

H_{∞} OPTIMAL CONTROL OF A COUNTER-CURRENT PROCESS

This dissertation has been submitted to the Department of Electrical and Electronic Engineering, University of Cape Town, in partial fulfilment of the requirements for an M.Sc. in Electrical Engineering.

by

Brendan Olivier

September 1991

The University of Cape Town has been given the right to reproduce this thesis in whole or in part. Copyright is held by the author.

The copyright of this thesis vests in the author. No quotation from it or information derived from it is to be published without full acknowledgement of the source. The thesis is to be used for private study or non-commercial research purposes only.

Published by the University of Cape Town (UCT) in terms of the non-exclusive license granted to UCT by the author.

ACKNOWLEDGEMENTS

The Council of Mineral Technology, MINTEK, for providing the necessary funding for this project.

Prof. Martin Braae of the Department of Electrical and Electronic Engineering, University of Cape Town, for his supervision and guidance on all aspects for the duration of this project.

Dr. Chris Swartz of the Department of Chemical Engineering, University of Cape Town, for his guidance as a co-supervisor and assistance with the Heat Exchanger simulation.

Dr. Ian Barker of MINTEK, for his advice and interest shown in this project.

Prof. Jannie Van Deventer of the Department of Metallurgical Engineering, University of Stellenbosch, for his assistance with the Carbon-in-Pulp simulation.

SYNOPSIS

The extraction of gold from ore that has been mined is the most important part of the process which eventually produces gold bullion. The process most commonly used today is that of Carbon-in-Pulp gold extraction (CIP). One of the main reasons for this is that it is the most economically efficient method of extracting gold from ore.

The process uses activated carbon to absorb gold from a cyanide leach solution. Slurry containing the gold bearing ore and the activated carbon flow in a counter-current fashion. This counter-current flow enables a high percentage of the gold to be recovered. Gold can then be recovered through an elution process. Large amounts of activated carbon are used in the process and a formal multivariable control study of the adsorption section of the CIP process could provide further economic savings by extracting more gold with controlled amounts of carbon.

A study was performed to identify the chemical mechanisms involved in the adsorption section of a CIP plant. It was felt that the workings of the process could best be established by designing a simple simulator of the process. The simulator was designed with four reactor tanks, in which the carbon absorbs gold from the leached slurry. The simulator uses a continuous transfer of carbon.

The mathematical equations describing the adsorption process were derived using a mass balance across each tank. The mechanism of gold adsorption onto the carbon used in the simulation was a simple two parameter first order adsorption model. A study was then done to examine the change in gold concentration in solution due a change in the amount of carbon in a reactor tank. A steady state analysis was performed before the dynamics of the gold concentration changes were observed. These two studies provided a good understanding of the chemical principles involved in the adsorption process.

Since data from an operational CIP plant was unavailable a control study was performed on a similar process. A counter-current Heat Exchanger Rig (HERIG) was used. A comparison needed to be performed to determine how well the HERIG mimics the CIP process.

The HERIG has two streams of water flowing counter-currently. One stream begins hot and is cooled down, the HOT stream, while the other begins cold and warms up, the COLD stream. The HOT stream flows inside copper piping which is wound helically inside each of four tanks. An exchange of heat takes place from the HOT water through the walls of the copper piping and into the COLD water inside the tanks. The HOT stream represents the concentration of gold in slurry, while the COLD stream represents the concentration of gold on carbon. The adsorption of gold onto the carbon is then mimicked by the transfer of heat from the HOT to the COLD stream.

In order to fully understand the operation of the HERIG, a simulation study was performed. This simulated model was a simplified version of the actual rig. The level changes of COLD water in the tanks were assumed to be instantaneous and the heat transfer coefficients were assumed to be the same for all four of the tanks. The calculation of the heat transfer coefficients was investigated thoroughly and care was taken to obtain accurate values.

The simulator designed was a lumped parameter model. The pipes containing the HOT stream were divided into many small section, in each of which a constant temperature was assumed. A sum of the contributions of all the sections submerged under the COLD water was used to calculate the heat transferred into the COLD water. The COLD water in the tank is stirred continuously and is assumed to be at a constant temperature.

The level of COLD water in each tank on the HERIG represents the mass of carbon in each of the tanks on a CIP plant. A change in the HOT water pipe temperature (concentration of Au in slurry) was examined as a function of a change in the level of COLD water in a tank (mass of carbon in a reactor). A steady state and dynamic analysis verified that trends observed from the CIP model were in fact mimicked by the trends observed on the HERIG. It was then decided to perform a formal control study of the HERIG since the numerous similarities found between the CIP and HERIG enabled relevant conclusions to be drawn about the control of CIP from the control of the HERIG.

In deciding what approach should be taken to design a closed loop controller for the HERIG, a study of the HERIG structure was undertaken using a Binary Interaction Matrix (BIM). This method enabled the decision to be made that the flows could be controlled using a single variable controller for each loop, and the levels could be controller in the same manner but with a precompensator to eliminate the interaction that exists between the loops. Closed loop step tests showed that the controllers designed for both the flows and levels worked properly.

With the levels under control, the temperature response to a changing water level could be examined. An open loop model was found for the fully interactive multivariable matrix. It was noticed that the system required a indepth multivariable study to be performed to bring the temperatures under closed loop control.

The method chosen to design such a controller was the relatively new approach of H_{∞} Optimal Control. It is a frequency based design method. The method involves synthesis as well as design stages. The synthesis problem is easily solved with the correct computing facilities, but the design section is one that required much attention. Before any design or synthesis was performed it was found that the plant model needed to be reduced in order (number of states). This not only saved on computational time but also allows for a controller with less states to be designed.

The synthesis of a stable controller for a stable closed loop system is an easily solved problem. It requires that the traditional closed loop arrangement be changed to fit the form of an open loop plant augmented with appropriate weighting functions. These weighting functions define the desired frequency domain characteristics of the closed loop system. The choice of appropriate weights is the design phase of the problem.

Once an augmented plant has been found, a simple iteration procedure finds the optimal controller for the augmented plant. The controller has zeros which cancel the augmented plants stable poles and the controller has poles that cancel the augmented plants left half plane zeros. The controller also has poles that will provide the necessary closed loop requirements.

- The design phase involves choosing weighting functions for the error, input and output signals that satisfy design requirements. The requirements of setpoint tracking and steady state disturbance rejection are easily obtained using a weight on the error signal. The requirements of robustness and sensor noise attenuation needs to be solved in a higher frequency range than the previously mentioned criteria, since they clash. This can be solved using a weight on the output.

One particular design objective not addressed to date in literature is that of input action limitation. On real plants the amount of input action available is limited. Thus the design need to be adjustable to cater for this. It was found that by attenuating the high frequency response of the input, with a weight on the input function, the input could be made to be a smooth function without any large input spikes.

- The response time of the closed loop system is determined by the cross-over frequency of two of the weighting functions. When the synthesis algorithm is applied it ensures that the closed loop system has a controller that gives the correct response time and a controller containing the minimum number of states. The interaction between the loops was found to be taken care of automatically by the synthesis procedure.

Thus the H_{∞} method provides a design method for easily obtaining a multivariable controller that satisfies all required frequency and time domain specifications. It was then required to be shown that this method could be successfully implemented on the HERIG.

The final controller chosen was included in software already written to control the flowrates and levels on the HERIG. Closed loop step tests were performed to examine the temperature response of the HERIG. The results were compared to simulated data of the expected output and input responses of the closed loop system. An analysis of this showed that excellent correlation between the two sets of results was found. Both the input action and the output behaved in time very much as expected and thus confirmed that H_{∞} designed controller was implemented successfully.

The conclusions drawn concerning the modelling of the CIP process were that a simple first order model provided a good understanding of the chemical processes at work in the adsorption section of a CIP plant. It was also found that the HERIG had very similar characteristics to the CIP process and enough correlation was found between the two to say that a formal control study of the HERIG would give insight into the control of the CIP process.

The conclusions arrived at from the study and implementation of the H_{∞} Optimal Method of designing feedback controller are that it is method with which frequency as well as time domain constraints can be handled in a design stages. A controller was successfully implemented on the HERIG and the results compared very well to the expected results.

Finally it can be said that H_{∞} can be successfully used to design closed loop controllers for counter-current processes. In particular the CIP process will benefit from using this method since it is a large multivariable problem with tight constraints on the amount of input action (mass of carbon) that can be used. The H_{∞} method provides a method of designing with input constraints and generating a controller that automatically takes care of interactions without the need for precompensation.

TABLE OF CONTENTS

SYNOPSIS	i
TABLE OF CONTENTS	viii
LIST OF FIGURES	xii
LIST OF TABLES	xxiv
LIST OF SYMBOLS	xxv
CHAPTER 1 : INTRODUCTION	1
REFERENCES	4
CHAPTER 2 : CARBON-IN-PULP GOLD EXTRACTION	7
2.1 : DESCRIPTION OF THE ADSORPTION SECTION	7
2.1.1 : NUMBER OF STAGES IN THE ADSORPTION SECTION	8
2.1.2 : MIXING OF THE TANKS	9
2.1.3 : INTERSTAGE SCREENING	9
2.1.4 : LEACHING OF THE ORE	9
2.1.5 : TRANSFER OF CARBON	9
2.1.6 : BACKMIXING OF SLURRY	10
2.2 : MODEL FOR ADSORPTION IN THE CIP PROCESS	10
2.2.1 : MASS BALANCE ACROSS A TANK	11
2.2.2 : MODEL FOR THE RATE OF ADSORPTION	11
2.2.3 : CHOOSING INPUTS AND OUTPUTS	12
2.2.4 : CHOOSING OPERATING CONDITIONS	13
2.3 : SIMULATION OF THE ADSORPTION SECTION	14
2.4 : ANALYSIS OF THE MODEL AT STEADY STATE	15
2.5 : ANALYSIS OF MODEL DYNAMICS	17
2.6 : CONCLUSIONS ON THE CIP MODEL	20
2.6.1 : ELEMENTS ON AND BELOW THE DIAGONAL	20
2.6.2 : ELEMENTS ABOVE THE DIAGONAL	21
REFERENCES	21
CHAPTER 3 : COUNTER-CURRENT HEAT EXCHANGER RIG	25
3.1 : DESCRIPTION OF THE HERIG	25
3.2 : COMPARISON BETWEEN CIP AND HERIG	27
3.2.1 : HOT STREAM OF WATER	27
3.2.2 : COLD STREAM OF WATER	27
3.2.3 : LEVEL OF WATER IN A TANK	28
3.2.4 : RATE OF HEAT TRANSFER	28
3.2.5 : TEMPERATURE GRADIENT	29
3.3 : MODEL FOR THE HERIG	29
3.3.1 : HEAT BALANCE ACROSS A TANK	30
3.3.2 : OVERALL HEAT TRANSFER COEFFICIENT TO WATER	32
3.3.2.1 : INNER HEAT TRANSFER COEFFICIENT	32
3.3.2.2 : OUTER HEAT TRANSFER COEFFICIENT	33
3.3.3 : OVERALL HEAT TRANSFER COEFFICIENT TO AIR	34
3.3.3.1 : INNER HEAT TRANSFER COEFFICIENT	34

3.3.3.2 : OUTER HEAT TRANSFER COEFFICIENT	34
3.3.4 : OPERATING CONDITIONS FOR THE HERIG	35
3.4 : SIMULATION OF THE HEAT EXCHANGER RIG	36
3.5 : ANALYSIS OF THE MODEL AT STEADY STATE	37
3.6 : ANALYSIS OF MODEL DYNAMICS	39
3.7 : CONCLUSIONS ON THE HERIG AND CIP SIMILARITIES	42
3.7.1 : CHOICE OF INPUTS AND OUTPUTS	43
3.7.2 : RESPONSE OF THE ON-DIAGONAL ELEMENTS	43
3.7.3 : RESPONSE OF THE OFF-DIAGONAL ELEMENTS	43
REFERENCES	44
CHAPTER 4 : STRUCTURAL ANALYSIS OF THE HERIG	46
4.1 : BINARY INTERACTION MATRIX (BIM) STUDY OF THE HERIG	46
4.1.1 : DESCRIPTION OF THE FULL SCALE BIM MATRIX	48
4.1.2 : REDUCED ORDER BIM MATRIX	48
4.1.3 : FINAL CONTROL STRUCTURE FOR THE HERIG	50
4.1.3.1 : CONTROL OF THE FLOWRATES	50
4.1.3.2 : CONTROL OF THE LEVELS	50
4.1.3.3 : CONTROL OF THE PIPE TEMPERATURES	51
4.1.4 : IMPLEMENTATION OF THE DESIRED CONTROL SCHEME	52
4.2 : CLOSED LOOP FLOW CONTROL	53
4.2.1 : CLOSED LOOP FLOW CONTROLLER	54
4.2.2 : FILTER FOR FLOW CONTROL	54
4.2.3 : CLOSED LOOP CONTROL STRUCTURE	55
4.3 : CLOSED LOOP LEVEL CONTROL	56
4.3.1 : PRE-COMPENSATOR FOR LEVEL CONTROL	59
4.3.2 : CLOSED LOOP LEVEL CONTROLLER	60
4.3.3 : FILTER FOR FLOW CONTROL	60
4.3.4 : CLOSED LOOP CONTROL STRUCTURE	61
4.4 : OPEN LOOP TEMPERATURE RESPONSES	66
4.5 : COMPARISON BETWEEN THE MODEL AND THE HERIG	71
4.5.1 : COMPARISON OF THE STEADY STATE GAINS	71
4.5.2 : COMPARISON OF THE SPEED OF RESPONSE	73
4.5.2.1 : DIAGONAL ELEMENTS	73
4.5.2.2 : LOWER DIAGONAL ELEMENTS	73
4.5.2.3 : UPPER DIAGONAL ELEMENTS	74
REFERENCES	74
CHAPTER 5 : H_{∞} OPTIMAL CONTROLLER DESIGN AND SYNTHESIS	76
5.1 : DEFINITION OF THE H_{∞} OPERATOR NORM	76
5.1.1 : SINGULAR VALUES	76
5.1.2 : PRINCIPLE GAINS	77
5.2 : ALGORITHM FOR THE DESIGN OF FEEDBACK CONTROLLERS	78
5.3 : OBTAINING A PLANT MODEL	82
5.4 : WEIGHTING FUNCTIONS SPECIFICATIONS	83
5.4.1 : DESIGN OF WEIGHT ON $S(s)$	85
5.4.2 : DESIGN OF WEIGHT ON $R(s)$	86
5.4.3 : DESIGN OF WEIGHT ON $T(s)$	87
5.5 : H_{∞} SYNTHESIS	88
5.5.1 : AUGMENT PLANT	88
5.5.2 : SCALE PLANT	91
5.5.3 : SCALE CONTROLS AND MEASUREMENTS	92

5.5.4 : H_{∞} SOLUTION	93
5.5.4.1 : TRANSFER FUNCTION P11 BE STRICTLY PROPER	93
5.5.4.2 : TRANSFER FUNCTION P22 BE STRICTLY PROPER	93
5.5.4.3 : TRANSFER FUNCTION P21 BE PROPER	93
5.5.4.4 : TRANSFER FUNCTION P12 BE PROPER	94
5.5.5 : OBTAIN CONTROLLER	96
5.6 : H_{∞} DESIGN	98
5.6.1 : SETPOINT TRACKING	98
5.6.2 : DISTURBANCE REJECTION	98
5.6.3 : ROBUSTNESS	99
5.6.4 : SPEED OF RESPONSE	99
5.6.5 : INTERACTION	99
5.6.6 : INPUT ACTION	100
REFERENCES	100
CHAPTER 6 : HERIG TEMPERATURE CONTROL USING H_{∞} DESIGN	103
6.1 : SOFTWARE USED TO PERFORM DESIGN	103
6.2 : MODEL REDUCTION	104
6.3 : DESIGN NUMBER 1	108
6.3.1 : SYNTHESIS STAGE 1 ($\Gamma = 1E6$)	109
6.3.2 : SYNTHESIS STAGE 2 ($\Gamma = 16$)	111
6.4 : DESIGN NUMBER 2	114
6.4.1 : SYNTHESIS STAGE 1 ($\Gamma = 1E6$)	115
6.4.2 : SYNTHESIS STAGE 2 ($\Gamma = 0.95$)	117
6.5 : DESIGN NUMBER 3	120
6.5.1 : SYNTHESIS STAGE 1 ($\Gamma = 1E6$)	121
6.5.2 : SYNTHESIS STAGE 2 ($\Gamma = 0.85$)	123
6.6 : DESIGN NUMBER 4	126
6.6.1 : SYNTHESIS STAGE 1 ($\Gamma = 1E6$)	127
6.6.2 : SYNTHESIS STAGE 2 ($\Gamma = 0.9$)	129
6.7 : IMPLEMENTATION OF THE CONTROLLER	132
REFERENCES	138
CHAPTER 7 : COMPARISON OF RESULTS	139
7.1 : COMPARISON OF OUTPUTS	139
7.1.1 : STEP IN SETPOINT 1	140
7.1.2 : STEP IN SETPOINT 2	141
7.1.3 : STEP IN SETPOINT 3	142
7.1.4 : STEP IN SETPOINT 4	143
7.2 : COMPARISON OF INPUTS	144
7.2.1 : STEP IN SETPOINT 1	145
7.2.2 : STEP IN SETPOINT 2	146
7.2.3 : STEP IN SETPOINT 3	147
7.2.4 : STEP IN SETPOINT 4	148
CHAPTER 8 : CONCLUSIONS	150
8.1 : MODELLING OF CIP	150
8.2 : MODELLING OF THE HERIG	151
8.3 : SIMILARITIES BETWEEN CIP AND THE HERIG	151
8.4 : H_{∞} DESIGN METHODOLOGY	152
8.5 : DESIGN OF A CONTROLLER FOR HERIG TEMPERATURE CONTROL	153

8.6 : CONTROL OF THE HEAT EXCHANGER RIG	154
8.7 : CONTROL OF THE CIP PROCESS	155
BIBLIOGRAPHY	156
APPENDIX A : CARBON-IN-PULP SIMULATION SOFTWARE	161
A.1 : MAIN PROGRAMME	161
A.2 : PLANT DESCRIPTION UNIT	166
A.3 : MATRIX HANDLING UNIT	168
A.4 : DATA HANDLING UNIT	170
APPENDIX B : CALCULATION OF CONSTANTS FOR THE HERIG	172
B.1 : PROPERTIES OF AIR AND WATER	172
B.2 : PHYSICAL LAYOUT OF THE HERIG	173
B.3 : OVERALL HEAT TRANSFER COEFFICIENT TO WATER	174
B.4 : OVERALL HEAT TRANSFER COEFFICIENT TO AIR	174
REFERENCES	175
APPENDIX C : HEAT EXCHANGER RIG SIMULATION SOFTWARE	176
C.1 : MAIN PROGRAMME	176
C.2 : PLANT DESCRIPTION UNIT	182
C.3 : MATRIX HANDLING UNIT	187
C.4 : DATA HANDLING UNIT	189
APPENDIX D : HERIG CONTROLLER SOFTWARE	191
D.1 : MAIN PROGRAMME	191
D.2 : PLANT DESCRIPTION UNIT	216
D.3 : MATRIX HANDLING UNIT	217
D.4 : DATA HANDLING UNIT	221
D.5 : INTERFACE UNIT	226
APPENDIX E : SYSTEM IDENTIFICATION SOFTWARE	228
APPENDIX F : OPEN LOOP TEMPERATURE TRANSFER FUNCTIONS	253
APPENDIX G : MATLAB SOFTWARE USED FOR H_{∞} DESIGNS	257
G.1 : H_{∞} DESIGN PROGRAMME	257
G.2 : BUTTERWORTH FILTER DESIGN PROGRAMME	260
G.3 : SAVING MATRIX IN TRANSFER FUNCTION FORM	262
G.4 : LOADING MATRIX IN TRANSFER FUNCTION FORM	266
G.5 : SAVING MATRIX IN STATE SPACE FORM	268
G.6 : LOADING MATRIX IN STATE SPACE FORM	269

LIST OF FIGURES

Figure 2.1	
THE CIP ADSORPTION CASCADE	8
Figure 2.2	
A CIP ADSORPTION REACTOR TANK	10
Figure 2.5.1	
CIP G11(s)	18
Figure 2.5.2	
CIP G21(s)	18
Figure 2.5.3	
CIP G31(s)	18
Figure 2.5.4	
CIP G41(s)	18
Figure 2.5.5	
CIP G12(s)	18
Figure 2.5.6	
CIP G22(s)	18
Figure 2.5.7	
CIP G32(s)	18
Figure 2.5.8	
CIP G42(s)	18
Figure 2.5.9	
CIP G13(s)	19
Figure 2.5.10	
CIP G23(s)	19
Figure 2.5.11	
CIP G33(s)	19
Figure 2.5.12	
CIP G43(s)	19
Figure 2.5.13	
CIP G14(s)	19
Figure 2.5.14	
CIP G24(s)	19
Figure 2.5.15	
CIP G34(s)	19
Figure 2.5.16	
CIP G44(s)	19
 Figure 3.1	
THE HERIG CONFIGURATION	26
Figure 3.2	
A HERIG REACTOR TANK	30
Figure 3.6.1	
HES G11(s)	40
Figure 3.6.2	
HES G21(s)	40
Figure 3.6.3	
HES G31(s)	40
Figure 3.6.4	

HES G41(s)	40
Figure 3.6.5	
HES G12(s)	40
Figure 3.6.6	
HES G22(s)	40
Figure 3.6.7	
HES G32(s)	40
Figure 3.6.8	
HES G42(s)	40
Figure 3.6.9	
HES G13(s)	41
Figure 3.6.10	
HES G23(s)	41
Figure 3.6.11	
HES G33(s)	41
Figure 3.6.12	
HES G43(s)	41
Figure 3.6.13	
HES G14(s)	41
Figure 3.6.14	
HES G24(s)	41
Figure 3.6.15	
HES G34(s)	41
Figure 3.6.16	
HES G44(s)	41
Figure 4.1.1	
FULL ORDER BIM MATRIX	47
Figure 4.1.2	
REDUCED ORDER BIM MATRIX	49
Figure 4.1.3	
OPEN LOOP FLOW CONTROL	50
Figure 4.1.4	
OPEN LOOP LEVEL CONTROL	51
Figure 4.1.5	
OPEN LOOP TEMPERATURE CONTROL	51
Figure 4.2.1	
FLOW G11(s)	53
Figure 4.2.2	
FLOW G21(s)	53
Figure 4.2.3	
FLOW G12(s)	53
Figure 4.2.4	
FLOW G22(s)	53
Figure 4.2.5	
OPEN LOOP FLOW MODEL	53
Figure 4.2.6	
CLOSED LOOP FLOW CONTROLLER	54
Figure 4.2.7	
FILTER FOR FLOW CONTROL	54
Figure 4.2.8	
CLOSED LOOP FLOW CONTROL	55
Figure 4.2.9	

FLOW H11(s)	55
Figure 4.2.10	
FLOW H21(s)	55
Figure 4.2.11	
FLOW H12(s)	55
Figure 4.2.12	
FLOW H22(s)	55
Figure 4.2.13	
FLOW U11(s)	56
Figure 4.2.14	
FLOW U21(s)	56
Figure 4.2.15	
FLOW U12(s)	56
Figure 4.2.16	
FLOW U22(s)	56
Figure 4.3.1	
LEVEL G11(s)	57
Figure 4.3.2	
LEVEL G21(s)	57
Figure 4.3.3	
LEVEL G31(s)	57
Figure 4.3.4	
LEVEL G41(s)	57
Figure 4.3.5	
LEVEL G12(s)	57
Figure 4.3.6	
LEVEL G22(s)	57
Figure 4.3.7	
LEVEL G32(s)	57
Figure 4.3.8	
LEVEL G42(s)	57
Figure 4.3.9	
LEVEL G13(s)	58
Figure 4.3.10	
LEVEL G23(s)	58
Figure 4.3.11	
LEVEL G33(s)	58
Figure 4.3.12	
LEVEL G43(s)	58
Figure 4.3.13	
LEVEL G14(s)	58
Figure 4.3.14	
LEVEL G24(s)	58
Figure 4.3.15	
LEVEL G34(s)	58
Figure 4.3.16	
LEVEL G44(s)	58
Figure 4.3.17	
OPEN LOOP LEVEL MODEL	59
Figure 4.3.18	
LEVEL PRECOMPENSATOR MATRIX	59
Figure 4.3.19	
CLOSED LOOP FLOW CONTROLLER	60
Figure 4.3.20	

FILTER FOR FLOW CONTROL	61
Figure 4.3.21	
CLOSED LOOP FLOW CONTROL	61
Figure 4.3.22	
LEVEL H11(s)	62
Figure 4.3.23	
LEVEL H21(s)	62
Figure 4.3.24	
LEVEL H31(s)	62
Figure 4.3.25	
LEVEL H41(s)	62
Figure 4.3.26	
LEVEL H12(s)	62
Figure 4.3.27	
LEVEL H22(s)	62
Figure 4.3.28	
LEVEL H32(s)	62
Figure 4.3.29	
LEVEL H42(s)	62
Figure 4.3.30	
LEVEL H13(s)	63
Figure 4.3.31	
LEVEL H23(s)	63
Figure 4.3.32	
LEVEL H33(s)	63
Figure 4.3.33	
LEVEL H43(s)	63
Figure 4.3.34	
LEVEL H14(s)	63
Figure 4.3.35	
LEVEL H24(s)	63
Figure 4.3.36	
LEVEL H34(s)	63
Figure 4.3.37	
LEVEL H44(s)	63
Figure 4.3.38	
LEVEL U11(s)	64
Figure 4.3.39	
LEVEL U21(s)	64
Figure 4.3.40	
LEVEL U31(s)	64
Figure 4.3.41	
LEVEL U41(s)	64
Figure 4.3.42	
LEVEL U12(s)	64
Figure 4.3.43	
LEVEL U22(s)	64
Figure 4.3.44	
LEVEL U32(s)	64
Figure 4.3.45	
LEVEL U42(s)	64
Figure 4.3.46	
LEVEL U13(s)	65
Figure 4.3.47	

LEVEL U23(s)	65
Figure 4.3.48	
LEVEL U33(s)	65
Figure 4.3.49	
LEVEL U43(s)	65
Figure 4.3.50	
LEVEL U14(s)	65
Figure 4.3.51	
LEVEL U24(s)	65
Figure 4.3.52	
LEVEL U34(s)	65
Figure 4.3.53	
LEVEL U44(s)	65
Figure 4.4.1	
TEMP G11(s)	67
Figure 4.4.2	
TEMP G21(s)	67
Figure 4.4.3	
TEMP G31(s)	67
Figure 4.4.4	
TEMP G41(s)	67
Figure 4.4.5	
TEMP G12(s)	67
Figure 4.4.6	
TEMP G22(s)	67
Figure 4.4.7	
TEMP G32(s)	67
Figure 4.4.8	
TEMP G42(s)	67
Figure 4.4.9	
TEMP G13(s)	68
Figure 4.4.10	
TEMP G23(s)	68
Figure 4.4.11	
TEMP G33(s)	68
Figure 4.4.12	
TEMP G43(s)	68
Figure 4.4.13	
TEMP G14(s)	68
Figure 4.4.14	
TEMP G24(s)	68
Figure 4.4.15	
TEMP G34(s)	68
Figure 4.4.16	
TEMP G44(s)	68
Figure 4.5.1	
GAINS FOR COLUMN 1	71
Figure 4.5.2	
GAINS FOR COLUMN 2	71
Figure 4.5.3	
GAINS FOR COLUMN 3	72
Figure 4.5.4	
GAINS FOR COLUMN 4	72

Figure 5.1	
FLOWCHART FOR H_{∞} BASED CONTROLLER DESIGN	79
Figure 5.2	
NOMINAL PLANT WITH CONTROLLER IN FEEDBACK	80
Figure 5.3	
CLOSED LOOP CONFIGURATION WITH WEIGHTS	83
Figure 5.4	
WEIGHTING FUNCTION w_1^{-1}	85
Figure 5.5	
WEIGHTING FUNCTION w_2^{-1}	86
Figure 5.6	
WEIGHTING FUNCTION w_3^{-1}	87
Figure 5.7	
AUGMENTED PLANT	89
Figure 5.8	
SCALE THE PLANT	91
Figure 5.9	
SCALE CONTROLS AND MEASUREMENTS	92
Figure 5.10	
H_{∞} CONTROLLER STRUCTURE	94
Figure 5.11	
SCALING THE CONTROLLER	96
Figure 6.2.1	
COMPARATIVE FREQUENCY RESPONSE	104
Figure 6.2.2	
COMPARATIVE STEP RESPONSE (STEP INPUT 1)	105
Figure 6.2.3	
COMPARATIVE STEP RESPONSE (STEP INPUT 2)	105
Figure 6.2.4	
COMPARATIVE STEP RESPONSE (STEP INPUT 3)	105
Figure 6.2.5	
COMPARATIVE STEP RESPONSE (STEP INPUT 4)	105
Figure 6.3.1	
WEIGHTING FUNCTIONS (DESIGN 1.1)	109
Figure 6.3.2	
PERFORMANCE FUNCTIONS (DESIGN 1.1)	109
Figure 6.3.3	
OUTPUT (Step Setpoint 1)	110
Figure 6.3.4	
OUTPUT (Step Setpoint 2)	110
Figure 6.3.5	
OUTPUT (Step Setpoint 3)	110
Figure 6.3.6	
OUTPUT (Step Setpoint 4)	110
Figure 6.3.7	
INPUT (Step Setpoint 1)	110
Figure 6.3.8	
INPUT (Step Setpoint 2)	110
Figure 6.3.9	
INPUT (Step Setpoint 3)	110
Figure 6.3.10	

INPUT (Step Setpoint 4)	110
Figure 6.3.11	
WEIGHTING FUNCTIONS (DESIGN 1.2)	111
Figure 6.3.12	
PERFORMANCE FUNCTIONS (DESIGN 1.2)	111
Figure 6.3.13	
OUTPUT (Step Setpoint 1)	112
Figure 6.3.14	
OUTPUT (Step Setpoint 2)	112
Figure 6.3.15	
OUTPUT (Step Setpoint 3)	112
Figure 6.3.16	
OUTPUT (Step Setpoint 4)	112
Figure 6.3.17	
INPUT (Step Setpoint 1)	112
Figure 6.3.18	
INPUT (Step Setpoint 2)	112
Figure 6.3.19	
INPUT (Step Setpoint 3)	112
Figure 6.3.20	
INPUT (Step Setpoint 4)	112
Figure 6.4.1	
WEIGHTING FUNCTIONS (DESIGN 2.1)	115
Figure 6.4.2	
PERFORMANCE FUNCTIONS (DESIGN 2.1)	115
Figure 6.4.3	
OUTPUT (Step Setpoint 1)	116
Figure 6.4.4	
OUTPUT (Step Setpoint 2)	116
Figure 6.4.5	
OUTPUT (Step Setpoint 3)	116
Figure 6.4.6	
OUTPUT (Step Setpoint 4)	116
Figure 6.4.7	
INPUT (Step Setpoint 1)	116
Figure 6.4.8	
INPUT (Step Setpoint 2)	116
Figure 6.4.9	
INPUT (Step Setpoint 3)	116
Figure 6.4.10	
INPUT (Step Setpoint 4)	116
Figure 6.4.11	
WEIGHTING FUNCTIONS (DESIGN 2.2)	117
Figure 6.4.12	
PERFORMANCE FUNCTIONS (DESIGN 2.2)	117
Figure 6.4.13	
OUTPUT (Step Setpoint 1)	118
Figure 6.4.14	
OUTPUT (Step Setpoint 2)	118
Figure 6.4.15	
OUTPUT (Step Setpoint 3)	118
Figure 6.4.16	
OUTPUT (Step Setpoint 4)	118
Figure 6.4.17	

