MULTIVARIABLE CONTROL OF A FLOTATION PLANT SIMULATOR

A thesis submitted to the Department of Electrical Engineering, University of Cape Town, in fulfilment of the requirements for an MSc. Degree in Electrical Engineering

by

I.P. FISHER

December 1988

Acknowledgments

I would like to thank the following :

The Council for Mineral Technology for providing the funding for this research.

My supervisor and mentor, Dr. M. Braae for his guidance and patience throughout this project.

Abstract

This dissertation describes the application of two multivariable frequency domain techniques in order to design controllers for a "flotation plant". A flotation plant simulator was designed and constructed at the University of Cape Town. The design of the multivariable controllers was based on a linear time invariant model (in s-domain) developed for the simulator. The two frequency domain techniques, Characteristic Loci (CL) and Inverse Nyquist Array (INA), were implemented in the form of CAD packages. The INA CAD package had already been written at the university but the CL CAD package had to be developed before the design of the controllers could proceed.

Multivariable controllers were designed using the two multivariable techniques in order to compare the techniques from a designer's point of view. These controllers were implemented on the flotation plant simulator and the response of the closed-loop system was correlated with the information provided by the two design techniques.

The two multivariable controllers did maintain control of the flotation plant simulator. The level of interaction on the Simulator was also reduced by the presence of the two controllers. The closed-loop characteristics of the simulator could be correlated with the predictions of the two design techniques. The philosophies of the two design techniques differ in that the CL technique is a powerful analyser of system characteristics, whereas the power of the INA technique lies in its simple design objective of system diagonal dominance.

Table of Contents

List of Illustrations

Chapter Six

Appendix F

Tables

Photographs

Appendix B

Displays

Chapter Four

Nomenclature

A list of symbols used and their meanings is given in this nomenclature. Note that matrices are represented by uppercase characters while vectors and scalars are represented by lowercase characters.

A matrix element is represented by the lowercase character corresponding to the uppercase character of the matrix with a pair of subscripted numbers which represent the row and column numbers of the element. Generally the subscript i denotes the row number and the subscript j denotes the column number.

Vectors are all column vectors unless otherwise stated.

Symbol	**Meaning**
A/D	Analog to digital conversion
CAD	Computer Aided Design
CL	Characteristic Loci
CLCP	Closed loop characteristic polynomial
C_1	System condition number
C_2	Condition number of dG(s)
det A	Determinant of matrix A
D	The Nyquist contour D
dG(s)	Multivariable perturbation of the plant, G(s)
D_1	Scaling matrix for K_∞
D_2	Scaling matrix for K_o
diag(a_i)	A diagonal matrix with the elements $a_1, a_2, \ldots, a_n$ on the diagonal
D/A	Digital to analog conversion

$e(s)$	A vector of the difference between the desired outputs and the actual outputs
$F_e(s)$	Return-difference matrix for system error vertex
$F_u(s)$	Return-difference matrix for system input vertex
$F_y(s)$	Return-difference matrix for system output vertex
$G(s)$	Input output matrix model of a multivariable system
$g_{ij}(s)$	The element of the matrix G(s) in the ith row and jth column
$H(s)$	Matrix of feedback functions, usually diagonal
H_L	Positive semidefinite Hermitian matrix often called the left moduli
H_R	Positive semidefinite Hermitian matrix often called the right moduli
I	Identity matrix
INA	Inverse Nyquist Array
$K(s)$	Matrix representing a multivariable controller
$K_c(s)$	Precompensator to diagonalise a multivariable system
K_o	Matrix to diagonalise the system at DC
$K_{pi}(s)$	Matrix with PI controller terms on the diagonal
K_∞	Matrix to diagonalise the system at high frequencies
LTI	Linear time invariant
NELM	An algorithm to implement a non-linear parameter adjustment technique

OLCP	Open loop characteristic polynomial
p_o	Number of poles in the right half s plane
$Q(s)$	Open-loop transfer function of the controlled system
$q_i(s)$	Characteristic transfer functions of $Q(s)$
$R(s)$	Matrix of transfer functions representing closed-loop system
$r(s)$	A vector of setpoints (desired values) for the outputs of the system
$R_i(s)$	Input relative stability matrix
$R_o(s)$	Output relative stability matrix
s	Laplace variable
$T_e(s)$	Return-ratio matrix for the system error vertex
$T_u(s)$	Return-ratio matrix for the system input vertex
$T_y(s)$	Return-ratio matrix for the system output vertex
$u(s)$	A vector of inputs to a system
$V(s)$	Inverted $W(s)$
$v_i(s)^t$	Rows of $V(s)$
w	Angular frequency in radians
$w_i(s)$	Linearly independent characteristic vectors associated with $q_i(s)$
$W(s)$	Indentically non-singular matrix where the columns consist of $w_i(s)$
$y(s)$	A vector of outputs from a system
A^*	Complex conjugate of A

$\|a\|$	The absolute value of a scalar or vector a
$\|A\|$	The norm of a matrix A
$\|x\|_2$	The Euclidian norm of vector x (l_2-norm)
$\|A\|_2$	The spectral norm of a matrix A
$<x,y>$	Inner product
Γ_i	The contour in the s plane onto which an element q_{ii} is mapped by D
$\propto(s)$	Signal transform vector
$\propto_i(s)$	Principal gain of system
$\theta_i(s)$	Principal phase of system
$\delta_i(s)$	Principal gain of dG(s)
$\epsilon_i(s)$	Principal phase of dG(s)
$\Omega_m(s)$	Phase modifier
$\theta_j(s)$	Misalignment angle
$\stackrel{D}{=}$	Equals by definition

Introduction

Flotation Systems

Flotation is one of the most widely used mechanisms for mineral extraction and refinement in the mineral industry [1]. Flotation is a process whereby the grains of mineral ore or compound in a pulp or slurry are caused to rise to the surface in a cell or tank by the action of bubbles of air. The grains are caught in a froth formed on the surface of the cell and are removed with the froth. While the grains that do not rise remain in the slurry and are drawn off the bottom of the cell. The refining of minerals being achieved by building up a network of these flotation cells.

Control of Flotation Processes

Due to the economic climate it is desirable to maintain strict control of the flotation process. There are two main aspects of flotation which need to be controlled :

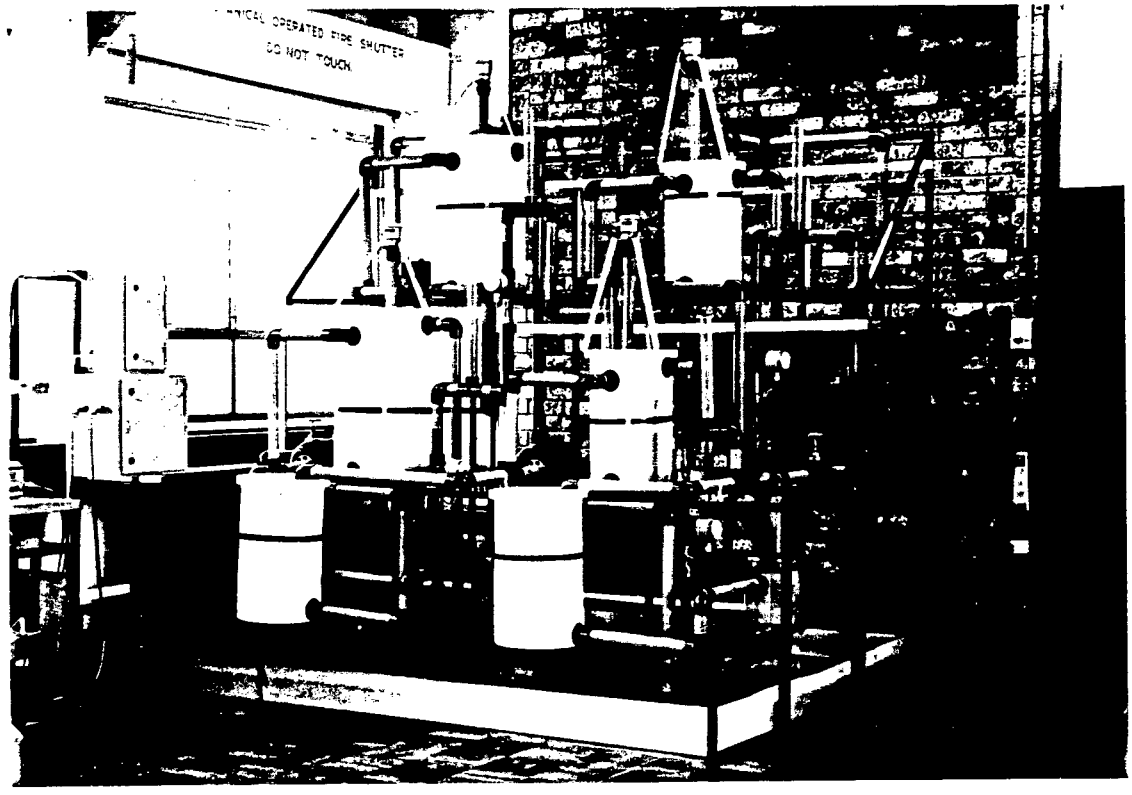
i) The chemical characteristics of the flotation cells (eg. chemical composition of the input to the flotation cells, froth formation in the flotation cells and chemical composition of the outputs of the flotation cells).

ii) The fluid dynamics of the flotation cells (eg. the fluid levels in the flotation cells and flow rates from the flotation cells).

This study of control techniques is only concerned with the second aspect of control, the fluid dynamics, of the flotation process. Since a flotation cell has at least one input and two outputs and a flotation process consists of many interconnected flotation cells, the system will require multivariable control of the fluid dynamics.

Flotation Plant Simulator

But, in order to study the fluid dynamics of a flotation plant and the effectiveness of various multivariable control schemes, a flotation plant simulator was required. Thus, a flotation plant simulator was designed and built in the control laboratory at the University of Cape Town (see Photograph I, below).



Photograph I The Flotation Plant Simulator

The flotation plant simulator included the appropriate instrumentation (eg. level probes, flow sensors, etc.) in order monitor and control the fluid dynamics of the system. All the instruments and actuators were interfaced to a computer to facilitate the data processing requirements of a variety of multivariable control schemes.

Multivariable Techniques

Numerous techniques have been developed for designing controllers for multivariable systems, but only two frequency domain techniques have been selected for the investigation of multivariable control of the flotation plant simulator. The two techniques are :

Inverse Nyquist Array (INA)

Characteristic Locus.

Computer Aided Design Systems

One of the major reasons for the revival of interest in the application of frequency response techniques to multivariable systems is the availability of computing power. Multivariable frequency domain techniques are computationally intensive and require the graphical representation of a set of complex variables.

Thus computer aided design (CAD) systems to implement the two frequency domain techniques mentioned above, have been developed at the University of Cape Town. The INA-CAD system had been developed at the time of the investigation [2]. Thus, the Characteristic Loci CAD system had to be developed in order to fulfill the requirements of the investigation.

Control of the Flotation Plant Simulator

The investigation of control of the Flotation Plant Simulator involved the following steps :

i) Designing multivariable controllers and compensators for the system using the INA and Characteristic Loci CAD systems.

ii) Implementing the controllers and compensators on the Flotation Plant Simulator.

iii) Compare the two design techniques. The techniques were compared from a designers point of view (ie. were problems within the system easily identified ?, etc.) and on the performance of the controllers.

Breakdown of Dissertation

The first chapter of this dissertation describes the design process and construction of the Flotation Plant Simulator. The procedure followed to calculate the model (in the frequency domain) of the Flotation Plant Simulator is explained in chapter two. The resultant Flotation Plant Simulator model is also presented in chapter two.

The background theory of characteristic loci as a method to describe multivariable feedback systems and the characteristic loci design technique are explained in chapter three. The characteristic loci theory and explanation have been included as a lead-in to chapter 4, which describes the design philosophy and capabilities of the Characteristic Loci CAD system.

The design and implementation of multivariable compensators and controllers for the Flotation Plant Simulator using the two frequency domain techniques (INA and characteristic loci) is described in chapters five and six respectively.

The conclusion contains a brief evaluation of the three main components of this dissertation ; The Flotation Plant Simulator, the Characteristic Loci CAD system and the investigation of control of the Flotation Plant Simulator. Comments are also made concerning future work and extensions of the systems (ie. the Flotation Plant Simulator and the Characteristic Loci CAD system) initiated through this project.

Chapter 1

Flotation Plant Simulator

To investigate various multivariable control techniques for flotation systems, a flotation plant simulator has been designed and built in the control laboratory at the University of Cape Town. The flotation plant simulator simulates the fluid dynamics of a flotation process circuit typical of the type found in the mineral extraction industry [3].

The flotation plant simulator had to satisfy the following requirements :-

i) The product circuit must be typical of the circuits used in the mineral extraction industry.

ii) The parameters measured on the flotation plant simulator (eg. product level in a particular flotation cell) must be similar to those parameters available on a real flotation system.

iii) The flotation plant simulator must provide the researcher with enough information to enable various control schemes to be investigated.

This chapter describes the design and construction of the flotation plant simulator. Firstly, the design is initialised with the simulation of a single flotation cell. These simulated flotation cells are then combined to realise the product circuit of a typical flotation plant. The design is followed by details of the construction of the flotation plant simulator (ie. pipe sizing, instrumentation, etc).

1.1 Design of Flotation Plant Simulator

The product circuit for the flotation plant simulator is of a typical flotation system [3]. This product circuit is shown schematically in figure 1.1 below.

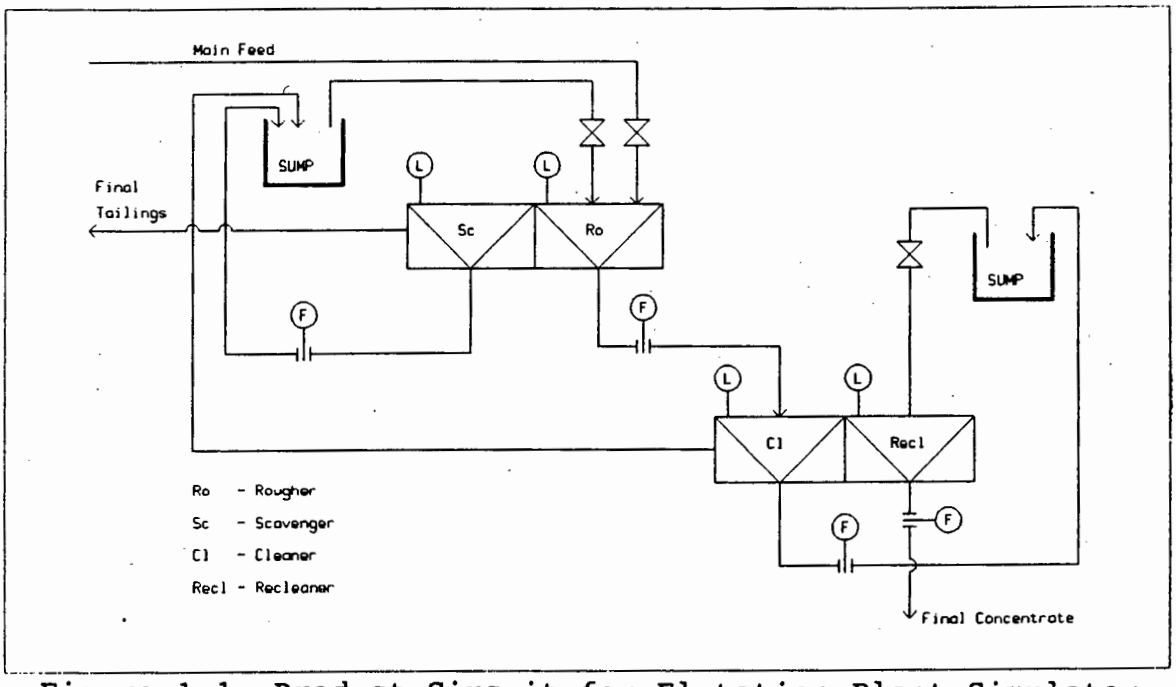


Figure 1.1 Product Circuit for Flotation Plant Simulator

In a typical flotation cell, the cell splits the input flow into two outputs :-

i) *Concentrate (or froth)* : The rate of concentrate output is dependent on the level of the product within that cell. If the level is too low, no concentrate will be output. If the level is too high, the concentrate output would be contaminated by the 'non-froth' components in that cell.

ii) *Tailings* : The rate of tailing output is normally controlled in order to control the level of product in the flotation cell. The maximum rate of tailing output is obviously dependent on the level of product in that flotation cell.

1.1.1 Flotation Cell Simulation

The configuration of each cell within the flotation plant simulator is described in figure 1.2, shown below. The product in the simulated circuit is water. This configuration permits the above mentioned characteristics of flotation cells to be simulated in the following manner :-

i) *Concentrate Output* : (Refer to figure 1.2) The inverted U-piece prevents any product being output until the level in the cell exceeds the height of the cross-piece. The breather pipe on the cross-piece prevents any product from being siphoned out of the cell once the product level drops below the level of the cross-piece.

   The main reason for the inverted U-piece (as opposed to a pipe section exiting the simulated cell at the froth level), is the positioning of the flow sensors : The flow sensors should ideally be placed in an upwardly flowing stream to ensure accurate measurement of flow velocity.

   The concentrate output is only monitored, not controlled directly.

ii) *Tailings Output* : (Refer to figure 1.2) The tailing out flow is controlled by a control valve (pneumatic), but the flow rate is not monitored. Thus the product level in the cell can be controlled by regulating the tailing output from the simulated cell.

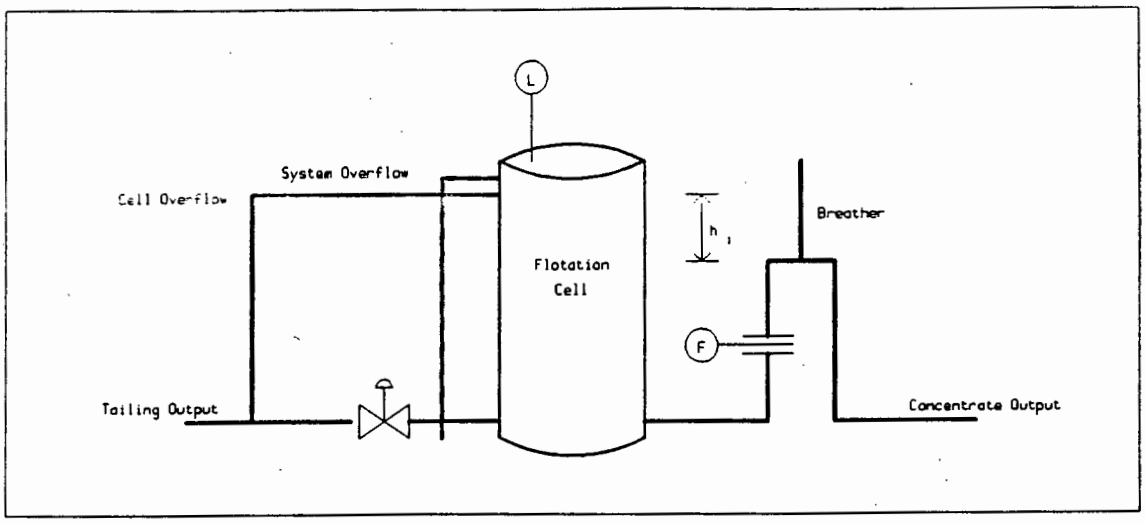


Figure 1.2 Flotation Plant Simulator Cell Configuration

The *Cell Overflow* (shown above in figure 1.2 on the simulator floatation cell) pipe that bypasses the control valve has been put there as a precaution. If the output from the concentrate and tailings outputs of the cell is less than the total input to the cell, flooding may result. This condition could result from poor control of the system. Thus, the Cell Overflow bypass of the control valve prevents (or delays) flooding of the cell, without the total mass of product within flotation circuit being reduced.

The *System Overflow* (shown above in figure 1.2) has been installed to prevent flooding in the case of the flotation simulator circuit containing an amount of product in excess of the entire circuits' capacity.

1.1.2 Realisation of the Flotation Simulator Product Circuit

The product circuit described in figure 1 of section 1.1 can be realised using the flotation cell simulation technique described previously in section 1.1.1. The simulated product circuit realisation is described by the drawing below (figure 1.3).

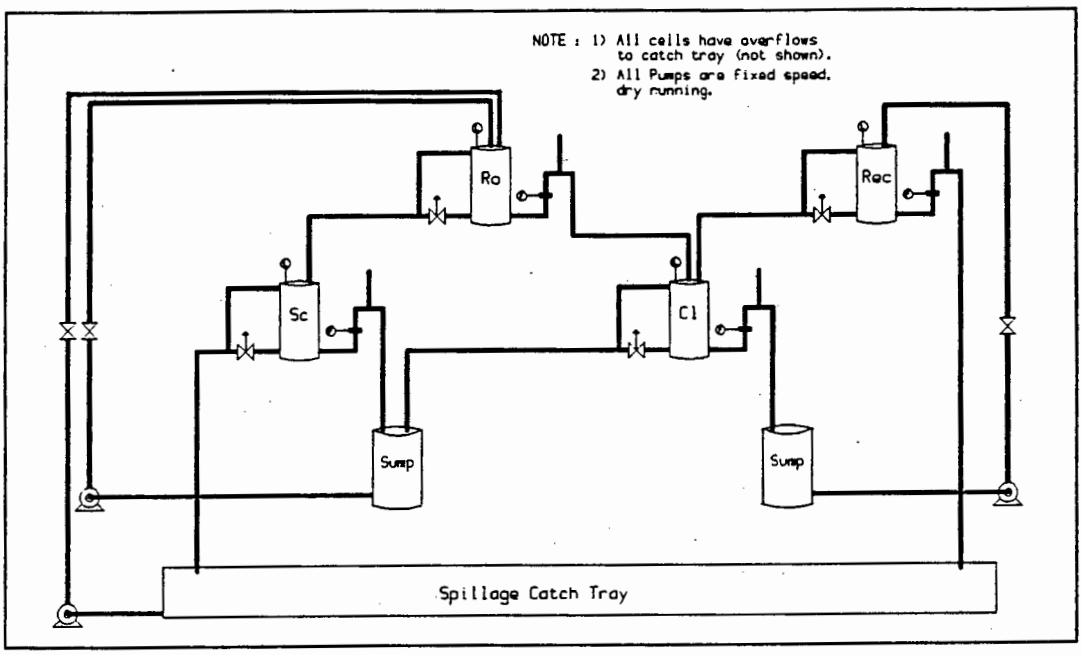


Figure 1.3 Realisation of Simulated Product Circuit

1.1.3 The Simulated Product Circuit Flow Rates

The flow rates from each input and output were selected by setting the system input and output flow rates to values that were proportionate to the mass flow of a typical flotation system. The remaining flow rates were then calculated and adjusted until all the net inflows to each cell equaled the net outflows from each cell.

The desired normal and extreme flow rates in the simulated product circuit are listed below in figure 1.4.

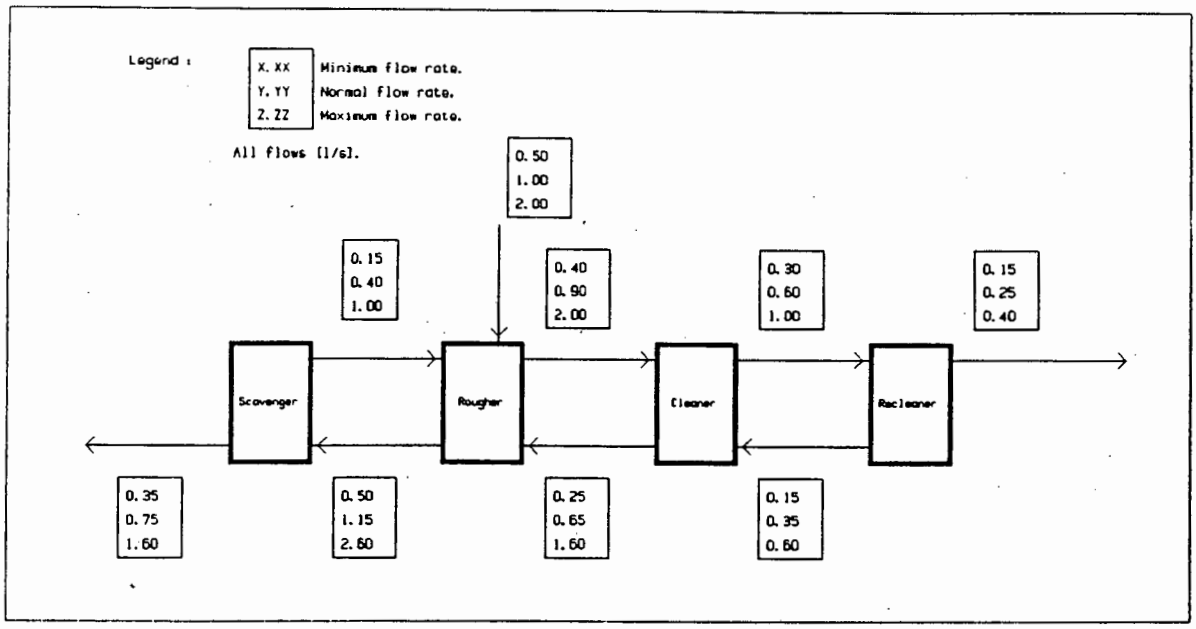


Figure 1.4 Normal and Extreme Flow Rates in the Simulated Product Circuit

1.2 Construction of the Flotation Simulator

1.2.1 Pipe Sizing to Accommodate Flow Rates

The sizing of the circuit pipes was calculated as follows :-

i) Calculate the minimum fluid velocity in each pipe section using the minimum head of water for that section of the circuit (the minimum height obtained from the drawings of the flotation simulator cells and the basic framework drawings, shown in appendix C). The fluid velocity given by

$$v = (2\ g\ h)^{\frac{1}{2}}$$

where : v - velocity [m/s]
h - height [m]
g - 9.8 [m/s^2]

(This equation obtained from an approximation of Benoullis' equations [1]).

The fluid velocity was always reduced to accommodate the pressure drop due to tank exits, elbow joints and pipe friction.

ii) Calculate the pipe diameter required to achieve the maximum flow rate for each section of the circuit given the minimum velocity (calculated above) and the flow rate given in figure 1.4 in section 1.1.3.

All the flow rates in the product circuit could be achieved using a combination of 25 mm and 50 mm diameter piping. But to simplify the construction (and reduce production costs) of the simulator, only 50 mm piping was used in the implementation of the flotation plant simulator.

1.2.2 Instrumentation on the Flotation Plant Simulator

Since the object of the project is to study the product dynamics of the flotation simulator, the variables of interest are product level in the tanks and product flow rates in the circuit. Thus, the instrumentation on the flotation simulator consists of level probes in the simulated flotation cells and flow sensors in the simulated froth outflows.

Ideally, to have made the flotation simulator dynamics completely observable flow sensors should have been installed in the tail outflows and main feed into the system. But, due to cost of the flow sensors and a limited budget, the installation of these instruments was not possible.

To control the product dynamics of the flotation simulator, pneumatic control valves were installed in the tail outflow of each tank.

A control valve should have been installed in the main feed to affect complete control of the product dynamics of the flotation simulator. But, as explained above, due to expense and limited budget this was not possible.

Specific information concerning the instruments and actuators is contained in the following appendices :-

i) Calculations required to select the instruments and actuators : Appendix A

ii) Installation (and manufacturer) of the instruments and actuators : Appendix B

iii) Circuit diagrams of the installation of the instruments and actuators : Appendix D

1.2.3 Monitoring the Instruments and Effecting Control

All the instruments (including actuators) on the flotation simulator are interfaced to a personal computer system (PC).

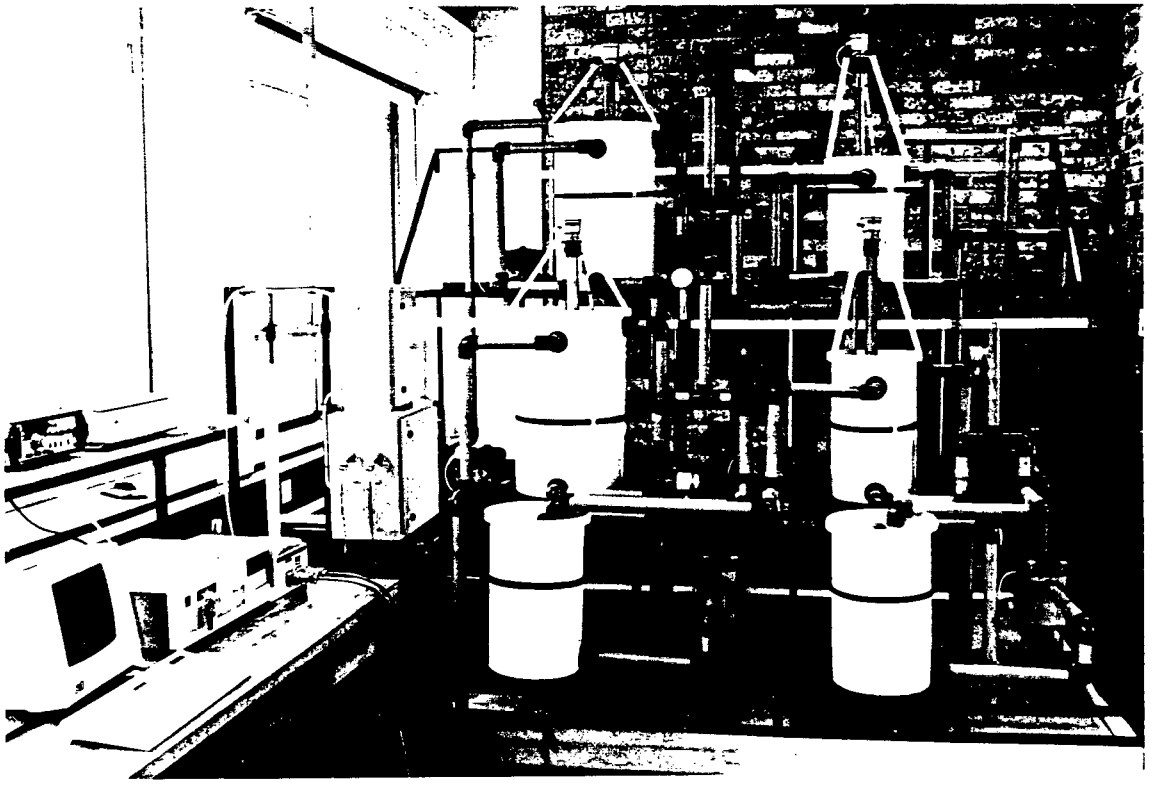
The instruments' analog outputs are converted to digital format using an internal add-on analog-to-digital (A/D) card for the PC. A similar configuration is used to interface the PC to the actuators except the PCs' digital format is converted to an analog signal using a digital-to-analog (D/A) card.

This configuration allows the instrument data to be used and/or modified in any way the researcher sees fit. The PC provides the computational power and flexibility required by a researcher to either monitor or implement various control schemes on the flotation simulator.

The equipment required to interface the PC to the flotation plant simulator is listed in appendix B, while the circuit diagrams are shown in appendix D.

1.3 Commissioning the Flotation Plant Simulator

The flotation plant simulator was commissioned on 1 June 1988 following one and half years of design, construction and testing. The photograph below describes the commissioned flotation plant simulator.



Photograph 1.1 The Flotation Plant Simulator

Chapter 2

Identification of the Flotation Plant Simulator Model

In order to investigate various multivariable control techniques on the flotation plant simulator, the system model has to be identified. Since this investigation is only interested in frequency domain techniques (as explained in the Introduction), the model is presented in the form of a matrix of polynomials in the frequency domain.

This chapter describes the identification of the flotation plant simulator model. The tests performed on the simulator are described first, followed by description of the method of analysis of the data obtained during the tests. Finally, the system model is presented to the reader.

2.1 Flotation Plant Simulator Open Loop Tests

2.1.1 The Linear Operating Region

Since the multivariable control techniques to be investigated in this report will require linear time invariant models ; the following open loop tests on the flotation plant simulator have only been performed in the *linear operating region*.

The flotation plant simulator is considered to be in the *linear operating region* when each cells' level is in the region between the *froth outflow level* and the *cell overflow*. Figure 2.1 below describes the linear operating region for one of the simulated flotation cells.

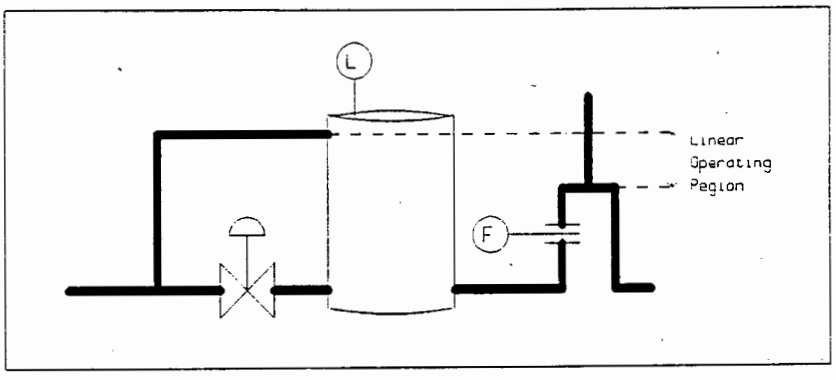


Figure 2.1 Linear Operating Region for a Flotation Cell

Thus, the flotation plant simulator model is only valid while the system is in the linear operating region. The model becomes invalid in any one of the cells' levels moves into a non-linear region (ie. the level drops below the froth outflow or rises above the cell overflow).

2.1.2 Preliminary Open Loop Tests

Preliminary open loop step tests were performed on the flotation simulator in order to asses the approximate time constants and characteristics of the flotation plant simulator. The procedure for executing the preliminary step tests was as follows :-

i) Set valves to a value such that the steady state achieved by the system is within the linear operating region.

ii) Wait until the system has achieved steady state.

iii) Once the system is in steady state, start recording the data (levels and froth flow rates) from the system.

iv) Step one of the valves (open or close), such that the systems' new steady state is still within the linear operating region.

v) Once the new steady state has been achieved, stop recording data from the system.

The data from one such step test is shown below in figure 2.2.

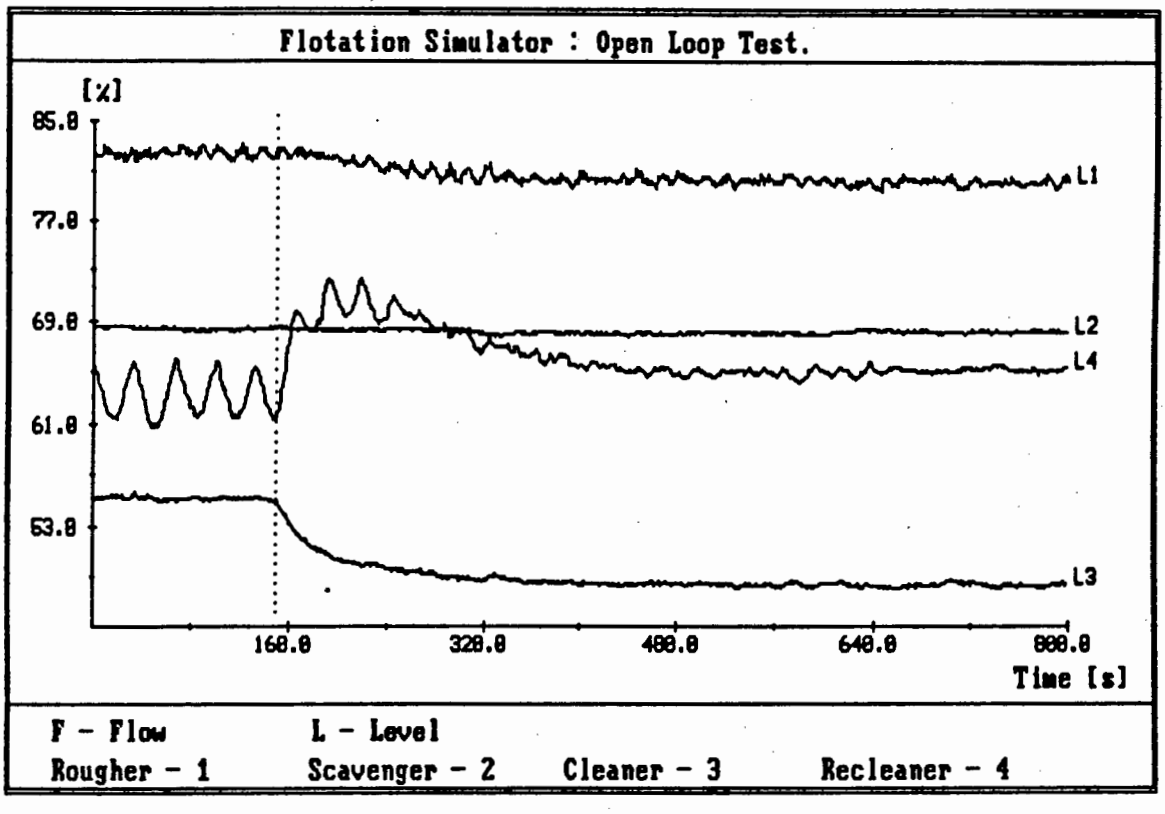


Figure 2.2 Open Loop Step Test - Recleaner Stepped

The following insight was gained from the preliminary tests :-

i) The fastest time constants are in the region of 50 seconds. Thus, a sampling frequency of 2 Hertz would be sufficient to record the data from the flotation plant simulator. And, thus it would also be appropriate to use seconds as the units to describe the time constants.

ii) The flotation simulator attains steady state within 500 seconds after a valve step up to fifteen percent. Thus, data from the flotation simulator must be recorded for at least 500 seconds after the valve step.

iii) Since to the froth flow from each cell was in direct proportion to the product level in the cell (but only while the level was above the froth outflow height or linear operating region), it was decided that the level in each cell would be used as the input to the control algorithm. The control algorithm would be designed to control the level in each cell.

iv) The model of the flotation simulator components are considered as lumped elements. This means that the model will identify the characteristics from valve input to level output in units of 'bits' input and output to the personal computer (ie. none of the components are modeled individually, such as the valves, level probes, etc.).

v) The oscillations visible on the Rougher and Recleaner levels is caused by the feedback sump pumps oscillating. The pumps drain the sumps faster than product is input to the sumps. While the sump is empty or refilling, the pumps cavitate until the level of product in the sump rises to the point which prevents the pumps from cavitating.

The oscillations are therefore a fault within the simulator and is not considered as part of the fluid dynamics of the flotation simulator.

vi) The values to which the valves should be set such that the steady state achieved by the flotation plant simulator would be in the linear operating region were found through the experience gained by performing the preliminary open loop step tests.

The valve settings are listed below (100% represents fully open) :-

Rougher	: 90 %
Scavenger	: 85 %
Cleaner	: 85 %
Recleaner	: 85 %

2.1.3 Final Open Loop Tests

The procedure followed for the final open loop step tests were similar to the tests performed in the preliminary open loop test. The procedure is as follows :-

i) Set the valves to the values used as initial values in the preliminary open loop step tests. These values enable the flotation plant simulator to achieve a steady state condition within the linear operating region.

ii) Wait for the flotation plant simulator to achieve steady state conditions.

iii) Once steady state has been achieved, start recording data from the system (levels and froth flow rates).

iv) Step one of the valves.

v) Stop recording data from the flotation plant simulator once the new steady state condition has been achieved (at least 500 seconds after the step).

NOTE : At no point during the open loop step test was any one of the cells allowed to enter a non-linear region, such as the level rising above the cell overflow or dropping below the froth outflow level.

The data obtained from the open loop step tests are contained in appendix E.

2.2 Developing the Flotation Plant Simulator Model

The model to be developed only concerns the product level in each cell, thus whenever the text refers to step test data, the data being referred to, is the cell product level data.

The process of taking the raw data from the open loop step tests and extracting the necessary modeling information consists of two main steps : Normalising the data and then calculating the transfer function in the frequency domain. The flowchart in figure 2.3 outlines the process.

Each transfer function is processed individually. In other words, the process outlined in figure 2.3 had to be performed for each cells' response to every step input to the system.

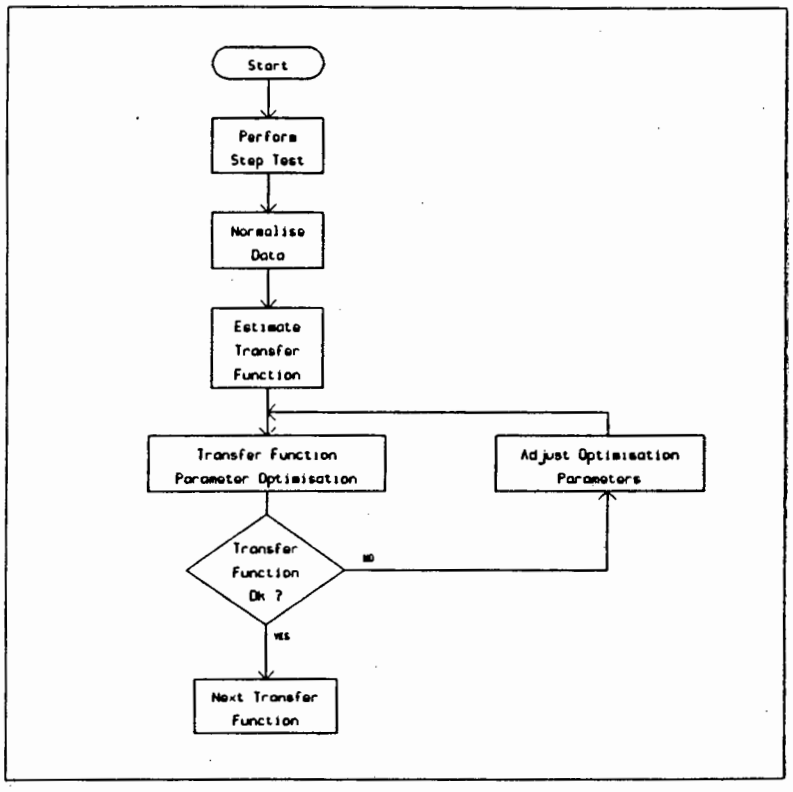


Figure 2.3 Brief Description of Process to Develop a Transfer Function for the Flotation Plant Simulator Model

2.2.1 Normalising the Data

The data from each tank for each step test performed was normalised in the following manner :-

i) The DC offset was subtracted. This is done by averaging the level before the step occurred and subtracting the average from the level data obtained during the test.

ii) The time delay was eliminated (but was recorded for insertion into the transfer function) from the data.

iii) The data was then divided by the magnitude of the step itself. This reduced the actual response to the response to a unit step.

This data was now used to develop the transfer functions to describe the characteristics of the flotation plant simulator.

2.2.2 Calculating the Transfer Functions

The transfer functions were calculated using an algorithm called NELM [2]. NELM requires an initial estimate of the transfer function and then tries to minimise the difference or error between the transfer functions' response and the given data.

2.2.3 Generating the Plant Transfer Function Matrix

The transfer function matrix for the Flotation Plant Simulator was generated column by column (each column of the matrix is generated from the results of a single step test). Each step

test was performed several times and the "column" of transfer functions calculated for each set of results. Each coefficient of the transfer functions were then averaged which resulted in an "averaged column" of transfer functions (of the plant matrix).

The normalised and modeled transfer functions are presented in appendix F. The normalised data presented with each transfer function (in appendix F) is the "best" test result obtained to fit the averaged transfer function.

2.3 The Flotation Plant Simulator Model

The system matrix of transfer functions is given as

$$\begin{bmatrix} y_1 \\ y_2 \\ y_3 \\ y_4 \end{bmatrix} = \begin{bmatrix} g_{11}(s) & g_{12}(s) & g_{13}(s) & g_{14}(s) \\ g_{21}(s) & g_{22}(s) & g_{23}(s) & g_{24}(s) \\ g_{31}(s) & g_{32}(s) & g_{33}(s) & g_{34}(s) \\ g_{41}(s) & g_{42}(s) & g_{43}(s) & g_{44}(s) \end{bmatrix} \begin{bmatrix} u_1 \\ u_2 \\ u_3 \\ u_4 \end{bmatrix}$$

where :

y_1 = Rougher Level
y_2 = Scavenger Level
y_3 = Cleaner Level
y_4 = Recleaner Level

u_1 = Rougher Valve
u_2 = Scavenger Valve
u_3 = Cleaner Valve
u_4 = Recleaner Valve

The input and output units for the transfer functions are in 'bits' ; since the data interface to the flotation simulator is via 12-bit analog/digital/analog computer interface devices. The transfer function time constants are in seconds, as are the time delays.

The transfer functions are given overleaf in table 2.1. The transfer function matrix and the poles and zeros of the transfer functions are contained in appendix G.

Element	Transfer Function	Delay
g_{11}	$\frac{(2.000 \times 10^{-2})s + (9.125 \times 10^{-5})}{s^2 + (6.460 \times 10^{-2})s + (7.300 \times 10^{-4})}$	0.0 [sec]
g_{12}	$\frac{(1.350 \times 10^{-2})}{s + (5.930 \times 10^{-3})}$	20.0 [sec]
g_{13}	$\frac{(-6.651 \times 10^{-3})}{s^2 + (2.948 \times 10^{-1})s + (4.434 \times 10^{-3})}$	0.0 [sec]
g_{14}	$\frac{(-7.060 \times 10^{-4})}{s + (7.865 \times 10^{-3})}$	0.0 [sec]
g_{21}	$\frac{(-1.342 \times 10^{-2})}{s + (2.975 \times 10^{-2})}$	0 .0 [sec]
g_{22}	$\frac{(1.640 \times 10^{-2})}{s + (9.574 \times 10^{-3})}$	0.0 [sec]
g_{23}	$\frac{(-4.890 \times 10^{-3})}{s + (1.418 \times 10^{-2})}$	20.0 [sec]
g_{24}	$\frac{(-2.860 \times 10^{-4})}{s + (7.580 \times 10^{-3})}$	0.0 [sec]

Table 2.1 Flotation Plant Simulator Transfer Functions.

Element	Transfer Function	Delay
g_{31}	$\frac{(7.610 \times 10^{-3})s + (1.007 \times 10^{-5})}{s^2 + (1.949 \times 10^{-2})s + (8.774 \times 10^{-5})}$	0.0 [sec]
g_{32}	$\frac{(9.388 \times 10^{-3})}{s + (3.356 \times 10^{-3})}$	32.0 [sec]
g_{33}	$\frac{(7.000 \times 10^{-2})s + (6.560 \times 10^{-5})}{s^2 + (5.820 \times 10^{-2})s + (4.100 \times 10^{-4})}$	0.0 [sec]
g_{34}	$\frac{(-5.871 \times 10^{-3})}{s + (2.190 \times 10^{-2})}$	0.0 [sec]
g_{41}	$\frac{(1.750 \times 10^{-2})s + (4.725 \times 10^{-5})}{s^2 + (2.400 \times 10^{-2})s + (1.350 \times 10^{-4})}$	27.0 [sec]
g_{42}	$\frac{(1.917 \times 10^{-2})}{s + (3.176 \times 10^{-3})}$	70.0 [sec]
g_{43}	$\frac{(2.654 \times 10^{-2})s + (1.182 \times 10^{-4})}{s^2 + (1.121 \times 10^{-2})s + (1.540 \times 10^{-4})}$	10.0 [sec]
g_{44}	$\frac{(1.500 \times 10^{-2})s + (2.813 \times 10^{-5})}{s^2 + (4.000 \times 10^{-2})s + (3.750 \times 10^{-4})}$	0.0 [sec]

Table 2.1 Flotation Plant Simulator Transfer Functions (contd).

Chapter 3

The Characteristic Locus Design Method

Multivariable frequency response methods have been developed in recent years which provide a logical, well defined approach to the multi-input multi-output control system design problem [4]. The methods are a natural extension of the single loop frequency response approach of Bode [5] and Nyquist [6], used widely in servo system design. Control schemes can be designed to specified stability margins, accuracy, speed of response, interaction levels and system integrity.

MacFarlane [7] documented the application of Bode's [5] concept of return difference and return ratio to the analysis of multivariable feedback systems. The matrix transfer functions are regarded as operators on linear vector spaces over the field of rational functions in the complex variable s. The eigenvalues of these operators are identified as characteristic transfer functions whose corresponding characteristic frequency responses provide a link between classical single loop-design techniques and multivariable system feedback theory. Macfarlane [7] also introduced the concept of a design technique based on the frequency response loci associated with a set of characteristic transfer functions for a multivariable feedback system.

A multivariable design technique based on the exploitation of the properties of linear vector spaces defined over base fields of functions of a complex variable was introduced by Belletrutti and MacFarlane [8,9] and developed further and formally presented by Kouvaritakis [10]. The results obtained demonstrated that the classical Bode-Nyquist theory is essentially the special scalar case of a completely general

vector theory. Thus the characteristic loci design technique is based on generalisations of the classical Bode-Nyquist approach.

The multivariable design technique described by Belletrutti and MacFarlane [8,9] and Kouvaritakis [10] provided methods for assessing stability, integrity, interaction and accuracy of the multivariable feedback system. The literature which followed the initial publications either studied the behavior of the root-loci under specific conditions (for example Kouvaritakis and Shaked [11]) or presented techniques for manipulating the root-loci (for example Owens [12]) of multivariable systems.

Although the initial literature [7,8,9] concerning the design technique presented a technique for assessing system stability (the encirclement theorem), no formal proof was presented. Rosenbrock and Cook [13] concluded that no such stability theorem could be obtained without some condition on the plant matrix. But, MacFarlane and Postlethwaite [14] published a comprehensive discussion of the background to the generalised stability criterion and provided a proof of the results.

In 1981, Postlethwaite [15] introduced the concepts of principal gain and principal phase and their use in analysis of feedback behavior for linear multivariable systems. A Nyquist-type stability criterion is presented in terms of these quantities and is used to characterise the robustness of the closed-loop stability property when the system model is subjected to a linear perturbation at any point in the feedback configuration.

This chapter is intended as an overview of the theory of the Characteristic Locus design technique and associated subject matter. The first section describes some fundamental relationships and conventions used throughout the text. This is

followed by the development of the open loop to closed loop relationship for multivariable systems expressed as a set of characteristic vectors. The performance analysis and characteristic locus design method is then explained, also in terms of the system characteristic values and vectors, which is followed by a few comments concerning the characteristic locus method as a multivariable design technique. The last section describes a technique for determining the robustness of a multivariable controller using the system principal gains and phases. This section is also concluded by comment on the technique.

3.1 Fundamental Multivariable Relationships

The multivariable feedback configuration used throughout this text is shown below in figure 3.1 where :-

- r(s) - vector of reference input transforms (order *m*)
- e(s) - vector of error transforms (order *m*)
- u(s) - vector of plant input transforms (order *l*)
- y(s) - vector of output transforms (order *m*)
- K(s) - *l* X *m* matrix of controller transfer functions
- G(s) - *m* X *l* matrix of plant transfer functions
- H(s) - *m* X *m* matrix of feedback transducer transfer functions
- I_m - *m* X *m* identity matrix.

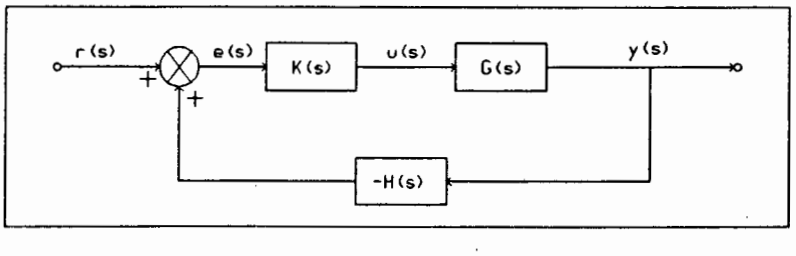


Figure 3.1 Multivariable Feedback System Configuration

It is convenient to denote the product G(s)K(s) which occurs repeatedly by

$$Q(s) = G(s)K(s) \tag{3-1}$$

where $Q(s)$ = System open-loop transfer function matrix.

The closed-loop transfer function matrix, R(s), for this system may be written in the form

$$R(s) = [I_m + G(s)K(s)H(s)]^{-1}G(s)K(s) \tag{3-2}$$

Suppose that in figure 3.1 all the feedback loops are broken at y(s), and that a signal transform vector $\alpha(s)$ is injected at this point. Then the returned signal transform vector is given as

$$-G(s)K(s)H(s)\alpha(s)$$

The difference between injected and returned signals is given by

$$[I_m + G(s)K(s)H(s)]\alpha(s) = F_y(s)\alpha(s) \tag{3-3}$$

where

$$F_y(s) = I_m + G(s)K(s)H(s) \tag{3-4}$$

is a square matrix defined as the system *return-difference matrix* [7] measured at the output side of the plant.

The matrix

$$T_y(s) = G(s)K(s)H(s) \tag{3-5}$$

is defined as the system *return-ratio matrix* measured at the output side of the plant. It then follows that

$$F_y(s) = I_m + T_y(s). \tag{3-6}$$

The above 'loop-breaking' approach to the definition of return-ratio and return-difference quantities need not be restricted to the system output vertex, y(s). For instance, if the feedback loops were broken at the points corresponding to the signals u(s) and e(s) respectively, and the above analysis repeated, the results would be

$$T_u(s) = K(s)H(s)G(s) \tag{3-7}$$
$$= \text{return-ratio matrix for plant input vertex.}$$

$$T_e(s) = H(s)G(s)K(s) \tag{3-8}$$
$$= \text{return-ratio matrix for system error vertex.}$$

The corresponding return-difference operators then become

$$F_u(s) = I_l + T_u(s) \tag{3-9}$$

$$F_e(s) = I_m + T_e(s) \tag{3-10}$$

In terms of these return difference matrices, simple algebraic manipulations give the following equivalent forms for the

closed-loop system transfer function matrix relating y(s) and r(s)

$$R(s) = [F_y(s)]^{-1}G(s)K(s)$$

$$= G(s)[F_u(s)]^{-1}K(s)$$

$$= G(s)K(s)[F_e(s)]^{-1} \qquad (3\text{-}11)$$

3.2 Open Loop / Closed Loop Relationships

Since the control engineer is concerned with the closed-loop performance of the system and the technique is based on the frequency response of system characteristic transfer functions and characteristic vectors, a set of open-loop to closed-loop relationships would be appropriate.

Let the *m* X *m* open-loop transfer function Q(s) matrix have a set of distinct characteristic transfer functions and associated linearly independent characteristic vectors denoted respectively by $q_i(s)$ and $w_i(s)$ for i = 1,2,..,*m* ; Form the identically non-singular matrix

$$W(s) = [w_1(s)\ w_2(s)\ .\ .\ .\ w_m(s)] \qquad (3\text{-}12)$$

and invert it to give

$$V(s) = W^{-1}(s) = \begin{bmatrix} v_1(s)^t \\ v_2(s)^t \\ \vdots \\ v_m(s)^t \end{bmatrix} \qquad (3\text{-}13)$$

where the symbol t denotes transposition so that $\{v_i(s)^t\}$ are the rows of $V(s)$. Then standard algebraic relationships [16] give that $Q(s)$ may be expressed in the form

$$Q(s) = W(s)[\text{diag}\{q_i(s)\}]V(s) \qquad (3\text{-}14)$$

Alternatively, in the dyadic form [16],

$$Q(s) = \sum_{i=1}^{m} q_i(s) w_i(s) v_i(s)^t \qquad (3\text{-}15)$$

For unity feedback systems, with $H(s) = I_m$, the closed-loop transfer function matrix as given in equation (3-2) may be written as

$$R(s) = [I_m + Q(s)]^{-1} Q(s) \qquad (3\text{-}16)$$

from which it can readily be shown that

$$R(s) = W(s)[\text{diag}\{q_i(s)/(1+q_i(s))\}]V(s) \qquad (3\text{-}17)$$

or, in dyadic form, that

$$R(s) = \sum_{i=1}^{m} \left[\frac{q_i(s)}{1 + q_i(s)} \right] w_i(s) v_i(s)^t \qquad (3\text{-}18)$$

Thus, for the case of unity feedback, two interesting relationships have emerged :-

i) The characteristic transfer functions of the open-loop and closed loop systems are both functions of the

characteristic transfer functions :-

Open-loop system : $q_i(s)$ for i=1,2,...,m

Closed-loop system : $q_i(s)/(1+q_i(s))$ for i=1,2,...,m

ii) The characteristic vectors are the same for both the open-loop and closed-loop systems, namely

Characteristic vectors : $w_i(s)$ for i=1,2,...,m

3.3 Performance Analysis

3.3.1 Stability

F(s) is the system return-difference matrix which can represent any of the corresponding quantities introduced in section 3.1.1.
Let the characteristic transfer functions of the return-ratio matrix T(s) be $t_i(s)$ for i=1,2,...,m. Then

$$\det F(s) = \det [I_m + T(s)]$$

$$= \prod_{i=1}^{m} [1 + t_i(s)] \qquad (3\text{-}19)$$

(NOTE : $t_i(s)$ are simply $q_i(s)$ when $H(s)=I_m$)

Let the $t_i(s)$ map the usual Nyquist contour into the set of *m characteristic loci* denoted by $t_i(jw)$, i=1,2,...,m. Then it may be shown that, if p_o is the number of right-half plane zeros in the open-loop characteristic polynomial, and ΣN_{ti} (i=1,2,..,m)

is the net sum of clockwise encirclements of the critical point (-1,0) in the complex plane contributed by the characteristic loci of T(s), the closed-loop system is stable if and only if

$$\sum_{i=1}^{m} N_{ti} = -p_o \qquad (3\text{-}20)$$

Clockwise encirclements are counted positive corresponding to a clockwise traversal of the nyquist contour.

The above encirclement theorem can be used in determining system closed-loop stability boundaries. More specifically, let the return-ratio matrix for a given system be T(s). Now apply a gain of k to each loop so that the return-ratio matrix for the modified system is

$$T_1(s) = kT(s)$$

The characteristic loci corresponding to $T_1(s)$ are equal to those of T(s) scaled by the factor k. In the complex plane, the stability of the new system is determined by applying the encirclement theorem to the original system characteristic loci with $(-1/k,0)$ as the critical point. Thus ,it is possible to find the limiting gain factor k which preserves overall stability when applied to each loop.

3.3.2 Interaction

The term interaction is used to denote the set of relationships influencing the way in which a reference input, r(s), applied

to input i, affects the set of outputs $\{y_j(s) : j <> i\}$.

Low Frequencies :
Consider the characteristic dyadic expansion of the closed-loop operator, R(s), given by equation (3-18) and evaluated for s=jw. This gives

$$R(s) = \sum_{i=1}^{m} \left[\frac{q_i(jw)}{1 + q_i(jw)} \right] w_i(jw) v_i(jw)^t \qquad (3\text{-}21)$$

Now at some low frequency, w_l, if high characteristic gains are imposed

$$|q_i(jw_l)| >> 1 \qquad \text{for } i=1,2,\ldots,m \qquad (3\text{-}22)$$

Then

$$R(jw_l) \rightarrow \sum_{i=1}^{m} w_i(jw_l) v_i(jw_l)^t = I_m \qquad (3\text{-}23)$$

and the closed-loop system is noninteracting at this frequency. Thus, at low frequencies, interaction can be suppressed by ensuring the moduli of all the characteristic loci are sufficiently large.

High Frequencies :
At high frequencies, condition (3-22) cannot be met due to the plant being band-limited, so

$$|q_i(jw_h)| << 1 \qquad \text{for } i=1,2,\ldots,m \qquad (3\text{-}24)$$

at the high frequency w_h. Then from equation (3-21)

$$R(jw_h) \rightarrow \sum_{i=1}^{m} q_i(jw_h) w_i(jw_h) v_i(jw_h)^t = Q(jw_h) \tag{3-25}$$

which means that any high frequency cross-couplings in $Q(jw)$ will pass straight through to $R(jw)$ despite the action of feedback.

One method of suppressing high frequency interaction is to ensure that the characteristic direction set of $Q(jw)$ is aligned with the standard basis set. Thus a convenient measure of alignment is the angle, as function of frequency, between the vectors $w_j(jw)$ and the vectors e_j for j=1,2,..,m (where e_j is a standard basis vector, the jth column of a unit matrix of appropriate order). The angle formed by the two vectors in the complex plane is called the *misalignment angle* and is given as

$$\cos \theta_j(jw) = \frac{|w_j(jw), e_j)|}{\|w_j(jw)\|} \tag{3-26}$$

where $w_j(jw)$ is that characteristic direction which produces the minimum $\theta_j(jw)$ at frequency w. Thus if $\theta_j(jw)$ is sufficiently small at high frequencies, interaction effects arising from the jth input will be correspondingly small.

(NOTE : $Q(jw)$ may only be considered diagonally dominant when all the θ_j are in exact alignment with standard basis vectors.)

In summary, to asses interaction over the frequency range, it is convenient to display the misalignment angles as a function of frequency together with a plot of the modulus of the corresponding characteristic loci against frequency.

3.3.3 Integrity

A multivariable feedback system is said to be of satisfactory integrity if the system remains stable under all combinations of a stipulated set of failure conditions.

To ensure integrity against failure of the output transducer in loop j say, the characteristic loci of the principal sub-matrix of the return-ratio matrix $T_y(s)$, obtained by deleting row j and column j, must satisfy the Nyquist stability criterion, as defined by equation (3-20).

To check for integrity against actuator and error monitoring channel failures, similar considerations to those above apply to $T_u(s)$ and $T_e(s)$ respectively. For the case of when $H(s) = I_m$, $T_e(s)=T_y(s)$ so that integrity against transducer failures automatically insures integrity in the error monitoring channels.

3.3.4 Accuracy

In general, accuracy can be defined as the degree to which actual system outputs follow the desired system outputs. That is we wish to have

$$y(jw) = r(jw)$$

The system accuracy will be high providing the moduli of the characteristic loci are suitably large at low frequencies, ie.

$$|q_i(jw)| >> 1 \qquad \text{for } i=1,2,\ldots,m.$$

An exception to this can be found in the case of badly skewed characteristic direction vectors; for an assessment of accuracy in this case, the method reverts to a different set of frequency response plots called the *principal gain loci* [15].

3.4 The Characteristic Locus Method

The general problem of multivariable feedback control system design can be looked on in terms of choosing the controller matrix, K(s), so that the characteristic loci (or eigenvalues) of the matrix G(s)K(s) have certain prescribed properties. The main forms of manipulation which must be performed via K(s) include :

i) Modifying the phases of appropriate sets of characteristic loci in order to achieve acceptable stability and integrity.

ii) Aligning the characteristic directions at high frequencies and balancing the gains of the characteristic loci at low frequencies in order to reduce the interaction to acceptable levels.

iii) Injecting gain into the phase compensated and aligned system to achieve satisfactory overall performance.

The final system controller has many criteria to satisfy simultaneously. Thus, K(s) is designed as the cascaded combination of several sub-controllers, $K_i(s)$, so that

$$K(s) = \prod_{i=1}^{m} K_i(s) \qquad (3\text{-}27)$$

where each $K_i(s)$ has a specific task to handle during the sequential synthesis.

The final controller matrix, K(s), must satisfy the following for obvious reasons :

i) The dynamical elements must be rational functions in *s*.

ii) The result of det K(s), must be identically non-singular and must not have any right-half plane zeros (to avoid non-minimum phase difficulties).

iii) All the poles of K(s) should lie in the open left-half plane.

The characteristic loci method is iterative ; the design process alternating between *system analysis* and *design decision* until the final specifications have been achieved.

Listed below are several types of controller factors which can be used in achieving the desired manipulations of the systems characteristic loci and directions. In order to simplify the descriptions of the controllers, unity feedback, $H(s) = I_m$, is assumed; then the transition from open-loop to closed-loop characteristics is straight-forward as described in section 3.3.

3.4.1 Permutation Matrix Controller

$$K_i(s) = [e_1 \cdots e_{q-1} e_p e_{q+1} \cdots e_{p-1} e_q e_{p+1} \cdots e_m] \qquad (3\text{-}28)$$

where $p>q$ and e_j is the jth column of I_m. This controller has the effect of interchanging columns p and q of the system G(s). This controller is used in the initial stages of design when it is required, for some specific technical reason, to reorder the inputs to G(s).

3.4.2 Elementary Transformation Matrix Controllers

The *mxm* elementary transformation matrices, $K_i(s)$, can be of the following two types :

NOTE : Zero elements not shown

i)
$$K_i(s) = \begin{bmatrix} 1 & & & & & \\ & 1 & & & & \\ & & \cdot & & & \\ & & & k_{jj}(s) & & \\ & & & & 1 & \\ & & & & & 1 \end{bmatrix} \qquad (3\text{-}29)$$

where the on-diagonal entry $k_{jj}(s)$ is a rational function of s, having all its poles and zeros in the open left-half plane.
Post-multiplying the plant, G(s), by the above matrix corresponds to the elementary column operation of multiplying all the elements of the jth column of G(s) by $k_{jj}(s)$.

ii)
$$K_i(s) = \begin{bmatrix} 1 & & & & & \\ & 1 & & & & \\ & & \cdot & & & \\ & & & 1 \;\; k_{jk}(s) & & \\ & & & & 1 & \\ & & & & & 1 \end{bmatrix} \qquad (3\text{-}30)$$

where the off-diagonal entry $k_{jk}(s)$ is a rational function of s, having all its poles in the open left-half plane. However its zeros may be in the right-half plane.

Post-multiplying the plant, G(s), by the above matrix corresponds to the elementary column operation of adding $k_{jk}(s)$ times the jth column of G(s) to the kth column of G(s).

In order to investigate the effect of the elementary transformation matrices, the analysis proceeds as follows :

$$|T(s)| = |Q(s)| = |G(s)||K(s)|$$

$$= \prod_{i=1}^{m} t_i(s) \qquad (3\text{-}31)$$

where $t_i(s)$ (i=1,2,..,m) are the characteristic loci of T(s).

Taking logorithms and separating the magnitude and phase characteristics gives

$$\sum_{i=1}^{m} \ln|t_i(s)| = \ln|\det K(jw)| + \sum_{i=1}^{m} \ln|q_i(jw)| \qquad (3\text{-}32)$$

and

$$\sum_{i=1}^{m} \text{phase}\{t_i(s)\} = \text{phase}\{\det K(jw)\} + \sum_{i=1}^{m} \text{phase}\{q_i(jw)\} \qquad (3\text{-}33)$$

where $q_i(jw)$ (i=1,2,..,m) are the characteristic loci of G(s).

If K(s) is an elementary controller described by equation (3-29) then

$$\det K(jw) = k_{jj}(jw) \qquad (3\text{-}34)$$

and equations (3-32) and (3-33) become

$$\sum_{i=1}^{m} \ln|t_i(s)| = \ln|k_{jj}(jw)| + \sum_{i=1}^{m} \ln|q_i(jw)| \qquad (3\text{-}35)$$

and

$$\sum_{i=1}^{m} \text{phase}\{t_i(s)\} = \text{phase}\{k_{jj}(jw)\} + \sum_{i=1}^{m} \text{phase}\{q_i(jw)\} \tag{3-36}$$

respectively. This shows that the gain and phase shift contributed by the controller term $k_{jj}(jw)$ is shared among some or all of the characteristic loci, $q_i(jw)$, to form the new characteristic loci, $t_i(jw)$.

If K(s) is an elementary controller described by equation (3-30) then

$$\det K(jw) = 1 \tag{3-37}$$

and equations (3-32) and (3-33), respectively, yield

$$\sum_{i=1}^{m} \ln|t_i(s)| = \sum_{i=1}^{m} \ln|q_i(jw)| \tag{3-38}$$

and

$$\sum_{i=1}^{m} \text{phase}\{t_i(s)\} = \sum_{i=1}^{m} \text{phase}\{q_i(jw)\} \tag{3-39}$$

Thus, any increase in the gain or phase margin in one or some of the characteristic loci, $t_i(jw)$, results in a corresponding decrease in the net gain and phase margins in one or some of the remaining characteristic loci, and vica-versa.

Note : The problem with the technique described above is that the control engineer does not have any information on how the gains and phases of the characteristic loci will be affected by the presence of a elementary transformation matrix.

3.4.3 Matrix P.I. Controller

$$K_i(s) = K_\beta + K_\delta/s \qquad (3\text{-}40)$$

where K_β and K_δ are *mxm* non-singular matrices of constants so that $K_i(s)$ is a matrix generalisation of the scalar P.I. controller. The controller may be designed by putting

$$K_\beta = K_\infty D_1 \qquad (3\text{-}41)$$

$$K_\delta = K_0 D_2 \qquad (3\text{-}42)$$

where K_0 and K_∞ respectively diagonalise the system at zero and high frequencies. The matrices D_2 and D_1 are used to adjust the weighting between zero and infinite frequencies in each column of $G(s)K_i(s)$.

An obvious choice for K_0 is the d.c. plant inverse :

$$K_0 = G^{-1}(0) \qquad (3\text{-}43)$$

which diagonalises the plant at d.c. and thus has the effects of :

i) Aligning the plant characteristic directions with the standard basis vectors at d.c., removing any d.c. interaction.

ii) Normalising the characteristic loci by assigning a gain of one to each locus at d.c. This makes it easier to meet low frequency performance specifications for cases where some of the plant characteristic loci are large at d.c. and others are relatively small.

To find K_∞, multiply $g_i(s)$, ie. row i of G(s), by s^{pi} where the integer pi is chosen so that as $|s|$ -> ∞ no element of $s^{pi}g_i(s)$ tends to infinity and not every element tends to zero. Now define the row vector

$$b_i = \lim_{s\infty} s^{pi} g_i(s) \qquad (3\text{-}44)$$

Repeat the above procedure for all the remaining rows of G(s) and then assemble the vectors b_i, i=1,2,...,m into a matrix B. Then

$$K_\infty = B^{-1} \qquad (3\text{-}45)$$

The above operations, in retrospect, will show that the magnitudes of the off-diagonal elements in each row of $G(s)K_\infty$ become arbitrarily small relative to the diagonal elements as $|s|$ -> ∞.

This form of matrix P.I. controller performs the following set of functions :

i) The controller eliminates steady state error and low frequency interaction by virtue of the integral action since ;

$$|q_i(s)| >> 1 \quad \text{as} \quad |s| \to 0$$

ii) The controller reduces high frequency interaction by ensuring that $Q(s) = G(s)K_i(s)$ approaches diagonal form as $|s|$ -> ∞. The characteristic vectors are therefore aligned with the standard basis vectors at high frequencies.

3.4.4 Commutative Controllers

The problem with designing or choosing the individual elements of K(s) (to achieve the desired modification of the characteristic loci of G(s)) is that the location of the eigenvalues of the product of two matrices is often not known. One solution to this problem is to ensure that the matrices commute (ie. the matrices have a common set of eigenvectors).

A commutative controller would be designed by synthesising m classical single-loop controllers in the eigenframework of the plant transfer function matrix, and then 'transforming back' to the original reference basis to obtain the required controller matrices.

Let $q_1(s),\ldots q_m(s)$ be the eigenvalues (characteristic transfer functions) of plant transfer function matrix, G(s), with corresponding eigenvector set $w_1(s),\ldots,w_m(s)$ (characteristic direction functions) and associated matrices W(s), V(s) as defined in equations (3-12) and (3-13) of section 3.2. For a commutative controller, the eigenframework of G(s) and Q(s) will be the same. G(s) can then be expressed in the dyadic form as

$$G(s) = \sum_{j=1}^{m} w_j(s) q_j(s) v_j(s)^t \qquad (3\text{-}46)$$

Let

$$K(s) = \sum_{k=1}^{m} w_k(s) k_k(s) v_k(s)^t \qquad (3\text{-}47)$$

Then

$$Q(s) = G(s)K(s)$$

$$= \sum_{i=1}^{m} w_i(s) q_i(s) k_i(s) v_i(s)^t \qquad (3\text{-}48)$$

since
$$v_i(s)^t w_j(s) = \begin{cases} 1 & \text{if } i = j \\ 0 & \text{if } i <> j \end{cases} \qquad (3\text{-}49)$$

as
$$V(s)W(s) = I_m \qquad (3\text{-}50)$$

Then
$$R(s) = \{I_m + G(s)K(s)\}^{-1}G(s)K(s)$$

$$= \sum_{i=1}^{m} w_i(s) \left[\frac{q_i(s)k_i(s)}{1 + q_i(s)k_i(s)} \right] v_i(s)^t$$

$$= W(s)\Omega(s)V(s) \qquad (3\text{-}51)$$

where
$$\Omega(s) = \text{diag} \left[\frac{q_i(s)k_i(s)}{1 + q_i(s)k_i(s)} \right] \qquad (3\text{-}52)$$

and
$$K(s) = W(s)\{\text{diag } k_i(s)\}V(s) \qquad (3\text{-}53)$$

From (3-51) it follows that
$$R(s) \rightarrow \sum_{i=1}^{m} w_i(s)v_i(s)^t$$

$$\text{as } k_i \rightarrow \infty \quad \text{for } i=1,2,\ldots,m$$

The above relationships exhibit the idea of the commutative controller. The transformation of basis to the plant eigenframework means expressing general signal transform vectors as linear combinations of those vectors (the

eigenvectors of G(s))) to which the plant appears as a set of single scalar transfer functions, namely the characteristic transfer functions (eigenvalues - $q_i(s)$). In this basis, the control engineer would try to carry out *m* single loop designs to choose the set of single loop controllers, $k_i(s)$ and finally transform back to obtain the actual controller K(s) by means of equation (3-53).

One of the major problems concerning this technique of designing multivariable controllers is that if Q(s) has significant off diagonal terms at high frequencies, the interaction terms cannot be suppressed by the design of scalar systems which specify the behavior of the system characteristic functions. This is due to the stability requirement that the gain is reduced at high frequencies, thus high frequency interaction cannot be suppressed. One solution to this problem is to first make the system diagonally dominant [17].

Despite this drawback, Kouvaritakis [10,18,19] has developed a technique which employs the method described above in the form of an approximate commutative controller. The controller commutates approximately with the system at a particular frequency.

3.5 Comments on the Characteristic Locus Method

i) *Loss of information* : Since the characteristic locus technique depends on the eigenvalues and eigenvectors of the system matrix ; If the system is upper or lower triangular the eigenvalues will not indicate the off-diagonal term(s) (ie. Information concerning the off-diagonal characteristics will not be shown). The system will appear to have no interaction from any off-diagonal elements or diagonally dominant.

ii) *Misleading Angles* : Small misalignment angles does not necessarily imply a system which is nearly diagonal or diagonally dominant. An example [9] of this is the system

$$Q(s) = \frac{1}{0.99(s+1)} \begin{bmatrix} 9 & 9 \\ -9 & 99.9 \end{bmatrix}$$

which produces two equal misalignment angles of 5.7 degrees at all frequencies. The criteria mentioned in section 3.3.2 above implies that the system is non-interacting although Q(s) is not diagonally dominant.

Another point concerning the calculation of misalignment angles (section 3.3.2) is that the characteristic directions (eigenvectors), w_i, can be compared with any of the elements of the standard basis set, $(e_1, e_2, \ldots, e_m)$ which will result in a matrix of misalignment angles. The literature does not specify which elements should be used in determining high frequency interaction.

iii) *Low/High frequency crossover ? :* The level of interaction is determined, with the characteristic loci technique, by the magnitude of the moduli of the characteristic loci (Bode plots) at low frequencies, and the alignment of the characteristic directions with the standard basis at high frequencies. But, the control engineer has no indication as to what constitutes "low frequency" or "high frequency" with respect to the system under investigation.

iv) *Identification of problems within the system :* Since the characteristic loci are a characteristic of the system matrix as whole it is virtually impossible to trace any characteristic of the loci back to the original system unless the system is diagonally dominant.

Since the technique cannot identify which part of the original system is the source of a particular problem, the technique may not provide enough information to enable the control engineer to design a compensator to solve the problem.

v) *A stability indicator at all times :* The stability and integrity of the system can be determined at any stage of compensation or controller design. This is unlike the Inverse Nyquist Array (INA) technique [17] which requires the system to be diagonally dominant before stability can be assessed.

3.6 Robustness in Multivariable Control System Design

Frequency response techniques have the advantage of being largely insensitive to small errors in the system model. Should the actual system suffer from large parameter variations or inaccuracies, however, then the control system should be designed to have a large degree of stability. The mere presence of feedback is not sufficient to guarantee the robustness of the stability property, and so techniques to asses the relative stability of a multivariable design have been developed [15].

For large and possibly dynamic perturbations, the robustness of the closed-loop stability property is characterised by the *principal gains* and *principal phases* of the input and output relative stability matrices. From Bode-type plots of the principal values, multivariable gain and phase margins can be defined which are analogous to the classical single-loop stability margins.

3.6.1 Relative Stability Matrices

In order to study the robustness of the closed-loop stability property with respect to large and possibly dynamic perturbations consider the feedback configuration of figure 3.2, where I+dG(s) represents a multivariable perturbation of the plant from G(s) to G(s)+G(s)dG(s). The configuration can be redrawn as shown in figure 3.3, where the perturbation dG(s) now appears in series with a transfer function $R_i(s)$, given by

$$R_i(s) \stackrel{D}{=} [I+(K(s)G(s))^{-1}]^{-1}$$

$R_i(s)$ will be called the *relative stability matrix* with respect

in figure 3.4. This configuration can redrawn as shown in figure 3.5, where dG(s) now appears in series with a transfer function $R_0(s)$, given by

$$R_0(s) \stackrel{D}{=} [I+(G(s)K(s))^{-1}]^{-1}$$

$R_0(s)$ will be called the *relative stability matrix* with respect to uncertainty at the plant output, or simply the *output relative stability matrix.*

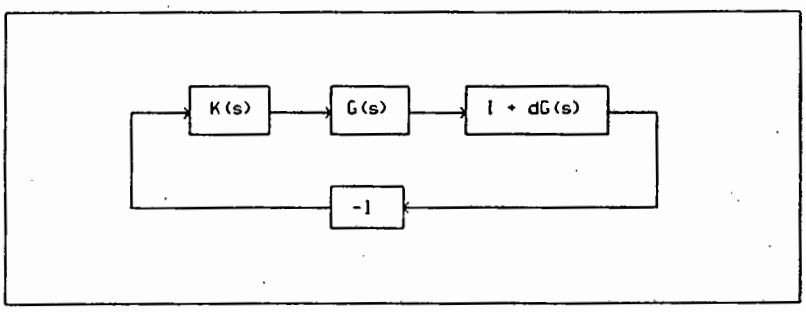


Figure 3.4 Dynamic Uncertainty at Plant Output

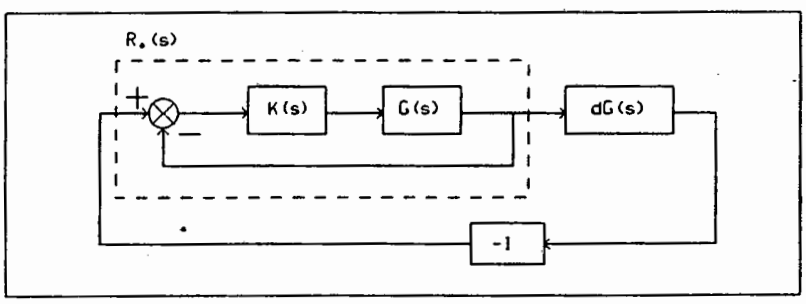


Figure 3.5 A rearrangement of Figure 3.4

3.6.2 Principal Gains and Principal Phases

The gain and phase information from which the multivariable gain and phase margins are derived stems from the polar decomposition of a complex matrix [20] which is now defined.

Analogous to the polar form of a complex number, a complex matrix R can be represented in the forms

$$R = UH_R \tag{3-54}$$

$$R = H_L U \tag{3-55}$$

where U is unitary and H_R, H_L are positive semidefinite Hermitian matrices. H_R and H_L are often called right and left moduli, and are uniquely determined as $(R^*R)^{\frac{1}{2}}$ and $(RR^*)^{\frac{1}{2}}$ respectively, where * denotes complex conjugate transpose, and U is uniquely defined via (3-54) and (3-55) when R is nonsingular.

Representations (3-54) and (3-55) are called *Polar Decompositions* of R, and are easily determined from the *Singular Value Decomposition* of R [21].

If R has the singular value decomposition

$$R = X\Sigma V^* \qquad ; \text{ X, V unitary, and } \Sigma \text{ diagonal and } \geq 0$$

then

$$\begin{aligned} R &= (XV^*)(V\Sigma V^*) \\ &= UH_R \end{aligned}$$

and

$$\begin{aligned} R &= (X\Sigma X^*)(XV^*) \\ &= H_L U \end{aligned}$$

From these polar decompositions the following key are defined :

i) The eigenvalues of R are called *characteristic gains*.
ii) The eigenvalues of the Hermitian part in either polar

decomposition of R are called *principal gains*, or *singular values*.

iii) The arguments of the eigenvalues of the unitary part of the polar decompositions of R are called *principal phases*.

These three quantities are related by two theorems :

Theorem 1 The magnitudes of the characteristic gains are bounded above and below by the maximum and minimum principal gains.

Theorem 2 If the principal phases have a spread of less than π (180 degrees) , then the arguments of the characteristic gains are bounded above and below by the maximum and minimum principal phases.

3.6.3 Multivariable Gain and Phase Margins

The theorems stated above in section 3.6.2 have been used as a basis by Postelthwaite, et al [15], to derive the robustness results discussed in this section.

Let $R(jw)$ have principal gains and principal phases

$$\alpha_1(w) \leq \alpha_2(w) \leq \ldots \leq \alpha_m(w)$$

and

$$\theta_1(w) \leq \theta_2(w) \leq \ldots \leq \theta_m(w)$$

respectively ; it is assumed that the $\{\theta_i(w)\}$ have a spread of less than π. Similarly, let the principal gains and phases of $dG(jw)$ be

$$\delta_1(w) \leq \delta_2(w) \leq \ldots \leq \delta_m(w)$$

and

$$\epsilon_1(w) \leq \epsilon_2(w) \leq \ldots \leq \epsilon_m(w)$$

respectively. Also, let the condition numbers [21] of $R(jw)$ and $dG(jw)$, using the l_2-induced norm (Euclidean norm of a vector, ie. $(x^*x)^{\frac{1}{2}}$), be $c_1(w)$ and $c_2(w)$, respectively, so that

$$c_1(w) \stackrel{D}{=} \alpha_m(w)/\alpha_1(w)$$

and

$$c_2(w) \stackrel{D}{=} \delta_m(w)/\delta_1(w)$$

and also define

$$\Omega_m(w) \stackrel{D}{=} \tan^{-1}\left[\frac{[c_1(w)-1]c_2(w)}{1 - [c_1(w)-1]c_2(w)}\right]$$

which will be referred to as a phase modifier.
Then the following theorems can be stated.

Theorem 3 (Small gain theorem) The perturbed closed-loop system remains stable, if

i) $dG(s)$ is stable, and

ii) $\delta_m(w)\alpha_m(w) < 1$, for all w.

Theorem 4 (Small phase theorem) The perturbed closed-loop system remains stable, if

i) $dG(s)$ is stable,

ii) $\{\theta_i(w) + \epsilon_j(w) : i,j = 1,2,\ldots,m\}$ have a spread of less than π, for all w,

iii) $[c_1(w)-1]c_2(w) < 1$, for all w,

iv) $\epsilon_1(w)+\theta_1(w)-\Omega_m(w) > -\pi$, for all w, and

v) $\epsilon_m(w)+\theta_m(w)+\Omega_m(w) < \pi$, for all w.

Corollary 1 By symmetry theorem 4 can be restated with c1(w) and c2(w) interchanged.

Corollary 2 As a consequence of theorems 3 and 4, and the perturbed closed-loop system remains stable, if, for some frequency w_b,

i) the conditions of theorem 4 are satisfied in the frequency range $[0,w_b]$, and

ii) the conditions of theorem 3 are satisfied in the frequency range $[w_b,\infty]$.

3.7 Comments on Principal Gains and Principal Phases

A perturbation dG(s) representing small parameter variations and high frequency dynamics that have not been modeled will typically have the following characteristics according to theorems 3 and 4 :

Low frequencies	-	A condition number of approximately 1. Small principal gains.
High frequencies	-	The maximum principal gain will increase. The minimum principal phase will exceed 180 degrees lag.

Consequently, when predicting perturbations for which the closed-loop system remains stable, a combination of theorems 3 and 4, as in corollary 2, is useful.

The small gain theorem (theorem 3) indicates that, at frequencies for which the maximum principal gain of the perturbation is large, the maximum principal gain of R(jw) should be designed to be small. This will normally be the case at high frequencies.

While at low frequencies, the small phase theorem (theorem 4) indicates that a perturbation with large gains can be tolerated providing R(jw) is designed to have a condition number close to 1 and a small spread of principal phases. This situation will normally be feasible at low frequencies, but at high frequencies when the phase lag inevitably exceeds 180 degrees, the maximum principal gain of R(jw) has necessarily to be made small.

Thus, it can be seen from the preceding remarks that useful information concerning the robustness of the closed-loop stability property can be extracted from Bode plots of the principal gains and phases on the input and output stability matrices.

Analysis of the multivariable case as opposed to the scalar case, moves from studying a single Bode magnitude and phase plot to a band of Bode plots (for each point of uncertainty) defined by the maximum and minimum principal gains and phases of the appropriate stability matrix. Also, as an indicator of the robustness of the stability property, gain and phase margins can be defined in terms of the maximum principal gain and minimum principal phase (maximum phase lag) by considering these values to be standard Bode magnitude and phase plots for a scalar system.

Chapter 4

Characteristic Loci Computer Aided Design System

This chapter deals with the design and implementation of a computer aided design (CAD) system to design controllers for multivariable systems using the Characteristic Loci technique. Although similar CAD systems are already in existence, there are several advantages for developing a CAD system on the university premises :-

i) Existing CAD packages are expensive and often specific hardware is required to implement the package.

ii) The source code for the CAD system would be available to developers if the package was developed on the premises. This is often not the case with bought CAD systems and would thus be difficult, if not impossible, and costly to develop or modify the CAD package.

iii) The Electrical Engineering Department at the university is at present building up a suite of CAD systems for personal computer systems. The common denominator between the various CAD systems being the database format. It would be difficult to buy a CAD system that could satisfy this requirement.

iv) There is not an abundance of information and experience concerning the Characteristic Loci technique at the university. Thus, development of a CAD system which implements the design technique would reveal the merits and problems associated with the design technique to the developers of the CAD system.

This chapter describes the main capabilities of the Characteristic Loci CAD (CL-CAD) package. This is followed by a brief explanation of the operation of the CL-CAD system in the form of a design example. The operational syntax for the CAD system is not mentioned in this chapter ; this information is contained in appendices H - K. Nor is this chapter a users' manual concerning the operation of the CL-CAD system ; the CL-CAD system does have an on-line help facility in order to assist the uninitiated user.

4.1 Characteristic Loci Design Procedure

In order to ensure a desirable closed-loop behavior of the multivariable system being investigated, the Characteristic Locus method prescribes a systematic procedure for the manipulation of the characteristic loci of the system. The characteristic loci being the frequency response plots of the eigenvalues of the open-loop transfer function matrix operator, G(s), of the multivariable system. This procedure aims at the construction of controllers, K(s), which commute with G(s) and thus yield a compensated transfer function matrix Q(s) = G(s)K(s), whose eigenvalues are the product of the eigenvalues of G(s) and K(s).

Thus, the CL-CAD system has been designed around the procedure described in the flowchart shown below in figure 4.1.

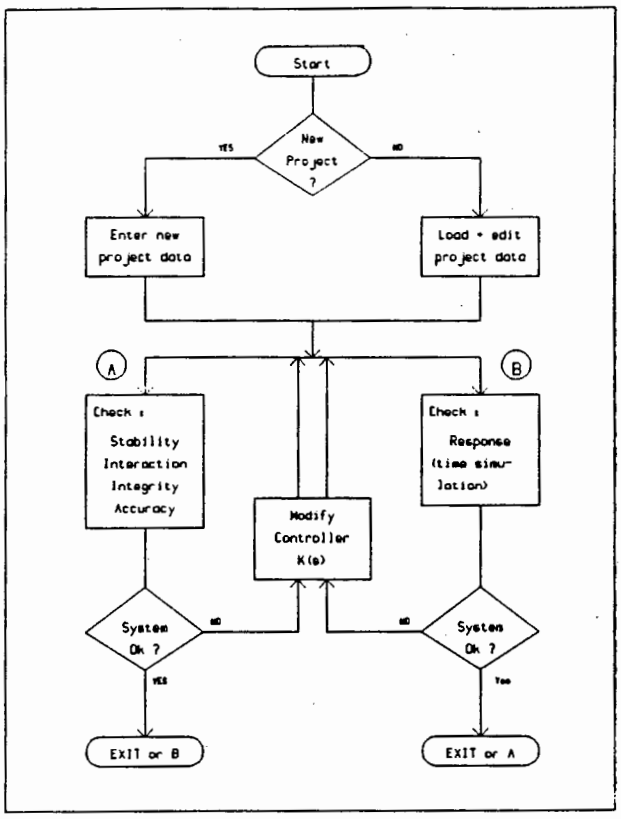


Figure 4.1 Flowchart Representing the Design Session using the CL-CAD System

4.2 Characteristic Loci CAD System Capabilities

4.2.1 System Representation

Since the Characteristic Loci technique is a frequency domain technique, the CAD package requires the system being investigated in the form of a matrix of polynomials in the s-domain. The CAD package allows square single to tenth order systems (ie. 10 inputs, 10 outputs), while each element or polynomial has a maximum order of ten for the numerator and denominator. A delay term (in seconds) is also included in each element or polynomial.

The CAD package maintains three polynomial matrices at any one time : System matrix : G(s), Controller matrix : K(s) and a temporary controller matrix : TEMP(s).

The CAD package has the ability to perform the following operations on all the matrices mentioned above :-

i) Save - The polynomial matrix information can be stored to a long term storage device (ie. to a file on disk).

ii) Load - The polynomial matrix information can be loaded from a long term storage device (ie. from a file on disk).

iii) Edit - Any coefficient of any element of the matrix can be edited or modified by the user at any stage of the design procedure.

iv) Initialise - The matrix of polynomials can be initialised to a zero or identity matrix.

v) Print - The matrix of polynomials can be printed out (if a printer is attached to the computer) at any stage of the design procedure.

4.2.2 System Characteristics

4.2.2.1 Characteristic Loci

The CL-CAD system enables designer to obtain the characteristic loci, bode plots and plots of the misalignment angles of the open loop system (G(s) or Q(s)) over any desired frequency range (limited by the accuracy of the computer being used).

The characteristic loci or eigenvalues of the system are obtained by evaluating the system at a particular frequency point and then using a QR [22] algorithm to calculate the eigenvalues and eigenvectors. The Bode-magnitude and misalignment angles are calculated using these values and are then presented to the user in the form of plots. This procedure is repeated for every frequency point over the range and increment specified by the user.

Two main problems are encountered when plotting the system characteristics :-

i) *Order of Eigenvalues :* The QR algorithm to calculate the eigenvalues and eigenvectors of a complex matrix isolates the largest eigenvalue first and the smallest last. Thus, the plot of the eigenvalues may not be continuous over the frequency range. Therefore, a tracking and sorting algorithm has to be implemented for each set of eigenvalues produced.

ii) *Misalignment angles :* These angles are calculated using the eigenvectors of the system and the basis set of vectors, but there is no indication in the literature on how to select the eigenvector and basis vector combination. Thus, the CL-CAD system calculates the angle between each eigenvector and all the basis vectors.

4.2.2.2 Principal Loci

The CL-CAD system does not include the facilities to calculate and display the principal gains and phases for the system under investigation. This capability was excluded as this CAD system was designed as an introduction to the Characteristic Loci technique. Thus, all the features of the CL-CAD system were kept as simple as possible.

4.2.3 Designing Multivariable Controllers

The CL-CAD system enables the designer to manipulate controller matrices but only performs basic controller design for the designer ie. the CAD system does not include any "intelligent" design features.

The CAD system enables the designer to perform the following operations on the controller matrices :-

i) Scalar matrix generation : This facility enables the designer to generate constant matrices in order to scale rows or columns.

ii) PI Matrix generation : The CAD system generates the basic PI matrix for the designer as described in chapter 3. The scaling of the individual columns within the PI matrix must be undertaken by the designer.

iii) Matrix editing : Any coefficient of any element of the matrix can be edited or modified by the user at any stage of the design procedure.

iv) Symbolic multiplication : The final system controller, K(s), has many criteria to satisfy simultaneously. This suggests that K(s) be designed as the cascaded combination of several sub-controllers, $K_i(s)$, so that

$$K(s) = \prod_{i=1}^{p} K_i(s)$$

where each $K_i(s)$ has a specific task to handle during the sequential synthesis.

Thus, the CL-CAD system enables the designer to generate a sub-controller in the polynomial matrix, TEMP(s), and then "add" the sub-controller to the system controller matrix, K(s), by using the symbolic matrix multiply facility provided in the CAD system.

v) System values : The designer can calculate the actual eigenvalues, eigenvectors and the eigenvector inverse at any frequency. These values can be output to a printer if so desired.

4.2.4 Time Simulation

The CL-CAD system provides the designer with a system time simulation facility. The CAD system uses a 4th order Runga-Kutta algorithm to perform the time simulation ; thus, the designers' choice of time step for the time simulation is critical when considering the accuracy of the simulation.

The CAD system has the facility that enables the designer to vary the system configuration for the time simulation, ie. The system can be either open-loop or closed-loop, the controller can be included or excluded in the loop. The designer is also able to step any system input at any point within the simulation.

4.2.5 Project Information

The CAD package parameters for each system the designer investigates will be different. These parameters include items such as : matrix filenames, system names, frequency sweep parameters, plot formats, axes scales, time simulation formats, etc. And since the design of a control system will require many design sessions, the CAD package enables the designer to save

the project information to disk file. So, whenever the designer starts a design session, the project information can be loaded in order to "customise" the CAD package to the system under investigation.

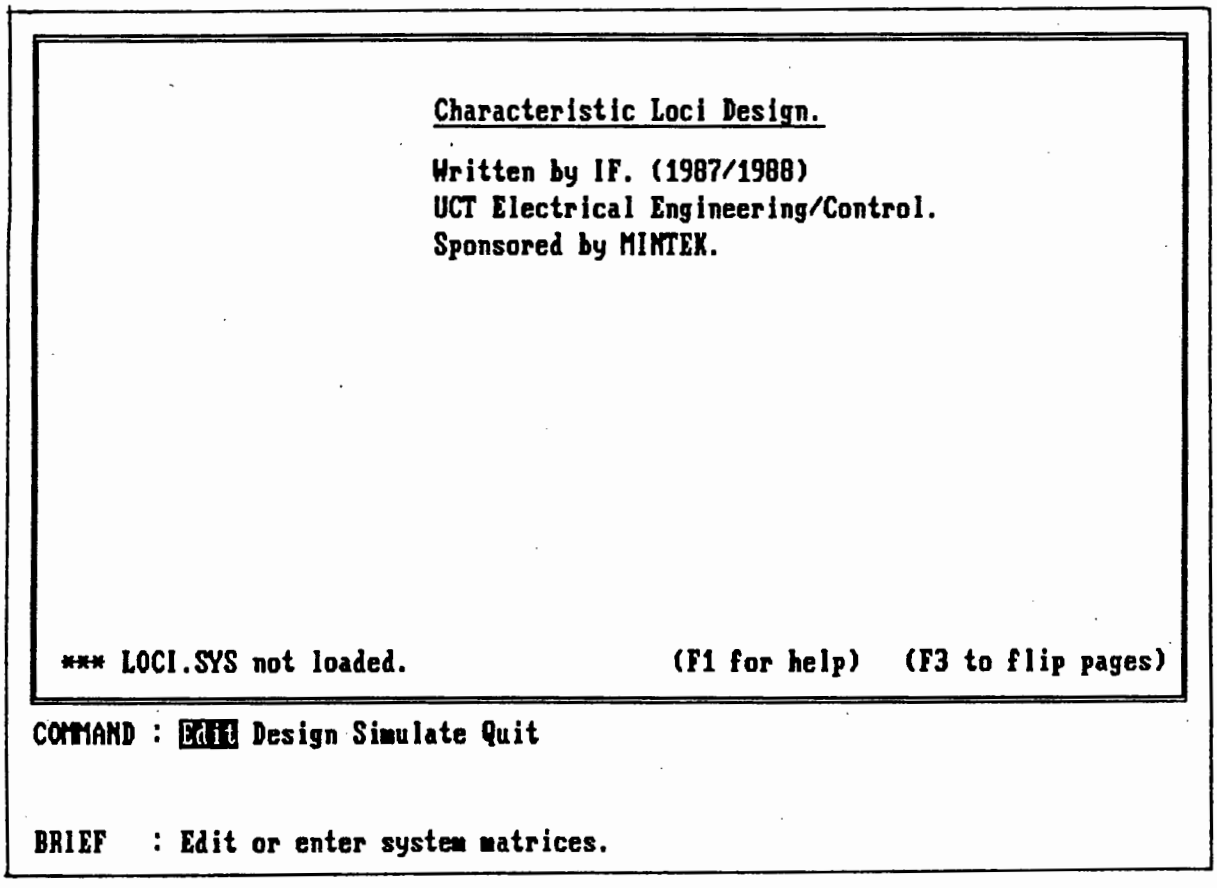
4.3 Appendix Listings for the CL-CAD System

The appendices listed below, contain all the specific information concerning the CL-CAD system :-

Appendix H - CAD system subroutine list.
Appendix I - CAD system source code listings.
Appendix J - CAD system disk file formats.
Appendix K - Development/execution information.

4.4 Design Example using the CL-CAD System

This section will present a brief overview of the operation of the CL-CAD system. The presentation will be in the form of description of the intended procedure that a control engineer would follow when investigating a multivariable system. The descriptions will be accompanied by print-outs of the CL-CAD system display at various stages of the design procedure. The display print-outs are as the user would see them, for example the startup display :



Display 4.1 Startup Display for the CL-CAD System

The following sections combine a worked example with the basic operation of the CL-CAD system. To assist the reader, comments concerning the worked example are in italics and comments regarding the CL-CAD system are in a normal typeface.

4.4.1 Description of Multivariable Example

The system considered here for the example is a pressurised flow-box [9], an important part of a paper making process. The state space representation is given below

$$\frac{d}{dt}\begin{bmatrix} H(t) \\ h(t) \end{bmatrix} = \begin{bmatrix} -0.03950 & 0.01145 \\ -0.01100 & 0.00000 \end{bmatrix}\begin{bmatrix} H(t) \\ h(t) \end{bmatrix} + \begin{bmatrix} 0.03362 & 1.03800 \\ 0.000966 & 0.00000 \end{bmatrix}\begin{bmatrix} u_1(t) \\ u_2(t) \end{bmatrix} \qquad (4\text{-}1)$$

$$\begin{bmatrix} y_1(t) \\ y_2(t) \end{bmatrix} = \begin{bmatrix} H(t) \\ h(t) \end{bmatrix} \qquad (4\text{-}2)$$

The system outputs are : *Flow box liquid level* - $h(t)$
Total head of stock - $H(t)$
The system inputs are : *Stock inflow* - $u_1(t)$
Air inflow - $u_2(t)$

The open loop characteristic polynomial is given by

$$\begin{aligned} OLCP &= \det(sI - A) \\ &= (s + 0.3949)(s + (0.032 \times 10^{-3})) \end{aligned} \qquad (4\text{-}3)$$

so that the number of right half plane zeros is zero and closed loop stability follows if and only if the net sum of clockwise encirclements of the (-1,0) point by the characteristic loci is zero. Or, using the notation used in chapter 3,

$$p_0 = 0 \qquad (4\text{-}4)$$

Thus closed loop stability follows iff

$$\sum_{i=1}^{2} n_{ti} = 0 \qquad (4\text{-}5)$$

The transfer function matrix is given by

$$G(s) = (sI - A)^{-1}B$$

$$= \begin{bmatrix} \dfrac{0.03362}{(s + 0.3949)} & \dfrac{1.03s}{\propto(s)} \\ \dfrac{((9.66 \times 10^{-4})s + (1.17 \times 10^{-5})}{\propto(s)} & \dfrac{-0.01141}{\propto(s)} \end{bmatrix} \qquad (4\text{-}6)$$

where $\propto(s) = s^2 + (0.395)s + (1.26 \times 10^{-4})$ (4-7)

4.4.2 Entering System Data into the CL-CAD System

After starting the CL-CAD system, the control engineer would then start by entering the system information which would consist of the following :-

i) Order of system (1X1 to 10X10 system).
ii) Input and output names.
iii) Project title and design group name.
iv) Standard filenames for plant and controller matrices.
v) Load and save pathnames for file operations.
vi) The plant transfer function matrix.

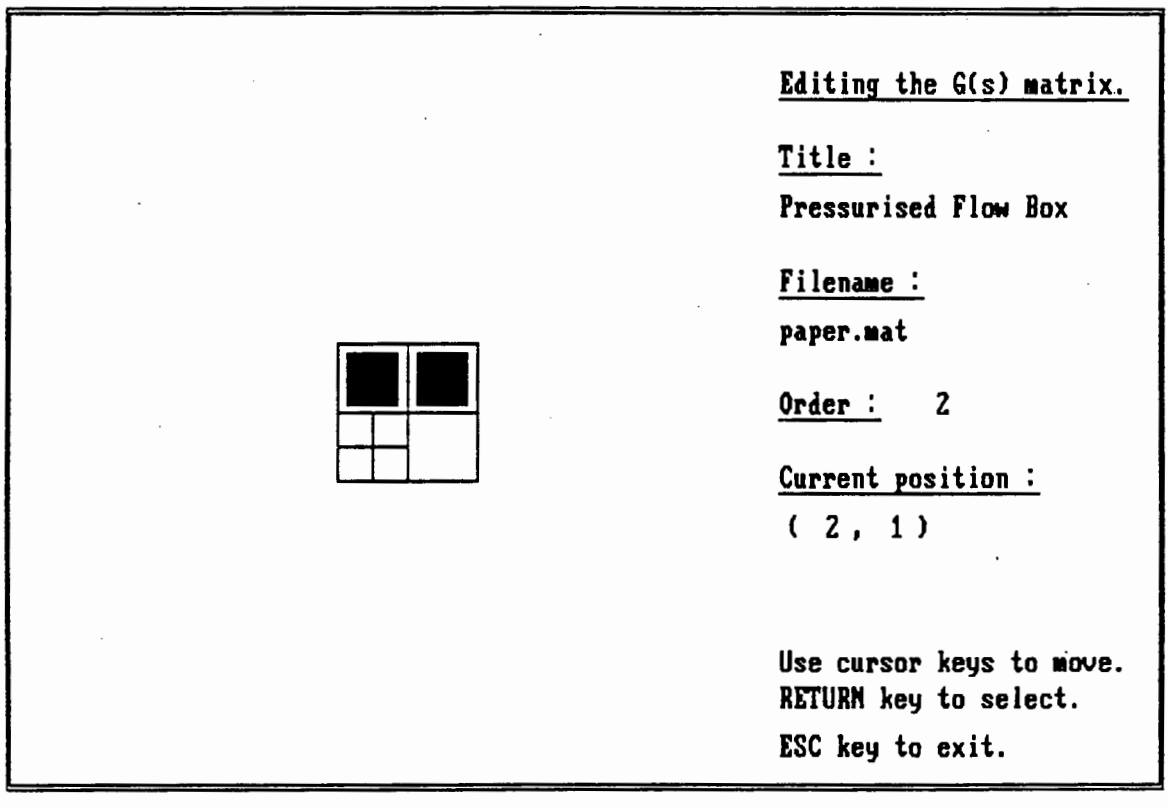
Entering items such as input and output names, project title, etc. are not crucial to the design procedure or operation of the CL-CAD system, but these items can help the user keep the whole system in perspective while working on the CAD system.

The most important data item to be entered into the CL-CAD system is the plant transfer function matrix. The CAD system presents the user with a grid that represents the plant matrix (see display 4.2 overleaf), each block represents an element of the matrix. If an element is a non-zero element, then that grid

block is "coloured-in". This enables the user to see the overall structure of the plant matrix at any time.

The user can edit any element in the plant matrix by moving a "select" cursor to that element on the grid and hitting a key. The CAD system then sets-up that matrix element for editing (editing a matrix element is described at a later stage).

The display below describes the grid structure, element (1,1) and (1,2) have been edited (or non-zero) and the "select" cursor is on element (2,1) which is zero at the moment.

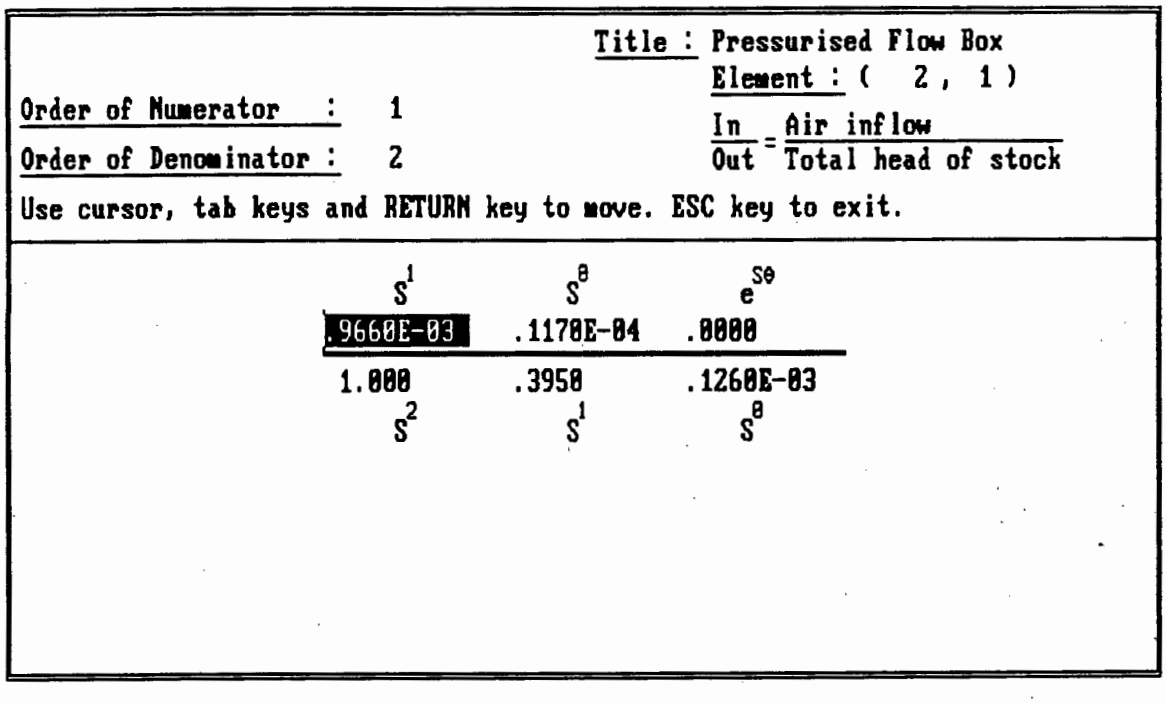


Display 4.2 Editing System Matrix - Matrix Structure

The display overleaf shows the user editing matrix element (2,1). The numerator and denominator can be in the range of zero to 10th order polynomials in the s-domain. The transfer

function poles and zeros are assumed to be in radians per second and the dead time is assumed to be in seconds.

Note, in display 4.3, that the user has customised the CAD system by entering project titles, input and output names, etc.



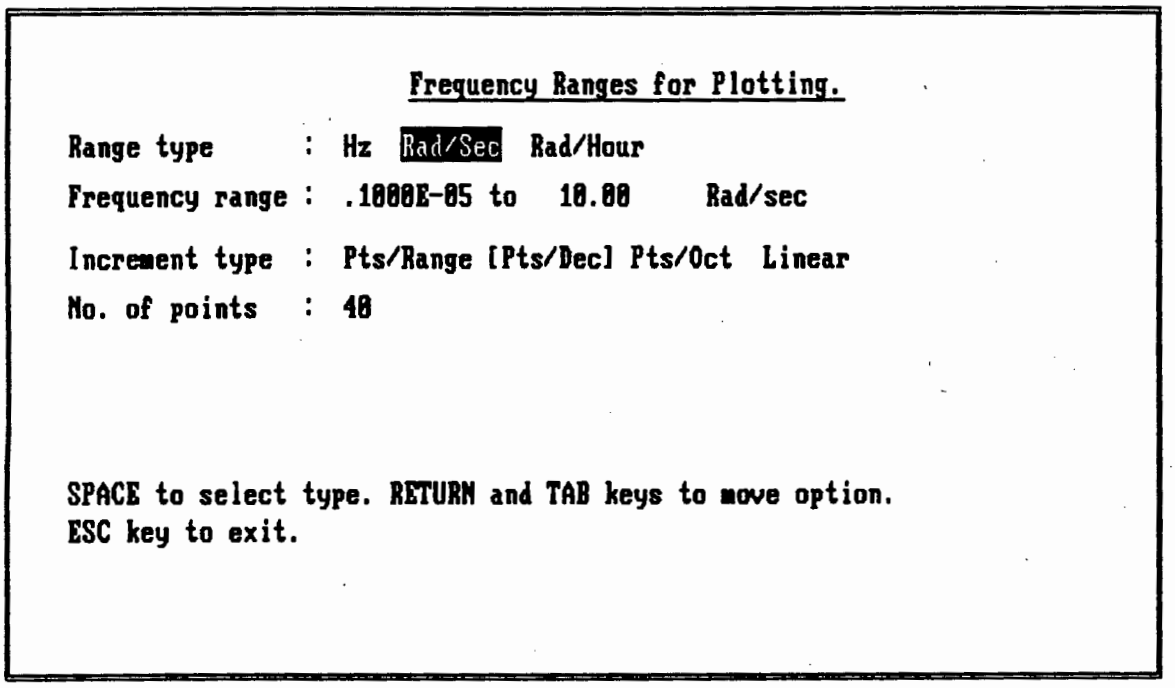
Display 4.3 Editing Polynomial/element $g_{21}(s)$

4.4.3 CAD Display Parameters

Once the user has entered the system data, the CAD system is almost in a position to calculate and present the plant characteristics. But, first the user must inform the CAD system of certain parameters, such as frequency range, axes scales and plotting formats.

The display overleaf describes the frequency sweep parameters edit page. The user can specify any combination of frequency

units, range and increment type. All numbers entered are checked for errors, for example : entering a negative number for the points per decade parameter.

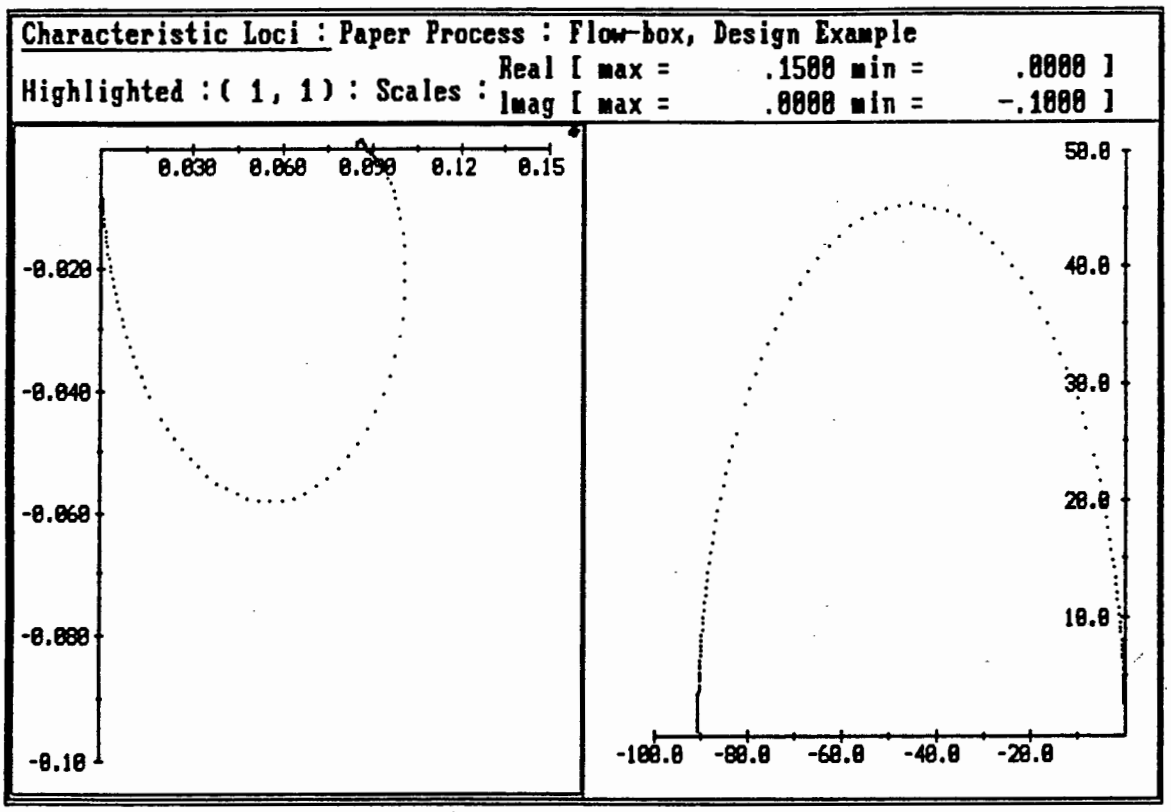


Display 4.4 Frequency Sweep Parameter Editing

Similar editing facilities exist in order to edit the axes scales and plot page formats. As with all the editing facilities available in the CAD system, as many error checks as possible are performed in order to prevent nonsensical results.

4.4.4 Displaying System Characteristics

The CAD system is now able to present the system characteristics to the user. The display shown overleaf is the characteristic loci of the system described in section 4.4.1 above over the frequency range (1.0×10^{-4} to 10.0) rad/sec.



Display 4.5 Characteristic Loci of G(s)

The characteristic loci of G(s) (display 4.5) reveal that for equal gains in each loop (ie. $k_1=k_2=k$), the stability condition (4-5) is satisfied providing

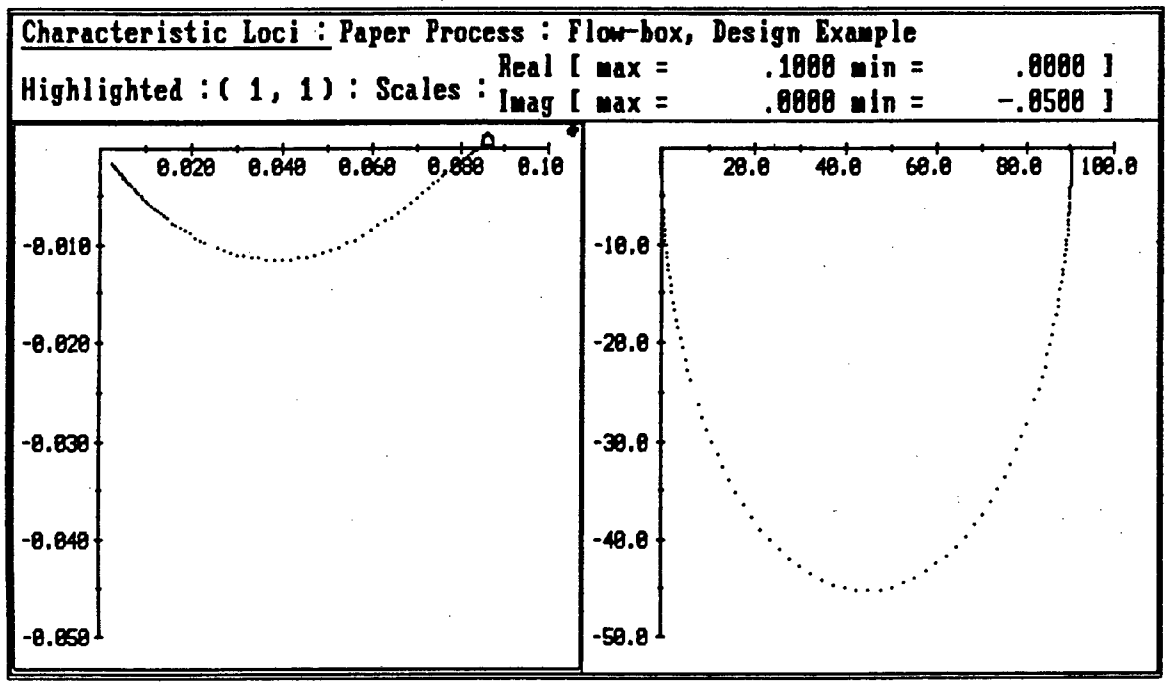
$$k < (1/90.6) = 0.0111 \qquad (4\text{-}8)$$

These stability margins are extremely small. To remedy this situation, a elementary transformation matrix controller is used to change the sign of the feedback round $g_{22}(s)$) without affecting $g_{11}(s)$. The controller matrix :

$$K_1 = \begin{bmatrix} 1 & 0 \\ 0 & -1 \end{bmatrix} \qquad (4\text{-}9)$$

The controller matrix, K(s), in the CL-CAD system is presented to the user in exactly the same format as the plant matrix, G(s). All operations that can be performed on G(s), such as editing, initialising, etc., can be performed on K(s). Thus, the user can enter the K_1 matrix into the CL-CAD system in exactly the same way as the plant matrix was entered.

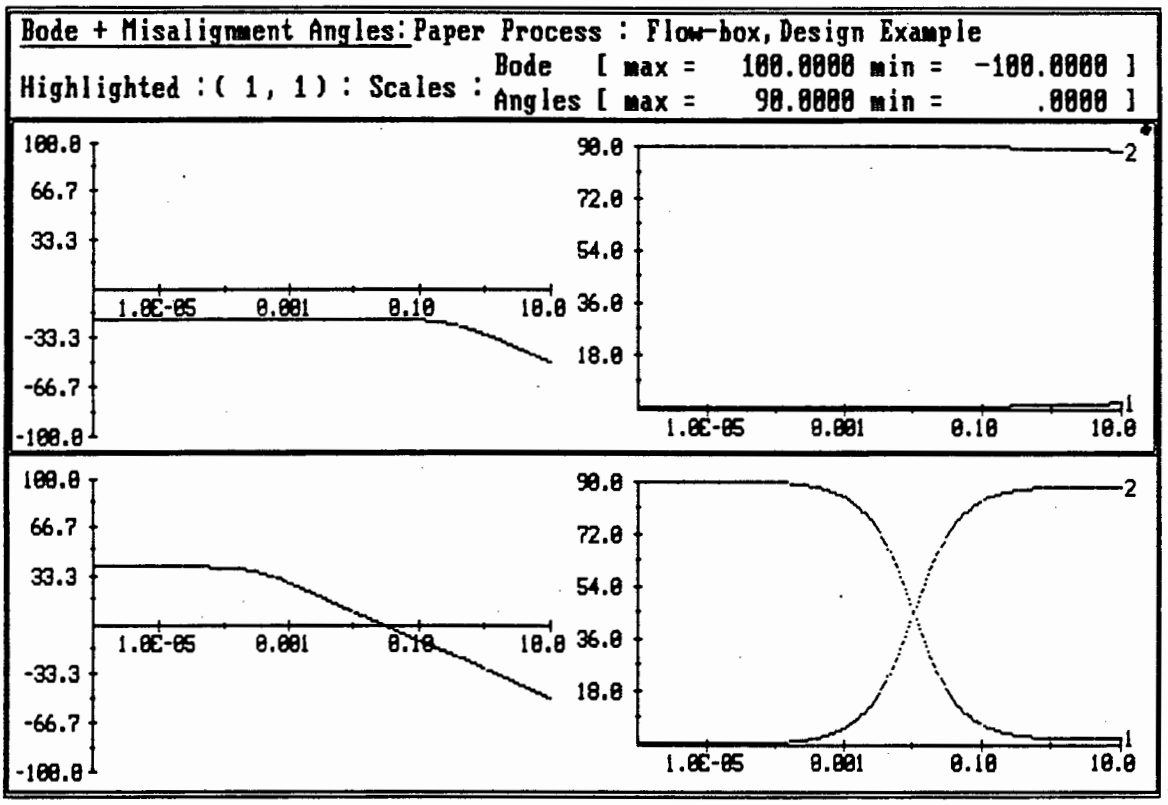
The characteristic loci of $Q_1(s)$, where $Q_1(s) = G(s)K_1$, are shown in display 4.6 below. The loci show that K_1 has improved the overall stability of the system for all combinations of loop gain between zero and arbitrarily high values.



Display 4.6 Characteristic Loci of $Q_1(s)$

The users' attention will now turn to interaction after satisfying the stability criteria. Display 4.7 overleaf shows the Bode magnitude and misalignment angle plots of $Q_1(s)$. Inspection of the plots reveals that interaction problems exist

at low and high frequencies (small magnitude of one of the loci at low frequencies and large set of misalignment angles at high frequencies). A matrix PI controller could be the next controller factor which could solve the interaction problem described above.



Display 4.7 Bode Magnitude and Misalignment Angle Plots of $Q_1(s)$

The numbers following the misalignment angle plots inform the user of which basis vector (see chapter 3) was used to generate that particular misalignment angle plot.

The CL-CAD system can generate the PI matrix mentioned above, but the scaling of the columns of the PI matrix must be undertaken by the user. The CL-CAD system will synthesise the PI matrix and place the result in a temporary matrix for editing by the user. Once the user has edited the temporary

matrix, the user can post multiply (symbolically) the controller matrix with the temporary matrix, in other words $K(s)=K_1*K_2(s)$, where $K_2(s)$ is the PI matrix residing in the temporary matrix.

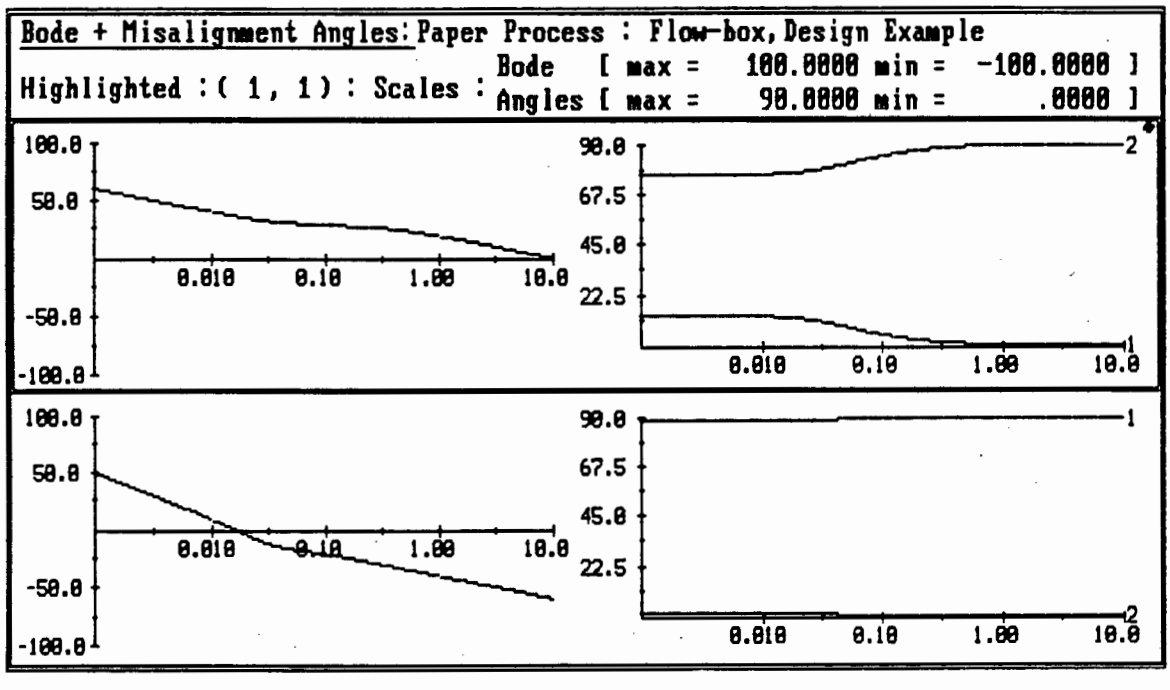
The PI matrix, $K_2(s)$, for this example

$$K_2(s) = \begin{bmatrix} \dfrac{11.72}{s} & 10.30 \\ -9.66 - \dfrac{0.012}{s} & 0.03362 + \dfrac{0.011}{s} \end{bmatrix} \qquad (4\text{-}10)$$

which gives

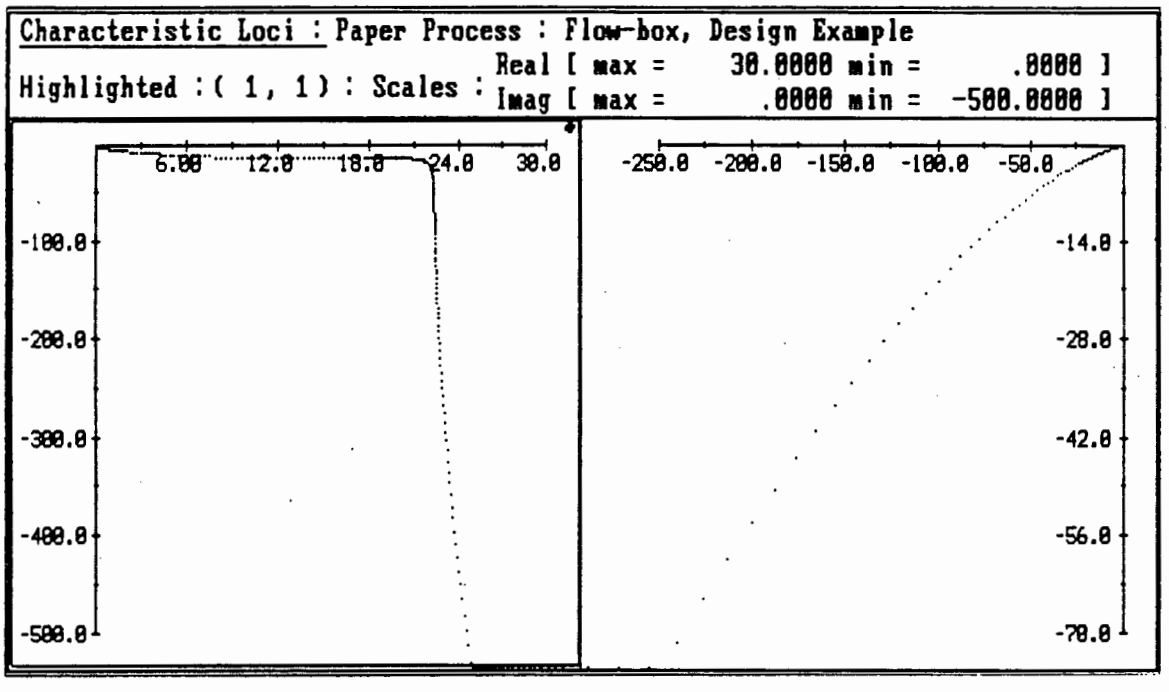
$$Q_2(s) = G(s)K_1K_2(s) = Q_1(s)K_2(s) \qquad (4\text{-}11)$$

Repeating the interaction analysis with $Q_2(s)$ produces the plots in display 4.8 where it can be seen that interaction at high and low frequencies has been improved.



Display 4.8 Bode Magnitude and Misalignment Angle Plots of $Q_2(s)$

Checking the characteristic loci of $Q_2(s)$, shown in display 4.9 below, reveals that system stability has not been impaired by the addition of $K_2(s)$.



Display 4.9 Characteristic Loci of $Q_2(s)$

The user would now adjust overall closed loop performance by tuning the design values of loop gain. This is done on the basis of the diagonal elements of $Q_2(s)$ and produces

$$K_3 = \begin{bmatrix} 10 & 0 \\ 0 & 100 \end{bmatrix} \qquad (4\text{-}12)$$

The user can generate this scalar matrix in the temporary matrix of the CL-CAD system and then post multiply (symbolically) the controller matrix with the temporary matrix as mentioned previously when dealing with the PI matrix.

The overall controller becomes

$$K(s) = K_1 K_2(s) K_3$$

$$K(s) = \begin{bmatrix} \dfrac{117.2}{s} & 1030 \\ -96.6 - \dfrac{0.12}{s} & 33.62 + \dfrac{1.10}{s} \end{bmatrix} \quad (4\text{-}13)$$

4.4.5 Time Simulation of System

The CL-CAD system has the facility to perform time simulations of the system. Similar to setting the display parameters for the system characteristics, the user must set the time simulation parameters. These include axes scales and system simulation format. The system simulation format page is shown in display 4.10 below. The user is currently editing the time step parameter.

Time Simulation Parameters.

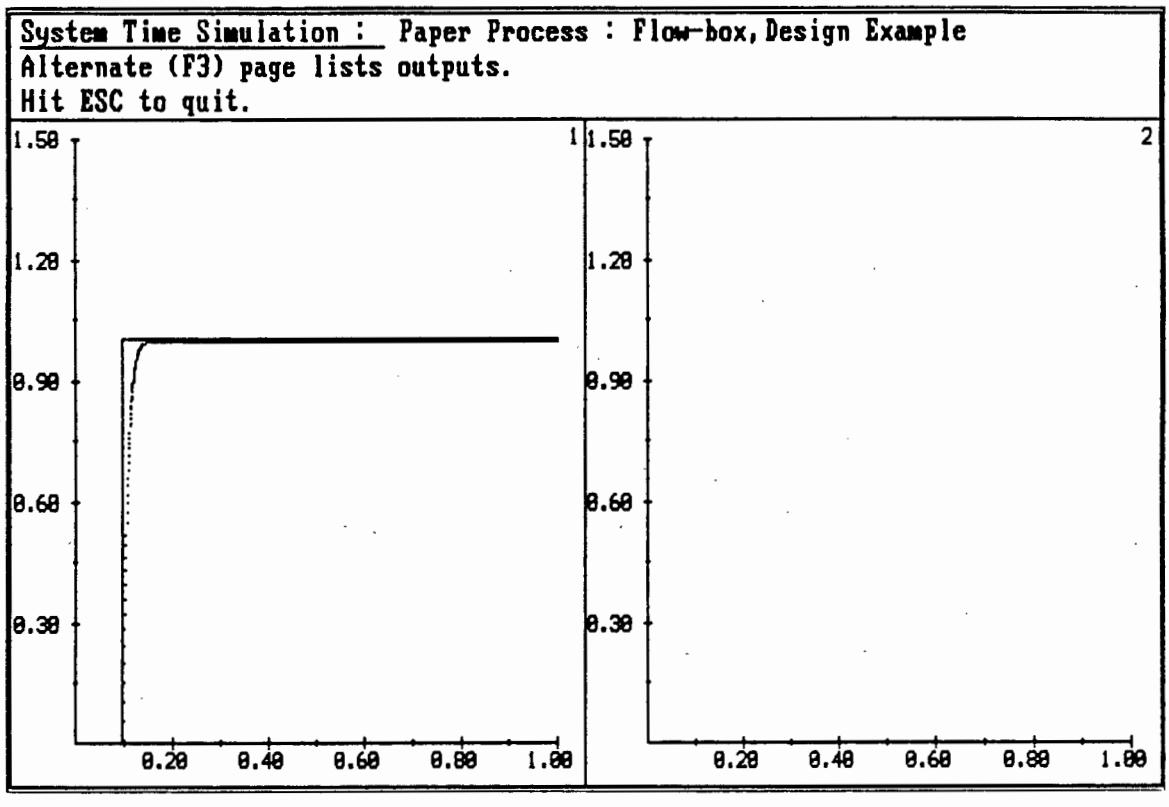
System type : Closed Open
Controller : [In] Out
Time duration [secs] : 5.000
Time step [secs] : .5000E-02

Step No.	Input No.	Increment	Time [secs] (after start)
1	1	1.000	.1000
2	2	1.000	2.500
3	0	.0000	.0000
4	0	.0000	.0000
5	0	.0000	.0000

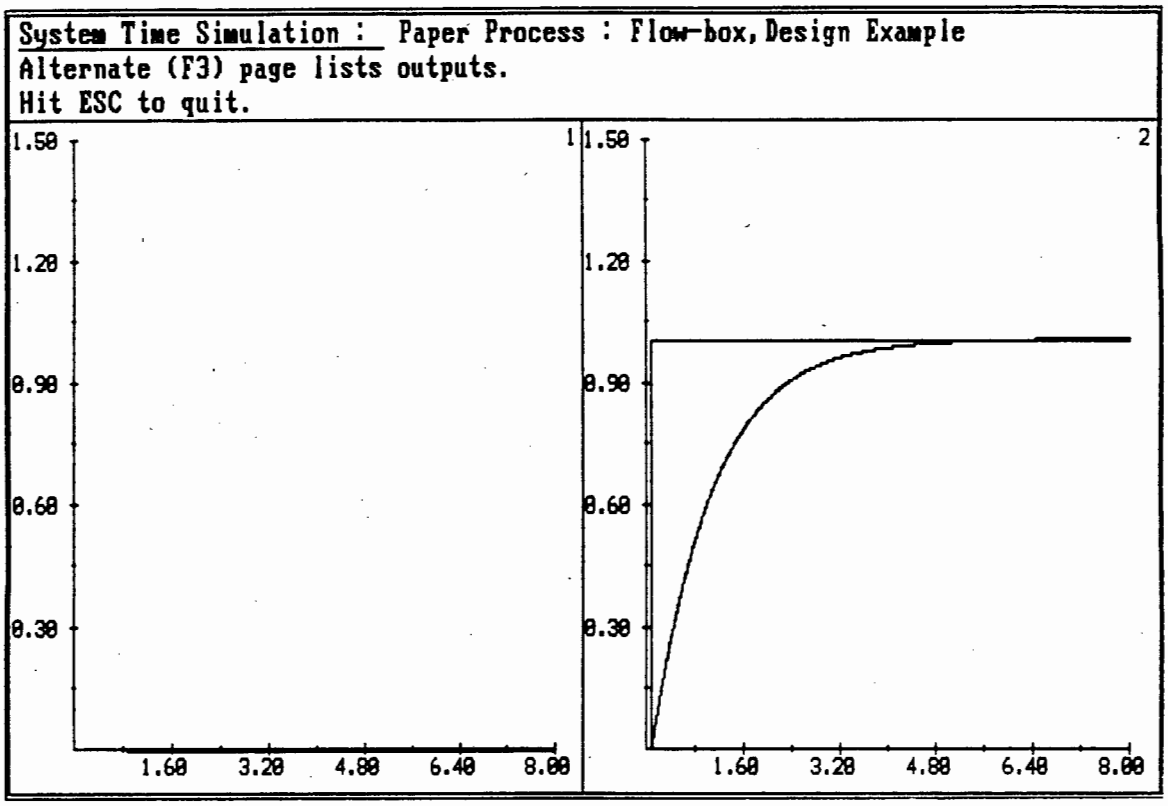
SPACE to select type. RETURN and TAB keys to move option.
ESC key to exit.

Display 4.10 Time Simulation Format Page

The transient responses for unit step changes in total head, H(t), and level, h(t), are shown in displays 4.11 below and 4.12 overleaf, where it can be seen that the closed loop responses are fast and that interaction is negligible.



Display 4.11 Transient Response of Compensated System - Setpoint H(t) Stepped



Display 4.12 Transient Response of Compensated System - Setpoint h(t) Stepped

4.4.6 Saving System Information

The CL-CAD system has the facility to save all information relevant to a particular project. The project information includes system names (input, output, titles, filenames, etc.), axes settings (loci , bode, misalignment angles and time simulation plots), frequency sweep parameters, plots formats, time simulation formats, etc. Saving this information prevents the user from re-entering the same project information whenever working on a particular project that is spread over several design sessions.

Chapter 5

Design of Control System for the Flotation Plant Simulator using the Characteristic Loci Technique

This chapter will describe the design procedure followed in order to synthesise multivariable controllers for the Flotation Plant Simulator using the Characteristic Loci (CL) technique. This frequency domain technique has been implemented in the form of CAD system on a personal computer as described in chapter 4.

The frequency domain technique requires a time invariant model (in the *s*-domain) of the Flotation Plant Simulator. The model was generated in chapter 2 (shown in appendix G) and all design procedures will be based on this model.

This chapter first describes the design procedure followed when applying the CL design technique to the Flotation Plant Simulator. The design procedure is followed by comments, made by the designer, concerning the technique. The second part of the chapter describes the implementation and testing of the control scheme.

5.1 Design of Control Scheme

Upon inspection of the plant matrix, G(s), it is apparent that none of the elements of G(s) have poles which lie in the left half plane. Thus the open loop characteristic polynomial of the system will have no roots in the left half plane, so that $p_o=0$ (using notation as in chapter 3) and closed loop stability follows if and only if

$$\sum_{i=1}^{4} N_{ti} = 0 \qquad (5\text{-}1)$$

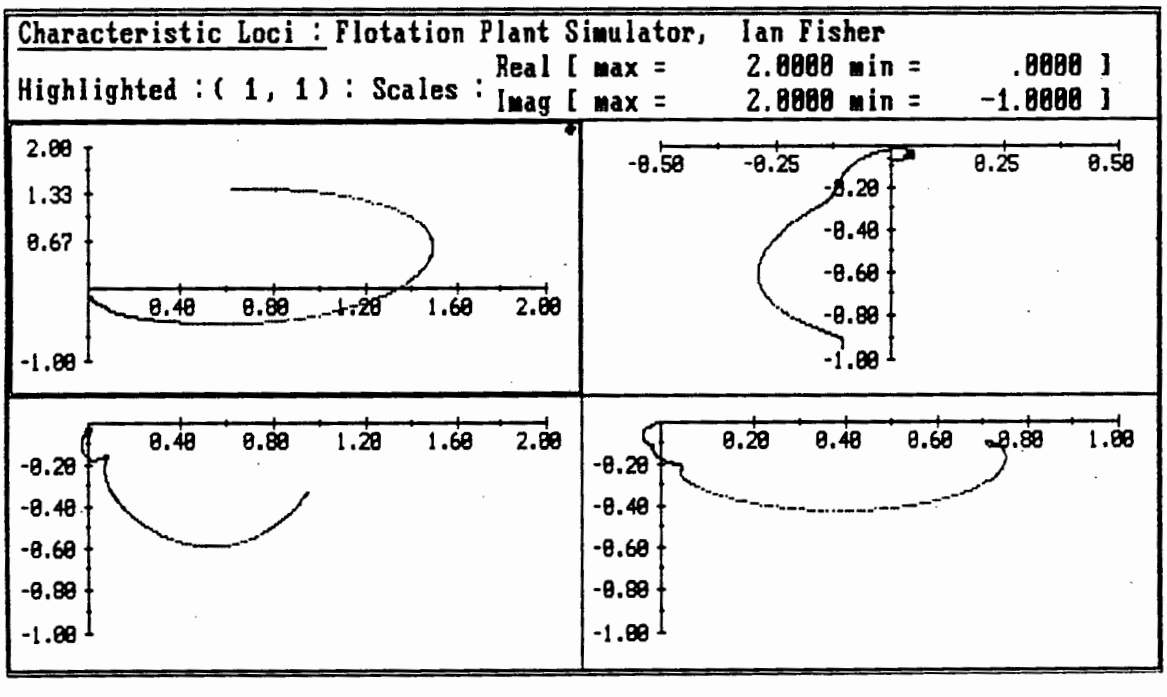


Figure 5.1 Characteristic loci of Flotation Plant Simulator or G(s) (Frequency : 0.005 -> 0.5 Rads/Sec).

