mathbf{B}$ and it is found that the resultant **G(s)** correctly equals the original expression for $G_{ft}(s)$ in (P.1).

The process states are now rearranged to ensure that two of them equal the process outputs.

Define new vector of states Z(t) such that:

$$X(t) = T\,Z(t) \qquad \text{(P.7)}$$

or, equivalently,

$$Z(t) = T^{-1}\,X(t) \qquad \text{(P.8)}$$

so,

$$d/dt(X(t)) = A\,X(t) + B\,U(t) \qquad \text{(P.9)}$$

$$T^{-1}\,d/dt(X(t)) = T^{-1}\,AX(t) + T^{-1}\,B\,U(t) \qquad \text{(P.10)}$$

$$d/dt(Z(t)) = T^{-1}\,A\,T\,Z(t) + T^{-1}\,B\,U(t) \qquad \text{(P.11)}$$

$$d/dt(Z(t)) = A'\,Z(t) + B'\,U(t) \qquad \text{(P.12)}$$

Also output equation:

$$\begin{aligned} Y(t) &= C\,X(t) \\ &= C\,T\,Z(t) \\ &= C'\,Z(t) \end{aligned} \qquad \text{(P.13)}$$

The process is thus transformed into a new state-space system description (P.12)..(P.13). This description can be used in

the design of a state feedback control law for the process provided that the system (P.12)..(P.13) is controllable and observable.

Now consider the output equation for the new system:

$$\mathbf{Y(t)} = \mathbf{C\ T\ Z(t)} \qquad \text{(P.14)}$$

It is thus possible to ensure that the states $z_1(t)$ and $z_4(t)$ equal the process outputs $y_1(t)$ and $y_2(t)$ by choosing **T** so that (P.14) is satisfied. This then implies that:

$$\mathbf{C\ T} = \begin{bmatrix} 1.0 & 0.0 & 0.0 & 0.0 \\ 0.0 & 0.0 & 0.0 & 1.0 \end{bmatrix} \qquad \text{(P.15)}$$

so,

$$\begin{bmatrix} K_{11} & K_{11} & 0.0 & 0.0 \\ 0.0 & 0.0 & K_{11} & K_{11} \end{bmatrix} \mathbf{T} = \begin{bmatrix} 1.0 & 0.0 & 0.0 & 0.0 \\ 0.0 & 0.0 & 0.0 & 1.0 \end{bmatrix} \qquad \text{(P.16)}$$

Solving for equation (P.16) yields the possible solution:

$$\mathbf{T} = \begin{bmatrix} 1/K_{11} & 1.0 & 0.0 & 0.0 \\ 0.0 & -K_{11}/K_{12} & 0.0 & 0.0 \\ 0.0 & 0.0 & -K_{22}/K_{21} & 0.0 \\ 0.0 & 0.0 & 1.0 & 1/K_{22} \end{bmatrix} \qquad \text{(P.17)}$$

Multiplying **C** with this solution for **T** produces the required result (P.20), meaning that the state space description of the process has been modified sufficiently to realize two of the states to be equal to the outputs.

The modified system matrices for the transformed system (P.12)..(P.13) then become

$$\mathbf{A'} = \mathbf{T}^{-1}\,\mathbf{A}\,\mathbf{T}$$

$$= \begin{bmatrix} K_{11} & K_{12} & 0.0 & 0.0 \\ 0.0 & -K_{12}/T_{12} & 0.0 & 0.0 \\ 0.0 & 0.0 & -1/T_{21} & 0.0 \\ 0.0 & 0.0 & K_{21} & K_{22} \end{bmatrix} \qquad \text{(P.18)}$$

$$\mathbf{B'} = \mathbf{T}^{-1}\,\mathbf{B}$$

$$= \begin{bmatrix} K_{11}/T_{11} & K_{12}/T_{12} \\ 0.0 & -K_{12}/(K_{11}T_{12}) \\ -K_{21}/(K_{22}T_{21}) & 0.0 \\ K_{21}/T_{21} & K_{22}/T_{22} \end{bmatrix} \qquad \text{(P.19)}$$

$$\mathbf{C'} = \begin{bmatrix} 1.0 & 0.0 & 0.0 & 0.0 \\ 0.0 & 0.0 & 0.0 & 1.0 \end{bmatrix} \qquad \text{(P.20)}$$

The matrices **A'**, **B'** and **C'** are once again checked by performing the multiplication $\mathbf{G(s)} = \mathbf{C'(sI - A')^{-1}B'}$ and it is found that the resultant **G(s)** again correctly equals the original expression for $G_{ft}(s)$ in (P.1).

APPENDIX Q

VERIFICATION OF PERFORMANCE INDEX DATA: SYSTEM 2

Q.1) Individual Performance Index Plots and Analysis

	Low Freq Band (x 10^{-3} rad/sec)		
	4.35	5.22	Maximum
$\|T_{YD}(s)\|_{max}$ (dB)	0.0	0.0	0.0
$\|T_{UD}(s)\|_{max}$ (dB)	-8.0	-7.7	-7.7
$\|T_{YN}(s)\|_{max}$ (dB)	-15.0	-15.6	-15.0
$\|T_{UN}(s)\|_{max}$ (dB)	-8.0	-7.7	-7.7
$\|SE_Y(s)\|_{max}$ (dB)	0.0	0.0	0.0
$\|SE_U(s)\|_{max}$ (dB)	-15.0	-16.1	-15.0
$\|ERR(s)\|_{max}$ (dB)	1.13	1.00	1.13
$\|T_{UR}(s)\|_{max}$ (dB)	0.0	0.0	0.0

Table Q.1: **Values of Each Performance Index at Two Indicated Frequencies in Low Frequency Band and the Maximum of these two values. (Obtained from Figures Q.1 to Q.4)**

	Intermediate Freq Band (x 10^{-2} rad/sec)		
	1.31	1.57	Maximum
$\|T_{YD}(s)\|_{max}$ (dB)	0.0	0.0	0.0
$\|T_{UD}(s)\|_{max}$ (dB)	-7.1	-7.1	-7.1
$\|T_{YN}(s)\|_{max}$ (dB)	-22.3	-24.1	-22.3
$\|T_{UN}(s)\|_{max}$ (dB)	-7.1	-7.1	-7.1
$\|SE_Y(s)\|_{max}$ (dB)	0.0	0.0	0.0
$\|SE_U(s)\|_{max}$ (dB)	-22.3	-24.1	-22.3
$\|ERR(s)\|_{max}$ (dB)	0.54	0.45	0.54
$\|T_{UR}(s)\|_{max}$ (dB)	0.0	0.0	0.0

Table Q.2: **Values of Each Performance Index at Two Indicated Frequencies in Interm. Frequency Band and the Maximum of these two values. (Obtained from Figures Q.1 to Q.4)**

	High Freq Band (x 10^{-2} rad/sec)		
	2.13	2.61	Maximum
$\|T_{YD}(s)\|_{max}$ (dB)	0.0	0.0	0.0
$\|T_{UD}(s)\|_{max}$ (dB)	-7.1	-7.1	-7.1
$\|T_{YN}(s)\|_{max}$ (dB)	-26.3	-28.1	-26.3
$\|T_{UN}(s)\|_{max}$ (dB)	-7.1	-7.1	-7.1
$\|SE_Y(s)\|_{max}$ (dB)	0.0	0.0	0.0
$\|SE_U(s)\|_{max}$ (dB)	-26.8	-28.6	-26.8
$\|ERR(s)\|_{max}$ (dB)	0.36	0.27	0.36
$\|T_{UR}(s)\|_{max}$ (dB)	0.0	0.0	0.0

Table Q.3: **Values of Each Performance Index at Two Indicated Frequencies in High Frequency Band and the Maximum of these two values. (Obtained from Figures Q.1 to Q.4)**

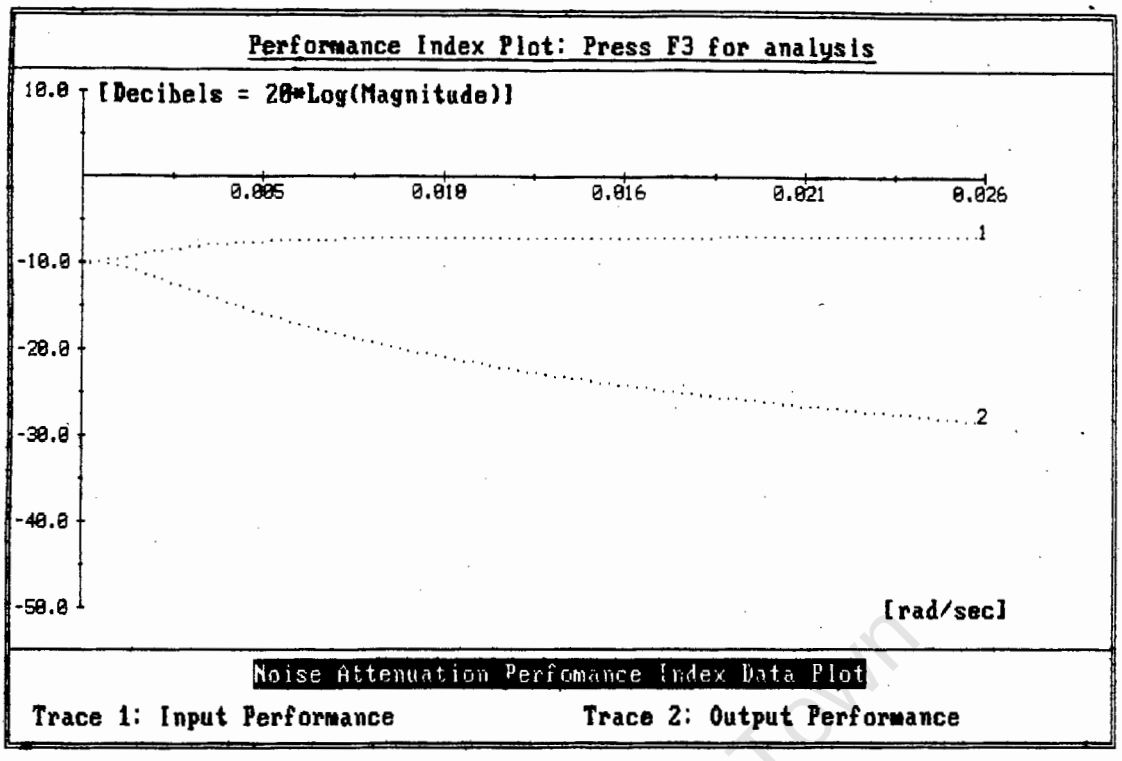


Figure Q.1: System 2 - Noise Transmission Index Plots

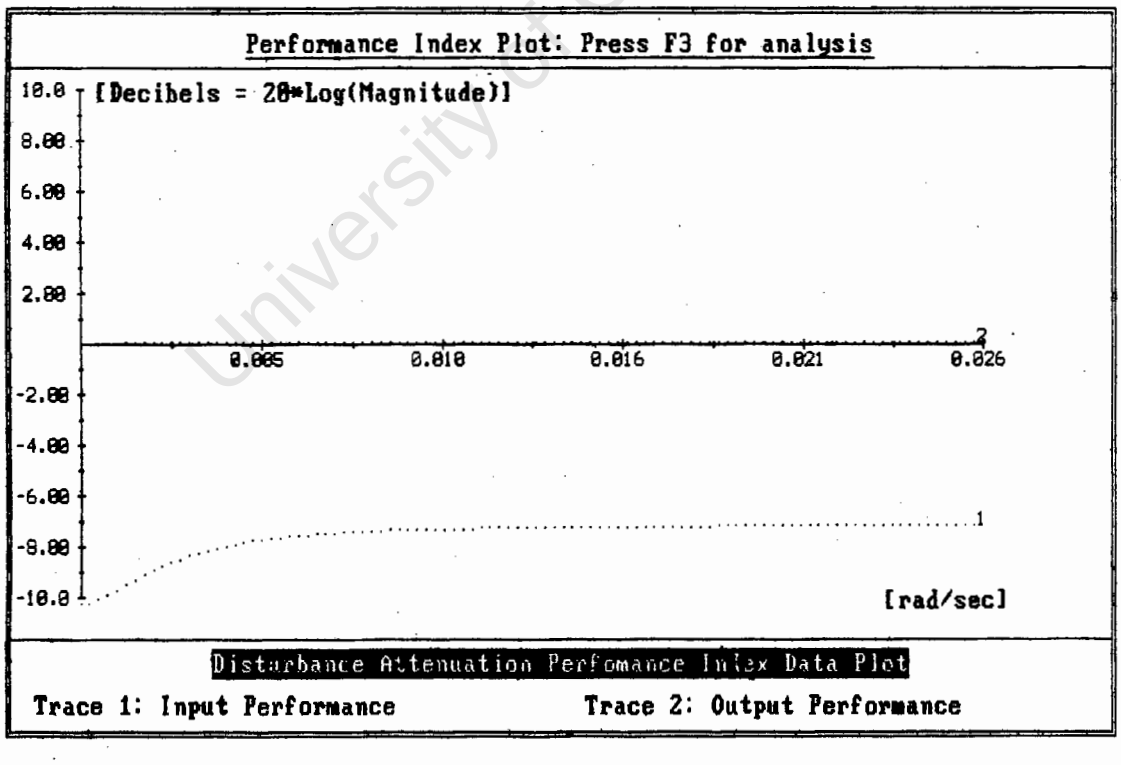


Figure Q.2: System 2 - Disturbance Transmission Index Plots

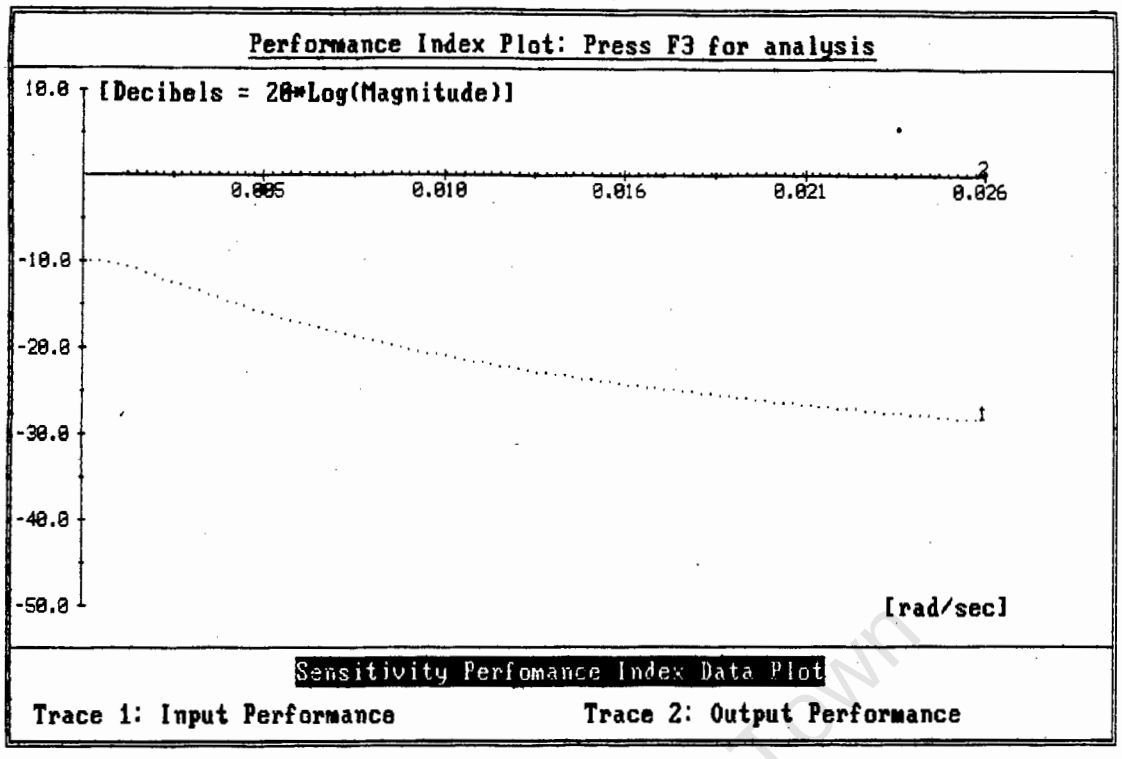


Figure Q.3: System 2 - Sensitivity Index Plots

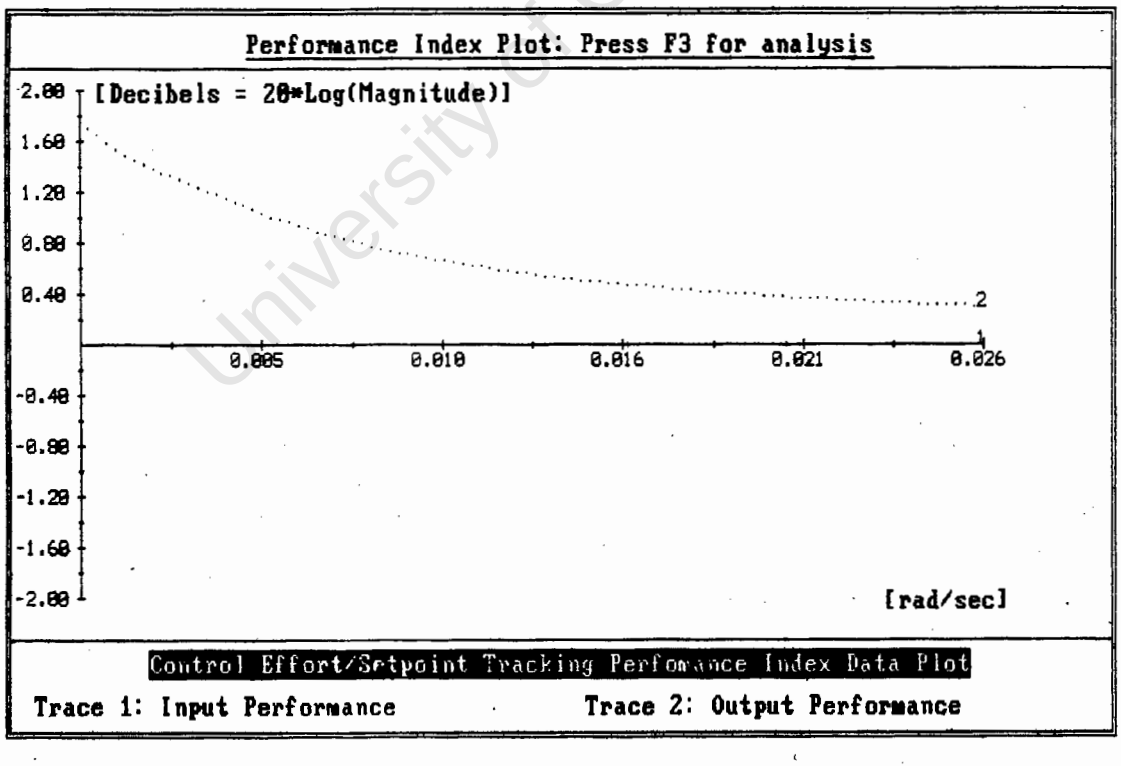


Figure Q.4: System 2 - Control Effort and Tracking Error Index Plots

Q.2) Simulated Input and Output Responses

	Low Freq Band (0.00435 to 0.00522 rad/sec)		
	MR_{ch1} (units)	MR_{ch2} (units)	MR_{max}/A_{pert} (dB)
Dist. to Output D->Y	10.4	10.0	0.34
Dist. to Input D->U	3.2	3.9	-8.18
Noise to Output N->Y	0.5	2.2	-13.15
Noise to Input N->U	3.3	3.9	-8.18
Setp. to Error R->E	11.0	12.0	1.58
Setp. to Input R->U	10.5	10.8	0.66

Table Q.4: **Maximum Simulated Amplitudes of Response, MR_{max} to Perturbations in Low Frequency Band with A_{pert} = 10 units.**
(Obtained from Figures Q.5 to Q.17)

	Intermediate Frequency Band (0.01305 to 0.01566 rad/sec)		
	MR_{ch1} (units)	MR_{ch2} (units)	MR_{max}/A_{pert} (dB)
Dist. to Output D->Y	10.0	10.5	0.42
Dist. to Input D->U	3.5	4.0	-7.96
Noise to Output N->Y	0.25	1.2	-18.42
Noise to Input N->U	3.5	4.1	-7.74
Setp. to Error R->E	10.2	11.0	0.82
Setp. to Input R->U	10.3	10.5	0.42

Table Q.5: **Maximum Simulated Amplitudes of Response, MR_{max} to Perturbations in Intermediate Frequency Band with A_{pert} = 10 units. (Obtained from Figures Q.5 to Q.17)**

	High Freq Band (0.00435 to 0.00522 rad/sec)		
	MR_{ch1} (units)	MR_{ch2} (units)	MR_{max}/A_{pert} (dB)
Dist. to Output D->Y	10.0	10.5	0.42
Dist. to Input D->U	3.5	4.0	-7.96
Noise to Output N->Y	0.15	0.79	-22.05
Noise to Input N->U	3.6	4.1	-7.74
Setp. to Error R->E	10.0	10.5	0.42
Setp. to Input R->U	10.3	10.3	0.25

Table Q.6: **Maximum Simulated Amplitudes of Response, MR_{max} to Perturbations in High Frequency Band with A_{pert} = 10 units.**
(Obtained from Figures Q.5 to Q.17)

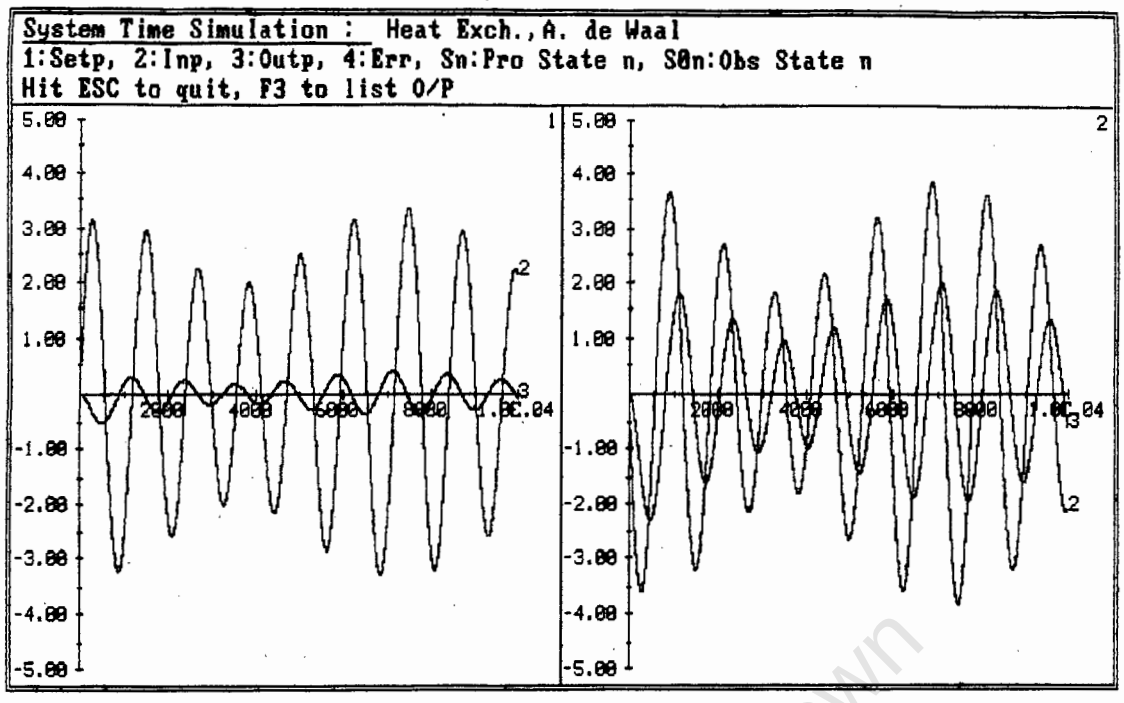


Figure Q.5: System 2 - Simulated Response of Inputs to Low Frequency Sensor Noise