INPUT (Step Setpoint 1)	118
Figure 6.4.18	
INPUT (Step Setpoint 2)	118
Figure 6.4.19	
INPUT (Step Setpoint 3)	118
Figure 6.4.20	
INPUT (Step Setpoint 4)	118
Figure 6.5.1	
WEIGHTING FUNCTIONS (DESIGN 3.1)	121
Figure 6.5.2	
PERFORMANCE FUNCTIONS (DESIGN 3.1)	121
Figure 6.5.3	
OUTPUT (Step Setpoint 1)	122
Figure 6.5.4	
OUTPUT (Step Setpoint 2)	122
Figure 6.5.5	
OUTPUT (Step Setpoint 3)	122
Figure 6.5.6	
OUTPUT (Step Setpoint 4)	122
Figure 6.5.7	
INPUT (Step Setpoint 1)	122
Figure 6.5.8	
INPUT (Step Setpoint 2)	122
Figure 6.5.9	
INPUT (Step Setpoint 3)	122
Figure 6.5.10	
INPUT (Step Setpoint 4)	122
Figure 6.5.11	
WEIGHTING FUNCTIONS (DESIGN 3.2)	123
Figure 6.5.12	
PERFORMANCE FUNCTIONS (DESIGN 3.2)	123
Figure 6.5.13	
OUTPUT (Step Setpoint 1)	124
Figure 6.5.14	
OUTPUT (Step Setpoint 2)	124
Figure 6.5.15	
OUTPUT (Step Setpoint 3)	124
Figure 6.5.16	
OUTPUT (Step Setpoint 4)	124
Figure 6.5.17	
INPUT (Step Setpoint 1)	124
Figure 6.5.18	
INPUT (Step Setpoint 2)	124
Figure 6.5.19	
INPUT (Step Setpoint 3)	124
Figure 6.5.20	
INPUT (Step Setpoint 4)	124
Figure 6.6.1	
WEIGHTING FUNCTIONS (DESIGN 4.1)	127
Figure 6.6.2	
PERFORMANCE FUNCTIONS (DESIGN 4.1)	127
Figure 6.6.3	
OUTPUT (Step Setpoint 1)	128
Figure 6.6.4	

OUTPUT (Step Setpoint 2)	128
Figure 6.6.5	
OUTPUT (Step Setpoint 3)	128
Figure 6.6.6	
OUTPUT (Step Setpoint 4)	128
Figure 6.6.7	
INPUT (Step Setpoint 1)	128
Figure 6.6.8	
INPUT (Step Setpoint 2)	128
Figure 6.6.9	
INPUT (Step Setpoint 3)	128
Figure 6.6.10	
INPUT (Step Setpoint 4)	128
Figure 6.6.11	
WEIGHTING FUNCTIONS (DESIGN 4.2)	129
Figure 6.6.12	
PERFORMANCE FUNCTIONS (DESIGN 4.2)	129
Figure 6.6.13	
OUTPUT (Step Setpoint 1)	130
Figure 6.6.14	
OUTPUT (Step Setpoint 2)	130
Figure 6.6.15	
OUTPUT (Step Setpoint 3)	130
Figure 6.6.16	
OUTPUT (Step Setpoint 4)	130
Figure 6.6.17	
INPUT (Step Setpoint 1)	130
Figure 6.6.18	
INPUT (Step Setpoint 2)	130
Figure 6.6.19	
INPUT (Step Setpoint 3)	130
Figure 6.6.20	
INPUT (Step Setpoint 4)	130
Figure 6.7.1	
TEMPERATURE H11(S)	134
Figure 6.7.2	
TEMPERATURE H21(S)	134
Figure 6.7.3	
TEMPERATURE H31(S)	134
Figure 6.7.4	
TEMPERATURE H41(S)	134
Figure 6.7.5	
TEMPERATURE H12(S)	134
Figure 6.7.6	
TEMPERATURE H22(S)	134
Figure 6.7.7	
TEMPERATURE H32(S)	134
Figure 6.7.8	
TEMPERATURE H42(S)	134
Figure 6.7.9	
TEMPERATURE H13(S)	135
Figure 6.7.10	
TEMPERATURE H23(S)	135
Figure 6.7.11	

TEMPERATURE H33(S)	135
Figure 6.7.12	
TEMPERATURE H43(S)	135
Figure 6.7.13	
TEMPERATURE H14(S)	135
Figure 6.7.14	
TEMPERATURE H24(S)	135
Figure 6.7.15	
TEMPERATURE H34(S)	135
Figure 6.7.16	
TEMPERATURE H44(S)	135
Figure 6.7.17	
TEMPERATURE U11(S)	136
Figure 6.7.18	
TEMPERATURE U21(S)	136
Figure 6.7.19	
TEMPERATURE U31(S)	136
Figure 6.7.20	
TEMPERATURE U41(S)	136
Figure 6.7.21	
TEMPERATURE U12(S)	136
Figure 6.7.22	
TEMPERATURE U22(S)	136
Figure 6.7.23	
TEMPERATURE U32(S)	136
Figure 6.7.24	
TEMPERATURE U42(S)	136
Figure 6.7.25	
TEMPERATURE U13(S)	137
Figure 6.7.26	
TEMPERATURE U23(S)	137
Figure 6.7.27	
TEMPERATURE U33(S)	137
Figure 6.7.28	
TEMPERATURE U43(S)	137
Figure 6.7.29	
TEMPERATURE U14(S)	137
Figure 6.7.30	
TEMPERATURE U24(S)	137
Figure 6.7.31	
TEMPERATURE U34(S)	137
Figure 6.7.32	
TEMPERATURE U44(S)	137
Figure 7.1.1	
OBSERVED AND SIMULATED H11(s)	140
Figure 7.1.2	
OBSERVED AND SIMULATED H21(s)	140
Figure 7.1.3	
OBSERVED AND SIMULATED H31(s)	140
Figure 7.1.4	
OBSERVED AND SIMULATED H41(s)	140
Figure 7.1.5	

OBSERVED AND SIMULATED H12(s)	141
Figure 7.1.6	
OBSERVED AND SIMULATED H22(s)	141
Figure 7.1.7	
OBSERVED AND SIMULATED H32(s)	141
Figure 7.1.8	
OBSERVED AND SIMULATED H42(s)	141
Figure 7.1.9	
OBSERVED AND SIMULATED H13(s)	142
Figure 7.1.10	
OBSERVED AND SIMULATED H23(s)	142
Figure 7.1.11	
OBSERVED AND SIMULATED H33(s)	142
Figure 7.1.12	
OBSERVED AND SIMULATED H43(s)	142
Figure 7.1.13	
OBSERVED AND SIMULATED H14(s)	143
Figure 7.1.14	
OBSERVED AND SIMULATED H24(s)	143
Figure 7.1.15	
OBSERVED AND SIMULATED H34(s)	143
Figure 7.1.16	
OBSERVED AND SIMULATED H44(s)	143
Figure 7.2.1	
OBSERVED AND SIMULATED U11(s)	145
Figure 7.2.2	
OBSERVED AND SIMULATED U21(s)	145
Figure 7.2.3	
OBSERVED AND SIMULATED U31(s)	145
Figure 7.2.4	
OBSERVED AND SIMULATED U41(s)	145
Figure 7.2.5	
OBSERVED AND SIMULATED U12(s)	146
Figure 7.2.6	
OBSERVED AND SIMULATED U22(s)	146
Figure 7.2.7	
OBSERVED AND SIMULATED U32(s)	146
Figure 7.2.8	
OBSERVED AND SIMULATED U42(s)	146
Figure 7.2.9	
OBSERVED AND SIMULATED U13(s)	147
Figure 7.2.10	
OBSERVED AND SIMULATED U23(s)	147
Figure 7.2.11	
OBSERVED AND SIMULATED U33(s)	147
Figure 7.2.12	
OBSERVED AND SIMULATED U43(s)	147
Figure 7.2.13	
OBSERVED AND SIMULATED U14(s)	148
Figure 7.2.14	
OBSERVED AND SIMULATED U24(s)	148
Figure 7.2.15	
OBSERVED AND SIMULATED U34(s)	148
Figure 7.2.16	

OBSERVED AND SIMULATED U44(s)	148
Figure F.1 TEMPERATURE RESPONSE G11(s)	253
Figure F.2 TEMPERATURE RESPONSE G21(s)	253
Figure F.3 TEMPERATURE RESPONSE G31(s)	253
Figure F.4 TEMPERATURE RESPONSE G41(s)	253
Figure F.5 TEMPERATURE RESPONSE G12(s)	254
Figure F.6 TEMPERATURE RESPONSE G22(s)	254
Figure F.7 TEMPERATURE RESPONSE G32(s)	254
Figure F.8 TEMPERATURE RESPONSE G42(s)	254
Figure F.9 TEMPERATURE RESPONSE G13(s)	255
Figure F.10 TEMPERATURE RESPONSE G23(s)	255
Figure F.11 TEMPERATURE RESPONSE G33(s)	255
Figure F.12 TEMPERATURE RESPONSE G43(s)	255
Figure F.13 TEMPERATURE RESPONSE G14(s)	256
Figure F.14 TEMPERATURE RESPONSE G24(s)	256
Figure F.15 TEMPERATURE RESPONSE G34(s)	256
Figure F.16 TEMPERATURE RESPONSE G44(s)	256

LIST OF TABLES

Table 2.1		
CIP STEADY STATE VALUES (NO INPUT STEPPED)		16
Table 2.2		
CIP STEADY STATE VALUES (STEP INPUT 1)		16
Table 2.3		
CIP STEADY STATE VALUES (STEP INPUT 2)		16
Table 2.4		
CIP STEADY STATE VALUES (STEP INPUT 3)		16
Table 2.5		
CIP STEADY STATE VALUES (STEP INPUT 4)		17
Table 3.1		
HES STEADY STATE VALUES (NO INPUT STEPPED)		37
Table 3.2		
HES STEADY STATE VALUES (STEP INPUT 1)		38
Table 3.3		
HES STEADY STATE VALUES (STEP INPUT 2)		38
Table 3.4		
HES STEADY STATE VALUES (STEP INPUT 3)		38
Table 3.5		
HES STEADY STATE VALUES (STEP INPUT 4)		38
Table 6.1		
SPECIFICATIONS FOR WEIGHTS IN DESIGN 1		108
Table 6.2		
SPECIFICATIONS FOR WEIGHTS IN DESIGN 2		114
Table 6.3		
SPECIFICATIONS FOR WEIGHTS IN DESIGN 3		120
Table 6.4		
SPECIFICATIONS FOR WEIGHTS IN DESIGN 4		126

LIST OF SYMBOLS

A_i	: Surface area of contact for pipe section	$[m^2]$
C_W	: Heat capacity of water	$[J/kg.K]$
C_A	: Heat capacity of air	$[J/kg.K]$
C_C^*	: Model parameter for adsorption model	$[g\text{ Au/Tonne carbon}]$
C_{Ci}	: Concentration of Au on carbon in tank i	$[g\text{ Au/Tonne carbon}]$
C_{Si}	: Concentration of Au in solution in tank i	$[g\text{ Au/Tonne slurry}]$
C_1	: Control valve setting for flowrate F1	$[\%]$
C_2	: Control valve setting for level L4	$[\%]$
C_3	: Control valve setting for level L3	$[\%]$
C_4	: Control valve setting for level L2	$[\%]$
C_5	: Control valve setting for level L1	$[\%]$
C_6	: Control valve setting for flowrate F2	$[\%]$
D_O	: Outer diameter of copper pipe	$[m]$
D_i	: Inner diameter of copper pipe	$[m]$
D_T	: Diameter of COLD water tank	$[m]$
F_P	: Flowrate of HOT water in pipe	$[m^3/s]$
F_T	: Flowrate of COLD water in tanks	$[m^3/s]$
F_C	: Flowrate of carbon	$[Tonne\text{ carbon/hr}]$
F_S	: Flowrate of slurry	$[Tonne\text{ slurry/hr}]$
F_1	: Flowrate of COLD water on HERIG	$[\%]$
F_2	: Flowrate of HOT water on HERIG	$[\%]$
$F_L(s)$	: Closed loop HERIG level filter	

$F_F(s)$	: Closed loop HERIG flowrate filter	[% / %]
$F_T(s)$	: Closed loop HERIG temperature filter	[°C / °C]
$G_L(s)$	: Open loop HERIG level response model	[% / %]
$G_F(s)$	: Open loop HERIG flowrate response model	[% / %]
$G_T(s)$	: Open loop HERIG temperature response model	[°C / %]
Gr_A	: Grahof number for air	[—]
g	: Gravitational acceleration	[m/s ²]
H_{OW}	: Outer heat transfer coefficint to water	[J/s.m ² .K]
H_{iW}	: Inner heat transfer coefficient to water	[J/s.m ² .K]
H_{OA}	: Outer heat transfer coefficint to air	[J/s.m ² .K]
H_{iA}	: Inner heat transfer coefficient to air	[J/s.m ² .K]
$H_L(s)$	: Closed loop HERIG level response model	[% / %]
$H_F(s)$	: Closed loop HERIG flowrate response model	[% / %]
$H_T(s)$	: Closed loop HERIG temperature response model	[°C / °C]
k_W	: Thermal conductivity of water	[J/s.m ² .K]
k_A	: Thermal conductivity of air	[J/s.m ² .K]
k_1	: Rate constant for adsorption model	[Tonne slurry/g Au*1/hr]
k_2	: Rate constant for adsorption model	[1/hr]
$K_C(s)$	: Closed loop HERIG level pre-compensator	[% / %]
$K_L(s)$	: Closed loop HERIG level controller	[% / %]
$K_F(s)$	: Closed loop HERIG flowrate controller	[% / %]
$K_T(s)$	: Closed loop HERIG temperature controller	[% / °C]
L_i	: Level of COLD water in a tank	

L1	: Level of COLD water in HERIG tank number 1	[%]
L2	: Level of COLD water in HERIG tank number 2	[%]
L3	: Level of COLD water in HERIG tank number 3	[%]
L4	: Level of COLD water in HERIG tank number 4	[%]
L _S	: Length of stirrers impeller blade	[m]
M _C	: Mass of carbon	[Tonne]
M _S	: Mass of slurry	[Tonne]
N _S	: Speed of revolution of stirrer	[1/s]
Nu _W	: Nusseldt number for water	[—]
Nu _A	: Nusseldt number for air	[—]
P _W	: Density of water	[kg/m ³]
P _A	: Density of air	[kg/m ³]
Pr _W	: Prandtl number for water	[—]
Pr _A	: Prandtl number for air	[—]
R _i	: Rate of Au adsorption onto carbon	[g Au/Tonne carbon*1/hr]
Re _W	: Reynolds number for water	[—]
S1	: Stirring rate of stirrer for HERIG tank number 1	[%]
S2	: Stirring rate of stirrer for HERIG tank number 2	[%]
S3	: Stirring rate of stirrer for HERIG tank number 3	[%]
S4	: Stirring rate of stirrer for HERIG tank number 4	[%]
T _{PN}	: Temperature of HOT water in pipe section n	

T_{Ti}	: Temperature of COLD water in tank number i	[K]
T_{RM}	: Average room temperature	[K]
T_{WL}	: Average wall temperature	[K]
T_1	: Temperature of HERIG COLD reservoir	[K]
T_2	: Temperature of HERIG COLD water in tank number 4	[°C]
T_3	: Temperature of HERIG COLD water in tank number 3	[°C]
T_4	: Temperature of HERIG COLD water in tank number 2	[°C]
T_5	: Temperature of HERIG COLD water in tank number 1	[°C]
T_6	: Temperature of HERIG HOT reservoir	[°C]
T_7	: Temperature of HERIG HOT water leaving tank 1	[°C]
T_8	: Temperature of HERIG HOT water leaving tank 2	[°C]
T_9	: Temperature of HERIG HOT water leaving tank 3	[°C]
T_0	: Temperature of HERIG HOT water leaving tank 4	[°C]
u_W	: Viscosity of water	[kg/m.s]
u_A	: Viscosity of air	[kg/m.s]
U_A	: Overall heat transfer coefficient to air	[J/s.m ² .K]
U_W	: Overall heat transfer coefficient to water	[J/s.m ² .K]
V_{PN}	: Volume of water in pipe section n	[m ³]
V_{Ti}	: Volume of water in tank number i	[m ³]

CHAPTER 1 : INTRODUCTION

Gold mining, South Africa's most important industry, started on the Witwatersrand in 1886¹. By 1890 the upper levels of ore were exhausted and severe difficulties were encountered with extracting gold from the deeper mined zones.

It was around this time that John MacArthur and the brothers Robert and William Forrest discovered the Cyanidation Process². In this process, gold-bearing ore is dissolved in a cyanide solution. The gold can then be recovered by filtration and zinc precipitation.

By 1894 it had been established that activated carbon could precipitate (absorb) gold from a cyanide solution³. Before carbon could be used in the extraction of gold, a process for eluting (stripping) the gold from the carbon needed to be developed. Research performed in the United States of America (USA) led to the Zadra elution process⁴. In this process gold is eluted by circulating a hot caustic cyanide solution through an elution column. The first major operation to employ Carbon-in-Pulp (CIP) recovery of gold was the Carlton Mill at Cripple Creek, Colorado, in 1951⁵ and not in 1973, as thought by many, at the Homestead Mine in South Dakota⁶.

This pioneering work done in the USA led to the Anglo American Research Laboratory (AARL) developing the AARL elution process and AARL electrowinning cells⁷. The AARL procedure pre-treats the carbon with a hot caustic cyanide solution prior to elution with hot deionised water in a single pass through an elution column. This equipment was used on the first local full scale CIP plant at the President Brand Gold Mine⁸.

Since 1980 the CIP process has been used in all new plants that have been commissioned⁹. The reasons for this are lower capital expenditure, lower operating costs and greater gold recovery when compared to zinc precipitation. The CIP process is now the preferred route of gold extraction and consequently has a great deal of research and development devoted to it.

Until recently a process control study of CIP has been hampered by the lack of instrumentation. MINTEK have now developed an on-line gold analyser for monitoring the amount of gold in solution¹⁰ and are developing an instrument to measure the concentration of carbon granules in an adsorption vessel. These measuring instruments open the way for a formal control study of the CIP process to be performed.

Step test data from a fully operational CIP plant is not available to the University and consequently a study needs to be performed on a process with similar properties to that of CIP. A counter-current heat exchanger rig (HERIG) has been built by Anton De Waal and is running successfully at the University of Cape Town (UCT)¹¹. Due to its counter-current nature, it is seen as a process that mimics CIP. It is hoped that by studying the control of the HERIG, insight may be gained into the control of an actual CIP plant.

A widely used method of controller design is the Multivariable Inverse Nyquist Array (INA) technique developed by Rosenbrock¹². It has been successfully used to design controllers for processes in commercial operation¹³. The INA method attempts to make an open loop system diagonally dominant, but cannot guarantee that this is possible¹⁴. Once diagonal dominance is achieved separate single-input single-output (SISO) controllers can be designed, one for each loop in the multi-input multi-output (MIMO) system.

Achieving diagonal dominance for MIMO systems of order greater than 2 becomes difficult, often at the expense of speed of response¹⁵. The INA method does show closed loop stability and interaction¹⁶, but does not take noise, control action and disturbance rejection specifications into account in the design.

A relatively new approach to feedback design is that of H_∞ Optimal Control¹⁷. The design method requires performance objectives on robustness, sensitivity, noise and control action to be specified in the frequency domain. An algorithm for solving the H_∞ problem, given the frequency domain specifications, is well documented¹⁸. If a controller can be found to fit the design specifications, the closed loop system is guaranteed to be stable. This method has successfully been used to design controllers for computer simulations^{19,20}, but to date there has been no reference to the use of H_∞ on an actual process.

It would help with the further optimising of the CIP gold extraction process if relevant conclusions could be drawn as to the effectiveness of a closed loop controller on the CIP process. Thus it is hoped that by studying the control of the HERIG, insight may be gained into the control of the CIP process.

The objectives of this thesis are thus :

1. To study the CIP process and to model the adsorption section of a multistage CIP plant.
2. To investigate the similarities and differences between the CIP process and the HERIG and to obtain a simple model of the HERIG based on thermal balances.
3. To obtain a model of the HERIG from step tests performed on it and compare this model to the theoretical one obtained.

4. To study the H_{∞} Optimal Control technique and hence evaluate this method as a means of solving the counter-current control problem.
5. To design and implement a multivariable controller for the HERIG using the H_{∞} design method, based on the HERIG step test data.
6. To draw conclusions on the implementation of an H_{∞} designed controller on the HERIG and to its application to the CIP process.

REFERENCES

1. JAMES H.E., DUNNE R.C., DE WAAL S.A. and LAXEN P.A., Established Procedures and New Advances in the Extraction of Gold, Uranium and Pyrite from Witwatersrand Ores, *Proceedings 12TH CMMI Congress*, Johannesburg, S. Afr. Inst. Min. Metall., pp.515-525, 1982.
2. POHLANDT-WATSON C., Historical, Analytical and Environmental Aspects of the Cyanide Process, *Cyanide in Gold Processing : The Next 100 Years*, Johannesburg, The South African Chemical Institute, pp.1-10, 1990.
3. ANONYMOUS, The Development of an Integrated Carbon-in-Pulp process for South African Conditions, Application Report No. 3, MINTEK, pp.1-12, 1987.
4. ADAMS M.D. and NICOL M.J., The Kinetics of the Elution of Gold from Activated Carbon, *GOLD 100 : Proceedings of the International Conference on Gold. Vol 2 : Extractive Metallurgy of Gold*, Johannesburg, SAIMM, pp.111-121, 1986.
5. FAST J.L., Carbon-in-Pulp Pioneering at the Carlton Mill, *E&MJ*, pp.56-57, June 1990.
6. HALL K.B., Homestake uses carbon-in-pulp process to recover gold from slimes, *World Mining*, vol.27 no. 12, pp.44-49, 1974.

7. LAXEN P.A., FLEMING C.A., HOLTRUM D.A. and RUBIN R. A Review of Pilot-Plant Testwork Conducted on the Carbon-in-Pulp Process for the Recovery of Gold, *Proceedings 12TH CMMI Congress*, Johannesburg, S. Afr. Inst. Min. Metall., pp.551-561, 1982.
8. YOUNG G.J.C., DOUGLAS W.D. and HAMPSHIRE M.J., Carbon in Pulp Process for Recovering Gold from Acid Plant Calcines at President Brand, *Mining Engineering*, pp.257-264, March 1984.
9. STANLEY G.G., The Extractive Metallurgy of Gold in South Africa : Volume 1, Johannesburg : Kelvin House, 1987.
10. ANONYMOUS, The Development of an Integrated Carbon-in-Pulp process for South African Conditions, Application Report No. 3, MINTEK, pp.1-12, 1987.
11. DE WAAL A., Performance Index Analysis of Control Systems Synthesised for a Counter-Current Process, M.Sc. Dissertation, Department of Electrical and Electronic Engineering, University of Cape Town, April 1990.
12. ROSENBROCK H.H., State-Space and Multivariable Theory, London : Nelson, 1970.
13. BRAAE M., Department of Electrical and Electronic Engineering, University of Cape Town, Private Communication.
14. MACIEJOWSKI J.M., Multivariable Feedback Design, Reading MA : Addison-Wesley, pp.177-188, 1989.
15. HENNING R., MINTEK, Private Communication.
16. MACIEJOWSKI J.M., Multivariable Feedback Design, Reading MA : Addison-Wesley, pp.168-177, 1989.
17. MACIEJOWSKI J.M., Multivariable Feedback Design, Reading MA : Addison-Wesley, pp.265-324, 1989.
18. DOYLE J.C., GLOVER K., KHARGONEKAR P.P. and FRANCIS B.A., State-Space Solutions to Standard H₂ and H_∞ Control Problems, *IEEE Trans. Automat. Contr.*, Vol. 34 no. 8, pp.831-847, 1989.

19. CRAIG I.K., Sensitivity of H_∞ Controller Designs to Structured Uncertainty, M.Sc. Dissertation, Department of Aeronautics and Astronautics, Massachusetts Institute of Technology, May 1989.
20. CHIANG R.Y. and SAFONOV M.G., Robust-Control Toolbox : for use with MATLAB and the Control Systems Toolbox, User's Guide, The Mathworks, Jun 1988.

CHAPTER 2 : CARBON-IN-PULP GOLD EXTRACTION

Gold bearing ore that has been mined is first milled, then separated by cyclones and further screened to separate out the fine ore. It then passes into the leaching section. Here, during the Cyanidation process, the gold bearing ore is forced into solution¹.

The leached slurry then passes into the CIP absorption section, where activated carbon is used to absorb gold from solution. From here gold bearing carbon is transported to an elution column, where the gold is stripped from the carbon and finally recovered by electrowinning². The adsorption section is the most important of them all³ and it is this section that is studied in this chapter.

2.1 : DESCRIPTION OF THE ADSORPTION SECTION

It is the adsorption section that gives CIP it's counter-current nature. The gold bearing slurry and the activated carbon, which precipitates the gold from solution, flow in opposite directions⁴.

A series of stages, tanks in which the slurry and carbon are mixed, are cascaded together to form the adsorption section. The slurry is pumped into the first stage and then flows downstream under the force of gravity. The carbon is added to the final stage and is transferred upstream, the amount of gold on the carbon increasing as it moves further upstream. Thus in the first tank concentrations, of gold in solution and gold on carbon, are at their highest. Similarly, in the last tank the concentrations are at their lowest.

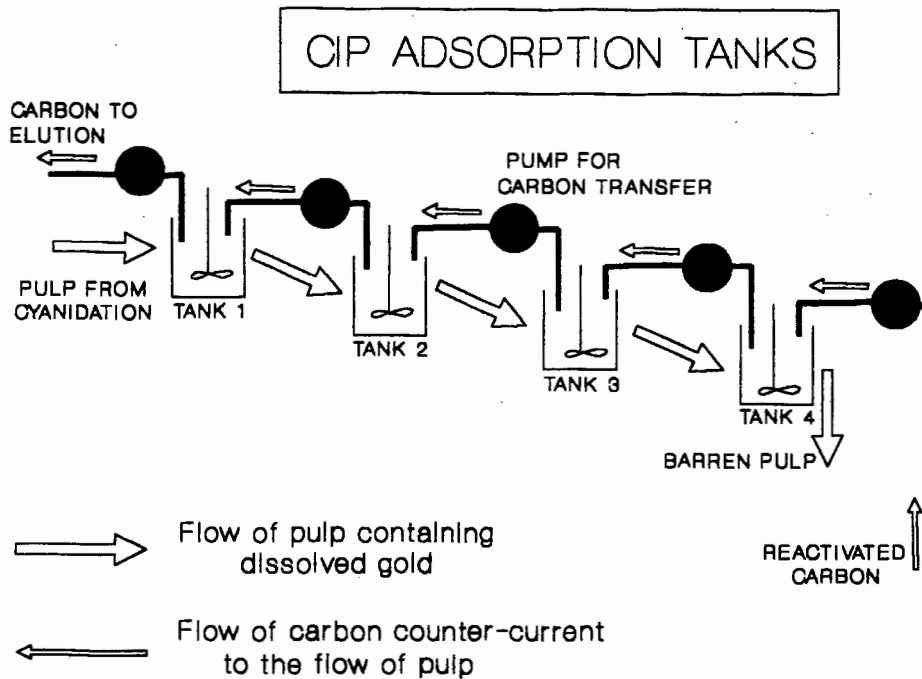


Figure 2.1 : THE CIP ADSORPTION CASCADE

Certain process aspects occur on operational plants and need to be mentioned but have been omitted in this simulation model since an understanding of the adsorption process requires only a simple model.

2.1.1.1 : NUMBER OF STAGES IN THE ADSORPTION SECTION

Under steady state conditions, the requirement of a certain barren gold value (gold in tailings) determines the amount of carbon needed⁵. As the number of stages is increased the total amount of carbon required is decreased. Coupled with this increase in number of tanks is an increase in gold lockup (gold retained in the adsorption circuit). Four stages will be used in this model.

2.1.2 : MIXING OF THE TANKS

Each of the tanks has a mechanical agitator, or stirrer, that in practice has a large effect on the distribution of slurry density and carbon concentration⁶. The approximation of a first order adsorption model has been used in previous simulations⁷. This assumes a perfectly mixed vessel and the first order assumption will be used here.

2.1.3 : INTERSTAGE SCREENING

The pulp flows downstream under the force of gravity. In order to keep the carbon from moving in the same direction, the pulp needs to pass through a screen. MINTEK developed the Equalised Pressure Air Cleaned (EPAC) screen which is widely used on operational plants⁸. For the model it will be assumed that all the carbon is retained in a tank by the screen.

2.1.4 : LEACHING OF THE ORE

The slurry entering the adsorption cascade is a water and solids mixture. The solids include ore that is still being leached into solution. Although models have been developed to incorporate leaching⁹, the model used in this thesis assumes that the slurry entering the first tank is fully leached.

2.1.5 : TRANSFER OF CARBON

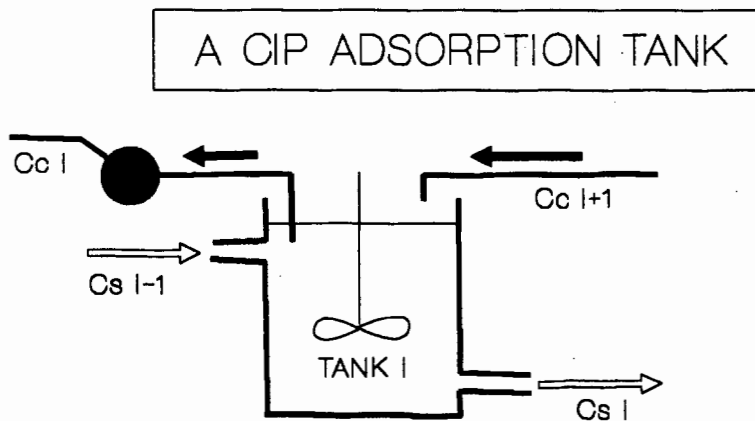
In practice, carbon is periodically transferred counter-current to the flow of pulp. Nicol¹⁰ has shown that there is very little difference in stage performance if a model is used in which the carbon is transferred continuously. It is the continuous counter-current transfer method that will be used in this thesis.

2.1.6 : BACKMIXING OF SLURRY

When pumping the carbon, a certain amount of backmixing is encountered¹¹. Backmixing is the transferral of slurry in the same direction as the carbon during the periods of carbon transfer. This occurs since the transfer of carbon using pumps normally results in a carbon-slurry mixture being transferred. This effect will not be considered.

2.2 : MODEL FOR ADSORPTION IN THE CIP PROCESS

Each tank in the cascade will be assumed to have the same reaction rates, as was done in previous CIP simulations¹². Thus a dynamic model for the entire adsorption section can be made up by modelling the mass balance across one reactor and cascading as many reactors as are required.