The characteristic loci of the Flotation Plant Simulator or G(s), shown in figure 5.1, reveal that for equal gains (ie. $k_1=k_2=k_3=k_4=k$) in each loop, the stability condition (5-1) is satisfied providing

$$k \geq 0 \qquad (5\text{-}2)$$

The integrity of the of the system is checked by observing the characteristic loci under a set of stipulated failure conditions. If loop *j* fails, the characteristic loci of the principal sub-matrix obtained by deleting row *j* and column *j* must satisfy the stability criteria (this method has been described in chapter 3).

The characteristic loci of the sub-matrix formed by deleting row 1 and column 1 of the original system (if loop 1 failed) is shown below in figure 5.2.

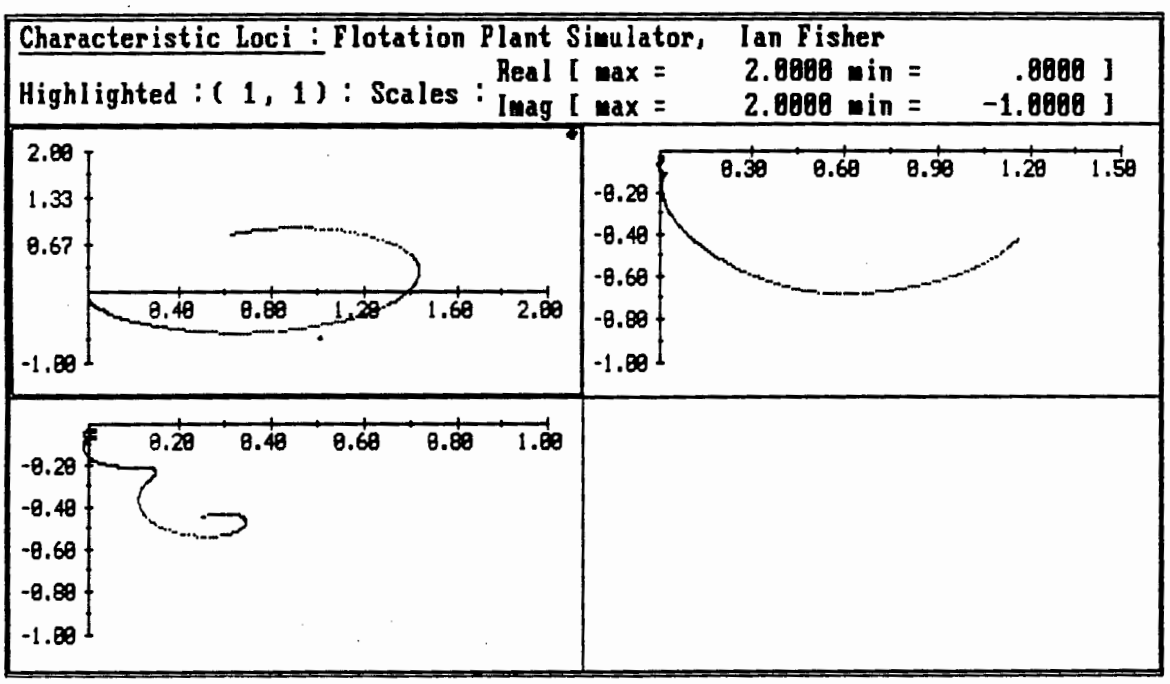


Figure 5.2 Checking integrity : Failure in loop 1
(Frequency : 0.005 -> 0.5 Rads/Sec)

The characteristic loci in figure 5.2 reveal that the system will remain stable under the following conditions

$$k_1 = 0 \quad k_2 \geq 0 \quad k_3 \geq 0 \quad k_4 \geq 0 \qquad (5\text{-}3)$$

The same technique can be applied to the remaining three loops and similar results for system stability under loop failure conditions are obtained as follows

$$k_2 = 0 \quad k_1 \geq 0 \quad k_3 \geq 0 \quad k_4 \geq 0 \qquad (5\text{-}4)$$
$$k_3 = 0 \quad k_1 \geq 0 \quad k_2 \geq 0 \quad k_4 \geq 0 \qquad (5\text{-}5)$$
$$k_4 = 0 \quad k_1 \geq 0 \quad k_2 \geq 0 \quad k_3 \geq 0 \qquad (5\text{-}6)$$

From these results, it can be safely assumed that stability of the system will be maintained if any loop fails as long as the remaining loops have positive gains.

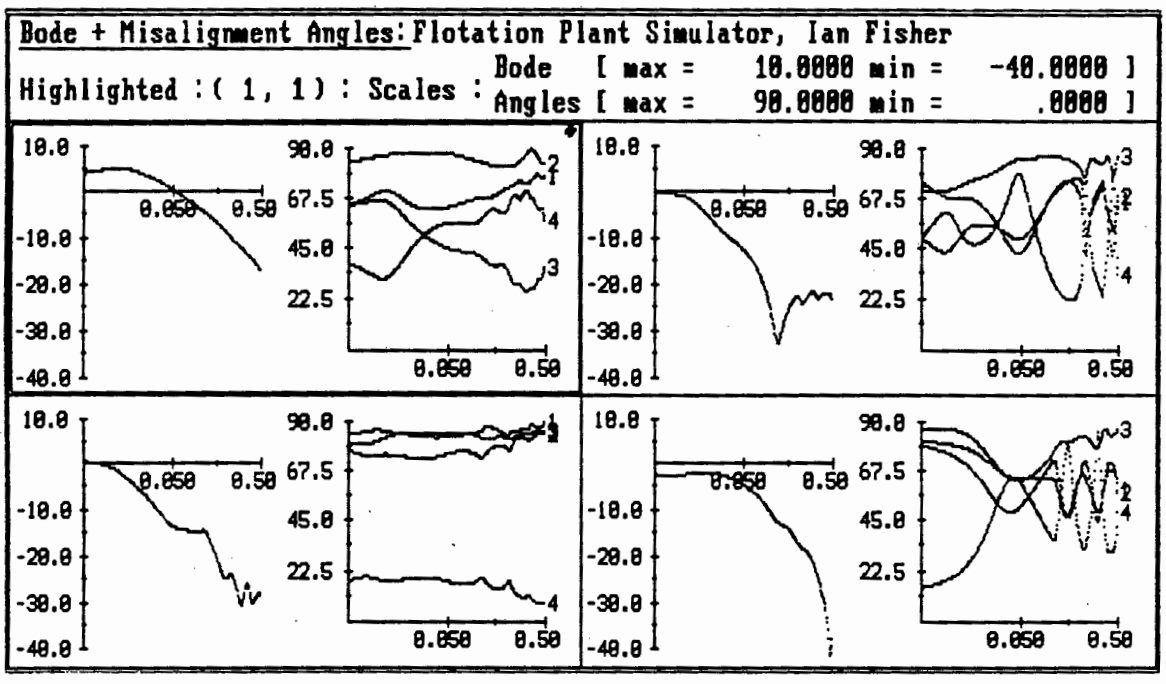


Figure 5.3 Bode magnitude and misalignment angles of G(s) (Frequency : 0.005 -> 0.5 Rads/Sec).

An inspection of the interaction analysis plots in figure 5.3, reveal the interaction problems associated with the plant. At low frequencies, the magnitudes (looking at the bode plots) of the loci are small which indicate interaction and the general performance characteristics are not expected to be good.

The misalignment angles do not provide any useful information except that the plant is interactive at high frequencies. The delays present in the system become significant at high frequencies and are seen in the form of loops on the loci, dips and crests on the bode magnitude plots and "oscillations" in the misalignment angle plots.

These "oscillations" in the misalignment angle plots are caused by the delay factors affecting the angle between the characteristic direction vectors and the standard basis vectors (delays, in the frequency domain, can be seen as a rotating vector of constant magnitude). This implies that if the elements containing the delays are not eliminated or reduced, then characteristic direction vectors will never be aligned with the standard basis vector set at high frequencies. And this implies that there will be coupling within the system at high frequencies.

In order to eliminate the interactive problem mentioned above, two "solutions" are suggested :

i) Reduce or compensate for the elements containing the delays such that the phase shifts due to delay factors do not affect the characteristic direction vectors.

ii) Ensure that $Q(s)=G(s)K(s)$ tends to zero, as frequency increases, before the phase shifts due to the delay

factors become significant in relation to the angle formed between the characteristic direction vectors and the standard basis vectors.

The first solution is more specific and appealing from an engineering point of view. But the implementation using the Characteristic Loci technique would be difficult as the designer is not given any clues as to where the most significant undesirable factors are within the system. And it thus becomes difficult to eliminate or compensate for those factors.

The second solution is within the capabilities of the design technique to ensure that the system is band-limited. This will reduce the effect of coupling within the system at high frequencies, but not eliminate the coupling.

In order to reduce interaction at low frequencies and remove steady state errors in the closed loop system, a PI matrix controller (as described in chapter 3) will now be cascaded with the plant. Recall that the PI matrix controller described in chapter 3

$$K(s) = K_\infty D_1 + K_0 D_2/s \qquad (5\text{-}7)$$

Selecting

$$D_1 = \begin{bmatrix} 0.01 & 0.00 & 0.00 & 0.00 \\ 0.00 & 0.10 & 0.00 & 0.00 \\ 0.00 & 0.00 & 0.01 & 0.00 \\ 0.00 & 0.00 & 0.00 & 0.01 \end{bmatrix} \qquad (5\text{-}8)$$

and

$$D_2 = \begin{bmatrix} 0.20 & 0.00 & 0.00 & 0.00 \\ 0.00 & 0.10 & 0.00 & 0.00 \\ 0.00 & 0.00 & 0.10 & 0.00 \\ 0.00 & 0.00 & 0.00 & 0.10 \end{bmatrix} \qquad (5\text{-}9)$$

The exception to the scaling matrices (5-8) and (5-9) is the diagonal elements of the original K_∞ and K_0 matrices. In other words, the diagonal elements of the K_∞ and K_0 matrices should remain unchanged and only the off-diagonal elements of K_∞ and K_0 should be multiplied by the factors in D_1 and D_2 respectively.

This results in the PI matrix controller

$$k_{11}(s) = \frac{(3.029\text{x}10^{1})s + (0.5621)}{s}$$

$$k_{12}(s) = \frac{(-2.627)s + (-0.1836)}{s}$$

$$k_{13}(s) = \frac{(1.943\text{x}10^{-2})s + (1.386\text{x}10^{-2})}{s}$$

$$k_{14}(s) = \frac{(2.851\text{x}10^{-3})s + (2.448\text{x}10^{-2})}{s}$$

$$k_{21}(s) = \frac{(0.2285)s + (7.518\text{x}10^{-3})}{s}$$

$$k_{22}(s) = \frac{(3.8297\text{x}10^{1})s + (0.1044)}{s}$$

$$k_{23}(s) = \frac{(1.732\times10^{-2})s + (6.037\times10^{-4})}{s}$$

$$k_{24}(s) = \frac{(2.483\times10^{-2})s + (1.191\times10^{-2})}{s}$$

$$k_{31}(s) = \frac{(-0.1071)s + (-0.1160)}{s}$$

$$k_{32}(s) = \frac{(-0.3316)s + (-1.230\times10^{-3})}{s}$$

$$k_{33}(s) = \frac{(1.243\times10^{1})s + (0.2285)}{s}$$

$$k_{34}(s) = \frac{(4.296\times10^{-2})s + (1.163\times10^{-2})}{s}$$

$$k_{41}(s) = \frac{(-4.859\times10^{-1})s + (5.736\times10^{-2})}{s}$$

$$k_{42}(s) = \frac{(-1.239)s + (2.955\times10^{-2})}{s}$$

$$k_{43}(s) = \frac{(-2.193\times10^{-1})s + (-3.471\times10^{-1})}{s}$$

$$k_{44}(s) = \frac{(5.556\times10^{1})s + (1.417)}{s}$$

The PI matrix controller cascaded with the plant gives

$$Q(s) = G(s)K_{pi}(s) \qquad (5\text{-}10)$$

Repeating the stability (figure 5.4 and figure 5.5) and interaction (figure 5.6) analysis :

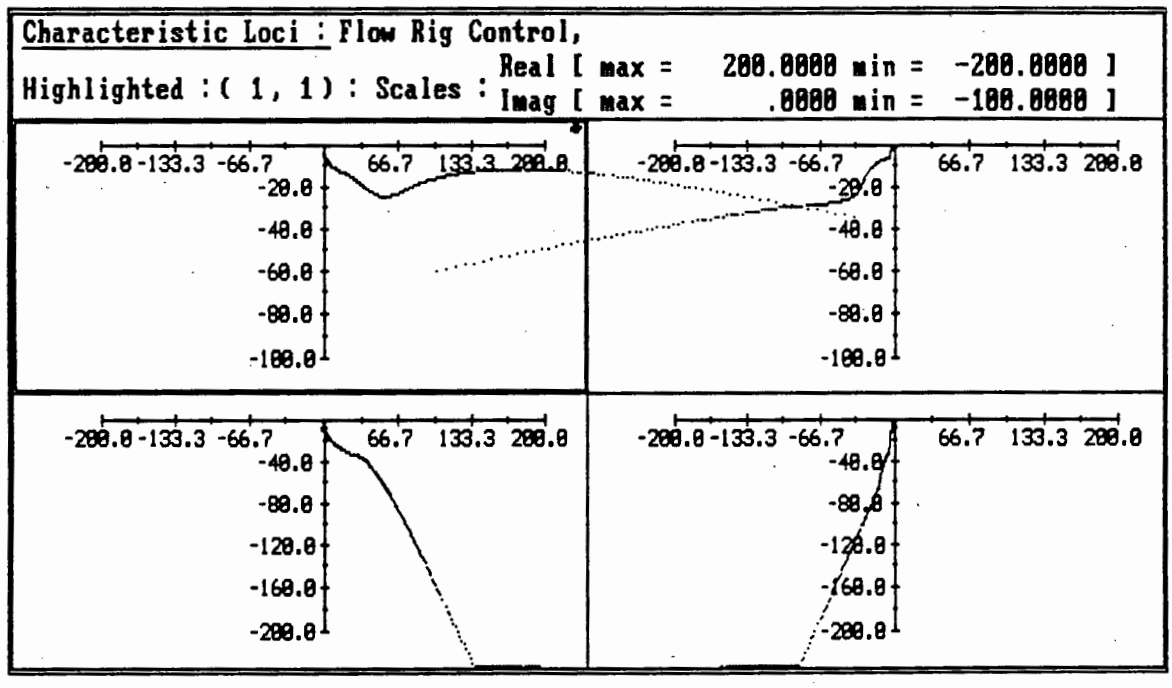


Figure 5.4 Characteristic loci of Q(s).
(Frequency : 0.001 -> 0.256 Rads/Sec)

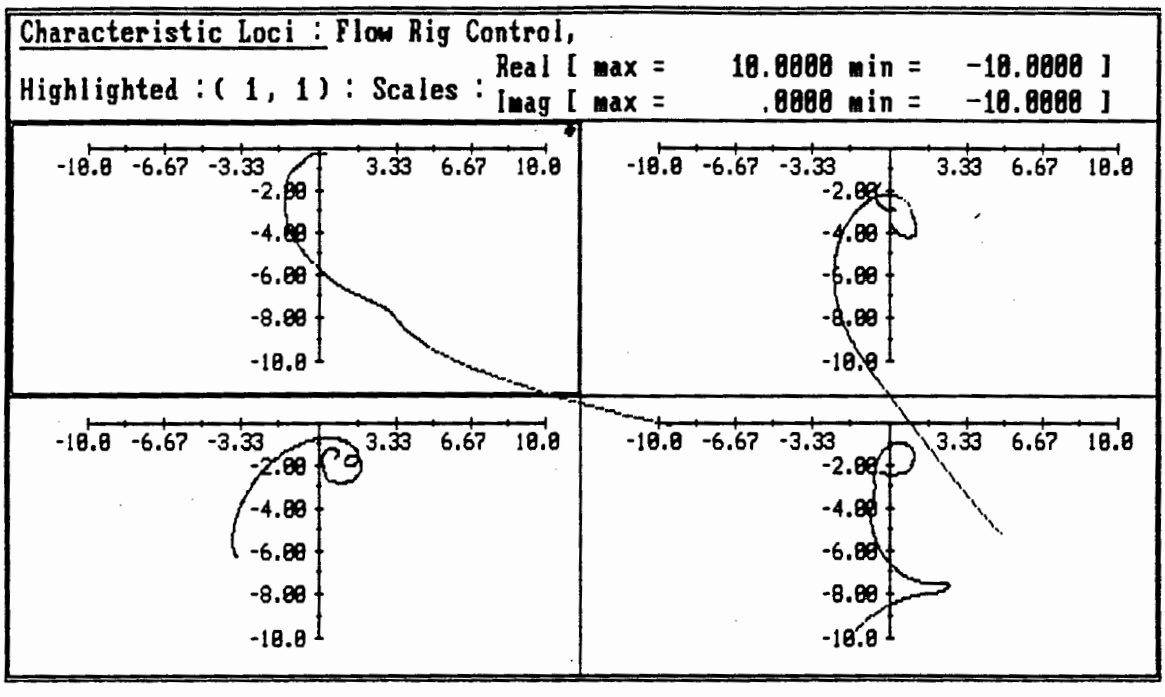


Figure 5.5 Characteristic loci of Q(s) - Origin detail.
(Frequency : 0.05 -> 0.5 Rads/Sec)

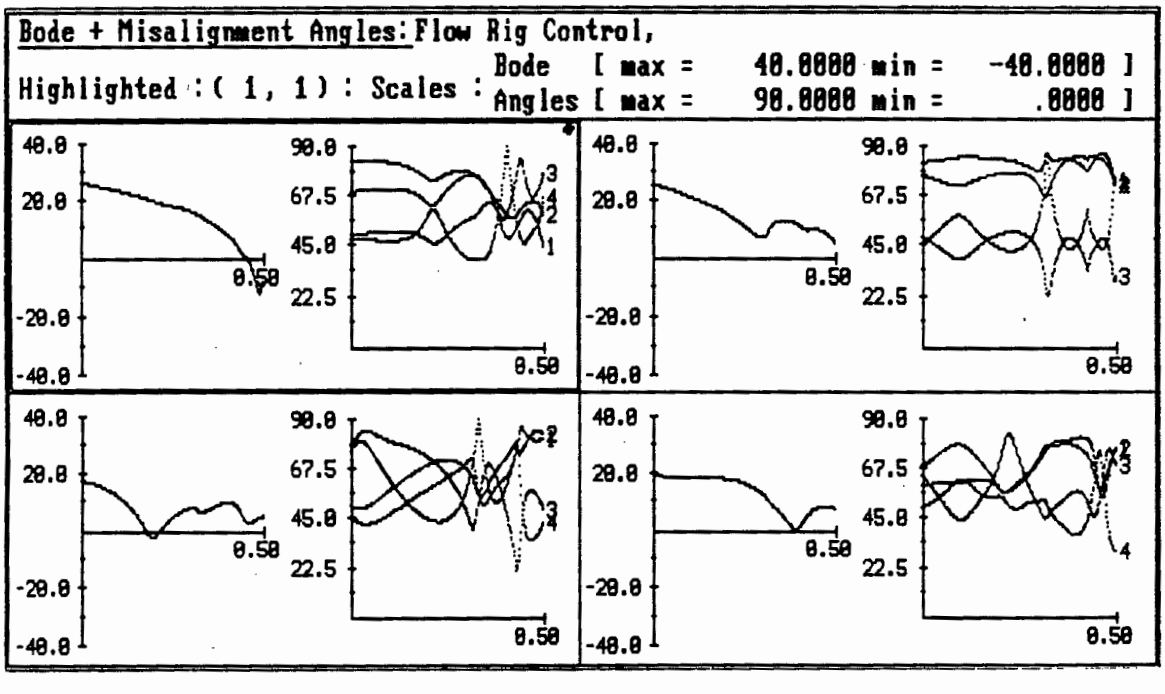


Figure 5.6 Bode magnitude and misalignment angles of Q(s).
(Frequency : 0.05 -> 0.5 Rads/Sec)

After inspecting figures 5.4 to 5.6, one can conclude the following :

i) The system still satisfies the stability criteria. Although the phase margin of one of the loci should be improved. This could be done using an off-diagonal phase advance controller (The effect of which is described in chapter 3). But, the Characteristic Loci technique does nor present a methodical technique for synthesising a controller in order to affect the phase of one particular locus and a 4X4 system presents too many permutations for a trial and error technique.

ii) The bode magnitude plots reveal high gain at low frequencies, which indicates low interaction at these frequencies.

iii) The misalignment angles still do not indicate any improvement in the alignment of the characteristic directions with the standard basis vectors. In other words, the delay factors are still affecting the characteristic directions at high frequencies.

The PI matrix attempts to diagonalise Q(s) and align the characteristic directions at high frequencies using a constant matrix. But, due to the fact that the delay factors are still significant at high frequencies, it is not possible to align the characteristic directions using a constant matrix.

iv) Q(s) does tend towards zero as frequency increases, which will reduce the effect of the high frequency coupling within the system.

The simulated transient responses for step changes to each setpoint are shown in the following figures where it is seen that interaction has not been completely eliminated from the system. But the response of the closed-loop system is faster when compared to the open-loop system presented in chapter 2.

The time simulations may not be completely realistic when compared to the original system in that the control values and the values represented by the transfer functions are not bounded as in a real plant situation. For example, it might not be possible to have a negative flow from one tank to another, whereas the transfer function model would permit such a condition. But, the time simulations will still provide the designer with some guidelines as to how the system is going to react to setpoint activity.

In the following plots, the graphs are numbered (in top-right corner) according to system input number : 1 = Rougher, 2 = Scavenger, 3 = Cleaner, 4 = Recleaner.

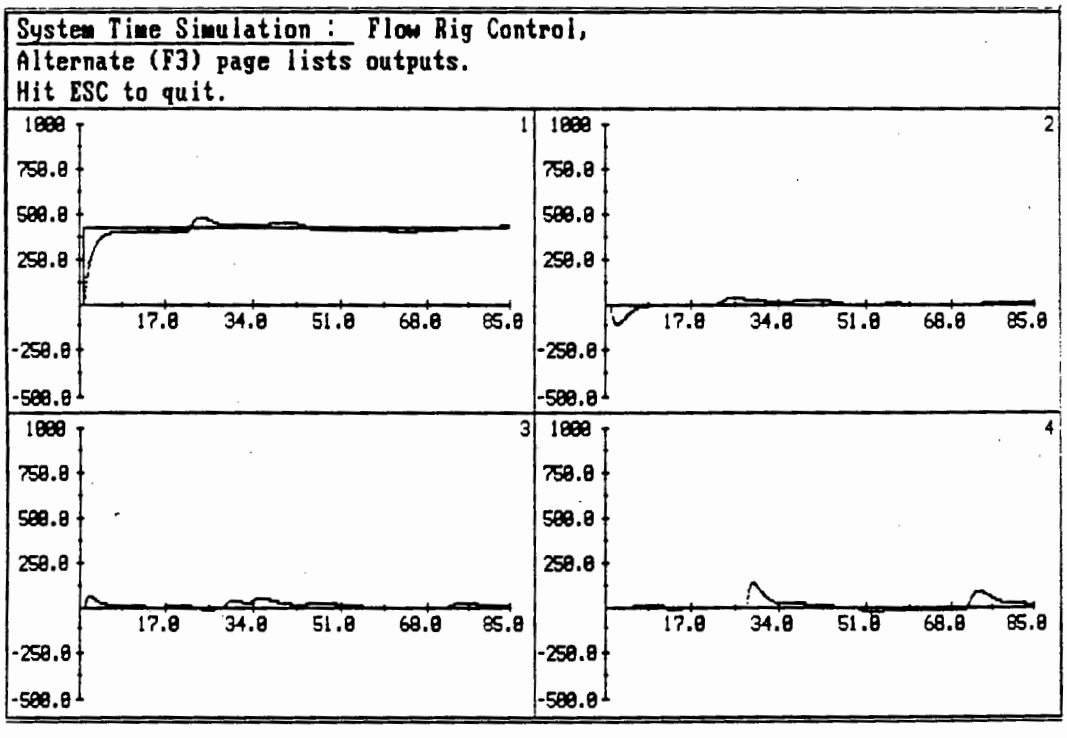


Figure 5.7 Time simulation of Q(s) : Rougher stepped.

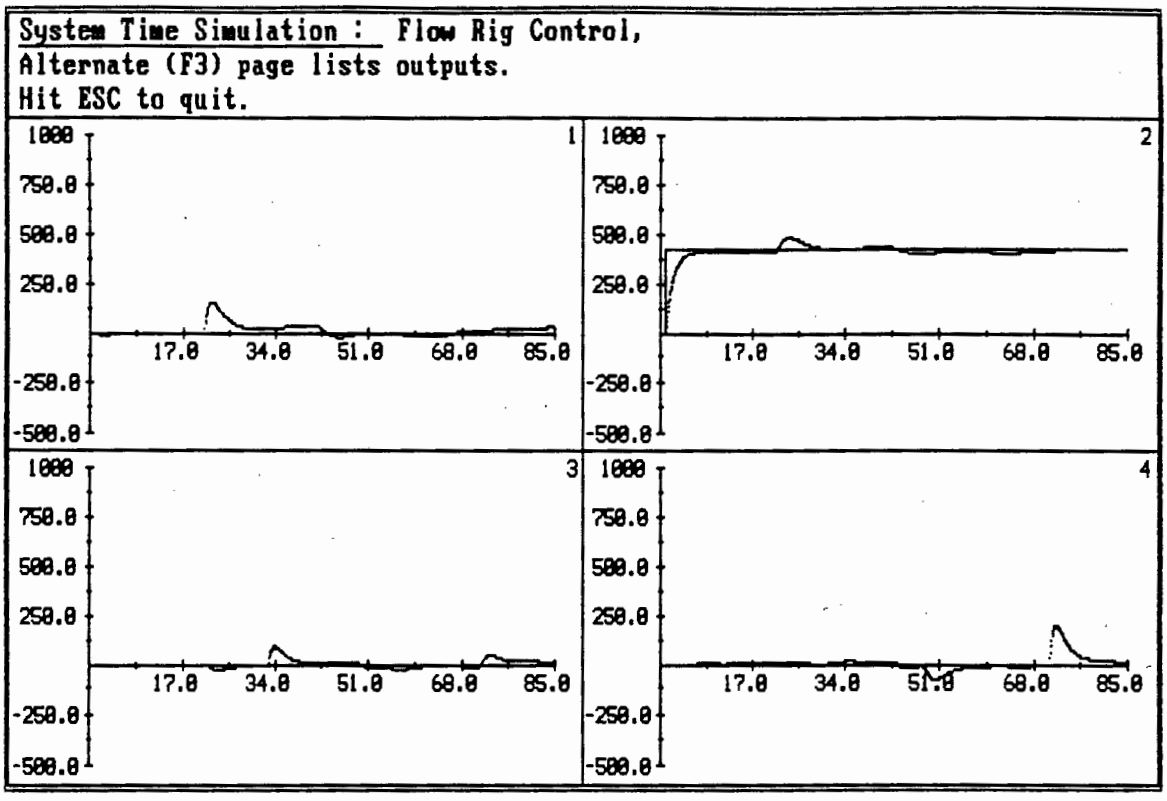


Figure 5.8 Time simulation of Q(s) : Scavenger stepped.

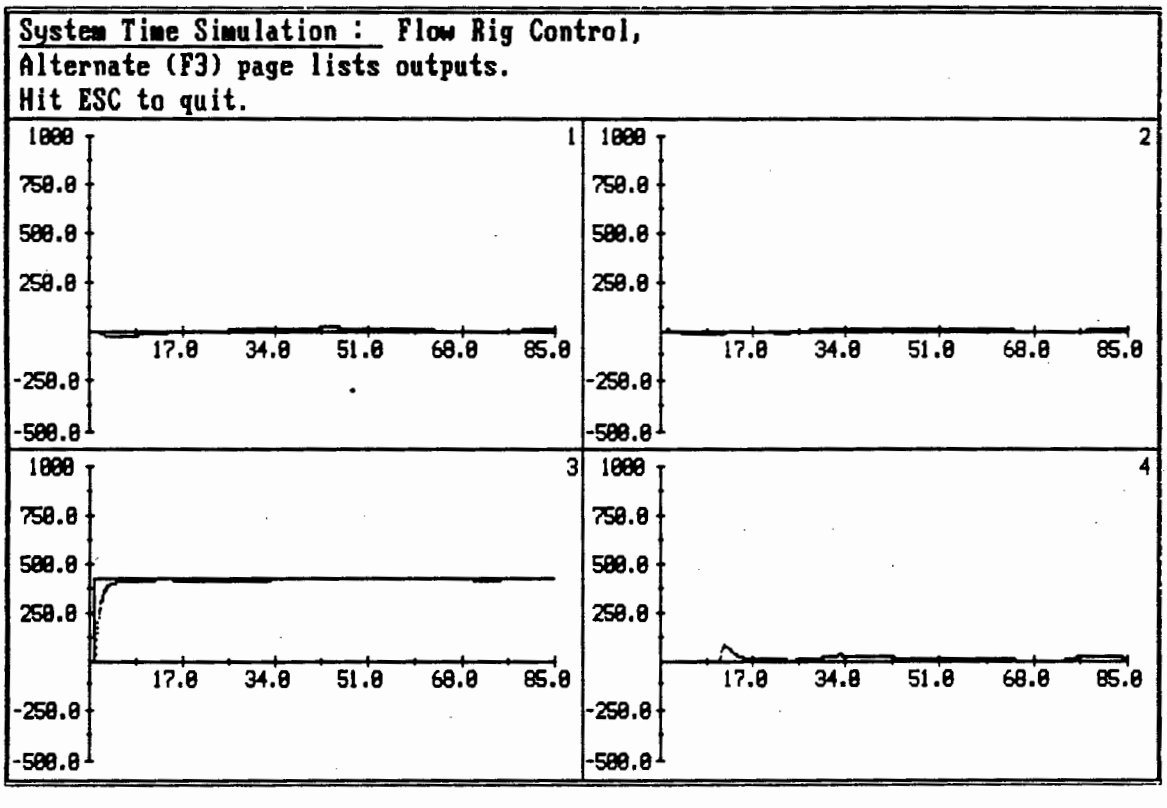


Figure 5.9 Time simulation of Q(s) : Cleaner stepped.

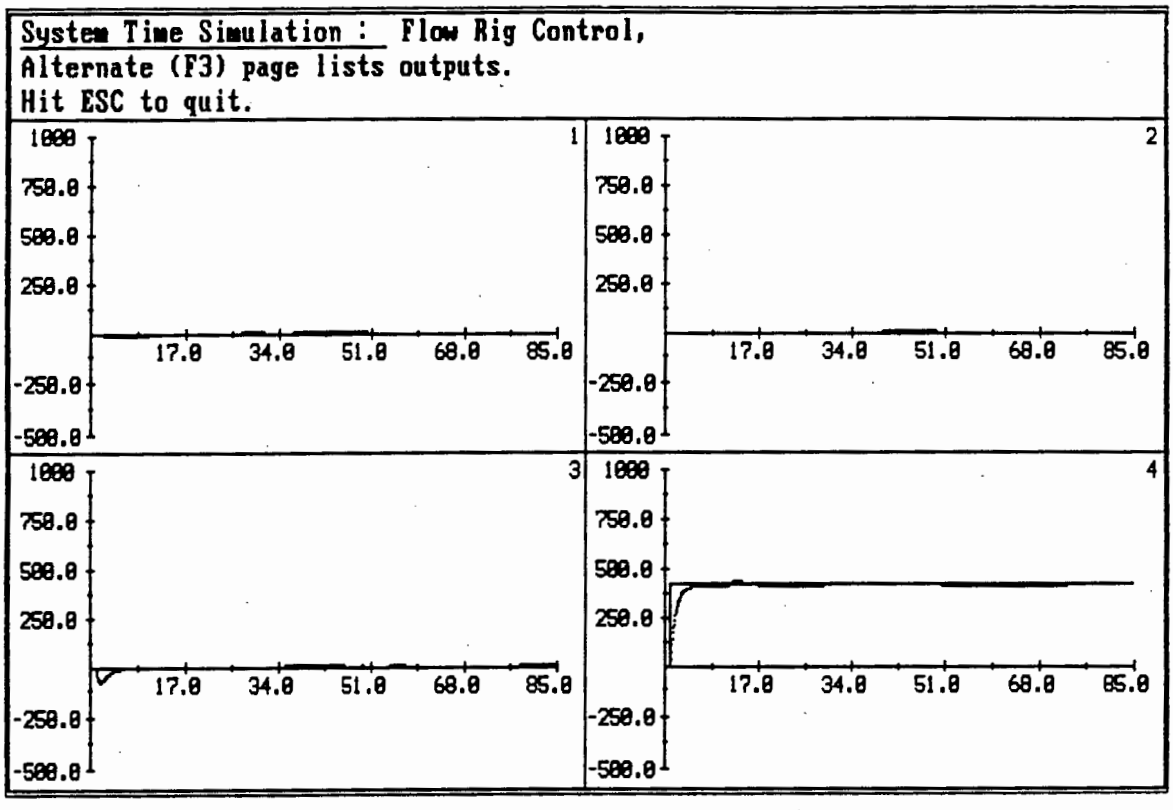


Figure 5.10 Time simulation of Q(s) : Recleaner stepped.

Designer's comments on the Characteristic Loci design technique :

i) The ability to check on system stability and integrity at any stage of the design process is the strongpoint of this design technique.

ii) A weakpoint of this technique is the lack of the ability to identify problematic elements or factors within the system.

iii) The accuracy and reliability of the interactive analysis, especially at high frequencies (or the misalignment angles), is questionable. One must always remember that misalignment angles are not an indicator of diagonal dominance of the system.

iv) The designer often appears to be working blind. For example, the design of the PI matrix is a methodical process but the selection of the scaling matrices, D_1 and D_2, tends to be a trial and error process and visualising the effect of the matrices on the system is difficult.

v) The use of a PI matrix controller to reduce interaction within the system is not a good concept. One can visualise the PI matrix as a constant matrix cascaded with a diagonal matrix with PI controllers on the diagonal. The designer is thus, trying to make the system diagonally dominant using only a constant matrix.

5.2 Implementation of Control Scheme

The control scheme developed in section 5.1 is to be implemented on a personal computer which is interfaced to the Flotation Plant Simulator. The controller, K(s), had to be converted into the z-domain to form K(z). The controller terms were transformed from the s-domain to the z-domain using tables and a sampling/control interval of one second.

Once implemented, the controller presented the problem of integral "wind-up". An example of the effects of integral wind-up is shown in figure 5.11 below (The solid line in each graph represents the setpoint setting of each cell), where all four setpoints are stepped simultaneously.

All the levels overshot their respective setpoints. The worst response of the four cells was the Scavenger, where the level overshot the setpoint for 250 seconds. Under these "wind up" conditions, the control values output to the system saturated for an undesirable length of time.

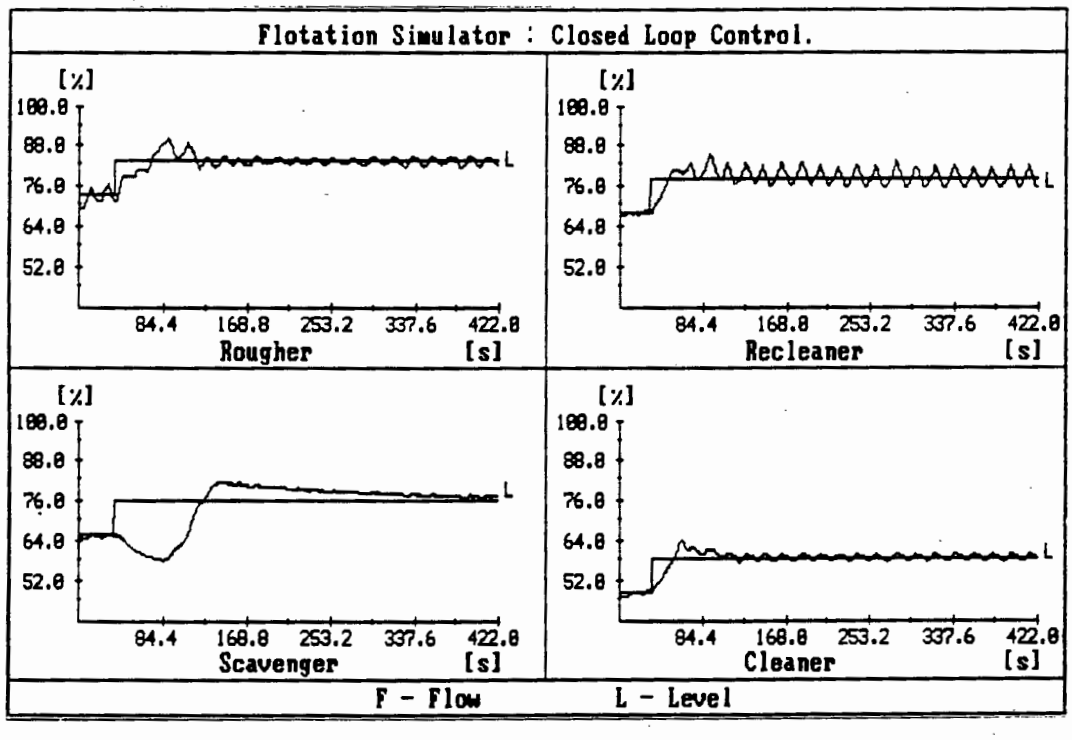


Figure 5.11 Flotation Plant Simulator : Example of "Wind-up".

In order to avoid this problem of integral "wind-up", a rate-algorithm was implemented. The rate algorithm can be described in four steps :

Step 1 : Differentiate the error signal.

Step 2 : Perform the controller calculation on the differentiated error signal.

Step 3 : Integrate the output of the controller calculation.

Step 4 : Do bounds limiting on the output of step 3 and output the result to the plant.

The performance of the rate algorithm can be seen in figure 5.12 which is the result of the same step test described in figure 5.11, where all four plant inputs are stepped simultaneously. Figure 5.13 shows the system actuator control signals during the same test.

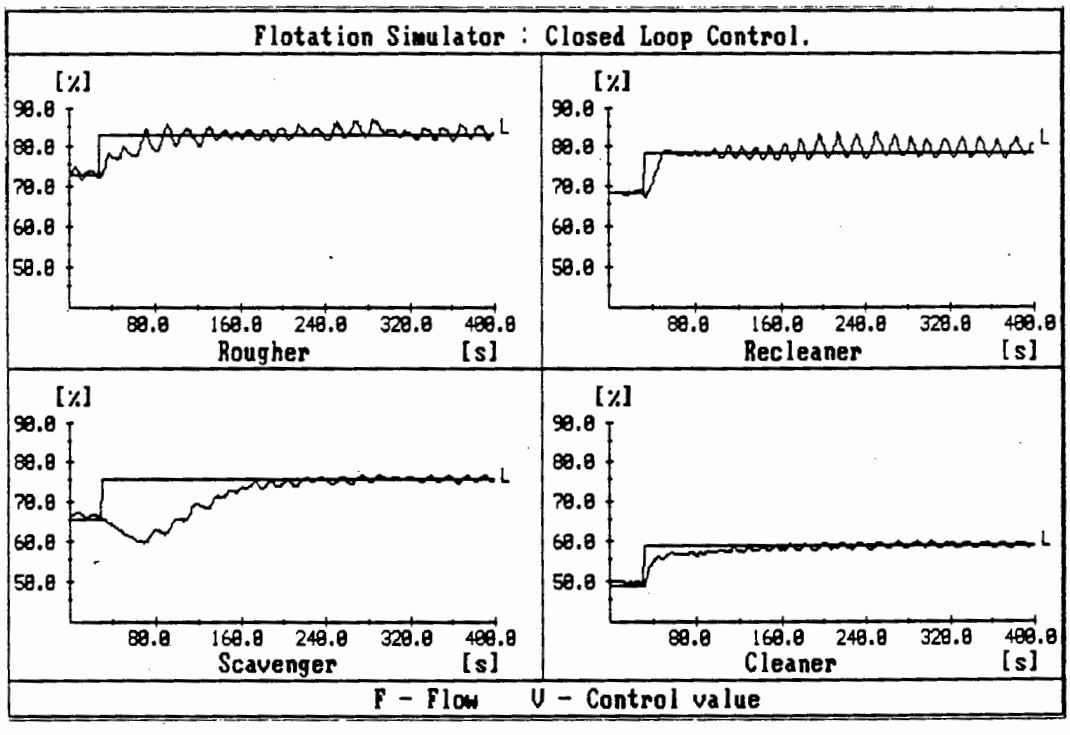


Figure 5.12 Flotation Plant Simulator : Rate algorithm implemented.

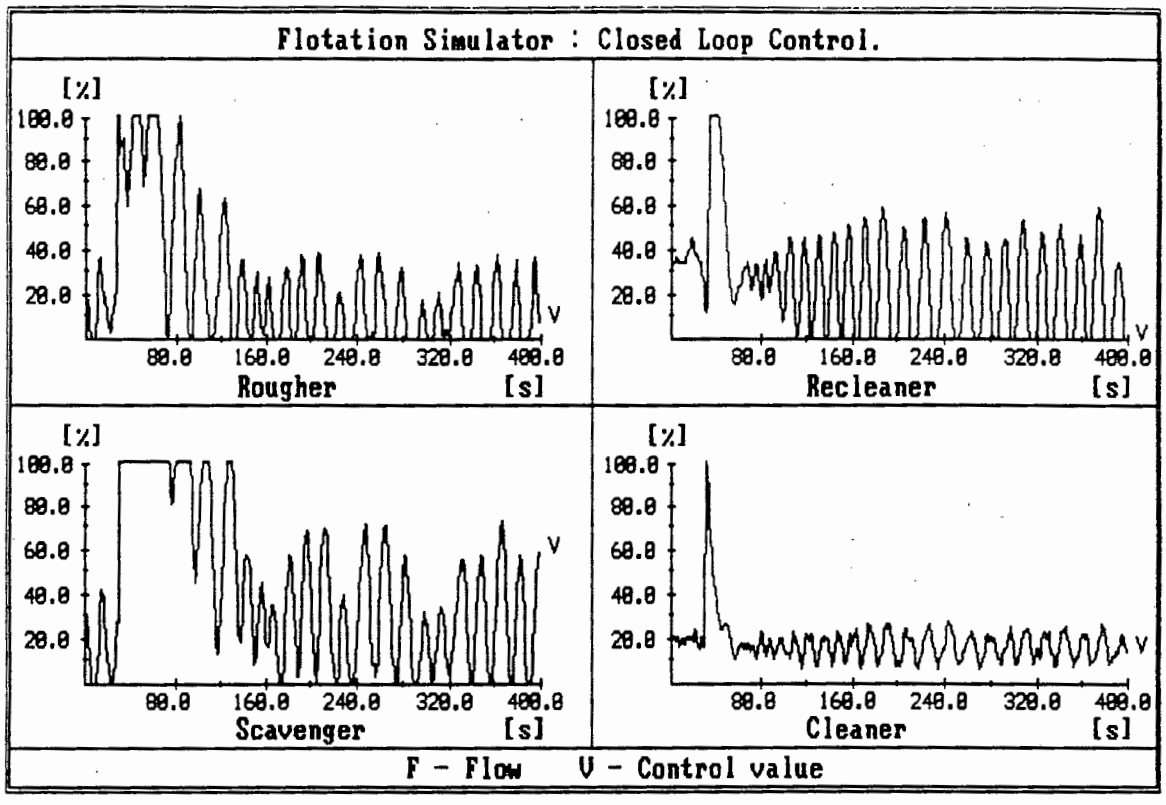


Figure 5.13 Flotation Plant Simulator : Rate algorithm implemented - actuator control values.

The performance of the system appears to have improved since the rate algorithm has been implemented. Figure 5.12 shows that no cell overshot its setpoint. And figure 5.13 shows that the control values output to the system actuators do not saturate for any significant length of time (ie. the "wind-up" effect has been eliminated). The Scavenger control value is an exception, but the excessive time span for which the Scavenger control value saturates can be explained as follows :

> When the system setpoints are stepped, the Scavenger level drops although the controller closes the Scavenger tail valve completely. This is due to the combined effect of the Rougher tail valve also being closed off completely and the Scavenger froth outflow being unregulated. In other words, there is no inflow to the Scavenger but there is an outflow, thus the level drops.

This effect in turn increases the error signal to the controller which attempts to increase the already saturated control signal. The controller will continue saturating the Scavenger control signal until the Scavenger level starts to rise (or the error signal starts to decrease).
In this case of the four setpoints being stepped simultaneously, the Rougher only starts to "feed" the Scavenger a significant quantity of product once the Rougher level has almost reached the setpoint.

The test of stepping all four setpoints simultaneously is a severe test of the system and one should expect the control values to saturate. But, under normal operating conditions, the control values do not saturate for any significant length of time. Thus, the results listed in the remainder of this chapter do not show the system control values.

The table below lists the approximate response times of the individual cells for this particular test. The two time columns represent the controller with the rate algorithm excluded and then included.

Cell	No Rate Algorithm [Secs]	Rate Algorithm [Secs]
Rougher	90	90
Scavenger	400	210
Cleaner	75	140
Recleaner	70	30

Table 5.1 Response times to setpoints stepped simultaneously

Table 5.1 shows that the response times of the system are generally faster, especially the Scavenger and the Recleaner. The exception to this characteristic is the Cleaner, where the

response time doubled when the rate algorithm was implemented. But this figure is not a good indication of the response of the Cleaner as one should note that the Recleaner had attained 70% of the setpoint change within 30 seconds. This result is similar to the response time of the Recleaner when no rate algorithm had been implemented.

Thus, from a time response, setpoint overshoot and "wind-up" point of view, the rate algorithm does improve the performance of the closed-loop system.

Once the rate-algorithm had been implemented and the performance of the controller had been observed it became obvious that tuning of some of the controller parameters was required. Only the parameters on the diagonal of the controller matrix were tuned. The modified PI terms on the controller diagonal were implemented as follows :

$$k_{11}(s) = \frac{(1.500\text{x}10^{1})s + (1.500)}{s}$$

$$k_{22}(s) = \frac{(3.500\text{x}10^{1})s + (1.000)}{s}$$

$$k_{33}(s) = \frac{(1.200\text{x}10^{1})s + (0.230)}{s}$$

$$k_{44}(s) = \frac{(1.500\text{x}10^{1})s + (2.000)}{s}$$

The remaining terms in the controller were not modified.

The simulated transient responses of the modified closed-loop system, $Q_2(s)$, for step changes to each setpoint are shown in the following figures. The following plots are numbered (in top-right corner) according to system input number : 1 = Rougher, 2 = Scavenger, 3 = Cleaner, 4 = Recleaner.

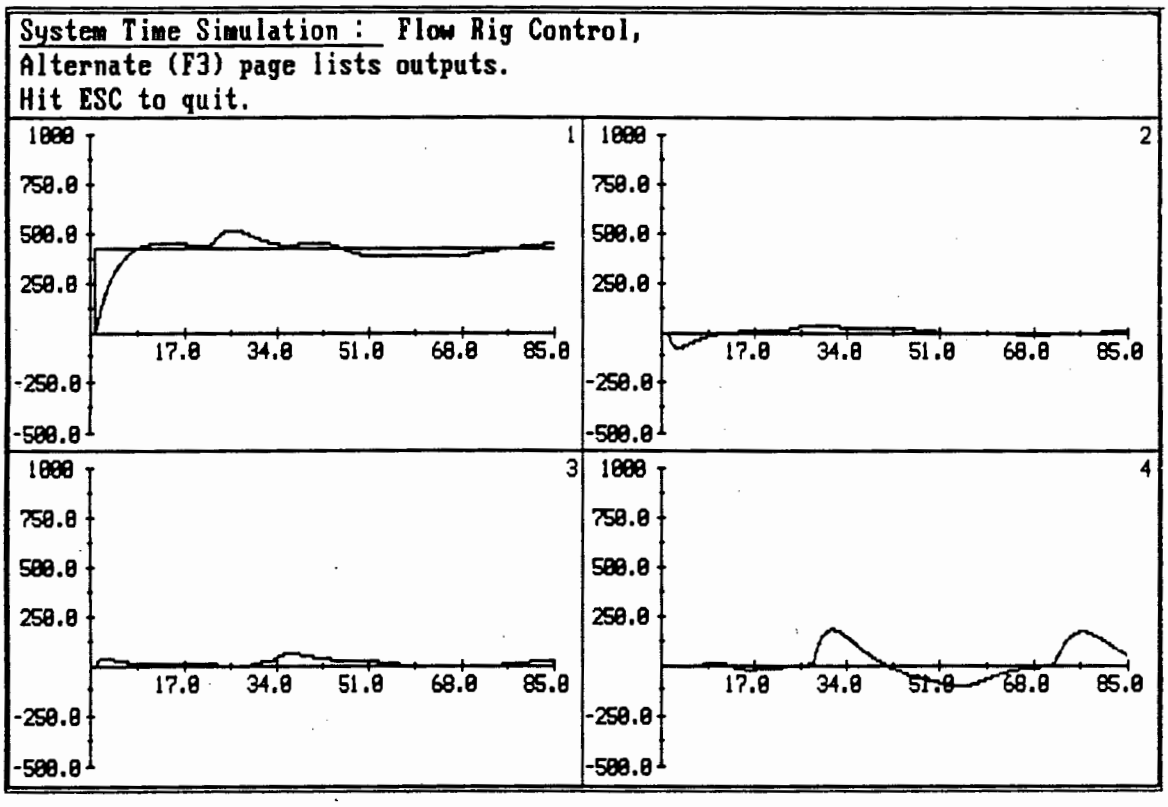


Figure 5.14 Time simulation of $Q_2(s)$: Rougher stepped.

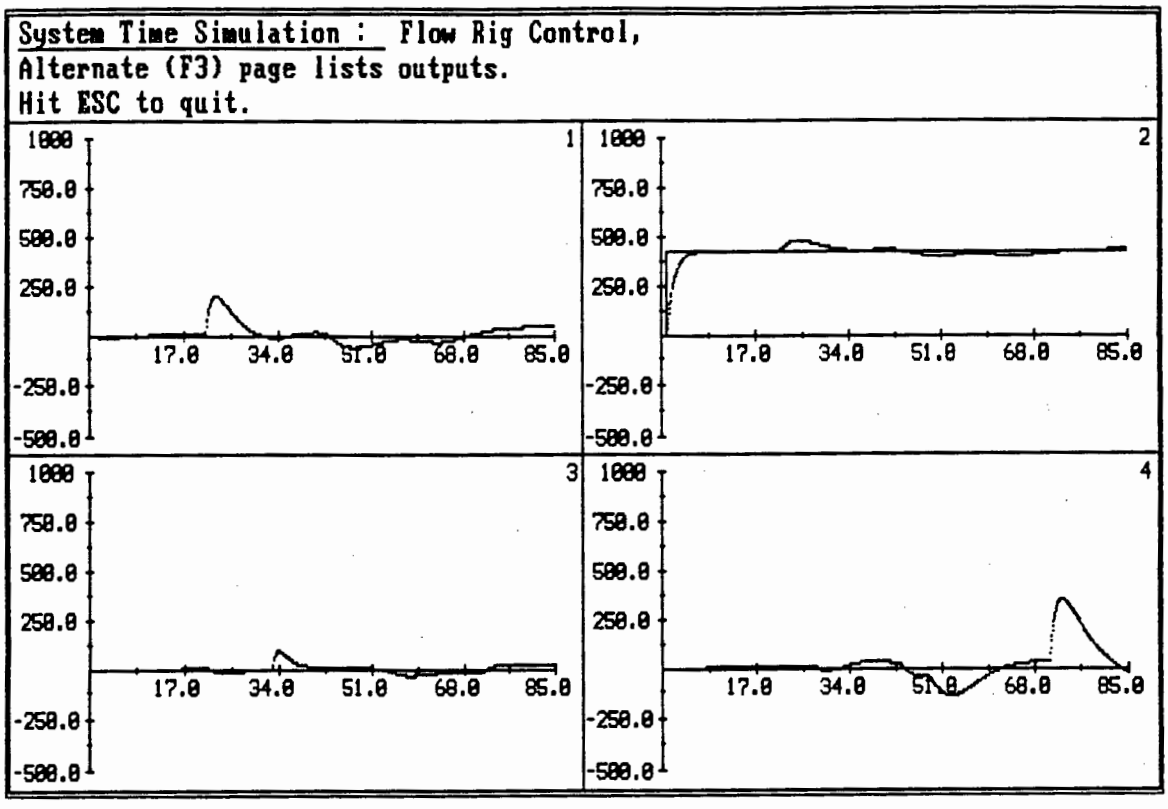


Figure 5.15 Time simulation of $Q_2(s)$: Scavenger stepped.

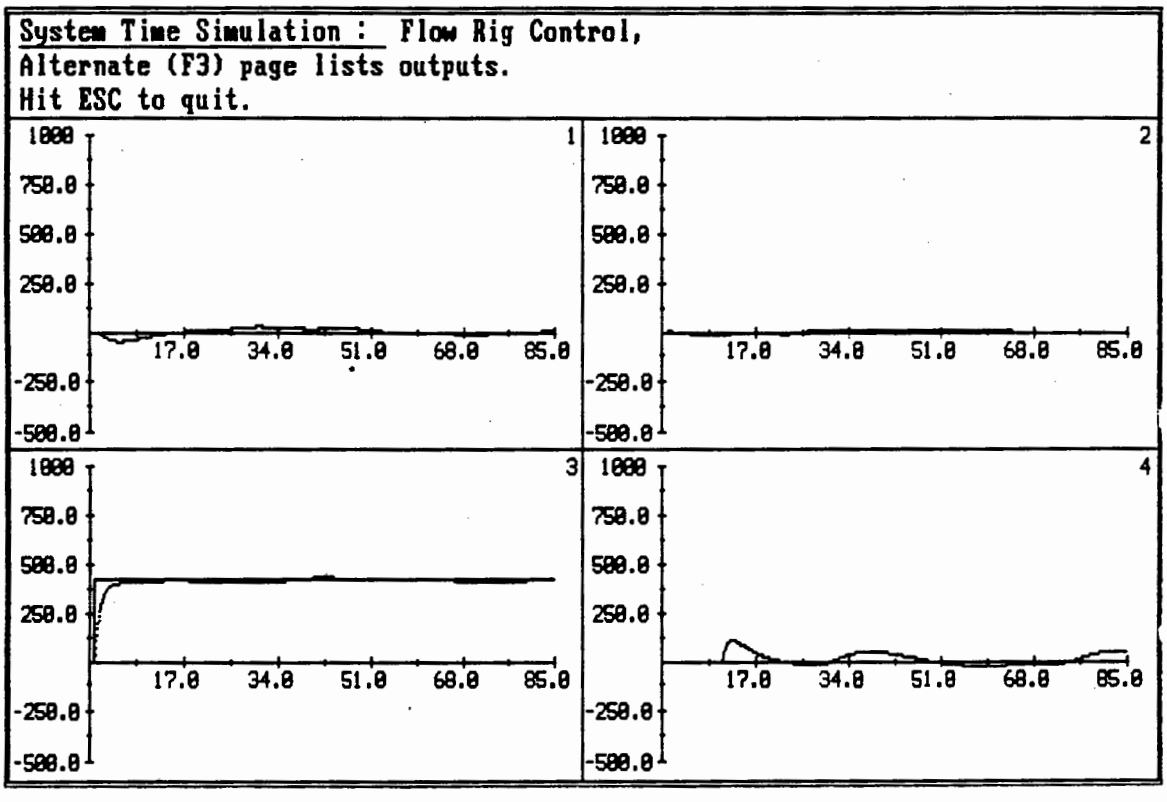


Figure 5.16 Time simulation of $Q_2(s)$: Cleaner stepped.

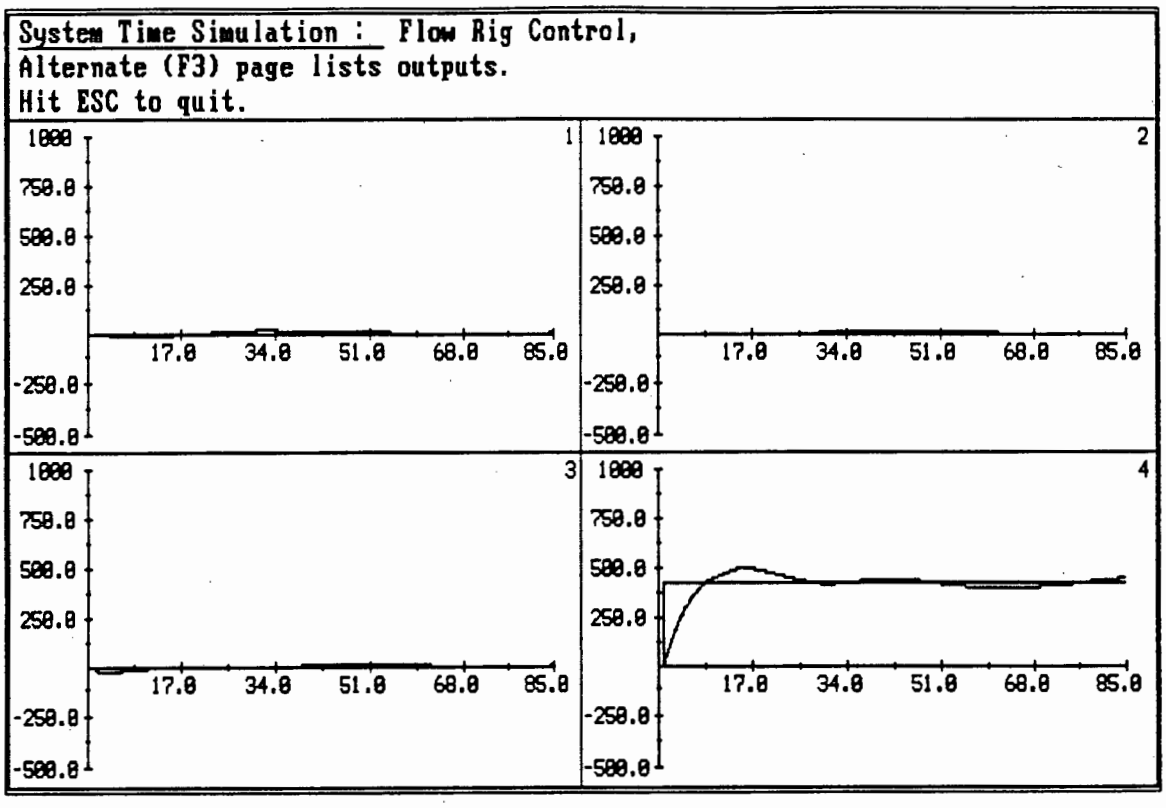


Figure 5.17 Time simulation of $Q_2(s)$: Recleaner stepped.

The transient response of the system to a single stepoint step are shown in the following figures. The setpoint is stepped 10% in each test. The solid line in each graph represents the setpoint setting for the level of each cell of the Flotation Plant Simulator.

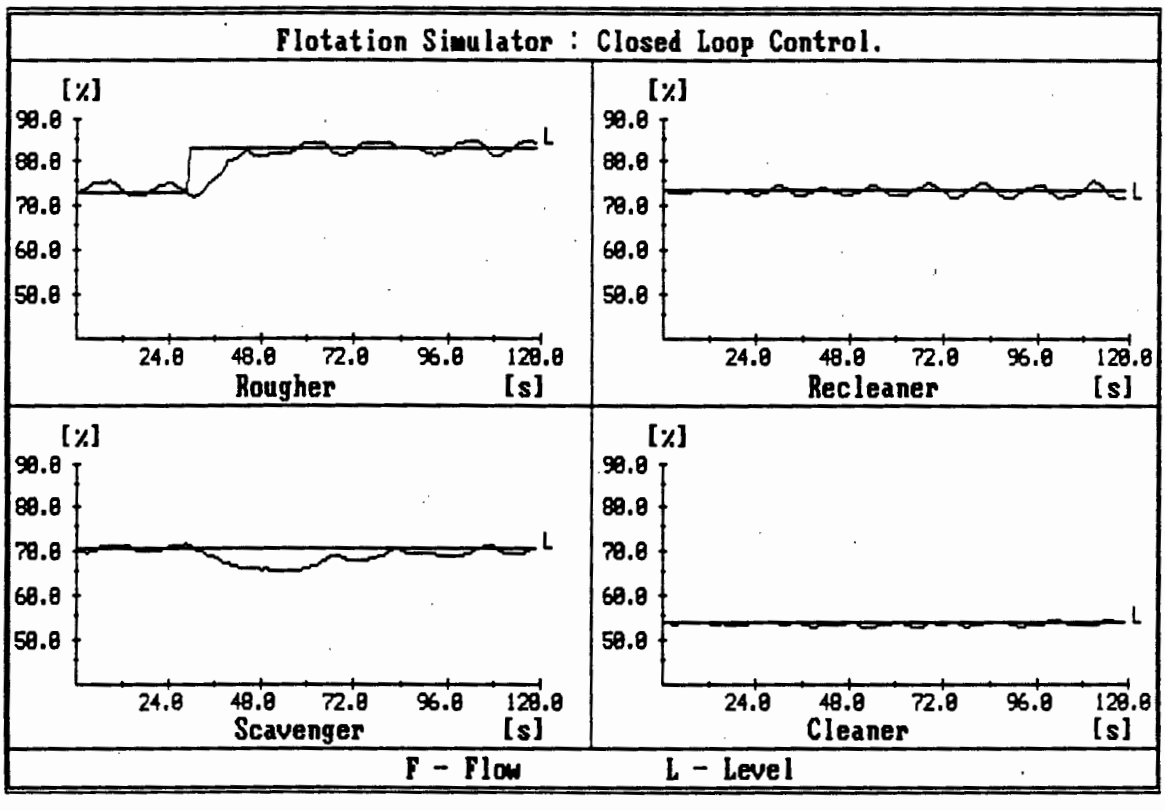


Figure 5.18 Flotation Plant Simulator : Rougher stepped 10%

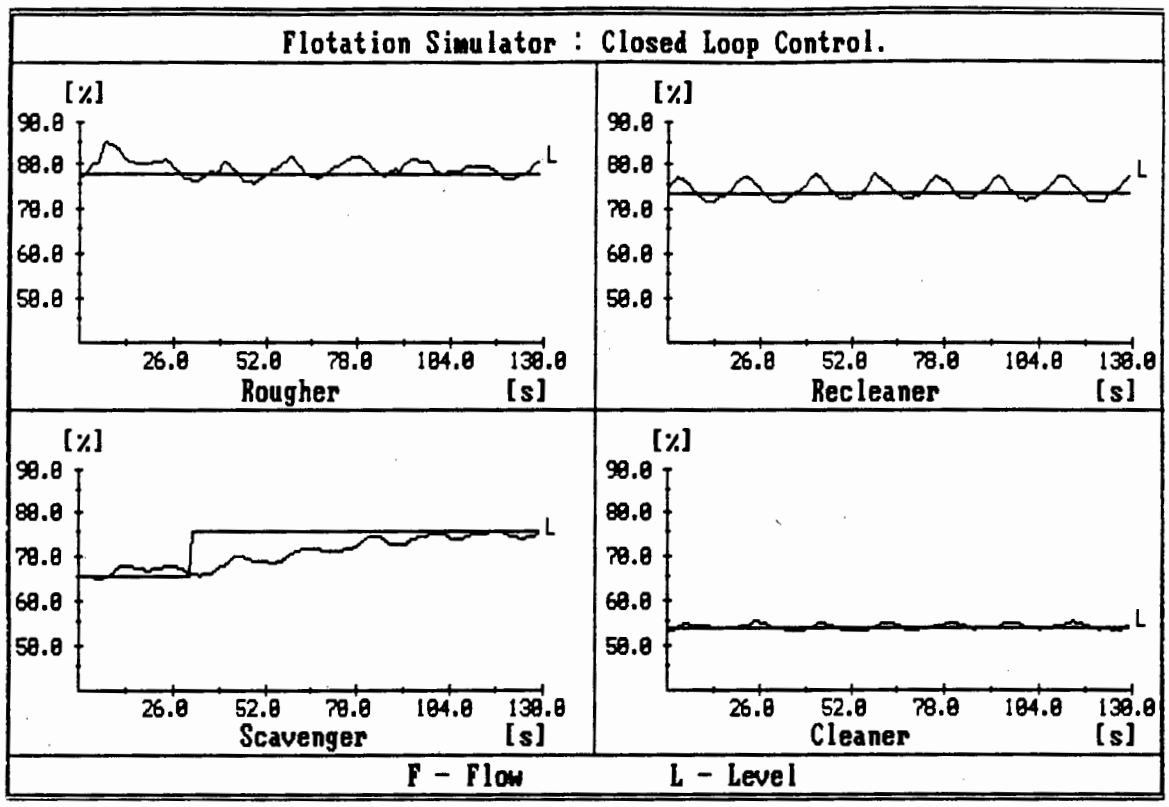


Figure 5.19 Flotation Plant Simulator : Scavenger stepped 10%

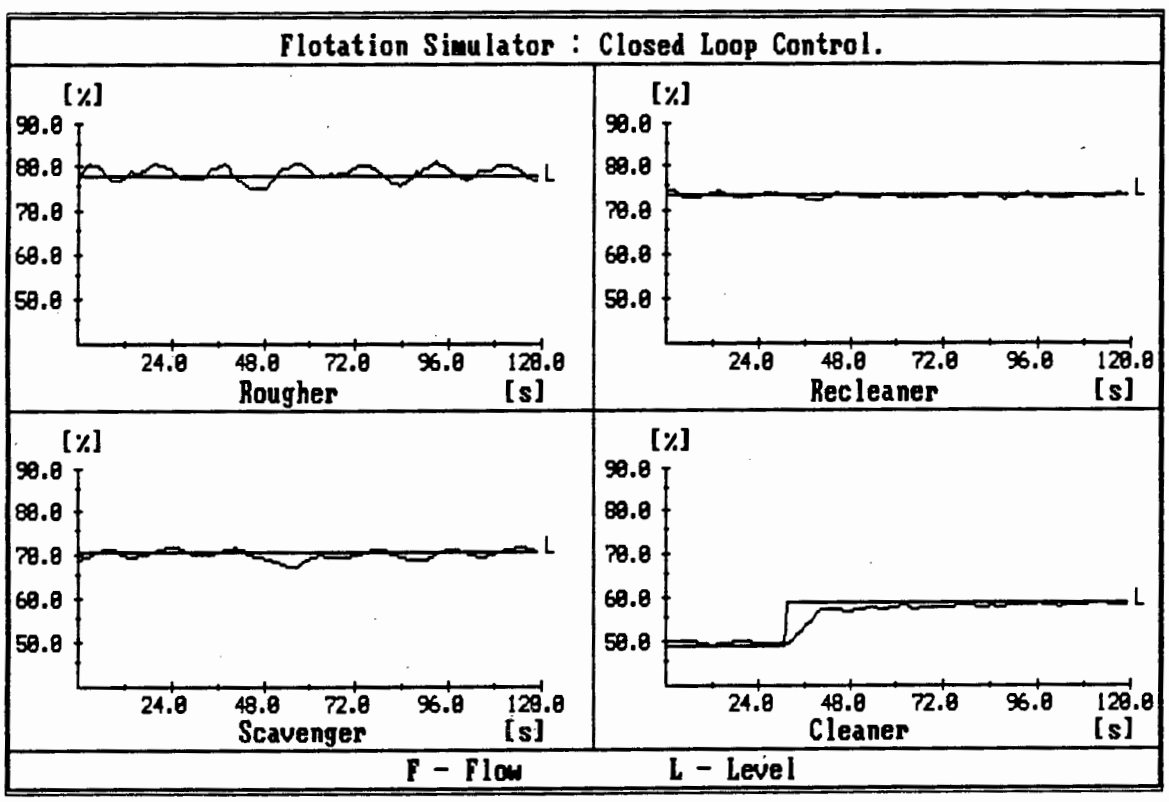


Figure 5.20 Flotation Plant Simulator : Cleaner stepped 10%

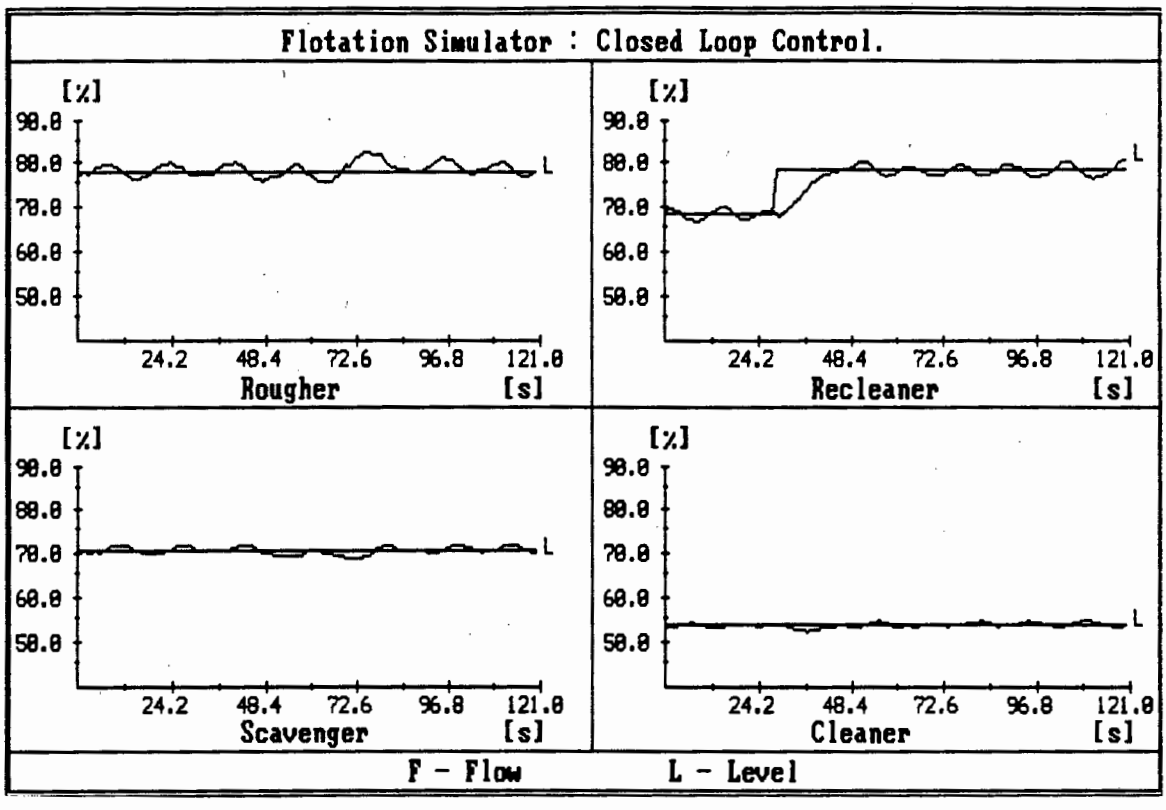


Figure 5.21 Flotation Plant Simulator : Recleaner stepped 10%

The oscillations visible on the cell levels, especially the Rougher and Recleaner levels is due to the pumps surging as described in appendix F, Section F.1. The controller has reduced the magnitude of the oscillations when comparing the responses of the open-loop system (chapter 2) and the closed loop system. But, the oscillations could not be eliminated completely due to the frequency of the oscillations being too high.

The table below lists the response times of the the system to a step change to one of the cells setpoint's. The two time columns represent the simulated time response and the actual time response of the cell whose setpoint was stepped.

Cell Stepped	Simulated Response [Secs]	Actual Response [Secs]
Rougher	18	20
Scavenger	10	100
Cleaner	10	25
Recleaner	25	25

Table 5.2 Simulated and actual time responses to single setpoint steps.

Rougher Step : Table 5.2 shows that there is not a large discrepancy between the simulated and actual response times of the Rougher to a setpoint step. But, comparing figures 5.14 and 5.18 reveals that the response curves are quite different. The actual Scavenger response to the Rougher step is more significant than indicated in the simulation.

This difference can be explained in terms of flow rates and control values being bounded in the actual system and unbounded in the time simulations. The values used in the actual controller calculations are kept within the system bounds by the presence of the rate algorithm. But, in the time simulation, the controller values and flow rates represented by the model exceed magnitudes that can exist on a real system. This can result in the time simulation responses being faster than those on the actual system. And, the amount of interaction in the simulation of the system will be greater than the interaction that occurs on the actual system.

The effects described above explain why i) the Rougher response is slower, ii) the Scavenger response is larger and iii) the interaction on the Recleaner is smaller than simulations predict (ie. The feed rates from the Rougher to the Scavenger and Recleaner are lower in the actual system than in the simulated system).

Scavenger Step : Figures 5.15 and 5.19 reveal a large discrepancy between the actual and simulated system response to a setpoint step on the Scavenger cell (results in Table 5.2). This discrepancy is due to actual system constraints limiting the control values as described above for the Rougher step. The feed rate from the Rougher to the Scavenger was less than predicted by the simulation. And similarly the Recleaner reaction to the Scavenger step was less significant than predicted by the simulation.

Cleaner Step : The simulated and actual response time of the Cleaner (figures 5.16 and 5.20) to a setpoint step differ by only 5 seconds. And the Recleaner response to the Cleaner step is less significant than predicted in the simulation. These differences can again be explained by limited values being used in the controller calculations and output to the system as discussed previously.

Recleaner Step : The Recleaner response to a setpoint step was predicted quite accurately by the time simulation (figures 5.17 and 5.21). The response times are the same but the simulation predicts that the level will overshoot the setpoint which does not actually occur. This discrepancy could be due to lower feed rates to the Recleaner than predicted by the time simulation.

The remaining cells' response (almost non-existent) to the Recleaner setpoint step is virtually identical as predicted by the time simulation.

The following figures show the system response to a disturbance of ±10% on each setpoint.

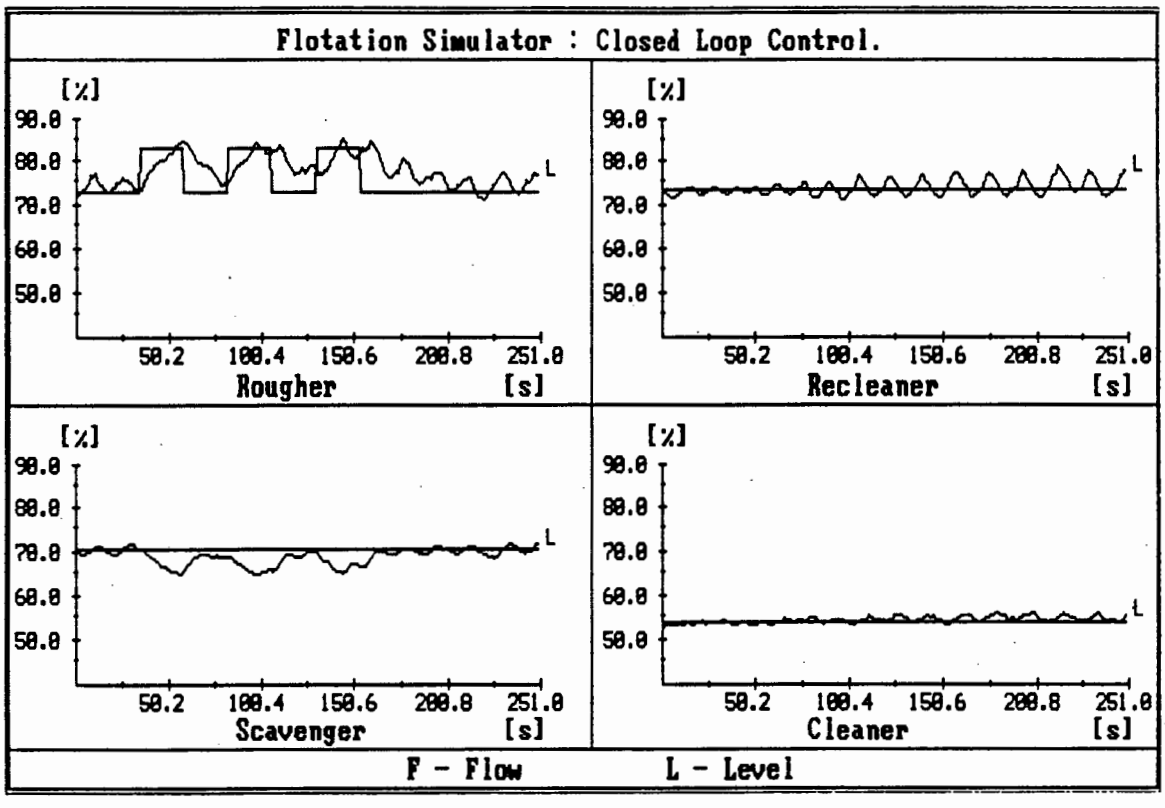


Figure 5.22 Flotation Plant Simulator: Rougher disturbed ±10%

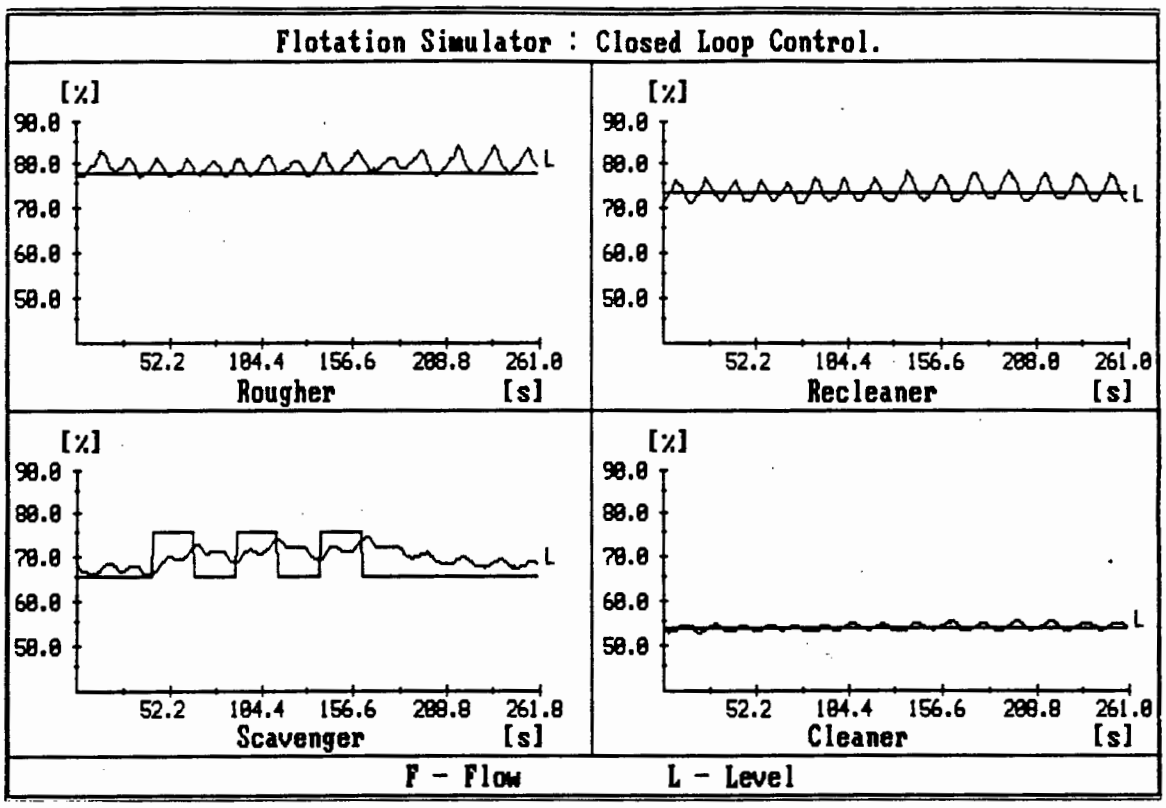


Figure 5.23 Flotation Plant Simulator: Scavenger disturbed ±10%

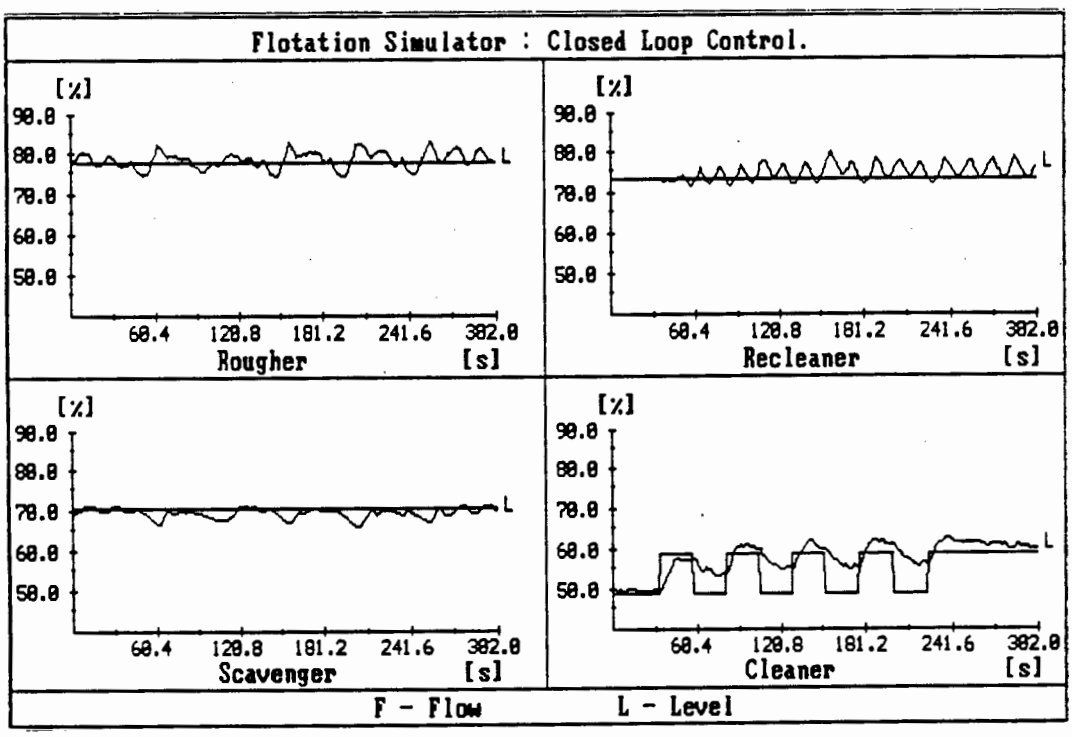


Figure 5.24 Flotation Plant Simulator: Cleaner disturbed ±10%

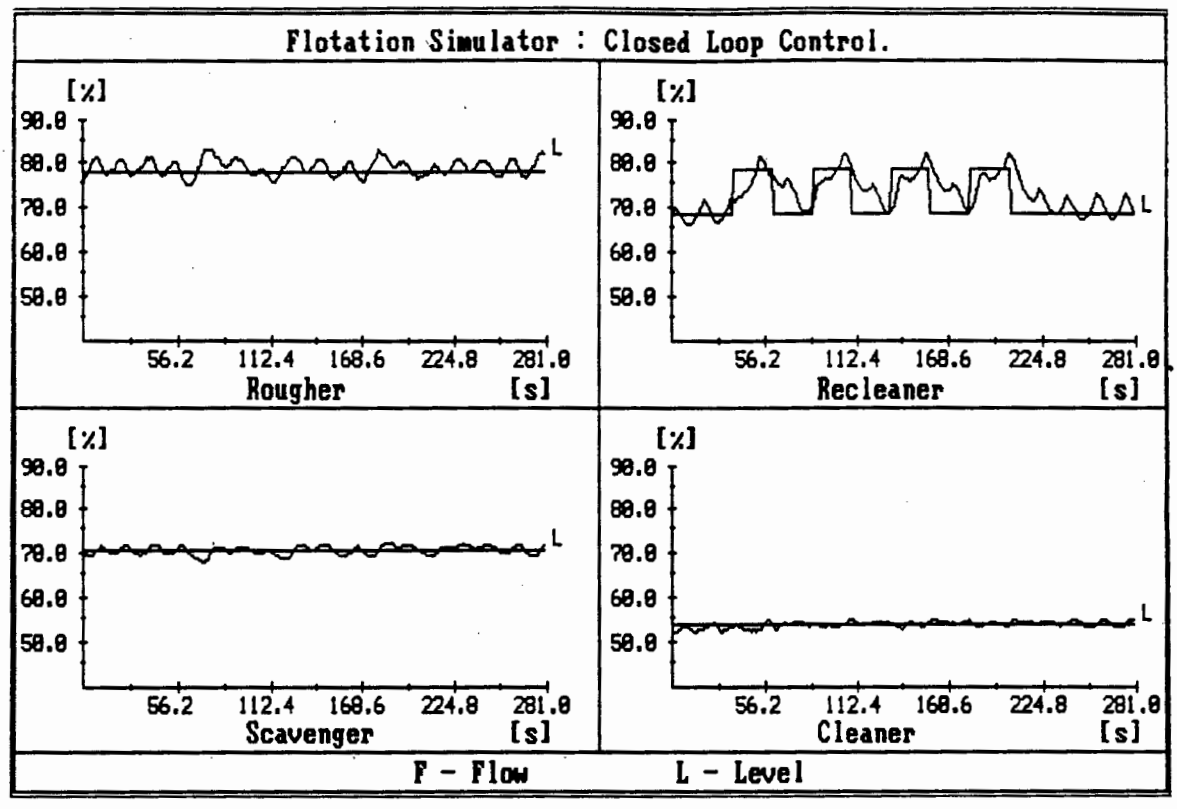


Figure 5.25 Flotation Plant Simulator: Recleaner disturbed ±10%

The system remains stable for all of the setpoint disturbance conditions described in figures 5.22 to 5.25. Also, the system responses to the setpoint disturbances do not reveal any new system artifacts.

Rougher Disturbance : Figure 5.22 reveals that the Scavenger's response to the Rougher setpoint disturbance is the most significant (apart from the Rougher itself) when compared to the remaining cells. This response is expected if one considers the system response to a single setpoint step on the Rougher, shown in figure 5.18.

The remaining cells (Cleaner and Recleaner) are not affected significantly by the Rougher setpoint disturbance. The only apparent interaction on the remaining cells (figure 5.22) is on the Recleaner, where the sump pump that feeds the Recleaner has increased the amplitude of its "surge-cavitate" cycle (as described in appendix F, section F.1). The increase in the

amplitude may be due to the slight variation in feed rate from the Cleaner during the Rougher setpoint disturbance. The efficacy of the sump pumps were sensitive to variations in feed rates to the sumps.

Scavenger Disturbance : Figure 5.23 reveals that the remainder of the system is not affected significantly by the Scavenger setpoint disturbance. This response is expected if one considers the system response to a single setpoint step on the Scavenger, shown in figure 5.19.

Cleaner Disturbance : The system response to a setpoint disturbance on the Cleaner, shown in figure 5.24, is as expected if one uses the experience gathered from the single setpoint step on the Cleaner (figure 5.20) and the system's response to previous setpoint disturbances : The interaction visible on the Rougher and Scavenger is as expected from the system response to a single setpoint step on the Cleaner. The interaction present on the Recleaner is the result of the sump pump "surge-cavitate" cycle amplitude increasing due to the varying feed rate from the Cleaner. (This sump pump characteristic was mentioned previously).

Recleaner Disturbance : The system response to a setpoint disturbance on the Recleaner, shown in figure 5.25, does not reveal any new artifacts when compared to the system response to a single setpoint step on the Recleaner (figure 5.21). There is a small amount of interaction on the Rougher and Scavenger but the magnitude is insignificant when compared to the noise provided by the sump pump "surge-cavitate" cycle (appendix F, section F.1).

The following figures show the response of the system to failures in the loops. Feedback (transducer) failures were induced by actually disconnecting the level-sensor output from the computer interface circuitry. Similarly, the computer interface outputs to the actuators were disconnected to simulate actuator failures.

Only two responses from each type of failure have been presented here as all the responses to the "failures" are similar. Comments made about the shown responses can be applied to the failure responses not shown.

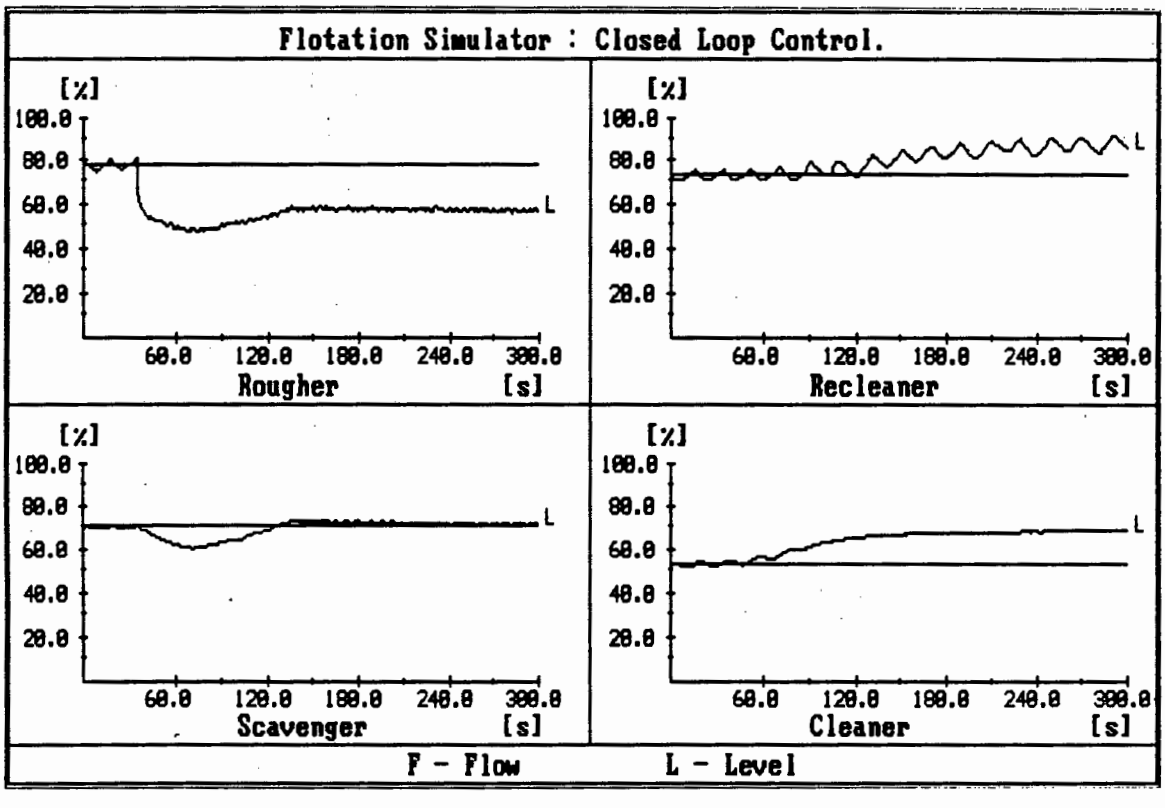


Figure 5.26 System response : Rougher level sensor failure.

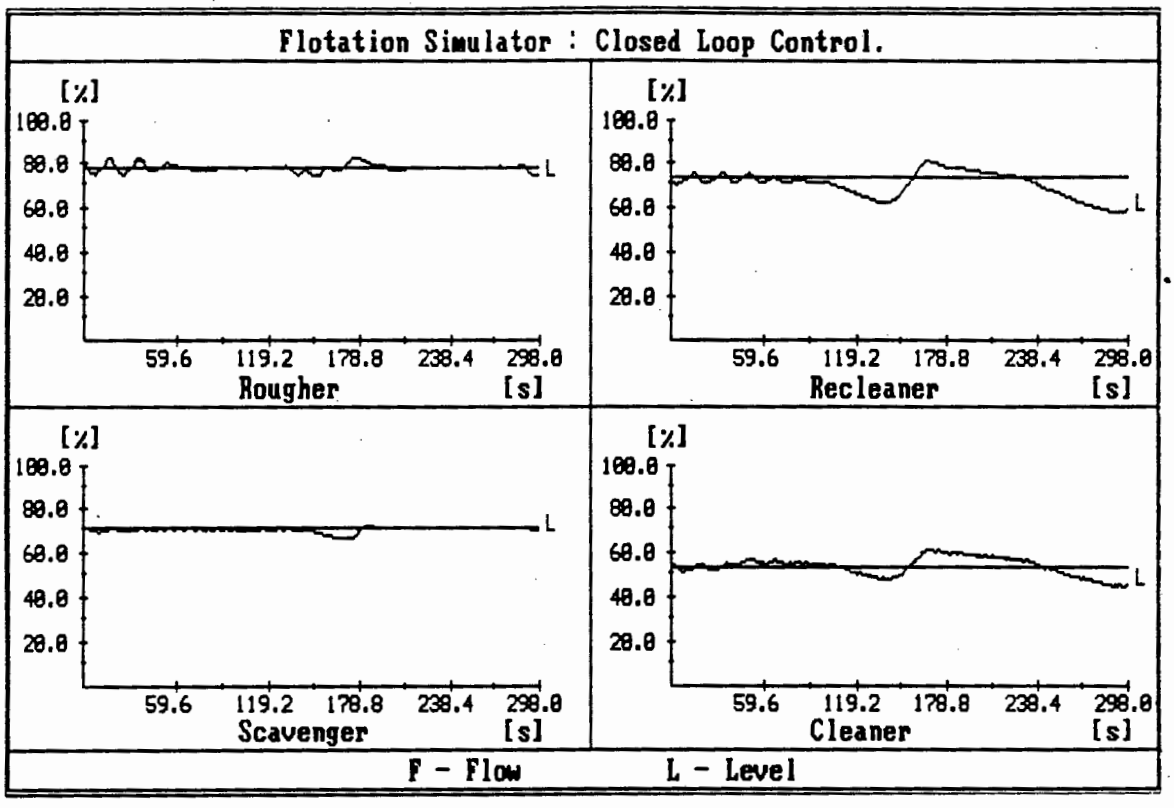


Figure 5.27 System response : Cleaner level sensor failure.

The levels indicated on the "failed-level-sensor" cell (in figures 5.26 and 5.27) are false. The level indicated is due to the computer interface circuitry having a floating (or non-grounded) input which can as a result "follow" one of the other inputs (see figure 5.27, where the cleaner level input follows the recleaner level).

The actual response of the system is to attempt to raise the level of the cell continuously (as shown in figure 5.26). This is due to the error signal being incorrect and the controller containing integrator terms which result in the control signals continuously increasing. The result is that the tank is continually filled to overflowing resulting in an unstable system. This phenomenon is shown in figure 5.26 where the controller tries to raise the "level" of the rougher which results in flooding the rougher and placing so much product in

the circuit that the system is placed beyond controllable bounds.

In the design phase of this controller, the integrity of the system was inspected. The result was that the remaining system would remain stable under single loop failure conditions if the remaining loops had positive gains (section 5.1). This does not appear to be the case in figure 5.26, where the system does not appear to be stable. But, the remainder of the system did remain stable while the product volume in the Flotation Plant Simulator remained within the controllable bounds of the system. In other words, the volume present in the product circuit reached a point where the Flotation Plant Simulator could not control the levels in the individual cells due to their product inputs being too great. Thus, it is possible to say that the theoretical and actual results do correlate as long as actual system constraints are ignored.

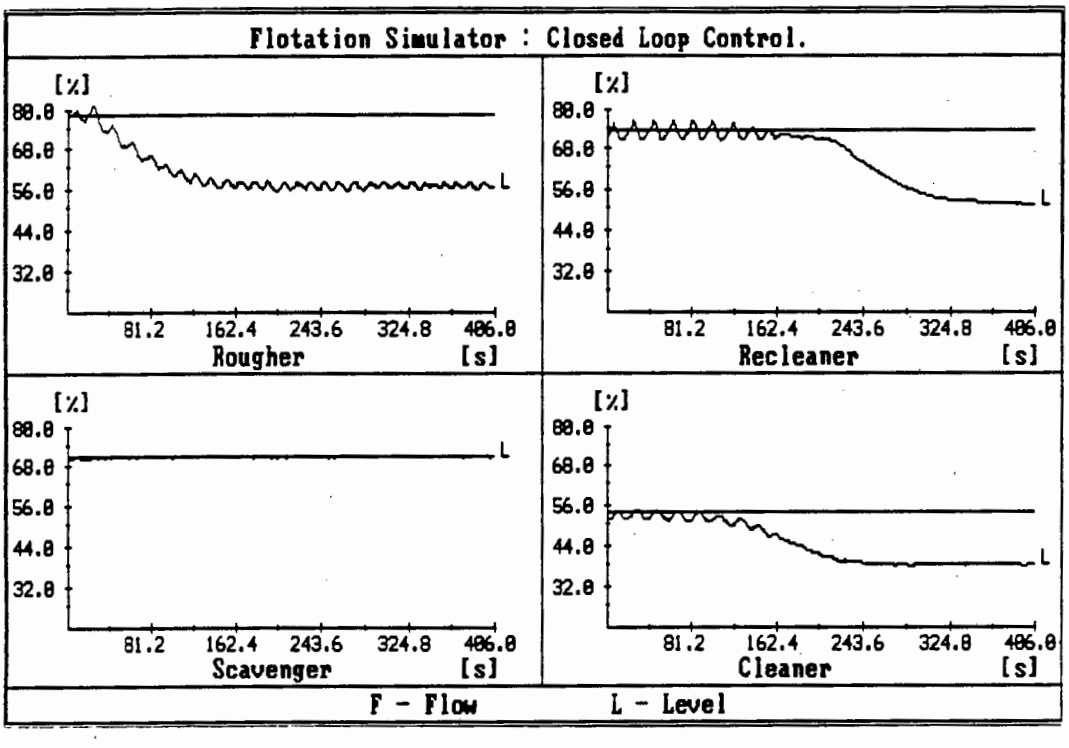


Figure 5.28 System response : Rougher actuator failure.

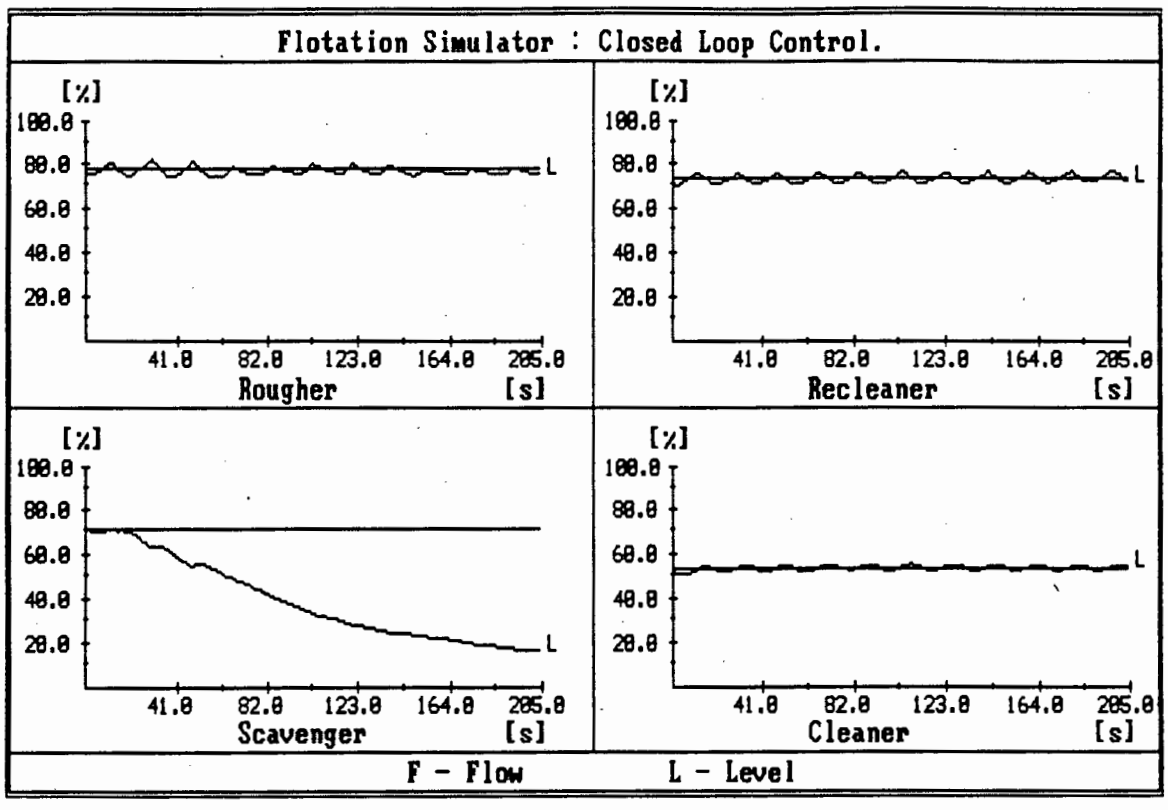


Figure 5.29 System response : Scavenger actuator failure.

The system does remain stable (results shown in figures 5.28 and 5.29) if any actuator fails. This is due to the fact that the valves are normally open when no signal is applied to the actuators. This obviously does not apply if the valve locks or jams in the closed position.

The error on one cell does not appear to affect the control of the other cells to any extent (figure 5.29). The cell's level response is obviously affected when the feed to that cell is cut-off or another cell which feeds that cell cannot provide enough product to maintain the required levels. This phenomenon can be seen in figure 5.28 where the Rougher feed to the Cleaner-Recleaner circuit is reduced. The Cleaner and Recleaner

try to maintain their required levels but their product input is just not high enough and the levels drop.

The theory in section 5.1, where the integrity of the system was inspected, revealed that the remaining system would be stable under single loop failure conditions if the remaining loops maintained positive gains. The system response to actuator failures does agree with the theory in that the system does remain stable whenever a single actuator fails.

5.3 Conclusion

The characteristic loci indicated that the closed-loop system would be stable and the bode plots predicted that interaction would be reduced at low frequencies, but the misalignment angles predicted that there would be interaction at high frequencies.

The simulated response of the closed-loop system confirmed the results of the Characteristic Loci technique, ie. the system did appear stable and interaction was apparent after steps to the setpoints, but the steady state errors were eliminated.

The actual response of the closed-loop system did, in general, agree with the theoretical predictions. The actual closed-loop system did appear to be stable and was still interactive, although the interaction had been reduced when compared to the open-loop system. The steady state errors of the closed-loop system were eliminated and the response times were similar to the predicted response times. There were discrepancies between the actual and simulated systems but explanations could be found for these discrepancies.

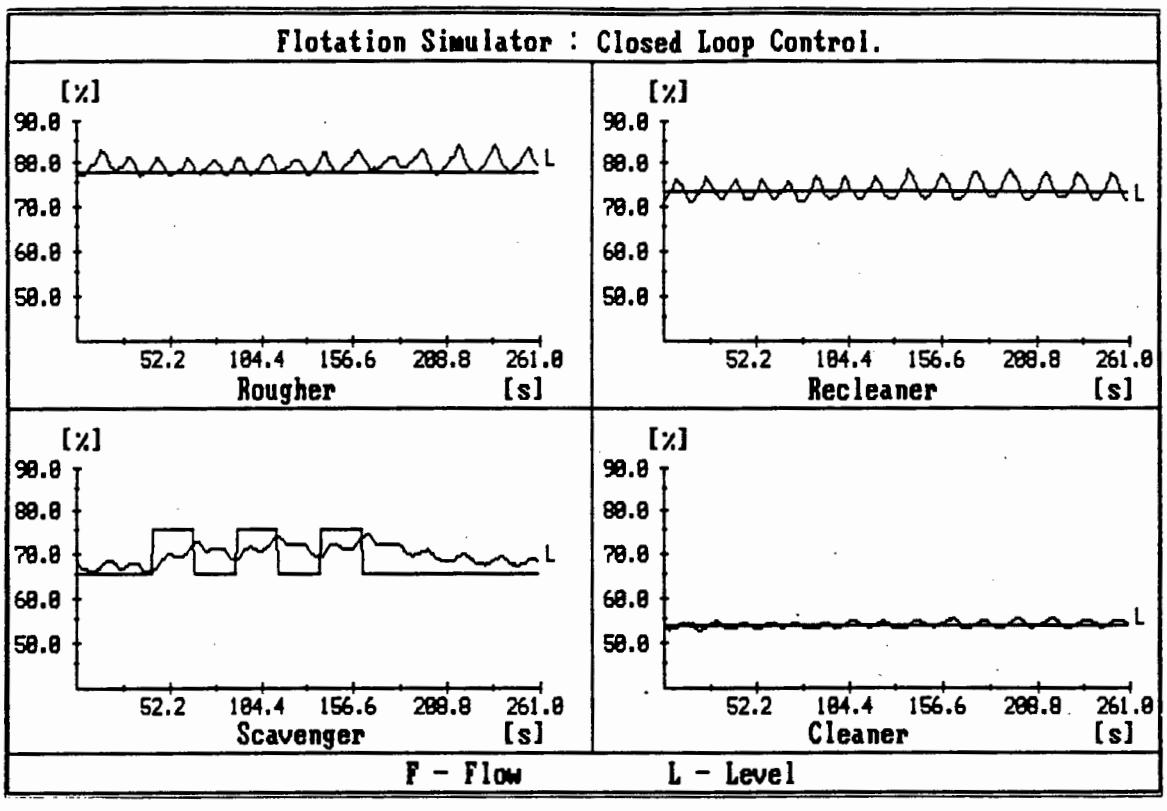


Figure 5.23 Flotation Plant Simulator: Scavenger disturbed ±10%

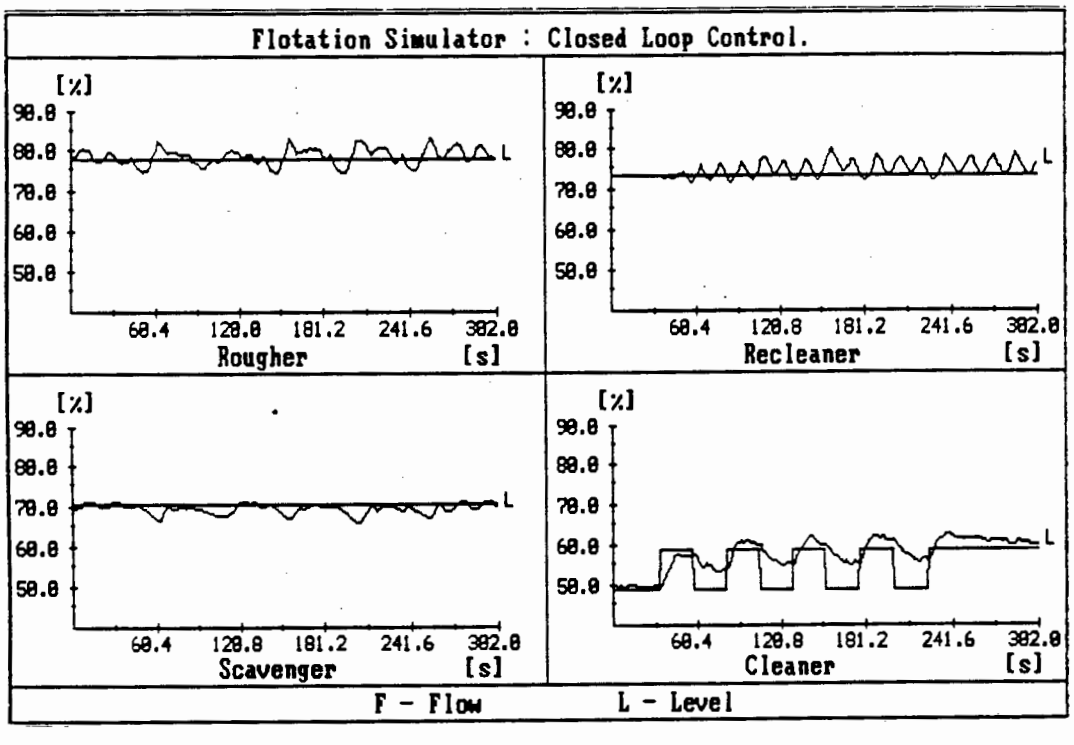


Figure 5.24 Flotation Plant Simulator: Cleaner disturbed ±10%

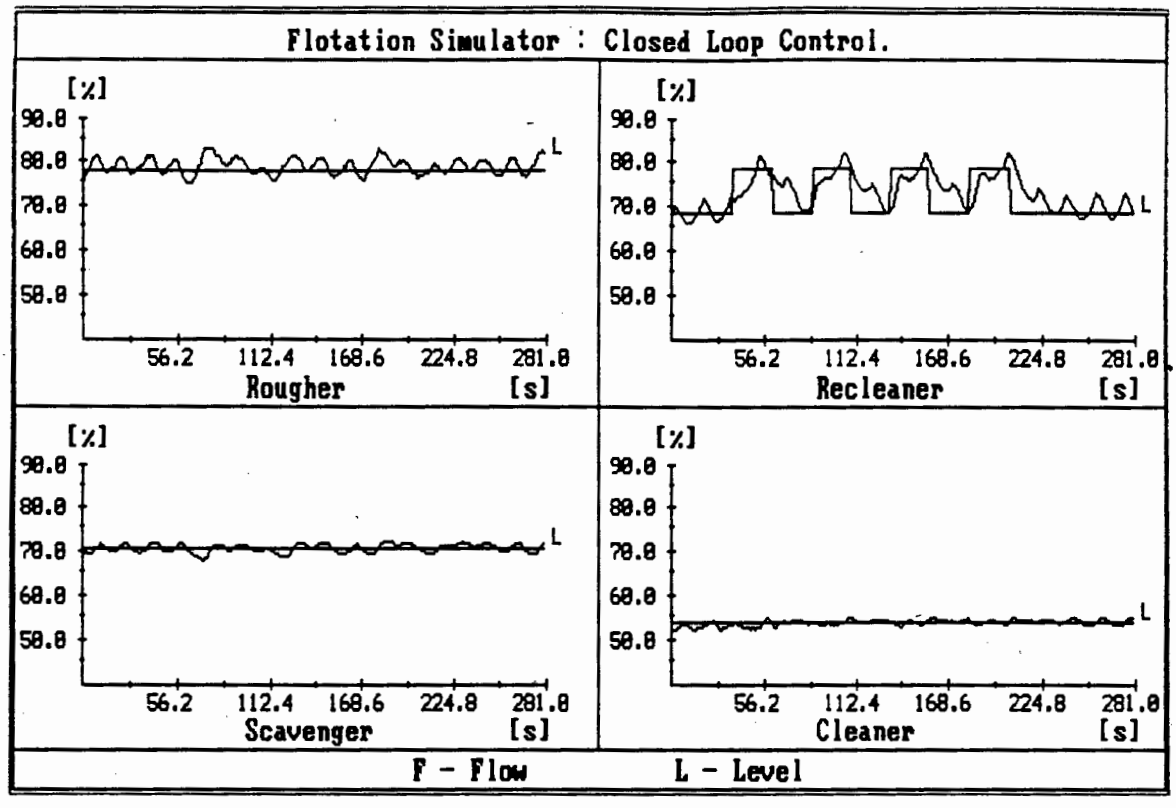


Figure 5.25 Flotation Plant Simulator: Recleaner disturbed ±10%

The system remains stable for all of the setpoint disturbance conditions described in figures 5.22 to 5.25. Also, the system responses to the setpoint disturbances do not reveal any new system artifacts.

Rougher Disturbance : Figure 5.22 reveals that the Scavenger's response to the Rougher setpoint disturbance is the most significant (apart from the Rougher itself) when compared to the remaining cells. This response is expected if one considers the system response to a single setpoint step on the Rougher, shown in figure 5.18.

The remaining cells (Cleaner and Recleaner) are not affected significantly by the Rougher setpoint disturbance. The only apparent interaction on the remaining cells (figure 5.22) is on the Recleaner, where the sump pump that feeds the Recleaner has increased the amplitude of its "surge-cavitate" cycle (as described in appendix F, section F.1). The increase in the

amplitude may be due to the slight variation in feed rate from the Cleaner during the Rougher setpoint disturbance. The efficacy of the sump pumps were sensitive to variations in feed rates to the sumps.

Scavenger Disturbance : Figure 5.23 reveals that the remainder of the system is not affected significantly by the Scavenger setpoint disturbance. This response is expected if one considers the system response to a single setpoint step on the Scavenger, shown in figure 5.19.

Cleaner Disturbance : The system response to a setpoint disturbance on the Cleaner, shown in figure 5.24, is as expected if one uses the experience gathered from the single setpoint step on the Cleaner (figure 5.20) and the system's response to previous setpoint disturbances : The interaction visible on the Rougher and Scavenger is as expected from the system response to a single setpoint step on the Cleaner. The interaction present on the Recleaner is the result of the sump pump "surge-cavitate" cycle amplitude increasing due to the varying feed rate from the Cleaner. (This sump pump characteristic was mentioned previously).

Recleaner Disturbance : The system response to a setpoint disturbance on the Recleaner, shown in figure 5.25, does not reveal any new artifacts when compared to the system response to a single setpoint step on the Recleaner (figure 5.21). There is a small amount of interaction on the Rougher and Scavenger but the magnitude is insignificant when compared to the noise provided by the sump pump "surge-cavitate" cycle (appendix F, section F.1).

The following figures show the response of the system to failures in the loops. Feedback (transducer) failures were induced by actually disconnecting the level-sensor output from the computer interface circuitry. Similarly, the computer interface outputs to the actuators were disconnected to simulate actuator failures.

Only two responses from each type of failure have been presented here as all the responses to the "failures" are similar. Comments made about the shown responses can be applied to the failure responses not shown.

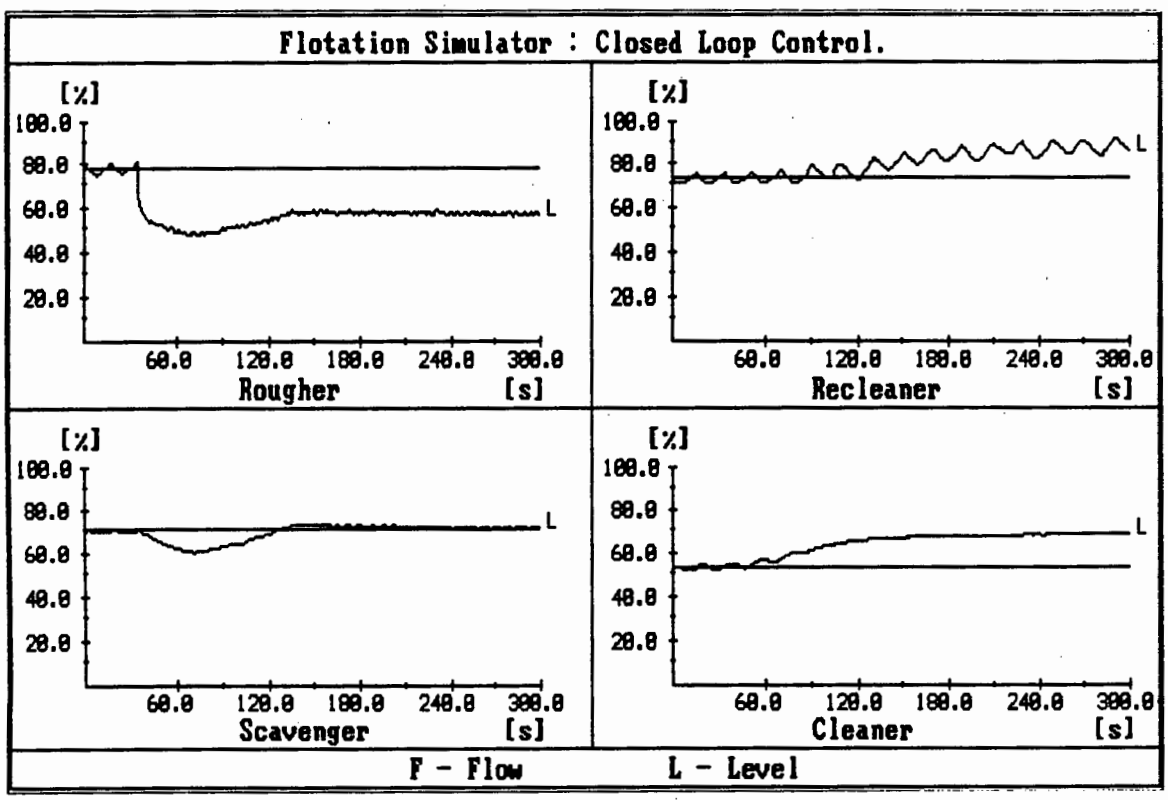


Figure 5.26 System response : Rougher level sensor failure.

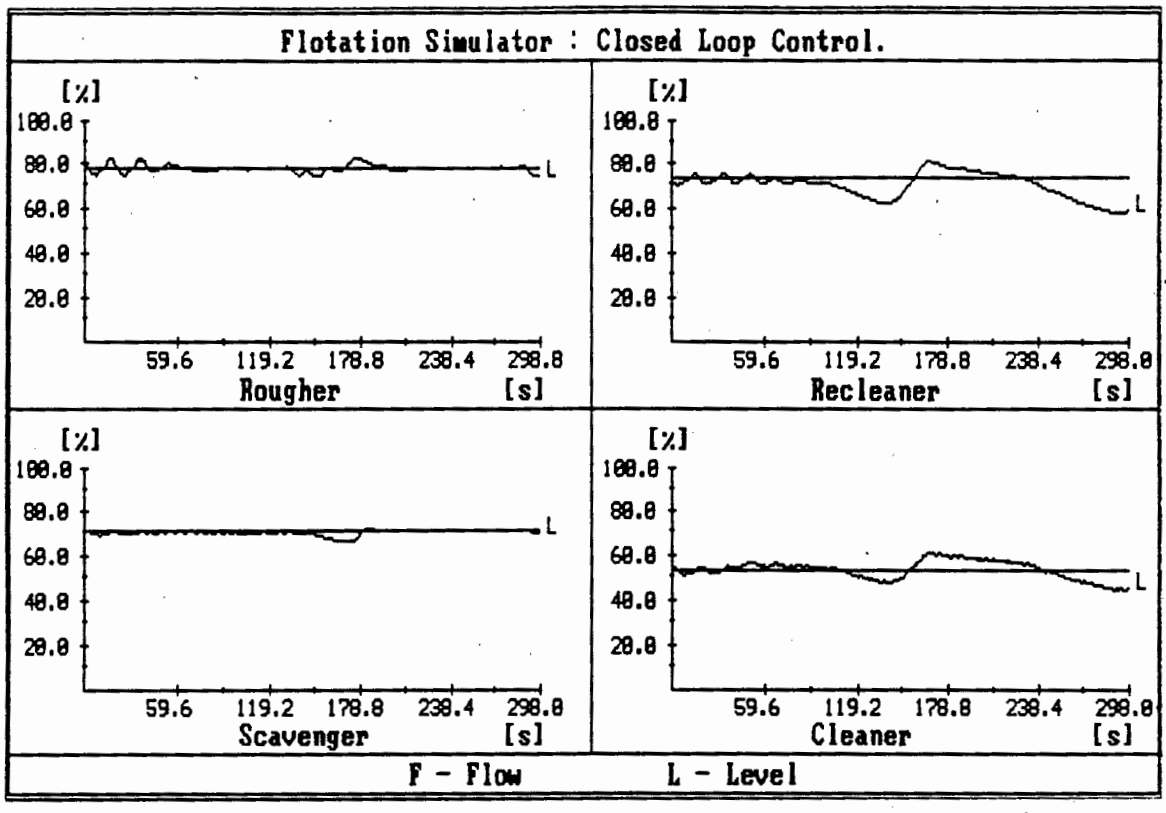


Figure 5.27 System response : Cleaner level sensor failure.

The levels indicated on the "failed-level-sensor" cell (in figures 5.26 and 5.27) are false. The level indicated is due to the computer interface circuitry having a floating (or non-grounded) input which can as a result "follow" one of the other inputs (see figure 5.27, where the cleaner level input follows the recleaner level).

The actual response of the system is to attempt to raise the level of the cell continuously (as shown in figure 5.26). This is due to the error signal being incorrect and the controller containing integrator terms which result in the control signals continuously increasing. The result is that the tank is continually filled to overflowing resulting in an unstable system. This phenomenon is shown in figure 5.26 where the controller tries to raise the "level" of the rougher which results in flooding the rougher and placing so much product in

the circuit that the system is placed beyond controllable bounds.

In the design phase of this controller, the integrity of the system was inspected. The result was that the remaining system would remain stable under single loop failure conditions if the remaining loops had positive gains (section 5.1). This does not appear to be the case in figure 5.26, where the system does not appear to be stable. But, the remainder of the system did remain stable while the product volume in the Flotation Plant Simulator remained within the controllable bounds of the system. In other words, the volume present in the product circuit reached a point where the Flotation Plant Simulator could not control the levels in the individual cells due to their product inputs being too great. Thus, it is possible to say that the theoretical and actual results do correlate as long as actual system constraints are ignored.

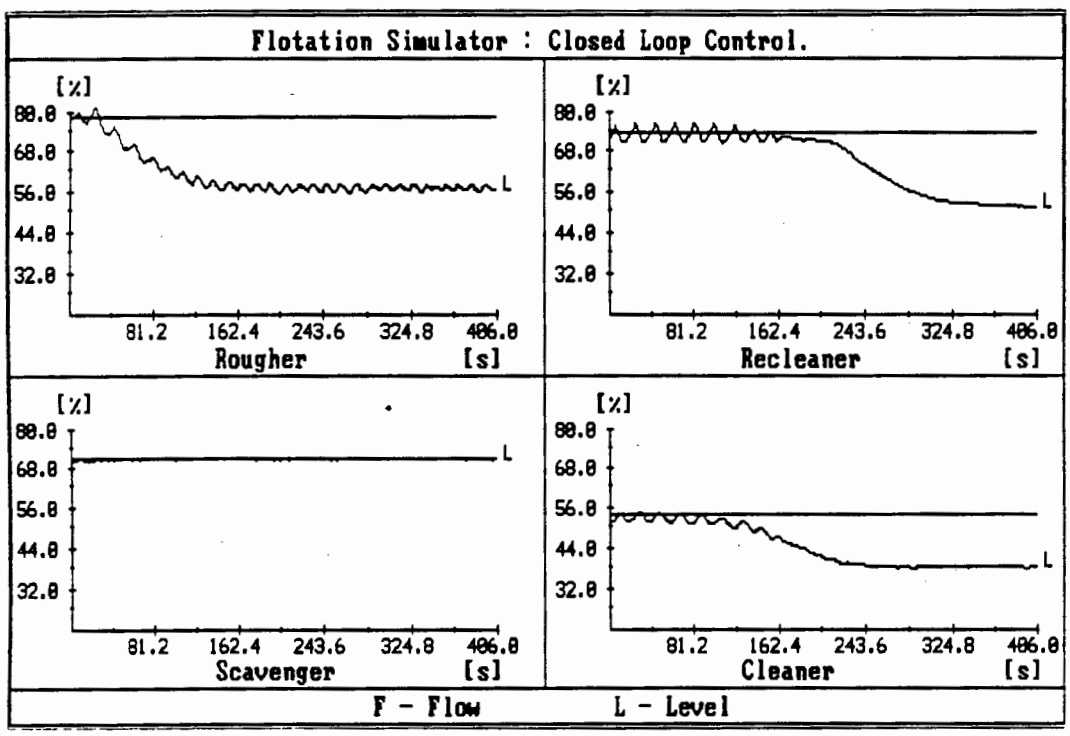


Figure 5.28 System response : Rougher actuator failure.

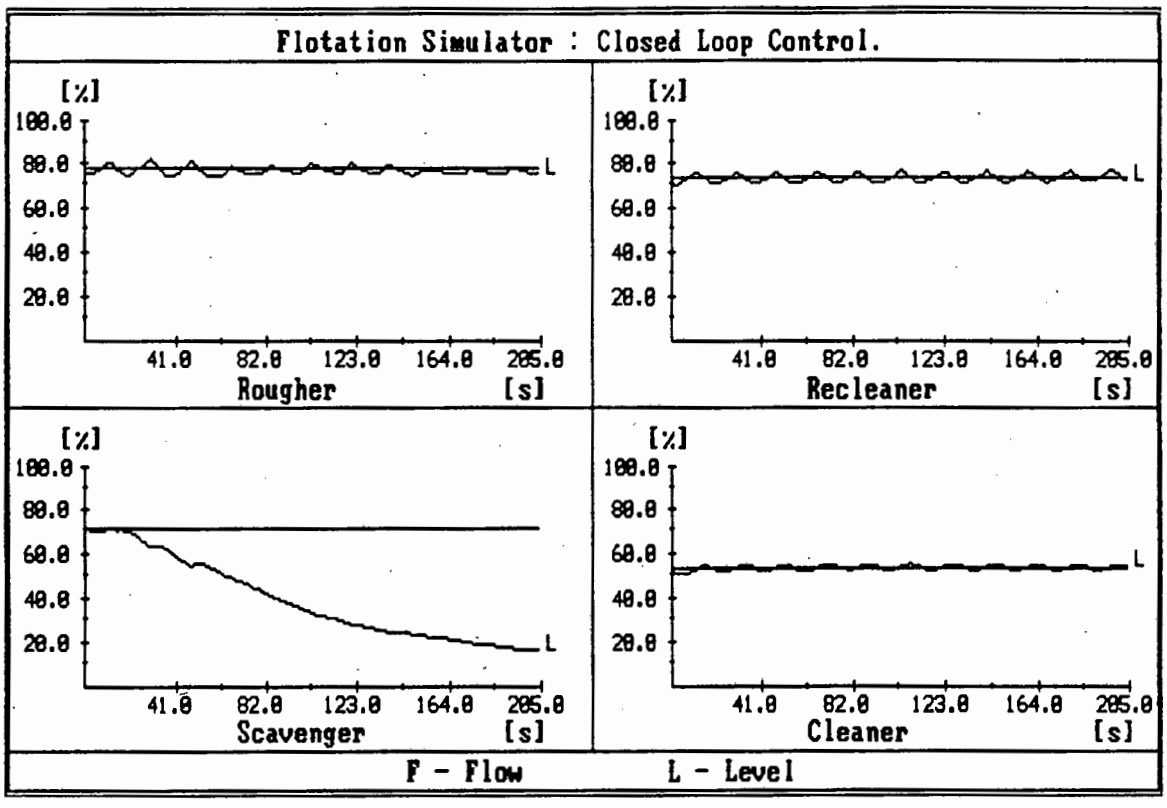


Figure 5.29 System response : Scavenger actuator failure.

The system does remain stable (results shown in figures 5.28 and 5.29) if any actuator fails. This is due to the fact that the valves are normally open when no signal is applied to the actuators. This obviously does not apply if the valve locks or jams in the closed position.

The error on one cell does not appear to affect the control of the other cells to any extent (figure 5.29). The cell's level response is obviously affected when the feed to that cell is cut-off or another cell which feeds that cell cannot provide enough product to maintain the required levels. This phenomenon can be seen in figure 5.28 where the Rougher feed to the Cleaner-Recleaner circuit is reduced. The Cleaner and Recleaner

Chapter 6

Design of Control System for the Flotation Plant Simulator using the Inverse Nyquist Array Technique

This chapter will describe the design procedure followed in order to synthesise multivariable controllers for the Flotation Plant Simulator using the Inverse Nyquist Array (INA) technique. This frequency domain technique has been implemented in the form of CAD [2] system on a personal computer as described in the Introduction.

The frequency domain technique requires a time invariant model of the Flotation Plant Simulator. The model was generated in chapter 2 and the design procedure will be based on this model.

This chapter first describes the design procedure followed when applying the INA design technique to the Flotation Plant Simulator. The design procedure is followed by comments, made by the designer, concerning the technique. The second part of the chapter describes the implementation and testing of the control scheme. Comments concerning the control of the Flotation Plant Simulator concludes this chapter.

6.1 The Closed Loop Model

The general closed-loop model considered by the INA design technique is shown below in figure 6.1.

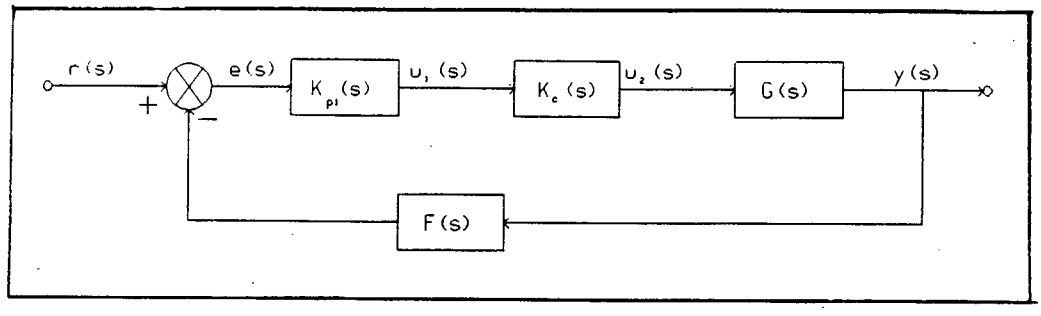


Figure 6.1 Closed-loop model for INA design technique

Where for a *m*-X-*m* system :

- r(s) - reference inputs or setpoint vector (order *m*)
- e(s) - error signal vector (order *m*)
- $u_1(s)$ - input vector to diagonal dominant system (order *m*)
- $u_2(s)$ - input vector to plant or system (order *m*)
- y(s) - system output vector (order *m*)
- $K_c(s)$ - pre-compensator matrix designed to make the system $G(s)K_c(s)$ diagonal dominant (order *m* X *m*)
- $K_{pi}(s)$ - diagonal matrix of single loop PI controllers (order *m* X *m*)
- G(s) - *m* X *m* matrix of plant transfer functions
- F(s) - feedback function matrix, normally an identity matrix *(order m X m)*

6.2 Design of Control Scheme

The INA design technique [2,16,17] specifies the design of a precompensator in order to make system diagonal dominant. Then single loop PI controllers can be designed for setpoint tracking. This procedure is followed in this design section, but the system could not be made completely diagonal dominant before moving into the design of the single loop PI controllers. The reason for this is explained in the appropriate section.

The INA diagram for $G^{-1}(s)$ with row dominance Gershgorin circles is shown in figure 6.2 below (the frequency range is given in the figure). It can be seen that the whole system is interactive, no cell in the Flotation Plant Simulator is dominant.

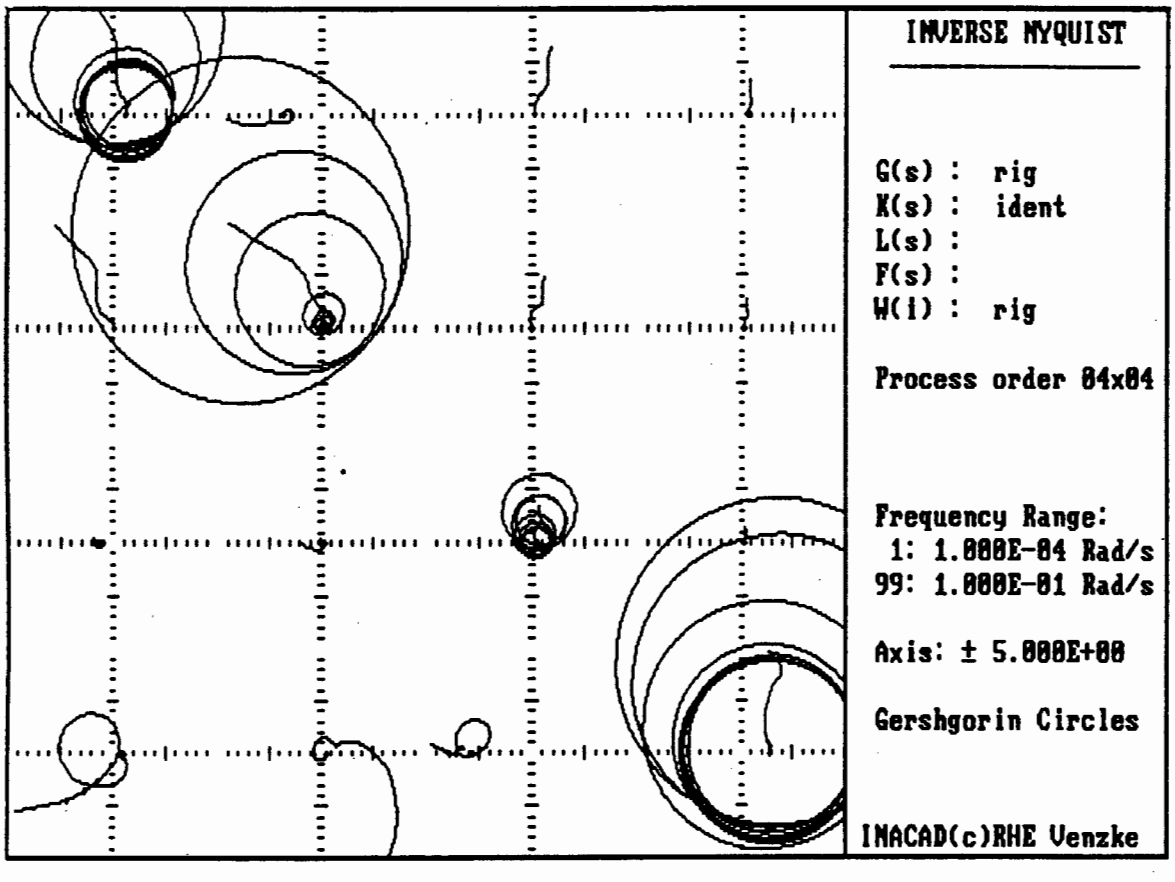


Figure 6.2 INA diagram of $G^{-1}(s)$ with Gershgorin circles.

Interaction within the system is reduced by eliminating or reducing particular elements within the system. The precompensator achieves this elimination by adding multiples of selected rows to other rows within the system.

The order of elimination can effect the performance and the complexity of the final precompensator. The order followed in this design resulted in the simplest precompensator after a number of attempts were made to diagonalise the system.

There is good possibility that another order of elimination would have resulted in a better precompensator. This is due to the fact there are twelve off diagonal elements in this system which results in a large number of permutations when it comes to selecting an order of elimination.

The design phase now proceeds with the design of the precompensator in order to eliminate specific elements in the system in an attempt to diagonalise the system.

The first element to be eliminated is (1,2), by adding a multiple of row 2 to row 1.

$$Row_1 = Row_1 + \alpha(s)Row_2 \qquad (6\text{-}1)$$

The multiple function (marked 'o' in the plot) and its approximation (marked 'x' in the plot) are shown in figure 6.3 overleaf.

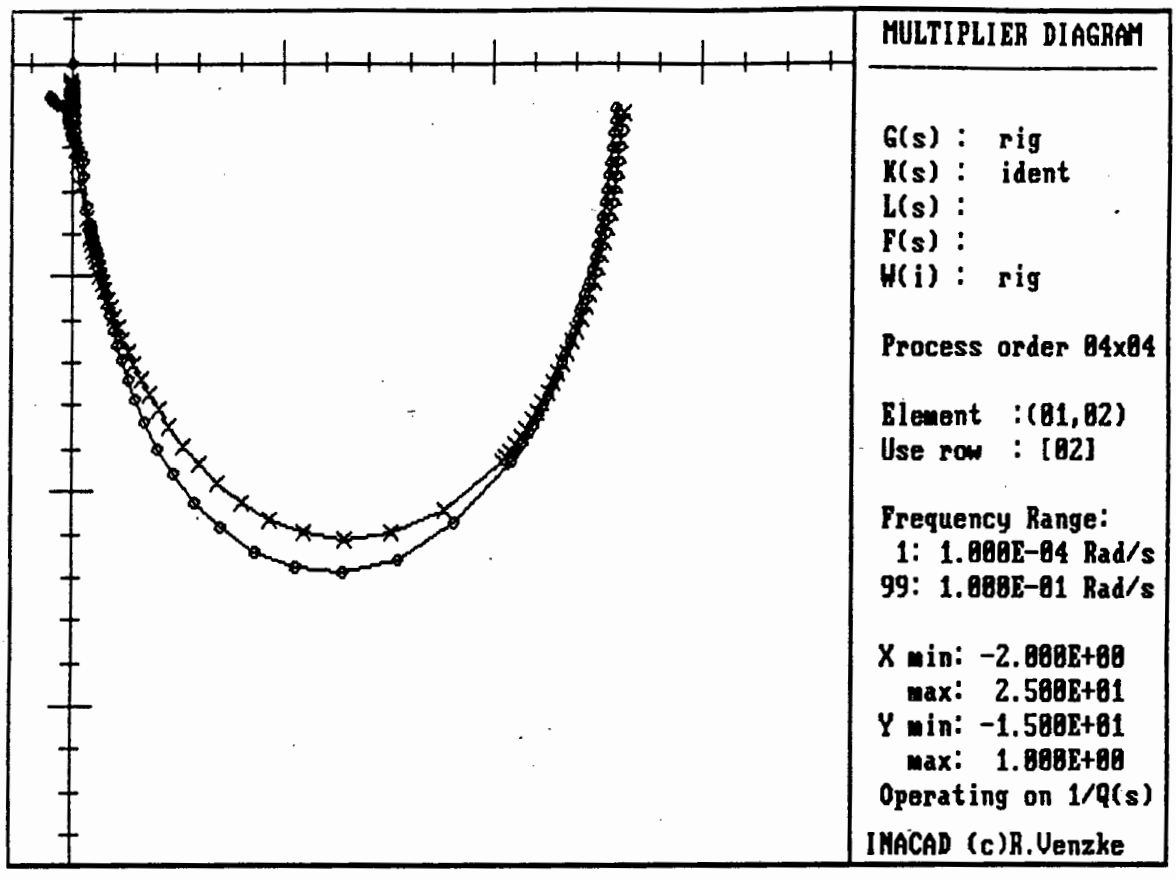


Figure 6.3 Multiple function to eliminate system element (1,2) using row 2.

The multiple function, $\alpha(s)$, is found by initially estimating the function and then refining the coefficients of the function using a non-linear parameter adjustment technique called NELM [2]. After estimating a function and using NELM, the multiple function to eliminate element (1,2) is

$$\alpha(s) = \frac{(-10.43)s + (19.77)}{(585.00)s + (1.112)} \qquad (6\text{-}2)$$

and the corresponding precompensator matrix

$$K_1(s) = \begin{bmatrix} 1 & \dfrac{(10.43)s + (-19.77)}{(585.00)s + (1.112)} & 0 & 0 \\ 0 & 1 & 0 & 0 \\ 0 & 0 & 1 & 0 \\ 0 & 0 & 0 & 1 \end{bmatrix} \qquad (6\text{-}3)$$

The result of this row operation is shown in figure 6.4 where one can see that element (1,2) has been reduced (slightly) at low frequencies.