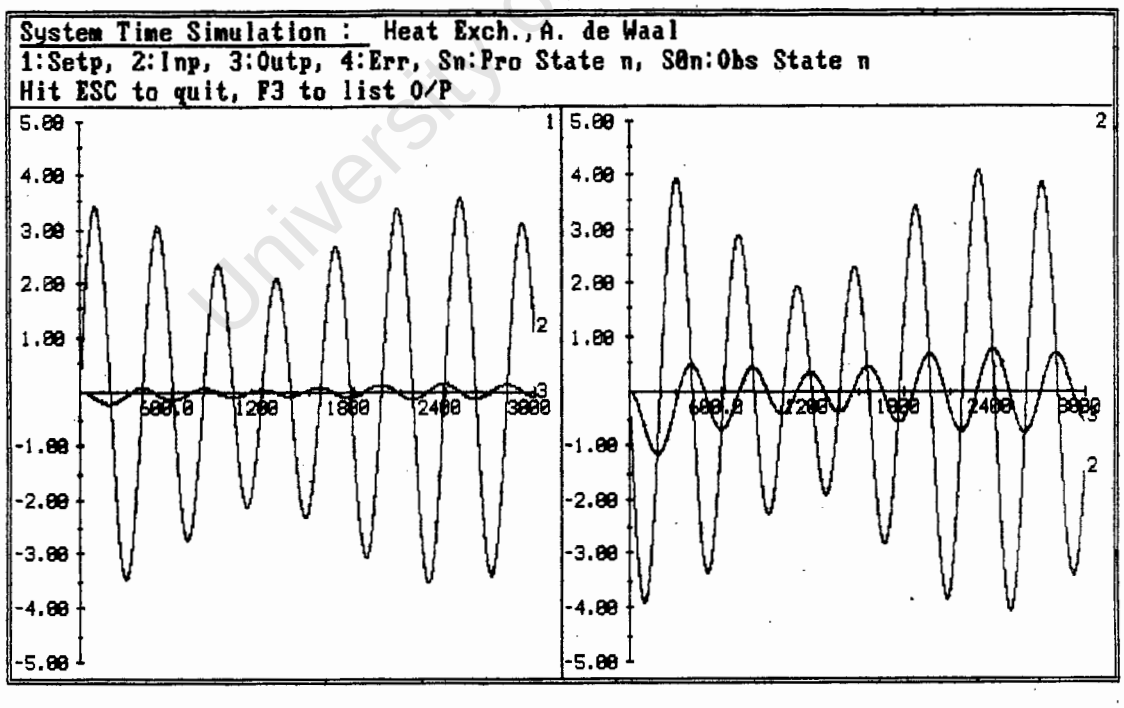


Figure Q.6: System 2 - Simulated Response of Inputs to Interm. Frequency Sensor Noise

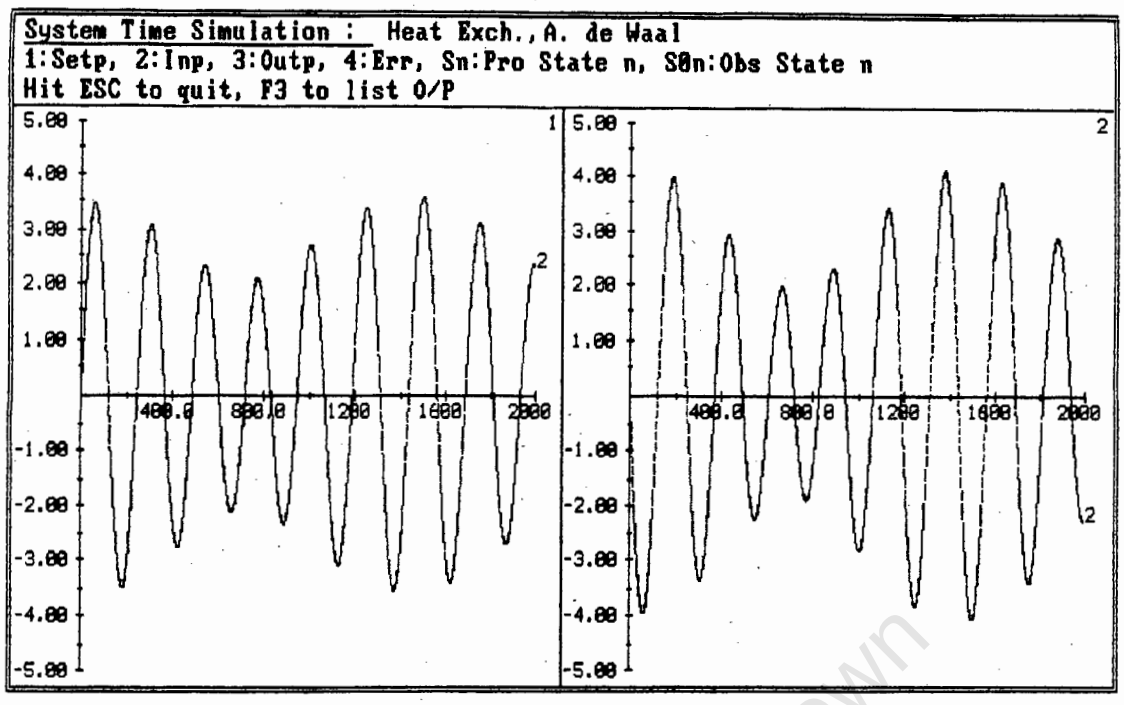


Figure O.7: System 2 - Simulated Response of Inputs to High Frequency Sensor Noise

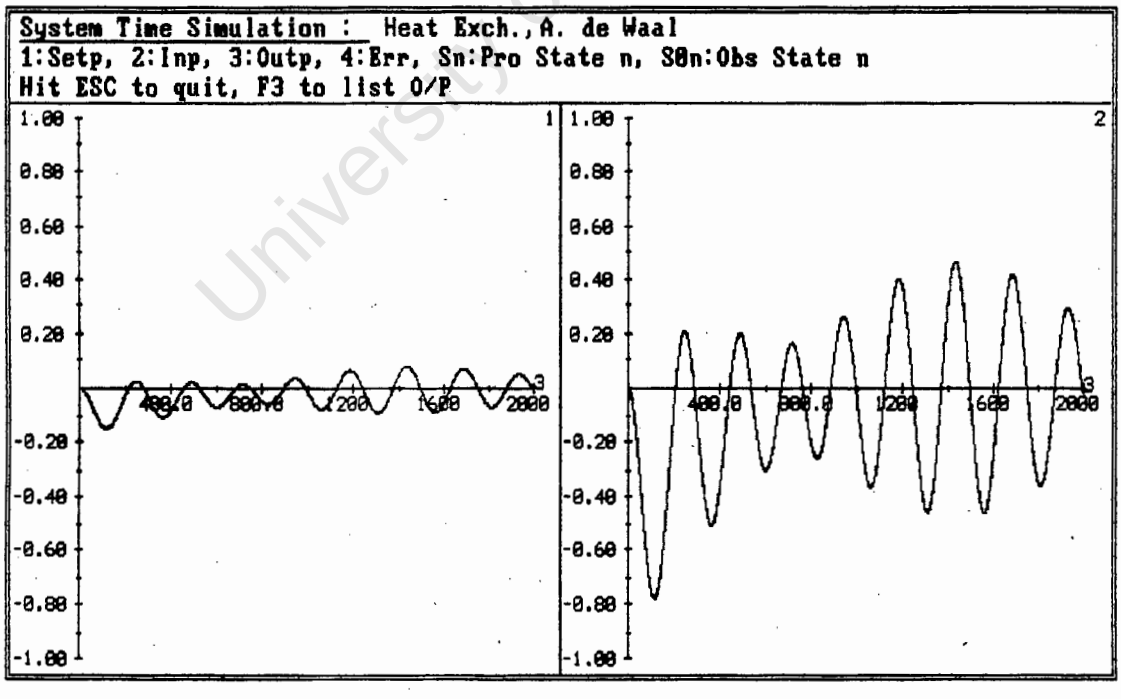


Figure O.8: System 2 - Simulated Response of Outputs to High Frequency Sensor Noise

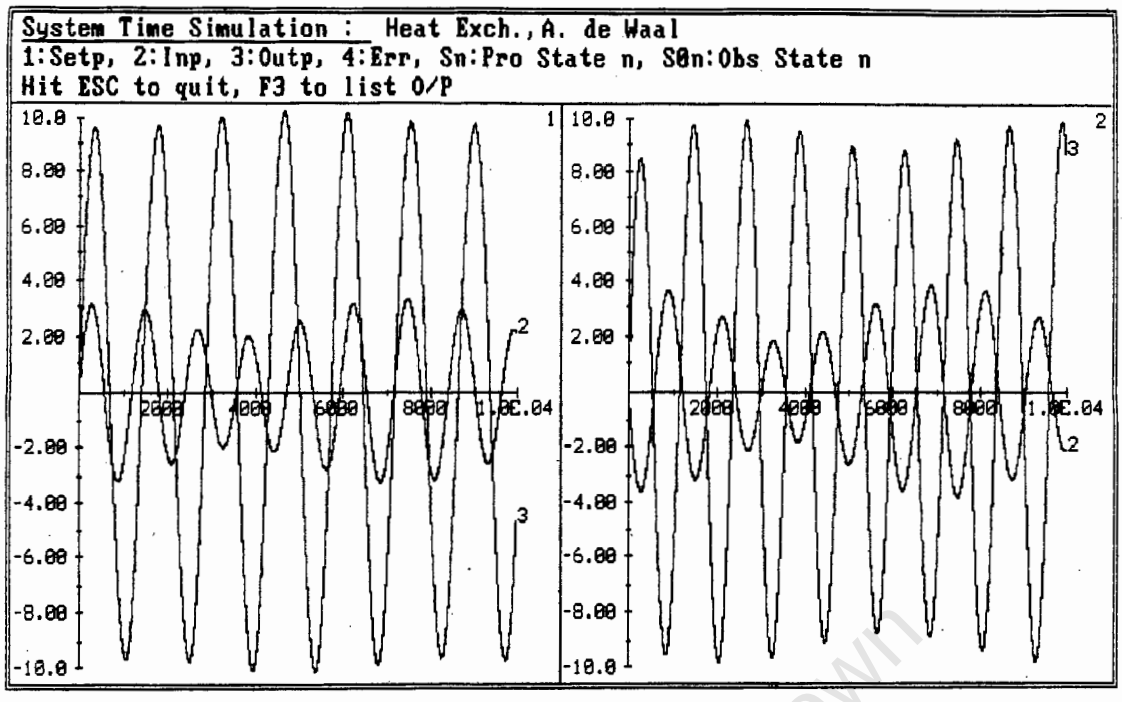


Figure O.9: System 2 - Simulated Response of Inputs to Low Frequency Disturbances

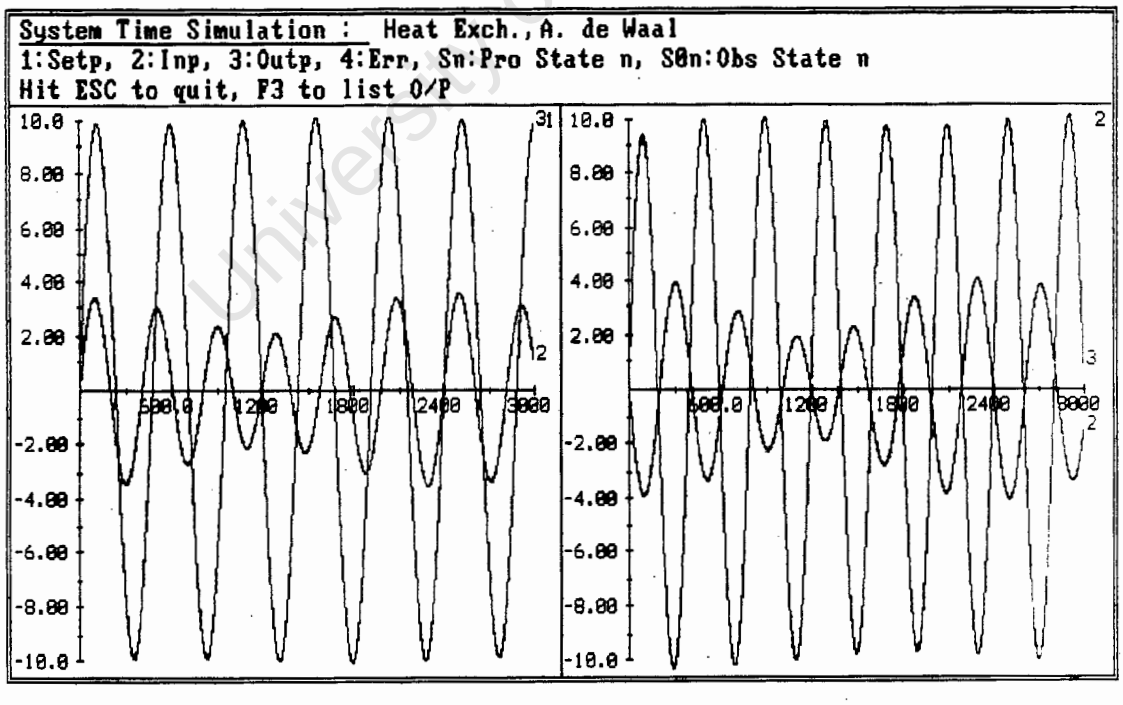


Figure O.10: System 2 - Simulated Response of Inputs to Interm. Frequency Disturbances

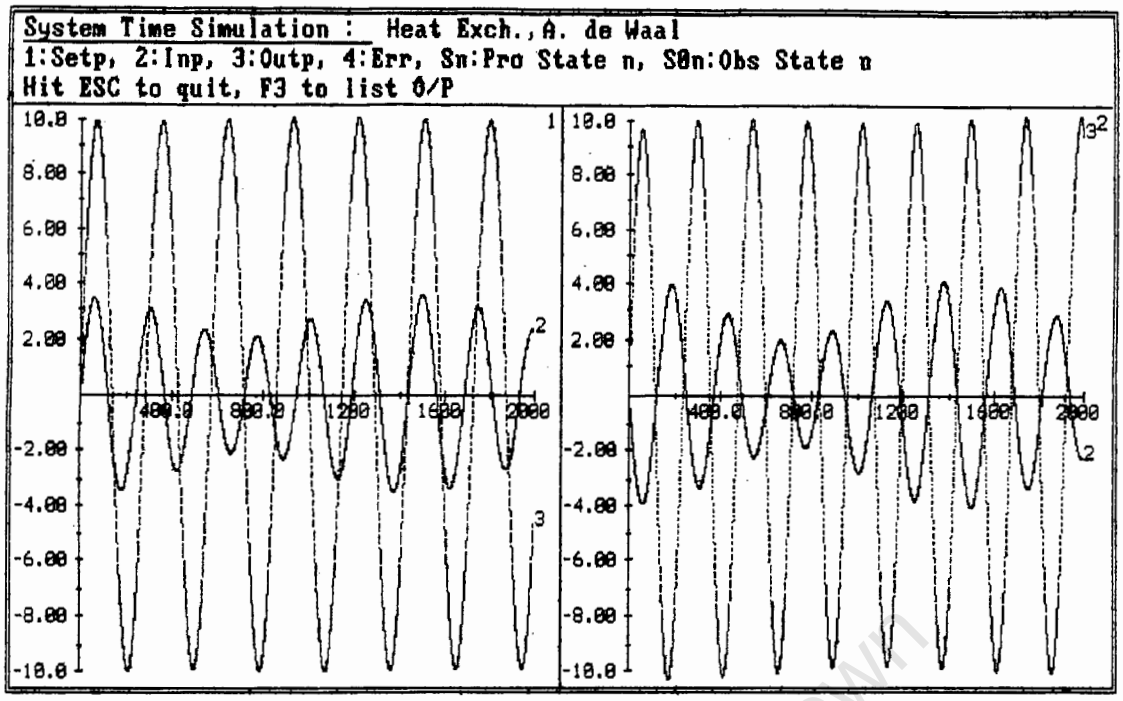


Figure O.11: System 2 - Simulated Response of Inputs to High Frequency Disturbances

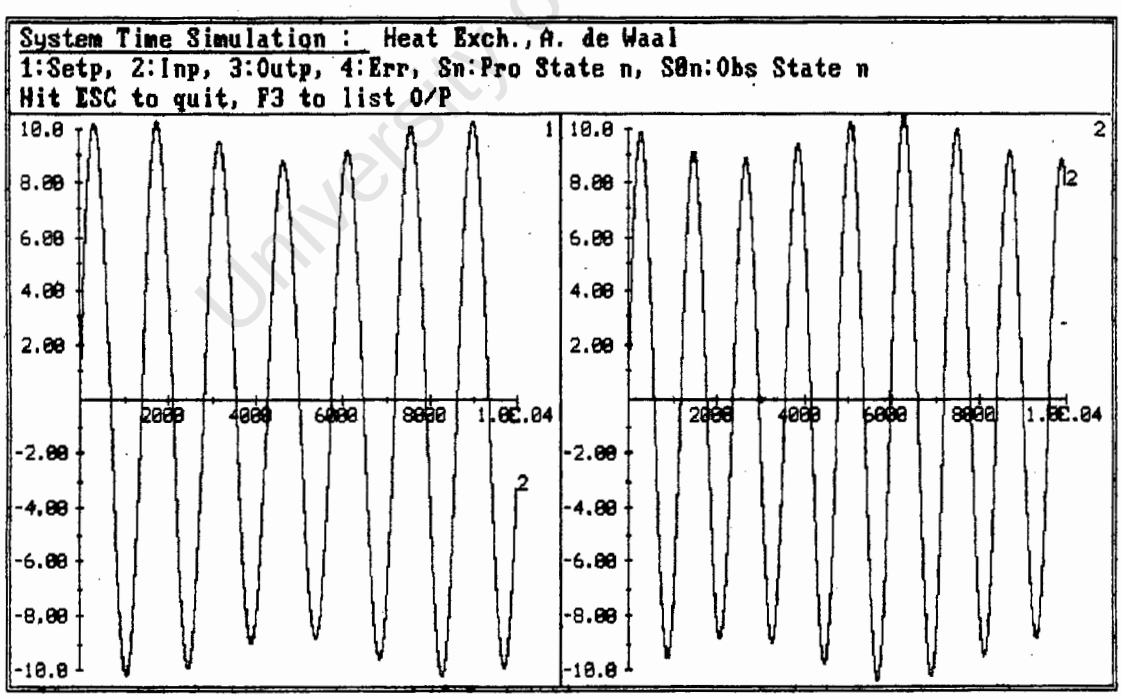


Figure O.12: System 2 - Simulated Response of Inputs to Low Frequency Setpoints

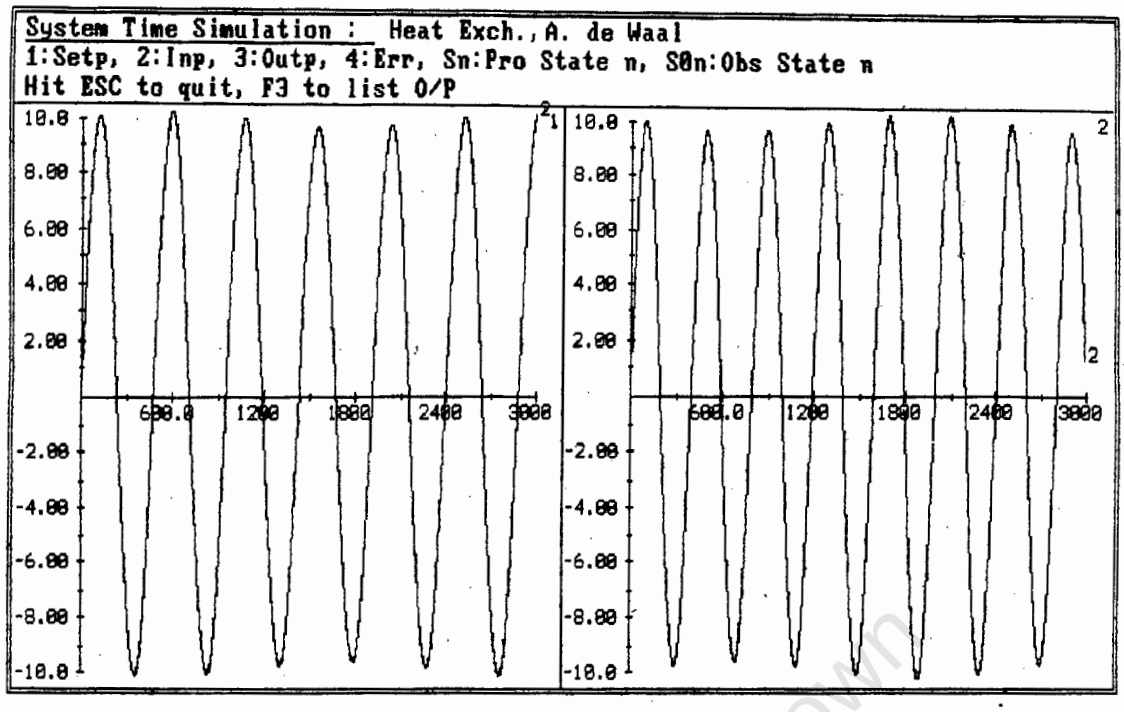


Figure O.13: System 2 - Simulated Response of Inputs to Interm. Frequency Setpoints