Slurry flowrate = F_s [Tonne slurry/hr]

Carbon flowrate = F_c [Tonne carbon/hr]

Mass of slurry in a tank = M_s [Tonne]

Mass of carbon in a tank = M_c [Tonne]

Concentration of Au in slurry = C_s [g Au/Tonne slurry]

Concentration of Au on carbon = C_c [g Au/Tonne carbon]

Figure 2.2 : A CIP ADSORPTION REACTOR TANK

Both flows are considered to be continuous and the tank is regarded as a continuous stirred batch reactor (CSTR)¹³.

2.2.1 : MASS BALANCE ACROSS A TANK

The rates of change of gold in solution, C_S , and the concentration of gold on carbon, C_C , can be described using a mass balance.

Rate of change of C_S in tank i :

$$\frac{dC_{Si}}{dt} = \frac{F_S}{M_S} [C_{Si-1} - C_{Si}] - \frac{M_C}{M_S} R_i \quad (2.1)$$

Rate of change of C_C in tank i :

$$\frac{dC_{Ci}}{dt} = \frac{F_C}{M_C} [C_{Ci+1} - C_{Ci}] + R_i \quad (2.2)$$

The factor R_i describes the rate at which gold is adsorbed onto the carbon. The mass balances described above have been checked and verified by Dr. I.J. Barker of MINTEK¹⁴.

2.2.2 : MODEL FOR THE RATE OF ADSORPTION

This is the most complex part of the CIP model. There are many factors that influence this rate¹⁵ and the development of an accurate rate model is the subject of ongoing research¹⁶. It has been found through data obtained on operational mines that the constants found for the rate expression vary from mine to mine¹⁷.

The first generally accepted CIP adsorption model was developed by Fleming, Nicol and Nicol in 1980¹⁸. However, this model does have its limitations. Firstly it assumes that C_S is at steady state, when in fact C_S varies as carbon is transferred. Secondly C_C has a significant effect on the rate of adsorption, a factor not catered for by this model.

Later Nicol, Fleming and Cromberge used a simple film diffusion mass transfer model¹⁹. They verified their model on a range of batch transfer data.

The model chosen for this simulator is the one proposed by Dixon²⁰. It takes the same form as that of Nicol, Fleming and Cromberg, but has two instead of one parameters. Menne²¹ has shown that this model fits continuous transfer data well and it is for this reason that the Dixon model was chosen.

$$R_i = k_1 C_{Si} (C_C^* - C_{Ci}) - k_2 C_{Ci} \quad (2.3)$$

The factors k_1 and k_2 are rate constants and C_C^* is a model parameter.

2.2.3 : CHOOSING INPUTS AND OUTPUTS

The choice of output is restricted to C_S since this is the only concentration for which an on-line measuring instrument has been developed. If further research allows C_C to be measured on-line then there is the further possibility of using C_C as an output.

The slurry flowrate cannot be varied as plants are designed to operate on a fixed tonnage of ore per month. Slurry flows downstream under gravity and consequently the mass of slurry in each tank cannot be varied. The pumps used to transfer carbon are generally constant speed drives and therefore the carbon flowrate cannot be changed.

The one variable that can be altered is the mass of carbon in each tank. The choice of the mass of carbon in a stage as an input variable has economic implications that need to be considered if a closed loop controller were designed for a CIP plant.

2.2.4 : CHOOSING OPERATING CONDITIONS

The parameters to be taken as operating conditions are those used by Stange²². These parameters are hypothetical and do not represent any real plant. The value of carbon mass is the initial mass in the tanks. The parameter values used are :

F_S	= 700	[Tonne slurry/hr]
F_C	= 0.6	[Tonne carbon/hr]
M_S	= 2592	[Tonne slurry]
M_C	= 14.4	[Tonne carbon]
C_{S0}	= 1.0	[g Au/Tonne slurry]
C_{C5}	= 50.0	[g Au/Tonne carbon]

The concentration of gold in solution that arrives from the leaching section is C_{S0} . After the gold has been stripped from carbon in the electrowinning cells, the carbon is regenerated. This regenerated carbon is fed into tank number four, but it still contains a small amount of gold, C_{C5} .

The constants used in the adsorption rate expression are :

$$k_1 = 0.017 \quad [1/(g \text{ Au/Tonne slurry}) \text{ 1/hr}]$$

$$k_2 = 0.005 \quad [1/hr]$$

$$C_C^* = 11\,650 \quad [g \text{ Au/Tonne carbon}]$$

2.3 : SIMULATION OF THE ADSORPTION SECTION

The equations derived and constants chosen were used to design a computer simulation of the adsorption section. Turbo Pascal was used as the computer language and a fourth order Runge-Kutta²³ technique used to solve the system of differential equations. A listing of the software written is given in Appendix A.

For a given set of inputs, M_C , the steady state values of C_S and C_C can be calculated. These values are found by rearranging equations 2.1 to 2.3 into the form presented in equations 2.4 and 2.5.

$$C_{Si} = \frac{F_S C_{Si-1} + M_C k_2 C_{Ci}}{F_S + M_C k_1 (C_C^* - C_{Ci})} \quad (2.4)$$

$$C_{Ci+1} = C_{Ci} - \frac{M_C}{F_C} \left[k_1 C_{Si} (C_C^* - C_{Ci}) - k_2 C_{Ci} \right] \quad (2.5)$$

The simulator first finds the steady state values for a given set of inputs. This is done by guessing C_{S1} and then calculating all values of C_S and C_C using equations 2.4 and 2.5. Once C_{C5} has been calculated, it can be compared to the specified value. This procedure is repeated until the calculated value of C_{C5} is equal to the given value.

A step test can then be performed, since the simulator is at steady state. If a series of four step tests are performed, the data obtained can be used to derive an open loop model for the CIP plant.

The open loop transfer function matrix, $G(s)$ will then be defined by :

$$\begin{bmatrix} Y_1 \\ Y_2 \\ Y_3 \\ Y_4 \end{bmatrix} = \begin{bmatrix} G_{11} & G_{12} & G_{13} & G_{14} \\ G_{21} & G_{22} & G_{23} & G_{24} \\ G_{31} & G_{32} & G_{33} & G_{34} \\ G_{41} & G_{42} & G_{43} & G_{44} \end{bmatrix} * \begin{bmatrix} U_1 \\ U_2 \\ U_3 \\ U_4 \end{bmatrix}$$

Where

Outputs Y_1 to $Y_4 = C_S$ in Tanks 1 to 4

Inputs U_1 to $U_4 = M_C$ in Tanks 1 to 4

Matrix G_{ij} = Open Loop Model

2.4 : ANALYSIS OF THE MODEL AT STEADY STATE

Before any step tests were performed the expected steady state values were calculated under five different conditions. These conditions are the mass of carbon in each tanks before any step is done and the masses in each tank after four successive step increases in M_C .

The steady state analysis is done in order to observe the effect additional carbon has on the amount of gold that can be recovered on the carbon and the amount of gold lost in the slurry tailings. The amount of interaction can be gauged by the magnitude of the gains.

Table 2.1 : CIP STEADY STATE VALUES (NO INPUT STEPPED)

Tank	M_C [T]	C_S [g/T]	C_C [g/T]
1	14.4	0.241853	1212.15
2	14.4	0.055560	327.65
3	14.4	0.013286	110.30
4	14.4	0.003871	60.98

Table 2.2 : CIP STEADY STATE VALUES (STEP INPUT 1)

Tank	M_C [T]	C_S [g/T]	C_C [g/T]
1	21.6	0.183341	1212.90
2	14.4	0.042161	260.13
3	14.4	0.010311	95.42
4	14.4	0.003226	58.27

Table 2.3 : CIP STEADY STATE VALUES (STEP INPUT 2)

Tank	M_C [T]	C_S [g/T]	C_C [g/T]
1	21.6	0.183363	1213.47
2	21.6	0.032058	260.73
3	14.4	0.008072	84.20
4	14.4	0.002741	56.22

Table 2.4 : CIP STEADY STATE VALUES (STEP INPUT 3)

Tank	M_C [T]	C_S [g/T]	C_C [g/T]
1	21.6	0.183379	1213.89
2	21.6	0.032071	261.17
3	21.6	0.006386	84.64
4	14.4	0.002376	54.68

Table 2.5 : CIP STEADY STATE VALUES (STEP INPUT 4)

Tank	M_C [T]	C_S [g/T]	C_C [g/T]
1	21.6	0.183392	1214.22
2	21.6	0.032081	261.51
3	21.6	0.006395	84.98
4	21.6	0.002101	55.01

The increase in carbon in each stage shows a decrease in C_S in the tailings and an increase in C_C in tank 1. This means a decrease in gold lost in tailing and an increase in gold recovered on the carbon.

The steady state data presented in tables 2.1 to 2.5 is used to evaluate the steady state gains of each transfer function. The matrix at zero frequency then becomes :

$$G(j0) = \begin{bmatrix} -0.008127 & 3.1E-6 & 2.2E-6 & 1.8E-6 \\ -0.001861 & -0.001403 & 1.8E-6 & 1.4E-6 \\ -0.000413 & -0.000311 & -0.000234 & 1.3E-6 \\ -0.000089 & -0.000067 & -0.000051 & -0.000038 \end{bmatrix}$$

These steady state gains indicate that there is negligible interaction from the upper diagonal elements. This gives the matrix a lower diagonal form. This form of a lower diagonal plant matrix indicates that there is no multivariable problem²⁴.

2.5 : ANALYSIS OF MODEL DYNAMICS

The dynamic analysis will give an indication as to the speed of response of the transfer functions and should also confirm the expected gains calculated in the previous section.

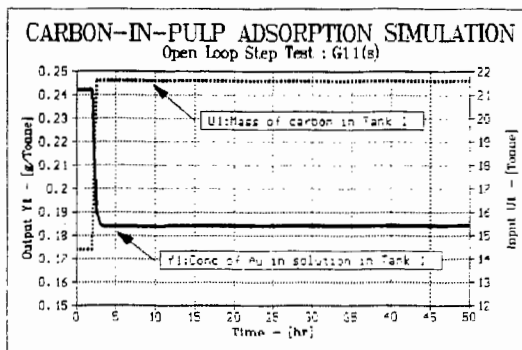


Figure 2.5.1 : CIP G11(s)

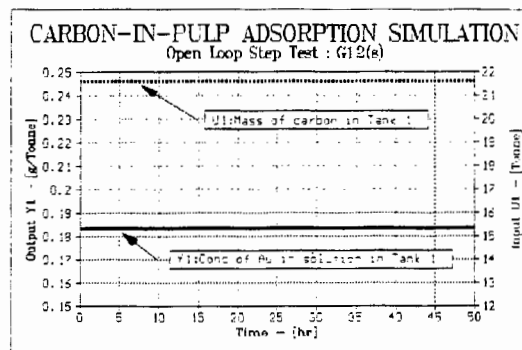


Figure 2.5.5 : CIP G12(s)

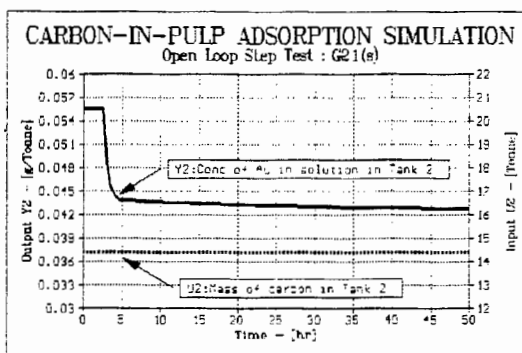


Figure 2.5.2 : CIP G21(s)

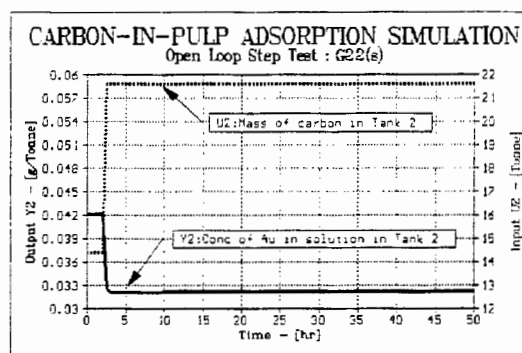


Figure 2.5.6 : CIP G22(s)

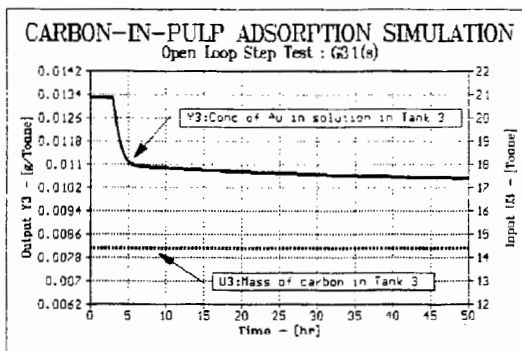


Figure 2.5.3 : CIP G31(s)

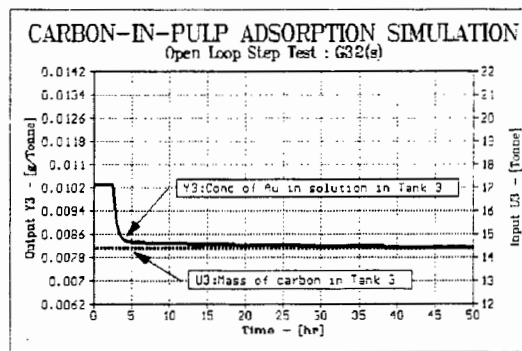


Figure 2.5.7 : CIP G32(s)

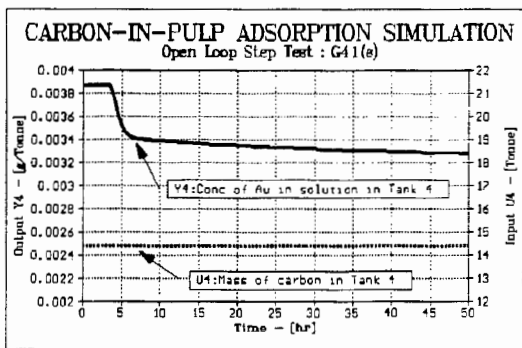


Figure 2.5.4 : CIP G41(s)

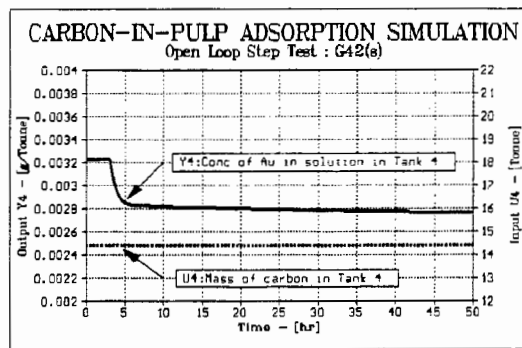


Figure 2.5.8 : CIP G42(s)

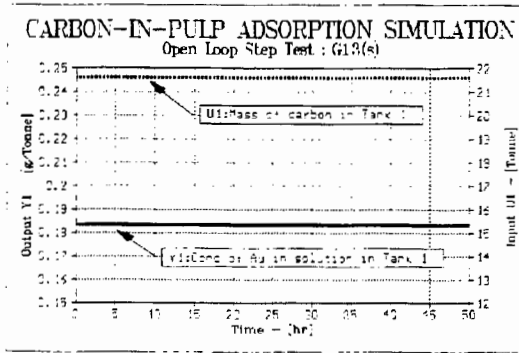


Figure 2.5.9 : CIP G13(s)

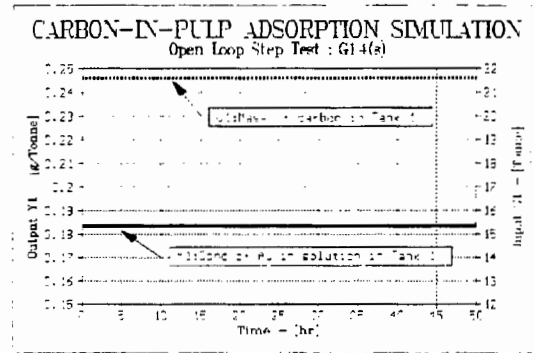


Figure 2.5.13 : CIP G14(s)

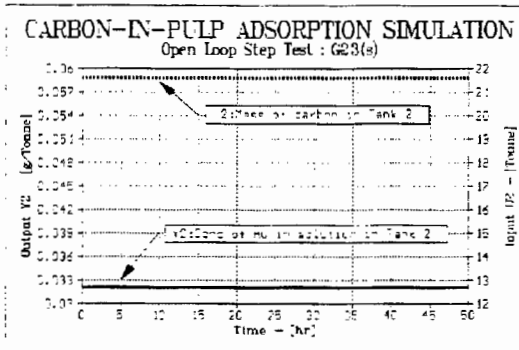


Figure 2.5.10 : CIP G23(s)

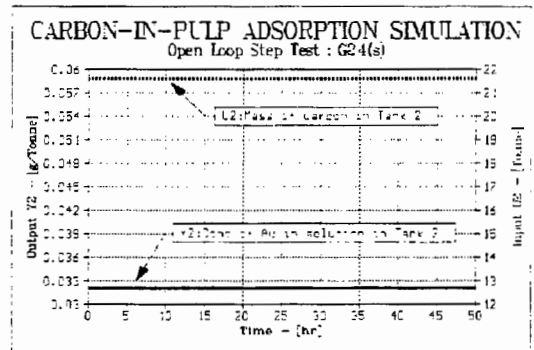


Figure 2.5.14 : CIP G24(s)

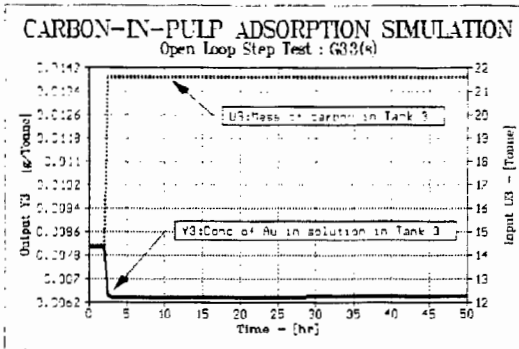


Figure 2.5.11 : CIP G33(s)

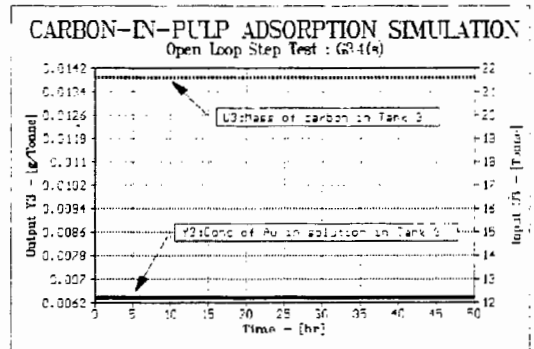


Figure 2.5.15 : CIP G34(s)

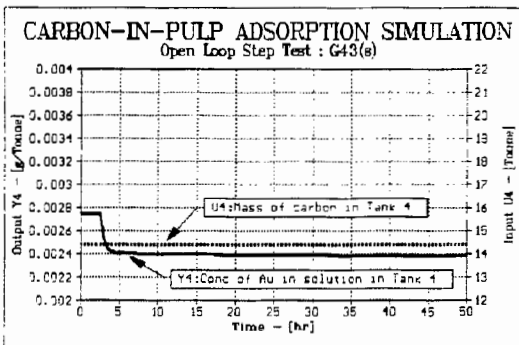


Figure 2.5.12 : CIP G43(s)

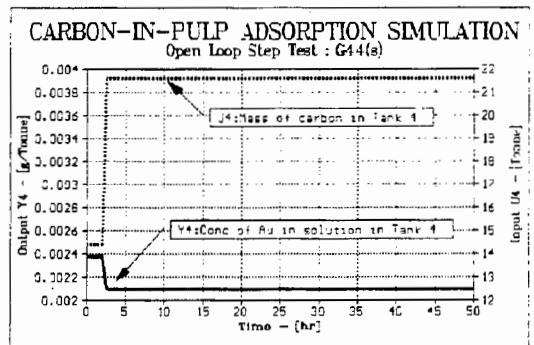


Figure 2.5.16 : CIP G44(s)

The simulations were performed over a long period of time, 50 hours. In this time the diagonal elements settled very quickly, within one hour. Some of the lower diagonal elements had not settled in this time, but it was assumed that since the diagonal elements had settled these would do so as well. Confirmation was also obtained of the expected gains and indication of negligible interaction from the upper diagonal elements.

2.6 : CONCLUSIONS ON THE CIP MODEL

The dynamic analysis confirms the gains expected from the steady state analysis. It further gives an indication of the speed of response. The reasons for the behaviour of the model will now be discussed.

2.6.1 : ELEMENTS ON AND BELOW THE DIAGONAL

The matrix elements on and below the diagonal have negative gains. The reason for this is that the addition of more carbon to a tank creates more surface area of contact for gold adsorption to take place and consequently more gold is absorbed onto the carbon. The concentration of gold in solution drops as a result.

The diagonal elements have an almost instantaneous rise. This is attributed to the assumption that there is an instantaneous change in M_C .

The slurry flows downstream under gravity. It has a short residence time in each tank due to the high flowrate. Because the slurry flows downstream quickly, the concentrations of gold in slurry in tanks downstream will also drop.

2.6.2 : ELEMENTS ABOVE THE DIAGONAL

The elements above the diagonal have a slight positive gain. The reason for this positive gain is that the carbon of now higher gold concentration (due to the added carbon) is pumped to preceding stages. The gold in solution in these stages now reacts with carbon of a higher gold loading. This causes less gold to be absorbed onto the carbon and thus the concentration in solution rises.

The gain is almost negligible due to the long residence time of the carbon. Since the carbon mass flowrate is nearly the same as the mass of carbon in the tank, the carbon in a tank spends almost 24 hours in a tank. Consequently it takes a long time for carbon of a higher gold loading to move to a preceding tank.

This concludes the study on the adsorption section of a CIP plant. It provides a basic understanding as to how a CIP plant reacts to changing amounts of carbon in the reactor tanks.

REFERENCES

1. POHLANDT-WATSON C., Historical, Analytical and Environmental Aspects of the Cyanide Process, *Cyanide in Gold Processing : The Next 100 Years*, Johannesburg, The South African Chemical Institute, 1990.
2. ADAMS M.D. and NICOL M.J., The Kinetics of the Elution of Gold from Activated Carbon, *GOLD 100 : Proceedings of the International Conference on Gold. Vol 2 : Extractive Metallurgy of Gold*, Johannesburg, SAIMM, pp. 111-121, 1986.

3. VAN DEVENTER J.S.J., Factors Influencing the Efficiency of a Carbon-in-Pulp Adsorption Circuit for the Recovery of Gold from Cyanided Pulps, *GOLD 100 : Proceedings of the International Conference on Gold. Vol 2 : Extractive Metallurgy of Gold*, Johannesburg, SAIMM, pp. 85-95, 1986.
4. STANLEY G.G., The Extractive Metallurgy of Gold in South Africa : Volume 1, Johannesburg : Kelvin House, pp.393-399, 1987.
5. STANLEY G.G., The Extractive Metallurgy of Gold in South Africa : Volume 1, Johannesburg : Kelvin House, pp.401-407, 1987.
6. TODTENHAUPT P., FORSCHNER P., GRIMSLEY D. and LASES A.B.M., Design of Agitators for the ERGO Carbon-in-Leach Plant, *GOLD 100 : Proceedings of the International Conference on Gold. Vol 2 : Extractive Metallurgy of Gold*, Johannesburg, SAIMM, pp. 195-207, 1986.
7. CARRIER C., HODOUIN D. and COURCHESNE, Dynamic Simulation of the CIP Gold Recovery Process, *Proceedings of the International Symposium on Gold Metallurgy*, Winipeg, pp.309-325, 1987.
8. STANLEY G.G., The Extractive Metallurgy of Gold in South Africa : Volume 1, Johannesburg : Kelvin House, pp.470-478, 1987.
9. STANGE W., The Simulation and Optimisation of the CIP Process, *Cyanide in Gold Processing : The Next 100 Years*, Johannesburg, The South African Chemical Institute, 1990.
10. NICOL M.J., FLEMING C.A. and CROMBERGE G., The Absorption of Gold Cyanide onto Activated Carbon. II. Application of the Kinetic Model to Multistage Absorption Circuits, *J. S. Afr. Inst. Min. Metall.*, Vol. 84, no. 3, pp.70-78, Mar 1984.

11. WOOLLACOTT L.C., STANGE W. and KING R.P., Towards more Effective Simulation of CIP-CIL Processes. Part I : The Modelling of Adsorption and Leaching, To be published in *J. S. Afr. Inst. Min. Metall.*, 1990.
12. VAN DER WALT T.J., Parametric Sensitivity Analysis of Abnormal Behaviour in the Dynamic Simulation of Carbon-in-Pulp Plants, B.sc. Dissertation, Department of Metallurgical Engineering, University of Stellenbosch, Nov 1989.
13. WILLIAMS D.F. and GLASSER D., The Modelling and Simulation of Processes for the absorption of Gold by Activated Charcoal, *J. S. Afr. Inst. Min. Metall.*, Vol. 85, no. 8, pp.237-243, Aug 1985.
14. BARKER I.J., MINTEK, Private Communication.
15. FLEMING C.A. and NICOL M.J., The Absorption of Gold Cyanide onto Activated Carbon. III. Factors Influencing the Rate of Loading and the equilibrium Capacity, *J. S. Afr. Inst. Min. Metall.*, Vol. 84, no. 4, pp.85-93, Apr 1984.
16. STANGE W., KING R.P. and WOOLLACOTT L.C., Towards more Effective Simulation of CIP-CIL Processes. Part II : A Population Balance Based Simulation Approach, To be published in *J. S. Afr. Inst. Min. Metall.*, 1990.
17. CARRIER C., HODOUIN D. and COURCHESNE, Dynamic Simulation of the CIP Gold Recovery Process, *Proceedings of the International Symposium on Gold Metallurgy*, Winnipeg, pp.309-325, 1987.
18. FLEMING C.E., NICOL M.J. and NICOL D.I., The Optimisation of a Carbon-in-Pulp Adsorption Circuit based on the Kinetics of Extraction of Aurocyanide by Activated Carbon, Presented at MINTEK Meeting "Ion Exchange and Solvent Extraction in Mineral Processing", MINTEK, Feb. 1980.

19. NICOL M.J., FLEMING C.A. and CROMBERGE G., The Absorption of Gold Cyanide onto Activated Carbon. I. The Kinetics of Absorption from pulps, *J. S. Afr. Inst. Min. Metall.*, Vol. 84, no. 2, pp.50-54, Feb 1984.
20. DIXON S., CHO E.H. and PITT C.H., The Interaction between Gold cyanide, Silver cyanide and High Surface Area Charcoal, *AIChE Symposium Series 74*, Vol. 74, no. 173, pp.75-83, 1978.
21. MENNE D., Predicting and Assessing Carbon-in-Pulp Performance, *CIM XIV, 1ST Mineral Processing Conference*, Toronto, 1982.
22. STANGE W., WOOLLACOTT L.C. and KING R.P., Towards more Effective Simulation of CIP-CIL Processes. Part III : Validation and Use of a New Simulator, To be published in *J. S. Afr. Inst. Min. Metall.*, 1990.
23. KREYSZIG E., Advanced Engineering Mathematics, New York : Wiley, pp.661-672, 1972.
24. BRAAE M., Lecture Notes : Advanced Control Systems, 1990.

CHAPTER 3 : COUNTER-CURRENT HEAT EXCHANGER RIG

A counter-current Heat Exchanger Rig (HERIG) was built and commissioned by Mr. A. De Waal at the University of Cape Town¹. It is felt that insight may be gained into the closed loop control of a CIP plant by studying the control of the HERIG, which mimics the CIP adsorption process.

The similarities and differences between the two needs to be studied to determine if they are indeed similar. But before a formal control study can be performed on the HERIG the principles of heat transfer taking place on the rig needs to be studied. It is felt that this is best achieved by designing a simulator (HES) of the heat exchanger rig.

3.1 : DESCRIPTION OF THE HERIG

The HERIG, displayed in Figure 3.1, consists of four tanks cascaded together. Water from a constant temperature cold reservoir is pumped into tank number 4. This COLD stream flows down towards tank number 1 under the force of gravity. The level in each tank is controlled by a valve on the outlet of each tank. Each of the tanks is stirred by means of a variable speed stirrer (VSS).

Inside each tank is a copper coil wound in a helical fashion. These coils contain water from a HOT stream. Heat is transferred from this HOT stream to the COLD stream water in the tank. The HOT stream originates from a constant temperature HOT reservoir. This HOT water is pumped in an opposite direction to the flow of COLD water. A counter-current temperature gradient is thus established across all four tanks, in both the HOT as well as the COLD streams.

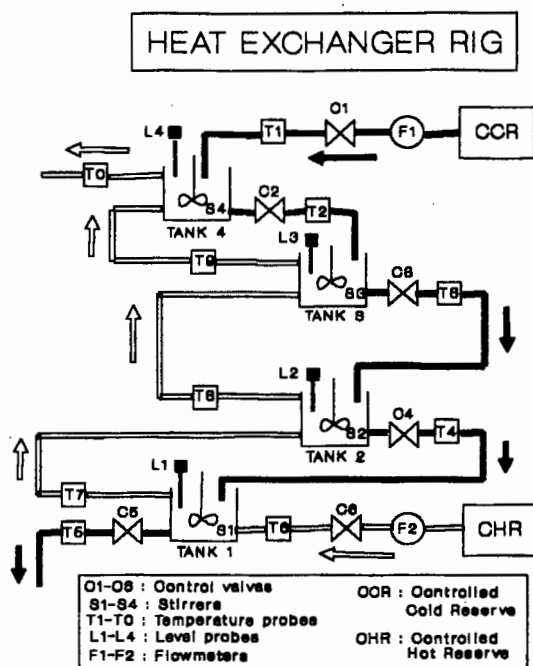


Figure 3.1 : THE HERIG CONFIGURATION

The flows out of each reservoir can be controlled by means of a valve. There are a further four valves connected to the outlets of the four tanks. All these valves can be controlled from a personal computer (PC) by means of a digital to analogue (D/A) card that is connected to a current to pressure (I/P) transducers which then set the valve openings. The four stirrers in each tank can also be controlled via the D/A card, giving a total of ten inputs that can be manipulated.

Readings for the four levels of COLD water in each tank are obtained from level probes connected through an analogue to digital (A/D) card. This card also has two flowmeter readings connected to it. The COLD stream reservoir temperature and the four COLD stream temperatures in the tanks can be read. Five corresponding readings in the HOT stream are also available. This provides a total of sixteen measurable outputs.

3.2 : COMPARISON BETWEEN CIP AND HERIG

On a CIP plant a flow of activated carbon adsorbs gold from a stream of slurry flow in a counter-current manner. The HERIG has a COLD stream of water absorbing heat from a HOT stream of water, also flowing counter-currently. Thus both processes have counter-current flows and the temperatures on the HERIG are analogous to the gold concentrations on a CIP plant.

3.2.1 : HOT STREAM OF WATER

The HOT stream mimics the flow of slurry on a CIP process. Water from a constant temperature reservoir is pumped at a constant flowrate through a length of pipe. In each tank heat is lost to the water in the tank. Thus a temperature gradient, going from a higher to a lower temperature, is established across all the tanks due to the loss of heat in each tank.