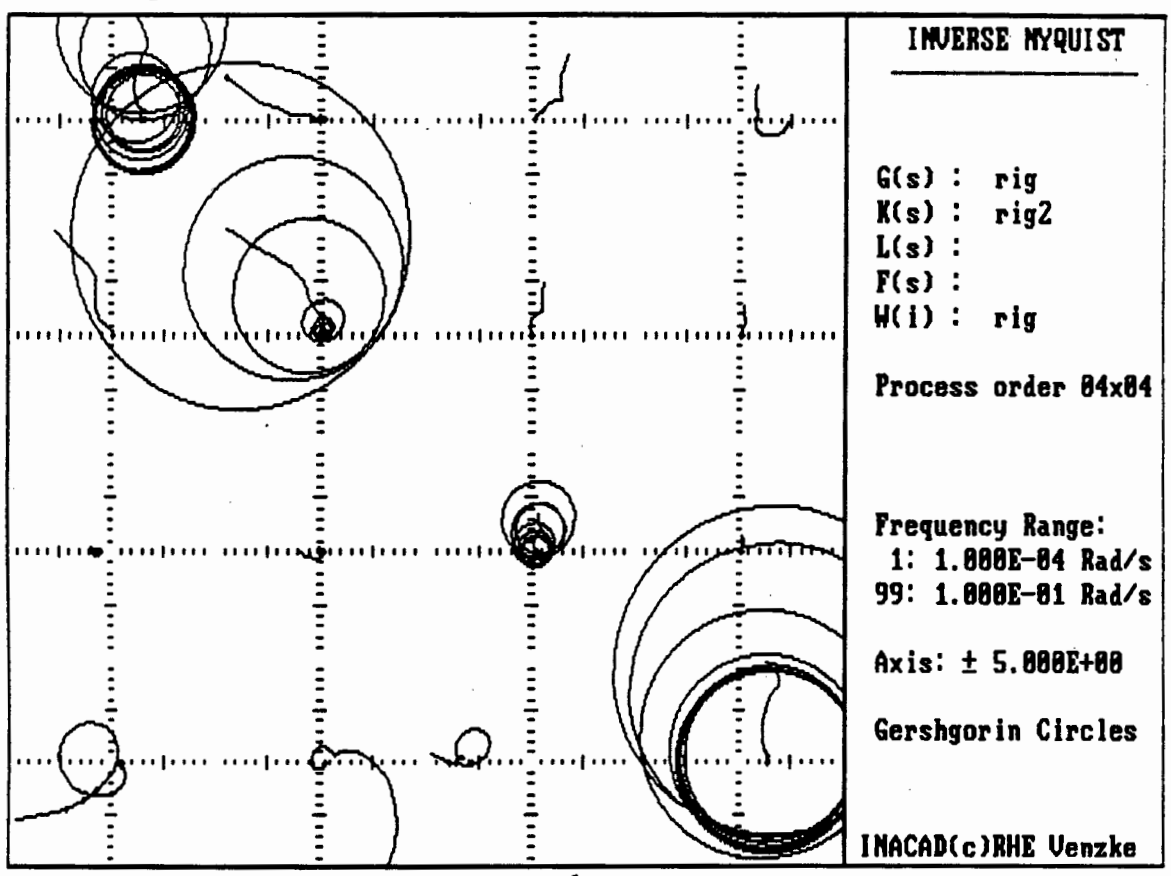


Figure 6.4 INA diagram of $G^{-1}(s)K_1$ with Gershgorin circles.

Now in a similar fashion, the element (2,1) of $G^{-1}(s)K_1$ is eliminated by adding a multiple of row 1 to row 2.

$$Row_2 = Row_2 + \alpha(s)Row_1 \qquad (6\text{-}4)$$

The multiple function (marked 'o') and its approximation (marked 'x') is shown in figure 6.5.

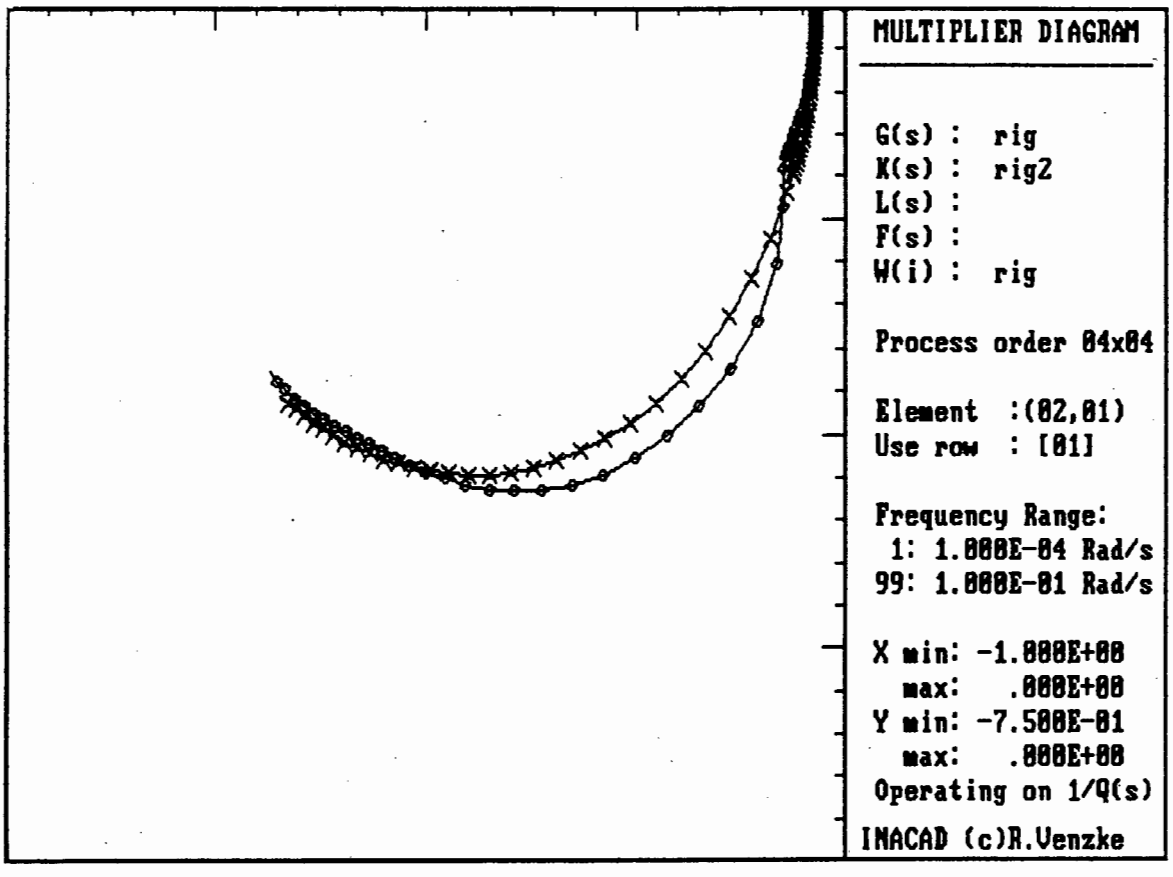


Figure 6.5 Multiple function to eliminate system element (2,1) using row 1.

The multiple function was then estimated and the coefficients optimised as described previously to result in

$$\alpha(s) = -\frac{(72.12)s + (0.153)}{(84.41)s + (4.514)} \qquad (6\text{-}5)$$

and the corresponding precompensator matrix

$$K_2(s) = \begin{bmatrix} 1 & 0 & 0 & 0 \\ \dfrac{(72.12)s + (0.153)}{(84.41)s + (4.514)0} & 1 & 0 & 0 \\ 0 & 0 & 1 & 0 \\ 0 & 0 & 0 & 1 \end{bmatrix} \quad (6\text{-}6)$$

The result of this row operation is shown in figure 6.6 where one can see that element (2,1) has been eliminated and the gershgorin circles on element (2,2) show a marked improvement.

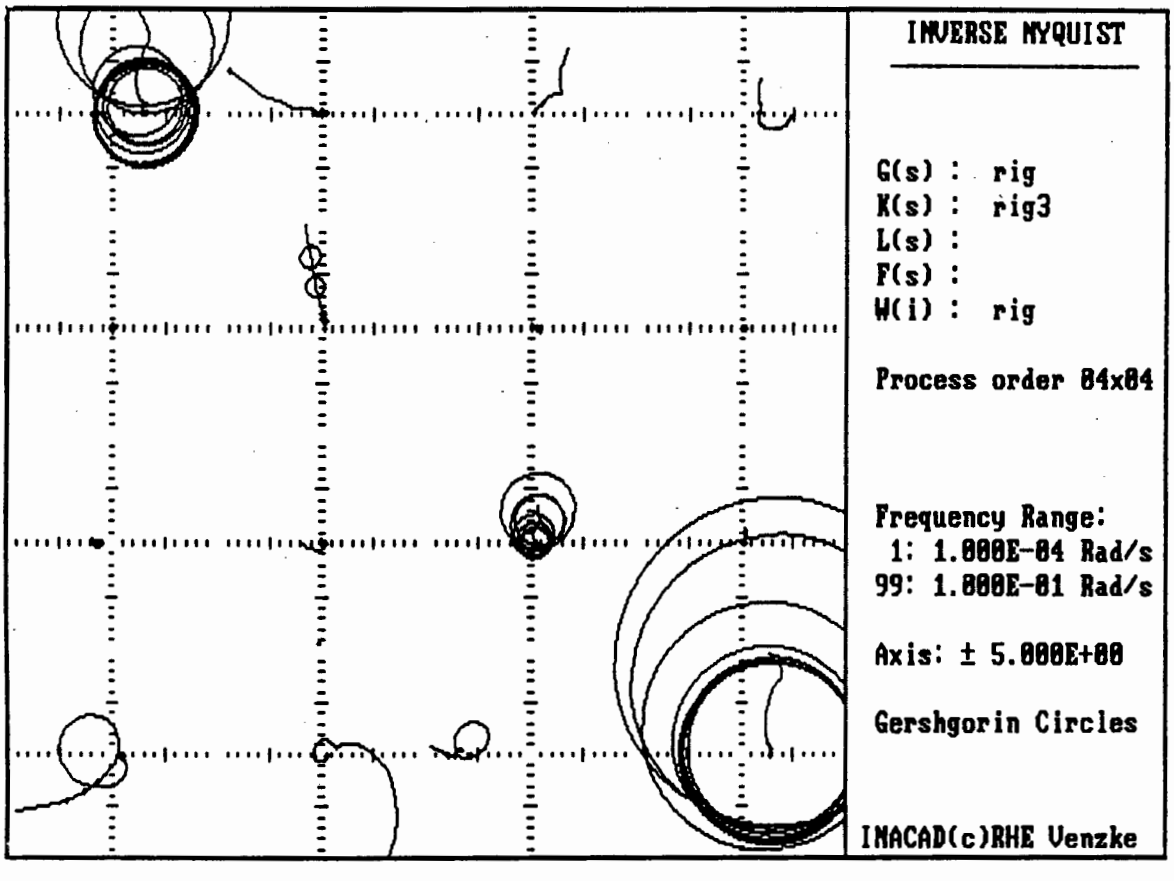


Figure 6.6 INA diagram of $G^{-1}(s)K_1K_2$ with Gershgorin circles.

Similarly, the element (1,4) of $G^{-1}(s)K_1K_2$ is eliminated by adding a multiple of row 3 to row 1.

$$Row_1 = Row_1 + \alpha(s)Row_3 \qquad (6\text{-}7)$$

The multiple function (marked 'o') and its approximation (marked 'x') is shown in figure 6.7.

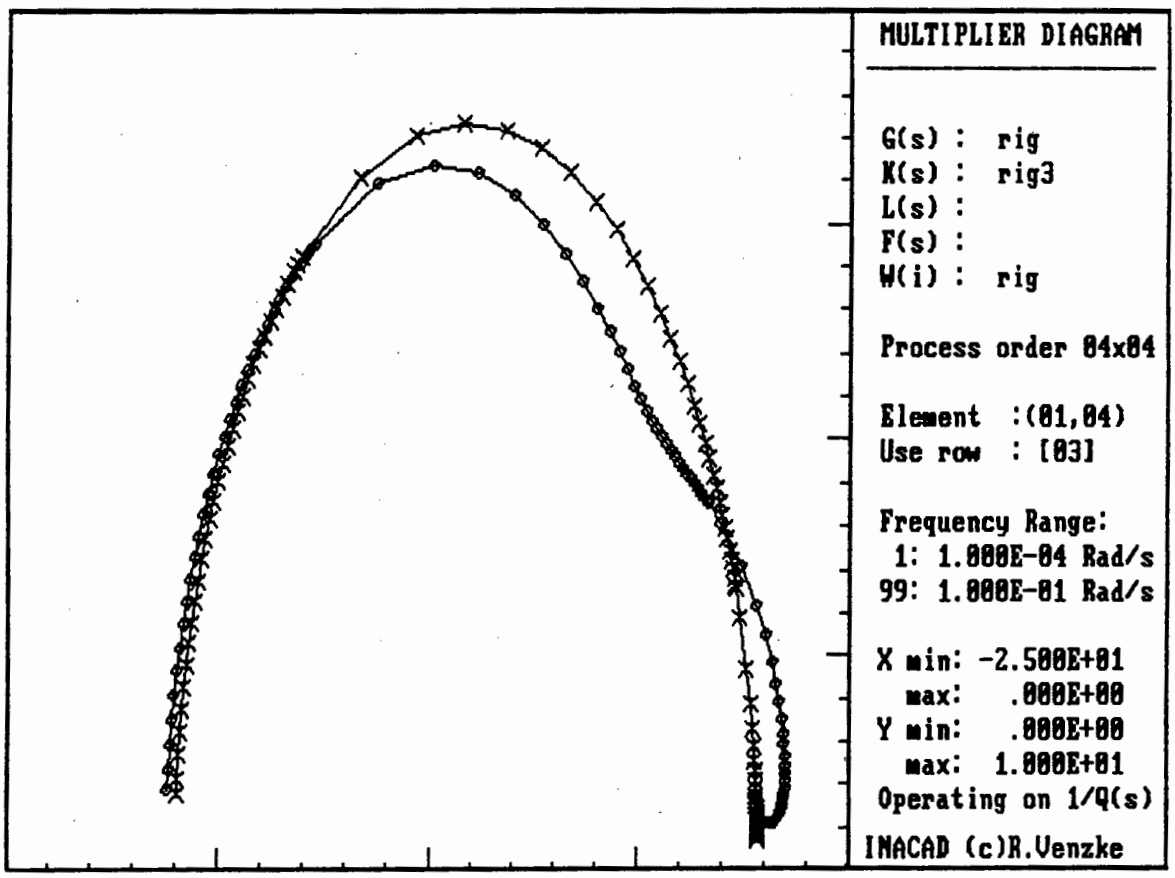


Figure 6.7 Multiple function to eliminate system element (1,4) using row 3.

The multiple function was then estimated and the coefficients optimised as described previously to result in

$$\alpha(s) = -\frac{(52.40)s + (0.767)}{(19.54)s + (0.0383)} \qquad (6\text{-}8)$$

and the corresponding precompensator matrix

$$K_3(s) = \begin{bmatrix} 1 & 0 & \dfrac{(52.40)s + (0.767)}{(19.54)s + (0.0383)} & 0 \\ 0 & 1 & 0 & 0 \\ 0 & 0 & 1 & 0 \\ 0 & 0 & 0 & 1 \end{bmatrix} \quad (6\text{-}9)$$

The result of this row operation is shown in figure 6.8 where one can see that element (1,4) has been eliminated and the gershgorin circles on element (1,1) have improved.

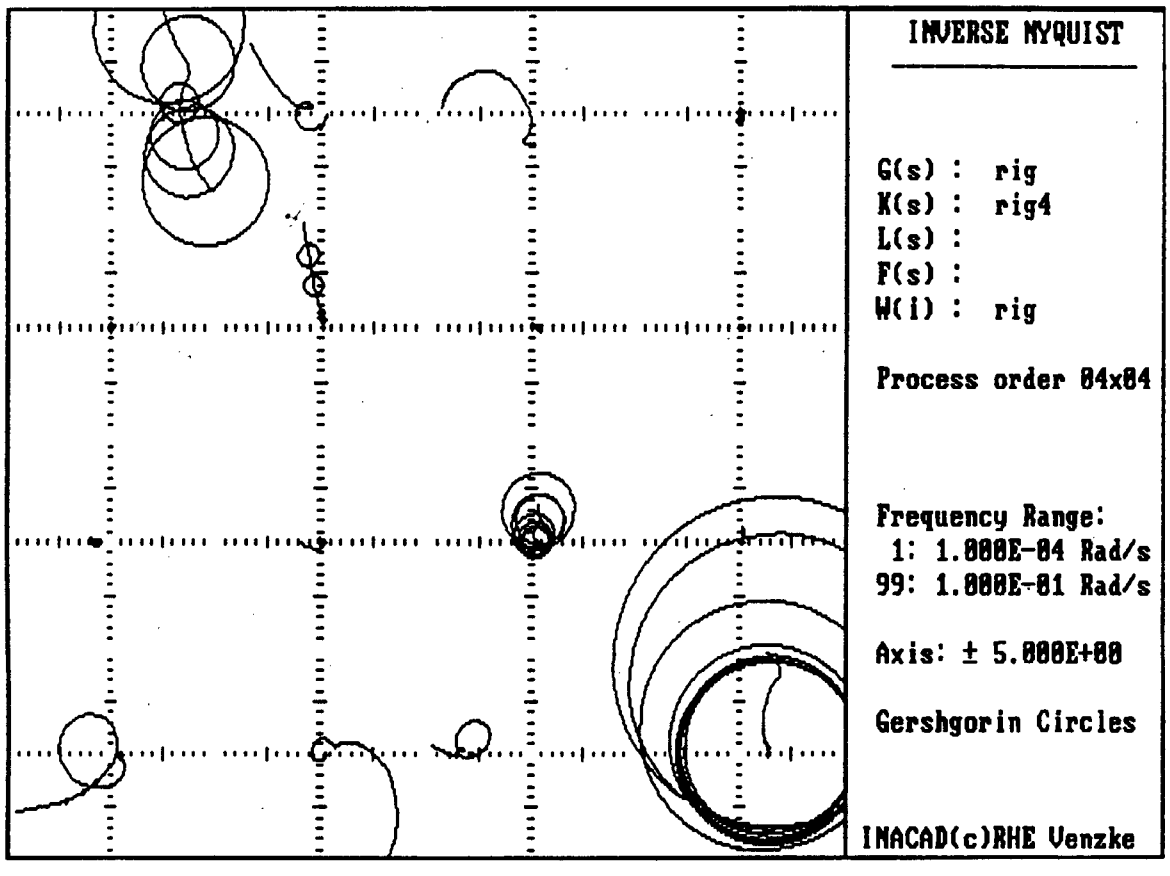


Figure 6.8 INA diagram of $G^{-1}(s)K_1K_2K_3$ with Gershgorin circles.

Similarly, the element (3,4) of $G^{-1}(s)K_1K_2K_3$ is eliminated by adding a multiple of row 4 to row 3.

$$Row_3 = Row_3 + \alpha(s)Row_4 \qquad (6\text{-}10)$$

The multiple function (marked 'o') and its approximation (marked 'x') is shown in figure 6.9.

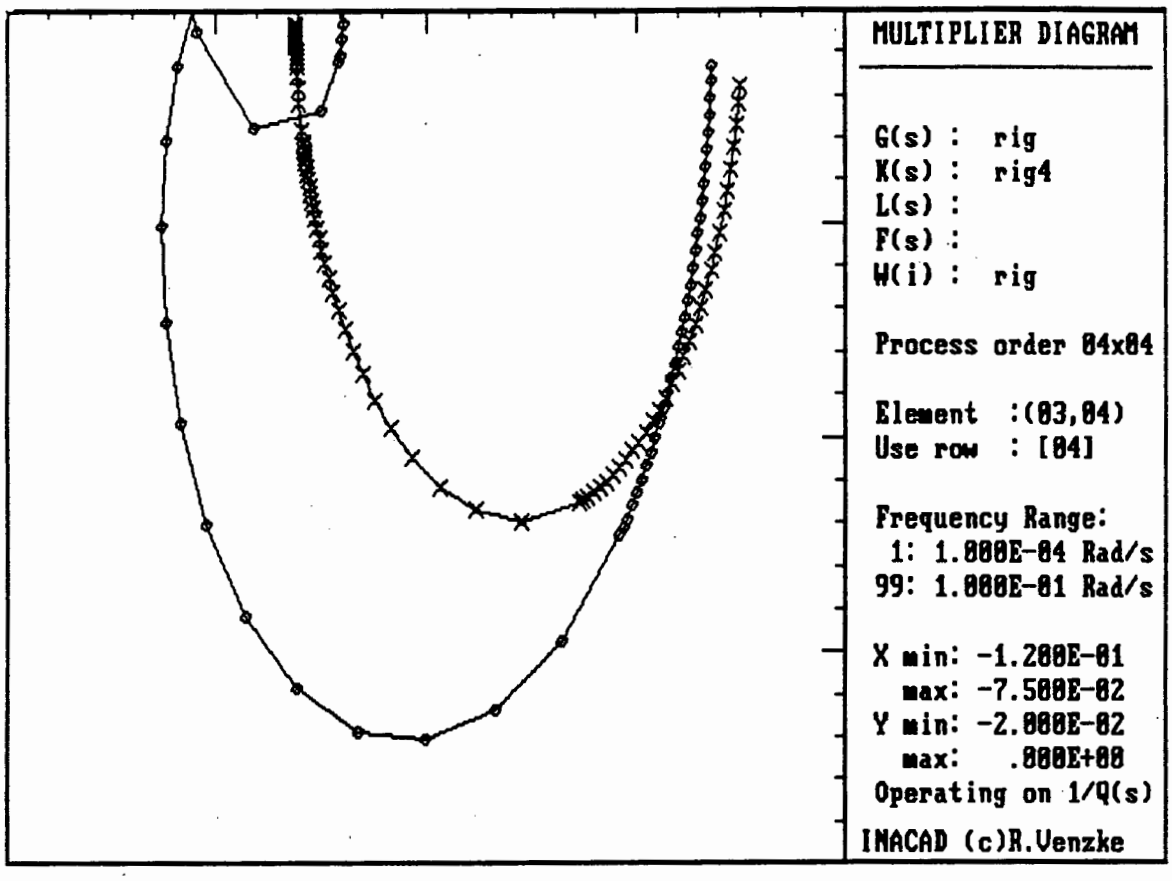


Figure 6.9 Multiple function to eliminate system element (3,4) using row 4.

The multiple function was then estimated and the coefficients optimised as described previously to result in

$$\alpha(s) = -\frac{(41.51)s + (0.0426)}{(397.42)s + (0.529)} \qquad (6\text{-}11)$$

and the corresponding precompensator matrix

$$K_4(s) = \begin{bmatrix} 1 & 0 & 0 & 0 \\ 0 & 1 & 0 & 0 \\ 0 & 0 & 1 & \dfrac{(41.51)s + (0.0426)}{(397.42)s + (0.529)} \\ 0 & 0 & 0 & 1 \end{bmatrix} \quad (6\text{-}12)$$

The result of this row operation is shown in figure 6.10 where one can see that element (3,4), although small initially has been reduced further and the gershgorin circles on element (3,3) have improved slightly.

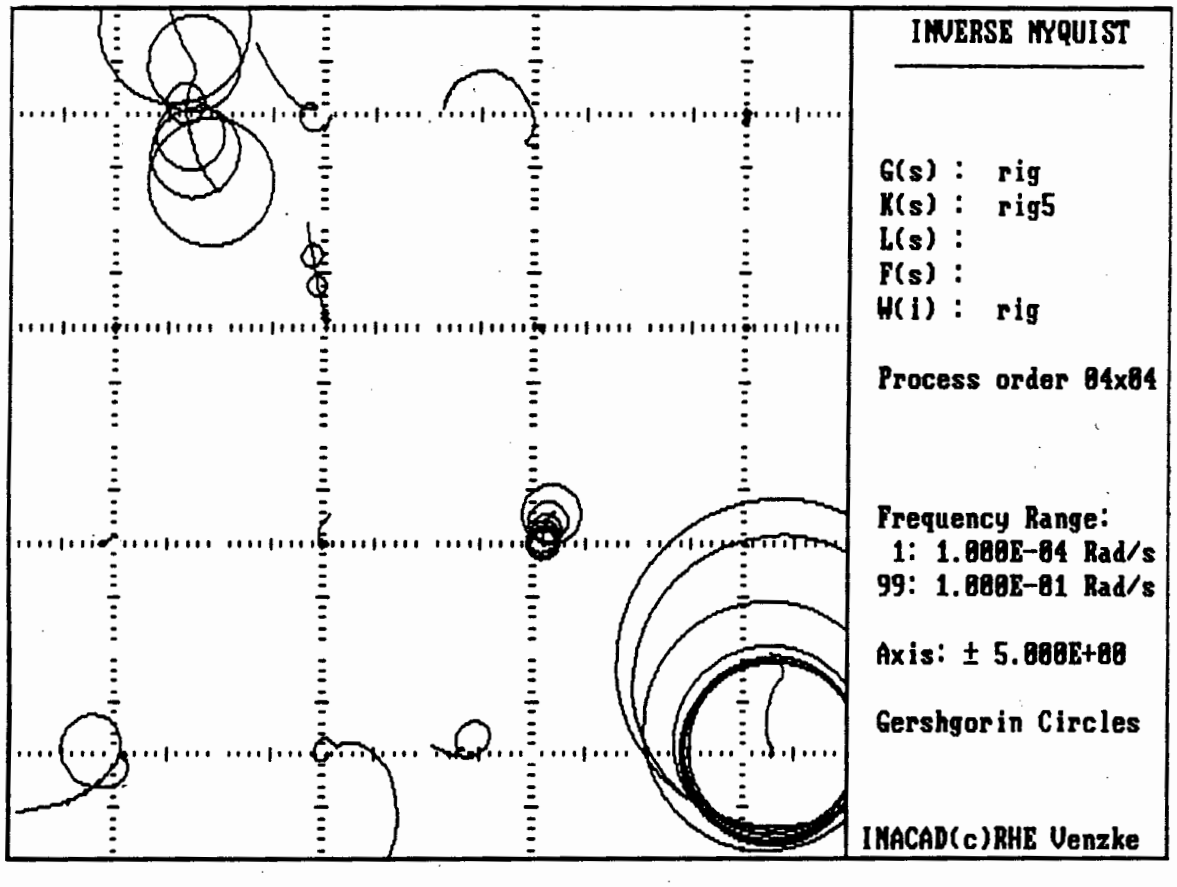


Figure 6.10 INA diagram of $G^{-1}(s)K_1K_2K_3K_4$ with Gershgorin circles.

To compensate or eliminate any other elements in the system require multiplier elements of the shape shown in figure 6.11 below. Multiplier functions required to diagonalise the system from this point nearly always displayed delay characteristics ie. the multiplier function would generate loops in the complex plane as a function of frequency. The elements whose elimination requires a "looping" multiplier function would thus be extremely difficult to eliminate and implement.

Since this is merely an investigation into the INA design technique (in comparison to the Characteristic Loci technique) and given the difficulties of proceeding with the design of the precompensator, it was decided to terminate the design of the precompensator at this point. The design procedure will now proceed to the design of the single loop PI controllers.

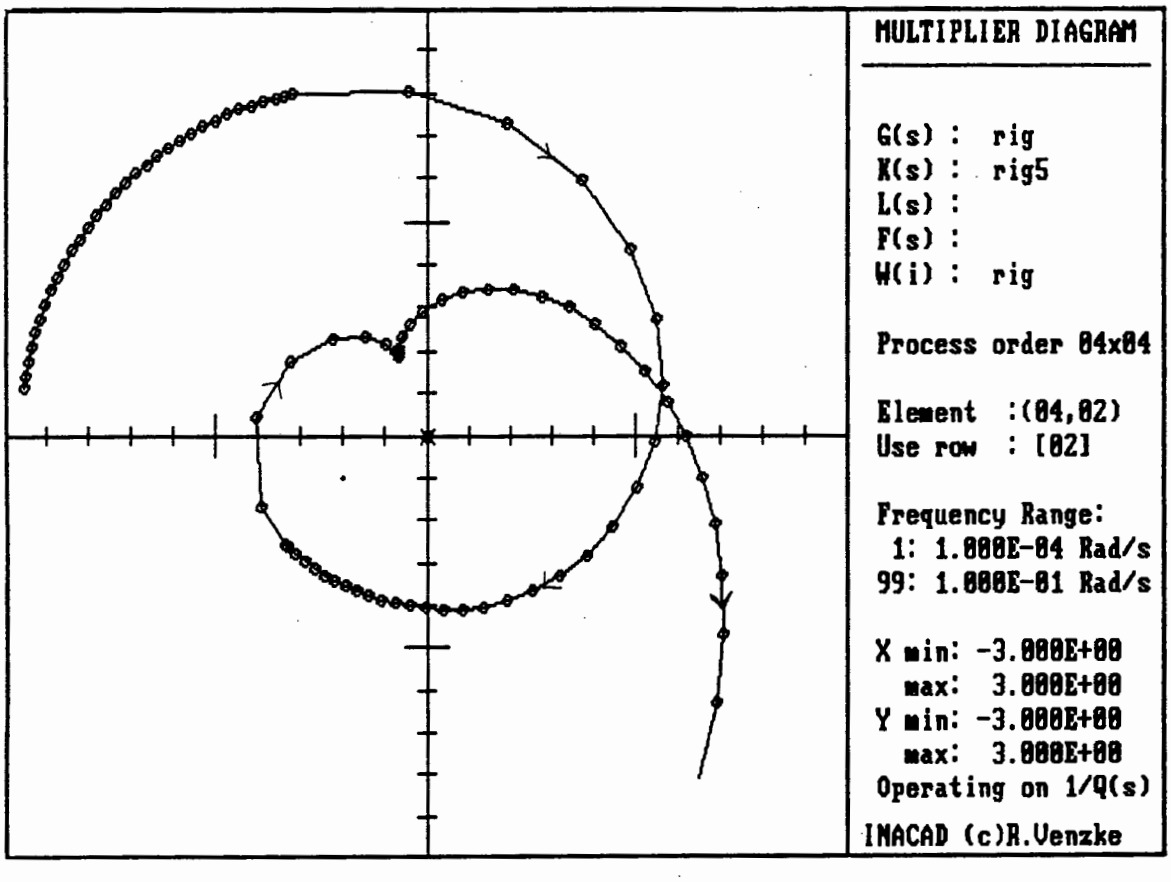


Figure 6.11 Multiple function required to eliminate system element (4,2) using row 2.

Although the precompensator did not diagonalise the system, the design proceeded with the development of single loop PI controllers. In order to assess system stability, the Direct Nyquist Array (DNA) was utilised.

The DNA specified very conservative values for the gains (ie. in the order of 10^{-5}) in order to guarentee system stability. These conservative values resulted in the closed-loop system performance being totally unacceptable. A controller that resulted in the system having an acceptable response was developed using a combination of experience gained on the system and time simulations. The stability of this closed-loop system was assessed using Characteristic Loci which indicated a stable system. The PI controller developed is as follows

$$K_{pi}(s) = \begin{bmatrix} \frac{(15.0)s + 1.5}{s} & & \\ & \frac{(38.0)s + 1.0}{s} & \\ & & \frac{(12.0)s + 0.3}{s} \\ & & \frac{(15.0)s + 2.0}{s} \end{bmatrix} \quad (6\text{-}13)$$

(NOTE: The PI terms are on the diagonal and the zero elements are not shown).

The resulting Direct Nyquist diagram is shown in figure 6.12. The two circles visible on the Nyquist diagrams of the diagonal elements represent the gain (M=1.3) circle, centred on the (-1,0) point and the sensitivity (R=1.0) circle, centred on the origin.

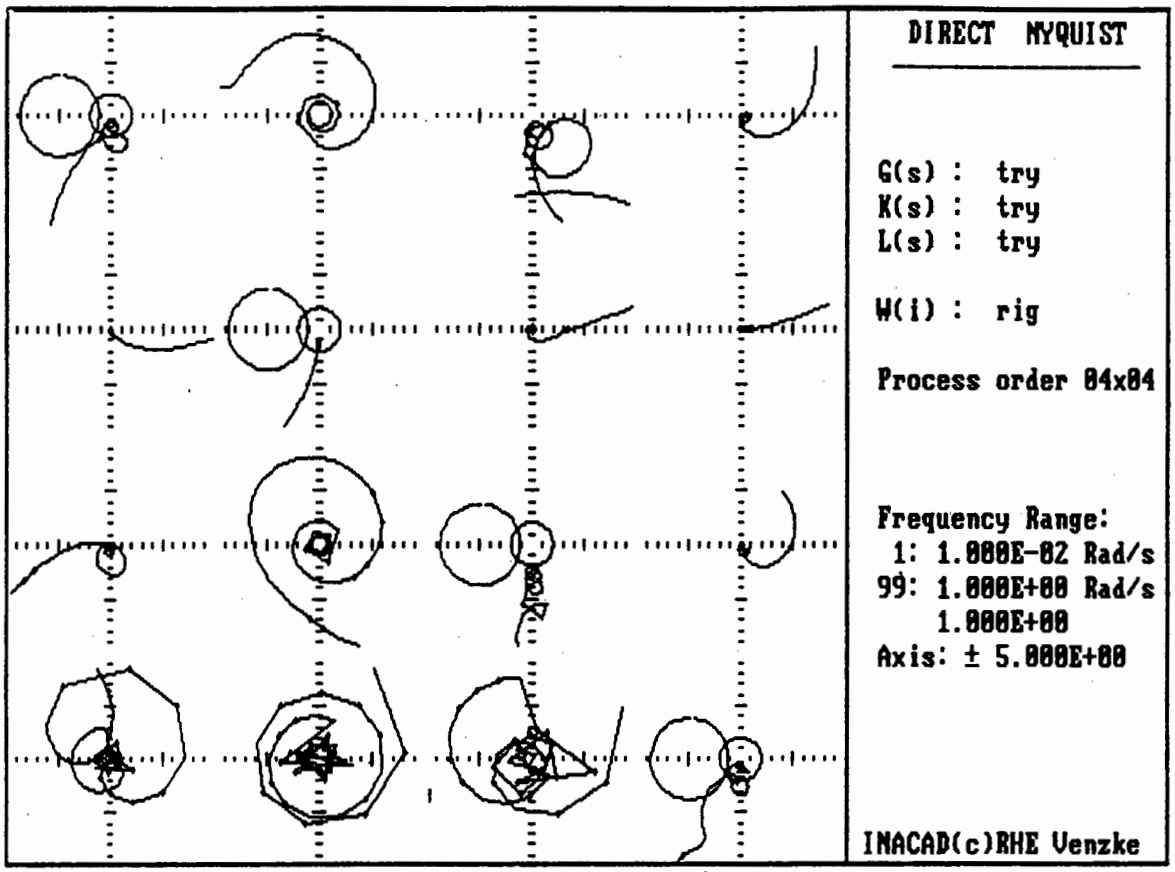


Figure 6.12 Direct Nyquist diagram for the final control system.

The simulated transient responses for step changes to each setpoint are shown in the following figures where it is seen that interaction has been reduced but not been completely eliminated from the system. But the response of the closed-loop system is faster when compared to the open-loop system presented in chapter 2.

As discussed in chapter 5, the time simulations are not completely realistic when compared to the actual system in that

the control values and the values represented by the transfer functions are not bounded as in a real plant situation.

In the following plots, the graphs are numbered (in top-right corner) according to system input number : 1 = Rougher, 2 = Scavenger, 3 = Cleaner, 4 = Recleaner.

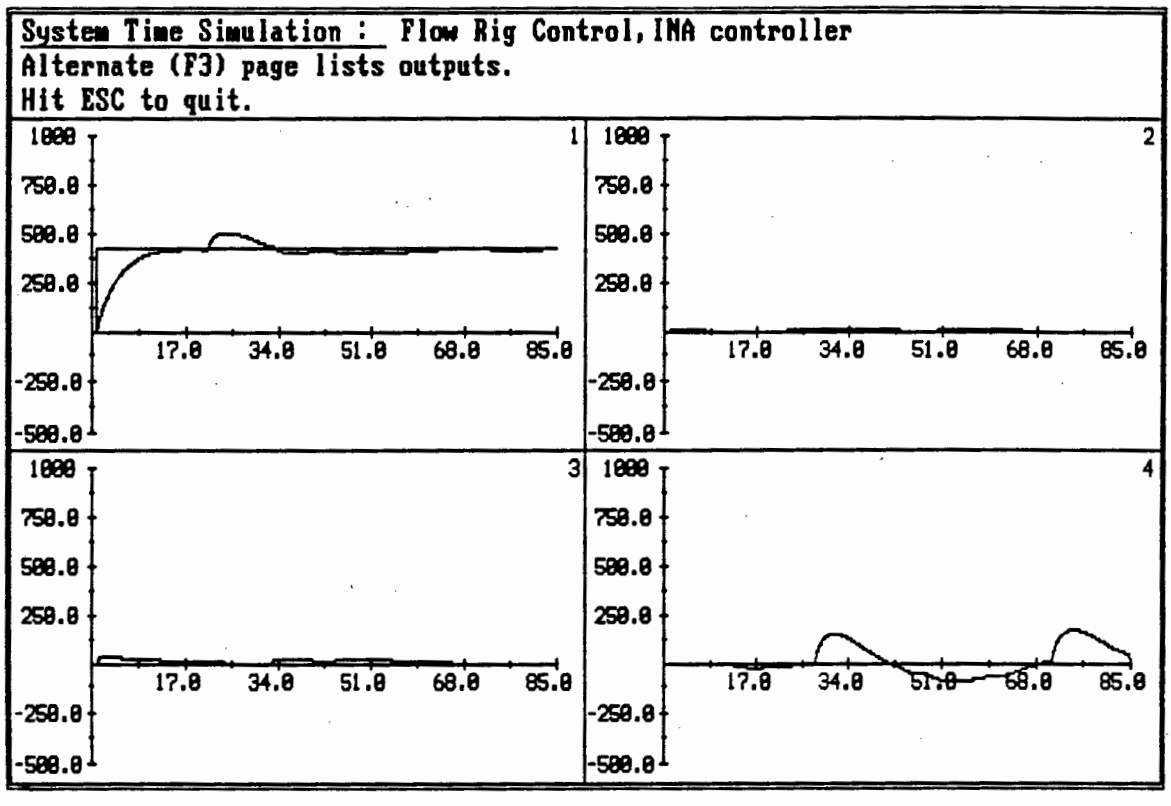


Figure 6.13 Simulated system : Rougher stepped 10%

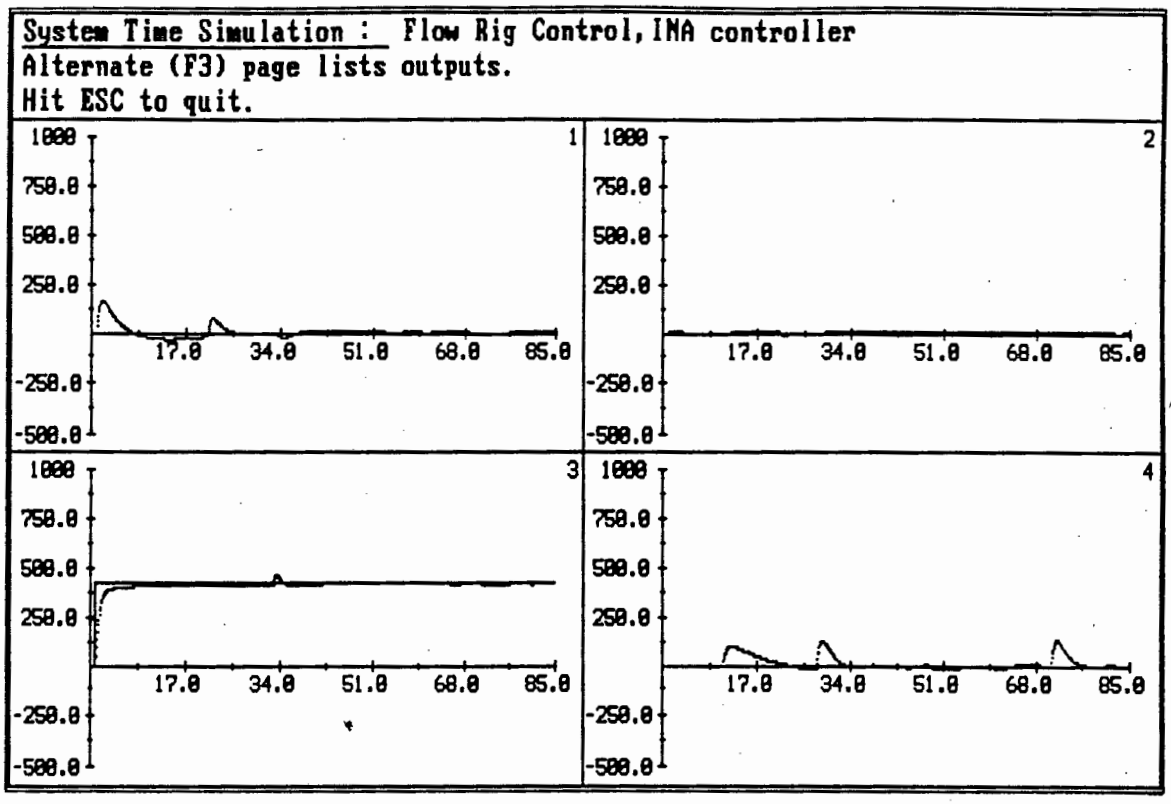


Figure 6.14 Simulated system : Scavenger stepped 10%

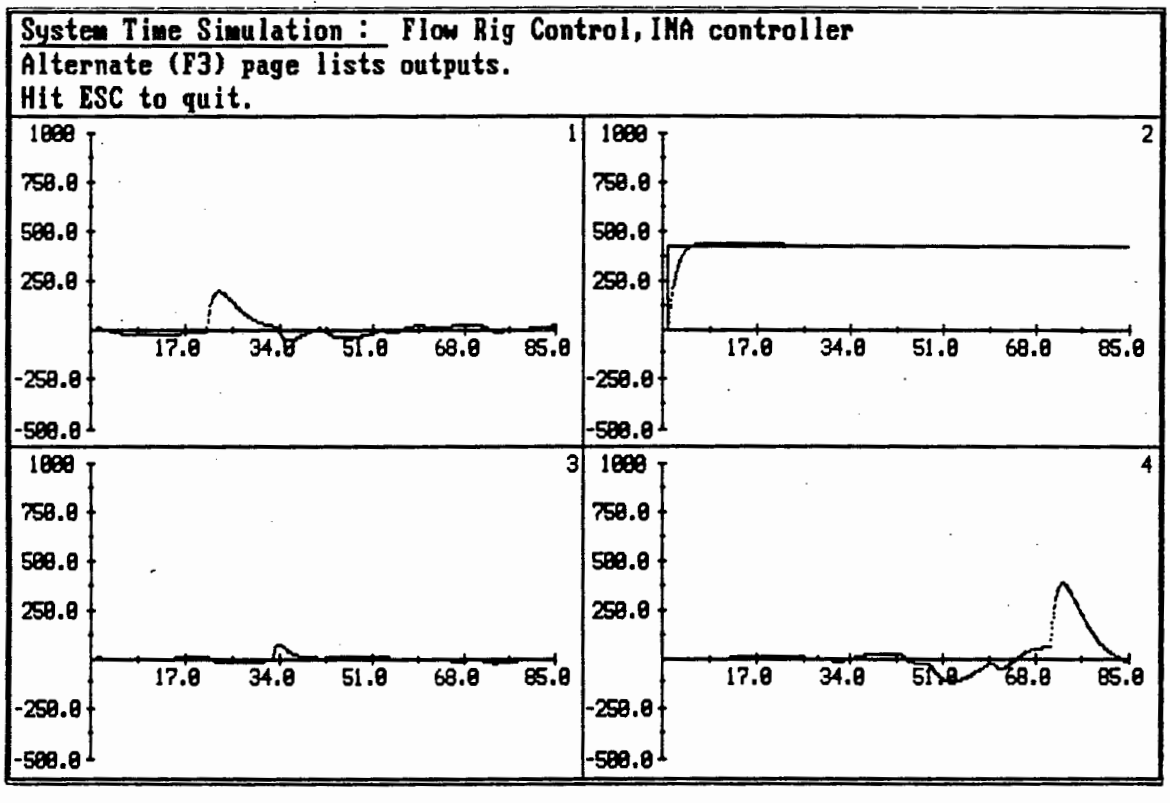


Figure 6.15 Simulated system : Cleaner stepped 10%

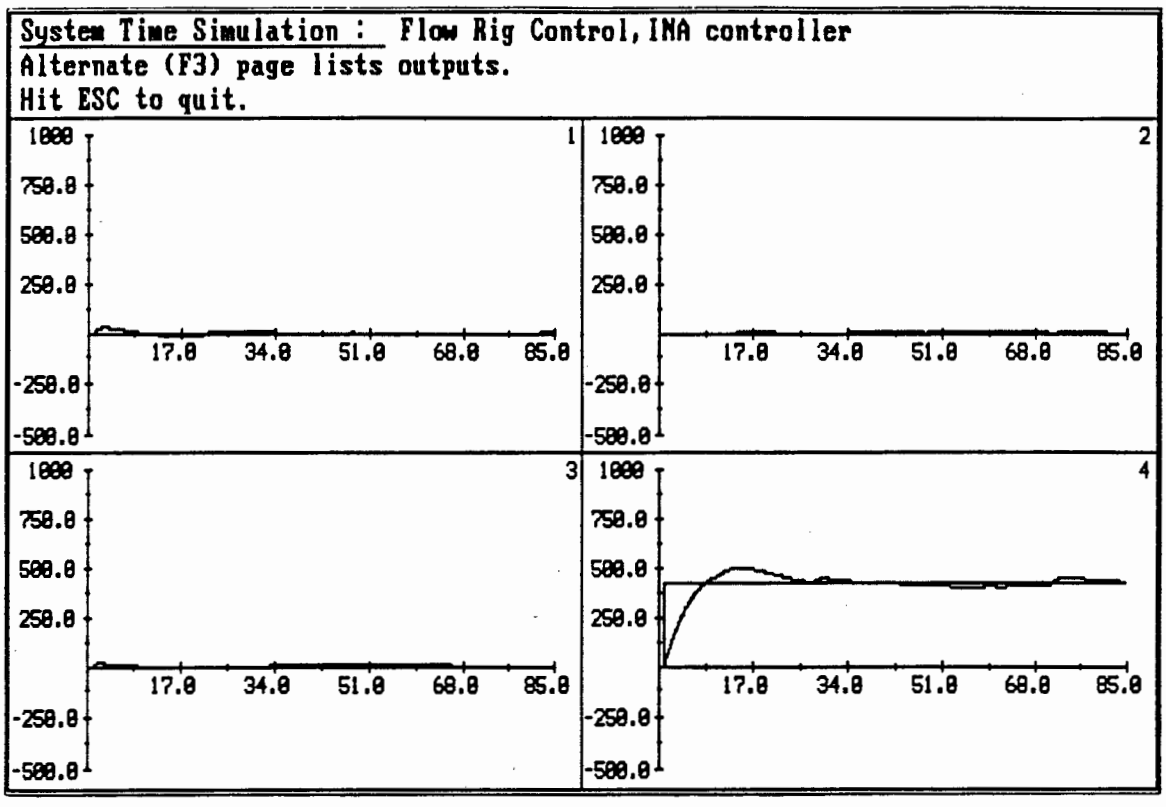


Figure 6.16 Simulated system : Recleaner stepped 10%

Comments on the final control system design and the INA design technique :

i) The INA diagram shown in figure 6.10 reveals that the system is not diagonally dominant. This implies that there will still be interaction, although reduced, in the system which includes the precompensator.

After designing the single loop PI controllers and simulating the closed-loop system (figures 6.13 to 6.16) it was revealed that there is still interaction within the system. But, as expected, the amount of interaction has been reduced when compared to the open-loop system presented in chapter 2.

ii) Since the system is not diagonal dominant (figure 6.10), no assessment can be made about the system stability or integrity.

The fact that the system must be diagonal dominant before system stability and integrity can be assessed is a restrictive pre-condition of the INA design technique.

This pre-condition can result in the designer synthesising an unnecessarily complex precompensator to diagonalise the system so that the system stability can be assessed. The complex precompensator could provide problems at a later stage such as designing the single loop PI controllers or the implementation of the precompensator.

iii) The most powerful feature of the INA technique is the methodical approach to achieving diagonal dominance. The INA technique presents the characteristics of the system in a format that enables the designer to identify the source of problems within a system. The technique then provides the designer with a simple process in order to eliminate or reduce the identified problem.

The technique does rely on the intuition of the designer in the problem identification and elimination phases of the design. But this should be seen as an advantage, not a disadvantage, since the designer will probably be able to visualise how the system is being manipulated and be able to judge or "feel" if the results make sense.

iv) The most significant difference between the Characteristic Loci technique and the INA technique, from a designers point of view, is that the Characteristic Loci technique has powerful analytical abilities while the power of the INA technique lies in its methodical approach to diagonal dominance.

6.3 Implementation of Control Scheme

The control scheme developed in section 6.2 is to be implemented on a personal computer which is interfaced to the Flotation Plant Simulator. The precompensator, $K_c(s)$, and the PI controller matrix, $K_{pi}(s)$, had to be converted into the z-domain to form $K_c(z)$ and $K_{pi}(z)$ respectively. The controller terms were transformed from the s-domain to the z-domain using tables and a sampling/control interval of one second.

Once implemented, the controller presented the problem of integral "wind-up", as in the implementation of the controller in chapter 5. An example of the effects of integral wind-up is shown in figure 6.17 below (The solid line in each graph represents the setpoint setting of each cell), where all four setpoints are stepped simultaneously.

All the levels,except the Scavenger, overshot their respective setpoints. The Scavenger did not reach the required setpoint within 370 seconds, which is due to terms in the precompensator "winding up" and causing the Scavenger actuator to open instead of closing. Under these "wind up" conditions, the control values output to the system saturated for an undesirable length of time.

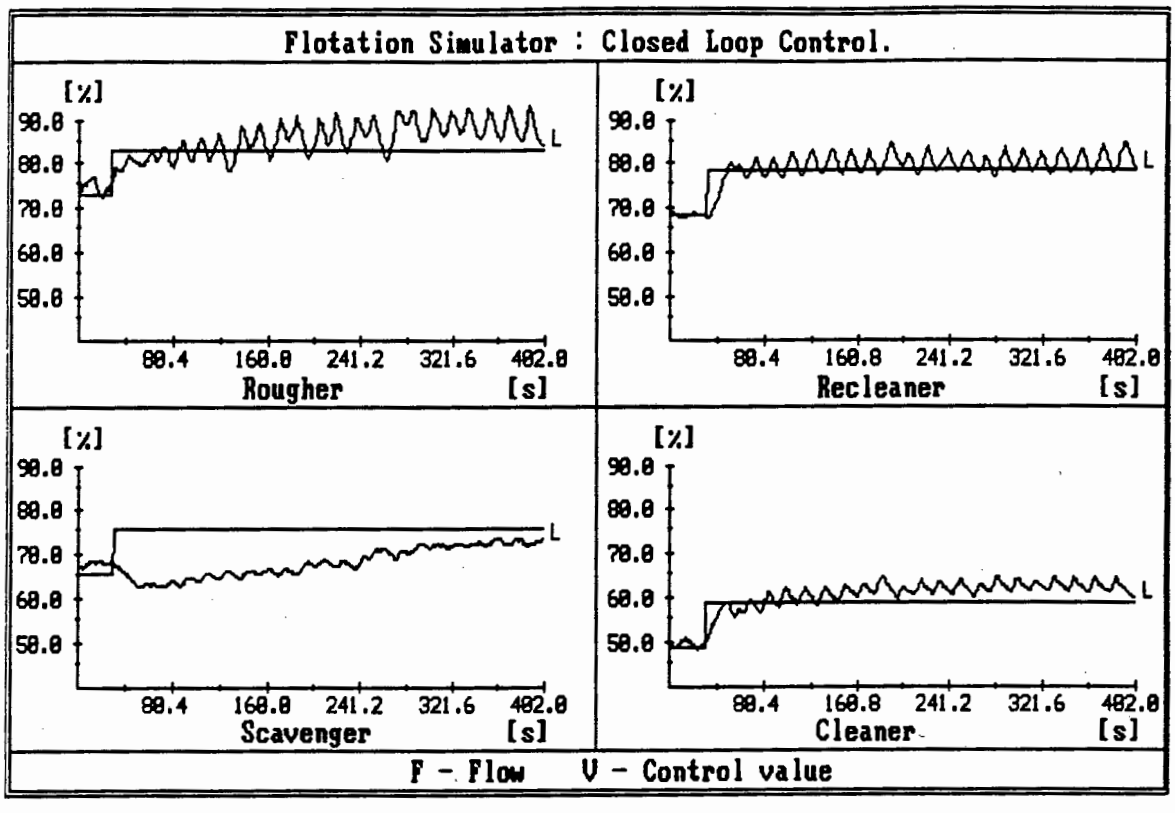


Figure 6.17 Flotation Plant Simulator : Example of "Wind-up".

In order to avoid this problem of integral "wind-up", a rate-algorithm was implemented. A description of the rate algorithm was described in chapter 5, section 5.2 and will not be (barring the results) discussed in this chapter.

The performance of the rate algorithm can be seen in figure 6.18 which is the result of the same step test described in figure 6.17, where all four plant inputs are stepped simultaneously. Figure 6.19 shows the system actuator control signals during the same test.

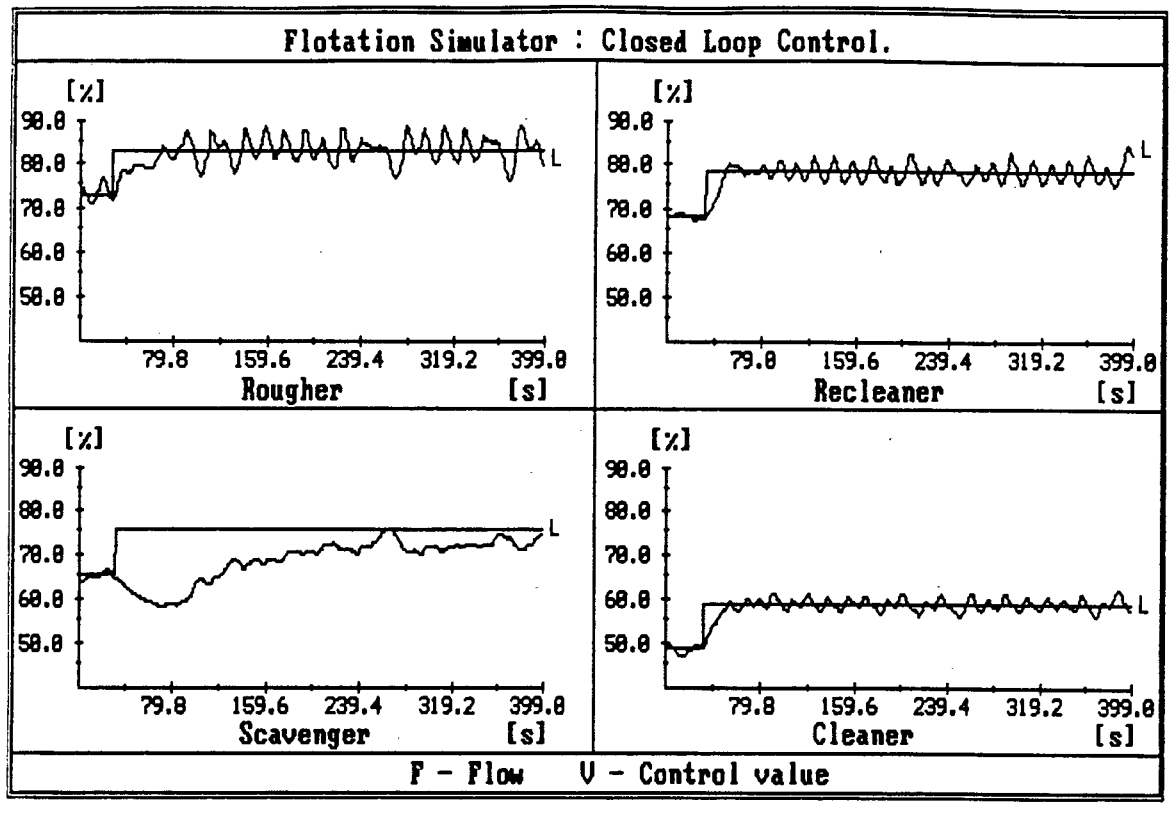


Figure 6.18 Flotation Plant Simulator : Rate algorithm implemented.

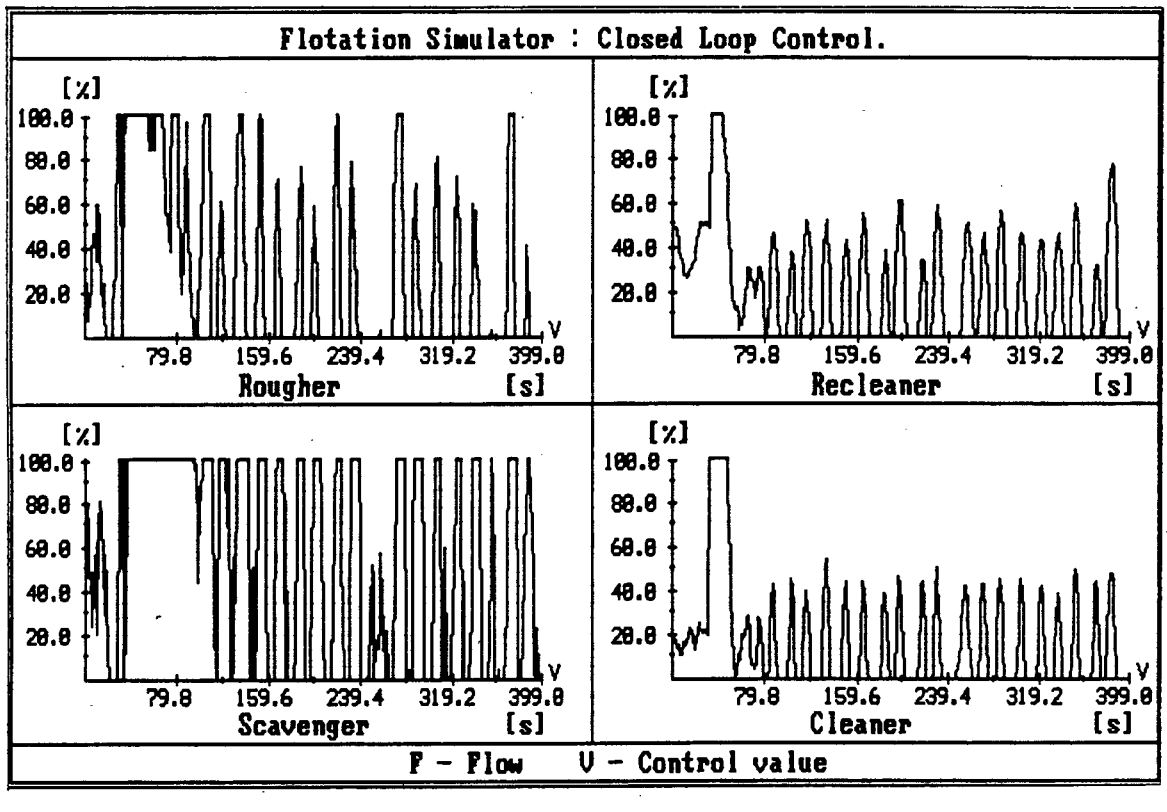


Figure 6.19 Flotation Plant Simulator : Rate algorithm implemented - actuator control values.

The performance of the system appears to have improved since the rate algorithm has been implemented. Figure 6.18 shows that no cell overshot its setpoint. But, the Scavenger still does not reach its setpoint within 370 seconds.

Figure 6.19 shows that the control values output to the Rougher and Scavenger actuators do saturate for a significant length of time. And the Scavenger control value has the undesirable characteristic of changing rapidly from one extreme to the other extreme.

But, one must note that this test of stepping all four setpoints simultaneously does place extreme demands on the system. The system control values do not saturate for any significant length of time under normal operating conditions. Thus, the results listed in the remainder of this chapter do not show the system control values.

The table below lists the approximate response times of the individual cells for this particular test. The two time columns represent the controller with the rate algorithm excluded and then included.

Cell	No Rate Algorithm [Secs]	Rate Algorithm [Secs]
Rougher	> 370	75
Scavenger	> 370	> 370
Cleaner	> 370	38
Recleaner	45	45

Table 6.1 Response times to setpoints stepped simultaneously

Table 6.1 shows that the response times of the system are generally faster, especially the Rougher and the Cleaner. The exception to this characteristic is the Scavenger, where the

response is quite different although the response time is approximately equal to the original controller implementation.

In the system with the rate algorithm, the Scavenger level drops further (than in the system without the rate algorithm), but rises faster after the initial drop in the level. After this point, the two Scavenger responses look alike in that they slowly rise to the setpoint.

In conclusion, the rate algorithm has improved the response times of some of the cells by reducing the amount of setpoint overshoot which was due to integral "wind up".

Unlike the controller in chapter 5, this controller's performance could not be visibly improved by tuning the PI controller terms. Thus, the closed-loop system remains unchanged after the implementation of the rate algorithm.

The transient response of the system to a step change to the setpoints is shown in the following figures. The setpoint is stepped 10% in each test. The solid line in each graph represents the setpoint setting for the level of each cell of the Flotation Plant Simulator.

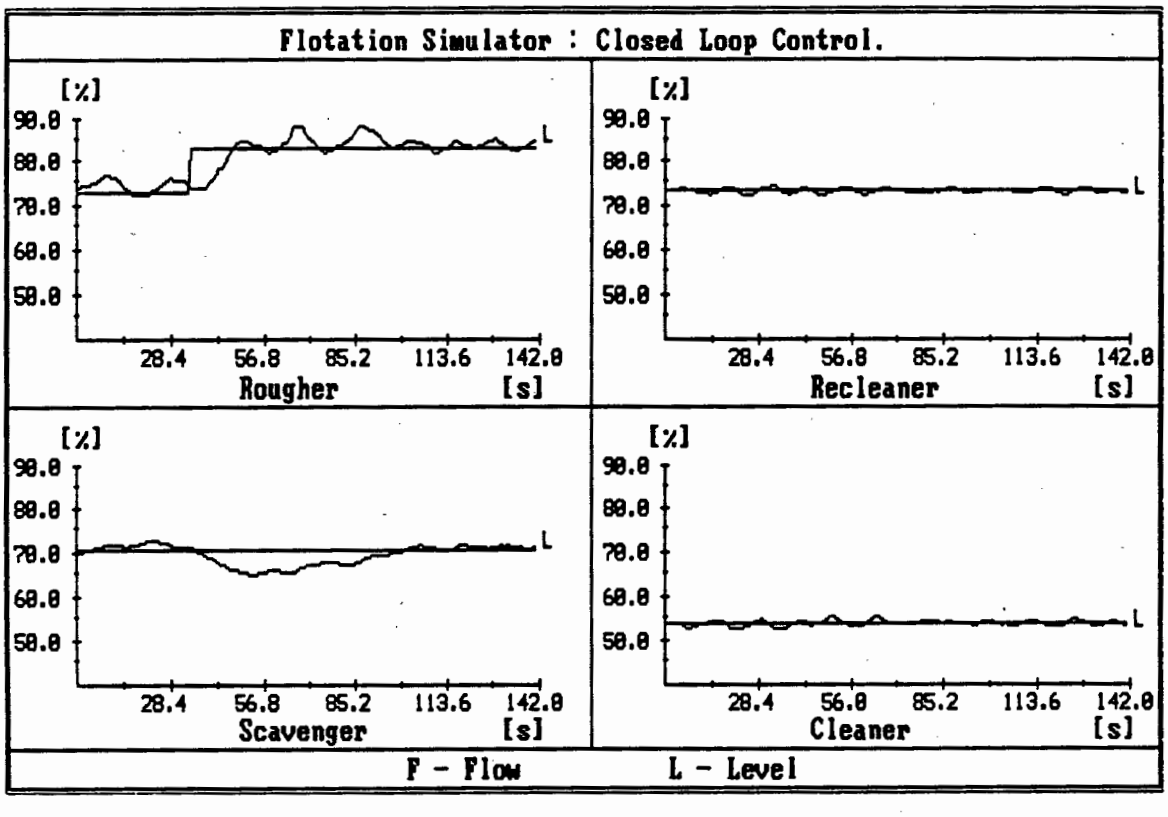


Figure 6.20 Flotation Plant Simulator : Rougher stepped 10%

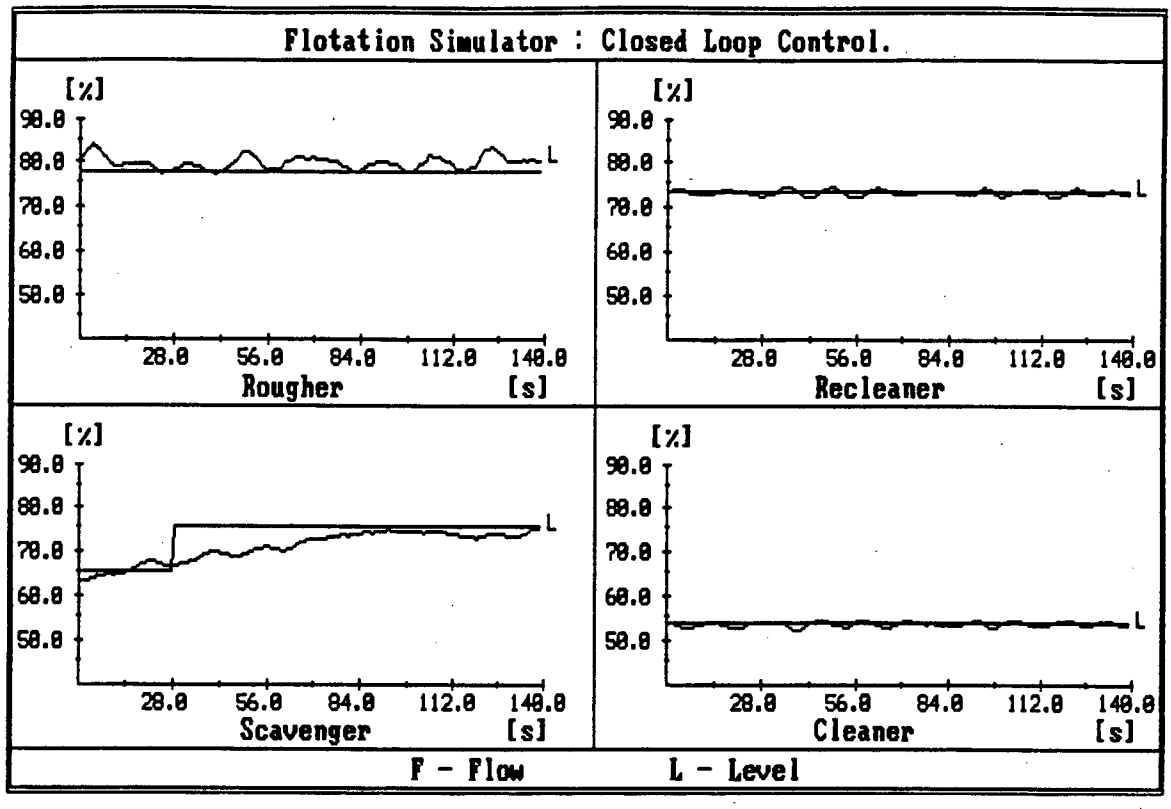


Figure 6.21 Flotation Plant Simulator : Scavenger stepped 10%

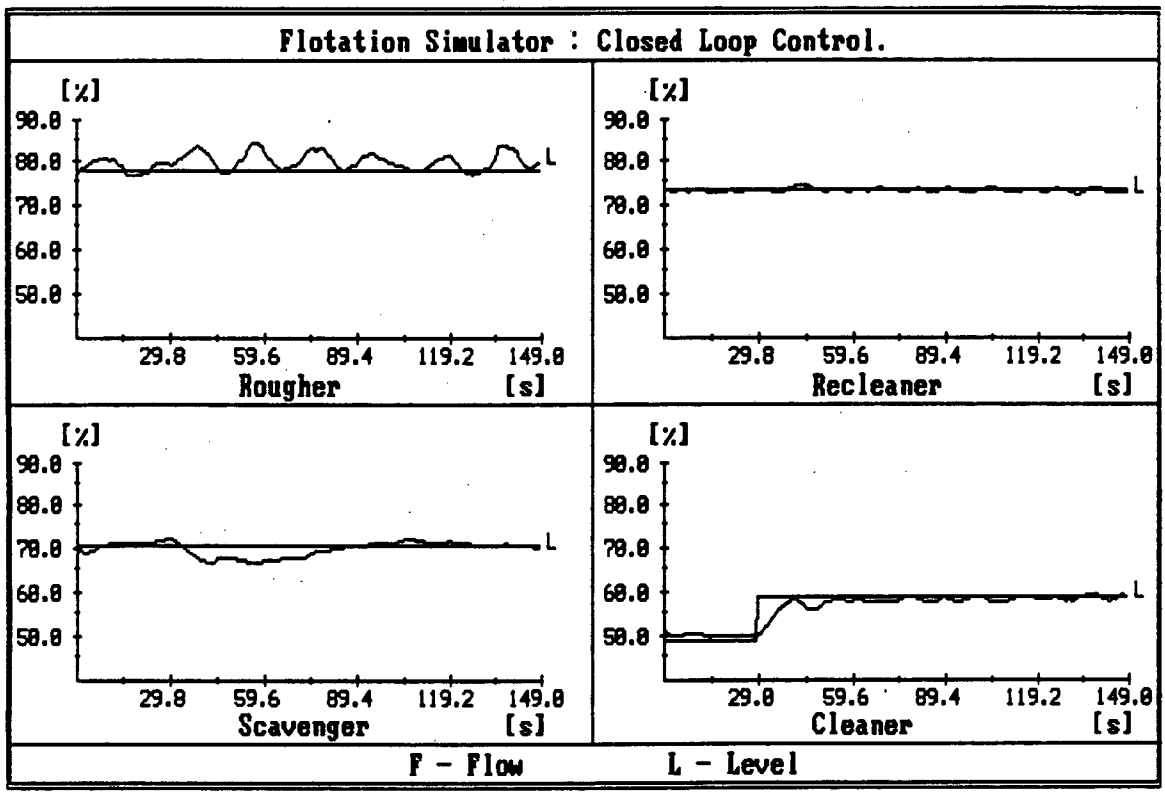


Figure 6.22 Flotation Plant Simulator : Cleaner stepped 10%

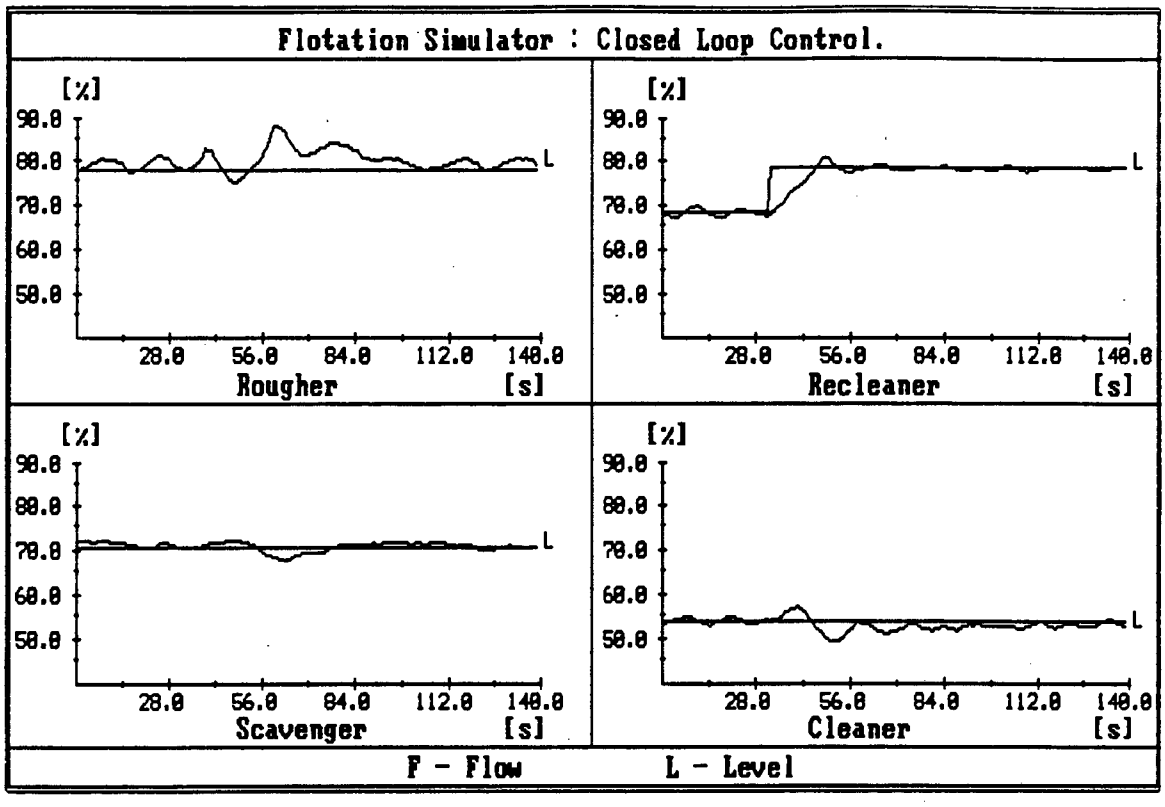


Figure 6.23 Flotation Plant Simulator : Recleaner stepped 10%

The oscillations visible on the cell levels, especially the Rougher and Recleaner levels is due to the pumps surging as described in appendix F, Section F.1. As in chapter 5, the controller has reduced the magnitude of the oscillations when comparing the responses of the open-loop system (chapter 2) and the closed loop system. But, the oscillations could not be eliminated completely due to the high frequency nature of the oscillations.

The table below lists the response times of the the system to a step change to one of the cells setpoints. The two time columns represent the simulated time response and the actual time response of the cell whose setpoint was stepped.

Cell Stepped	Simulated Response [Secs]	Actual Response [Secs]
Rougher	15	20
Scavenger	17	100
Cleaner	10	20
Recleaner	30	20

Table 6.2 Simulated and actual time responses to single setpoint steps.

Table 6.2 shows that the actual response times for the closed-loop system are as expected. But, there are some discrepancies (especially the Scavenger response time) between the simulated and actual response times of the closed-loop system.

The biggest discrepancy in table 6.2 is the Scavenger response time. This discrepancy appears to be the result of the combined effects of i) bounded flow rates on the system and limited controller values as explained in chapter 5, and ii) off-diagonal controller terms dominating the Scavenger control value (due to limiting of all the controller values) in spite of the Scavenger having a large error signal.

The effect of bounded flow rates and control values causing the discrepancies between the actual and simulated closed-loop

system was discussed in chapter 5. This effect explains the discrepancies (except for noise) between the actual and simulated closed-loop system responses shown in figures 6.13 to 6.16 and figures 6.20 to 6.23.

The following figures show the system response to a disturbance of ±10% on each setpoint.

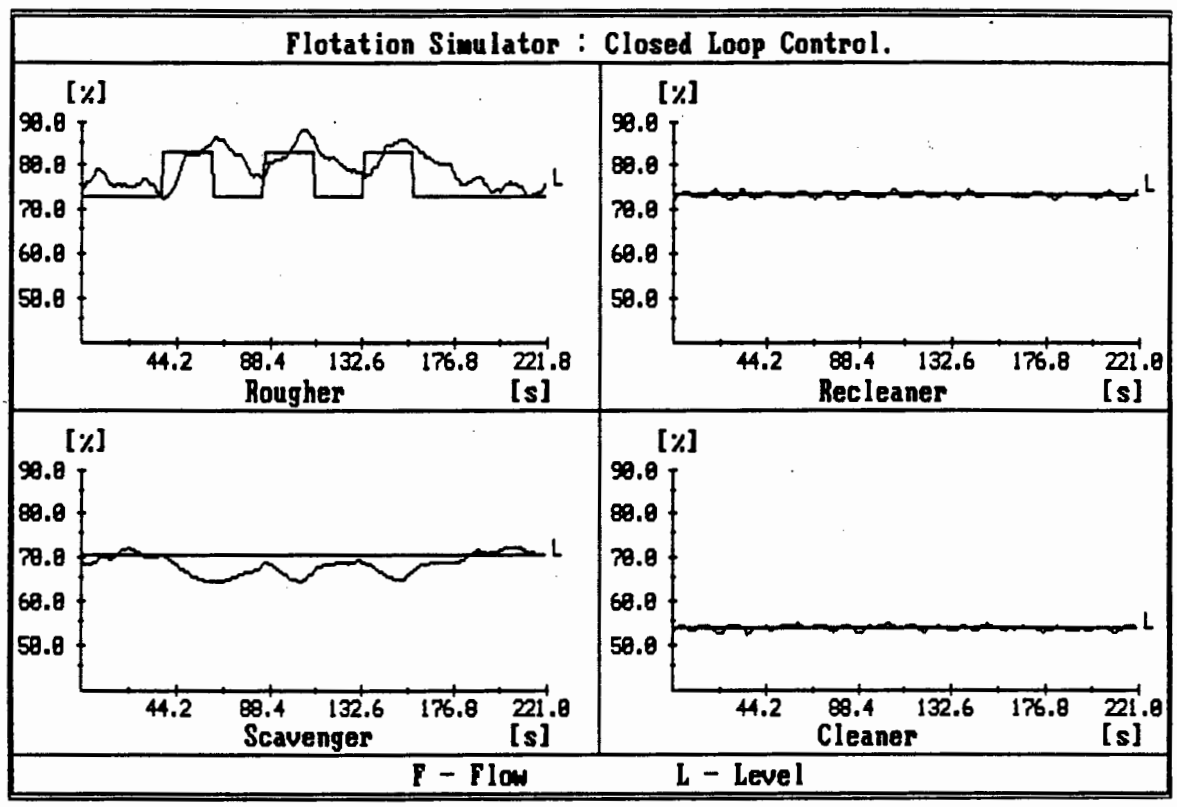


Figure 6.24 Flotation Plant Simulator: Rougher disturbed ±10%

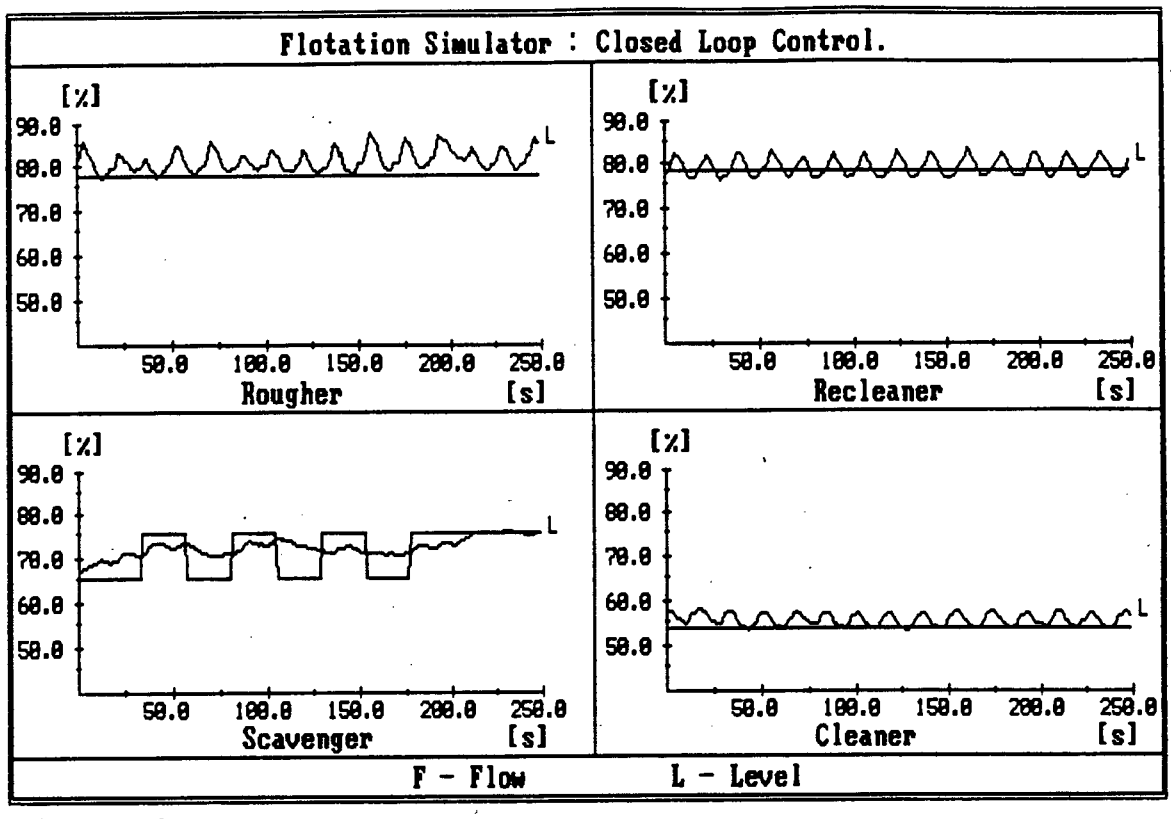


Figure 6.25 Flotation Plant Simulator: Scavenger disturbed ±10%

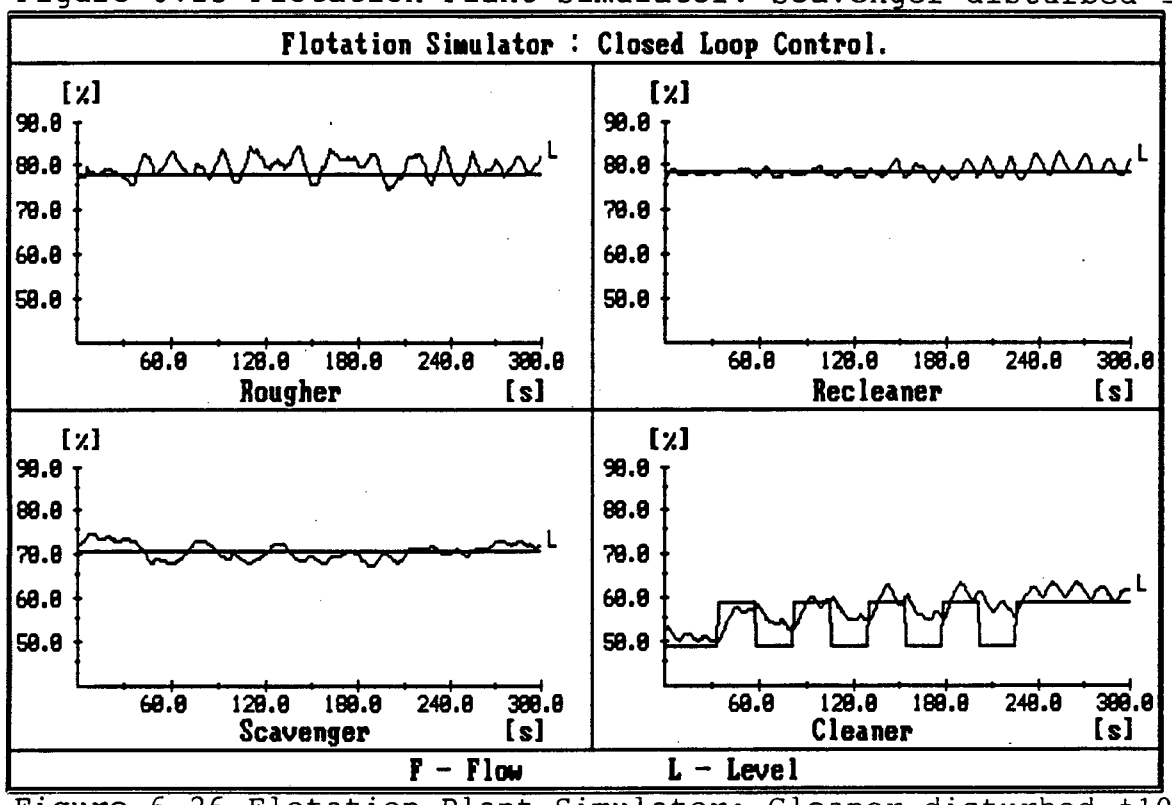


Figure 6.26 Flotation Plant Simulator: Cleaner disturbed ±10%

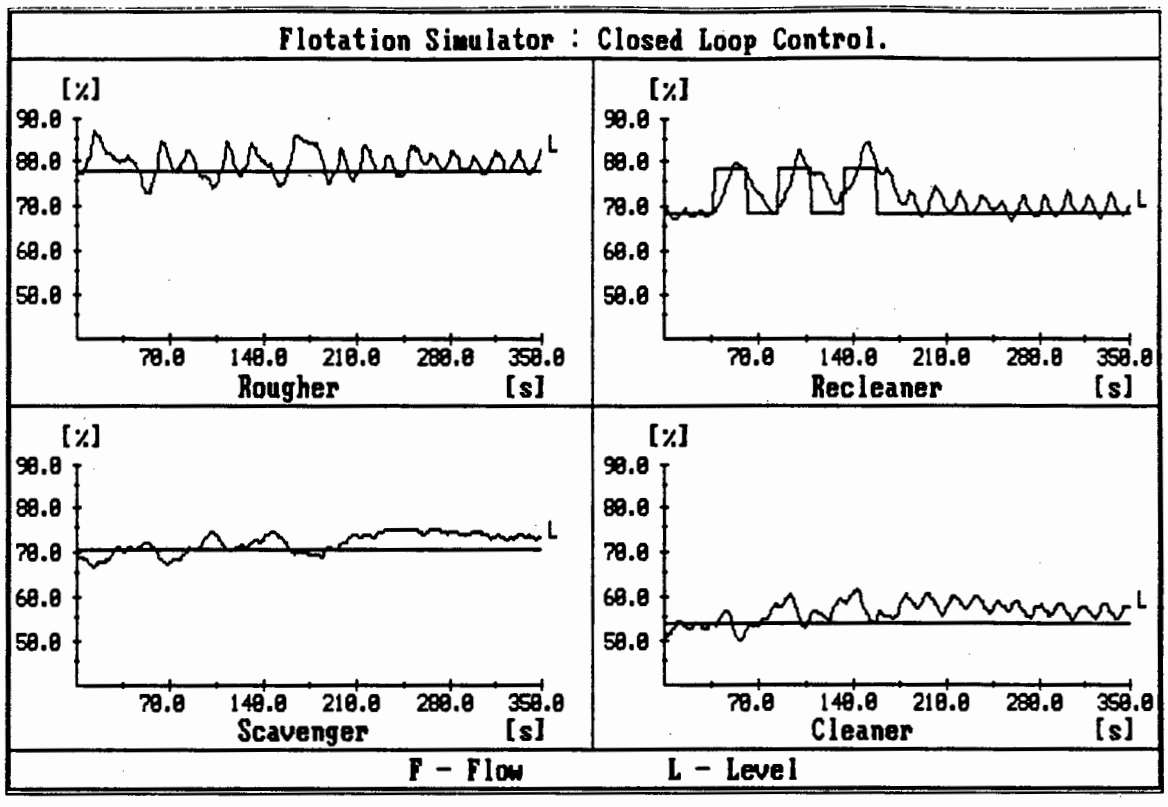


Figure 6.27 Flotation Plant Simulator: Recleaner disturbed ±10%

The closed-loop system remains stable for all of the setpoint disturbance conditions described in figures 6.24 to 6.27. The closed-loop system responses to the setpoint disturbances are similar to the responses to single setpoint steps.

The disturbance on the Rougher and Scavenger setpoints did not result in any visible interaction on the Cleaner and Recleaner. A similar result occurred when the Rougher and Scavenger setpoints were stepped (see figures 6.20 and 6.21).

The disturbance on the Cleaner setpoint resulted in the Rougher and Scavenger being disturbed but not the Recleaner. This

system response is also similar to the case of a single setpoint step to the Recleaner (see figure 6.22).

The disturbance to the Recleaner setpoint had the most significant effect on the closed-loop system. All the remaining cells react to the disturbance. The amount of interaction on the remaining cells is similar to the case of a single setpoint step to the Recleaner (see figure 6.23).

6.4 Conclusion

Although the designed precompensator did not diagonalise the system, single loop PI controllers were designed and the entire control scheme was implemented on the Flotation Plant Simulator.

The controller did succeed in eliminating steady state errors, improving response times and reducing the amount of interaction. The closed-loop did appear to be stable - this characteristic could not be assessed in theory due to the system not being diagonal dominant.

In general, the theoretical and actual closed-loop system characteristic could be correlated and the discrepancies could be explained.

Chapter 7

Conclusions

7.1 Flotation Plant Simulator

The Flotation Plant Simulator (chapter 1) design and implementation can be considered a success as the system functions as expected. The only problematic area on the Flotation Plant Simulator is the "surge-cavitate" cycle of the sump pumps (explained in appendix F, section F.1).

The Flotation Plant Simulator provides an interesting control problem in that the system is completely interactive and a number of the interactive components have delays. Also, the Simulator has many non-linear operating regions. This combination of characteristics implies a complicated model as well as a complicated solution to the control problem. The Simulator can thus be considered a tool in the development of modeling, controller design and controller implementation techniques.

The instrumentation installed on the system is of the type found in industrial applications and presents "real" (ie. noisy and non-linear) plant data to the user of the Simulator. This aspect of the Simulator makes it a valuable teaching instrument in the university environment where students are often "shielded" from real plant problems such as noisy and non-linear instrumentation.

There is room for improvement on the Flotation Plant Simulator such as a solution to the "surge-cavitate" problem on the sump pumps and the installation of more instrumentation. In particular, a control valve and perhaps a flowmeter on the main feed into the system (main feed to Rougher, see figure 1.3)

could have a significant affect on the performance of the Simulator. A suitable controller would have to be designed to use this additional feature. But one should note that if the full system is to be controlled as well as using the additional control valve on the main feed, the system will become a 5th order system. In other words, an even more complicated system.

7.2 Modeling of the Flotation Plant Simulator

Judging by the discrepancy in results of the time simulations and actual responses of the closed-loop system in chapters 5 and 6, it has become obvious that the model presented in chapter 2 is not sufficient. The model was only considered valid in the "linear operating region" (see chapter 2), but it appears that the model is only accurate for a narrow band within that region as well. The reason for this is that there are non-linearities in the system : i) instrument non-linearities, ii) actuator non-linearities, iii) system dynamic non-linearities (ie. the gain of a relationship between two cells is not constant).

If the Flotation Plant Simulator is to be used as a teaching instrument for control techniques, a suitable model will have to be synthesised. This model will have to consider all the non-linear regions in the system such as the cell empty and full extremes, instrument non-linearities and especially the change in dynamics when individual cell levels rise above the froth outflow level (see figure 1.2).

7.3 The Characteristic Loci CAD System

The specific aim of a computer aided design (CAD) system to implement the Characteristic Loci design technique was to make the technique available in the form of a design tool for the students at the university.

The Characteristic Loci CAD system (CL-CAD) is a user friendly (on-line help is available) package that runs on a personal computer. The CL-CAD package enables the user to enter a multivariable system in the form of a transfer function matrix (in frequency domain) and then extract the characteristic loci information from the system.

The CL-CAD system does tend to lack design "tools" to assist the user in designing a suitable multivariable controller for the system. These "tools" or features were not included because the CL-CAD package was used as a means of introducing the design technique. The inclusion of these features in the CL-CAD package would have complicated the introduction of the design technique.

The CL-CAD system has been designed such that additional features can be included at a later stage. These additional features could consist of the following :

i) Design "tools" : For example to synthesise commutative controllers.

ii) Integrity analysis.

iii) Model and controller sensitivity analysis.

7.4 The Characteristic Loci Technique

In the course of the work concerning the Characteristic Loci design technique it has become apparent that the most significant aspect of the technique is the system stability and integrity analysis. As explained in chapter 3, system stability and integrity can be inspected at any point in the design of a control scheme.

The main drawback of the technique is that the designer has difficulty visualising what the loci represent when the actual plant is being considered. This results in the designer having the impression of working blind.

The Characteristic Loci technique does reveal problems within the system but does not enable the designer to identify and isolate the source of the problems. This is due to the fact that the loci represent the system as a whole. Thus, the designer will have difficulty in removing an undesirable characteristic from one particular locus as any modifications to the system tends to affect all the loci.

7.4.1 Control of the Flotation Plant Simulator

After comparing the performance of the closed-loop system presented in chapter 5 and the open-loop system described in chapter 2, the following conclusions can be made :

i) The closed-loop system response is faster than the open-loop system.
ii) The controller has eliminated steady state error in the closed-loop system.
iii) The controller has reduced the amount of interaction that was apparent in the open-loop system.

iv) The controller did manage to reduce noise present on the open-loop system (ie. the amplitude of the "surge-cavitate" cycle of the sump pumps was reduced).

Thus, the controller designed using the Characteristic Loci technique can be considered satisfactory, although the interaction could be reduced even further. The system did respond as expected from the theory apart from a few discrepancies such as the simulated response of the closed-loop system being i) optimistic about the response times and ii) pessimistic about the amount of interaction present on the system.

Another fact that should be mentioned is that the controller may not be considered satisfactory from a design point of view for the following reason : The PI matrix controller can be split into two parts, a diagonal matrix with PI controllers for the diagonal elements cascaded with a constant matrix. This implies that the controller tries to diagonalise the system with a constant matrix. This constant matrix may not be sophisticated enough to diagonalise the system properly.

This technique of using a PI matrix controller is certainly one of the earlier methods of diagonalising a multivariable system. There are more modern techniques mentioned in the literature [10,12] which do enable the design of dynamic compensators to diagonalise a multivariable system.

In conclusion, the Characteristic Loci technique does provide valuable insight to system characteristics but does not satisfy all the designer's requirements of a design technique. The

mathematical elegance of the technique is lost as the designer cannot associate the actual system with the technique's representation of the system. The more experience the author gained by working with the Characteristic Loci technique, the more apparent it became that it is not as methodical as the Inverse Nyquist Array (INA) technique [16,17].

7.5 The Inverse Nyquist Array (INA) Technique

The most striking feature of the INA technique is the simple methodical approach to achieving diagonal dominance. The technique's success is mainly due to the presentation of system characteristics. The method of displaying the system characteristics enables the designer to identify problematic areas within the system. The technique then provides the designer with a method to eliminate or reduce particular elements that are the source of the problems.

The main disadvantage of the INA technique is that stability cannot be accurately assessed until diagonal dominance has been achieved. This is not always possible (as is the case in chapter 6) without designing an excessively complicated precompensator.

7.5.1 Control of the Flotation Plant Simulator

Comparing the performance of the closed-loop system presented in chapter 6 and the open-loop system described in chapter 2, one can come to the same conclusions listed in section 7.4.1 of this chapter.

The controller performed satisfactorily, although one feels that the interaction in the closed-loop system could be reduced even further than at present. The theoretical predictions of

the closed-loop response were fairly accurate even if the simulated response times tended to be optimistic. In all, the closed-loop system did respond as expected from the theory.

In conclusion, both multivariable frequency domain design techniques discussed in this thesis have their merits and disadvantages. Their design philosophies are different, the Characteristic Loci technique tries a "unified" approach to diagonalising the system (ie. all problems are solved at once). Whereas, the INA technique approaches the problem of diagonalising a system with a "divide-and-conquer" philosophy.

A good approach by a designer of multivariable control systems would be to make full use of the major advantages of the two techniques, which are :

i) The INA technique provides a simple design objective in aiming for diagonal dominance.

ii) The Characteristic Loci technique leads to ready assessment of overall system stability, integrity and performance properties.

References

1. Perry J.H., Eds., Chemical Engineers Handbook, 4th ed., New York, McGraw-Hill, 1963.

2. Venzke R.H.E., Comparison of INA technique to the Pole Assignment Technique, MSc. Dissertation (in preparation), 1988.

3. Twidle T.R. et al, "Improvements in stabilising control at Black Mountain", Journal of the South African Institute of Mining and Metallurgy, January 1986, p.15-24.

4. Bell D.J., Cook P.A. and Munro N., Eds., Design of Modern Control Systems, London, Peter Peregrinus, 1982.

5. Bode H.W., Network Analysis and Feedback Amplifier Design, Van Nostrand, Princeton, 1905.

6. Nyquist H., "Regeneration theory", Bell Systems Tech. J., Vol. 11, p.126-147, 1932.

7. MacFarlane A.G.J., "Return-difference and return-ratio matrices and their use in analysis and design of multivariable feedback control systems", Proc. IEE, Vol. 117, No. 10, p.2037-2049, October 1970.

8. Belletrutti J.J. and MacFarlane A.G.J., "Characteristic loci techniques in multivariable control system design", IEE Proc., Vol. 118, p.1291-1297, 1971.

9. MacFarlane A.G.J. and Belletrutti J.J., "The Characteristic Locus Design method", Automatica, Vol. 9, p.575-588, 1973.

10. Kouvaritakis B.A., "Theory and practice of the characteristic locus design method", Proc. IEE, Vol. 126, no. 6, p.542-548, June 1979.

11. Kouvaritakis B.A. and Shaked U., "Asymptotic behavior of root loci of linear multivariable systems", Int. J. Control, Vol. 23, No. 3, p.297-340, 1976.

12. Owens D.H., "Compensation theory for multivariable root loci", Proc. IEE, Vol. 126, No. 6, p.538-541, June 1979.

13. Rosenbrock H.H. and Cook P.A., "Stability and the eigenvalues of G(s)", Int. J. Control, Vol. 1, p.99-104, 1975.

14. MacFarlane A.G.J. and Postlethwaite I., "The generalised Nyquist stability criterion and multivariable root loci", Int. J. Control, Vol. 25, p.81-127, 1977.

15. Postlethwaite I. et al, "Principal Gains and Principal Phases in the Analysis of Linear Multivariable Feedback Systems", IEEE Transactions on Automatic Control, Vol. AC-26, No. 1, p.32-46, February 1981.

16. Munro N., Modern Approaches to Control Systems Design, Stevenage, Peter Peregrinus, 1979.

17. Rosenbrock H.H., :Design of linear multivariable control systems using the inverse Nyquist array", Proc. IEE, Vol. 116, p.1929-1936, 1969.

18. Edmunds J. and Kouvaritakis B.A., "Extensions of the frame alignment technique and their use in the characteristic locus design technique", Int. J. Control, Vol. 29, No. 5, p.787-796, 1979.

19. Kouvaritakis B.A., "Complex Align : A technique for the characteritsic locus method of design", IEE Proc., Vol. 129, No. 1, p.1-5, January 1982.

20. Gantmacher F.R., Theory of Matrices, Chelsea, New York, 1959.

21. Stewart G.W., Introduction to Matrix Computations, Academic Press, New York, 1973.

22. Martin and Wilkinson, , "Procedure COMHES", Num. Math., No. 12, p.349-368, 1968.

23. Martin W.L, Handbook of Industrial Piping, London, Isaac Pitman, 1961.

24. King R.C., Eds., Piping Handbook, 5th ed., New York, McGraw-Hill, 1967.

Bibliography

1. Applications of Multivariable Systems Theory, London, 1982. Collected papers from a symposium, 'Applications of Multivariable Systems Theory', Plymouth, 1982.

2. Astrom K.J. and Wittenmark B., Computer Controlled Systems, Prentice Hall, 1984.

3. Bell D.J., Cook P.A. and Munro N., Eds., Design of Modern Control Systems, London, Peter Peregrinus, 1982.

4. Belletrutti J.J. and MacFarlane A.G.J., "Characteristic loci techniques in multivariable control system design", IEE Proc., Vol. 118, p.1291-1297, 1971.

5. Bode H.W., Network Analysis and Feedback Amplifier Design, Van Nostrand, Princeton, 1905.

6. Chavel I., Eigenvalues in Riemannian Geometry, Academic Press, 1984.

7. Edmunds J. and Kouvaritakis B.A., "Extensions of the frame alignment technique and their use in the characteristic locus design technique", Int. J. Control, Vol. 29, No. 5, p.787-796, 1979.

8. Gantmacher F.R., Theory of Matrices, Chelsea, New York, 1959.

9. Hughes F.M. and Mallouppa A., "Frequency Response Methods for Nuclear Station Boiler Control", Automatica, Vol. 12, No. 3, p.201-210, 1976.

10. King R.C., Eds., Piping Handbook, 5th ed., New York, McGraw-Hill, 1967.

11. Kouvaritakis B.A. and Shaked U., "Asymptotic behavior of root loci of linear multivariable systems", Int. J. Control, Vol. 23, No. 3, p.297-340, 1976.

12. Kouvaritakis B.A., "Theory and practice of the characteristic locus design method", Proc. IEE, Vol. 126, no. 6, p.542-548, June 1979.

13. Kouvaritakis B.A., "Complex Align : A technique for the characteritsic locus method of design", IEE Proc., Vol. 129, No. 1, p.1-5, January 1982.

14. Layton J., Multivariable Control Theory, Stevenage, Herts, Peter Peregrinus, 1976.

15. MacFarlane A.G.J., "Return-difference and return-ratio matrices and their use in analysis and design of multivariable feedback control systems", Proc. IEE, Vol. 117, No. 10, p.2037-2049, October 1970.

16. MacFarlane A.G.J. and Belletrutti J.J., "The Characteristic Locus Design method", Automatica, Vol. 9, p.575-588, 1973.

17. MacFarlane A.G.J. and Postlethwaite I., "The generalised Nyquist stability criterion and multivariable root loci", Int. J. Control, Vol. 25, p.81-127, 1977.

18. Martin W.L, Handbook of Industrial Piping, London, Isaac Pitman, 1961.

19. Martin and Wilkinson, , "Procedure COMHES", Num. Math., No. 12, p.349-368, 1968.

20. Munro N., Modern Approaches to Control Systems Design, Stevenage, Peter Peregrinus, 1979.

21. Nyquist H., "Regeneration theory", Bell Systems Tech. J., Vol. 11, p.126-147, 1932.

22. Owens D.H., "Compensation theory for multivariable root loci", Proc. IEE, Vol. 126, No. 6, p.538-541, June 1979.

23. Patel R.V. and Munro N., Multivariable System Theory and Design, Oxford, Pergamon, 1982.

24. Paraskevopoulos P.N., "Eigenvalue assignment of linear multivariable 2-dimensional systems", Proc. IEE, Vol. 126, No. 11, p.1204-1208, November 1979.

25. Perry J.H., Eds., Chemical Engineers Handbook, 4th ed., New York, McGraw-Hill, 1963.

26. Postlethwaite I. et al, "Principal Gains and Principal Phases in the Analysis of Linear Multivariable Feedback Systems", IEEE Transactions on Automatic Control, Vol. AC-26, No. 1, p.32-46, February 1981.

27. Rase H.F., Piping Design for Process Plants, New York, Wiley, 1963.

28. Rosenbrock H.H., :Design of linear multivariable control systems using the inverse Nyquist array", Proc. IEE, Vol. 116, p.1929-1936, 1969.

29. Rosenbrock H.H. and Cook P.A., "Stability and the eigenvalues of G(s)", Int. J. Control, Vol. 1, p.99-104, 1975.

30. Stewart G.W., Introduction to Matrix Computations, Academic Press, New York, 1973.

31. Twidle T.R. et al, "Improvements in stabilising control at Black Mountain", Journal of the South African Institute of Mining and Metallurgy, January 1986, p.15-24.

32. Venzke R.H.E., Comparison of INA technique to the Pole Assignment Technique, MSc. Dissertation (in preparation), 1988.

33. Wilkinson J.H., The Algebraic Eigenvalue Problem, Oxford, Clarendon Press, 1965.

34. Wilkinson J.H., Linear Algebra, C. Reinsch, Berlin, Springer, 1971.

Appendix A Flotation Simulator Plant Instrument Calculations

A.1 Flowmeters

The flowmeters selected for use on the flotation simulator measure the flow by detecting the motion of a small paddle wheel inserted into the flow. Due to the inertia of the paddle wheel, the sensors require a minimum flow velocity of 0.31 [m/s] in order to produce reliable data.

Thus the piping must be sized to fulfill two criteria :-

i) The maximum volume throughput must be maintained when the maximum flow velocity is attained.

ii) The flow velocity when minimum volume throughput is maintained must be greater than the flowmeters' minimum velocity.

But, for a narrow pipe section and a high flow velocity, the pressure drop across the pipe section will become unacceptable. The resultant pressure drop will prevent the required volume throughput from being achieved.

A solution to this problem is to use a short narrowed pipe section, into which the flow sensor is inserted. The remainder of the piping is kept wide in comparison to the volume throughput, ie. wide pipe section will result in low flow velocities and thus a small pressure drop in the remaining pipe section.

The minimum diameter and length of the narrow pipe section for the flow sensor is calculated as follows :-

i) The maximum head of water attainable for the concentrate output is h_1 in figure A.1 On the flotation plant simulator h_1 has been set to 0.2 [m].

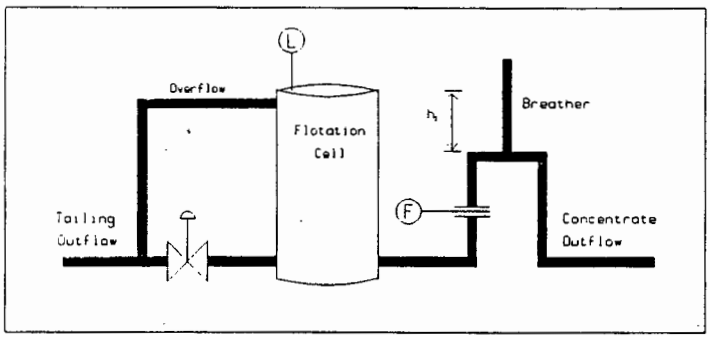


Figure A.1 Flotation Simulator Cell Configuration.

Thus the maximum velocity : $V^2 = 2\ g\ h$
(from Benoullis' equations [2])

Where :- V - flow velocity [m/s]
g - gravitational acceleration [m/s^2]
h - head of water [m]

Thus, on the flotation plant simulator : V = 1.98 [m/s]

But, due to friction components, tank exits and discontinuities in the piping, a flow velocity of 1.5 [m/s] was used in the calculations.

ii) The minimum diameter of the pipe section can be calculated using the maximum volume rate and the velocity of the flow (figure A.2) contains the desired flow rates on the flotation plant simulator) :-

The area of the pipe section : $A = \frac{K\ Vol}{Vel}$

Where : A = Area [m^2]
K = 0.001 Constant
Vol = Volume rate [l/s]
Vel = Flow velocity [m/s]

Thus, diameter of the pipe can be calculated.

Since schedule 80 PVC piping is to be used, the closest approximation to the calculated diameter should be taken as the final diameter of the pipe section.

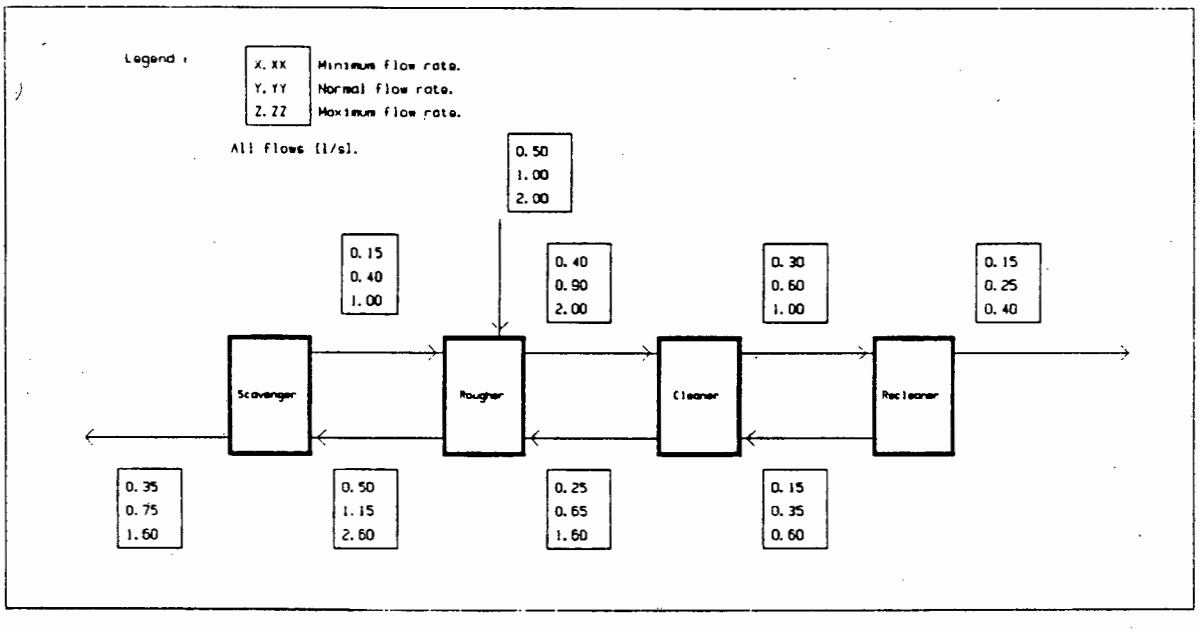


Figure A.2 Normal and Extreme Flow Rates on the Flotation Plant Simulator.

iii) New flow velocities are now calculated using the original volume rates and the final pipe diameter.

iv) The Reynolds number is calculated using the new flow velocities and the final pipe diameter [2, 3, 4].

Reynolds Number : $R_e = \frac{Vel\ D\ p}{u}$

Where : Vel = Flow velocity [m/s]
D = Pipe diameter [m]
u = Viscosity of water at 25°C
= 8.9×10^{-4} [Kg(mass)/m sec]
p = Density of water at 25° C
= 1×10^{3} [Kg/m^3]

Thus, the friction factor can be read off Moody charts [3] using the calculated Reynolds number.

v) The friction factor is used to calculate the (L_e/D) factor. This is the length to diameter ratio for the narrow pipe section (for the flowmeter) and must not be exceeded, otherwise the pressure drop across the section will become unacceptable.

Table A.1 below shows the result of the calculations described above.

Cell	Flow rate		Reynolds number	Friction factor	Ratio
Pipe Dia.	Type	[m/s]	R_e	f	L_e/D
Scavenger	Max.	1.25	6700	0.035	29
32.0 [mm]	Normal	0.40	2600	0.050	20
Rougher	Max.	1.50	10500	0.030	33
37.5 [mm]	Normal	0.80	5600	0.038	26
Cleaner	Max.	1.25	6700	0.035	29
32.0 [mm]	Normal	0.75	4000	0.042	23
Recleaner	Max.	1.50	4700	0.040	25
18.3 [mm]	Normal	0.95	2950	0.045	22

Table A.1 Flow Sensor Pipe Section Sizing Calculations.

The pipe diameters quoted in table A.1 are the inner diameter specifications for schedule 80 PVC pipes.

A.2 Control Valves

The tailings output of each 'flotation' cell in the flotation plant simulator is to be regulated by a pneumatically actuated control valve.

Before the control valves can be selected, the required valve coefficient of each valve must be calculated. The calculated control valve constants [K_v] are listed in Table A.2 below.

The K_v values were calculated using the following equation [2] :-

$$K_v = (dP/F)^{\frac{1}{2}}$$

where :- K_v - valve coeff. [m^3/hr] at 1 bar pressure drop.
dP - pressure drop [bars]
F - volume rate [m^3/hr]

Cell	Flow rate		Head	K_v
	Type	[l/s]	[m]	[m^3/hr]
Scavenger	Min.	0.35	0.5	6.0
	Normal	0.75	0.5	12.2
	Max.	1.60	0.5	26.0
Rougher	Min.	0.50	0.5	8.0
	Normal	1.15	0.5	19.0
	Max.	2.60	0.5	42.0
Cleaner	Min	0.25	0.5	5.0
	Normal	0.65	0.5	12.6
	Max.	1.60	0.5	31.0
Recleaner	Min.	0.15	0.5	3.0
	Normal	0.35	0.5	7.0
	Max.	0.60	0.5	12.0

Table A.2 The Calculated Valve Coefficients.

Appendix B Equipment for the Flotation Simulator

The following is the list of equipment used to construct the flotation simulator.

B.1 Tanks

B.1.1 Flotation Cells

All the tanks used were cylindrical, open top tanks constructed of plastic. The tanks were sized as follows :-

Cell	Diameter [mm]	Height [mm]	Volume [l]
Rougher	450	730	100
Scavenger	450	730	100
Cleaner	345	570	45
Recleaner	345	570	45
Sump 1	345	570	45
Sump 2	345	570	45

Table B.1 Dimensions of the tanks to simulate flotation cells.

B.1.2 Spillage Catch Tray

The spillage catch tray was designed to cover the entire floor area used by the flotation simulator. This was done to ensure that any overflow from the simulator would flow into the catch tray.

A second reason for the large surface area of the catch tray : the catch tray is used as the main feed sump and requires a large volume. The large area of the catch tray enables the height of the walls of the catch tray to be kept to a minimum while still ensuring a large volume. The low walls on the catch tray simplified the construction of the catch tray, as the walls do not have to support a large pressure head.

The spillage catch tray dimensions :-

Size : 2900 X 2000 X 170 [mm]

Volume : 800 [l]

B.2 Piping

All the piping used on the flotation simulator was :-

50 mm class IV PVC
(inner diameter : 46 mm)

This piping is fitted into all the T-pieces and elbows using a PVC bonding agent.

The narrowed PVC pipe sections for the flow sensors were not constructed of the piping mentioned above, but Schedule 80 PVC T-pieces provided with the flow sensors.

B.3 Framework

The framework was constructed using 1 inch (25.4 mm) square steel tubing. The framework consists of three rectangular frames bolted together to form the three tiers on which the tanks are placed. Diamond mesh (3 mm)is used as the platform material on which the tanks rest.

The diagram which describes the basic framework structure for the Flotation Plant Simulator is contained in appendix C, figure C.2.

B.4 Pumps

The pumps used for the main feed and for the Rougher and Recleaner feedback circuits, are as follows :-

Manufacturer : Speck

Model : 40/9
motor rating : 0.4 [KW]
Supply : 220 Vac

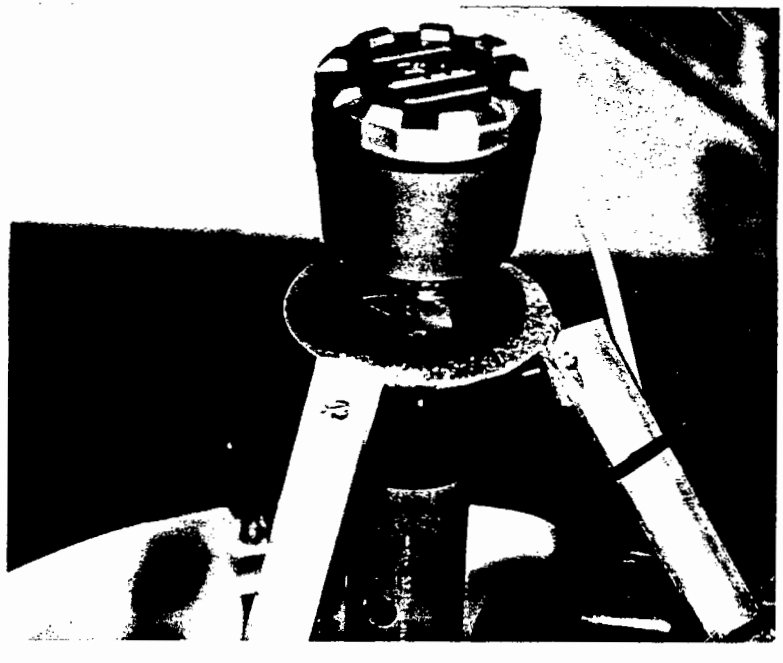
(capable of pumping a 3.0 [m] head of water at a maximum of 2.2 [l/s])

B.5 Level Sensors

Capacitive level probes were selected to measure the level of product in each flotation simulator cell. The following devices were chosen :-

Manufacturer : Endress and Hauser
Model : FM4C20 - Amplifier
EC112 - Insertion piece
11301 - Probe (1 [m])
The device is capable of a variety of output formats (ie. 4-20 mA, 0-20 mA and 0-10 V)

Due to the fact that tanks were made out of plastic, a one inch aluminium strip was placed inside, on the wall of the tank. This aluminium strip acted as the reference electrode for the capacitance measurement. The probe was inserted into the centre of the tank and held steady using an aluminium bracket (see photograph below).



Photograph B.1 The Level Sensor Installation on the Scavenger on the Flotation Simulator

B.6 Flow Sensors

The flow sensors selected for the flotation plant simulator use a small paddle wheel inserted into the flow to measure the flow velocity. Each flow sensor is supplied with a schedule 80 PVC T-piece which ensures the instrument is inserted the correct distance into the flow. The sizing of each T-piece has been documented in Appendix 1.1.

The following sensors were selected :-

Manufacturer : Signet

Model : Flow transmitter (2 wire)
Type : 3-8500-OP

T-pieces : Rougher : PV8T015
Scavenger : PV8T015
Cleaner : PV8T012
Recleaner : PV8T007

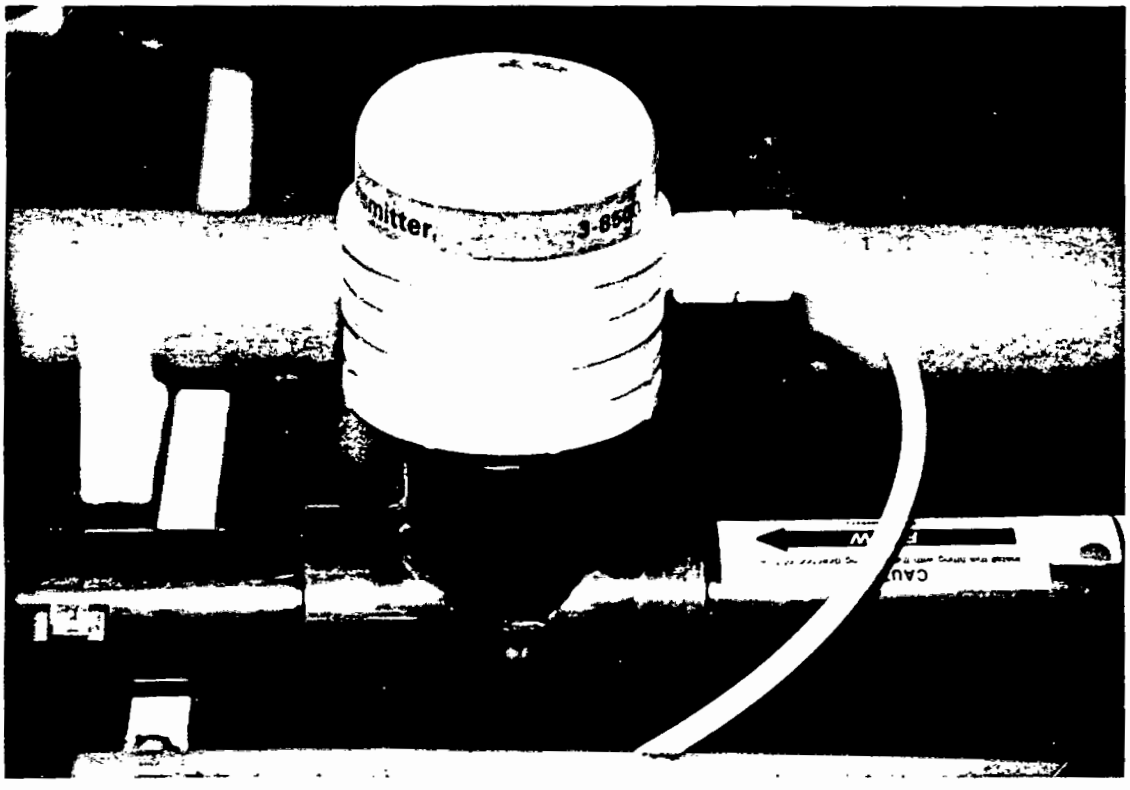
The device outputs a 4-20 mA signal in proportion to the flow velocity.

The flow transmitters require a 24 Vdc stabilised power supply. The following power supplies were selected to power the flow sensors :-

Manufacturer : Action Instruments

Model : AP9046 DC Power Supply
24 Vdc regulated output up to 65mA.
220 Vac input voltage.

The photograph below shows the flow sensor installation on the Cleaner.



Photograph B.2 The Flow Sensor Installation on the Cleaner on the Flotation Plant Simulator

B.7 Control Valves

The size of the control valves selected for the flotation plant simulator have been calculated in Appendix A.2 The valves selected are :-

Manufacturer : Gemu Valves

Model : Rougher : 690/50/D 0114-2
Scavenger : 690/40/D 0114-2
Cleaner : 690/40/D 0114-2
Recleaner : 690/25/D 0114-2

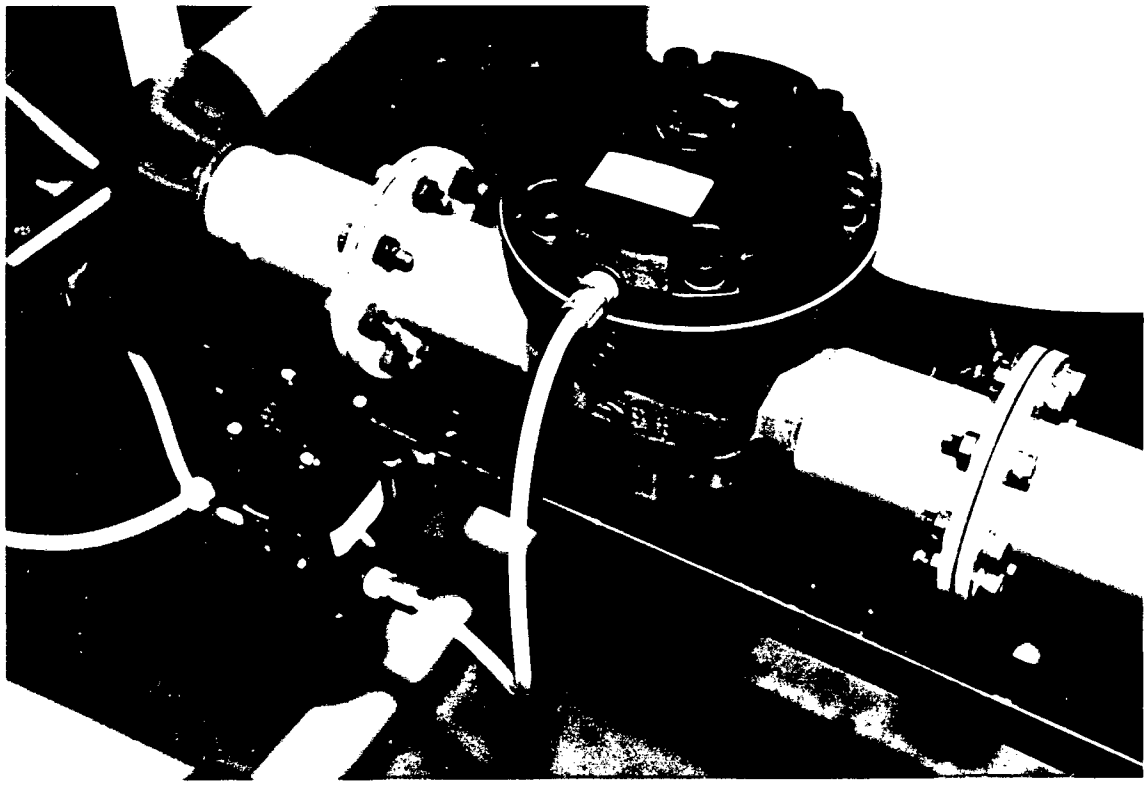
(All the valves have a plastic diaphragm and are pneumatically actuated. The valves are of the normally open type.)

Each valves requires a current to pressure (I/P) converter in order to convert the electrical control signal into valve action. The following I/P converters were selected :-

Manufacturer : John Watson and Smith LTD

Model : 100X
The instrument produces a pressure of 0.2-1.0 bar for an input signal of 4-20 mA.
Supply pressure : 1.4 - 7.0 bar.

The photograph below describes the valve and I/P converter installation on the Cleaner.



Photograph B.3 The Valve and I/P Converter Installation on the Cleaner on the Flotation Plant Simulator

An additional piece of equipment was required by the valve control system : The control signal is a voltage output (0-10 V) from an computer interface card and thus has to be converted into a current (4-20 mA). The voltage to current converter was selected as follows :-

Manufacturer : TCL

Model : TCL
0-10 V input
4-20 mA output
220 Vac Supply

B.8 Pneumatic System

The I/P converters require clean, regulated compressed air in order to actuate the pneumatic control valves. The selected compressor and air receiver:-

Compressor :

Manufacturer : ITT Pneumotive
Model : GH-8104
Oil-less
4 cylinder
1.5 HP Motor
100 PSI Compressor

Air Receiver :

Manufacturer : Arlec Engineering Works
Model : Cylindrical Welded Air Receiver
Capacity : 0.150 cubic metres
Maximum Working Pressure : 1035 kPa

The air from the compressor/receiver system was then fed through a water trap/air filter unit. The filter was a standard 25 micron air filter. The unit selected was as follows :-

Manufacturer : Hyflo
Model : AF3000-02

The air from the water trap/air filter unit was then fed through a pressure regulator. The regulator selected was as follows :-

Manufacturer : John Watson and Smith LTD
Model : 10-B Manostat Precision Pressure Regulator

B.9 Computer System

To enable a variety of control schemes to be implemented on the flotation plant simulator, all instruments are interfaced to a personal computer. The personal computer is configured as follows :-

Type : IBM compatible PC-AT (10 MHz)
640 KBytes of RAM
Intel 80287 Numeric Coprocessor
1 X 20 MByte Hard Disk Drive
1 X 360 KByte Floppy Disk Drive
Hercules Graphics Card and Monitor

In order to interface the computer to the flotation plant simulator, two analog/digital (A/D) interface cards were selected :-

Manufacturer : Data Translation

Model : DT2815
12 bit D/A interface card
8 analog output channels (0-5 V)

Model : DT757
Termination panel required by DT2815

Model : DT2801

12 bit A/D/A interface card

16 analog input channels (0-10 V)

2 analog output channels (0-10 V)

16 digital I/O channels

Model : DT707

Termination panel required by DT2801

The diagram (figure B.1) below describes the logical layout of the interface system.