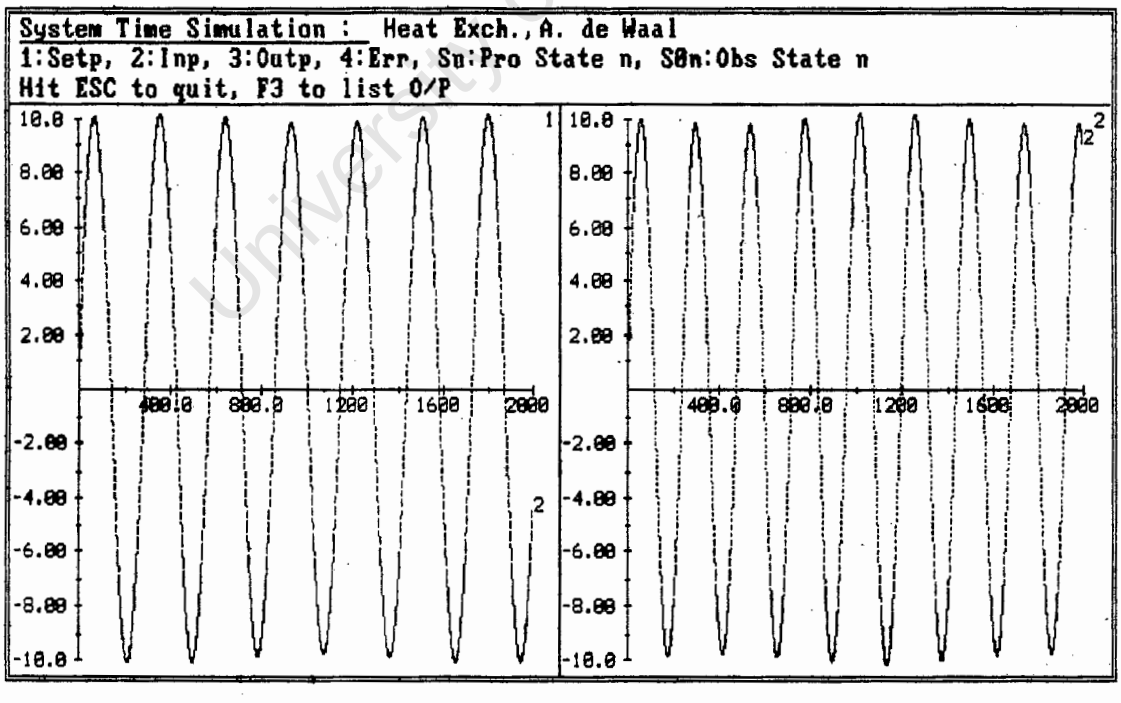


Figure O.14: System 2 - Simulated Response of Inputs to High Frequency Setpoints

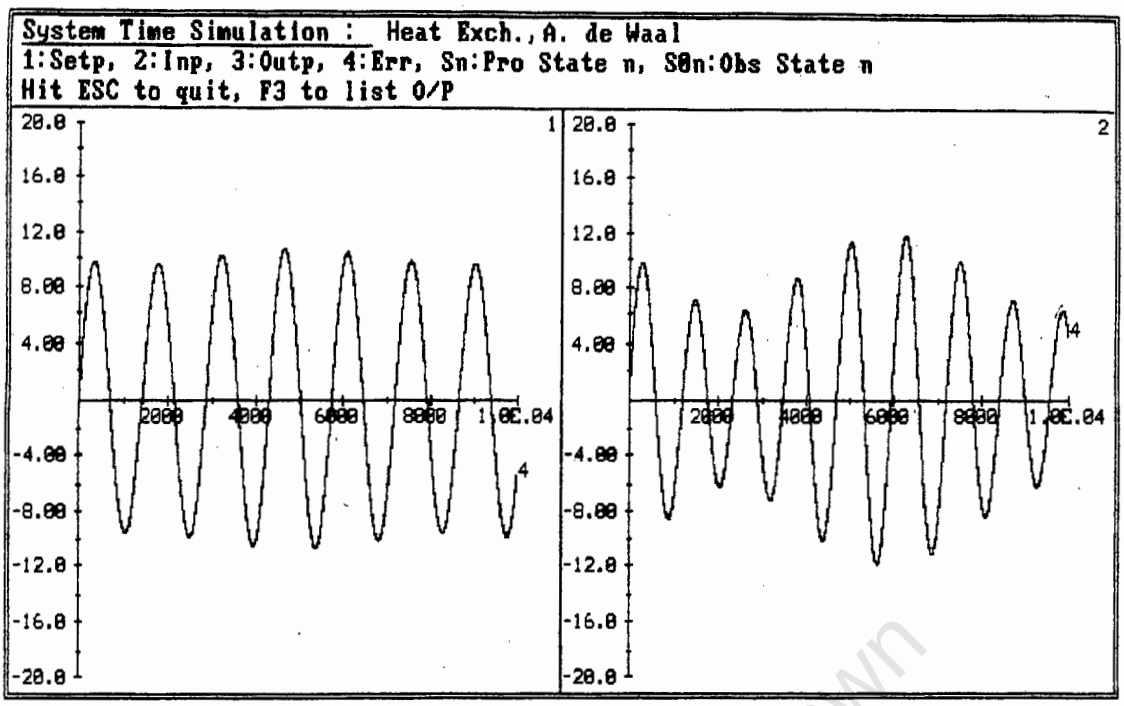


Figure O.15: System 2 - Simulated Response of Errors to Low Frequency Setpoints

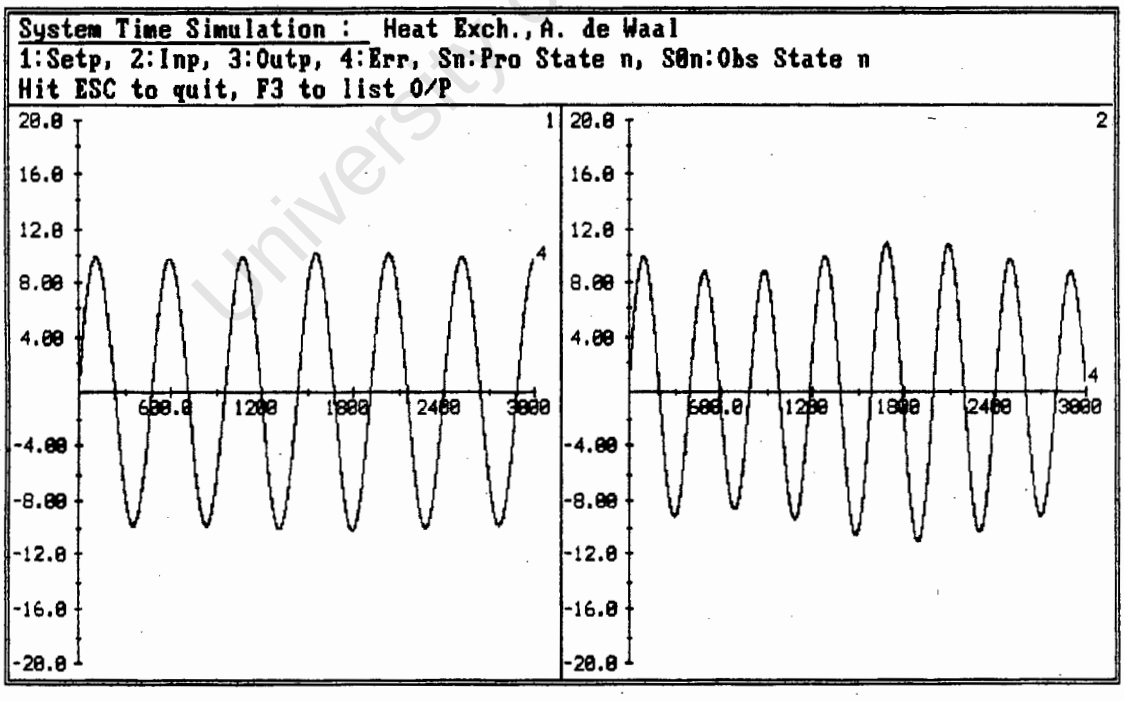


Figure O.16: System 2 - Simulated Response of Errors to Interm. Frequency Setpoints

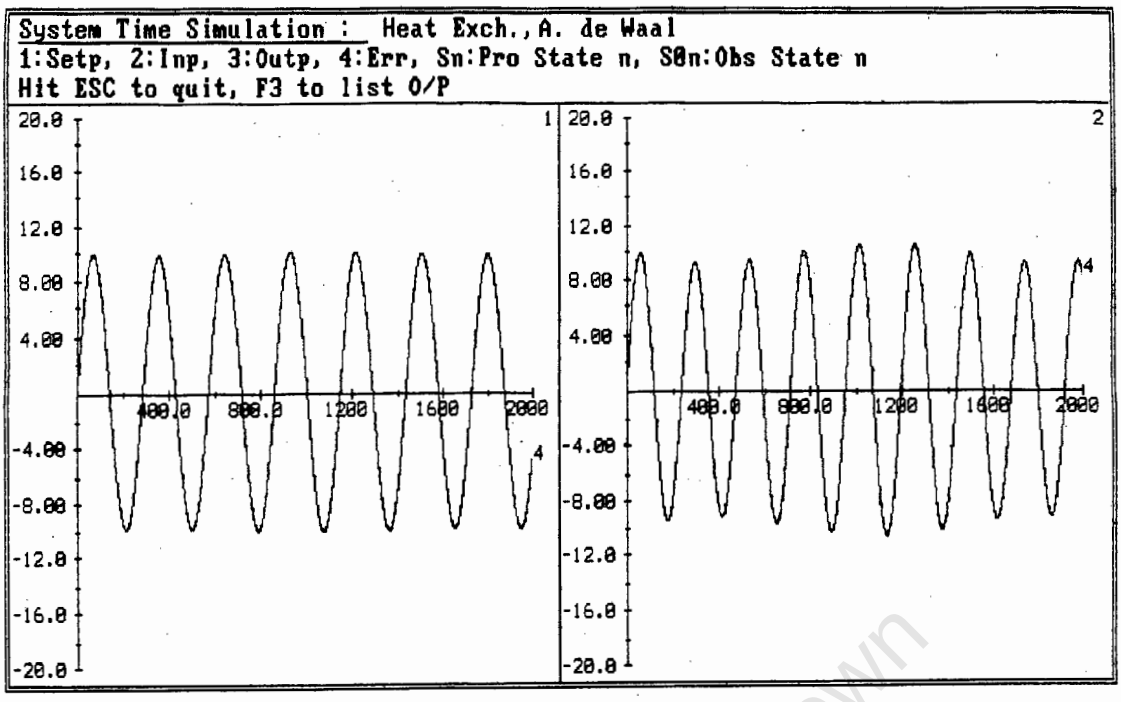


Figure O.17: System 2 - Simulated Response of Errors to High Frequency Setpoints

Q.3) Observed Process Input, Output and Error Responses

	Low Freq Band (0.00435 to 0.00522 rad/sec)		
	MR_{ch1} (units)	MR_{ch2} (units)	MR_{max}/A_{pert} (dB)
Dist. to Output D->Y	264.7	266.5	0.34
Dist. to Input D->U	83.6	97.7	-8.37
Noise to Output N->Y	12.5	67.7	-11.55
Noise to Input N->U	98.2	97.7	-8.32
Setp. to Error R->E	270.0	300.0	1.38
Setp. to Input R->U	267.7	267.7	0.64

Table Q.7: **Maximum Observed Amplitudes of Response, MR_{max} to Perturbations in Low Frequency Band with A_{pert} = 10 units.**
(Obtained from Figures Q.18 to Q.41)

	Intermediate Frequency Band (0.01305 to 0.01566 rad/sec)		
	MR_{ch1} (units)	MR_{ch2} (units)	MR_{max}/A_{pert} (dB)
Dist. to Output D->Y	257.7	284.1	0.90
Dist. to Input D->U	94.1	107.7	-7.52
Noise to Output N->Y	6.1	39.1	-16.32
Noise to Input N->U	94.7	108.9	-7.42
Setp. to Error R->E	259.4	308.9	1.63
Setp. to Input R->U	269.1	272.1	0.52

Table Q.8: **Maximum Observed Amplitudes of Response, MR_{max} to Perturbations in Intermediate Frequency Band with A_{pert} = 10 units. (Obtained from Figures Q.18 to Q.41)**

	High Freq Band (0.00435 to 0.00522 rad/sec)		
	MR_{ch1} (units)	MR_{ch2} (units)	MR_{max}/A_{pert} (dB)
Dist. to Output D->Y	257.7	275.3	0.63
Dist. to Input D->U	94.1	108.3	-7.47
Noise to Output N->Y	4.6	20.0	-22.14
Noise to Input N->U	95.9	110.0	-7.34
Setp. to Error R->E	257.7	300.0	1.38
Setp. to Input R->U	273.5	270.6	0.57

Table Q.9: **Maximum Observed Amplitudes of Response, MR_{max} to Perturbations in High Frequency Band with A_{pert} = 10 units.**
(Obtained from Figures Q.18 to Q.41)