A CIP plant operates on a fixed throughput of slurry (tonnes per hour). The concentration of gold in solution from the leaching section is nearly constant. In the adsorption stages of a CIP cascade the gold concentration in slurry decreases the further it travels down the cascade, due to the adsorption of gold onto activated carbon.

3.2.2 : COLD STREAM OF WATER

The COLD stream operates in a similar manner to the carbon on a CIP plant. The level of COLD water in each tank can be controlled, as can the mass of carbon in each reactor for a plant that transfers carbon continuously.

The water begins at a constant COLD temperature, while regenerated carbon arrives with a low but constant gold loading. The water heats up in each tank as heat is transferred from the HOT stream into the COLD stream. In much the same way the gold loading on carbon increases as more gold is adsorbed from the slurry solution.

3.2.3 : LEVEL OF WATER IN A TANK

On a CIP plant an increase in the mass of carbon in a reactor causes the gold concentration in solution to decrease. Similarly on the HERIG if the level of COLD water in the tank is raised, more surface area of the pipe is submerged and the temperature of the HOT water in the pipe decreases, due to more heat being transferred into the tank.

As the level of water in a tank is raised, the surface area of contact with the helically wound copper pipe is increased. This increase in surface area allows more heat to be transferred from the HOT water inside the pipe and into the COLD water. On the CIP process, when more carbon is added to a reactor tank more gold is adsorbed from solution due to the increased surface area.

Consequently the level of water in a tank is analogous to the mass of carbon in a CIP reactor tank.

3.2.4 : RATE OF HEAT TRANSFER

The expression for the rate of gold adsorption onto carbon has parameters of constant value. The rate at which heat is transferred into the COLD stream of water is proportional to the rate of stirring. Thus on the HERIG the stirrers are kept at constant speed.

The expressions for the respective rates of transfer are very different. As a result of this the shape of the dynamic response of the HERIG will be different to that of the CIP model.

3.2.5 : TEMPERATURE GRADIENT

The COLD stream reacts to an increase in level in the same manner as carbon behaves to increase the mass of carbon in a reactor. Thus it is expected that the transfer functions for the HERIG will be positive above the diagonal and negative on and below the diagonal; the same as was found for the CIP simulator.

The gradient of concentrations on the CIP plant is great, the concentration of Au in slurry dropping by a few orders of magnitude. On the HERIG the HOT stream temperatures do not drop an order of magnitude. Thus it can be expected that the gains of the transfer functions will be very different to the CIP model gains.

3.3 : MODEL FOR THE HERIG

A model for the HERIG will be designed using heat balances. As with the CIP model, each tank is considered to have the same rate of transfer. The geometry of the copper pipe in the tank will not be taken into account. It will be assumed that there is a length of straight pipe in each tank.

A stabilising level controller will be designed for the HERIG. This will ensure that the levels never overflow or run dry. A change in the level will however not occur instantaneously, but it is known from the work done by De Waal that the level response time is far quicker than the temperature response time. For this reason the model will assume instantaneous level changes, similar to the assumption made about carbon mass for the CIP model.

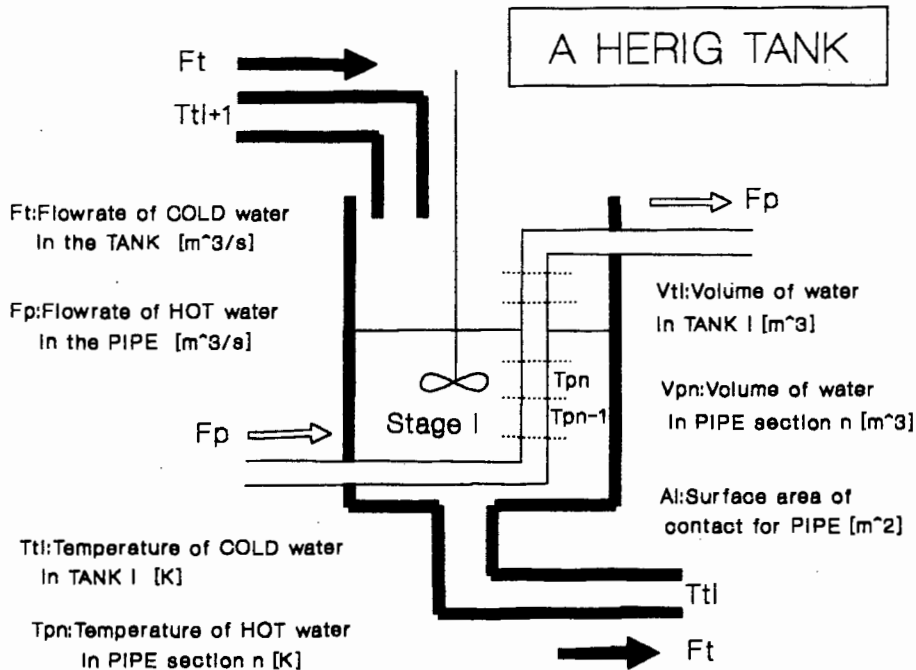


Figure 3.2 : A HERIG REACTOR TANK

The derivation of a set of heat balance equations to describe the heat exchanger requires the use of physical constants for water and air, as well as parameters from the actual sizes of pipe and tanks used on the HERIG. These values are tabulated in Appendix B.

3.3.1 : HEAT BALANCE ACROSS A TANK

To describe the heat balance across the heat exchanger, a lumped parameter model was used². In this type of model the helically wound pipe in each tank is assumed to be a long straight pipe. This length of pipe is divided into a number of small sections, in each of which the temperature is constant.

The temperature of the COLD water in the tank is assumed to be constant. The dynamic behaviour of a heat exchanger with constant wall temperature, the temperature of the COLD water in this case, is well documented³.

The level of water in each tank determines the number of pipe sections that are submerged under the COLD water in the tank. The number of pipe sections submerged in a tank is denoted by L_i . Each submerged section of pipe transfers heat into the COLD tank water. The balance across the pipe sections for $1 \leq n \leq L_i$:

$$\frac{dT_{Pn}}{dt} = \frac{F_P}{V_{Pi}} [T_{Pn-1} - T_{Pn}] - \frac{U_W A_i}{P_W C_W V_{Pi}} [T_{Pn} - T_{Ti}] \quad (3.1)$$

The sections of pipe not submerged under the COLD water lose heat to the surrounding air, which is assumed to be at a constant value of room temperature T_{RM} . The balance across pipe section for $L_i < n \leq \text{Max } L_i$:

$$\frac{dT_{Pn}}{dt} = \frac{F_P}{V_{Pi}} [T_{Pn-1} - T_{Pn}] - \frac{U_A A_i}{P_W C_W V_{Pi}} [T_{Pn} - T_{RM}] \quad (3.2)$$

The COLD water in each tank accumulates heat from each submerged section of pipe. Thus the heat transferred into the tank is found by summing the contributions of each section of HOT water pipe up to the level of water in the tank. The balance for the tank is then :

$$\frac{dT_{Ti}}{dt} = \frac{F_T}{V_{Ti}} [T_{Ti+1} - T_{Ti}] + \sum_{n=1}^{L_i} \frac{U_W A_i}{P_W C_W V_{Ti}} [T_{Pn} - T_{Ti}] \quad (3.3)$$

These differential equations have terms describing the rate at which heat is transferred from the HOT to the COLD water. Equations 3.1 and 3.3 are for the submerged sections of pipe and thus use an overall heat transfer coefficient to water, U_W . Similarly Equation 3.2 uses a coefficient of transfer to air, U_A . The calculation of these constants will be presented next.

3.3.2 : OVERALL HEAT TRANSFER COEFFICIENT TO WATER

In transferring heat from the HOT to the COLD water, three thermal resistance have to be overcome. They are the transfer from HOT fluid through the inner wall, the resistance of the copper pipe and finally the transfer from the pipe through the outer wall.

Using the simplification that the copper pipe wall resistance is negligible, the overall heat transfer coefficient to water is⁴ :

$$U_W = \frac{1}{\frac{1}{H_{OW}} + \frac{D_O}{D_i} \frac{1}{H_{iW}}} \left[\frac{J}{\text{sm}^2\text{K}} \right] \quad (3.4)$$

This overall coefficient is now only dependant on an inner and an outer heat transfer coefficient.

3.3.2.1 : INNER HEAT TRANSFER COEFFICIENT

The assumption is made that the flow in the pipe is turbulent. The coefficient for the transfer of heat from the HOT water in the pipe and through the inner wall is⁵ :

$$H_{iW} = Nu_W \frac{k_W}{D_i} \left[\frac{J}{sm^2 K} \right] \quad (3.5)$$

This requires calculation of the Nusseldt number for the water in the pipe, Nu_W . The water in the pipe is assumed to be turbulent and Nu_W can be calculated using the relation⁶ :

$$Nu_W = 0.023 Re_W^{0.8} Pr_W^{0.4} \quad (3.6)$$

The Reynolds, Re_W , and Prandtl, Pr_W , numbers for the water in the pipe are calculated as follows⁷ :

$$Re_W = \frac{p_W (F_P/A_i) D_i}{u_W} \quad (3.7)$$

$$Pr_W = \frac{C_W u_W}{k_W} \quad (3.8)$$

Flow is assumed to be turbulent when the Reynolds number is greater than 3000⁸. When checked with the data in Appendix B, Re_W was found to be 6750. This confirms the assumption made about the flow in the pipe being turbulent.

3.3.2.2 : OUTER HEAT TRANSFER COEFFICIENT

The water in the tank is mechanically agitated. Thus the stirred COLD water receives heat from the submerged coils at a rate governed by⁹ :

$$H_{OW} = 0.87 \frac{k_W}{D_T} \left[\frac{L_S^2 N_{SPW}}{u_W} \right]^{2/3} \left[\frac{C_W u_W}{k_W} \right]^{1/3} \left[\frac{J}{sm^2 K} \right] \quad (3.9)$$

3.3.3 : OVERALL HEAT TRANSFER COEFFICIENT TO AIR

Using the same simplification that the copper pipe wall resistance is negligible, the overall heat transfer coefficient to air is :

$$U_A = \frac{1}{\frac{1}{H_{OA}} + \frac{D_O}{D_i} \frac{1}{H_{iA}}} \quad \left[\frac{J}{sm^2 K} \right] \quad (3.10)$$

This overall coefficient is dependant on an inner and an outer heat transfer coefficient.

3.3.3.1 : INNER HEAT TRANSFER COEFFICIENT

The inner coefficient used here is the same as was used previously, since the heat transfer from inside the pipe and through the inner wall remains the same.

$$H_{iA} = Nu_W \frac{k_W}{D_i} \quad \left[\frac{J}{sm^2 K} \right] \quad (3.11)$$

3.3.3.2 : OUTER HEAT TRANSFER COEFFICIENT

The transfer of heat through the outer copper layer to the surrounding air is assumed to be natural convection. Since the coils are wound horizontally, the natural convection from horizontal cylinders is :

$$H_{OA} = Nu_A \frac{k_A}{D_O} \quad \left[\frac{J}{sm^2 K} \right] \quad (3.12)$$

Where the Nusseldt number for air is¹⁰ :

$$Nu_A = 0.53 \left[Gr_A Pr_A \right]^{1/4} \quad (3.13)$$

The Prandtl, Pr_A , and Grashof, Gr_A , numbers for air are¹¹ :

$$Pr_A = \frac{C_A u_A}{k_A} \quad (3.14)$$

$$Gr_A = g \frac{2}{(T_{RM} + T_{WL})} \frac{(T_{WL} - T_{RM}) D_i^3}{(u_A / p_A)^2} \quad (3.15)$$

These balance equations have been checked and verified¹¹.

3.3.4 : OPERATING CONDITIONS FOR THE HERIG

The operating conditions set for the reservoir temperatures and flowrates, are those that will be used on the actual rig.

Hot reserve temperature	$T_{P0} = 58 \text{ } [^{\circ}\text{C}]$
Cold reserve temperature	$T_{T5} = 23 \text{ } [^{\circ}\text{C}]$
Hot stream flowrate	$F_P = 2.5 \text{ [l/min]}$
Cold stream flowrate	$F_T = 5.0 \text{ [l/min]}$

These operating conditions complete the data required to perform a numerical simulation of the HERIG. The calculations of the heat transfer coefficients, together with the values of the constants needed, is given in Appendix B.

3.4 : SIMULATION OF THE HEAT EXCHANGER RIG

As with the CIP simulation, a computer simulation was written using Turbo Pascal. The differential equations were once again solved using a fourth order Runge-Kutta routine. A listing of the coding of the heat exchanger simulator is given in Appendix C.

In order to find the steady state values of temperatures, equations 3.1 to 3.3 needed to be rearranged. The pipes in each tank were all divided into 20 sections. Thus in each tank number i , the temperature in pipe section n can be calculated. For the sections submerged under water :

$$T_{Pn} = \left[\frac{P_W C_W F_{Pi}}{P_W C_W F_{Pi} + U_W A_i} \right] T_{Pn-1} + \left[\frac{U_W A_i}{P_W C_W F_{Pi} + U_W A_i} \right] T_{Ti} \quad (3.16)$$

While for the sections losing heat to the surrounding air :

$$T_{Pn} = \left[\frac{P_W C_W F_{Pi}}{P_W C_W F_{Pi} + U_A A_i} \right] T_{Pn-1} + \left[\frac{U_A A_i}{P_W C_W F_{Pi} + U_A A_i} \right] T_{RM} \quad (3.17)$$

Once all the temperatures for the submerged pipe sections have been calculated, the temperature of the water in the tank is calculated as :

$$T_{Ti+1} = T_{Ti} - \sum_{n=1}^{L_i} \frac{U_W A_i}{P_W C_W F_{Ti}} [T_{Pn} - T_{Ti}] \quad (3.18)$$

The method of finding the correct steady state conditions is similar to that used for the CIP simulator. For the first tank, T_{T1} needs to be guessed. Then T_{Pn} can be calculated for all sections of pipe in tank 1 (equations 3.16 to 3.17). The temperature of the water in tank 2 can also be calculated (equation 3.18). This is done until T_{T5} is found.

The calculated value of T_{T5} can then be compared to the specified value. Based on the difference of these values the guess of T_{T1} is altered and the procedure is repeated until T_{T5} is sufficiently close to the specified value.

As with the CIP model, the HES is a four input and four output system. The open loop transfer function matrix, $G(s)$ will then be defined by :

$$\begin{bmatrix} Y_1 \\ Y_2 \\ Y_3 \\ Y_4 \end{bmatrix} = \begin{bmatrix} G_{11} & G_{12} & G_{13} & G_{14} \\ G_{21} & G_{22} & G_{23} & G_{24} \\ G_{31} & G_{32} & G_{33} & G_{34} \\ G_{41} & G_{42} & G_{43} & G_{44} \end{bmatrix} * \begin{bmatrix} U_1 \\ U_2 \\ U_3 \\ U_4 \end{bmatrix}$$

Where

Outputs Y_1 to $Y_4 = T_p$ in Tanks 1 to 4

Inputs U_1 to $U_4 = L$ in Tanks 1 to 4

Matrix G_{ij} = Open Loop Model

3.5 : ANALYSIS OF THE MODEL AT STEADY STATE

Using equations 3.15 to 3.17 the steady state values under five different conditions can be evaluated. These conditions correspond to the level of water in each tank before and after four different steps.

Table 3.1 : HES STEADY STATE VALUES (NO INPUT STEPPED)

Tank	L [%]	T_p [°C]	T_T [°C]
1	20.0	53.810057	33.204784
2	20.0	47.692402	31.195115
3	20.0	40.142961	28.254764
4	20.0	36.818572	24.611485

Table 3.2 : HES STEADY STATE VALUES (STEP INPUT 1)

Tank	L [%]	T _P [°C]	T _T [°C]
1	80.0	46.318420	35.027110
2	20.0	41.688307	29.202427
3	20.0	35.974555	26.977039
4	20.0	33.458509	24.219643

Table 3.3 : HES STEADY STATE VALUES (STEP INPUT 2)

Tank	L [%]	T _P [°C]	T _T [°C]
1	80.0	47.176122	36.720861
2	80.0	36.046529	31.325621
3	20.0	32.057690	25.776418
4	20.0	30.301209	23.851447

Table 3.4 : HES STEADY STATE VALUES (STEP INPUT 3)

Tank	L [%]	T _P [°C]	T _T [°C]
1	80.0	47.808345	37.969346
2	80.0	37.328833	32.890654
3	80.0	29.120060	27.668025
4	20.0	27.933248	23.575302

Table 3.5 : HES STEADY STATE VALUES (STEP INPUT 4)

Tank	L [%]	T _P [°C]	T _T [°C]
1	80.0	48.117318	38.579490
2	80.0	37.955505	33.655498
3	80.0	29.995690	28.592467
4	80.0	26.737066	24.625893

Tables 3.1 to 3.5 show that for an increase in the level of water in each tank, the temperature of the water in the pipe for that tank decreases. Also T_P decreases in all tanks further down the cascade and T_P increases further upstream.

The open loop model at steady state is :

$$G(j0) = \begin{bmatrix} -0.124861 & 0.014295 & 0.010537 & 0.005150 \\ -0.100068 & -0.094030 & 0.021372 & 0.010445 \\ -0.069473 & -0.065281 & -0.048961 & 0.014594 \\ -0.056001 & -0.052622 & -0.039466 & -0.019936 \end{bmatrix}$$

This steady state analysis has shown that the signs of the transfer functions are the same for the HERIG and CIP. This indicates that the level of COLD water does mimic the mass of carbon in a tank and the temperature of the HOT water does also mimic the concentration of gold in solution. The magnitudes of the functions are less for the lower diagonal than for the on diagonals in a column. This is the same for the CIP model. Thus by scaling of the inputs and outputs the HERIG could be made to look very similar to CIP.

The upper diagonal elements have a larger gain when compared to the on diagonals than was found on the CIP model. This is due to the ratio of residence times for the HOT and COLD streams being less than the ratio for the slurry and carbon streams.

3.6 : ANALYSIS OF MODEL DYNAMICS

Four step tests are needed to extract the sixteen open loop transfer functions. It was found that a long time period was needed, 1000 seconds, before steady state was reached. The dynamic responses will be plotted in order to compare the shape and speed of response of the HERIG to that of the CIP model. It can be expected that they will be different due to the different mechanisms of transfer for heat and for gold adsorption.

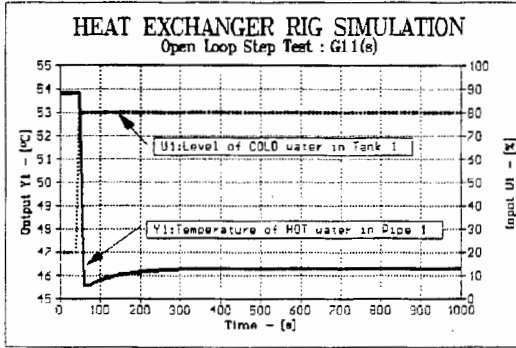


Figure 3.6.1 : HES G11(s)

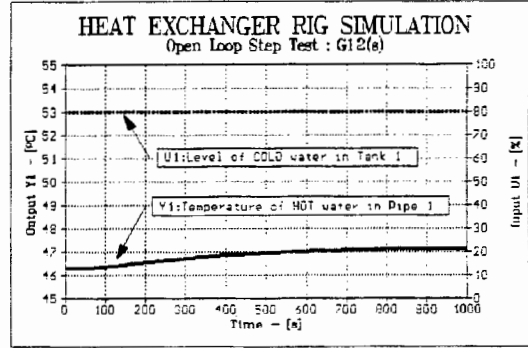


Figure 3.6.5 : HES G12(s)

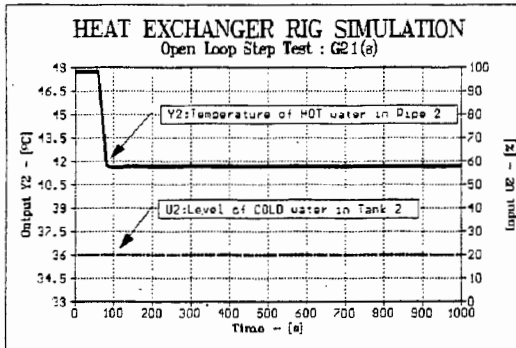


Figure 3.6.2 : HES G21(s)

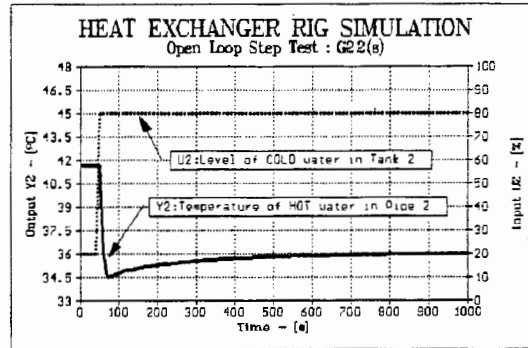


Figure 3.6.6 : HES G22(s)

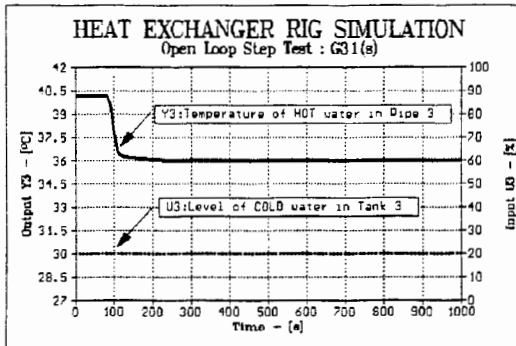


Figure 3.6.3 : HES G31(s)

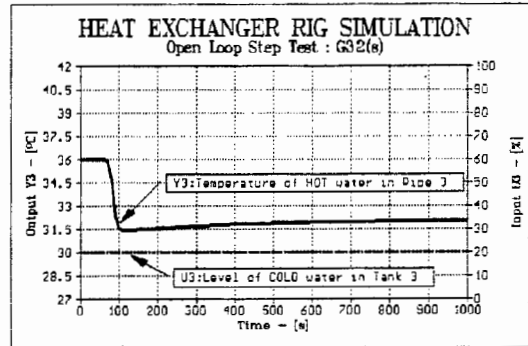


Figure 3.6.7 : HES G32(s)

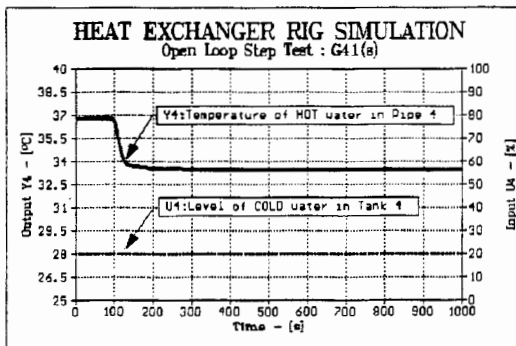


Figure 3.6.4 : HES G41(s)

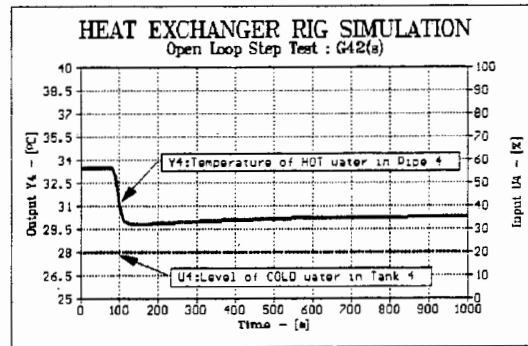


Figure 3.6.8 : HES G42(s)

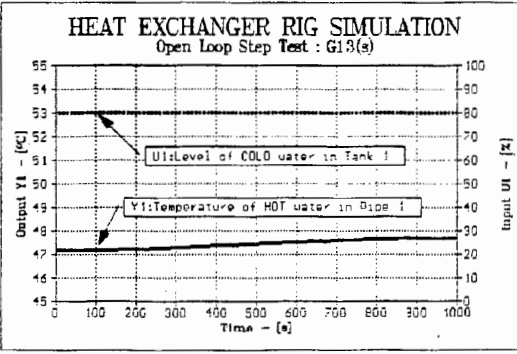


Figure 3.6.9 : HES G13(s)

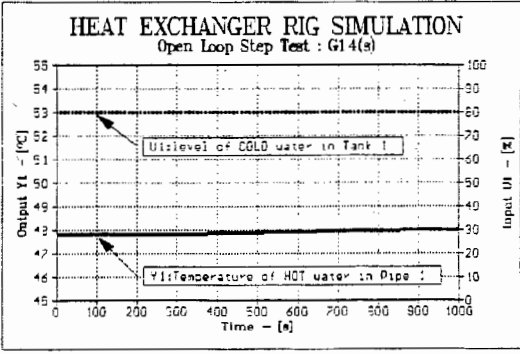


Figure 3.6.13 : HES G14(s)

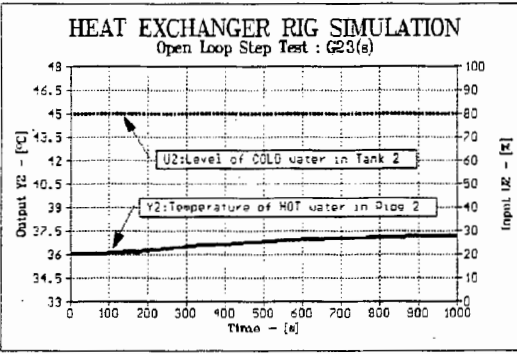


Figure 3.6.10 : HES G23(s)

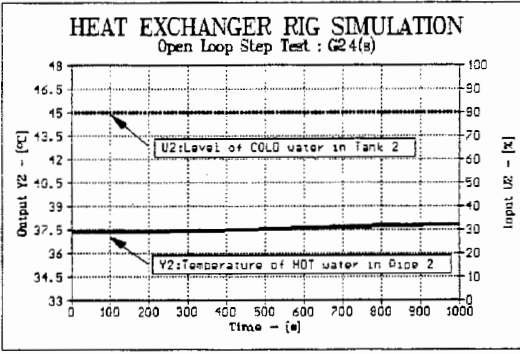


Figure 3.6.14 : HES G24(s)

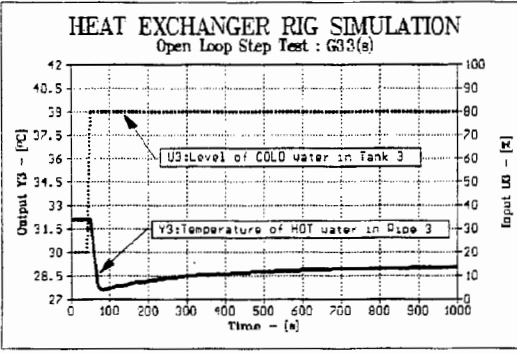


Figure 3.6.11 : HES G33(s)

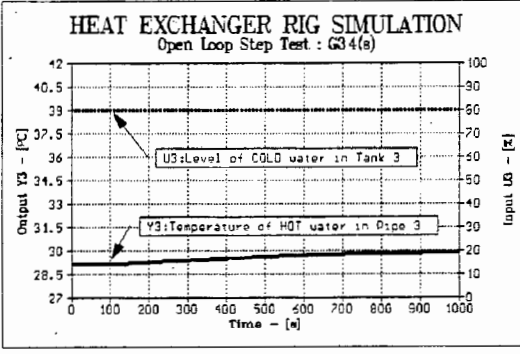


Figure 3.6.15 : HES G34(s)

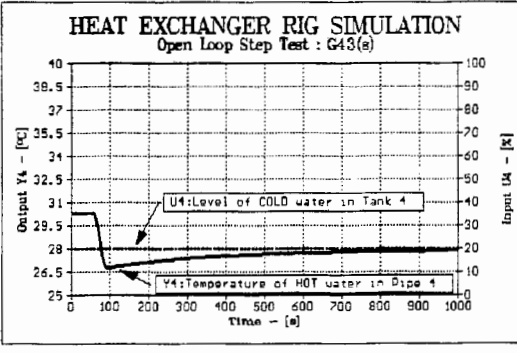


Figure 3.6.12 : HES G43(s)

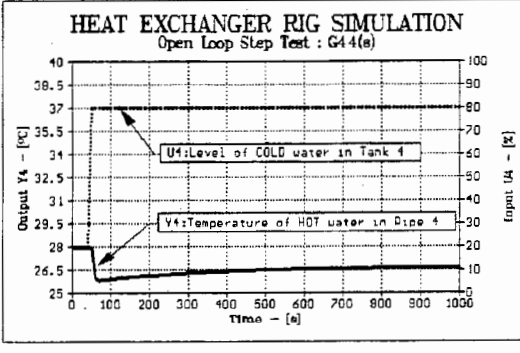


Figure 3.6.16 : HES G44(s)

The on-diagonal elements show a second order response that is characterised by two complex poles. The overshoot is caused by the instantaneous change in level. As the surface area is increased a far greater amount of heat is transferred extremely quickly. If the level response was not assumed to be instantaneous it is expected that there will be no overshoot.

The lower diagonals show a decrease in the HOT pipe temperature. This is caused by the HOT water in the previous tanks being cooled by the increased level of COLD water and this now colder HOT water being pumped to the succeeding tanks.

The upper diagonals show a second order rise that is characterised by two real poles, one slow and one fast. The temperature rises due to less heat being transferred out of the pipe since the temperature of the COLD water in the tank has risen. As the temperature in the PIPE section downstream decreases, due to the increased level of COLD water, more heat is transferred from pipe sections further upstream in an attempt to keep a thermal balance. This is happening at the same time as less heat is being lost to the COLD water. These two processes happen at two different rates, causing a second order response.

3.7 : CONCLUSIONS ON THE HERIG AND CIP SIMILARITIES

This chapter has so far presented a model of the HERIG and an analysis of the dynamic and steady state responses. The model was partly derived to gain an understanding of the HERIG and to observe the similarities between the HERIG and the CIP process. It was hoped that the HERIG would mimic the CIP process and the conclusions drawn in this respect will now be summarised.

3.7.1 : CHOICE OF INPUTS AND OUTPUTS

The signs of the corresponding transfer functions for the CIP model and those on the HERIG are the same. This indicates that the inputs and outputs on both processes behave in the same manner. On the HERIG the tank temperature can be measured. Thus if instrumentation is developed to measure on-line loading of gold on carbon, the HERIG may further mimic the CIP process.

3.7.2 : RESPONSE OF THE ON-DIAGONAL ELEMENTS

Both the CIP and HERIG on-diagonal responses show an overshoot. They both look likely to be modelled by a second order function with two complex poles. The overshoot of the HERIG model is more than that of the CIP model and consequently the pole positions will be different. The fact that they could both be modelled by the same class of function is very encouraging.

3.7.3 : RESPONSE OF THE OFF-DIAGONAL ELEMENTS

The lower diagonal elements shown a remarkably similar type of response. The type of function has not been identified but with scaling of the inputs and outputs the corresponding functions could be made to look very similar indeed.

The upper diagonal elements show the only marked difference in the processes. They are both positive gains, but the magnitudes are very different when compared to the respective on-diagonals. The CIP elements are much smaller than the corresponding HERIG elements and thus will pose far less of a control problem.

Thus in conclusion of the simulation study of the HERIG, it is felt that the HERIG does mimic the CIP process in many respects. A formal control study of the HERIG plant can be performed and the conclusions drawn as to the control of this process can then be related to the control of the CIP process.