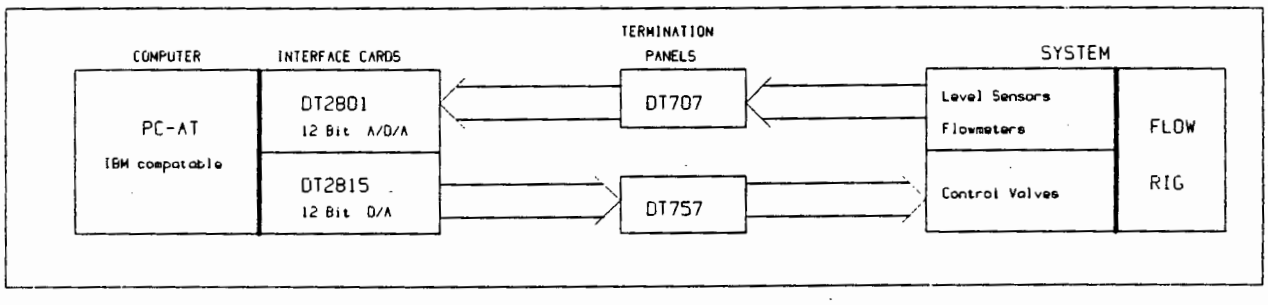


Figure B.1 The Logical Layout of Computer Interface to the Flotation Plant Simulator

Appendix C Drawings of the Flotation Plant Simulator

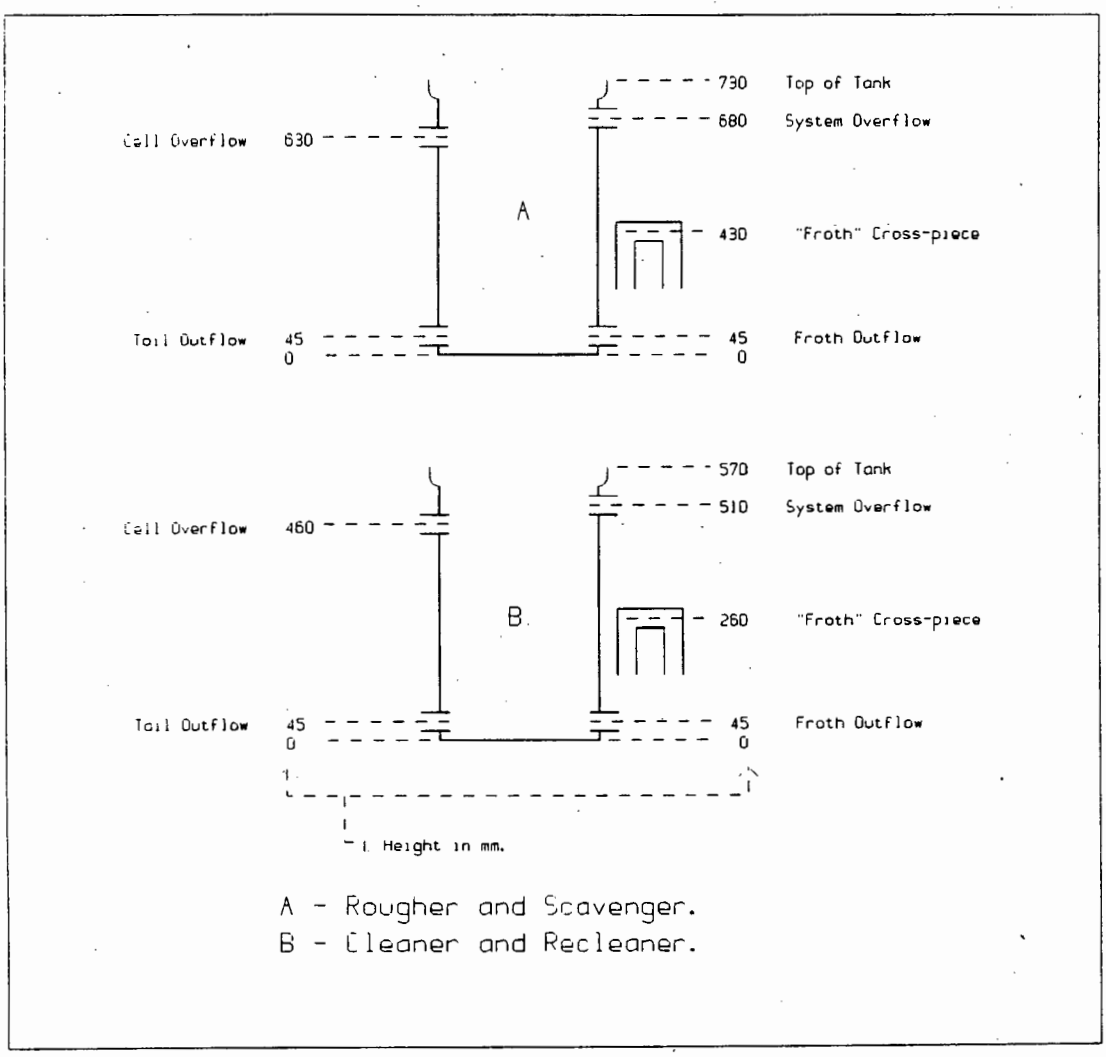


Figure C.1 Relative Heights of Exits form Simulated Flotation Cell

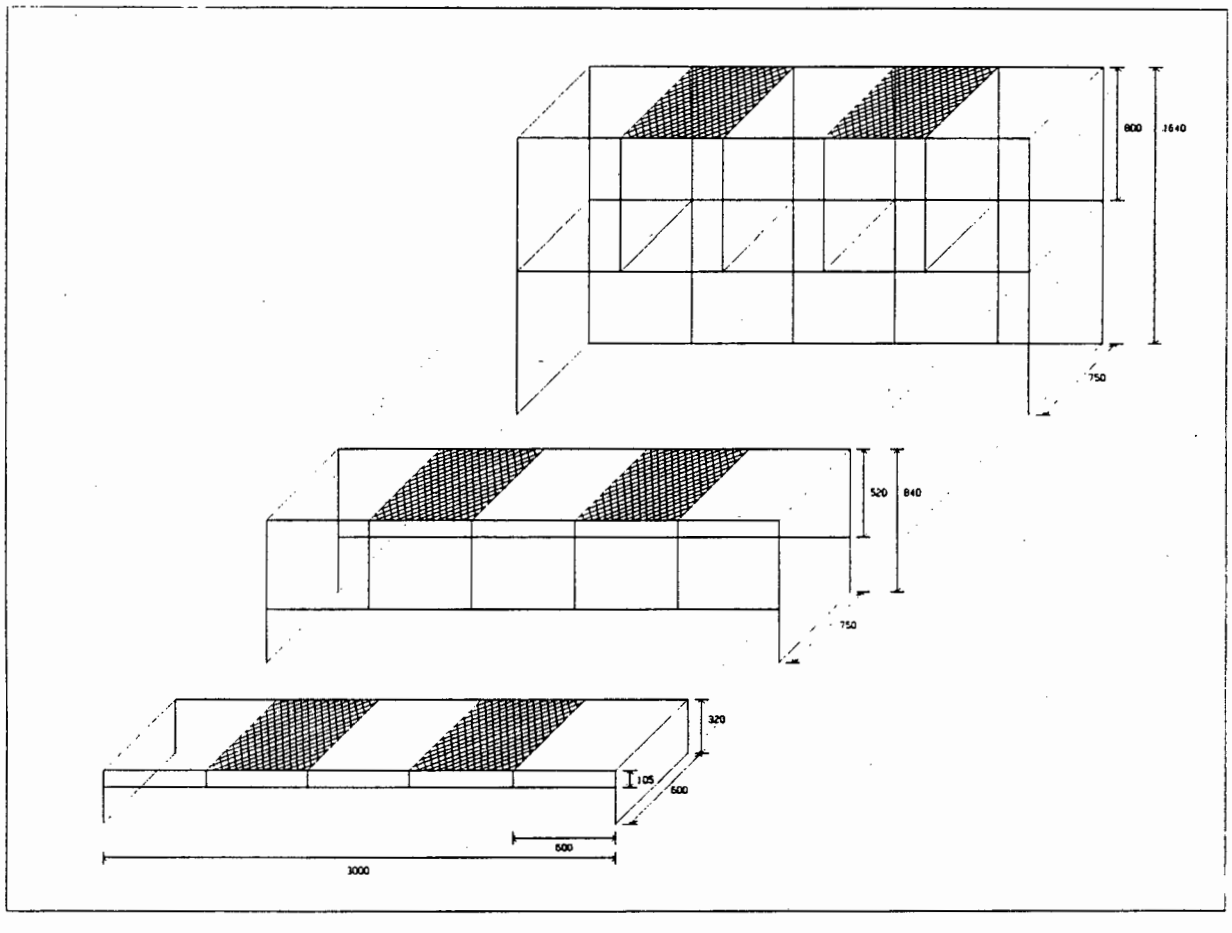


Figure C.2 Flotation Plant Simulator Framework Dimensions

Appendix D The Instrument Circuit Diagrams for the Flotation Plant Simulator

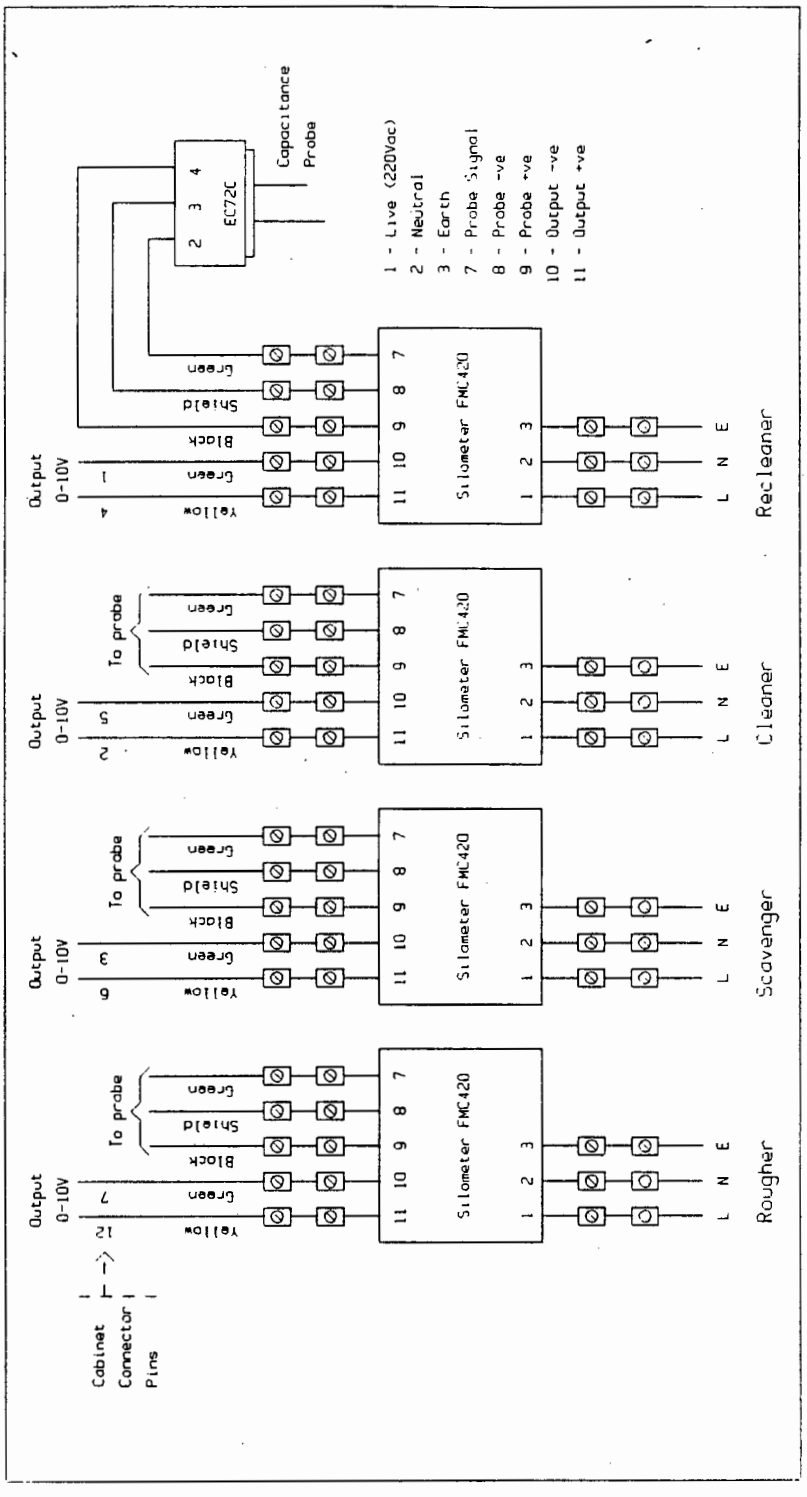


Figure D.1 The Circuit Diagram for the Level Sensors

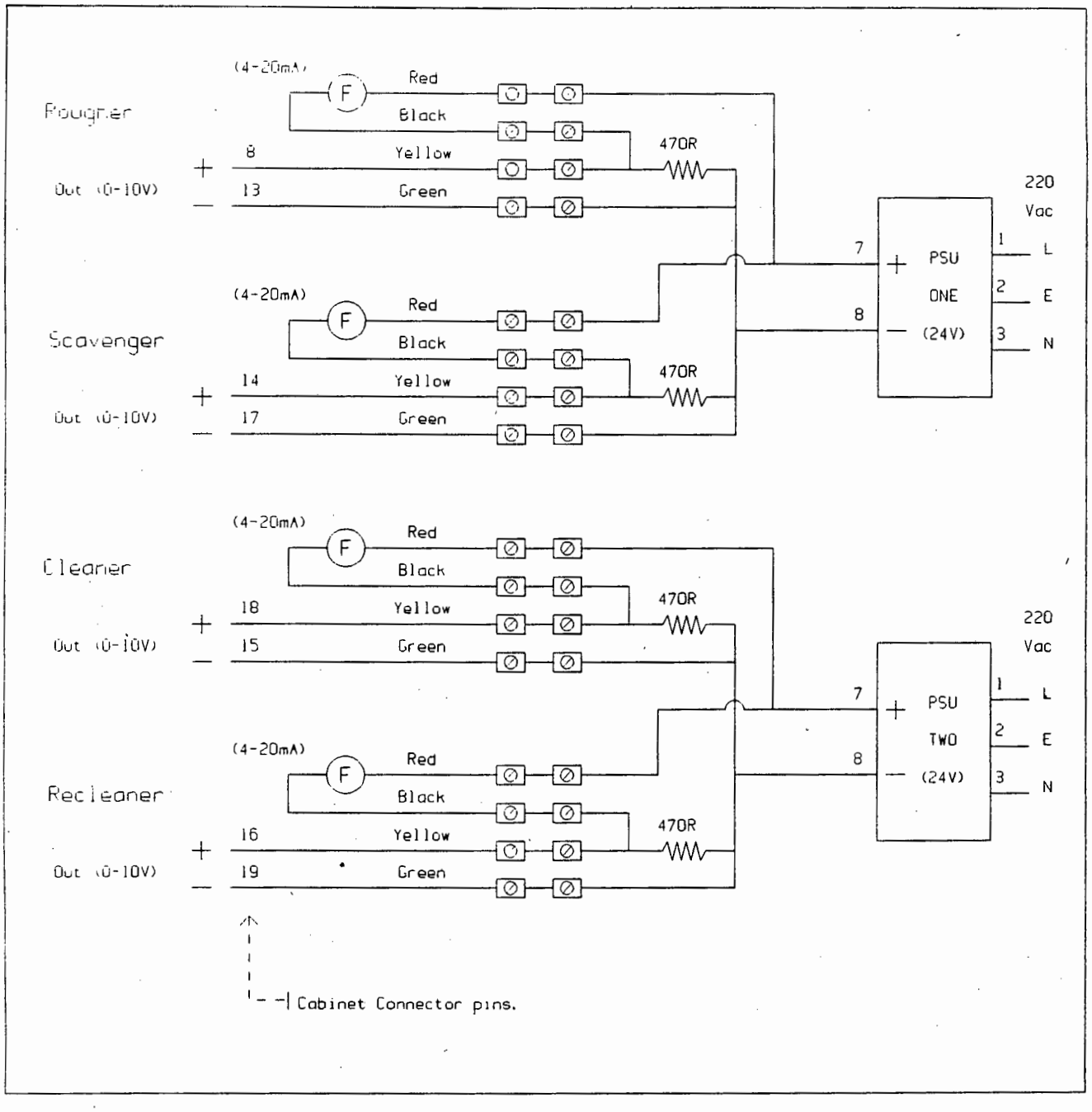


Figure D.2 The Circuit Diagram for the Flow Sensors

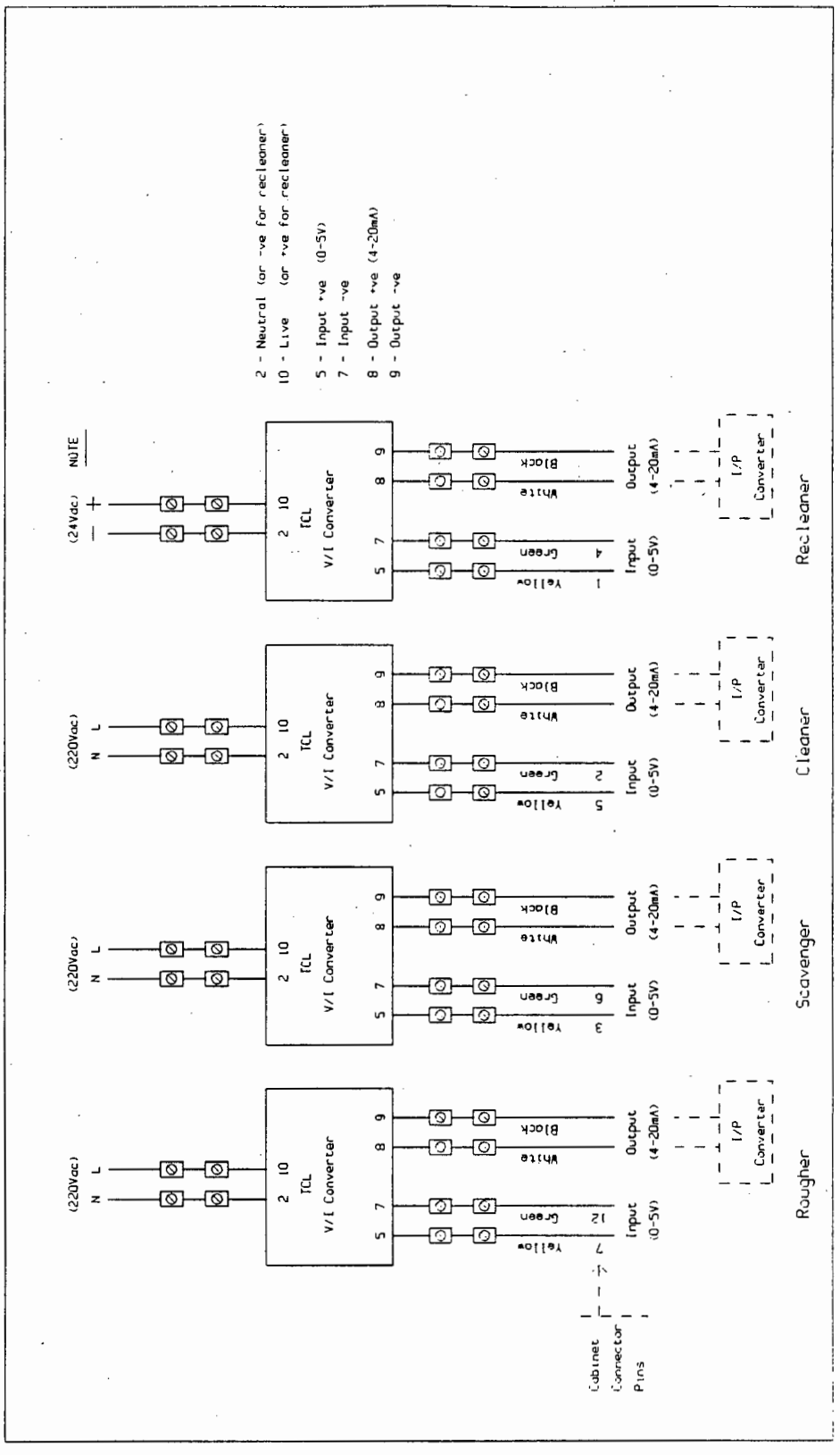


Figure D.3 The Circuit Diagram for the Actuators

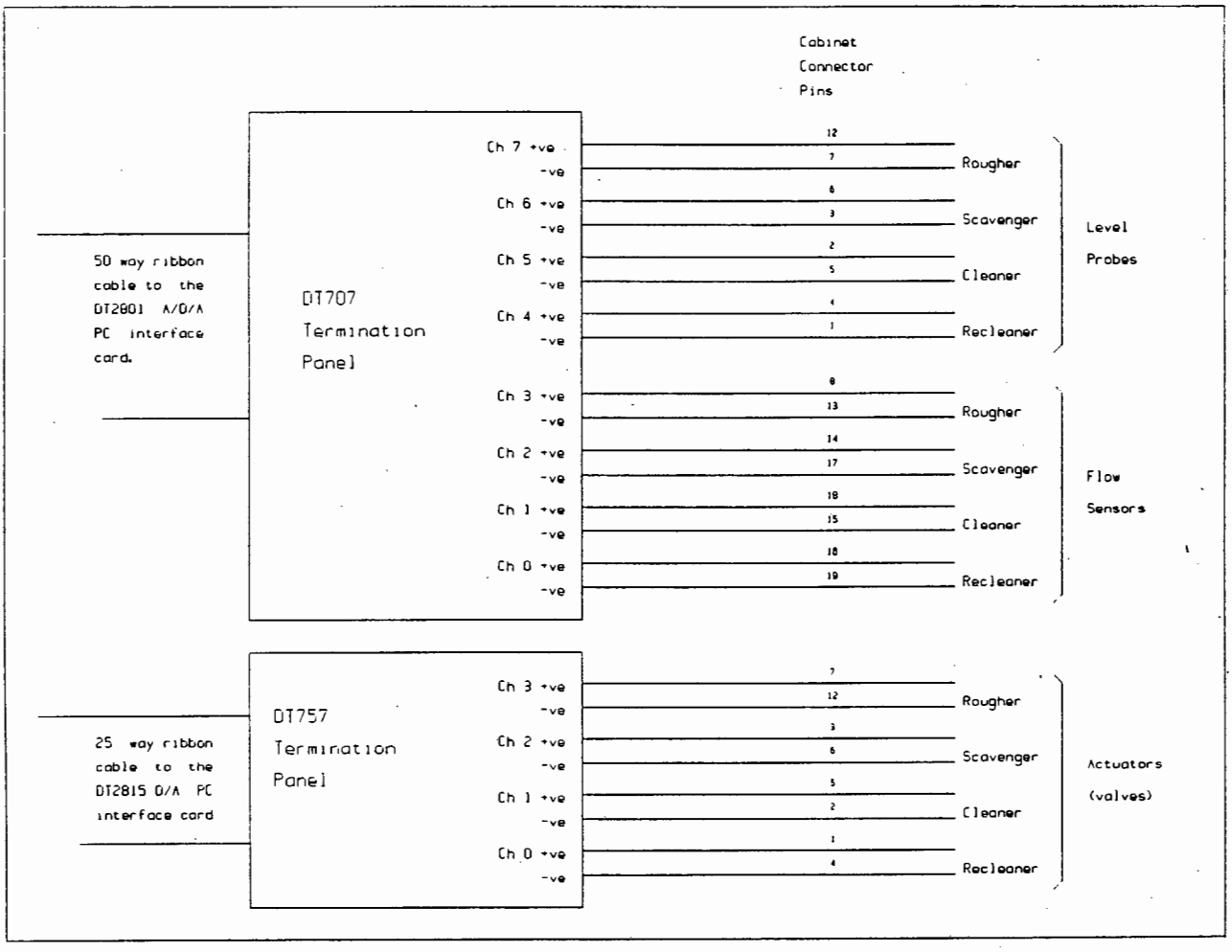


Figure D.4 The Channel and Pin Numbers Used on the A/D/A Interface to the Personal Computer

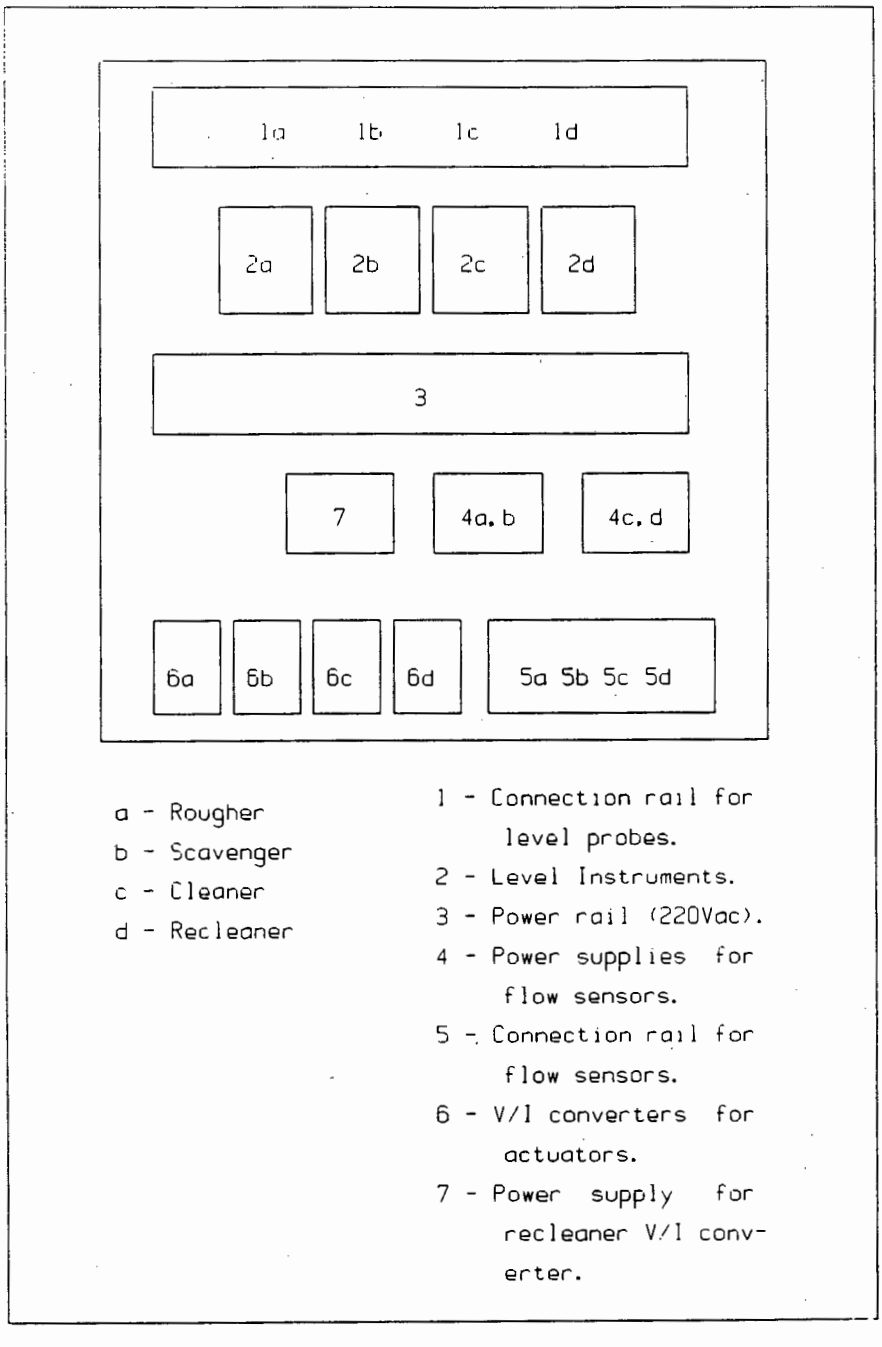


Figure D.5 The Physical Layout of the Instrument Cabinet

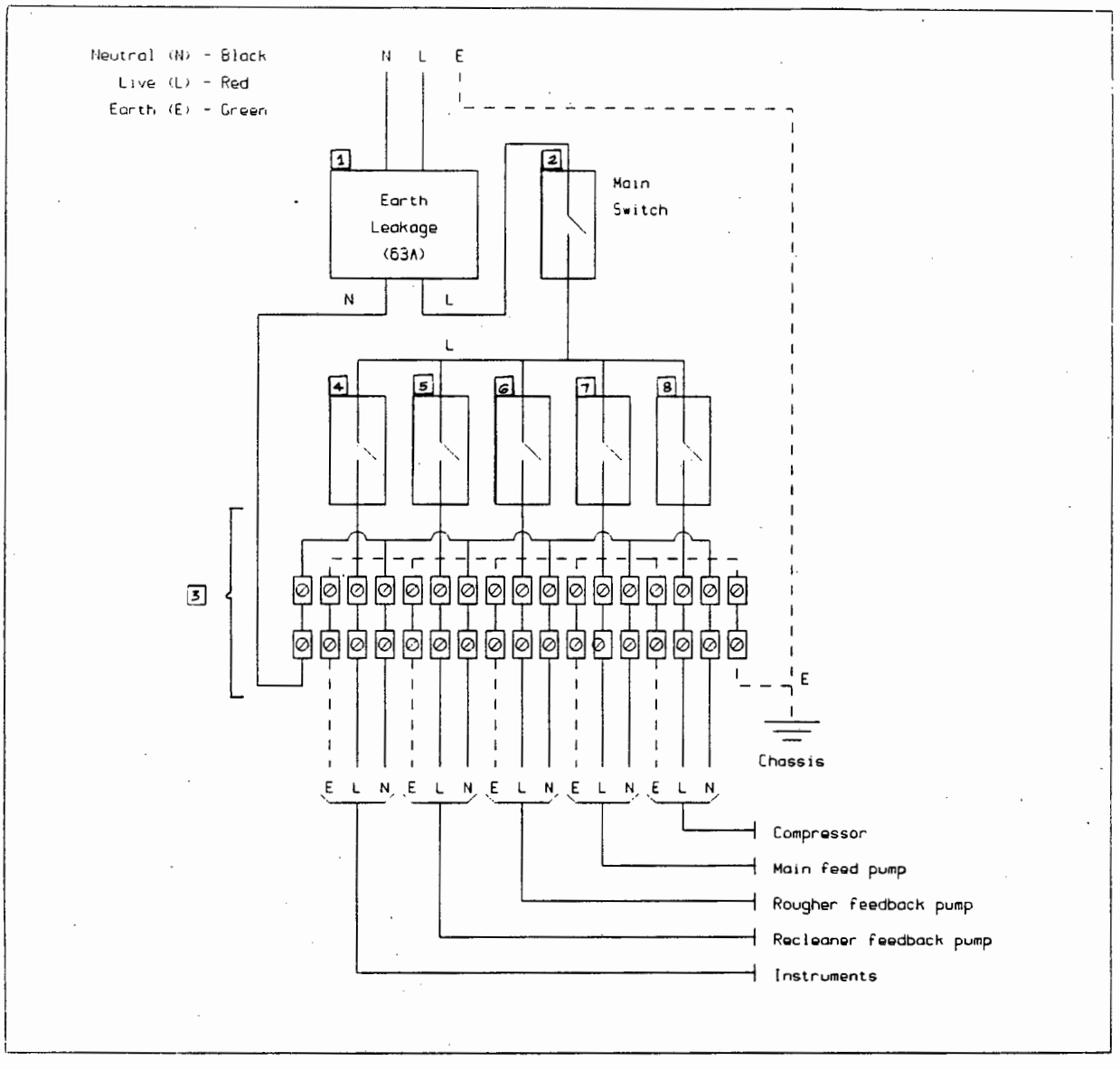
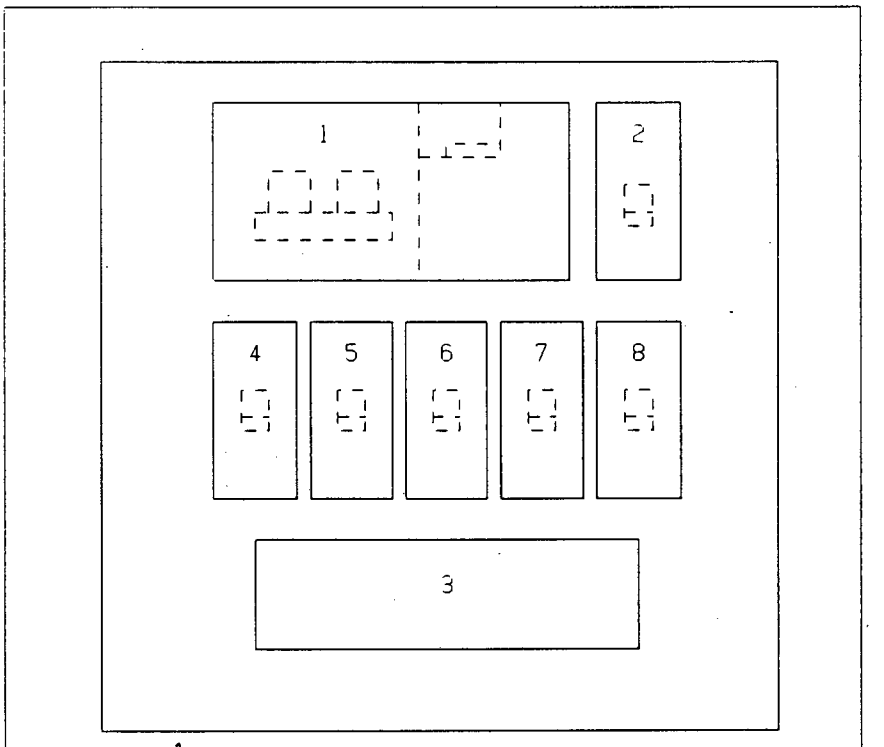


Figure D.6 The Diagrams of the Power Circuits
(The numbers in the blocks refer to parts of the circuit which are represented in the block diagram in figure D.7 overleaf).



1 - Earth leakage unit (63 A).
2 - Main switch (63 A).
3 - Power distribution rail.
4 - Instrumentation power switch.
5 - Recleaner feedback pump switch.
6 - Rougher feedack pump switch.
7 - Main feed pump switch.
8 - Compressor power switch.

Figure D.7 The Physical Layout of the Power Cabinet

Appendix E Flotation Plant Simulator Open Loop Step Tests

The open loop test data presented in this appendix is merely a sample of the many open loop step tests performed on the flotation plant simulator.

This data represents the product level in each cell of the Flotation Plant Simulator and has not been normalised (see chapter 2, section 2.2.1).

The dotted vertical line on each graph represents the point at which one of the valves was stepped.

The data is presented in the following order :-

E.1 Open Loop Step Test - Rougher

The open loop response of the flotation plant simulator to a step on the Rougher tailing valve is shown below in figure E.1.

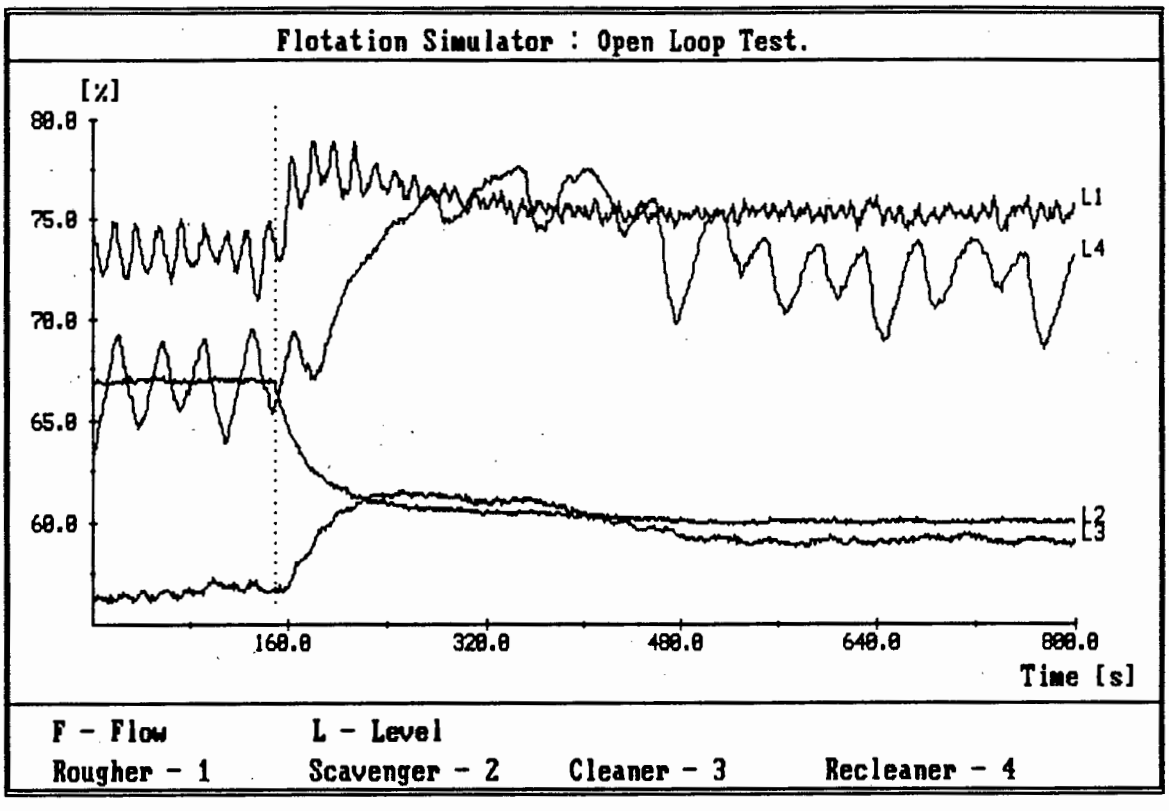


Figure E.1 Open Loop Response to Rougher Tailing Valve Step.

The system configuration for the step test was as follows :-

Initial Valve States : (100% = fully open ; 0% = fully closed)
Rougher : 90 %
Scavenger : 85 %
Cleaner : 85 %
Recleaner : 85 %

Valve Stepped : Rougher tail valve stepped : -15%
(ie. valve closed by 15%)

Sampling Frequency : 2 Hz

E.2 Open Loop Step Test - Scavenger

The open loop response of the flotation plant simulator to a step on the Scavenger tailing valve is shown below in figure E.2.

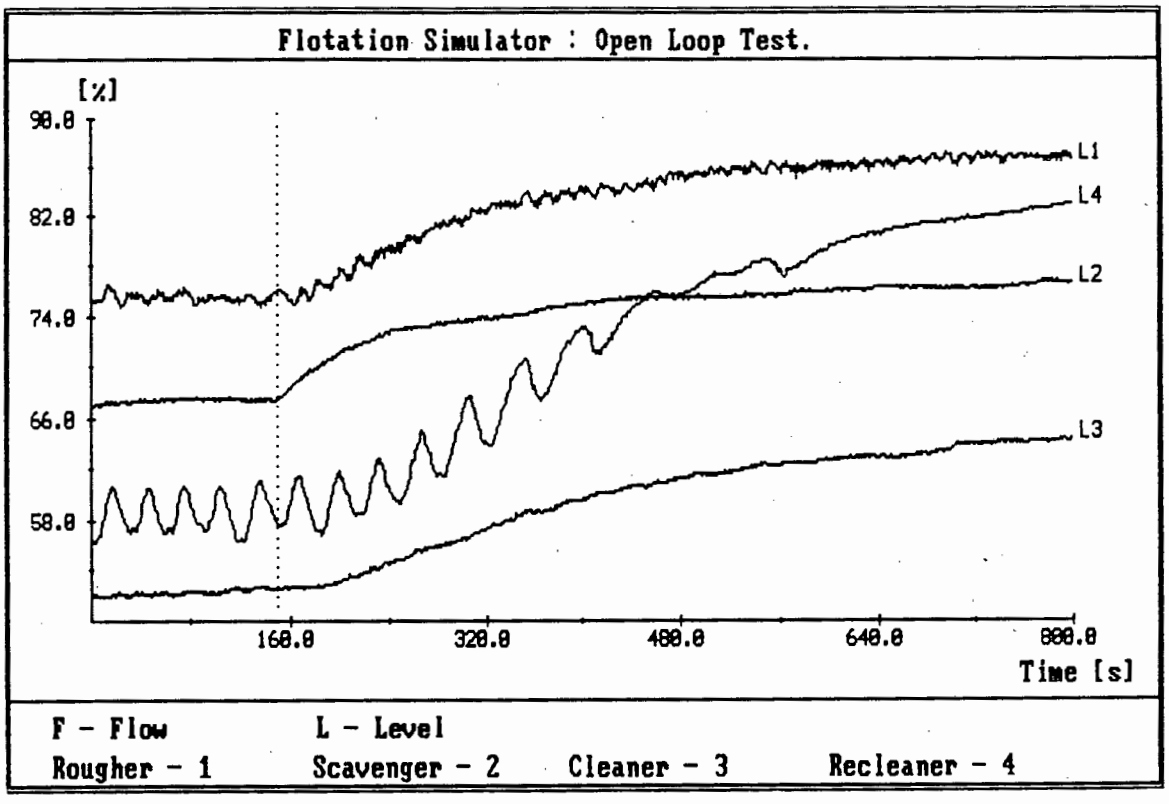


Figure E.2 Open Loop Response to Scavenger Tailing Valve Step.

The system configuration for the step test was as follows :-

Initial Valve States : (100% = fully open ; 0% = fully closed)
- Rougher : 90 %
- Scavenger : 85 %
- Cleaner : 85 %
- Recleaner : 85 %

Valve Stepped : Scavenger tail valve stepped : -5%
(ie. valve closed by 5%)

Sampling Frequency : 2 Hz

E.3 Open Loop Step Test - Cleaner

The open loop response of the flotation plant simulator to a step on the Cleaner tailing valve is shown below in figure E.3.

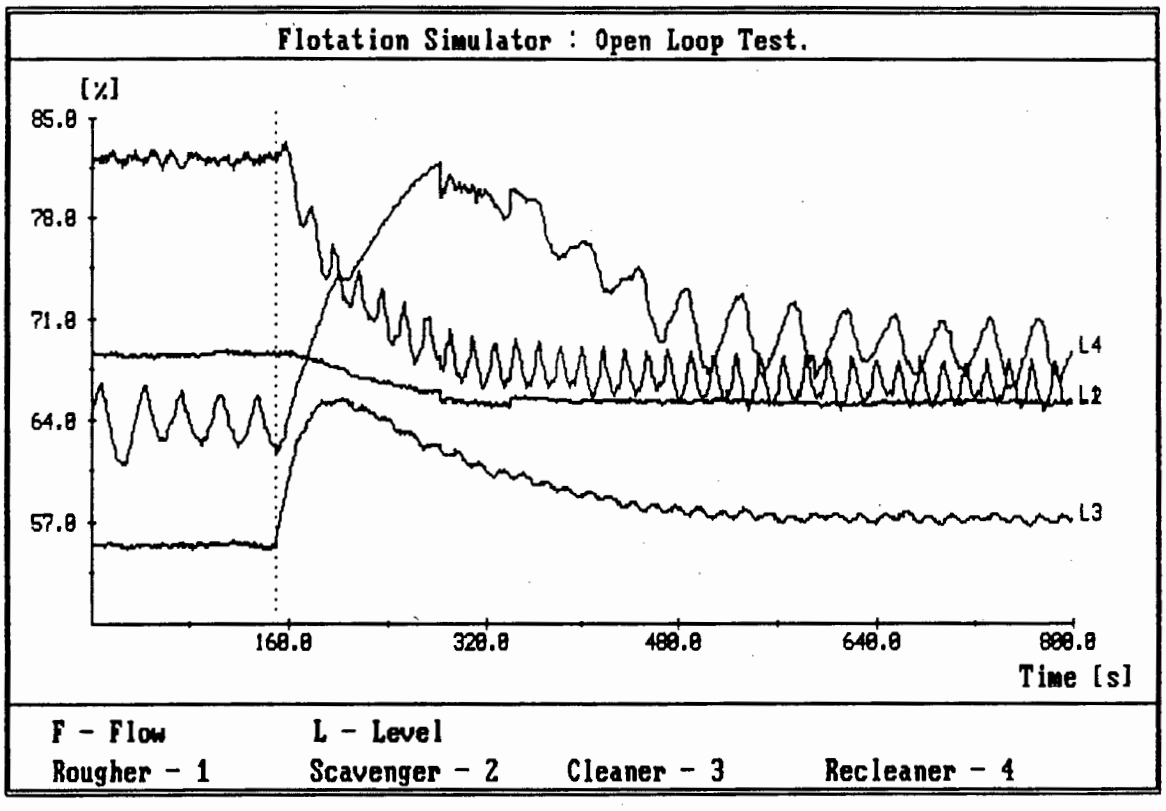


Figure E.3 Open Loop Response to Cleaner Tailing Valve Step.

The system configuration for the step test was as follows :-

Initial Valve States : (100% = fully open ; 0% = fully closed)
Rougher : 90 %
Scavenger : 85 %
Cleaner : 85 %
Recleaner : 85 %

Valve Stepped : Scavenger tail valve stepped : -10%
(ie. valve closed by 10%)

Sampling Frequency : 2 Hz

E.4 Open Loop Step Test - Recleaner

The open loop response of the flotation plant simulator to a step on the Recleaner tailing valve is shown below in figure E.4.

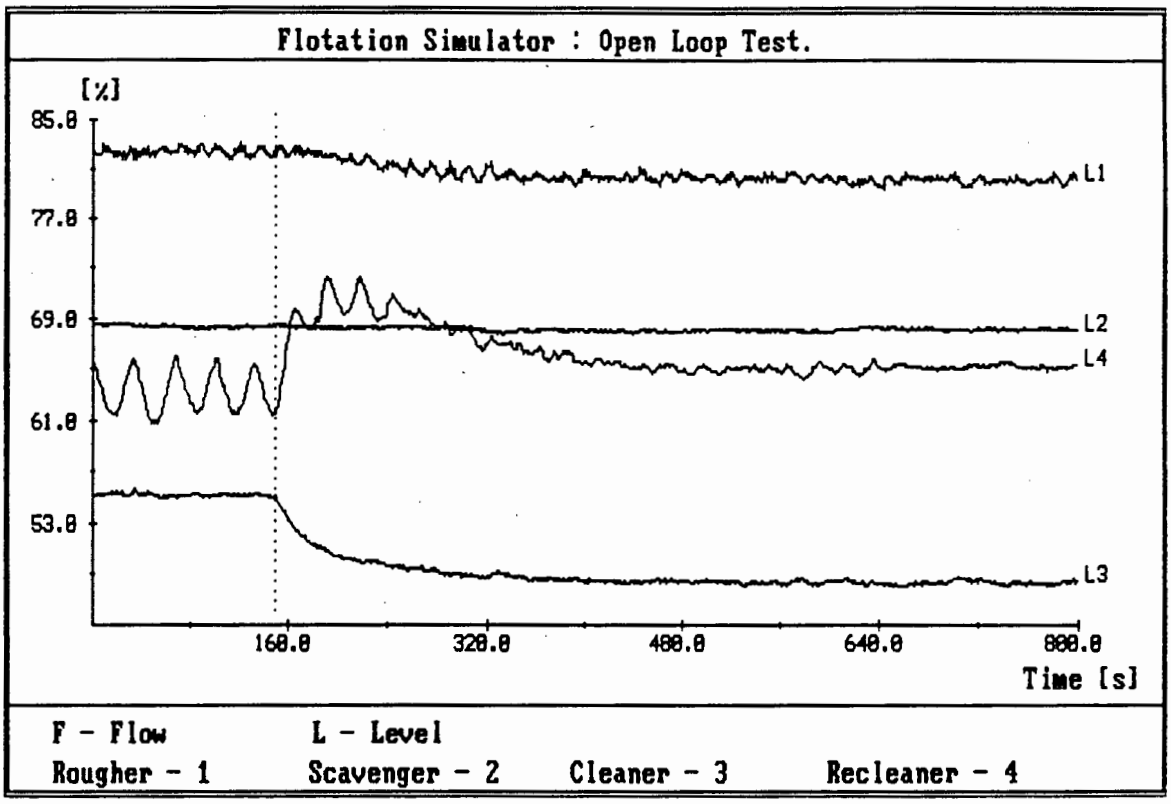


Figure E.4 Open Loop Response to Recleaner Tailing Valve Step.

The system configuration for the step test was as follows :-

Initial Valve States : (100% = fully open ; 0% = fully closed)
- Rougher : 90 %
- Scavenger : 85 %
- Cleaner : 85 %
- Recleaner : 85 %

Valve Stepped : Recleaner tail valve stepped : -25%
(ie. valve closed by 25%)

Sampling Frequency : 2 Hz

Appendix F Data Analysis of Flotation Plant Simulator Open Loop Test Data

This appendix contains the results of the analysis of the data obtained from the open loop step tests performed on the flotation plant simulator. All the data presented has been normalised (see chapter 2, section 2.2.1) and the transfer functions have been optimised using the NELM [2] algorithm (see chapter2, section 2.2.2).

The normalised data presented with each transfer function is the "best" test result obtained to fit the averaged transfer function as explained in chapter 2, section 2.2.3.

The normalised and modeled data is presented in the following order :-

F.1 Comments on the Data and Model

i) *Model kept simple as possible :* The order of a number of the transfer functions presented in this dissertation (figures F.1, F.2, F.6, F.9, F.13, F.14, F.16) appear to be too low when compared to the actual data taken from the Flotation Plant Simulator. This was done in order to keep the transfer function matrix as simple as possible, which was achieved by only modeling the "significant orders" in the responses recorded on the simulator.

ii) *High frequency components not modeled :* A number of the transfer functions presented in this appendix (figures F.1, F.4, F.8, F.9, F.12, F.16) do not appear to model an additional high frequency component. The high frequency component is due to the sump pumps surging.

The surging of the sump pumps can be explained as follows : The pumps drain the sumps faster than product is input to the sumps. While the sump is empty or refilling, the pumps cavitate until the level of product in the sump rises to the point which prevents the pumps from cavitating. The pump will then surge, draining the sump again. The whole process is then repeated.

The "surge-cavitate" cycle of the sump pumps manifests itself in the form of the high frequency components visible in the figures mentioned above. The period and amplitude of the pump "surge-cavitate" cycle depend on the rate of input to the sumps, which in turn is dependent on the level of product in the cells feeding the sumps. Thus as the level of product in the cells vary, the "surge-cavitate" cycle characteristics vary. An example of this

can be seen in figure F.1 where the high frequency component characteristics change as the product levels in the cells change.

The reason for not modeling these high frequency components is mentioned in i) above, ie. the model was kept simple as possible. Another reason was that the high frequency components were not considered as part of the flotation dynamics problem, in other words the sump pumps surging is a problem with the simulator rather than the flotation dynamics.

iii) *Elements that have not reached steady state :* Several of the recorded responses and modeled transfer functions presented in this appendix (figures F.3, F.4, F.5, F.7, F.8, F.12) do not appear to have reached steady state conditions. This abbreviation is for the sake of presentation in that all the graphs have a time span of 550 seconds. The data analysis procedure followed to obtain the transfer functions always proceeded to steady state conditions.

iv) *Anomalies in the data :* Two graphs presented in this appendix (figures F.10, F.12) reveal an anomaly. These irregularities in the data were due to spurious problems in the Flotation Plant Simulator instrumentation. These anomalies were not common and were easily identifiable as irregularities when they occurred.

F.2 Rougher Level - Rougher Tail Valve Step

The normalised Rougher level open loop response to the Rougher tail valve step and the associated simulated transfer function is shown below in figure F.1.

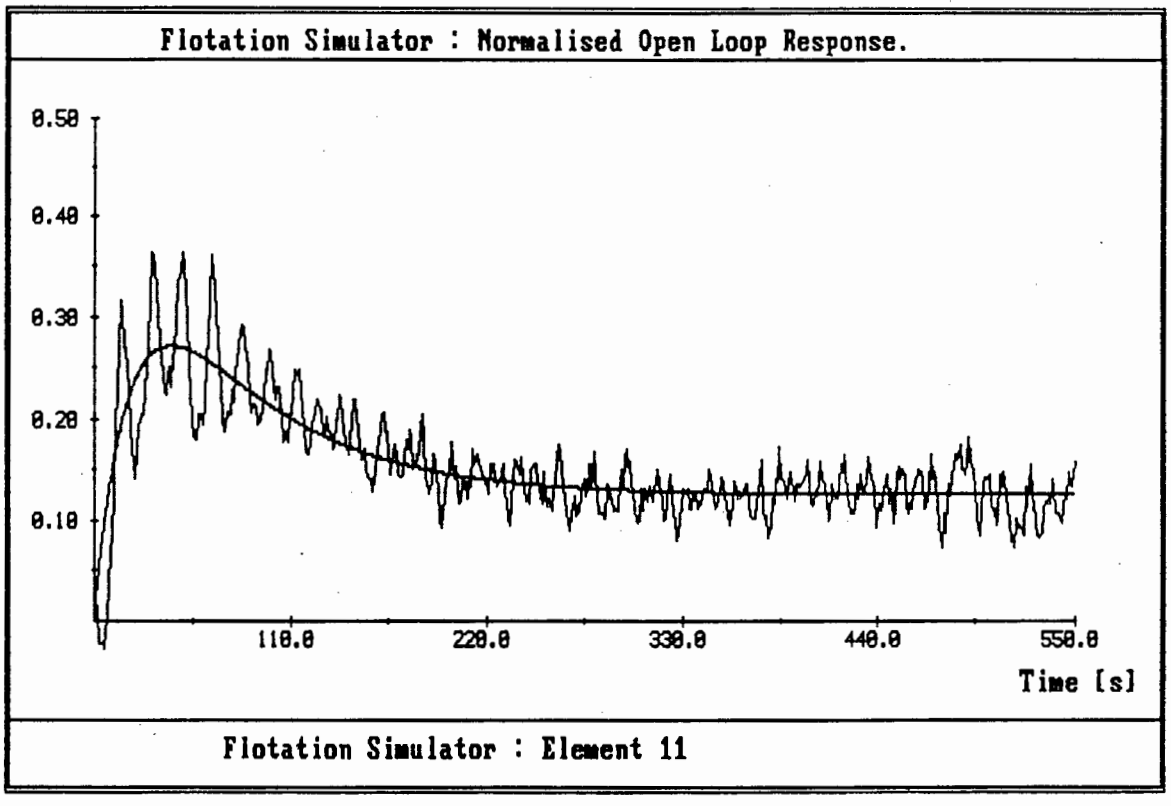


Figure F.1 Open Loop Response and Simulated Transfer Function : $g_{11}(s)$

The transfer function :

$$g_{11} = \frac{(2.000 \times 10^{-2})s + (9.125 \times 10^{-5})}{s^2 + (6.460 \times 10^{-2})s + (7.300 \times 10^{-4})}$$

Delay = 0.0 seconds

F.3 Scavenger Level - Rougher Tail Valve Step

The normalised Scavenger level open loop response to the Rougher tail valve step and the associated simulated transfer function is shown below in figure F.2.

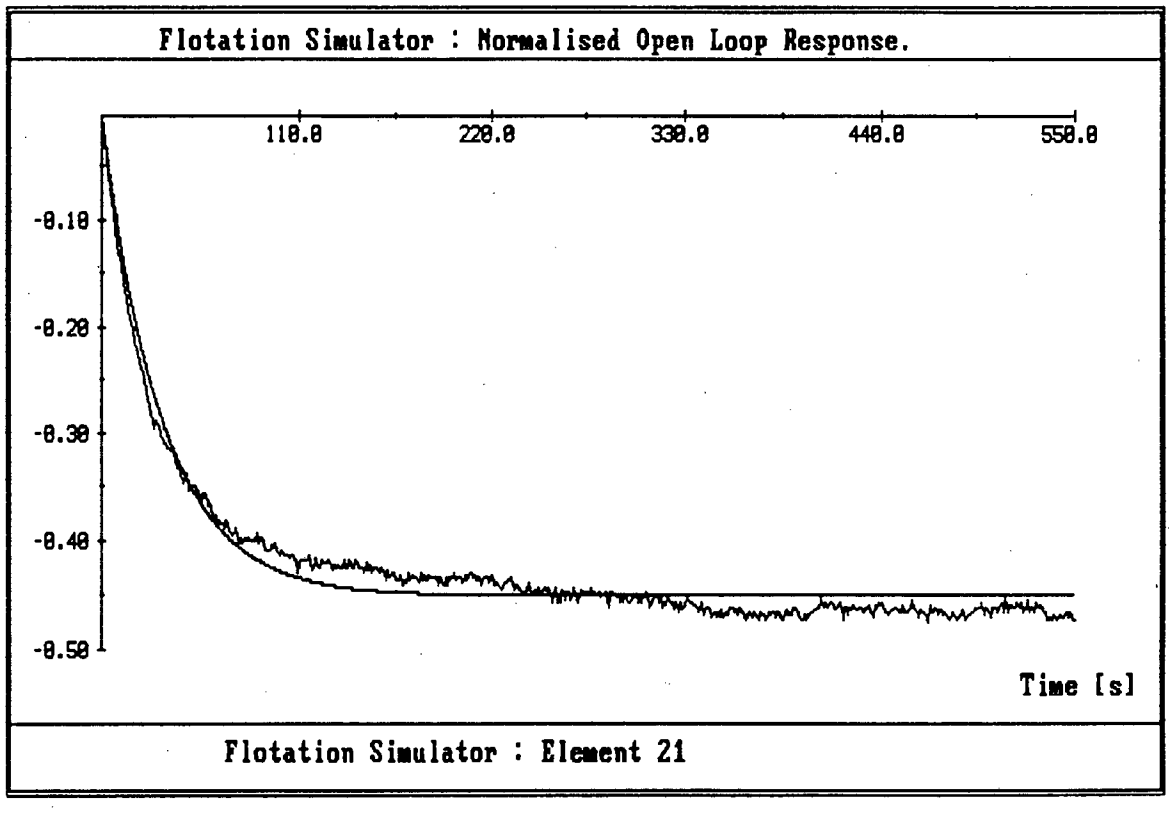


Figure F.2 Open Loop Response and Simulated Transfer Function : $g_{21}(s)$

The transfer function :

$$g_{21} = \frac{(-1.342 \times 10^{-2})}{s + (2.975 \times 10^{-2})}$$

Delay = 0.0 seconds

F.4 Cleaner Level - Rougher Tail Valve Step

The normalised Cleaner level open loop response to the Rougher tail valve step and the associated simulated transfer function is shown below in figure F.3.

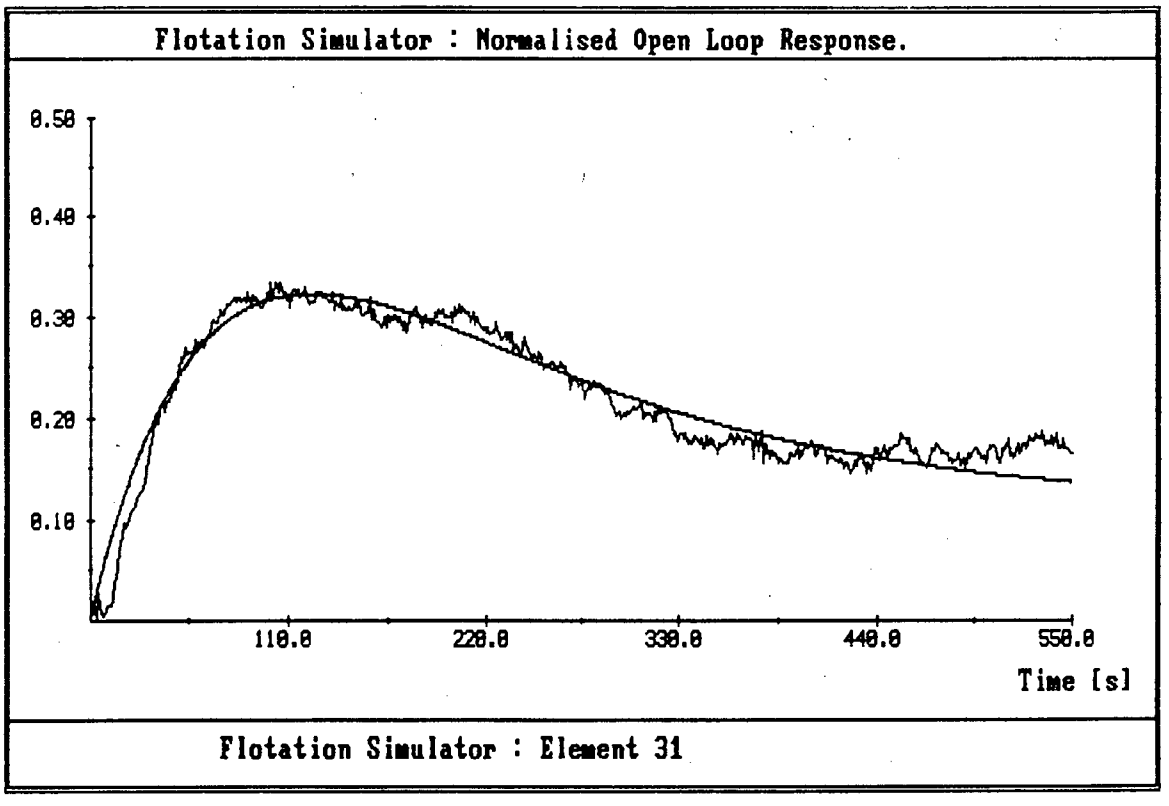


Figure F.3 Open Loop Response and Simulated Transfer Function : $g_{31}(s)$

The transfer function :

$$g_{31} = \frac{(7.610 \times 10^{-3})s + (1.007 \times 10^{-5})}{s^2 + (1.949 \times 10^{-2})s + (8.774 \times 10^{-5})}$$

Delay = 0.0 seconds

F.5 Recleaner Level - Rougher Tail Valve Step

The normalised Recleaner level open loop response to the Rougher tail valve step and the associated simulated transfer function is shown below in figure F.4.

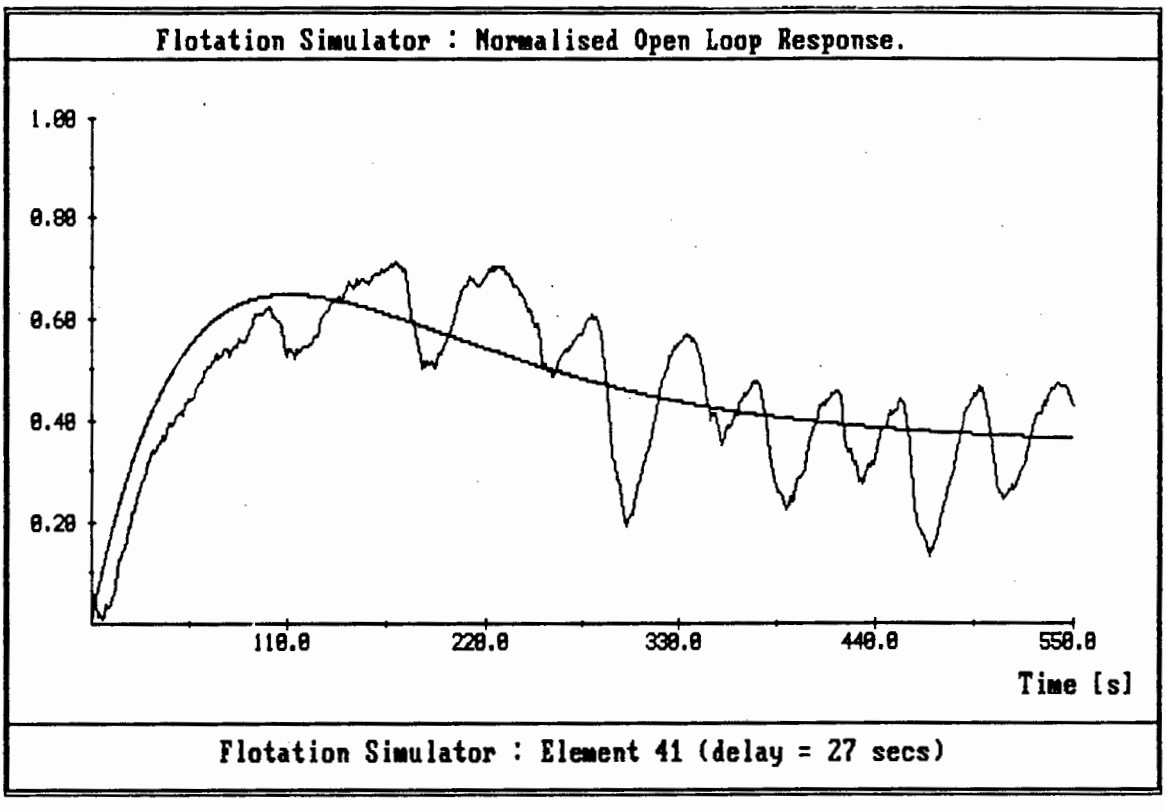


Figure F.4 Open Loop Response and Simulated Transfer Function : $g_{41}(s)$

The transfer function :

$$g_{41} = \frac{(1.750 \times 10^{-2})s + (4.725 \times 10^{-5})}{s^2 + (2.400 \times 10^{-2})s + (1.350 \times 10^{-5})}$$

Delay = 27.0 seconds

F.6 Rougher Level - Scavenger Tail Valve Step

The normalised Rougher level open loop response to the Scavenger tail valve step and the associated simulated transfer function is shown below in figure F.5.

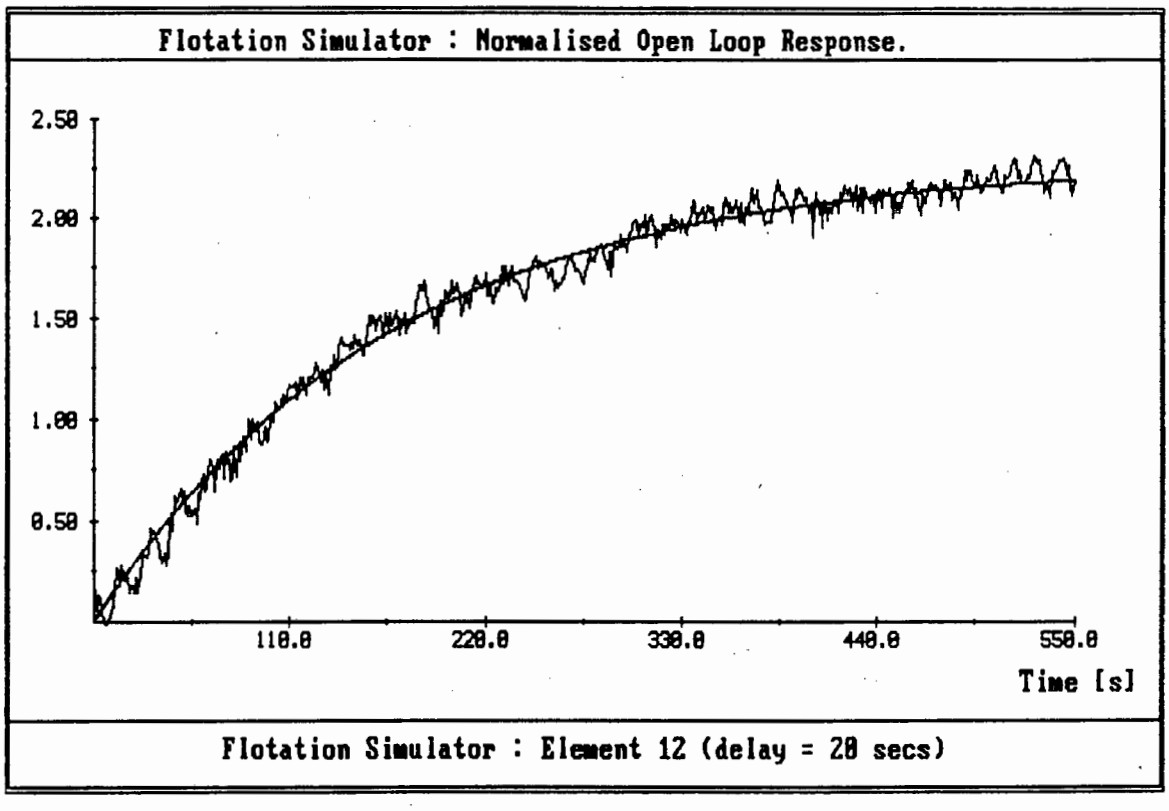


Figure F.5 Open Loop Response and Simulated Transfer Function : $g_{12}(s)$

The transfer function :

$$g_{12} = \frac{(1.350 \times 10^{-2})}{s + (5.930 \times 10^{-3})}$$

Delay = 20.0 seconds

F.7 Scavenger Level - Scavenger Tail Valve Step

The normalised Scavenger level open loop response to the Scavenger tail valve step and the associated simulated transfer function is shown below in figure F.6.

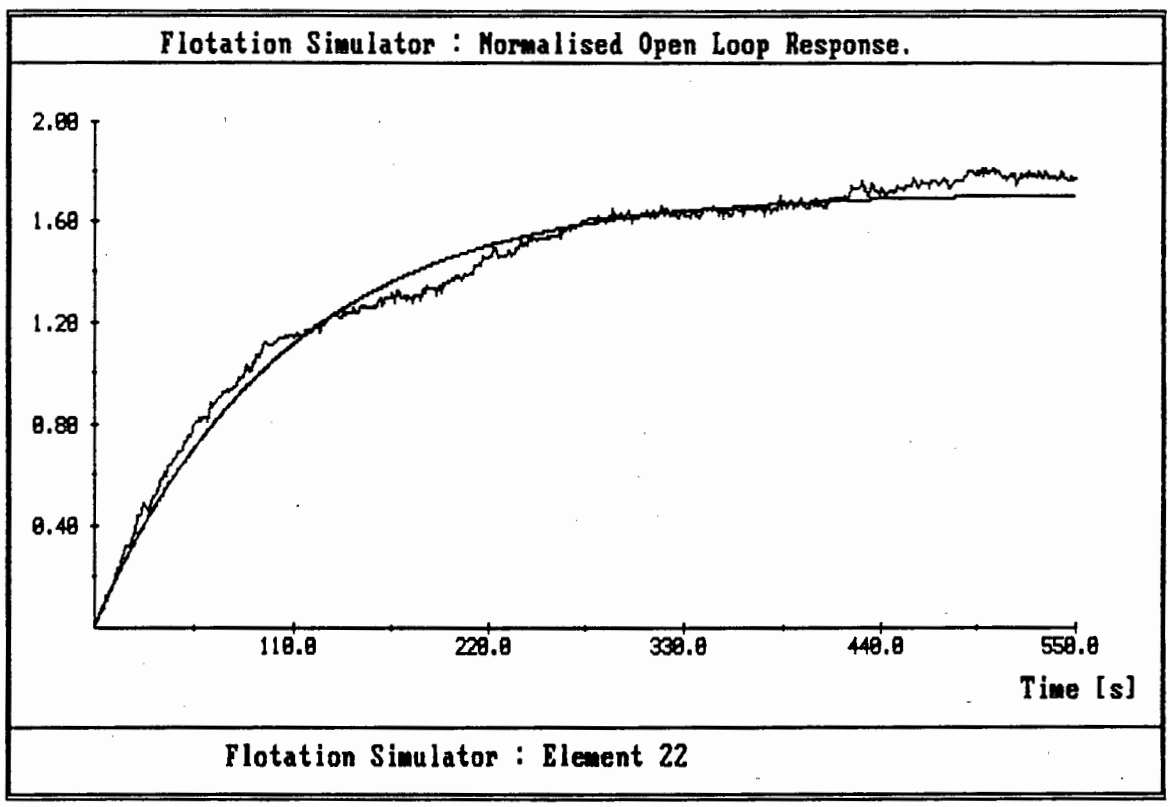


Figure F.6 Open Loop Response and Simulated Transfer Function : $g_{22}(s)$

The transfer function :

$$g_{22} = \frac{(1.640 \times 10^{-2})}{s + (9.574 \times 10^{-3})}$$

Delay = 0.0 seconds

F.8 Cleaner Level - Scavenger Tail Valve Step

The normalised Cleaner level open loop response to the Scavenger tail valve step and the associated simulated transfer function is shown below in figure F.7.

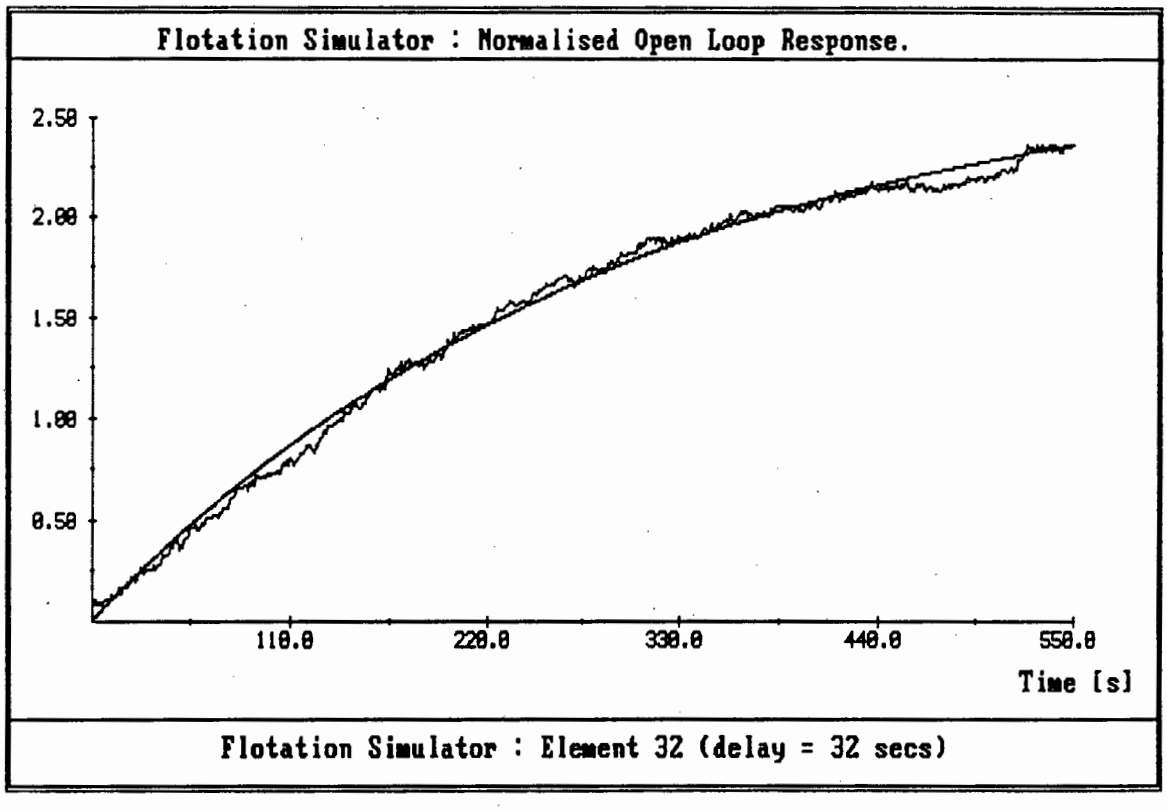


Figure F.7 Open Loop Response and Simulated Transfer Function : $g_{32}(s)$

The transfer function :

$$g_{32} = \frac{(9.388 \times 10^{-3})}{s + (3.356 \times 10^{-3})}$$

Delay = 32.0 seconds

F.9 Recleaner Level - Scavenger Tail Valve Step

The normalised Recleaner level open loop response to the Scavenger tail valve step and the associated simulated transfer function is shown below in figure F.8.

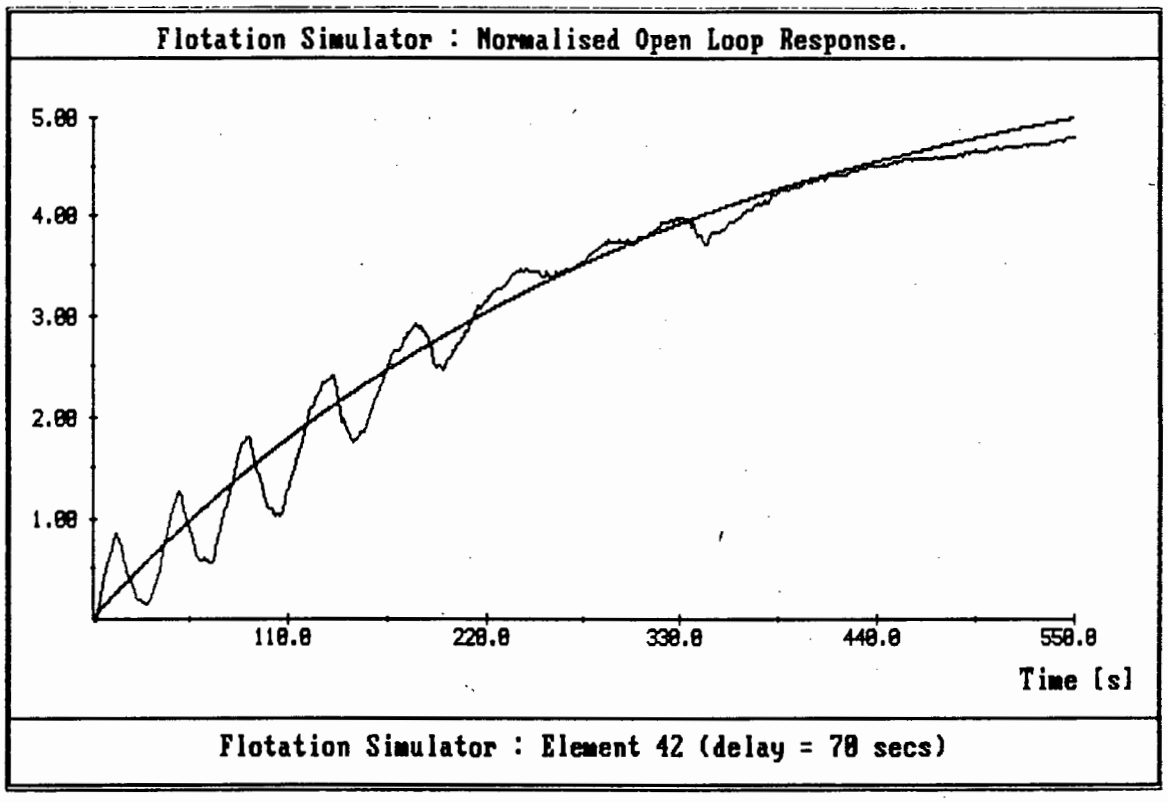


Figure F.8 Open Loop Response and Simulated Transfer Function : $g_{42}(s)$

The transfer function :

$$g_{42} = \frac{(1.917 \times 10^{-2})}{s + (3.176 \times 10^{-3})}$$

Delay = 70.0 seconds

F.10 Rougher Level - Cleaner Tail Valve Step

The normalised Rougher level open loop response to the Cleaner tail valve step and the associated simulated transfer function is shown below in figure F.9.

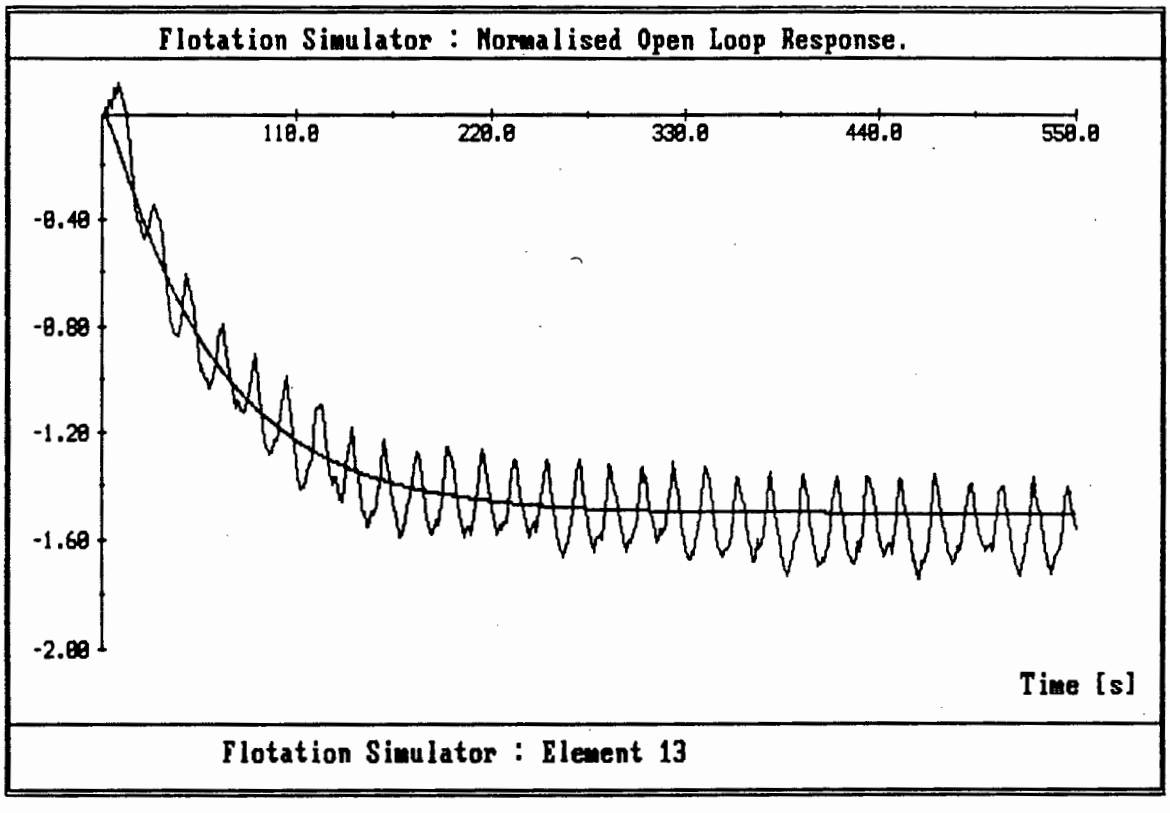


Figure F.9 Open Loop Response and Simulated Transfer Function : $g_{13}(s)$

The transfer function :

$$g_{13} = \frac{(-6.651 \times 10^{-3})}{s^2 + (2.948 \times 10^{-1})s + (4.434 \times 10^{-3})}$$

Delay = 0.0 seconds

F.11 Scavenger Level - Cleaner Tail Valve Step

The normalised Scavenger level open loop response to the Cleaner tail valve step and the associated simulated transfer function is shown below in figure F.10.

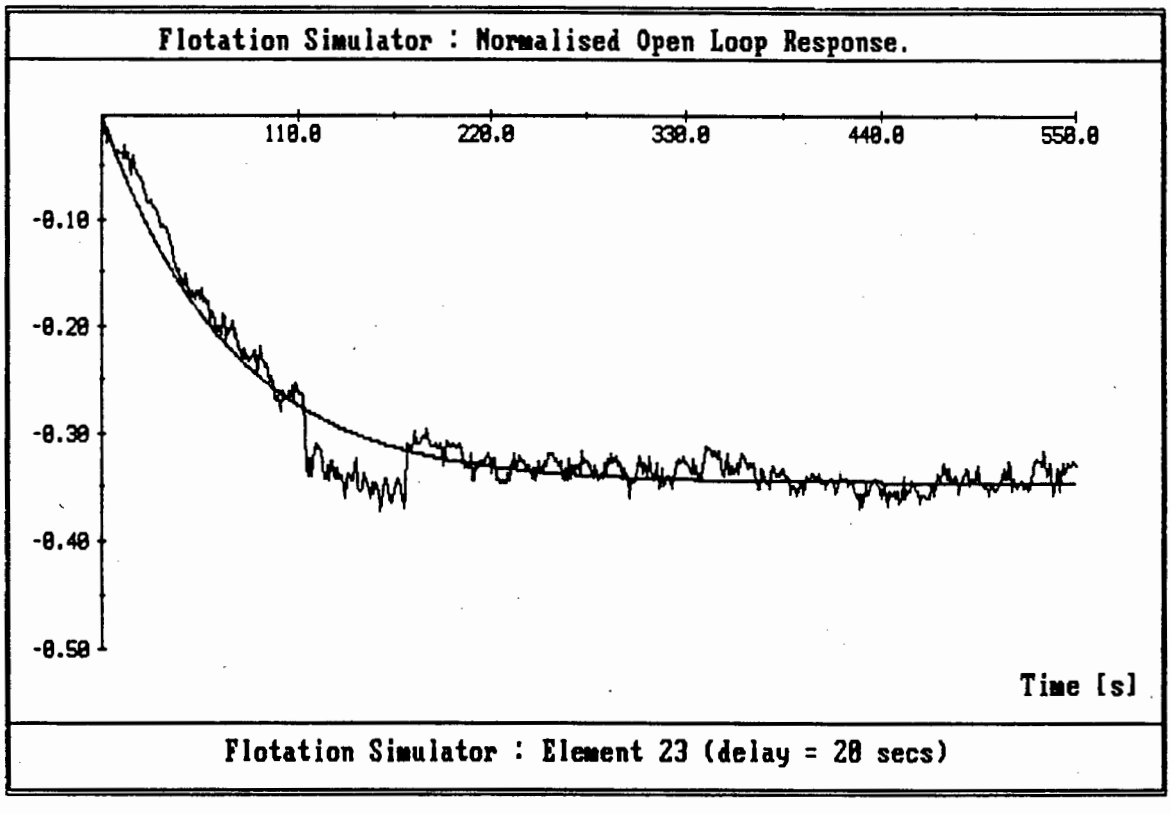


Figure F.10 Open Loop Response and Simulated Transfer Function : $g_{23}(s)$

The transfer function :

$$g_{23} = \frac{(-4.890 \times 10^{-3})}{s + (1.418 \times 10^{-2})}$$

Delay = 20.0 seconds

F.12 Cleaner Level - Cleaner Tail Valve Step

The normalised Cleaner level open loop response to the Cleaner tail valve step and the associated simulated transfer function is shown below in figure F.11.

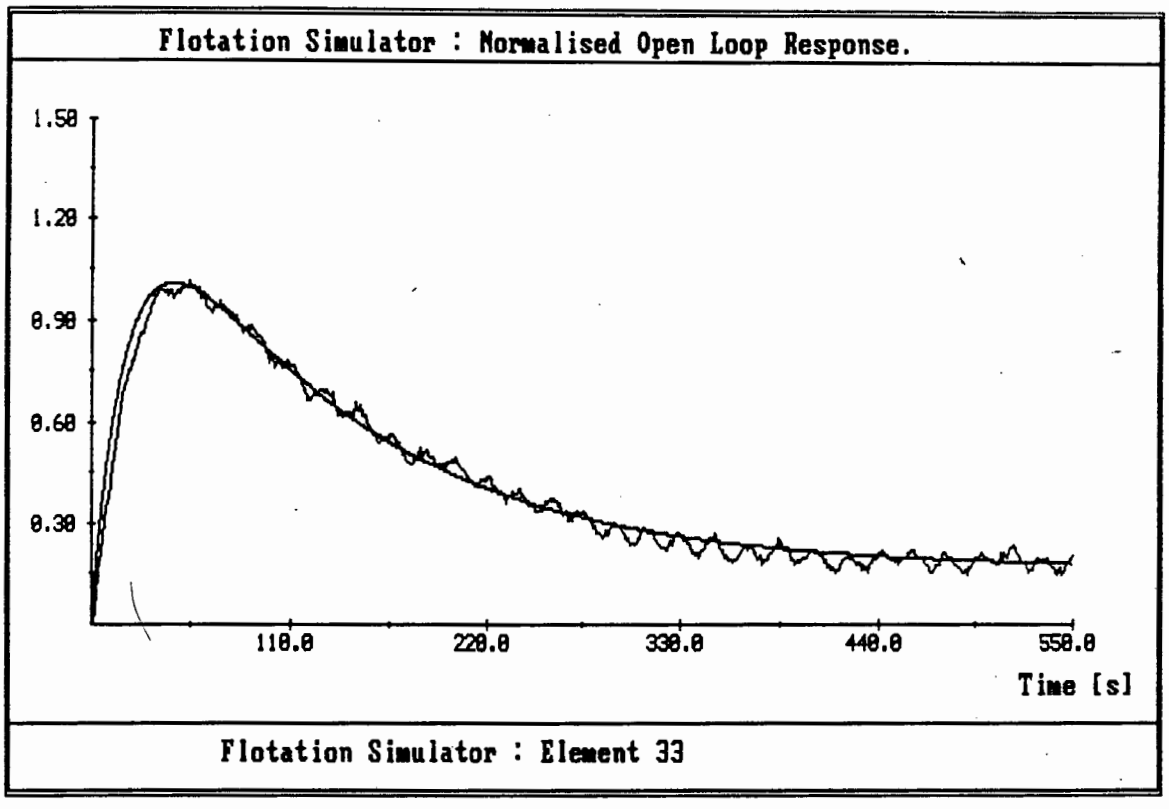


Figure F.11 Open Loop Response and Simulated Transfer Function : $g_{33}(s)$

The transfer function :

$$g_{33} = \frac{(7.000 \times 10^{-2})s + (6.560 \times 10^{-5})}{s^2 + (5.820 \times 10^{-2})s + (4.100 \times 10^{-4})}$$

Delay = 0.0 seconds

F.13 Recleaner Level - Cleaner Tail Valve Step

The normalised Recleaner level open loop response to the Cleaner tail valve step and the associated simulated transfer function is shown below in figure F.12.

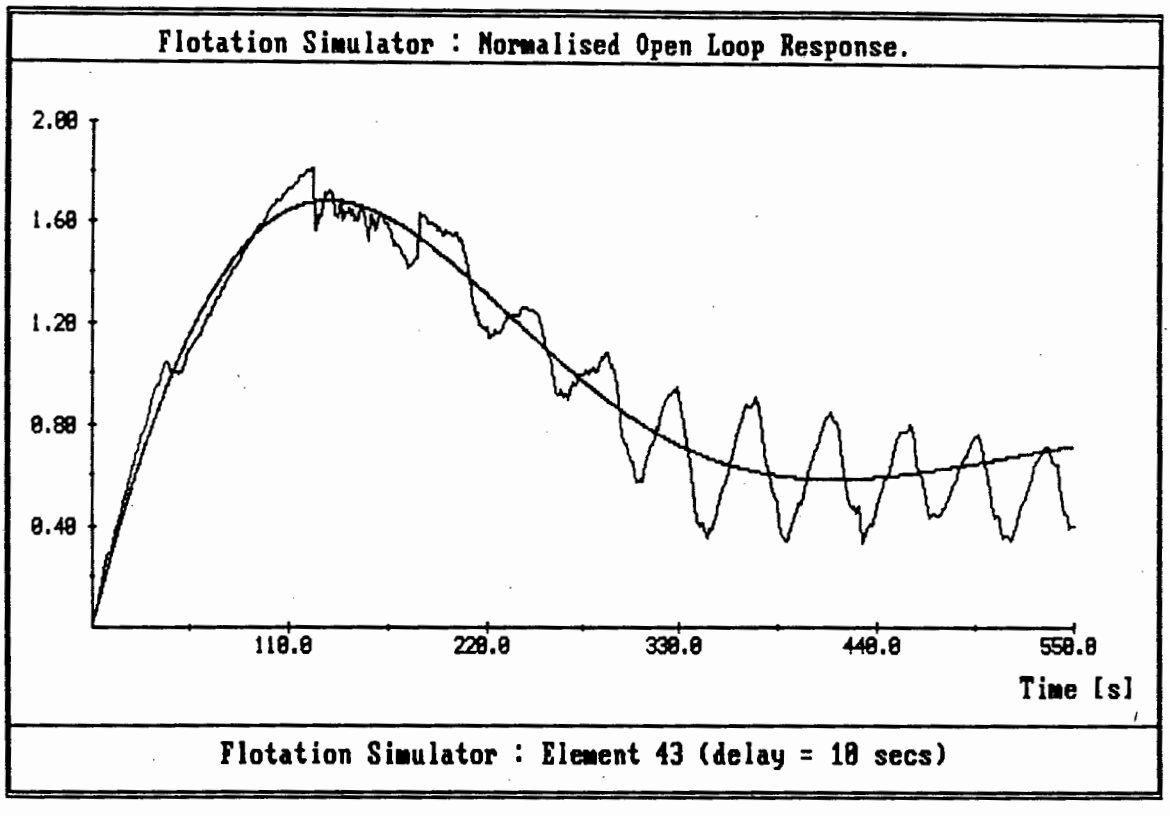


Figure F.12 Open Loop Response and Simulated Transfer Function : $g_{43}(s)$

The transfer function :

$$g_{43} = \frac{(2.654 \times 10^{-2})s + (1.182 \times 10^{-4})}{s^2 + (1.121 \times 10^{-2})s + (1.540 \times 10^{-4})}$$

Delay = 10.0 seconds

F.14 Rougher Level - Recleaner Tail Valve Step

The normalised Rougher level open loop response to the Recleaner tail valve step and the associated simulated transfer function is shown below in figure F.13.

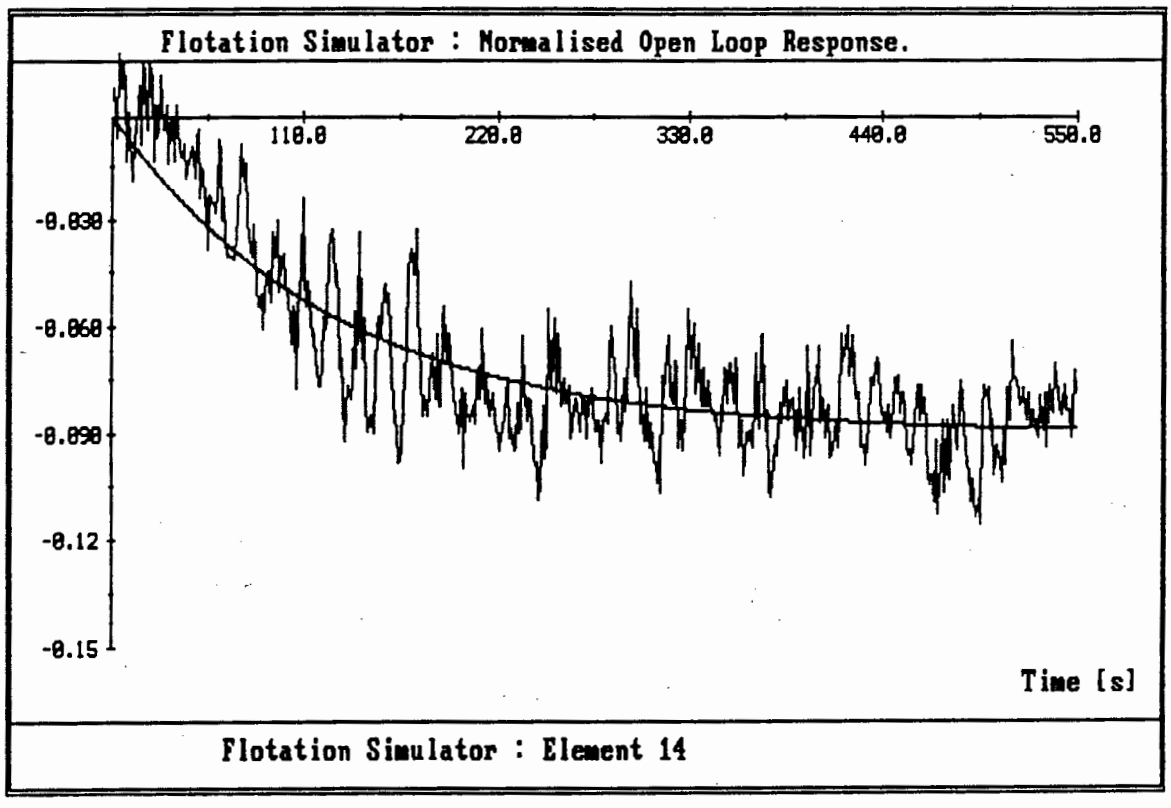


Figure F.13 Open Loop Response and Simulated Transfer Function : $g_{14}(s)$

The transfer function :

$$g_{14} = \frac{(-7.060 \times 10^{-4})}{s + (7.865 \times 10^{-3})}$$

Delay = 0.0 seconds

F.15 Scavenger Level - Recleaner Tail Valve Step

The normalised Scavenger level open loop response to the Recleaner tail valve step and the associated simulated transfer function is shown below in figure F.14.

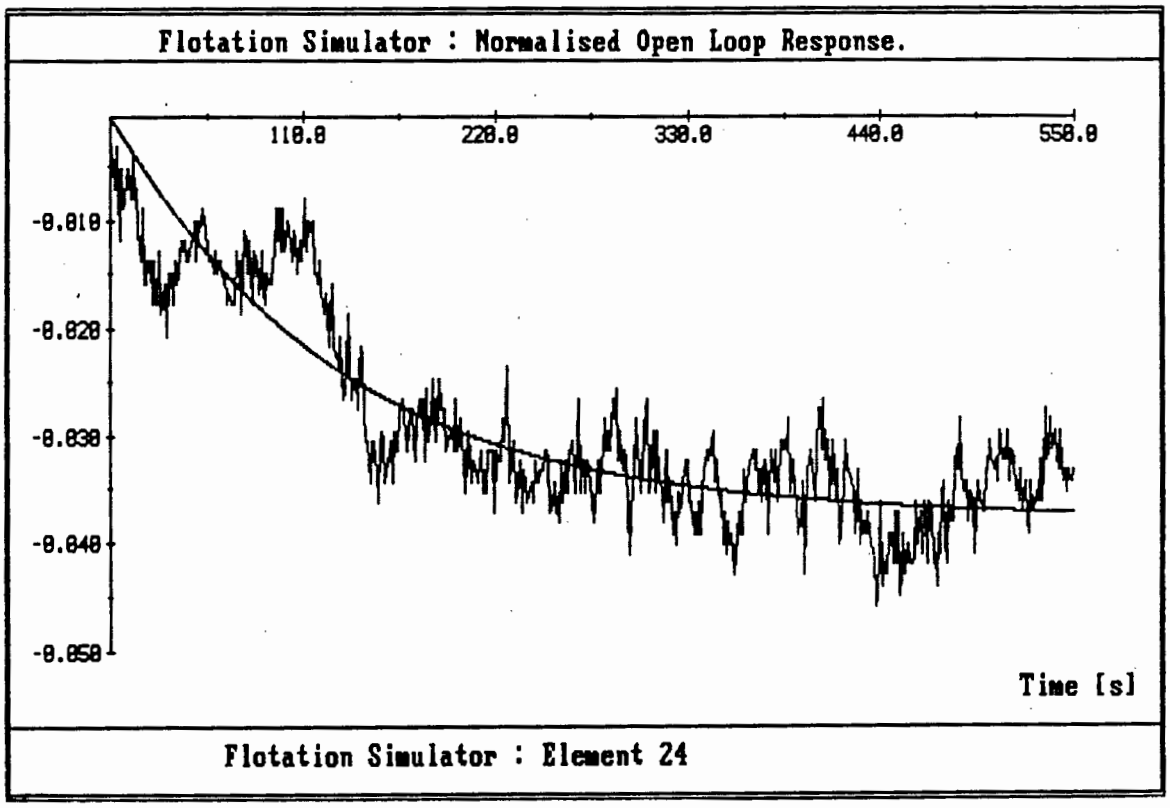


Figure F.14 Open Loop Response and Simulated Transfer Function : $g_{24}(s)$

The transfer function :

$$g_{24} = \frac{(-2.860 \times 10^{-4})}{s + (7.580 \times 10^{-3})}$$

Delay = 0.0 seconds

F.16 Cleaner Level - Recleaner Tail Valve Step

The normalised Cleaner level open loop response to the Recleaner tail valve step and the associated simulated transfer function is shown below in figure F.15.

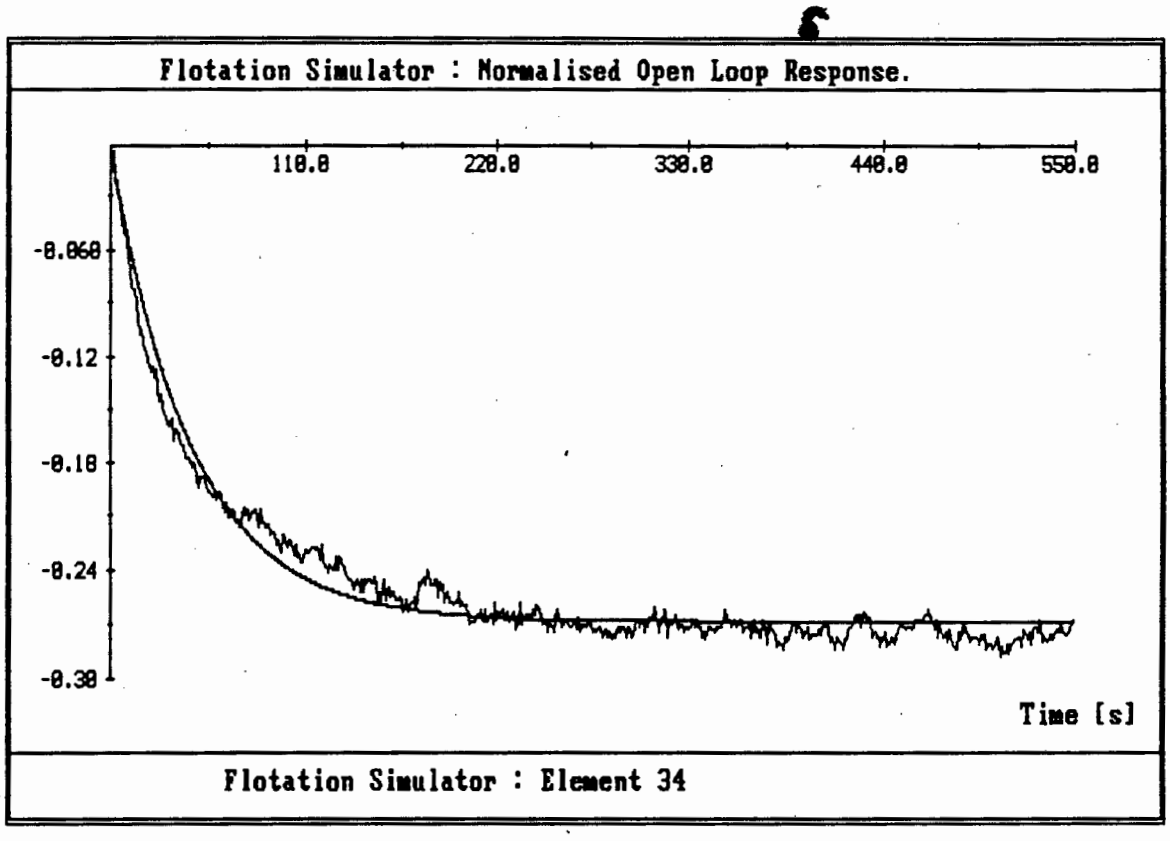


Figure F.15 Open Loop Response and Simulated Transfer Function : $g_{34}(s)$

The transfer function :

$$g_{34} = \frac{(-5.871 \times 10^{-3})}{s + (2.190 \times 10^{-2})}$$

Delay = 0.0 seconds

F.17 Recleaner Level - Recleaner Tail Valve Step

The normalised Recleaner level open loop response to the Recleaner tail valve step and the associated simulated transfer function is shown below in figure F.16.

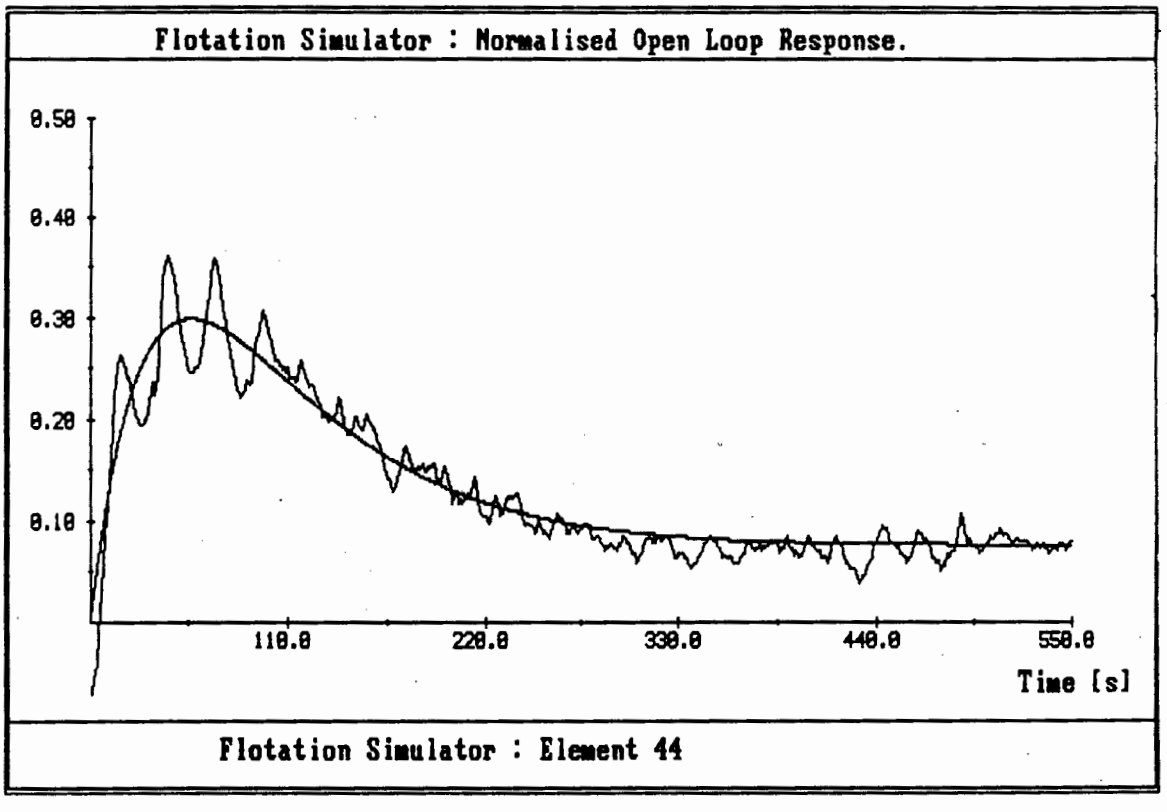


Figure F.16 Open Loop Response and Simulated Transfer Function : $g_{44}(s)$

The transfer function :

$$g_{44} = \frac{(1.500 \times 10^{-2})s + (2.813 \times 10^{-5})}{s^2 + (4.000 \times 10^{-2})s + (3.750 \times 10^{-4})}$$

Delay = 0.0 seconds

Appendix G Flotation Plant Simulator Transfer Function Matrix

The system matrix of transfer functions is given as

$$\begin{bmatrix} y_1 \\ y_2 \\ y_3 \\ y_4 \end{bmatrix} = \begin{bmatrix} g_{11}(s) & g_{12}(s) & g_{13}(s) & g_{14}(s) \\ g_{21}(s) & g_{22}(s) & g_{23}(s) & g_{24}(s) \\ g_{31}(s) & g_{32}(s) & g_{33}(s) & g_{34}(s) \\ g_{41}(s) & g_{42}(s) & g_{43}(s) & g_{44}(s) \end{bmatrix} \begin{bmatrix} u_1 \\ u_2 \\ u_3 \\ u_4 \end{bmatrix}$$

where :

	[Units]	
y_1 = Rougher Level	[bits]	(12 bit A/D)
y_2 = Scavenger Level	[bits]	(12 bit A/D)
y_3 = Cleaner Level	[bits]	(12 bit A/D)
y_4 = Recleaner Level	[bits]	(12 bit A/D)
u_1 = Rougher Valve	[bits]	(12 bit D/A)
u_2 = Scavenger Valve	[bits]	(12 bit D/A)
u_3 = Cleaner Valve	[bits]	(12 bit D/A)
u_4 = Recleaner Valve	[bits]	(12 bit D/A)

NOTE : All the time constants and time delays are given in seconds.

Element	Transfer Function	Delay
g_{11}	$\frac{(2.000 \times 10^{-2})S + (9.125 \times 10^{-5})}{S^2 + (6.460 \times 10^{-2})S + (7.300 \times 10^{-4})}$ Zeros : -4.563×10^{-3} Poles : -1.460×10^{-2} -5.000×10^{-2}	0.0 [sec]
g_{12}	$\frac{(1.350 \times 10^{-2})}{S + (5.930 \times 10^{-3})}$ Zeros : Poles : -5.93×10^{-3}	20.0 [sec]
g_{13}	$\frac{(-6.651 \times 10^{-3})}{S^2 + (2.948 \times 10^{-1})S + (4.434 \times 10^{-3})}$ Zeros : Poles : -1.590×10^{-2} -2.789×10^{-1}	0.0 [sec]
g_{14}	$\frac{(-7.060 \times 10^{-4})}{S + (7.865 \times 10^{-3})}$ Zeros : Poles : -7.865×10^{-3}	0.0 [sec]

Figure G.1 Flotation Plant Simulator Model.

Element	Transfer Function	Delay
g_{21}	$\frac{(-1.342 \times 10^{-2})}{S + (2.975 \times 10^{-2})}$ Zeros : Poles : -2.975×10^{-2}	0 .0 [sec]
g_{22}	$\frac{(1.640 \times 10^{-2})}{S + (9.574 \times 10^{-3})}$ Zeros : Poles : -9.574×10^{-3}	0.0 [sec]
g_{23}	$\frac{(-4.890 \times 10^{-3})}{S + (1.418 \times 10^{-2})}$ Zeros : Poles : -1.418×10^{-2}	20.0 [sec]
g_{24}	$\frac{(-2.860 \times 10^{-4})}{S + (7.580 \times 10^{-3})}$ Zeros : Poles : -7.580×10^{-3}	0.0 [sec]

Figure G.1 Flotation Plant Simulator Model (contd.)

Element	Transfer Function	Delay
g_{31}	$\frac{(7.610 \times 10^{-3})S + (1.007 \times 10^{-5})}{S^2 + (1.949 \times 10^{-2})S + (8.774 \times 10^{-5})}$ Zeros : -1.323×10^{-3} Poles : -7.100×10^{-3} -1.240×10^{-2}	0.0 [sec]
g_{32}	$\frac{(9.388 \times 10^{-3})}{S + (3.356 \times 10^{-3})}$ Zeros : Poles : -3.356×10^{-3}	32.0 [sec]
g_{33}	$\frac{(7.000 \times 10^{-2})S + (6.560 \times 10^{-5})}{S^2 + (5.820 \times 10^{-2})S + (4.100 \times 10^{-4})}$ Zeros : -9.370×10^{-4} Poles : -8.200×10^{-3} -5.000×10^{-2}	0.0 [sec]
g_{34}	$\frac{(-5.871 \times 10^{-3})}{S + (2.190 \times 10^{-2})}$ Zeros : Poles : -2.190×10^{-2}	0.0 [sec]

Figure G.1 Flotation Plant Simulator Model (contd.)

Element	Transfer Function	Delay
g_{41}	$\frac{(1.750 \times 10^{-2})S + (4.725 \times 10^{-5})}{S^2 + (2.400 \times 10^{-2})S + (1.350 \times 10^{-4})}$ Zeros : -2.700×10^{-3} Poles : -9.000×10^{-3} -1.500×10^{-2}	27.0 [sec]
g_{42}	$\frac{(1.917 \times 10^{-2})}{S + (3.176 \times 10^{-3})}$ Zeros : Poles : -3.176×10^{-3}	70.0 [sec]
g_{43}	$\frac{(2.654 \times 10^{-2})S + (1.182 \times 10^{-4})}{S^2 + (1.121 \times 10^{-2})S + (1.540 \times 10^{-4})}$ Zeros : -4.454×10^{-3} Poles : $[(-5.600 \times 10^{-3})$ $+- j(1.110 \times 10^{-2})]$	10.0 [sec]
g_{44}	$\frac{(1.500 \times 10^{-2})S + (2.813 \times 10^{-5})}{S^2 + (4.000 \times 10^{-2})S + (3.750 \times 10^{-4})}$ Zeros : -1.875×10^{-3} Poles : -1.500×10^{-2} -2.500×10^{-2}	0.0 [sec]

Figure G.1 Flotation Plant Simulator Model (contd.)

Appendix H Characteristic Loci CAD System Subroutine List

H.1 CAD System Routines

H.1.2 CAD Entry Point and Initialising Routines

Subroutine Name	Source Filename	Description
LOCI	LOCI.FOR	Entry point for the Characteristic Loci CAD system.
INISYS	INISYS.FOR	To initialise system variables.
GETSYS	GETSYS.FOR	To load the 'loci.sys' configuration file.
INMENU	INMENU.FOR	To initialise the menus and screens.
NAMES	NAMES.FOR	To drive the system names and titles menu.
PROJNM	NAMES.FOR	Enables the user to edit the project titles.
SYSNM	NAMES.FOR	Enables the user to edit the system I/O names.
EDTNMS	NAMES.FOR	Enables the user to edit either the input or output names.
PARPOS	NAMES.FOR	To calculate highlight position for system names.

H.1.2 Project Routines

Subroutine Name	Source Filename	Description
PROJ	PROJ.FOR	To drive the menu for project information management.
SAVPRJ	PROJ.FOR	To save project information to disk.
LDPRJ	PROJ.FOR	To load project information from disk.
STPRJ	PROJ.FOR	To perform the store operation for SAVPRJ.
LODPRJ	PROJ.FOR	To perform the load operation for LDPRJ.

H.1.3 Polynomial Matrix Utility Routines

Subroutine Name	Source Filename	Description
NEWMAT	NEWMAT.FOR	To drive the system editing menu.
GETMAT	NEWMAT.FOR	To drive the matrix editing menu.
EDITM	EDTMAT.FOR	To edit a matrix of polynomials.
MOVCRS	EDTMAT.FOR	To move the matrix edit cursor.
DWSQRS	EDTMAT.FOR	To draw the edit matrix grid.
DWSHAP	EDTMAT.FOR	To draw the required shapes on the edit matrix grid.
WRTITL	EDTMAT.FOR	To write a string as a title (ie. underlined).
SETBLK	EDTMAT.FOR	To scan through the matrix and find non-zero polynomial elements.
UPBLK	EDTMAT.FOR	To update the blocks on the grid once a polynomial has been edited.
POLY	POLY.FOR	To print and permit editing of a polynomial.
DWPOLY	POLY.FOR	Draw the polynomial being edited.

H.1.3 Polynomial Matrix Utility Routines (contd.)

Subroutine Name	Source Filename	Description
EDTPOL	POLY.FOR	Enables the user to edit a polynomial from the matrix.
CALPOS	POLY.FOR	Calculate the position of the number (from polynomial) on the screen.
PRPOW	POLY.FOR	To print out the power of S in the polynomial.
CLRMAT	CLR1.FOR	To drive the menu for clearing the matrices.
CLRALL	CLR1.FOR	To clear all the system matrices.
CLRM	CLR2.FOR	To clear a system matrix.
IDMAT	CLR2.FOR	To initialise a system matrix to an identity matrix.
DOFILE	FILES.FOR	Drives the menu for system file and path names.
GETFIL	FILES.FOR	To edit the system file and path names.
DIR	FILES.FOR	Enables the user to list directories.
SAVEF	FMAT.FOR	To save matrix information to disk.

H.1.3 Polynomial Matrix Utility Routines (contd.)

Subroutine Name	Source Filename	Description
SAVMAT	FMAT.FOR	To perform the matrix save to disk operation.
LOADF	FMAT.FOR	To load matrix information from disk.
LODMAT	FMAT.FOR	To perform the matrix load from disk operation.
PRDERR	FMAT.FOR	To print an appropriate disk error message.
MAKNAM	FMAT.FOR	To construct a filename, including path name, from substrings.
DODIR	DODIR.FOR	To print out a small directory list at the bottom of a page.
CHPRT	PRPOLY.FOR	Checks if the user wishes to print out the matrix of polynomials.
PRPOLY	PRPOLY.FOR	To print out the matrix of polynomials to the printer.
PRTITL	PRPOLY.FOR	To print the page header.
PREQTN	PRPOLY.FOR	Prints out an element of the polynomial matrix.
PRSYM	PRPOLY.FOR	To print either the numerator or denominator elements.

H.1.4 Characteristic Loci (and associated) Routines

Subroutine Name	Source Filename	Description
DESIGN	DESIGN.FOR	To drive the main design menu.
PLOTS	DESIGN.FOR	To initiate the plotting procedures.
FRANGE	DESIGN.FOR	To enable the user to edit the frequency sweep parameters.
SELAXE	DESIGN.FOR	To enable the user to edit the plot page parameters.
CALFRE	CALFRE.FOR	To calculate the individual frequencies for plotting.
PTSRNG	CALFRE.FOR	To calculate the frequency pts. with equal spread over the range.
PTSDEC	CALFRE.FOR	To calculate the frequency pts. with a LOG increment over the range.
PTSOCT	CALFRE.FOR	To calculate the frequency pts. with an OCTAVE increment.
FREINC	CALFRE.FOR	To calculate the frequency pts. with a constant increment.
PRFTYP	FODDS.FOR	Prints the frequency unit parameters.
SELFRE	FODDS.FOR	To highlight the frequency unit option.

H.1.4 Characteristic Loci (and associated) Routines (contd.)

Subroutine Name	Source Filename	Description
PRITYP	FODDS.FOR	Prints the frequency increment type parameter.
SELINC	FODDS.FOR	To highlight frequency increment option.
PRSET	FODDS.FOR	To print plot page options.
SELSET	FODDS.FOR	To highlight plot page options.
PRSYS	FODDS.FOR	To print plot system options.
SYSET	FODDS.FOR	To highlight the plot system option.
PRANGE	FODDS.FOR	Prints the frequency sweep parameters.
SCALES	SCALES.FOR	Enables the user to edit the axes scales for loci, angle and bode plots.
PRELEM	SCALES.FOR	To print a set of scales and I/O names.
EDELEM	SCALES.FOR	Enables the user to edit axes scales.
DOLOCI	DWLOCI.FOR	To plot the characteristic loci.

H.1.4 Characteristic Loci (and associated) Routines (contd.)

Subroutine Name	Source Filename	Description
DOBODE	DWBODE.FOR	To plot the Bode diagrams and misalignment angles.
DOPLOT	DOPLOT.FOR	To control the plotting of loci, angles and bode plots.
VIEWPL	DOPLOT.FOR	To isolate a single plot being displayed.
VIEWL	PRINFO.FOR	To isolate a single loci plot
VIEWM	PRINFO.FOR	To isolate a single bode and misalignment angle plot.
IDBOX	PRINFO.FOR	Draws a box around a plot and places a dot in the upper right corner.
PRINFO	PRINFO.FOR	Print the isolated plots' axes scale limits.
DOGRAF	DOG.FOR	To calculate and plot the loci, angles and bode diagram values.
EVPOLY	DOGRAF.FOR	To evaluate a matrix of polynomials at a particular frequency.
CALPOL	DOGRAF.FOR	To evaluate a polynomial at a particular frequency.
PLTSYS	DOGRAF.FOR	To plot a set of points on loci and/or bode and angle diagrams.

H.1.4 Characteristic Loci (and associated) Routines (contd.)

Subroutine Name	Source Filename	Description
CHKDUP	DOGRAF.FOR	To ensure that no eigenvalue is selected more than once during sorting.
GETANG	DOGRAF.FOR	To produce the current track angle of an eigenvalue.
CHKSTP	DOGRAF.FOR	Checks if the user wishes to quit the current plot.
TRACK	TRACK.FOR	To track and sort a set of eigenvalues.

H.1.5 Controller Utility Routines

Subroutine Name	Source Filename	Description
GETK	GETK.FOR	To drive the controller generation menu.
EDTWRK	GETK.FOR	To setup call to the matrix edit menu.
SCALAR	GETK.FOR	To generate a temporary scalar matrix.
SCELEM	GETK.FOR	To print input/output names for scalar matrix.
MATMUL	GETK.FOR	To drive the menu for matrix symbolic multiplication.
CHKOP	GETK.FOR	Checks if the user wishes to proceed with matrix (temp) clear operation.
SETZER	GETK.FOR	To initialise a matrix of polynomials (temporary matrix).
PIMAT	PIMAT.FOR	To generate the basic matrix PI controller.
GETK0	PIMAT.FOR	Generates the P term of the matrix PI controller (at DC).
GETKI	PIMAT.FOR	Generates I term of the matrix PI controller (at high frequency).
DOPI	PIMAT.FOR	Combine elements from GETK0 and GETKI into a symbolic PI matrix.

H.1.5 Controller Utility Routines (contd.)

Subroutine Name	Source Filename	Description
SYMULT	SYMULT.FOR	To symbolically multiply two polynomial matrices.
POLMUL	SYMULT.FOR	To symbolically multiply two polynomials and add to a second polynomial.
PMULT	SYMULT.FOR	Symbolically multiplies two polynomials (just numerator or denominator).
POLZER	SYMULT.FOR	Checks if polynomial has zero elements.
PRTMAT	PRMAT.FOR	To print the eigen-system and inverse on a printer.
PRHEAD	PRMAT.FOR	To print eigen-system header on the printer.
PRVECT	PRMAT.FOR	Prints a single -value, -vector and inverse on the printer.
VECTOR	VECTOR.FOR	To present the eigen-system at some frequency to the user.
PRVECS	VECTOR.FOR	To print the eigen-system and inverse on graphics page 1.
PRFREQ	VECTOR.FOR	To print the frequency and units for the current displayed eigen-system.
BLKEL	VECTOR.FOR	To highlight an element number for eigen-system being displayed.

H.1.6 Time Simulation Routines (contd.)

Subroutine Name	Source Filename	Description
STPOPT	STEPOPT.FOR	Enables the user to edit the time simulation axes.
LOPTYP	STEPOPT.FOR	Prints the loop type for editing of time simulation parameters.
SELLOP	STEPOPT.FOR	To highlight the loop type option when editing time simulation params.
CONTYP	STEPOPT.FOR	Prints the controller type for editing time simulation parameters.
SELCON	STEPOPT.FOR	To highlight controller type option when editing time simulation params.

H.1.7 On-line Help Routines

Subroutine Name	Source Filename	Description
DOHELP	DOHELP.FOR	To initiate the help facility.
PRHELP	DOHELP.FOR	To controlling routine for the help facility.
PRPAGE	DOHELP.FOR	To print a single page of help information.
FINDPG	DOHELP.FOR	To find a particular page in the help file.
DRVHLP	DOHELP.FOR	To control the help screens within a help section.
BLKSEC	DOHELP.FOR	To highlight a help menu header option.
FLIPG1	HELP.ASM	To flip graphics page 1 in or out of a memory buffer.
INKEY2	HELP.ASM	The keyboard routine for the help facility.

H.1.8 Menu Driver Routines

Subroutine Name	Source Filename	Description
GET_MENU	DOMENU.ASM	To obtain the start address of the new menu text.
PR_MENU	DOMENU.ASM	Prints out a new menu to the screen.
DOMENU	DOMENU.ASM	Controls the selection of options within a menu.
MOVE_OPT	DOMENU.ASM	To move the highlight to the next option.
CHG_ATR	DOMENU.ASM	To change the attribute of a displayed option.
CHG_ATR2	DOMENU.ASM	To change the attribute of an option once it has been selected.
PR_BRIEF	DOMENU.ASM	To print an options' help line.
MTEXT (macros)	MTEXT.ASM	Macros for setting up menu data segments at compile time.

H.2 Low-level Utility Routines

H.2.1 Graphics Mode I/O Routines

Subroutine Name	Source Filename	Description
NOBLNK	STRIN.ASM	To remove trailing blanks from numeric strings.
MARKER	STRIN.ASM	To print out the string edit cursor.
SCANIN	STRIN.ASM	Scans input numeric string and converts lowercase to uppercase.
STRIN	STRIN.ASM	Enable user to input or edit a string.
STRPRT	STRIN.ASM	Prints out the string to be edited.
DELCHR	STRIN.ASM	Deletes a character from the string being edited.
INSCHR	STRIN.ASM	Inserts a character into the string being edited.
STREDT	STRIN.ASM	Controls the editing of a character string.
PRTNUM	NUMIN.FOR	To print a number on any one of the graphics pages.
GETNUM	NUMIN.FOR	To input or edit a number on any one of the graphics pages.
NUMSTR	NUMIN.FOR	To convert a number into a character string.

H.2.1 Graphics Mode I/O Routines (contd.)

Subroutine Name	Source Filename	Description
WIPSCR	UTIL.ASM	To clear one of the defined graphics pages.
WRTSTR	UTIL.ASM	To print a string at x,y on a particular graphics page.

H.2.2 String Manipulation Routines

Subroutine Name	Source Filename	Description
APPEND	STROPS.ASM	To append a substring to a string.
CMPSTR	STROPS.ASM	To identify a substring within a string.
LENSTR	STROPS.ASM	Calculates the length of the string (first non-blank char. from right).
STRCPY	STROPS.ASM	To copy a substring to a string.
COPYST	STROPS.ASM	To copy substrings from a string into a second string.
RJUST	STROPS.ASM	To remove leading blanks from a string.

H.2.3 Graphics Mode Plotting and Axes Routines

Subroutine Name	Source Filename	Description
DWAXES	AXES.FOR	To draw a set of axes on a given area on a graphics page.
YDATA	AXES.FOR	To adjust the window dimensions for the Y axes.
XLIN	AXES.FOR	To adjust the window dimensions for the linear type X axes.
XDECAD	AXES.FOR	To adjust the window dimensions for the decade type X axes.
XOCTAV	AXES.FOR	To adjust the window dimensions for the octave type X axes.
GETDEC	AXES.FOR	Calculates numbers for the decade type X axes.
GETOCT	AXES.FOR	Calculates numbers for the octave type X axes.
DOGNUM	AXES.FOR	Calculates numbers for linear type X axes and does string conversion.
DOAX	AXES.FOR	Prints the set of axes.
DYTICK	AXES.FOR	Draws the ticks and numbers on the Y axis.
DXTICK	AXES.FOR	Draws the ticks and numbers on the X axis.

H.2.3 Graphics Mode Plotting and Axes Routines (contd.)

Subroutine Name	Source Filename	Description
PNUMS	AXES.FOR	Converts numbers into printable strings.
DWNUMB	AXES.FOR	Prints a number string on a set of axes.
PLOTPT	PLOT.FOR	To plot a point on a set of previously defined axes.
DRAWPT	PLOT.FOR	To draw a set of points on a set of previously defined axes.

H.2.4 Complex Matrix Operations

Subroutine Name	Source Filename	Description
QZVECA	MATH.FOR	Calculates the eigen-values and vectors of the system : A*x = lambda*x.
COMHES	MATH.FOR	Initial stage of the eigen-value solution.
COMLR2	MATH.FOR	Final stage of the eigen-value solution.
CMULT	MATH.FOR	To multiply two complex matrices together.
COMINV	MATH.FOR	To compute the inverse of a complex matrix.

H.2.5 Keyboard Routines

Subroutine Name	Source Filename	Description
GETKEY	GETKEY.ASM	Low level keyboard utility routine.
INKEY	GETKEY.ASM	High level interface to keyboard utility routine.
CHKFUNC	GETKEY.ASM	Checks if a function has been entered.
FUNC3	GETKEY.ASM	Perform function 3 : Flip display pages.
FUNC1	GETKEY.ASM	Perform function 1 : Help facility.

H.2.6 Low-level System Utility Routines

Subroutine Name	Source Filename	Description
INDERR	DERROR.ASM	To hook the fatal error interrupt vector (23H).
REDERR	DERROR.ASM	To replace the original fatal error interrupt vector.
DISKERR	DERROR.ASM	Ensures failure of the disk operation in any disk fault occurs.
GETERR	DERROR.ASM	To query the latest disk or drive error.
RSTERR	DERROR.ASM	To clear the latest disk or drive error number.
ARC	UTIL.ASM	To draw a quarter circle.
BLKFIL	UTIL.ASM	To fill a rectangular block.
BOX	UTIL.ASM	To print a rectangle.
CIRC	UTIL.ASM	To print a circle of given radius.
CLRSCR	UTIL.ASM	To clear a graphics page.
DEFDRV	UTIL.ASM	To determine the current default drive.

H.2.6 Low-level System Utility Routines (contd.)

Subroutine Name	Source Filename	Description
DISP	UTIL.ASM	To display a page on the screen.
DLINE	UTIL.ASM	Draws a line on the current graphics page.
DO_TONE	UTIL.ASM	Switches the error tone on/off depending on tone counter.
ERTONE	UTIL.ASM	To set the error tone on by setting the tone counter.
FFIRST	UTIL.ASM	To find the first file in a directory search.
FNEXT	UTIL.ASM	To find the next file in the directory search.
CHKDIR	UTIL.ASM	Check if the filename returned by FFIRST or FNEXT is a directory name.
FILL	UTIL.ASM	To fill a polygon.
GETPG	UTIL.ASM	Returns the number of the page currently being displayed.
GETWPG	UTIL.ASM	Returns page number to which data is being written.
GETATT	UTIL.ASM	Returns the current write attribute.

H.2.6 Low-level System Utility Routines (contd.)

Subroutine Name	Source Filename	Description
GMODE	UTIL.ASM	To switch to graphics mode.
GPAGE	UTIL.ASM	Selects page to which data is written.
INTONE	UTIL.ASM	Initialises the tone interrupt vector.
INT10	UTIL.ASM	Interface to memory resident graphics sub-routines.
LEVEL	UTIL.ASM	To set the write intensity level.
MOVE	UTIL.ASM	To move the graphics cursor to position x,y.
PLOT	UTIL.ASM	To plot a point at x,y.
PRINTCH	UTIL.ASM	To print a single character to current graphics page.
PRSTR	UTIL.ASM	To print a string to the current graphics page.
ESC_CH	UTIL.ASM	To perform ESC sequences for PRSTR.
RETONE	UTIL.ASM	To reset the tone interrupt vector.

H.2.6 Low-level System Utility Routines (contd.)

Subroutine Name	Source Filename	Description
TMODE	UTIL.ASM	To switch to text mode.
TO_UPPER	UTIL.ASM	To convert characters to uppercase.
INTSCR	SCREEN.ASM	Initialises the screen.
RSTSCR	SCREEN.ASM	To reset the screen to text mode.

H.3 CAD System INCLUDE Files

Subroutine Name	Source Filename	Description
KMATS	SYSMAT.INC	K(s) polynomial matrix.
GMATS	SYSMAT.INC	G(s) polynomial matrix.
K2MATS	SYSMAT.INC	Temporary K(s) polynomial matrix.
MATS2	SYSMAT.INC	Temporary storage for matrix operations.
SYSNMS	SYSNMS.INC	System I/O names, matrix names and system titles.
FNAMES	FNAMES.INC	Filenames and pathnames.
PLOTFR	PLOTFR.INC	Calculated plot frequencies.
PLOTPG	PLOTPG.INC	Graph and plot parameters.
PLOCI	PLOCI.INC	Arrays for characteristic loci axes settings.
PBODE	PLOCI.INC	Arrays for Bode plot axes settings.
PMISA	PLOCI.INC	Arrays for Misalignment angle plot axes settings.

H.3 CAD System INCLUDE Files (contd.)

Subroutine Name	Source Filename	Description
(PARAMETERS)	KEYS.INC	Keyboard constants (not a common block).
TIME	TIME.INC	State variables for the time simulation (part of).
TIME1	TIME.INC	State variables for the time simulation (part of).
TIME2	TIME.INC	State variables for the time simulation (part of).
TIME3	TIME.INC	Arrays for time simulation axes settings.
K3MATS	K3MAT.INC	Temporary polynomial matrix.

H.4 CAD System Subroutine Names - Index Listing

Subroutine Name	Source Filename	Source Listing Page Number
This Page : APPEND - DODIR		
APPEND	STROPS.ASM	I - 218
ARC	UTIL.ASM	I - 236
BLKEL	VECTOR.FOR	I - 123
BLKFIL	UTIL.ASM	I - 237
BLKSEC	DOHELP.FOR	I - 147
BOX	UTIL.ASM	I - 237
CALFRE	CALFRE.FOR	I - 65
CALPOL	DOGRAF.FOR	I - 95
CALPOS	POLY.FOR	I - 34
CHG_ATR	DOMENU.ASM	I - 201
CHG_ATR2	DOMENU.ASM	I - 202
CHKDIR	UTIL.ASM	I - 243
CHKDUP	DOGRAF.FOR	I - 97
CHKFUNC	GETKEY.ASM	I - 227
CHKOP	GETK.FOR	I - 105
CHKQT	SIMUL.FOR	I - 130
CHKSTP	DOGRAF.FOR	I - 98
CHPRT	PRPOLY.FOR	I - 53
CIRC	UTIL.ASM	I - 238
CLRALL	CLR1.FOR	I - 37
CLRM	CLR2.FOR	I - 39
CLRMAT	CLR1.FOR	I - 37
CLRSCR	UTIL.ASM	I - 239
CMPSTR	STROPS.ASM	I - 219
CMULT	MATH.FOR	I - 179
COMHES	MATH.FOR	I - 171
COMINV	MATH.FOR	I - 179
COMLR2	MATH.FOR	I - 173
CONTYP	STEPOPT.FOR	I - 141
COPYST	STROPS.ASM	I - 222
DEFDRV	UTIL.ASM	I - 239
DELCHR	STRIN.ASM	I - 214
DESIGN	DESIGN.FOR	I - 57
DIR	FILES.FOR	I - 42
DISKERR	DERROR.ASM	I - 232
DISP	UTIL.ASM	I - 239
DLINE	UTIL.ASM	I - 239
DOAX	AXES.FOR	I - 159
DOBODE	DWBODE.FOR	I - 84
DODIR	DODIR.FOR	I - 51

H.4 CAD System Subroutine Names - Index Listing (contd.)

Subroutine Name	Source Filename	Source Listing Page Number
This Page :	DOFILE - GETATT	
DOFILE	FILES.FOR	I - 41
DOGNUM	AXES.FOR	I - 158
DOGRAF	DOG.FOR	I - 93
DOHELP	DOHELP.FOR	I - 143
DOLOCI	DWLOCI.FOR	I - 82
DOMENU	DOMENU.ASM	I - 199
DOPI	PIMAT.FOR	I - 110
DOPLOT	DOPLOT.FOR	I - 86
DOSIM	SIMUL.FOR	I - 124
DO_TONE	UTIL.ASM	I - 240
DRAWPT	PLOT.FOR	I - 168
DRVHLP	DOHELP.FOR	I - 146
DWAXES	AXES.FOR	I - 148
DWINPS	SIMUL.FOR	I - 131
DWNUMB	AXES.FOR	I - 166
DWPOLY	POLY.FOR	I - 31
DWSHAP	EDTMAT.FOR	I - 27
DWSQRS	EDTMAT.FOR	I - 26
DXTICK	AXES.FOR	I - 162
DYTICK	AXES.FOR	I - 160
EDELEM	SCALES.FOR	I - 79
EDITM	EDTMAT.FOR	I - 24
EDSCAL	SIMSCL.FOR	I - 134
EDTNMS	NAMES.FOR	I - 11
EDTPOL	POLY.FOR	I - 33
EDTWRK	GETK.FOR	I - 102
ERTONE	UTIL.ASM	I - 240
ESC_CH	UTIL.ASM	I - 250
EVPOLY	DOGRAF.FOR	I - 95
FFIRST	UTIL.ASM	I - 241
FILL	UTIL.ASM	I - 243
FINDPG	DOHELP.FOR	I - 145
FLIPG1	HELP.ASM	I - 193
FNEXT	UTIL.ASM	I - 242
FRANGE	DESIGN.FOR	I - 59
FREINC	CALFRE.FOR	I - 68
FUNC1	GETKEY.ASM	I - 229
FUNC3	GETKEY.ASM	I - 229
GETANG	DOGRAF.FOR	I - 98
GETATT	UTIL.ASM	I - 244

H.4 CAD System Subroutine Names - Index Listing (contd.)

Subroutine Name	Source Filename	Source Listing Page Number
This Page :	GETDEC - MATMUL	
GETDEC	AXES.FOR	I - 156
GETERR	DERROR.ASM	I - 233
GETFIL	FILES.FOR	I - 41
GETK	GETK.FOR	I - 102
GETK0	PIMAT.FOR	I - 108
GETKEY	GETKEY.ASM	I - 226
GETKI	PIMAT.FOR	I - 109
GETMAT	NEWMAT.FOR	I - 22
GETNUM	NUMIN.FOR	I - 257
GETOCT	AXES.FOR	I - 157
GETPG	UTIL.ASM	I - 244
GETSYS	GETSYS.FOR	I - 8
GETWPG	UTIL.ASM	I - 244
GET_MENU	DOMENU.ASM	I - 198
GMODE	UTIL.ASM	I - 245
GPAGE	UTIL.ASM	I - 245
IDBOX	PRINFO.FOR	I - 90
IDMAT	CLR2.FOR	I - 39
INDERR	DERROR.ASM	I - 231
INISIM	SIMUL.FOR	I - 129
INISYS	INISYS.FOR	I - 5
INKEY	GETKEY.ASM	I - 227
INKEY2	HELP.ASM	I - 194
INMENU	INMENU.FOR	I - 9
INSCHR	STRIN.ASM	I - 215
INT10	UTIL.ASM	I - 246
INTONE	UTIL.ASM	I - 245
INTSCR	SCREEN.ASM	I - 255
LDPRJ	PROJ.FOR	I - 16
LENSTR	STROPS.ASM	I - 221
LEVEL	UTIL.ASM	I - 247
LOADF	FMAT.FOR	I - 46
LOCI	LOCI.FOR	I - 3
LODMAT	FMAT.FOR	I - 47
LODPRJ	PROJ.FOR	I - 19
LOPSET	FODDS.FOR	I - 75
LOPTYP	STEPOPT.FOR	I - 139
MAKNAM	FMAT.FOR	I - 49
MARKER	STRIN.ASM	I - 210
MATMUL	GETK.FOR	I - 104

H.4 CAD System Subroutine Names - Index Listing (contd.)

Subroutine Name	Source Filename	Source Listing Page Number
This Page :	MOVCRS - PRSTR	
MOVCRS	EDTMAT.FOR	I - 25
MOVE	UTIL.ASM	I - 248
MOVE_OPT	DOMENU.ASM	I - 201
MTEXT	MTEXT.ASM	I - 205
NAMES	NAMES.FOR	I - 10
NEWMAT	NEWMAT.FOR	I - 22
NOBLNK	STRIN.ASM	I - 210
NSTEP	SIMUL.FOR	I - 126
NUMSTR	NUMIN.FOR	I - 260
PARPOS	NAMES.FOR	I - 12
PIMAT	PIMAT.FOR	I - 108
PLOT	UTIL.ASM	I - 248
PLOTPT	PLOT.FOR	I - 167
PLOTS	DESIGN.FOR	I - 57
PLTSIM	SIMUL.FOR	I - 130
PLTSYS	DOGRAF.FOR	I - 96
PMULT	SYMULT.FOR	I - 115
PNUMS	AXES.FOR	I - 164
POLMUL	SYMULT.FOR	I - 113
POLY	POLY.FOR	I - 30
POLZER	SYMULT.FOR	I - 115
PRANGE	FODDS.FOR	I - 76
PRDERR	FMAT.FOR	I - 48
PRELEM	SCALES.FOR	I - 78
PREQTN	PRPOLY.FOR	I - 55
PRFREQ	VECTOR.FOR	I - 122
PRFTYP	FODDS.FOR	I - 70
PRHEAD	PRMAT.FOR	I - 118
PRHELP	DOHELP.FOR	I - 144
PRINFO	PRINFO.FOR	I - 90
PRINTCH	UTIL.ASM	I - 248
PRITYP	FODDS.FOR	I - 71
PROJ	PROJ.FOR	I - 14
PROJNM	NAMES.FOR	I - 10
PRPAGE	DOHELP.FOR	I - 144
PRPOLY	PRPOLY.FOR	I - 53
PRPOW	POLY.FOR	I - 35
PRSCAL	SIMSCL.FOR	I - 134
PRSET	FODDS.FOR	I - 73
PRSTR	UTIL.ASM	I - 249

H.4 CAD System Subroutine Names - Index Listing (contd.)

Subroutine Name	Source Filename	Source Listing Page Number
This Page :	PRSYM - STRCPY	
PRSYM	PRPOLY.FOR	I - 55
PRSYS	FODDS.FOR	I - 74
PRTITL	PRPOLY.FOR	I - 54
PRTMAT	PRMAT.FOR	I - 117
PRTNUM	NUMIN.FOR	I - 257
PRVECS	VECTOR.FOR	I - 122
PRVECT	PRMAT.FOR	I - 118
PR_BRIEF	DOMENU.ASM	I - 203
PR_MENU	DOMENU.ASM	I - 198
PTSDEC	CALFRE.FOR	I - 66
PTSOCT	CALFRE.FOR	I - 67
PTSRNG	CALFRE.FOR	I - 65
QZVECA	MATH.FOR	I - 171
REDERR	DERROR.ASM	I - 232
RETONE	UTIL.ASM	I - 251
RJUST	STROPS.ASM	I - 224
RSTERR	DERROR.ASM	I - 233
RSTSCR	SCREEN.ASM	I - 256
SAVEF	FMAT.FOR	I - 44
SAVMAT	FMAT.FOR	I - 45
SAVPRJ	PROJ.FOR	I - 14
SCALAR	GETK.FOR	I - 103
SCALES	SCALES.FOR	I - 77
SCANIN	STRIN.ASM	I - 211
SCELEM	GETK.FOR	I - 104
SELAXE	DESIGN.FOR	I - 62
SELCON	STEPOPT.FOR	I - 141
SELFRE	FODDS.FOR	I - 71
SELINC	FODDS.FOR	I - 72
SELLOP	STEPOPT.FOR	I - 140
SELSET	FODDS.FOR	I - 74
SETBLK	EDTMAT.FOR	I - 28
SETZER	GETK.FOR	I - 106
SIMAT	SIMUL.FOR	I - 125
SIMGRF	SIMUL.FOR	I - 127
SIMSCL	SIMSCL.FOR	I - 133
SIMUL	SIMUL.FOR	I - 124
STPOPT	STEPOPT.FOR	I - 137
STPRJ	PROJ.FOR	I - 17
STRCPY	STROPS.ASM	I - 221

H.4 CAD System Subroutine Names - Index Listing (contd.)