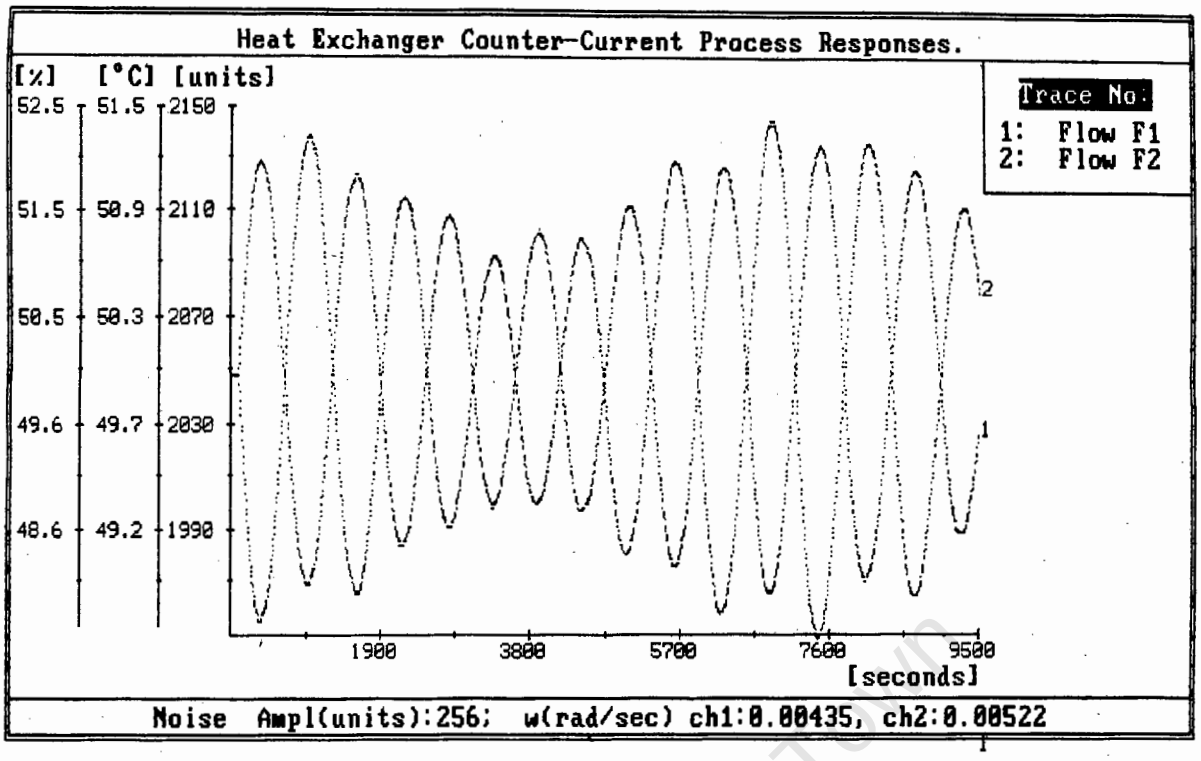


Figure Q.18: System 2 - Low Freq F1, F2 Response to Dist

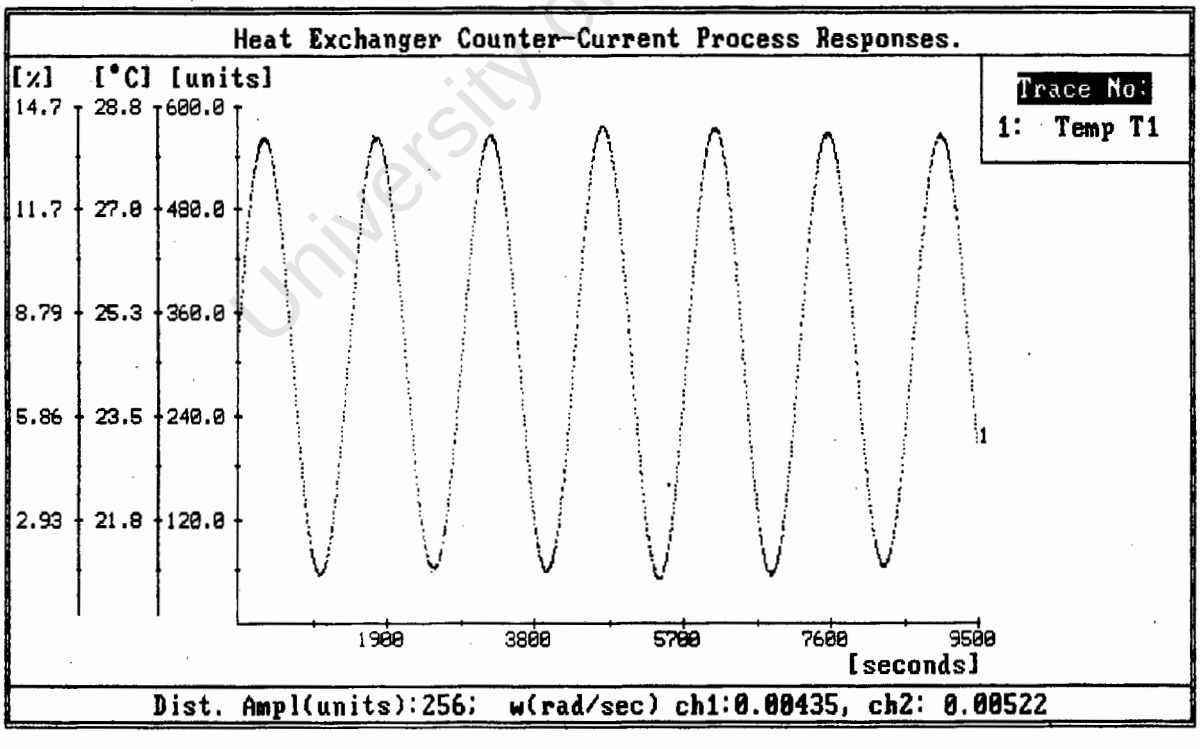


Figure Q.19: System 2 - Low Freq T1 Response to Dist

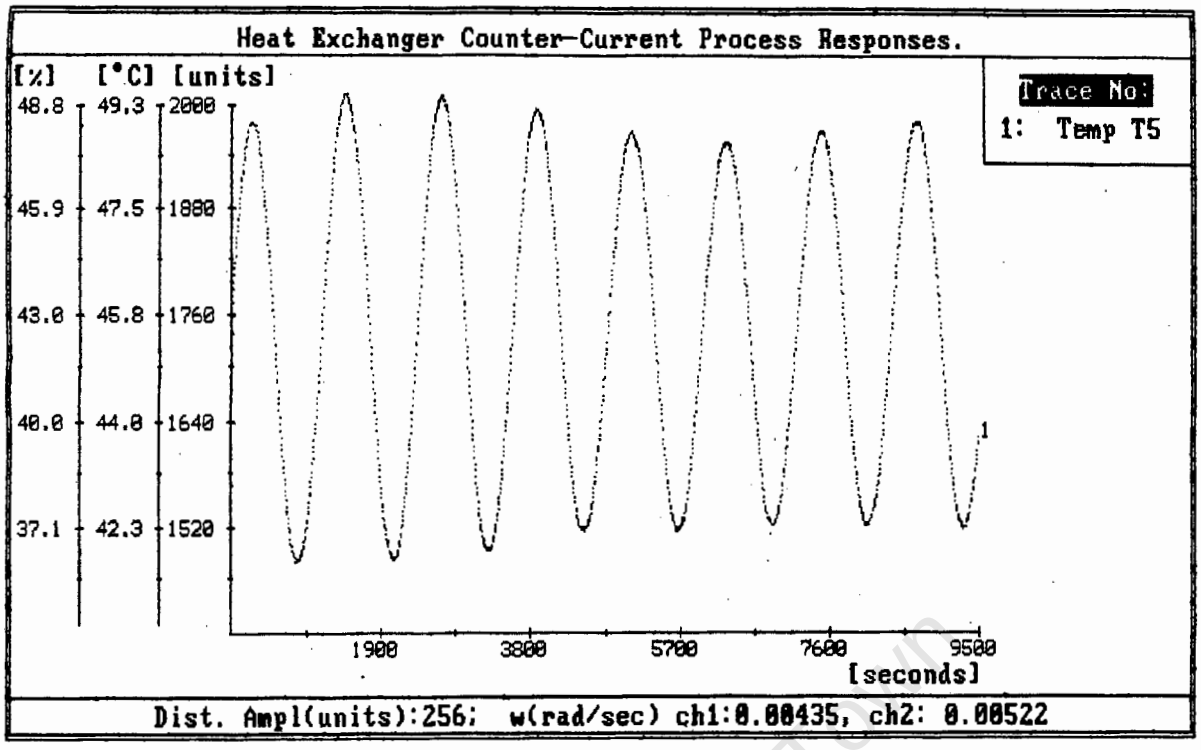


Figure Q.20: System 2 - Low Freq T5 Response to Dist

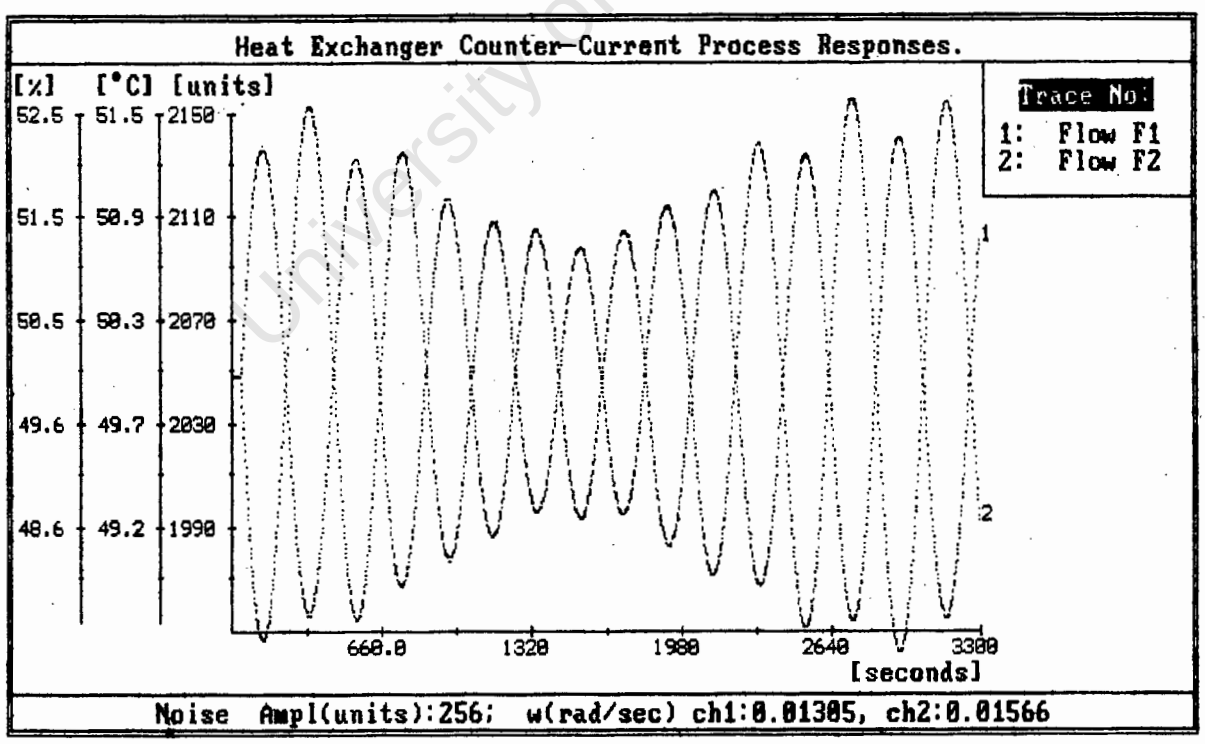


Figure Q.21: System 2 - Int Freq F1, F2 Response to Dist

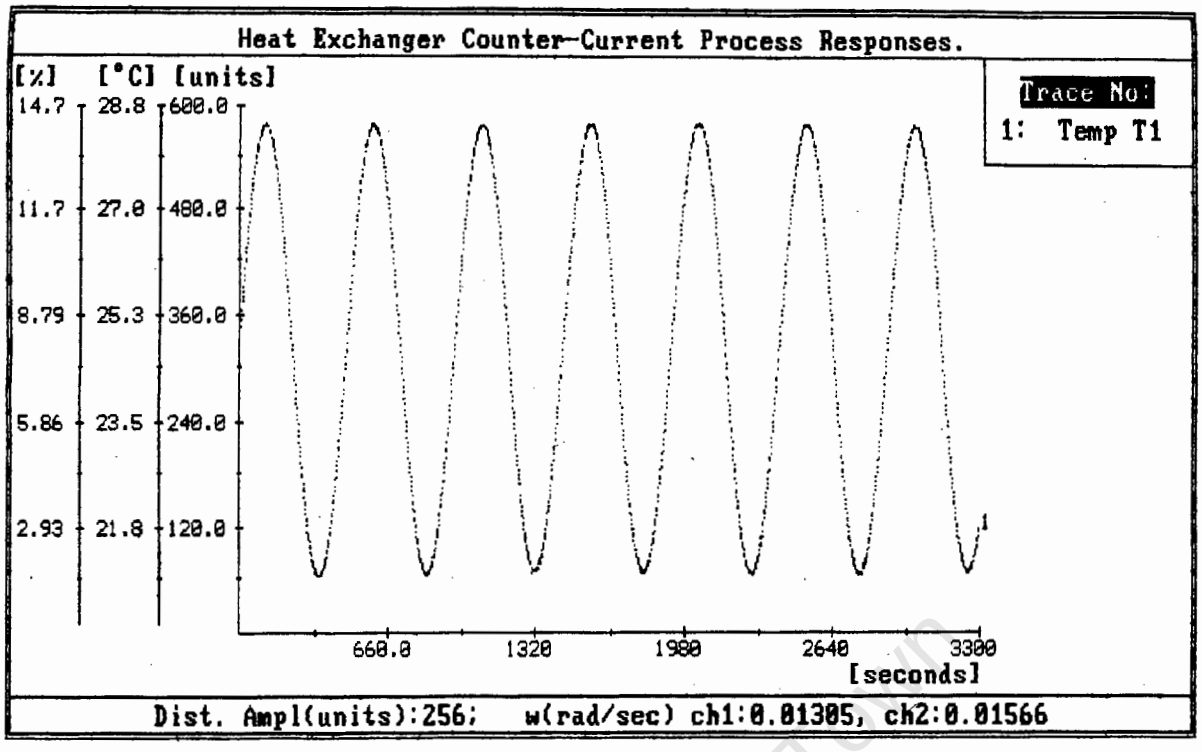


Figure Q.22: System 2 - Int Freq T1 Response to Dist

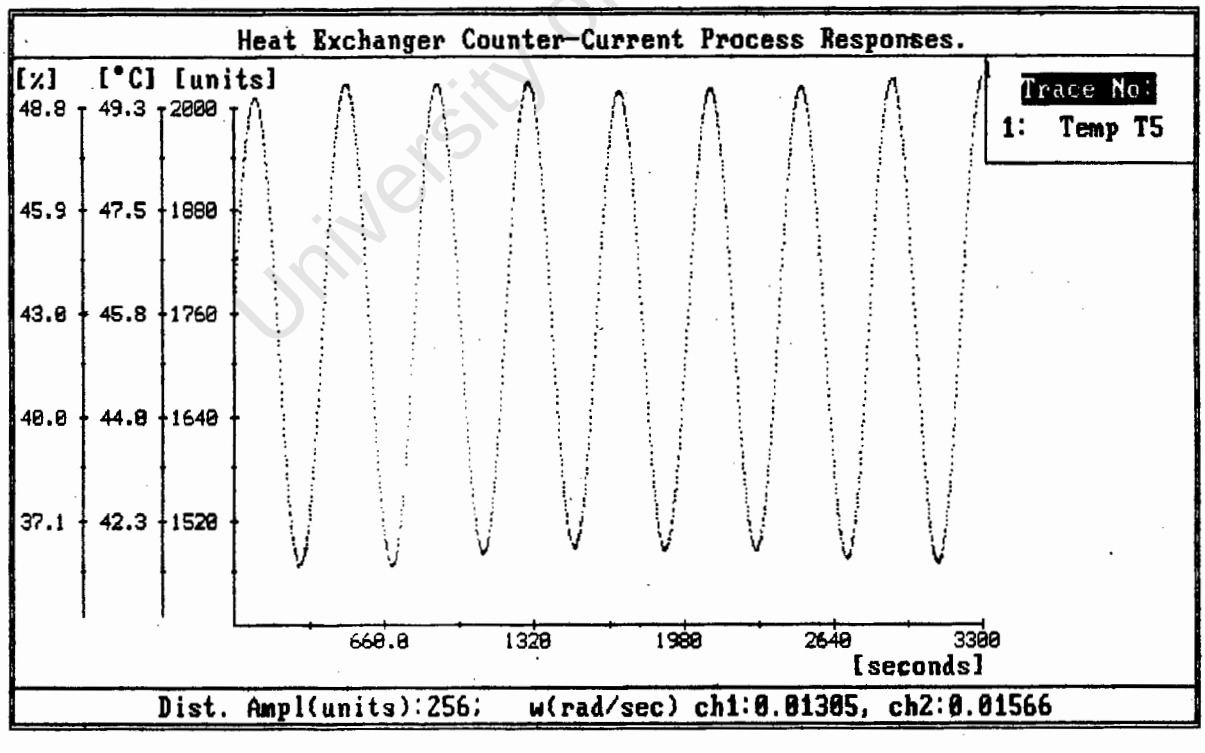


Figure Q.23: System 2 - Int Freq T5 Response to Dist

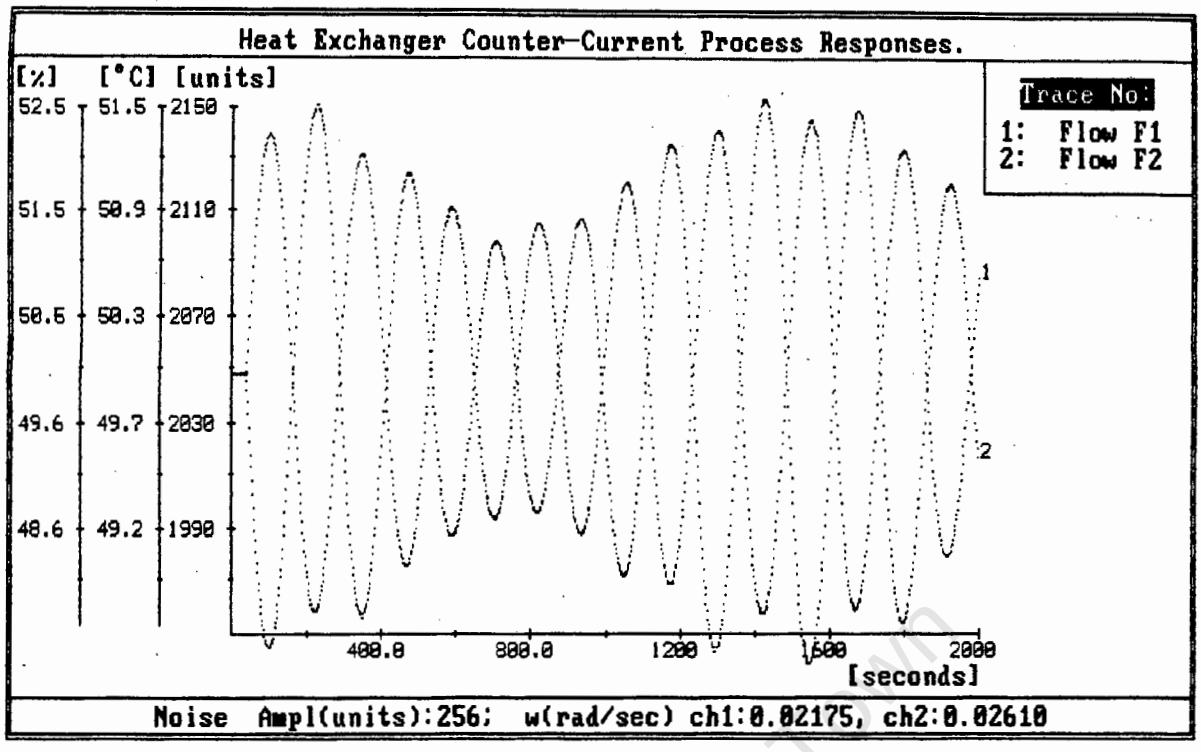


Figure Q.24: System 2 - High Freq F1, F2 Response to Dist

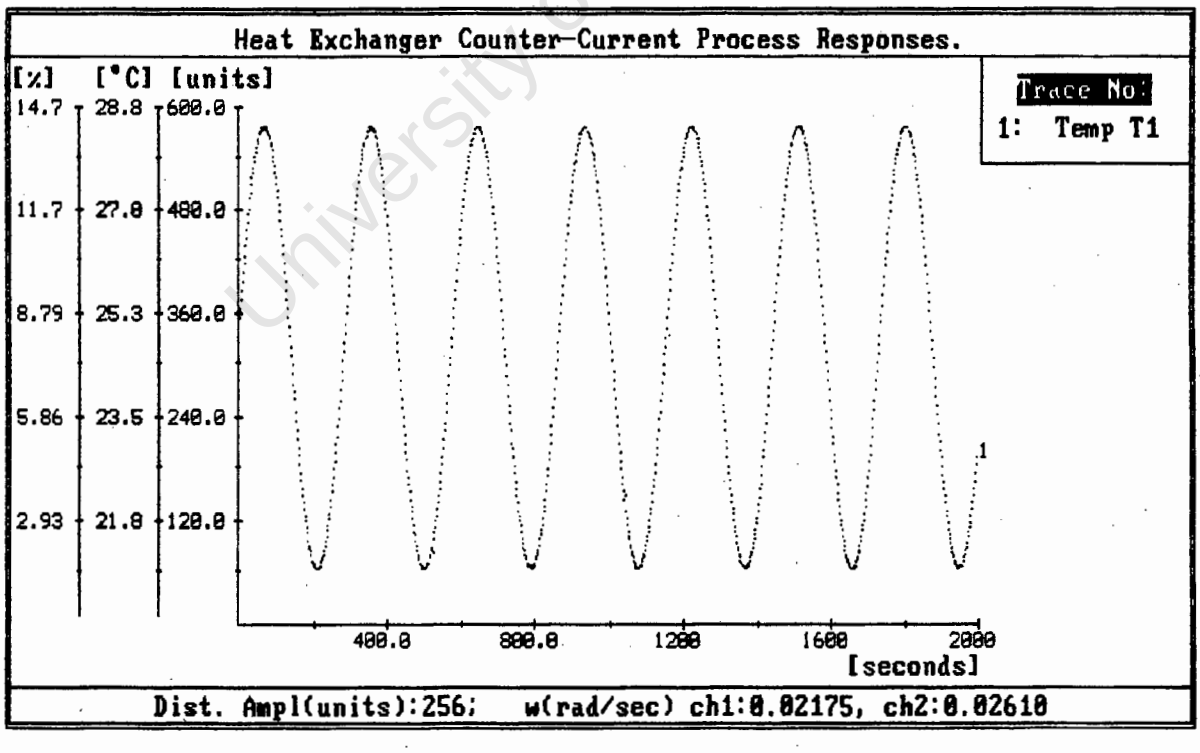


Figure Q.25: System 2 - High Freq T1 Response to Dist

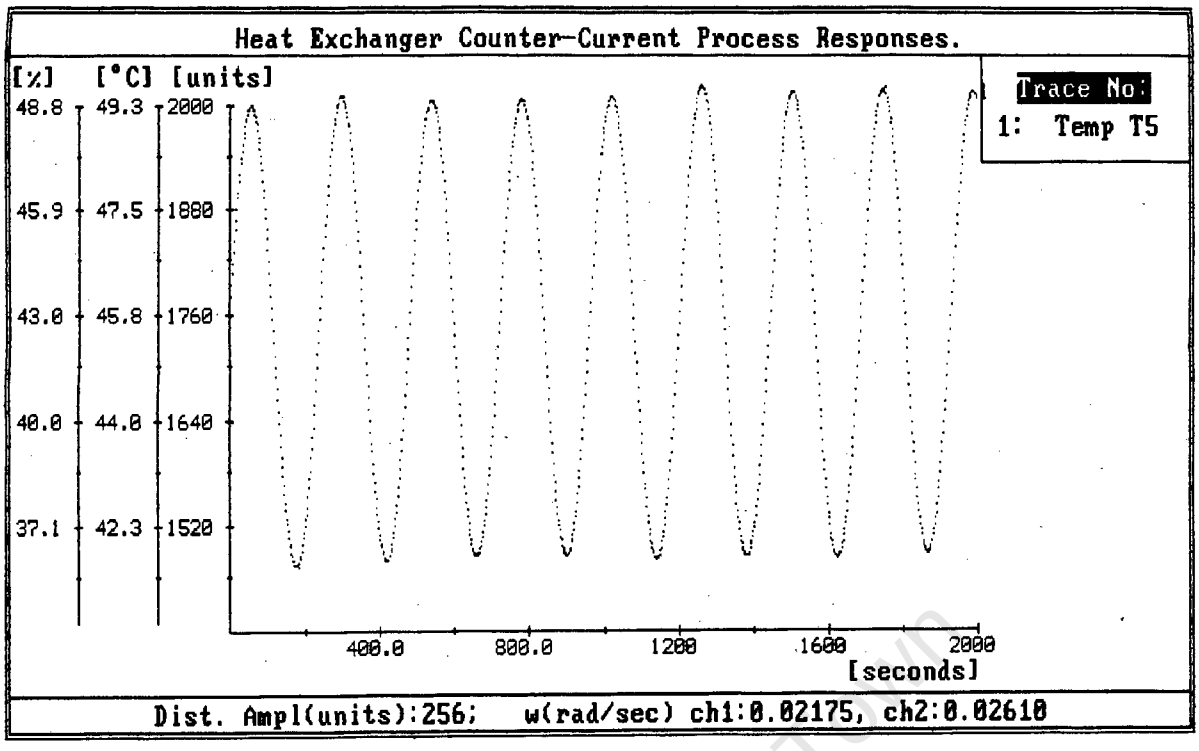


Figure Q.26: System 2 - High Freq T5 Response to Dist

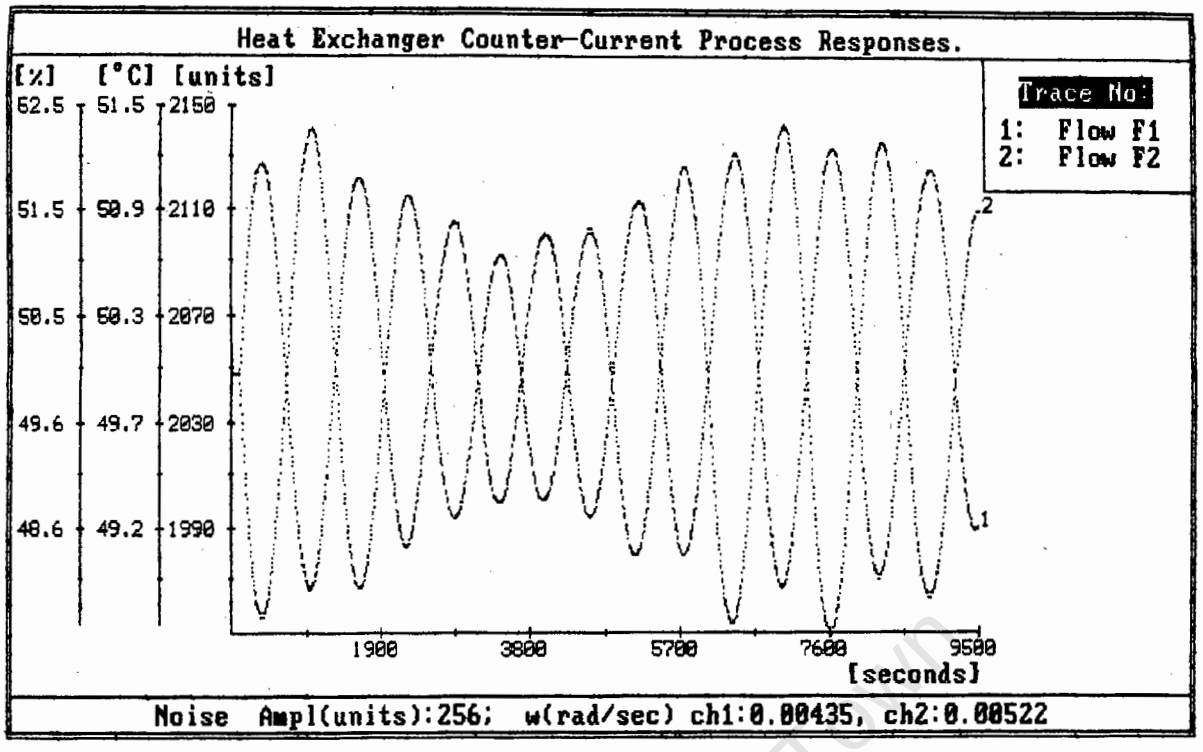


Figure Q.27: System 2 - Low Freq F1, F2 Response to Noise

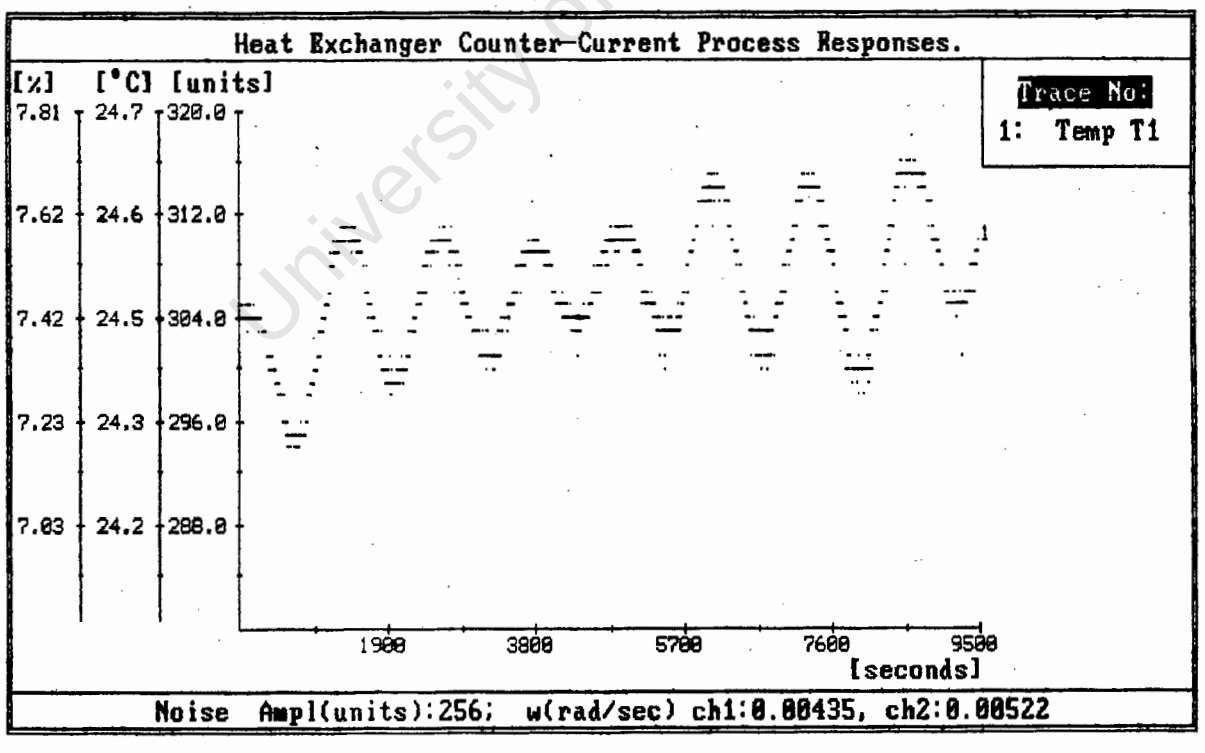


Figure Q.28: System 2 - Low Freq T1 Response to Noise

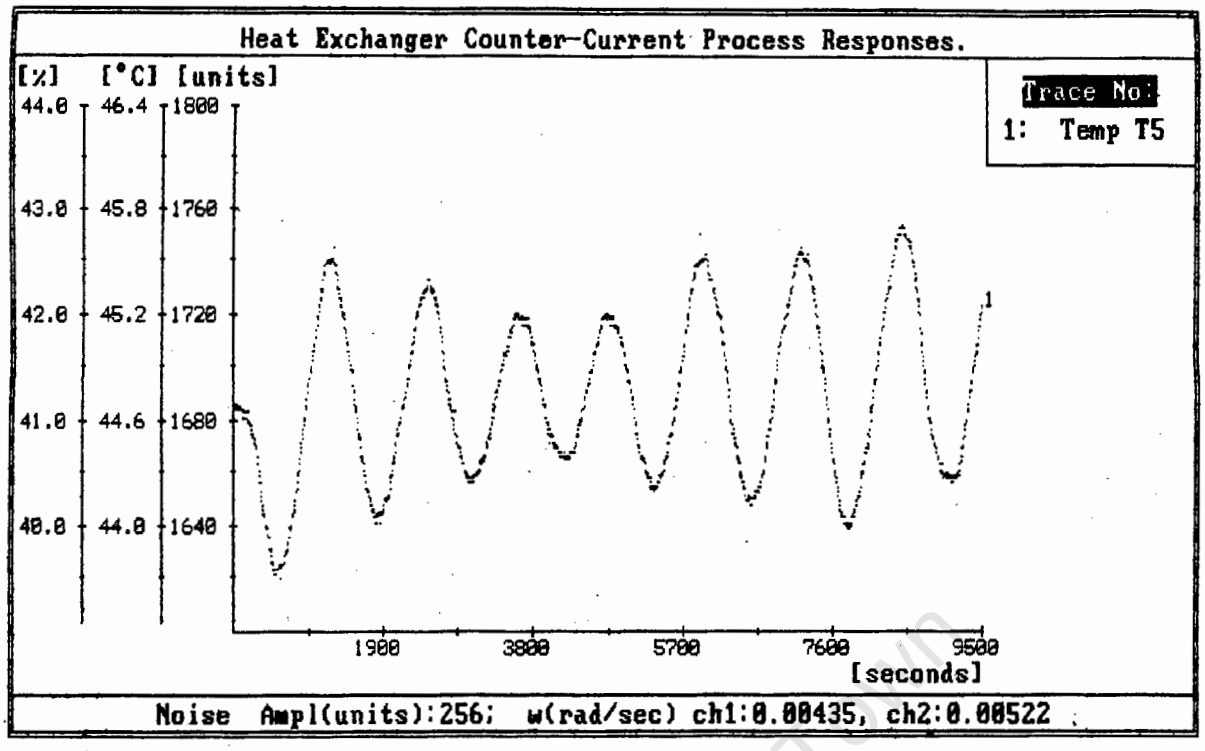