REFERENCES

1. DE WAAL A., Performance Index Analysis of Control Systems Synthesised for a Counter-Current Process, M.Sc. Dissertation, Department of Electrical Engineering, University of Cape Town, April 1990.
2. HIMMELBLAU D.M. and BISCHOFF K.B., Process Analysis and Simulation, Wiley, pp.222-223, 1968.
3. SEINFELD J.H. and LAPIDUS L., Mathematical Modeling in Chemical Engineering Volume 3 Process Modeling, Estimation and Identification, New Jersey : Prentice-Hall, pp.134-138, 1974.
4. KERN D.Q., Process Heat Transfer, Tokyo : McGraw-Hill, pp. 86-87, 1950.
5. HOLMAN J.P., Heat Transfer, Tokyo : McGraw-Hill, pp. 210-212, 1976.
6. HOLMAN J.P., Heat Transfer, Tokyo : McGraw-Hill, pp. 187-190, 1976.
7. HOLMAN J.P., Heat Transfer, Tokyo : McGraw-Hill, pp. 147-172, 1976.
8. WELTY J.R., WICKS C.E. and WILSON R.E., Fundamentals of Momentum, Heat and Mass Transfer, New York : Wiley, pp. 169-170, 1976.
9. KERN D.Q., Process Heat Transfer, Tokyo : McGraw-Hill, pp. 720-727, 1950.
10. HOLMAN J.P., Heat Transfer, Tokyo : McGraw-Hill, pp. 235-243, 1976.
11. HOLMAN J.P., Heat Transfer, Tokyo : McGraw-Hill, pp. 250-251, 1976.

12. SWARTZ C.L.E., Department of Chemical Engineering,
University of Cape Town, Private Communication.

CHAPTER 4 : STRUCTURAL ANALYSIS OF THE HERIG

The HERIG is a MIMO system consisting of 10 inputs and 16 outputs, as shown by Figure 3.1. As it stands at present a closed loop control system cannot be designed to track setpoints since there are more outputs to be controlled than inputs that can be manipulated¹. Not only does the number of controlled outputs have to be reduced, but certain inputs may not affect any outputs or certain outputs be affected by any inputs.

A structural study can be performed on the HERIG to determine where there is interaction. This kind of analysis is done using a Binary Interaction Matrix (BIM)². The method does not quantify the interaction between inputs and outputs but merely specifies whether there is or is not an input-output connection.

4.1 : BINARY INTERACTION MATRIX (BIM) STUDY OF THE HERIG

The BIM matrix is a matrix of ones ('1') and zeros ('0'). It has as many rows as there outputs and as many columns as there are inputs. A matrix element containing a '1' indicates an interaction exists between the corresponding input and output. A '0' indicates there is no interaction. Once again it must be stressed that a '1' element gives no value to the magnitude of the actual interaction.

For the HERIG the ten inputs consist of six control valves and four stirring rates. The sixteen outputs are made up of ten temperature sensors, four level probes and two flowmeters. The BIM matrix presented will represent a '1' with a 'X' and a '0' will be represented by a blank. This makes the matrix more easy to read.

The full scale BIM matrix for the HERIG is then :

	C1	C2	C3	C4	C5	C6	S1	S2	S3	S4
F1	X									
F2						X				
L1				X	X					
L2			X	X						
L3		X	X							
L4	X	X								
T1										
T2	X	X	X	X	X	X	X	X	X	X
T3	X	X	X	X	X	X	X	X	X	X
T4	X	X	X	X	X	X	X	X	X	X
T5	X	X	X	X	X	X	X	X	X	X
T6										
T7	X	X	X	X	X	X	X	X	X	X
T8	X	X	X	X	X	X	X	X	X	X
T9	X	X	X	X	X	X	X	X	X	X
T0	X	X	X	X	X	X	X	X	X	X

Figure 4.1.1 : FULL ORDER BIM MATRIX

where :

F1,F2 : Flowrates in Tank and Pipe
 L1,L2,L3,L4 : Levels in Tanks 1 to 4
 T1 : Temperature of COLD reserve
 T5,T4,T3,T2 : COLD water temperature in Tanks 1 to 4
 T6 : Temperature of HOT reserve
 T7,T8,T9,T0 : HOT water temperature in Pipes 1 to 4
 C1 : Control valve for Tank flowrate
 C5,C4,C3,C2 : Control valve for Tanks 1 to 4
 C6 : Control valve for Pipe flowrate
 S1,S2,S3,S4 : Stirring rates for Tanks 1 to 4

4.1.1 : DESCRIPTION OF THE FULL SCALE BIM MATRIX

The temperatures T1 and T6 are unaffected by any of the inputs. They represent the reservoir temperatures which are fixed and cannot be altered by the plant inputs.

The stirring rates affect the rates of heat transfer in the respective tanks. Consequently they will affect the temperatures of the COLD water in the tank and the HOT water in the Pipe. There is no affect on the flowrates if the stirrers are changed and if splashing is regarded as a disturbance, then they do not change the levels either.

Control valve C6 changes the flow of water in the pipe. This will also affect the temperatures of the HOT water in the pipe and consequently the temperatures of the COLD tank water, because of the heat transfer. Control valve C1 affects the flowrate of COLD water coming from the COLD reservoir. The water from the cold reserve flows into the last tank and thus C1 also affects the level of water in tank 4.

Valves C2, C3 and C4 are at the outlets of tanks 4, 3 and 2 respectively. Thus they affect not only the level of their respective tanks but also the level of the tanks further upstream, since this is the direction in which the COLD tank water flows. Control valve C5 only affects the flow out of tank number 1.

4.1.2 : REDUCED ORDER BIM MATRIX

The rate of gold absorption onto carbon is assumed to be constant for a given CIP plant. Thus an attempt will be made to keep the heat transfer rate in a tank constant. This can be done by fixing the stirring rate. If this is done, then S1 to S4 can no longer be considered to be inputs.

As was explained in the previous section, T1 and T6 cannot be altered by any of the inputs. Thus they are both disregarded as controllable outputs. There is no on-line instrumentation available to measure the loading of gold on carbon. This concentration is mimicked by the COLD stream temperature. Thus the tank temperatures will also be discounted as controllable outputs.

The BIM matrix has now been reduced to one that has 6 inputs and 10 outputs.

	C1	C2	C3	C4	C5	C6
F1	X					
F2						X
L1				X	X	
L2			X	X		
L3		X	X			
L4	X	X				
T7	X	X	X	X	X	X
T8	X	X	X	X	X	X
T9	X	X	X	X	X	X
T0	X	X	X	X	X	X

Figure 4.1.2 : REDUCED ORDER BIM MATRIX

This reduced order model still shows more outputs than inputs. However the next section will use this reduced order model to decide on the final control strategy to be adopted.

4.1.3 : FINAL CONTROL STRUCTURE FOR THE HERIG

There are certain conditions that exist on an operational CIP plant that need to be considered when designing a controller for the HERIG.

4.1.3.1 : CONTROL OF THE FLOWRATES

A CIP plant operated in continuous mode has a fixed flowrate of carbon. The plants also run on a fixed tonnage of slurry per day. This means that the HERIG must also be run with both the HOT and the COLD streams of water being pumped at a constant flowrate. Thus valves C1 and C6 can be used to control flowrates F1 and F2 respectively.

$$\begin{bmatrix} F1 \\ F2 \end{bmatrix} = \begin{bmatrix} G_{F11} & 0 \\ 0 & G_{F22} \end{bmatrix} * \begin{bmatrix} C1 \\ C6 \end{bmatrix}$$

Figure 4.1.3 : OPEN LOOP FLOW CONTROL

Since there is no interaction, two single variable designs can be performed on each loop independently. The step tests performed to obtain $G_F(s)$ will be presented in section 4.2, as will the design of a suitable closed loop controller.

4.1.3.2 : CONTROL OF THE LEVELS

The pipe temperature will be controlled using the level of COLD water in each tank. The levels need to be under closed loop control before level setpoints can be used to control temperatures. They will be controlled using valves C2 to C5.

$$\begin{bmatrix} L1 \\ L2 \\ L3 \\ L4 \end{bmatrix} = \begin{bmatrix} G_{L11} & G_{L12} & 0 & 0 \\ 0 & G_{L22} & G_{L23} & 0 \\ 0 & 0 & G_{L33} & G_{L34} \\ 0 & 0 & 0 & G_{L44} \end{bmatrix} * \begin{bmatrix} C5 \\ C4 \\ C3 \\ C2 \end{bmatrix}$$

Figure 4.1.4 : OPEN LOOP LEVEL CONTROL

This transfer function matrix has an upper diagonal structure. This indicates that there is no multivariable problem and single variable designs can be carried out on each loop independently. The interaction can be eliminated by designing a suitable pre-compensator that eliminates the upper diagonal elements. This will all be presented in section in section 4.3.

4.1.3.3 : CONTROL OF THE PIPE TEMPERATURES

The level setpoints can now be used to control the pipe temperatures. This can be done since it is known from the work done by De Waal³ that the level responses will be much faster than the temperature responses.

$$\begin{bmatrix} T7 \\ T8 \\ T9 \\ T8 \end{bmatrix} = \begin{bmatrix} G_{T11} & G_{T12} & G_{T13} & G_{T14} \\ G_{T21} & G_{T22} & G_{T23} & G_{T24} \\ G_{T31} & G_{T32} & G_{T33} & G_{T34} \\ G_{T41} & G_{T42} & G_{T43} & G_{T44} \end{bmatrix} * \begin{bmatrix} L1 \\ L2 \\ L3 \\ L4 \end{bmatrix}$$

Figure 4.1.5 : OPEN LOOP TEMPERATURE CONTROL

This matrix is of full order. It poses a challenging multivariable problem. The design of a suitable controller for such a system will be discussed in the chapters that follow.

4.1.4 : IMPLEMENTATION OF THE DESIRED CONTROL SCHEME

A Turbo Pascal software package was written to run the HERIG from a personal computer. The package displays all the current values of the outputs as well as the current settings for the inputs. Data can be logged and displayed graphically. The package can run with the flows, levels and temperatures in either closed or open loop. For closed loop control, the controllers are implemented in state space form. A fourth order Runge-Kutta routine implements the controller numerically. A listing of the software package is presented in Appendix D.

In order to design a controller a model needs to be found for all transfer functions. Data from step tests performed on the rig were stored on disk. A software programme was written in Turbo Pascal which took as its input the data from such a step test. The programme then offers the user a choice of transfer functions to fit to the data. The user can alter all parameters of a given transfer function, until they are satisfied that the function approximates the data adequately.

The package is merely a tool to implement various trial and error approximations. It was not intended to be a fully fledged system identification software package. A listing of the code written is given in Appendix E.

4.2 : CLOSED LOOP FLOW CONTROL

The flowrates in both the Pipes and the Tanks are kept constant during the running of the rig. A closed loop controller will be designed to do this, based on a model for the flowrates obtained from step tests. The open loop flow step test results are :

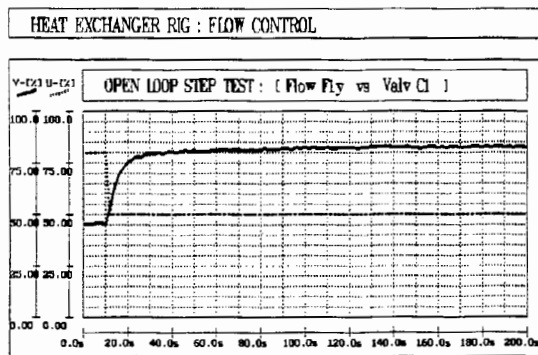


Figure 4.2.1 : FLOW G11(s)

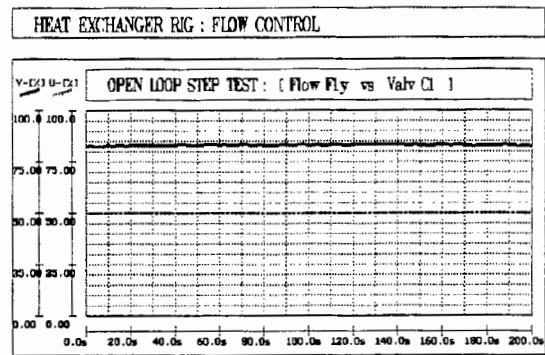


Figure 4.2.3 : FLOW G12(s)

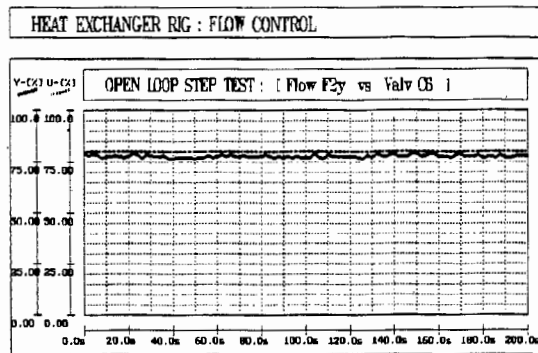


Figure 4.2.2 : FLOW G21(s)

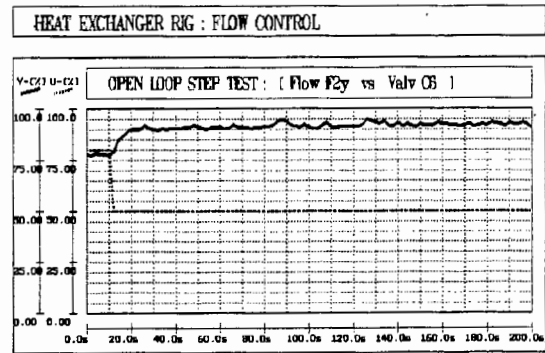


Figure 4.2.4 : FLOW G22(s)

These step tests confirmed the expected diagonal structure of $G_F(s)$. The diagonal elements are easily identified as first order functions. The open loop model is :

$$G_F(s) = \begin{bmatrix} -1.24 \frac{0.15}{s+0.15} & 0 \\ 0 & -0.50 \frac{0.15}{s+0.15} \end{bmatrix}$$

Figure 4.2.5 : OPEN LOOP FLOW MODEL

4.2.1 : CLOSED LOOP FLOW CONTROLLER

This open loop model is diagonally dominant and thus two PI controllers can be designed independently for each loop. Further there will be no need to design a pre-compensator since there is no interaction. The controller was chosen to give a closed loop response that is as fast as the open loop model, without using excessive control action.. The closed loop flow controller designed is :

$$K_F(s) = \left[\begin{array}{c|c} -\frac{s+0.2}{s} & 0 \\ \hline 0 & -\frac{s+0.2}{s} \end{array} \right]$$

Figure 4.2.6 : CLOSED LOOP FLOW CONTROLLER

4.2.2 : FILTER FOR FLOW CONTROL

When implemented on the rig, a filter, $F_F(s)$ was inserted in the feedback path of the closed loop system. The breakpoint frequency of the filter was chosen to be at least ten times faster than the open loop model $G_F(s)$. The filter is intended to eliminate high frequency signal noise.

$$F_F(s) = \left[\begin{array}{c|c} \frac{2.0}{s+2.0} & 0 \\ \hline 0 & \frac{2.0}{s+2.0} \end{array} \right]$$

Figure 4.2.7 : FILTER FOR FLOW CONTROL

4.2.3 : CLOSED LOOP CONTROL STRUCTURE

The HERIG flowrates under closed loop control from a PC has the following structure :

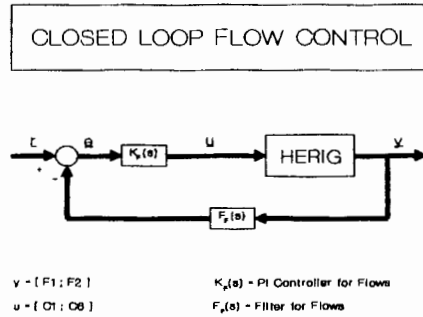


Figure 4.2.8 : CLOSED LOOP FLOW CONTROL

The results of closed loop flow step tests are :

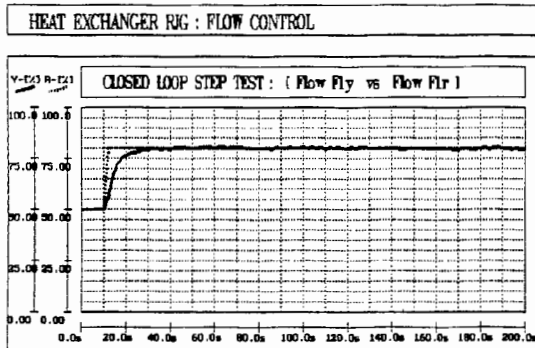


Figure 4.2.9 : FLOW H11(s)

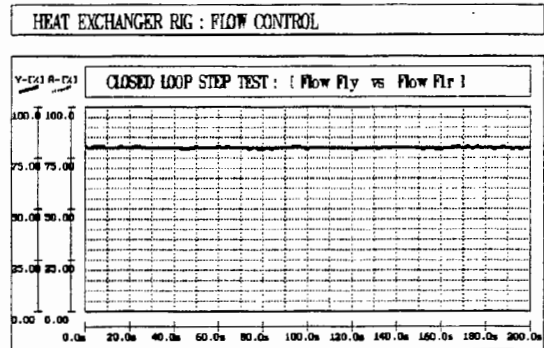


Figure 4.2.11 : FLOW H12(s)

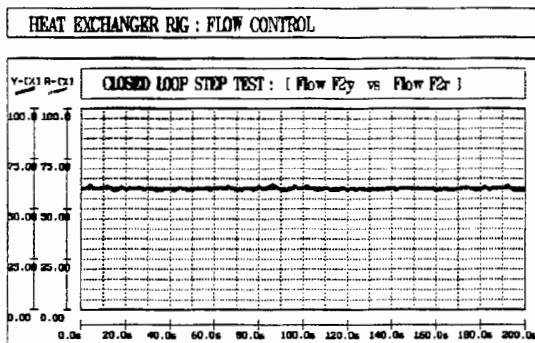


Figure 4.2.10 : FLOW H21(s)

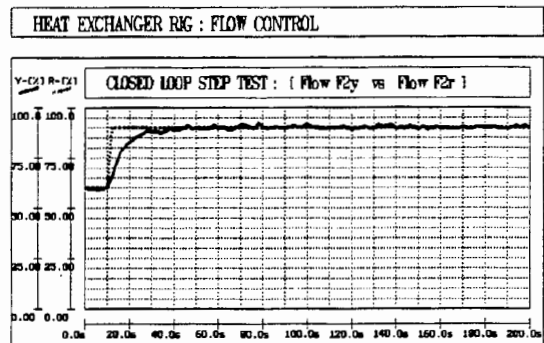


Figure 4.2.12 : FLOW H22(s)

These graphs show that the HERIG flows track their setpoints and do not show any interaction between the loops; as was expected. The input action required to achieve these responses are :

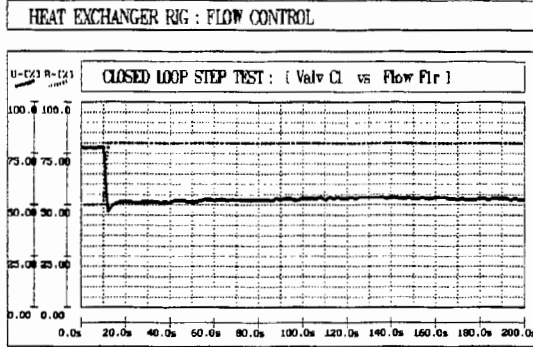


Figure 4.2.13 : FLOW U11(s)

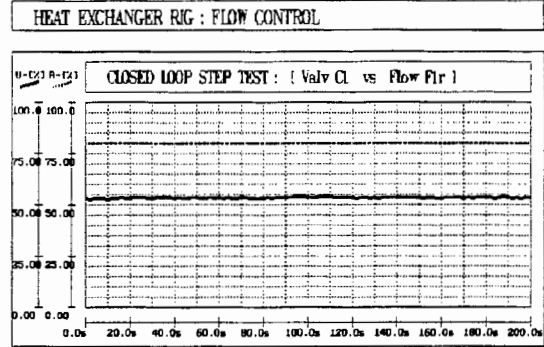


Figure 4.2.15 : FLOW U12(s)

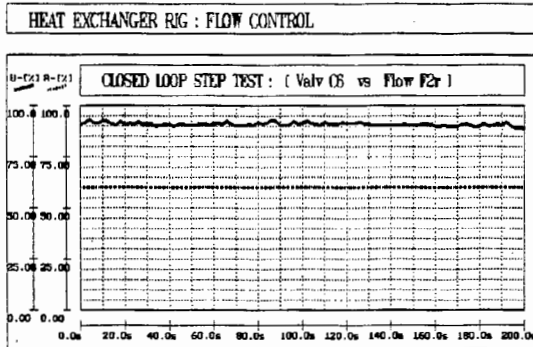


Figure 4.2.14 : FLOW U21(s)

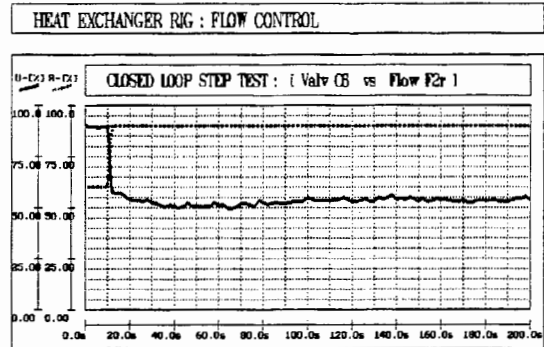


Figure 4.2.16 : FLOW U22(s)

4.3 : CLOSED LOOP LEVEL CONTROL

The structural study shows that the open loop model should be upper diagonal. The tanks are small in volume, 22 liters, and it is expected that they will have the characteristics of an integrator. The open loop level step test results are shown graphically in the following two pages.

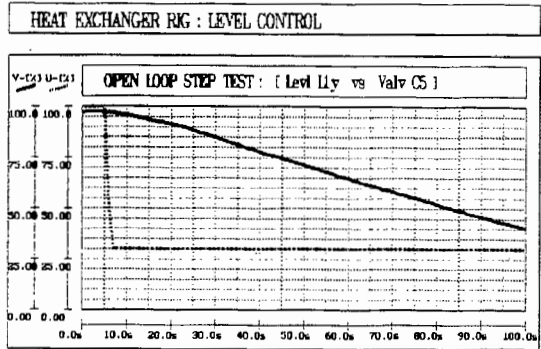


Figure 4.3.1 : LEVEL G11(s)

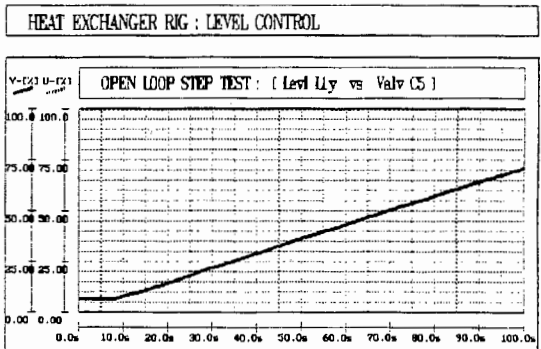


Figure 4.3.5 : LEVEL G12(s)

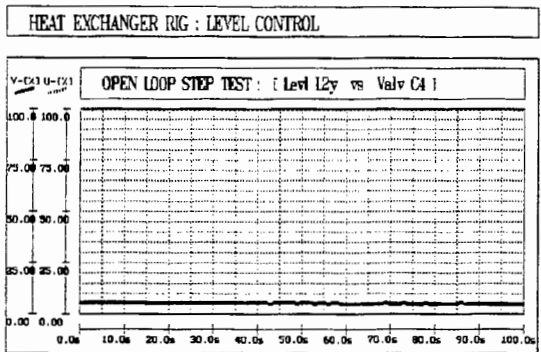


Figure 4.3.2 : LEVEL G21(s)

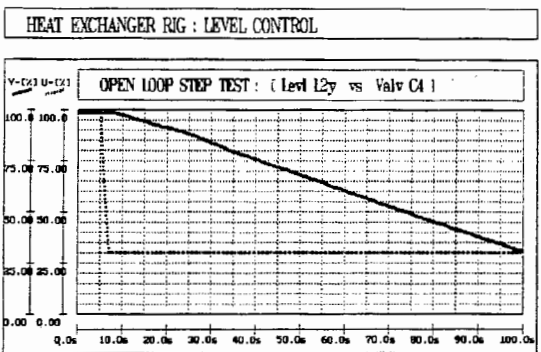


Figure 4.3.6 : LEVEL G22(s)

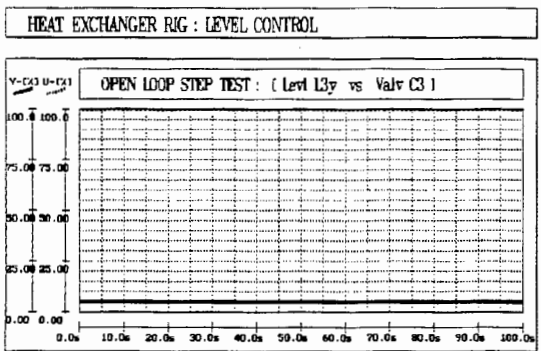


Figure 4.3.3 : LEVEL G31(s)

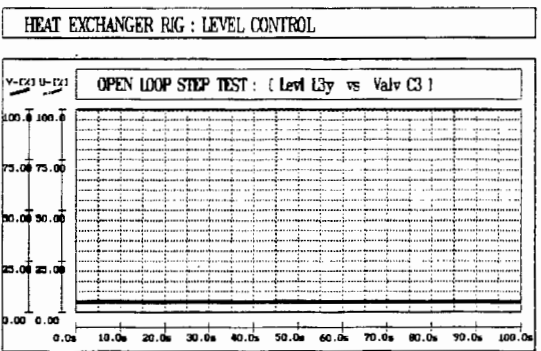


Figure 4.3.7 : LEVEL G32(s)

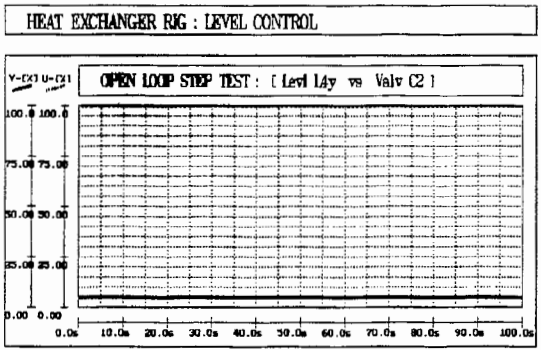


Figure 4.3.4 : LEVEL G41(s)

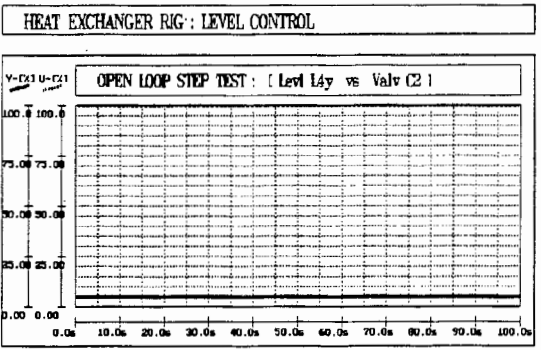


Figure 4.3.8 : LEVEL G42(s)

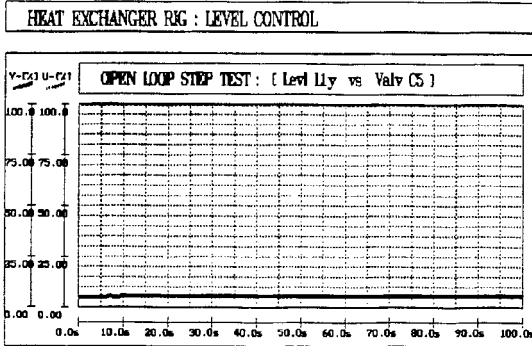


Figure 4.3.9 : LEVEL G13(s)

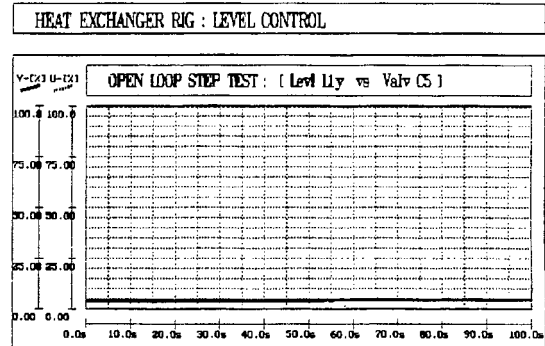


Figure 4.3.13: LEVEL G14(s)

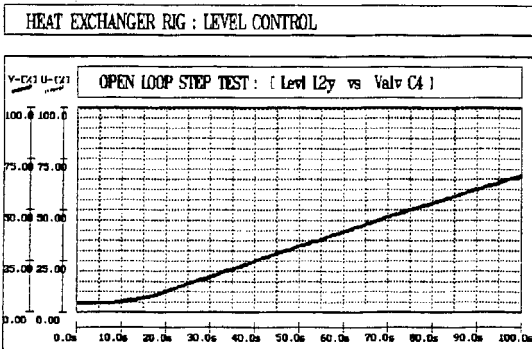


Figure 4.3.10: LEVEL G23(s)

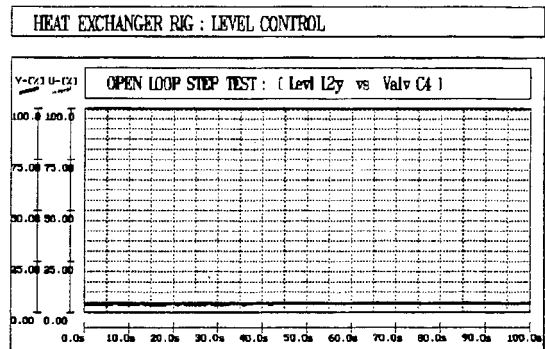


Figure 4.3.14: LEVEL G24(s)

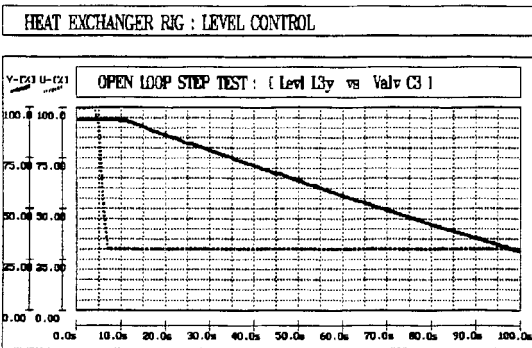


Figure 4.3.11: LEVEL G33(s)

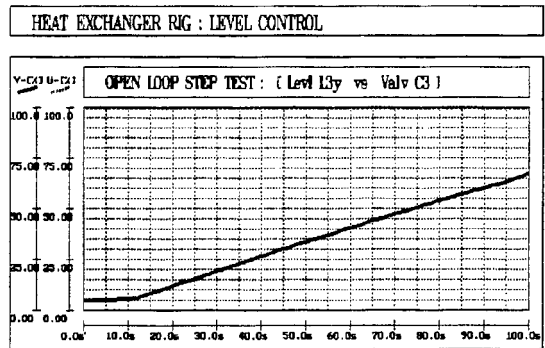


Figure 4.3.15: LEVEL G34(s)

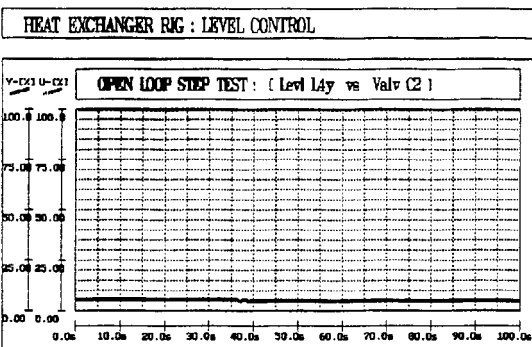


Figure 4.3.12: LEVEL G43(s)

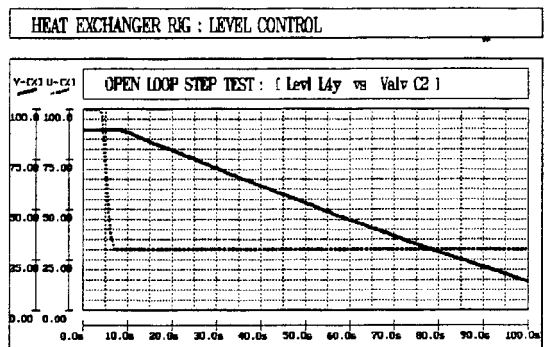


Figure 4.3.16: LEVEL G44(s)

The step tests confirm the expected structure of $G_L(s)$. The elements of the model are all integrators, the gains of which can be found using the slopes of the graphs and the step sizes. The dead times seen in the graphs are small and have been ignored. The open loop model is :

$$G_L(s) = \begin{bmatrix} \frac{0.009}{s} & \frac{-0.01}{s} & 0 & 0 \\ 0 & \frac{0.0105}{s} & \frac{-0.01}{s} & 0 \\ 0 & 0 & \frac{0.0105}{s} & \frac{-0.01}{s} \\ 0 & 0 & 0 & \frac{0.012}{s} \end{bmatrix}$$

Figure 4.3.17 : OPEN LOOP LEVEL MODEL

4.3.1 : PRE-COMPENSATOR FOR LEVEL CONTROL

Since the open loop model is upper diagonal and all elements in the matrix are integrators, a pre-compensator can be designed to diagonalise the open loop level model.

$$K_C(s) = \begin{bmatrix} 1 & 1.1111 & 1.0582 & 1.0078 \\ 0 & 1 & 0.9524 & 0.9070 \\ 0 & 0 & 1 & 0.9524 \\ 0 & 0 & 0 & 1 \end{bmatrix}$$

Figure 4.3.18 : LEVEL PRECOMPENSATOR MATRIX

4.3.2 : CLOSED LOOP LEVEL CONTROLLER

The pre-compensator $K_C(s)$ diagonalises the model $G_L(s)$. Thus a diagonal PI control matrix can be designed to achieve setpoint tracking for all of the loops. The controller was chosen to give as fast a response as was possible, without making the valves too sensitive to level fluctuations.

$$K_L(s) = \begin{bmatrix} 20 \frac{1+2000s}{2000s} & 0 & 0 & 0 \\ 0 & 20 \frac{1+2000s}{2000s} & 0 & 0 \\ 0 & 0 & 20 \frac{1+2000s}{2000s} & 0 \\ 0 & 0 & 0 & 20 \frac{1+2000s}{2000s} \end{bmatrix}$$

Figure 4.3.19 : CLOSED LOOP FLOW CONTROLLER

This controller needed to make the loop very fast since level setpoints will be used to control temperatures. Thus the level control will be an inner loop with the temperature control being a slower outer loop.