Subroutine Name	Source Filename	Source Listing Page Number
This Page :	STREDT - YDATA	
STREDT	STRIN.ASM	I - 216
STRIN	STRIN.ASM	I - 211
STRPRT	STRIN.ASM	I - 213
SYMULT	SYMULT.FOR	I - 112
SYSET	FODDS.FOR	I - 75
SYSNM	NAMES.FOR	I - 11
TMODE	UTIL.ASM	I - 251
TO_UPPER	UTIL.ASM	I - 252
TRACK	TRACK.FOR	I - 100
UPBLK	EDTMAT.FOR	I - 29
VECTOR	VECTOR.FOR	I - 120
VIEWL	PRINFO.FOR	I - 89
VIEWM	PRINFO.FOR	I - 89
VIEWPL	DOPLOT.FOR	I - 86
WIPSCR	UTIL.ASM	I - 252
WRTITL	EDTMAT.FOR	I - 27
WRTSTR	UTIL.ASM	I - 252
XDECAD	AXES.FOR	I - 152
XLIN	AXES.FOR	I - 151
XOCTAV	AXES.FOR	I - 154
YDATA	AXES.FOR	I - 149

Appendix I Characteristic Loci CAD System Code Listings

The code listings for the CAD system contained in this appendix are grouped according to source file and appear in the following order :

```
C
C     FILE              : LOCI.FOR
C ***************************************************************************
CN    MODULE NAME       : loci (program - not a subroutine))
CA    FUNCTION          : The start of the Characteristic Loci CAD package.
CS    CALL SEQUENCE     : loci (from DOS)
CI    INPUT PARAMETERS  : None.
C
CO    OUTPUT PARAMETERS: None.
C
CG    GLOBAL VARIABLES  : None.
CM    MODULES CALLED    : design (for), DOMENU (asm),inisys (for), inmenu (for),
C                         INTONE (asm), newmat (for), RETONE (asm), getsys (for)
C                         RSTSCR (asm), wrtitl (for), WRTSTR (asm), simul (for),
C                         REDERR (asm).
C
CE    ERROR CONDITIONS  : INT10 GRAPHIX package not resident in the system,
C                         HERCULES card not installed.
C
CC    COMMENTS          : This is the start of the Characteristic Loci CAD
C                         package. The initial menu is controlled from this
C                         routine.
C                         This routine calls subroutines to perform the
C                         following functions :-
C                           i) Initialises the 2 graphics pages.
C                          ii) Initialises the system variables.
C                         iii) Initialises the system timer interrupt for
C                              error tone operation.
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
C                         To run the Characteristic Loci package :-
C                           i) Put the HERCULES card into the full screen mode
C                              ie. type 'HGC FULL <CR>'
C                          ii) Make the INT10 GRAPHIX package resident in the
C                              system, ie. type 'INT10 <CR>'.
C                         iii) Run the CAD package, ie. type 'LOCI <CR>'.
C
C ***************************************************************************

      program LOCI
      implicit integer*2 (D)

      integer*2 opt1

      call INTONE()
      call inmenu()
      call inisys()
      call getsys()
      call WRTSTR(0,500,258,17,'(Hit F1 for help)')
99    continue
      call wrtitl(0,250,35,27,'Characteristic Loci Design.')
      opt1 = DOMENU(1)
      call WRTSTR(0,250,35,28,'                              ')
      if (opt1.eq.1) then
         call newmat()
      elseif (opt1.eq.2) then
         call design()
      elseif (opt1.eq.3) then
         call simul()
      endif
      if (opt1.ne.4) goto 99
      call REDERR()
      call RSTSCR()
      call RETONE()
      stop 'Characteristic Loci CAD Terminated.'
```

```
      end

C ********************************************************************************
CH    REVISION HISTORY :
C     VERSION       BY                DATE        COMMENT
C     1.00          Ian Fisher        23/06/88    Creation.
C     FILEND           :
```

```
C
C     FILE             : INISYS.FOR
C ***************************************************************************
C
CN    MODULE NAME      : INISYS
CA    FUNCTION         : To initialise system variables.
CS    CALL SEQUENCE    : call inisys()
CI    INPUT PARAMETERS : All system variables in the following INCLUDE files :-
C                        FNAMES.INC  -  File and system path names.
C                        PLOCI.INC   -  Plot scales and axes data.
C                        PLOTPG.INC  -  Plot format data.
C                        SYSMAT.INC  -  System polynomial matrices.
C                        SYSNMS.INC  -  System names and titles.
C                        TIME.INC    -  Time simulation parameters.
CO    OUTPUT PARAMETERS: Initialised system variables.
CG    GLOBAL VARIABLES :
CM    MODULES CALLED   : APPEND (asm).
C
CE    ERROR CONDITIONS : Can't think of any, except bad compile and link.
C
CC    COMMENTS         : Initialises the system to default parameters.
C
C ***************************************************************************
      subroutine inisys()

$include: 'fnames.inc'
$include: 'ploci.inc'
$include: 'plotpg.inc'
$include: 'sysmat.inc'
$include: 'sysnms.inc'
$include: 'time.inc'
      integer*2 numb

      do 99 i = 1,10
         do 100 j = 1,10
            do 101 k = 1,13
               kmat(i,j,1,k) = 0.0
               kmat(i,j,2,k) = 0.0
               gmat(i,j,1,k) = 0.0
               gmat(i,j,2,k) = 0.0
               k2mat(i,j,1,k) = 0.0
               k2mat(i,j,2,k) = 0.0
101         continue
100      continue
99    continue
      do 96 i = 1,10
         kmat(i,i,1,1) = 1.0
         kmat(i,i,2,1) = 1.0
         k2mat(i,i,1,1) = 1.0
         k2mat(i,i,2,1) = 1.0
         gmat(i,i,1,1) = 1.0
         gmat(i,i,2,1) = 1.0
96    continue

      order = 1
      matok = .FALSE.
      do 98 i = 1,100
         calcr(i) = 0.0
         calci(i) = 0.0
         zr(i) = 0.0
         zi(i) = 0.0
98    continue
      do 97 i = 1,10
         alfr(i) = 0.0
         alfi(i) = 0.0
         beta(i) = 0.0
         inpnms(i) = 'Input                '
         outnms(i) = 'Output               '
         if (i.gt.9) then
            numb = i/10
            call APPEND(25,inpnms(i),1,char(48+numb))
            call APPEND(25,outnms(i),1,char(48+numb))
```

```
            numb = i - numb*10
            call APPEND(25,inpnms(i),1,char(48+numb))
            call APPEND(25,outnms(i),1,char(48+numb))
         else
            call APPEND(25,inpnms(i),1,char(48+i))
            call APPEND(25,outnms(i),1,char(48+i))
         endif
97    continue

      gname  = 'Matrix = G(s)            '
      kname  = 'Matrix = K(s)            '
      lname  = 'Matrix = L(s)            '
      prjnm  = '                         '
      engnms = '                         '
      outpth = 'a:\                      '
      inpath = 'a:\                      '
      gfnam  = '.gs                      '
      kfnam  = '.ks                      '
      lfnam  = '.ls                      '
      prjfil = '.prj                     '
      prport = 'PRN  '


      ftype  = 2
      fstart = 0.0001
      fstop  = 1.0
      inctyp = 2
      increm = 0.1
      incpts = 10
      plset1 = 1
      pmset1 = 0
      plset2 = 0
      pmset2 = 1
      pmat1  = 1
      pmat2  = 2
      fback1 = 1
      fback2 = 2

      do 90 i = 1,10
         xblim(1,i) = 100.0
         xblim(2,i) = -50.0
         xblim(3,i) = 0.0
         xmlim(1,i) = 90.0
         xmlim(2,i) = 0.0
         xmlim(3,i) = 0.0
90    continue
         yblim(1) = 100.0
         yblim(2) = 0.0
         yblim(3) = 0.0
         ymlim(1) = 100.0
         ymlim(2) = 0.0
         ymlim(3) = 0.0

      do 91 i = 1,10
         xllim(1,i) = 100.0
         xllim(2,i) = -100.0
         xllim(3,i) = 0.0
         yllim(1,i) = 100.0
         yllim(2,i) = -100.0
         yllim(3,i) = 0.0
91    continue

      cloop = 1
      coninc = 1
      ti = 0.0
      dt = 0.05
      tend = 10.0
      do 80 i = 1,5
         stpinp(i) = 0
         stpdat(i) = 0.0
         stpinc(i) = 0.0
80    continue
```

```
      do 81 i = 1,10
         ytlim(1,i) = 10.0
         ytlim(2,i) = -10.0
         ytlim(3,i) = 0.0
81    continue

      return
      end

C ********************************************************************************
CH    REVISION HISTORY :
C     VERSION       BY                DATE        COMMENT
C     1.00          Ian Fisher        23/06/88    Creation.
C     FILEND           :
```

```
C
C     FILE             : GETSYS.FOR
C ***************************************************************************
CN    MODULE NAME      : GETSYS
CA    FUNCTION         : To load the 'loci.sys' configuration file.
CS    CALL SEQUENCE    : call getsys()
CI    INPUT PARAMETERS : None.
CO    OUTPUT PARAMETERS: None.
CG    GLOBAL VARIABLES : fnames.inc
CM    MODULES CALLED   : ERTONE (asm), WRTSTR (asm).
CE    ERROR CONDITIONS : ?
C
CC    COMMENTS         : The routine tries to load up the configuration file :
C                        'loci.sys'.   The   file   contains   the   following
C                        information :-
C                          i) Save pathname.
C                         ii) Load pathname.
C                        iii) Printer name (eg. PRN or LPT1)
C
C                        If the file cannot be found then an error message is
C                        printed  and  no  other  attempt is made to  read  a
C                        configuration file.
C ***************************************************************************
      subroutine getsys()
$include: 'fnames.inc'
      integer*2 ferr,funit
      funit = 6
      open(funit,FILE='loci.sys',STATUS='OLD',IOSTAT=ferr)
      if (ferr.eq.0) then
         read(funit,'(a)',iostat=ferr,err=300) outpth
         read(funit,'(a)',iostat=ferr,err=300) inpath
         read(funit,'(a)',iostat=ferr,err=300) prport
      endif
300   continue
      if (ferr.ne.0) then
         call ERTONE()
         call WRTSTR(0,20,258,24,'*** LOCI.SYS not loaded.')
      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     FILE              : INMENU.FOR
C ***************************************************************************
CN    MODULE NAME       : inmenu
CA    FUNCTION          : To initialise the menu screens and menus.
CS    CALL SEQUENCE     : call inmenu()
CI    INPUT PARAMETERS  : None.
CO    OUTPUT PARAMETERS: None.
CG    GLOBAL VARIABLES  : None.
CM    MODULES CALLED    : BOX (asm), GPAGE (asm), INTSCR (asm), INDERR (asm).
CE    ERROR CONDITIONS  : ?
C
CC    COMMENTS          : Just clears all the pages, setups the frames around
C                         the screens and prints out the menu headers.
C
C ***************************************************************************
      subroutine inmenu()

      call INTSCR()
      call INDERR()
      call BOX(1,271,717,270)
      call BOX(3,269,713,266)
      call GPAGE(1)
      call BOX(1,319,717,318)
      call BOX(3,317,713,314)
      call GPAGE(0)

      return
      end

C ***************************************************************************
CH    REVISION HISTORY :
C     VERSION       BY                DATE        COMMENT
C     1.00          Ian Fisher        23/06/88    Creation.
C     FILEND           :
```

```
C
C     FILE              : NAMES.FOR
C ***************************************************************************
CN    MODULE NAME       : names
CA    FUNCTION          : To drive the system names and titles menu.
CS    CALL SEQUENCE     : call names()
CI    INPUT PARAMETERS  : None.
CO    OUTPUT PARAMETERS: None.
CG    GLOBAL VARIABLES  : None.
CM    MODULES CALLED    : DOMENU (asm), WIPSCR (asm), wrtitl (for), WRTSTR (asm)
C                         projnm (for), sysnm (for).
CE    ERROR CONDITIONS  : ?
C
CC    COMMENTS          : Calls the appropriate routine apon request of the
C                         user.
C                         The routine returns to the calling routine if the
C                         ESC key is entered as an option.
C
C ***************************************************************************
      subroutine names()
      implicit integer*2 (D)
      integer*2 opt13
99    continue
      call wrtitl(0,250,35,24,'System Titles and Names.')
      opt13 = DOMENU(13)
      call WRTSTR(0,250,35,25,'                          ')
      if (opt13.eq.1) then
         call projnm()
      elseif (opt13.eq.2) then
         call sysnm()
      endif
      if (opt13.gt.0) goto 99
      call WIPSCR(0)
      return
      end

C ***************************************************************************
CN    MODULE NAME       : projnm
CA    FUNCTION          : To allow the user to edit the project titles.
CS    CALL SEQUENCE     : call projnm()
CI    INPUT PARAMETERS  : None.
CO    OUTPUT PARAMETERS: None.
CG    GLOBAL VARIABLES  : sysnms.inc
C                         keys.inc
CM    MODULES CALLED    : STRIN (asm), WIPSCR (asm), wrtitl (for), WRTSTR (asm).
CE    ERROR CONDITIONS  : ?
C
CC    COMMENTS          : Permits the user to edit the project title and the
C                         engineer or group name.
C
C ***************************************************************************
      subroutine projnm()
      implicit integer*2 (S)
$include: 'sysnms.inc'
$include: 'keys.inc'
      integer*2 key,pos
      call WIPSCR(0)
      call wrtitl(0,250,30,22,'Editing Project Names.')
      call wrtitl(0,40,60,23,'Project Name or Title   :')
      call wrtitl(0,40,85,25,'Design Team or Engineer :')
      call WRTSTR(0,40,120,36,'RETURN, TAB and cursor keys to move.')
      call WRTSTR(0,40,135,16,'ESC key to exit.')
      call WRTSTR(0,275,85,25,engnms)
      pos = 1
399   continue
      if (pos.eq.1) then
         key = STRIN(0,275,60,25,prjnm)
      elseif (pos.eq.2) then
         key = STRIN(0,275,85,25,engnms)
      endif
      if ((key.eq.tabk).or.(key.eq.downk).or.(key.eq.upk).or.
     +    (key.eq.rtabk).or.(key.eq.retk)) then
```

```
         pos = pos + 1
         if (pos.gt.2) pos = 1
      endif
      if (key.ne.esck) goto 399
      call WIPSCR(0)
      return
      end

C ***************************************************************************
CN    MODULE NAME       : sysnm
CA    FUNCTION          : To enable the user to edit the system I/O names.
CS    CALL SEQUENCE     : call sysnm()
CI    INPUT PARAMETERS  : None.
CO    OUTPUT PARAMETERS: None.
CG    GLOBAL VARIABLES  : sysmat.inc
C                         sysnms.inc
CM    MODULES CALLED    : DOMENU (asm), WIPSCR (asm), WRTSTR (asm), edtnms (for)
C                         wrtitl (for).
CE    ERROR CONDITIONS  : ?
C
CC    COMMENTS          : This routine allows the  user  to  edit  the  system
C                         input and output names. The user will have to select
C                         which names to edit : input or output.
C                         The routine will return to the calling  routine once
C                         the user has finished editing the names.
C
C ***************************************************************************
      subroutine sysnm()
      implicit integer*2 (D)
$include: 'sysmat.inc'
$include: 'sysnms.inc'
      integer*2 opt132
99    continue
      call wrtitl(0,250,35,23,'System Parameter Names.')
      opt132 = DOMENU(132)
      call WRTSTR(0,250,35,25,'                          ')
      if (opt132.eq.1) then
         call edtnms(order,inpnms,37,
     +               'Editing System Input Parameter Names.')
      elseif (opt132.eq.2) then
         call edtnms(order,outnms,38,
     +               'Editing System Output Parameter Names.')
      endif
      if (opt132.ne.0) goto 99
      call WIPSCR(0)
      return
      end

C ***************************************************************************
CN    MODULE NAME       : edtnms.
CA    FUNCTION          : To present the names to the user for editing.
CS    CALL SEQUENCE     : call edtnms(order,names,length,message)
CI    INPUT PARAMETERS  :     order - (integer*2) Order of the system.
C                         names(10) - (character*25) System name matrix.
C                            length - (integer*2) Length of message being passed.
C                           message - (character*length) Message from calling
C                                                        routine.
CO    OUTPUT PARAMETERS: None.
CG    GLOBAL VARIABLES  : keys.inc
CM    MODULES CALLED    : INKEY (asm), prtnum (for), STRIN (asm), wrtitl (for),
C                         WIPSCR (asm), WRTSTR (asm), parpos (for).
CE    ERROR CONDITIONS  : ?
C
CC    COMMENTS          : Allows the user to edit 'order' number of names of
C                         the system. The user  finishes the editing session
C                         by hitting an ESC.
C
C ***************************************************************************
      subroutine edtnms(n,parnms,len,msg)
      implicit integer*2 (I,S)
      integer*2 n
      character*25 parnms(10)
```

```
      integer*2 len
      character*25 msg
$include: 'keys.inc'
      integer*2 xpos,ypos,pos,key
      xpos = 125
      ypos = 55
      call WIPSCR(0)
      call wrtitl(0,200,20,len,msg)
      if (n.lt.1) then
         call WRTSTR(0,40,40,22,'No parameters to name.')
         call WRTSTR(0,40,70,20,'Hit ESC to continue.')
98       continue
         key = INKEY(1)
         if (key.ne.esck) goto 98
         call WIPSCR(0)
      else
         do 99 i = 1,n
               if (i.eq.1) then
                  call wrtitl(0,15,40,11,'Parameter :')
               elseif (i.eq.11) then
                  call wrtitl(0,365,40,11,'Parameter :')
                  xpos = 465
                  ypos = 55
               endif
               call prtnum(0,(xpos-55),ypos,1,4,'(i2)',2,i)
               call WRTSTR(0,xpos,ypos,25,parnms(i))
               ypos = ypos + 20
99       continue
         pos = 1
         xpos = 125
         ypos = 55
100      continue
         call parpos(pos,xpos,ypos)
         key = STRIN(0,xpos,ypos,25,parnms(pos))
         if ((key.eq.retk).or.(key.eq.downk)) then
            pos = pos + 1
            if (pos.gt.n) pos = 1
         elseif (key.eq.tabk) then
            if (n.gt.10) then
               if ((pos.le.10).and.((pos+10).le.n)) then
                  pos = pos + 10
               elseif ((pos.le.10).and.((pos+10).gt.n)) then
                  pos = pos + 1
                  if ((pos.gt.n).or.(pos.eq.11)) pos = 1
               elseif (pos.gt.10) then
                  pos = pos - 9
                  if (pos.eq.11) pos = 1
               endif
            else
               pos = pos + 1
               if (pos.gt.n) pos = 1
            endif
         elseif (key.eq.rtabk) then
            if (n.gt.10) then
               if ((pos.le.10).and.((pos+9).le.n)) then
                  pos = pos + 9
               elseif ((pos.le.10).and.((pos+9).gt.n)) then
                  pos = pos - 1
                  if (pos.lt.1)  pos = n
               elseif (pos.gt.10) then
                  pos = pos - 10
                  if (pos.eq.0) pos = 10
               endif
            else
               pos = pos - 1
               if (pos.lt.1) pos = n
            endif
         elseif (key.eq.upk) then
            pos = pos - 1
            if (pos.lt.1) pos = n
         endif
         if (key.ne.esck) goto 100
```

```
         call WIPSCR(0)
      endif
      return
      end

C ***************************************************************************
CN    MODULE NAME      : parpos
CA    FUNCTION         : To calculate highlight position for system name.
CS    CALL SEQUENCE    : call parpos(pos,x,y)
CI    INPUT PARAMETERS : pos - (integer*2) Position of the name within the
C                                          name array.
CO    OUTPUT PARAMETERS: x,y - (integer*2) Position of the name highlight.
CG    GLOBAL VARIABLES : None.
CM    MODULES CALLED   : None.
CE    ERROR CONDITIONS : ?
C
CC    COMMENTS         : Just calculates the X,Y co-ords of the name highlight
C                        which is dependent on the name position in the array.
C
C ***************************************************************************
      subroutine parpos(pos,xpos,ypos)
      integer*2 pos,xpos,ypos
      xpos = 125
      ypos = 55
      if (pos.gt.10) then
         xpos = 465
         ypos = ypos + (pos-11)*20
      else
         ypos = ypos + (pos-1)*20
      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     FILE             : PROJ.FOR
C ***************************************************************************
CN    MODULE NAME      : proj
CA    FUNCTION         : To drive the menu for project information management.
CS    CALL SEQUENCE    : call proj()
CI    INPUT PARAMETERS : None.
CO    OUTPUT PARAMETERS: None.
CG    GLOBAL VARIABLES : None
CM    MODULES CALLED   : DOMENU (asm), WIPSCR (asm), wrtitl (for), ldprj (for),
C                        savprj (for), dir (for).
CE    ERROR CONDITIONS : ?
C
CC    COMMENTS         : Calls subroutines to perform functions  upon  request
C                        of the user. The ESC key causes the routine to return
C                        control to the calling routine.
C
C ***************************************************************************
      subroutine proj()
      implicit integer*2 (D)
      integer*2 opt112
      call WIPSCR(0)
      call wrtitl(0,250,30,31,'Project Information Operations.')
99    continue
      opt112 = DOMENU(112)
      call WIPSCR(0)
      if (opt112.eq.1) then
         call ldprj()
      elseif (opt112.eq.2) then
         call savprj()
      elseif (opt112.eq.3) then
         call dir()
      endif
      if (opt112.gt.2) goto 99
      return
      end

C ***************************************************************************
CN    MODULE NAME      : savprj
CA    FUNCTION         : To save project information to disk.
CS    CALL SEQUENCE    : call savprj()
CI    INPUT PARAMETERS : None.
CO    OUTPUT PARAMETERS: None.
CG    GLOBAL VARIABLES : keys.inc
C                        fnames.inc
CM    MODULES CALLED   : BLKFIL (asm), LEVEL (asm), dodir (for),  LENSTR (asm),
C                        ERTONE (asm), maknam (for), prderr (for), stprj (for),
C                        STRIN (asm), wrtitl (for), WRTSTR (asm), RSTERR (asm),
C                        GETERR (asm).
CE    ERROR CONDITIONS : ?
C
CC    COMMENTS         : Opens the project file, first prompting  the user to
C                        check the file name. The routine performs  all  disk
C                        checks. If any errors, error  messages  are  printed
C                        and the routine quits, otherwise a routine is called
C                        to save the project information.
C
C ***************************************************************************
      subroutine savprj()
      implicit integer*2 (c,G,L,S)
$include: 'keys.inc'
$include: 'fnames.inc'
      integer*2 key,ferr,derr,serr,funit,len
      character*1 yesno
      character*50 usenam
      funit = 5
      call maknam(prjfil,outpth,usenam)
      call wrtitl(0,40,60,35,'Saving Project Information to File.')
      call wrtitl(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.')
      call dodir(0,180,5,'*.prj',outpth)
```

```
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()
         prjfil = usenam
         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(0,215,110,25,usenam)
               call RSTERR()
               open(funit,FILE=usenam,STATUS='NEW',IOSTAT=ferr)
               if (ferr.eq.0) then
                  serr = stprj(funit)
                  close(funit)
                  if (serr.eq.0) then
                     call WRTSTR(0,40,130,26,
     +                    'Project information saved.')
                  else
                     call WRTSTR(0,40,130,42,
     +                  '*** Error : Project information not saved.')
                  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,(9*len),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='NEW',IOSTAT=ferr)
               if (ferr.eq.0) then
                  serr = stprj(funit)
                  close(funit)
                  if (serr.eq.0) then
                     call WRTSTR(0,40,170,26,
     +                           'Project information saved.')
                  else
                     call WRTSTR(0,40,170,42,
     +                   '*** Error : Project information not saved.')
                  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(0,40,170,40,
     +              'Project information operation cancelled.')
         endif
      endif
      else
         call WRTSTR(0,40,110,40,
     +                'Project information operation cancelled.')
      endif
      close(funit)
      return
      end

C ***************************************************************************
CN    MODULE NAME      : ldprj
CA    FUNCTION         : To load project information from disk.
CS    CALL SEQUENCE    : call ldprj()
CI    INPUT PARAMETERS : None
CO    OUTPUT PARAMETERS: None.
CG    GLOBAL VARIABLES : keys.inc
C                        sysmat.inc
c                        fnames.inc
CM    MODULES CALLED   : ERTONE (asm), lodprj (for), dodir (for), RSTERR (asm),
C                        GETERR (asm), maknam (for), prderr (for), STRIN (asm),
C                        wrtitl (for), WRTSTR (asm).
CE    ERROR CONDITIONS : ?
C
CC    COMMENTS         : Opens the project file, first prompting the user to
C                        check the file name. The routine performs all disk
C                        checks. If any errors, error messages are printed
C                        and the routine quits, otherwise a routine is called
C                        to load the project information.
C
C ***************************************************************************
      subroutine ldprj()
      implicit integer*2 (c,G,l,S)
$include: 'keys.inc'
$include: 'sysmat.inc'
$include: 'fnames.inc'
      integer*2 key,ferr,derr,lerr,funit
      character*50 usenam
      funit = 6
      call wrtitl(0,40,60,38,'Loading Project 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.')
      call dodir(0,180,5,'*.prj',inpath)
      call maknam(prjfil,inpath,usenam)
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()
         prjfil = usenam
         open(funit,FILE=usenam,STATUS='OLD',IOSTAT=ferr)
         if (ferr.eq.0) then
            call WRTSTR(0,40,110,32,
     +           'Loading project information ....')
            lerr = lodprj(funit)
            close(funit)
            if (lerr.eq.0) then
               call WRTSTR(0,330,110,27,
     +                     'Project information loaded.')
            else
               call WRTSTR(0,40,130,43,
     +              '*** Error : Project information not loaded.')
            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(0,40,140,40,
     +        'Project information operation cancelled.')
       endif
       return
       end

C ******************************************************************************
CN     MODULE NAME       : stprj
CA     FUNCTION          : To store project information to disk.
CS     CALL SEQUENCE     : error = stprj(unit)
CI     INPUT PARAMETERS  : unit - (integer*2) Storage unit number.
CO     OUTPUT PARAMETERS: error - (integer*2) Error code if any error.
CG     GLOBAL VARIABLES  : sysnms.inc
C                          fnames.inc
C                          sysmat.inc
C                          plotpg.inc
C                          ploci.inc
C                          time.inc
CM     MODULES CALLED    : ERTONE (asm).
CE     ERROR CONDITIONS  : ?
C
CC     COMMENTS          : Stores the project 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
C ******************************************************************************
       integer*2 function stprj(funit)
       integer*2 funit
$include: 'sysnms.inc'
$include: 'fnames.inc'
$include: 'sysmat.inc'
$include: 'plotpg.inc'
$include: 'ploci.inc'
$include: 'time.inc'
       integer*2 chkio
       write(funit,'(a)',iostat=chkio,err=200)
     + '; Characteristic Design : Project file.'
       write(funit,'(a)',iostat=chkio,err=200)
     + '; ************************************'
       write(funit,'(a)',iostat=chkio,err=200) '; Project Title :'
       write(funit,'(a)',iostat=chkio,err=200) '; ***************'
       write(funit,'(a)',iostat=chkio,err=200) prjnm
       write(funit,'(a)',iostat=chkio,err=200)
     + '; Design Group or Engineer :'
       write(funit,'(a)',iostat=chkio,err=200)
     + '; *************************'
       write(funit,'(a)',iostat=chkio,err=200) engnms
       write(funit,'(a)',iostat=chkio,err=200) '; Matrix names.'
       write(funit,'(a)',iostat=chkio,err=200) '; *************'
       write(funit,'(a)',iostat=chkio,err=200) '; System Matrix : G(s) :'
       write(funit,'(a)',iostat=chkio,err=200) '; ***********************'
       write(funit,'(a)',iostat=chkio,err=200) gname
       write(funit,'(a)',iostat=chkio,err=200)
     + '; Controller Matrix : K(s) :'
       write(funit,'(a)',iostat=chkio,err=200)
     + '; ***************************'
       write(funit,'(a)',iostat=chkio,err=200) kname
       write(funit,'(a)',iostat=chkio,err=200) '; Order of System :'
       write(funit,'(a)',iostat=chkio,err=200) '; *****************'
```

```
      write(funit,'(i2)',iostat=chkio,err=200) order
      write(funit,'(a)',iostat=chkio,err=200) '; System Inputs :'
      write(funit,'(a)',iostat=chkio,err=200) '; ***************'
      do 980 i = 1,order
         write(funit,'(a)',iostat=chkio,err=200) inpnms(i)
980   continue

      write(funit,'(a)',iostat=chkio,err=200) '; System Outputs :'
      write(funit,'(a)',iostat=chkio,err=200) '; ****************'
      do 981 i = 1,order
         write(funit,'(a)',iostat=chkio,err=200) outnms(i)
981   continue

      write(funit,'(a)',iostat=chkio,err=200) '; System File Names.'
      write(funit,'(a)',iostat=chkio,err=200) '; ******************'
      write(funit,'(a)',iostat=chkio,err=200) '; G(s) Filename :'
      write(funit,'(a)',iostat=chkio,err=200) '; ***************'
      write(funit,'(a)',iostat=chkio,err=200) gfnam
      write(funit,'(a)',iostat=chkio,err=200) '; K(s) Filename :'
      write(funit,'(a)',iostat=chkio,err=200) '; ***************'
      write(funit,'(a)',iostat=chkio,err=200) kfnam
      write(funit,'(a)',iostat=chkio,err=200) '; System Path Names.'
      write(funit,'(a)',iostat=chkio,err=200) '; ******************'
      write(funit,'(a)',iostat=chkio,err=200) '; Save Path :'
      write(funit,'(a)',iostat=chkio,err=200) '; ************'
      write(funit,'(a)',iostat=chkio,err=200) outpth
      write(funit,'(a)',iostat=chkio,err=200) '; Load Path :'
      write(funit,'(a)',iostat=chkio,err=200) '; ************'
      write(funit,'(a)',iostat=chkio,err=200) inpath
      write(funit,'(a)',iostat=chkio,err=200)
     +      '; Frequency Sweep Parameters'
      write(funit,'(a)',iostat=chkio,err=200)
     +      '; *************************'
      write(funit,'(3g10.4)',iostat=chkio,err=200)
     +      fstart,fstop,increm
      write(funit,'(3i8)',iostat=chkio,err=200) ftype,inctyp,incpts
      write(funit,'(a)',iostat=chkio,err=200) '; Plot page settings.'
      write(funit,'(a)',iostat=chkio,err=200) '; ********************'
      write(funit,'(4i8)',iostat=chkio,err=200)
     +      plset1,pmset1,pmat1,fback1
      write(funit,'(4i8)',iostat=chkio,err=200)
     +      plset2,pmset2,pmat2,fback2
      write(funit,'(a)',iostat=chkio,err=200) '; Axes scales.'
      write(funit,'(a)',iostat=chkio,err=200) '; *************'
      do 950 i = 1,order
         write(funit,'(3g15.6)',iostat=chkio,err=200)
     +        (xllim(j,i), j=1,3)
         write(funit,'(3g15.6)',iostat=chkio,err=200)
     +        (yllim(j,i), j=1,3)
950   continue
      do 951 i =1,order
         write(funit,'(3g15.6)',iostat=chkio,err=200)
     +        (xblim(j,i), j=1,3)
951   continue
      write(funit,'(3g15.6)',iostat=chkio,err=200)
     +     (yblim(j), j=1,3)
      do 952 i = 1,order
         write(funit,'(3g15.6)',iostat=chkio,err=200)
     +        (xmlim(j,i), j=1,3)
952   continue
      write(funit,'(3g15.6)',iostat=chkio,err=200)
     +     (ymlim(j), j=1,3)
      write(funit,'(a)',iostat=chkio,err=200)
     +'; ************************************************************'
      write(funit,'(a)',iostat=chkio,err=200)
     +      '; Time Simulation Parameters:'
      write(funit,'(a)',iostat=chkio,err=200)
     +'; *************************'
      write(funit,'(2g10.4)',iostat=chkio,err=200) tend,dt
      write(funit,'(2i2)',iostat=chkio,err=200) cloop,coninc
      write(funit,'(5g10.4)',iostat=chkio,err=200) (stpinc(i), i=1,5)
      write(funit,'(5g10.4)',iostat=chkio,err=200) (stpdat(i), i=1,5)
```

```
      write(funit,'(5i2)',iostat=chkio,err=200) (stpinp(i), i=1,5)
      do 953 i = 1,order
         write(funit,'(3g10.4)',iostat=chkio,err=200)
     +        (ytlim(j,i), j=1,3)
953   continue

      write(funit,'(a)',iostat=chkio,err=200)
     +'; ************************************************************'
      if (chkio.eq.0) goto 201
200   call ERTONE()
201   continue
      stprj = chkio
      return
      end

C ******************************************************************************
CN    MODULE NAME       : lodprj
CA    FUNCTION          : To load project information from disk.
CS    CALL SEQUENCE     : error = lodprj(unit)
CI    INPUT PARAMETERS  : unit - (integer*2) Storage unit number.
CO    OUTPUT PARAMETERS: error - (integer*2) Error code if any error.
CG    GLOBAL VARIABLES  : sysnms.inc
C                         fnames.inc
C                         sysmat.inc
C                         plotpg.inc
C                         ploci.inc
C                         time.inc
CM    MODULES CALLED    : ERTONE (asm).
CE    ERROR CONDITIONS  : ?
C
CC    COMMENTS          : Loads the project 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
C ******************************************************************************
      integer*2 function lodprj(funit)
      integer*2 funit
$include: 'sysnms.inc'
$include: 'fnames.inc'
$include: 'sysmat.inc'
$include: 'plotpg.inc'
$include: 'ploci.inc'
$include: 'time.inc'
      integer*2 chkio
      character*1 chkch
      read(funit,'(a)',iostat=chkio,err=300) chkch
      read(funit,'(a)',iostat=chkio,err=300) chkch
      read(funit,'(a)',iostat=chkio,err=300) chkch
      read(funit,'(a)',iostat=chkio,err=300) chkch
      if (chkch.ne.';') goto 300
      read(funit,'(a)',iostat=chkio,err=300) prjnm

      read(funit,'(a)',iostat=chkio,err=300) chkch
      read(funit,'(a)',iostat=chkio,err=300) chkch
      if (chkch.ne.';') goto 300
      read(funit,'(a)',iostat=chkio,err=300) engnms

      read(funit,'(a)',iostat=chkio,err=300) chkch
      read(funit,'(a)',iostat=chkio,err=300) chkch
      read(funit,'(a)',iostat=chkio,err=300) chkch
      read(funit,'(a)',iostat=chkio,err=300) chkch
      if (chkch.ne.';') goto 300
      read(funit,'(a)',iostat=chkio,err=300) gname

      read(funit,'(a)',iostat=chkio,err=300) chkch
      read(funit,'(a)',iostat=chkio,err=300) chkch
      if (chkch.ne.';') goto 300
      read(funit,'(a)',iostat=chkio,err=300) kname

      read(funit,'(a)',iostat=chkio,err=300) chkch
      read(funit,'(a)',iostat=chkio,err=300) chkch
```

```
      if (chkch.ne.';') goto 300
      read(funit,'(i2)',iostat=chkio,err=300) order

      read(funit,'(a)',iostat=chkio,err=300) chkch
      read(funit,'(a)',iostat=chkio,err=300) chkch
      if (chkch.ne.';') goto 300

      do 888 i = 1,order
         read(funit,'(a)',iostat=chkio,err=300) inpnms(i)
888   continue
      read(funit,'(a)',iostat=chkio,err=300) chkch
      read(funit,'(a)',iostat=chkio,err=300) chkch
      if (chkch.ne.';') goto 300

      do 887 i = 1,order
         read(funit,'(a)',iostat=chkio,err=300) outnms(i)
887   continue

      read(funit,'(a)',iostat=chkio,err=300) chkch
      read(funit,'(a)',iostat=chkio,err=300) chkch
      read(funit,'(a)',iostat=chkio,err=300) chkch
      read(funit,'(a)',iostat=chkio,err=300) chkch
      if (chkch.ne.';') goto 300
      read(funit,'(a)',iostat=chkio,err=300) gfnam

      read(funit,'(a)',iostat=chkio,err=300) chkch
      read(funit,'(a)',iostat=chkio,err=300) chkch
      if (chkch.ne.';') goto 300
      read(funit,'(a)',iostat=chkio,err=300) kfnam

      read(funit,'(a)',iostat=chkio,err=300) chkch
      read(funit,'(a)',iostat=chkio,err=300) chkch
      read(funit,'(a)',iostat=chkio,err=300) chkch
      read(funit,'(a)',iostat=chkio,err=300) chkch
      if (chkch.ne.';') goto 300
      read(funit,'(a)',iostat=chkio,err=300) outpth

      read(funit,'(a)',iostat=chkio,err=300) chkch
      read(funit,'(a)',iostat=chkio,err=300) chkch
      if (chkch.ne.';') goto 300
      read(funit,'(a)',iostat=chkio,err=300) inpath
      read(funit,'(a)',iostat=chkio,err=300) chkch
      read(funit,'(a)',iostat=chkio,err=300) chkch
      read(funit,'(3g10.4)',iostat=chkio,err=300)
     +      fstart,fstop,increm
      read(funit,'(3i8)',iostat=chkio,err=300) ftype,inctyp,incpts
      read(funit,'(a)',iostat=chkio,err=300) chkch
      read(funit,'(a)',iostat=chkio,err=300) chkch
      if (chkch.ne.';') goto 300
      read(funit,'(4i8)',iostat=chkio,err=300)
     +      plset1,pmset1,pmat1,fback1
      read(funit,'(4i8)',iostat=chkio,err=300)
     +      plset2,pmset2,pmat2,fback2
      read(funit,'(a)',iostat=chkio,err=300) chkch
      read(funit,'(a)',iostat=chkio,err=300) chkch
      if (chkch.ne.';') goto 300
      do 850 i = 1,order
         read(funit,'(3g15.6)',iostat=chkio,err=300)
     +        (xllim(j,i), j=1,3)
         read(funit,'(3g15.6)',iostat=chkio,err=300)
     +        (yllim(j,i), j=1,3)
850   continue
      do 851 i =1,order
         read(funit,'(3g15.6)',iostat=chkio,err=300)
     +        (xblim(j,i), j=1,3)
851   continue
      read(funit,'(3g15.6)',iostat=chkio,err=300)
     +     (yblim(j), j=1,3)
      do 852 i = 1,order
         read(funit,'(3g15.6)',iostat=chkio,err=300)
     +        (xmlim(j,i), j=1,3)
852   continue
```

```
      read(funit,'(3g15.6)',iostat=chkio,err=300)
     +     (ymlim(j), j=1,3)
      read(funit,'(a)',iostat=chkio,err=300) chkch
      read(funit,'(a)',iostat=chkio,err=300) chkch
      read(funit,'(a)',iostat=chkio,err=300) chkch
      if (chkch.ne.';') goto 300
      read(funit,'(2g10.4)',iostat=chkio,err=300) tend,dt
      read(funit,'(2i2)',iostat=chkio,err=300) cloop,coninc
      read(funit,'(5g10.4)',iostat=chkio,err=300) (stpinc(i), i=1,5)
      read(funit,'(5g10.4)',iostat=chkio,err=300) (stpdat(i), i=1,5)
      read(funit,'(5i2)',iostat=chkio,err=300) (stpinp(i), i=1,5)
      do 853 i = 1,order
         read(funit,'(3g10.4)',iostat=chkio,err=300)
     +        (ytlim(j,i), j=1,3)
853   continue
      if (chkio.eq.0) goto 301
300   call ERTONE()
301   continue
      lodprj = chkio
      return
      end

C ********************************************************************************
CH    REVISION HISTORY :
C     VERSION       BY                DATE        COMMENT
C     1.00          Ian Fisher        23/06/88    Creation.
C     FILEND           :
```

```
C
C     FILE              : NEWMAT.FOR
C ***************************************************************************
CN    MODULE NAME       : newmat
CA    FUNCTION          : Drives the system editing menu.
CS    CALL SEQUENCE     : call newmat()
CI    INPUT PARAMETERS  : None.
CO    OUTPUT PARAMETERS: None.
CG    GLOBAL VARIABLES  : sysmat.inc
C                         fnames.inc
C                         sysnms.inc
CM    MODULES CALLED    : clrmat (for), dofile (for), DOMENU (asm), names (for),
C                         getmat (for), proj (for), wrtitl (for),  WRTSTR (asm),
C                         WIPSCR (asm).
CE    ERROR CONDITIONS  : ?
C
CC    COMMENTS          : Calls subroutines to perform system editing functions
C                         upon request of the  user. The  ESC  key  causes  the
C                         routine to return control to the calling routine.
C
C ***************************************************************************
      subroutine newmat()
      implicit integer*2 (D)
$include: 'sysmat.inc'
$include: 'fnames.inc'
$include: 'sysnms.inc'
      integer*2 opt11
      character*25 mtitle
      character*4 match
      character*4 defdir
      call WIPSCR(0)
99    continue
      call wrtitl(0,250,35,24,'Editing System Matrices.')
      opt11 = DOMENU(11)
      call WRTSTR(0,250,35,25,'                          ')
      if (opt11.eq.1) then
         call dofile()
      elseif (opt11.eq.2) then
         call proj()
      elseif (opt11.eq.3) then
         mtitle = 'Editing the G(s) matrix. '
         match = 'G(s)'
         defdir = '*.gs'
         call getmat(match,mtitle,24,order,gmat,gname,gfnam,defdir,4)
      elseif (opt11.eq.4) then
         mtitle = 'Editing the K(s) matrix. '
         match = 'K(s)'
         defdir = '*.ks'
         call getmat(match,mtitle,24,order,kmat,kname,kfnam,defdir,3)
      elseif (opt11.eq.5) then
         call clrmat()
      elseif (opt11.eq.6) then
         call names()
      endif
      if (opt11.ne.0) goto 99
      call WIPSCR(0)
      return
      end

C ***************************************************************************
CN    MODULE NAME       : getmat
CA    FUNCTION          : To drive the matrix editing menu.
CS    CALL SEQUENCE     : call getmat(match,mtitle,titlen,n,mat,mname,
C                                     fname,funit)
CI    INPUT PARAMETERS  :           match - (character*4) Matrix print name.
C                                  mtitle - (character*25) Title to be printed
C                                                          on the screen.
C                                  titlen - (integer*2) Length of mtitle.
C                                       n - (integer*2) Order of the system.
C                           mat(10,10,2,13) - (real*4) The matrix of polynomials.
C                                   mname - (character*25) Matrix title.
C                                   fname - (character*25) Matrix filename.
```

```
C                                  defdir - (character*4) Default directory
C                                           search string.
C                                   funit - (integer*2) System unit number for
C                                                   matrix file.
CO    OUTPUT PARAMETERS: None.
CG    GLOBAL VARIABLES : None.
CM    MODULES CALLED   : DOMENU (asm), loadf (for), savef (for), editm (for),
C                        dir (for),  clrm  (for),  idmat (for),  chprt (for),
C                        WIPSCR (asm), wrtitl (for).
CE    ERROR CONDITIONS : ?
C
CC    COMMENTS         : Calls subroutines to perform the editing functions
C                        on the matrix upon request of the user. The ESC key
C                        returns control to the calling routine.
C
C ******************************************************************************
      subroutine getmat(match,mtitle,titlen,n,mat,mname,fname,defdir,
     +                  funit)
      implicit integer*2 (D)
      character*4 match
      character*25 mtitle
      integer*2 titlen,n
      real*4 mat(10,10,2,13)
      character*25 mname,fname
      character*4 defdir
      integer*2 funit

      integer*2 opt113
      call WIPSCR(1)
      call WIPSCR(0)
      call wrtitl(0,10,18,titlen,mtitle)
      call wrtitl(1,480,35,titlen,mtitle)
60    continue
      opt113 = DOMENU(113)
      if ((opt113.ne.0).and.(opt113.ne.7)) then
         call WIPSCR(0)
         call wrtitl(0,10,18,titlen,mtitle)
         call wrtitl(1,480,35,titlen,mtitle)
      endif
      if (opt113.eq.1) then
         call loadf(n,mat,mname,fname,defdir,funit)
      elseif (opt113.eq.2) then
         call savef(n,mat,mname,fname,defdir,funit)
      elseif (opt113.eq.3) then
         call editm(n,mat,mname,fname)
      elseif (opt113.eq.4) then
         call dir()
      elseif (opt113.eq.5) then
         call clrm(mat,4,match)
      elseif (opt113.eq.6) then
         call idmat(mat,4,match)
      elseif (opt113.eq.7) then
         call chprt(n,mat,mname,fname)
      endif
      if (opt113.ne.0) goto 60
      call WIPSCR(0)
      return
      end

C ******************************************************************************
CH    REVISION HISTORY :
C     VERSION       BY                DATE        COMMENT
C     1.00          Ian Fisher        23/06/88    Creation.
C     FILEND           :
```

```
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(10,10,2,13) - (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,mat,mname,fname)
      implicit integer*2 (I,S,g)
      integer*2 n
      real*4 mat(10,10,2,13)
      character*25 mname,fname
$include: 'keys.inc'
      integer*2 i,j,pos,key,nold
      call wipscr(0)
      call wrtitl(1,480,65,7,'Title :')
      call WRTSTR(1,480,85,25,mname)
      call wrtitl(1,480,115,10,'Filename :')
      call WRTSTR(1,480,135,25,fname)
      call wrtitl(1,480,165,7,'Order :')
      call prtnum(1,560,165,1,4,'(i3)',3,n)
      call WRTSTR(1,480,270,24,'Use tab keys and RETURN ')
      call WRTSTR(1,480,285,14,'key to select.')
      call WRTSTR(1,480,305,16,'ESC key to exit.')
      if ((n.gt.0).and.(n.lt.11)) then
         call dwsqrs(n,1)
         call setblk(n,mat,1)
      endif
      pos = 1
      i = 1
      j = 1
70    continue
      if (pos.eq.1) then
         key = STRIN(1,480,85,25,mname)
      elseif (pos.eq.2) then
         key = STRIN(1,480,135,25,fname)
      elseif (pos.eq.3) then
         nold = n
71       continue
         key = getnum(1,560,165,1,4,'(i3)',3,n)
         if ((n.lt.1).or.(n.gt.10)) then
           call ERTONE()
           call WRTSTR(1,480,235,25,'*** ERROR: 0 < Order ≤ 10')
         else
```

```
            call WRTSTR(1,480,235,25,'                              ')
         endif
         if ((n.lt.1).or.(n.gt.10)) goto 71
         if (nold.ne.n) then
            call dwsqrs(nold,0)
            call setblk(nold,mat,0)
            i = 1
            j = 1
            call dwsqrs(n,1)
            call setblk(n,mat,1)
         endif
      elseif (pos.eq.4) then
         if ((n.gt.0).and.(n.lt.21)) then
            call wrtitl(1,480,195,18,'Current position :')
            call dwshap(n,i,j,2,2)
102         continue
            call WRTSTR(1,485,215,1,'(')
            call prtnum(1,500,215,1,4,'(i2)',2,i)
            call WRTSTR(1,525,215,1,',')
            call prtnum(1,540,215,1,4,'(i2)',2,j)
            call WRTSTR(1,565,215,1,')')
            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 poly(n,mat,mname,i,j)
               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(1,480,195,19,'                      ')
            call WRTSTR(1,480,215,19,'                      ')
            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.')
         endif
      endif
      if ((key.eq.tabk).or.(key.eq.retk).or.(key.eq.downk)) then
         pos = pos + 1
         if (pos.gt.4) pos = 1
      elseif ((key.eq.rtabk).or.(key.eq.upk)) then
         pos = pos - 1
         if (pos.lt.1) pos = 4
      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 ***************************************************************************
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(10,10,2,13) - (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(10,10,2,13)
       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.11)) 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     FILE             : POLY.FOR
C ***************************************************************************
CN    MODULE NAME      : poly
CA    FUNCTION         : To print and permit editing of a polnomial.
CS    CALL SEQUENCE    : call poly(n,mat,mname,row,col)
CI    INPUT PARAMETERS :              n - (integer*2) The order of the system.
C                       mat(10,10,2,13) - (real*4) The matrix of polynomials.
C                              mname - (character*25) The matrix title.
C                            row,col - (integer*2) The  position  of  the
C                                                  polynomial in the matrix.
CO    OUTPUT PARAMETERS: None.
CG    GLOBAL VARIABLES : keys.inc
C                        sysmat.inc
C                        sysnms.inc
CM    MODULES CALLED   : DLINE (asm),  DISP (asm),  dwpoly (for), edtpol (for),
C                        ERTONE (asm),  GPAGE (asm), LENSTR (asm),  MOVE (asm),
C                        getnum (for), prtnum (for), wrtitl (for), WRTSTR (asm)
CE    ERROR CONDITIONS : ?
C
CC    COMMENTS         : This routine prints out the polynomial and associated
C                        information.  The  user  is  allowed  to  edit  this
C                        information. The  ESC  key  returns  control  to  the
C                        calling routine.
C
C ***************************************************************************
      subroutine poly(n,mat,mname,row,col)
      implicit integer*2 (I,g,L)
      integer*2 n
      real*4 mat(10,10,2,13)
      character*25 mname
      integer*2 row,col
$include: 'keys.inc'
$include: 'sysmat.inc'
$include: 'sysnms.inc'
      integer*2 key,numer,denom,pos,numold,denold,len

      call DISP(0)
      call GPAGE(0)
      call wrtitl(0,368,18,7,'Title :')
      call WRTSTR(0,440,18,25,mname)
      call wrtitl(0,440,33,9,'Element :')
      call WRTSTR(0,530,33,1,'(')
      call prtnum(0,555,33,1,4,'(i2)',2,row)
      call WRTSTR(0,580,33,1,',')
      call prtnum(0,595,33,1,4,'(i2)',2,col)
      call WRTSTR(0,620,33,1,')')
      call wrtitl(0,440,52,3,'In ')
      call WRTSTR(0,440,66,3,'Out')
      call WRTSTR(0,472,59,1,'=')
      call WRTSTR(0,485,52,25,'                         ')
      call WRTSTR(0,485,66,25,'                         ')
      len = LENSTR(25,inpnms(row))
      if (LENSTR(25,outnms(col)).gt.len) then
         len = LENSTR(25,outnms(col))
      endif
      call wrtitl(0,485,52,len,inpnms(row))
      call WRTSTR(0,485,66,len,outnms(col))
      call wrtitl(0,10,45,22,'Order of Numerator   :')
      numer = int (mat(row,col,1,13))
      denom = int (mat(row,col,2,13))
      call prtnum(0,220,45,1,4,'(i3)',3,numer)
      call wrtitl(0,10,65,22,'Order of Denominator :')
      call prtnum(0,220,65,1,4,'(i3)',3,denom)
      call WRTSTR(0,10,85,61,
     +'Use cursor, tab keys and RETURN key to move. ESC key to exit.')
      call dwpoly(n,mat,row,col)
      call MOVE(4,95)
      call DLINE(715,95)
      pos = 1

99    continue
```

```
      if (pos.eq.1) then
         numer = int (mat(row,col,1,13))
         numold = numer
98       continue
         key = getnum(0,220,45,1,4,'(i3)',3,numer)
         if ((numer.lt.0).or.(numer.gt.10)) then
           call ERTONE()
           call WRTSTR(0,260,45,14,'0≤Numerator≤10')
         else
           call WRTSTR(0,260,45,18,'                  ')
         endif
         if ((numer.lt.0).or.(numer.gt.10)) goto 98
         mat(row,col,1,13) = float (numer)
         if (numold.ne.numer) then
            call dwpoly(n,mat,row,col)
         endif
      elseif (pos.eq.2) then
         denom = int (mat(row,col,2,13))
         denold = denom
97       continue
         key = getnum(0,220,65,1,4,'(i3)',3,denom)
         if ((denom.lt.0).or.(denom.gt.10)) then
           call ERTONE()
           call WRTSTR(0,260,65,16,'0≤Denominator≤10')
         else
           call WRTSTR(0,260,65,20,'                    ')
         endif
         if ((denom.lt.0).or.(denom.gt.10)) goto 97
         mat(row,col,2,13) = float (denom)
         if (denold.ne.denom) then
            call dwpoly(n,mat,row,col)
         endif
      elseif (pos.eq.3) then
c         call dwpoly(n,mat,row,col)
         call edtpol(n,mat,row,col,key)
      endif
      if ((key.eq.downk).or.(key.eq.tabk).or.(key.eq.retk)) then
         pos = pos + 1
         if (pos.gt.3) pos = 1
      elseif ((key.eq.rtabk).or.(key.eq.upk)) then
         pos = pos - 1
         if (pos.lt.1) pos = 3
      endif
      if (key.ne.esck) goto 99
      call DISP(1)
      call GPAGE(1)
      return
      end

C *************************************************************************
CN    MODULE NAME      : dwpoly
CA    FUNCTION         : Draw the polynomial being edited.
CS    CALL SEQUENCE    : call dwpoly(n,mat,row,col)
CI    INPUT PARAMETERS :              n - (integer*2) The order of the system.
C                       mat(10,10,2,13) - (real*4) The matrix of polynomials.
C                              row,col - (integer*2) The  position  of  the
C                                                    polynomial in the matrix.
CO    OUTPUT PARAMETERS: None.
CG    GLOBAL VARIABLES :
CM    MODULES CALLED   : DLINE (asm),  BLKFIL (asm),  gnum  (for),  MOVE (asm),
C                        prpow (for), prtnum (for), WIPSCR (asm), WRTSTR (asm).
CE    ERROR CONDITIONS : ?
C
CC    COMMENTS         : Prints out the polynomial being edited.
C
C *************************************************************************
      subroutine dwpoly(n,mat,row,col)
      integer*2 n
      real*4 mat(10,10,2,13)
      integer*2 row,col
      integer*2 xcent,xpos,ypos,xdelta,temp1,temp2,numer,denom
      integer*2 prpos
```

```
      call WIPSCR(2)
      numer = int (mat(row,col,1,13)) + 1
      denom = int (mat(row,col,2,13)) + 1
      xcent = 360
      xdelta = 12*9
      if (numer.gt.5) then
         temp1 = numer - 5
         xpos = xcent - (3*xdelta)
      else
         temp1 = 1
         xpos = xcent - int ((real(numer+1)/2.0)*real(xdelta))
      endif
      ypos = 135
      prpos = 0
      do 200 i = numer,temp1,-1
         call prtnum(0,(xpos+prpos*xdelta),ypos,3,7,'(g10.4)',10,
     +              mat(row,col,1,i))
         call WRTSTR(0,(xpos+int((real(prpos+1)-0.6)*xdelta)),
     +              (ypos-15),1,'S')
         call prpow(0,(xpos+int((real(prpos+1)-0.6)*xdelta)),
     +              (ypos-15),(i-1))
         prpos = prpos + 1
200   continue
      if (numer.lt.6) then
         call prtnum(0,(xpos+prpos*xdelta),ypos,3,7,'(g10.4)',10,
     +              mat(row,col,1,12))
         call WRTSTR(0,(xpos+int((real(prpos+1)-0.6)*xdelta)),
     +              (ypos-15),1,'e')
         call gnum(0,(xpos+int((real(prpos+1)-0.6)*xdelta)+8),
     +             (ypos-23),'s')
         call gnum(0,(xpos+int((real(prpos+1)-0.6)*xdelta)+15),
     +             (ypos-23),char(233))
      endif
      xpos = 36
      ypos = 220
      prpos = 0
      if (temp1.gt.1) then
         do 203 i = (temp1-1),1,-1
            call prtnum(0,(xpos+prpos*xdelta),ypos,3,7,'(g10.4)',10,
     +                 mat(row,col,1,i))
            call WRTSTR(0,(xpos+int((real(prpos+1)-0.6)*xdelta)),
     +                 (ypos-15),1,'S')
            call prpow(0,(xpos+int((real(prpos+1)-0.6)*xdelta)),
     +                 (ypos-15),(i-1))
            prpos = prpos + 1
203      continue
      endif
      if (numer.gt.5) then
         call prtnum(0,(xpos+prpos*xdelta),ypos,3,7,'(g10.4)',10,
     +              mat(row,col,1,12))
         call WRTSTR(0,(xpos+int((real(prpos+1)-0.6)*xdelta)),
     +              (ypos-15),1,'e')
         call gnum(0,(xpos+int((real(prpos+1)-0.6)*xdelta)+8),
     +             (ypos-23),'s')
         call gnum(0,(xpos+int((real(prpos+1)-0.6)*xdelta)+15),
     +             (ypos-23),char(233))
      endif

      if (denom.gt.6) then
         temp2 = denom - 5
         xpos = xcent - (3*xdelta)
      else
         temp2 = 1
         xpos = xcent - int ((real(denom)/2.0)*real(xdelta))
      endif
      ypos = 155
      prpos = 0
      do 201 i = denom,temp2,-1
         call prtnum(0,(xpos+(prpos)*xdelta),ypos,3,7,'(g10.4)',10,
     +              mat(row,col,2,i))
         call WRTSTR(0,(xpos+int((real(prpos+1)-0.6)*xdelta)),
```

```
     +               (ypos+20),1,'S')
         call prpow(0,(xpos+int((real(prpos+1)-0.6)*xdelta)),
     +               (ypos+20),(i-1))
         prpos = prpos + 1
201   continue
      if (temp2.gt.1) then
         prpos = 0
         xpos = 36
         ypos  = 240
         do 204 i = (temp2-1),1,-1
            call prtnum(0,(xpos+(prpos)*xdelta),ypos,3,7,'(g10.4)',10,
     +                  mat(row,col,2,i))
            call WRTSTR(0,(xpos+int((real(prpos+1)-0.6)*xdelta)),
     +                  (ypos+20),1,'S')
            call prpow(0,(xpos+int((real(prpos+1)-0.6)*xdelta)),
     +                  (ypos+20),(i-1))
            prpos = prpos + 1
204      continue
      endif

      if ((numer+1).gt.denom) then
         temp1 = numer+1
      else
         temp1 = denom
      endif

      if (temp1.gt.6) then
         call MOVE(4,185)
         call DLINE(715,185)
         call BLKFIL(36,141,(6*xdelta),2)
         temp1 = temp1 - 6
         call BLKFIL(36,226,(temp1*xdelta),2)
      else
         xpos = xcent - int ((real(temp1)/2.0)*real(xdelta))
         call BLKFIL(xpos,141,(temp1*xdelta),2)
      endif
      return
      end

C ***************************************************************************
CN    MODULE NAME      : edtpol
CA    FUNCTION         : Enables the user to edit a polynomial.
CS    CALL SEQUENCE    : call edtpol(n,mat,row,col,key)
CI    INPUT PARAMETERS :                n - (integer*2) The order of the system.
C                       mat(10,10,2,13) - (real*4) The matrix of polynomials.
C                              row,col - (integer*2) The   position   of   the
C                                                    polynomial in the matrix.
CO    OUTPUT PARAMETERS: key - (integer*2) The  user  entered key that this
C                                          routine interprets as a non-edit
C                                          key.
CG    GLOBAL VARIABLES : keys.inc
CM    MODULES CALLED   : calpos (for), getnum (for).
CE    ERROR CONDITIONS : ?
C
CC    COMMENTS         : The routine assumes that the polynomial has already
C                        been  printed. The routine permits the user to edit
C                        the elements of the polynomial. The ESC key returns
C                        control to the calling routine.
C
C ***************************************************************************
      subroutine edtpol(n,mat,row,col,key)
      implicit integer*2 (g)
      integer*2 n
      real*4 mat(10,10,2,13)
      integer*2 row,col,key
$include: 'keys.inc'
      integer*2 xpos,ypos,numer,denom,polord,numden

      numden = 1
      polord = 1
      numer = int (mat(row,col,1,13)) + 2
      denom = int (mat(row,col,2,13)) + 1
```

```
300   continue
      call calpos(numer,denom,polord,numden,xpos,ypos)
      if (numden.eq.1) then
         key = getnum(0,xpos,ypos,3,7,'(g10.4)',10,
     +                mat(row,col,numden,(numer-polord)))
      elseif (numden.eq.2) then
         key = getnum(0,xpos,ypos,3,7,'(g10.4)',10,
     +                mat(row,col,numden,(denom-polord+1)))
      elseif (numden.eq.3) then
         key = getnum(0,xpos,ypos,3,7,'(g10.4)',10,
     +                mat(row,col,1,12))
      endif
      if ((key.eq.tabk).or.(key.eq.retk)) then
         polord = polord + 1
         if ((numden.eq.1).and.(polord.gt.(numer-1))) then
            polord = numer
            numden = 3
         elseif ((numden.eq.2).and.(polord.gt.denom)) then
            polord = 1
            numden = 1
         elseif (numden.eq.3) then
            polord = 1
            numden = 2
         endif
      elseif (key.eq.rtabk) then
         polord = polord - 1
         if ((numden.eq.1).and.(polord.lt.1)) then
            polord = denom
            numden = 2
         elseif ((numden.eq.2).and.(polord.lt.1)) then
            numden = 3
            polord = denom
         elseif (numden.eq.3) then
            polord = numer-1
            numden = 1
         endif
      elseif ((key.eq.upk).or.(key.eq.downk)) then
         if ((numden.eq.1).or.(numden.eq.3)) then
            numden = 2
            if (polord.gt.denom) polord = denom
         else
            numden = 1
            if (polord.ge.numer) then
               polord = numer
               numden = 3
            endif
         endif
      endif
      if (key.ne.esck) goto 300
      key = retk
      return
      end

C ***************************************************************************
CN    MODULE NAME       : calpos
CA    FUNCTION          : Calculate position of number on the screen, while
C                         polynomial is edited.
CS    CALL SEQUENCE     : call calpos(num,den,order,numden,x,y)
CI    INPUT PARAMETERS  : num - (integer*2) Order of numerator.
C                         den - (integer*2) Order of denominator.
C                         order - (integer*2) Current position of edit.
C                         numden - (integer*2) Numerator or denominator.
CO    OUTPUT PARAMETERS: x,y - (integer*2) Returned x,y co-ords of number.
CG    GLOBAL VARIABLES  : None.
CM    MODULES CALLED    : None.
CE    ERROR CONDITIONS  : ?
C
CC    COMMENTS          : Calculates the position of the number on the graphics
C                         screen so that the number can be edited. This routine
C                         uses the state variables of the polynomial editor to
C                         determine the position of the number. These states are
```

```
C                        :- Order of numerator,
C                           Order of denominator,
C                           Position of editor within the polynomial,
C                           Numerator or denominator.
C
C ***************************************************************************
      subroutine calpos(num,den,order,numden,xpos,ypos)
      integer*2 num,den,order,numden,xpos,ypos
      integer*2 xcent,xdelta,temp1,temp2

      xcent = 360
      xdelta = 12*9
      if (numden.eq.1) then
         if (order.gt.6) then
            xpos = 36 + (order-7)*xdelta
            ypos = 220
         else
            ypos = 135
            if (num.gt.6) then
               xpos = xcent - 3*xdelta
            else
               xpos = xcent - int ((real(num)/2.0)*real(xdelta))
            endif
            xpos = xpos + (order-1)*xdelta
         endif
      elseif (numden.eq.2) then
         if (order.gt.6) then
            xpos = 36 + (order-7)*xdelta
            ypos = 240
         else
            ypos = 155
            if (den.gt.6) then
               xpos = xcent - 3*xdelta
            else
               xpos = xcent - int ((real(den)/2.0)*real(xdelta))
            endif
            xpos = xpos + (order-1)*xdelta
         endif
      elseif (numden.eq.3) then
         if (num.lt.7) then
            xpos = xcent - int ((real(num)/2.0)*real(xdelta))
            xpos = xpos + (num-1)*xdelta
            ypos = 135
         else
            xpos = 36 + (num-7)*xdelta
            ypos = 220
         endif
      endif
      return
      end

C ***************************************************************************
CN    MODULE NAME       : prpow
CA    FUNCTION          : To print out the power of S in the polynomial.
CS    CALL SEQUENCE     : call prpow(page,x,y,power)
CI    INPUT PARAMETERS :  page - (integer*2) Page on which the power is to
C                                            be written.
C                           x,y - (integer*2) X,y co-ords of polynomial element.
C                         power - (integer*2) The power of the element.
CO    OUTPUT PARAMETERS: None.
CG    GLOBAL VARIABLES : None.
CM    MODULES CALLED   : gnum (for).
CE    ERROR CONDITIONS : ?
C
CC    COMMENTS         : Calculates  the  position of the power and then prints
C                        out the power using graphic numbers on the appropriate
C                        screen.
C
C ***************************************************************************
      subroutine prpow(pg,xpos,ypos,pow)
      integer*2 pg,xpos,ypos,pow
```

```
C
C     FILE              : CLR1.FOR
C ***************************************************************************
CN    MODULE NAME       : clrmat
CA    FUNCTION          : To drive the menus for clearing the matrices.
CS    CALL SEQUENCE     : call clrmat()
CI    INPUT PARAMETERS  : None.
CO    OUTPUT PARAMETERS: None.
CG    GLOBAL VARIABLES  : sysmat.inc
CM    MODULES CALLED    : clrall (for), clrm (for), DOMENU (asm), WIPSCR (asm),
C                         wrtitl (for).
CE    ERROR CONDITIONS  : ?
C
CC    COMMENTS          : Call subroutine to perform functions upon the users'
C                         request. The ESC key returns control to the calling
C                         routine.
C
C ***************************************************************************
      subroutine clrmat()
      implicit integer*2 (D)
$include: 'sysmat.inc'
      integer*2 opt115

      call WIPSCR(0)
      call wrtitl(0,280,35,18,'Clearing Matrices.')
99    continue
      opt115 = DOMENU(115)
      call WIPSCR(0)
      call wrtitl(0,280,35,18,'Clearing Matrices.')
      if (opt115.eq.1) then
         call clrm(gmat,4,'G(s)')
      elseif (opt115.eq.2) then
         call clrm(kmat,4,'K(s)')
      elseif (opt115.eq.3) then
         call clrall()
      endif
      if (opt115.gt.3) goto 99
      return
      end

C ***************************************************************************
CN    MODULE NAME       : clrall
CA    FUNCTION          : To clear all the system matrices.
CS    CALL SEQUENCE     : call clrall()
CI    INPUT PARAMETERS  : None.
CO    OUTPUT PARAMETERS: None.
CG    GLOBAL VARIABLES  : sysmat.inc
C                         keys.inc
CM    MODULES CALLED    : STRIN (asm), WRTSTR (asm).
CE    ERROR CONDITIONS  : ?
C
CC    COMMENTS          : The routine asks the user for confirmation of the
C                         clearing operation, only then  does  the  routine
C                         zero all the elements of the system matrices.
C
C ***************************************************************************
      subroutine clrall()
      implicit integer*2 (S)
$include: 'sysmat.inc'
$include: 'keys.inc'
      integer*2 key
      character*1 yesno

      yesno = 'n'
      call WRTSTR(0,40,60,55,
     +'Are you sure you want to clear all the matrices (y/n) ?')
      call WRTSTR(0,40,110,26,'Hit RETURN to enter reply.')
      call WRTSTR(0,40,125,16,'Hit ESC to exit.')
      key = STRIN(0,540,60,1,yesno)
      if (((yesno.eq.'y').or.(yesno.eq.'Y')).and.(key.eq.retk)) then
         call WRTSTR(0,40,80,30,'Clearing all the matrices ....')
         do 200 i = 1,10
```

```
            do 201 j = 1,10
               do 202 k = 1,13
                  kmat(i,j,1,k) = 0.0
                  kmat(i,j,2,k) = 0.0
                  gmat(i,j,1,k) = 0.0
                  gmat(i,j,2,k) = 0.0
202            continue
201         continue
200      continue
         call WRTSTR(0,310,80,9,'Complete.')
      else
         call WRTSTR(0,40,80,26,'Clear operation cancelled.')
      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     FILE              : CLR2.FOR
C ***************************************************************************
CN    MODULE NAME       : clrm
CA    FUNCTION          : To clear a system matrix.
CS    CALL SEQUENCE     : call clrm(mat,length,msg)
CI    INPUT PARAMETERS  : mat(10,10,2,13) - (real*4) The matrix to be cleared.
C                                 length - (integer*2) Length of message.
C                                    msg - (character*length) Message passed by
C                                                          calling routine.
CO    OUTPUT PARAMETERS: None.
CG    GLOBAL VARIABLES  : keys.inc
CM    MODULES CALLED    : STRIN (asm), WRTSTR (asm).
CE    ERROR CONDITIONS  : ?
C
CC    COMMENTS          : The routine first ask the user to confirm the clear
C                         operation, and only then zeros the elements of the
C                         matrix.
C
C ***************************************************************************
      subroutine clrm(mat,msglen,matmsg)
      implicit integer*2 (S)
      real*4 mat(10,10,2,13)
      integer*2 msglen
      character*25 matmsg
$include: 'keys.inc'
      integer*2 key
      character*1 yesno

      yesno = 'n'
      call WRTSTR(0,40,60,35,'Are you sure you want to clear the ')
      call WRTSTR(0,355,60,msglen,matmsg)
      call WRTSTR(0,(355+msglen*9),60,15,' matrix (y/n) ?')
      call WRTSTR(0,40,110,26,'Hit RETURN to enter reply.')
      call WRTSTR(0,40,125,16,'Hit ESC to exit.')
      key = STRIN(0,(500+msglen*9),60,1,yesno)
      if (((yesno.eq.'y').or.(yesno.eq.'Y')).and.(key.eq.retk)) then
         call WRTSTR(0,40,80,13,'Clearing the ')
         call WRTSTR(0,160,80,msglen,matmsg)
         call WRTSTR(0,(160+msglen*9),80,11,' matrix....')
         do 100 i = 1,10
            do 101 j = 1,10
               do 102 k = 1,13
                  mat(i,j,1,k) = 0.0
                  mat(i,j,2,k) = 0.0
102            continue
101         continue
100      continue
         call WRTSTR(0,(260+msglen*9),80,9,'Complete.')
      else
         call WRTSTR(0,40,80,26,'Clear operation cancelled.')
      endif
      return
      end

C ***************************************************************************
CN    MODULE NAME       : IDMAT
CA    FUNCTION          : To initialise a system matrix to an identity matrix.
CS    CALL SEQUENCE     : call idmat(mat,msg,length)
CI    INPUT PARAMETERS  : mat(10,10,2,13) - (real*4) The matrix to be cleared.
C                                    msg - (character*length) Message passed by
C                                                          calling routine.
C                                 length - (integer*2) Length of message.
CO    OUTPUT PARAMETERS: None.
CG    GLOBAL VARIABLES  : keys.inc
CM    MODULES CALLED    : STRIN (asm), WRTSTR (asm).
CE    ERROR CONDITIONS  : ?
C
CC    COMMENTS          : The routine first asks the user to confirm the
C                         initialise operation, and only then sets the matrix
C                         to an identity matrix.
C
```

```
C ***************************************************************************
      subroutine idmat(mat,msglen,matmsg)
      implicit integer*2 (S)
      real*4 mat(10,10,2,13)
      integer*2 msglen
      character*25 matmsg
$include: 'keys.inc'
      integer*2 key
      character*1 yesno

      yesno = 'n'
      call WRTSTR(0,40,60,40,'Are you sure you want to initialise the ')
      call WRTSTR(0,400,60,msglen,matmsg)
      call WRTSTR(0,(400+msglen*9),60,7,' matrix')
      call WRTSTR(0,40,75,29,'to an identity matrix (y/n) ?')
      call WRTSTR(0,40,120,26,'Hit RETURN to enter reply.')
      call WRTSTR(0,40,135,16,'Hit ESC to exit.')
      key = STRIN(0,310,75,1,yesno)
      if (((yesno.eq.'y').or.(yesno.eq.'Y')).and.(key.eq.retk)) then
         call WRTSTR(0,40,95,16,'Initialising the')
         call WRTSTR(0,190,95,msglen,matmsg)
         call WRTSTR(0,(190+msglen*9),95,11,' matrix....')
         do 100 i = 1,10
            do 101 j = 1,10
               do 102 k = 1,13
                  mat(i,j,1,k) = 0.0
                  mat(i,j,2,k) = 0.0
102            continue
               if (i.eq.j) then
                  mat(i,j,1,1) = 1.0
                  mat(i,j,2,1) = 1.0
               endif
101         continue
100      continue
         call WRTSTR(0,(290+msglen*9),95,9,'Complete.')
      else
         call WRTSTR(0,40,95,31,'Initialise operation cancelled.')
      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     FILE             : FILES.FOR
C ***************************************************************************
CN    MODULE NAME      : dofile
CA    FUNCTION         : Drives the menus for system files and paths.
CS    CALL SEQUENCE    : call dofile().
CI    INPUT PARAMETERS : None.
CO    OUTPUT PARAMETERS: None.
CG    GLOBAL VARIABLES : None.
CM    MODULES CALLED   : dir (for), DOMENU (asm), getfil (for), wrtitl (for),
C                        WIPSCR (asm)
CE    ERROR CONDITIONS : ?
C
CC    COMMENTS         : Calls subroutines to perform functions upon request
C                        of the user. The  ESC  key  returns  control to the
C                        calling routine.
C
C ***************************************************************************
      subroutine dofile()
      implicit integer*2 (D)
      integer*2 opt111
      call wrtitl(0,160,35,46,
     +     'System Filenames, Pathnames and Configuration.')
99    continue
      opt111 = DOMENU(111)
      if (opt111.eq.1) then
         call getfil()
      elseif (opt111.eq.2) then
         call dir()
      endif
      if (opt111.gt.1) goto 99
      call WIPSCR(0)
      return
      end

C ***************************************************************************
CN    MODULE NAME      : getfil
CA    FUNCTION         : To edit the system file and path names.
CS    CALL SEQUENCE    : call getfil()
CI    INPUT PARAMETERS : None.
CO    OUTPUT PARAMETERS: None.
CG    GLOBAL VARIABLES : fnames.inc
C                        keys.inc
CM    MODULES CALLED   : STRIN (asm), WIPSCR (asm), WRTSTR (asm), wrtitl (for).
CE    ERROR CONDITIONS : ?
C
CC    COMMENTS         : Permits the user to edit the system file and path
C                        names.
C
C ***************************************************************************
      subroutine getfil()
      implicit integer*2 (S)
$include: 'fnames.inc'
$include: 'keys.inc'
      integer*2 key,pos
      pos = 1
      call WIPSCR(0)
      call wrtitl(0,40,60,17,'System Filenames.')
      call WRTSTR(0,40,85,7,'G(s) : ')
      call WRTSTR(0,105,85,25,gfnam)
      call WRTSTR(0,40,110,7,'K(s) : ')
      call WRTSTR(0,105,110,25,kfnam)
      call wrtitl(0,350,60,17,'System Pathnames.')
      call WRTSTR(0,350,85,12,'Save path : ')
      call WRTSTR(0,460,85,25,outpth)
      call WRTSTR(0,350,110,12,'Load path : ')
      call WRTSTR(0,460,110,25,inpath)
      call WRTSTR(0,40,160,44,
     +            'Use cursor, tab keys and RETURN key to move.')
      call WRTSTR(0,40,175,16,'ESC key to exit.')
199   continue
      if (pos.eq.1) then
```

```
          key = STRIN(0,105,85,25,gfnam)
       elseif (pos.eq.2) then
          key = STRIN(0,460,85,25,outpth)
       elseif (pos.eq.3) then
          key = STRIN(0,105,110,25,kfnam)
       elseif (pos.eq.4) then
          key = STRIN(0,460,110,25,inpath)
       endif
       if (key.eq.downk) then
          pos = pos + 2
          if (pos.ge.6) then
             pos = 2
          elseif (pos.gt.4) then
             pos = 1
          endif
       elseif ((key.eq.tabk).or.(key.eq.retk)) then
          pos = pos + 1
          if (pos.gt.4) pos = 1
       elseif (key.eq.rtabk) then
          pos = pos - 1
          if (pos.lt.1) pos = 4
       elseif (key.eq.upk) then
          pos = pos - 2
          if (pos.eq.0) then
             pos = 4
          elseif (pos.lt.1) then
             pos = 4
          endif
       endif
       if (key.ne.esck) goto 199
       call WIPSCR(0)
       return
       end

C ******************************************************************************
CN     MODULE NAME       : dir
CA     FUNCTION          : Permits the user to list directories.
CS     CALL SEQUENCE     : call dir()
CI     INPUT PARAMETERS  : None.
CO     OUTPUT PARAMETERS: None.
CG     GLOBAL VARIABLES  : keys.inc
CM     MODULES CALLED    : BLKFIL (asm), FFIRST (asm), FNEXT (asm),  INKEY (asm),
C                          LENSTR (asm), LEVEL (asm), prderr (for), prtnum (for),
C                          RJUST (asm), STRIN (asm), WIPSCR (asm),  wrtitl (for),
C                          WRTSTR (asm), RSTERR (asm), GETERR (asm).
CE     ERROR CONDITIONS  : ?
C
CC     COMMENTS          : Enables the user to list directories : wild characters
C                          are  allowed  in  the  directory  specifications. This
C                          does perform error checking on the  drive  to  prevent
C                          the system from crashing with a disk error.
C
C ******************************************************************************
       subroutine dir()
       implicit integer*2 (C,D,F,G,I,L,S)
$include: 'keys.inc'
       character*31 dirstr
       character*15 retstr
       integer*2 key,len,len2,find,xpos,ypos,fcount,pos,derr,drive
       dirstr = 'a:*.*'
       call WIPSCR(0)
       call wrtitl(0,70,25,17,'Directory Search.')
       call WRTSTR(0,40,45,33,'Enter search string or filename :')
       call WRTSTR(0,40,85,31,'Hit RETURN to search for files.')
       call WRTSTR(0,40,100,16,'Hit ESC to exit.')
       key = STRIN(0,350,45,30,dirstr)
       if (key.eq.retk) then
          call WRTSTR(0,40,45,65,
     +    '
     +    ')
          call WRTSTR(0,40,85,31,'                                        ')
          call WRTSTR(0,40,100,16,'                ')
```

```
         call wrtitl(0,20,45,15,'Search string :')
         call RJUST(30,dirstr)
         len = LENSTR(30,dirstr)
         call WRTSTR(0,165,45,len,dirstr)
         call LEVEL(2)
         call BLKFIL(165,48,(len*9),14)
         call LEVEL(1)
         call WRTSTR(0,40,265,16,'Hit ESC to exit.')

         fcount = 0
         key = 0
         xpos = 20
         ypos = 65
         call RSTERR()
         find = FFIRST(len,dirstr,len2,retstr)
         if (find.eq.0) then
102         continue
            fcount = fcount + 1
            call WRTSTR(0,xpos,ypos,len2,retstr)
            xpos = xpos + 140
            if (xpos.gt.680) then
               xpos = 20
               ypos = ypos + 15
            endif
            if (ypos.gt.250) then
               call WRTSTR(0,400,265,23,'Hit RETURN to continue.')
103            continue
               key = INKEY(1)
               if ((key.ne.esck).and.(key.ne.retk)) goto 103
               if (key.eq.retk) then
                  call WIPSCR(0)
                  call wrtitl(0,70,25,17,'Directory Search.')
                  call wrtitl(0,20,45,15,'Search string :')
                  len = LENSTR(30,dirstr)
                  call WRTSTR(0,165,45,len,dirstr)
                  call LEVEL(2)
                  call BLKFIL(165,48,(len*9),14)
                  call LEVEL(1)
                  call WRTSTR(0,40,265,16,'Hit ESC to exit.')
                  xpos = 20
                  ypos = 65
               endif
            endif
            if (key.eq.esck) goto 104
            call RSTERR()
            find = FNEXT(len2,retstr)
            if (find.eq.0) goto 102
104         continue
         endif
         derr = GETERR()
         if (derr.ne.-1) then
            call prderr(0,40,75,derr)
         endif
         if (key.ne.esck) then
            call prtnum(0,395,45,1,4,'(i4)',4,fcount)
            call WRTSTR(0,440,45,12,'Files found.')
101         continue
            key = INKEY(1)
            if (key.ne.esck) goto 101
         endif
      else
         call WRTSTR(0,40,65,27,'Search operation cancelled.')
      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     FILE              : FMAT.FOR
C ***************************************************************************
CN    MODULE NAME       : savef
CA    FUNCTION          : To save matrix information to disk.
CS    CALL SEQUENCE     : call savef(n,mat,mname,fname,funit)
CI    INPUT PARAMETERS  :                n - (integer*2) Order of the system.
C                         mat(10,10,2,13) - (real*4) The matrix to be saved.
C                                   mname - (character*25) The matrix title.
C                                   fname - (character*25) The matrix filename.
C                                  defdir - (character*4) Default directory
C                                           search string.
C                                   funit - (integer*2) The drive unit number.
CO    OUTPUT PARAMETERS: None.
CG    GLOBAL VARIABLES : keys.inc
C                         sysmat.inc
C                         fnames.inc
CM    MODULES CALLED    : BLKFIL (asm), dodir (for), RSTERR (asm), GETERR (asm),
C                         ERTONE (asm), LEVEL (asm), maknam (for), prderr (for),
C                         savmat (for), STRIN (asm), wrtitl (for), WRTSTR (asm).
CE    ERROR CONDITIONS  : ?
C
CC    COMMENTS          : Opens the matrix  file, first prompting  the user to
C                         check the file name. The routine performs  all  disk
C                         checks. If any errors, error  messages  are  printed
C                         and the routine quits, otherwise a routine is called
C                         to save the matrix information.
C
C ***************************************************************************
      subroutine savef(n,mat,mname,fname,defdir,funit)
      implicit integer*2 (c,G,L,S)
      integer*2 n
      real*4 mat(10,10,2,13)
      character*25 mname,fname
      character*4 defdir
      integer*2 funit
$include: 'keys.inc'
$include: 'sysmat.inc'
$include: 'fnames.inc'
      integer*2 key,ferr,derr,serr,len
      character*1 yesno
      character*50 usenam
      call maknam(fname,outpth,usenam)
      call wrtitl(0,40,60,22,'Saving Matrix to File.')
      call wrtitl(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.')
      call dodir(0,180,4,defdir,outpth)
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(0,215,110,25,usenam)
               open(funit,FILE=usenam,STATUS='NEW',IOSTAT=ferr)
               if (ferr.eq.0) then
                  serr = savmat(n,mat,mname,funit)
                  close(funit)
                  if (serr.eq.0) then
                     call WRTSTR(0,40,130,22,'Matrix save completed.')
                  else
                     call WRTSTR(0,40,130,38,
     +                    '*** Error : Matrix save not completed.')
                  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='NEW',IOSTAT=ferr)
               if (ferr.eq.0) then
                  serr = savmat(n,mat,mname,funit)
                  close(funit)
                  if (serr.eq.0) then
                     call WRTSTR(0,40,170,22,
     +                    'Matrix save completed.')
                  else
                     call WRTSTR(0,40,170,38,
     +                    '*** Error : Matrix 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(0,40,170,32,
     +                 'Matrix save operation cancelled.')
            endif
         endif
      else
         call WRTSTR(0,40,170,32,
     +              'Matrix save operation cancelled.')
      endif
      close(funit)
      return
      end

C ********************************************************************************
CN    MODULE NAME       : savmat
CA    FUNCTION          : To save the matrix data to disk.
CS    CALL SEQUENCE     : error = savmat(n,mat,mname,funit)
CI    INPUT PARAMETERS :                n - (integer*2) Order of the system.
C                        mat(10,10,2,13) - (real*4) The matrix to be saved.
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         : 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
C ***************************************************************************
      integer*2 function savmat(n,mat,mname,funit)
      integer*2 n
      real*4 mat(10,10,2,13)
      character*25 mname
      integer*2 funit
      integer*2 maxord,ordr,chkio
      maxord = 0
      do 999 i = 1,n
         do 998 j = 1,n
            if (int (mat(i,j,1,13)).gt.maxord) then
               maxord = int (mat(i,j,1,13))
            endif
            if (int (mat(i,j,2,13)).gt.maxord) then
               maxord = int (mat(i,j,2,13))
            endif
998      continue
999   continue

      write(funit,'(3i6)',iostat=chkio,err=200) maxord,n,n
      if (chkio.eq.0) then
         do 997 i = 1,n
            do 996 j = 1,n
               write(funit,'(g10.4)',iostat=chkio,err=200)
     +               mat(i,j,1,12)
               ordr = int (mat(i,j,1,13))
               write(funit,'(12g10.4)',iostat=chkio,err=200)
     +               real (ordr),(mat(i,j,1,k), k=1,(ordr+1))
               ordr = int (mat(i,j,2,13))
               write(funit,'(12g10.4)',iostat=chkio,err=200)
     +               real (ordr),(mat(i,j,2,k), k=1,(ordr+1))
996         continue
997      continue
         write(funit,'(a)',iostat=chkio,err=200) '; Matrix name :'
         write(funit,'(a)',iostat=chkio,err=200) '; *************'

         write(funit,'(a)',iostat=chkio,err=200) mname
      endif

      if (chkio.eq.0) goto 201
200   call ERTONE()
201   continue
      savmat = chkio
      return
      end

C ***************************************************************************
CN    MODULE NAME      : loadf
CA    FUNCTION         : To load matrix information from disk.
CS    CALL SEQUENCE    : loadf(n,mat,mname,fname,funit)
CI    INPUT PARAMETERS :              n - (integer*2) Order of the system.
C                       mat(10,10,2,13) - (real*4) The matrix to be saved.
C                            mname - (character*25) The matrix title.
C                            fname - (character*25) The matrix filename.
C                           defdir - (character*4) Default directory
C                                    search string.
C                            funit - (integer*2) The drive unit number.
CO    OUTPUT PARAMETERS: None.
CG    GLOBAL VARIABLES : keys.inc
C                        sysmat.inc
C                        fnames.inc
CM    MODULES CALLED   : dodir (for), RSTERR (asm), GETERR (asm), ERTONE (asm),
C                        lodmat (for), maknam (for), prderr (for), STRIN (asm),
C                        wrtitl (for), WRTSTR (asm).
CE    ERROR CONDITIONS : ?
```

```
C
CC    COMMENTS          : Opens the matrix  file, first prompting  the user to
C                         check the file name. The routine performs  all  disk
C                         checks. If any errors, error  messages  are  printed
C                         and the routine quits, otherwise a routine is called
C                         to load the matrix information.
C
C ***************************************************************************
      subroutine loadf(n,mat,mname,fname,defdir,funit)
      implicit integer*2 (c,G,l,S)
      integer*2 n
      real*4 mat(10,10,2,13)
      character*25 mname,fname
      character*4 defdir
      integer*2 funit
$include: 'keys.inc'
$include: 'sysmat.inc'
$include: 'fnames.inc'
      integer*2 key,ferr,derr,lerr
      character*50 usenam
      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.')
      call dodir(0,180,4,defdir,inpath)
99    continue
      call maknam(fname,inpath,usenam)
      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 ....')
            lerr = lodmat(n,mat,mname,funit)
            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.')
            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
      return
      end

C ***************************************************************************
CN    MODULE NAME       : lodmat
CA    FUNCTION          : To load matrix data from disk.
CS    CALL SEQUENCE     : error = lodmat(n,mat,mname,funit)
CI    INPUT PARAMETERS :                n - (integer*2) Order of the system.
C                         mat(10,10,2,13) - (real*4) The matrix to be saved.
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 ***************************************************************************
      integer*2 function lodmat(n,mat,mname,funit)
      integer*2 n
      real*4 mat(10,10,2,13)
      character*25 mname
      integer*2 funit
      integer*2 maxord,n2,chkio
      character*1 chkch

      do 250 i = 1,10
         do 251 j = 1,10
            do 252 k = 1,13
               mat(i,j,1,k) = 0.0
               mat(i,j,2,k) = 0.0
252         continue
251      continue
250   continue

      read(funit,'(3i6)',iostat=chkio,err=300) maxord,n,n2
      if (chkio.eq.0) then
         do 897 i = 1,n
            do 896 j = 1,n
               read(funit,'(g10.4)',iostat=chkio,err=300)
     +              mat(i,j,1,12)
               read(funit,'(12g10.4)',iostat=chkio,err=300)
     +             mat(i,j,1,13),
     +             (mat(i,j,1,k), k=1,(int(mat(i,j,1,13))+1))
               read(funit,'(12g10.4)',iostat=chkio,err=300)
     +             mat(i,j,2,13),
     +             (mat(i,j,2,k), k=1,(int(mat(i,j,2,13))+1))
896         continue
897      continue
      endif
      if (chkio.eq.0) goto 301
300   call ERTONE()
301   continue
      lodmat = chkio
      return
      end

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 ******************************************************************************
CN    MODULE NAME       : maknam
CA    FUNCTION          : To construct a filename, including path names.
CS    CALL SEQUENCE     : call maknam(fname,pname,usenam)
CI    INPUT PARAMETERS  : fname - (character*25) The file name.
C                         mname - (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
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     FILEND           :
```

```
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),  wrtitl (for), LENSTR (asm),
C                        COPYST (asm), RSTERR (asm), GETERR (asm), prtnum (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
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 wrtitl(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 prtnum(0,495,ylimit+15,1,4,'(i2)',2,fcount)
```

```
            call WRTSTR(0,525,ylimit+15,10,'printed / ')
            call prtnum(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     FILEND           :
```

```
C
C     FILE              : PRPOLY.FOR
C ***************************************************************************
CN    MODULE NAME       : CHPRT
CA    FUNCTION          : To check if the user wishes to print out the matrix.
CS    CALL SEQUENCE     : call chprt(n,mat,name,filnam)
CI    INPUT PARAMETERS  :
C                    n - (integer*2) Order of the system.
C      mat(10,10,2,13) - (real*4) The matrix to be printed.
C                 name - (character*25) The matrix title.
C               filnam - (character*25) The matrix filename.
C
CO    OUTPUT PARAMETERS: None.
CG    GLOBAL VARIABLES  : keys.inc
CM    MODULES CALLED    : WIPSCR (asm), wrtitl (for), WRTSTR (asm), INKEY (asm),
C                         prpoly (for).
CE    ERROR CONDITIONS  : ?
C
CC    COMMENTS          : Checks if the user wishes to print the matrix, if so
C                         the appropriate routines are called.
C                         ESC returns control to the calling routine.
C ***************************************************************************
      subroutine chprt(n,mat,name,filnam)
      implicit integer*2 (G,I)
      integer*2 n
      real*4 mat(10,10,2,13)
      character*25  name,filnam
$include: 'keys.inc'
      call WIPSCR(0)
      call wrtitl(0,250,40,31,'Printing Matrix of Polynomials.')
      call WRTSTR(0,40,70,27,'Hit RETURN to print matrix,')
      call WRTSTR(0,40,85,15,'or ESC to quit.')
      call WRTSTR(0,40,100,32,'Hit CTRL-Q to quit during print.')
99    continue
      key = INKEY(1)
      if ((key.ne.esck).and.(key.ne.retk).and.(key.ne.4352)) goto 99
      if (key.eq.retk) then
         call WRTSTR(0,40,125,17,'Print started....')
         call prpoly(n,mat,name,filnam)
      else
         call WRTSTR(0,40,125,16,'Print cancelled.')
      endif
      call WIPSCR(0)
      return
      end

C ***************************************************************************
CN    MODULE NAME       : PRPOLY
CA    FUNCTION          : To print out the matrix of polynomials.
CS    CALL SEQUENCE     : call prpoly(n,mat,name,filnam)
CI    INPUT PARAMETERS  :
C                    n - (integer*2) Order of the system.
C      mat(10,10,2,13) - (real*4) The matrix to be printed.
C                 name - (character*25) The matrix title.
C               filnam - (character*25) The matrix filename.
C
CO    OUTPUT PARAMETERS: None.
CG    GLOBAL VARIABLES  : keys.inc
C                         fnames.inc
CM    MODULES CALLED    : INKEY (asm), prtitl (for), preqtn (for), ERTONE (asm).
CE    ERROR CONDITIONS  : ?
C
CC    COMMENTS          : Opens the printer file and if there are no errors, the
C                         routine starts to call subroutines to print  headings,
C                         and the polynomials.
C                         The user can quit the print by entering : CTRL-Q.
C ***************************************************************************
      subroutine prpoly(n,mat,name,filnam)
      implicit integer*2 (G,I)
      integer*2 n
      real*4 mat(10,10,2,13)
      character*25  name,filnam
```

```
$include: 'keys.inc'
$include: 'fnames.inc'

      integer*2 i,j,key,prpg,linum,chkio,ferr,derr,flush,ctrlq
      linum = 0
      prpg = 1
      ctrlq = 4352
      flush = INKEY(2)
      call RSTERR()
      open(1,FILE='PRN',IOSTAT=ferr)
      if (ferr.eq.0) then
         call prtitl(n,prpg,name,filnam)
         linum = 0
         key = 0
         do 999 i = 1,n
           do 998 j = 1,n
              if (key.eq.0) then
                 key = INKEY(4)
                 if (key.ne.ctrlq) key = 0
              endif
              if (key.eq.0) then
                 if ((56-linum).lt.9) then
                    prpg = prpg + 1
                    call prtitl(n,prpg,name,filnam)
                    linum = 0
                 endif
                 call preqtn(i,j,mat,linum)
              endif
998         continue
999      continue
      endif
      if (chkio.eq.0) goto 201
200   continue
      call ERTONE()
201   continue
      close(1)
      return
      end

C ************************************************************************
CN    MODULE NAME      : PRTITL
CA    FUNCTION         : To print the page header.
CS    CALL SEQUENCE    : call prtitl(n,prpg,name,filnam)
CI    INPUT PARAMETERS :
C                   n - (integer*2) Order of the system.
C                prpg - (integer*2) The page number.
C                name - (character*25) The matrix title.
C              filnam - (character*25) The matrix filename.
C
CO    OUTPUT PARAMETERS: None.
CG    GLOBAL VARIABLES : None.
CM    MODULES CALLED   : None.
CE    ERROR CONDITIONS : ?
C
CC    COMMENTS         : Prints a form feed and then prints the page header.
C ************************************************************************
      subroutine prtitl(n,prpg,name,filnam)
      integer*2 n,prpg
      character*25 name,filnam

      integer*2 chkio
      write(1,"(a)',iostat=chkio,err=200) '1'
      write(1,10,iostat=chkio,err=200) name,filnam,prpg
10    format(' Title : ',a,' Filename : ',a,' Pg : ',i3)
      write(1,11,iostat=chkio,err=200)
11    format(' -------                                  ----------
     +              --------')
      write(1,12,iostat=chkio,err=200) n
12    format(' System Order = ',i4)
200   continue
      return
      end
```

```
C ******************************************************************************
CN     MODULE NAME       : PREQTN
CA     FUNCTION          : Prints a polynomial on the printer.
CS     CALL SEQUENCE     : call preqtn(row,col,mat,linum)
CI     INPUT PARAMETERS :
C                   row
C                   col - (integer*2) Position of polynomial in the matrix.
C       mat(10,10,2,13) - (real*4) The matrix.
C                 linum - (integer*2) The line count.
C
CO     OUTPUT PARAMETERS: None.
CG     GLOBAL VARIABLES : sysnms.inc
CM     MODULES CALLED   : prsym (for).
CE     ERROR CONDITIONS : ?
C
CC     COMMENTS          : Prints the input and output names, the dead time and
C                          the numerator and denominator as seperate units.
C ******************************************************************************
       subroutine preqtn(row,col,mat,linum)
       integer*2 row,col
       real*4 mat(10,10,2,13)
       integer*2 linum
$include: 'sysnms.inc'
       integer*2 chkio
       write(1,'(a)',iostat=chkio,err=200) '  '
       write(1,10,iostat=chkio,err=200) row,col
10     format(' Element: (',i2,',',i2,')')
       write(1,13,iostat=chkio,err=200) inpnms(row),outnms(col)
13     format(' Input : ',a,'  Output : ',a)
       write(1,11,iostat=chkio,err=200) mat(row,col,1,12)
11     format(' Dead Time : ',g10.4)
       call prsym(row,col,1,mat,'Numerator  ',linum)
       call prsym(row,col,2,mat,'Denominator',linum)
       write(1,12,iostat=chkio,err=200)
12     format(' ------------------------------------------------------
      +----------------------')
       linum = linum + 7
200    continue
       return
       end

C ******************************************************************************
CN     MODULE NAME       : PRSYM
CA     FUNCTION          : To print a single polynomial eg. numerator.
CS     CALL SEQUENCE     : call prsym(row,col,numden,mat,name,linum)
CI     INPUT PARAMETERS :
C                   row
C                   col - (integer*2) The position of the polynomial.
C                numden - (integer*2) Numerator or denominator.
C       mat(10,10,2,13) - (real*4) The matrix.
C                  name - (character*25) The matrix title.
C                 linum - (integer*2) The line count.
C
CO     OUTPUT PARAMETERS: None.
CG     GLOBAL VARIABLES : None.
CM     MODULES CALLED   : None.
CE     ERROR CONDITIONS : ?
C
CC     COMMENTS          : Prints out the numerator or denominator on the printer
C ******************************************************************************
       subroutine prsym(row,col,numden,mat,name,linum)
       integer*2 row,col,numden
       real*4 mat(10,10,2,13)
       character*11 name
       integer*2 linum

       integer*2 i,order,chrs,el,chkio
       write(1,100,iostat=chkio,err=200) name
100    format(' ',a,':',\)
       order = int(mat(row,col,numden,13))
       chrs = 0
```

```
      do 99 i = (order+1),1,-1
         if ((79-chrs).lt.17) then
            write(1,'(a)',iostat=chkio,err=200) '  '
            write(1,101,iostat=chkio,err=200)
101         format('           ',\)
           chrs = 0
           linum = linum + 1
        endif
         el = i - 1
        if (el.ge.10) then
            write(1,102,iostat=chkio,err=200) mat(row,col,numden,i),el
        else
            write(1,103,iostat=chkio,err=200) mat(row,col,numden,i),el
        endif
102      format('  (',g10.4,')S^',i2,\)
103      format('  (',g10.4,')S^',i1,\)
         chrs = chrs + 17
99    continue
      write(1,'(a)',iostat=chkio,err=200) '  '
      linum = linum + 1
200   continue
      return
      end

C ********************************************************************************
CH    REVISION HISTORY :
C     VERSION       BY                DATE        COMMENT
C     1.00          Ian Fisher        23/06/88    Creation.
C     FILEND           :
```

```
C
C     FILE              : DESIGN.FOR
C ***************************************************************************
CN    MODULE NAME       : design
CA    FUNCTION          : To drive the design section of the menu.
CS    CALL SEQUENCE     : call design()
CI    INPUT PARAMETERS  : None.
C
CO    OUTPUT PARAMETERS: None.
C
CG    GLOBAL VARIABLES  : None.
CM    MODULES CALLED    : DOMENU (asm), frange (for),  getk (for), newmat (for),
C                         plots (for), scales (for), selaxe (for), WIPSCR (asm),
C                         wrtitl (for), WRTSTR (asm).
C
CE    ERROR CONDITIONS  : Bad compile and link, otherwise dunno.
C
CC    COMMENTS          : Controls the design menu, calling appropriate routines
C                         upon request of the user.
C
C ***************************************************************************
      subroutine design()
      implicit integer*2 (D)

      integer*2 opt12

99    continue
      call wrtitl(0,170,35,45,
     +           'Design Controllers using Characteristic Loci.')
      opt12 = DOMENU(12)
      call WRTSTR(0,170,35,46,
     +           '                                                ')
      if (opt12.eq.1) then
         call plots()
      elseif (opt12.eq.2) then
         call getk()
      elseif (opt12.eq.3) then
         call frange()
      elseif (opt12.eq.4) then
         call scales()
      elseif (opt12.eq.5) then
         call selaxe()
      elseif (opt12.eq.6) then
         call newmat()
      endif
      if (opt12.ne.0) goto 99
      call WIPSCR(0)
      return
      end

C ***************************************************************************
CN    MODULE NAME       : plots
CA    FUNCTION          : To initiate plotting procedures.
CS    CALL SEQUENCE     : call plots()
CI    INPUT PARAMETERS  : None.
C
CO    OUTPUT PARAMETERS: None
C
CG    GLOBAL VARIABLES  : plotpg.inc
C                         plotfr.inc
CM    MODULES CALLED    : DOMENU (asm), doplot (for), WIPSCR (asm), BOX (asm),
C                         wrtitl (for), WRTSTR (asm), prtnum (for).
C
CE    ERROR CONDITIONS  : ?
C
CC    COMMENTS          : Displays the current graphics page settings (1 and 2)
C                         (which are user definable). The routine also requests
C                         the user to select which set of user defined plot page
C                         parameters is to be used while plotting.
C
C ***************************************************************************
      subroutine plots()
```

```
      implicit integer*2 (c,D)

$include: 'plotpg.inc'
$include: 'plotfr.inc'
      integer opt121,numpts

      call WIPSCR(0)
      call wrtitl(0,250,20,31,'Plotting System Characteristics.')
      call wrtitl(0,40,50,11,'Setting 1 :')
      call WRTSTR(0,145,50,27,'Loop Characteristic      : ')
      call WRTSTR(0,145,70,27,'Characteristic Loci Page : ')
      call WRTSTR(0,145,90,22,'Bode and Misalignment ')
      call WRTSTR(0,145,105,27,'Angles Page             : ')
      call BOX(30,120,485,90)
      if (pmat1.eq.1) then
         call WRTSTR(0,390,50,4,'G(s)')
      else
         call WRTSTR(0,390,50,13,'Q(s)=K(s)G(s)')
      endif
      if (plset1.le.1) then
         call prtnum(0,390,70,1,4,'(i1)',1,plset1)
      else
         call WRTSTR(0,390,70,10,'No Display')
      endif
      if (pmset1.le.1) then
         call prtnum(0,390,105,1,4,'(i1)',1,pmset1)
      else
         call WRTSTR(0,390,105,10,'No Display')
      endif

      call wrtitl(0,40,165,11,'Setting 2 :')
      call WRTSTR(0,145,165,27,'System to be plotted     : ')
      call WRTSTR(0,145,185,27,'Characteristic Loci Page : ')
      call WRTSTR(0,145,205,22,'Bode and Misalignment ')
      call WRTSTR(0,145,220,27,'Angles Page             : ')
      call BOX(30,235,485,90)
      if (pmat2.eq.1) then
         call WRTSTR(0,390,165,4,'G(s)')
      else
         call WRTSTR(0,390,165,13,'Q(s)=K(s)G(s)')
      endif
      if (plset2.le.1) then
         call prtnum(0,390,185,1,4,'(i1)',1,plset2)
      else
         call WRTSTR(0,390,185,10,'No Display')
      endif
      if (pmset2.le.1) then
         call prtnum(0,390,220,1,4,'(i1)',1,pmset2)
      else
         call WRTSTR(0,390,220,10,'No Display')
      endif
      call BOX(520,235,190,205)
      call wrtitl(0,530,50,15,'Frequency Sweep')
      call WRTSTR(0,530,75,8,'Units : ')
      call WRTSTR(0,530,95,8,'Start : ')
      call WRTSTR(0,530,115,8,'Stop  : ')
      call wrtitl(0,530,140,9,'Increment')
      if (inctyp.eq.1) then
         call prtnum(0,530,160,1,4,'(i3)',3,incpts)
         call WRTSTR(0,565,160,14,'[Points/Range]')
      elseif (inctyp.eq.2) then
         call prtnum(0,530,160,1,4,'(i3)',3,incpts)
         call WRTSTR(0,565,160,15,'[Points/Decade]')
      elseif (inctyp.eq.3) then
         call prtnum(0,530,160,1,4,'(i3)',3,incpts)
         call WRTSTR(0,565,160,15,'[Points/Octave]')
      endif
      if (ftype.eq.1) then
         call WRTSTR(0,605,75,7,'[Hertz]')
         call prtnum(0,605,95,3,7,'(g10.4)',10,(fstart/(6.2831853)))
         call prtnum(0,605,115,3,7,'(g10.4)',10,(fstop/(6.2831853)))
         if (inctyp.eq.4) then
```

```
            call WRTSTR(0,530,180,8,'Step  : ')
            call prtnum(0,605,180,3,7,'(g10.4)',10,(increm/(6.2831853)))
            call WRTSTR(0,530,160,15,'Units : [Hertz]')
         endif
      elseif (ftype.eq.2) then
         call WRTSTR(0,605,75,9,'[Rad/Sec]')
         call prtnum(0,605,95,3,7,'(g10.4)',10,fstart)
         call prtnum(0,605,115,3,7,'(g10.4)',10,fstop)
         if (inctyp.eq.4) then
            call WRTSTR(0,530,180,8,'Step  : ')
            call prtnum(0,605,180,3,7,'(g10.4)',10,increm)
            call WRTSTR(0,530,160,17,'Units : [Rad/Sec]')
         endif
      else
         call WRTSTR(0,605,75,10,'[Rad/hour]')
         call prtnum(0,605,95,3,7,'(g10.4)',10,(fstart/3600.0))
         call prtnum(0,605,115,3,7,'(g10.4)',10,(fstop/3600.0))
         if (inctyp.eq.4) then
            call WRTSTR(0,530,180,8,'Step  : ')
            call prtnum(0,605,180,3,7,'(g10.4)',10,(increm/(3600.0)))
            call WRTSTR(0,530,160,18,'Units : [Rad/Hour]')
         endif
      endif
      numpts = calfre()
      numpts = 0
      do 280 i = 1,300
         if (freqs(i).ne.(-1.0)) then
            numpts = numpts + 1
         endif
280   continue
      call wrtitl(0,530,205,14,'Total Points :')
      call prtnum(0,665,205,1,4,'(i3)',3,numpts)

299   continue
      opt121 = DOMENU(121)
      if (opt121.eq.1) then
         call doplot(1)
      elseif (opt121.eq.2) then
         call doplot(2)
      endif
      call WIPSCR(0)
      return
      end

C ***************************************************************************
CN    MODULE NAME      : frange
CA    FUNCTION         : To permit the user to edit the frequency sweep
C                        parameters.
CS    CALL SEQUENCE    : call frange()
CI    INPUT PARAMETERS : None.
C
CO    OUTPUT PARAMETERS: None.
C
CG    GLOBAL VARIABLES : plotpg.inc
C                        keys.inc
C                        plotfr.inc
CM    MODULES CALLED   : ERTONE (asm), INKEY (asm), getnum (for), prange (for),
C                        prftyp (for), prityp (for), WIPSCR (asm), wrtitl (for)
C                        WRTSTR (asm).
C
CE    ERROR CONDITIONS : ?
C
CC    COMMENTS         : This routine permits the user to edit the frequency
C                        sweep parameters such as :- start and end frequencies
C                        defined in hertz, radians/sec or radians/hour,
C                        frequency increment type, etc.
C                        This information is stored in the variables defined
C                        in PLOTPG.INC.
C
C
C ***************************************************************************
      subroutine frange()
```

```
      implicit integer*2 (c,I,g)
$include: 'plotpg.inc'
$include: 'keys.inc'
$include: 'plotfr.inc'

      integer*2 pos,key,ferr

      pos = 1
      call WIPSCR(0)
      call wrtitl(0,250,35,30,'Frequency Ranges for Plotting.')
      call WRTSTR(0,40,60,17,'Range type      :')
      call WRTSTR(0,40,80,17,'Frequency range :')
      call WRTSTR(0,300,80,2,'to')
      call WRTSTR(0,40,105,17,'Increment type  :')
      call prange()
      call prityp(inctyp,1)
199   continue
      call WRTSTR(0,40,200,57,
     +'                                                          ')
      call WRTSTR(0,40,215,57,
     +'                                                          ')

      call WRTSTR(0,40,200,36,'RETURN, TAB and cursor keys to move.')
      call WRTSTR(0,40,215,16,'ESC key to exit.')
      if (pos.eq.1) then
         call prftyp(ftype,2)
         call WRTSTR(0,40,200,57,
     +    'SPACE to select type. RETURN and TAB keys to move option.')
198      continue
         key = INKEY(1)
         if (key.eq.8192) then
            ftype = ftype + 1
            if (ftype.gt.3) ftype = 1
            call prftyp(ftype,3)
            call prange()
         endif
         if ((key.ne.tabk).and.(key.ne.retk).and.(key.ne.rtabk).and.
     +      (key.ne.esck)) goto 198
         call prftyp(ftype,1)
      elseif (pos.eq.2) then
190      continue
         key = getnum(0,200,80,3,7,'(g10.4)',10,fstar2)
         if (ftype.eq.1) then
            fstart = fstar2*2.0*3.14159265
         elseif (ftype.eq.2) then
            fstart = fstar2
         elseif (ftype.eq.3) then
            fstart = fstar2*3600.0
         endif
         if (fstart.lt.(0.0)) then
            call WRTSTR(0,40,150,33,
     +                  '*** Error : Start frequency < 0.0')
            call ERTONE()
         else
            call WRTSTR(0,40,150,33,
     +                  '                                    ')
         endif
         if (fstart.lt.(0.0)) goto 190
      elseif (pos.eq.3) then
197      continue
         key = getnum(0,335,80,3,7,'(g10.4)',10,fstop2)
         if (ftype.eq.1) then
            fstop = fstop2*2.0*3.14159265
         elseif (ftype.eq.2) then
            fstop = fstop2
         elseif (ftype.eq.3) then
            fstop = fstop2*3600.0
         endif
         if (fstop.lt.(0.0)) then
            call WRTSTR(0,40,150,31,'*** Error : End frequency < 0.0')
            call ERTONE()
         else
```

```
            call WRTSTR(0,40,150,31,'                                    ')
         endif
         if (fstop.lt.(0.0)) goto 197
         if (fstop.le.fstart) then
            call WRTSTR(0,40,150,44,
     +      '*** Error : End frequency ≤ Start frequency.')
            call ERTONE()
         else
            call WRTSTR(0,40,150,44,
     +      '                                                ')
         endif
         if (fstop.le.fstart) goto 197
       elseif (pos.eq.4) then
         call WRTSTR(0,40,200,57,
     +   'SPACE to select type. RETURN and TAB keys to move option.')
         call prityp(inctyp,2)
192      continue
         key = INKEY(1)
         if (key.eq.8192) then
            inctyp = inctyp + 1
            if (inctyp.gt.4) inctyp = 1
            call prityp(inctyp,3)
            call prange()
         endif
         if ((key.ne.tabk).and.(key.ne.retk).and.(key.ne.rtabk).and.
     +      (key.ne.esck)) goto 192
         call prityp(inctyp,1)
       elseif (pos.eq.5) then
         if (inctyp.eq.4) then
196         continue
            key = getnum(0,200,125,1,7,'(g10.4)',10,incre2)
            if (ftype.eq.1) then
               increm = incre2*2.0*3.14159265
            elseif (ftype.eq.2) then
               increm = incre2
            elseif (ftype.eq.3) then
               increm = incre2*3600.0
            endif
            if (increm.le.(0.0)) then
               call WRTSTR(0,40,150,26,'*** Error : Increment < 0.')
               call ERTONE()
            else
               call WRTSTR(0,40,150,26,'                          ')
            endif
            if (increm.le.(0.0)) goto 196
         else
195         continue
            key = getnum(0,200,125,1,4,'(i3)',3,incpts)
            if (incpts.gt.300) then
               call WRTSTR(0,40,150,31,
     +                     '*** Error : No. of points > 300')
               call ERTONE()
            elseif (incpts.lt.1) then
               call WRTSTR(0,40,150,31,
     +                     '*** Error : No. of points < 1  ')
               call ERTONE()
            else
               call WRTSTR(0,40,150,31,
     +                     '                                ')
            endif
            if ((incpts.gt.300).or.(incpts.lt.1)) goto 195
         endif
       endif
       if ((key.eq.retk).or.(key.eq.tabk).or.(key.eq.downk)) then
         pos = pos + 1
         if (pos.gt.5) pos = 1
       elseif ((key.eq.rtabk).or.(key.eq.upk)) then
         pos = pos - 1
         if (pos.lt.1) pos = 5
       endif
       if (key.ne.esck) goto 199
       if (fstop.le.fstart) then
```

```
         call WRTSTR(0,40,150,44,
     +   '*** Error : End frequency ≤ Start frequency.')
         call ERTONE()
         pos = 2
       else
         call WRTSTR(0,40,150,44,
     +   '                                                ')
       endif
       if (fstop.le.fstart) goto 199
       ferr = calfre()
       if (ferr.eq.-1) then
         call WRTSTR(0,40,200,57,
     +   '                                                ')
         call WRTSTR(0,40,215,57,
     +   '                                                ')
         call WRTSTR(0,40,200,36,
     +              '*** Warning : Number of points > 300')
         call WRTSTR(0,40,215,36,'Hit ESC to continue. RETURN to edit.')
         call ERTONE()
180      continue
         key = INKEY(1)
         if (key.eq.retk) goto 199
         if (key.ne.esck) goto 180
       endif

       call WIPSCR(0)
       return
       end

C ***************************************************************************
CN     MODULE NAME      : selaxe
CA     FUNCTION         : To edit the plot page parameters.
CS     CALL SEQUENCE    : call selaxe()
CI     INPUT PARAMETERS : None.
C
CO     OUTPUT PARAMETERS: None.
C
CG     GLOBAL VARIABLES : plotpg.inc
C                         keys.inc
CM     MODULES CALLED   : ERTONE (asm),  INKEY (asm), prset (for),  prsys (for),
C                         WIPSCR (asm), wrtitl (for), WRTSTR (asm).
C
CE     ERROR CONDITIONS : ?
C
CC     COMMENTS         : This routine permits the user to edit the plot page
C                         parameters :-
C                           i) The page on which the system loci are plotted.
C                          ii) The page on which the bode and misalignment
C                              angles are plotted.
C                         iii) The loop characteristic : closed or open loop.
C                          iv) The system to be plotted : G(s) or Q(s)=K(s)G(s).
C
C                         The user is allowed to setup two sets of parameters,
C                         SETTING 1 and SETTING 2. The choice of setting to be
C                         used will be prompted at the time of the plot.
C
C ***************************************************************************
       subroutine selaxe()
       implicit integer*2 (I)
$include: 'plotpg.inc'
$include: 'keys.inc'

       integer*2 key,pos

       call WIPSCR(0)
       call wrtitl(0,250,20,21,'Plot Page Definition.')
       call wrtitl(0,40,45,11,'Setting 1 :')
       call WRTSTR(0,150,45,22,'System to Plot      : ')
       call WRTSTR(0,150,62,22,'Characteristic Loci : ')
       call WRTSTR(0,150,79,22,'Bode Diagrams and     ')
       call WRTSTR(0,150,96,22,'Misalignment angles : ')
```

```
      call wrtitl(0,40,153,11,'Setting 2 :')
      call WRTSTR(0,150,153,22,'System to Plot      : ')
      call WRTSTR(0,150,170,22,'Characteristic Loci : ')
      call WRTSTR(0,150,187,22,'Bode Diagrams and     ')
      call WRTSTR(0,150,204,22,'Misalignment angles : ')
      call WRTSTR(0,40,265,45,
     +'RETURN and TAB keys to move. Hit ESC to exit.')
      pos = 1
      call prset(355,62,plset1,1)
      call prset(355,96,pmset1,1)
      call prsys(355,153,pmat2,1)
      call prset(355,170,plset2,1)
      call prset(355,204,pmset2,1)
299   continue
      if (pos.eq.1) then
         call prsys(355,45,pmat1,2)
280      continue
         key = INKEY(1)
         if (key.eq.8192) then
            pmat1 = pmat1 + 1
            if (pmat1.gt.2) pmat1 = 1
            call prsys(355,45,pmat1,3)
         endif
         if ((key.ne.retk).and.(key.ne.tabk).and.(key.ne.rtabk).and.
     +       (key.ne.esck)) goto 280
         call prsys(355,45,pmat1,1)
      elseif (pos.eq.2) then
         call prset(355,62,plset1,2)
298      continue
         key = INKEY(1)
         if (key.eq.8192) then
            plset1 = plset1 + 1
            if (plset1.gt.2) plset1 = 0
            call prset(355,62,plset1,3)
         endif
         if ((key.ne.retk).and.(key.ne.tabk).and.(key.ne.rtabk).and.
     +       (key.ne.esck)) goto 298
         call prset(355,62,plset1,1)
      elseif (pos.eq.3) then
         call prset(355,96,pmset1,2)
297      continue
         key = INKEY(1)
         if (key.eq.8192) then
            pmset1 = pmset1 + 1
            if (pmset1.gt.2) pmset1 = 0
            call prset(355,96,pmset1,3)
         endif
         if ((key.ne.retk).and.(key.ne.tabk).and.(key.ne.rtabk).and.
     +       (key.ne.esck)) goto 297
         if ((pmset1.ne.2).and.(plset1.eq.pmset1)) then
            call WRTSTR(0,40,115,61,
     +'*** Error : Setting_1 : Loci page = Bode + Misalignment page.')
            call ERTONE()
         else
            call WRTSTR(0,40,115,61,
     +'                                                            ')
         endif
         if ((pmset1.ne.2).and.(plset1.eq.pmset1)) goto 297
         call prset(355,96,pmset1,1)
      elseif (pos.eq.4) then
         call prsys(355,153,pmat2,2)
278      continue
         key = INKEY(1)
         if (key.eq.8192) then
            pmat2 = pmat2 + 1
            if (pmat2.gt.2) pmat2 = 1
            call prsys(355,153,pmat2,3)
         endif
         if ((key.ne.retk).and.(key.ne.tabk).and.(key.ne.rtabk).and.
     +       (key.ne.esck)) goto 278
         call prsys(355,153,pmat2,1)
      elseif (pos.eq.5) then
```

```
         call prset(355,170,plset2,2)
296      continue
         key = INKEY(1)
         if (key.eq.8192) then
            plset2 = plset2 + 1
            if (plset2.gt.2) plset2 = 0
            call prset(355,170,plset2,3)
         endif
         if ((key.ne.retk).and.(key.ne.tabk).and.(key.ne.rtabk).and.
     +       (key.ne.esck)) goto 296
         call prset(355,170,plset2,1)
       elseif (pos.eq.6) then
         call prset(355,204,pmset2,2)
295      continue
         key = INKEY(1)
         if (key.eq.8192) then
            pmset2 = pmset2 + 1
            if (pmset2.gt.2) pmset2 = 0
            call prset(355,204,pmset2,3)
         endif
         if ((key.ne.retk).and.(key.ne.tabk).and.(key.ne.rtabk).and.
     +       (key.ne.esck)) goto 295
         if ((pmset2.ne.2).and.(plset2.eq.pmset2)) then
            call WRTSTR(0,40,225,61,
     +'*** Error : Setting_2 : Loci page = Bode + Misalignment page.')
            call ERTONE()
         else
            call WRTSTR(0,40,225,61,
     +'                                                              ')
         endif
         if ((pmset2.ne.2).and.(plset2.eq.pmset2)) goto 295
         call prset(355,204,pmset2,1)
       endif
       if ((key.eq.retk).or.(key.eq.tabk)) then
         pos = pos + 1
         if (pos.gt.6) pos = 1
       elseif (key.eq.rtabk) then
         pos = pos - 1
         if (pos.lt.1) pos = 6
       endif
       if (key.ne.esck) goto 299
       if ((pmset1.ne.2).and.(plset1.eq.pmset1)) then
         call WRTSTR(0,40,115,61,
     +'*** Error : Setting_1 : Loci page = Bode + Misalignment page.')
         call ERTONE()
         pos = 2
       else
         call WRTSTR(0,40,115,61,
     +'                                                              ')
       endif
       if ((pmset1.ne.2).and.(plset1.eq.pmset1)) goto 299
       if ((pmset2.ne.2).and.(plset2.eq.pmset2)) then
         call WRTSTR(0,40,225,61,
     +'*** Error : Setting_2 : Loci page = Bode + Misalignment page.')
         call ERTONE()
         pos = 5
       else
         call WRTSTR(0,40,225,61,
     +'                                                              ')
       endif
       if ((pmset2.ne.2).and.(plset2.eq.pmset2)) goto 299

       call WIPSCR(0)
       return
       end

C ******************************************************************************
CH     REVISION HISTORY :
C      VERSION       BY                DATE        COMMENT
C      1.00          Ian Fisher        23/06/88    Creation.
C      FILEND           :
```

```
C
C     FILE              : CALFRE
C ***************************************************************************
CN    MODULE NAME       : calfre
CA    FUNCTION          : To calculate the individual frequencies for plotting.
CS    CALL SEQUENCE     : pts = calfre()
CI    INPUT PARAMETERS  : Parameters passed globally through PLOTPG.INC
CO    OUTPUT PARAMETERS: pts - (INTEGER*2) The number of points calculated.
C                                          pts = -1 if the range included more
C                                                than 300 points.
CG    GLOBAL VARIABLES  : plotpg.inc
C                         plotfr.inc
CM    MODULES CALLED    : ptsrng (for), ptsdec (for), ptsoct (for), freinc (for)
CE    ERROR CONDITIONS  : ?
C
CC    COMMENTS          : The routine checks the increment type and then calls
C                         the appropriate routine to calculate the points. The
C                         routine fills the remainder of the array of
C                         frequencies with -1, if the range does not exceed
C                         300 points.
C
C ***************************************************************************
      integer*2 function calfre()
      implicit integer*2 (f,p)
$include: 'plotpg.inc'
$include: 'plotfr.inc'
      real*4 fst2,fstp2,inc2
      integer*2 i

      if (ftype.eq.1) then
         fst2 = fstart/(2.0*3.14159265)
         fstp2 = fstop/(2.0*3.14159265)
         inc2 = increm/(2.0*3.14159265)
      elseif (ftype.eq.2) then
         fst2 = fstart
         fstp2 = fstop
         inc2 = increm
      else
         fst2 = fstart/(2.0*3.14159265*3600.0)
         fstp2 = fstop/(2.0*3.14159265*3600.0)
         inc2 = increm/(2.0*3.14159265*3600.0)
      endif
      if (inctyp.eq.1) then
         calfre = ptsrng(fst2,fstp2,incpts)
      elseif (inctyp.eq.2) then
         calfre = ptsdec(fst2,fstp2,incpts)
      elseif (inctyp.eq.3) then
         calfre = ptsoct(fst2,fstp2,incpts)
      else
         calfre = freinc(fst2,fstp2,inc2)
      endif
      do 99 i = 1,300
         if (freqs(i).ne.-1) then
            if (ftype.eq.1) then
               freqs(i) = freqs(i)*(2.0*3.14159265)
            elseif (ftype.eq.3) then
               freqs(i) = freqs(i)*(2.0*3.14159265*3600.0)
            endif
         endif
99    continue
      return
      end

C ***************************************************************************
CN    MODULE NAME       : PTSRNG
CA    FUNCTION          : To calculate the frequency points with equal increment
C                         over the frequency range.
CS    CALL SEQUENCE     : err = ptsrng(fstart,fstop,pts)
CI    INPUT PARAMETERS :
C               fstart - (real*4) The start frequency.
C                fstop - (real*4) The end frequency.
C                  pts - (integer*2) The number of points over the range.
```

```
C
CO    OUTPUT PARAMETERS:
C                 err - (integer*2) The error return :  0 - no error.
C                                                      -1 - error, too many
C                                                           points.
CG    GLOBAL VARIABLES : plotfr.inc
CM    MODULES CALLED   : None.
CE    ERROR CONDITIONS : ?
C
CC    COMMENTS         : Calculates the constant frequency step and then steps
C                        through the range calculating the points.
C
C ***************************************************************************
      integer*2 function ptsrng(fstart,fstop,pts)
      real*4 fstart,fstop
      integer*2 pts
$include: 'PLOTFR.INC'
       real*4 incr
       integer*2 i,pts2

      if (pts.le.1) then
         freqs(1) = fstart
         freqs(2) = fstop
         do 199 i = 3,300
            freqs(i) = -1.0
199      continue
         ptsrng = 0
      elseif (pts.gt.1) then
         if (pts.gt.300) then
            ptsrng = -1
            pts2 = 300
         else
            ptsrng = 0
            pts2 = pts
         endif
         incr = (fstop - fstart)/(real (pts2-1))
         do 198 i = 1,300
            if (i.le.pts2) then
               freqs(i) = fstart + ((i-1)*incr)
            else
               freqs(i) = -1.0
            endif
198      continue
      endif
      return
      end

C ***************************************************************************
CN    MODULE NAME      : PTSDEC
CA    FUNCTION         : To calculate the frequency points with LOG increments
C                        over the frequency range.
CS    CALL SEQUENCE    : err = ptsrng(fstart,fstop,pts)
CI    INPUT PARAMETERS :
C              fstart - (real*4) The start frequency.
C               fstop - (real*4) The end frequency.
C                 pts - (integer*2) The number of points per decade.
C
CO    OUTPUT PARAMETERS:
C                 err - (integer*2) The error return :  0 - no error.
C                                                      -1 - error, too many
C                                                           points.
CG    GLOBAL VARIABLES : plotfr.inc
CM    MODULES CALLED   : None.
CE    ERROR CONDITIONS : ?
C
CC    COMMENTS         : Calculates the frequency step/decade and then steps
C                        through the range calculating the points.
C
C ***************************************************************************
      integer*2 function ptsdec(fstart,fstop,pts)
      real*4 fstart,fstop
      integer*2 pts
```

```
$include: 'plotfr.inc'
      real*8 fact,fst2
      integer*2 i,count,strti

      fst2 = fstart
      fact = 1/(real (pts))
      count = 1
      ptsdec = 0
      strti = 1
      freqs(count) = fst2
      count = count + 1
      if (pts.lt.1) then
         freqs(count) = fstop
         do 299 i = 3,300
            freqs(i) = -1.0
299      continue
      else
298      continue
         do 297 i = strti,pts
            if (count.lt.300) then
               freqs(count) = fst2*(10.0**(fact*i))
               if (freqs(count).le.fstop) then
                  count = count + 1
               else
                  freqs(count) = -1.0
               endif
            else
               ptsdec = -1
            endif
297      continue
         if (pts.gt.1) then
            strti = 2
         endif
         fst2 = fst2*10.0
         if ((fst2.le.fstop).and.(count.le.300).and.(ptsdec.ne.-1))
     +      goto 298
         if ((count.lt.300).and.(ptsdec.ne.-1)) then
            do 296 i = count,300
               freqs(i) = -1.0
296         continue
         endif
      endif
      return
      end

C ******************************************************************************
CN    MODULE NAME       : PTSOCT
CA    FUNCTION          : To calculate the frequency points with octave
C                         increments over the frequency range.
CS    CALL SEQUENCE     : err = ptsrng(fstart,fstop,pts)
CI    INPUT PARAMETERS :
C              fstart - (real*4) The start frequency.
C               fstop - (real*4) The end frequency.
C                 pts - (integer*2) The number of points per octave.
C
CO    OUTPUT PARAMETERS:
C                 err - (integer*2) The error return :  0 - no error.
C                                                      -1 - error, too many
C                                                           points.
CG    GLOBAL VARIABLES : plotfr.inc
CM    MODULES CALLED   : None.
CE    ERROR CONDITIONS : ?
C
CC    COMMENTS         : Calculates the frequency step/octave and then steps
C                        through the range calculating the points.
C
C ******************************************************************************
      integer*2 function ptsoct(fstart,fstop,pts)
      real*4 fstart,fstop
      integer*2 pts
$include: 'plotfr.inc'
      real*8 fact,fst2
```

```
      integer*2 i,count,strti

      fst2 = fstart
      fact = 1/(real (pts))
      count = 1
      ptsoct = 0
      strti = 1
      freqs(count) = fst2
      count = count + 1
      if (pts.lt.1) then
         freqs(count) = fstop
         do 399 i = 3,300
            freqs(i) = -1.0
399      continue
      else
398      continue
         do 397 i = strti,pts
            if (count.lt.300) then
               freqs(count) = fst2*(2.0**(fact*i))
               if (freqs(count).le.fstop) then
                  count = count + 1
               else
                  freqs(count) = -1.0
               endif
            else
               ptsoct = -1
            endif
397      continue
         fst2 = fst2*2.0
         if (pts.gt.1) then
            strti = 2
         endif
         if ((fst2.le.fstop).and.(count.le.300).and.(ptsoct.ne.-1))
     +      goto 398
         if ((count.lt.300).and.(ptsoct.ne.-1)) then
            do 396 i = count,300
               freqs(i) = -1.0
396         continue
         endif
      endif
      return
      end

C ******************************************************************************
CN    MODULE NAME       : FREINC
CA    FUNCTION          : To calculate the frequency points with constant
C                         increment.
CS    CALL SEQUENCE     : err = freinc(fstart,fstop,inc)
CI    INPUT PARAMETERS :
C              fstart - (real*4) The start frequency.
C               fstop - (real*4) The end frequency.
C                 inc - (real*4) The frequency increment.
C
CO    OUTPUT PARAMETERS:
C                 err - (integer*2) The error return :  0 - no error.
C                                                      -1 - error, too many
C                                                           points.
CG    GLOBAL VARIABLES : plotfr.inc
CM    MODULES CALLED   : None.
CE    ERROR CONDITIONS : ?
C
CC    COMMENTS         : Steps through the range calculating the points.
C
C ******************************************************************************
      integer*2 function freinc(fstart,fstop,inc)
      real*4 fstart,fstop,inc
$include: 'plotfr.inc'
      integer*2 i

      if (inc.le.(0.0)) then
         freqs(1) = fstart
         freqs(2) = fstop
```

```
         do 499 i = 3,300
            freqs(i) = -1.0
499      continue
         freinc = 0
      else
         do 498 i =1,300
            freqs(i) = fstart + (i-1)*inc
            if (freqs(i).gt.fstop) then
               freqs(i) = -1.0
            endif
498      continue
         if ((freqs(300).ne.(-1.0)).and.(freqs(300).lt.fstop)) then
            freinc = -1
         else
            freinc = 0
         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     FILE              : FODDS.FOR
C ***************************************************************************
CN    MODULE NAME       : prftyp
CA    FUNCTION          : Prints the frequency unit parameters.
CS    CALL SEQUENCE     : call prftyp(type,symbol)
CI    INPUT PARAMETERS :   type - (integer*2) Frequency parameter :-
C                                             1 - Hertz
C                                             2 - Rad/sec
C                                             3 - Rad/hour
C                          symbol - (integer*2) Type of symbol to be displayed :-
C                                             1 - Insert brackets on option.
C                                             2 - Return option to normal.
C                                             3 - Highlight option.
CO    OUTPUT PARAMETERS: None.
CG    GLOBAL VARIABLES : None.
CM    MODULES CALLED   : WRTSTR (asm), COPYST (asm), LEVEL (asm), selfre (for).
C
CE    ERROR CONDITIONS : Clueless.
C
CC    COMMENTS         : This prints the frequency unit parameter while
C                        editing the user is editing the parameters.
C
C ***************************************************************************
      subroutine prftyp(type,sym)
      integer*2 type,sym
      integer*2 type2
      character*25 fstr
      if (sym.eq.1) then
         fstr = 'HzRad/SecRad/Hour'
         if (type.eq.1) then
            call COPYST(25,fstr,0,1,'[',0,1)
            call COPYST(25,fstr,3,1,']',0,1)
         else
            call COPYST(25,fstr,0,1,' ',0,1)
            call COPYST(25,fstr,3,1,' ',0,1)
         endif
         if (type.eq.2) then
            call COPYST(25,fstr,4,1,'[',0,1)
            call COPYST(25,fstr,12,1,']',0,1)
         else
            call COPYST(25,fstr,4,1,' ',0,1)
            call COPYST(25,fstr,12,1,' ',0,1)
         endif
         if (type.eq.3) then
            call COPYST(25,fstr,13,1,'[',0,1)
            call COPYST(25,fstr,22,1,']',0,1)
         else
            call COPYST(25,fstr,13,1,' ',0,1)
            call COPYST(25,fstr,22,1,' ',0,1)
         endif
         call LEVEL(0)
         call selfre(type)
         call LEVEL(1)
         call WRTSTR(0,200,60,25,fstr)
      else
         fstr = ' Hz  Rad/Sec  Rad/Hour'
         if (sym.eq.3) then
            type2 = type - 1
            if (type2.lt.1) type2 = 3
            call LEVEL(2)
            call selfre(type2)
         endif
         if (sym.eq.2) then
            call LEVEL(1)
            call WRTSTR(0,200,60,25,fstr)
         endif
         call LEVEL(2)
         call selfre(type)
         call LEVEL(1)
      endif
      return
```

```
      end

C ***************************************************************************
CN    MODULE NAME       : selfre
CA    FUNCTION          : To highlight frequency unit option.
CS    CALL SEQUENCE     : call selfre(type)
CI    INPUT PARAMETERS  : type - (integer*2) Which option to highlight :-
C                                            1 - Hertz.
C                                            2 - Rad/sec.
C                                            3 - Rad/hour.
CO    OUTPUT PARAMETERS: None.
CG    GLOBAL VARIABLES  : None.
CM    MODULES CALLED    : BLKFIL (asm).
CE    ERROR CONDITIONS  : ?
C
CC    COMMENTS          : Routine draws a block around the option requested by
C                         calling routine.
C
C ***************************************************************************
      subroutine selfre(type)
      integer*2 type
      if (type.eq.1) then
         call BLKFIL(209,62,18,14)
      elseif (type.eq.2) then
         call BLKFIL(245,62,63,14)
      elseif (type.eq.3) then
         call BLKFIL(326,62,72,14)
      endif
      return
      end

C ***************************************************************************
CN    MODULE NAME       : prityp
CA    FUNCTION          : Prints the frequency increment type parameter.
CS    CALL SEQUENCE     : call prityp(type,symbol)
CI    INPUT PARAMETERS  :   type - (integer*2) Frequency parameter :-
C                                              1 - Points/range.
C                                              2 - Points/decade.
C                                              3 - Points/octave.
C                                              4 - Linear frequency increments.
C                         symbol - (integer*2) Type of symbol to be displayed :-
C                                              1 - Insert brackets on option.
C                                              2 - Highlight current option.
C                                              3 - Return previous option to
C                                                  normal.
CO    OUTPUT PARAMETERS: None.
CG    GLOBAL VARIABLES  : None.
CM    MODULES CALLED    : WRTSTR (asm), COPYST (asm), LEVEL (asm), selinc (for).
C
CE    ERROR CONDITIONS  : Clueless.
C
CC    COMMENTS          : This prints the frequency increment parameter while
C                         editing the user is editing the parameters.
C
C ***************************************************************************
      subroutine prityp(type,sym)
      integer*2 type,sym
      integer*2 type2
      character*40 fstr
      if (sym.eq.1) then
         fstr = 'Pts/RangePts/DecPts/OctLinear'
         if (type.eq.1) then
            call COPYST(38,fstr,0,1,'[',0,1)
            call COPYST(38,fstr,10,1,']',0,1)
         else
            call COPYST(38,fstr,0,1,' ',0,1)
            call COPYST(38,fstr,10,1,' ',0,1)
         endif
         if (type.eq.2) then
            call COPYST(38,fstr,11,1,'[',0,1)
            call COPYST(38,fstr,19,1,']',0,1)
         else
```

```
            call COPYST(38,fstr,11,1,' ',0,1)
            call COPYST(38,fstr,19,1,' ',0,1)
         endif
         if (type.eq.3) then
            call COPYST(38,fstr,20,1,'[',0,1)
            call COPYST(38,fstr,28,1,']',0,1)
         else
            call COPYST(38,fstr,20,1,' ',0,1)
            call COPYST(38,fstr,28,1,' ',0,1)
         endif
         if (type.eq.4) then
            call COPYST(38,fstr,29,1,'[',0,1)
            call COPYST(38,fstr,36,1,']',0,1)
         else
            call COPYST(38,fstr,29,1,' ',0,1)
            call COPYST(38,fstr,36,1,' ',0,1)
         endif
         call LEVEL(0)
         call selinc(type)
         call LEVEL(1)
         call WRTSTR(0,200,105,38,fstr)
      else
         fstr = ' Pts/Range  Pts/Dec  Pts/Oct  Linear'
         if (sym.eq.3) then
            type2 = type - 1
            if (type2.lt.1) type2 = 4
            call LEVEL(2)
            call selinc(type2)
         endif
         if (sym.eq.2) then
            call LEVEL(1)
            call WRTSTR(0,200,105,40,fstr)
         endif
         call LEVEL(2)
         call selinc(type)
         call LEVEL(1)
      endif
      return
      end

C ***************************************************************************
CN    MODULE NAME      : selinc
CA    FUNCTION         : To highlight frequency increment option.
CS    CALL SEQUENCE    : call selinc(type)
CI    INPUT PARAMETERS : type - (integer*2) Which option to highlight :-
C                                            1 - points/range.
C                                            2 - points/decade.
C                                            3 - points/octave.
C                                            4 - Linear frequency increments.
CO    OUTPUT PARAMETERS: None.
CG    GLOBAL VARIABLES : None.
CM    MODULES CALLED   : BLKFIL (asm).
CE    ERROR CONDITIONS : ?
C
CC    COMMENTS         : Routine draws a block around the option requested by
C                        calling routine.
C
C ***************************************************************************
      subroutine selinc(type)
      integer*2 type
      if (type.eq.1) then
         call BLKFIL(209,107,81,14)
      elseif (type.eq.2) then
         call BLKFIL(308,107,63,14)
      elseif (type.eq.3) then
         call BLKFIL(389,107,63,14)
      elseif (type.eq.4) then
         call BLKFIL(470,107,54,14)
      endif
      return
      end
```

```
C ***************************************************************************
CN    MODULE NAME       : prset
CA    FUNCTION          : To print plot page options.
CS    CALL SEQUENCE     : call prset(x,y,type,symbol)
CI    INPUT PARAMETERS :     x,y - (integer*2) X,Y co-ords of options.
C                          type - (integer*2) Option to highlight :-
C                                             0 - Page 0.
C                                             1 - Page 1.
C                                             2 - No display.
C                        symbol - (integer*2)
C                                             1 - Bracket option.
C                                             2 - Highlight current option.
C                                             3 - Remove highlight from
C                                                 previous option.
CO    OUTPUT PARAMETERS: None.
CG    GLOBAL VARIABLES : None.
CM    MODULES CALLED   : WRTSTR (asm), COPYST (asm), LEVEL (asm), selset (for).
CE    ERROR CONDITIONS : ?
C
CC    COMMENTS          : This routine prints the plot page options while the
C                         user is editing these parameters.
C
C ***************************************************************************
      subroutine prset(x,y,type,sym)
      integer*2 x,y,type,sym
      integer*2 type2
      character*30 fstr
      if (sym.eq.1) then
         fstr = 'Page_0Page_1No display'
         if (type.eq.0) then
            call COPYST(30,fstr,0,1,'[',0,1)
            call COPYST(30,fstr,7,1,']',0,1)
         else
            call COPYST(30,fstr,0,1,' ',0,1)
            call COPYST(30,fstr,7,1,' ',0,1)
         endif
         if (type.eq.1) then
            call COPYST(30,fstr,8,1,'[',0,1)
            call COPYST(30,fstr,15,1,']',0,1)
         else
            call COPYST(30,fstr,8,1,' ',0,1)
            call COPYST(30,fstr,15,1,' ',0,1)
         endif
         if (type.eq.2) then
            call COPYST(30,fstr,16,1,'[',0,1)
            call COPYST(30,fstr,27,1,']',0,1)
         else
            call COPYST(30,fstr,16,1,' ',0,1)
            call COPYST(30,fstr,27,1,' ',0,1)
         endif
         call LEVEL(0)
         call selset(x,y,type)
         call LEVEL(1)
         call WRTSTR(0,x,y,30,fstr)
      else
         fstr = ' Page_0  Page_1  No display'
         if (sym.eq.3) then
            type2 = type - 1
            if (type2.lt.0) type2 = 2
            call LEVEL(2)
            call selset(x,y,type2)
         endif
         if (sym.eq.2) then
            call LEVEL(1)
            call WRTSTR(0,x,y,27,fstr)
         endif
         call LEVEL(2)
         call selset(x,y,type)
         call LEVEL(1)
      endif
      return
      end
```

```
C ***************************************************************************
CN    MODULE NAME      : selset
CA    FUNCTION         : To highlight plot page options.
CS    CALL SEQUENCE    : call selset(x,y,type)
CI    INPUT PARAMETERS : x,y - (integer*2) X,Y co-ords of options.
C                        type - (integer*2) Option to highlight :-
C                                           0 - Page 0.
C                                           1 - Page 1.
C                                           2 - No display.
CO    OUTPUT PARAMETERS: None.
CG    GLOBAL VARIABLES : None.
CM    MODULES CALLED   : BLKFIL (asm).
CE    ERROR CONDITIONS : ?
C
CC    COMMENTS         : Routine draws a block around the option requested by
C                        calling routine.
C
C ***************************************************************************
      subroutine selset(x,y,type)
      integer*2 x,y,type
      if (type.eq.0) then
         call BLKFIL(x+9,y+2,54,14)
      elseif (type.eq.1) then
         call BLKFIL((x+81),y+2,54,14)
      elseif (type.eq.2) then
         call BLKFIL((x+153),y+2,90,14)
      endif
      return
      end

C ***************************************************************************
CN    MODULE NAME      : prsys
CA    FUNCTION         : To print system to be plotted options.
CS    CALL SEQUENCE    : call prset(x,y,type,symbol)
CI    INPUT PARAMETERS :    x,y - (integer*2) X,Y co-ords of options.
C                         type - (integer*2) Option to highlight :-
C                                            1 - G(s)
C                                            2 - Q(s) = K(s)G(s)
C                        symbol - (integer*2)
C                                            1 - Bracket option.
C                                            2 - Highlight current option.
C                                            3 - Remove highlight from
C                                                previous option.
CO    OUTPUT PARAMETERS: None.
CG    GLOBAL VARIABLES : None.
CM    MODULES CALLED   : WRTSTR (asm), COPYST (asm), LEVEL (asm), syset (for).
CE    ERROR CONDITIONS : ?
C
CC    COMMENTS         : This routine prints the system to be plotted options
C                        while the user is editing these parameters.
C
C ***************************************************************************
      subroutine prsys(x,y,type,sym)
      integer*2 x,y,type,sym
      integer*2 type2
      character*22 fstr
      if (sym.eq.1) then
         fstr = 'G(s)Q(s)=K(s)G(s)'
         if (type.eq.1) then
            call COPYST(22,fstr,0,1,'[',0,1)
            call COPYST(22,fstr,5,1,']',0,1)
         else
            call COPYST(22,fstr,0,1,' ',0,1)
            call COPYST(22,fstr,5,1,' ',0,1)
         endif
         if (type.eq.2) then
            call COPYST(22,fstr,6,1,'[',0,1)
            call COPYST(22,fstr,20,1,']',0,1)
         else
            call COPYST(22,fstr,6,1,' ',0,1)
            call COPYST(22,fstr,20,1,' ',0,1)
```

```
         endif
         call LEVEL(0)
         call syset(x,y,type)
         call LEVEL(1)
         call WRTSTR(0,x,y,22,fstr)
      else
         fstr = ' G(s)  Q(s)=K(s)G(s)'
         if (sym.eq.3) then
            type2 = type - 1
            if (type2.lt.1) type2 = 2
            call LEVEL(2)
            call syset(x,y,type2)
         endif
         if (sym.eq.2) then
            call LEVEL(1)
            call WRTSTR(0,x,y,22,fstr)
         endif
         call LEVEL(2)
         call syset(x,y,type)
         call LEVEL(1)
      endif
      return
      end

C ***************************************************************************
CN    MODULE NAME       : syset
CA    FUNCTION          : To highlight an option for system to plotted options.
CS    CALL SEQUENCE     : call syset(x,y,type)
CI    INPUT PARAMETERS :    x,y - (integer*2) X,Y co-ords of options.
C                         type - (integer*2) Option to highlight :-
C                                            1 - G(s)
C                                            2 - Q(s) = K(s)G(s)
CO    OUTPUT PARAMETERS: None.
CG    GLOBAL VARIABLES : None.
CM    MODULES CALLED   : BLKFIL (asm).
CE    ERROR CONDITIONS : ?
C
CC    COMMENTS          : Routine draws a block around the option requested by
C                         calling routine.
C
C ***************************************************************************
      subroutine syset(x,y,type)
      integer*2 x,y,type
      if (type.eq.1) then
         call BLKFIL(x+9,y+2,36,14)
      elseif (type.eq.2) then
         call BLKFIL((x+63),y+2,117,14)
      endif
      return
      end

C ***************************************************************************
CN    MODULE NAME       : lopset
CA    FUNCTION          : To highlight feedback characteristic options.
CS    CALL SEQUENCE     : call lopset(x,y,type)
CI    INPUT PARAMETERS :    x,y - (integer*2) X,Y co-ords of options.
C                         type - (integer*2) Option to highlight :-
C                                            1 - Open loop.
C                                            2 - Closed Loop.
CO    OUTPUT PARAMETERS: None.
CG    GLOBAL VARIABLES : None.
CM    MODULES CALLED   : BLKFIL (asm).
CE    ERROR CONDITIONS : ?
C
CC    COMMENTS          : Routine draws a block around the option requested by
C                         calling routine.
C
C ***************************************************************************
      subroutine lopset(x,y,type)
      integer*2 x,y,type
      if (type.eq.1) then
         call BLKFIL(x+9,y+2,81,14)
```

```
      elseif (type.eq.2) then
         call BLKFIL((x+108),y+2,99,14)
      endif
      return
      end

C ********************************************************************************
CN    MODULE NAME       : prange
CA    FUNCTION          : Prints the frequency sweep parameters.
CS    CALL SEQUENCE     : call prange()
CI    INPUT PARAMETERS  : None.
CO    OUTPUT PARAMETERS : None.
CG    GLOBAL VARIABLES  : PLOTPG.INC
CM    MODULES CALLED    : prtnum (for), WRTSTR (asm).
CE    ERROR CONDITIONS  : ?
C
CC    COMMENTS          : Prints the frequency sweep parameters while the user
C                         is editing these parameters.
C                         The frequency parameters are stored in rads/sec, thus
C                         the routine converts the parameters for display,
C                         according to frequency type selected.
C
C ********************************************************************************
      subroutine prange()
$include: 'plotpg.inc'
      character*8 fstr
      if (ftype.eq.1) then
         fstar2 = fstart/(2.0*3.14159265)
         fstop2 = fstop/(2.0*3.14159265)
         incre2 = increm/(2.0*3.14159265)
         fstr = 'Hz      '
      elseif (ftype.eq.2) then
         fstar2 = fstart
         fstop2 = fstop
         incre2 = increm
         fstr = 'Rad/sec '
      elseif (ftype.eq.3) then
         fstar2 = fstart/3600.0
         fstop2 = fstop/3600.0
         incre2 = increm/3600.0
         fstr = 'Rad/Hour'
      endif
      call prtnum(0,200,80,3,7,'(g10.4)',10,fstar2)
      call prtnum(0,335,80,3,7,'(g10.4)',10,fstop2)
      call WRTSTR(0,435,80,8,fstr)
      if (inctyp.lt.4) then
         call WRTSTR(0,40,125,31,'No. of points   :              ')
         call prtnum(0,200,125,1,4,'(i3)',3,incpts)
         call WRTSTR(0,300,125,8,'         ')
      else
         call WRTSTR(0,40,125,17,'Increment       :')
         call prtnum(0,200,125,3,7,'(g10.4)',10,incre2)
         call WRTSTR(0,300,125,8,fstr)
      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     FILE              : SCALES.FOR
C ***************************************************************************
CN    MODULE NAME       : scales
CA    FUNCTION          : To enable the user to edit the axes scales.
CS    CALL SEQUENCE     : call scales()
CI    INPUT PARAMETERS  : None.
CO    OUTPUT PARAMETERS: None.
CG    GLOBAL VARIABLES  : sysmat.inc
C                         ploci.inc
C                         keys.inc
CM    MODULES CALLED    : DISP (asm), DLINE (asm),  edelem (for),  INKEY (asm),
C                         LEVEL (asm), MOVE (asm), prtnum (for), prelem (for) ,
C                         WIPSCR (asm), wrtitl (for), WRTSTR (asm).
CE    ERROR CONDITIONS  : ?
C
CC    COMMENTS          : Prints out all the axes limits for  the characteristic
C                         loci, misalignment angles and the bode plots. The user
C                         can then edit the scales accordingly.
C                         The ESC key returns control to the calling routine.
C
C ***************************************************************************
      subroutine scales()
      implicit integer*2 (g,I)
$include: 'sysmat.inc'
$include: 'ploci.inc'
$include: 'keys.inc'
      integer*2 pos,key,elem,lastel,xpos,ypos

      call WIPSCR(1)
      call DISP(1)
      call LEVEL(1)
      call wrtitl(1,250,17,27,'Plot Axes Scale Definition.')
      call MOVE(5,25)
      call DLINE(714,25)
      call MOVE(185,25)
      call DLINE(185,316)
      call wrtitl(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 prtnum(1,xpos+10,ypos,1,4,'(i2)',2,i)
            call WRTSTR(1,xpos+30,ypos,1,',')
            call prtnum(1,xpos+40,ypos,1,4,'(i2)',2,i)
            call WRTSTR(1,xpos+60,ypos,1,')')
            ypos = ypos + 14
99       continue
         call wrtitl(1,400,40,3,'In ')
         call WRTSTR(1,400,54,3,'Out')
         call WRTSTR(1,436,47,1,'=')
         call wrtitl(1,500,75,3,'Max')
         call wrtitl(1,600,75,3,'Min')
         call wrtitl(1,200,95,21,'Characteristic Loci :')
         call WRTSTR(1,400,95,7,'Real : ')
         call WRTSTR(1,400,110,7,'Imag : ')
         call wrtitl(1,200,135,26,'Bode Diagrams       : [dB]')
         call wrtitl(1,200,160,27,'Misalignment Angles : [Deg]')
         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 prelem(lastel,elem)
98       continue
         key = INKEY(1)
         if (key.eq.downk) then
            lastel = elem
            elem = elem + 1
```

```
            if (elem.gt.order) elem = 1
            call prelem(lastel,elem)
         elseif (key.eq.upk) then
            lastel = elem
            elem = elem - 1
            if (elem.lt.1) elem = order
            call prelem(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 edelem(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      : prelem
CA    FUNCTION         : To print out a set of scales and I/O names.
CS    CALL SEQUENCE    : call prelem(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 : ploci.inc
C                       sysnms.inc
CM    MODULES CALLED   : BLKFIL (asm), LENSTR (asm), LEVEL (asm), prtnum (for),
C                       WRTSTR (asm), wrtitl (for).
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
C ***************************************************************************
      subroutine prelem(lastel,elem)
      implicit integer*2 (L)
      integer*2 lastel,elem
$include: 'ploci.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 wrtitl(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 prtnum(1,470,95,4,7,'(g10.4)',10,xllim(1,elem))
      call prtnum(1,570,95,4,7,'(g10.4)',10,xllim(2,elem))
      call prtnum(1,470,110,4,7,'(g10.4)',10,yllim(1,elem))
```

```
      call prtnum(1,570,110,4,7,'(g10.4)',10,yllim(2,elem))
      call prtnum(1,470,135,4,7,'(g10.4)',10,xblim(1,elem))
      call prtnum(1,570,135,4,7,'(g10.4)',10,xblim(2,elem))
      call prtnum(1,470,160,4,7,'(g10.4)',10,xmlim(1,elem))
      call prtnum(1,570,160,4,7,'(g10.4)',10,xmlim(2,elem))
      return
      end

C ******************************************************************************
CN    MODULE NAME       : edelem
CA    FUNCTION          : Enables the user to edit the axes scales.
CS    CALL SEQUENCE     : call edelem(element)
CI    INPUT PARAMETERS  : element - (integer*2) The element to be edited.
CO    OUTPUT PARAMETERS: None.
CG    GLOBAL VARIABLES  : ploci.inc
C                         keys.inc
CM    MODULES CALLED    : ERTONE (asm), getnum (for), WRTSTR (asm), BOX (asm),
C                         LEVEL (asm), BLKFIL (asm).
CE    ERROR CONDITIONS  : ?
C
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
C ******************************************************************************
      subroutine edelem(elem)
      implicit integer*2 (g)
      integer*2 elem
$include: 'ploci.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 = getnum(1,470,95,4,7,'(g10.4)',10,xllim(1,elem))
         if (xllim(1,elem).lt.(0.0)) then
            call WRTSTR(1,200,200,36,
     +                  '*** Error : Loci maximum limit < 0.0')
            call ERTONE()
         else
            call WRTSTR(1,200,200,36,
     +                  '                                    ')
         endif
         if (xllim(1,elem).lt.(0.0)) goto 198
      elseif (pos.eq.2) then
197      continue
         key = getnum(1,570,95,4,7,'(g10.4)',10,xllim(2,elem))
         if (xllim(2,elem).gt.(0.0)) then
            call WRTSTR(1,200,200,36,
     +                  '*** Error : Loci minimum limit > 0.0')
            call ERTONE()
         else
            call WRTSTR(1,200,200,36,
     +                  '                                    ')
         endif
         if (xllim(1,elem).lt.xllim(2,elem)) then
            call WRTSTR(1,200,220,35,
     +      '*** Error : Loci minimum > maximum.')
            call ERTONE()
         else
            call WRTSTR(1,200,220,35,
     +      '                                   ')
         endif
```

```
      if ((xllim(2,elem).gt.(0.0)).or.
     +     (xllim(1,elem).lt.xllim(2,elem))) goto 197
      elseif (pos.eq.3) then
196      continue
         key = getnum(1,470,110,4,7,'(g10.4)',10,yllim(1,elem))
         if (yllim(1,elem).lt.(0.0)) then
            call WRTSTR(1,200,200,36,
     +                  '*** Error : Loci maximum limit < 0.0')
            call ERTONE()
         else
            call WRTSTR(1,200,200,36,
     +                  '                                      ')
         endif
         if (yllim(1,elem).lt.(0.0)) goto 196
      elseif (pos.eq.4) then
193      continue
         key = getnum(1,570,110,4,7,'(g10.4)',10,yllim(2,elem))
         if (yllim(2,elem).gt.(0.0)) then
            call WRTSTR(1,200,200,36,
     +                  '*** Error : Loci minimum limit > 0.0')
            call ERTONE()
         else
            call WRTSTR(1,200,200,36,
     +                  '                                      ')
         endif
         if (yllim(1,elem).lt.yllim(2,elem)) then
            call WRTSTR(1,200,220,35,
     +      '*** Error : Loci minimum > maximum.')
            call ERTONE()
         else
            call WRTSTR(1,200,220,35,
     +      '                                  ')
         endif
         if ((yllim(2,elem).gt.(0.0)).or.
     +     (yllim(1,elem).lt.yllim(2,elem))) goto 193
      elseif (pos.eq.5) then
195      continue
         key = getnum(1,470,135,4,7,'(g10.4)',10,xblim(1,elem))
         if (xblim(1,elem).lt.(0.0)) then
            call WRTSTR(1,200,200,36,
     +                  '*** Error : Bode maximum limit < 0.0')
            call ERTONE()
         else
            call WRTSTR(1,200,200,36,
     +                  '                                      ')
         endif
         if (xblim(1,elem).lt.(0.0)) goto 195
      elseif (pos.eq.6) then
192      continue
         key = getnum(1,570,135,4,7,'(g10.4)',10,xblim(2,elem))
         if (xblim(2,elem).gt.(0.0)) then
            call WRTSTR(1,200,200,36,
     +                  '*** Error : Bode minimum limit > 0.0')
            call ERTONE()
         else
            call WRTSTR(1,200,200,36,
     +                  '                                      ')
         endif
         if (xblim(1,elem).lt.xblim(2,elem)) then
            call WRTSTR(1,200,220,35,
     +      '*** Error : Bode minimum > maximum.')
            call ERTONE()
         else
            call WRTSTR(1,200,220,35,
     +      '                                  ')
         endif
         if ((xblim(2,elem).gt.(0.0)).or.
     +     (xblim(1,elem).lt.xblim(2,elem))) goto 192
      elseif (pos.eq.7) then
194      continue
         key = getnum(1,470,160,4,7,'(g10.4)',10,xmlim(1,elem))
         if (xmlim(1,elem).lt.(0.0))then
```

```
            call WRTSTR(1,200,200,50,
     +      '*** Error : Misalignment angle maximum limit < 0.0')
            call ERTONE()
         else
            call WRTSTR(1,200,200,50,
     +      '                                                  ')
         endif
         if (xmlim(1,elem).lt.(0.0)) goto 194
       elseif (pos.eq.8) then
191      continue
         key = getnum(1,570,160,4,7,'(g10.4)',10,xmlim(2,elem))
         if (xmlim(2,elem).gt.(0.0)) then
            call WRTSTR(1,200,200,49,
     +      '*** Error : Misalignment angle minimum limit > 0.0')
            call ERTONE()
         else
            call WRTSTR(1,200,200,49,
     +      '                                                  ')
         endif
         if (xmlim(1,elem).lt.xmlim(2,elem)) then
            call WRTSTR(1,200,220,49,
     +      '*** Error : Misalignment angle minimum > maximum.')
            call ERTONE()
         else
            call WRTSTR(1,200,220,49,
     +      '                                                  ')
         endif
         if ((xmlim(2,elem).gt.(0.0)).or.
     +       (xmlim(1,elem).lt.xmlim(2,elem))) goto 191
       endif
       if (key.eq.downk) then
          pos = pos + 2
          if (pos.eq.9) then
             pos = 1
          elseif (pos.gt.8) then
             pos = 2
          endif
       elseif (key.eq.upk) then
          pos = pos - 2
          if (pos.eq.0) then
             pos = 8
          elseif (pos.lt.1) then
             pos = 7
          endif
       elseif ((key.eq.retk).or.(key.eq.tabk)) then
          pos = pos + 1
          if (pos.gt.8) pos = 1
       elseif (key.eq.rtabk) then
          pos = pos - 1
          if (pos.lt.1) pos = 8
       endif
       if (key.ne.esck) goto 199
       xllim(3,elem) = 0.0
       yllim(3,elem) = 0.0
       xblim(3,elem) = 0.0
       yblim(3) = 0.0
       xmlim(3,elem) = 0.0
       ymlim(3) = 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      FILEND           :
```

```
C
C     FILE             : DWLOCI.FOR
C ***************************************************************************
CN    MODULE NAME      : doloci
CA    FUNCTION         : To plot the characteristic loci.
CS    CALL SEQUENCE    : call doloci(page)
CI    INPUT PARAMETERS : page - (integer*2) Page on which the loci are to be
C                                            plotted.
CO    OUTPUT PARAMETERS: None.
CG    GLOBAL VARIABLES : sysmat.inc
C                        sysnms.inc
C                        plotpg.inc
C                        ploci.inc
CM    MODULES CALLED   : DISP (asm),  DLINE (asm),  LENSTR (asm),   MOVE (asm),
C                        dwaxes (for), WIPSCR (asm), WRTSTR (asm), LEVEL (asm),
C                        wrtitl (for).
CE    ERROR CONDITIONS : ?
C
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
C ***************************************************************************
      subroutine doloci(pgdisp)
      implicit integer*2 (L)
      integer*2 pgdisp
$include: 'sysmat.inc'
$include: 'sysnms.inc'
$include: 'plotpg.inc'
$include: 'ploci.inc'
      integer*2 len

      call DISP(pgdisp)
      call WIPSCR(pgdisp)
      if (pgdisp.eq.0) then
         lwmax   = 711
         lwstrt = 4
         lhmax   = 222
         lhstrt = 268
      else
         lwmax   = 711
         lwstrt = 4
         lhmax   = 270
         lhstrt = 316
      endif
      call wrtitl(pgdisp,10,(lhstrt-lhmax-32),21,
     +            'Characteristic Loci :')
      call WRTSTR(pgdisp,205,(lhstrt-lhmax-32),25,prjnm)
      len = LENSTR(25,prjnm)
      call WRTSTR(pgdisp,(205+len*9),(lhstrt-lhmax-32),2,', ')
      call WRTSTR(pgdisp,(205+(len+2)*9),(lhstrt-lhmax-32),25,engnms)
      if (order.le.1) then
         lwidth = 1
      elseif (order.le.4) then
         lwidth = 2
      elseif (order.le.9) then
         lwidth = 3
      elseif (order.le.16) then
         lwidth = 4
      else
         lwidth = 5
      endif
      if ((order.gt.0).and.(order.lt.11)) then
         lheigh = int (order/lwidth)
         if ((real(order)/real(lwidth)).gt.(real(int(order/lwidth))))
     +                                  lheigh = lheigh + 1
         lxdel = int(lwmax/lwidth)
         lydel = int(lhmax/lheigh)
         call LEVEL(1)
```

```
         do 199 i = 1,(lwidth-1)
            call MOVE((i*lxdel+lwstrt),(lhstrt-lheigh*lydel))
            call DLINE((i*lxdel+lwstrt),lhstrt)
199      continue
         do 198 i = 1,lheigh
            call MOVE(lwstrt,(lhstrt-i*lydel))
            call DLINE((lwstrt+lwmax-1),(lhstrt-i*lydel))
198      continue
         j = 1
         k = 1
         do 99 i = 1,order
            lgraph(1) = lwstrt + 9 + (j-1)*lxdel
            lgraph(2) = lhstrt - lhmax - 9 + k*lydel
            lgraph(3) = lxdel - 18
            lgraph(4) = lydel - 15
c            lgraph(1) = lwstrt + 6 + (j-1)*lxdel
c            lgraph(2) = lhstrt - lhmax - 4 + k*lydel
c            lgraph(3) = lxdel - 12
c            lgraph(4) = lydel - 10
            call dwaxes(pgdisp,1,1,xllim(1,i),yllim(1,i),lgraph,
     +                  lscale(1,i),lcentre(1,i))
            j = j + 1
            if (j.gt.lwidth) then
               j = 1
               k = k + 1
            endif
99       continue
      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     FILE              : DWBODE.FOR
C ***************************************************************************
CN    MODULE NAME       : dobode
CA    FUNCTION          : To plot the bode diagrams and misalignment angles.
CS    CALL SEQUENCE     : call dobode(page)
CI    INPUT PARAMETERS  : page - (integer*2) Page on which the loci are to be
C                                            plotted.
CO    OUTPUT PARAMETERS: None.
CG    GLOBAL VARIABLES  : sysmat.inc
C                         sysnms.inc
C                         plotpg.inc
C                         ploci.inc
CM    MODULES CALLED    : DISP (asm),  DLINE (asm),  LENSTR (asm),  MOVE (asm),
C                         dwaxes (for), WIPSCR (asm), WRTSTR (asm), wrtitl (for)
C                         LEVEL (asm).
CE    ERROR CONDITIONS  : ?
C
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
C ***************************************************************************
      subroutine dobode(pgdisp)
      implicit integer*2 (L)
      integer*2 pgdisp
$include: 'sysmat.inc'
$include: 'sysnms.inc'
$include: 'plotpg.inc'
$include: 'ploci.inc'
      integer*2 len,xtype

      call DISP(pgdisp)
      call WIPSCR(pgdisp)
      if (pgdisp.eq.0) then
         mwmax   = 711
         mwstrt = 4
         mhmax   = 222
         mhstrt = 268
      else
         mwmax   = 711
         mwstrt = 4
         mhmax   = 270
         mhstrt = 316
      endif
      call wrtitl(pgdisp,10,(mhstrt-mhmax-32),27,
     +           'Bode + Misalignment Angles:')
      call WRTSTR(pgdisp,255,(mhstrt-mhmax-32),25,prjnm)
      len = LENSTR(25,prjnm)
      call WRTSTR(pgdisp,(255+len*9),(mhstrt-mhmax-32),2,', ')
      call WRTSTR(pgdisp,(255+(len+2)*9-7),(mhstrt-mhmax-32),25,engnms)
      if (order.le.2) then
         mwidth = 1
      elseif (order.le.8) then
         mwidth = 2
      elseif (order.le.15) then
         mwidth = 3
      else
         mwidth = 4
      endif
      if ((order.gt.0).and.(order.lt.11)) then
         mheigh = int (order/mwidth)
         if ((real(order)/real(mwidth)).gt.(real(int(order/mwidth))))
     +                                    mheigh = mheigh + 1
         mxdel = int(mwmax/mwidth)
         mydel = int(mhmax/mheigh)
         call LEVEL(1)
         do 99 i = 1,(mwidth-1)
            call MOVE((i*mxdel+mwstrt),(mhstrt-mheigh*mydel))
```

```
            call DLINE((i*mxdel+mwstrt),mhstrt)
99       continue
         do 98 i = 1,mheigh
            call MOVE(mwstrt,(mhstrt-i*mydel))
            call DLINE((mwstrt+mwmax-1),(mhstrt-i*mydel))
98       continue
         j = 1
         k = 1
         xtype = 1
         yblim(1) = fstop
         yblim(2) = fstart
         yblim(3) = yblim(2)
         if (inctyp.eq.1) then
            xtype = 1
         elseif (inctyp.eq.2) then
            xtype = 2
         elseif (inctyp.eq.3) then
            xtype = 3
         elseif (inctyp.eq.4) then
            xtype = 1
         endif
         do 199 i = 1,order
            bgraph(1) = mwstrt + (j-1)*mxdel + 6
            bgraph(2) = mhstrt - mhmax + k*mydel - 2
            bgraph(3) = mxdel/2 - 12
            bgraph(4) = mydel - 7
            call dwaxes(pgdisp,1,xtype,yblim,xblim(1,i),bgraph,
     +                  bscale(1,i),bcentre(1,i))
            mgraph(1) = mwstrt +(j-1)*mxdel + mxdel/2
            mgraph(2) = mhstrt - mhmax + k*mydel - 3
            mgraph(3) = mxdel/2 - 8
            mgraph(4) = mydel - 8
            call dwaxes(pgdisp,1,xtype,yblim,xmlim(1,i),mgraph,
     +                  mscale(1,i),mcentre(1,i))
            j = j + 1
            if (j.gt.mwidth) then
               j = 1
               k = k + 1
            endif
199      continue
      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     FILE              : DOPLOT.FOR
C ***************************************************************************
CN    MODULE NAME       : doplot
CA    FUNCTION          : To control the plotting of loci, angles and bode plots
CS    CALL SEQUENCE     : call doplot(setting)
CI    INPUT PARAMETERS  : setting - (integer*2) The set of plot parameters to
C                                                use to control the plots.
C                                                (range: 1 or 2)
CO    OUTPUT PARAMETERS: None.
CG    GLOBAL VARIABLES  : plotpg.inc
CM    MODULES CALLED    : dobode (for), doloci (for), dograf (for), viewpl (for)
CE    ERROR CONDITIONS  : ?
C
CC    COMMENTS          : Depending on the plot parameters, the routine
C                         initialises the plot pages. Once all the plots are
C                         complete, a subroutine is called to permit the user
C                         to isolate a single plot.
C
C ***************************************************************************
      subroutine doplot(seting)
      integer*2 seting
$include: 'plotpg.inc'
      if (seting.eq.1) then
         if ((plset1.eq.0).or.(plset1.eq.1)) then
            call doloci(plset1)
         endif
         if ((pmset1.eq.0).or.(pmset1.eq.1)) then
            call dobode(pmset1)
         endif
         if ((plset1.eq.0).or.(plset1.eq.1).or.
     +       (pmset1.eq.0).or.(pmset1.eq.1)) then
            call dograf(plset1,pmset1,pmat1)
            call viewpl(plset1,pmset1)
            if (plset1.lt.2) call WIPSCR(plset1)
            if (pmset1.lt.2) call WIPSCR(pmset1)
         endif
      else
         if ((plset2.eq.0).or.(plset2.eq.1)) then
            call doloci(plset2)
         endif
         if ((pmset2.eq.0).or.(pmset2.eq.1)) then
            call dobode(pmset2)
         endif
         if ((plset2.eq.0).or.(plset2.eq.1).or.
     +       (pmset2.eq.0).or.(pmset2.eq.1)) then
            call dograf(plset2,pmset2,pmat2)
            call viewpl(plset2,pmset2)
            if (plset2.lt.2) call WIPSCR(plset2)
            if (pmset2.lt.2) call WIPSCR(pmset2)
         endif
      endif
      return
      end

C ***************************************************************************
CN    MODULE NAME       : viewpl
CA    FUNCTION          : To isolate a single plot.
CS    CALL SEQUENCE     : call viewpl(lpage,mpage)
CI    INPUT PARAMETERS  : lpage - (integer*2) The characteristic loci plot page.
C                         mpage - (integer*2) The bode and misalignment angle
C                                             plot page.
CO    OUTPUT PARAMETERS: None.
CG    GLOBAL VARIABLES  : keys.inc
C                         plotpg.inc
C                         sysmat.inc
CM    MODULES CALLED    : GETPG (asm), INKEY (asm), prinfo (for), viewl (for),
C                         viewm (for).
CE    ERROR CONDITIONS  : ?
C
CC    COMMENTS          : This routine enables the user to isolate a single plot
C                         on the page. Information concerning that plot is
```

```
C                       displayed  at the top of  the  page. This  information
C                       includes :- Axes scales, I/O names.
C                       Since   all   the  information   cannot  be  presented
C                       simultaneously,  the  user switch  between information
C                       displayed using  the  F4  key. The F4 key affects both
C                       pages  (if  both  have  plots  on  them) : F3  to flip
C                       between the pages.
C
C ***************************************************************************
      subroutine viewpl(lpage,mpage)
      implicit integer*2 (G,I)
      integer*2 lpage,mpage
$include: 'keys.inc'
$include: 'plotpg.inc'
$include: 'sysmat.inc'
      integer*2 key,lpos,lold,mpos,mold,pg,f4stat,f4key,oldf4

      f4key = 62
      f4stat = 1
      oldf4 = 2
      lpos = 1
      mpos = 1
      call viewl(lpage,lpos,2)
      call viewm(mpage,mpos,2)
      call prinfo(f4stat,oldf4,lpage,mpage,lpos,mpos)
99    continue
      key = INKEY(1)
      lold = lpos
      mold = mpos
      pg = GETPG()
      if ((key.eq.tabk).or.(key.eq.retk).or.(key.eq.rightk)) then
         if (pg.eq.lpage) then
            lpos = lpos + 1
            if (lpos.gt.order) lpos = 1
         else
            mpos = mpos + 1
            if (mpos.gt.order) mpos = 1
         endif
      elseif ((key.eq.rtabk).or.(key.eq.leftk)) then
         if (pg.eq.lpage) then
            lpos = lpos - 1
            if (lpos.lt.1) lpos = order
         else
            mpos = mpos - 1
            if (mpos.lt.1) mpos = order
         endif
      elseif (key.eq.upk) then
         if (pg.eq.lpage) then
            lpos = lpos - lwidth
            if (lpos.lt.1) then
               lpos = lpos+lwidth + (lheigh-1)*lwidth
               if (lpos.gt.order) lpos = lpos - lwidth
            endif
         else
            mpos = mpos - mwidth
            if (mpos.lt.1) then
               mpos = mpos+mwidth + (mheigh-1)*mwidth
               if (mpos.gt.order) mpos = mpos - mwidth
            endif
         endif
      elseif (key.eq.downk) then
         if (pg.eq.lpage) then
            lpos = lpos + lwidth
            if (lpos.gt.order) then
               lpos = lpos-lwidth - (lheigh-1)*lwidth
               if (lpos.lt.1) lpos = lpos + lwidth
            endif
         else
            mpos = mpos + mwidth
            if (mpos.gt.order) then
               mpos = mpos-mwidth - (mheigh-1)*mwidth
               if (mpos.lt.1) mpos = mpos + mwidth
```

```
            endif
         endif
      elseif (key.eq.f4key) then
         oldf4 = f4stat
         if (f4stat.eq.1) then
            f4stat = 2
         else
            f4stat = 1
         endif
         call prinfo(f4stat,oldf4,lpage,mpage,lpos,mpos)
      endif
      if (pg.eq.lpage) then
         mpos = lpos
      else
         lpos = mpos
      endif
      if ((lpos.ne.lold).and.(mpos.ne.mold)) then
         call prinfo(f4stat,oldf4,lpage,mpage,lpos,mpos)
         call viewl(lpage,lold,2)
         call viewm(mpage,mold,2)
         call viewl(lpage,lpos,2)
         call viewm(mpage,mpos,2)
      endif
      if (key.ne.esck) goto 99
      return
      end

C ***************************************************************************
CH    REVISION HISTORY :
C     VERSION       BY                DATE        COMMENT
C     1.00          Ian Fisher        23/06/88    Creation.
C     FILEND           :
```

```
C
C     FILE             : PRINFO.FOR
C ***************************************************************************
CN    MODULE NAME      : viewl
CA    FUNCTION         : To isolate a single loci plot.
CS    CALL SEQUENCE    : call viewl(page,position,level)
CI    INPUT PARAMETERS :     page - (integer*2) Graphics page of loci.
C                        position - (integer*2) Position of the loci.
C                           level - (integer*2) Write intensity.
CO    OUTPUT PARAMETERS: None.
CG    GLOBAL VARIABLES : plotpg.inc
CM    MODULES CALLED   : GPAGE (asm), LEVEL (asm), idbox (for).
CE    ERROR CONDITIONS : ?
C
CC    COMMENTS         : Calculates the x,y position of the loci plot to
C                        be isolated, sets the write intensity and then
C                        calls a routine to draw a box around the plot.
C
C ***************************************************************************
      subroutine viewl(vpage,pos,vlev)
      integer*2 vpage,pos,vlev
$include: 'plotpg.inc'
      integer*2 lx,ly,lw,lh,row,col
      if (vpage.lt.2) then
         row = int (pos/lwidth)
         if ((real(pos)/real(lwidth)).gt.(real(int(pos/lwidth))))
     +                                 row = row + 1
         col = pos - (row-1)*lwidth
         lx = lwstrt + (col-1)*lxdel + 1
         ly = lhstrt - (lheigh - row)*lydel - 1
         lw = lxdel - 2
         lh = lydel - 2
         call GPAGE(vpage)
         call LEVEL(vlev)
         call idbox(lx,ly,lw,lh)
         call LEVEL(1)
      endif
      return
      end

C ***************************************************************************
CN    MODULE NAME      : viewm
CA    FUNCTION         : To isolate a bode and misalignment angle plot.
CS    CALL SEQUENCE    : call viewm(page,position,level)
CI    INPUT PARAMETERS :     page - (integer*2) Graphics page of loci.
C                        position - (integer*2) Position of the loci.
C                           level - (integer*2) Write intensity.
CO    OUTPUT PARAMETERS: None.
CG    GLOBAL VARIABLES : plotpg.inc
CM    MODULES CALLED   : GPAGE (asm), LEVEL (asm), idbox (for).
CE    ERROR CONDITIONS : ?
C
CC    COMMENTS         : Calculates  the x,y position of the bode and angle
C                        plot  to be isolated, sets the write intensity and
C                        then calls a routine to draw a box around the plot.
C
C ***************************************************************************
      subroutine viewm(vpage,pos,vlev)
      integer*2 vpage,pos,vlev
$include: 'plotpg.inc'
      integer*2 mx,my,mw,mh,row,col
      if (vpage.lt.2) then
         row = int (pos/mwidth)
         if ((real(pos)/real(mwidth)).gt.(real(int(pos/mwidth))))
     +                                 row = row + 1
         col = pos - (row-1)*mwidth
         mx = mwstrt + (col-1)*mxdel + 1
         my = mhstrt - (mheigh - row)*mydel - 1
         mw = mxdel - 2
         mh = mydel - 2
         call GPAGE(vpage)
         call LEVEL(vlev)
```

```
         call idbox(mx,my,mw,mh)
         call LEVEL(1)
      endif
      return
      end

C ***************************************************************************
CN    MODULE NAME       : idbox
CA    FUNCTION          : To draw a box around a plot and place a dot in the
C                         upper right corner.
CS    CALL SEQUENCE     : call idbox(x,y,width,height)
CI    INPUT PARAMETERS  :    x,y - (integer*2) The x,y co-ords of the box -
C                                              lower left corner.
C                          width - (integer*2) Width in dots.
C                         height - (integer*2) Height in dots.
CO    OUTPUT PARAMETERS: None.
CG    GLOBAL VARIABLES  : None.
CM    MODULES CALLED    : BOX (asm), CIRC (asm), PLOT (asm).
CE    ERROR CONDITIONS  : ?
C
CC    COMMENTS          : Draws a box around a plot and then places a dot in
C                         in the upper right corner.
C
C ***************************************************************************
      subroutine idbox(x,y,wid,hei)
      integer*2 x,y,wid,hei
      call BOX(x,y,wid,hei)
      call CIRC((x+wid-6),(y-hei+3),3)
      call CIRC((x+wid-6),(y-hei+3),2)
      call PLOT((x+wid-6),(y-hei+3))
      call PLOT((x+wid-7),(y-hei+3))
      call PLOT((x+wid-5),(y-hei+3))
      return
      end

C ***************************************************************************
CN    MODULE NAME       : prinfo
CA    FUNCTION          : To print either the isolated plots' I/O names or
C                         the axes scales.
CS    CALL SEQUENCE     : call prinfo(state,oldstate,lpage,mpage,lpos,mpos)
CI    INPUT PARAMETERS  :    state - (integer*2) The current state of the F4
C                                                key.
C                         oldstate - (integer*2) The old state of the F4 key.
C                            lpage - (integer*2) The loci graphics page.
C                            mpage - (integer*2) The bode + angles graphics page
C                             lpos - (integer*2) The current loci position.
C                             mpos - (integer*2) The cureent bode + angles
C                                                position.
CO    OUTPUT PARAMETERS: None.
CG    GLOBAL VARIABLES  : ploci.inc
C                         plotpg.inc
C                         sysnms.inc
CM    MODULES CALLED    : LEVEL (asm), LENSTR (asm), wrtitl (for), prtnum (for),
C                         WRTSTR (asm).
CE    ERROR CONDITIONS  : ?
C
CC    COMMENTS          : Whenever the loci or bode+misalignment angles are
C                         being displayed, the user can isolate one of the plots
C                         and can display information concerning that plot.
C                         Information such as I/O names and axes scales. But
C                         all this information can not be printed at once. Thus
C                         the F4 key is provided to toggle information
C                         concerning the isolated plot.
C
C ***************************************************************************
      subroutine prinfo(f4stat,oldf4,lpage,mpage,lpos,mpos)
      implicit integer*2 (L)
      integer*2 f4stat,oldf4,lpage,mpage,lpos,mpos
$include: 'ploci.inc'
$include: 'plotpg.inc'
$include: 'sysnms.inc'
      integer*2 nlen
```

```
      call LEVEL(1)
      if (lpage.le.1) then
         call WRTSTR(lpage,10,(lhstrt-lhmax-9),13,'Highlighted :')
         call WRTSTR(lpage,130,(lhstrt-lhmax-9),1,'(')
         call prtnum(lpage,140,(lhstrt-lhmax-9),1,4,'(i2)',2,lpos)
         call WRTSTR(lpage,162,(lhstrt-lhmax-9),1,',')
         call prtnum(lpage,171,(lhstrt-lhmax-9),1,4,'(i2)',2,lpos)
         call WRTSTR(lpage,193,(lhstrt-lhmax-9),1,')')
c         if (f4stat.ne.oldf4) then
c            call WRTSTR(lpage,202,(lhstrt-lhmax-9),15,'               ')
c            call WRTSTR(lpage,248,(lhstrt-lhmax-18),51,
c     +           '                                                   ')
c            call WRTSTR(lpage,248,(lhstrt-lhmax-4),51,
c     +           '                                                   ')
c         endif
c         if (f4stat.eq.1) then
c            call WRTSTR(lpage,202,(lhstrt-lhmax-9),5,'  =  ')
c            nlen = LENSTR(25,inpnms(lpos))
c            if (LENSTR(25,outnms(lpos)).gt.nlen) then
c               nlen = LENSTR(25,outnms(lpos))
c            endif
c            call WRTSTR(lpage,248,(lhstrt-lhmax-18),25,
c     +                  '                         ')
c            call WRTSTR(lpage,248,(lhstrt-lhmax-4),25,
c     +                  '                         ')
c            call wrtitl(lpage,248,(lhstrt-lhmax-18),nlen,inpnms(lpos))
c            call WRTSTR(lpage,248,(lhstrt-lhmax-4),nlen,outnms(lpos))
c         else
c            if (f4stat.ne.oldf4) then
         call WRTSTR(lpage,210,(lhstrt-lhmax-9),10,': Scales :')
         call WRTSTR(lpage,305,(lhstrt-lhmax-18),13,
     +              'Real [ max = ')
         call WRTSTR(lpage,305,(lhstrt-lhmax-4),13,
     +              'Imag [ max = ')
         call WRTSTR(lpage,525,(lhstrt-lhmax-18),6,'min = ')
         call WRTSTR(lpage,525,(lhstrt-lhmax-4),6,'min = ')
         call WRTSTR(lpage,680,(lhstrt-lhmax-18),1,']')
         call WRTSTR(lpage,680,(lhstrt-lhmax-4),1,']')
c            endif
         call prtnum(lpage,425,(lhstrt-lhmax-18),4,7,'(f10.4)',
     +              10,xllim(1,lpos))
         call prtnum(lpage,425,(lhstrt-lhmax-4),4,7,'(f10.4)',
     +              10,yllim(1,lpos))
         call prtnum(lpage,580,(lhstrt-lhmax-18),4,7,'(f10.4)',
     +              10,xllim(2,lpos))
         call prtnum(lpage,580,(lhstrt-lhmax-4),4,7,'(f10.4)',
     +              10,yllim(2,lpos))
c         endif
      endif
      if (mpage.le.1) then
c         if (f4stat.ne.oldf4) then
c            call WRTSTR(mpage,202,(mhstrt-mhmax-9),15,'               ')
c            call WRTSTR(mpage,248,(mhstrt-mhmax-18),51,
c     +           '                                                   ')
c            call WRTSTR(mpage,248,(mhstrt-mhmax-4),51,
c     +           '                                                   ')
c         endif
         call WRTSTR(mpage,10,(mhstrt-mhmax-9),13,'Highlighted :')
         call WRTSTR(mpage,130,(mhstrt-mhmax-9),1,'(')
         call prtnum(mpage,140,(mhstrt-mhmax-9),1,4,'(i2)',2,mpos)
         call WRTSTR(mpage,162,(mhstrt-mhmax-9),1,',')
         call prtnum(mpage,171,(mhstrt-mhmax-9),1,4,'(i2)',2,mpos)
         call WRTSTR(mpage,193,(mhstrt-mhmax-9),6,')  =  ')
c         if (f4stat.eq.1) then
c            nlen = LENSTR(25,inpnms(mpos))
c            if (LENSTR(25,outnms(mpos)).gt.nlen) then
c               nlen = LENSTR(25,outnms(mpos))
c            endif
c            call WRTSTR(mpage,248,(mhstrt-mhmax-18),25,
c     +                  '                         ')
c            call WRTSTR(mpage,248,(mhstrt-mhmax-4),25,
c     +                  '                         ')
```

```
c            call wrtitl(mpage,248,(mhstrt-mhmax-18),nlen,inpnms(mpos))
c            call WRTSTR(mpage,248,(mhstrt-mhmax-4),nlen,outnms(mpos))
c         else
c            if (f4stat.ne.oldf4) then
          call WRTSTR(mpage,210,(mhstrt-mhmax-9),10,': Scales :')
          call WRTSTR(mpage,305,(mhstrt-mhmax-18),15,
     +                'Bode    [ max = ')
          call WRTSTR(mpage,305,(mhstrt-mhmax-4),15,
     +                'Angles [ max = ')
          call WRTSTR(mpage,540,(mhstrt-mhmax-18),6,'min = ')
          call WRTSTR(mpage,540,(mhstrt-mhmax-4),6,'min = ')
          call WRTSTR(mpage,695,(mhstrt-mhmax-18),1,']')
          call WRTSTR(mpage,695,(mhstrt-mhmax-4),1,']')
c            endif
          call prtnum(mpage,440,(mhstrt-mhmax-18),4,7,'(f10.4)',
     +                10,xblim(1,lpos))
          call prtnum(mpage,440,(mhstrt-mhmax-4),4,7,'(f10.4)',
     +                10,xmlim(1,lpos))
          call prtnum(mpage,595,(mhstrt-mhmax-18),4,7,'(f10.4)',
     +                10,xblim(2,lpos))
          call prtnum(mpage,595,(mhstrt-mhmax-4),4,7,'(f10.4)',
     +                10,xmlim(2,lpos))
c         endif
       endif
       oldf4 = f4stat
       return
       end

C ******************************************************************************
CH     REVISION HISTORY :
C      VERSION       BY                DATE        COMMENT
C      1.00          Ian Fisher        23/06/88    Creation.
C      FILEND           :
```

```
C
C     FILE              : DOG.FOR
C ***************************************************************************
CN    MODULE NAME       : dograf
CA    FUNCTION          : To calculate all values and plot them.
CS    CALL SEQUENCE     : call dograf(lpage,mpage,pmat)
CI    INPUT PARAMETERS  : lpage - (INTEGER*2) The page on which the loci
C                                               are plotted.
C                         mpage - (INTEGER*2) The page on which the angles
C                                               are plotted.
C                          pmat - (INTEGER*2) The type of system :
C                                               0 : controller included.
C                                               1 : controller excluded.
CO    OUTPUT PARAMETERS: None.
CG    GLOBAL VARIABLES  : plotfr.inc
C                         sysmat.inc
C                         ploci.in
C                         plotpg.inc
CM    MODULES CALLED    :
CE    ERROR CONDITIONS  : ?
C
CC    COMMENTS          : Steps through the calculated frequencies and
C                         calculates the eigen values or loci ,tracks them and
C                         plots them on the desired axes.
C
C ***************************************************************************
      subroutine dograf(lpage,mpage,pmat)
      implicit integer*2 (c,I)
      integer*2 lpage,mpage,pmat
$include: 'sysmat.inc'
$include: 'plotfr.inc'
$include: 'ploci.inc'
$include: 'plotpg.inc'
      real*4 fr
      integer*2 i,materr
      integer*2 key
      character*20 frqstr

      frqstr = '[]'
      if (ftype.eq.1) then
         frqstr = '[Hertz]    '
      elseif (ftype.eq.2) then
         frqstr = '[Rads/Sec] '
      elseif (ftype.eq.3) then
         frqstr = '[Rads/Hour]'
      endif
      if ((lpage.eq.0).or.(mpage.eq.0)) then
         call WRTSTR(0,10,(lhstrt-lhmax-18),50,
     +   '                                                  ')
         call WRTSTR(0,10,(lhstrt-lhmax-18),12,
     +   'Frequency = ')
         call WRTSTR(0,235,(lhstrt-lhmax-18),11,frqstr)
         call WRTSTR(0,10,(lhstrt-lhmax-4),25,
     +   'Hit ESC to quit plotting.')
      endif
      if ((lpage.eq.1).or.(mpage.eq.1)) then
         call WRTSTR(1,10,(lhstrt-lhmax-18),50,
     +   '                                                  ')
         call WRTSTR(1,10,(lhstrt-lhmax-18),12,
     +   'Frequency = ')
         call WRTSTR(1,235,(lhstrt-lhmax-18),11,frqstr)
         call WRTSTR(1,10,(lhstrt-lhmax-4),25,
     +   'Hit ESC to quit plotting.')
      endif
      i = 1
      key = 0
      if (freqs(i).ne.(-1.0)) then
99       continue
         if ((lpage.eq.0).or.(mpage.eq.0)) then
            call prtnum(0,130,(lhstrt-lhmax-18),3,7,'(g10.4)',10,
     +                  freqs(i))
         endif
```

```
            if ((lpage.eq.1).or.(mpage.eq.1)) then
               call prtnum(1,130,(lhstrt-lhmax-18),3,7,'(g10.4)',10,
     +                     freqs(i))
            endif
            if (pmat.eq.1) then
               call evpoly(freqs(i),order,gmat,calcr,calci)
            else
               call evpoly(freqs(i),order,gmat,calcr2,calci2)
               call evpoly(freqs(i),order,kmat,calcr3,calci3)
               materr = cmult(order,order,calcr2,calci2,
     +                        order,order,calcr3,calci3,
     +                        calcr,calci)
            endif
            call qzveca(order,calcr,calci,alfr,alfi,zr,zi,materr)
            if (materr.eq.0) then
               call track(i,order,alfr,alfi,zr,zi,mpage)
               call pltsys(order,alfr,alfi,zr,zi,freqs(i),lpage,mpage)
            else
               call ERTONE()
            endif
            i = i + 1
            key = INKEY(4)
            if (key.ne.0) then
               call chkstp(key,lpage,mpage)
               if ((lpage.eq.0).or.(mpage.eq.0)) then
                  call WRTSTR(0,10,(lhstrt-lhmax-18),50,
     +                 '                                                  ')
                  call WRTSTR(0,10,(lhstrt-lhmax-18),12,
     +                 'Frequency = ')
                  call WRTSTR(0,235,(lhstrt-lhmax-18),11,frqstr)
                  call WRTSTR(0,10,(lhstrt-lhmax-4),25,
     +                 'Hit ESC to quit plotting.')
               endif
               if ((lpage.eq.1).or.(mpage.eq.1)) then
                  call WRTSTR(1,10,(lhstrt-lhmax-18),50,
     +                 '                                                  ')
                  call WRTSTR(1,10,(lhstrt-lhmax-18),12,
     +                 'Frequency = ')
                  call WRTSTR(1,235,(lhstrt-lhmax-18),11,frqstr)
                  call WRTSTR(1,10,(lhstrt-lhmax-4),25,
     +                 'Hit ESC to quit plotting.')
               endif
            endif
            if ((freqs(i).ne.(-1.0)).and.(i.le.300).and.((key.eq.0).
     +          or.(key.eq.2))) goto 99
         endif
         if ((lpage.eq.0).or.(mpage.eq.0)) then
            call WRTSTR(0,10,(lhstrt-lhmax-18),50,
     +           '                                                  ')
            call WRTSTR(0,10,(lhstrt-lhmax-4),50,
     +           '                                                  ')
         endif
         if ((lpage.eq.1).or.(mpage.eq.1)) then
            call WRTSTR(1,10,(lhstrt-lhmax-18),50,
     +           '                                                  ')
            call WRTSTR(1,10,(lhstrt-lhmax-4),50,
     +           '                                                  ')
         endif
         return
         end

C ******************************************************************************
CH    REVISION HISTORY :
C     VERSION       BY                DATE        COMMENT
C     1.00          Ian Fisher        23/06/88    Creation.
C     FILEND           :
```

```
C     FILE             : DOGRAF.FOR
C ***************************************************************************
CN    MODULE NAME      : EVPOLY
CA    FUNCTION         : To evaluate a matrix of polynomials at a particular
C                        frequency.
CS    CALL SEQUENCE    : call evpoly(fr,n,mat,calr,cali)
CI    INPUT PARAMETERS :
C                  fr - (real*4) Frequency value.
C                   n - (integer*2) Order of the system.
C     mat(10,10,2,13) - (real*4) The matrix of polynomials.
C
CO    OUTPUT PARAMETERS:
C           calr(n,n)
C           cali(n,n) - (real*4) The resultant complex matrix.
C
CG    GLOBAL VARIABLES : None.
CM    MODULES CALLED   : calpol (for).
CE    ERROR CONDITIONS : ?
C
CC    COMMENTS         : Step through the entire matrix calling routines to
C                        evaluate each polynomial in turn and  placing  the
C                        results in a constant complex matrix.
C ***************************************************************************
      subroutine evpoly(fr,n,mat,calr,cali)
      implicit complex*8 (c)
      implicit integer*2 (I)
      real*4 fr
      integer*2 n
      real*4 mat(10,10,2,13),calr(n,n),cali(n,n)
      complex*8 num,den,dead,fpol
      integer*2 i,j
      integer*2 key
      real*4 t1,t2

      do 199 i = 1,n
         do 198 j = 1,n
            num = calpol(i,j,1,mat,fr)
            den = calpol(i,j,2,mat,fr)
            dead = cmplx((0.0),(-1.0*fr*mat(i,j,1,12)))
            dead = cexp(dead)
            fpol = (num*dead)/den
            calr(i,j) = real (fpol)
            cali(i,j) = aimag(fpol)
198      continue
199   continue

      return
      end

C ***************************************************************************
CN    MODULE NAME      : CALPOL
CA    FUNCTION         : To evaluate a poynomial at a particular frequency.
CS    CALL SEQUENCE    : val = calpol(i,j,nord,mat,fr)
CI    INPUT PARAMETERS :
C                 i,j - (integer*2) Position of polynomial in matrix.
C                nord - (integer*2) Numerator or denominator.
C     mat(10,10,2,13) - (real*4) The matrix of polynomials.
C                  fr - (real*4) The frequency of evaluation.
C
CO    OUTPUT PARAMETERS:
C                 val - (complex*8) The result of polynomial evaluation.
CG    GLOBAL VARIABLES : None.
CM    MODULES CALLED   : None.
CE    ERROR CONDITIONS : ?
C
CC    COMMENTS         : Steps through from 0th to max. power and calculates
C                        each element and adding to a running total.
C ***************************************************************************
      complex*8 function calpol(i,j,nord,mat,fr)
      integer*2 i,j,nord
      real*4 mat(10,10,2,13),fr
      integer*2 polord,k
```

```
      complex*8 calc
      calc = cmplx(0.0,0.0)
      polord = int (mat(i,j,nord,13))
      do 299 k = 1,(polord+1)
         if (k.eq.1) then
            calc = calc +
     +        (cmplx(mat(i,j,nord,k),(0.0)))
         else
            calc = calc +
     +        (cmplx(mat(i,j,nord,k),(0.0)))*((cmplx((0.0),fr))**(k-1))
         endif
299   continue
      calpol = calc
      if ((polord.eq.0).and.(nord.eq.2).and.
     +    (abs(mat(i,j,nord,0)).lt.(1.0e-15))) then
         calpol = cmplx(1.0,0.0)
      endif
      return
      end

C ***************************************************************************
CN    MODULE NAME       : PLTSYS
CA    FUNCTION          : To plot a set of points on the loci or bode diagrams.
CS    CALL SEQUENCE     : call pltsys(n,xpts,ypts,zr,zi,frq,lpg,mpg)
CI    INPUT PARAMETERS :
C                    n - (integer*2) Order of system.
C               xpts(10)
C               ypts(10) - (real*4) The Characteristic loci points.
C                zr(n,n)
C                zi(n,n) - (real*4) The eigenvectors.
C                    frq - (real*4) The frequency of evaluation.
C                    lpg - (integer*2) The Char. loci page.
C                    mpg - (integer*2) The bode and misalignment angle page.
C
CO    OUTPUT PARAMETERS: None.
CG    GLOBAL VARIABLES : ploci.inc
C                        plotpg.inc
CM    MODULES CALLED   : plotpt (for).
CE    ERROR CONDITIONS : ?
C
CC    COMMENTS         : Plots the loci, bode and misalignment angles on the
C                        appropriate pages and on the appropriate set of axes.
C                        If the bode and misalignment angles are to be plotted,
C                        then these values are calculated just before plotting.
C ***************************************************************************
      subroutine pltsys(n,xpts,ypts,zr,zi,frq,lpg,mpg)
      integer*2 n
      real*4 xpts(10),ypts(10),zr(n,n),zi(n,n),frq
      integer*2 lpg,mpg
$include: 'ploci.inc'
$include: 'plotpg.inc'
      real*8 cent(4),scal(2)

      integer*2 xtype,k
      real*4 tbode,tangl
      real*8 num1,num2

      xtype = 1
      if ((lpg.eq.0).or.(lpg.eq.1)) then
         do 399 i = 1,n
            do 398 j = 1,4
               cent(j) = lcentre(j,i)
398         continue
            scal(1) = lscale(1,i)
            scal(2) = lscale(2,i)
            call plotpt(lpg,xtype,xpts(i),ypts(i),cent,scal)
399      continue
      endif
      if ((mpg.eq.0).or.(mpg.eq.1)) then
         xtype = 1
         if (inctyp.eq.1) then
            xtype = 1
```

```
         elseif (inctyp.eq.2) then
            xtype = 2
         elseif (inctyp.eq.3) then
            xtype = 3
         elseif (inctyp.eq.4) then
            xtype = 1
         endif
         do 389 i = 1,n
            do 388 j = 1,4
               cent(j) = bcentre(j,i)
388         continue
            scal(1) = bscale(1,i)
            scal(2) = bscale(2,i)
            tbode = 20.0*log10(sqrt((xpts(i)*xpts(i))
     +                        + (ypts(i)*ypts(i))))
            call plotpt(mpg,xtype,frq,tbode,cent,scal)

            num2 = 0.0
            do 386 j = 1,n
               num2 = zr(j,i)*zr(j,i) + zi(j,i)*zi(j,i) + num2
386         continue
            do 387 j = 1,4
               cent(j) = mcentre(j,i)
387         continue
            scal(1) = mscale(1,i)
            scal(2) = mscale(2,i)
            do 384 k = 1,n
               num1 = sqrt(zr(k,i)**2 + zi(k,i)**2)
c               num1 = sqrt(zr(i,i)**2 + zi(i,i)**2)
               if (num2.gt.(0.0)) then
                  num1 = num1/sqrt(num2)
                  if ((num1.le.(1.0)).and.(num1.ge.(0.0))) then
                     tangl = (180.0/3.14159265)*acos(num1)
                     call plotpt(mpg,xtype,frq,tangl,cent,scal)
                  endif
               endif
384         continue
389      continue
      endif
      return
      end

C ***************************************************************************
CN    MODULE NAME       : CHKDUP
CA    FUNCTION          : To check that no eigenvalue is selected more than once
C                         during tracking.
CS    CALL SEQUENCE     : chk = chkdup(mindst,i,j)
CI    INPUT PARAMETERS :
C
C           mindst(10) - (integer*2) The current set of eigenvalue
C                        associations.
C                    i - (integer*2) The current position in the 'mindst' array
C                    j - (integer*2) The value to check for in the 'mindst'
C                        array.
C
CO    OUTPUT PARAMETERS:
C                  chk - (integer*2) The condition returned :
C                        0 : Value has not been selected yet.
C                        1 : Value found in the array.
CG    GLOBAL VARIABLES : None.
CM    MODULES CALLED   : None.
CE    ERROR CONDITIONS : ?
C
CC    COMMENTS         : Steps through the array checking for the value.
C ***************************************************************************
      integer*2 function chkdup(mindst,i,j)
      integer*2 mindst(10),i,j
      integer*2 k
      chkdup = 0
      do 550 k = 1,(i-1)
         if ((mindst(k).eq.j).and.(chkdup.eq.0)) then
            chkdup = 1
```

```
         endif
550   continue
      return
      end

C ***************************************************************************
CN    MODULE NAME       : GETANG
CA    FUNCTION          : To produce the current track angle of an eigenvalue.
CS    CALL SEQUENCE     : angle = getang(dy,dx)
CI    INPUT PARAMETERS :
C                  dy - (real*4) Change in imaginary axis.
C                  dx - (real*4) Change in real axis.
C
CO    OUTPUT PARAMETERS:
C               angle - (real*4) The resultant angle in radians.
C
CG    GLOBAL VARIABLES : None.
CM    MODULES CALLED   : None.
CE    ERROR CONDITIONS : ?
C
CC    COMMENTS         : Straight trigonometry.
C ***************************************************************************
      real*4 function getang(dy,dx)
      real*4 dy,dx
      integer*2 sx,sy
      real*4 temp
      sy = 1
      if (dy.lt.(0.0)) sy = -1
      sx = 1
      if (dx.lt.(0.0)) sx = -1
      if ((sx.gt.0).and.(sy.gt.0)) then
         temp = 0.0
      elseif ((sx.gt.0).and.(sy.lt.0)) then
         temp = 3.141592654*1.5
      elseif ((sx.lt.0).and.(sy.lt.0)) then
         temp = 3.141592654
      else
         temp = 3.141592654/2
      endif
      if (abs(dx).lt.(1.0e-10)) then
         getang = 3.141592654/2 + temp
      else
         getang = atan((abs(dy))/(abs(dx))) + temp
      endif
      return
      end

C ***************************************************************************
CN    MODULE NAME       : CHKSTP
CA    FUNCTION          : Checks if the user wishes to quit the plot.
CS    CALL SEQUENCE     : call chkstp(key,lpg,mpg)
CI    INPUT PARAMETERS :
C                 lpg - (integer*2) The char. loci page.
C                 mpg - (integer*2) The bode and misalignment angle page.
C
CO    OUTPUT PARAMETERS:
C                 key - (integer*2) The returned key or state.
C
CG    GLOBAL VARIABLES : keys.inc
CM    MODULES CALLED   : INKEY (asm), GETPG (asm), WRTSTR (asm), STRIN (asm).
CE    ERROR CONDITIONS : ?
C
CC    COMMENTS         : Obtains confirmation from the user to quit the plot
C                        of char. loci, etc. before the frequency sweep has
C                        run to completion.
C ***************************************************************************
      subroutine chkstp(key,lpg,mpg)
      implicit integer*2 (G,I,S)
      integer*2 key,lpg,mpg
      integer*2 flush,pg,hstrt,hmax
      character*1 yesno
$include: 'keys.inc'
```

```
      flush = INKEY(2)
      if (key.eq.esck) then
         pg = GETPG()
         if ((lpg.ne.pg).and.(mpg.ne.pg)) then
            if ((lpg.eq.1).or.(lpg.eq.0)) then
               pg = lpg
            elseif ((mpg.eq.1).or.(mpg.eq.0)) then
               pg = mpg
           endif
         endif
         if (pg.eq.0) then
            hmax   = 222
            hstrt = 268
         else
            hmax   = 270
            hstrt = 316