Figure O.29: System 2 - Low Freq T5 Response to Noise

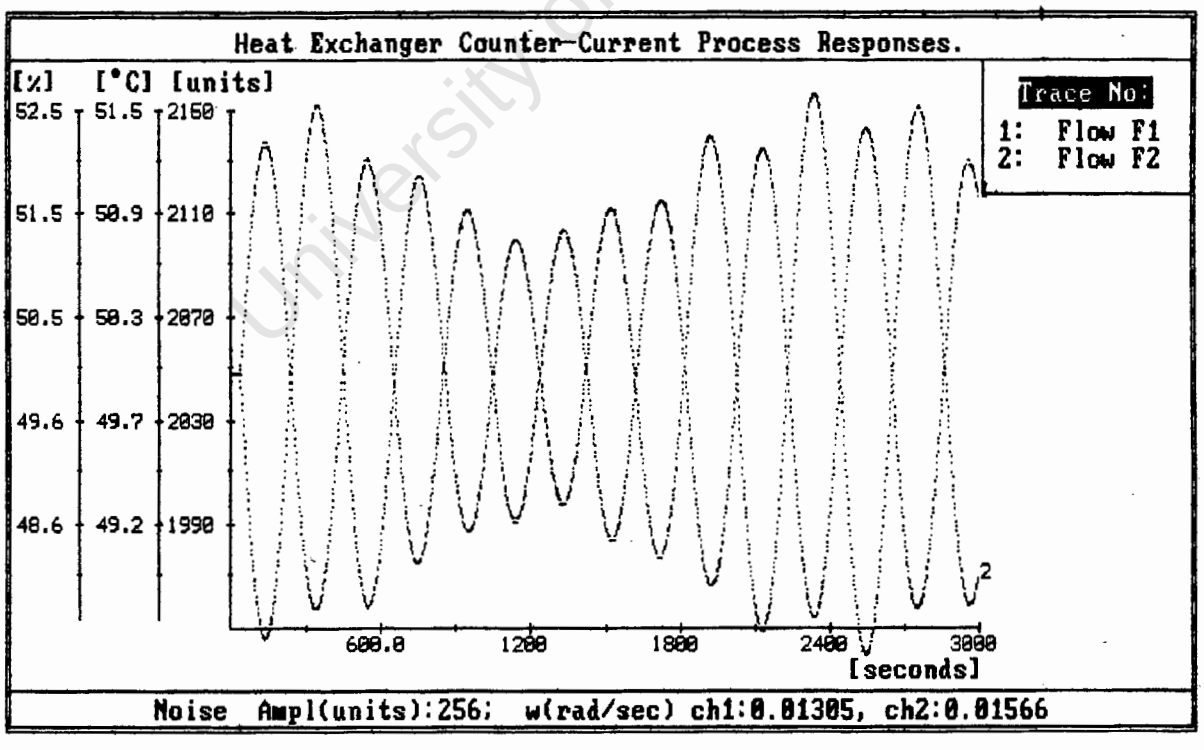


Figure O.30: System 2 - Int Freq F1, F2 Response to Noise

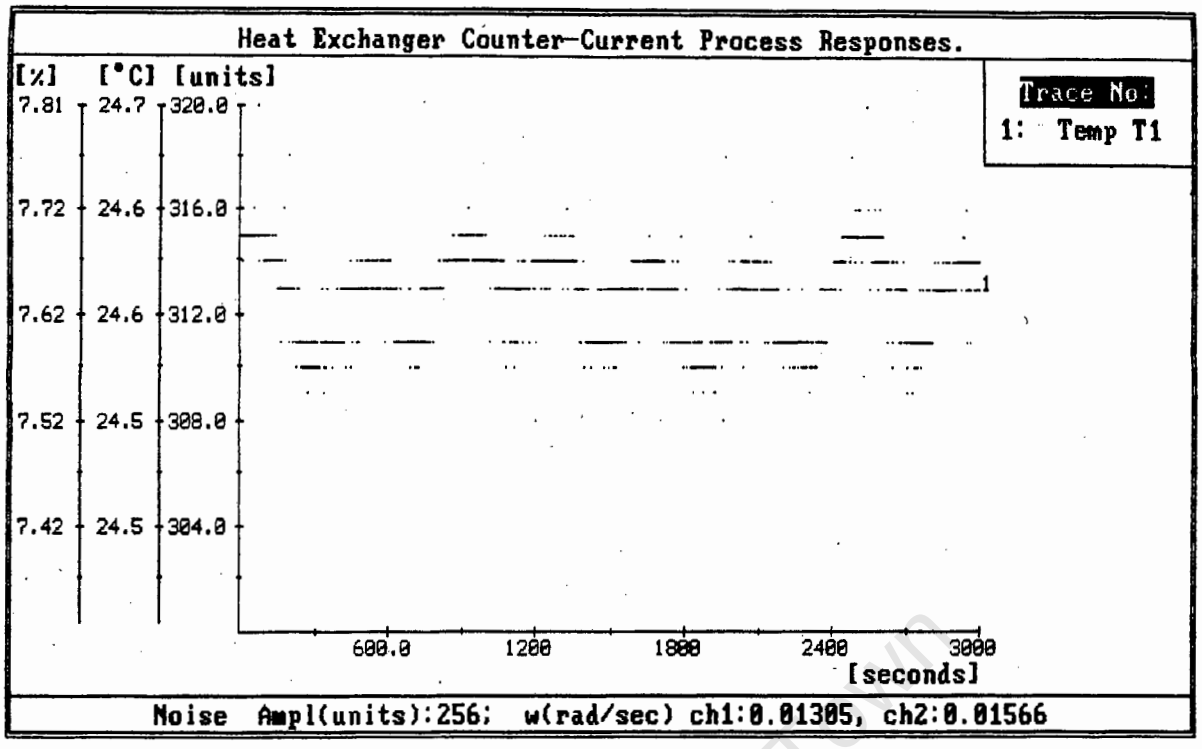


Figure Q.31: System 2 - Int Freq T1 Response to Noise

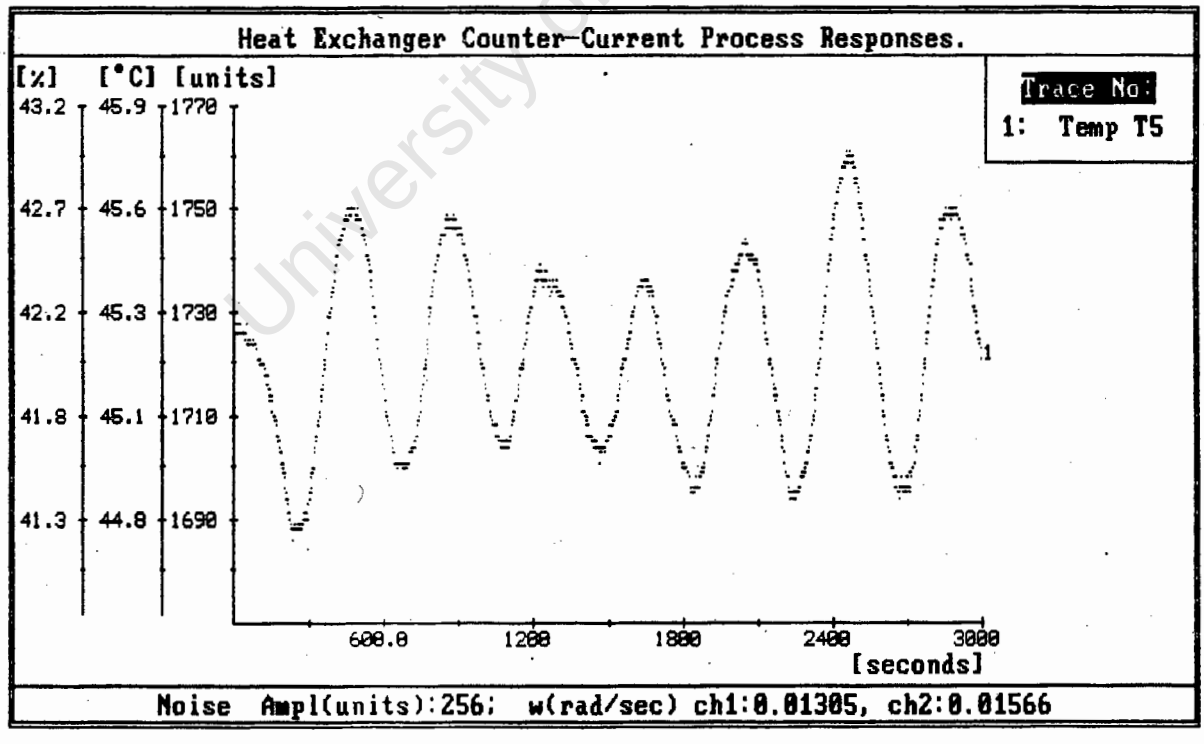


Figure Q.32: System 2 - Int Freq T5 Response to Noise

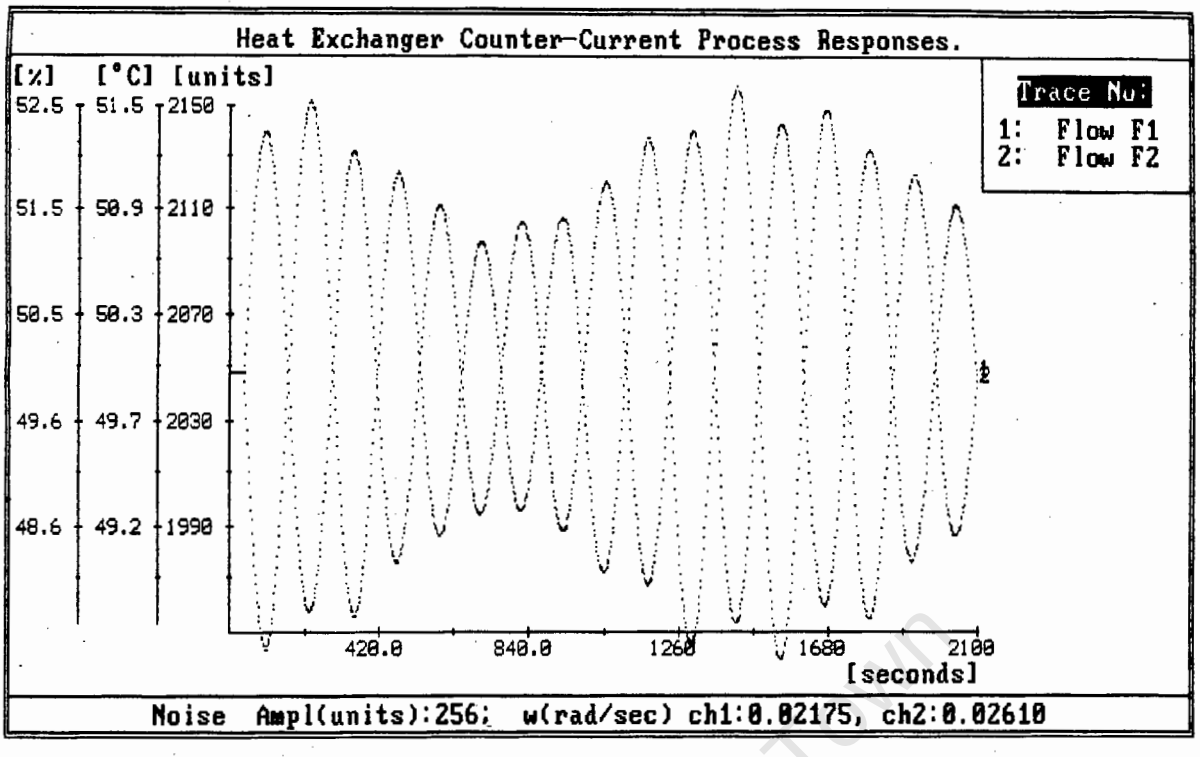


Figure O.33: System 2 - High Freq F1, F2 Response to Noise

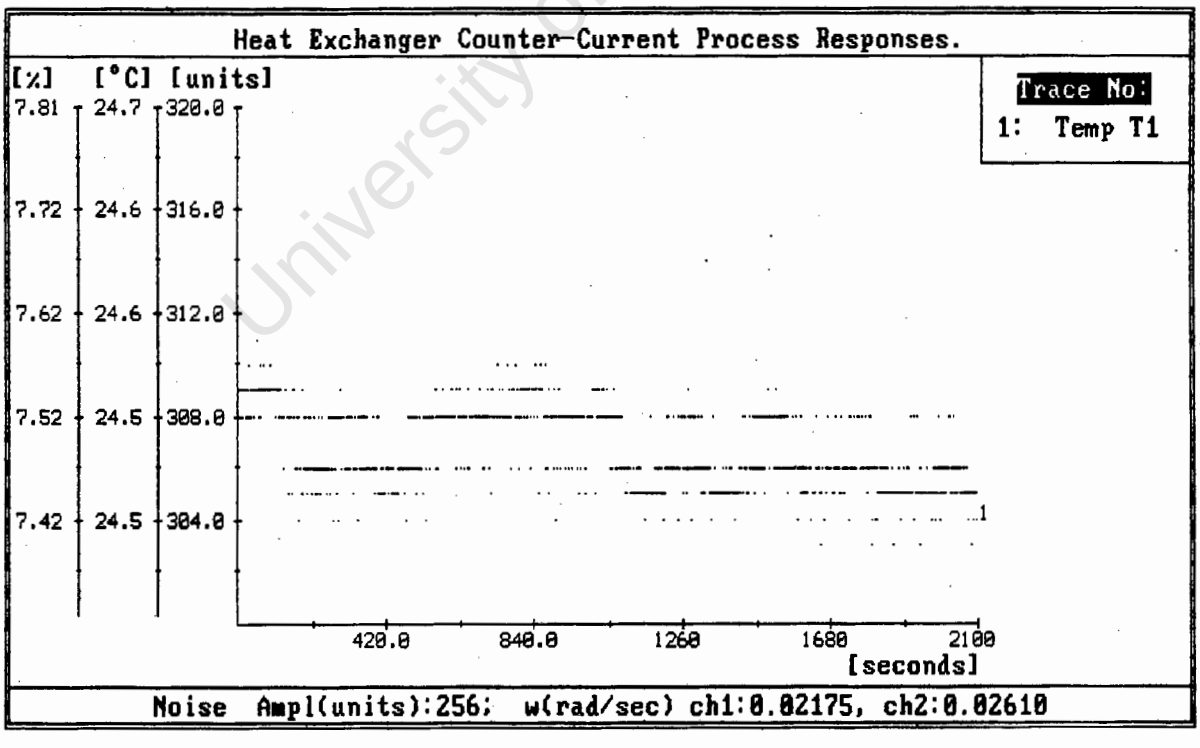


Figure O.34: System 2 - High Freq T1 Response to Noise

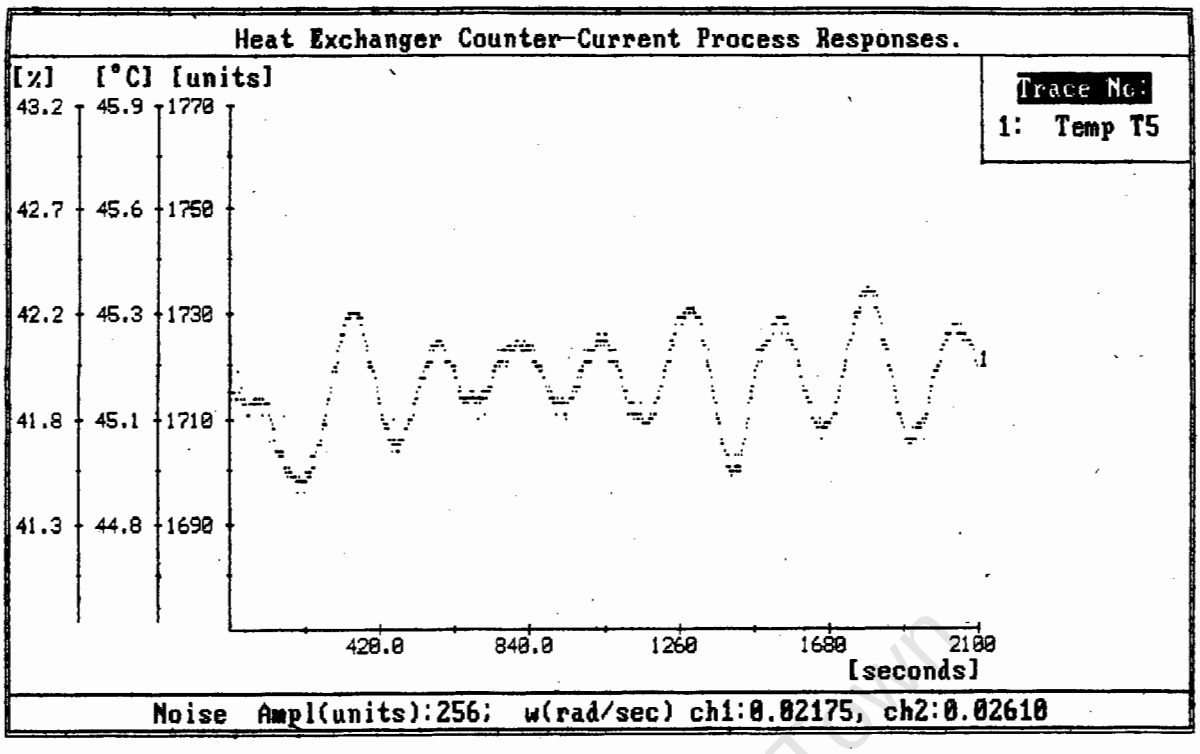


Figure Q.35: System 2 - High Freq T5 Response to Noise

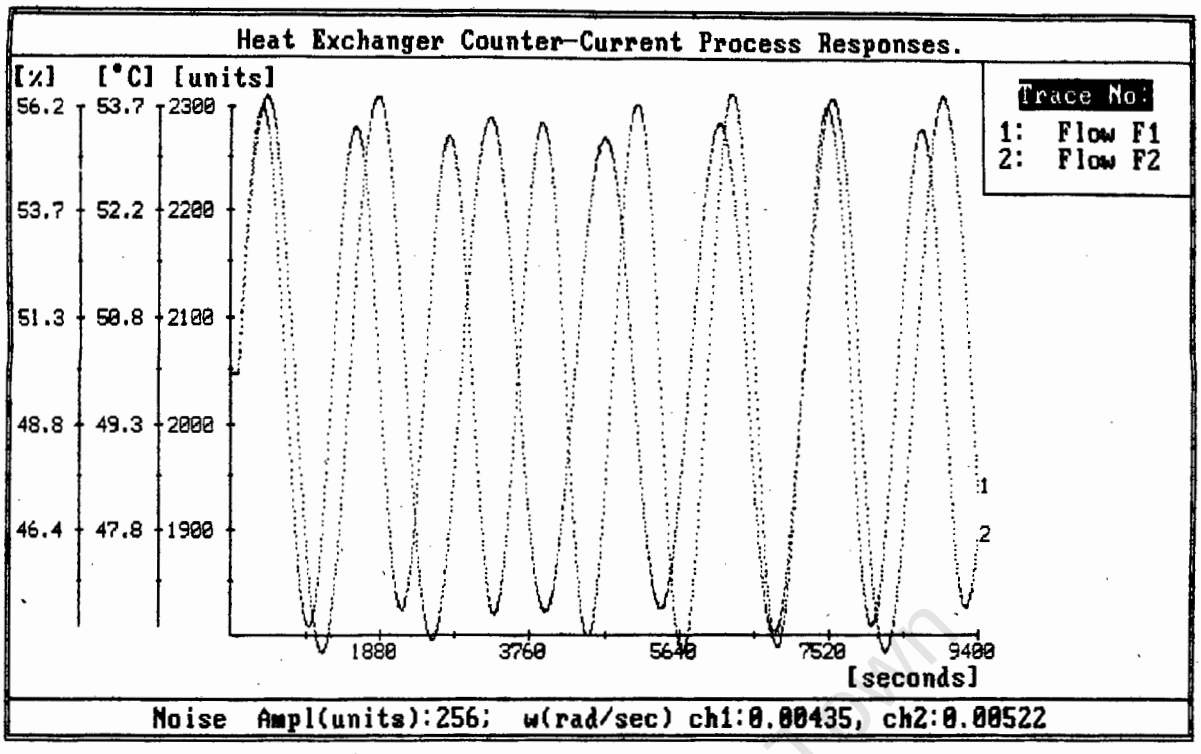


Figure Q.36: System 2 - Low Freq F1, F2 Response to Setp

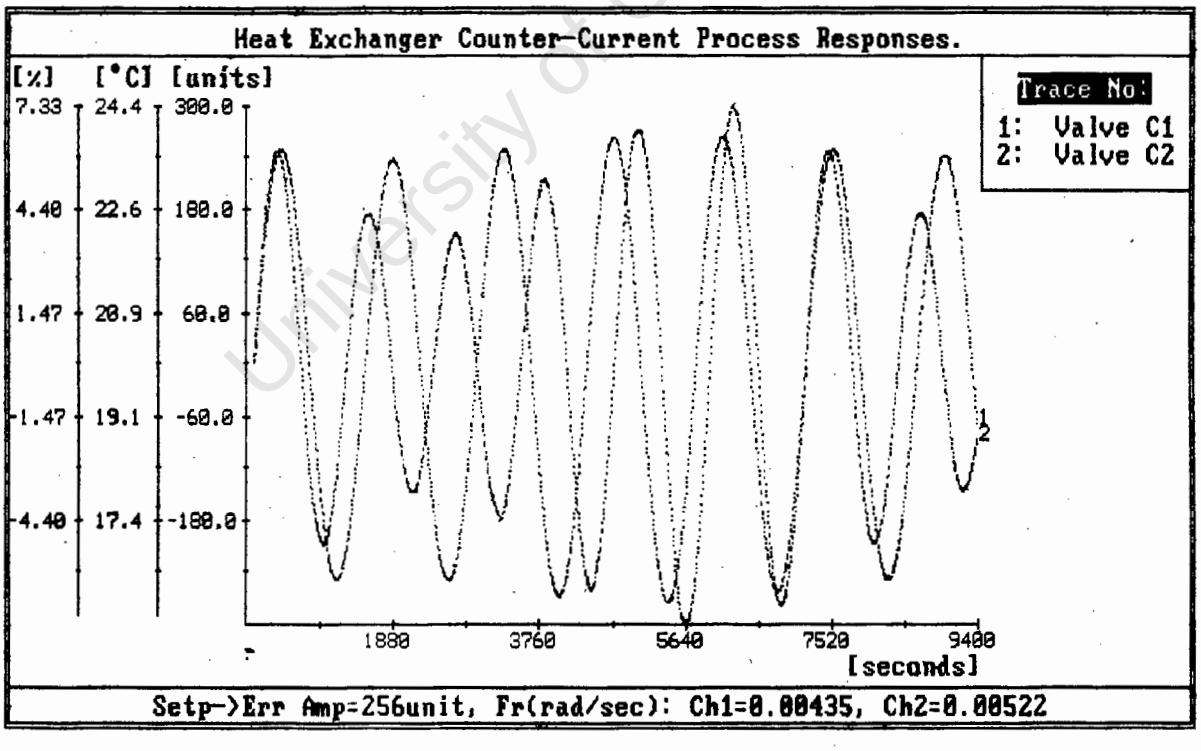


Figure Q.37: System 2 - Low Freq Error Response to Setp

Trace 1: Err(T1); Trace 2: Err(T5)

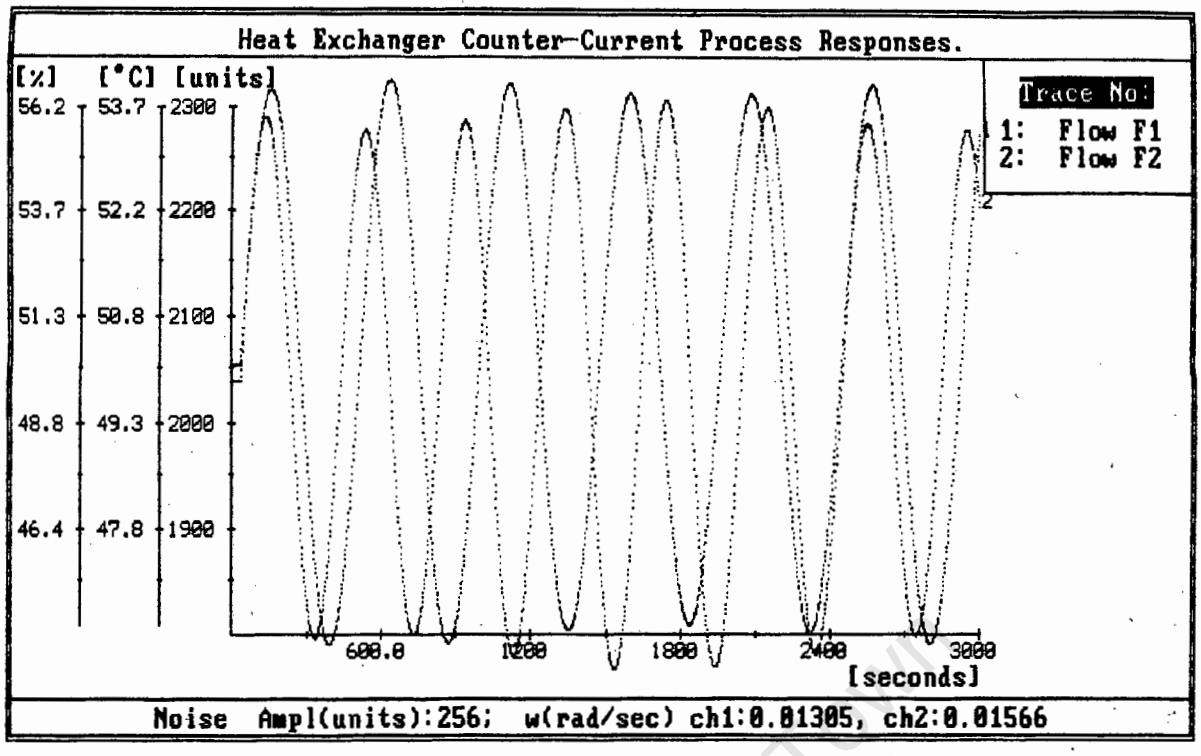


Figure O.38: System 2 - Int Freq F1, F2 Response to Setp

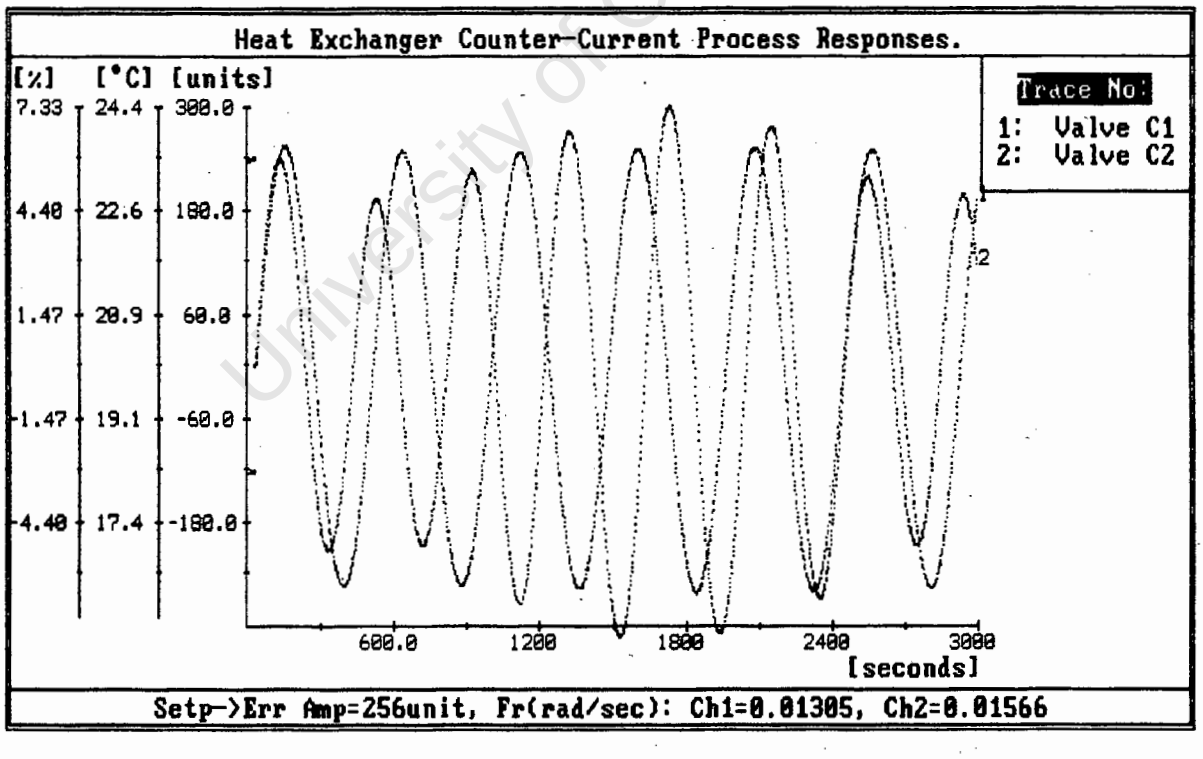


Figure O.39: System 2 - Interm Freq Error Response to Setp

Trace 1: Err(T1); Trace 2: Err(T5)