4.3.3 : FILTER FOR FLOW CONTROL

As with the flow control a filter had to be designed to filter out any noise on the level readings. This filter also serves to filter any splashing that might be caused by the stirrers.

$$F_L(s) = \begin{bmatrix} \frac{1.0}{s+1.0} & 0 & 0 & 0 \\ 0 & \frac{1.0}{s+1.0} & 0 & 0 \\ 0 & 0 & \frac{1.0}{s+1.0} & 0 \\ 0 & 0 & 0 & \frac{1.0}{s+1.0} \end{bmatrix}$$

Figure 4.3.20 : FILTER FOR FLOW CONTROL

4.3.4 : CLOSED LOOP CONTROL STRUCTURE

The HERIG flowrates under closed loop control from a PC has the following structure :

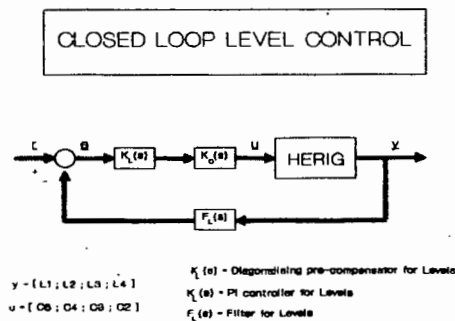


Figure 4.3.21 : CLOSED LOOP FLOW CONTROL

Closed loop level step tests were performed on the HERIG. The level response, as well as the control action used, are displayed in the graphs that follow.

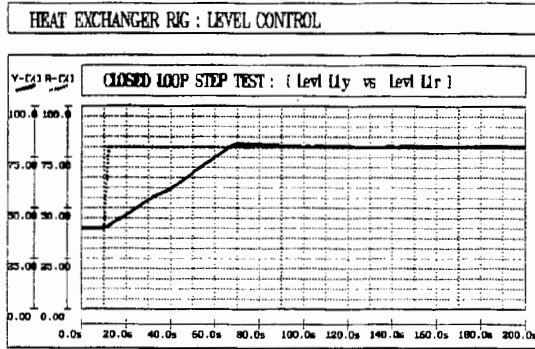


Figure 4.3.22: LEVEL H11(s)

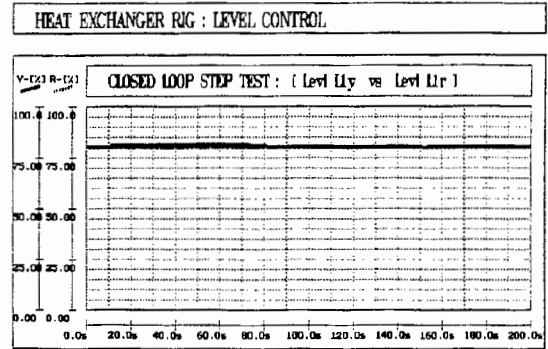


Figure 4.3.26: LEVEL H12(s)

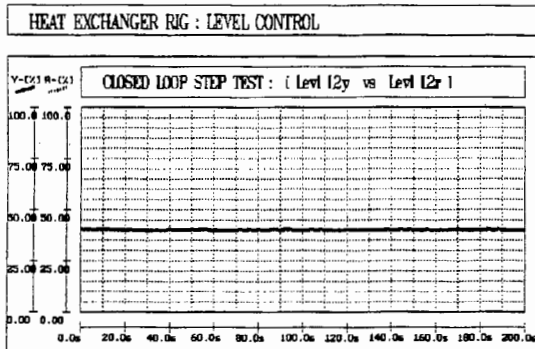


Figure 4.3.23: LEVEL H21(s)

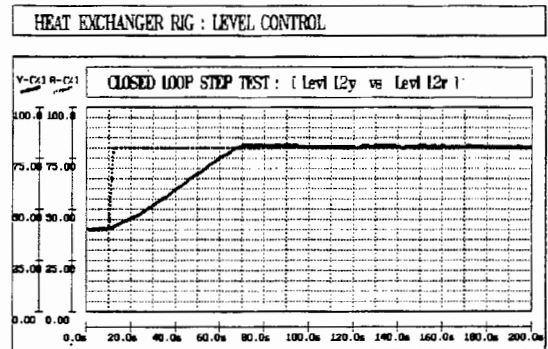


Figure 4.3.27: LEVEL H22(s)

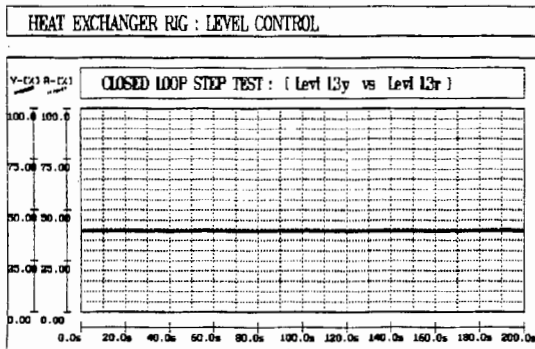


Figure 4.3.24: LEVEL H31(s)

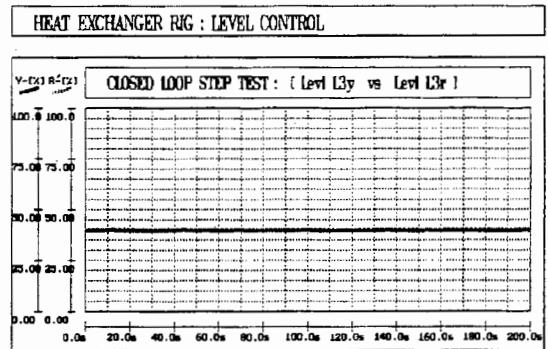


Figure 4.3.28: LEVEL H32(s)

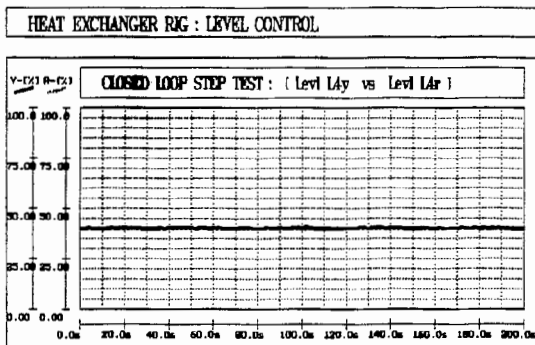


Figure 4.3.25: LEVEL H41(s)

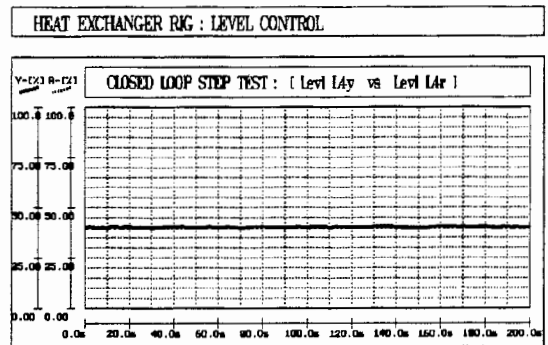


Figure 4.3.29: LEVEL H42(s)

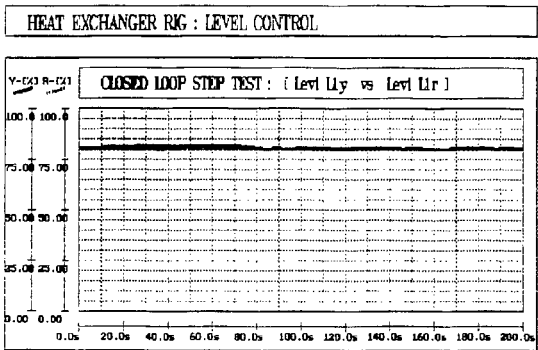


Figure 4.3.30: LEVEL H13(s)

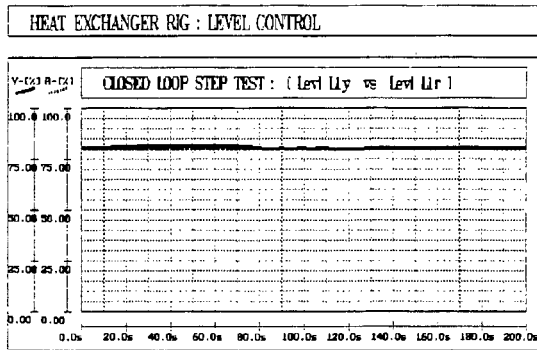


Figure 4.3.34: LEVEL H14(s)

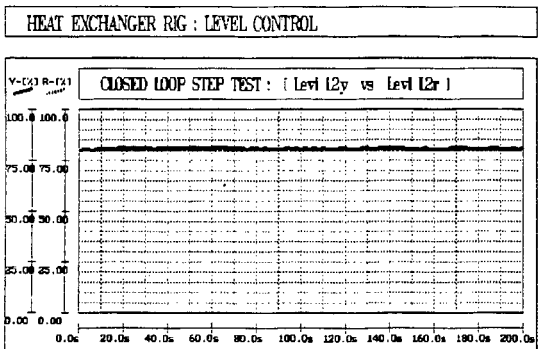


Figure 4.3.31: LEVEL H23(s)

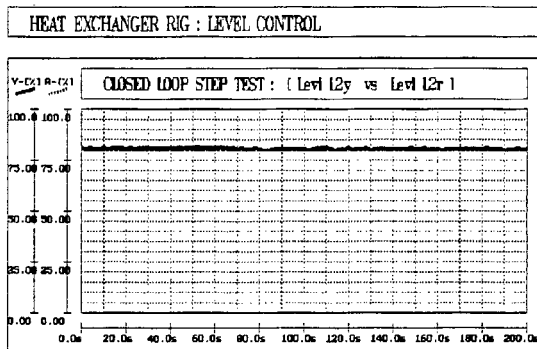


Figure 4.3.35: LEVEL H24(s)

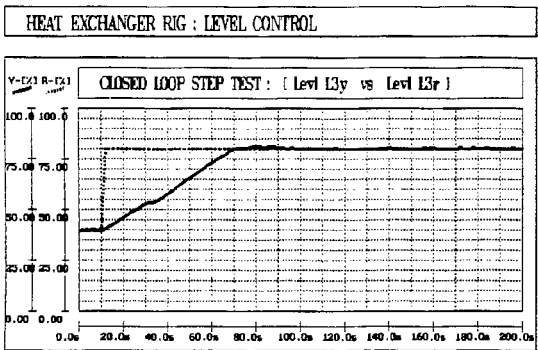


Figure 4.3.32: LEVEL H33(s)

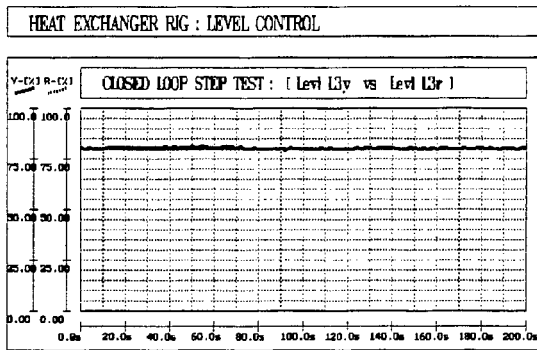


Figure 4.3.36: LEVEL H34(s)

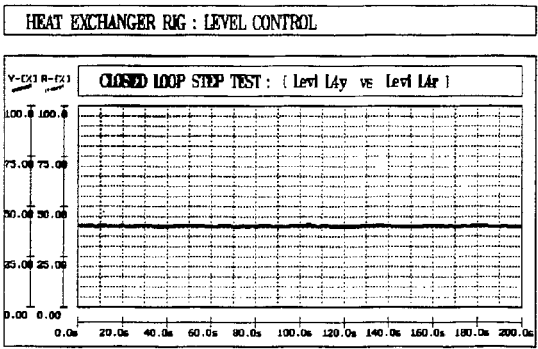


Figure 4.3.33: LEVEL H43(s)

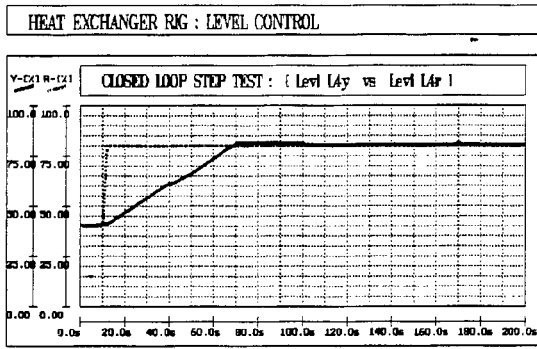


Figure 4.3.37: LEVEL H44(s)

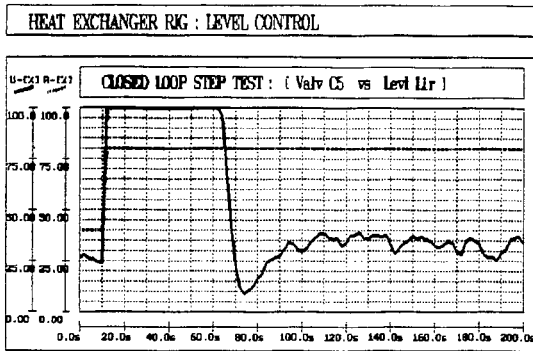


Figure 4.3.38: LEVEL U11(s)

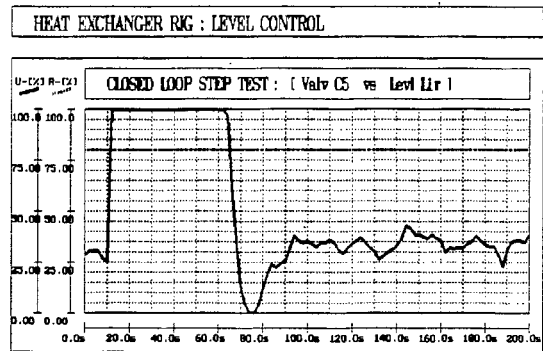


Figure 4.3.42: LEVEL U12(s)

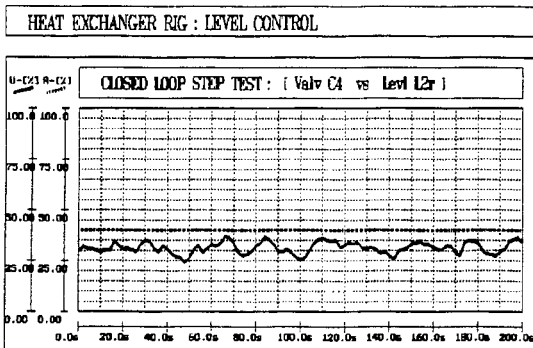


Figure 4.3.39: LEVEL U21(s)

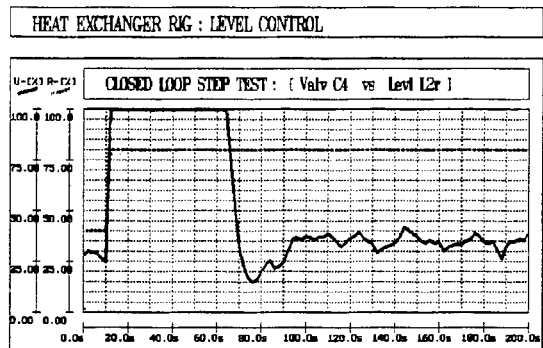


Figure 4.3.43: LEVEL U22(s)

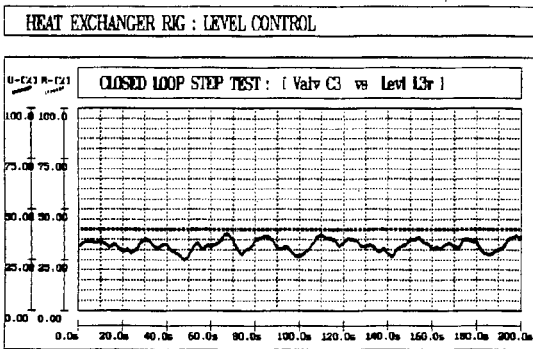


Figure 4.3.40: LEVEL U31(s)

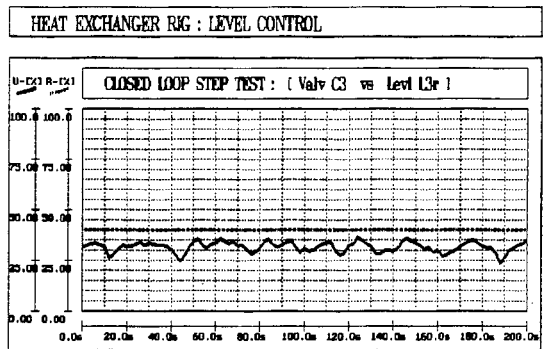


Figure 4.3.44: LEVEL U32(s)

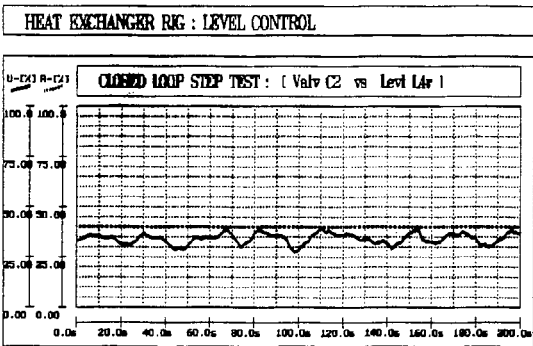


Figure 4.3.41: LEVEL U41(s)

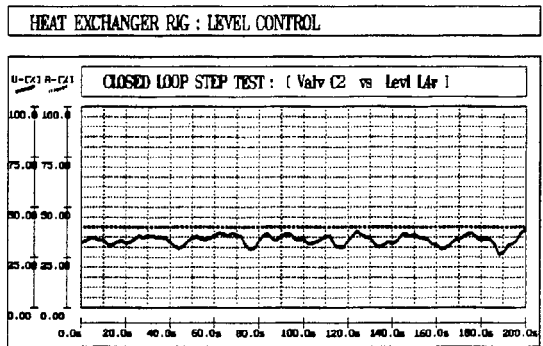


Figure 4.3.45: LEVEL U42(s)

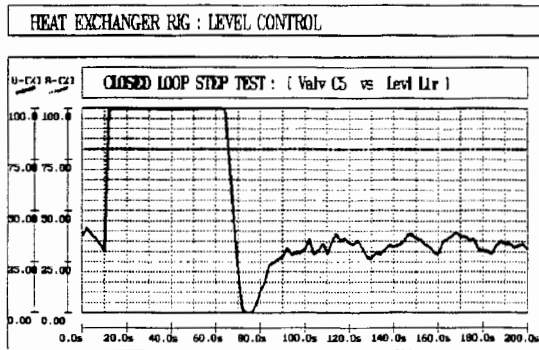


Figure 4.3.46: LEVEL U13(s)

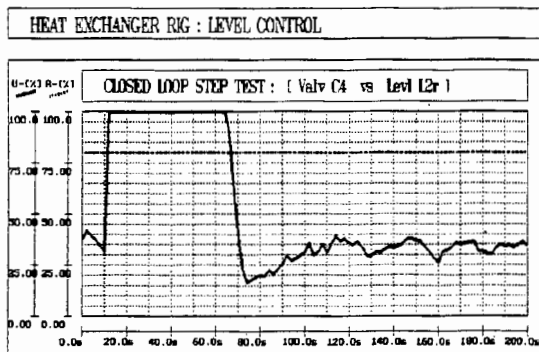


Figure 4.3.47: LEVEL U23(s)

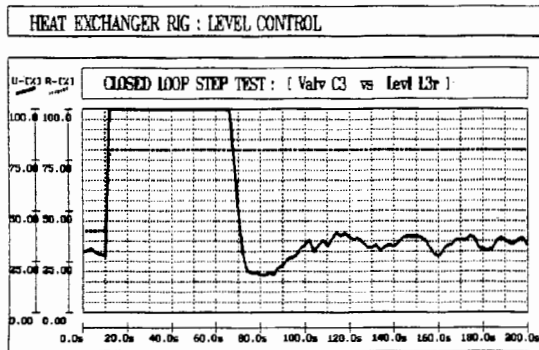


Figure 4.3.48: LEVEL U33(s)

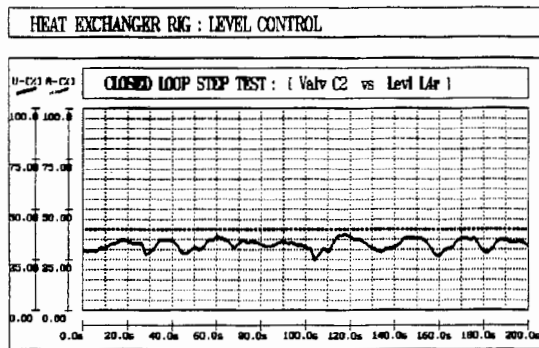


Figure 4.3.49: LEVEL U43(s)

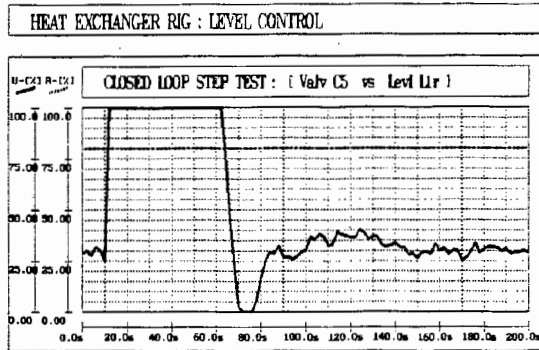


Figure 4.3.50: LEVEL U14(s)

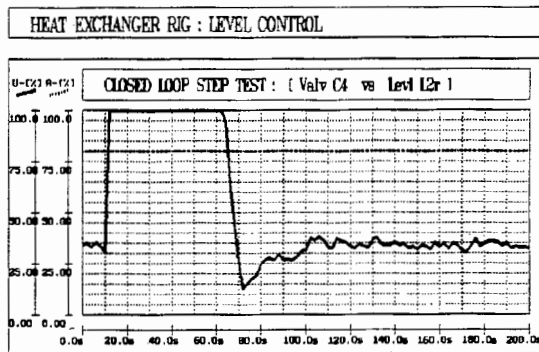


Figure 4.3.51: LEVEL U24(s)

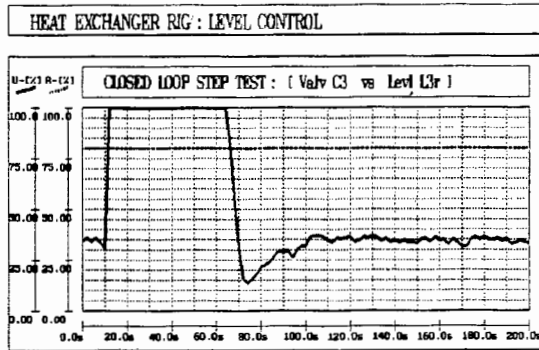


Figure 4.3.52: LEVEL U34(s)

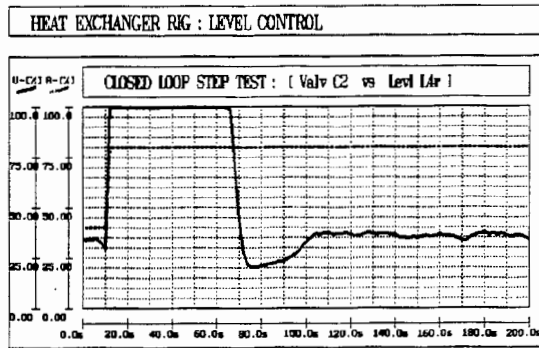


Figure 4.3.53: LEVEL U44(s)

The graphs presented so far have shown that closed loop flow control can be achieved without excessive input action. the level control appears to be like bang-bang control. The speed of the level control needs to be as fast as possible if the levels are going to be used in an inner loop to control temperatures.

4.4 : OPEN LOOP TEMPERATURE RESPONSES

An open loop model needs to be found for the HERIG temperature response, $G_T(s)$. The rig is first brought under level and flow control and then only are step tests done to extract the necessary transfer functions.

The step tests will be performed under the same conditions to those used for the simulator. The flowrates will be the same, as will the reservoir temperatures. The HERIG was allowed to reach a steady state with the all the levels being at 20%. Then as was done on the simulator, four successive step were done, each a raise of 60% in the level. The time period for data logging was kept the same, so that a comparison of the settling times between the actual rig and the simulator could be done.

One technical problem encountered on the rig was a drift in the temperature of the HOT reservoir. The HERIG was designed such that the heated COLD water leaving the first tank is recycled into the HOT reservoir. This results in the reservoir heating up by a few degrees and thus causing the temperature of the reservoir to drift slightly above the temperature to which the controller is set.

This drift was most noticeable in the first step test done, where all four pipe temperatures show an upward drift due to the slightly increasing reservoir temperature. The rise is not more than one to two degrees and did not interfere in later step tests.