         endif
         call WRTSTR(pg,10,(hstrt-hmax-4),50,
     +        '                                                  ')
         call WRTSTR(pg,10,(hstrt-hmax-4),33,
     +        'Do you want quit plotting (y/n) ?')
         yesno = 'n'
111      continue
         key = STRIN(pg,315,(hstrt-hmax-4),1,yesno)
         if (key.ne.retk) goto 111
         if ((yesno.eq.'y').or.(yesno.eq.'Y')) then
            key = 1
         elseif (yesno.eq.'q') then
            key = 2
         else
            key = 0
         endif
      else
         key = 0
      endif
      call WRTSTR(pg,10,(hstrt-hmax-4),50,
     +     '                                                  ')
      return
      end

C ******************************************************************************
CH    REVISION HISTORY :
C     VERSION       BY                DATE      COMMENT
C     1.00          Ian Fisher        23/06/88  Creation.
C     FILEND           :
```

```
C
C     FILE             : TRACK.FOR
C ***************************************************************************
CN    MODULE NAME      : TRACK
CA    FUNCTION         : To track a set of eigenvalues.
CS    CALL SEQUENCE    : call track(pntnum,n,eigr,eigi,frq,er,ei,mpg)
CI    INPUT PARAMETERS :
C                pntnum - (integer*2) The state of eigenvalues :
C                           1  : First set of eigenvalues - no tracking yet.
C                           2  : Second set of eigenvalues - no tracking yet.
C                           >2 : Track eigenvalues.
C                     n - (integer*2) Order of the system.
C              eigr(10)
C              eigi(10) - (real*4) The eigenvalues.
C               er(n,n)
C               ei(n,n) - (real*4) The eigenvectors.
C
CO    OUTPUT PARAMETERS:
C              eigr(10)
C              eigi(10) - (real*4) The sorted eigenvalues.
C               er(n,n)
C               ei(n,n) - (real*4) The sorted eigenvectors.
C
CG    GLOBAL VARIABLES : plotfr.inc
C                        sysmat.inc
CM    MODULES CALLED   : getang (for), chkdup (for).
CE    ERROR CONDITIONS : ?
C
CC    COMMENTS         : The routine tracks the eigenvalues by  using  the last
C                        two points to predict  the next eigenvalue. The method
C                        then goes through each  eigenvalue and selects the one
C                        eigenvalue that is closest to the predicted value.
C                        The value associations  are  kept  in  a table for the
C                        transformations at a later stage and for checking that
C                        no duplication of association takes place.
C ***************************************************************************
      subroutine track(pntnum,n,eigr,eigi,er,ei,mpg)
      implicit integer*2 (c,I)
      implicit real*4 (g)
      integer*2 pntnum,n
      real*4 eigr(10),eigi(10)
      real*4 er(n,n)
      real*4 ei(n,n)
      integer*2 mpg
$include: 'plotfr.inc'
$include: 'sysmat.inc'
      real*4 hypot1,hypot2,minhyp
      real*4 minang,angl,angl2,diff
      integer*2 minele,dupl,i,j
      real*4 nptsi(10),nptsr(10)
      integer*2 mindst(10)
      integer*2 key

      do 480 i = 1,n
         mindst(i) = 0
480   continue
      if (pntnum.eq.1) then
         do 499 i = 1,n
            evalr2(i) = eigr(i)
            evali2(i) = eigi(i)
499      continue
      elseif (pntnum.eq.2) then
         do 498 i = 1,n
            evalr1(i) = eigr(i)
            evali1(i) = eigi(i)
498      continue
      elseif (pntnum.gt.2) then
         do 494 i = 1,n
            hypot1 = sqrt((abs(evalr1(i))-abs(evalr2(i)))**2 +
     +                    (abs(evali1(i))-abs(evali2(i)))**2)
            angl=getang((evali1(i)-evali2(i)),(evalr1(i)-evalr2(i)))
            nptsr(i) = evalr1(i) + hypot1*cos(angl)
```

```
            nptsi(i) = evali1(i) + hypot1*sin(angl)
494      continue
         do 493 i = 1,n
            do 492 j = 1,n
               dupl = chkdup(mindst,i,j)
               if (dupl.eq.0) then
                  hypot1 = sqrt((abs(eigr(j))-abs(nptsr(i)))**2 +
     +                    (abs(eigi(j))-abs(nptsi(i)))**2)
                  if (mindst(i).eq.0) then
                     minhyp = hypot1
                     mindst(i) = j
                  else
                     if (hypot1.lt.minhyp) then
                        minhyp = hypot1
                        mindst(i) = j
                     endif
                  endif
               endif
492         continue
493      continue
         do 495 i = 1,n
            evalr2(i) = evalr1(i)
            evali2(i) = evali1(i)
495      continue
         do 490 i = 1,n
            evalr1(i) = eigr(mindst(i))
            evali1(i) = eigi(mindst(i))
490      continue
         do 489 i = 1,n
            eigr(i) = evalr1(i)
            eigi(i) = evali1(i)
489      continue
         if ((mpg.eq.0).or.(mpg.eq.1)) then
            do 470 i = 1,n
               do 469 j = 1,n
                  calcr2(((j-1)*n)+i) = er(j,(mindst(i)))
469            continue
470         continue
            do 468 i = 1,n
               do 465 j = 1,n
                  er(j,i) = calcr2(((j-1)*n)+i)
465            continue
468         continue
            do 370 i = 1,n
               do 369 j = 1,n
                  calcr2(((j-1)*n)+i) = ei(j,(mindst(i)))
369            continue
370         continue
            do 368 i = 1,n
               do 365 j = 1,n
                  ei(j,i) = calcr2(((j-1)*n)+i)
365            continue
368         continue
         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     FILE              : GETK.FOR
C ***************************************************************************
CN    MODULE NAME       : GETK
CA    FUNCTION          : To drive the controller generation menu.
CS    CALL SEQUENCE     : call getk()
CI    INPUT PARAMETERS  : None.
CO    OUTPUT PARAMETERS: None.
CG    GLOBAL VARIABLES  : None.
CM    MODULES CALLED    : WIPSCR (asm), DOMENU (asm), WRTSTR (asm), scalar (for)
C                         pimat (for), vector (for), matmul (for), edtwrk (for).
CE    ERROR CONDITIONS  : ?
C
CC    COMMENTS          : The routine drives the menu for controller generation
C                         and editing. This is achieved by calling  subroutines
C                         to perform the tasks upon request of the user.
C                         The ESC key returns control to the calling routine.
C ***************************************************************************
      subroutine getk()
      implicit integer*2 (D)

      integer*2 opt122
      call WIPSCR(0)
99    continue
      call wrtit1(0,250,35,18,'Controller Design.')
      opt122 = DOMENU(122)
      call WRTSTR(0,250,35,19,'                    ')
      if (opt122.eq.1) then
         call scalar()
      elseif (opt122.eq.2) then
         call pimat()
      elseif (opt122.eq.3) then
         call vector()
      elseif (opt122.eq.4) then
         call matmul()
      elseif (opt122.eq.5) then
         call edtwrk()
      endif
      if (opt122.ne.0) goto 99
      call WIPSCR(0)
      return
      end

C ***************************************************************************
CN    MODULE NAME       : EDTWRK
CA    FUNCTION          : To setup the call to the matrix edit menu.
CS    CALL SEQUENCE     : call edtwrk()
CI    INPUT PARAMETERS  : None.
CO    OUTPUT PARAMETERS: None.
CG    GLOBAL VARIABLES  : sysmat.inc
CM    MODULES CALLED    : getmat (for), WIPSCR (asm).
CE    ERROR CONDITIONS  : ?
C
CC    COMMENTS          : Sets  up  the  variables  for  a call to the symbolic
C                         matrix editing menu. This involves giving the routine
C                         a matrix title, filename and screen heading.
C ***************************************************************************
      subroutine edtwrk()
$include: 'sysmat.inc'

      character*25 name,fname,mtitle
      character*4 match,defdir
      mtitle = 'Editing the Work Matrix'
      name = 'Temporary Work Matrix.'
      fname = 'temp.ks'
      match = 'Temp'
      defdir = '*.ks'
      call getmat(match,mtitle,23,order,k2mat,name,fname,defdir,5)
      call WIPSCR(0)
      return
      end
```

```
C ***************************************************************************
CN    MODULE NAME       : SCALAR
CA    FUNCTION          : To generate a temporary scalar matrix.
CS    CALL SEQUENCE     : call scalar()
CI    INPUT PARAMETERS  : None.
CO    OUTPUT PARAMETERS: None.
CG    GLOBAL VARIABLES  : sysmat.inc
C                         keys.inc
CM    MODULES CALLED    : WIPSCR (asm),  DISP (asm),  LEVEL (asm), wrtitl (for),
C                         chkop (for), setzer (for), WRTSTR (asm), prtnum (for),
C                         scelem (for), getnum (for).
CE    ERROR CONDITIONS  : ?
C
CC    COMMENTS          : Enables the user to create a scalar matrix. The user
C                         can put any scalar value on each diagonal element of
C                         an identity matrix. The resultant scalar matrix is
C                         left in the temporary work matrix.
C                         The routine first clears the work matrix (after
C                         obtaining the users' permission) and inserts an
C                         matrix. The user is then allowed to edit the
C                         coefficients of the 0th power of the numerator
C                         diagonal elements.
C                         A duplicate facility is provided if any elements are
C                         to be repeated on the diagonal (CTRL-Q).
C                         The ESC key returns control to the calling routine.
C ***************************************************************************
      subroutine scalar()
      implicit integer*2 (c,g)
$include: 'sysmat.inc'
$include: 'keys.inc'
      integer*2 xpos,ypos,i,pos,pos2,key,ctrlq,chk
      call WIPSCR(1)
      call DISP(1)
      call LEVEL(1)

      call wrtitl(1,150,20,25,'Scalar Matrix Definition.')
      chk = 0
      chk = chkop(1)
      if (chk.eq.0) then
         call setzer(k2mat,1.0)
         call wrtitl(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 prtnum(1,xpos+10,ypos,1,4,'(i2)',2,i)
               call WRTSTR(1,xpos+30,ypos,1,',')
               call prtnum(1,xpos+40,ypos,1,4,'(i2)',2,i)
               call WRTSTR(1,xpos+60,ypos,8,')   K = ')
               call prtnum(1,(xpos+132),ypos,3,7,'(g10.4)',10,
     +                     k2mat(i,i,1,1))
               ypos = ypos + 14
99          continue
            call wrtitl(1,400,40,3,'In ')
            call WRTSTR(1,400,54,3,'Out')
            call WRTSTR(1,436,47,1,'=')
            call WRTSTR(1,400,90,30,'Return, TAB and cursor keys to')
            call WRTSTR(1,400,105,32,'move. CTRL-Q to duplicate field.')
            call WRTSTR(1,400,120,12,'ESC to quit.')

            pos = 1
            ypos = 39
            xpos = xpos + 132
            ctrlq = 4352
98          continue
            call scelem(pos)
            key = getnum(1,xpos,(ypos+(pos-1)*14),3,7,'(g10.4)',10,
     +                   k2mat(pos,pos,1,1))
            if ((key.eq.downk).or.(key.eq.retk).or.(key.eq.tabk)) then
               pos = pos + 1
               if (pos.gt.order) pos = 1
```

```
            elseif ((key.eq.upk).or.(key.eq.rtabk)) then
               pos = pos - 1
               if (pos.lt.1) pos = order
            elseif (key.eq.ctrlq) then
               pos2 = pos + 1
               if (pos2.gt.order) pos2 = 1
               k2mat(pos2,pos2,1,1) = k2mat(pos,pos,1,1)
               pos = pos + 1
               if (pos.gt.order) pos = 1
            endif
            if (key.ne.esck) goto 98
            call wipscr(1)
         endif
      else
         call WRTSTR(1,40,60,34,'Scalar matrix operation cancelled.')
      endif
      call wipscr(1)
      return
      end

C ***************************************************************************
CN    MODULE NAME       : SCELEM
CA    FUNCTION          : To print input/output names for scalar matrix
C                         generation.
CS    CALL SEQUENCE     : call scelem(elem)
CI    INPUT PARAMETERS :
C                elem - (integer*2) The element number.
C
CO    OUTPUT PARAMETERS: None.
CG    GLOBAL VARIABLES : sysnms.inc
CM    MODULES CALLED   : WRTSTR (asm), LENSTR (asm), wrtitl (for).
CE    ERROR CONDITIONS : ?
C
CC    COMMENTS          : To print the input and output names of a particular
C                         diagonal element.
C ***************************************************************************
      subroutine scelem(elem)
      implicit integer*2 (L)
      integer*2 elem
$include: 'sysnms.inc'
      integer*2 len
      call WRTSTR(1,455,40,26,'                          ')
      call WRTSTR(1,455,54,26,'                          ')
      len = LENSTR(25,inpnms(elem))
      if (LENSTR(25,outnms(elem)).gt.len) then
         len = LENSTR(25,outnms(elem))
      endif
      call wrtitl(1,455,40,len,inpnms(elem))
      call WRTSTR(1,455,54,len,outnms(elem))
      return
      end

C ***************************************************************************
CN    MODULE NAME       : MATMUL
CA    FUNCTION          : To drive the menu for symbolic multiplication.
CS    CALL SEQUENCE     : call matmul()
CI    INPUT PARAMETERS : None.
CO    OUTPUT PARAMETERS: None.
CG    GLOBAL VARIABLES : sysmat.inc
C                         keys.inc
CM    MODULES CALLED   : WIPSCR (asm), wrtitl (for), DOMENU (asm), WRTSTR (asm)
C                         symult (for), ERTONE (asm), INKEY (asm).
CE    ERROR CONDITIONS : ?
C
CC    COMMENTS          : The menu driver for the symbolic matrix multiplication
C                         routine.
C                         Only two choices are provided :
C                             K(s) X Work_matrix  or
C                              Work_matrix X K(s)
C
C                         The routine also does a check if the multiply was
C                         completed without error, if not an error message is
```

```
C                       printed out.
C                       The ESC key returns control to the calling routine.
C
C ***************************************************************************
      subroutine matmul()
      implicit integer*2 (D,I)
$include: 'sysmat.inc'
$include: 'keys.inc'
      integer*2 opt1226,err,key
      call WIPSCR(0)
99    continue
      call wrtitl(0,250,35,31,'Symbolic Matrix Multiplication.')
      opt1226 = DOMENU(1226)
      if (opt1226.eq.1) then
         call WRTSTR(0,150,60,39,
     +        'Multiplying rows of K(s) by the columns')
         call WRTSTR(0,150,75,23,
     +        'of the work matrix.....')
         err = 0
         call symult(order,kmat,k2mat,1,err)
      elseif (opt1226.eq.2) then
         call WRTSTR(0,150,60,38,
     +        'Multiplying rows of the work matrix by')
         call WRTSTR(0,150,75,23,
     +        'the columns of K(s)....')
         err = 0
         call symult(order,kmat,k2mat,2,err)
      elseif (opt1226.eq.3) then
         call WRTSTR(0,150,60,51,
     +        'Multiplying rows of G(s) by the columns of K(s)....')
         err = 0
         call symult(order,gmat,kmat,1,err)
      elseif (opt1226.eq.4) then
         call WRTSTR(0,150,60,31,
     +        'Multiplying rows of G(s) by the')
         call WRTSTR(0,150,75,26,
     +        'columns of work matrix....')
         err = 0
         call symult(order,gmat,k2mat,1,err)
      endif
      if ((opt1226.ge.1).and.(opt1226.le.4)) then
         if (err.eq.0) then
            call WRTSTR(0,360,75,5,'Done.')
         elseif (err.eq.1) then
            call ERTONE()
            call WRTSTR(0,150,95,59,
     +'*** Warning : Operation not complete : polynomial order < 0')
         elseif (err.eq.2) then
            call ERTONE()
            call WRTSTR(0,150,95,60,
     +'*** Warning : Operation not complete : polynomial order > 10')
            call WRTSTR(0,150,120,22,'Hit RETURN to proceed.')
         endif
         call WRTSTR(0,150,120,22,'Hit RETURN to proceed.')
555      continue
         key = INKEY(1)
         if (key.ne.retk) goto 555
      endif
      if ((opt1226.lt.0).or.(opt1226.gt.4)) goto 99
      call WIPSCR(0)
      return
      end

C ***************************************************************************
CN    MODULE NAME       : CHKOP
CA    FUNCTION          : To check if the user wishes to proceed with an
C                         operation that will clear thetemporary matrix.
CS    CALL SEQUENCE     : reply = chkop(pg)
CI    INPUT PARAMETERS :
C                   pg - (integer*2) The page on which the request is to be
C                        printed.
C
```

```
CO    OUTPUT PARAMETERS:
C                reply - (integer*2) State of user reply :
C                        0 : answer = yes
C                        1 : answer = no
C
CG    GLOBAL VARIABLES : keys.inc
CM    MODULES CALLED   : WRTSTR (asm), STRIN (asm).
CE    ERROR CONDITIONS : ?
C
CC    COMMENTS         : Requests the users permission to clear the work matrix
C                        before a controller generation routine uses the matrix
C ***************************************************************************
      integer*2 function chkop(pg)
      implicit integer*2 (S)
      integer*2 pg
$include: 'keys.inc'
      integer*2 key
      character*1 yesno
      yesno = 'n'
      call WRTSTR(pg,40,60,51,
     +     'The following operation will clear the work matrix.')
      call WRTSTR(pg,40,75,30,'Do you wish to proceed (y/n) ?')
      call WRTSTR(pg,40,110,26,'Hit RETURN to enter reply.')
      call WRTSTR(pg,40,125,16,'Hit ESC to exit.')
      key = STRIN(pg,315,75,1,yesno)
      if (((yesno.eq.'y').or.(yesno.eq.'Y')).and.(key.eq.retk)) then
         chkop = 0
      else
         chkop = 1
      endif
      call WRTSTR(pg,40,60,51,
     +     '                                                    ')
      call WRTSTR(pg,40,75,32,'                                    ')
      call WRTSTR(pg,40,110,26,'                          ')
      call WRTSTR(pg,40,125,16,'                ')
      return
      end

C ***************************************************************************
CN    MODULE NAME      : SETZER
CA    FUNCTION         : To initialise a matrix of polynomials.
CS    CALL SEQUENCE    : call setzer(mat,val)
CI    INPUT PARAMETERS :
C                 val - (integer*2) The value to be put on the diagonal
C                       element.
C
CO    OUTPUT PARAMETERS:
C      mat(10,10,2,13) - (real*4) The matrix of polynomials.
C
CG    GLOBAL VARIABLES : None.
CM    MODULES CALLED   : None.
CE    ERROR CONDITIONS : ?
C
CC    COMMENTS         : Zeros the entire symbolic matrix, except the
C                        coefficient of the 0th power of the numerators on the
C                        diagonal, 'val' is placed in these locations.
C ***************************************************************************
      subroutine setzer(mat,val)
      real*4 mat(10,10,2,13),val
      integer*2 i,j,k
      do 300 i = 1,10
         do 301 j = 1,10
            do 302 k = 1,13
               mat(i,j,1,k) = 0.0
               mat(i,j,2,k) = 0.0
302         continue
            if (i.eq.j) then
               mat(i,j,1,1) = val
               mat(i,j,2,1) = val
            endif
301      continue
300   continue
```

```
      return
      end

C ********************************************************************************
CH    REVISION HISTORY :
C     VERSION       BY                DATE        COMMENT
C     1.00          Ian Fisher        23/06/88    Creation.
C     FILEND           :
```

```
C
C     FILE              : PIMAT.FOR
C ***************************************************************************
CN    MODULE NAME       : PIMAT
CA    FUNCTION          : To generate the initial stage of a matrix PI.
CS    CALL SEQUENCE     : call pimat()
CI    INPUT PARAMETERS  : None.
CO    OUTPUT PARAMETERS: None.
CG    GLOBAL VARIABLES  : sysmat.inc
C                         keys.inc
CM    MODULES CALLED    : WIPSCR (asm), WRTSTR (asm), wrtitl (for), setzer (for)
C                         INKEY (asm), getk0 (for), getki (for), dopi (for).
CE    ERROR CONDITIONS  : ?
C
CC    COMMENTS          : The main controlling routine for the generation of the
C                         matrix PI controller.
C                         The routine obtains the users permission to clear the
C                         work matrix and then starts the matrix generation.
C                         The inverse constant matrices at f=0 and f=infinity
C                         are then calculated and the symbolic PI matrix is
C                         generated. The resultant matrix is left in the work
C                         matrix.
C                         The inverse constant matrices are not scaled for
C                         optimum performance. This is left to the user.
C ***************************************************************************
      subroutine pimat()
      implicit integer*2 (c,I)
$include: 'sysmat.inc'
$include: 'keys.inc'
      integer*2 chk,i,j,key

      call wipscr(1)
      call wrtitl(1,150,20,21,'PI Matrix Definition.')
      chk = 0
      chk = chkop(1)
      if (chk.eq.0) then
         call setzer(k2mat,0.0)
         do 300 i = 1,100
            calcr(i) = 0.0
            calci(i) = 0.0
300      continue
         call WRTSTR(1,40,50,24,'Generating : K0 = 1/G(0)')
         call getk0(order,gmat,calcr)
         call WRTSTR(1,40,70,24,'Generating : Ki = 1/G(i)')
         call getki(order,gmat,calci)
         call WRTSTR(1,40,90,23,'Constructing PI matrix.')
         call dopi(order,k2mat,calcr,calci)
         call WRTSTR(1,40,110,9,'Finished.')
         call WRTSTR(1,40,140,22,'Hit RETURN to proceed.')
111      continue
         key = INKEY(1)
         if (key.ne.retk) goto 111
      else
         call WRTSTR(1,40,50,30,'PI Matrix operation cancelled.')
      endif
      call WIPSCR(1)
      return
      end

C ***************************************************************************
CN    MODULE NAME       : GETK0
CA    FUNCTION          : Generates K term of matrix PI controller.
CS    CALL SEQUENCE     : call getk0(n,mat,work)
CI    INPUT PARAMETERS :
C                    n - (integer*2) Order of the system.
C      mat(10,10,2,13) - (real*4) The matrix of polynomials.
C
CO    OUTPUT PARAMETERS:
C            work(n,n) - (real*4) The resultant constant matrix.
C
CG    GLOBAL VARIABLES : sysmat.inc
CM    MODULES CALLED   : cominv (for).
```

```
CE    ERROR CONDITIONS : ?
C
CC    COMMENTS         : Calculates the constant matrix from the system matrix
C                        ie. G(s), at f=0. The constant matrix is then
C                        inverted.
C ***************************************************************************
      subroutine getk0(n,mat,work)
      integer*2 n
      real*4 mat(10,10,2,13),work(n,n)
      integer*2 i,j
      real*4 num,den
      character*24 name
      complex*8 c,c2
$include: 'sysmat.inc'
      do 300 i = 1,100
         calcr2(i) = 0.0
300   continue
      do 99 i = 1,n
         do 199 j = 1,n
            num = mat(i,j,1,1)
            den = mat(i,j,2,1)
            if ((int(mat(i,j,2,13)).eq.0).and.(abs(den).lt.(1.0e-12)))
     +         then
                work(i,j) = num
            else
               work(i,j) = num/den
            endif
199      continue
99    continue
      name = 'test1.mat'
      c2 = (0.0,0.0)
      call cominv(n,work,calcr2,c)
      if (c.eq.c2) then
         call WRTSTR(1,270,50,37,
     +        '*** Warning : Matrix Inversion Error.')
         do 201 i = 1,10
            do 202 j = 1,10
               work(i,j) = 0.0
202         continue
201      continue
      endif
      return
      end

C ***************************************************************************
CN    MODULE NAME      : GETKI
CA    FUNCTION         : Generates I term of matrix PI controller.
CS    CALL SEQUENCE    : call getki(n,mat,work)
CI    INPUT PARAMETERS :
C                   n - (integer*2) Order of the system.
C      mat(10,10,2,13) - (real*4) The matrix of polynomials.
C
CO    OUTPUT PARAMETERS:
C            work(n,n) - (real*4) The resultant constant matrix.
C
CG    GLOBAL VARIABLES : sysmat.inc
CM    MODULES CALLED   : cominv (for).
CE    ERROR CONDITIONS : ?
C
CC    COMMENTS         : To find Ki, the following  operations are performed :
C
C                          i) Multiply each row, g(s),  of  the  system matrix
C                             ,G(s), by (s^p) ; where (p) is an  integer  such
C                             that as  |s| -> infinity no element of (s^p)g(s)
C                             tends to infinity and not every element tends to
C                             zero.
C
C                         ii) Now define the row vector,
C                                b = lim  (s^p)g(s)
C                                    (as s -> infinity)
C
C                        iii) Repeat (i) and (ii) for  all  the rows to create
```

```
C                              matrix B. Then  Ki = inverse(B).
C ***************************************************************************
      subroutine getki(n,mat,work)
      integer*2 n
      real*4 mat(10,10,2,13),work(n,n)
      integer*2 i,j,mindif,num,den,num2,den2
      character*24 name
      complex*8 c,c2
$include: 'sysmat.inc'
      do 300 i = 1,100
         calcr2(i) = 0.0
300   continue
      do 99 i = 1,n
         mindif = 20
         do 199 j = 1,n
            num = int(mat(i,j,1,13))
            den = int(mat(i,j,2,13))
            if ((num.eq.0).and.(abs(mat(i,j,1,1)).lt.(1.0e-12))) then
               num = 30
            else
               num = den-num
            endif
            if (num.lt.mindif) mindif = num
199      continue
         if (mindif.le.10) then
            do 299 j = 1,n
               den = int(mat(i,j,2,13))
               num = int(mat(i,j,1,13))
               if (mindif.ge.0) then
                  num2 = num + mindif
                  if (den.gt.num2) then
                     work(i,j) = 0.0
                  elseif (den.eq.num2) then
                     work(i,j) = mat(i,j,1,(num+1))/mat(i,j,2,(den+1))
                  else
                     work(i,j) = 0.0
                  endif
               else
                  den2= den + abs(mindif)
                  if (den2.gt.num) then
                     work(i,j) = 0.0
                  elseif (den2.eq.num) then
                     work(i,j) = mat(i,j,1,(num+1))/mat(i,j,2,(den+1))
                  else
                     work(i,j) = 0.0
                  endif
               endif
299         continue
         endif
99    continue
      name = 'test2.mat'
      c2 = (0.0,0.0)
      call cominv(n,work,calcr2,c)
      if (c.eq.c2) then
         call WRTSTR(1,270,70,37,
     +        '*** Warning : Matrix Inversion Error.')
         do 201 i = 1,10
            do 202 j = 1,10
               work(i,j) = 0.0
202         continue
201      continue
      endif
      return
      end

C ***************************************************************************
CN    MODULE NAME      : DOPI
CA    FUNCTION         : To combine the matrices from GETK0 and GETKI into
C                        the symbolic PI matrix.
CS    CALL SEQUENCE    : call dopi(n,mat,k0,ki)
CI    INPUT PARAMETERS :
C                   n - (integer*2) Order of the system.
```

```
C      mat(10,10,2,13) - (real*4) The resultant PI matrix of polynomials.
C              k0(n,n)
C             ki(n,n) - (real*4) The constant matrices generated previously.
C
CO    OUTPUT PARAMETERS:
C      mat(10,10,2,13) - (real*4) The resultant PI matrix of polynomials.
C
CG    GLOBAL VARIABLES : None.
CM    MODULES CALLED   : None.
CE    ERROR CONDITIONS : ?
C
CC    COMMENTS         : This routine generates the  symbolic PI matrix from
C                        the constant matrices  :  Ki  and K0. The PI matrix
C                        is left in the work matrix.
C
C
C ***************************************************************************
      subroutine dopi(n,mat,k0,ki)
      integer*2 n
      real*4 mat(10,10,2,13),k0(n,n),ki(n,n)
      integer*2 i,j
      do 99 i = 1,n
         do 98 j = 1,n
            if (abs(k0(i,j)).gt.(1.0e-12)) then
               mat(i,j,1,13) = 0.0
               mat(i,j,1,1)  = k0(i,j)
               mat(i,j,2,13) = 1.0
               mat(i,j,2,1)  = 0.0
               mat(i,j,2,2)  = 1.0
                if (abs(ki(i,j)).gt.(1.0e-12)) then
                   mat(i,j,1,13) = 1.0
                   mat(i,j,1,2)  = ki(i,j)
                endif
            elseif (abs(ki(i,j)).gt.(1.0-12)) then
               mat(i,j,1,13) = 0.0
               mat(i,j,1,1)  = ki(i,j)
               mat(i,j,2,13) = 0.0
               mat(i,j,2,1)  = 0.0
            else
               mat(i,j,1,13) = 0.0
               mat(i,j,1,1)  = 0.0
               mat(i,j,2,13) = 0.0
               mat(i,j,2,1)  = 0.0
            endif
98       continue
99    continue
      return
      end

C ***************************************************************************
CH    REVISION HISTORY :
C     VERSION       BY                DATE        COMMENT
C     1.00          Ian Fisher        23/06/88    Creation.
C     FILEND           :
```

```
C
C     FILE             : SYMULT.FOR
C ***************************************************************************
CN    MODULE NAME      : SYMULT
CA    FUNCTION         : To multiply, symbolically, two matrices.
CS    CALL SEQUENCE    : call symult(order,kmat,k2mat,dir,err)
CI    INPUT PARAMETERS :
C                 order - (integer*2) The order of the system.
C      kmat(10,10,2,13) - (real*4) The first matrix.
C     k2mat(10,10,2,13) - (real*4) The second matrix.
C                   dir - (integer*2) The multiply order value :
C                         1 : kmat = kmat X k2mat
C                         2 : kmat = k2mat X kmat
C
CO    OUTPUT PARAMETERS:
C      kmat(10,10,2,13) - (real*4) The resultant matrix.
C                   err - (integer*2) Error return :
C                         0 : no error.
C                         1 : Error - operation not complete.
C
CG    GLOBAL VARIABLES : k3mat.inc
CM    MODULES CALLED   : polmul (for).
CE    ERROR CONDITIONS : ?
C
CC    COMMENTS         : Steps through each element, calling routines to
C                        multiply the rows and columns of the two matrices.
C                        The routine does perform checks for error after each
C                        row/column multiply.
C ***************************************************************************
      subroutine symult(order,kmat,k2mat,dir,err)
      implicit integer*2 (p)
      integer*2 order
      real*4 kmat(10,10,2,13),k2mat(10,10,2,13)
      integer*2 dir,err
$include: 'k3mat.inc'
      real*4 tpoly(2,13)
      integer*2 i,j,err2
      err = 0
      err2 = 0
      do 333 i = 1,order
         do 334 j = 1,order
            do 335 k = 1,13
               k3mat(i,j,1,k) = 0.0
               k3mat(i,j,2,k) = 0.0
335         continue
334      continue
333   continue
      do 99 i = 1,order
         do 199 j = 1,order
            do 98 k = 1,13
               tpoly(1,k) = 0.0
               tpoly(2,k) = 0.0
98          continue
            do 97 k = 1,order
               if (dir.eq.1) then
                  err2 = polmul(i,k,kmat,k,j,k2mat,tpoly)
               else
                  err2 = polmul(k,j,kmat,i,k,k2mat,tpoly)
               endif
               if ((err2.ne.0).and.(err.eq.0)) then
                  err = err2
               endif
97          continue
            if (err2.eq.0) then
               do 96 k = 1,13
                  k3mat(i,j,1,k) = tpoly(1,k)
                  k3mat(i,j,2,k) = tpoly(2,k)
96             continue
            else
               k3mat(i,j,1,1) = 0.0
               k3mat(i,j,2,1) = 0.0
               k3mat(i,j,1,13) = 0.0
```

```
               k3mat(i,j,2,13) = 0.0
            endif
            err2 = 0
199      continue
99    continue
      do 433 i = 1,order
         do 434 j = 1,order
            do 435 k = 1,13
               kmat(i,j,1,k) = k3mat(i,j,1,k)
               kmat(i,j,2,k) = k3mat(i,j,2,k)
435         continue
434      continue
433   continue
      return
      end

C ***************************************************************************
CN    MODULE NAME       : POLMUL
CA    FUNCTION          : To muliply a two polynomials and add to a second
C                         polynomial.
CS    CALL SEQUENCE     : err = polmul(row1,col1,kmat,row2,col2,k2mat,temp)
CI    INPUT PARAMETERS :
C                 row1
C                 col1 - (integer*2) Position of first polynomial.
C     kmat(10,10,2,13) - (real*4) The first matrix of polynomials.
C                 row2
C                 col2 - (integer*2) Position of second polynomial.
C    k2mat(10,10,2,13) - (real*4) The second matrix of polynomials.
C           temp(2,13) - (real*4) The second 'running' total polynomial.
C
CO    OUTPUT PARAMETERS:
C                  err - (integer*2) The error return :
C                        0 : no error.
C                        1 : error - operation not complete.
C           temp(2,13) - (real*4) The second 'running' total polynomial.
C
CG    GLOBAL VARIABLES : None.
CM    MODULES CALLED   : pmult (for), polzer (for).
CE    ERROR CONDITIONS :
C
CC    COMMENTS         : Multiplies two polynomials together and adds the
C                        resultant polynomial to a 'running' total type
C                        polynomial.
C ***************************************************************************
      integer*2 function polmul(row1,col1,kmat,row2,col2,k2mat,temp)
      implicit integer*2 (I,p)
      integer*2 row1,col1
      real*4 kmat(10,10,2,13)
      integer*2 row2,col2
      real*4 k2mat(10,10,2,13)
      real*4 temp(2,13)

      real*4 t1poly(13),t2poly(13),t3poly(13)
      integer*2 i,g1ord,g2ord,err,key

c *** kn1*kn2
      g1ord = int(kmat(row1,col1,1,13))
      g2ord = int(k2mat(row2,col2,1,13))
      do 98 i = 0,g1ord
         t1poly(i+1) = kmat(row1,col1,1,(i+1))
98    continue
      t1poly(13) = real(g1ord)
      t1poly(12) = kmat(row1,col1,1,12)
      do 97 i = 0,g2ord
         t2poly(i+1) = k2mat(row2,col2,1,(i+1))
97    continue
      t2poly(13) = real(g2ord)
      t2poly(12) = k2mat(row2,col2,1,12)
      err = 0
      err = pmult(t1poly,t2poly)
c *** kd1*kd2
      if (err.eq.0) then
```

```
         g1ord = int(kmat(row1,col1,2,13))
         g2ord = int(k2mat(row2,col2,2,13))
         do 96 i = 0,g1ord
            t2poly(i+1) = kmat(row1,col1,2,(i+1))
96       continue
         t2poly(13) = real(g1ord)
         t2poly(12) = kmat(row1,col1,2,12)

         do 95 i = 0,g2ord
            t3poly(i+1) = k2mat(row2,col2,2,(i+1))
95       continue
         t3poly(13) = real(g2ord)
         t3poly(12) = k2mat(row2,col2,2,12)
         err = pmult(t2poly,t3poly)
      endif
c*******************************
      if (err.eq.0) then
         if (polzer(t1poly).ne.0) then
            if (polzer(t2poly).eq.0) then
               t2poly(1) = 1.0
               t2poly(13) = 0.0
            endif
            do 88 i = 1,13
               t3poly(i) = temp(1,i)
88          continue
            if (polzer(t3poly).eq.0) then
               do 79 i = 1,13
                  temp(1,i) = t1poly(i)
                  temp(2,i) = t2poly(i)
79             continue
            else
               do 87 i = 1,13
                  t3poly(i) = temp(2,i)
87             continue
               if (polzer(t3poly).eq.0) then
                  temp(2,1) = 1.0
                  temp(2,13) = 0.0
               endif
c *********    nold*kd1
               g1ord = int(temp(1,13))
               do 94 i = 0,g1ord
                  t3poly(i+1) = temp(1,(i+1))
94             continue
               t3poly(13) = real(g1ord)
               err = pmult(t3poly,t2poly)
               t3poly(12) = temp(1,12)
               if (err.eq.0) then
                  do 93 i = 1,13
                     temp(1,i) = t3poly(i)
93                continue
c *********       dold*kn1
                  g1ord = int(temp(2,13))
                  do 92 i = 0,g1ord
                     t3poly(i+1) = temp(2,(i+1))
92                continue
                  t3poly(13) = real(g1ord)
                  t3poly(12) = temp(2,12)
                  err = pmult(t1poly,t3poly)
               endif
c *********    (nold*k1) + (dold*kn1)
               if (err.eq.0) then
                  do 91 i = 1,12
                     temp(1,i) = temp(1,i) + t1poly(i)
91                continue
                  if (int(temp(1,13)).lt.int(t1poly(13))) then
                     temp(1,13) = int(t1poly(13))
                  endif
c *********       dold*kd1
                  g1ord = int(temp(2,13))
                  do 90 i = 0,g1ord
                     t3poly(i+1) = temp(2,(i+1))
90                continue
```

```
                  t3poly(13) = real(g1ord)
                  t3poly(12) = temp(2,12)
                  err = pmult(t3poly,t2poly)
               endif
               if (err.eq.0) then
                  do 89 i = 1,13
                     temp(2,i) = t3poly(i)
89                continue
               endif
            endif
         endif
      endif
      polmul = err
      return
      end

C ***************************************************************************
CN    MODULE NAME       : PMULT
CA    FUNCTION          : Multiplies two polynomials.
CS    CALL SEQUENCE     : err = pmult(poly1,poly2)
CI    INPUT PARAMETERS :
C           poly1(13) - (real*4) Polynomial 1.
C           poly2(13) - (real*4) Polynomial 2.
C
CO    OUTPUT PARAMETERS:
C           poly1(13) - (real*4) Polynomial 1 - resultant.
C                 err - (integer*2) The error return :
C                       0 : no error.
C                       1 : error - operation not complete.
CG    GLOBAL VARIABLES : None.
CM    MODULES CALLED   : None.
CE    ERROR CONDITIONS : ?
C
CC    COMMENTS         : Multiplies and adds the coefficients of the two
C                        polynomials in the standard order.
C
C ***************************************************************************
      integer*2 function pmult(poly1,poly2)
      real*4 poly1(13),poly2(13)

      real*4 temp(11)
      integer*2 g1ord,g2ord,i,j
      g1ord = int(poly1(13))
      g2ord = int(poly2(13))
      if ((g1ord.lt.0).or.(g2ord.lt.0)) then
         pmult = 1
      elseif ((g1ord+g2ord).gt.10) then
         pmult = 2
      else
         do 98 i = 1,11
            temp(i) = 0.0
98       continue
         do 99 i = 0,g1ord
            do 199 j = 0,g2ord
               temp(i+j+1) = temp(i+j+1) + poly1(i+1)*poly2(j+1)
199         continue
99       continue
         do 198 i = 1,11
            poly1(i) = temp(i)
198      continue
         poly1(12) = poly1(12)+poly2(12)
         poly1(13) = real(g1ord+g2ord)
         pmult = 0
      endif
      return
      end
C ***************************************************************************
CN    MODULE NAME       : POLZER
CA    FUNCTION          : Checks if a polynomial is zero.
CS    CALL SEQUENCE     : chk = polzer(poly)
CI    INPUT PARAMETERS :
C           poly(13) - (real*4) Polynomial.
```

```
     +**************************')
        endif
      endif
      if (chkio.eq.0) goto 201
200   continue
      call ERTONE()
201   continue
      close(1)
      return
      end

C ******************************************************************************
CN    MODULE NAME       : PRHEAD
CA    FUNCTION          : To print a header at the top the page.
CS    CALL SEQUENCE     : call prhead(n,prpg,freq,ftype)
CI    INPUT PARAMETERS :
C                   n - (integer*2) Order of teh system.
C                prpg - (integer*2) The page count.
C                freq - (real*4) Frequency of evaluation.
C               ftype - (real*4) Frequency unit type.
C
CO    OUTPUT PARAMETERS: None.
CG    GLOBAL VARIABLES : None.
CM    MODULES CALLED   : None.
CE    ERROR CONDITIONS : ?
C
CC    COMMENTS          : The routine sends a formfeed to the printer and then
C                         prints the header.
C ******************************************************************************
      subroutine prhead(n,prpg,freq,ftype)
      integer*2 n,prpg
      real*4 freq
      integer*2 ftype

      integer*2 chkio
      write(1,'(a)',iostat=chkio,err=200) '1'
      write(1,100,iostat=chkio,err=200) ' Eigen System at F= '
      if (ftype.eq.1) then
         freq2 = freq/6.2831853
         write(1,101,iostat=chkio,err=200) freq
         write(1,100,iostat=chkio,err=200) '  [Hz]     '
      elseif (ftype.eq.2) then
         freq2 = freq
         write(1,101,iostat=chkio,err=200) freq
         write(1,100,iostat=chkio,err=200) '  [Rad/Sec]'
      else
         freq2 = freq/3600.0
         write(1,101,iostat=chkio,err=200) freq
         write(1,100,iostat=chkio,err=200) '  [Rad/Hr] '
      endif
      write(1,100,iostat=chkio,err=200) '   System order : '
      write(1,102,iostat=chkio,err=200) n
      write(1,100,iostat=chkio,err=200) '     Page : '
      write(1,102,iostat=chkio,err=200) prpg
      write(1,'(a)',iostat=chkio,err=200) ' '
      write(1,100,iostat=chkio,err=200)
     +' ----------------------------------------'
      write(1,100,iostat=chkio,err=200)
     +'   -------------------'
      write(1,100,iostat=chkio,err=200)
     +'     -----------'
      write(1,99,iostat=chkio,err=200)  ' '
200   continue
99    format(a)
100   format(a,\)
101   format(g10.4,\)
102   format(i4,\)
      return
      end

C ******************************************************************************
CN    MODULE NAME       : PRVECT
```

```
CA    FUNCTION         : Prints a single eigenvalue and it's associated
C                        eigenvector (and inverse).
CS    CALL SEQUENCE    : call prvect(el,n,alfr,alfi,zr,zi,calcr2,calci2)
CI    INPUT PARAMETERS :
C                   el - (integer*2) The element to be printed.
C                    n - (integer*2) Order of teh system.
C             alfr(n,n)
C             alfi(n,n) - (real*4) Eigenvalues.
C               zr(n,n)
C               zi(n,n) - (real*4) Eigenvectors.
C           calcr2(n,n)
C           calci2(n,n) - (real*4) Inverse eigenvectors.
C
CO    OUTPUT PARAMETERS: None.
CG    GLOBAL VARIABLES : None.
CM    MODULES CALLED   : None.
CE    ERROR CONDITIONS : ?
C
CC    COMMENTS         : Sends the data to the printer unit.
C ************************************************************************
      subroutine prvect(el,n,alfr,alfi,zr,zi,calcr2,calci2)
      integer*2 el,n
      real*4 alfr(n),alfi(n)
      real*4 zr(n,n),zi(n,n)
      real*4 calcr2(n,n),calci2(n,n)

      integer*2 chkio,i
      write(1,99,iostat=chkio,err=200)  ' '
      write(1,10,iostat=chkio,err=200)  el,alfr(el),alfi(el)
10    format(' Column(',i2,')  :  Eigenvalue : Real = ',g10.4,
     +       '   Imag = ',g10.4)
      write(1,99,iostat=chkio,err=200)  ' '
      write(1,11,iostat=chkio,err=200)
      write(1,13,iostat=chkio,err=200)
11    format(' Eigenvector :    Real        Imag      Inverse :    Real
     +       Imag')
13    format(' -------------    ----        ----      ---------    ----
     +       ----')
      do 998 i = 1,n
         write(1,12,iostat=chkio,err=200)  zr(i,el),zi(i,el),
     +                                     calcr2(i,el),calci2(i,el)
12       format('               ',g10.4,'  ',g10.4,'                ',
     +          g10.4,'  ',g10.4)
998   continue
      write(1,99,iostat=chkio,err=200)  ' '
200   continue
99    format(a)
100   format(a,\)
101   format(g10.4,\)
102   format(i4,\)
      return
      end

C ************************************************************************
CH    REVISION HISTORY :
C     VERSION       BY                DATE        COMMENT
C     1.00          Ian Fisher        23/06/88    Creation.
C     FILEND           :
```

```
$include: 'lst.inc'
C
C     FILE              : VECTOR.FOR
C ***************************************************************************
CN    MODULE NAME       : VECTOR
CA    FUNCTION          : To present the eigenvalues and vectors of the system
C                         matrix at some frequency to the user.
CS    CALL SEQUENCE     : call vector()
CI    INPUT PARAMETERS  : None.
CO    OUTPUT PARAMETERS: None.
CG    GLOBAL VARIABLES  : keys.inc
C                         sysmat.inc
C                         plotpg.inc
CM    MODULES CALLED    : WIPSCR (asm), GPAGE (asm), DISP (asm), MOVE (asm),
C                         DLINE (asm), wrtitl (for), WRTSTR (asm), prtnum (for),
C                         getnum (for), evpoly (for), qzveca (for), cominv (for)
C                         prvecs (for), INKEY (asm), blkel (for), prtmat (for).
CE    ERROR CONDITIONS  : ?
C
CC    COMMENTS          : This routine calculates and presents the eigenvalues
C                         and eigenvectors (and inverse eigenvectors) of the
C                         system matrix, G(s), at some particular frequency. The
C                         frequency is specified by the user.
C                         Only one eigenvalue and it's associated vector (and
C                         it's inverse) can be displayed at once. But the entire
C                         set can be printed out on a printer.
C                         The ESC key returns control to the calling routine.
C ***************************************************************************
      subroutine vector()
      implicit integer*2 (I,g)
$include: 'keys.inc'
$include: 'sysmat.inc'
$include: 'plotpg.inc'

      integer*2 xpos,ypos,pos,key,pos2,materr,ctrlt
      real*4 freq
      complex*8 c,c2
      ctrlt = 5120
      call WIPSCR(1)
      call GPAGE(1)
      call DISP(1)
      call MOVE(270,4)
      call DLINE(270,316)
      call wrtitl(1,20,20,26,'Eigen-vectors and -values.')
      call WRTSTR(1,20,40,6,'Freq =')
      call WRTSTR(1,20,283,27,'RETURN, TAB and cursor keys')
      call WRTSTR(1,20,297,21,'to move. ESC to quit.')
      call WRTSTR(1,20,311,25,'CTRL-T to print matrices.')
      xpos = 20
      ypos = 60
      call WRTSTR(1,xpos,ypos,12,'Row Number :')
      xpos = xpos + 13*9
      pos = 1
      freq = 1.0
      do 100 i = 1,order
         call prtnum(1,xpos,((i-1)*15+ypos),1,4,'(i2)',2,i)
100   continue
      call wrtitl(1,422,20,4,'Real')
      call wrtitl(1,530,20,4,'Imag')
      call WRTSTR(1,280,36,12,'Eigenvalue :')
      call wrtitl(1,328,56,11,'Eigenvector')
      call wrtitl(1,567,56,7,'Inverse')
      call wrtitl(1,306,73,4,'Real')
      call wrtitl(1,414,73,4,'Imag')
      call wrtitl(1,527,73,4,'Real')
      call wrtitl(1,635,73,4,'Imag')
      do 101 i = 1,100
         calcr(i) = 0.0
         calci(i) = 0.0
         calcr2(i) = 0.0
         calci2(i) = 0.0
         zr(i) = 0.0
```

```
         zi(i) = 0.0
101   continue
      do 102 i = 1,10
         alfr(i) = 0.0
         alfi(i) = 0.0
102   continue
      call prvecs(1,order,alfr,alfi,zr,zi,calcr2,calci2)
      call prfreq(ftype,freq)
99    continue
      if (pos.eq.1) then
         if (ftype.eq.1) then
            freq = (freq/6.2831853)
         elseif (ftype.eq.2) then
            freq = freq
         else
            freq = (freq/3600.0)
         endif
91       continue
         key = getnum(1,83,40,3,7,'(g10.4)',10,freq)
         if (freq.lt.(0.0)) call ERTONE()
         if (freq.lt.(0.0)) goto 91
         if (ftype.eq.1) then
            freq = (freq*6.2831853)
         elseif (ftype.eq.2) then
            freq = freq
         else
            freq = (freq*3600.0)
         endif
         if (key.ne.esck) then
            call evpoly(freq,order,gmat,calcr,calci)
            call qzveca(order,calcr,calci,alfr,alfi,zr,zi,materr)
            if (materr.eq.0) then
               do 200 i = 1,100
                  calcr2(i) = zr(i)
                  calci2(i) = zi(i)
200            continue
               c2 = (0.0,0.0)
               call cominv(order,calcr2,calci2,c)
               if (c2.eq.c) then
                  do 202 i = 1,100
                     calcr2(i) = 0.0
                     calci2(i) = 0.0
202               continue
               endif
            else
               do 201 i = 1,100
                  calcr2(i) = 0.0
                  calci2(i) = 0.0
201            continue
            endif
         endif
      else
         call prvecs((pos-1),order,alfr,alfi,zr,zi,calcr2,calci2)
         key = INKEY(1)
      endif
      if ((key.eq.retk).or.(key.eq.tabk).or.(key.eq.downk)) then
         if (pos.gt.1) call blkel(pos)
         pos = pos + 1
         if (pos.gt.(order+1)) then
            pos = 1
         else
            call blkel(pos)
         endif
      elseif ((key.eq.rtabk).or.(key.eq.upk)) then
         if (pos.gt.1) call blkel(pos)
         pos = pos - 1
         if (pos.lt.1) then
            pos = order+1
            call blkel(pos)
         elseif (pos.gt.1) then
            call blkel(pos)
         endif
```

```
      elseif (key.eq.ctrlt) then
         call WRTSTR(1,20,311,25,'CTRL-Q to quit printing. ')
         call prtmat(order,alfr,alfi,zr,zi,calcr2,calci2,freq,ftype)
         call WRTSTR(1,20,311,25,'CTRL-T to print matrices.')
      endif
      if (key.ne.esck) goto 99
      call WIPSCR(1)
      return
      end

C ***************************************************************************
CN    MODULE NAME       : PRVECS
CA    FUNCTION          : To print the eigen-vector and inverse on graphics
C                         page 1.
CS    CALL SEQUENCE     : call prvecs(el,n,alfr,alfi,zr,zi,calcr2,calci2)
CI    INPUT PARAMETERS :
C                  el - (integer*2) The element to print.
C                   n - (integer*2) The order of the system.
C             alfr(n)
C             alfi(n) - (real*4) The eigenvalues.
C             zr(n,n)
C             zi(n,n) - (real*4) The eigenvectors.
C          calcr2(n,n)
C          calci2(n,n) - (real*4) The inverse of the eigenvectors.
C
CO    OUTPUT PARAMETERS: None.
CG    GLOBAL VARIABLES : None.
CM    MODULES CALLED   : prtnum (for).
CE    ERROR CONDITIONS : ?
C
CC    COMMENTS          : Prints the real and imaginary parts of the eigenvalue,
C                         the associated eigenvector and the inverse on graphics
C                         page 1.
C ***************************************************************************
      subroutine prvecs(el,n,alfr,alfi,zr,zi,calcr2,calci2)
      integer*2 el,n
      real*4 alfr(n),alfi(n)
      real*4 zr(n,n),zi(n,n)
      real*4 calcr2(n,n),calci2(n,n)

      integer*2 i,ypos
      ypos = 89
      call prtnum(1,395,36,3,7,'(g10.4)',10,alfr(el))
      call prtnum(1,503,36,3,7,'(g10.4)',10,alfi(el))
      do 99 i = 1,n
         call prtnum(1,279,(ypos+(i-1)*16),3,7,'(g10.4)',10,
     +               zr(i,el))
         call prtnum(1,387,(ypos+(i-1)*16),3,7,'(g10.4)',10,
     +               zi(i,el))
         call prtnum(1,500,(ypos+(i-1)*16),3,7,'(g10.4)',10,
     +               calcr2(i,el))
         call prtnum(1,608,(ypos+(i-1)*16),3,7,'(g10.4)',10,
     +               calci2(i,el))
99    continue
      return
      end

C ***************************************************************************
CN    MODULE NAME       : PRFREQ
CA    FUNCTION          : To print the frequency (including units).
CS    CALL SEQUENCE     : call prfreq(ftype,freq)
CI    INPUT PARAMETERS :
C               ftype - (integer*2) Frequency type :
C                       1 : Hertz
C                       2 : Rads/Second
C                       3 : Rads/Hour
C                freq - (real*4) The frequency.
C
CO    OUTPUT PARAMETERS: None.
CG    GLOBAL VARIABLES : None.
CM    MODULES CALLED   : WRTSTR (asm), prtnum (for).
CE    ERROR CONDITIONS : ?
```

```
C
CC    COMMENTS          : To print the frequency and associated units on page 1.
C                         The frequency is always  stored  in  rad/sec, thus the
C                         value is adjusted before printing.
C ***************************************************************************
      subroutine prfreq(ftype,freq)
      integer*2 ftype
      real*4 freq
      if (ftype.eq.1) then
         call WRTSTR(1,183,40,9,'[Hz]      ')
         call prtnum(1,83,40,3,7,'(g10.4)',10,(freq/(6.2831853)))
      elseif (ftype.eq.2) then
         call WRTSTR(1,183,40,9,'[Rad/Sec]')
         call prtnum(1,83,40,3,7,'(g10.4)',10,freq)
      else
         call WRTSTR(1,183,40,9,'[Rad/Hr] ')
         call prtnum(1,83,40,3,7,'(g10.4)',10,(freq/3600.0))
      endif
      return
      end

C ***************************************************************************
CN    MODULE NAME       : BLKEL
CA    FUNCTION          : To back highlight an element number.
CS    CALL SEQUENCE     : call blkel(el)
CI    INPUT PARAMETERS  :
C                   el - (integer*2) The element number.
C
CO    OUTPUT PARAMETERS: None.
CG    GLOBAL VARIABLES  : None.
CM    MODULES CALLED    : LEVEL (asm), BLKFIL (asm).
CE    ERROR CONDITIONS  : ?
C
CC    COMMENTS          : Back highlights an element number as the user steps
C                         through the eiegnvalues and eigenvectors.
C ***************************************************************************
      subroutine blkel(el)
      integer*2 el
      call LEVEL(2)
      call BLKFIL(137,(61+(el-2)*15),18,14)
      call LEVEL(1)
      return
      end

C ***************************************************************************
CH    REVISION HISTORY :
C     VERSION       BY                  DATE        COMMENT
C     1.00          Ian Fisher          23/06/88    Creation.
C     FILEND           :
```

```
C
C     FILE              : SIMUL.FOR
C ***************************************************************************
CN    MODULE NAME       : SIMUL
CA    FUNCTION          : To 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    : wrtitl (for), DOMENU (asm), stpopt (for), simscl (for),
C                         dosim (for), newmat (for), WIPSCR (asm).
CE    ERROR CONDITIONS  : ?
C
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 wrtitl(0,250,35,23,'System Time Simulation.')
      opt14 = DOMENU(14)
      if (opt14.eq.1) then
         call stpopt()
      elseif (opt14.eq.2) then
         call simscl()
      elseif (opt14.eq.3) then
         call dosim()
      elseif (opt14.eq.4) then
         call newmat()
      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  : time.inc
C                         sysmat.inc
C                         keys.inc
CM    MODULES CALLED    : inisim (for), simgrf (for), simat (for), prtnum (for),
C                         pltsim (for), chkqt (for), WRTSTR (asm),  INKEY (asm),
C                         WIPSCR (asm), dwinps (for).
CE    ERROR CONDITIONS  : ?
C
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)
$include: 'time.inc'
$include: 'sysmat.inc'
$include: 'keys.inc'

      integer*2 i,key,flush,err,delmax
      real*4 ti2
      delmax = (12000 - mod(12000,(order*order)))/(order*order)
      call inisim()
      err = inidly(delmax)
      if (err.eq.0) then
         call simgrf()
         call dwinps()
```

```
      flush = INKEY(2)

      do 600 i = 1,order
         esim(i) = 0.0
600   continue
      call simat(order,delmax,gmat,gx,ysim,yptr,esim,yout,dt)
      call simat(order,delmax,kmat,kx,usim,uptr,esim,uout,dt)
      err = inidly(delmax)
      ti2 = -1.0
799   continue
        call prtnum(1,75,(thstrt-thmax-18),3,7,'(g10.4)',10,ti)
        do 701 i = 1,5
           if ((ti.ge.stpdat(i)).and.(ti2.lt.stpdat(i))) then
              if ((stpinp(i).gt.0).and.(stpinp(i).le.order)) then
                 rsim(stpinp(i)) = rsim(stpinp(i)) + stpinc(i)
              endif
           endif
701     continue
        if (cloop.eq.0) then
           do 800 i = 1,order
              esim(i) = rsim(i) - yout(i)
800        continue
        else
           do 801 i = 1,order
              esim(i) = rsim(i)
801        continue
        endif
        if (coninc.eq.0) then
           call simat(order,delmax,kmat,kx,usim,uptr,esim,uout,dt)
c           call pltsim(order,ti,uout)
           do 802 i = 1,order
              esim(i) = uout(i)
c              if (esim(i).gt.(4095.0)) then
c                 esim(i) = 4095.0
c              elseif (esim(i).lt.(0.0)) then
c                 esim(i) = 0.0
c              endif
802        continue
        endif
        call simat(order,delmax,gmat,gx,ysim,yptr,esim,yout,dt)
        call pltsim(order,ti,yout)
        ti2 = ti
        ti = ti + dt
        key = INKEY(4)
        if (key.ne.0) then
           call chkqt(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 (((ti-0.5*dt).lt.tend).and.(key.eq.0)) goto 799
      call WRTSTR(1,10,(thstrt-thmax-18),34,
     +           'Alternate (F3) page lists outputs.')
      call WRTSTR(1,10,(thstrt-thmax-4),31,
     +           'Hit ESC to quit.               ')
710   continue
      key = INKEY(1)
      if (key.ne.esck) goto 710
      call WIPSCR(1)
      call WIPSCR(0)
    endif
    return
    end

C ***************************************************************************
CN    MODULE NAME      : SIMAT
CA    FUNCTION         : To generate the outputs at (t+1).
CS    CALL SEQUENCE    : call simat(n,mat,mx,in,out,dt)
CI    INPUT PARAMETERS :
C                  n - (integer*2) Order of the system.
```

```
C      mat(10,10,2,13) - (real*4) The matrix of polynomials.
C       mx(10,10,2,11) - (real*4) The time simulation states for 'mat'.
C              in(10) - (real*4) The matrix inputs.
C                  dt - (real*4) The time step.
C
CO    OUTPUT PARAMETERS:
C          out(10,200) - (real*4) The matrix outputs.
C       mx(10,10,2,11) - (real*4) The time simulation states for 'mat'.
C
CG    GLOBAL VARIABLES : None.
CM    MODULES CALLED   : nstep (for), dodly (for).
CE    ERROR CONDITIONS : ?
C
CC    COMMENTS         : Accumalates the outputs of each polynomial simulation
C                        and adds the output to the appropriate matrix output.
C ***************************************************************************
      subroutine simat(n,max,mat,mx,sim,ptr,in,out,dt)
      integer*2 n,max
      real*4 mat(10,10,2,13),mx(10,10,2,11)
      real*4 sim(n,n,max)
      integer*2 ptr(10,10,2)
      real*4 in(10),out(10),dt

      real*4 nout
      integer*2 i,j
      do 700 i = 1,n
         out(i) = 0.0
         do 701 j = 1,n
            nout = 0.0
            call nstep(mat,mx,i,j,in(j),nout)
            call dodly(n,max,i,j,sim,ptr,nout)
            out(i) = out(i) + nout
701      continue
700   continue
      return
      end

C ***************************************************************************
CN    MODULE NAME      : NSTEP
CA    FUNCTION         : Simulate one time step.
CS    CALL SEQUENCE    : call nstep(mat,mx,row,col,in,out)
CI    INPUT PARAMETERS :
C      mat(10,10,2,13) - (real*4) The matrix of polynomials.
C       mx(10,10,2,11) - (real*4) The states of the time simulation.
C             row,col - (integer*2) The element of the canal to simulate.
C                  in - (real*4) The input to the element.
C                 out - (real*4) The output of the element.
C
CO    OUTPUT PARAMETERS:
C       mx(10,10,2,11) - (real*4) The states of the time simulation.
C                  out - (real*4) The output of the element.
C
CG    GLOBAL VARIABLES : time.inc
CM    MODULES CALLED   : None.
CE    ERROR CONDITIONS : ?
C
CC    COMMENTS         : The routine performs the simulation by breaking the
C                        polynomial down into a series of first order
C                        integrations. The first order integrations are
C                        achieved using the 4th order Runga-Kutta method.
C
C ***************************************************************************
      subroutine nstep(mat,mx,row,col,in,out)
      real*4 mat(10,10,2,13),mx(10,10,2,11)
      integer*2 row,col
      real*4 in,out
$include: 'time.inc'
      integer*2 nord,dord
      real*4 sx(11),dx(11)
      integer*2 i,j

      nord = mat(row,col,1,13)
```

```
      dord = mat(row,col,2,13)

c *** y = g(x,u) equations.
c ***     Equation for g(s).
      out = 0.0
      if ((dord.eq.0).and.(abs(mat(row,col,2,1)).lt.(1.0e-9))) then
         out = 0.0
      else
         do 96 i = 0,nord
            out = out + mat(row,col,1,(i+1))*mx(row,col,1,(i+1))
96       continue
      endif

      do 93 i = 1,dord
         mx(row,col,2,i) = mx(row,col,1,i)
         sx(i) = 0.0
         dx(i) = 0.0
93    continue

      if ((dord.eq.0).and.(abs(mat(row,col,2,1)).lt.(1.0e-9))) then
         mx(row,col,1,(dord+1)) = 0.0
      else
         mx(row,col,1,(dord+1)) = 0.0
         do 97 i = 1,dord
            mx(row,col,1,(dord+1)) =
     +      mx(row,col,1,(dord+1)) - mat(row,col,2,i)*mx(row,col,1,i)
97       continue
         if (abs(mat(row,col,2,(dord+1))).gt.(1.0e-12)) then
            mx(row,col,1,(dord+1)) = (mx(row,col,1,(dord+1))+in)/
     +                               mat(row,col,2,(dord+1))
         endif
      endif
c *** dx = f(x,u) Equations.
      do 92 i = 1,4
c ***    equations for g(s).
         if (dord.gt.0) then
            dx(dord) = 0.0
            do 91 j = 1,dord
               dx(dord) = dx(dord) - mat(row,col,2,j)*mx(row,col,1,j)
91          continue
            if (abs(mat(row,col,2,(dord+1))).gt.(1.0e-12)) then
               dx(dord) = (dx(dord)+in)/mat(row,col,2,(dord+1))
            endif
         endif
         if (dord.gt.1) then
            do 90 j = 1,(dord-1)
               dx(j) = mx(row,col,1,(j+1))
90          continue
         endif
c ***    Runga-Kutta
         do 87 j = 1,dord
            sx(j) = sx(j)+c1(i)*dx(j)
            if (i.lt.4) then
               mx(row,col,1,j) = mx(row,col,2,j) + c2(i)*dx(j)*dt
            elseif (i.eq.4) then
               mx(row,col,1,j) = mx(row,col,2,j) + sx(j)*dt/6.0
            endif
87       continue
92    continue

      return
      end

C ********************************************************************************
CN    MODULE NAME      : SIMGRF
CA    FUNCTION         : To plot all time simulation axes.
CS    CALL SEQUENCE    : call simgrf()
CI    INPUT PARAMETERS : None.
CO    OUTPUT PARAMETERS: None.
CG    GLOBAL VARIABLES : sysmat.inc
C                        time.inc
C                        sysnms.inc
```

```
CM    MODULES CALLED   : WIPSCR (asm), wrtitl (for), prtnum (for), WRTSTR (asm)
C                        LENSTR (asm),  LEVEL (asm),  MOVE (asm),  DLINE (asm),
C                        dwaxes (for), dwnumb (for).
CE    ERROR CONDITIONS : ?
C
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: 'sysmat.inc'
$include: 'time.inc'
$include: 'sysnms.inc'
      integer*2 i,j,k,len,chkio,xpos,ypos
      character*10 tstr

      xpos = 40
      ypos = 50
      call WIPSCR(0)
      call wrtitl(0,200,17,17,'System Parameters')
      call wrtitl(0,40,35,3,'No.')
      call wrtitl(0,100,35,5,'Input')
      call wrtitl(0,350,35,6,'Output')
      do 800 i = 1,order
         call prtnum(0,xpos,ypos,1,4,'(i2)',2,i)
         call WRTSTR(0,(xpos+60),ypos,25,outnms(i))
         call WRTSTR(0,(xpos+310),ypos,25,inpnms(i))
         ypos = ypos + 15
800   continue

      call DISP(1)
      call WIPSCR(1)

      twmax   = 711
      twstrt = 4
      thmax   = 270
      thstrt = 316
      call wrtitl(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 = 5
      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
            call dwaxes(1,1,1,xtlim(1),ytlim(1,i),tgraph,
     +                  tscale(1,i),tcentr(1,i))
            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          : To initialise the states and arrays for the time
C                         simulation.
CS    CALL SEQUENCE     : call inisim()
CI    INPUT PARAMETERS  : None.
CO    OUTPUT PARAMETERS: None.
CG    GLOBAL VARIABLES  : time.inc
C                         sysmat.inc
CM    MODULES CALLED    : None.
CE    ERROR CONDITIONS  : ?
C
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: 'sysmat.inc'
      integer*2 i,j,k
      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
      do 600 i = 1,order
         do 601 j = 1,order
            do 602 k = 1,11
               gx(i,j,1,k) = 0.0
               gx(i,j,2,k) = 0.0
               kx(i,j,1,k) = 0.0
               kx(i,j,2,k) = 0.0
602         continue
601      continue
600   continue

      xtlim(1) = tend
      xtlim(2) = 0.0
      xtlim(3) = 0.0
```

```
      do 88 i = 1,order
         esim(i) = 0.0
         rsim(i) = 0.0
88    continue

      do 87 i = 1,12000
         usim(i) = 0.0
         ysim(i) = 0.0
87    continue

      ti = 0.0
      return
      end

C ***************************************************************************
CN    MODULE NAME       : PLTSIM
CA    FUNCTION          : To plot the outputs from the system time 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(10) - (real*4) The system outputs.
C
CO    OUTPUT PARAMETERS: None.
CG    GLOBAL VARIABLES : time.inc
CM    MODULES CALLED   : plotpt (for).
CE    ERROR CONDITIONS : ?
C
CC    COMMENTS          : Plots each set of points on the resepective set of
C                         axes.
C ***************************************************************************
      subroutine pltsim(n,t,y)
      integer*2 n
      real*4 t,y(10)
$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,y(i),cent,scal)
399   continue
      return
      end

C ***************************************************************************
CN    MODULE NAME       : CHKQT
CA    FUNCTION          : To check if the user wishes to quit the 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 : time.inc
C                        keys.inc
CM    MODULES CALLED   : INKEY (asm), WRTSTR (asm), STRIN (asm).
CE    ERROR CONDITIONS : ?
C
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 ********************************************************************************
CN    MODULE NAME       : DWINPS
CA    FUNCTION          : To plot the inputs to the system.
CS    CALL SEQUENCE     : call dwinps()
CI    INPUT PARAMETERS  : None.
CO    OUTPUT PARAMETERS: None.
CG    GLOBAL VARIABLES  : sysmat.inc
C                         time.inc
CM    MODULES CALLED    : drawpt (for).
CE    ERROR CONDITIONS  : ?
C
CC    COMMENTS          : Sweeps through the input arrays and plotting the
C                         inputs to the system on the apropriate axes.
C ********************************************************************************
      subroutine dwinps()
$include: 'sysmat.inc'
$include: 'time.inc'
      real*4 xpt(12),ypt(12)
      real*8 cent(4),scal(2)
      integer*2 i,j,ptind
      do 99 i = 1,order
         ptind = 1
         xpt(ptind) = 0.0
         ypt(ptind) = 0.0
         do 98 j = 1,5
            if (stpinp(j).eq.i) then
               ptind = ptind + 1
               xpt(ptind) = stpdat(j)
               ypt(ptind) = ypt(ptind-1)
               ptind = ptind + 1
               xpt(ptind) = stpdat(j)
               ypt(ptind) = stpinc(j) + ypt(ptind-1)
            endif
98       continue
         if (ptind.gt.1) then
            ptind = ptind + 1
            xpt(ptind) = tend
            ypt(ptind) = ypt(ptind-1)
            cent(1) = tcentr(1,i)
            cent(2) = tcentr(2,i)
            cent(3) = tcentr(3,i)
            cent(4) = tcentr(4,i)
            scal(1) = tscale(1,i)
            scal(2) = tscale(2,i)
            call drawpt(1,1,xpt,ypt,ptind,cent,scal,2)
         endif
99    continue
      return
```

```
      end

C ********************************************************************************
CH    REVISION HISTORY :
C     VERSION       BY                DATE        COMMENT
C     1.00          Ian Fisher        23/06/88    Creation.
C     FILEND           :
```

```
C
C     FILE              : SIMSCL.FOR
C ***************************************************************************
CN    MODULE NAME       : SIMSCL
CA    FUNCTION          : To enable the user to edit the time simulation axes
C                         scales.
CS    CALL SEQUENCE     : call simscl()
CI    INPUT PARAMETERS  : None.
CO    OUTPUT PARAMETERS: None.
CG    GLOBAL VARIABLES  : time.inc
C                         sysmat.inc
C                         keys.inc
CM    MODULES CALLED    : WIPSCR (asm),  DISP  (asm),  LEVEL (asm),  MOVE (asm),
C                         DLINE (asm), WRTSTR (asm), prtnum (for), wrtitl (for),
C                         prscal (for), INKEY (asm).
CE    ERROR CONDITIONS  : ?
C
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: 'sysmat.inc'
$include: 'keys.inc'
      integer*2 pos,key,elem,lastel,xpos,ypos

      call WIPSCR(1)
      call DISP(1)
      call LEVEL(1)
      call wrtitl(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 wrtitl(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 prtnum(1,xpos+10,ypos,1,4,'(i2)',2,i)
            call WRTSTR(1,xpos+30,ypos,1,',')
            call prtnum(1,xpos+40,ypos,1,4,'(i2)',2,i)
            call WRTSTR(1,xpos+60,ypos,1,')')
            ypos = ypos + 14
99       continue
         call wrtitl(1,400,40,3,'In ')
         call WRTSTR(1,400,54,3,'Out')
         call WRTSTR(1,436,47,1,'=')
         call wrtitl(1,200,95,14,'Output Scale :')
         call wrtitl(1,400,95,9,'Maximum :')
         call wrtitl(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          : To print out a 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 : time.inc
C                         sysnms.inc
CM     MODULES CALLED    : BLKFIL (asm), LENSTR (asm), LEVEL (asm), prtnum (for),
C                          WRTSTR (asm), wrtitl (for).
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
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 wrtitl(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 prtnum(1,490,95,4,7,'(g10.4)',10,ytlim(1,elem))
       call prtnum(1,490,110,4,7,'(g10.4)',10,ytlim(2,elem))
       return
       end

C ***************************************************************************
CN     MODULE NAME       : edscal
CA     FUNCTION          : Enables the user to 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 : time.inc
C                        keys.inc
CM    MODULES CALLED   : LEVEL (asm), BLKFIL (asm), box (for), ERTONE (asm),
C                        getnum (for), WRTSTR (asm).
CE    ERROR CONDITIONS : ?
C
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
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 = getnum(1,490,95,4,7,'(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 = getnum(1,490,110,4,7,'(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     FILEND           :
```

```
C
C     FILE              : STEPOPT.FOR
C ***************************************************************************
CN    MODULE NAME       : STPOPT
CA    FUNCTION          : To allow the user to edit the time simulation
C                         parameters.
CS    CALL SEQUENCE     : call stpopt()
CI    INPUT PARAMETERS  : None.
CO    OUTPUT PARAMETERS: None.
CG    GLOBAL VARIABLES  : time.inc.
C                         keys.inc
C                         sysmat.inc
CM    MODULES CALLED    : WIPSCR (asm), WRTSTR (asm), wrtitl (for), prtnum (for)
C                         INKEY (asm), loptyp (for), contyp (for), getnum (for),
C                         ERTONE (asm).
CE    ERROR CONDITIONS  : ?
C
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 stpopt()
      implicit integer*2 (I,g,S)
$include: 'time.inc'
$include: 'keys.inc'
$include: 'sysmat.inc'
      integer*2 pos,pos2,key,ferr,i,key2
      real*4 temp1

      pos = 1
      call WIPSCR(0)
      call wrtitl(0,250,25,27,'Time Simulation Parameters.')
      call WRTSTR(0,40,50,22,'System type          :')
      call WRTSTR(0,40,70,22,'Controller           :')
      call WRTSTR(0,40,90,22,'Time duration [secs] :')
      call WRTSTR(0,40,110,22,'Time step     [secs] :')
      call wrtitl(0,20,135,8,'Step No.')
      call wrtitl(0,110,135,9,'Input No.')
      call wrtitl(0,210,135,9,'Increment')
      call wrtitl(0,320,135,11,'Time [secs]')
      call WRTSTR(0,430,135,13,'(after start)')
      call contyp(coninc,1)
      call PRTNUM(0,240,90,3,7,'(g10.4)',10,tend)
      call PRTNUM(0,240,110,3,7,'(g10.4)',10,dt)
      do 888 i = 1,5
         call PRTNUM(0,40,(140+i*15),1,4,'(i3)',3,i)
         call PRTNUM(0,140,(140+i*15),1,4,'(i2)',2,stpinp(i))
         call PRTNUM(0,210,(140+i*15),3,7,'(g10.4)',10,stpinc(i))
         call PRTNUM(0,320,(140+i*15),3,7,'(g10.4)',10,stpdat(i))
888   continue

199   continue
      call WRTSTR(0,40,235,57,
     +'                                                          ')
      call WRTSTR(0,40,250,57,
     +'                                                          ')
      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 WRTSTR(0,40,235,57,
     +    'SPACE to select type. RETURN and TAB keys to move option.')
         call loptyp(cloop,2)
198      continue
         key = INKEY(1)
         if (key.eq.8192) then
            cloop = cloop + 1
            if (cloop.gt.1) cloop = 0
            call loptyp(cloop,3)
         endif
         if ((key.ne.tabk).and.(key.ne.retk).and.(key.ne.rtabk).and.
```

```
     +      (key.ne.esck)) goto 198
            call loptyp(cloop,1)
          elseif (pos.eq.2) then
            call WRTSTR(0,40,235,57,
     +       'SPACE to select type. RETURN and TAB keys to move option.')
            call contyp(coninc,2)
190         continue
            key = INKEY(1)
            if (key.eq.8192) then
               coninc = coninc + 1
               if (coninc.gt.1) coninc = 0
               call contyp(coninc,3)
            endif
            if ((key.ne.tabk).and.(key.ne.retk).and.(key.ne.rtabk).and.
     +      (key.ne.esck)) goto 190
             call contyp(coninc,1)
          elseif (pos.eq.3) then
197         continue
            key = getnum(0,240,90,3,7,'(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.4) then
195         continue
            key = getnum(0,240,110,3,7,'(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.5) then
            pos2 = 1
            do 180 i = 5,17,3
               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 = getnum(0,140,(140+pos2*15),1,4,'(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 PRTNUM(0,673,(140+pos2*15),1,4,'(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 = 6,18,3
               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 = getnum(0,210,(140+pos2*15),3,7,'(g10.4)',10,
```

```
     +                    stpinc(pos2))
          endif
          pos2 = 1
          do 177 i = 7,19,3
             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 = getnum(0,320,(140+pos2*15),3,7,'(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.4) then
             pos = pos + 3
          else
             pos = pos + 1
          endif
          if (pos.gt.19) pos = 1
       elseif ((key.eq.retk).or.(key.eq.tabk)) then
          pos = pos + 1
          if (pos.gt.19) pos = 1
       elseif (key.eq.upk) then
          if (pos.gt.6) then
             pos = pos - 3
          elseif (pos.eq.6) then
             pos = pos - 2
          else
             pos = pos - 1
          endif
          if (pos.lt.1) pos = 19
       elseif (key.eq.rtabk) then
          pos = pos - 1
          if (pos.lt.1) pos = 14
       endif
       if (key.ne.esck) goto 199
       call WIPSCR(0)
       return
       end

C ******************************************************************************
CN     MODULE NAME       : LOPTYP
CA     FUNCTION          : To generate the string 'loop type' - for editing and
C                          printing.
CS     CALL SEQUENCE     : call loptyp(type,sym)
CI     INPUT PARAMETERS :
C                  type - (integer*2) Loop type : 0 - closed
C                                                 1 - open
C                   sym - (integer*2) Operation : 1 : printing
C                                                 2 : first edit
C                                                 3 : change option
C
CO     OUTPUT PARAMETERS: None.
```

```
CG    GLOBAL VARIABLES : None.
CM    MODULES CALLED   : COPYST (asm), LEVEL (asm), WRTSTR (asm), sellop (for).
CE    ERROR CONDITIONS : ?
C
CC    COMMENTS         : Generates a standard string and then inserts chars
C                        according to loop type and operation. The printing
C                        of the string is dependent on operation.
C ***************************************************************************
      subroutine loptyp(type,sym)
      integer*2 type,sym
      integer*2 type2
      character*25 fstr
      if (sym.eq.1) then
         fstr = 'ClosedOpen'
         if (type.eq.0) then
            call COPYST(25,fstr,0,1,'[',0,1)
            call COPYST(25,fstr,7,1,']',0,1)
         else
            call COPYST(25,fstr,0,1,' ',0,1)
            call COPYST(25,fstr,7,1,' ',0,1)
         endif
         if (type.eq.1) then
            call COPYST(25,fstr,8,1,'[',0,1)
            call COPYST(25,fstr,13,1,']',0,1)
         else
            call COPYST(25,fstr,8,1,' ',0,1)
            call COPYST(25,fstr,13,1,' ',0,1)
         endif
         call LEVEL(0)
         call sellop(type)
         call LEVEL(1)
         call WRTSTR(0,245,50,25,fstr)
      else
         fstr = ' Closed  Open'
         if (sym.eq.3) then
            type2 = type - 1
            if (type2.lt.0) type2 = 1
            call LEVEL(2)
            call sellop(type2)
         endif
         if (sym.eq.2) then
            call LEVEL(1)
            call WRTSTR(0,245,50,25,fstr)
         endif
         call LEVEL(2)
         call sellop(type)
         call LEVEL(1)
      endif
      return
      end

C ***************************************************************************
CN    MODULE NAME      : SELLOP
CA    FUNCTION         : To back highlight the 'loop type' string.
CS    CALL SEQUENCE    : call sellop(type)
CI    INPUT PARAMETERS :
C                 type - (integer*2) The loop type : 0 - closed
C                                                    1 - open
C
CO    OUTPUT PARAMETERS: None.
CG    GLOBAL VARIABLES : None.
CM    MODULES CALLED   : BLKFIL (asm).
CE    ERROR CONDITIONS : ?
C
CC    COMMENTS         : Back highlights a string by filling in the block when
C                        print level is set to XOR mode.
C ***************************************************************************
      subroutine sellop(type)
      integer*2 type
      if (type.eq.0) then
         call BLKFIL(254,52,54,14)
      elseif (type.eq.1) then
```

```
         call BLKFIL(326,52,36,14)
      endif
      return
      end

C ***************************************************************************
CN    MODULE NAME       : CONTYP
CA    FUNCTION          : To generate the string 'controller state' - for
C                         editing and printing.
CS    CALL SEQUENCE     : call contyp(type,sym)
CI    INPUT PARAMETERS :
C                 type - (integer*2) Loop type : 0 - in
C                                                1 - out
C                  sym - (integer*2) Operation : 1 : printing
C                                                2 : first edit
C                                                3 : change option
C
CO    OUTPUT PARAMETERS: None.
CG    GLOBAL VARIABLES : None.
CM    MODULES CALLED   : COPYST (asm), LEVEL (asm), WRTSTR (asm), selcon (for).
CE    ERROR CONDITIONS : ?
C
CC    COMMENTS          : Generates a standard string and then inserts chars
C                         according to controller state and operation.  The
C                         printing of the string is dependent on operation.
C ***************************************************************************
      subroutine contyp(type,sym)
      integer*2 type,sym
      integer*2 type2
      character*25 fstr
      if (sym.eq.1) then
         fstr = 'InOut'
         if (type.eq.0) then
            call COPYST(25,fstr,0,1,'[',0,1)
            call COPYST(25,fstr,3,1,']',0,1)
         else
            call COPYST(25,fstr,0,1,' ',0,1)
            call COPYST(25,fstr,3,1,' ',0,1)
         endif
         if (type.eq.1) then
            call COPYST(25,fstr,4,1,'[',0,1)
            call COPYST(25,fstr,8,1,']',0,1)
         else
            call COPYST(25,fstr,4,1,' ',0,1)
            call COPYST(25,fstr,8,1,' ',0,1)
         endif
         call LEVEL(0)
         call selcon(type)
         call LEVEL(1)
         call WRTSTR(0,245,70,25,fstr)
      else
         fstr = ' In  Out'
         if (sym.eq.3) then
            type2 = type - 1
            if (type2.lt.0) type2 = 1
            call LEVEL(2)
            call selcon(type2)
         endif
         if (sym.eq.2) then
            call LEVEL(1)
            call WRTSTR(0,245,70,25,fstr)
         endif
         call LEVEL(2)
         call selcon(type)
         call LEVEL(1)
      endif
      return
      end

C ***************************************************************************
CN    MODULE NAME       : SELCON
CA    FUNCTION          : To back highlight the 'loop type' string.
```

```
CS    CALL SEQUENCE    : call selcon(type)
CI    INPUT PARAMETERS :
C                type - (integer*2) The controller state : 0 - in
C                                                          1 - out
C
CO    OUTPUT PARAMETERS: None.
CG    GLOBAL VARIABLES : None.
CM    MODULES CALLED   : BLKFIL (asm).
CE    ERROR CONDITIONS : ?
C
CC    COMMENTS         : Back highlights a string by filling in the block when
C                        print level is set to XOR mode.
C ***************************************************************************
      subroutine selcon(type)
      integer*2 type
      if (type.eq.0) then
         call BLKFIL(254,72,18,14)
      elseif (type.eq.1) then
         call BLKFIL(290,72,27,14)
      endif
      return
      end

C ***************************************************************************
CH    REVISION HISTORY :
C     VERSION       BY              DATE        COMMENT
C     1.00          Ian Fisher      23/06/88    Creation.
C     FILEND           :
```

```
C     FILE              : DOHELP.FOR
C ***************************************************************************
CN    MODULE NAME       : DOHELP
CA    FUNCTION          : To initiate the help facility.
CS    CALL SEQUENCE     : call dohelp()
CI    INPUT PARAMETERS  : None.
CO    OUTPUT PARAMETERS : None.
CG    GLOBAL VARIABLES  : keys.inc
CM    MODULES CALLED    : GETPG (asm), GETWPG (asm), GETATT (asm), FLIPG1 (asm),
C                         DISP  (asm),  LEVEL  (asm),  GPAGE (asm),  BOX  (asm),
C                         prhelp (for), ERTONE (asm), WRTSTR (asm), INKEY2 (asm)
CE    ERROR CONDITIONS  : ?
C
CC    COMMENTS          : The routine  first  saves  all  the attributes of the
C                         current system (eg. displayed page, write page, write
C                         attributes) and then calls a routine to dump graphics
C                         page 1 to a buffer in memory (simultaneously clearing
C                         page 1).
C                         The routine then checks if the help  file,  LOCI.HLP,
C                         can be found ; If so then control is  passed onto the
C                         main help routines, else an error message is printed.
C ***************************************************************************
C     The HELP format :
C                          i) The file is split up into sections, each section
C                             can contain any number of pages.
C                             (Section 0 is the menu header).
C
C                         ii) Each page is seperated by a control field/record :
C                                    '* sect page maxpages'
C                             or in FORTRAN format
C                                  format(a1,i4,i4,i4)
C                           The '*' must appear in column 1.
C
C                        iii) The first column of any information  must be a blank
C                             (which is not printed out). This is used to identify
C                             control fields from information fields.
C
C ***************************************************************************
      subroutine dohelp()
      implicit integer*2 (I,G)

$include: 'keys.inc'
      integer*2 key,oldpg,oldwr,oldatt,ferr
      oldpg = GETPG()
      oldwr = GETWPG()
      oldatt = GETATT()
      call flipg1(1)
      call DISP(1)
      call LEVEL(1)
      call GPAGE(1)
      call BOX(1,319,717,318)
      call BOX(3,317,713,314)
      open(6,FILE='LOCI.HLP',STATUS='OLD',IOSTAT=ferr)
      if (ferr.eq.0) then
         call prhelp(6)
      else
         call ERTONE()
         call WRTSTR(1,150,30,47,
     +   '*** The help data file : LOCI.HLP, could not be')
         call WRTSTR(1,150,45,16,'found or opened.')
         call WRTSTR(1,150,70,24,'Hit ESC to continue.....')
100      continue
         key = INKEY2()
         if (key.ne.esck) goto 100
      endif
      close(6)
      call flipg1(2)
      call LEVEL(oldatt)
      call DISP(oldpg)
      call GPAGE(oldwr)
      return
      end
```

```
$PAGE
C ********************************************************************************
CN     MODULE NAME       : PRHELP
CA     FUNCTION          : The controlling routine of the help facility.
CS     CALL SEQUENCE     : call prhelp(funit)
CI     INPUT PARAMETERS :
C                  funit - (integer*2) The unit number for the source of help
C                          information.
C
CO     OUTPUT PARAMETERS: None.
CG     GLOBAL VARIABLES : keys.inc
CM     MODULES CALLED   : prpage (for), blksec (for), drvhlp (for), INKEY2 (asm)
C                          ERTONE (asm).
CE     ERROR CONDITIONS : ?
C
CC     COMMENTS          : This routine  prints out the main help menu header and
C                          permits the user to select a section of help.
C                          The ESC or  F1  key  returns  control  to  the calling
C                          routine.
C ********************************************************************************
       subroutine prhelp(funit)
       implicit integer*2 (I)
       integer*2 funit
$include: 'keys.inc'
       integer*2 key,i,chkio,sect,pg,maxpg,count
       character*1 chkch

       rewind funit
       read(funit,'(a)',iostat=chkio,err=300) chkch
       if (chkch.eq.'*') then
          backspace funit
          read(funit,102,iostat=chkio,err=300) chkch,sect,pg,maxpg
102       format(a1,i4,i4,i4)
          backspace funit
          call prpage(sect,pg,funit)
          count = 1
          call blksec(count)
99        continue
          key = INKEY2()
          if (key.eq.upk) then
             call blksec(count)
             count = count - 1
             if (count.lt.1) count = 10
             call blksec(count)
          elseif (key.eq.downk) then
             call blksec(count)
             count = count + 1
             if (count.gt.10) count = 1
             call blksec(count)
          elseif (key.eq.retk) then
             call drvhlp(count,key,funit)
             if (key.ne.59) then
                rewind funit
                call prpage(sect,pg,funit)
                call blksec(count)
             endif
          endif
          if ((key.ne.esck).and.(key.ne.59)) goto 99
       endif
300    continue
       if (chkio.ne.0) then
          call ERTONE()
       endif
       return
       end
$PAGE
C ********************************************************************************
CN     MODULE NAME       : PRPAGE
CA     FUNCTION          : To print a single page of help information.
CS     CALL SEQUENCE     : call prpage(sect,pg,funit)
CI     INPUT PARAMETERS :
```

```
C                sect - (integer*2) The section number of the help to print.
C                  pg - (integer*2) The page number of the help to print.
C               funit - (integer*2) The unit  number  for  the source of help
C                       information.
C
CO    OUTPUT PARAMETERS: None.
CG    GLOBAL VARIABLES : None.
CM    MODULES CALLED   : WRTSTR (asm), findpg (for)
CE    ERROR CONDITIONS : ?
C
CC    COMMENTS         : The routine prints out the  data from the current
C                        file pointer until a 'help  file control char' is
C                        encountered.
C                        The routine does perform a  'backspace' operation
C                        before exit; thus the file pointer is pointing at
C                        the next page control char.
C ***************************************************************************
      subroutine prpage(sect,pg,funit)
      integer*2 sect,pg,funit

      character*1 chkch
      character*78 hlpstr,cl1
      integer*2 tsect,tpg,tmax,chkio,xpos,ypos
      call findpg(sect,pg,funit)
      hlpstr = ' '
      cl1 = ' '
      xpos = 10
      ypos = 17
99    continue
      read(funit,100,iostat=chkio,err=300) chkch,hlpstr
100   format(a1,a)
      if ((chkch.ne.'*').and.(chkch.ne.'+')) then
         call WRTSTR(1,xpos,ypos,77,cl1)
         call WRTSTR(1,xpos,ypos,77,hlpstr)
      else
199      continue
         call WRTSTR(1,xpos,ypos,77,cl1)
         ypos = ypos + 14
         if (ypos.le.315) goto 199
      endif
      ypos = ypos + 14
      if ((chkch.ne.'*').and.(chkch.ne.'+')) goto 99
      backspace funit
300   continue
      return
      end
$PAGE
C ***************************************************************************
CN    MODULE NAME      : FINDPG
CA    FUNCTION         : To find a page of information in the help file.
CS    CALL SEQUENCE    : call findpg(sect,pg,funit)
CI    INPUT PARAMETERS :
C                sect - (integer*2) The section number of the help to find.
C                  pg - (integer*2) The page number of the help to find.
C               funit - (integer*2) The unit  number  for  the source of help
C                       information.
C
CO    OUTPUT PARAMETERS: File pointer at the start of the requested page.
CG    GLOBAL VARIABLES : None.
CM    MODULES CALLED   : None.
CE    ERROR CONDITIONS : ?
C
CC    COMMENTS         : This routine scans  through  a  file  searching  for a
C                        particular section and page number.
C                        The scan starts at the current file  pointer  position
C                        and checks the first  control  record : If the section
C                        and page number is beyond that desired then a 'rewind'
C                        operation is performed  and  the  scan starts from the
C                        start of the help file.
C ***************************************************************************
      subroutine findpg(sect,pg,funit)
      integer*2 sect,pg,funit
```

```
99    continue
      if (oldpos.ne.pos) then
         call prpage(sect,pos,funit)
      endif
      oldpos = pos
      key = INKEY2()
      if (key.eq.pgdnk) then
         pos = pos + 1
         if (pos.gt.tmax) then
            pos = pos - 1
            call ERTONE()
         endif
      elseif (key.eq.pgupk) then
         pos = pos - 1
         if (pos.lt.1) then
            pos = pos + 1
            call ERTONE()
         endif
      endif
      if ((key.ne.esck).and.(key.ne.59)) goto 99
      if (key.eq.esck) key = 0
300   continue
      return
      end
$PAGE
C *****************************************************************************
CN    MODULE NAME       : BLKSEC
CA    FUNCTION          : To back highlight a menu header option.
CS    CALL SEQUENCE     : call blksec(sect)
CI    INPUT PARAMETERS :
C                 sect - (integer*2) The section number.
C
CO    OUTPUT PARAMETERS: None.
CG    GLOBAL VARIABLES : None.
CM    MODULES CALLED    : LEVEL (asm), BLKFIL (asm).
CE    ERROR CONDITIONS : ?
C
CC    COMMENTS          : The routine fills a block around the option while
C                         the write attribute is set to XOR.
C *****************************************************************************
      subroutine blksec(sect)
      integer*2 sect

      call LEVEL(2)
      call BLKFIL(45,(17+3*14+sect*14+3),320,14)
      call LEVEL(1)
      return
      end

C *****************************************************************************
CH    REVISION HISTORY :
C     VERSION       BY                  DATE        COMMENT
C     1.00          Ian Fisher          23/06/88    Creation.
C     FILEND           :
```

```
C
C     FILE              : AXES.FOR
C ***************************************************************************
CN    MODULE NAME       : DWAXES
CA    FUNCTION          : To draw a set of axes on a graphics page,
C                         given an area on the page and the axes
C                         dimensions.
CS    CALL SEQUENCE     : call dwaxes(page,nums,xtype,x,y,graph,
C                                     scale,centre)
CI    INPUT PARAMETERS :
C                      page - (integer*2) The page to which
C                             the axes are to be drawn.
C                      nums - (integer*2) The axes routine
C                             function number :-
C                             0 : No numbers on the axes.
C                             1 : Numbers on the axes.
C                             2 : No axes or numbers are to be
C                                 drawn.
C                     xtype - (integer*2) Type of x-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                 graph(4) - (integer*2) Dimensions of the
C                            area on the graphics page in
C                            which the set of axes are to be
C                            drawn.
C                            graph(1): X co-ord of the area
C                                    : lower left corner.
C                            graph(2): Y co-ord of the area
C                                    : lower left corner.
C                            graph(3): Width of area speci-
C                                      fied in dots.
C                            graph(4): 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.
C
CG    GLOBAL VARIABLES : None.
CM    MODULES CALLED   : ydata (for), xlin (for), xdecad (for), xoctav (for),
C                        doax (for).
CE    ERROR CONDITIONS : ?
C
CC    COMMENTS         : The routine first calculates the scales and
C                        centres of the set of axes, taking into
C                        account factors such as numbering along the
```

```
C                       axes. If  number  are  to  be printed on the
C                       axes, then  the  total  area  is adjusted to
C                       ensure that  the text will not extend beyond
C                       the defined area.
C
C                       Each  axis  is  drawn  separately  (ie.  The
C                       number  of  ticks  on the axes is calculated
C                       and then drawn  with  numbering, if the user
C                       requested numbering, along the axis).
C
C                       The X axes can be numbered using three types
C                       of systems : Linear, decade or octave scales.
C                       The SCALE array is adjust accordingly for the
C                       type of axes used.
C
C ***************************************************************************
      subroutine dwaxes(pg,nums,xtype,x,y,graph,scale,centre)
      implicit integer*2 (I)
      integer*2 pg,nums,xtype
      real*8 x(3),y(3)
      integer*2 graph(4)
      real*8 scale(2),centre(4),yspc,xspc

      integer*2 xch,ych,zeroy,gr2(4),key,i
      character*15 y1str(10),y2str(10),x1str(10),x2str(10)
      integer*2    y1len(10),y2len(10),x1len(10),x2len(10)
      integer*2    ylind,y2ind,x1ind,x2ind,y1max,x1max

      xch = 5
      ych = 6
      centre(3) = x(3)
      centre(4) = y(3)
      call ydata(nums,y,gr2,graph,scale,centre,y1str,y1len,ylind,
     +           y2str,y2len,y2ind,y1max,yspc)
      if (xtype.eq.1) then
         call xlin(nums,x,gr2,graph,scale,centre,x1str,x1len,x1ind,
     +             x2str,x2len,x2ind,y1max,xspc)
      elseif (xtype.eq.2) then
         call xdecad(nums,x,gr2,graph,scale,centre,x1str,x1len,xlind,
     +               x2str,x2len,x2ind,y1max,xspc)
      elseif (xtype.eq.3) then
         call xoctav(nums,x,gr2,graph,scale,centre,x1str,x1len,xlind,
     +               x2str,x2len,x2ind,y1max,xspc)
      endif

      call doax(pg,nums,xtype,x,y,gr2,graph,scale,centre,x1str,x1len,
     +          x1ind,x2str,x2len,x2ind,y1str,y1len,y1ind,y2str,
     +          y2len,y2ind,xspc,yspc)
      return
      end

C ***************************************************************************
CN    MODULE NAME       : YDATA
CA    FUNCTION          : To adjust window dimensions for the Y axes.
CA    CALL SEQUENCE     : call ydata(axnum,y,gr2,graph,scale,centre,
C                                    y1str,y1len,y1ind,y2str,y2len,y2ind,
C                                    y1max,yspc)
CI    INPUT PARAMETERS :
C                        axnum - (integer*2) The   axes   routine
C                                function number.
C                          y(3) - (real*8) Y axes dimensions :
C                                          y(1): Ymaximum.
C                                          y(2): Yminimum.
C                                          y(3): Y axes or origin.
C                     graph(4) - (integer*2) Dimensions  of   the
C                                area on  the  graphics  page  in
C                                which  the set of axes are to be
C                                drawn.
C
CO    OUTPUT PARAMETERS:
C                       gr2(4) - (integer*2) Adjusted  dimensions
C                                of  the  area  on  the  graphics
```

```
C                              page  in  which  the set of axes
C                              are to be drawn
C                    scale(2) - (real*8) The   axes    to   page
C                              scaling   factors   (  dots/axis
C                              unit).
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                   ylstr(10) - (character*15) The numbers for
C                              the positive y axes in string
C                              format - to be printed on the
C                              axes.
C                   y2str(10) - (character*15) The numbers for
C                              the negative y axes in string
C                              format - to be printed on the
C                              axes.
C                   yllen(10) - (integer*2) The  length  of  the
C                              positive y axes number strings.
C                   y2len(10) - (integer*2) The  length  of  the
C                              negative y axes number strings.
C                       ylind - (integer*2) The number of strings
C                              to be printed on  the positive y-
C                              axes.
C                       y2ind - (integer*2) The number of strings
C                              to be printed on  the negative y-
C                              axes.
C                       ylmax - (integer*2) Length of longest
C                              number string.
C                        yspc - (real*8)  The   spacing  between
C                              ticks on the y axes.
C
CG    GLOBAL VARIABLES : None.
CM    MODULES CALLED   : calaxe (for), aspace (for), dognum (for).
CE    ERROR CONDITIONS : ?
C
CC    COMMENTS         : The  routine  calculates  which  numbers  are  to  be
C                        printed on  the  y  axes.  The  numbers  are converted
C                        to printable strings. The dimensions of  the  graphics
C                        window is then adjusted to handle the printed strings.
C
C ******************************************************************************
      subroutine ydata(axnum,y,gr2,graph,scale,centre,ylstr,yllen,ylind,
     +                 y2str,y2len,y2ind,ylmax,yspc)
      implicit integer*2 (c)
      integer*2 axnum
      real*8 y(3)
      integer*2 gr2(4),graph(4)
      real*8 scale(2),centre(4)
      character*15 ylstr(10),y2str(10)
      integer*2    yllen(10),y2len(10)
      integer*2    ylind,y2ind,ylmax
      real*8 yspc

      integer*2 xch,ych,ytick

      xch = 5
      ych = 6

      if (axnum.eq.1) then
         gr2(2) = graph(2) - int(0.9*ych)
         gr2(4) = graph(4) - int(1.8*ych)
         scale(2) = (real(gr2(4))/(y(1)-y(2)))
         centre(2) = gr2(2) - (gr2(4) - calaxe(y,gr2(4)))
         if (((y(3)-y(2))*scale(2)).lt.(2*ych)) then
             gr2(2) = graph(2) - int(2.5*ych)
             gr2(4) = graph(4) - int(3.5*ych)
             scale(2) = (real(gr2(4))/(y(1)-y(2)))
             centre(2) = gr2(2) - (gr2(4) - calaxe(y,gr2(4)))
         endif
      else
         gr2(2) = graph(2)-2
```

```
          gr2(4) = graph(4)-4
          scale(2) = (real(gr2(4))/(y(1)-y(2)))
          centre(2) = gr2(2) - (gr2(4) - calaxe(y,gr2(4)))
        endif
        if (axnum.lt.2) then
           if (axnum.eq.0) then
              ych = 4
           endif
           ytick = 6
799        continue
           ytick = ytick - 1
           call aspace(yspc,y,ytick,y1ind,y2ind)
           if (((yspc*scale(2)).lt.(2.8*real(ych))).and.
     +         (ytick.gt.1)) goto 799
           call dognum(yspc,y,y1str,y1len,y1ind,
     +                 y2str,y2len,y2ind,y1max)

        endif
       return
       end

C ***************************************************************************
CN     MODULE NAME       : XLIN
CA     FUNCTION          : To adjust window dimensions for linear X axes.
CS     CALL SEQUENCE     : call xlin(axnum,x,gr2,graph,scale,centre,
C                                    x1str,x1len,x1ind,x2str,x2len,x2ind,
C                                    y1max,xspc)
CI     INPUT PARAMETERS :
C                 axnum - (integer*2) The   axes   routine
C                         function number.
C                  x(3) - (real*8) X axes dimensions :
C                                  x(1): Xmaximum.
C                                  x(2): Xminimum.
C                                  x(3): X axes or origin.
C              graph(4) - (integer*2) Dimensions  of   the
C                         area on  the  graphics  page  in
C                         which  the set of axes are to be
C                         drawn.
C
CO     OUTPUT PARAMETERS:
C                       gr2(4) - (integer*2) Adjusted  dimensions
C                                of  the  area  on  the  graphics
C                                page  in  which  the set of axes
C                                are to be drawn
C                     scale(2) - (real*8) The   axes    to   page
C                                scaling   factors   (  dots/axis
C                                unit).
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                    x1str(10) - (character*15) The numbers for
C                                the positive x axes in string
C                                format - to be printed on the
C                                axes.
C                    x2str(10) - (character*15) The numbers for
C                                the negative x axes in string
C                                format - to be printed on the
C                                axes.
C                    x1len(10) - (integer*2) The  length  of  the
C                                positive x axes number strings.
C                    x2len(10) - (integer*2) The  length  of  the
C                                negative x axes number strings.
C                        xlind - (integer*2) The number of strings
C                                to be printed on  the positive x-
C                                axes.
C                        x2ind - (integer*2) The number of strings
C                                to be printed on  the negative x-
C                                axes.
C                        y1max - (integer*2) Length of longest
C                                number string on y axes.
C                         xspc - (real*8)  The   spacing  between
```

```
C                              ticks on the x axes.
C
CG    GLOBAL VARIABLES : None.
CM    MODULES CALLED   : aspace (for), dognum (for), calaxe (for).
CE    ERROR CONDITIONS : ?
C
CC    COMMENTS         : The routine calculates which numbers are to be
C                        printed on the x axes. The numbers are converted
C                        to printable strings. The dimensions of the graphics
C                        window is then adjusted to handle the printed strings.
C
C ***************************************************************************
      subroutine xlin(axnum,x,gr2,graph,scale,centre,x1str,x1len,x1ind,
     +                x2str,x2len,x2ind,y1max,xspc)
      implicit integer*2 (c)
      integer*2 axnum
      real*8 x(3)
      integer*2 gr2(4),graph(4)
      real*8 scale(2),centre(4)
      character*15 x1str(10),x2str(10)
      integer*2    x1len(10),x2len(10)
      integer*2    x1ind,x2ind,y1max
      real*8 xspc

      integer*2 xtick,x1max,xch,ych

      xch = 5
      ych = 6
      if (axnum.lt.2) then
         if (axnum.eq.0) then
            xch = 2
         endif
         xtick = 6
99       continue
         xtick = xtick - 1
         call aspace(xspc,x,xtick,x1ind,x2ind)
         call dognum(xspc,x,x1str,x1len,x1ind,
     +               x2str,x2len,x2ind,x1max)
         if ((int(0.70*(real(xch*x2len(1))))).gt.
     +       (int(real(y1max)+3.5)*xch)) then
            gr2(1) = graph(1) + int(0.70*(real(xch*x2len(1))))
            gr2(3) = graph(3) - int(0.70*(real(xch*x2len(1))))
         else
            gr2(1) = graph(1) + (int(real(y1max)+3.5)*xch)
            gr2(3) = graph(3) - (int(real(y1max)+3.5)*xch)
         endif
         gr2(3) = gr2(3) - int(0.70*(real(xch*x1len(1))))
         scale(1) = (real(gr2(3))/(x(1)-x(2)))
         centre(1) = gr2(1) + (gr2(3) - calaxe(x,gr2(3)))
         if (((xspc*scale(1)).lt.(1.5*real(x1max*xch))).and.
     +       (xtick.gt.1)) goto 99
      else
         gr2(1) = graph(1)+1
         gr2(3) = graph(3)-2
         scale(1) = (real(gr2(3))/(x(1)-x(2)))
         centre(1) = gr2(1) + (gr2(3) - calaxe(x,gr2(3)))
      endif
      return
      end

C ***************************************************************************
CN    MODULE NAME      : XDECAD
CA    FUNCTION         : To adjust window dimensions for decade type X axes.
CS    CALL SEQUENCE    : call xdecad(axnum,x,gr2,graph,scale,centre,
C                                    x1str,x1len,x1ind,x2str,x2len,x2ind,
C                                    y1max,xspc)
CI    INPUT PARAMETERS :
C               axnum - (integer*2) The   axes   routine
C                       function number.
C                x(3) - (real*8) X axes dimensions :
C                                x(1): Xmaximum.
C                                x(2): Xminimum.
```

```
C                                x(3): X axes or origin.
C              graph(4) - (integer*2) Dimensions of the
C                         area on the graphics page in
C                         which the set of axes are to be
C                         drawn.
C
CO     OUTPUT PARAMETERS:
C                        gr2(4) - (integer*2) Adjusted dimensions
C                                 of the area on the graphics
C                                 page in which the set of axes
C                                 are to be drawn
C                      scale(2) - (real*8) The axes to page
C                                 scaling factors ( dots/axis
C                                 unit).
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                     x1str(10) - (character*15) The numbers for
C                                 the positive x axes in string
C                                 format - to be printed on the
C                                 axes.
C                     x2str(10) - (character*15) The numbers for
C                                 the negative x axes in string
C                                 format - to be printed on the
C                                 axes.
C                     x1len(10) - (integer*2) The length of the
C                                 positive x axes number strings.
C                     x2len(10) - (integer*2) The length of the
C                                 negative x axes number strings.
C                         xlind - (integer*2) The number of strings
C                                 to be printed on the positive x-
C                                 axes.
C                         x2ind - (integer*2) The number of strings
C                                 to be printed on the negative x-
C                                 axes.
C                         ylmax - (integer*2) Length of longest
C                                 number string on y axes.
C                          xspc - (real*8) The spacing between
C                                 ticks on the x axes.
C
CG     GLOBAL VARIABLES : None.
CM     MODULES CALLED   : getdec (for), pnums (for), calaxe (for).
CE     ERROR CONDITIONS : ?
C
CC     COMMENTS         : The routine calculates which numbers are to be
C                         printed on the x axes. The numbers are converted
C                         to printable strings. The dimensions of the graphics
C                         window is then adjusted to handle the printed strings.
C
C ***************************************************************************
      subroutine xdecad(axnum,x,gr2,graph,scale,centre,x1str,x1len,
     +                  x1ind,x2str,x2len,x2ind,ylmax,xspc)
      implicit integer*2 (c)
      integer*2 axnum
      real*8 x(3)
      integer*2 gr2(4),graph(4)
      real*8 scale(2),centre(4)
      character*15 x1str(10),x2str(10)
      integer*2    x1len(10),x2len(10)
      integer*2    x1ind,x2ind,ylmax
      real*8 xspc
      real*4 x1num(10)

      integer*2 skip,decs,x1max,xch,ych,i
      real*8 x2(3)

      x2ind = 0
      xch = 5
      ych = 6
      x2(1) = x(1)
      x2(2) = x(3)
```

```
      x2(3) = x(3)
      skip = 1
      decs = 0
      if (axnum.lt.2) then
777      continue
         call getdec(x2,skip,x1num,x1ind,decs)
         call pnums(x1num,x1ind,x1str,x1len)
         x1max = 0
         do 776 i = 1,x1ind
            if (x1len(i).gt.x1max) then
               x1max = x1len(i)
            endif
776      continue
         if (axnum.eq.1) then
            gr2(1) = graph(1) + (int(real(y1max)+3.5)*xch)
            gr2(3) = graph(3) - (int(real(y1max)+3.5)*xch)
            gr2(3) = gr2(3) - int(0.70*(real(xch*x1len(1))))
         else
            gr2(1) = graph(1) + 4
            gr2(3) = graph(3) - 6
         endif
         if (decs.gt.1) then
            scale(1) = (real(gr2(3))/((real(decs-1))))
         else
            scale(1) = real(gr2(3))
         endif
         xspc = real(skip)
         centre(1) = gr2(1) + (gr2(3) - calaxe(x2,gr2(3)))
         skip = skip + 1
         if (x1ind.gt.5) goto 777
         if ((real(skip)*scale(1).lt.(3.0*real(x1max*xch))).and.
     +        (x1ind.gt.1)) goto 777
      else
         call getdec(x2,1,x1num,x1ind,decs)
         gr2(1) = graph(1)
         gr2(3) = graph(3)
         scale(1) = (real(gr2(3))/((real(decs-1))))
         centre(1) = gr2(1) + (gr2(3) - calaxe(x,gr2(3)))
      endif
      return
      end

C ******************************************************************************
CN    MODULE NAME      : XOCTAV
CA    FUNCTION         : To adjust window dimensions for octave type X axes.
CS    CALL SEQUENCE    : call xoctav(axnum,x,gr2,graph,scale,centre,
C                                    x1str,x1len,x1ind,x2str,x2len,x2ind,
C                                    y1max,xspc)
CI    INPUT PARAMETERS :
C               axnum - (integer*2) The   axes   routine
C                       function number.
C                  x(3) - (real*8) X axes dimensions :
C                                  x(1): Xmaximum.
C                                  x(2): Xminimum.
C                                  x(3): X axes or origin.
C              graph(4) - (integer*2) Dimensions  of   the
C                       area on  the  graphics  page  in
C                       which  the set of axes are to be
C                       drawn.
C
CO    OUTPUT PARAMETERS:
C                     gr2(4) - (integer*2) Adjusted  dimensions
C                              of  the  area  on  the  graphics
C                              page  in  which  the set of axes
C                              are to be drawn
C                   scale(2) - (real*8) The   axes    to   page
C                              scaling   factors   (  dots/axis
C                              unit).
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                  x1str(10) - (character*15) The numbers for
C                              the positive x axes in string
C                              format - to be printed on the
C                              axes.
C                  x2str(10) - (character*15) The numbers for
C                              the negative x axes in string
C                              format - to be printed on the
C                              axes.
C                  x1len(10) - (integer*2) The length of the
C                              positive x axes number strings.
C                  x2len(10) - (integer*2) The length of the
C                              negative x axes number strings.
C                      x1ind - (integer*2) The number of strings
C                              to be printed on the positive x-
C                              axes.
C                      x2ind - (integer*2) The number of strings
C                              to be printed on the negative x-
C                              axes.
C                      y1max - (integer*2) Length of longest
C                              number string on y axes.
C                       xspc - (real*8) The spacing between
C                              ticks on the x axes.
C
CG    GLOBAL VARIABLES : None.
CM    MODULES CALLED   : getoct (for), pnums (for), calaxe (for).
CE    ERROR CONDITIONS : ?
C
CC    COMMENTS         : The routine calculates which numbers are to be
C                        printed on the x axes. The numbers are converted
C                        to printable strings. The dimensions of the graphics
C                        window is then adjusted to handle the printed strings.
C
C ***************************************************************************
      subroutine xoctav(axnum,x,gr2,graph,scale,centre,x1str,x1len,
     +                  x1ind,x2str,x2len,x2ind,y1max,xspc)
      implicit integer*2 (c)
      integer*2 axnum
      real*8 x(3)
      integer*2 gr2(4),graph(4)
      real*8 scale(2),centre(4)
      character*15 x1str(10),x2str(10)
      integer*2    x1len(10),x2len(10)
      integer*2    x1ind,x2ind,y1max
      real*8 xspc
      real*4 x1num(10)

      integer*2 skip,octs,x1max,xch,ych,i,key
      real*8 x2(3)

      x2ind = 0
      xch = 5
      ych = 6
      x2(1) = x(1)
      x2(2) = x(3)
      x2(3) = x(3)
      skip = 1
      octs = 0
      if (axnum.lt.2) then
666      continue
         call getoct(x2,skip,x1num,x1ind,octs)
         call pnums(x1num,x1ind,x1str,x1len)
         x1max = 0
         do 665 i = 1,x1ind
            if (x1len(i).gt.x1max) then
               x1max = x1len(i)
            endif
665      continue
         if (axnum.eq.1) then
            gr2(1) = graph(1) + (int(real(y1max)+3.5)*xch)
            gr2(3) = graph(3) - (int(real(y1max)+3.5)*xch)
            gr2(3) = gr2(3) - int(0.70*(real(xch*x1len(1))))
         else
```

```
            gr2(1) = graph(1) + 4
            gr2(3) = graph(3) - 6
         endif
         if (octs.gt.1) then
            scale(1) = (real(gr2(3))/((real(octs-1))))
         else
            scale(1) = real(gr2(3))
         endif
         xspc = real(skip)
         centre(1) = gr2(1) + (gr2(3) - calaxe(x2,gr2(3)))
         skip = skip + 1
         if (x1ind.gt.5) goto 666
         if ((real(skip)*scale(1).lt.(3.0*real(x1max*xch))).and.
     +        (x1ind.gt.1)) goto 666
      else
         call getoct(x2,1,x1num,x1ind,octs)
         gr2(1) = graph(1)
         gr2(3) = graph(3)
         scale(1) = (real(gr2(3))/((real(octs-1))))
         centre(1) = gr2(1) + (gr2(3) - calaxe(x,gr2(3)))
      endif

      return
      end

C ***************************************************************************
CN    MODULE NAME       : GETDEC                                              '
CA    FUNCTION          : To calculate which numbers appear on the x-axes for
C                         a decade type x-axes.
CS    CALL SEQUENCE     : call getdec(x,skip,x1num,x1ind,decs)
CI    INPUT PARAMETERS :
C                 x(3) - (real*8) X axes dimensions :
C                                 x(1): Xmaximum.
C                                 x(2): Xminimum.
C                                 x(3): X axes or origin.
C                 skip - (integer*2) The number of decades to skip for each
C                        decade number that appears on the x-axes.
C
CO    OUTPUT PARAMETERS:
C              x1num(10) - (real*4) The numbers to appear on the x-axes.
C                  x1ind - (integer*2) The number of decades to be printed.
C                   decs - (integer*2) The total number of decades over the
C                        specified range.
C
CG    GLOBAL VARIABLES : None.
CM    MODULES CALLED   : None.
CE    ERROR CONDITIONS : ?
C
CC    COMMENTS         : Sweeps through the specified range counting the number
C                        of decades. Certain decades are put into an array
C                        according to the skip factor. The values put into the
C                        array are eventually converted into strings and
C                        printed on the x-axes.
C
C ***************************************************************************
      subroutine getdec(x,skip,x1num,x1ind,decs)
      real*8 x(3)
      integer*2 skip
      real*4 x1num(10)
      integer*2 x1ind,decs

      integer*2 decno,count,i
      real*8 tdec

      if (x(3).gt.(0.0)) then
         decno = 1
         tdec = x(3)
         do 888 i = 1,20
            if ((abs(tdec-x(1)).gt.(1.0e-5)).and.(tdec.lt.x(1))) then
               tdec = tdec*10.0
               decno = decno + 1
            endif
```

```
888      continue
         decs = decno
         xlind = 0
887      continue
            xlind = xlind + 1
            xlnum(xlind) = tdec
         do 886 i = 1,skip
            tdec = tdec/10.0
886      continue
         decno = decno - skip
         if ((decno.gt.1).and.(xlind.lt.10)) goto 887
      endif
      return
      end

C ***************************************************************************
CN    MODULE NAME       : GETOCT
CA    FUNCTION          : To calculate which numbers appear on the x-axes for
C                         a octave type x-axes.
CS    CALL SEQUENCE     : call getdec(x,skip,xlnum,xlind,decs)
CI    INPUT PARAMETERS :
C                x(3) - (real*8) X axes dimensions :
C                                x(1): Xmaximum.
C                                x(2): Xminimum.
C                                x(3): X axes or origin.
C                skip - (integer*2) The number of octaves to skip for each
C                       octave number that appears on the x-axes.
C
CO    OUTPUT PARAMETERS:
C              xlnum(10) - (real*4) The numbers to appear on the x-axes.
C                  xlind - (integer*2) The number of octaves to be printed.
C                   decs - (integer*2) The total number of octaves over the
C                          specified range.
C
CG    GLOBAL VARIABLES : None.
CM    MODULES CALLED   : None.
CE    ERROR CONDITIONS : ?
C
CC    COMMENTS         : Sweeps through the specified range counting the number
C                        of  octaves. Certain  octaves  are  put  into an array
C                        according to the skip factor. The values put  into the
C                        array  are  eventually   converted  into  strings  and
C                        printed on the x-axes.
C
C ***************************************************************************
      subroutine getoct(x,skip,xlnum,xlind,octs)
      real*8 x(3)
      integer*2 skip
      real*4 xlnum(10)
      integer*2 xlind,octs

      integer*2 octno,count,i
      real*8 toct

      if (x(3).gt.(0.0)) then
         octno = 1
         toct = x(3)
         do 555 i = 1,20
            if ((abs(toct-x(1)).gt.(1.0e-5)).and.(toct.lt.x(1))) then
               toct = toct*2.0
               octno = octno + 1
            endif
555      continue
         octs = octno
         xlind = 0
554      continue
            xlind = xlind + 1
            xlnum(xlind) = toct
         do 553 i = 1,skip
            toct = toct/2.0
553      continue
         octno = octno - skip
```

```
         if ((octno.gt.1).and.(xlind.lt.10)) goto 554
      endif
      return
      end
c *************************************************************************
C +++ ---------------------------------------------------------------------
C +++ Part of the calculation for the origin of the axes.
C +++
C +++ ---------------------------------------------------------------------
      integer*2 function calaxe(g,gdots)
      real*8 g(3)
      integer*2 gdots
      calaxe = ((g(1)-g(3))/(g(1)-g(2)))*(real(gdots))
      return
      end

c *************************************************************************
C +++    space - Distance between each tick.
C +++     g(3) - Max, Min, Origin of axes.
C +++ scale(2) - X, Y scaling factors (dots per axes unit).
C +++    ntick - Number of ticks on the axes.
C +++    nomax - Number of ticks on the Origin-Max axis.
C +++    nomin - Number of ticks on the Min-Origin axis.
C +++
C +++ ---------------------------------------------------------------------
      subroutine aspace(space,g,ntick,nomax,nomin)
      real*8 space,g(3)
      integer*2 ntick,nomax,nomin

      if (abs(g(1)-g(3)).ge.abs(g(3)-g(2))) then
         space = abs(g(1) - g(3))/(real(ntick))
         nomax = ntick
         nomin = (abs(g(3)-g(2))/space)+1
      else
         space = abs(g(3) - g(2))/(real(ntick))
         nomax = (abs(g(1)-g(3))/space)+1
         nomin = ntick
      endif
      return
      end

c *************************************************************************
CN    MODULE NAME       : DOGNUM
CA    FUNCTION          : To calculate and convert the numbers for linear
C                         type axis.
CS    CALL SEQUENCE     : call dognum(gspc,g,g1str,g1len,g1ind,
C                                     g2str,g2len,g2ind,g1max)
CI    INPUT PARAMETERS :
C                gspc - (real*8) The   spacing  between
C                                 ticks on the axis.
C                g(3) - (real*8) Axis dimensions :
C                                 g(1): maximum.
C                                 g(2): minimum.
C                                 g(3): opposite axis or origin.
C
CO    OUTPUT PARAMETERS:
C           g1str(10) - (character*15) The numbers for the positive axes
C                       in string format - to be printed on the axes.
C           g2str(10) - (character*15) The numbers for the negative axes
C                       in string format - to be printed on the axes.
C           g1len(10) - (integer*2) The  length  of  the  positive  axes
C                       number strings.
C           g2len(10) - (integer*2) The  length  of  the  negative  axes
C                       number strings.
C               g1ind - (integer*2) The number of strings to be  printed
C                       on  the positive axes.
C               g2ind - (integer*2) The number of strings to be  printed
C                       on  the negative axes.
C               g1max - (integer*2) The maximum  length  of  the  number
C                       strings.
CG    GLOBAL VARIABLES : None.
CM    MODULES CALLED   : pnums (for).
```

```
CE    ERROR CONDITIONS : ?
C
CC    COMMENTS         : Calculates the number to be put on the axis and
C                        then converts the number into a string for printing.
C                        The strings and the length of the strings are stored
C                        in arrays for printing at a later stage. The maximum
C                        string length is also calculated for axes dimension
C                        adjustment.
C
C ***************************************************************************
      subroutine dognum(gspc,g,g1str,g1len,g1ind,
     +                  g2str,g2len,g2ind,g1max)
      real*8 gspc,g(3)
      character*15 g1str(10),g2str(10)
      integer*2    g1len(10),g2len(10)
      integer*2    g1ind,g2ind,g1max

      real*4 tnum(10)
      integer*2 i
      do 699 i = 1,g1ind
         tnum(i) = g(1) - (gspc*(real(i-1)))
699   continue
      call pnums(tnum,g1ind,g1str,g1len)
      do 698 i = 1,g2ind
         tnum(i) = g(2) + (gspc*(real(i-1)))
698   continue
      call pnums(tnum,g2ind,g2str,g2len)

      g1max = 0
      do 697 i = 1,g1ind
         if (g1len(i).gt.g1max) then
            g1max = g1len(i)
         endif
697   continue
      do 696 i = 1,g2ind
         if (g2len(i).gt.g1max) then
            g1max = g2len(i)
         endif
696   continue
      return
      end

C ***************************************************************************
CN    MODULE NAME      : DOAX
CA    FUNCTION         : To print the set axes.
CS    CALL SEQUENCE    : call doax(pg,axnum,xtype,x,y,gr2,graph,scale,centre,
C                                  x1str,x1len,x1ind,x2str,x2len,x2ind,y1str,
C                                  y1len,y1ind,y2str,y2len,y2ind,xspc,yspc)
CI    INPUT PARAMETERS :
C                  pg - (integer*2) The graphics page on which the set of
C                       axes is to be drawn.
C               axnum - (integer*2) The axes routine function number.
C               xtype - (integer*2) Type of x-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             gr2(4) - (integer*2) Adjusted dimensions of the area on
C                      the graphics page in which the set of axes are
C                      to be drawn
C            scale(2) - (real*8) The axes to page scaling factors
C                      (dots/axis unit).
C          centre(4) - (real*8) The origin of the axes on the actual
C                      graphics page and the axes origin (as given in
C                      x(3) and y(3)).
```

```
C           x1str(10) - (character*15) The numbers for the positive
C                       x-axes in string format - to be printed on the
C                       axes.
C           x2str(10) - (character*15) The numbers for the negative
C                       x-axes in string format - to be printed on the
C                       axes.
C           x1len(10) - (integer*2) The length of the positive x-axes
C                       number strings.
C           x2len(10) - (integer*2) The length of the negative x-axes
C                       number strings.
C               x1ind - (integer*2) The number of strings to be printed
C                       on the positive x-axes.
C               x2ind - (integer*2) The number of strings to be printed
C                       on the negative x-axes.
C           y1str(10) - (character*15) The numbers for the positive
C                       y-axes in string format - to be printed on the
C                       axes.
C           y2str(10) - (character*15) The numbers for the negative
C                       y-axes in string format - to be printed on the
C                       axes.
C           y1len(10) - (integer*2) The length of the positive y-axes
C                       number strings.
C           y2len(10) - (integer*2) The length of the negative y-axes
C                       number strings.
C               y1ind - (integer*2) The number of strings to be printed
C                       on the positive y-axes.
C               y2ind - (integer*2) The number of strings to be printed
C                       on the negative y-axes.
C                xspc - (real*8) The spacing between ticks on the x axes.
C                yspc - (real*8) The spacing between ticks on the y axes.
C
CO     OUTPUT PARAMETERS:
CG     GLOBAL VARIABLES : None.
CM     MODULES CALLED   : DISP (asm), GPAGE (asm), MOVE (asm), DLINE (asm),
C                         dytick (for), dxtick (for).
CE     ERROR CONDITIONS : ?
C
CC     COMMENTS         : Draws the axes lines on the requested graphics page
C                         and then calls routines to draw the ticks on the axes
C                         and then number the ticks.
C ***************************************************************************
      subroutine doax(pg,axnum,xtype,x,y,gr2,graph,scale,centre,x1str,
     +                x1len,x1ind,x2str,x2len,x2ind,y1str,y1len,y1ind,
     +                y2str,y2len,y2ind,xspc,yspc)
      integer*2 pg,axnum,xtype
      real*8 x(3),y(3)
      integer*2 gr2(4),graph(4)
      real*8 scale(2),centre(4)
      character*15 y1str(10),y2str(10),x1str(10),x2str(10)
      integer*2    y1len(10),y2len(10),x1len(10),x2len(10)
      integer*2    y1ind,y2ind,x1ind,x2ind
      real*8 xspc,yspc

      call DISP(pg)
      call GPAGE(pg)
      if (axnum.ne.2) then
         call MOVE(gr2(1),int(centre(2)))
         call DLINE((gr2(1)+gr2(3)),int(centre(2)))
         call dytick(pg,axnum,y,gr2,scale,centre,
     +               y1str,y1len,y1ind,y2str,y2len,y2ind,yspc)
         call MOVE(int(centre(1)),gr2(2))
         call DLINE(int(centre(1)),(gr2(2)-gr2(4)))
         call dxtick(pg,axnum,x,gr2,scale,centre,
     +               x1str,x1len,x1ind,x2str,x2len,x2ind,xspc)
      endif
      return
      end

C ***************************************************************************
CN     MODULE NAME      : DYTICK
CA     FUNCTION         : To draw the ticks and numbers on the Y axis.
CS     CALL SEQUENCE    : call (pg,axnum,y,gr2,scale,centre,y1str,y1len,y1ind,
```

```
C                                y2str,y2len,y2ind,yspc)
CI    INPUT PARAMETERS :
C                    pg - (integer*2) The graphics page on which the set of
C                         axes is to be drawn.
C                 axnum - (integer*2) The axes routine function number.
C                 y(3) - (real*8) Y axes dimensions :
C                                    y(1): Ymaximum.
C                                    y(2): Yminimum.
C                                    y(3): Y axes or origin.
C               gr2(4) - (integer*2) Adjusted dimensions of the area on
C                        the graphics page in which the set of axes are
C                        to be drawn
C              scale(2) - (real*8) The axes to page scaling factors
C                        (dots/axis unit).
C             centre(4) - (real*8) The origin of the axes on the actual
C                        graphics page and the axes origin (as given in
C                        x(3) and y(3)).
C             ylstr(10) - (character*15) The numbers for the positive
C                        y-axes in string format - to be printed on the
C                        axes.
C             yllen(10) - (integer*2) The length of the positive y-axes
C                        number strings.
C                 ylind - (integer*2) The number of strings to be printed
C                        on the positive y-axes.
C             y2str(10) - (character*15) The numbers for the negative
C                        y-axes in string format - to be printed on the
C                        axes.
C             y2len(10) - (integer*2) The length of the negative y-axes
C                        number strings.
C                 y2ind - (integer*2) The number of strings to be printed
C                        on the negative y-axes.
C                  yspc - (real*8) The spacing between ticks on the y axes.
C
CO    OUTPUT PARAMETERS: None.
CG    GLOBAL VARIABLES : None.
CM    MODULES CALLED   : MOVE (asm), DLINE (asm), dwnumb (for).
CE    ERROR CONDITIONS : ?
C
CC    COMMENTS         : Draws the ticks on the y axis and then prints the
C                        numbers associated with each tick.
C ***************************************************************************
      subroutine dytick(pg,axnum,y,gr2,scale,centre,
     +                  ylstr,yllen,ylind,y2str,y2len,y2ind,yspc)
      integer*2 pg,axnum
      real*8 y(3)
      integer*2 gr2(4)
      real*8 scale(2),centre(4)
      character*15 ylstr(10),y2str(10)
      integer*2    yllen(10),y2len(10),ylind,y2ind
      real*8 yspc

      integer*2 xt,yt,i,xch,ych

      xch = 5
      ych = 6
      do 99 i = 0,(ylind-1)
         xt = int(centre(1) - 2.0)
         yt = (gr2(2)-gr2(4))+(real(i)*yspc*scale(2))
         if (abs(int(centre(2)-yt)).gt.(int(yspc*scale(2))-5))
     +     then
            call MOVE(xt,yt)
            call DLINE(xt+4,yt)
            if ((int(yspc*scale(2)).gt.int(1.2*ych)).and.
     +         (axnum.eq.1)) then
               call dwnumb(pg,(xt-(int((yllen(i+1)+3)*xch))),
     +              (yt+int(0.5*ych)),ylstr(i+1),yllen(i+1))
            endif
            yt =  yt + (0.5)*yspc*scale(2)
            if (abs(int(centre(2))-yt).gt.int(0.5*ych)) then
               xt = xt + 1
               call MOVE(xt,yt)
               call DLINE(xt+2,yt)
```

```
            endif
         elseif ((i.eq.0).and.((abs(centre(2)-yt)).gt.2)) then
            call MOVE(xt,yt)
            call DLINE(xt+4,yt)
            if (((abs(centre(2)-yt)).gt.int(1.2*ych)).and.
     +         (axnum.eq.1)) then
               call dwnumb(pg,(xt-(int((y1len(i+1)+3)*xch))),
     +              (yt+int(0.5*ych)),y1str(i+1),y1len(i+1))
            endif
         endif
99    continue

      do 98 i = 0,(y2ind-1)
         xt = int(centre(1)-2)
         yt = gr2(2)-(real(i)*yspc*scale(2))
         if (abs(int(centre(2)-yt)).gt.(int(yspc*scale(2))-5))
     +     then
            call MOVE(xt,yt)
            call DLINE(xt+4,yt)
            if ((int(yspc*scale(2)).gt.int(1.2*ych)).and.
     +         (axnum.eq.1)) then
               call dwnumb(pg,(xt-(int((y2len(i+1)+3)*xch))),
     +              (yt+int(0.5*ych)),y2str(i+1),y2len(i+1))
            endif
            yt =  yt - (0.5)*yspc*scale(2)
            if (abs(int(centre(2)-yt)).gt.int(0.5*ych)) then
               xt = xt + 1
               call MOVE(xt,yt)
               call DLINE(xt+2,yt)
            endif
         elseif ((i.eq.0).and.((abs(centre(2)-yt)).gt.2)) then
            call MOVE(xt,yt)
            call DLINE(xt+4,yt)
            if (((abs(centre(2)-yt)).gt.int(1.2*ych)).and.
     +         (axnum.eq.1)) then
               call dwnumb(pg,(xt-(int((y2len(i+1)+3)*xch))),
     +              (yt+int(0.5*ych)),y2str(i+1),y2len(i+1))
            endif
         endif
98    continue

      return
      end

C ***************************************************************************
CN    MODULE NAME       : DXTICK
CA    FUNCTION          : To draw the ticks and numbers on the X axis.
CS    CALL SEQUENCE     : call dxtick(pg,axnum,x,gr2,scale,centre,x1str,x1len,
C                                     x1ind,x2str,x2len,x2ind,xspc)
CI    INPUT PARAMETERS :
C                   pg - (integer*2) The graphics page on which the set of
C                        axes is to be drawn.
C                 axnum - (integer*2) The axes routine function number.
C                 x(3) - (real*8) X axes dimensions :
C                                 x(1): Xmaximum.
C                                 x(2): Xminimum.
C                                 x(3): X axes or origin.
C               gr2(4) - (integer*2) Adjusted dimensions of the area on
C                        the graphics page in which the set of axes are
C                        to be drawn
C            scale(2) - (real*8) The axes to page scaling factors
C                        (dots/axis unit).
C            centre(4) - (real*8) The origin of the axes on the actual
C                        graphics page and the axes origin (as given in
C                        x(3) and y(3)).
C            x1str(10) - (character*15) The numbers for the positive
C                        x-axes in string format - to be printed on the
C                        axes.
C            x1len(10) - (integer*2) The length of the positive x-axes
C                        number strings.
C                xlind - (integer*2) The number of strings to be printed
C                        on the positive x-axes.
```

```
C           x2str(10) - (character*15) The   numbers  for  the   negative
C                       x-axes  in  string  format - to be printed on the
C                       axes.
C           x2len(10) - (integer*2) The  length  of  the  negative x-axes
C                       number strings.
C               x2ind - (integer*2) The number  of strings  to be printed
C                       on  the negative x-axes.
C                xspc - (real*8) The spacing between ticks on the x axes.
C
CO     OUTPUT PARAMETERS: None.
CG     GLOBAL VARIABLES : None.
CM     MODULES CALLED   : MOVE (asm), DLINE (asm), dwnumb (for).
CE     ERROR CONDITIONS : ?
C
CC     COMMENTS         : Draws the ticks on the x axis and  then  prints  the
C                         numbers associated with each tick.
C ***************************************************************************
       subroutine dxtick(pg,axnum,x,gr2,scale,centre,
      +                  x1str,x1len,x1ind,x2str,x2len,x2ind,xspc)
       integer*2 pg,axnum
       real*8 x(3)
       integer*2 gr2(4)
       real*8 scale(2),centre(4)
       character*15 x1str(10),x2str(10)
       integer*2    x1len(10),x2len(10),x1ind,x2ind
       real*8 xspc

       integer*2 xt,yt,i,xch,ych
       real*8 xd

       xch = 5
       ych = 6

       do 199 i = 0,(x1ind-1)
          xt = (gr2(1)+gr2(3))-int(real(i)*xspc*scale(1))
          yt = int(centre(2)-2)
          if (abs(int(centre(1)-xt)).gt.(int(xspc*scale(1)/2.0))) then
             call MOVE(xt,yt)
             call DLINE(xt,yt+4)
             if ((int(xspc*scale(1)).gt.
      +           (int((real(x1len(i+1))/2.0+1.6)*xch))).and.
      +          (axnum.eq.1)) then
                call dwnumb(pg,
      +             (xt-(int((real(x1len(i+1))/2.0+1.6)*xch))),
      +             (yt+2*ych),x1str(i+1),x1len(i+1))
             endif
             xt =  xt - (0.5)*xspc*scale(1)
             if (abs(int(centre(1)-xt)).gt.(int(xspc*scale(1)/2.0)))
      +        then
                yt = yt + 1
                call MOVE(xt,yt)
                call DLINE(xt,yt+2)
             endif
          elseif ((i.eq.0).and.(abs(int(centre(1)-xt)).gt.2)) then
             call MOVE(xt,yt)
             call DLINE(xt,yt+4)
             if ((abs(int(centre(1)-xt)).gt.
      +           (int((real(x1len(i+1))/2.0+1.6)*xch))).and.
      +           (axnum.eq.1)) then
                call dwnumb(pg,
      +             (xt-(int((real(x1len(i+1))/2.0+1.6)*xch))),
      +             (yt+2*ych),x1str(i+1),x1len(i+1))
             endif
          endif
199    continue

       do 198 i = 0,(x2ind-1)
          xt = (real(i)*xspc*scale(1))+gr2(1)
          yt = int(centre(2)-2)
          if (abs(int(centre(1)-xt)).gt.(int(xspc*scale(1)/2.0))) then
             call MOVE(xt,yt)
             call DLINE(xt,yt+4)
```

```
            if ((int(xspc*scale(1)).gt.
     +         (int((real(x1len(i+1))/2.0+1.6)*xch))).and.
     +         (axnum.eq.1)) then
               call dwnumb(pg,
     +              (xt-(int((real(x2len(i+1))/2.0+1.6)*xch))),
     +              (yt+2*ych),x2str(i+1),x2len(i+1))
            endif
            xt =  xt + (0.5)*xspc*scale(1)
            if (abs(int(centre(1)-xt)).gt.(int(xspc*scale(1)/2.0)))
     +        then
               yt = yt + 1
               call MOVE(xt,yt)
               call DLINE(xt,yt+2)
            endif
         elseif ((i.eq.0).and.(abs(int(centre(1)-xt)).gt.2)) then
            call MOVE(xt,yt)
            call DLINE(xt,yt+4)
            if ((abs(int(centre(1)-xt)).gt.
     +          (int((real(x2len(i+1))/2.0+1.6)*xch))).and.
     +          (axnum.eq.1)) then
               call dwnumb(pg,
     +              (xt-(int((real(x2len(i+1))/2.0+1.6)*xch))),
     +              (yt+2*ych),x2str(i+1),x2len(i+1))
            endif
         endif
198   continue
      return
      end

C ***************************************************************************
CN    MODULE NAME       : PNUMS
CA    FUNCTION          : To generate number strings from the actual numbers.
CS    CALL SEQUENCE     : call pnums(nums,count,nstr,nlen)
CI    INPUT PARAMETERS :
C              nums(10) - (real*4) The actual numbers to be converted.
C                 count - (integer*2) The maximum element. (number of numbers)
C
CO    OUTPUT PARAMETERS:
C              nstr(10) - (character*15) The number strings.
C              nlen(10) - (integer*2) The length of each number string.
C
CG    GLOBAL VARIABLES : None.
CM    MODULES CALLED   : RJUST (asm), LENSTR (asm), COPYST (asm).
CE    ERROR CONDITIONS : ?
C
CC    COMMENTS         : The  routine  uses  list  directed I/O functions to
C                        convert the numbers into strings. The format of the
C                        conversion is dependent on  the order of the number
C                        being converted.
C
C ***************************************************************************
      subroutine pnums(nums,count,nstr,nlen)
      implicit integer*2 (C,L)
      real*4 nums(10)
      integer*2 count
      character*15 nstr(10)
      integer*2 nlen(10)

      integer*2 i,j,order,ordset,chkio,pos,numlen,len2
      real*4 temp
      character*15 outstr

      if (count.gt.0) then
         do 99 i = 1,count
            temp = nums(i)
            temp = abs(temp)
            if (temp.lt.(1.0)) then
               order = -1
               ordset = 0
               do 77 j = 1,15
                  if (ordset.eq.0) then
                     if ((temp*(10.0**(abs(order)))).lt.(1.0)) then
```

```
                  order = order - 1
               else
                  ordset = 1
               endif
            endif
77          continue
         else
            order = 0
            ordset = 0
            do 78 j = 1,15
               if (ordset.eq.0) then
                  if ((temp*(10.0**(-order))).ge.(1.0)) then
                     order = order + 1
                  else
                     ordset = 1
                  endif
               endif
78          continue
            order = order - 1
         endif
         chkio = 0
         outstr  = '                '
         nstr(i) = '                '
         if (order.gt.3) then
            temp = (nums(i)/10.0)
            write(outstr,'(g10.2)',iostat=chkio,err=299) temp
            len2 = 0
            call rjust(15,outstr)
            numlen = LENSTR(15,outstr)
            if (nums(i).lt.(0.0)) then
               call copyst(15,nstr(i),(len2+1),1,'-',0,1)
               len2 = len2 + 1
            endif
            call copyst(15,nstr(i),(len2+1),numlen,outstr,(len2+1),1)
            call copyst(15,nstr(i),(len2+2),1,'.',0,1)
            call copyst(15,nstr(i),(len2+3),numlen,outstr,
     +                  (len2+2),(numlen-len2+2))
         elseif (order.eq.3) then
            write(nstr(i),'(i5)',iostat=chkio,err=299) int(nums(i))
         elseif ((order.ge.1).and.(order.le.2)) then
            write(nstr(i),'(f6.1)',iostat=chkio,err=299) nums(i)
         elseif (order.eq.0) then
            write(nstr(i),'(f6.2)',iostat=chkio,err=299) nums(i)
         elseif (order.eq.-1) then
            write(nstr(i),'(f6.2)',iostat=chkio,err=299) nums(i)
            call rjust(15,nstr(i))
            if (nums(i).lt.(0.0)) then
               call copyst(15,nstr(i),1,1,'0',0,1)
            else
               call copyst(15,nstr(i),0,1,'0',0,1)
            endif
         elseif ((order.le.-2).and.(order.ge.-3)) then
            write(nstr(i),'(f6.3)',iostat=chkio,err=299) nums(i)
            call rjust(15,nstr(i))
            if (nums(i).lt.(0.0)) then
               call copyst(15,nstr(i),1,1,'0',0,1)
            else
               call copyst(15,nstr(i),0,1,'0',0,1)
            endif
         elseif (order.lt.-3) then
            temp = (nums(i)/10.0)
            write(outstr,'(g10.2)',iostat=chkio,err=299) temp
            len2 = 0
            call rjust(15,outstr)
            numlen = LENSTR(15,outstr)
            if (nums(i).lt.(0.0)) then
               call copyst(15,nstr(i),(len2+1),1,'-',0,1)
               len2 = len2 + 1
            endif
            call copyst(15,nstr(i),(len2+1),numlen,outstr,(len2+1),1)
            call copyst(15,nstr(i),(len2+2),1,'.',0,1)
            call copyst(15,nstr(i),(len2+3),numlen,outstr,
```

```
     +                    (len2+2),(numlen-len2+2))
            endif
            call rjust(15,nstr(i))
            nlen(i) = LENSTR(15,nstr(i))
299         if (chkio.ne.0) then
              outstr = '0.00'
            endif
99        continue
      endif
      return
      end

C ***************************************************************************
CN    MODULE NAME       : DWNUMB
CA    FUNCTION          : To print a number on a set of axes.
CS    CALL SEQUENCE     : call dwnumb(pg,x,y,num,len)
CI    INPUT PARAMETERS :
C                  pg - (integer*2) The page on which the number is to be
C                       drawn.
C                 x,y - (integer*2)  The  x,y  co-ords.  of  the  number.
C                       The x,y co-ords. represent the lower left corner.
C                 num - (character*15) The number string.
C                 len - (integer*2) The length of the string.
C
CO    OUTPUT PARAMETERS: None.
CG    GLOBAL VARIABLES : None.
CM    MODULES CALLED   : COPYST (asm), gnum (for).
CE    ERROR CONDITIONS : ?
C
CC    COMMENTS         : Extracts one character at a time from the string and
C                        draws the digit in the appropriate position  on  the
C                        requested page.
C
C ***************************************************************************
      subroutine dwnumb(pg,x,y,num,len)
      integer*2 pg,x,y
      character*15 num
      integer*2 len

      character*1 xych
      integer*2 xch,ych,i

      xch = 7
      ych = 6
      do 99 i = 1,len
      call copyst(1,xych,0,len,num,(i-1),1)
      call gnum(pg,(x+xch*(i-1)),y,xych)
99    continue
      return
      end
C ***************************************************************************
CH    REVISION HISTORY :
C     VERSION       BY                DATE        COMMENT
C     0.00          Ian Fisher        23-11-87    Finally commented.
C     0.10          The same          02-06-88    x-axes : lin,decade,octaves
C     FILEND           :
```

```
C
C     FILE              : PLOT.FOR
C                       : Set of routines to plot points on a set of axes
C                       : previously defined.
C ***************************************************************************
C     MODULE NAME       : PLOTPT
CA    FUNCTION          : To plot a point on a set of axes.
CS    CALL SEQUENCE     : call plotpt(page,xtype,x,y,centre,scale)
CI    INPUT PARAMETERS  : page - (integer*2) Page on which to plot point.
C                         xtype - (integer*2) Type of x-axes to be drawn :
C                                 1 : linear
C                                 2 : decade
C                                 3 : octave
C                         x,y   - (real*4) X,Y co-ordinates.
C                         centre - (real*8) array of centre co-ords :-
C                                           centre(1) - x centre on page,
C                                           centre(2) - y centre on page,
C                                           centre(3) - x centre on user axes,
C                                           centre(4) - y centre on user axes.
C                         scale - (real*8) array of scaling factors :-
C                                          scale(1) - x dots per axes unit.
C                                          scale(2) - y dots per axes unit.
C
CO    OUTPUT PARAMETERS: Point plotted on the page requested.
CG    GLOBAL VARIABLES  : None.
CM    MODULES CALLED    : GPAGE (asm), PLOT (asm).
CE    ERROR CONDITIONS  :
C
CC    COMMENTS          : The routine scales the given x,y co-ords for the
C                         set of axes and shifts the points according to the
C                         given page centres.
C
C ***************************************************************************
      subroutine plotpt(pg,xtype,x,y,centre,scale)
      integer*2 pg,xtype
      real*4 x,y
      real*8 centre(4),scale(2)

      integer*2 xp,yp
      real*8 xtemp,ytemp
      if (xtype.eq.1) then
         xtemp = ((x-centre(3))*scale(1)+centre(1))
      elseif (xtype.eq.2) then
         xtemp = ((log10(x)-log10(centre(3)))*scale(1)+centre(1))
      elseif (xtype.eq.3) then
         xtemp = ((log(x)-log(centre(3)))*scale(1)+centre(1))
      endif
      ytemp = int(centre(2)-(y-centre(4))*scale(2))
      call gpage(pg)
      if (pg.eq.0) then
         if (xtemp.le.(3.0)) then
            xp = 4
         elseif (xtemp.ge.(715.0)) then
            xp = 714
         else
            xp = int(xtemp)
         endif
         if (ytemp.le.(3.0)) then
            yp = 4
         elseif (ytemp.ge.(269.0)) then
            yp = 268
         else
            yp = int(ytemp)
         endif
      else
         if (xtemp.le.(3.0)) then
            xp = 4
         elseif (xtemp.ge.(715.0)) then
            xp = 714
         else
            xp = int(xtemp)
         endif
```

```
         if (ytemp.le.(3.0)) then
            yp = 4
         elseif (ytemp.ge.317) then
            yp = 317
         else
            yp = int(ytemp)
         endif
      endif
      call PLOT(xp,yp)
      return
      end

C ***************************************************************************
CN    MODULE NAME       : DRAWPT
CA    FUNCTION          : To draw a set of points to a graphics page and to
C                         join the points if required.
CS    CALL SEQUENCE     : call drawpt(page,xtype,x,y,maxpts,centre,scale,pts)
CI    INPUT PARAMETERS  : page - (integer*2) Page on which to plot point.
C                         xtype - (integer*2) Type of x-axes to be drawn :
C                                 1 : linear
C                                 2 : decade
C                                 3 : octave
C                          x,y  - (real*4) arrays of X,Y co-ordinates,
C                                          length = maxpts.
C                          maxpts - (integer*2) Length of X,Y arrays.
C                          centre - (real*8) array of centre co-ords :-
C                                            centre(1) - x centre on page,
C                                            centre(2) - y centre on page,
C                                            centre(3) - x centre on user axes,
C                                            centre(4) - y centre on user axes.
C                          scale - (real*8) array of scaling factors :-
C                                           scale(1) - x dots per axes unit.
C                                           scale(2) - y dots per axes unit.
C                          pts - (integer*2) Option to draw points or join the
C                                           points. (1 - draw points,
C                                                    2 - join the points)
CO    OUTPUT PARAMETERS: Points or lines to the page requested.
CG    GLOBAL VARIABLES  : None.
CM    MODULES CALLED    : DLINE (asm), GPAGE (asm), MOVE (asm), PLOT (asm).
CE    ERROR CONDITIONS  : ?
C
CC    COMMENTS          : The routine scales the co-ords and shifts the points
C                         onto the axes. The routine  either  draws  points or
C                         lines depending on the call parameter: pts.
C
C ***************************************************************************

      subroutine drawpt(pg,xtype,x,y,size,centre,scale,pts)
      integer*2 pg,xtype,size
      real*4 x(size),y(size)
      real*8 centre(4),scale(2)
      integer*2 pts

      integer*2 xp,yp
      real*8 xtemp,ytemp
      call GPAGE(pg)
      if (pts.eq.1) then
         do 99 i = 1,size,1
            if (xtype.eq.1) then
               xtemp = int((x(i)-centre(3))*scale(1)+centre(1))
            elseif (xtype.eq.2) then
              xtemp = int((log10(x(i))-
     +                log10(centre(3)))*scale(1)+centre(1))
            elseif (xtype.eq.(3.0)) then
               xtemp = int((log(x(i))-
     +                log(centre(3)))*scale(1)+centre(1))
            endif
            ytemp = int(centre(2)-(y(i)-centre(4))*scale(2))
            if (pg.eq.0) then
               if (xtemp.le.(3.0)) then
                  xp = 4
               elseif (xtemp.ge.(715.0)) then
```

```
                  xp = 714
               else
                  xp = int(xtemp)
               endif
               if (ytemp.le.(3.0)) then
                  yp = 4
               elseif (ytemp.ge.(269.0)) then
                  yp = 268
               else
                  yp = int(ytemp)
               endif
            else
               if (xtemp.le.(3.0)) then
                  xp = 4
               elseif (xtemp.ge.(715.0)) then
                  xp = 714
               else
                  xp = int(xtemp)
               endif
               if (ytemp.le.(3.0)) then
                  yp = 4
               elseif (ytemp.ge.317) then
                  yp = 317
               else
                  yp = int(ytemp)
               endif
            endif
         call PLOT(XP,YP)
99       continue
       else
         if (xtype.eq.1) then
            xtemp = int((x(1)-centre(3))*scale(1)+centre(1))
         elseif (xtype.eq.2) then
            xtemp = int((log10(x(1))
     +              -log10(centre(3)))*scale(1)+centre(1))
         elseif (xtype.eq.(3.0)) then
            xtemp = int((log(x(1))-log(centre(3)))*scale(1)+centre(1))
         endif
         ytemp = int(centre(2)-(y(1)-centre(4))*scale(2))
         if (pg.eq.0) then
            if (xtemp.le.(3.0)) then
               xp = 4
            elseif (xtemp.ge.(715.0)) then
               xp = 714
            else
               xp = int(xtemp)
            endif
            if (ytemp.le.(3.0)) then
               yp = 4
            elseif (ytemp.ge.(269.0)) then
               yp = 268
            else
               yp = int(ytemp)
            endif
         else
            if (xtemp.le.(3.0)) then
               xp = 4
            elseif (xtemp.ge.(715.0)) then
               xp = 714
            else
               xp = int(xtemp)
            endif
            if (ytemp.le.(3.0)) then
               yp = 4
            elseif (ytemp.ge.317) then
               yp = 317
            else
               yp = int(ytemp)
            endif
         endif
         call MOVE(xp,yp)
         do 98 i = 2,size,1
```

```
            if (xtype.eq.1) then
               xtemp = int((x(i)-centre(3))*scale(1)+centre(1))
            elseif (xtype.eq.2) then
               xtemp = int((log10(x(i))
     +                -log10(centre(3)))*scale(1)+centre(1))
            elseif (xtype.eq.(3.0)) then
               xtemp = int((log(x(i))-
     +                log(centre(3)))*scale(1)+centre(1))
            endif
            ytemp = int(centre(2)-(y(i)-centre(4))*scale(2))
            if (pg.eq.0) then
               if (xtemp.le.(3.0)) then
                  xp = 4
               elseif (xtemp.ge.(715.0)) then
                  xp = 714
               else
                  xp = int(xtemp)
               endif
               if (ytemp.le.(3.0)) then
                  yp = 4
               elseif (ytemp.ge.(269.0)) then
                  yp = 268
               else
                  yp = int(ytemp)
               endif
            else
               if (xtemp.le.(3.0)) then
                  xp = 4
               elseif (xtemp.ge.(715.0)) then
                  xp = 714
               else
                  xp = int(xtemp)
               endif
               if (ytemp.le.(3.0)) then
                  yp = 4
               elseif (ytemp.ge.317) then
                  yp = 317
               else
                  yp = int(ytemp)
               endif
            endif
            call DLINE(xp,yp)
98       continue
      endif
      return
      end

C ***************************************************************************
CH    REVISION HISTORY :
C     VERSION       BY                DATE        COMMENT
C     0.00          Ian Fisher        23-11-87    Finally commented.
C     0.10          The same          02-06-88    Xaxes: lin,decade,octave
C     FILEND           :
```

```
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 > 10 : 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(20)

      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 MIN0
      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
```

```
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 = MIN0(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 ***************************************************************************
CN    MODULE NAME       : CMULT
CA    FUNCTION          : To multiply 2 complex matrices together.
CS    CALL SEQUENCE     : err = cmult(ma,na,ar,ai,mb,nb,br,bi,cr,ci)
CI    INPUT PARAMETERS :        ma - (integer*2) Row dimension of matrix A.
C                               na - (integer*2) Column dimension of matrix A.
C                         ar(ma,na) - (real*4) Real part of matrix A.
C                         ai(ma,na) - (real*4) Imaginary part of matrix A.
C                               mb - (integer*2) Row dimension of matrix B.
C                               nb - (integer*2) Column dimension of matrix B.
C                         br(mb,nb) - (real*4) Real part of matrix B.
C                         bi(mb,nb) - (real*4) Imaginary part of matrix B.
C
CO    OUTPUT PARAMETERS:       err - (integer*2) Error return :-
C                                                 0 = Multiplication OK.
C                                                -1 = Incompatible indices.
C                         cr(ma,nb) - (real*4) Real part of resultant matrix.
C                         ci(ma,nb) - (real*4) Imaginary  part  of  resultant
C                                              matrix.
C
CG    GLOBAL VARIABLES : None.
CM    MODULES CALLED   : None.
CE    ERROR CONDITIONS : ?
CC    COMMENTS         : Does simple matrix indices check before doing normal
C                        matrix multiplication.
C ***************************************************************************
      integer*2 function cmult(ma,na,ar,ai,mb,nb,br,bi,cr,ci)
      integer*2 ma,na
      real*4 ar(ma,na),ai(ma,na)
      integer*2 mb,nb
      real*4 br(mb,nb),bi(mb,nb),cr(ma,nb),ci(ma,nb)

      real*4 tempr,tempi

      if (na.eq.mb) then
         do 99 i = 1,nb
            do 98 j = 1,ma
               cr(j,i) = 0.0
               ci(j,i) = 0.0
               do 97 k = 1,na
                  cr(j,i) = cr(j,i) + ar(j,k)*br(k,i) - ai(j,k)*bi(k,i)
                  ci(j,i) = ci(j,i) + ar(j,k)*bi(k,i) + ai(j,k)*br(k,i)
97             continue
98          continue
99       continue
         cmult = 0
      else
         cmult = -1
      endif
      return
      end

C ***************************************************************************
CN    MODULE NAME       : COMINV
CA    FUNCTION          : To compute the inverse of a compex matrix.
CS    CALL SEQUENCE     : call cominv(n,br,bi,c)
CI    INPUT PARAMETERS :
C                     n - (integer*2) Order of the system.
C               br(n*n)
C               bi(n,n) - (real*4) The complex matrix to be inverted.
C
CO    OUTPUT PARAMETERS:
C               br(n*n)
C               bi(n,n) - (real*4) The complex matrix to be inverted.
C                     c - (complex*8) Return value ???????
```

```
CG    GLOBAL VARIABLES : None.
CM    MODULES CALLED   : None.
CE    ERROR CONDITIONS : ?
C
CC    COMMENTS         : Inverts a complex matrix using the Gauss-Jordan of
C                        elimination with complete pivoting.
C ***************************************************************************
      SUBROUTINE COMINV(N,BR,BI,C)

      INTEGER*2 N
      REAL*4 BR(*),BI(*)
      COMPLEX*8 C

      INTEGER*2  IR(10),IC(10)
C               IR,IC are the pivot point of the matrix
      REAL*8 AR(225),AI(225),TEMPR,TEMPI
      COMPLEX*16 D,HOLD,CAL1,CAL2
      INTEGER*2  I,J,K,L,II,JJ,III,M,IJ,IIJ,IT,KI,KJ,LP

      DO 101 I = 1,N*N
            AR(I) = BR(I)
            AI(I) = BI(I)
101   CONTINUE
      DO 10 I = 1,N
         IR(I) = I
         IC(I) = I
10    CONTINUE
      D=(1.,0.)
      M=-N
      DO 70 I = 1,N
         M = M+N+1
         IF (I.NE.N) THEN
C             BIG=cdABS(A(M))
            BIG = SQRT(AR(M)*AR(M) + AI(M)*AI(M))
            II = I
            JJ = I
            DO 20 K = I,N
               LP = (I-2)*N+K
               DO 20 L = I,N
                  LP = LP+N
                  IF ((SQRT(AR(LP)*AR(LP) + AI(LP)*AI(LP))).GT.BIG) THEN
C                     BIG = CdABS(A(LP))
                     BIG = SQRT(AR(LP)*AR(LP) + AI(LP)*AI(LP))
                     II = K
                     JJ = L
                  ENDIF
20          CONTINUE
22          IF (II.NE.I) THEN
               IJ = I-N
               IIJ = II-N
               DO 25 J = 1,N
                  IJ = IJ+N
                  IIJ = IIJ+N
                  TEMPR = AR(IJ)
                  TEMPI = AI(IJ)
                  AR(IJ) = AR(IIJ)
                  AI(IJ) = AI(IIJ)
                  AR(IIJ) = TEMPR
                  AI(IIJ) = TEMPI
25             CONTINUE
               IT = IR(I)
               IR(I) = IR(II)
               IR(II) = IT
               D = -D
            ENDIF
            IF (JJ.NE.I) THEN
               IJ = (I-1)*N
               IIJ = (JJ-1)*N
               DO 35 J = 1,N
                  IJ = IJ+1
                  IIJ = IIJ+1
                  TEMPR = AR(IJ)
```

```
                TEMPI = AI(IJ)
                AR(IJ) = AR(IIJ)
                AI(IJ) = AI(IIJ)
                AR(IIJ) = TEMPR
                AI(IIJ) = TEMPI
35            CONTINUE
              IT = IC(JJ)
              IC(JJ) = IC(I)
              IC(I) = IT
              D = -D
          ENDIF
        ENDIF

        CAL1 = CMPLX(AR(M),AI(M))
        CAL1 = D*CAL1
        IF (CDABS(CAL1).lt.(1.d-150)) THEN
           C=(0,0)
           RETURN
        ENDIF
        CAL1 = CMPLX(AR(M),AI(M))
        D = D*CAL1
        TEMP = CAL1
        AR(M) = 1.0
        AI(M) = 0.0
        IJ = I-N
        DO 50 J = 1,N
           IJ = IJ+N
           AR(IJ) = AR(IJ)/TEMP
           AI(IJ) = AI(IJ)/TEMP
50      CONTINUE
        KI = (I-1)*N
        DO 70 K = 1,N
           KI = KI+1
           IF (K.NE.I) THEN
              CAL1 = CMPLX(AR(KI),AI(KI))
              AR(KI) = 0.0
              AI(KI) = 0.0
              KJ = K-N
              IJ = I-N
              DO 60 J = 1,N
                 KJ = KJ+N
                 IJ = IJ+N
                 CAL2 = CMPLX(AR(IJ),AI(IJ))
                 HOLD = -CAL1*CAL2
                 CAL2 = CMPLX(AR(KJ),AI(KJ))
                 CAL2 = CAL2 + HOLD
                 AR(KJ) = REAL(CAL2)
                 AI(KJ) = AIMAG(CAL2)
60            CONTINUE
           ENDIF
70    CONTINUE
72    CONTINUE
      DO 80 I = 1,N
        IF (IR(I).EQ.I) GOTO 80
           K = IR(I)
           JI = (I-1)*N
           JK = (K-1)*N
           DO 85 J = 1,N
              JI = JI+1
              JK = JK+1
              TEMPR = AR(JI)
              TEMPI = AI(JI)
              AR(JI) = AR(JK)
              AI(JI) = AI(JK)
              AR(JK) = TEMPR
              AI(JK) = TEMPI
85         CONTINUE
           IR(I) = IR(K)
           IR(K) = K
           GOTO 72
80    CONTINUE
82    CONTINUE
```

```
      DO 90 I = 1,N
         IF (IC(I).EQ.I) GOTO 90
            K = IC(I)
            IJ = I-N
            KJ = K-N
            DO 75 J = 1,N
               IJ = IJ+N
               KJ = KJ+N
               TEMPR = AR(IJ)
               TEMPI = AI(IJ)
               AR(IJ) = AR(KJ)
               AI(IJ) = AI(KJ)
               AR(KJ) = TEMPR
               AI(KJ) = TEMPI
75          CONTINUE
            IC(I) = IC(K)
            IC(K) = K
            GOTO 82
90    CONTINUE
      DO 100 I = 1,N*N
         BR(I) = AR(I)
         BI(I) = AI(I)
100   CONTINUE
      C=D
      RETURN
      END

C ***************************************************************************
CH    REVISION HISTORY :
C     VERSION       BY                DATE        COMMENT
C     1.00          Ian Fisher        23/06/88    Creation.
C     FILEND           :
```

```
$NOLIST
C
C     INCLUDE FILE : SYSMAT.INC
C ***************************************************************************
CN    COMMON NAME  : kmats
CA    DESCRIPTION  : K(s) polynomial matrix.
CP    PARAMETERS   :
C            order - Order of the system.
C             kmat - The K(s) matrix : elements :-
C
C               mat(row,col,1,order) - Numerator polynomial.
C               mat(row,col,2,order) - Denominator polynomial.
C
C                  mat(row,col,1,13) - Order of numerator.
C                  mat(row,col,2,13) - Order of denominator.
C
C                  mat(row,col,1,12) - Element dead time.
C
C                   mat(row,col,1,1) - Numerator element order 0.
C                  mat(row,col,1,11) - Numerator element order 10.
C ***************************************************************************
      integer*2 order
      real*4 kmat
      dimension kmat(10,10,2,13)
      common /kmats/ kmat

C ***************************************************************************
CN    COMMON NAME  : gmats
CA    DESCRIPTION  : G(s) polynomial matrix.
CP    PARAMETERS   :
C             gmat - The G(s) matrix : elements see 'kmat' above.
C ***************************************************************************
      real*4 gmat
      dimension gmat(10,10,2,13)
      common /gmats/ gmat

C ***************************************************************************
CN    COMMON NAME  : k2mats
CA    DESCRIPTION  : Temporary K(s) polynomial matrix.
CP    PARAMETERS   :
C           k2mat - Temporary K(s) matrix : elements see 'kmat' above.
C ***************************************************************************
      real*4 k2mat
      dimension k2mat(10,10,2,13)
      common /k2mats/ k2mat

C ***************************************************************************
CN    COMMON NAME  : mats2
CA    DESCRIPTION  : Temporary storage for matrix operations.
CP    PARAMETERS   :
C            calcr - Matrix calculation area : real part.
C            calci - Matrix calculation area : imaginary part.
C           calcr2 - temporary storage for real part.
C           calci2 - temporary storage for imaginary part.
C           calcr3 - temporary storage for real part.
C           calci3 - temporary storage for imaginary part.
C             alfr - Eigen value and vector alfa array : real part.
C             alfi - Eigen value and vector alfa array : imaginary part.
C             beta - Eigen value and vector beta array.
C               zr - Eigen vector array : real part.
C               zi - Eigen vector array : imaginary part.
C           matok - Error return flag from eigen routines.
C ***************************************************************************
      real*4 calcr(100),calci(100)
      real*4 calcr2(100),calci2(100),calcr3(100),calci3(100)
      real*4 alfr(10),alfi(10),beta(10)
      real*4 zr(100),zi(100)
      logical*2 matok
      common /mats2/ order,calcr,calci,calcr2,calci2,calcr3,calci3,
     +               alfr,alfi,beta,zr,zi,matok

C ***************************************************************************
```

```
$NOLIST
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            gname - G(s) matrix title.
C            kname - K(s) matrix title.
C            lname - ?
C            prjnm - Project title.
C           engnms - Engineer or design team name.
C ***************************************************************************
      dimension inpnms(10),outnms(10)
      character*25 inpnms,outnms,gname,kname,lname,prjnm,engnms
      common /sysnms/ inpnms,outnms,gname,kname,lname,prjnm,engnms

C ***************************************************************************
$LIST
```

```
$NOLIST
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 ***************************************************************************
$LIST
```

```
$NOLIST
C
C     INCLUDE FILE : PLOTFR
C ***************************************************************************
CN    COMMON NAME  : plotfr
CA    DESCRIPTION  : Calculated plot frequencies.
CP    PARAMETERS   :
C         freqs - The calculated frequencies.
C ***************************************************************************

      real*4 freqs(300),evalr1(10),evali1(10),evalr2(10),evali2(10)
      common /plotfr/ freqs,evalr1,evali1,evalr2,evali2

C ***************************************************************************
$LIST
```

```
$NOLIST
C
C     INCLUDE FILE : PLOTPG.INC
C ***************************************************************************
CN    COMMON NAME  : plotpg
CA    DESCRIPTION  : Graph and plot parameters.
CP    PARAMETERS   :
C            ftype - Frequency type (1=Hz, 2=rads/sec, 3=rads/hour).
C           fstart - Start frequency (rad/sec).
C            fstop - Stop frequency (rads/sec).
C           fstar2 - Temporary start frequency for editing.
C           fstop2 - Temporary stop frequency for editing.
C           inctyp - Increment type (1=points/range, 2=points/decade,
C                                    3=points/octave, 4=frequency step).
C           increm - Frequency step if 'inctyp' = 4 (rads/sec).
C           incre2 - Temporary frequency step while editing.
C           incpts - Number of points if 1 <= 'inctyp' <= 3.
C
C           plset1 - Setting 1 : Loci plot page.
C           pmset1 - Setting 1 : Bode and misalignment angle plot page.
C           plset2 - Setting 2 : Loci plot page.
C           pmset2 - Setting 2 : Bode and misalignment angle plot page.
C            pmat1 - Setting 1 : System configuration (1=G(s), 2=Q(s)).
C            pmat2 - Setting 2 : System configuration (1=G(s), 2=Q(s)).
C           fback1 - Setting 1 : Feedback configuration (1=open, 2=closed).
C           fback2 - Setting 2 : Feedback configuration (1=open, 2=closed).
C
C            mwmax - Bode + misalignment : Maximum width (dots).
C           mwstrt - Bode + misalignment : X start point.
C            mhmax - Bode + misalignment : Maximum height (dots).
C           mhstrt - Bode + misalignment : Y start point.
C           mwidth - Bode + misalignment : No. of plots across the page.
C           mheigh - Bode + misalignment : No. of plots down the page.
C            mxdel - Bode + misalignment : Width of one plot (dots).
C            mydel - Bode + misalignment : Height of one plot (dots).
C
C            lwmax - Loci : Maximum width (dots).
C           lwstrt - Loci : X start point.
C            lhmax - Loci : Maximum height (dots).
C           lhstrt - Loci : Y start point.
C           lwidth - Loci : No. of plots across the page.
C           lheigh - Loci : No. of plots down the page.
C            lxdel - Loci : Width of one plot (dots).
C            lydel - Loci : Height of one plot (dots).
C
C ***************************************************************************
      integer*2 ftype
      real*4 fstart,fstop,fstar2,fstop2
      integer*2 inctyp
      real*4 increm,incre2
      integer*2 incpts
      integer*2 plset1,pmset1,plset2,pmset2,pmat1,pmat2,fback1,fback2
      integer*2 mwmax,mwstrt,mhmax,mhstrt,mwidth,mheigh,mxdel,mydel
      integer*2 lwmax,lwstrt,lhmax,lhstrt,lwidth,lheigh,lxdel,lydel

      common /plotpg/ ftype,fstart,fstop,fstar2,fstop2,inctyp,
     +                increm,incre2,incpts,plset1,pmset1,plset2,pmset2,
     +                pmat1,pmat2,fback1,fback2,
     +                mwmax,mwstrt,mhmax,mhstrt,mwidth,mheigh,mxdel,
     +                mydel,lwmax,lwstrt,lhmax,lhstrt,lwidth,lheigh,
     +                lxdel,lydel

C ***************************************************************************
$LIST
```

```
$NOLIST
C
C     INCLUDE FILE : PLOCI.INC
C ***************************************************************************
CN    COMMON NAME  : ploci
CA    DESCRIPTION  : Arrays for characteristic loci axes settings.
CP    PARAMETERS   :
C            xllim - X axes bounds.
C            yllim - Y axes bounds.
C           lscale - Axes scales (dots/unit).
C          lcentre - Axes centre (dots and units).
C           lgraph - Temporary array used to draw axes.
C ***************************************************************************

      real*8 xllim(3,10),yllim(3,10),lscale(2,10),lcentre(4,10)
      integer*2 lgraph(4)
      common /ploci/ xllim,yllim,lscale,lcentre,lgraph

C ***************************************************************************
CN    COMMON NAME  : pbode
CA    DESCRIPTION  : Bode plot axes settings.
CP    PARAMETERS   :
C            xblim - Magnitude axes limits.
C            yblim - Frequency axes limits.
C           bscale - Axes scales (dots/unit).
C          bcentre - Axes centre (dots and units).
C           bgraph - Temporary array used to draw axes.
C ***************************************************************************
      real*8 xblim(3,10),yblim(3),bscale(2,10),bcentre(4,10)
      integer*2 bgraph(4)
      common /pbode/ xblim,yblim,bscale,bcentre,bgraph

C ***************************************************************************
CN    COMMON NAME  : pmisa
CA    DESCRIPTION  : Misalignment angles axes settings.
CP    PARAMETERS   :
C            xmlim - Magnitude axes limits.
C            ymlim - Frequency axes limits.
C           mscale - Axes scales (dots/unit).
C          mcentre - Axes centre (dots and units).
C           mgraph - Temporary array used to draw axes.
C ***************************************************************************
      real*8 xmlim(3,10),ymlim(3),mscale(2,10),mcentre(4,10)
      integer*2 mgraph(4)
      common /pmisa/ xmlim,ymlim,mscale,mcentre,mgraph

C ***************************************************************************
$LIST
```

```
$NOLIST
C
C     INCLUDE FILE : KEYS.INC
C ***************************************************************************
CN    COMMON NAME  : Not a common block.
CA    DESCRIPTION  : Keyboard constants.
CP    PARAMETERS   :
C

C ***************************************************************************
      integer*2 upk,downk,leftk,rightk,homek,endk,pgupk,pgdnk,
     +          esck,retk,tabk,rtabk
      parameter (upk    = 72,
     +           downk  = 0080,
     +           leftk  = 0075,
     +           rightk = 0077,
     +           homek  = 0071,
     +           endk   = 0079,
     +           pgupk  = 0073,
     +           pgdnk  = 0081,
     +           esck   = 6912,
     +           retk   = 3328,
     +           tabk   = 2304,
     +           rtabk  = 0015)

C ***************************************************************************
$LIST
```

```
$NOLIST
C
C     INCLUDE FILE : TIME.INC
C ***************************************************************************
CN    COMMON NAME  : time
CA    DESCRIPTION  : Part of the state variables for the time simulation.
CP    PARAMETERS   :
C                  gx - State variables of G(s) for time simulation.
C                       gx(i,j,1,k) - element Gij, state 'x', order k.
C                       gx(i,j,2,k) - element Gij, state 'x0', order k.
C ***************************************************************************
      real*4 gx(10,10,2,11)
      common /time/ gx

C ***************************************************************************
CN    COMMON NAME  : time1
CA    DESCRIPTION  : Part of the state variables for the time simulation.
CP    PARAMETERS   :
C                  kx - State variables of K(s) for time simulation.
C                       kx(i,j,1,k) - element Kij, state 'x', order k.
C                       kx(i,j,2,k) - element Kij, state 'x0', order k.
C ***************************************************************************
      real*4 kx(10,10,2,11)
      common /time1/ kx

C ***************************************************************************
CN    COMMON NAME  : time2
CA    DESCRIPTION  : Part of the state variables for the time simulation.
CP    PARAMETERS   :
C             ysim(i,j) - Plant simulated outputs, i = i th output.
C                         j - elements used to implement delays.
C               esim(i) - Error signals - i th error signal (Ri-Yi)
C               rsim(i) - Setpoint values - i th setpoint.
C               yout(i) - Output from the system, G(s), polynomials.
C               uout(i) - Output from the controller, K(s), polynomials.
C                    ti - Current time.
C                    dt - Time increment.
C                 tend - Time limit.
C                 c1(i) - Runga-Kutta constants.
C                 c2(i) - Runga-Kutta constants.
C           yptr(i,j,k) - Delay array pointer table for G(s) :
C                         i,j - position of element.
C                           k - 1 : output pointer
C                               2 : input pointer.
C                 cloop - Closed(1) or open(0) loop state.
C                coninc - Controller included(1) or excluded(0).
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 ysim(12000),esim(10),rsim(10),yout(10),uout(10),
     +       ti,dt,tend,c1(4),c2(4)
      integer*2 yptr(10,10,2),cloop,coninc,stpinp(5)
      real*4 stpdat(5),stpinc(5)
      integer*2 twmax,twstrt,thmax,thstrt,twidth,theigh,txdel,tydel
      common /time2/ ysim,esim,rsim,yout,uout,ti,dt,tend,
     +               c1,c2,yptr,cloop,coninc,stpinp,stpdat,stpinc,
     +               twmax,twstrt,thmax,thstrt,twidth,theigh,
     +               txdel,tydel

C ***************************************************************************
CN    COMMON NAME  : time3
CA    DESCRIPTION  : Part of the state variables for the time simulation.
CP    PARAMETERS   :
```

```
C              usim(i,j) - Controller simulated inputs to plant,
C                          i = i th output.
C                          j - elements used to implement delays.
C            yptr(i,j,k) - Delay array pointer table for G(s) :
C                          i,j - position of element.
C                            k - 1 : output pointer
C                                2 : input pointer.
C ***************************************************************************
      real*4 usim(12000)
      integer*2 uptr(10,10,2)
      common /time3/ usim,uptr

C ***************************************************************************
CN    COMMON NAME  : time4
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 /time4/ xtlim,ytlim,tscale,tcentr,tgraph

C ***************************************************************************
$LIST
```

```
$NOLIST
C
C     INCLUDE FILE : K3MAT.INC
C ***************************************************************************
CN    COMMON NAME  : k3mats
CA    DESCRIPTION  : Temporary polynomial matrix.
CP    PARAMETERS   :
C           k3mat - Temporary matrix : elements see 'kmat' above.
C ***************************************************************************
      real*4 k3mat
      dimension k3mat(10,10,2,13)
      common /k3mats/ k3mat

C ***************************************************************************
$LIST
```

```
; MODULE : HELP routines.
; ************************

TITLE HELP routines and utilities.
;     ************************

; ***************************************************************************
; REVISION HISTORY :
; VERSION       BY              DATE        COMMENT
;    1.00       Ian Fisher      23/06/88    Creation.
;
; ***************************************************************************

; Procedures.
; ***********
            PUBLIC      INKEY2      ; HELP function keyboard function.
            PUBLIC      FLIPG1      ; Flip screen memory into buffer.

; INT 21H : Function calls and returns.
; *************************************
CHAR_TRUE   EQU         0FFH        ; Return if a char in buffer.
FLUSH_KB    EQU         00CH        ; Function : flush buffer.
NO_PROCESS  EQU         000H        ; Function : no function,
                                    ;            while flushing.
READ_KB     EQU         008H        ; Function : read ASCII char.
STATUS_KB   EQU         00BH        ; Function : read keyb. status.

; General data segment.
; *********************
DATA        SEGMENT PUBLIC 'DATA'
DATA        ENDS

; Menu data areas.
; ****************
MENU_DATA   SEGMENT WORD PUBLIC 'FAR_DATA'
PAGE1       DB          32*1024 DUP(0)
            DB          0
MENU_DATA   ENDS

DGROUP      GROUP DATA
CODE        SEGMENT 'CODE'
            ASSUME CS:CODE
            ASSUME DS:DGROUP
            ASSUME ES:DGROUP

;---------------------------------------------------------------------------
FLIPG1      PROC FAR
; Function: To flip page out/in of graphics page 1.
; Inputs:   FORTRAN : call FLIPG1(op)
;                op - (integer*2) The operation : 1 - flip page 1 into memory.
;                                                 2 - flip memory into page 1.
; Outputs:  None.
; Calls:    None.
; Destroys: Flags.
;
; Description: The routine does a 32K dump from screen memory into a buffer
;              or vica-versa depending on user request.
;
;---------------------------------------------------------------------------
            PUSH        BP                      ; Save the registers.
            PUSH        DS                      ;
            PUSH        ES                      ;
            PUSH        AX                      ;
            PUSH        BX                      ;
            PUSH        CX                      ;
            PUSH        DX                      ;
            MOV         BP,SP                   ;
            LES         BX,DWORD PTR [BP+18]    ; Load the operation number.
            MOV         AX,WORD PTR ES:[BX]     ;
            CMP         AX,2                    ;
            JE          FLIP_IN                 ; If (operation = 1) then
            MOV         BX,0B800H               ;    Set source address in
```

```
            MOV         ES,BX                   ;         ES:BX
            MOV         BX,0                    ;
            MOV         AX,MENU_DATA            ;     Set destination address in
            MOV         DS,AX                   ;         DS:AX
            MOV         AX,OFFSET MENU_DATA:PAGE1;
            JMP         FL_EXIT                 ;
FLIP_IN:                                        ; Elseif (operation = 2) then
            MOV         BX,MENU_DATA            ;     Set source address in
            MOV         ES,BX                   ;         ES:BX
            MOV         BX,OFFSET MENU_DATA:PAGE1;
            MOV         AX,0B800H               ;     Set destination address in
            MOV         DS,AX                   ;         DS:AX
            MOV         AX,0                    ;
FL_EXIT:                                        ; Endif
            MOV         CX,(16*1024)            ; Set count to 32K.
NEXT_FLIP:                                      ; While (count > 0)
            MOV         DX,WORD PTR ES:[BX]     ;   Load word from source.
            MOV         WORD PTR ES:[BX],0      ;   Set source to zero.
            ADD         BX,2                    ;   Increment source ptr.
            XCHG        AX,BX                   ;
            MOV         WORD PTR DS:[BX],DX     ;   Load word into destination.
            ADD         BX,2                    ;   Increment destination ptr.
            XCHG        AX,BX                   ;
            LOOP        NEXT_FLIP               ; Endwhile.
                                                ;
            POP         DX                      ; Restore registers.
            POP         CX                      ;
            POP         BX                      ;
            POP         AX                      ;
            POP         ES                      ;
            POP         DS                      ;
            POP         BP                      ;
            RET         04H                     ; Pop parameters and exit.
FLIPG1      ENDP                                ;

;-------------------------------------------------------------------------------
INKEY2      PROC FAR
; NOTE : This routine is a duplicate of GETKEY. This routine was created for
;        the HELP facility as the GETKEY routine  is  not re-entrant and the
;        HELP facility requires keyboard inputs.
;
; Function: FORTRAN Version :To read in a single character.
; Inputs:   FORTRAN: key = INKEY2()
;           key - (integer*2) Return value from routine.
; Outputs:  AX  - contains  the  return  code  or return character for
;                 the calling routine.
; Calls:    INT21H.
; Destroys: Flags.
;
; Description: Routine returns the ASCII (upper byte or shifted left 8 times)
;              code of the key typed (the char is  not echoed to the screen).
;              The  routine  first flushes the keyboard buffer before waiting
;              for a key to be typed.
;
;-------------------------------------------------------------------------------
            PUSH        BP                      ; Save the registers.
            PUSH        ES                      ;
            PUSH        BX                      ;
            MOV         BP,SP                   ;
                                                ;-----------------------------
            MOV         AH,FLUSH_KB             ;   Set function to flush the
            MOV         AL,NO_PROCESS           ;   buffer and do nothing else.
            INT         21H                     ;   Execute the function.
                                                ;
            MOV         AH,READ_KB              ;   Set function to read char.
            INT         21H                     ;   Execute function.
                                                ;   (Function waits for key to
                                                ;    entered before returning)
READ_CH:    CMP         AL,0                    ;   Has ALT, Function or cursor
            JNE         ONE_CH                  ;   key been entered ?
            MOV         AH,STATUS_KB            ;    Keyboard status function.
            INT         21H                     ;    Get status.
```

```
; MODULE : Menu routines.
; ***********************

TITLE Handles the output and control of menus.
;     ***************************************

; ****************************************************************************
; REVISION HISTORY :
; VERSION       BY                  DATE        COMMENT
;    1.00       Ian Fisher          23/06/88    Creation.
;
; ****************************************************************************

; Routines.
; *********
            PUBLIC      DOMENU                  ; To print out and control
                                                ; menus.

            EXTRN       ERTONE:FAR              ; Sound error tone.
            EXTRN       GETKEY:FAR              ; To read in keys from the
                                                ; keyboard.
            EXTRN       INT10:FAR               ; Calls the interrupt routine.
            EXTRN       PRINTCH:FAR             ; Prints a character on screen.
            EXTRN       PRSTR:FAR               ; Prints a str. on the screen.
            EXTRN       TO_UPPER:FAR            ; Converts lower to upper case.

; Variables.
; **********
            EXTRN       XPOS:WORD               ; X co-ord. of text cursor.
            EXTRN       YPOS:WORD               ; Y co-ord. of text cursor.

; Keyboard routine function numbers.
; **********************************
GET_ASCII   EQU         01H         ; Return ASCII key.
FLUSH_ASCII EQU         02H         ; Flush buffer.
KEY_HIT     EQU         03H         ; Check if key hit.
IF_HIT_GET  EQU         04H         ; If key hit, then get key.

; 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.

; Escape characters.
; ******************
ESCD        EQU         0441BH      ; Set display page.
ESCJ        EQU         04A1BH      ; Move cursor to x,y.
ESCK        EQU         04B1BH      ; Delete x number of characters.
ESCL        EQU         04C1BH      ; Set intensity to x.
ESCP        EQU         0501BH      ; Set page to which data is written.

; Keys from keyboard.
; *******************
ESC_KEY     EQU         1B00H       ; The ESC key.
CR_KEY      EQU         0D00H       ; The ENTER or RETURN key.
LDEL_KEY    EQU         0800H       ; The left delete key.
REV_TAB     EQU         000FH       ; The REVERSE TAB key
SPC_KEY     EQU         2000H       ; The space bar.
TAB_KEY     EQU         0900H       ; The TAB key.

; Routine constants and flags.
; ****************************
DIR_FWD     EQU         01          ; Highlight next option.
DIR_REV     EQU         02          ; Highlight previous option.
NORMATR     EQU         01          ; Normal intensity.
NO_MENU     EQU         0FFH        ; No error when printing new menu.
PAGE0       EQU         00          ; Select page 0.
PAGE1       EQU         01          ; Select page 1.
XORATR      EQU         02          ; XOR type intensity.
```

```
ZERO        EQU         00          ; Just zero.

; Menu and character constants.
; *****************************
MAXLEN      EQU         12          ; Maximum length of option.
SCR_X       EQU         2           ; X co-ord of command headers.
XCH         EQU         09          ; Character width in graphics mode.
XMENLIM     EQU         571         ; X co-ord limit for menu.
XMENST      EQU         92          ; X co-ord for start of menu.
XPOSLIM     EQU         710         ; X limit for printing chars.
YBRIEF      EQU         331         ; Y co-ord for brief.
YCH         EQU         14          ; Character height in graphics mode.
YMENST      EQU         287         ; Y co-ord for start of menu.
YPOSLIM     EQU         347         ; Y limit for printing chars.

; GRAPHIX routine function numbers.
; *********************************
BLKFIL      EQU         04AH        ; Fill a block.
CLRSCR      EQU         042H        ; Clear the screen.
DISP        EQU         045H        ; Display a page.
LINE        EQU         049H        ; Draw a line.
GMODE       EQU         040H        ; Define graphics mode.
GPAGE       EQU         043H        ; Page to which data is written.
LEVEL       EQU         044H        ; Intensity level.
CURSOR      EQU         048H        ; Move the imaginary cursor.
TEXT        EQU         04BH        ; Output text to the page.
TMODE       EQU         041H        ; To switch to text mode.

; General data segment.
; *********************
DATA        SEGMENT PUBLIC 'DATA'
DATA        ENDS

; Menu data areas.
; ****************
TABLE_DATA  SEGMENT WORD PUBLIC 'FAR_DATA'
TABLE_DATA  ENDS

; All the working data required by menus.
; ***************************************
MENU_DATA   SEGMENT WORD PUBLIC 'FAR_DATA'
MENU_ADR    DW          0           ; Start address of current menu.
OPT_COUNT   DW          0           ; Current position in the menu.
OPT_MAX     DW          0           ; Max. number of options.
OPT_TABLE   DW          200 DUP(0)  ; Table containg : First char of option.
                                    ;                  X co-ord of option.
                                    ;                  Y co-ord of option.
                                    ;                  Length of option.
                                    ;                  Start address of brief.
TEMPLEN     DW          0           ; Temporary : Length of option.
TEMPOFF     DW          0           ;             Address of brief.
TEMPX       DW          0           ;             X co-ord of start of option.
TEMPY       DW          0           ;             Y co-ord of start of option.
TEMP_CH     DW          0           ;             Option first character.
TEMP_COUNT  DW          0           ;             Position within menu.

BRIEF_CLR   DW          ESCJ,XMENST,YBRIEF,ESCK,68,ESCJ,XMENST,YBRIEF
            DB          0
CLR_MENU    DW          ESCJ,XMENST,YMENST,ESCK,68,ESCJ,XMENST,(YMENST+YCH)
            DW          ESCK,68,ESCJ,XMENST,YBRIEF,ESCK,68,ESCJ,XMENST,YMENST
            DB          0
RESET_ATR   DW          ESCL,NORMATR,0
SET_ATR     DW          ESCL,XORATR,0
SET_MENU    DW          ESCD,0,ESCP,0,ESCL,NORMATR,0
MENU_DATA   ENDS

; ****************************************************************************
DGROUP      GROUP DATA
CODE        SEGMENT 'CODE'
            ASSUME CS:CODE
            ASSUME DS:DGROUP
            ASSUME ES:DGROUP
```

```
;-------------------------------------------------------------------------
GET_MENU    PROC FAR
; Function: To obtain the start address of the new menu text.
; Inputs:   AX - number of the menu to be printed out.
; Outputs:  AX - error return code (No menu = 1, No error = 0)
;           BX - address of the menu text.
; Calls:    None.
; Destroys: BX, Flags.
;
; Description: Compares  input  number (AX) with known menu numbers and if a
;              corresponding number is found the offset from _DATA is placed
;              in BX and AX is set to zero.
;              If no menu is found, then  BX is set to zero and AX is set to
;              one.
;-------------------------------------------------------------------------

            ASSUME      DS:TABLE_DATA           ;
            PUSH        DS                      ;
            MOV         BX,TABLE_DATA           ;
            MOV         DS,BX                   ;
            MOV         BX,ZERO                 ; Load start address of table.
NEXT_ADR:   CMP         AX,WORD PTR DS:[BX]     ; Check against item in table.
            JE          GET_ADR                 ; If equal, get the address.
            CMP         WORD PTR DS:[BX],EOM    ; Else, check if the end of
            JE          ADR_ERROR               ;       the table.
            ADD         BX,4                    ; Check the next table entry.
            JMP         NEXT_ADR                ;
GET_ADR:    MOV         AX,WORD PTR DS:[BX+2]   ; Load the menu address from
            MOV         BX,AX                   ; the table.
            MOV         AX,ZERO                 ; Set error return code.
            JMP         GET_EXIT                ; Exit.
ADR_ERROR:  MOV         AX,NO_MENU              ; Set error return code.
            XOR         BX,BX                   ; Zero address register.
GET_EXIT:                                       ;
            POP         DS                      ;
            ASSUME      DS:DGROUP               ;
            RET                                 ; End.
GET_MENU    ENDP

;-------------------------------------------------------------------------
PR_MENU     PROC FAR
; Function: To print out a menu on the screen.
; Inputs:   AX - number of menu to print out.
; Outputs:  Menu on the screen.
;           AX - Return code to caller (No menu = 1, No error = 0).
; Calls:    CLR_MENU, GET_MENU, PRSTR, PRINTCH, INT10.
; Destroys: AX, flags
;
; Description: First clears the last menu from the screen and then prints out
;              the menu requested by the calling procedure. This routine also
;              sets  up  the  menu  table,  which  aids the menu handler when
;              manipulating the options.
;
;-------------------------------------------------------------------------
            ASSUME      ES:MENU_DATA            ;
            PUSH        BX                      ; Save the registers.
            PUSH        CX                      ;
            PUSH        DX                      ;
            PUSH        DI                      ;
            PUSH        BP                      ;
            CALL        FAR PTR GET_MENU        ; Get the menu text start addr.
            MOV         WORD PTR ES:MENU_ADR,BX ;
            CMP         AX,NO_MENU              ; If not a valid menu, then
            JNE         NOTPR_EXIT              ;
            JMP         PR_EXIT                 ;    exit.
NOTPR_EXIT: MOV         BX,OFFSET MENU_DATA:SET_MENU; else,
            CALL        FAR PTR PRSTR           ;         Initialise menu page.
            MOV         BX,OFFSET MENU_DATA:CLR_MENU; Clear the last menu.
            CALL        FAR PTR PRSTR           ;
            MOV         WORD PTR ES:OPT_MAX,ZERO; Initialise menu max. option
                                                ; counter.
            MOV         WORD PTR ES:TEMPLEN,ZERO; Initialise: Len. of option,
```

```
                MOV         WORD PTR ES:TEMPOFF,ZERO;              Offset in table.
                MOV         BX,WORD PTR ES:MENU_ADR ;
FIRST_CH:       MOV         CX,WORD PTR ES:XPOS     ;
                MOV         WORD PTR ES:TEMPX,CX    ;
                MOV         CX,WORD PTR ES:YPOS     ;
                MOV         WORD PTR ES:TEMPY,CX    ;
                MOV         AH,BYTE PTR ES:[BX]     ; Get the first char. of each
                XOR         AL,AL                   ; option, to be placed in the
                MOV         WORD PTR ES:TEMP_CH,AX  ; option lookup table.
NEXT_CH:                                            ;
                MOV         AL,BYTE PTR ES:[BX]     ; Get next char of option.
                INC         BX                      ; Increment pointer.
                CMP         AL,HLP                  ; If char = HLP , then
                JE          GET_BRIEF               ;    get start addr. of brief.
                CMP         AL,EOM                  ; elseif char = EOM , then
                JNE         NOT_EOM                 ;
                JMP         END_MENU                ;    exit the menu routine.
NOT_EOM:                                            ; elseif len(option) < max_len,
                CMP         WORD PTR ES:TEMPLEN,MAXLEN;                          then
                JE          NEXT_CH                 ;
                CALL        FAR PTR PRINTCH         ;    Print the character.
                INC         WORD PTR ES:TEMPLEN     ;    Increment option length.
                JMP         NEXT_CH                 ; Endif - check next char.
                                                    ;
GET_BRIEF:      MOV         DX,BX                   ; Keep copy of option pointer.
                MOV         BX,OFFSET MENU_DATA:OPT_TABLE; Load menu table pointer.
                ADD         BX,WORD PTR ES:TEMPOFF  ;
                MOV         CX,WORD PTR ES:TEMP_CH  ;
                MOV         WORD PTR ES:[BX],CX     ;
                MOV         CX,WORD PTR ES:TEMPX    ; Load X start pos. of option.
                MOV         WORD PTR ES:[BX+2],CX   ;
                MOV         CX,WORD PTR ES:TEMPY    ; Load Y start pos. of option.
                MOV         WORD PTR ES:[BX+4],CX   ;
                MOV         CX,WORD PTR ES:TEMPLEN  ; Load option length.
                MOV         WORD PTR ES:[BX+6],CX   ;
                MOV         WORD PTR ES:[BX+8],DX   ; Offset of brief.
                ADD         WORD PTR ES:TEMPOFF,10  ; Increment menu table pointer.
                MOV         BX,DX                   ; Restore option pointer.
                INC         WORD PTR ES:OPT_MAX     ;
NOT_END:        MOV         AL,BYTE PTR ES:[BX]     ; Skip to the start of the