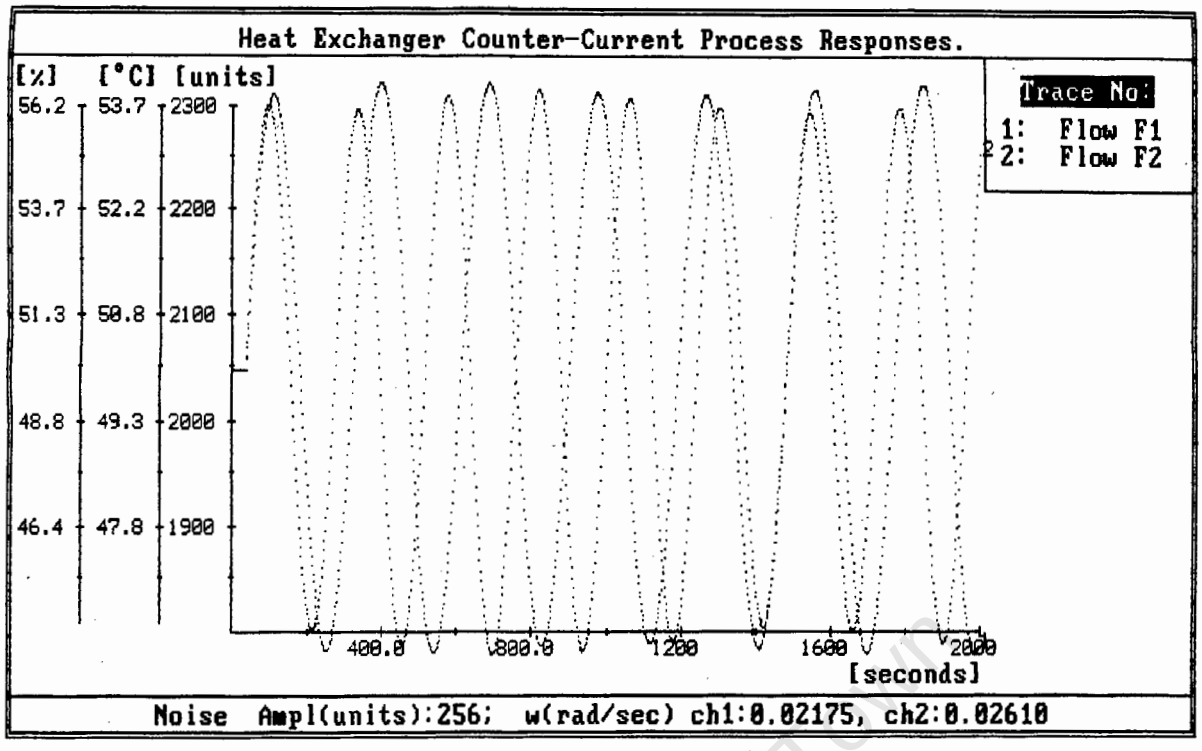


Figure Q.40: System 2 - High Freq F1, F2 Response to Setp

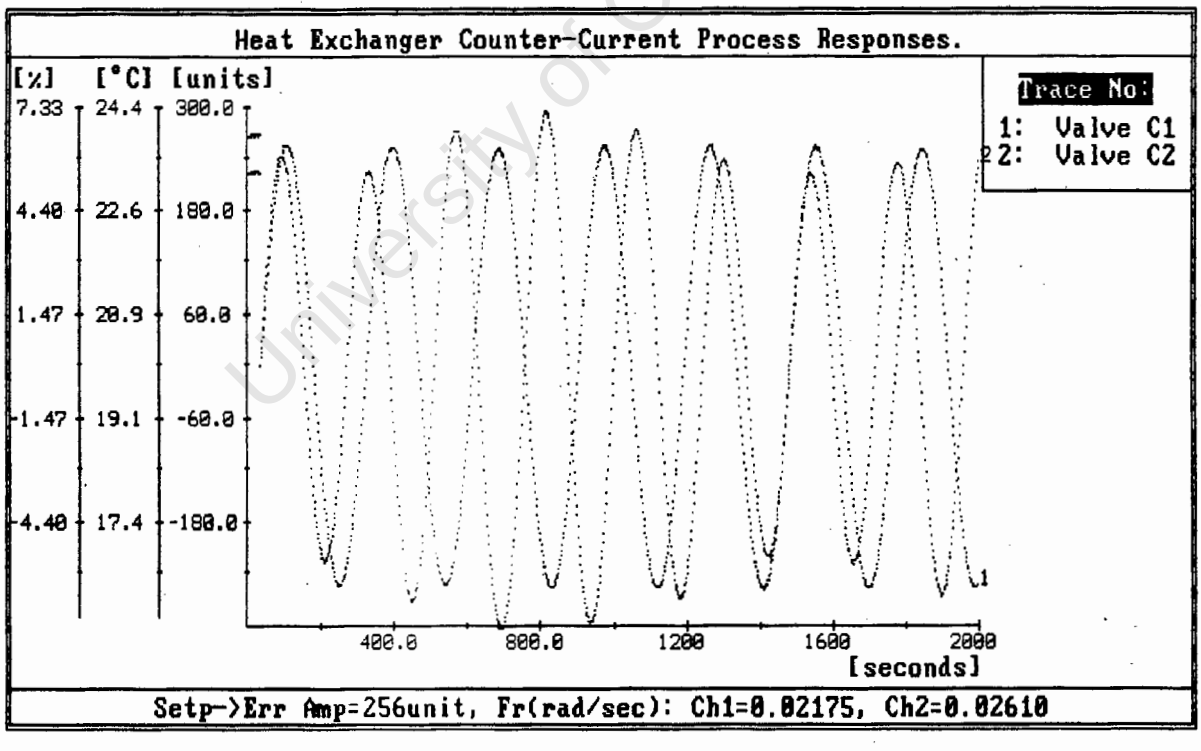


Figure Q.41: System 2 - High Freq Error Response to Setp

Trace 1: Err(T1); Trace 2: Err(T5)

APPENDIX R

VERIFICATION OF PERFORMANCE INDEX DATA: SYSTEM 3

R.1) Individual Performance Index Plots and Analysis

	Low Freq Band (x 10^{-3} rad/sec)		
	4.35	5.22	Maximum
$\|T_{YD}(s)\|_{max}$ (dB)	-5.4	-4.2	-4.2
$\|T_{UD}(s)\|_{max}$ (dB)	34.2	35.0	35.0
$\|T_{YN}(s)\|_{max}$ (dB)	0.9	0.9	0.9
$\|T_{UN}(s)\|_{max}$ (dB)	34.2	35.0	35.0
$\|SE_Y(s)\|_{max}$ (dB)	-5.4	-4.2	-4.2
$\|SE_U(s)\|_{max}$ (dB)	0.2	0.0	0.2
$\|ERR(s)\|_{max}$ (dB)	0.0	0,0	0.0
$\|T_{UR}(s)\|_{max}$ (dB)	-6.7	-6.0	-6.0

Table R.1: **Values of Each Performance Index at Two Indicated Frequencies in Low Frequency Band and the Maximum of these two values. (Obtained from Figures R1 to R4)**

	Intermediate Freq Band (x 10^{-2} rad/sec)		
	1.31	1.57	Maximum
$\|T_{YD}(s)\|_{max}$ (dB)	0.0	1.1	1.1
$\|T_{UD}(s)\|_{max}$ (dB)	39.5	40.0	40.0
$\|T_{YN}(s)\|_{max}$ (dB)	1.8	1.8	1.8
$\|T_{UN}(s)\|_{max}$ (dB)	39.5	40.0	40.0
$\|SE_Y(s)\|_{max}$ (dB)	0.0	1.1	1.1
$\|SE_U(s)\|_{max}$ (dB)	0.0	0.0	0.0
$\|ERR(s)\|_{max}$ (dB)	0.0	0.0	0.0
$\|T_{UR}(s)\|_{max}$ (dB)	-2.0	-1.1	-1.1

Table R.2: **Values of Each Performance Index at Two Indicated Frequencies in Interm. Frequency Band and the Maximum of these two values. (Obtained from Figures R1 to R4)**

	High Freq Band (x 10^{-2} rad/sec)		
	2.13	2.61	Maximum
$\|T_{YD}(s)\|_{max}$ (dB)	1.6	2.1	2.1
$\|T_{UD}(s)\|_{max}$ (dB)	40.6	41.5	41.5
$\|T_{YN}(s)\|_{max}$ (dB)	1.8	1.8	1.8
$\|T_{UN}(s)\|_{max}$ (dB)	40.6	41.5	41.5
$\|SE_Y(s)\|_{max}$ (dB)	1.6	2.1	2.1
$\|SE_U(s)\|_{max}$ (dB)	0.0	0.0	0.0
$\|ERR(s)\|_{max}$ (dB)	0.0	0.0	0.0
$\|T_{UR}(s)\|_{max}$ (dB)	-0.7	-0.2	-0.2

Table R.3: **Values of Each Performance Index at Two Indicated Frequencies in High Frequency Band and the Maximum of these two values. (Obtained from Figures R1 to R4)**

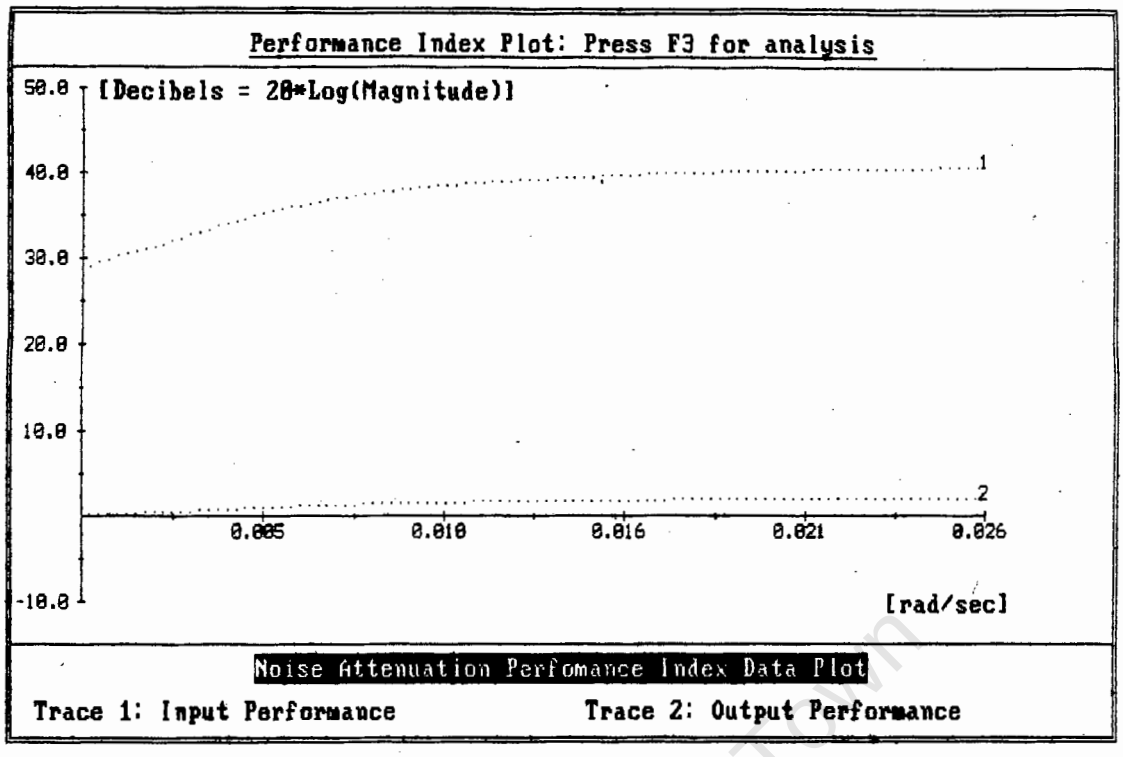


Figure R.1: Design 14 - Noise Transmission Index Plots

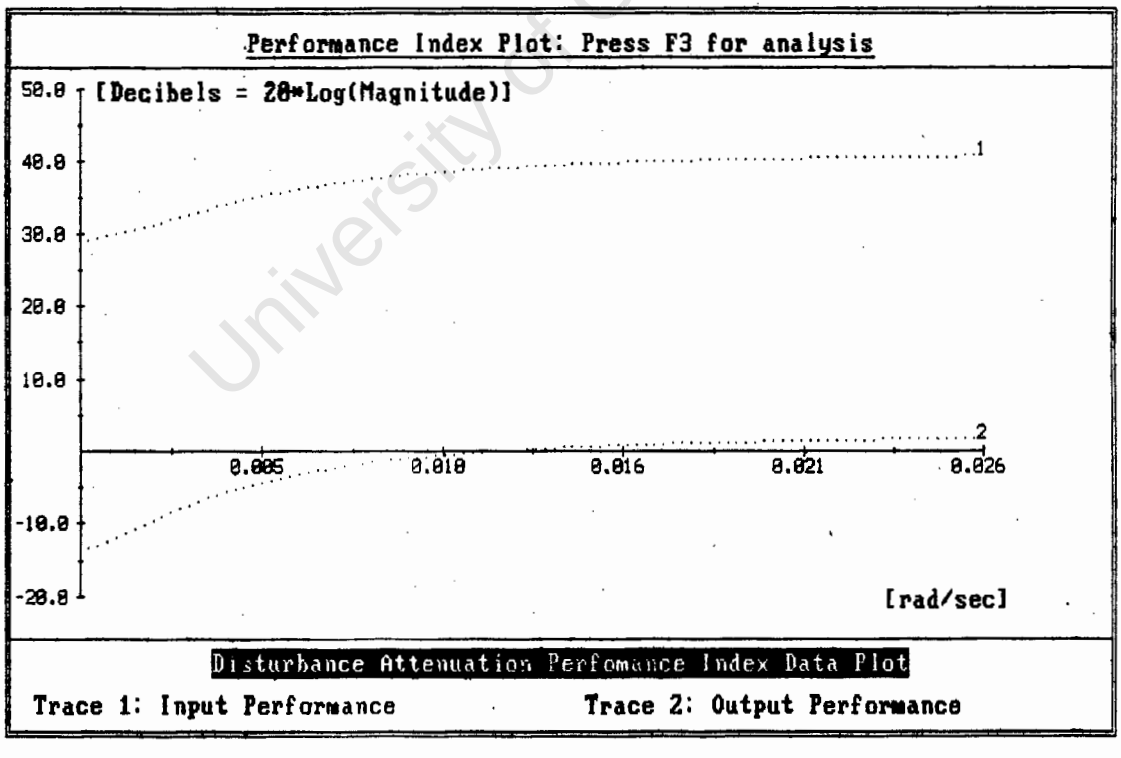


Figure R.2: Design 14 - Disturbance Transmission Index Plots

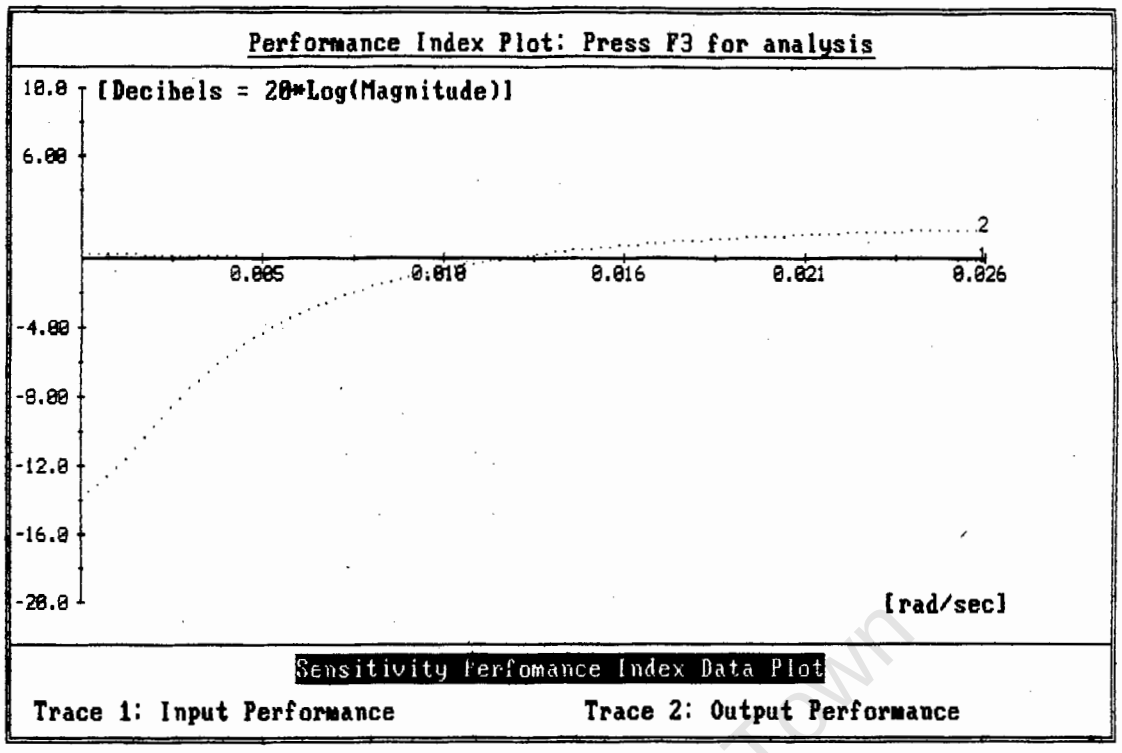


Figure R.3: Design 14 - Sensitivity Index Plots

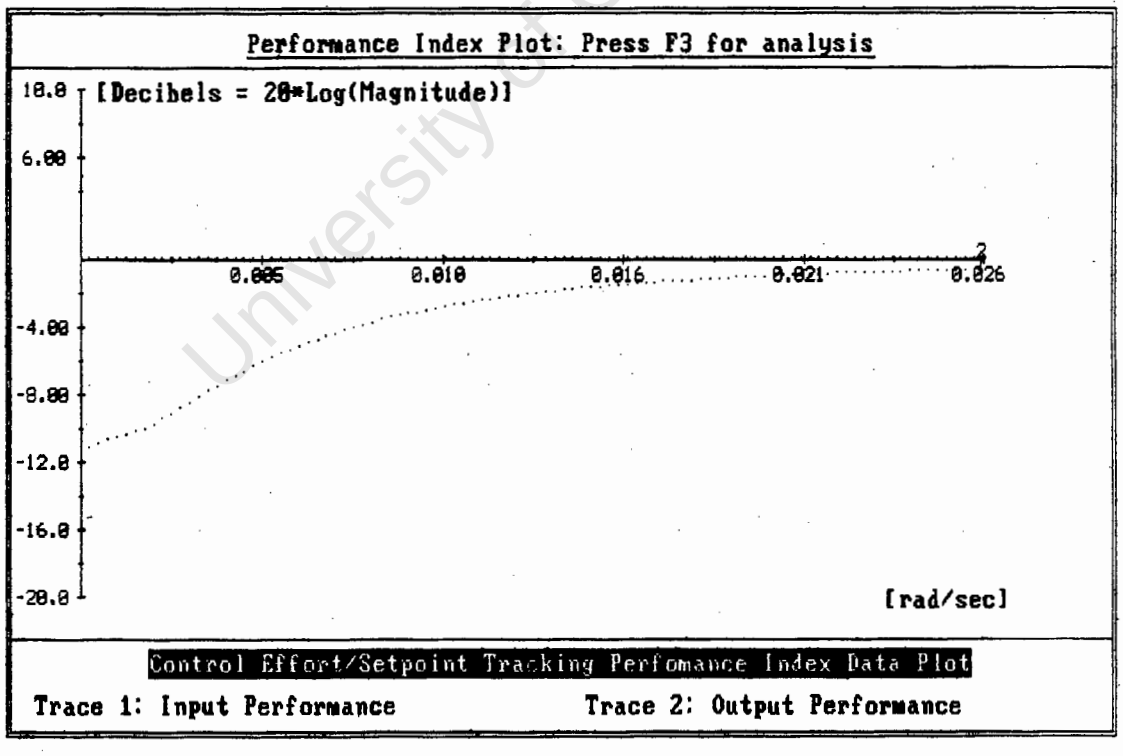


Figure R.4: Design 14 - Control Effort and Tracking Error Index Plots

R.2) Simulated Input and Output Responses

	Low Freq Band (0.00435 to 0.00522 rad/sec)		
	MR_{ch1} (units)	MR_{ch2} (units)	MR_{max}/A_{pert} (dB)
Dist. to Output D->Y	4.2	4.9	-6.17
Dist. to Input D->U	435.0	420.0	32.77
Noise to Output N->Y	9.0	13.0	2.28
Noise to Input N->U	435.0	420.0	32.77
Setp. to Error R->E	10.0	10.0	0.00
Setp. to Input R->U	4.8	4.5	-6.38

Table R.4: **Maximum Simulated Amplitudes of Response, MR_{max} to Perturbations in Low Frequency Band with A_{pert} = 10 units.**
(Obtained from Figures R.5 to R.17)

	Intermediate Frequency Band (0.01305 to 0.01566 rad/sec)		
	MR_{ch1} (units)	MR_{ch2} (units)	MR_{max}/A_{pert} (dB)
Dist. to Output D->Y	8.0	9.2	-0.72
Dist. to Input D->U	750.0	750.0	37.50
Noise to Output N->Y	6.9	16.0	4.08
Noise to Input N->U	750.0	750.0	37.50
Setp. to Error R->E	10.0	10.0	0.00
Setp. to Input R->U	8.0	8.0	-1.94

Table R.5: **Maximum Simulated Amplitudes of Response, MR_{max} to Perturbations in Intermediate Frequency Band with A_{pert} = 10 units. (Obtained from Figures R.5 to R.17)**