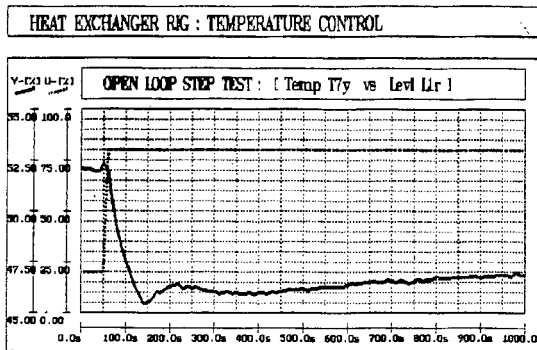


Figure 4.4.1 : TEMP G11(s)

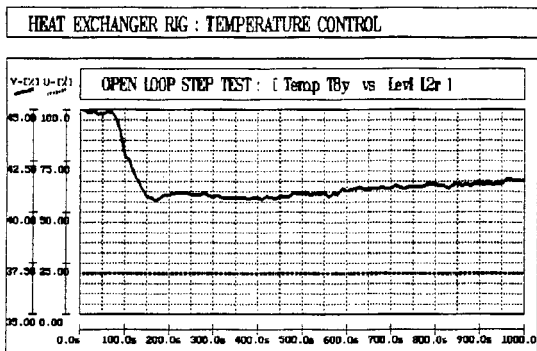


Figure 4.4.2 : TEMP G21(s)

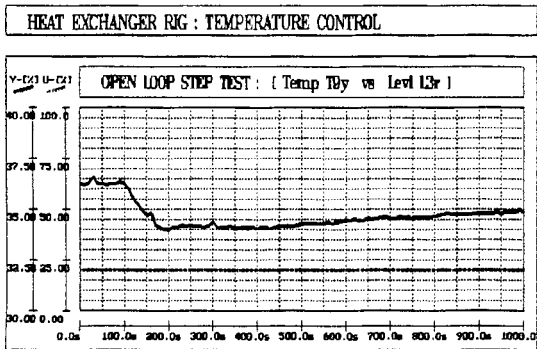


Figure 4.4.3 : TEMP G31(s)

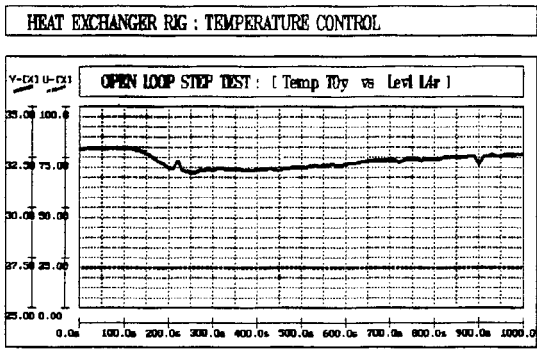


Figure 4.4.4 : TEMP G41(s)

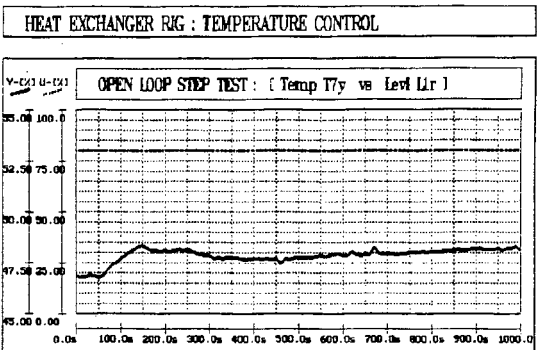


Figure 4.4.5 : TEMP G12(s)

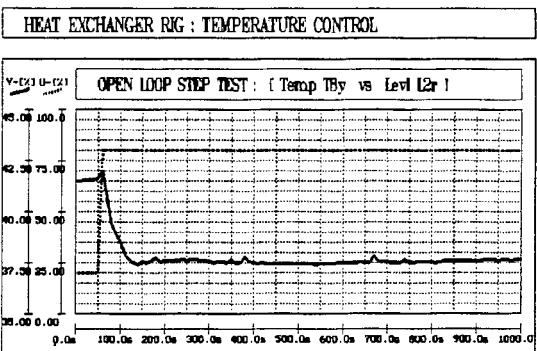


Figure 4.4.6 : TEMP G22(s)

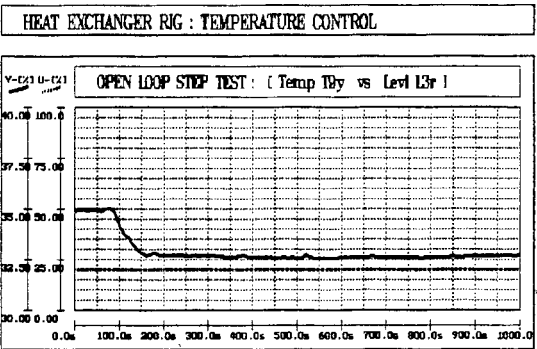


Figure 4.4.7 : TEMP G32(s)

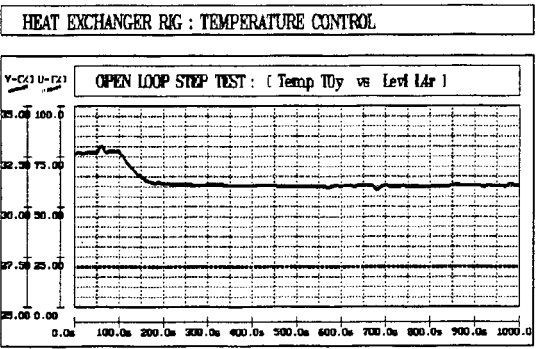


Figure 4.4.8 : TEMP G42(s)

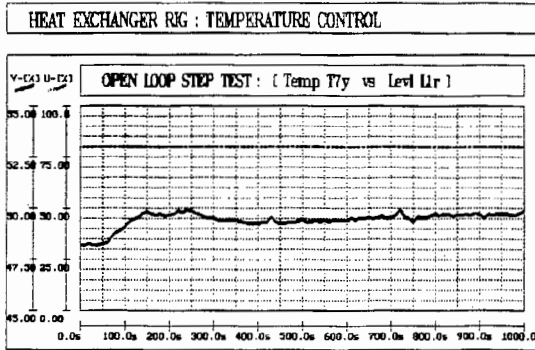


Figure 4.4.9 : TEMP G13(s)

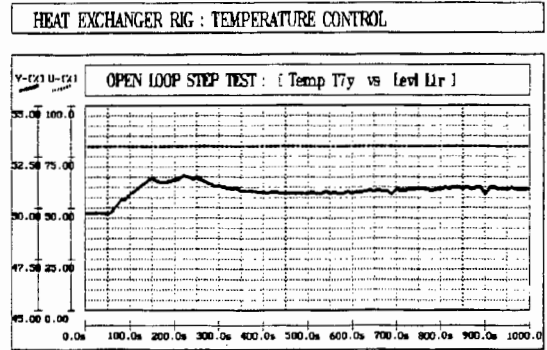


Figure 4.4.13 : TEMP G14(s)

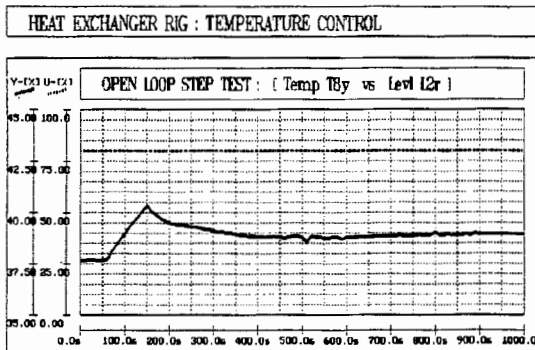


Figure 4.4.10 : TEMP G23(s)

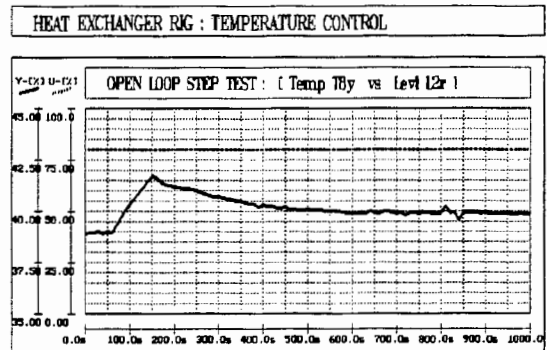


Figure 4.4.14 : TEMP G24(s)

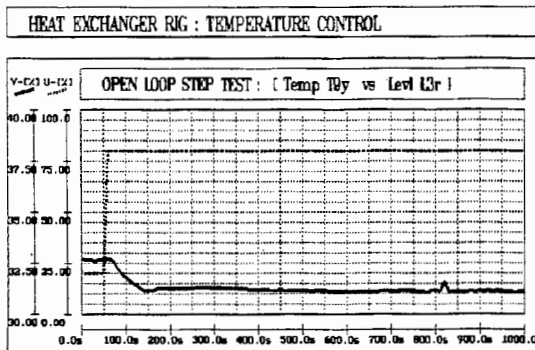


Figure 4.4.11 : TEMP G33(s)

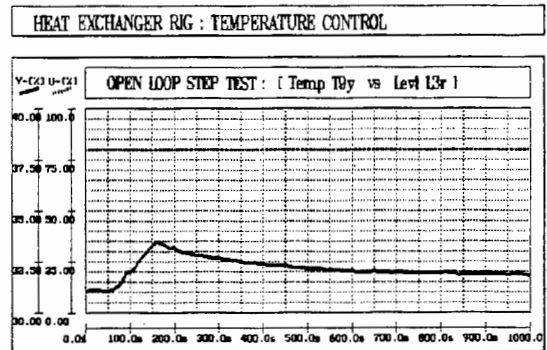


Figure 4.4.15 : TEMP G34(s)

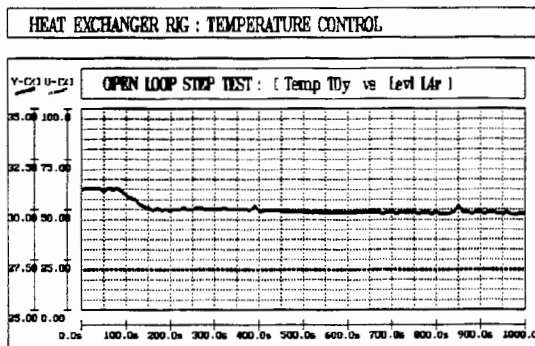


Figure 4.4.12 : TEMP G43(s)

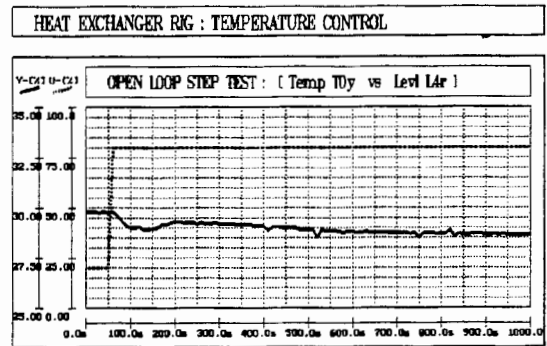


Figure 4.4.16 : TEMP G44(s)

Transfer functions were fitted to the open loop data, using a software package written for this purpose. A listing of the code written is given in Appendix E. Plots of all the functions fitted are presented in Appendix F. They show the data plotted against the estimated transfer function. The functions that were fitted to the response from the first step test, G_{T11} to G_{T41} , ignored the drift that was mentioned previously.

Transfer functions from the first step :

$$G_{T11}(s) = -0.1 \left[\frac{0.02}{s + 0.02} \right]$$

$$G_{T21}(s) = -0.07 \left[\frac{0.02}{s + 0.02} \right] e^{-20s}$$

$$G_{T31}(s) = -0.035 \left[\frac{0.02}{s + 0.02} \right] e^{-50s}$$

$$G_{T41}(s) = -0.02 \left[\frac{0.02}{s + 0.02} \right] e^{-80s}$$

Transfer functions from the second step :

$$G_{T12}(s) = 0.02 \left[\frac{0.12}{5} \frac{(s+0.005)}{(s+0.012)(s+0.01)} \right]$$

$$G_{T22}(s) = -0.065 \left[\frac{0.03}{s + 0.03} \right]$$

$$G_{T32}(s) = -0.04 \left[\frac{0.03}{s + 0.03} \right] e^{-40s}$$

$$G_{T42}(s) = -0.03 \left[\frac{0.02}{s + 0.02} \right] e^{-50s}$$

Transfer functions from the third step :

$$G_{T13}(s) = 0.025 \left[\frac{0.12}{5} \frac{(s+0.005)}{(s+0.012)(s+0.01)} \right]$$

$$G_{T23}(s) = 0.02 \left[\frac{0.12}{2} \frac{(s+0.002)}{(s+0.012)(s+0.01)} \right]$$

$$G_{T33}(s) = -0.025 \left[\frac{0.02}{s + 0.02} \right]$$

$$G_{T43}(s) = -0.02 \left[\frac{0.02}{s + 0.02} \right] e^{-40s}$$

Transfer functions from the fourth step :

$$G_{T14}(s) = 0.02 \left[\frac{0.12}{4} \frac{(s+0.004)}{(s+0.012)(s+0.01)} \right]$$

$$G_{T24}(s) = 0.015 \left[\frac{0.12}{1.5} \frac{(s+0.0015)}{(s+0.012)(s+0.01)} \right]$$

$$G_{T34}(s) = 0.012 \left[\frac{0.12}{1.5} \frac{(s+0.0015)}{(s+0.012)(s+0.01)} \right]$$

$$G_{T44}(s) = -0.02 \left[\frac{0.003}{s + 0.003} \right]$$

4.5 : COMPARISON BETWEEN THE MODEL AND THE HERIG

In Chapter 3 a model for the HERIG was developed. This model was derived entirely from thermal and mass balances. The heat transfer coefficients were calculated from equations and not measured on the rig. This section will examine how well this model compares to the actual data obtained from the rig, as presented in the form of the derived transfer function matrix $G_T(s)$.

4.5.1 : COMPARISON OF THE STEADY STATE GAINS

The steady state gains from the transfer functions of section 4.4 and the calculated steady state gains of the model, see section 3.5, are plotted on the same graphs.

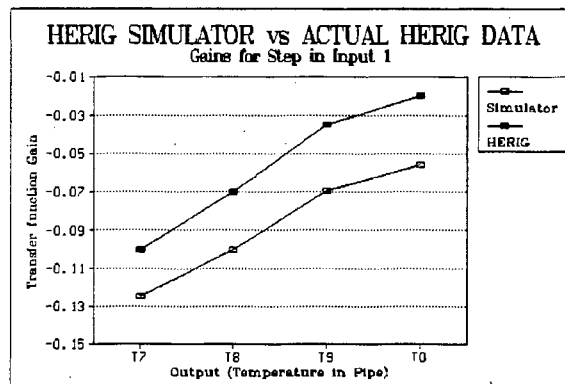


Figure 4.5.1 : GAINS FOR COLUMN 1

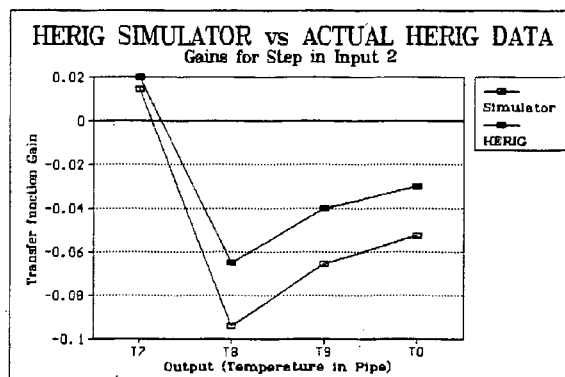


Figure 4.5.2 : GAINS FOR COLUMN 2

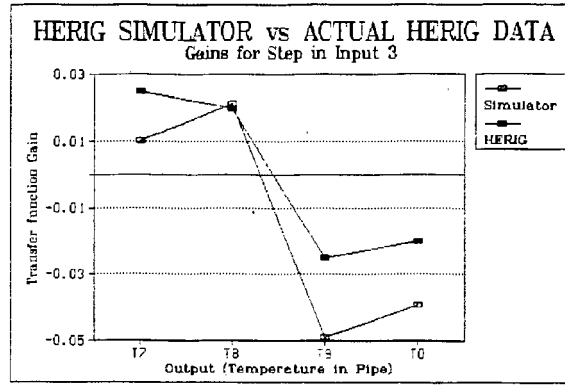


Figure 4.5.3 : GAINS FOR COLUMN 3

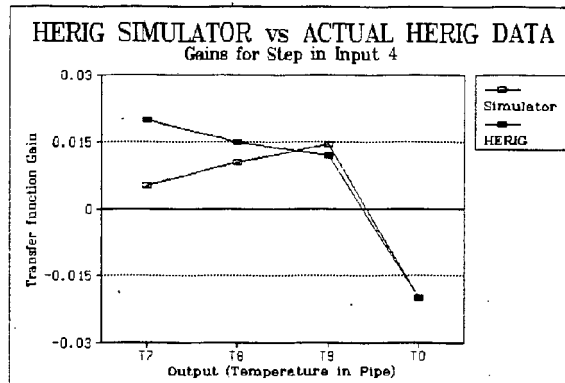


Figure 4.5.4 : GAINS FOR COLUMN 4

These tabulated gains indicate that the model describes the HERIG fairly well at steady state. The signs of all the corresponding functions are the same. The values for the diagonal elements correspond well, but the elements furthestmost from the diagonal are the worst. Most of the gains are less than one order of magnitude in difference, which is considered reasonable for controller design⁵.

The gains for the elements above the diagonal show different trends. The simulator indicated an expected decrease in gain for elements further from the diagonal. Step test data indicates that the gains in fact do the opposite. This was attributed to the drift in the reservoir temperature. The reservoir water is first pumped through Tank 1, and thus the gains for the first tanks are higher than expected.

4.5.2 : COMPARISON OF THE SPEED OF RESPONSE

A comparison will be made separately between the diagonal elements, the upper diagonals and the lower diagonals.

4.5.2.1 : DIAGONAL ELEMENTS

The settling times of the diagonal elements for the simulator go from 300s in Pipe 1 to 800s in Pipe 4 (see Figures 3.6.1 to 3.6.16). The HERIG data shows corresponding settling times ranging from 400s for Pipe 1 to a little more than 1000s for Pipe 4 (see Appendix F). The slower settling times for the rig is due to level response of the rig.

The simulator assumed an instantaneous level response, but the HERIG levels take around 80s to settle (see Figures 4.3.22 to 4.3.37). This level settling time could also account for the fact that the rig data did not show the slight overshoot that the simulator predicted.

4.5.2.2 : LOWER DIAGONAL ELEMENTS

Data from the HERIG temperature step tests show that all the lower diagonal responses are all first orders with some dead time. The simulated responses show first order responses, but with no dead time. The dead in the rig comes about as the level controller requires around 80s to settle. This means the temperature only begins to change noticeably once the level has settled.

Some of the model lower diagonal transfer functions exhibit an overshoot that is not observed in the rig data. This phenomena also occurred for the diagonal elements and the explanation is the same as is offered in the previous section.

4.5.2.3 : UPPER DIAGONAL ELEMENTS

The simulator predicted a second order response characterised by two real poles. This type of system does not have any overshoot. The HERIG results showed a distinct overshoot. The models fitted to these functions are characterised by two real poles and a dominant zero that causes the overshoot.

Thus the data for the upper diagonal elements have the expected pole characteristics, but exhibit an unexpected zero. The shapes of all these responses are the same though, as are the shapes of the simulated responses.

A model has been found for the temperature response of the HERIG to a change in water level. This model corresponds well to the model predicted by the simulation of the rig. The final chapters of this dissertation will deal with the design and implementation of a robust multivariable controller for the rig temperatures.

REFERENCES

1. BRAAE M., Lecture notes : Advanced Control Systems, 1990.

2. GEAR A.B.J., The Design of Decentralized Controllers for Large Scale Systems, M.Sc. Dissertation, Department of Electrical and Electronic Engineering, University of Cape Town, Feb 1988.
3. DE WAAL A., Performance Index Analysis of Control Systems Synthesised for a Counter-Current Process, M.Sc. Dissertation, Department of Electrical and Electronic Engineering, University of Cape Town, April 1990.
4. BRAAE M., Lecture notes : Control Engineering I, 1988.
5. BRAAE M., Private Communication.

CHAPTER 5 : H_{∞} OPTIMAL CONTROLLER DESIGN AND SYNTHESIS

The H_{∞} Optimal method of designing multivariable feedback controllers is a relatively new approach¹. It is a frequency domain based method of robust, uncertainty tolerant, controller design. The method uses singular value bode plots to evaluate the multivariable feedback system performance².

This chapter will present the definition of the H_{∞} norm of a transfer function matrix and explain its meaning. An algorithm will then be presented in which the synthesis and design phases of obtaining a controller will be presented. This chapter attempts to provide a guideline for anyone wishing to use H_{∞} as a design method.

5.1 : DEFINITION OF THE H_{∞} OPERATOR NORM

In single variable systems the closed loop performance is determined by the variation of the loop gain with frequency, assuming that closed loop stability is achieved. For multivariable systems, the problem arises that a matrix does not have a unique gain. Multivariable systems can be described by their singular values, or principle gains³.

5.1.1 : SINGULAR VALUES

For a multivariable system the norm $\|G(s)u(s)\|$ depends on the direction of the vector $u(s)$. Using the notation that $\|x\|$ denotes any vector norm, an INDUCED MATRIX NORM is defined by,

$$\|G\| = \sup_{x \neq 0} \frac{\|Gx\|}{\|x\|} \quad (5.1)$$

The vector norm can be described by the Euclidean vector norm,

$$\|x\| = \sqrt{x^H x} \quad (5.2)$$

This is calculated as the positive square root of the complex conjugate transpose of x , denoted x^H , multiplied with the vector x . The induced matrix norm then becomes the HILBERT or SPECTRAL NORM,

$$\|G\|_S = \bar{\sigma} \quad (5.3)$$

This Hilbert norm is calculated as the maximum eigenvalue of $G^H G$. For a square $n \times n$ matrix G , n positive square roots of the eigenvalues of $G^H G$ can be calculated. They are called *SINGULAR VALUES*. The maximum singular value is then the Hilbert norm.

5.1.2 : PRINCIPLE GAINS

In section 5.1.1 the singular values were calculated for a constant matrix G . Instead in practice we have a frequency dependent matrix $G(s)$ and the singular values now depend on frequency. They are then called *PRINCIPLE GAINS*. The Hilbert norm is now frequency dependent,

$$\|G(j\omega)\|_S = \bar{\sigma}(j\omega) \quad (5.3)$$

The principle gains can be ordered as $\sigma_1 \geq \sigma_2 \geq \dots \geq \sigma_N$, then the maximum and minimum principle gains can be denoted as $\bar{\sigma}$ and $\underline{\sigma}$ respectively.

This measure of transfer function 'gain', the principle gain, is evaluated at individual frequencies. It is useful to have a cruder measure of a single number to describe the 'gain' of a transfer function matrix. One such measure is the H_∞ norm⁴,

$$\|G\|_\infty = \sup_w \bar{\sigma}(G(j\omega)) \quad (5.4)$$

This norm, being a single number, is calculated as the maximum value that the maximum principle gain of a matrix $G(j\omega)$ reaches.

The usefulness and meaning of the H_∞ norm is shown in a theorem proved by Vidyasagar⁵. Suppose an input signal $u(t)$ has finite energy, it is useful to know what is the maximum increase in energy that can occur between the input and the output for a given system. Vidyasagar has proved that this maximum amplification is in fact the H_∞ norm as described in equation 5.4 above.

5.2 : ALGORITHM FOR THE DESIGN OF FEEDBACK CONTROLLERS

When designing a controller for a given process, a closed loop design is preferred to an open loop design since it has the advantages of robustness in the face of process variations, rejection of external disturbances and the ability to stabilise unstable processes⁶.

All these advantages can be translated into objectives that can be specified, for example a minimum attenuation required for a disturbance at a particular frequency. The choice of these objectives form the design section of the control problem.

The synthesis of a controller that satisfies the design objectives is a problem that has been solved by Doyle⁷. An algorithm for obtaining such a controller is presented in the following flowchart.

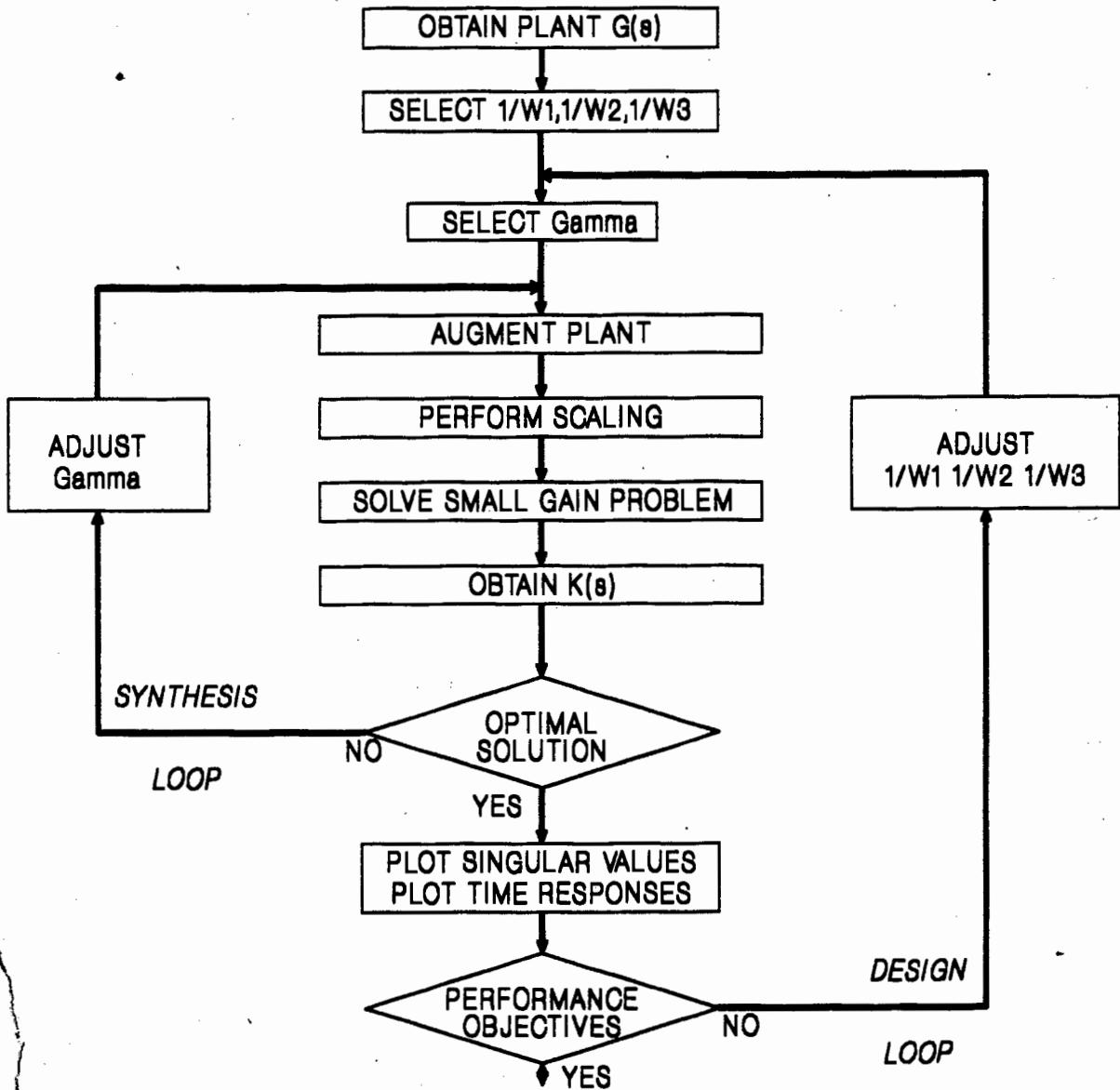


Figure 5.1 : FLOWCHART FOR H_{∞} BASED CONTROLLER DESIGN

The synthesis technique developed by Doyle is called Γ -iteration. The plant and the desired objectives, specified as weighting functions, are collected together to form a nominal plant $P(s)$. Together with the controller $K(s)$, it forms a feedback loop.

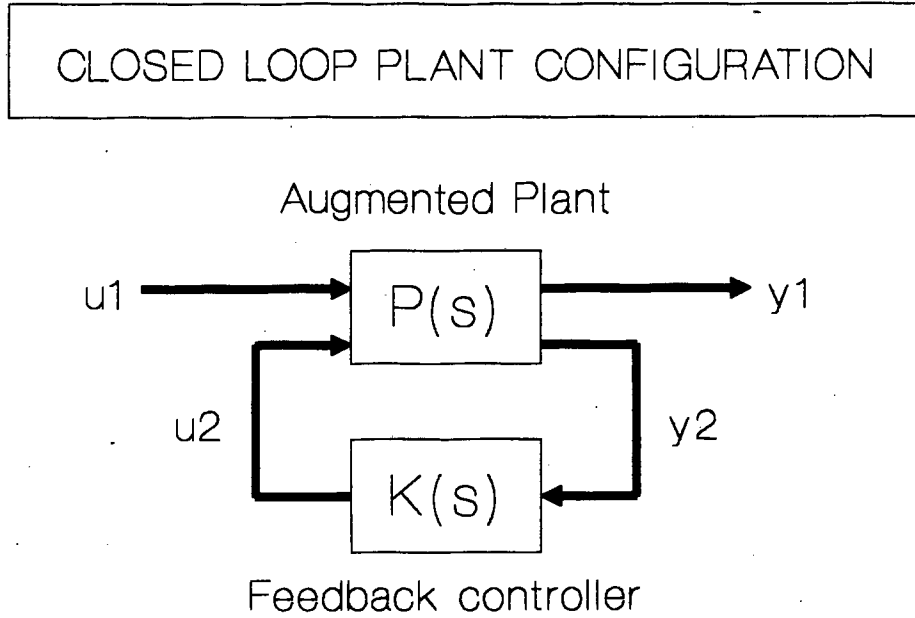


Figure 5.2 : NOMINAL PLANT WITH CONTROLLER IN FEEDBACK

If we suppose that $P(s)$ can be partitioned such that

$$P(s) = \begin{bmatrix} P_{11}(s) & P_{12}(s) \\ P_{21}(s) & P_{22}(s) \end{bmatrix} \quad (5.5)$$

Then in terms of figure 5.2, we can write

$$\begin{aligned} y_1 &= P_{11} * u_1 + P_{12} * u_2 \\ y_2 &= P_{21} * u_1 + P_{22} * u_2 \end{aligned} \quad (5.6)$$

The variables y_2 and u_2 can be eliminated, using $u_2 = K y_2$, which results in

$$y_1 = \left[P_{11} + P_{12}K(I - P_{22}K)^{-1}P_{21} \right] * u_1 \quad (5.7)$$

This is more conveniently written as

$$y_1 = F_1(P, K) * u_1 \quad (5.8)$$

small gain theorem

The robust control problem stated in 5.8 is called an H_∞ small gain problem⁸. It is solved if given a plant $P(s)$, a stabilising controller $K(s)$ can be found such that the closed loop function $F_1(P, K)$ is internally stable and its infinity-norm is less than or equal to one.

$$\|F_1(P, K)\|_\infty \leq 1 \quad (5.9)$$

The Γ -iteration algorithm developed by Doyle involves choosing a $\Gamma > 0$ and finding a controller $K(s)$ that satisfies the following problem.

$$\|F_1(P, K)\|_\infty \leq \Gamma \quad (5.10)$$

The algorithm then involves choosing a large value of Γ to start with and then gradually decreasing it. The Γ value chosen is included in the plant and the small gain problem of equation 5.9 is solved to give a stabilising controller.

It has been shown that as Γ is increased to very large values, the Glover-Doyle algorithm generates a controller that converges to a controller found using the LQG method⁹.

~~✱~~ For this non-optimal case, the algorithm will give a controller with the same number of states as the plant $P(s)$.

The minimum value of Γ is that for which there is no smaller value for which a stabilising controller can be found. During the designs that the author has performed it was noticed that the optimal value found for Γ yielded a controller with the least possible number of states.

5.3 : OBTAINING A PLANT MODEL

A description for the plant is usually obtained from step tests. From these tests a matrix of transfer functions is found that describe the plant dynamics and steady state values. The matrix elements are presented in the form of Laplace transforms.

The solution of the H_{∞} problem uses a plant described in state space form, $G(s) : [A \ B; C \ D]$. Thus the transfer function matrix needs to be converted from transfer function matrix to state space matrix form. This process often results in a system with a large number of states, many of them superfluous.

Thus a model reduction is performed and the H_{∞} method done on this reduced order model. This not only saves on computational time but also allows a controller of lower order (less states) to be found than would be possible with a full-order plant model.

The reduction technique used for this dissertation is the Optimal Hankel minimum degree approximation. It is a method developed by Safonov and is documented and coded as a function that can be used with MATLAB¹⁰. The reduced order model should be checked with the original in the frequency as well as the time domains. This involves finding the minimum order model for which the principle gain plots and the time domain simulations remain virtually the same.

5.4 : WEIGHTING FUNCTIONS SPECIFICATIONS

For multivariable systems the performance specifications are expressed in terms of principle gains. The H_{∞} method requires weighting functions to be placed on the signals that measure the required performance functions. The traditional closed loop feedback configuration and weighting functions are shown in the figure below.