                INC         BX                      ; next option.
                CMP         AL,ZERO                 ;
                JNE         NOT_END                 ;
                MOV         AL,' '                  ; Print ' ' between options.
                CALL        FAR PTR PRINTCH         ;
                CMP         WORD PTR ES:XPOS,XMENLIM; onto the next line.
                JLE         DO_NEXT                 ;
                CMP         WORD PTR ES:YPOS,(YMENST+YCH);
                JE          END_MENU                ;
                MOV         WORD PTR ES:XPOS,XMENST ;
                ADD         WORD PTR ES:YPOS,YCH    ;
DO_NEXT:        MOV         WORD PTR ES:TEMPLEN,ZERO;
                JMP         FIRST_CH                ; Print next option.
                                                    ;
END_MENU:       MOV         BX,OFFSET MENU_DATA:OPT_TABLE;
                ADD         BX,WORD PTR ES:TEMPOFF  ; Put an EOM char at the end
                MOV         WORD PTR ES:[BX],EOM    ; of the lookup table.
                MOV         AX,ZERO                 ; Set error return code.
PR_EXIT:                                            ;
                POP         BP                      ; Restore the registers.
                POP         DI                      ;
                POP         DX                      ;
                POP         CX                      ;
                POP         BX                      ;
                ASSUME      ES:DGROUP               ;
                RET                                 ; Exit.
PR_MENU         ENDP

;-------------------------------------------------------------------------------
DOMENU          PROC FAR
; Function:     To control the selection of options within an menu.
; Inputs:       Fortran passed parameter - Number of menu to control.
```

```
;           (Address of integer is passed on the stack.)
; Outputs:  AX, DX - integer passed back in fortran intger format.
;           Registers pass back number of option selected. Error code
;           is passed  back  if an invalid menu was attempted. (error
;           code = -1)
; Calls:    PR_MENU, CHG_ATR, CHG_ATR2, PR_BRIEF, GETKEY, MOVE_OPT,
;           TO_UPPER, ERTONE.
; Destroys: Flags, AX, DX.
;
; Description: Routine calls other subroutines to print out the new menus
;              and to set up the menu vector table. The menu vector table
;              contains  the start co-ordinates and length of each option
;              within the menu. The table also contains the start address
;              of  each  options' brief.  This  aids  this  routine when
;              controlling the menu.
;              If an option is selected, the routine checks if the option
;              exists within the menu. If the option is  valid,  then the
;              position of the option within the menu  is  passed back to
;              the  calling  routine.  Else  control  remains within this
;              routine.
;-------------------------------------------------------------------------
            PUSH        BP                      ; Save the registers.
            PUSH        ES                      ;
            PUSH        BX                      ;
            MOV         BP,SP                   ;
            LES         BX,DWORD PTR [BP+10]    ; Load address of parameter.
            MOV         AX,WORD PTR ES:[BX]     ; Load value of parameter.
                                                ;-----------------------------
            ASSUME      ES:MENU_DATA            ;
            MOV         BX,MENU_DATA            ;
            MOV         ES,BX                   ;
            CALL        FAR PTR PR_MENU         ; Print out the selected menu.
            CMP         AX,NO_MENU              ; If errors, then exit.
            JNE         RUNMENU                 ;
            JMP         SET_RET                 ;
RUNMENU:    MOV         WORD PTR ES:OPT_COUNT,1 ; Initialise option counter.
            CALL        FAR PTR CHG_ATR         ; Highlight first option.
            CALL        FAR PTR PR_BRIEF        ;


NEXT_OPT:   MOV         AX,1                    ;
            CALL        FAR PTR GETKEY          ; Wait for key to be entered.
                                                ; Case key :
            CMP         AX,TAB_KEY              ;  TAB:
            JNE         NOT_TAB                 ;
            MOV         AX,DIR_FWD              ;      Set direction forward.
            CALL        FAR PTR MOVE_OPT        ;      Highlight next option.
            JMP         NEXT_OPT                ;      Loop.
NOT_TAB:    CMP         AX,SPC_KEY              ;  SPACEBAR:
            JNE         NOT_BAR                 ;
            MOV         AX,DIR_FWD              ;      Set direction forward.
            CALL        FAR PTR MOVE_OPT        ;      Highlight next option.
            JMP         NEXT_OPT                ;      Loop.
NOT_BAR:    CMP         AX,REV_TAB              ;  REVERSE TAB:
            JNE         NOT_REVTAB              ;
            MOV         AX,DIR_REV              ;      Set direction reverse.
            CALL        FAR PTR MOVE_OPT        ;      Highlight prev. option.
            JMP         NEXT_OPT                ;      Loop.
NOT_REVTAB: CMP         AX,LDEL_KEY             ;  LEFT DELETE:
            JNE         NOT_LDEL                ;
            MOV         AX,DIR_REV              ;      Set direction reverse.
            CALL        FAR PTR MOVE_OPT        ;      Highlight prev. option.
            JMP         NEXT_OPT                ;      Loop.
NOT_LDEL:   CMP         AX,ESC_KEY              ;  ESC:
            JNE         NOT_ESCKEY              ;
            MOV         AX,ZERO                 ;      Return option zero.
            JMP         SET_RET                 ;      Exit.
NOT_ESCKEY: CMP         AX,CR_KEY               ;  RETURN:
            JNE         NOT_CR                  ;
            CALL        FAR PTR CHG_ATR         ;
            CALL        FAR PTR CHG_ATR2        ;
            MOV         AX,WORD PTR ES:OPT_COUNT;      Return option counter.
```

```
                JMP         SET_RET                 ;       Exit.
                                                    ;   OTHERWISE:
NOT_CR:         MOV         BX,OFFSET MENU_DATA:OPT_TABLE; Check input character
                MOV         WORD PTR ES:TEMP_COUNT,1;       against first chars
                CALL        FAR PTR TO_UPPER        ;
CHK_NEXT:       CMP         AX,WORD PTR ES:[BX]     ;       stored in the menu
                JNE         INCR_OPT                ;       lookup table.
                MOV         AX,WORD PTR ES:TEMP_COUNT;      If a char. is found, the
                CALL        FAR PTR CHG_ATR         ;
                MOV         WORD PTR ES:OPT_COUNT,AX;       position of the option
                CALL        FAR PTR CHG_ATR2        ;       is returned and the
                CALL        FAR PTR PR_BRIEF        ;
                JMP         SET_RET                 ;       routine is exited.
INCR_OPT:       CMP         WORD PTR ES:[BX],EOM    ;
                JE          ERROR_OPT               ;
                INC         WORD PTR ES:TEMP_COUNT  ;       If no match is found,
                ADD         BX,10                   ;       an error tone is sounded
                JMP         CHK_NEXT                ;       and the routine waits
ERROR_OPT:      CALL        FAR PTR ERTONE          ;       for the next key.
                JMP         NEXT_OPT                ;
                                                    ; Endcase.
SET_RET:        XOR         DX,DX                   ;
                MOV         SP,BP                   ; Restore registers.
                POP         BX                      ;
                POP         ES                      ;
                POP         BP                      ;
                ASSUME      ES:DGROUP               ;
                RET         04H                     ; Pop first parameter address
DOMENU          ENDP

;-------------------------------------------------------------------------------
MOVE_OPT        PROC FAR
; Function: To move the select highlight to the next option.
; Inputs:   AX - contains the direction of the highlight move.
;               (1 = forward ; 2 = reverse)
; Outputs:  Altered menu option list.
; Calls:    CHG_ATR, PR_BRIEF.
; Destroys: AX, Flags.
;
; Description: This routine reprints the previously highlighted option in
;              normal text and then prints the newly selected item with a
;              reverse attribute.
;              The routine also modifies the option pointer, OPT_COUNT.
;              This includes the wraparound at the end of the menu.
;              The new option brief is also printed out.
;
;-------------------------------------------------------------------------------
                ASSUME      ES:MENU_DATA            ;
                PUSH        BX                      ; Save the register.
                CALL        FAR PTR CHG_ATR         ; Change the current option.
                MOV         BX,WORD PTR ES:OPT_COUNT; Load the option counter.
                CMP         AX,DIR_FWD              ; Check on direction of option
                JNE         BACK_OPT                ; select.
                INC         BX                      ; If forward, then increment.
                CMP         BX,WORD PTR ES:OPT_MAX  ;    If counter > MAX_OPT
                JLE         HI_OPT                  ;       then wraparound to 1.
                MOV         BX,1                    ;
                JMP         HI_OPT                  ;
BACK_OPT:       DEC         BX                      ; If reverse, then decrement.
                CMP         BX,1                    ;    If counter < 1, then
                JGE         HI_OPT                  ;       wraparound to MAX_OPT.
                MOV         BX,WORD PTR ES:OPT_MAX  ;
HI_OPT:         MOV         WORD PTR ES:OPT_COUNT,BX; Save the new option counter.
                CALL        FAR PTR CHG_ATR         ; Change the new option.
                CALL        FAR PTR PR_BRIEF        ; Print the new option brief.
                POP         BX                      ; Restore the registers.
                ASSUME      ES:DGROUP               ;
                RET                                 ; Exit.
MOVE_OPT        ENDP

;-------------------------------------------------------------------------------
CHG_ATR         PROC FAR
```

```
; Function: To change the attribute of an option.
; Inputs:   OPT_COUNT - position of the option within the menu.
; Outputs:  Modified option in the menu.
; Calls:    INT10, PRSTR.
; Destroys: Flags.
;
; Description: The routine changes  the attribute of the option according
;              to the option  pointer. The print intensity is set so that
;              the output is XORed with the screen, effectively reversing
;              the attribute.
;-------------------------------------------------------------------------
            ASSUME      ES:MENU_DATA            ;
            PUSH        BP                      ; Save the registers.
            PUSH        DI                      ;
            PUSH        AX                      ;
            PUSH        BX                      ;
            PUSH        CX                      ;
            PUSH        DX                      ;
            MOV         BX,OFFSET MENU_DATA:SET_ATR; Set intensity mode to XOR.
            CALL        FAR PTR PRSTR           ;
            MOV         BX,OFFSET MENU_DATA:OPT_TABLE; Load start of lookup table.
            MOV         AX,WORD PTR ES:OPT_COUNT;   Load the option counter.
            DEC         AX                      ;
            MOV         CX,10                   ;
            MUL         CX                      ;   Get start of option in the
            ADD         BX,AX                   ;   lookup table.
            MOV         AX,WORD PTR ES:[BX+2]   ;   Get X position.
            MOV         DI,AX                   ;
            MOV         AX,WORD PTR ES:[BX+4]   ;   Get Y position.
            ADD         AX,2                    ;
            MOV         BP,AX                   ;
            MOV         AX,WORD PTR ES:[BX+6]   ;   Get option length.
            MOV         CX,XCH                  ;
            MUL         CX                      ;   Calculate length of option
            MOV         CX,AX                   ;   on the graphics page.
            MOV         BX,YCH                  ;   Height of option on page.
            SUB         BX,2                    ;
            MOV         AH,BLKFIL               ;   XOR the block.
            CALL        FAR PTR INT10           ;
            MOV         BX,OFFSET MENU_DATA:RESET_ATR; Reset intensity level to
            CALL        FAR PTR PRSTR           ;       white on black.
            POP         DX                      ; Restore the registers.
            POP         CX                      ;
            POP         BX                      ;
            POP         AX                      ;
            POP         DI                      ;
            POP         BP                      ;
            ASSUME      ES:DGROUP               ;
            RET                                 ; Exit.
CHG_ATR     ENDP

;-------------------------------------------------------------------------
CHG_ATR2    PROC FAR
; Function: To change the attribute of an option once it has been selected.
; Inputs:   OPT_COUNT - position of the option within the menu.
; Outputs:  Modified option in the menu.
; Calls:    INT10.
; Destroys: Flags.
;
; Description: The routine changes  the attribute of the option according
;              to the option  pointer. The print intensity is set so that
;              the output is XORed with the screen, effectively reversing
;              the attribute. This routine is only called once an option
;              is selected, and results in the option being surrounded by
;              a box rather than just reversed.
;-------------------------------------------------------------------------
            ASSUME      ES:MENU_DATA            ;
            PUSH        BP                      ; Save the registers.
            PUSH        DI                      ;
            PUSH        AX                      ;
            PUSH        BX                      ;
            PUSH        CX                      ;
```

```
            PUSH        DX                          ;
            MOV         BX,OFFSET MENU_DATA:OPT_TABLE; Load  start of lookup
                                                    ;       table.
            MOV         AX,WORD PTR ES:OPT_COUNT;   Load the option counter.
            DEC         AX                          ;
            MOV         CX,10                       ;
            MUL         CX                          ;   Get start of option in the
            ADD         BX,AX                       ;   lookup table.
            MOV         AX,WORD PTR ES:[BX+2]       ;   Get X position.
            MOV         DI,AX                       ;
            MOV         AX,WORD PTR ES:[BX+4]       ;   Get Y position.
            ADD         AX,2                        ;
            MOV         BP,AX                       ;
            MOV         AX,WORD PTR ES:[BX+6]       ;   Get option length.
            MOV         CX,XCH                      ;
            MUL         CX                          ;   Calculate length of option
            MOV         CX,AX                       ;   on the graphics page.
            MOV         BX,YCH                      ;   Height of option on page.
            SUB         BX,2                        ;
            MOV         AH,CURSOR                   ; Move the graphics cursor.
            CALL        FAR PTR INT10               ;
            MOV         AH,LINE                     ; Draw bottom horizontal.
            ADD         DI,CX                       ;
            CALL        FAR PTR INT10               ;
            SUB         BP,BX                       ; Draw right vertical.
            CALL        FAR PTR INT10               ;
            SUB         DI,CX                       ; Draw top horizontal.
            CALL        FAR PTR INT10               ;
            ADD         BP,BX                       ; Draw left vertical.
            CALL        FAR PTR INT10               ;
            POP         DX                          ; Restore the registers.
            POP         CX                          ;
            POP         BX                          ;
            POP         AX                          ;
            POP         DI                          ;
            POP         BP                          ;
            ASSUME      ES:DGROUP                   ;
            RET                                     ; Exit.
CHG_ATR2    ENDP

;-------------------------------------------------------------------------------
PR_BRIEF    PROC FAR
; Function: To print an options' brief.
; Inputs:   OPT_COUNT - position of option within the menu.
; Outputs:  Option brief on the screen.
; Calls:    PRSTR, INT10.
; Destroys: Flags.
;
; Description: First deletes the previous options' brief and locates the
;              start  of the new options' brief and uses PRSTR to output
;              the brief.
;-------------------------------------------------------------------------------
            ASSUME      ES:MENU_DATA                ;
            PUSH        AX                          ; Save the registers.
            PUSH        BX                          ;
            PUSH        CX                          ;
            PUSH        DX                          ;
            MOV         AH,LEVEL                    ; Set level to normal text.
            MOV         AL,NORMATR                  ;
            CALL        FAR PTR INT10               ;
            MOV         BX,OFFSET MENU_DATA:BRIEF_CLR; Clear the previous brief.
            CALL        FAR PTR PRSTR               ;
            MOV         AX,WORD PTR ES:OPT_COUNT;   Load option counter.
            DEC         AX                          ;
            MOV         BX,OFFSET MENU_DATA:OPT_TABLE; Calculate the postion of
            MOV         CX,10                       ;
            MUL         CX                          ;   the option in the lookup
            ADD         BX,AX                       ;   table and get  the brief
            ADD         BX,8                        ;   start address.
            MOV         AX,WORD PTR ES:[BX]         ;
            MOV         BX,AX                       ;
            CALL        FAR PTR PRSTR               ;   Print the brief.
```

```
                POP         DX                      ; Restore the registers.
                POP         CX                      ;
                POP         BX                      ;
                POP         AX                      ;
                ASSUME      ES:DGROUP               ;
                RET                                 ; Exit.
PR_BRIEF        ENDP

CODE            ENDS                                ;
                END                                 ;
                                                    ;
```

```
; MODULE : Menu macro routines.
;          ********************

TITLE Sets up the menu data segments.
;     *******************************

; ****************************************************************************
; 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

; ****************************************************************************
; 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          'Edit',HLP,'Edit or enter system matrices.',0
            DB          'Design',HLP,'Design controllers and compensators.',0
            DB          'Simulate',HLP,'Time simulation of the system.',0
            DB          'Quit',HLP,'End design session.',0
            DB          EOM
ENDMENU

STARTMENU   11
            DB          'Files',HLP,'Filenames, pathnames and system'
            DB          ' configuration.',0
            DB          'Project',HLP,'Load or save project information.',0
            DB          'G(s)',HLP,'Edit system matrix.',0
            DB          'K(s)',HLP,'Edit controller matrix.',0
            DB          'Clear',HLP,'Clear matrices.',0
            DB          'Names',HLP,'Enter project titles and names.',0
            DB          EOM
ENDMENU

STARTMENU   111
            DB          'Names',HLP,'Matrix filenames and pathnames.',0
            DB          'Directory',HLP,'Search a directory for files.',0
            DB          EOM
ENDMENU

STARTMENU   112
            DB          'Load',HLP,'Load project information from disk.',0
            DB          'Save',HLP,'Save project information to disk.',0
            DB          'Directory',HLP,'Search a directory for files.',0
            DB          EOM
ENDMENU

STARTMENU   113
            DB          'Load',HLP,'Load matrix from diskette.',0
            DB          'Save',HLP,'Save matrix to diskette.',0
            DB          'Edit',HLP,'Edit the matrix.',0
            DB          'Directory',HLP,'Search a directory for files.',0
            DB          'Clear',HLP,'Initialise matrix to zero matrix.',0
            DB          'Identity',HLP,'Initialise matrix to identity matrix.',0
            DB          'Print',HLP,'Print out the matrix of polynomials.',0
            DB          EOM
ENDMENU

STARTMENU   115
            DB          'G(s)',HLP,'Clear the system matrix.',0
            DB          'K(s)',HLP,'Clear the controller matrix.',0
            DB          'All',HLP,'Clear all the matrices.',0
            DB          EOM
ENDMENU

STARTMENU   12
            DB          'Plot',HLP,'Draw loci, misalignment angles, etc.',0
            DB          'Controller',HLP,'Generate controller matrix.',0
            DB          'Range',HLP,'Define frequency and dimensions.',0
            DB          'Scales',HLP,'Define max and min values for axes.',0
            DB          'Format',HLP,'Define which pages the graphs are'
            DB          ' to be drawn on.',0
            DB          'Edit',HLP,'Edit controller or compensator.',0
            DB          EOM
ENDMENU

STARTMENU   121
```

```
            DB          '1_Setting',HLP,'Use "Setting_1" plot page definition.',0
            DB          '2_Setting',HLP,'Use "Setting_2" plot page definition.',0
            DB          EOM
ENDMENU

STARTMENU 122
            DB          'Scalar',HLP,'Generate a scalar matrix controller in work
matrix.',0
            DB          'PI_Matrix',HLP,'Generate matrix PI controller in work matrix
from G(s).',0
            DB          'Vectors',HLP,'Eigen_vectors and _values of G(s) at a
particular frequency.',0
            DB          'Multiply',HLP,'Multiply K(s) by work matrix (symbolic).',0
            DB          'Edit',HLP,'Edit the elements of work matrix.',0
            DB          EOM
ENDMENU

STARTMENU   1226
            DB          'K(s)_X_Work',HLP,'Rows of K(s) multiplied columns of the work
matrix.',0
            DB          'Work_X_K(s)',HLP,'Rows of the work matrix multiplied columns
of K(s).',0
            DB          'G(s)_X_K(s)',HLP,'Rows of G(s) multiplied columns of K(s).',0
            DB          'G(s)_X_Work',HLP,'Rows of G(s) multiplied columns of the work
matrix.',0
            DB          EOM
ENDMENU

STARTMENU   13
            DB          'Project',HLP,'Names associated with the project.',0
            DB          'System',HLP,'System parameter names.',0
            DB          EOM
ENDMENU

STARTMENU   132
            DB          'Inputs',HLP,'Edit system input parameter names.',0
            DB          'Outputs',HLP,'Edit system output parameter names.',0
            DB          EOM
ENDMENU

STARTMENU   14
            DB          'Format',HLP,'Format of time simulation.',0
            DB          'Scales',HLP,'Adjust time simulation axes.',0
            DB          'Go',HLP,'Start the time simulation.',0
            DB          'Edit',HLP,'Edit or enter system parameters.',0
            DB          EOM
ENDMENU

STARTMENU   0
ENDMENU

; ******************************************************************************
            END
```

```
; MODULE : String I/O routines.
; ****************************

TITLE To read in and print out strings : FORTRAN type calls.
;     ****************************************************

; ************************************************************************
; REVISION HISTORY :
; VERSION       BY                DATE        COMMENT
;    1.00       Ian Fisher        23/06/88    Creation.
;
; ************************************************************************

; Routines.
; *********
            PUBLIC      NOBLNK      ; To remove trailing blanks from strings.
            PUBLIC      SCANIN      ; Scan input string, convert lower case to
                                    ; upper case.
            PUBLIC      STRIN       ; To input strings.

            EXTRN       ERTONE:FAR  ; Switch error tone on.
            EXTRN       GETKEY:FAR  ; Keyboard functions.
            EXTRN       INT10:FAR   ; INT10H Buffer routine.
            EXTRN       PRINTCH:FAR ; Print a char.
            EXTRN       PRSTR:FAR   ; Print a string.
            EXTRN       TO_UPPER:FAR; Converts alpha to upper case.

; Variables.
; **********
            EXTRN       DISPAGE:WORD; Page currently displayed.
            EXTRN       WRITPG:WORD ; Page to which data is being written.
            EXTRN       XPOS:WORD   ; X position of text cursor.
            EXTRN       YPOS:WORD   ; Y position of text cursor.

; Keyboard routine function numbers.
; **********************************
GET_ASCII   EQU         01H         ; Return ASCII key.
FLUSH_ASCII EQU         02H         ; Flush buffer.
KEY_HIT     EQU         03H         ; Check if key hit.
IF_HIT_GET  EQU         04H         ; If key hit, then get key.

; 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.

; Escape characters.
; ******************
ESCJ        EQU         04A1BH      ; Move cursor to x,y.
ESCK        EQU         04B1BH      ; Delete x number of characters.
ESCL        EQU         04C1BH      ; Set intensity to x.

; Keys from keyboard.
; *******************
ESC_KEY     EQU         1B00H       ; The ESC key.
CR_KEY      EQU         0D00H       ; The ENTER or RETURN key.
END_KEY     EQU         0047H       ; The END key on keypad.
HOME_KEY    EQU         004FH       ; The HOME key on keypad.
LDEL_KEY    EQU         0800H       ; Delete char to left of marker.
LEFT_KEY    EQU         004BH       ; Left cursor key.
RDEL_KEY    EQU         0053H       ; Delete char to right of marker.
REV_TAB     EQU         000FH       ; The REVERSE TAB key
RIGHT_KEY   EQU         004DH       ; Right cursor key.
TAB_KEY     EQU         0900H       ; The TAB key.

; Routine constants and flags.
; ****************************
DIR_FWD     EQU         01          ; Highlight next option.
DIR_REV     EQU         02          ; Highlight previous option.
```

```
LDEL        EQU         01          ; Delete char to left of marker.
NORMATR     EQU         01          ; Normal intensity.
NO_MENU     EQU         0FFH        ; No error when printing new menu.
PAGE0       EQU         00          ; Select page 0.
PAGE1       EQU         01          ; Select page 1.
RDEL        EQU         02          ; Delete char to right of marker.
XORATR      EQU         02          ; XOR type intensity.
ZERO        EQU         00          ; Just zero.

; Menu and character constants.
; *****************************
MAXLEN      EQU         12          ; Maximum length of option.
SCR_X       EQU         2           ; X co-ord of command headers.
XCH         EQU         09          ; Character width in graphics mode.
XMENLIM     EQU         571         ; X co-ord limit for menu.
XMENST      EQU         92          ; X co-ord for start of menu.
XPOSLIM     EQU         710         ; X limit for printing chars.
YBRIEF      EQU         331         ; Y co-ord for brief.
YCH         EQU         15          ; Character height in graphics mode.
YMENST      EQU         287         ; Y co-ord for start of menu.
YPOSLIM     EQU         347         ; Y limit for printing chars.

; GRAPHIX routine function numbers.
; *********************************
BFIL        EQU         04AH        ; Fill a block.
CLEAR       EQU         042H        ; Clear the screen.
DISPLAY     EQU         045H        ; Display a page.
LINE        EQU         049H        ; Draw a line.
GMODE       EQU         040H        ; Define graphics mode.
WRPAGE      EQU         043H        ; Page to which data is written.
ATRIB       EQU         044H        ; Intensity level.
CURSOR      EQU         048H        ; Move the imaginary cursor.
TEXT        EQU         04BH        ; Output text to the page.
TMODE       EQU         041H        ; To switch to text mode.

; General data segment.
; *********************
DATA        SEGMENT PUBLIC 'DATA'
DATA        ENDS

; Menu data areas.
; ****************
TABLE_DATA  SEGMENT WORD PUBLIC 'FAR_DATA'
TABLE_DATA  ENDS

; ***************************************************************************
MENU_DATA   SEGMENT WORD PUBLIC 'FAR_DATA'
CURSPOS     DW          0
NUMTYPE     DW          0
STRATRS     DW          0
STRLEN      DW          0
TEMPOFF     DW          0
TEMPSEG     DW          0
XTEMP       DW          0
YTEMP       DW          0
KTABLE      DB          'ABCDEFGHIJKLMNOPQRSTUVWXYZ'
            DB          'abcdefghijklmnopqrstuvwxyz'
            DB          '0123456789`-=\;,./~!@#$%^&*()_+|":<>?{}[] '
            DB          39
KTABLEND    DW          ($-KTABLE)
ITABLE      DB          '0123456789-+'
ITABLEND    DW          ($-ITABLE)
FTABLE      DB          '0123456789-+eEdD.'
FTABLEND    DW          ($-FTABLE)
MENU_DATA   ENDS

; ***************************************************************************
DGROUP      GROUP DATA
CODE        SEGMENT 'CODE'
            ASSUME CS:CODE
            ASSUME DS:DGROUP
            ASSUME ES:DGROUP
```

```
;-------------------------------------------------------------------------
NOBLNK      PROC FAR
; Function: To remove trailing blanks from number strings.
; Inputs:   FORTRAN : call NOBLNK(length,string)
;           length - TYPE : INTEGER*2
;           string - TYPE : CHARACTER*length
; Outputs:  Number string with no trailing blanks.
; Calls:    None.
; Destroys: Flags.
;
; Description: Shifts the entire string to the right, inserting blanks in
;              left side, until there are no blanks on the right side.
;-------------------------------------------------------------------------
            PUSH        BP                      ; Save the registers.
            PUSH        DI                      ;
            PUSH        ES                      ;
            PUSH        AX                      ;
            PUSH        BX                      ;
            PUSH        CX                      ;
            MOV         BP,SP                   ; Load stack pointer.
            LES         BX,DWORD PTR [BP+20]    ; Load length.
            MOV         DI,WORD PTR ES:[BX]     ;
            LES         BX,DWORD PTR [BP+16]    ; Load string ptr to (ES:BX).
            CMP         DI,ZERO                 ; If (length <= 0) then EXIT.
            JLE         BLNKEXIT                ;
            DEC         DI                      ; Decrement length.
            MOV         BP,DI                   ; Keep a copy of (length-1).
            MOV         AX,ZERO                 ; Initialise counter.
                                                ; Repeat
NEXTCHK:    MOV         DI,BP                   ;   Retrieve (length-1).
            INC         AX                      ;   Increment counter.
            CMP         BYTE PTR ES:[BX+DI],' ' ;   If (last_char <> ' ') then
            JNE         BLNKEXIT                ;                          EXIT.
            CMP         AX,BP                   ;   If(counter>(length-1)) then
            JG          BLNKEXIT                ;                          EXIT.
NEXTSHF:    CMP         DI,ZERO                 ;   While (count:DI > 0) do
            JE          INBLNK                  ;
            MOV         CL,BYTE PTR ES:[BX+DI-1];     string[DI]=string[DI-1].
            MOV         BYTE PTR ES:[BX+DI],CL  ;
            DEC         DI                      ;     Decrement count:DI.
            JMP         NEXTSHF                 ;   Enddo.
INBLNK:     MOV         BYTE PTR ES:[BX],' '    ;   Insert ' ' at start of
            JMP         NEXTCHK                 ;   the string.
BLNKEXIT:                                       ; Until ((last_char<>' ') or
                                                ;        (counter>(length-1))).
            POP         CX                      ; Restore the registers.
            POP         BX                      ;
            POP         AX                      ;
            POP         ES                      ;
            POP         DI                      ;
            POP         BP                      ;
            RET         08H                     ; Pop stack and EXIT.
NOBLNK      ENDP

;-------------------------------------------------------------------------
MARKER      PROC FAR
; Function: To print out a cursor to the screen.
; Inputs:   XTEMP, YTEMP : position of the cursor.
;           CURSPOS      : current position of cursor.
; Outputs:  To the screen.
; Calls:    INT10.
; Destroys: Flags.
;
; Description: XORs a vertical cursor to the screen at the position
;              according to XTEMP, YTEMP.
;-------------------------------------------------------------------------
            PUSH        BP                      ; Save the registers.
            PUSH        DI                      ;
            PUSH        ES                      ;
            PUSH        AX                      ;
            PUSH        CX                      ;
```

```
            MOV         AL,XORATR               ; Set write attribute to XOR.
            MOV         AH,ATRIB                ;
            CALL        FAR PTR INT10           ;
            ASSUME      ES:MENU_DATA            ; Use MENU_DATA segment.
            MOV         AX,MENU_DATA            ;
            MOV         ES,AX                   ;
            MOV         AX,WORD PTR ES:CURSPOS  ; Load marker position.
            DEC         AX                      ; Calculate the X position of
            MOV         CX,XCH                  ; the marker in the string
            MUL         CX                      ; for printing on the screen.
            ADD         AX,WORD PTR ES:XTEMP    ;
            MOV         DI,AX                   ;
            MOV         BP,WORD PTR ES:YTEMP    ; Get the Y position of the
            ADD         BP,4                    ; marker.
            MOV         AH,CURSOR               ; Move to the calculated point.
            CALL        FAR PTR INT10           ;
            SUB         BP,(YCH+1)              ;
            MOV         AH,LINE                 ; Create marker by drawing a
            CALL        FAR PTR INT10           ; vertical line.
            MOV         AL,NORMATR              ;
            MOV         AH,ATRIB                ; Reset write attribute to
            CALL        FAR PTR INT10           ; normal.
            ASSUME      ES:DGROUP               ;
            POP         CX                      ; Restore registers.
            POP         AX                      ;
            POP         ES                      ;
            POP         DI                      ;
            POP         BP                      ;
            RET                                 ; Exit.
MARKER      ENDP

;-------------------------------------------------------------------------------
SCANIN      PROC FAR
; Function: To scan input string, convert lower case to upper case.
; Inputs:   FORTRAN : call SCANIN(length,string)
;           length - TYPE : INTEGER*2
;           string - TYPE : CHARACTER*length
; Outputs:  Modified string.
; Calls:    TO_UPPER.
; Destroys: Flags.
;
; Description: Just checks each char. in string and converts if necessary.
;-------------------------------------------------------------------------------
            PUSH        BP                      ; Save the registers.
            PUSH        DI                      ;
            PUSH        ES                      ;
            PUSH        AX                      ;
            PUSH        BX                      ;
            MOV         BP,SP                   ; Load the stack pointer.
            LES         BX,DWORD PTR [BP+18]    ; Load the length.
            MOV         DI,WORD PTR ES:[BX]     ;
            LES         BX,DWORD PTR [BP+14]    ; Load string ptr to (ES:BX).
            CMP         DI,ZERO                 ; If (length <= 0) then EXIT.
            JLE         SCANEXIT                ;
                                                ; Repeat
NEXTSCAN:   DEC         DI                      ;   Decrement length or ptr.
            MOV         AH,BYTE PTR ES:[BX+DI]  ;   Load string[ptr] char.
            CALL        FAR PTR TO_UPPER        ;   Convert to upper case.
            MOV         BYTE PTR ES:[BX+DI],AH  ;   Store string[ptr] char.
            CMP         DI,ZERO                 ;
            JNE         NEXTSCAN                ; Until (ptr:DI = 0).
SCANEXIT:                                       ;
            POP         BX                      ; Restore the registers.
            POP         AX                      ;
            POP         ES                      ;
            POP         DI                      ;
            POP         BP                      ;
            RET         08H                     ; Pop the stack and EXIT.
SCANIN      ENDP

;-------------------------------------------------------------------------------
STRIN       PROC FAR
```

```
; Function: To prompt user for input string.
; Inputs:   FORTRAN : key = STRIN(page,x,y,maxlen,stringvar)
;                          page - type: INTEGER*2  (range: 0 or 1)
;                 x,y,maxlen - type: INTEGER*2
;                  stringvar - type: Array of chars.
;                        key - type: INTEGER*2.
; Outputs:  Input string echoed to screen.
;           Edited string in stringvar.
;           Key contains the returned non-string key.
; Calls:   INT10, STREDIT.
; Destroys: Flags.
;
; Description: The routine echos all input chars. to the page  requested. Only
;              printable ASCII chars. are accepted by the routine to be placed
;              in the string.
;              The routine will display the current contents of the string  up
;              to maxlen chars. (thus if no string  is  to  be  displayed, set
;              maxlen = 0).
;              Insert mode is assumed and any new chars entered will not over-
;              write what is displayed, but will be inserted in the string.
;              The  max.  number  of  chars.  in  the  string  will  be maxlen
;              The following keys have an effect on the string :-
;
;                   left, right cursor keys - move cursor within string.
;                             DEL, Back DEL - delete chars. in string.
;                                    RETURN - finished editing string.
;                 All printable ASCII chars. - Inserted into string.
;
;                 F1,..,F10,TAB,reverse TAB,
;                 up and down cursor keys.   - Control  is  passed  back  to
;                                              calling routine.  The  key is
;                                              passed  back  and  the edited
;                                              string.
;
;                   All other keys are ignored.
;
;-------------------------------------------------------------------------------
            PUSH        BP                      ; Save the registers.
            PUSH        DI                      ;
            PUSH        DS                      ;
            PUSH        ES                      ;
            PUSH        BX                      ;
            PUSH        CX                      ;
            PUSH        DX                      ;
            MOV         BP,SP                   ;
            MOV         AL,NORMATR              ;
            MOV         AH,ATRIB                ; Set write attribute to
            CALL        FAR PTR INT10           ; normal.
            LES         BX,DWORD PTR [BP+34]    ; Load page number to which
            MOV         AX,WORD PTR ES:[BX]     ; data is written.
            AND         AX,1                    ; Set the page number.
            MOV         AH,WRPAGE               ;
            CALL        FAR PTR INT10           ;
            XOR         AH,AH                   ;
            MOV         AH,DISPLAY              ; Display the page.
            CALL        FAR PTR INT10           ;
            ASSUME      DS:MENU_DATA            ; Use MENU_DATA segment.
            MOV         BX,MENU_DATA            ;
            MOV         DS,BX                   ;
            LES         BX,DWORD PTR [BP+30]    ; Load X co-ord of string.
            MOV         CX,WORD PTR ES:[BX]     ;
            MOV         WORD PTR DS:XPOS,CX     ; Store X co-ord.
            MOV         WORD PTR DS:XTEMP,CX    ;
            LES         BX,DWORD PTR [BP+26]    ; Load Y co-ord of string.
            MOV         CX,WORD PTR ES:[BX]     ;
            MOV         WORD PTR DS:YPOS,CX     ; Store Y co-ord.
            MOV         WORD PTR DS:YTEMP,CX    ;
            LES         BX,DWORD PTR [BP+22]    ; Load the length of the
            MOV         CX,WORD PTR ES:[BX]     ; string.
            MOV         WORD PTR DS:STRLEN,CX   ;
                                                ;
            LES         BX,DWORD PTR [BP+18]    ; Load the start address of
```

```
                                                ; the string.
            CALL        FAR PTR STREDIT         ;
                                                ;
            ASSUME      ES:DGROUP               ;
            ASSUME      DS:DGROUP               ;
            POP         DX                      ; Restore the registers.
            POP         CX                      ;
            POP         BX                      ;
            POP         ES                      ;
            POP         DS                      ;
            POP         DI                      ;
            POP         BP                      ;
            RET         14H                     ; Exit.
STRIN       ENDP

;-------------------------------------------------------------------------------
STRPRT      PROC FAR
; Function: To print out a string to be edited.
; Inputs:                 AX - Attributes of string (1 = string, reverse
;                                                   2 = string, normal).
;                      ES:BX - string to be output.
;           MENU_DATA:STRLEN - Max. length of string.
;       MENU_DATA:XPOS,XTEMP - Position of string : X co-ord.
;       MENU_DATA:YPOS,YTEMP -                      Y co-ord.
; Outputs:  To the screen.
; Calls:    INT10, PRINTCH.
; Destroys: Flags.
;
; Description: Prints out MAXLEN number of chars from the string and
;              then blanks the rest of the string for STRLEN chars.
;
;-------------------------------------------------------------------------------
            PUSH        BP                      ; Save the registers.
            PUSH        DI                      ;
            PUSH        DS                      ;
            PUSH        BX                      ;
            PUSH        CX                      ;
            ASSUME      DS:MENU_DATA            ; Use the MENU_DATA segment.
            MOV         CX,MENU_DATA            ;
            MOV         DS,CX                   ;
            MOV         CX,WORD PTR DS:XTEMP    ; Set X and Y cursor co-ords
            MOV         WORD PTR DS:XPOS,CX     ; to start of string.
            MOV         CX,WORD PTR DS:YTEMP    ;
            MOV         WORD PTR DS:YPOS,CX     ;
            MOV         WORD PTR DS:STRATRS,AX  ; Save the print type.
            CMP         AX,3                    ;
            JE          ERASTR                  ;
            PUSH        BX                      ; Save the string offset.
            MOV         AL,NORMATR              ; Set attributes to normal.
            MOV         AH,ATRIB                ;
            CALL        FAR PTR INT10           ;
            MOV         DI,WORD PTR DS:XTEMP    ; Erase the block where the
            MOV         BP,WORD PTR DS:YTEMP    ; string will be printed.
            ADD         BP,2                    ; The erase block is a block
            MOV         AX,WORD PTR DS:STRLEN   ; of highlight.
            MOV         CX,XCH                  ;
            MUL         CX                      ;
            MOV         CX,AX                   ;
            MOV         BX,YCH                  ;
            SUB         BX,2                    ;
            MOV         AH,BFIL                 ;
            CALL        FAR PTR INT10           ;
            POP         BX                      ; Restore string offset.
ERASTR:     MOV         AL,XORATR               ; Set attributes to XOR.
            MOV         AH,ATRIB                ;
            CALL        FAR PTR INT10           ;
            MOV         CX,ZERO                 ;
PRNEXT:     CMP         CX,WORD PTR DS:STRLEN   ; Print out the string. The
            JGE         END_PR                  ; length of string is given
            MOV         AL,BYTE PTR ES:[BX]     ; by STRLEN.
            CALL        FAR PTR PRINTCH         ;
            INC         BX                      ;
```

```
            INC         CX                          ;
            JMP         PRNEXT                      ;
END_PR:                                             ;
            MOV         AX,WORD PTR DS:STRATRS      ; If (no highlight required)
            CMP         AX,2                        ;  then
            JNE         PRTEXIT                     ;    Print a block over the
            MOV         AL,XORATR                   ;    string with the attribute
            MOV         AH,ATRIB                    ;    of XOR.
            CALL        FAR PTR INT10               ;
            MOV         DI,WORD PTR DS:XTEMP        ;
            MOV         BP,WORD PTR DS:YTEMP        ;
            ADD         BP,2                        ;
            MOV         AX,WORD PTR DS:STRLEN       ;
            MOV         CX,XCH                      ;
            MUL         CX                          ;
            MOV         CX,AX                       ;
            MOV         BX,YCH                      ;
            SUB         BX,2                        ;
            MOV         AH,BFIL                     ;
            CALL        FAR PTR INT10               ;
PRTEXIT:    MOV         AL,NORMATR                  ;
            MOV         AH,ATRIB                    ;
            CALL        FAR PTR INT10               ;
                                                    ; Endif.
            ASSUME      DS:DGROUP                   ;
            POP         CX                          ; Restore the registers.
            POP         BX                          ;
            POP         DS                          ;
            POP         DI                          ;
            POP         BP                          ;
            RET                                     ; Exit.
STRPRT      ENDP

;-------------------------------------------------------------------------------
DELCHR      PROC FAR
; Function: To delete a char from a string being edited.
; Inputs:                 AX - Char to be deleted from left or right
;                              (1 - left, 2 - right).
;                         ES:BX - the string to be edited.
;           MENU_DATA:STRLEN - max length of string.
;           MENU_DATA:CURSPOS - position of marker in string.
; Outputs:  Edited string.
; Calls:    MARKER, ERTONE.
; Destroys: Flags.
;
; Description: Checks if a character can be edited from the string, if the
;              char can be deleted then the operation is performed otherwise
;              the error tone is sounded. The string is not reprinted.
;-------------------------------------------------------------------------------
            PUSH        DI                          ; Save the registers.
            PUSH        DS                          ;
            PUSH        AX                          ;
            PUSH        CX                          ;
            ASSUME      DS:MENU_DATA                ; Use MENU_DATA segment.
            MOV         CX,MENU_DATA                ;
            MOV         DS,CX                       ;
            CALL        FAR PTR MARKER              ; Switch marker off.
            CMP         AX,LDEL                     ; If (left delete key) then
            JNE         DO_RDEL                     ;
            CMP         WORD PTR DS:CURSPOS,1       ;    If (marker at left end)
            JLE         NO_DEL                      ;     then, sound tone + exit.
            DEC         WORD PTR DS:CURSPOS         ;    Else
            JMP         DO_DEL                      ;     Decrement marker position
                                                    ;     and delete char.
                                                    ;    Endif.
DO_RDEL:    MOV         AX,WORD PTR DS:STRLEN       ; Elseif (marker at right end)
            CMP         WORD PTR DS:CURSPOS,AX      ;  then, sound tone and exit.
            JG          NO_DEL                      ; Else
DO_DEL:     MOV         DI,WORD PTR DS:CURSPOS      ;    Load marker position.
                                                    ;    While (not at end of str)
NEXT_DEL:   CMP         DI,WORD PTR DS:STRLEN       ;
            JE          DOBLANK                     ;
```

```
            MOV         AL,BYTE PTR ES:[BX+DI]  ;       Shift one char to left,
            MOV         BYTE PTR ES:[BX+DI-1],AL;       from position of marker.
            INC         DI                      ;       Increment pointer.
            JMP         NEXT_DEL                ;     Endwhile.
DOBLANK:    MOV         BYTE PTR ES:[BX+DI-1],' ';    Insert blank on the end
            JMP         DEL_EXIT                ;     of the string.
NO_DEL:                                         ; Endif.
            CALL        FAR PTR ERTONE          ; Sound error tone.
DEL_EXIT:                                       ;
            CALL        FAR PTR MARKER          ; Reprint the string.
            ASSUME      DS:DGROUP               ;
            POP         CX                      ; Restore the registers.
            POP         AX                      ;
            POP         DS                      ;
            POP         DI                      ;
            RET                                 ; Exit.
DELCHR      ENDP

;-------------------------------------------------------------------------------
INSCHR      PROC FAR
; Function: To insert a char into the string being edited.
; Inputs:               ES:BX - string being edited.
;                          AL - char to be inserted.
;            MENU_DATA:STRLEN - max length of string.
;           MENU_DATA:CURSPOS - postion of marker.
; Outputs:  Edited string.
; Calls:    MARKER, ERTONE.
; Destroys: Flags.
;
; Description: Inserts the character according to marker postion. Characters
;              at the end of the string are lost.
;-------------------------------------------------------------------------------
            PUSH        DI                      ; Save the registers.
            PUSH        DS                      ;
            PUSH        AX                      ;
            PUSH        BX                      ;
            PUSH        CX                      ;
            ASSUME      DS:MENU_DATA            ; Use MENU_DATA segment.
            MOV         CX,MENU_DATA            ;
            MOV         DS,CX                   ;
            MOV         DI,WORD PTR DS:STRLEN   ; Load the string length.
            CMP         WORD PTR DS:CURSPOS,DI  ; If (marker at end) then
            JG          NO_INS                  ;   Sound tone and exit.
            CMP         WORD PTR DS:CURSPOS,DI  ; Elesif (marker 1 char from
            JE          DO_CHR                  ;         the end) then
                                                ;   Overwrite last char;
                                                ; Else
SHF_NEXT:   DEC         DI                      ;   While (pointer <> marker)
            MOV         CL,BYTE PTR ES:[BX+DI-1];     Shift chars from marker,
            MOV         BYTE PTR ES:[BX+DI],CL  ;     1 char to the  right  to
            CMP         DI,WORD PTR DS:CURSPOS  ;     make place for new char.
            JNE         SHF_NEXT                ;   Endwhile.
DO_CHR:     MOV         BYTE PTR ES:[BX+DI-1],AL;   Insert new char into
            MOV         CX,WORD PTR DS:CURSPOS  ;   new space.
            CMP         CX,WORD PTR DS:STRLEN   ;   Move marker if not at the
            JG          INS_EXIT                ;   end of the string.
            CALL        FAR PTR MARKER          ;
            INC         WORD PTR DS:CURSPOS     ;
            CALL        FAR PTR MARKER          ;
            JMP         INS_EXIT                ; Endif.
NO_INS:                                         ;
            CALL        FAR PTR ERTONE          ; Sound error tone.
INS_EXIT:                                       ;
            ASSUME      DS:DGROUP               ;
            POP         CX                      ; Restore registers.
            POP         BX                      ;
            POP         AX                      ;
            POP         DS                      ;
            POP         DI                      ;
            RET                                 ;Exit.
INSCHR      ENDP
```

```
;-------------------------------------------------------------------------------
STREDIT     PROC FAR
; Function: To allow the user to edit a string.
; Inputs:             ES:BX - String to be edited.
;           MENU_DATA:STRLEN - Length of string.
;            MENU_DATA:XTEMP - Start position of string : X co-ord.
;            MENU_DATA:YTEMP -                            Y co-ord.
; Outputs:  Edited string,
;           AX - Latest unidentifable key.
; Calls:    GETKEY, STRPRT, DELCHR, INSCHR, MARKER.
; Destroys: Flags.
;
; Description: Allows the user to edit the string. Any keys not identified
;              by the routine (including those in the printable char table)
;              are accepted by the routine as termination of the editing.
;              The string and the key are passed back to the calling routine.
;-------------------------------------------------------------------------------
            PUSH        DI                      ;
            PUSH        DS                      ;
            PUSH        CX                      ;
            ASSUME      DS:MENU_DATA            ;
            MOV         CX,MENU_DATA            ;
            MOV         DS,CX                   ;
            MOV         AX,ES                   ;
            MOV         WORD PTR DS:TEMPSEG,AX  ;
            MOV         WORD PTR DS:TEMPOFF,BX  ;
            MOV         AX,1                    ; the string.
            CALL        FAR PTR STRPRT          ; Print out the string.
            MOV         WORD PTR DS:CURSPOS,1   ; Initialise marker position.
            CALL        FAR PTR MARKER          ; Print the marker.
NEXTKEY:                                        ;
            MOV         AX,GET_ASCII            ; Wait for key to be input.
            CALL        GETKEY                  ;
                                                ; Case (input_key) :
            CMP         AX,LEFT_KEY             ;  LEFT_CURSOR:
            JNE         NOT_LEFT                ;
            CMP         WORD PTR DS:CURSPOS,1   ;     If (cursor not at start)
            JLE         NEXTKEY                 ;      then
            CALL        FAR PTR MARKER          ;        Move cursor one char
            DEC         WORD PTR DS:CURSPOS     ;        to the left.
            CALL        FAR PTR MARKER          ;
            JMP         NEXTKEY                 ;     Endif.
NOT_LEFT:                                       ;
            CMP         AX,RIGHT_KEY            ;  RIGHT_CURSOR:
            JNE         NOT_RIGHT               ;
            MOV         AX,WORD PTR DS:STRLEN   ;     If (cursor not at end)
            CMP         WORD PTR DS:CURSPOS,AX  ;      then
            JG          NEXTKEY                 ;
            CALL        FAR PTR MARKER          ;        Move cursor one char
            INC         WORD PTR DS:CURSPOS     ;        to the right.
            CALL        FAR PTR MARKER          ;
            JMP         NEXTKEY                 ;     Endif.
NOT_RIGHT:                                      ;
            CMP         AX,END_KEY              ;
            JNE         NOT_ENDK                ;
            CALL        FAR PTR MARKER          ;
            MOV         WORD PTR DS:CURSPOS,1   ;
            CALL        FAR PTR MARKER          ;
            JMP         NEXTKEY                 ;
NOT_ENDK:                                       ;
            CMP         AX,HOME_KEY             ;
            JNE         NOT_HOME                ;
            CALL        FAR PTR MARKER          ;
            MOV         AX,WORD PTR DS:STRLEN   ;

            PUSH        ES                      ;
            PUSH        BX                      ;
            MOV         BX,WORD PTR DS:TEMPSEG  ;       Load string start adr.
            MOV         ES,BX                   ;
            MOV         BX,WORD PTR DS:TEMPOFF  ;
            ADD         BX,AX                   ;
            INC         AX                      ;
```

```
NEXT_END:                                       ;
            CMP         AX,0                    ;
            JLE         END_MARK                ;
            DEC         AX                      ;
            DEC         BX                      ;
            CMP         BYTE PTR ES:[BX],' '    ;
            JE          NEXT_END                ;
END_MARK:                                       ;
            POP         BX                      ;
            POP         ES                      ;
            INC         AX                      ;
            MOV         WORD PTR DS:CURSPOS,AX  ;
            CALL        FAR PTR MARKER          ;
            JMP         NEXTKEY                 ;
NOT_HOME:                                       ;
            CMP         AX,RDEL_KEY             ; RIGHT_DELETE:
            JNE         NOT_RDEL                ;
            MOV         AX,3                    ;
            CALL        FAR PTR STRPRT          ;
            MOV         AX,RDEL                 ;   Set delete direction.
            CALL        FAR PTR DELCHR          ;   Delete the char.
            MOV         AX,3                    ;   Set print type.
            CALL        FAR PTR STRPRT          ;   Print out string.
            JMP         NEXTKEY                 ;
NOT_RDEL:                                       ;
            CMP         AX,LDEL_KEY             ; LEFT_DELETE:
            JNE         NOT_LDEL                ;
            MOV         AX,3                    ;
            CALL        FAR PTR STRPRT          ;
            MOV         AX,LDEL                 ;  Set delete direction.
            CALL        FAR PTR DELCHR          ;  Delete the char.
            MOV         AX,3                    ;  Set print type.
            CALL        FAR PTR STRPRT          ;  Print out string.
            JMP         NEXTKEY                 ;
NOT_LDEL:                                       ;
            MOV         BX,AX                   ; OTHERWISE:
            ASSUME      ES:MENU_DATA            ;   Use MENU_DATA segment.
            MOV         CX,MENU_DATA            ;
            MOV         ES,CX                   ;
            MOV         DI,OFFSET MENU_DATA:KTABLE; Load start address of
            MOV         CX,WORD PTR DS:KTABLEND ;   lookup table of printable
            MOV         AL,AH                   ;   chars.
            XOR         AH,AH                   ;
            REPNE       SCASB                   ;   Scan the table.
            JNE         STREXIT                 ;   If (key identified) then
            MOV         BX,WORD PTR DS:TEMPSEG  ;      Load string start adr.
            MOV         ES,BX                   ;
            MOV         BX,WORD PTR DS:TEMPOFF  ;
            PUSH        AX                      ;
            MOV         AX,3                    ;      Erase string.
            CALL        FAR PTR STRPRT          ;
            POP         AX                      ;
            CALL        FAR PTR INSCHR          ;      Insert the char.
            MOV         AX,3                    ;      Set print type.
            CALL        FAR PTR STRPRT          ;      Print out string.
            JMP         NEXTKEY                 ;    Else
STREXIT:    MOV         CX,BX                   ;      Keep a copy of key.
            CALL        FAR PTR MARKER          ;      Switch marker off.
            MOV         BX,WORD PTR DS:TEMPSEG  ;      Load string start adr.
            MOV         ES,BX                   ;
            LES         BX,DWORD PTR [BP+18]    ;      Load string start adr.
            MOV         AX,2                    ;      Set print type.
            CALL        FAR PTR STRPRT          ;      Print out string.
            MOV         AX,CX                   ;      Restore key.
            ASSUME      DS:DGROUP               ;
            POP         CX                      ;
            POP         DS                      ;
            POP         DI                      ;
            RET                                 ;
STREDIT     ENDP

CODE        ENDS                                ;
```

```
; MODULE : STROPS
; ***************

TITLE To Implement string operations for FORTRAN.
;     ****************************************

; ********************************************************************************
; REVISION HISTORY :
; VERSION       BY                DATE        COMMENT
;    1.00       Ian Fisher        23/06/88    Creation.
;
; ********************************************************************************

; Routines.
; *********
            PUBLIC      APPEND      ; To append two strings.
            PUBLIC      CMPSTR      ; To search for a substring.
            PUBLIC      COPYST      ; To copy a section of a string.
            PUBLIC      LENSTR      ; To find the first non-blank from left.
            PUBLIC      RJUST       ; To remove leading blanks from a string.
            PUBLIC      STRCPY      ; To copy a string.

; General data segment.
; *********************
DATA        SEGMENT PUBLIC 'DATA'
DATA        ENDS

; Menu data areas.
; ****************
TABLE_DATA  SEGMENT WORD PUBLIC 'FAR_DATA'
TABLE_DATA  ENDS

MENU_DATA   SEGMENT WORD PUBLIC 'FAR_DATA'
STRING2     DB          100 DUP(0)
MENU_DATA   ENDS

; ********************************************************************************
DGROUP      GROUP DATA
CODE        SEGMENT 'CODE'
            ASSUME CS:CODE
            ASSUME DS:DGROUP
            ASSUME ES:DGROUP

;--------------------------------------------------------------------------------
APPEND      PROC FAR
; Function: To append string2 to string1
; Inputs:   FORTRAN : call APPEND(len1,str1,len2,str2)
;           len1  - (integer*2) Length of string1.
;           str1  - (character*len1) Target string.
;           len2  - (integer*2) Length of string2.
;           str2  - (character*len2) String to be appended.
; Outputs:  str2 appended to str1.
; Calls:    None.
; Destroys: Flags.
;
; Description: Appends string2 onto string1 from the first none blank
;              character from the right of string1.
;--------------------------------------------------------------------------------
            PUSH        BP                      ; Save the registers.
            PUSH        DI                      ;
            PUSH        SI                      ;
            PUSH        DS                      ;
            PUSH        ES                      ;
            PUSH        AX                      ;
            PUSH        BX                      ;
            PUSH        CX                      ;
            PUSH        DX                      ;
            MOV         BP,SP                   ; Load stack pointer.
            LES         BX,DWORD PTR [BP+34]    ;
            MOV         DI,WORD PTR ES:[BX]     ; Load len1.
            CMP         DI,0                    ; If (len1 <= 0) then EXIT.
            JLE         EXITAPP                 ;
```

```
            DEC         DI                          ; Decrement len1.
            MOV         DX,DI                       ; Keep a copy of (len1-1).
            LES         BX,DWORD PTR [BP+26]        ;
            MOV         CX,WORD PTR ES:[BX]         ; Load len2.
            CMP         CX,0                        ; If (len2 <= 0) then EXIT.
            JLE         EXITAPP                     ;
            DEC         CX                          ; Decrement len2.
            LES         BX,DWORD PTR [BP+30]        ; Load pointer to str1.
                                                    ;
CHKCH:      CMP         BYTE PTR ES:[BX+DI],' '     ; Repeat
            JNE         FINCHK                      ;
            CMP         DI,0                        ;
            JE          FINCHK                      ;
            DEC         DI                          ;   Decrement pointer:DI.
            JMP         CHKCH                       ; Until ((str1[DI] <> ' ') and
                                                    ;                (DI = 0)).
FINCHK:     CMP         BYTE PTR ES:[BX+DI],' '     ; If (str[DI] <> ' ') then
            JE          NOINC                       ;
            INC         DI                          ;    Increment DI.
                                                    ; Endif.
NOINC:                                              ;
            MOV         AX,ES                       ; Move str1 pointer (ES:BX) to
            MOV         DS,AX                       ;                   (DS:AX).
            MOV         AX,BX                       ;
            LES         BX,DWORD PTR [BP+22]        ; Load str2 pointer to (ES:BX).
            MOV         BP,DX                       ; Load len1.
            MOV         SI,0                        ; Initialise str2 pointer.
                                                    ;
APPCH:      CMP         DI,BP                       ; While ((DI <= (len1-1)) and
            JG          EXITAPP                     ;
            CMP         SI,CX                       ;        (SI <= (len2-1))) do
            JG          EXITAPP                     ;
            MOV         DL,BYTE PTR ES:[BX+SI]      ;   Load byte from str2.
            INC         SI                          ;   Increment str2 pointer:SI.
            XCHG        AX,BX                       ;
            MOV         BYTE PTR DS:[BX+DI],DL      ;   Store byte to str1.
            INC         DI                          ;   Increment str1 pointer:DI.
            XCHG        AX,BX                       ;
            JMP         APPCH                       ; Enddo.
EXITAPP:                                            ;
            POP         DX                          ; Restore the registers.
            POP         CX                          ;
            POP         BX                          ;
            POP         AX                          ;
            POP         ES                          ;
            POP         DS                          ;
            POP         SI                          ;
            POP         DI                          ;
            POP         BP                          ;
            RET         10H                         ; Pop the stack and exit.
APPEND      ENDP

;-------------------------------------------------------------------------------
CMPSTR      PROC FAR
; Function: To find a substring within a string.
; Inputs:   FORTRAN : pos = CMPSTR(len1,str1,len2,str2)
;           len1 - (integer*2) Length of target string.
;           str1 - (character*len1) String to be searched.
;           len2 - (integer*2) Length of substring.
;           str2 - (character*len2) Substring.
; Outputs:  pos  - (integer*2) Position of substring in string.
;                  (pos = 0, if substring not found.)
; Calls:    None.
; Destroys: Flags.
;
; Description: Searches for substring within target string. If the
;              substring is identified, then the routine returns the
;              position of the substring within the target. If the
;              substring is not found, then a value of zero is returned.
;-------------------------------------------------------------------------------
            PUSH        BP                          ; Save the registers.
            PUSH        DI                          ;
```

```
                PUSH    SI                      ;
                PUSH    DS                      ;
                PUSH    ES                      ;
                PUSH    BX                      ;
                PUSH    CX                      ;
                PUSH    DX                      ;
                MOV     BP,SP                   ; Load the stack pointer.
                LES     BX,DWORD PTR [BP+20]    ; Load str2 pointer to (DS:AX).
                MOV     AX,ES                   ;
                MOV     DS,AX                   ;
                MOV     AX,BX                   ;
                LES     BX,DWORD PTR [BP+24]    ;
                MOV     DX,WORD PTR ES:[BX]     ; Load len2.
                LES     BX,DWORD PTR [BP+32]    ;
                MOV     CX,WORD PTR ES:[BX]     ; Load len1.
                LES     BX,DWORD PTR [BP+28]    ; Load str1 pointer to (ES:BX).
                                                ;
                MOV     SI,0                    ; Initialise flag to "no_match"
                CMP     CX,0                    ; If (len1 <= 0) then EXIT.
                JLE     EXITCMP                 ;
                DEC     CX                      ; Decrement len1.
                CMP     DX,0                    ; If (len2 <= 0) then EXIT.
                JLE     EXITCMP                 ;
                DEC     DX                      ; Decrement len2.
                                                ; count1 = len1-1.
                                                ; count2 = len2-1.
CHKSTR:                                         ;Repeat
                CMP     CX,DX                   ;  If (count1 < count2) then
                JGE     STR1OK                  ;
                MOV     SI,0                    ;    Set flag to "no_match".
                JMP     EXITCMP                 ;    EXIT.
                                                ;  else
STR1OK:         MOV     DI,0                    ;    Initialise pointer:DI.
CHKAGAIN:                                       ;    Repeat
                PUSH    DX                      ;      Save (len2-1).
                XCHG    AX,BX                   ;
                MOV     DL,BYTE PTR DS:[BX+DI]  ;      Load byte from str2[DI].
                XCHG    AX,BX                   ;
                CMP     DL,BYTE PTR ES:[BX+DI]  ;      If (str1[DI]=str2[DI])
                JNE     INCSTR1                 ;       then
                POP     DX                      ;         Restore (len2-1).
                INC     DI                      ;         Increment pointer:DI.
                CMP     DI,DX                   ;      Endif.
                JLE     CHKAGAIN                ;
                JMP     FOUND                   ;    Until((string match found)
                                                ;        or(str1[DI]<>str2[DI])
                                                ;        or(DI > (len2-1)).
                                                ;   Endif
                                                ;   If (str1[DI]<>str2[DI])then
INCSTR1:        POP     DX                      ;      Restore (len2-1).
                INC     BX                      ;      Increment str1 pointer.
                DEC     CX                      ;      Decrement count1.
                                                ;   endif.
                JMP     CHKSTR                  ; Until ((count1 < count2) or
FOUND:                                          ;        (string match found)).
                LES     BX,DWORD PTR [BP+32]    ; Load len1.
                MOV     AX,WORD PTR ES:[BX]     ;
                SUB     AX,CX                   ; Calculate the position of the
                MOV     SI,AX                   ; string match.
EXITCMP:        MOV     AX,SI                   ; Load position into AX as the
                                                ; return value.
                POP     DX                      ; Restore the registers.
                POP     CX                      ;
                POP     BX                      ;
                POP     ES                      ;
                POP     DS                      ;
                POP     SI                      ;
                POP     DI                      ;
                POP     BP                      ;
                RET     10H                     ; Pop the stack and EXIT.
CMPSTR          ENDP
```

```
;-------------------------------------------------------------------------
LENSTR      PROC FAR
; Function: To find the first non-blank from the left.
; Inputs:   FORTRAN : pos = LENSTR(len1,str1)
;           len1 - (integer*2) Length of string.
;           str1 - (character*len1) Target string.
; Outputs:  pos - (integer*2) Position of the first non-blank char from left.
; Calls:    None.
; Destroys: Flags.
;
; Description: Just comapares the input string chars from left until a
;              non-blank char is found.
;-------------------------------------------------------------------------
            PUSH        BP                      ; Save the registers.
            PUSH        DI                      ;
            PUSH        ES                      ;
            PUSH        BX                      ;
            MOV         BP,SP                   ; Load the stack pointer.
            LES         BX,DWORD PTR [BP+16]    ;
            MOV         AX,WORD PTR ES:[BX]     ; Load len1.
            CMP         AX,0                    ;
            JLE         EXITLEN                 ; If (len1 <= 0) then EXIT.
            DEC         AX                      ; Decrement len1.
            MOV         DI,AX                   ; Keep a copy of (len1-1).
            LES         BX,DWORD PTR [BP+12]    ; Load str1 pointer to (ES:BX).
                                                ;
NEXTLEN:    CMP         BYTE PTR ES:[BX+DI],' ' ; While ((strl[DI] = ' ') and
            JNE         FINLEN                  ;        (DI >= 0 )) do
            DEC         DI                      ;   Decrement pointer:DI.
            CMP         DI,0                    ;
            JGE         NEXTLEN                 ; Enddo.
FINLEN:     MOV         AX,DI                   ; Load pointer (length) to AX
            INC         AX                      ; as return value.
EXITLEN:                                        ;
            POP         BX                      ; Restore the registers.
            POP         ES                      ;
            POP         DI                      ;
            POP         BP                      ;
            RET         08H                     ; Pop stack and EXIT.
LENSTR      ENDP

;-------------------------------------------------------------------------
STRCPY      PROC FAR
; Function: To copy a string.
; Inputs:   FORTRAN : call STRCPY(len1,string1,len2,string2)
;                len1, len2 - type : INTEGER*2
;           string1, string2 - type : CHARACTER*len
;           string1 - destination
;           string2 - source
; Outputs:  string1 = string2 (if len1 > len2 then string padded with blanks.)
; Calls:    None.
; Destroys: Flags.
;
; Description: Copies string2 to string1, if len1 greater than len2 then
;              string1 is padded with blanks.
;-------------------------------------------------------------------------
            PUSH        BP                      ; Save the registers.
            PUSH        DI                      ;
            PUSH        DS                      ;
            PUSH        ES                      ;
            PUSH        AX                      ;
            PUSH        BX                      ;
            PUSH        CX                      ;
            PUSH        DX                      ;
            MOV         BP,SP                   ; Load the stack pointer.
            LES         BX,DWORD PTR [BP+32]    ;
            MOV         DI,WORD PTR ES:[BX]     ; Load len1.
            LES         BX,DWORD PTR [BP+28]    ; Load str1 pointer to (DS:AX).
            MOV         AX,ES                   ;
            MOV         DS,AX                   ;
            MOV         AX,BX                   ;
            LES         BX,DWORD PTR [BP+24]    ;
```

```
            MOV         DX,WORD PTR ES:[BX]     ; Load len2.
            LES         BX,DWORD PTR [BP+20]    ; Load str2 pointer to (ES:BX).
            MOV         BP,0                    ; Initialise string pointer:BP.
NEXTCH:     CMP         DI,BP                   ; While (BP < len1) do
            JE          CPY_EXIT                ;
            CMP         DX,BP                   ;   If (BP <= len2) then
            JLE         NEXTBLNK                ;
            MOV         CL,BYTE PTR ES:[BX]     ;      Load char from str2.
            XCHG        AX,BX                   ;
            MOV         BYTE PTR DS:[BX],CL     ;      store char in str1.
            XCHG        AX,BX                   ;
            INC         BP                      ;      Increment pointer:BP.
            INC         AX                      ;      Increment str1 pointer.
            INC         BX                      ;      Increment str2 pointer.
            JMP         NEXTCH                  ;
NEXTBLNK:   XCHG        AX,BX                   ;   else
            MOV         BYTE PTR DS:[BX],' '    ;      Store ' ' in str1.
            XCHG        AX,BX                   ;
            INC         BP                      ;      Increment pointer:BP.
            INC         AX                      ;      Increment str1 pointer.
                                                ;   Endif.
            JMP         NEXTCH                  ; Enddo.
CPY_EXIT:                                       ;
            POP         DX                      ; Restore the registers.
            POP         CX                      ;
            POP         BX                      ;
            POP         AX                      ;
            POP         ES                      ;
            POP         DS                      ;
            POP         DI                      ;
            POP         BP                      ;
            RET         10H                     ; Pop stack and EXIT.
STRCPY      ENDP

;-------------------------------------------------------------------------------
COPYST      PROC FAR
; Function: To copy sections of a string.
; Inputs:   FORTRAN : call COPYST(len1,str1,pos1,len2,str2,pos2,num)
;           len1, len2 - type : INTEGER*2
;           str1, str2 - type : CHARACTER*len
;           str1 - destination
;           str2 - source
;           pos1 - (integer*2) Start location of copy.
;           pos2 - (integer*2) Start location of copy.
;            num - (integer*2) Number of chars. to copy.
; Outputs:  Modified str1.
; Calls:    None.
; Destroys: Flags.
;
; Description: Copies string2 to string1, starting at pos1 and pos2
;              respectively, and copy num characters.
;-------------------------------------------------------------------------------
            PUSH        BP                      ; Save the registers.
            PUSH        DI                      ;
            PUSH        SI                      ;
            PUSH        DS                      ;
            PUSH        ES                      ;
            PUSH        AX                      ;
            PUSH        BX                      ;
            PUSH        CX                      ;
            PUSH        DX                      ;
            MOV         BP,SP                   ; Load the stack pointer.
            LES         BX,DWORD PTR [BP+22]    ;
            MOV         DX,WORD PTR ES:[BX]     ; Load num.
            CMP         DX,0                    ; If (num <= 0) then EXIT.
            JG          NUMOK                   ;
            JMP         EXITCPY                 ;
NUMOK:      LES         BX,DWORD PTR [BP+26]    ;
            MOV         AX,WORD PTR ES:[BX]     ; Load pos2.
            CMP         AX,0                    ; If (pos2 < 0) then EXIT.
            JGE         POS2OK                  ;
            JMP         EXITCPY                 ;
```

```
POS2OK:     LES         BX,DWORD PTR [BP+34]    ;
            MOV         CX,WORD PTR ES:[BX]     ; Load len2.
            CMP         CX,0                    ; If (len2 <= 0) then EXIT.
            JLE         EXITCPY                 ;
            CMP         AX,CX                   ; If (len2 < pos2) then EXIT.
            JG          EXITCPY                 ;
            MOV         BX,AX                   ;
            ADD         BX,DX                   ;
            CMP         BX,CX                   ; If ((pos2+num) > len2) then
            JLE         NUM2OK                  ;
            MOV         DX,CX                   ;
            SUB         DX,AX                   ;    num = len2 - pos2
NUM2OK:                                         ; Endif.
            LES         BX,DWORD PTR [BP+38]    ;
            MOV         AX,WORD PTR ES:[BX]     ; Load pos1.
            CMP         AX,0                    ; If (pos1 < 0) then EXIT.
            JL          EXITCPY                 ;
            LES         BX,DWORD PTR [BP+46]    ;
            MOV         CX,WORD PTR ES:[BX]     ; Load len1.
            CMP         CX,0                    ; If (len1 <= 0) then EXIT.
            JLE         EXITCPY                 ;
            CMP         AX,CX                   ; If (pos1 > len1) then EXIT.
            JG          EXITCPY                 ;
            MOV         BX,AX                   ;
            ADD         BX,DX                   ;
            CMP         BX,CX                   ; If ((pos1+num) > len1) then
            JLE         NUM1OK                  ;
            MOV         DX,CX                   ;    num = len1 - pos1.
            SUB         DX,AX                   ; Endif.
NUM1OK:                                         ;
            LES         BX,DWORD PTR [BP+42]    ; Load str1 pointer to (ES:BX).
            MOV         SI,CX                   ; Initialise ptr:SI with len1.
            DEC         SI                      ; Decrement ptr:SI.
            SUB         SI,DX                   ; ptr:SI = ptr:SI - num
            MOV         DI,CX                   ; Load ptr:DI with len1.
            DEC         DI                      ; Decrement ptr:DI.
            PUSH        DX                      ; Save num.
SHFTCH:                                         ;
            CMP         SI,AX                   ; While (ptr:SI >= pos1) do
            JL          CPYCH                   ;
            MOV         DL,BYTE PTR ES:[BX+SI]  ;   str1[ptr] = str1[ptr-num]
            MOV         BYTE PTR ES:[BX+DI],DL  ;
            DEC         SI                      ;   Decrement ptr:SI.(len1-num)
            DEC         DI                      ;   Decrement ptr:DI.(len1)
            JMP         SHFTCH                  ; Enddo.
CPYCH:                                          ;
            MOV         SI,AX                   ; Keep a copy of pos1.
            MOV         AX,ES                   ; Move str1 pointer to (DS:AX).
            MOV         DS,AX                   ;
            MOV         AX,BX                   ;
            LES         BX,DWORD PTR [BP+26]    ;
            MOV         DI,WORD PTR ES:[BX]     ; Load pos2.
            LES         BX,DWORD PTR [BP+30]    ; Load str2 pointer to (ES:BX).
            POP         DX                      ; Recall num.
            MOV         BP,DX                   ; Initialise counter:BP.
                                                ;
NEXTCPY:    CMP         BP,0                    ; While (counter:BP > 0) do
            JLE         EXITCPY                 ;
            MOV         DL,BYTE PTR ES:[BX+DI]  ;   str1[ptr1+pos1] =
            INC         DI                      ;                str2[ptr2+pos2].
            XCHG        AX,BX                   ;
            MOV         BYTE PTR DS:[BX+SI],DL  ;
            INC         SI                      ;
            XCHG        AX,BX                   ;
            DEC         BP                      ;
            JMP         NEXTCPY                 ;
                                                ; Enddo.
EXITCPY:                                        ;
            POP         DX                      ; Restore registers.
            POP         CX                      ;
            POP         BX                      ;
            POP         AX                      ;
```

```
            POP         ES                      ;
            POP         DS                      ;
            POP         SI                      ;
            POP         DI                      ;
            POP         BP                      ;
            RET         1CH                     ; Pop stack and EXIT.
COPYST      ENDP

;-------------------------------------------------------------------------
RJUST       PROC FAR
; Function: To remove leading blanks from a string
; Inputs:   FORTRAN : call RJUST(len1,str1)
;           len1 - (integer*2) Length of string.
;           str1 - (character*len1) The target string.
; Outputs:  Right justified string.
; Calls:    None.
; Destroys: Flags.
;
; Description: Shifts the entire string right one char. for every blank
;              found before any non-blank chars. Blanks are shifted on
;              from the left. The routine will loop a maximum of 'len1'
;              times. This is to prevent infinite loops if a blank string
;              is passed to the routine.
;-------------------------------------------------------------------------
            PUSH        BP                      ; Save the registers.
            PUSH        DI                      ;
            PUSH        SI                      ;
            PUSH        ES                      ;
            PUSH        AX                      ;
            PUSH        BX                      ;
            PUSH        CX                      ;
            MOV         BP,SP                   ; Load the stack pointer.
            LES         BX,DWORD PTR [BP+22]    ;
            MOV         AX,WORD PTR ES:[BX]     ; Load len1.
            CMP         AX,0                    ; If (len1 <= 0) the EXIT.
            JLE         EXITRJUST               ;
            LES         BX,DWORD PTR [BP+18]    ; Load str1 pointer to (ES:BX).
            MOV         DI,0                    ; Initialise counter:DI.
                                                ;
NEXTJUST:   CMP         DI,AX                   ; While ((DI < len1) and
            JGE         EXITRJUST               ;       ((1st str1 char)<>' '))
            CMP         BYTE PTR ES:[BX],' '    ;                              do
            JNE         EXITRJUST               ;
            MOV         SI,0                    ;   Initialise pointer:SI.
NEXTSHF:    CMP         SI,AX                   ;   While (SI < len1) do
            JGE         INSBLNK                 ;
            MOV         CL,BYTE PTR ES:[BX+SI+1];     str1[SI] = str1[SI+1].
            MOV         BYTE PTR ES:[BX+SI],CL  ;
            INC         SI                      ;     Increment pointer:SI.
            JMP         NEXTSHF                 ;   Enddo.
INSBLNK:    MOV         SI,AX                   ;
            MOV         BYTE PTR ES:[BX+SI-1],' ';  str1[len1] = ' '.
            INC         DI                      ;   Increment counter:DI.
            JMP         NEXTJUST                ; Enddo.
EXITRJUST:                                      ;
            POP         CX                      ; Restore the registers.
            POP         BX                      ;
            POP         AX                      ;
            POP         ES                      ;
            POP         SI                      ;
            POP         DI                      ;
            POP         BP                      ;
            RET         08H                     ; Pop the stack and EXIT.
RJUST       ENDP

CODE        ENDS                                ;
            END                                 ;
;-------------------------------------------------------------------------
```

```
; MODULE : Keyboard routines.
; ***************************

TITLE Keyboard character input routine.
;     *********************************

; ********************************************************************************
; REVISION HISTORY :
; VERSION       BY                  DATE        COMMENT
;    1.00       Ian Fisher          23/06/88    Creation.
;
; ********************************************************************************

; Procedures.
; ***********
            PUBLIC      GETKEY      ; General keyboard routine.
            PUBLIC      INKEY       ; FORTRAN Version of GETKEY.

            EXTRN       INT10:FAR   ; Graphics interrupt call.
            EXTRN       DOHELP:FAR  ; Help file driver.
            EXTRN       PRSCRS:FAR  ; Print graphics screens.

; Variables.
; **********
            EXTRN       DISPAGE:WORD; Current page being displayed.

; Function select values.
; ***********************
GET_ASCII   EQU         01H         ; Return ASCII key.
FLUSH_ASCII EQU         02H         ; Flush buffer.
KEY_HIT     EQU         03H         ; Check if key hit.
IF_HIT_GET  EQU         04H         ; Check if key hit and get key.

; INT 21H : Function calls and returns.
; *************************************
CHAR_TRUE   EQU         0FFH        ; Return if a char in buffer.
FLUSH_KB    EQU         00CH        ; Function : flush buffer.
NO_PROCESS  EQU         000H        ; Function : no function,
                                    ;            while flushing.
READ_KB     EQU         008H        ; Function : read ASCII char.
STATUS_KB   EQU         00BH        ; Function : read keyb. status.
TRUE        EQU         001H        ; Value of logical true.

; Function key flags.
; *******************
DOREP       EQU         01          ; Do repeat key function.
NOREP       EQU         00          ; Do not repeat key function.

; Function keys.
; **************
F1          EQU         003BH       ;
F2          EQU         003CH       ;
F3          EQU         003DH       ;
F4          EQU         003EH       ;
F5          EQU         003FH       ;
F6          EQU         0040H       ;
F7          EQU         0041H       ;
F8          EQU         0042H       ;
F9          EQU         0043H       ;
F10         EQU         0044H       ;
ALTF9       EQU         0112        ; ALT-F9
ALTF10      EQU         0113        ; ALT-F10

; General data segment.
; *********************
DATA        SEGMENT PUBLIC 'DATA'
REPFLAG     DW          NOREP
TEMPKEY     DW          0
TEMPFUNC    DW          0
DATA        ENDS

DGROUP      GROUP DATA
```

```
CODE        SEGMENT 'CODE'
            ASSUME CS:CODE
            ASSUME DS:DGROUP
            ASSUME ES:DGROUP

;--------------------------------------------------------------------------
GETKEY      PROC FAR
; Function: To read in a single character or check if key hit.
; Inputs:   AX - function for routine to perform.
; Outputs:  AX - contains  the  return  code  or return character for
;                the calling routine.
; Calls:    DOS interrupt routine : INT 21H.
; Destroys: Flags.
;
; Description: The routine is passed an integer which is used to determine the
;              function required. The parameter to function relationship is as
;              follows :-
;                      Parameter     Function
;                      ---------     --------
;                         1          Return the ASCII code of key typed.
;                                    (The char is not echoed.)
;                                    Waits for char. to be typed.
;
;                         2          The keyboard buffer  is  flushed  and no
;                                    other function is performed.
;
;                         3          Tests if a key has been typed. Returns 0
;                                    if no key was typed. 1 if key typed.
;
;                         4          Tests if  a key was typed: if a key was
;                                    typed then the  ASCII  code is returned
;                                    else the routine returns a 0.
;
;--------------------------------------------------------------------------
            PUSH        ES                          ;
            PUSH        BX                          ;
            ASSUME      ES:DGROUP                   ;
            MOV         BX,DGROUP                   ;
            MOV         ES,BX                       ;
            MOV         WORD PTR ES:TEMPFUNC,AX     ;
DOAGAIN:                                            ;
            MOV         WORD PTR ES:REPFLAG,NOREP;
            CMP         AL,GET_ASCII                ; Get ASCII char ?
            JNE         NOT_ASCII                   ;
            MOV         AH,READ_KB                  ;   Set function to read char.
            INT         21H                         ;   Execute function.
                                                    ;   (Function waits for key to
                                                    ;    entered before returning)
READ_CH:    CMP         AL,0                        ;   Has ALT, Function or cursor
            JNE         ONE_CH                      ;   key been entered ?
            MOV         AH,STATUS_KB                ;    Keyboard status function.
            INT         21H                         ;    Get status.
            CMP         AL,CHAR_TRUE                ;    If a key has been hit ?
            JNE         NO_CHAR                     ;       then : read char,
            MOV         AH,READ_KB                  ;              return = char
            INT         21H                         ;       else : return = 0;
            XOR         AH,AH                       ;
            JMP         CHKF                        ;
NO_CHAR:    XOR         AX,AX                       ;    Zero rest of return value.
            JMP         CHKF                        ;
ONE_CH:     MOV         AH,AL                       ;
            XOR         AL,AL                       ;
            JMP         CHKF                        ; Exit.
CHKF:       MOV         WORD PTR ES:TEMPKEY,AX      ;
            CALL        FAR PTR CHKFUNC             ;
            MOV         AX,WORD PTR ES:TEMPFUNC     ;
            CMP         WORD PTR ES:REPFLAG,DOREP;
            JE          DOAGAIN                     ;
            MOV         AX,WORD PTR ES:TEMPKEY      ;
            JMP         GET_EXIT                    ;
                                                    ;
NOT_ASCII:                                          ;----------------------------
```

```
            CMP         AL,FLUSH_ASCII          ; Flush the keyboard buffer ?
            JNE         NOT_FLUSH               ;
            MOV         AH,FLUSH_KB             ;   Set function to flush the
            MOV         AL,NO_PROCESS           ;   buffer and do nothing else.
            INT         21H                     ;   Execute the function.
            XOR         AX,AX                   ;   Zero the return value.
            JMP         GET_EXIT                ; Exit.
NOT_FLUSH:                                      ;-----------------------------
            CMP         AL,KEY_HIT              ; Obtain status of keyboard ?
            JNE         NOT_HIT                 ;
            MOV         AH,STATUS_KB            ;   Set function to read state.
            INT         21H                     ;   Execute the function.
            CMP         AL,CHAR_TRUE            ;   Is there a char ?
            JNE         STAT_EXIT               ;      then : return = 1
            MOV         AL,TRUE                 ;      else : return = 0;
STAT_EXIT:  XOR         AH,AH                   ;   Zero rest of return value.
            JMP         GET_EXIT                ; Exit.
NOT_HIT:                                        ;-----------------------------
            CMP         AL,IF_HIT_GET           ; If key hit, get ASCII char ?
            JNE         NOT_IFHIT               ;
            MOV         AH,STATUS_KB            ;    Keyboard status function.
            INT         21H                     ;    Get status.
            CMP         AL,CHAR_TRUE            ;    If a key has been hit ?
            JNE         NO_INPUT                ;       then : read char,
            MOV         AH,READ_KB              ;              return = char
            INT         21H                     ;       else : return = 0;
            JMP         READ_CH                 ;
                                                ;
NO_INPUT:   XOR         AX,AX                   ;    Zero rest of return value.
            JMP         GET_EXIT                ; Exit.
NOT_IFHIT:                                      ;-----------------------------
GET_EXIT:                                       ;
            POP         BX                      ;
            POP         ES                      ;
            RET                                 ; Exit.
GETKEY      ENDP

;-------------------------------------------------------------------------------
INKEY       PROC FAR
; Function: FORTRAN Version :To read in a single character or check if key hit.
; Inputs:   FORTRAN: key = INKEY(code)
;           key - (integer*2) Return value from routine.
;           code - (integer*2) Function number.
; Outputs:  AX  - contains  the  return  code  or return character for
;                 the calling routine.
; Calls:    GETKEY.
; Destroys: Flags.
;
; Description: Same as for GETKEY above.
;
;-------------------------------------------------------------------------------
            PUSH        BP                      ; Save the registers.
            PUSH        ES                      ;
            PUSH        BX                      ;
            MOV         BP,SP                   ;
            LES         BX,DWORD PTR [BP+10]    ; Load address of parameter.
            MOV         AX,WORD PTR ES:[BX]     ; Load value of parameter.
                                                ;-----------------------------
            CALL        FAR PTR GETKEY          ;
            MOV         SP,BP                   ; Restore registers.
            POP         BX                      ;
            POP         ES                      ;
            POP         BP                      ;
            RET         04H                     ; Pop first parameter address
                                                ; and return to caller.
INKEY       ENDP

;-------------------------------------------------------------------------------
CHKFUNC     PROC FAR
; Function: Checks if a function key has been entered.
; Inputs:   AX - key entered.
; Outputs:  If function key hit, then that function is invoked,
```

```
            ASSUME      ES:DGROUP               ; Use DGROUP segment.
            MOV         AX,DGROUP               ;
            MOV         ES,AX                   ;
            MOV         WORD PTR ES:REPFLAG,DOREP; Set keyboard repeat function
            POP         AX                      ;
            POP         ES                      ;
            JMP         FEXIT                   ;
NOT_ALTF9:                                      ;
            CMP         AX,ALTF10               ;
            JNE         NOT_ALTF10              ;
            PUSH        ES                      ;
            PUSH        AX                      ;
            MOV         AX,1                    ;
            CALL        FAR PTR PRSCRS          ;
            ASSUME      ES:DGROUP               ; Use DGROUP segment.
            MOV         AX,DGROUP               ;
            MOV         ES,AX                   ;
            MOV         WORD PTR ES:REPFLAG,DOREP; Set keyboard repeat function
            POP         AX                      ;
            POP         ES                      ;
            JMP         FEXIT                   ;
NOT_ALTF10:                                     ;
FEXIT:                                          ;
            POP         BX                      ;
            POP         ES                      ;
            RET                                 ;
CHKFUNC     ENDP

;-------------------------------------------------------------------------------
FUNC3       PROC FAR
; Function: To perform function 3, flip display pages.
; Inputs:   DATA:DISPAGE - Page currently displayed.
;           DATA:REPFLAG - Inform keyboard routine to repeat function.
; Outputs:  DISPAGE contains new page.
;           Screen display changes to alternate page.
; Calls:    INT10.
; Destroys: Flags.
;
; Description: XORs current value of page and then calls routine to display
;              that page. The function repeat flag, for the keyboard routine
;              is also set.
;
;-------------------------------------------------------------------------------
            PUSH        ES                      ; Save registers.
            PUSH        AX                      ;
            ASSUME      ES:DGROUP               ; Use DGROUP segment.
            MOV         AX,DGROUP               ;
            MOV         ES,AX                   ;
            MOV         AX,WORD PTR ES:DISPAGE  ;  Load display state var.
            XOR         AL,1                    ;  Flip state.
            MOV         WORD PTR ES:DISPAGE,AX  ;  Execute display function.
            MOV         AH,45H                  ;
            CALL        FAR PTR INT10           ;
            MOV         WORD PTR ES:REPFLAG,DOREP; Set keyboard repeat function
            POP         AX                      ;  flag.
            POP         ES                      ;
            RET                                 ; Exit.
FUNC3       ENDP

;-------------------------------------------------------------------------------
FUNC1       PROC        FAR
; Function: To perform function 1, initiate HELP facility.
; Inputs:   DATA:REPFLAG - Inform keyboard routine to repeat function.
; Outputs:  None.
; Calls:    dohelp (for).
; Destroys: Flags.
;
; Description: Saves the registers, calls the help routine and upon regaining
;              control, sets the flag to repeat the keyboard function
;-------------------------------------------------------------------------------
            PUSH        DS                      ; Save the registers.
            PUSH        ES                      ;
```

```
                PUSH    AX                      ;
                PUSH    BX                      ;
                PUSH    CX                      ;
                PUSH    DX                      ;
                PUSH    SI                      ;
                PUSH    DI                      ;
                PUSH    BX                      ;
                ASSUME  ES:DGROUP               ; Use the DGROUP segment
                ASSUME  DS:DGROUP               ;
                MOV     BX,DGROUP               ;
                MOV     ES,BX                   ;
                MOV     DS,BX                   ;
                POP     BX                      ;
                CALL    FAR PTR DOHELP          ; Call DOHELP()
                ASSUME  ES:DGROUP               ;
                MOV     AX,DGROUP               ;
                MOV     ES,AX                   ;
                MOV     WORD PTR ES:REPFLAG,DOREP; Set keyboard function repeat
                POP     DI                      ;  flag.
                POP     SI                      ;  Restore registers.
                POP     DX                      ;
                POP     CX                      ;
                POP     BX                      ;
                POP     AX                      ;
                POP     ES                      ;
                POP     DS                      ;
                RET                             ; Exit.
FUNC1           ENDP

CODE            ENDS
                END                             ;
;-------------------------------------------------------------------------------
```

```
; MODULE : To handle critical errors.
; ***********************************

; ****************************************************************************
; REVISION HISTORY :
; VERSION       BY              DATE        COMMENT
;    1.00       Ian Fisher      23/06/88    Creation.
;
; ****************************************************************************

; Routines.
; *********
            PUBLIC      INDERR      ; Initialise the INT24H vector.
            PUBLIC      REDERR      ; Reset the INT24h vector.
            PUBLIC      RSTERR      ; Reset error variable.
            PUBLIC      GETERR      ; Read the error variable.

; Variables.
; **********
            PUBLIC      DERROR      ; Error variable.

; Fatal error interrupt vector address.
; *************************************
ERROFFSET   EQU         090H        ; Critical error interrupt offset.
ERRSEG      EQU         092H        ; Critical error interrupt segment.

; General data segment.
; *********************
DATA        SEGMENT PUBLIC 'DATA'
INT24VEC    DD          0
DERROR      DW          0
DATA        ENDS

; ****************************************************************************
DGROUP      GROUP DATA
CODE        SEGMENT 'CODE'
            ASSUME CS:CODE
            ASSUME DS:DGROUP
            ASSUME ES:DGROUP

;----------------------------------------------------------------------------
INDERR      PROC FAR
; Function: To grab the fatal error interrupt vector (24H)
; Inputs:   None.
; Outputs:  INT 24H vector replaced.
; Calls:    None.
; Destroys: None.
;
; Description: Holds off interrupts while replacing the vector. The
;              old vector is kept in order to call the old routine if
;              the new routine cannot handle the error and for replacement
;              at the end of the program.
;
;----------------------------------------------------------------------------
            PUSH        DS                          ;
            PUSH        ES                          ;
            PUSH        AX                          ;
            CLI                                     ;
            ASSUME      DS:DGROUP                   ;
            MOV         AX,DGROUP                   ;
            MOV         DS,AX                       ;
            MOV         AX,0                        ;
            MOV         ES,AX                       ;
                                                    ; Error handler.
            MOV         AX,WORD PTR ES:[ERROFFSET]; Save old offset.
            MOV         WORD PTR DS:INT24VEC,AX ;
            MOV         AX,WORD PTR ES:[ERRSEG] ;   Save old segment.
            MOV         WORD PTR DS:INT24VEC[2],AX;
            MOV         AX,OFFSET DISKERR       ;   Load new offset.
            MOV         WORD PTR ES:[ERROFFSET],AX;
            MOV         AX,SEG DISKERR          ;
            MOV         WORD PTR ES:[ERRSEG],AX ;   Load new segment.
```

```
                                                ;
            MOV         WORD PTR DS:DERROR,-1   ; Initialise error variable.
            POP         AX                      ;
            POP         ES                      ;
            POP         DS                      ;
            STI                                 ;
            RET                                 ;
INDERR      ENDP                                ;

;-------------------------------------------------------------------------------
REDERR      PROC FAR
; Function: To replace the fatal error interrupt vector (24H).
; Inputs:   Old fatal error vector.
; Outputs:  Reset INT 24H vector.
; Calls:    None.
; Destroys: Flags.
;
; Description: Holds off interrupts while replacing the old vector.
;
;-------------------------------------------------------------------------------
            PUSH        DS                      ;
            PUSH        ES                      ;
            PUSH        AX                      ;
            CLI                                 ;
            ASSUME      DS:DGROUP               ;
            MOV         AX,DGROUP               ;
            MOV         DS,AX                   ;
            MOV         AX,0                    ;
            MOV         ES,AX                   ;
            MOV         AX,WORD PTR DS:INT24VEC ;   Restoring interrupt vector
            MOV         WORD PTR ES:[ERROFFSET],AX; 25 to state before program
            MOV         AX,WORD PTR DS:INT24VEC[2];   execution. (DOS error
            MOV         WORD PTR ES:[ERRSEG],AX ;   handler interrupt.)
            POP         AX                      ;
            POP         ES                      ;
            POP         DS                      ;
            STI                                 ;
            RET                                 ;
REDERR      ENDP                                ;

;-------------------------------------------------------------------------------
DISKERR     PROC FAR
; Function: If a critical error occurs, then this routine ensures that the
;           operation fails.
; Inputs:   AH - If bit 7 clear then error is a disk error else ?.
;           DI - error code.
; Outputs:  AL - Error return code for DOS to make decision on - this routine
;                always returns 3 which instructs DOS to quit the operation.
;           DGROUP:DERROR - The error code for the application program
;                           to access.
; Calls:    None.
; Destroys: Flags.
;
; Description: The routine sets AL such that DOS will fail the operation,
;              passing the fail back to FORTRAN.
;              The error code  will be  placed  in DERROR for the application
;              program to access.
;-------------------------------------------------------------------------------
            PUSH        ES                      ;
            PUSH        BX                      ;
            MOV         BX,DGROUP               ;
            MOV         ES,BX                   ;
            MOV         BH,AH                   ;
            AND         BH,80H                  ;
            CMP         BH,0                    ;
            JMP         FATAL_ERR               ;
            JE          FATAL_ERR               ;
            PUSHF                               ;
            CALL        DWORD PTR ES:INT24VEC   ;
            JMP         HAND_EXIT               ;
FATAL_ERR:                                      ;
            MOV         WORD PTR ES:DERROR,DI   ;
```

```
            MOV         AL,3                    ;
HAND_EXIT:                                      ;
            POP         BX                      ;
            POP         ES                      ;
            IRET                                ;
DISKERR     ENDP                                ;

;-------------------------------------------------------------------------------
GETERR      PROC FAR
; Function: To query the latest disk/drive error.
; Inputs:   FORTRAN : err = GETERR()
; Outputs:  err - (INTEGER*2) The error number as passed to the DOS
;                             fatal error interrupt handler.
; Calls:    None.
; Destroys: Flags.
;
; Description: Just places the contents of DROUP:DERROR into AX.
;-------------------------------------------------------------------------------
            PUSH        ES                      ;
            ASSUME      ES:DGROUP               ;
            MOV         AX,DGROUP               ;
            MOV         ES,AX                   ;
            MOV         AX,WORD PTR ES:DERROR   ;
            POP         ES                      ;
            RET                                 ;
GETERR      ENDP                                ;

;-------------------------------------------------------------------------------
RSTERR      PROC FAR
; Function: To reset the disk/drive error number.
; Inputs:   FORTRAN : call RSTERR()
; Outputs:  None.
; Calls:    None.
; Destroys: Flags.
;
; Description: Just sets DGROUP:DERROR to -1.
;-------------------------------------------------------------------------------
            PUSH        ES                      ;
            PUSH        AX                      ;
            ASSUME      ES:DGROUP               ;
            MOV         AX,DGROUP               ;
            MOV         ES,AX                   ;
            MOV         WORD PTR ES:DERROR,-1   ;
            POP         AX                      ;
            POP         ES                      ;
            RET                                 ;
RSTERR      ENDP                                ;

CODE        ENDS                                ;
            END                                 ;
;-------------------------------------------------------------------------------
```

```
; MODULE : Utility routines.
; *************************

TITLE Set of utilities for use with the HERCULES card.
;     ********************************************

; ****************************************************************************
; REVISION HISTORY :
; VERSION       BY                DATE        COMMENT
;    1.00       Ian Fisher        23/06/88    Creation.
;
; ****************************************************************************

; Routines.
; *********
            PUBLIC      ARC         ; Draw an arc.
            PUBLIC      BLKFIL      ; Fill a block.
            PUBLIC      BOX         ; To draw a rectangle.
            PUBLIC      CIRC        ; Draw a circle.
            PUBLIC      CLRSCR      ; Clear the screen.
            PUBLIC      DEFDRV      ; Returns the default drive number.
            PUBLIC      DISP        ; Display a page.
            PUBLIC      DLINE       ; Draw a line.
            PUBLIC      ERTONE      ; To set the error tone on.
            PUBLIC      FFIRST      ; Find first file in directory search.
            PUBLIC      FNEXT       ; Find next file in directory search.
            PUBLIC      FILL        ; To fill a polygon.
            PUBLIC      GETPG       ; Return page being displayed.
            PUBLIC      GETWPG      ; Return page to which data is written.
            PUBLIC      GETATT      ; Return current screen attribute.
            PUBLIC      GMODE       ; To switch to graphics mode.
            PUBLIC      GPAGE       ; Select page to which data is written.
            PUBLIC      INTONE      ; To initialise the tone interupt.
            PUBLIC      INT10       ; To call the interrupt.
            PUBLIC      LEVEL       ; Sets intensity level.
            PUBLIC      MOVE        ; Move the graphics cursor.
            PUBLIC      PLOT        ; Plot a point.
            PUBLIC      TMODE       ; To switch to text mode.
            PUBLIC      WIPSCR      ; To clear one of the screens.

            PUBLIC      PRINTCH     ; To print out a single character.
            PUBLIC      PRSTR       ; To print out a string.
            PUBLIC      RETONE      ; Reset the tone interrupt vector.
            PUBLIC      TO_UPPER    ; Convert lower to upper.
            PUBLIC      WRTSTR      ; FORTRAN: Write string at x,y.

; Variables.
; **********
            PUBLIC      WRITPG      ; Current page to which data is written.
            PUBLIC      DISPAGE     ; Current page being displayed.
            PUBLIC      ATTPG       ; Current attribute.
            PUBLIC      XPOS        ; X co-ord. of text cursor.
            PUBLIC      YPOS        ; Y co-ord of text cursor.

; Keyboard routine function numbers.
; **********************************
GET_ASCII   EQU         01H         ; Return ASCII key.
FLUSH_ASCII EQU         02H         ; Flush buffer.
KEY_HIT     EQU         03H         ; Check if key hit.
IF_HIT_GET  EQU         04H         ; If key hit, then get key.

; 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.

; Escape characters.
; ******************
ESCD        EQU         0441BH      ; Set page to display.
```

```
ESCJ        EQU         04A1BH      ; Move cursor to x,y.
ESCK        EQU         04B1BH      ; Delete x number of characters.
ESCL        EQU         04C1BH      ; Set intensity to x.
ESCP        EQU         0501BH      ; Set page to which data is written.

; Keys from keyboard.
; *******************
ESC_KEY     EQU         1B00H       ; The ESC key.
CR_KEY      EQU         0D00H       ; The ENTER or RETURN key.
REV_TAB     EQU         000FH       ; The REVERSE TAB key
TAB_KEY     EQU         0900H       ; The TAB key.

; Routine constants and flags.
; ****************************
DIR_FWD     EQU         01          ; Highlight next option.
DIR_REV     EQU         02          ; Highlight previous option.
NORMATR     EQU         01          ; Normal intensity.
NO_MENU     EQU         0FFH        ; No error when printing new menu.
PAGE0       EQU         00          ; Select page 0.
PAGE1       EQU         01          ; Select page 1.
XORATR      EQU         02          ; XOR type intensity.
ZERO        EQU         00          ; Just zero.

; Menu and character constants.
; *****************************
MAXLEN      EQU         12          ; Maximum length of option.
SCR_X       EQU         2           ; X co-ord of command headers.
XCH         EQU         09          ; Character width in graphics mode.
XMENLIM     EQU         571         ; X co-ord limit for menu.
XMENST      EQU         92          ; X co-ord for start of menu.
XPOSLIM     EQU         715         ; X limit for printing chars.
YBRIEF      EQU         331         ; Y co-ord for brief.
YCH         EQU         14          ; Character height in graphics mode.
YMENST      EQU         287         ; Y co-ord for start of menu.
YPOSLIM     EQU         347         ; Y limit for printing chars.

; GRAPHIX routine function numbers.
; *********************************
BFIL        EQU         04AH        ; Fill a block.
CLEAR       EQU         042H        ; Clear the screen.
DISPLAY     EQU         045H        ; Display a page.
LINE        EQU         049H        ; Draw a line.
GRAPHM      EQU         040H        ; Define graphics mode.
WRPAGE      EQU         043H        ; Page to which data is written.
ATRIB       EQU         044H        ; Intensity level.
CURSOR      EQU         048H        ; Move the imaginary cursor.
TEXT        EQU         04BH        ; Output text to the page.
TEXTM       EQU         041H        ; To switch to text mode.

; Tone constants and addresses.
; *****************************
FREQU       EQU         480         ; Tone frequency value.
TONADR      EQU         042H        ; The tone port address.
TONE_PORT   EQU         061H        ; Tone enable port.
SET_NOTONE  EQU         0FCH        ; Switch the tone off.
SET_TONE    EQU         003H        ; Switch the tone on.
TONCONT     EQU         0B6H        ; Tone control word.
TONELEN     EQU         02          ; Tone duration. (TONELEN/18 secs)
TONOFFSET   EQU         070H        ; The tone interrupt routine : offset.
TONSEG      EQU         072H        ; the tone interrupt routine : segment.
CONT8253    EQU         043H        ; 8253 control port.

; Disk check constants.
; *********************
DRVA        EQU         00H         ; Disk drive A.
DRVD        EQU         03H         ; Disk drive D.
HEAD0       EQU         00H         ; Drive head 0.
NUMSECT     EQU         01H         ; Number of sectors.
NO_DISK     EQU         01H         ; Time out disk error code
NO_ERROR    EQU         00H         ; No disk error return code.
PROT_ERR    EQU         03H         ; Write protect error return code.
READSECT    EQU         02H         ; Function code to read sector.
```

```
RESET_DISK  EQU         00H         ; To reset the disk drive.
SECT1       EQU         01H         ; Sector 0.
TIME_OUT    EQU         080H        ; Time out code from BIOS.
TRACK0      EQU         00H         ; Track 0.
WRIT_PROT   EQU         02H         ; Disk is write protected.
WRITSECT    EQU         03H         ; Function code to write sector.
UNDEF_ERR   EQU         03H         ; Undefined disk error return code.

; General data segment.
; *********************
DATA        SEGMENT PUBLIC 'DATA'
DISPAGE     DW          0
WRITPG      DW          0
ATTPG       DW          0
DRIVE_NO    DW          0
DATA        ENDS

; Interrupt vector segment.
; *************************
INTRSEG     SEGMENT     AT  0
INTRSEG     ENDS

; Menu data areas.
; ****************
TABLE_DATA  SEGMENT WORD PUBLIC 'FAR_DATA'
TABLE_DATA  ENDS

; ***************************************************************************
MENU_DATA   SEGMENT WORD PUBLIC 'FAR_DATA'
XPOS        DW          0           ; PRSTR : X co-ordinate.
YPOS        DW          0           ; PRSTR : Y co-ordinate.
TONE_COUNT  DW          0           ; Count for error tone.
TONEOFF     DW          0           ; Old tone vector offset.
TONESEG     DW          0           ; Old tone vector segment.
DTAOFF      DW          0           ; DTA address offset.
DTASEG      DW          0           ; DTA address segment.
CLR0_MSG    DW          ESCP,0,ESCL,1,ESCJ,4,14,ESCK,1422
            DW          ESCJ,4,265,ESCK,79,0
CLR1_MSG    DW          ESCP,1,ESCL,1,ESCJ,4,14,ESCK,1738
            DW          ESCJ,4,313,ESCK,79,0
CLR2_MSG    DW          ESCP,0,ESCL,1,ESCJ,4,110,ESCK,948
            DW          ESCJ,4,265,ESCK,79,0
MENU_DATA   ENDS

; ***************************************************************************
DGROUP      GROUP DATA
CODE        SEGMENT 'CODE'
            ASSUME CS:CODE
            ASSUME DS:DGROUP
            ASSUME ES:DGROUP

;---------------------------------------------------------------------------
ARC         PROC FAR
; Function: To draw a quater circle.
; Inputs:   FORTRAN : call ARC(x,y,radius,quadrant)
;           All passed parameters are of type : INTEGER*2
; Outputs:  Arc on the screen.
; Calls:    INT10.
; Destroys: Flags.
;
; Description: Draws a quarter circle in the quad specified. The quad is
;              defined as follows :-
;
;                           2  |  1
;                          ----------
;                           3  |  4
;---------------------------------------------------------------------------
            PUSH        BP                      ; Save the registers.
            PUSH        DI                      ;
            PUSH        ES                      ;
            PUSH        AX                      ;
            PUSH        BX                      ;
```

```
            PUSH        CX                      ;
            MOV         BP,SP                   ;
            LES         BX,DWORD PTR [BP+28]    ; Load X co-ord.
            MOV         DI,WORD PTR ES:[BX]     ;
            LES         BX,DWORD PTR [BP+20]    ; Load radius.
            MOV         CX,WORD PTR ES:[BX]     ;
            LES         BX,DWORD PTR [BP+16]    ; Load quadrant.
            MOV         AX,WORD PTR ES:[BX]     ;
            LES         BX,DWORD PTR [BP+24]    ; Load Y co-ord.
            MOV         BP,WORD PTR ES:[BX]     ;
            MOV         BX,CX                   ; Transfer radius.
            MOV         AH,4CH                  ;
            CALL        FAR PTR INT10           ; Draw arc.
            POP         CX                      ; Restore the registers.
            POP         BX                      ;
            POP         AX                      ;
            POP         ES                      ;
            POP         DI                      ;
            POP         BP                      ;
            RET         10H                     ; Exit to FORTRAN.
ARC         ENDP

;-------------------------------------------------------------------------------
BLKFIL      PROC FAR
; Function: To fill a rectangular block.
; Inputs:   FORTRAN : call BLKFIL(x,y,width,height)
;           All parameters are of the type : INTEGER*2
; Outputs:  To the graphics screen.
; Calls:    INT10.
; Destroys: Flags.
;
; Description: Fills the rectangular block, where the  given  x,y  co-ords
;              are the position of the lower left corner of the rectangle.
;-------------------------------------------------------------------------------
            PUSH        BP                      ; Save the registers.
            PUSH        DI                      ;
            PUSH        ES                      ;
            PUSH        AX                      ;
            PUSH        BX                      ;
            PUSH        CX                      ;
            MOV         BP,SP                   ;
            LES         BX,DWORD PTR [BP+28]    ; Load X co-ord.
            MOV         DI,WORD PTR ES:[BX]     ;
            LES         BX,DWORD PTR [BP+20]    ; Load width.
            MOV         CX,WORD PTR ES:[BX]     ;
            LES         BX,DWORD PTR [BP+16]    ; Load height.
            MOV         AX,WORD PTR ES:[BX]     ;
            LES         BX,DWORD PTR [BP+24]    ; Load Y co-ord.
            MOV         BP,WORD PTR ES:[BX]     ;
            MOV         BX,AX                   ;
            MOV         AH,BFIL                 ;
            CALL        FAR PTR INT10           ; Draw the block.
            POP         CX                      ; Restore the registers.
            POP         BX                      ;
            POP         AX                      ;
            POP         ES                      ;
            POP         DI                      ;
            POP         BP                      ;
            RET         10H                     ; Exit to FORTRAN.
BLKFIL      ENDP

;-------------------------------------------------------------------------------
BOX         PROC FAR
; Function: To print a rectangle.
; Inputs:   FORTRAN : call BOX(x,y,width,height)
;           x,y,width,height - type : INTEGER*2
; Outputs:  Box on the graphics screen.
; Calls:    INT10H.
; Destroys: Flags.
;
; Description: Uses the values given to draw a box on the graphics
;              screen.
```

```
;-------------------------------------------------------------------------------
            PUSH        BP                      ; Save the registers.
            PUSH        DI                      ;
            PUSH        ES                      ;
            PUSH        AX                      ;
            PUSH        BX                      ;
            PUSH        CX                      ;
            PUSH        DX                      ;
            MOV         BP,SP                   ;
            LES         BX,DWORD PTR [BP+30]    ;
            MOV         AX,WORD PTR ES:[BX]     ;
            LES         BX,DWORD PTR [BP+22]    ;
            MOV         CX,WORD PTR ES:[BX]     ;
            LES         BX,DWORD PTR [BP+18]    ;
            MOV         DX,WORD PTR ES:[BX]     ;
            LES         BX,DWORD PTR [BP+26]    ;
            MOV         DI,WORD PTR ES:[BX]     ;
            MOV         BX,DI                   ;
            MOV         DI,AX                   ; Load X co-ord.
            MOV         BP,BX                   ; Load Y co-ord.
            MOV         AH,CURSOR               ; Move the graphics cursor.
            CALL        FAR PTR INT10           ;
            MOV         AH,LINE                 ; Draw bottom horizontal.
            ADD         DI,CX                   ;
            CALL        FAR PTR INT10           ;
            SUB         BP,DX                   ; Draw right vertical.
            CALL        FAR PTR INT10           ;
            SUB         DI,CX                   ; Draw top horizontal.
            CALL        FAR PTR INT10           ;
            ADD         BP,DX                   ; Draw left vertical.
            CALL        FAR PTR INT10           ;
            POP         DX                      ; Restore registers.
            POP         CX                      ;
            POP         BX                      ;
            POP         AX                      ;
            POP         ES                      ;
            POP         DI                      ;
            POP         BP                      ;
            RET         10H                     ; Exit.
BOX         ENDP

;-------------------------------------------------------------------------------
CIRC        PROC FAR
; Function: To print a circle on the graphics screen.
; Inputs:   FORTRAN : call CIRC(x,y,radius)
;           All parameters are of the type : INTEGER*2
; Outputs:  To the screen.
; Calls:    INT10.
; Destroys: Flags.
;
; Description: Draws a circle centred at x,y and with the given radius.
;-------------------------------------------------------------------------------
            PUSH        BP                      ; Save the registers.
            PUSH        DI                      ;
            PUSH        ES                      ;
            PUSH        AX                      ;
            PUSH        BX                      ;
            MOV         BP,SP                   ;
            LES         BX,DWORD PTR [BP+22]    ; Load X co-ord.
            MOV         DI,WORD PTR ES:[BX]     ;
            LES         BX,DWORD PTR [BP+14]    ; Load the radius.
            MOV         AX,WORD PTR ES:[BX]     ;
            LES         BX,DWORD PTR [BP+18]    ; Load the Y co-ord.
            MOV         BP,WORD PTR ES:[BX]     ;
            MOV         BX,AX                   ; Transfer radius.
            MOV         AH,4DH                  ;
            CALL        FAR PTR INT10           ; Draw the circle.
            POP         BX                      ; Restore the registers.
            POP         AX                      ;
            POP         ES                      ;
            POP         DI                      ;
            POP         BP                      ;
```

```
            RET         0CH                     ; Exit to FORTRAN.
CIRC        ENDP

;-------------------------------------------------------------------------------
CLRSCR      PROC FAR
; Function: To clear the page.
; Inputs:   FORTRAN : call CLRSCR
; Outputs:  To the screen.
; Calls:    INT10.
; Destroys: Flags.
;
; Description: Clears the current page to which data is being written.
;-------------------------------------------------------------------------------
            PUSH        AX                      ; Save the registers.
            MOV         AH,CLEAR                ;
            CALL        FAR PTR INT10           ; Clear the page.
            POP         AX                      ; Restore the registers.
            RET                                 ; Exit to FORTRAN.
CLRSCR      ENDP

;-------------------------------------------------------------------------------
DEFDRV      PROC FAR
; Function: To determine the current default drive.
; Inputs:   FORTRAN : drive = DEFDRV()
; Outputs:  drive - (integer*2) Drive number (A=0, B=1, C=2,...)
; Calls:    INT21H.
; Destroys: Flags.
;
; Description: Uses DOS call to determine default drive.
;-------------------------------------------------------------------------------
            MOV         AH,19H                  ; DOS function number.
            INT         21H                     ; Call to DOS.
            XOR         AH,AH                   ;
            RET                                 ;
DEFDRV      ENDP

;-------------------------------------------------------------------------------
DISP        PROC FAR
; Function: To display a page on the screen.
; Inputs:   FORTRAN : call DISP(page)
;           Parameter of the type : INTEGER*2  (Range: 0 or 1)
; Outputs:  To the screen.
; Calls:   INT10.
; Destroys: Flags.
;
; Description: Displays either page 0 or 1 on the graphics screen.
;-------------------------------------------------------------------------------
            PUSH        BP                      ; Save the registers.
            PUSH        ES                      ;
            PUSH        AX                      ;
            PUSH        BX                      ;
            MOV         BP,SP                   ;
            LES         BX,DWORD PTR [BP+12]    ; Load the page number.
            MOV         AX,WORD PTR ES:[BX]     ;
            AND         AX,1                    ;
            MOV         AH,DISPLAY              ; Display the page.
            CALL        FAR PTR INT10           ;
            POP         BX                      ; Restore the registers.
            POP         AX                      ;
            POP         ES                      ;
            POP         BP                      ;
            RET         04H                     ; Exit to FORTRAN.
DISP        ENDP

;-------------------------------------------------------------------------------
DLINE       PROC FAR
; Function: Draws a line on the current page.
; Inputs:   FORTRAN : call DLINE(x,y)
;           All parameters of the type : INTEGER*2
; Outputs:  To the current page.
; Calls:   INT10.
; Destroys: Flags.
```

```
;
; Description: Draws a line from the current cursor position to the
;              x,y co-ords given.
;-------------------------------------------------------------------------------
            PUSH        BP                      ; Save the registers.
            PUSH        DI                      ;
            PUSH        ES                      ;
            PUSH        AX                      ;
            PUSH        BX                      ;
            MOV         BP,SP                   ;
            LES         BX,DWORD PTR [BP+18]    ; Load X co-ord.
            MOV         DI,WORD PTR ES:[BX]     ;
            LES         BX,DWORD PTR [BP+14]    ; Load Y co-ord.
            MOV         BP,WORD PTR ES:[BX]     ;
            MOV         AH,LINE                 ;
            CALL        FAR PTR INT10           ; Draw the line.
            POP         BX                      ; Restore the registers.
            POP         AX                      ;
            POP         ES                      ;
            POP         DI                      ;
            POP         BP                      ;
            RET         08H                     ; Exit to FORTRAN.
DLINE       ENDP

;-------------------------------------------------------------------------------
DO_TONE     PROC FAR
; Function: Switches the error tone on/off.
; Inputs:   MENU_DATA:TONE_COUNT - The tone decrement counter.
; Outputs:  Error tone on/off.
; Calls:    None.
; Destroys: Flags.
;
; Description: This is an interrupt routine that is executed  at approx.
;              18 Hz. The counter : TONE_COUNT is decremented every time
;              this routine is executed and the counter is  greater than
;              zero. If the counter is zero, the  tone  is switched off,
;              else the tone is switched on.
;-------------------------------------------------------------------------------
            PUSH        ES                      ;
            PUSH        AX                      ;
            ASSUME      ES:MENU_DATA            ;
            MOV         AX,MENU_DATA            ;
            MOV         ES,AX                   ;
            CMP         WORD PTR ES:TONE_COUNT,ZERO; The tone is turned  on as
            JE          TONE_OFF                ;    long as the tone count is
            IN          AL,TONE_PORT            ;    greater than zero.
            OR          AL,SET_TONE             ;    Each time the  tone  count
            OUT         TONE_PORT,AL            ;    is checked, the tone count
            DEC         WORD PTR ES:TONE_COUNT  ;    is decremented by one.
            JMP         TONE_EXIT               ;
TONE_OFF:   IN          AL,TONE_PORT            ; Switch the tone off.
            AND         AL,SET_NOTONE           ;
            OUT         TONE_PORT,AL            ;
TONE_EXIT:                                      ;
            ASSUME      ES:DGROUP               ;
            POP         AX                      ;
            POP         ES                      ;
            IRET                                ;
DO_TONE     ENDP

;-------------------------------------------------------------------------------
ERTONE      PROC FAR
; Function: To set the error tone on.
; Inputs:   None.
; Outputs:  Error tone will sound.
; Calls:    None.
; Destroys: Flags.
;
; Description: Sets the time count which informs the Timer interrupt
;              to switch the tone on for the period of the count.
;-------------------------------------------------------------------------------
            PUSH        ES                      ; Save the registers.
```

```
            PUSH        AX                          ;
            ASSUME      ES:MENU_DATA                ; Use MENU_DATA segment.
            MOV         AX,MENU_DATA                ;
            MOV         ES,AX                       ;
            MOV         WORD PTR ES:TONE_COUNT,TONELEN; Load the tone length.
            ASSUME      ES:DGROUP                   ;
            POP         AX                          ; Restore the registers.
            POP         ES                          ;
            RET                                     ; Exit.
ERTONE      ENDP

;-------------------------------------------------------------------------------
FFIRST      PROC FAR
; Function: To find first file in directory search.
; Inputs:   FORTRAN : find = FFIRST(len1,str1,len2,str2)
;           len1 - (integer*2) Length of input search string.
;           str1 - (character*len1) Search string.
;
;           NB: THE  DEFINED  LENGTH  OF "STR1" SHOULD BE ("LEN1"+1) AS
;               THIS ROUTINE HAS TO CONVERT "STR1" TO AN ASCIIZ STRING.
;
; Outputs:  find - (integer*2) Return code ( 0 - successful,
;                                           -1 - Fail or error.)
;           len2 - (integer*2) Length of returned string.
;           str2 - (character*len2) Filename if search successful.
; Calls:    INT21H, CHKDIR.
; Destroys: Flags.
;
; Description: Finds address of DTA first, converts the string to ASCIIZ next.
;              Then attempts to find the first file.  Sets 'find' accordingly,
;              and  if  the  search  successful,  calculates the length of the
;              string.
;-------------------------------------------------------------------------------
            PUSH        BP                          ; Save the registers.
            PUSH        DI                          ;
            PUSH        DS                          ;
            PUSH        ES                          ;
            PUSH        BX                          ;
            PUSH        CX                          ;
            PUSH        DX                          ;
            MOV         BP,SP                       ; Load stack pointer.
            ASSUME      DS:MENU_DATA                ; Use MENU_DATA segment.
            MOV         AX,MENU_DATA                ;
            MOV         DS,AX                       ;
            MOV         AH,2FH                      ; Locate the DTA by using a
            INT         21H                         ; DOS call.
            MOV         WORD PTR DS:DTAOFF,BX       ; Save the address of the DTA.
            MOV         AX,ES                       ;
            MOV         WORD PTR DS:DTASEG,AX       ;
            LES         BX,DWORD PTR [BP+30]        ; Load len1.
            MOV         AX,WORD PTR ES:[BX]         ;
            CMP         AX,0                        ; If (len1 <= 0) then set error
            JLE         SETFERR                     ;                      and EXIT.
            MOV         DI,AX                       ; Keep a copy of len1.
            LES         BX,DWORD PTR [BP+26]        ; Load str1 ptr to (ES:BX).
            MOV         BYTE PTR ES:[BX+DI],0       ; Convert string to ASCIIZ.
            MOV         DX,BX                       ; Load up registers for DOS
            MOV         BX,ES                       ; call to perform FIND FIRST.
            MOV         DS,BX                       ;
            MOV         CX,14H                      ;
            MOV         AH,4EH                      ;
            INT         21H                         ; DOS call.
            JC          SETFERR                     ; If (carry set) then set error
            ASSUME      ES:MENU_DATA                ;                      and EXIT.
            MOV         AX,MENU_DATA                ;
            MOV         ES,AX                       ; Use MENU_DATA segment.
            MOV         AX,WORD PTR ES:DTASEG       ; Retrieve DTA address
            MOV         DS,AX                       ; to (DS:AX).
            MOV         AX,WORD PTR ES:DTAOFF       ;
            LES         BX,DWORD PTR [BP+18]        ; Load str2 ptr to (ES:BX)
            MOV         DI,0                        ; Initialise counter:DI.
                                                    ; Repeat
```

```
NEXTFCH:    XCHG        AX,BX                       ;
            MOV         DL,BYTE PTR DS:[BX+DI+30];  Load file name char
            XCHG        AX,BX                       ;   from DTA.
            CMP         DL,0                        ;
            JE          ENDFCH                      ;
            MOV         BYTE PTR ES:[BX+DI],DL      ;   Store filename char in
                                                    ;   return string.
            INC         DI                          ;   Increment counter:DI.
            JMP         NEXTFCH                     ; Until (char = 0).
                                                    ; (Returned str is not ASCIIZ)
ENDFCH:     CALL        FAR PTR CHKDIR              ;
            LES         BX,DWORD PTR [BP+22]        ; Load len2 ptr.
            MOV         WORD PTR ES:[BX],DI         ; Store len2.
            JMP         SETF_OK                     ; Set OK return value and EXIT.
SETFERR:    MOV         AX,-1                       ; Set ERROR return.
            JMP         FIRSTEXIT                   ;
SETF_OK:    MOV         AX,0                        ;
FIRSTEXIT:                                          ;
            ASSUME      ES:DGROUP                   ;
            ASSUME      DS:DGROUP                   ;
            POP         DX                          ; Restore the registers.
            POP         CX                          ;
            POP         BX                          ;
            POP         ES                          ;
            POP         DS                          ;
            POP         DI                          ;
            POP         BP                          ;
            RET         10H                         ; Pop the stack and EXIT.
FFIRST      ENDP

;-------------------------------------------------------------------------------
FNEXT       PROC FAR
; Function: To find the next file in the directory search.
; Inputs:   FORTRAN : find = FNEXT(len2,str2)
; Outputs:  find - (integer*2) Return code ( 0 - successful,
;                                           -1 - Fail or error.)
;           len2 - (integer*2) Length of returned string.
;           str2 - (character*len2) Filename if search successful.
; Calls:    INT21H, CHKDIR.
; Destroys: Flags.
;
; Description: Finds the next file in the directory search using the DTA
;              set up in FFIRST.
;-------------------------------------------------------------------------------
            PUSH        BP                          ; Save the registers.
            PUSH        DI                          ;
            PUSH        DS                          ;
            PUSH        ES                          ;
            PUSH        BX                          ;
            PUSH        CX                          ;
            PUSH        DX                          ;
            MOV         BP,SP                       ; Load the stack pointer.
            ASSUME      ES:MENU_DATA                ; Use the MENU_DATA segment.
            MOV         AX,MENU_DATA                ;
            MOV         ES,AX                       ;
            MOV         AH,4FH                      ; DOS call to perform the
            INT         21H                         ; FIND NEXT function.
            JC          SETNERR                     ; If (carry set) then set error
                                                    ;                   and EXIT.
            MOV         AX,WORD PTR ES:DTASEG       ; Load the DTA address.
            MOV         DS,AX                       ;
            MOV         AX,WORD PTR ES:DTAOFF       ;
            LES         BX,DWORD PTR [BP+18]        ; Load str2 ptr to (ES:BX).
            MOV         DI,0                        ; Initialise counter:DI.
                                                    ; Repeat
NEXTNCH:    XCHG        AX,BX                       ;
            MOV         DL,BYTE PTR DS:[BX+DI+30];  Load filename char from
            XCHG        AX,BX                       ;   the DTA.
            CMP         DL,0                        ;
            JE          ENDNCH                      ;
            MOV         BYTE PTR ES:[BX+DI],DL      ;   Store the filename char to
                                                    ;   str2 for return.
```

```
            INC         DI                          ;   Increment counter:DI.
            JMP         NEXTNCH                     ; Until (char = 0).
ENDNCH:     CALL        FAR PTR CHKDIR              ;
            LES         BX,DWORD PTR [BP+22]        ; Load len2 ptr to (ES:BX).
            MOV         WORD PTR ES:[BX],DI         ; Store the length to len2.
            JMP         SETN_OK                     ; Set OK return and EXIT.
SETNERR:    MOV         AX,-1                       ; Set ERROR return.
            JMP         FNEXTEXIT                   ;
SETN_OK:    MOV         AX,0                        ;
FNEXTEXIT:                                          ;
            ASSUME      ES:DGROUP                   ;
            ASSUME      DS:DGROUP                   ;
            POP         DX                          ; Restore the registers.
            POP         CX                          ;
            POP         BX                          ;
            POP         ES                          ;
            POP         DS                          ;
            POP         DI                          ;
            POP         BP                          ;
            RET         08H                         ; Pop the stack and EXIT.
FNEXT       ENDP

;-------------------------------------------------------------------------------
CHKDIR      PROC FAR
; Function: To check if file name returned by FFIRST and FNEXT is a directory.
; Inputs:   DS:AX - DTA set up by FFIRST or FNEXT.
;           ES:BX - string or filename.
;           DI    - char counter.
; Outputs:  <DIR> - appended to string if the file is directory.
;           DI    - char counter set to 14.
; Calls:    None.
; Destroys: Flags.
;
; Description: Checks the file attribute returned in the DTA (byte 22). If
;              the 5 bit is set, then the file is a directory.
;-------------------------------------------------------------------------------
            PUSH        AX                          ; Save the registers.
            PUSH        BX                          ;
            PUSH        CX                          ;
            XCHG        AX,BX                       ;
            MOV         CL,BYTE PTR DS:[BX+21]      ; Load file attribute.
            AND         CL,10H                      ;
            CMP         CL,10H                      ; If (file_type <> DIR) then
            JNE         DIREXIT                     ;                       EXIT.
            XCHG        AX,BX                       ;
DIRBLNK:    CMP         DI,9                        ; Blank out rest of string
            JGE         DODIRTXT                    ; until char 10.
            MOV         BYTE PTR ES:[BX+DI],' '     ;
            INC         DI                          ;
            JMP         DIRBLNK                     ;
DODIRTXT:                                           ;
            MOV         BYTE PTR ES:[BX+9],'<'      ; Insert the "<DIR>."
            MOV         BYTE PTR ES:[BX+10],'D'     ;
            MOV         BYTE PTR ES:[BX+11],'I'     ;
            MOV         BYTE PTR ES:[BX+12],'R'     ;
            MOV         BYTE PTR ES:[BX+13],'>'     ;
            MOV         DI,14                       ;
DIREXIT:                                            ;
            POP         CX                          ; Restore the registers.
            POP         BX                          ;
            POP         AX                          ;
            RET                                     ; Exit.
CHKDIR      ENDP

;-------------------------------------------------------------------------------
FILL        PROC FAR
; Function: To fill a polygon.
; Inputs:   FORTRAN : call FILL(x,y)
;           x,y - (integer*2) X,Y co-ordinates of seed in polygon.
; Outputs:  To screen.
; Calls:    INT10.
; Destroys: Flags.
```

```
;
; Description: Fill the polygon with white if black (or vica-versa). The
;              polygon should not contain islands or peninsulas.
;-------------------------------------------------------------------------------
            PUSH        BP                      ; Save the registers.
            PUSH        DI                      ;
            PUSH        ES                      ;
            PUSH        AX                      ;
            PUSH        BX                      ;
            MOV         BP,SP                   ; Load stack pointer.
            LES         BX,DWORD PTR [BP+18]    ; Load x co-ord.
            MOV         DI,WORD PTR ES:[BX]     ;
            LES         BX,DWORD PTR [BP+14]    ; Load y co-ord.
            MOV         BP,WORD PTR ES:[BX]     ;
            MOV         AH,4EH                  ; Set function number.
            CALL        FAR PTR INT10           ; Do call.
            POP         BX                      ; Restore the registers.
            POP         AX                      ;
            POP         ES                      ;
            POP         DI                      ;
            POP         BP                      ;
            RET         08H                     ; Pop the stack and EXIT.
FILL        ENDP

;-------------------------------------------------------------------------------
GETPG       PROC FAR
; Function: To return page being displayed.
; Inputs:   FORTRAN : page = GETPG()
; Outputs:  page - (integer*2) Page being displayed.
; Calls:    None.
; Destroys: Flags.
;
; Description: Puts value of DATA:DISPAGE in AX and returns.
;-------------------------------------------------------------------------------
            PUSH        ES                      ; Save the register.
            ASSUME      ES:DGROUP               ; Use DGROUP segment.
            MOV         AX,DGROUP               ;
            MOV         ES,AX                   ;
            MOV         AX,WORD PTR ES:DISPAGE  ; Load the display page var.
            POP         ES                      ; Restore register.
            RET                                 ; Exit.
GETPG       ENDP

;-------------------------------------------------------------------------------
GETWPG      PROC FAR
; Function: To return page to which data is written.
; Inputs:   FORTRAN : page = GETWPG()
; Outputs:  page - (integer*2) Page being to which data is written.
; Calls:    None.
; Destroys: Flags.
;
; Description: Puts value of DATA:WRITPG in AX and returns.
;-------------------------------------------------------------------------------
            PUSH        ES                      ; Save the register.
            ASSUME      ES:DGROUP               ; Use DGROUP segment.
            MOV         AX,DGROUP               ;
            MOV         ES,AX                   ;
            MOV         AX,WORD PTR ES:WRITPG   ; Load the write page var.
            POP         ES                      ; Restore register.
            RET                                 ; Exit.
GETWPG      ENDP

;-------------------------------------------------------------------------------
GETATT      PROC FAR
; Function: To attribute of current page.
; Inputs:   FORTRAN : att = GETATT()
; Outputs:  att - (integer*2) Page attribute.
; Calls:    None.
; Destroys: Flags.
;
; Description: Puts value of DATA:ATTPG in AX and returns.
;-------------------------------------------------------------------------------
```

```
            PUSH        ES                      ; Save the register.
            ASSUME      ES:DGROUP               ; Use DGROUP segment.
            MOV         AX,DGROUP               ;
            MOV         ES,AX                   ;
            MOV         AX,WORD PTR ES:ATTPG    ; Load the attribute variable.
            POP         ES                      ; Restore register.
            RET                                 ; Exit.
GETATT      ENDP

;-------------------------------------------------------------------------------
GMODE       PROC FAR
; Function: To switch to graphics mode.
; Inputs:   FORTRAN : call GMODE()
; Outputs:  Altered mode.
; Calls:    INT10.
; Destroys: Flags.
;
; Description: Sets AX register to function call, then executes INT10.
;-------------------------------------------------------------------------------
            PUSH        AX                      ;
            MOV         AH,GRAPHM               ;
            CALL        FAR PTR INT10           ;
            POP         AX                      ;
            RET
GMODE       ENDP

;-------------------------------------------------------------------------------
GPAGE       PROC FAR
; Function: Selects page to which data is written.
; Inputs:   FORTRAN : call GPAGE(page)
;           Parameter is of type : INTEGER*2   (Range : 0 or 1)
; Outputs:  To the current page.
; Calls:    INT10.
; Destroys: Flags.
;
; Description: All data is now written, subsequent to GPAGE, to the page
;              specified as the parameter.
;-------------------------------------------------------------------------------
            PUSH        BP                      ; Save the registers.
            PUSH        ES                      ;
            PUSH        AX                      ;
            PUSH        BX                      ;
            MOV         BP,SP                   ;
            LES         BX,DWORD PTR [BP+12]    ; Load the page number.
            MOV         AX,WORD PTR ES:[BX]     ;
            AND         AX,1                    ;
            MOV         AH,WRPAGE               ;
            CALL        FAR PTR INT10           ; Set the page number.
            POP         BX                      ; Restore the registers.
            POP         AX                      ;
            POP         ES                      ;
            POP         BP                      ;
            RET         04H                     ; Exit to FORTRAN.
GPAGE       ENDP

;-------------------------------------------------------------------------------
INTONE      PROC FAR
; Function: To initialise the tone interrupt vector.
; Inputs:   None.
; Outputs:  Modified interrupt vector 1CH.
;           Old vector stored : MENU_DATA:TONEOFF
;                               MENU_DATA:TONESEG
; Calls:    None.
; Destroys: Flags.
;
; Description: Replaces the interrupt vector with the address of the tone
;              interrupt routine : DO_TONE.
;-------------------------------------------------------------------------------
            CLI                                 ; Hold interrupts.
            PUSH        AX                      ; Save status of processor.
            PUSH        ES                      ;
            PUSH        DS                      ;
```

```
                ASSUME      DS:MENU_DATA            ;
                MOV         AX,MENU_DATA            ;
                MOV         DS,AX                   ;
                                                    ;
                MOV         AX,INTRSEG              ; Load interrupt vector seg.
                MOV         ES,AX                   ;
                                                    ; 18 Hz SYSTEM INTERRUPT.
                MOV         AX,WORD PTR ES:[TONOFFSET]; Save old offset.
                MOV         WORD PTR DS:TONEOFF,AX  ;
                MOV         AX,WORD PTR ES:[TONSEG] ;   Save old segment.
                MOV         WORD PTR DS:TONESEG,AX  ;
                MOV         AX,OFFSET DO_TONE       ;   Load new offset.
                MOV         WORD PTR ES:[TONOFFSET],AX;
                MOV         WORD PTR ES:[TONSEG],CS ;   Load new segment.
                                                    ;
                MOV         AL,TONCONT              ; Setting tone pitch.
                OUT         CONT8253,AL             ;
                MOV         AL,LOW FREQU            ;
                OUT         TONADR,AL               ;
                MOV         AL,HIGH FREQU           ;
                OUT         TONADR,AL               ;
                                                    ;
                POP         DS                      ; Retrieve processor status.
                POP         ES                      ;
                POP         AX                      ;
                STI                                 ;
                RET                                 ;
INTONE          ENDP

;-------------------------------------------------------------------------
INT10           PROC FAR
; Function: Routine to preserve registers and to maintain some state variables.
; Inputs:   Call variables to INT10H.
; Outputs:  DGROUP:WRITPG - Page to which data is written.
;            DGROUP:ATTPG - The current write attribute.
;          DGROUP:DISPAGE - The page currently displayed.
;
; Calls:    INT10H.
; Destroys: Flags.
;
; Description: The routine saves all the registers  and  then checks if the
;              operation is one of the following :-
;               WRPAGE - Page to which data is written,
;                ATRIB - Write attribute,
;              DISPLAY - Page which is diplayed.
;
;              If so, the routine saves the state in the  appropriate state
;              variable and then executes the function.
;-------------------------------------------------------------------------
                PUSH        BP                      ; Save the registers.
                PUSH        DS                      ;
                PUSH        ES                      ;
                PUSH        DI                      ;
                PUSH        SI                      ;
                PUSH        AX                      ;
                PUSH        BX                      ;
                PUSH        CX                      ;
                PUSH        DX                      ;
                CMP         AH,WRPAGE               ; If (operation = GPAGE) then
                JNE         NOTWRIT                 ;
                PUSH        ES                      ;    Save registers.
                PUSH        AX                      ;
                PUSH        BX                      ;
                ASSUME      ES:DGROUP               ;    Use DGROUP segment.
                MOV         BX,DGROUP               ;
                MOV         ES,BX                   ;
                XOR         AH,AH                   ;
                MOV         WORD PTR ES:WRITPG,AX   ;    Save page number to which
                POP         BX                      ;    data is written.
                POP         AX                      ;
                POP         ES                      ;
                JMP         DOINT10                 ;
```

```
NOTWRIT:                                        ; Elseif (operation = LEVEL)
            CMP         AH,ATRIB                ;   then
            JNE         NOTATT                  ;
            PUSH        ES                      ;    Save registers.
            PUSH        AX                      ;
            PUSH        BX                      ;
            ASSUME      ES:DGROUP               ;    Use DGROUP segment.
            MOV         BX,DGROUP               ;
            MOV         ES,BX                   ;
            XOR         AH,AH                   ;
            MOV         WORD PTR ES:ATTPG,AX    ;    Save write attribute.
            POP         BX                      ;
            POP         AX                      ;
            POP         ES                      ;
            JMP         DOINT10                 ;
NOTATT:                                         ; Elseif (operation = DISP)
            CMP         AH,DISPLAY              ;   then
            JNE         NOTDIS                  ;
            PUSH        ES                      ;    Save registers.
            PUSH        AX                      ;
            PUSH        BX                      ;
            ASSUME      ES:DGROUP               ;    Use DGROUP segment.
            MOV         BX,DGROUP               ;
            MOV         ES,BX                   ;
            XOR         AH,AH                   ;
            MOV         WORD PTR ES:DISPAGE,AX  ;    Save displayed page number
            POP         BX                      ;
            POP         AX                      ;
            POP         ES                      ;
            JMP         DOINT10                 ;
NOTDIS:                                         ; Endif
DOINT10:                                        ;
            INT         10H                     ; Do graphics function.
            POP         DX                      ; Restore the registers.
            POP         CX                      ;
            POP         BX                      ;
            POP         AX                      ;
            POP         SI                      ;
            POP         DI                      ;
            POP         ES                      ;
            POP         DS                      ;
            POP         BP                      ;
            RET                                 ; Exit.
INT10       ENDP

;-------------------------------------------------------------------------------
LEVEL       PROC FAR
; Function: To set the intensity level.
; Inputs:   FORTRAN : call LEVEL(intensity)
;           Parameter is of type : INTEGER*2   (Range 0, 1 or 2)
; Outputs:  To the current page.
; Calls:   INT10.
; Destroys: Flags.
;
; Description: All data written subsequent to the LEVEL call will be
;              written with intensity specified.
;-------------------------------------------------------------------------------
            PUSH        BP                      ; Save the registers.
            PUSH        ES                      ;
            PUSH        AX                      ;
            PUSH        BX                      ;
            MOV         BP,SP                   ;
            LES         BX,DWORD PTR [BP+12]    ; Load the intensity level.
            MOV         AX,WORD PTR ES:[BX]     ;
            MOV         AH,ATRIB                ;
            CALL        FAR PTR INT10           ; Set the intensity level.
            POP         BX                      ; Restore the registers.
            POP         AX                      ;
            POP         ES                      ;
            POP         BP                      ;
            RET         04H                     ; Exit to FORTRAN.
LEVEL       ENDP
```

```
;-------------------------------------------------------------------------------
MOVE        PROC FAR
; Function: To move the graphics cursor to x,y
; Inputs:   FORTRAN : call MOVE(x,y)
;           All the parameters are of the type : INTEGER*2
; Outputs:  To the screen.
; Calls:    INT10.
; Destroys: Flags.
;
; Description: The imaginary graphics cursor is moved to the location
;              specified by x,y.
;-------------------------------------------------------------------------------
            PUSH        BP                      ; Save the registers.
            PUSH        DI                      ;
            PUSH        ES                      ;
            PUSH        AX                      ;
            PUSH        BX                      ;
            MOV         BP,SP                   ;
            LES         BX,DWORD PTR [BP+18]    ; Load X co-ord.
            MOV         DI,WORD PTR ES:[BX]     ;
            LES         BX,DWORD PTR [BP+14]    ; Load Y co-ord.
            MOV         BP,WORD PTR ES:[BX]     ;
            MOV         AH,CURSOR               ;
            CALL        FAR PTR INT10           ; Move the graphics cursor.
            POP         BX                      ; Restore the registers.
            POP         AX                      ;
            POP         ES                      ;
            POP         DI                      ;
            POP         BP                      ;
            RET         08H                     ; Exit to FORTRAN.
MOVE        ENDP

;-------------------------------------------------------------------------------
PLOT        PROC FAR
; Function: To plot a point at x,y.
; Inputs:   FORTRAN : call PLOT(x,y)
;           All parameters are of the type : INTEGER*2
; Outputs:  To the screen.
; Calls:    INT10.
; Destroys: Flags.
;
; Description: A single point is printed on the current page at
;              the location specifed by x,y.
;-------------------------------------------------------------------------------
            PUSH        BP                      ; Save the registers.
            PUSH        DI                      ;
            PUSH        ES                      ;
            PUSH        AX                      ;
            PUSH        BX                      ;
            MOV         BP,SP                   ;
            LES         BX,DWORD PTR [BP+18]    ; Load X co-ord.
            MOV         DI,WORD PTR ES:[BX]     ;
            LES         BX,DWORD PTR [BP+14]    ; Load Y co-ord.
            MOV         BP,WORD PTR ES:[BX]     ;
            MOV         AH,46H                  ;
            CALL        FAR PTR INT10           ; Plot the point.
            POP         BX                      ; Restore the registers.
            POP         AX                      ;
            POP         ES                      ;
            POP         DI                      ;
            POP         BP                      ;
            RET         08H                     ; Exit to FORTRAN.
PLOT        ENDP

;-------------------------------------------------------------------------------
PRINTCH     PROC FAR
; Function: To print a char to the screen.
; Inputs:   AL : character to be printed.
; Outputs:  To the screen.
; Calls:    INT10.
; Destroys: Flags.
```

```
;
; Description: The  routine  puts  the  character  on  the  screen
;              according  to  the  screen  pointers : XPOS and Ypos.
;              After  the  write the screen pointers are incremented
;              to the next character.
;
;-------------------------------------------------------------------------
            PUSH        ES                      ; Save the registers.
            PUSH        BP                      ;
            PUSH        DI                      ;
            PUSH        AX                      ;
            PUSH        BX                      ;
            ASSUME      ES:MENU_DATA            ;
            MOV         BX,MENU_DATA            ; Load the MENU_DATA segment.
            MOV         ES,BX                   ;
            MOV         AH,TEXT                 ; Set the interrupt function.
            MOV         BX,WORD PTR ES:XPOS     ; X co-ordinate.
            MOV         DI,BX                   ;
            MOV         BX,WORD PTR ES:YPOS     ; Y co-ordinate.
            MOV         BP,BX                   ;
            CALL        FAR PTR INT10           ;
            ADD         WORD PTR ES:XPOS,XCH    ; Increment X co-ord.
            CMP         WORD PTR ES:XPOS,XPOSLIM; If X co-ord > X-limit, then
            JL          POS_OK                  ;
            MOV         WORD PTR ES:XPOS,4      ;    Set to 1 char. from edge.
            ADD         WORD PTR ES:YPOS,YCH    ;    Increment Y co-ord.
            CMP         WORD PTR ES:YPOS,YPOSLIM;    If Y co-ord > Y-limit,then
            JL          POS_OK                  ;
            MOV         WORD PTR ES:YPOS,(YCH+3);       Set to 1 char from top.
                                                ;    Endif.
                                                ; Endif.
POS_OK:     POP         BX                      ; Restore the registers.
            POP         AX                      ;
            POP         DI                      ;
            POP         BP                      ;
            POP         ES                      ;
            ASSUME      ES:DGROUP               ;
            RET                                 ; Exit.
PRINTCH     ENDP

;-------------------------------------------------------------------------
PRSTR       PROC FAR
; Function: To print a string of characters.
; Inputs:   BX : Offset of the start of the string.
;           ES : assumed to contain the correct segment for the string.
; Outputs:  To the screen.
; Calls:    PRINTCH, ESC_CH, INT10.
; Destroys: Flag, BX.
;
; Description: This routine prints out the a string of characters starting
;              at BX and ending in a 0. ie. all strings must be terminated
;              with a zero.
;              Three <ESC> sequences are available :-
;              1) ESC,'J',x,y  : To position the cursor at a position
;                                before printing the string.
;              2) ESC,'K',len  : To clear 'len' characters from the current
;                                cursor position.
;              3) ESC,'L',attr : To set the attribute for a particular
;                                string.
;              4) ESC,'P',page : To set the page to which data is written.
;              5) ESC,'D',page : To set the page that is displayed.
;
;               NOTE: x, y, len, attr and page : are all of size :WORD.
;
;-------------------------------------------------------------------------
            PUSH        AX                      ; Save the registers.
            PUSH        BX                      ;
            PUSH        CX                      ;
            MOV         AH,ATRIB                ;
            MOV         AL,NORMATR              ;
            CALL        FAR PTR INT10           ;
                                                ;
```

```
NEXT_STR:   MOV         AL,BYTE PTR ES:[BX]     ; Load char from string.
            INC         BX                      ; Increment string pointer.
            CMP         AL,ZERO                 ; If the char is zero, then
            JE          END_STR                 ; str. print is complete, exit.
            CMP         AL,ESC                  ; If char = ESC, then check esc
            JNE         NOT_ESC                 ; escape sequence for string
            CALL        FAR PTR ESC_CH          ; print.
            JMP         NEXT_STR                ;
NOT_ESC:    CALL        FAR PTR PRINTCH         ; If not zero, or an ESC char,
            JMP         NEXT_STR                ; then print the char and loop
                                                ; for the next char.
END_STR:                                        ;
            POP         CX                      ; Restore the registers.
            POP         BX                      ;
            POP         AX                      ;
            RET                                 ; End.
PRSTR       ENDP

;-------------------------------------------------------------------------------
ESC_CH      PROC FAR
; Function: To perform the escape sequences.
; Inputs:   ES:BX - the string being output.
; Outputs:  Screen pointer : for ESC J.
;           To the screen  : for ESC K.
;           AH - attribute : for ESC L.
; Calls:    PRINTCH, INT10.
; Destroys: Flags.
;
; Description: The routine detects which ESC sequence is being used
;              and performs the necessary function to pointers or screen.
;
;-------------------------------------------------------------------------------
            MOV         AL,BYTE PTR ES:[BX]     ; Load the ESC sequence char.
            INC         BX                      ; Increment the string pointer.
                                                ;
            CMP         AL,'J'                  ; Case ESC sequence char of:
            JNE         NOT_ESCJ                ;   'J' : Set the X, Y pointers
            ASSUME      DS:MENU_DATA            ;         to the values follow-
            PUSH        DS                      ;         the ESC,J sequence.
            MOV         AX,MENU_DATA            ;
            MOV         DS,AX                   ;
            MOV         AX,WORD PTR ES:[BX]     ;
            MOV         WORD PTR DS:XPOS,AX     ;
            MOV         AX,WORD PTR ES:[BX+2]   ;
            MOV         WORD PTR DS:YPOS,AX     ;
            POP         DS                      ;
            ASSUME      DS:DGROUP               ;
            ADD         BX,4                    ;
            JMP         ESC_EXIT                ;
                                                ;
NOT_ESCJ:   CMP         AL,'K'                  ;   'K' : Blank the text from
            JNE         NOT_ESCK                ;         the current X,Y ptrs.
            MOV         CX,WORD PTR ES:[BX]     ;         The number of chars
            ADD         BX,2                    ;         to be blanked is held
NEXT_BL:    MOV         AL,' '                  ;         in the word following
            CALL        FAR PTR PRINTCH         ;         the ESC, K sequence.
            LOOP        NEXT_BL                 ;
            JMP         ESC_EXIT                ;
                                                ;
NOT_ESCK:   CMP         AL,'L'                  ;   'L' : Set the string print
            JNE         NOT_ESCL                ;         intensity level.
            MOV         AX,WORD PTR ES:[BX]     ;
            ADD         BX,2                    ;
            MOV         AH,ATRIB                ;
            CALL        FAR PTR INT10           ;
            JMP         ESC_EXIT                ;
                                                ;
NOT_ESCL:   CMP         AL,'P'                  ;   'P' : Set the page to which
            JNE         NOT_ESCP                ;         data is written.
            MOV         AX,WORD PTR ES:[BX]     ;
            AND         AX,1                    ;
            ADD         BX,2                    ;
```

```
            MOV         AH,WRPAGE                   ;
            CALL        FAR PTR INT10               ;
            JMP         ESC_EXIT                    ;
                                                    ;
NOT_ESCP:   CMP         AL,'D'                      ;   'D' : Set the page to be
            JNE         NOT_ESCD                    ;         displayed.
            MOV         AX,WORD PTR ES:[BX]         ;
            ADD         BX,2                        ;
            AND         AX,1                        ;
            MOV         AH,DISPLAY                  ;
            CALL        FAR PTR INT10               ;
                                                    ;
NOT_ESCD:                                           ; Endcase.
ESC_EXIT:   RET                                     ; Exit.
ESC_CH      ENDP

;-------------------------------------------------------------------------------
RETONE      PROC FAR
; Function: To reset the tone interrupt vector.
; Inputs:   Old interrupt vector : MENU_DATA:TONEOFF
;                                  MENU_DATA:TONESEG
; Outputs:  Reset interrupt vector.
; Calls:    None.
; Destroys: Flags.
;
; Description: Replaces the interrupt vector with the stored values
;              for the offset and segment.
;-------------------------------------------------------------------------------
            CLI                                     ; Hold interrupts.
            PUSH        AX                          ; Save status of processor.
            PUSH        ES                          ;
            PUSH        DS                          ;
            ASSUME      DS:MENU_DATA                ;
            MOV         AX,MENU_DATA                ;
            MOV         DS,AX                       ;
            MOV         AX,INTRSEG                  ; Load interrupt vector seg.
            MOV         ES,AX                       ;
                                                    ;
            MOV         AX,WORD PTR DS:TONEOFF      ;   Restoring interrupt vector
            MOV         WORD PTR ES:[TONOFFSET],AX; 1C to state before program
            MOV         AX,WORD PTR DS:TONESEG      ;   execution. (18  Hz  system
            MOV         WORD PTR ES:[TONSEG],AX     ;   interrupt.)
            IN          AL,TONE_PORT                ;
            AND         AL,SET_NOTONE               ;
            OUT         TONE_PORT,AL                ;
                                                    ;
            ASSUME      DS:DGROUP                   ;
            POP         DS                          ; Retrieve processor status.
            POP         ES                          ;
            POP         AX                          ;
            STI                                     ; Restore interrupts.
            RET                                     ;
RETONE      ENDP

;-------------------------------------------------------------------------------
TMODE       PROC FAR
; Function: To switch to text mode.
; Inputs:   FORTRAN : call TMODE()
; Outputs:  Switched modes.
; Calls:    INT10.
; Destroys: Flags.
;
; Description: Loads AX register with function call, then executes INT10.
;-------------------------------------------------------------------------------
            PUSH        AX                          ;
            MOV         AH,TEXTM                    ;
            CALL        FAR PTR INT10               ;
            POP         AX                          ;
            RET                                     ;
TMODE       ENDP

;-------------------------------------------------------------------------------
```

```
TO_UPPER    PROC FAR
; Function: To convert character to upper case.
; Inputs:   AH - char to be converted.
; Outputs:  AH - converted char.
; Calls:    None.
; Destroys: Flags.
;
; Description: If the character is in the range 'a' to 'z' the char is
;              converted to upper case.
;
;-------------------------------------------------------------------------
            CMP         AH,'a'                  ;
            JL          TO_EXIT                 ;
            CMP         AH,'z'                  ;
            JG          TO_EXIT                 ;
            SUB         AH,('a'-'A')            ;
TO_EXIT:                                        ;
            RET                                 ;
TO_UPPER    ENDP

;-------------------------------------------------------------------------
WIPSCR      PROC FAR
; Function: To clear one of the graphics pages.
; Inputs:   FORTRAN : call WIPSCR(page)
;           page - (integer*2) Page to be cleared (0 or1)
; Outputs:  Cleared page.
; Calls:    PRSTR.
; Destroys: Flags.
;
; Description: Checks which page to be cleared, then clears the page
;              using blank spaces. This is different to CLRSCR, in that the
;              borders around each page are not disturbed.
;-------------------------------------------------------------------------
            PUSH        BP                      ; Save the registers.
            PUSH        ES                      ;
            PUSH        AX
            PUSH        BX                      ;
            MOV         BP,SP                   ;
            LES         BX,DWORD PTR [BP+12]    ; Load page number.
            MOV         AX,WORD PTR ES:[BX]     ;
            ASSUME      ES:MENU_DATA            ;
            MOV         BX,MENU_DATA            ;
            MOV         ES,BX                   ;
            CMP         AL,0                    ;
            JNE         CLR_P1                  ;
            MOV         BX,OFFSET MENU_DATA:CLR0_MSG;
            JMP         CLRPG                   ;
CLR_P1:     CMP         AL,1                    ;
            JNE         CLR_P2                  ;
            MOV         BX,OFFSET MENU_DATA:CLR1_MSG;
            JMP         CLRPG                   ;
CLR_P2:                                         ;
            MOV         BX,OFFSET MENU_DATA:CLR2_MSG;
CLRPG:      CALL        PRSTR                   ;
            POP         BX                      ; Restore the registers.
            POP         AX                      ;
            POP         ES                      ;
            POP         BP                      ;
            RET         04H                     ; Exit to FORTRAN.
WIPSCR      ENDP

;-------------------------------------------------------------------------
WRTSTR      PROC FAR
; Function: FORTRAN : To print a string at x,y.
; Inputs:   FORTRAN : call wrtstr(page,x,y,length,string)
;           page   - INTEGER*2 : Page to which string is written. (0 or 1)
;            x,y   - INTEGER*2 : X and Y co-ord. of start of string.
;            len   - INTEGER*2 : Length of string.
;           string - String to be printed.
; Outputs:  String on the page at x,y.
; Calls:    INT10H, PRINTCH.
; Destroys: Flags.
```

```
;
; Description: Switches to the page requested,  moves  the  cursor to  the
;              required position and prints the string. The routine leaves
;              the page set to the page specified in the parameter list.
;-------------------------------------------------------------------------
            PUSH        BP                      ; Save the registers.
            PUSH        ES                      ;
            PUSH        AX                      ;
            PUSH        BX                      ;
            PUSH        CX                      ;
            MOV         BP,SP                   ;
            LES         BX,DWORD PTR [BP+30]    ; Load the page number.
            MOV         AX,WORD PTR ES:[BX]     ;
            AND         AX,1                    ;
            MOV         AH,WRPAGE               ;
            CALL        FAR PTR INT10           ; Set the page number.
            LES         BX,DWORD PTR [BP+26]    ; Load X co-ord.
            MOV         AX,WORD PTR ES:[BX]     ;
            LES         BX,DWORD PTR [BP+22]    ; Load Y co-ord.
            MOV         CX,WORD PTR ES:[BX]     ;
            ASSUME      ES:MENU_DATA            ; Load MENU_DATA segment.
            MOV         BX,MENU_DATA            ;
            MOV         ES,BX                   ;
            MOV         WORD PTR ES:XPOS,AX     ; Set the PRINTCH cursor
            MOV         WORD PTR ES:YPOS,CX     ; X,Y pointers.
            ASSUME      ES:DGROUP               ;
            LES         BX,DWORD PTR [BP+18]    ; Load the length of the
            MOV         CX,WORD PTR ES:[BX]     ; string.
            LES         BX,DWORD PTR [BP+14]    ; Load start addr. of string.
            CMP         CX,ZERO                 ; If length <> 0 , then
            JE          NO_CHAR                 ;    While not_end_of_string do
WRIT_CH:    MOV         AL,BYTE PTR ES:[BX]     ;      Load next char.
            INC         BX                      ;      Increment pointer.
            CALL        FAR PTR PRINTCH         ;      Print char.
            LOOP        WRIT_CH                 ;    Endwhile.
                                                ; Endif.
NO_CHAR:                                        ;
            POP         CX                      ; Restore registers.
            POP         BX                      ;
            POP         AX                      ;
            POP         ES                      ;
            POP         BP                      ;
            RET         14H                     ; Exit to FORTRAN.
WRTSTR      ENDP

CODE        ENDS                                ;
            END                                 ;
;-------------------------------------------------------------------------
```

```
; MODULE : Screen routines.
; ************************

TITLE Initialises and resets the screen for graphics.
;     ********************************************

; ******************************************************************************
; REVISION HISTORY :
; VERSION       BY                DATE        COMMENT
;    1.00       Ian Fisher        23/06/88    Creation.
;
; ******************************************************************************

; Routines.
; *********
            PUBLIC      INTSCR      ; To initialise the screen.
            PUBLIC      RSTSCR      ; To reset the screen to text mode.

            EXTRN       BOX:FAR     ; To print out a box.
            EXTRN       INT10:FAR   ; To call the interrupt.
            EXTRN       PRINTCH:FAR ; To print a character to graphics screen.
            EXTRN       PRSTR:FAR   ; To print a string to graphics screen.

; Keyboard routine function numbers.
; **********************************
GET_ASCII   EQU         01H         ; Return ASCII key.
FLUSH_ASCII EQU         02H         ; Flush buffer.
KEY_HIT     EQU         03H         ; Check if key hit.
IF_HIT_GET  EQU         04H         ; If key hit, then get key.

; 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.

; Escape characters.
; ******************
ESCJ        EQU         04A1BH      ; Move cursor to x,y.
ESCK        EQU         04B1BH      ; Delete x number of characters.
ESCL        EQU         04C1BH      ; Set intensity to x.

; Keys from keyboard.
; *******************
ESC_KEY     EQU         1B00H       ; The ESC key.
CR_KEY      EQU         0D00H       ; The ENTER or RETURN key.
REV_TAB     EQU         000FH       ; The REVERSE TAB key
TAB_KEY     EQU         0900H       ; The TAB key.

; Routine constants and flags.
; ****************************
DIR_FWD     EQU         01          ; Highlight next option.
DIR_REV     EQU         02          ; Highlight previous option.
NORMATR     EQU         01          ; Normal intensity.
NO_MENU     EQU         0FFH        ; No error when printing new menu.
PAGE0       EQU         00          ; Select page 0.
PAGE1       EQU         01          ; Select page 1.
XORATR      EQU         02          ; XOR type intensity.
ZERO        EQU         00          ; Just zero.

; Menu and character constants.
; *****************************
MAXLEN      EQU         12          ; Maximum length of option.
SCR_X       EQU         2           ; X co-ord of command headers.
XCH         EQU         09          ; Character width in graphics mode.
XMENLIM     EQU         571         ; X co-ord limit for menu.
XMENST      EQU         92          ; X co-ord for start of menu.
XPOSLIM     EQU         710         ; X limit for printing chars.
YBRIEF      EQU         331         ; Y co-ord for brief.
YCH         EQU         14          ; Character height in graphics mode.
```

```
YMENST      EQU         287         ; Y co-ord for start of menu.
YPOSLIM     EQU         347         ; Y limit for printing chars.

; GRAPHIX routine function numbers.
; *********************************
BLKFIL      EQU         04AH        ; Fill a block.
CLRSCR      EQU         042H        ; Clear the screen.
DISP        EQU         045H        ; Display a page.
DLINE       EQU         049H        ; Draw a line.
GMODE       EQU         040H        ; Define graphics mode.
GPAGE       EQU         043H        ; Page to which data is written.
LEVEL       EQU         044H        ; Intensity level.
MOVE        EQU         048H        ; Move the imaginary cursor.
TEXT        EQU         04BH        ; Output text to the page.
TMODE       EQU         041H        ; To switch to text mode.

; General data segment.
; *********************
DATA        SEGMENT PUBLIC 'DATA'
DATA        ENDS

; Menu data areas.
; ****************
TABLE_DATA  SEGMENT WORD PUBLIC 'FAR_DATA'
TABLE_DATA  ENDS

;*****************************************************************************
MENU_DATA   SEGMENT WORD PUBLIC 'FAR_DATA'
INIT_MSG    DW          ESCJ,SCR_X,YMENST
            DB          'COMMAND : '
            DW          ESCJ,SCR_X,YBRIEF
            DB          'BRIEF   : ',0
MENU_DATA   ENDS

; *****************************************************************************
DGROUP      GROUP DATA
CODE        SEGMENT 'CODE'
            ASSUME CS:CODE
            ASSUME DS:DGROUP
            ASSUME ES:DGROUP

;-----------------------------------------------------------------------------
INTSCR      PROC FAR
; Function: To initialise the screen.
; Inputs:   None.
; Outputs:  Pages 0 and 1 cleared, routine with page 0 displayed and all
;           data being written to page 0.
; Calls:    INT10, PRSTR.
; Destroys: Flags.
;
; Description: This routine sets the Hercules card into graphics mode and
;              clears both pages. The routine  exits  with all data being
;              written and displayed on page 0.
;
;-----------------------------------------------------------------------------
            PUSH        BP                  ; Save the registers.
            PUSH        DI                  ;
            PUSH        ES                  ;
            PUSH        AX                  ;
            PUSH        BX                  ;
            MOV         AH,GMODE            ; Set graphics mode.
            CALL        FAR PTR INT10       ;
            MOV         AH,GPAGE            ; Write to page 1.
            MOV         AL,PAGE1            ;
            CALL        FAR PTR INT10       ;
            MOV         AH,CLRSCR           ; Clear page 1.
            CALL        FAR PTR INT10       ;
            MOV         AH,GPAGE            ; Write to page 0.
            MOV         AL,PAGE0            ;
            CALL        FAR PTR INT10       ;
            MOV         AH,CLRSCR           ; Clear page 0.
            CALL        FAR PTR INT10       ;
```

```
            MOV         AH,DISP                 ; Display page 0.
            MOV         AL,PAGE0                ;
            CALL        FAR PTR INT10           ;
            ASSUME      ES:MENU_DATA            ;
            MOV         AX,MENU_DATA            ;
            MOV         ES,AX                   ;
            MOV         BX,OFFSET MENU_DATA:INIT_MSG;
            CALL        FAR PTR PRSTR           ;
            ASSUME      ES:DGROUP               ;
            POP         BX                      ; Restore register.
            POP         AX                      ;
            POP         ES                      ;
            POP         DI                      ;
            POP         BP                      ;
            RET                                 ; Exit.
INTSCR      ENDP

;-------------------------------------------------------------------------------
RSTSCR      PROC FAR
; Function: To reset the screen to text mode.
; Inputs:   None.
; Outputs:  Modified screen state.
; Calls:    INT10H
; Destroys: Flags.
;
; Description: Clears both pages and then switches to the text mode.
;-------------------------------------------------------------------------------
            PUSH        AX                      ;
            MOV         AH,CLRSCR               ;
            CALL        FAR PTR INT10           ;
            MOV         AH,TMODE                ;
            CALL        FAR PTR INT10           ;
            POP         AX                      ;
            RET                                 ;
RSTSCR      ENDP

CODE        ENDS                                ;
            END                                 ;
;-------------------------------------------------------------------------------
```

```
C     FILE              : NUMIN.FOR
C                         Set of routines to print, input and edit numbers from
C                         the graphics screen. (Hercules card).
C ***************************************************************************
CN    MODULE NAME       : PRTNUM
CA    FUNCTION          : To print a number to a graphics page.
CS    CALL SEQUENCE     : call prtnum(page,x,y,type,flen,format,width,number)
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                         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                         number - (type) The number to be printed.
C
CO    OUTPUT PARAMETERS: None.
CG    GLOBAL VARIABLES  : None.
CM    MODULES CALLED    : WRTSTR (asm), numstr (for)
CE    ERROR CONDITIONS  : The format string must not be greater than 40 chars.
C                         The number input must one of the 4 types.
CC    COMMENTS          : The routine converts the input number to a string,
C                         and then just outputs the string to the appropriate
C                         page.
C ***************************************************************************
      subroutine prtnum(pg,x,y,type,flen,form,width,num)
      integer*2 pg,x,y,type,flen
      character*40 form
      integer*2 width
      real*8 num

      character*40 prstr
      call numstr(type,flen,form,num,width,prstr)
      call WRTSTR(pg,x,y,width,prstr)
      return
      end

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,number)
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                         number - (type) 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),  tri2 (for),  tri4 (for),  trr4 (for),
C                         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,num)
      implicit integer*2 (I,S)

      integer*2 pg,x,y,type,flen
      character*40 form
      integer*2 width
      real*8 num

      character*40 edtstr,edtst2,form2
      real*8 tempr8
      real*4 tempr4
      integer*4 tempi4
      integer*2 tempi2
      integer*2 key,ctrla,f2key,repflg,twid,chkio

      ctrla = 0256
      f2key = 60
      chkio = 0
      escflg = 0

      call numstr(type,flen,form,num,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)
      call STRCPY(40,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) tempi2
            call tri2(num,tempi2)
         elseif (type.eq.2) then
            read(edtstr,form2,iostat=chkio,end=103,err=103) tempi4
            call tri4(num,tempi4)
         elseif (type.eq.3) then
            read(edtstr,form2,iostat=chkio,end=103,err=103) tempr4
            call trr4(num,tempr4)
         elseif (type.eq.4) then
            read(edtstr,form2,iostat=chkio,end=103,err=103) tempr8
            num = tempr8
```

```
         endif
         call prtnum(pg,x,y,type,flen,form,width,num)
      endif
      if (repflg.eq.1) goto 101
      getnum = key
      if (chkio.eq.0) then
         return
      endif
103   call ERTONE()
      goto 102
      return
      end

C ***************************************************************************
C     Dummy routines to overcome problem of pointer type mismatches.

      subroutine tri2(num1,num2)
      integer*2 num1,num2
      num1 = num2
      return
      end
C ***************************************************************************
      subroutine tri4(num1,num2)
      integer*4 num1,num2
      num1 = num2
      return
      end
C ***************************************************************************
      subroutine trr4(num1,num2)
      real*4 num1,num2
      num1 = num2
      return
      end
C ***************************************************************************
CN    MODULE NAME       : NUMSTR
CA    FUNCTION          : To convert a number to a string.
CS    CALL SEQUENCE     : call numstr(type,lform,format,number,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                         number - (type) 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), i2func (for), i4func (for), r4func (for)
C                         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,num,lstr,outstr)
      integer ntype
      integer lform
      character*40 form
      real*8 num
      integer lstr
      character*40 outstr

      character*40 form2,out2
      real*8 temp
      integer*2 chkio

      chkio = 0
```

```
      call STRCPY(40,form2,lform,form)
      if (ntype.eq.1) then
         temp = i2func(num)
         write(out2,form2,iostat=chkio,err=200) int (temp)
      elseif (ntype.eq.2) then
         temp = i4func(num)
         write(out2,form2,iostat=chkio,err=200) int (temp)
      elseif (ntype.eq.3) then
         temp = r4func(num)
         write(out2,form2,iostat=chkio,err=200) temp
      else
         temp = num
         write(out2,form2,iostat=chkio,err=200) temp
      endif
200   if (chkio.ne.0) then
         call ERTONE()
      endif
      call STRCPY(lstr,outstr,40,out2)
      return
      END

C ******************************************************************************
C Routines to convert the input numbers to the required type and to take into
C account pointer mismatches.

      real*8 function i2func(i2)
      integer*2 i2
      i2func = dble (i2)
      return
      end
C ******************************************************************************
      real*8 function i4func(i4)
      integer*4 i4
      i4func = dble (i4)
      return
      end
C ******************************************************************************
      real*8 function r4func(r4)
      real*4 r4
      r4func = dble (r4)
      return
      end

C ******************************************************************************
CH    REVISION HISTORY :
C     VERSION       BY                DATE        COMMENT
C     1.00          Ian Fisher        23/06/88    Creation.
C     FILEND           :
```