	High Freq Band (0.00435 to 0.00522 rad/sec)		
	MR_{ch1} (units)	MR_{ch2} (units)	MR_{max}/A_{pert} (dB)
Dist. to Output D->Y	9.1	11.0	0.82
Dist. to Input D->U	870.0	870.0	38.79
Noise to Output N->Y	5.6	16.9	4.56
Noise to Input N->U	870.0	870.0	38.79
Setp. to Error R->E	10.0	10.0	0.00
Setp. to Input R->U	9.0	9.0	-0.91

Table R.6: **Maximum Simulated Amplitudes of Response, MR_{max} to Perturbations in High Frequency Band with A_{pert} = 10 units.**
(Obtained from Figures R.5 to R.17)

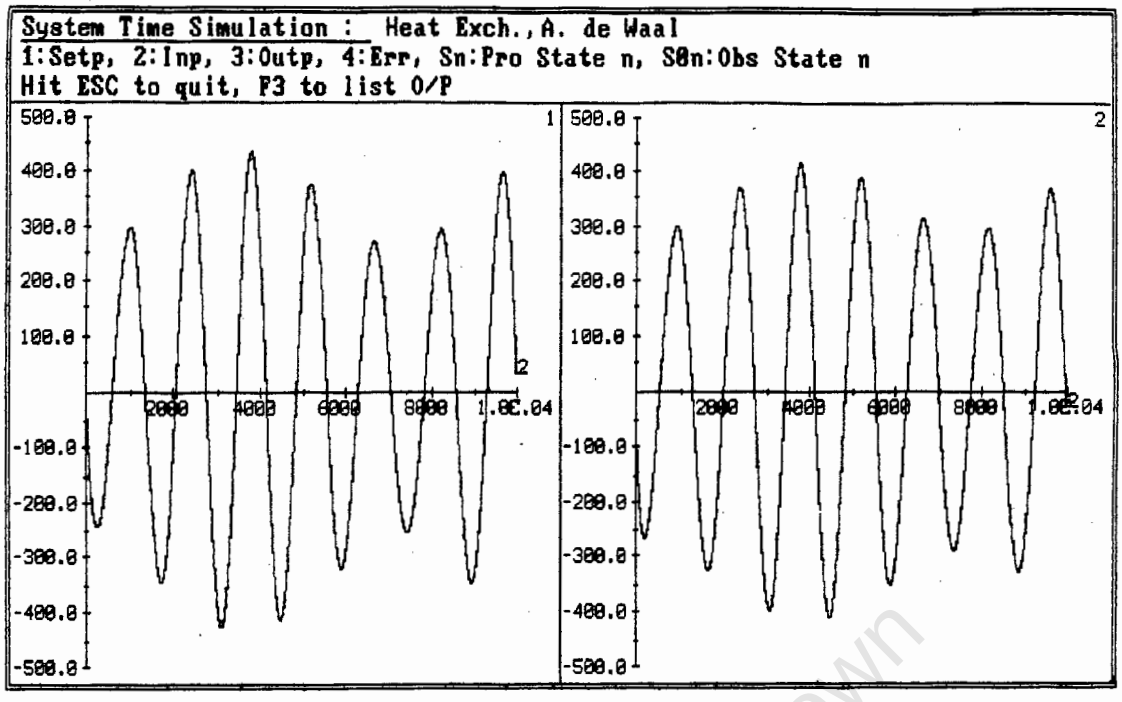


Figure R.5: System 3 - Simulated Response of Inputs to Low Frequency Sensor Noise

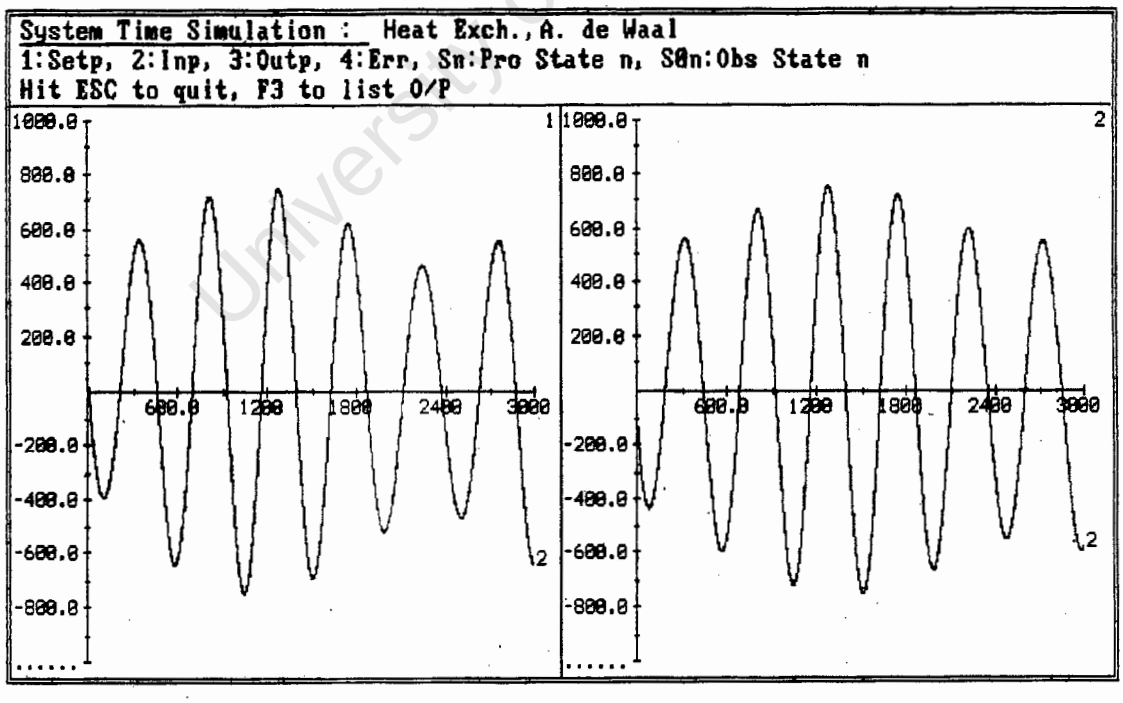


Figure R.6: System 3 - Simulated Response of Inputs to Interm. Frequency Sensor Noise

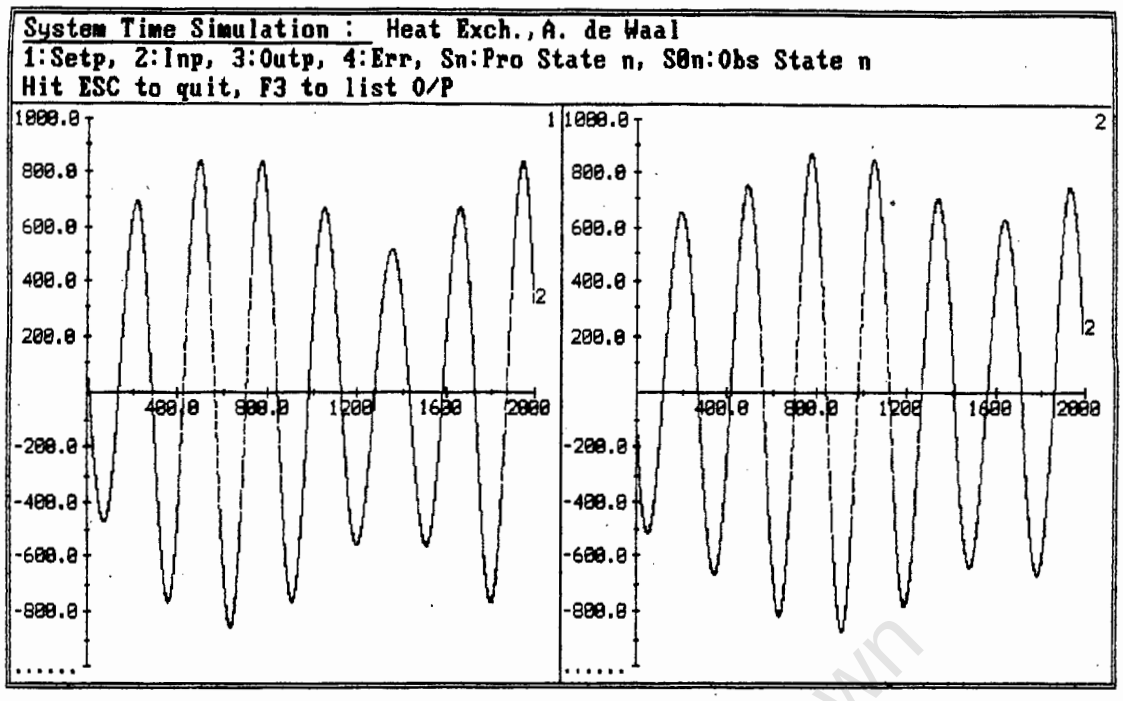


Figure R.7: System 3 - Simulated Response of Inputs to High Frequency Sensor Noise

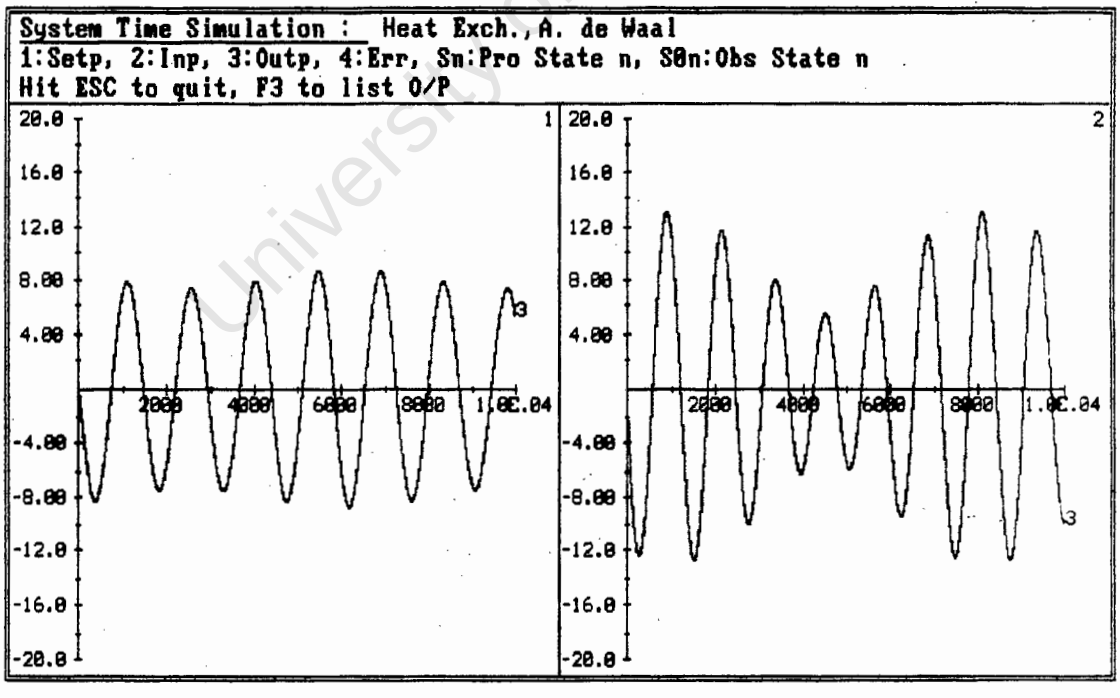


Figure R.8: System 3 - Simulated Response of Outputs to Low Frequency Sensor Noise

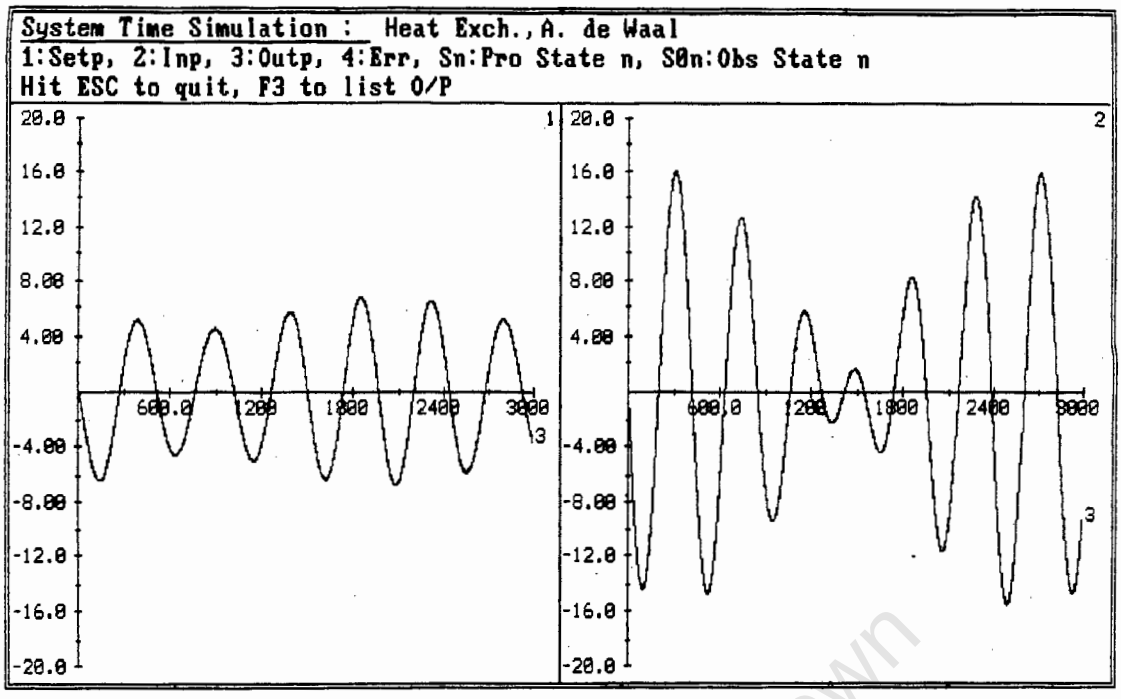


Figure R.9: System 3 - Simulated Response of Outputs to Interm. Frequency Sensor Noise

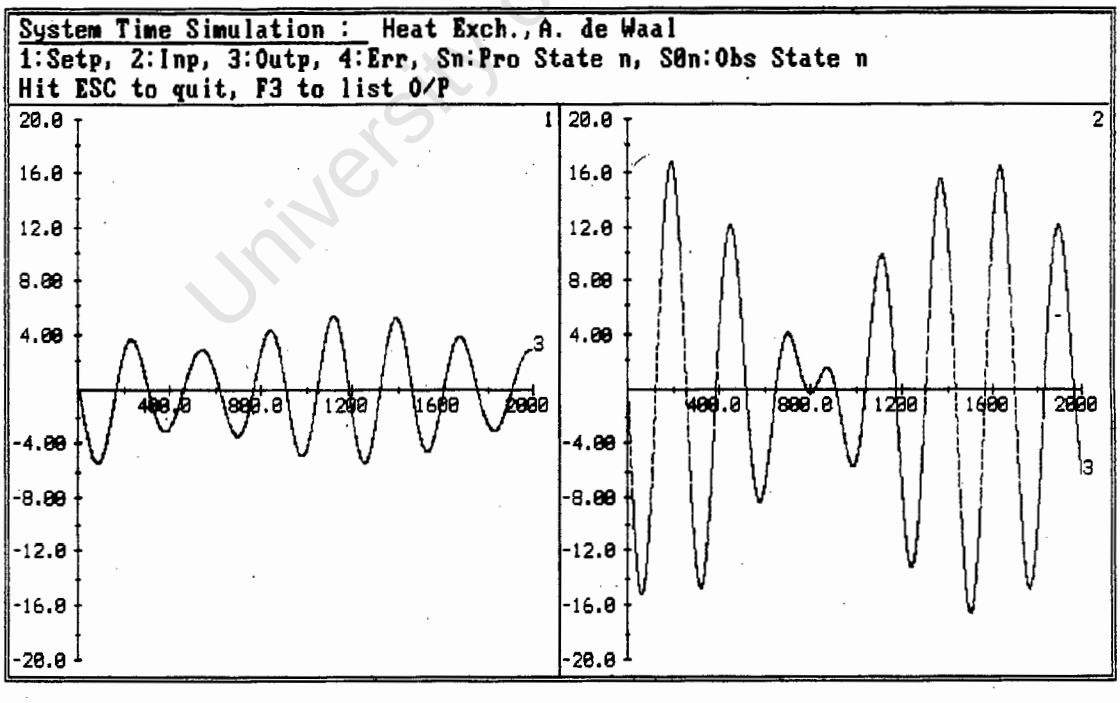


Figure R.10: System 3 - Simulated Response of Outputs to High Frequency Sensor Noise

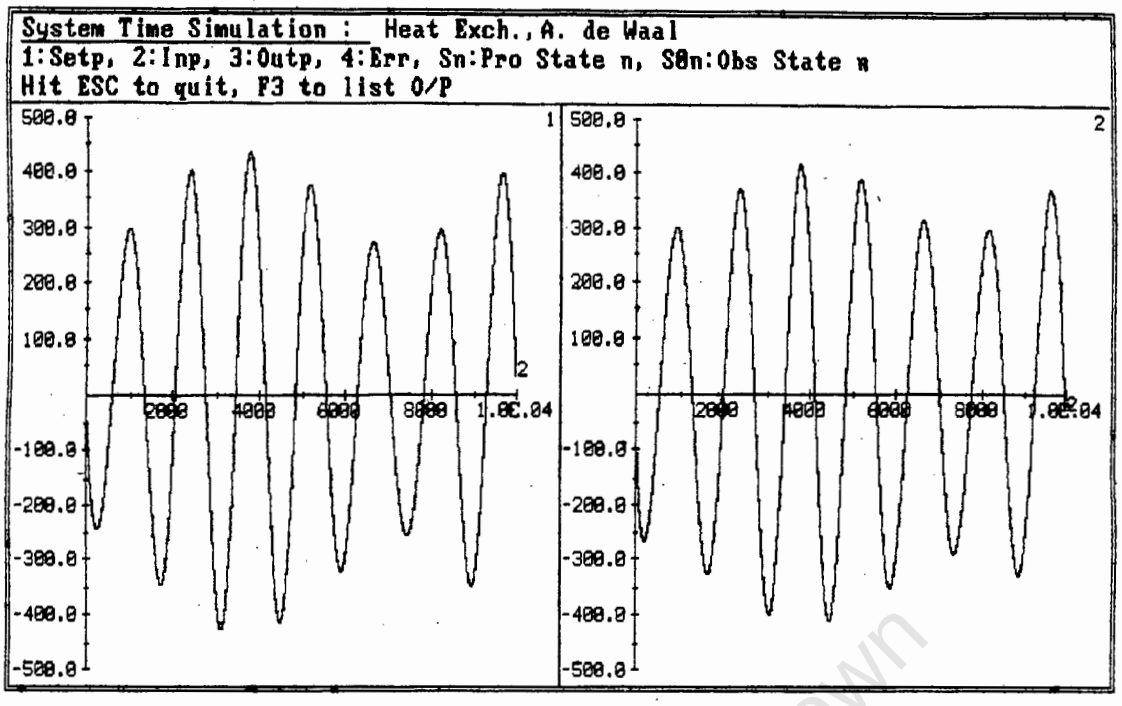


Figure R.11: System 3 - Simulated Response of Inputs to Low Frequency Disturbances

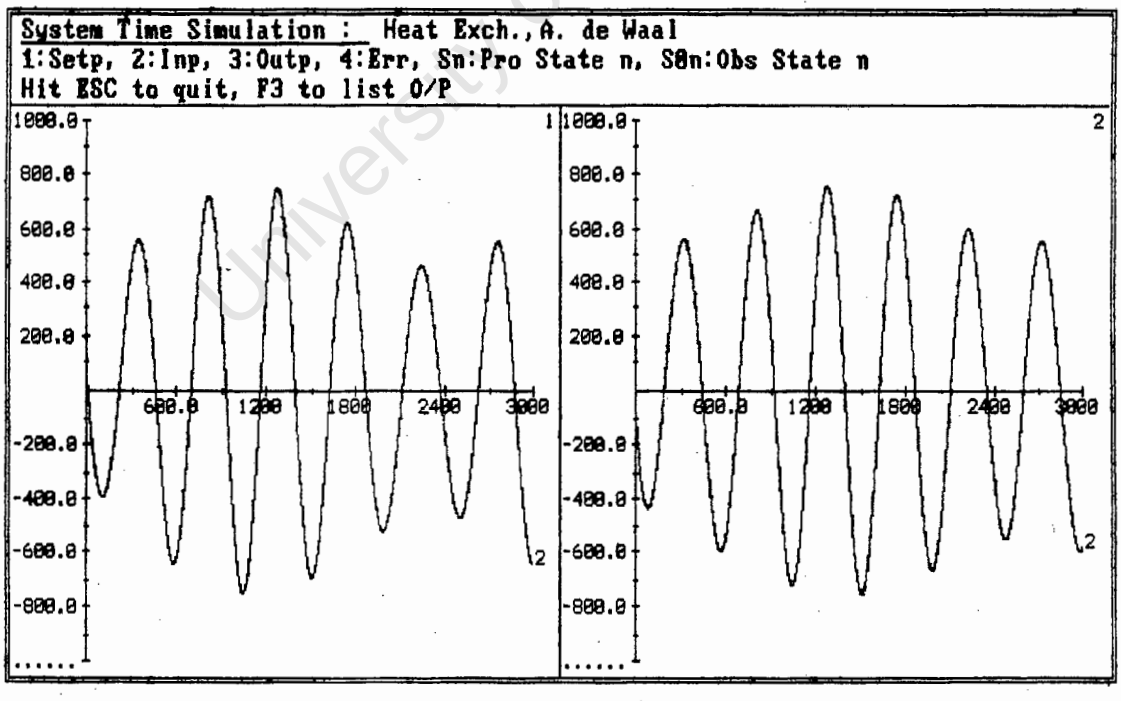


Figure R.12: System 3 - Simulated Response of Inputs to Interm. Frequency Disturbances

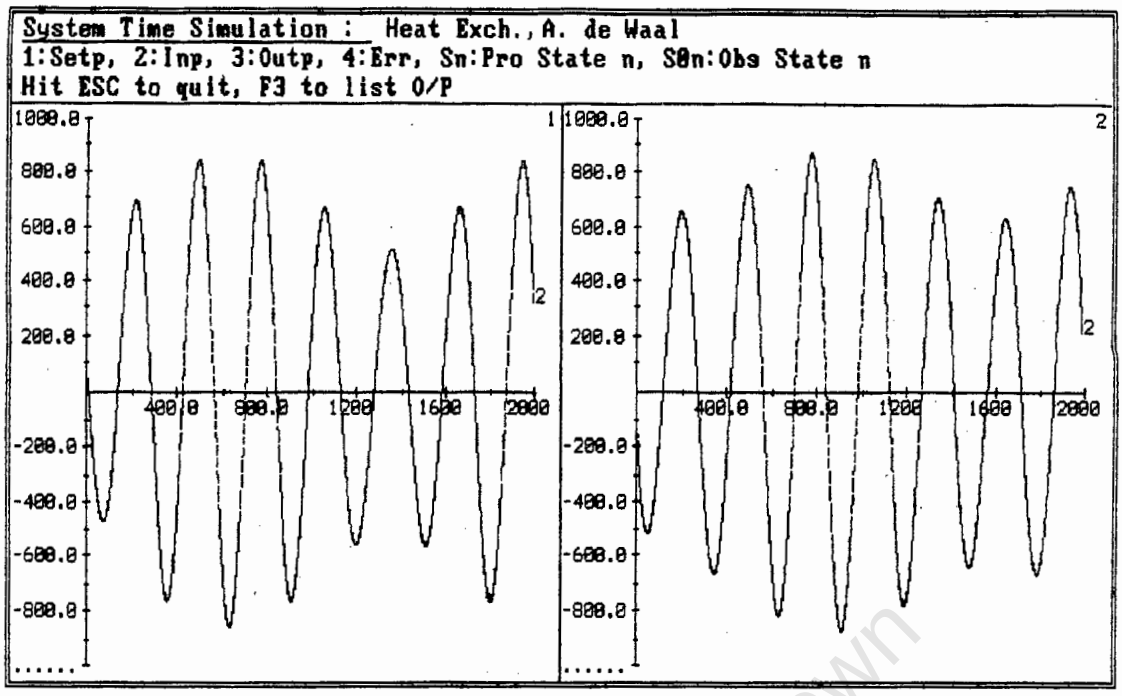


Figure R.13: System 3 - Simulated Response of Inputs to High Frequency Disturbances

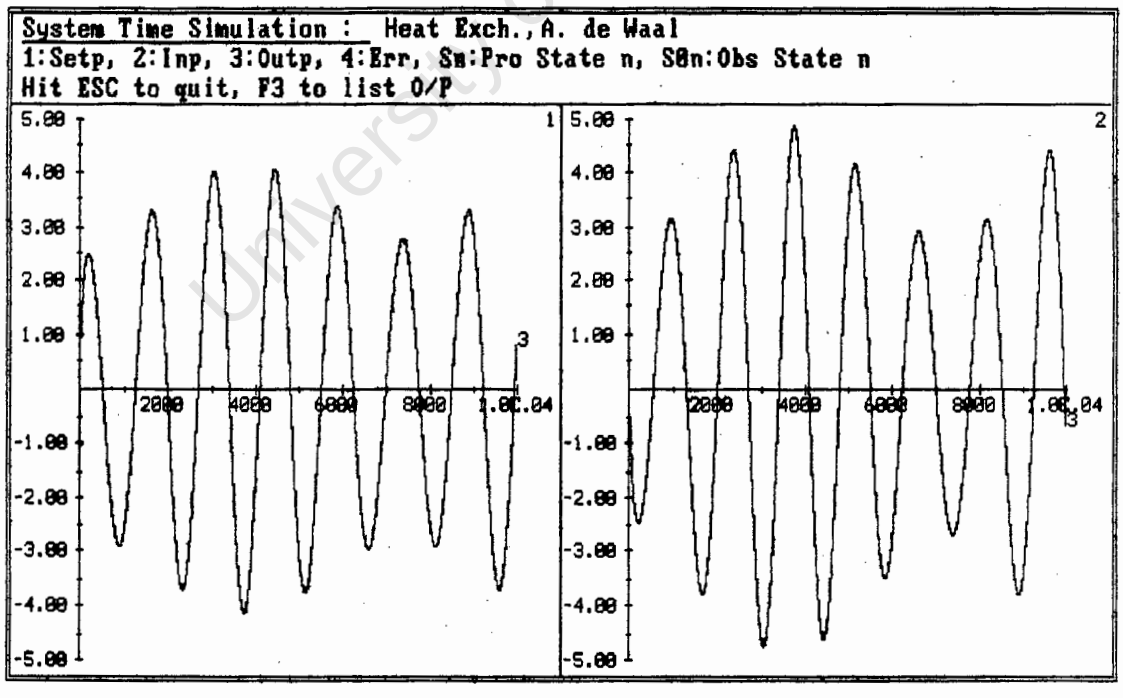


Figure R.14: System 3 - Simulated Response of Outputs to Low Frequency Disturbances

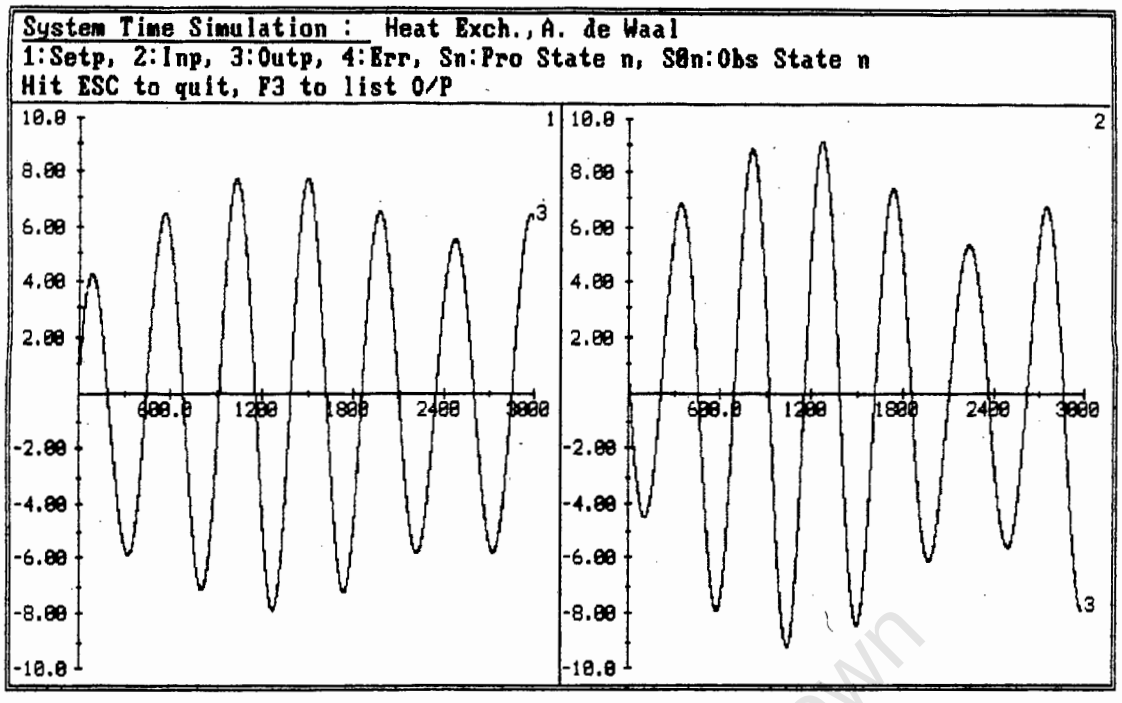


Figure R.15: System 3 - Simulated Response of Outputs to Interm. Frequency Disturbances

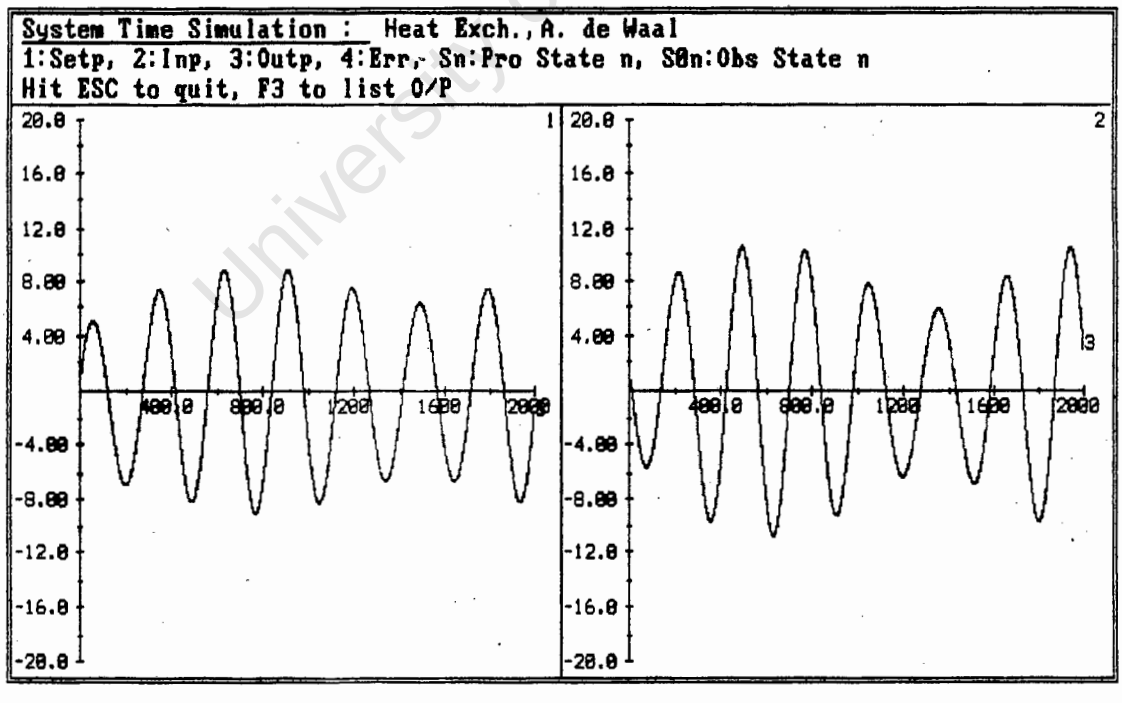


Figure R.16: System 3 - Simulated Response of Outputs to High Frequency Disturbances

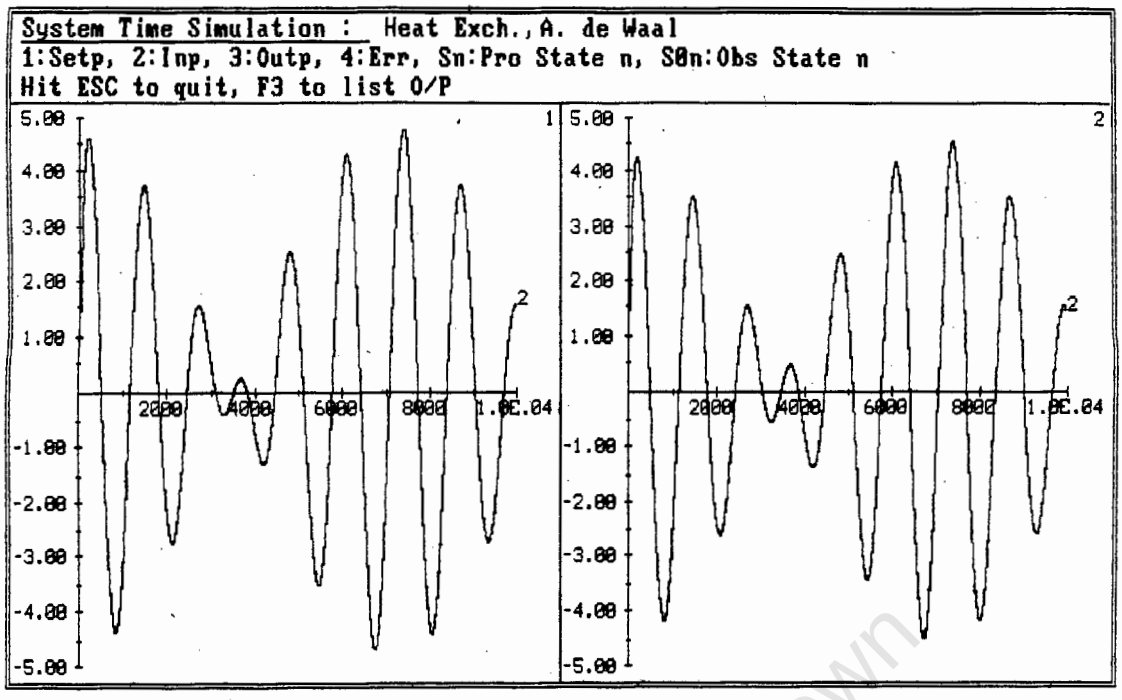


Figure R.17: System 3 - Simulated Response of Inputs to Low Frequency Setpoints

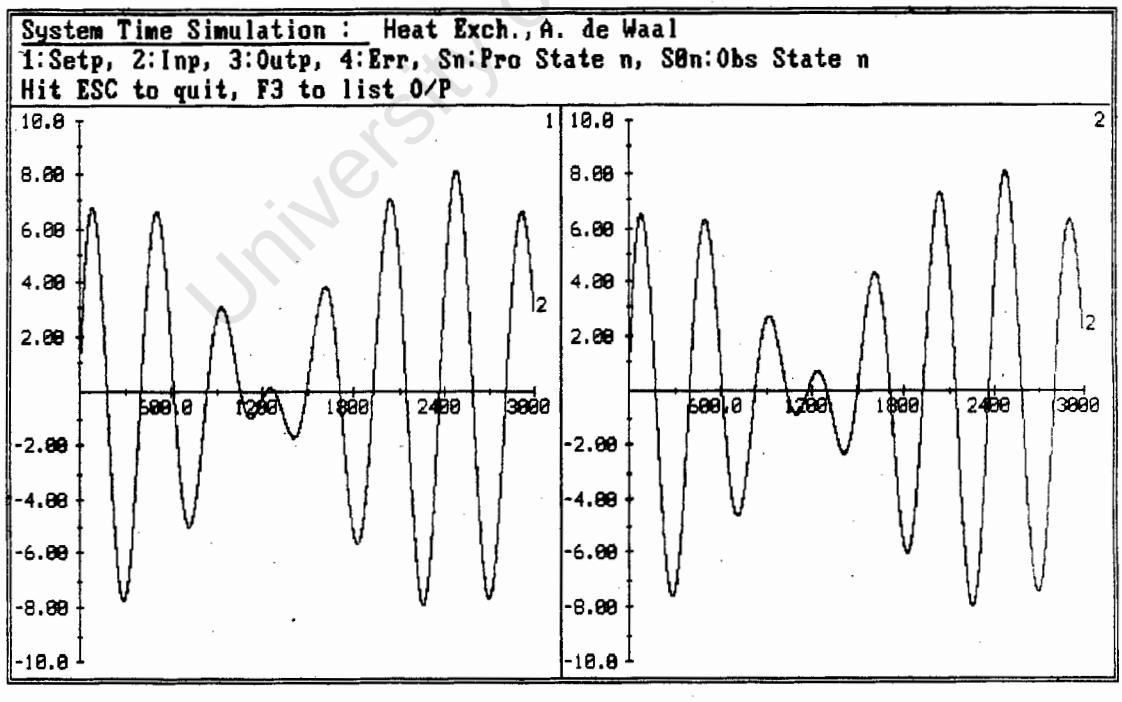


Figure R.18: System 3 - Simulated Response of Inputs to Interm. Frequency Setpoints

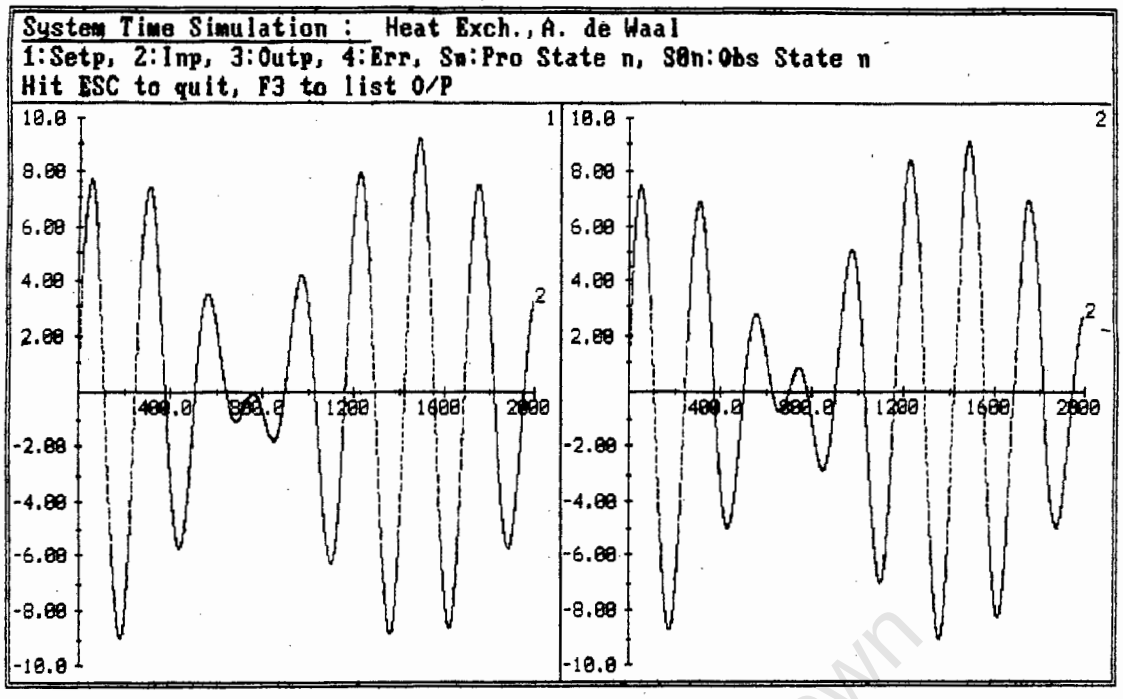


Figure R.19: System 3 - Simulated Response of Inputs to High Frequency Setpoints

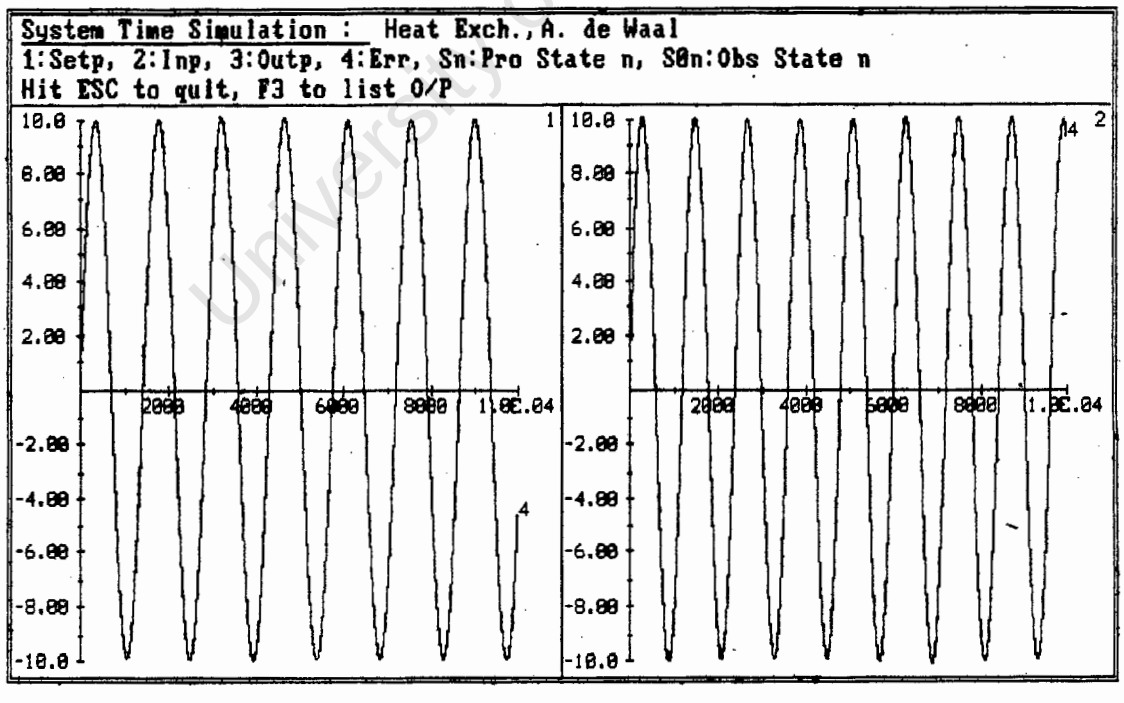


Figure R.20: System 3 - Simulated Response of Errors to Low Frequency Setpoints

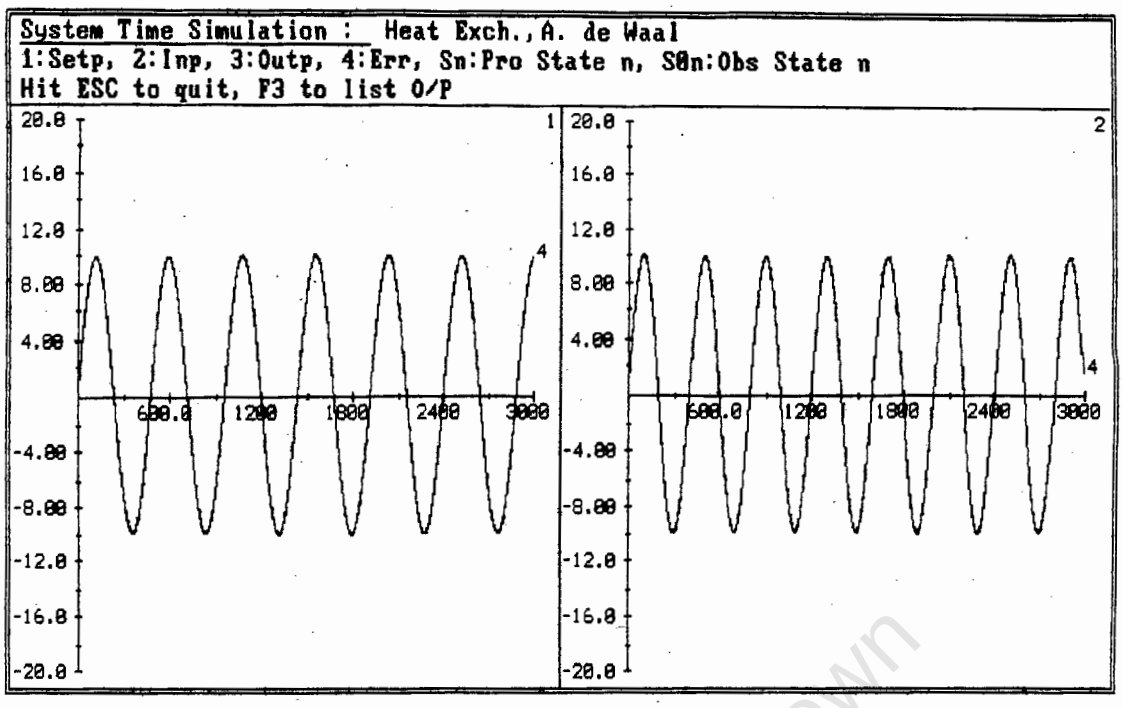


Figure R.21: System 3 - Simulated Response of Errors to Interm. Frequency Setpoints

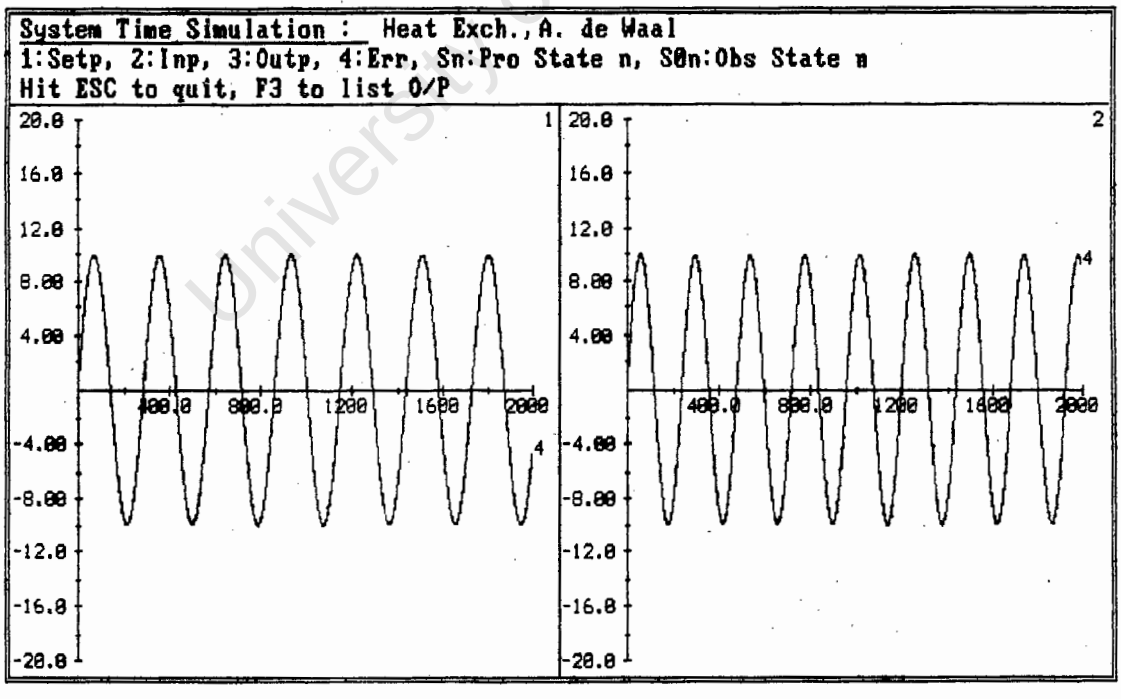


Figure R.22: System 3 - Simulated Response of Errors to High Frequency Setpoints