Appendix J Characteristic Loci CAD System File Formats

The disk file formats for project and matrix files created by the Characteristic Loci CAD system are listed below.

Note : 1) Each block represents a new line.
(If the variable is an array and the 'variable name' has a letter for the index number, then each element of the array appears on a new line).
2) Every header and underline has a ';' as the first character on the line.
3) Array variables elements are put on a new line until all elements have been recorded.
4) The files are written in ASCII format

J.1 Project Information Disk File Format

Variable format	Variable Name	Description
a		Title
a		Underline
a		Project title header
a		Underline
a	prjnm	Project title
a		Design group header
a		Underline
a	engnms	Design group name
a		Matrix name header
a		Underline
a		System matrix header
a		Underline
a	gname	System matrix name.

Variable format	Variable Name	Description
a		Controller matrix header
a		Underline
a	kname	Controller matrix name
a		System order header
a		Underline
i2	order	Order of system
a		System input name header
a		Underline
a	inpnms(n)	System input names (n = order)
a		System output name header
a		Underline
a	outnms(n)	System output names (n = order)
a		System filename header
a		Underline
a		System matrix filename header
a		Underline
a	gfnam	Sytem matrix filename
a		Controller matrix filename header
a		Underline
a	kfnam	Controller matrix filename

Variable format	Variable Name	Description
a		System pathname header
a		Underline
a		Save pathname header
a		Underline
a	outpth	Save pathname
a		Load pathname header
a		Underline
a	inpath	Load pathname
a		Frequency sweep parameter header
a		Underline
g10.4 g10.4 g10.4	fstart fstop increm	Start frequency End frequency Frequency increment
i8 i8 i8	ftype inctyp incpts	Frequency units Frequency increment mode Frequency increment points
a		Plot page settings header
a		Underline
i8 i8 i8 i8	plset1 pmset1 pmat1 fback1	Loci page : setting 1 Bode page : setting 1 In- or excluding controller : setting 1 No longer in use
i8 i8 i8 i8	plset2 pmset2 pmat2 fback2	Loci page : setting 2 Bode page : setting 2 In- or excluding controller : setting 2 No longer in use

Variable format	Variable Name	Description
a		Loci axes scales header
a		Underline
g10.4	xllim(3,n)	X-axis limits for loci (n = order)
g10.4	yllim(3,n)	Y-axis limits for loci (n = order)
g10.4	xblim(3,n)	Y-axis limits for Bode plots (n = order)
g10.4	yblim(3)	X-axis limits for Bode plots
g10.4	xmlim(3,n)	Y-axis limits for angle plots (n = order)
g10.4	ymlim(3)	X-axis limits for angle plots
a		Underline
a		Time simulation header
a		Underline
g10.4 g10.4	tend dt	Time limit Time step
i2 i2	cloop coninc	System loop status Controller status
g10.4	stpinc(5)	System step input times
g10.4	stpdat(5)	System step sizes
i2	stpinp(5)	System inputs to step
g10.4	ytlim(3,n)	Time simulation y-axis limits (n = order)
a		Underline

J.2 Matrix Information Disk File Format

Variable format	Variable Name	Description
i6	maxord	Maximum order of polynomial
i6	n	Number of rows in the system
i6	n	Number of columns in the system
g10.4	mat(i,j,1,12)	Delay or lag
g10.4	mat(i,j,1,13)	Order of Numerator
g10.4	mat(i,j,1,m)	Numerator coefficients (m = order of numerator)
g10.4	mat(i,j,2,13)	Order of Denominator
g10.4	mat(i,j,2,m)	Denominator coefficients (m = order of denominator)
a		Matrix name header
a		Underline
a	mname	Matrix name

Appendix K Development/Execution Information for the Characteristic Loci CAD (CL-CAD) System

This appendix describes computer hardware and software requirements necessary to execute and/or develop the CL-CAD system.

K.1 Computer Hardware Required by the CL-CAD System

The following computer hardware is required to execute or develop the CL-CAD system :

Personal computer :
- i) PC-XT or PC-AT (IBM compatible or similar machine capable of running DOS version 2.0 or higher)
- ii) 640 Kbytes of RAM
- iii) Hercules Graphics Card and monitor
- iv) Hard storage device (eg. floppy disk)

Optional hardware :
- i) Maths Co-processor (Intel 8087)
- ii) 20 Mbyte hard disk drive
- iii) Dot matrix printer (EPSON FX type)

K.3.3 Linking the Modules

All the object modules (filename with ".obj" extensions) must now be linked which will result in the creation of the executable file (filename with ".exe" extension).

The command line to line all the object modules is as follows :

```
link loci+inisys+mtext+getsys+
        newmat+fmat+poly+names+files+proj+clr1+clr2+
               fodds+edtmat+dodir+
        design+doplot+prinfo+dwbode+dwloci+scales+
               getk+calfre+dog+dograf+track+symult+pimat+
               vector+prmat+prpoly+
        simul+ stepopt+simscl+delay+
        dohelp+help+
        domenu+util+screen+getkey+strin+strops+numin+axes+
        plot+gnums+inmenu+derror+pscr+math
        ,,,/SEGMENTS:255/EXEPACK;
```

The two linker switches : EXEPACK - this switch causes the linker to create the smallest executable file possible.

SEGMENTS:255 - this switch (and parameter) sets the maximum number of data segments that can be linked into the executable file.

The following software is required to execute the CL-CAD system :

i) CAD system software : LOCI.EXE
LOCI.SYS
LOCI.HLP

ii) INT10 memory resident graphics support software for the Hercules graphics card.

To following software is required to develop the CL-CAD system :

Compilers : i) Microsoft FORTRAN (version 3.31 or higher)
ii) Microsoft Macro Assembler (version 4.0 or higher)

Linker : i) Microsoft Overlay Linker (version 3.51 or higher)

K.2 Executing the CL-CAD System

The following steps should be followed in order to execute the CL-CAD system :

step 1 : Insure that the Hercules graphics card is in FULL page mode ; type the following on the command line (at the DOS prompt) :

HGC FULL SAVE <return>

Where "<return>" implies hitting the ENTER key.

step 2 : Insure that the INT10 memory resident graphics support software has been installed ; enter the following command :

INT10 <return>

step 3 : To start the CL-CAD system, enter the following command :

LOCI <return>

The CAD system should now be in operation. Hitting the F1 key enables the on-line help facility.

K.3 Development Information

K.3.1 FORTRAN Modules

All the FORTRAN modules were compiled using the following commands (As an example, to compile the module "example.for") :

```
FOR1 example,,,;    <return>
```

Then, if no compile errors were found

```
PAS2                <return>
```

Then, if no errors were detected, the compiler will have generated the file : "example.obj" (All the ".obj" modules are linked at a later stage, see section K.3.3).

K.3.2 Assembler Modules

All the Assembler modules were assembled using the following commands (For example, to assemble the module "example.asm") :

```
MASM example,,,;    <return>
```

If no errors are detected, the assembler will have generated the file : "example.obj" (All the ".obj" modules are linked at a later stage, see section K.3.3).