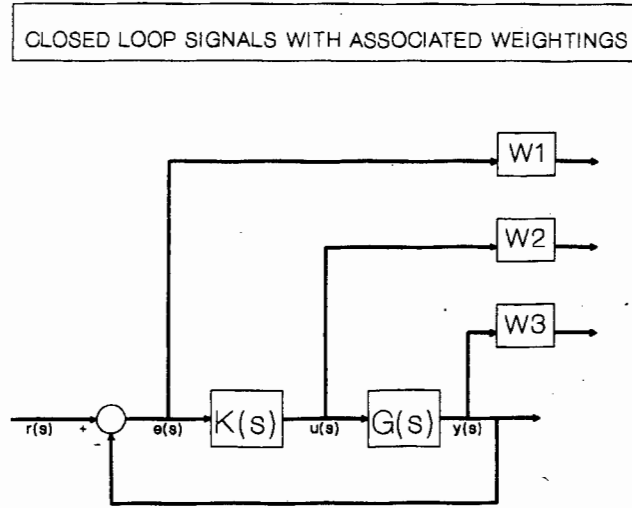


Figure 5.3 : CLOSED LOOP CONFIGURATION WITH WEIGHTS

The functions that measure the performance of the closed loop system are

$$\text{Sensitivity } S(s) = \frac{e(s)}{r(s)} = (I + G(s)K(s))^{-1}I \quad (5.11)$$

$$\text{Input Action } R(s) = \frac{u(s)}{r(s)} = (I + G(s)K(s))^{-1}K(s) \quad (5.12)$$

$$\text{Robustness } T(s) = \frac{y(s)}{r(s)} = (I + G(s)K(s))^{-1}G(s)K(s) \quad (5.13)$$

The closed loop requirements, specified in terms of principle gains can typically be described as follows¹¹.

- (1) SENSITIVITY : The measured output should not be affected by added disturbances at the output. The transfer function between the output and disturbances is the same as between the error and the setpoint. Thus we have a requirement that $\bar{\sigma}[S(s)]$ should be as small as possible.
- (2) ROBUSTNESS : Uncertainty in the plant can be described in terms of multiplicative and additive uncertainties. It is common to lump these two into a single multiplicative uncertainty¹². The smaller the maximum principle gain of the robustness function, the greater will be the size of the smallest destabilising perturbation, and hence the greater the stability margin. Hence a requirement is that $\bar{\sigma}[T(s)]$ be as small as possible. This also has the effect of attenuating sensor noise.
- (3) SETPOINT TRACKING : The most elementary of performance objectives is that the outputs reach their specified values. This results in limit being imposed on both the minimum and maximum principle gains. The requirement is that $\bar{\sigma}[T(s)] \approx 1$ and also $\underline{\sigma}[T(s)] \approx 1$.
- (4) MINIMISE CONTROL ACTION : Most real systems have a limit on the amount of input action that can be delivered. This poses the problem that if a designed controller has too large a control action, the inputs will hit limits and the linear theory to predict the outputs will be invalid¹³. Thus a further requirement is to keep $\bar{\sigma}[K(s)]$ as small as possible.

These requirements conflict with one another, for example 2 conflicts with 3. Thus the solution is to achieve each specification within a specified frequency range.

5.4.1 : DESIGN OF WEIGHT ON $S(s)$

The design objective of sensitivity can be achieved by keeping $\bar{\sigma}[S(s)]$ less than some weighting function at low frequencies. High frequency disturbance rejection needs to be sacrificed to accommodate the robustness requirements.

$$\begin{aligned}\bar{\sigma}[S(s)] &\leq W_1^{-1} \\ \|W_1 S\|_\infty &\leq 1\end{aligned}\tag{5.14}$$

The design of a suitable weight W_1^{-1} then becomes a filter design problem. Typically W_1^{-1} would be as depicted in the figure below.

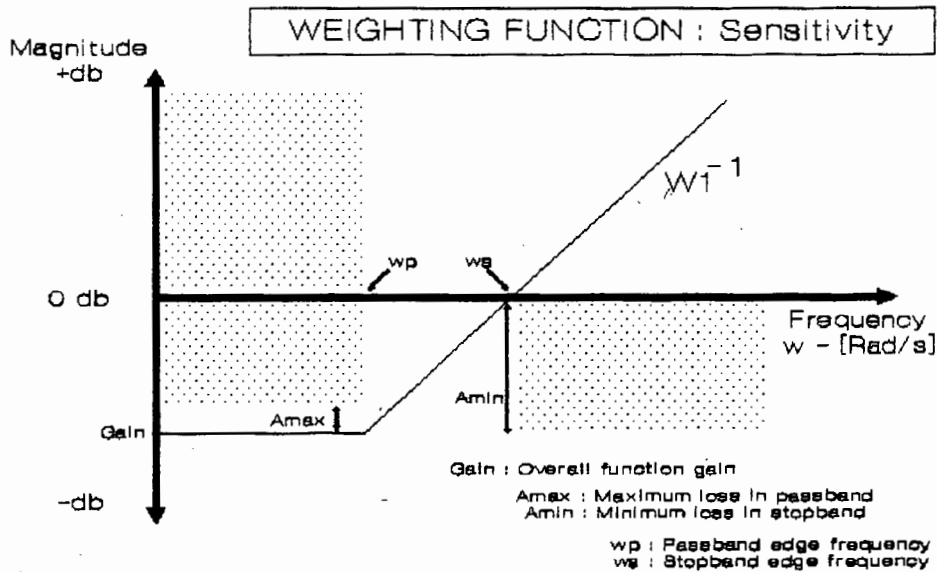


Figure 5.4 : WEIGHTING FUNCTION W_1^{-1}

Since the function W_1 , not its inverse W_1^{-1} , needs to be multiplied with $S(s)$, W_1 need to have a realisable state space form. Figure 5.4 indicates W_1 should be designed as a LOW PASS filter with a gain offset. The method used in this dissertation is the Butterworth design method¹⁴.

5.4.2 : DESIGN OF WEIGHT ON $R(s)$

The algorithm developed by Glover and Doyle requires that the weight W_2 be non zero. If it is not present certain state space matrix rank conditions will not be satisfied. This weighting function also serves the purpose of satisfying the input action design objective.

$$\begin{aligned} \bar{\sigma}[R(s)] &\leq W_2^{-1} \\ \|W_2 R\|_\infty &\leq 1 \end{aligned} \quad (5.15)$$

In controller designs documented to date¹⁵ the weighting function W_2^{-1} has been chosen as a constant, to satisfy the required rank condition.

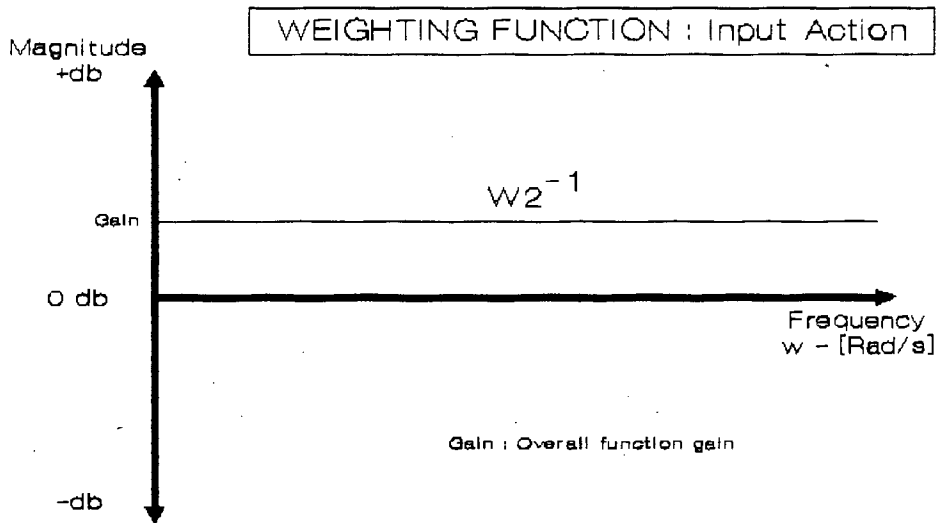


Figure 5.5 : WEIGHTING FUNCTION W_2^{-1}

It has been shown that W_2^{-1} need not be included to satisfy the rank conditions¹⁶. But this requires W_3^{-1} to be chosen in a specific form and the plant model to be amended.

This does not make for a versatile means of formulating a general strategy to be adopted when designing controllers. This together with the usefulness of w_2^{-1} in limiting the input action has led to it being included in the weights used in this dissertation.

5.4.3 : DESIGN OF WEIGHT ON $T(s)$

The design objective of setpoint tracking will be achieved if $\bar{\sigma}[S(s)] \ll 1$ at low frequencies, since¹⁷ $S(s) + T(s) = I$. The robustness specification can be achieved if w_3^{-1} is chosen such that $\bar{\sigma}[T(s)]$ is small at high frequencies. The conflicting requirements have now been resolved by dividing the problem in frequency.

$$\begin{aligned} \bar{\sigma}[T(s)] &\leq w_3^{-1} \\ \|w_3 T\|_\infty &\leq 1 \end{aligned} \quad (5.16)$$

The design of a suitable weight w_3^{-1} is then also a filter design problem. Typically w_3^{-1} would be as depicted in the figure below.

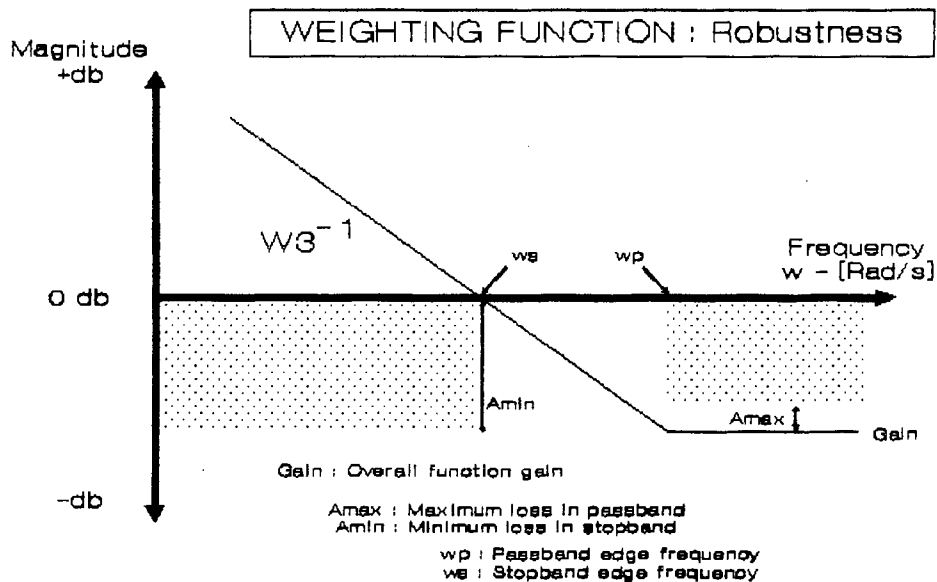


Figure 5.6 : WEIGHTING FUNCTION w_3^{-1}

As with W_1 , the weight W_3 needs to have a state space realisation. It is evident from figure 5.6 that W_3 needs to be designed as a HIGH PASS filter with associated gain. Once again the Butterworth method is used¹⁸.

No mention has yet been made on how to choose the filter parameters to achieve the design objectives. There is no documentation in literature to date to suggest how one goes about doing this. Part of the aim of this dissertation is to formulate such a scheme. This process will be viewed as a design stage and will be discussed in in section 5.6.

5.5 : H_∞ SYNTHESIS

This section details how the plant model and weighting functions are adapted to form an augmented plant, a form in which the H_∞ problem can be solved. The rank conditions that need to be satisfied will be defined, as will the scaling required to achieve certain matrix conditions.

Once this has been done a stabilising controller can be found by solving two Ricatti equations. The method of finding such a controller will be explained briefly. Finally the scaling performed on the plant need to be incorporated into the controller.

The scheme that will be laid out in this section is that adopted by Craig¹⁹, which is the algorithm formulated by Glover and Doyle.

5.5.1 : AUGMENT PLANT

The closed loop plant configuration of figure 5.3 needs to be converted, augmented, into the form shown in figure 5.2. The plant and weights are assumed to both have realizable state space forms.

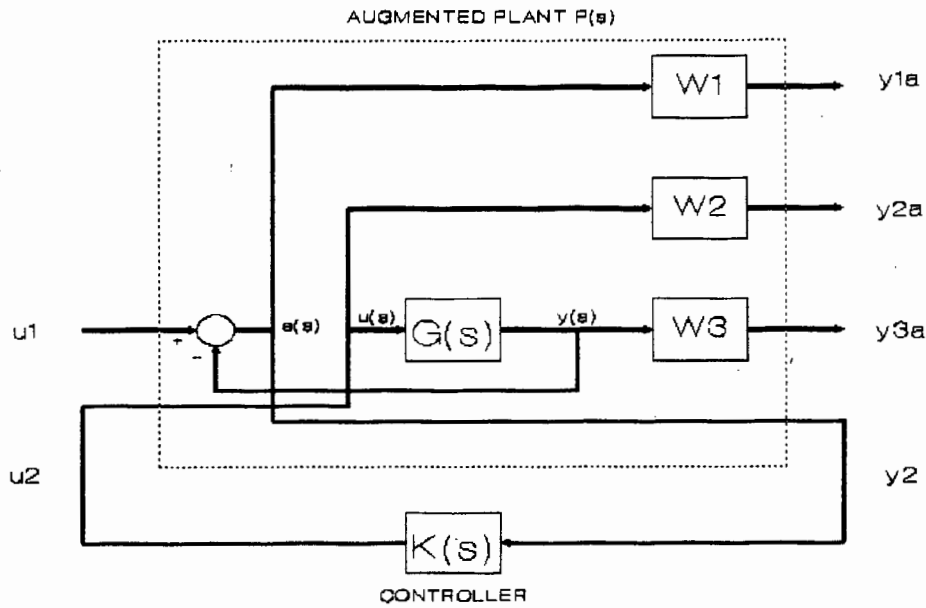


Figure 5.7 : AUGMENTED PLANT

Using equations 5.5 and 5.6 this augmented plant has the following description

$$\begin{aligned} y1 &= \begin{bmatrix} W_1 \\ 0 \\ 0 \end{bmatrix} u1 + \begin{bmatrix} -W_1 G \\ W_2 \\ W_3 G \end{bmatrix} u2 \\ y2 &= \begin{bmatrix} I \end{bmatrix} u1 + \begin{bmatrix} -G \end{bmatrix} u2 \end{aligned} \quad (5.17)$$

Assuming the plant has a state space realisation of

$$G(s) = \left[\begin{array}{c|c} A_P & B_P \\ \hline C_P & D_P \end{array} \right] \quad (5.18)$$

Assuming the weights have state space realisations of

$$W_i(s) = \left[\begin{array}{c|c} A_{Wi} & B_{Wi} \\ \hline C_{Wi} & D_{Wi} \end{array} \right] \quad (5.19)$$

Using the state space descriptions above, the augmented plant can be realised in state space form. The augmented plant then has realisation

$$P(s) = \left[\begin{array}{c|cc} A & B_1 & B_2 \\ \hline C_1 & D_{11} & D_{12} \\ C_2 & D_{21} & D_{22} \end{array} \right] \quad (5.20)$$

Where the matrices are as follows

$$A = \left[\begin{array}{cccc} A_p & 0 & 0 & 0 \\ -B_{w1}C_p & A_{w1} & 0 & 0 \\ 0 & 0 & A_{w2} & 0 \\ B_{w3}C_p & 0 & 0 & A_{w3} \end{array} \right] \quad (5.21)$$

$$[B_1 | B_2] = \left[\begin{array}{c|cc} 0 & B_p & \\ B_{w1} & -B_{w1}D_p & \\ 0 & B_{w2} & \\ 0 & B_{w3}D_p & \end{array} \right] \quad (5.22)$$

$$\left[\begin{array}{c} C_1 \\ C_2 \end{array} \right] = \left[\begin{array}{cccc} -D_{w1}C_p & C_{w1} & 0 & 0 \\ 0 & 0 & C_{w2} & 0 \\ D_{w3}C_p & 0 & 0 & C_{w3} \\ \hline -C_p & 0 & 0 & 0 \end{array} \right] \quad (5.23)$$

$$\left[\begin{array}{c|c} D_{11} & D_{12} \\ \hline D_{21} & D_{22} \end{array} \right] = \left[\begin{array}{c|c} D_{w1} & -D_{w1}D_p \\ 0 & D_{w2} \\ 0 & D_{w3}D_p \\ \hline I & -D_p \end{array} \right] \quad (5.24)$$

This concludes the state space description of the augmented plant and weighting functions.

5.5.2 : SCALE PLANT

The augmented plant needs to be scaled to accommodate the Γ iteration procedure. A scaled plant incorporates the Γ in equation 5.10 and a controller is found to solve the original specification of equation 5.9.

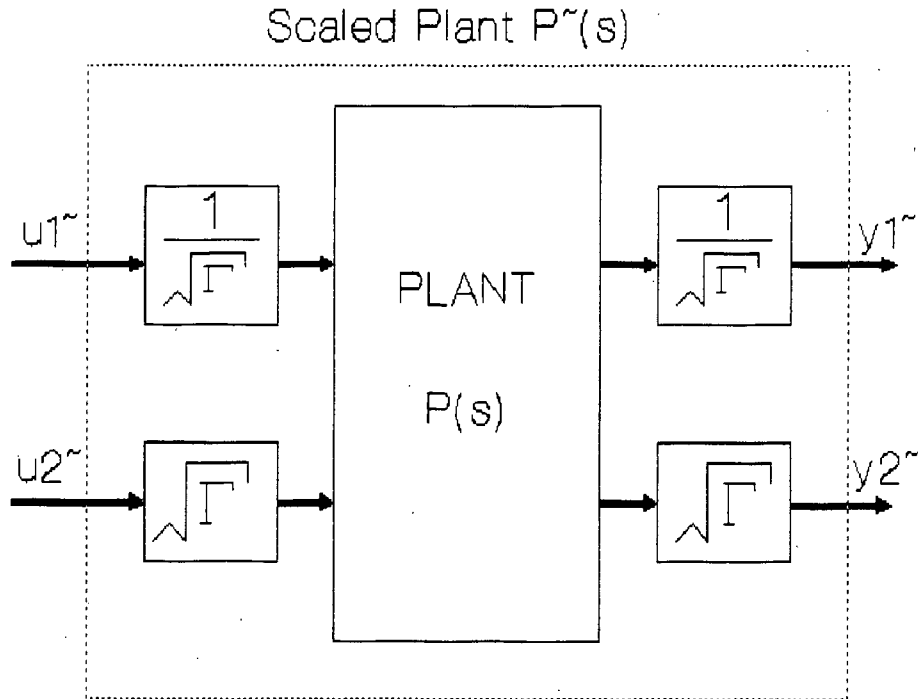


Figure 5.8 : SCALE THE PLANT

The scaled plant, $\tilde{P}(s)$, then has a realisations of

$$\tilde{P}(s) = \left[\begin{array}{c|cc} A & B_1/\sqrt{\Gamma} & B_2^*/\sqrt{\Gamma} \\ \hline C_1/\sqrt{\Gamma} & D_{11}/\Gamma & D_{12} \\ C_2^*/\sqrt{\Gamma} & D_{21} & D_{22}^*\Gamma \end{array} \right] \quad (5.25)$$

5.5.3 : SCALE CONTROLS AND MEASUREMENTS

The error and input signal, which form the inputs and outputs respectively to the controller, need to be scaled such that

$$\begin{aligned} D_{12}^T D_{12} &= I \\ D_{21} D_{21}^T &= I \end{aligned} \quad (5.26)$$

Two matrices, S_u and S_y , are used for the scaling. They can be computed using Cholesky decomposition to solve the following

$$\begin{aligned} S_u^T S_u &= D_{12}^T D_{12} \\ S_y^{-1} (S_y^{-1})^T &= D_{21} D_{21}^T \end{aligned} \quad (5.27)$$

The scaling is performed by inserting the matrices as shown in the figure below

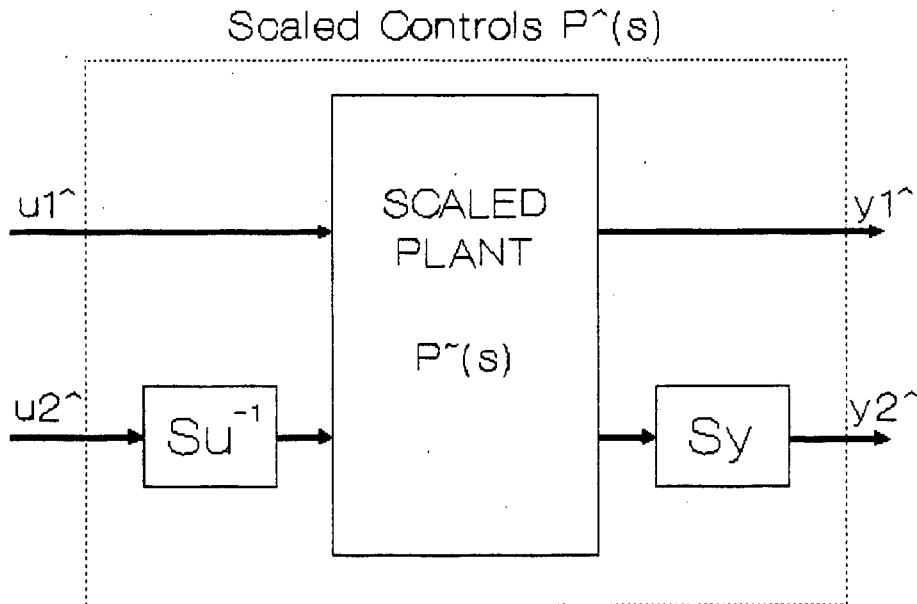


Figure 5.9 : SCALE CONTROLS AND MEASUREMENTS

The new plant matrix, $P^\wedge(s)$, then has a realisations of

$$P^\wedge(s) = \left[\begin{array}{c|cc} \tilde{A} & \tilde{B}_1 & \tilde{B}_2 s^{-1} \\ \hline \tilde{C}_1 & \tilde{D}_{11} & \tilde{D}_{12} s^{-1} \\ \tilde{S}y\tilde{C}_2 & \tilde{S}y\tilde{D}_{21} & \tilde{S}y\tilde{D}_{22} s^{-1} \end{array} \right] \quad (5.28)$$

5.5.4 : H_∞ SOLUTION

The solution will assume that the plant $P(s)$ has been scaled as indicated in sections 5.5.2 and 5.5.3. The procedure further requires that the augmented plant $P(s)$, the scaled version, have certain matrix properties.

5.5.4.1 : TRANSFER FUNCTION P_{11} BE STRICTLY PROPER

For $P_{11}(s)$ to be strictly proper, D_{11} must be zero. If as suggested in section 5.4.1 weighting function W_1 is designed as a low pass filter, then this will automatically be fulfilled since D_{W1} will be zero.

5.5.4.2 : TRANSFER FUNCTION P_{22} BE STRICTLY PROPER

All real plants exhibit high frequency roll-off²⁰. This means that D_p will be zero for real plants and consequently D_{22} will be zero as well, thus ensuring a strictly proper function $P_{22}(s)$.

5.5.4.3 : TRANSFER FUNCTION P_{21} BE PROPER

For $P_{21}(s)$ to be proper, it must be ensured that D_{21} is non-zero. In the standard feedback configuration used D_{21} is equal to the identity matrix and thus this condition is satisfied.

5.5.4.4 : TRANSFER FUNCTION P_{12} BE PROPER

If, as stated in section 5.5.4.2, real plants are studied then it is evident from equation 5.24 that in order for D_{12} to be non-zero (thus satisfying $P_{12}(s)$ be proper) the element D_{W2} must be non-zero. But since we have chosen to include weighting W_2 this can be satisfied if W_2 is designed as a constant, or as will be suggested later as a high pass filter.

A compensator can now be found for the scaled version of the augmented plant, since all the conditions are guaranteed to be met.

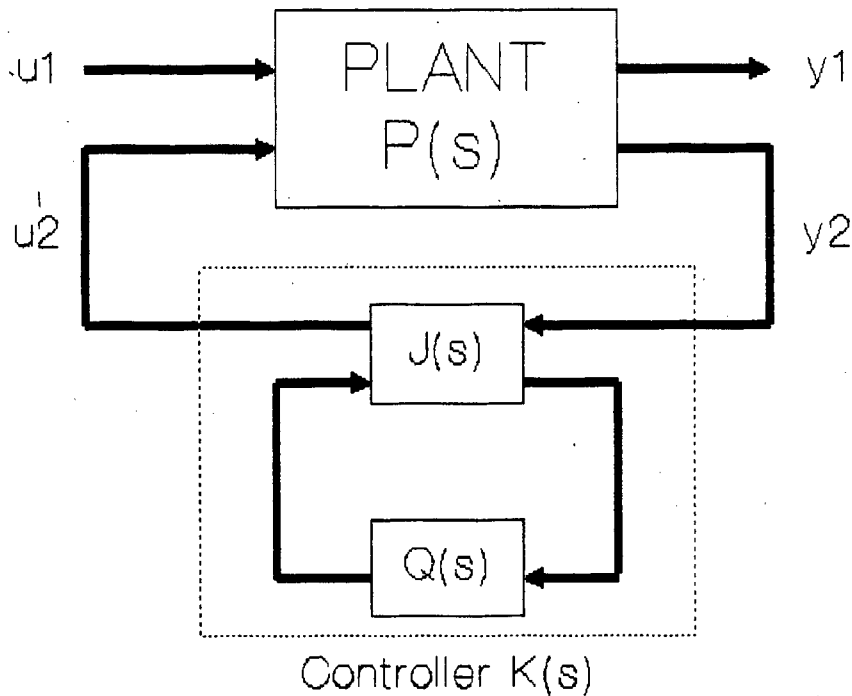


Figure 5.10 : H_∞ CONTROLLER STRUCTURE

A legitimate choice²¹ for Q is, $Q(s) = 0$. This value will be used throughout. The plant $P(s)$: $[A, B_1, B_2, C_1, C_2, D_{11}, D_{12}, D_{21}, D_{22}]$ is assumed to be scaled for Γ , control signals and measurements.

The state space realisation of $J(s)$ is

$$J(s) = \left[\begin{array}{c|c} A_J & B_J \\ \hline C_J & D_J \end{array} \right] \quad (5.29)$$

where the matrices are as follows

$$A_J = A - K_F C_2 - B_2 K_C + Y_\infty C_1^T (C_1 - D_{12} K_C) \quad (5.30)$$

$$B_J = [K_F \mid K_{F1}] \quad (5.31)$$

$$C_J = \left[\begin{array}{c} -K_C \\ K_{C1} \end{array} \right] \quad (5.32)$$

$$D_J = \left[\begin{array}{c|c} 0 & -I \\ \hline I & 0 \end{array} \right] \quad (5.33)$$

with

$$K_C = (B_2^T X_\infty + D_{12}^T C_1) (I - Y_\infty X_\infty)^{-1} \quad (5.34)$$

$$K_{C1} = (D_{12} B_1^T - C_2) (I - Y_\infty X_\infty)^{-1} \quad (5.35)$$

$$K_F = (Y_\infty C_2^T + B_1 D_{12}^T) \quad (5.36)$$

$$K_{F1} = (Y_\infty C_1^T D_{12} + B_2) \quad (5.37)$$

X_∞ is the unique solution of the Algebraic Riccati equation

$$\begin{aligned} (A-B_2D_{12}^TC_1)^TX_\infty + X_\infty(A-B_2D_{12}^TC_1) - X_\infty(B_2B_2^T-B_1B_1^T)X_\infty \\ + (C_1-D_{12}D_{12}^TC_1)^T(C_1-D_{12}D_{12}^TC_1) = 0 \end{aligned} \quad (5.38)$$

Y_∞ is the unique solution of the Algebraic Riccati equation

$$\begin{aligned} (A-B_1D_{21}^TC_2)Y_\infty + Y_\infty(A-B_1D_{21}^TC_2)^T - Y_\infty(C_2C_2^T-C_1C_1^T)Y_\infty \\ + (B_1-B_1D_{21}^TD_{21})^T(B_1-B_1D_{21}^TD_{12}) = 0 \end{aligned} \quad (5.39)$$

5.5.5 : OBTAIN CONTROLLER

The scaling that was performed on the plant $P(s)$ needs to be incorporated into the controller. This will result in the final controller to be implemented.

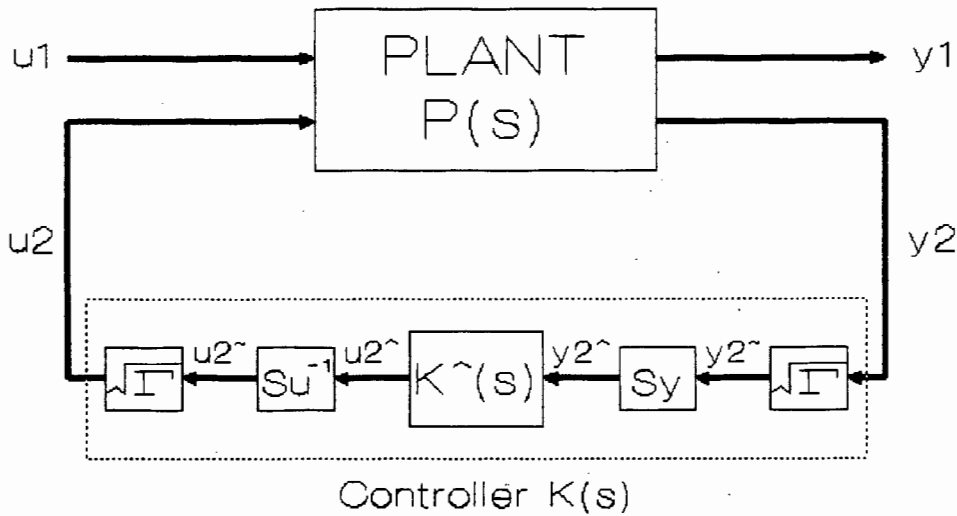


Figure 5.11 : SCALING THE CONTROLLER

With $Q(s) = 0$, the state space description of the compensator $K^\wedge(s)$ is

$$K^\wedge(s) = \left[\begin{array}{c|c} A_K^\wedge & B_K^\wedge \\ \hline C_K^\wedge & D_K^\wedge \end{array} \right] \quad (5.40)$$

where the matrices are as follows

$$A_K^\wedge = A - K_F C_2 - B_2 K_C + Y_\infty C_1^T (C_1 - D_{12} K_C) \quad (5.41)$$

$$B_K^\wedge = K_F \quad (5.42)$$

$$C_K^\wedge = K_C \quad (5.43)$$

$$D_K^\wedge = 0 \quad (5.44)$$

Incorporating the scaling, the final controller $K(s)$ has a realisation

$$K(s) = \left[\begin{array}{c|c} A_K & B_K \\ \hline C_K & D_K \end{array} \right] \quad (5.45)$$

where the matrices are as follows

$$A_K = A_K^\wedge \quad (5.46)$$

$$B_K = \sqrt{\Gamma} * B_K^\wedge * S_Y \quad (5.47)$$

$$C_K = \sqrt{\Gamma} * S_U^{-1} * C_K^\wedge \quad (5.48)$$

$$D_K = \Gamma * S_U^{-1} * D_K^\wedge * S_Y \quad (5.49)$$

5.6 : H_{∞} DESIGN

The decisions on how to adjust the weighting functions to achieve the desired performance objectives are design choices. The choices need to be made with the help of time simulations and principle gain plots of the closed loop system that the H_{∞} synthesis generates.

This section will outline the effects the weights have on the various objectives. These effects are based on the design experience the author has gained by using the H_{∞} method. The means of achieving certain of the frequency domain objectives, for example roll-off of $T(s)$ at high frequency, has been discussed in the literature. Little emphasis has however been placed on the means of achieving certain time domain specifications. In particular none of the references used by the author has considered the limiting of input action, a very real problem encountered on industrial plants²².

5.6.1 : SETPOINT TRACKING

The low frequency gain of W_1^{-1} determines if setpoint tracking will be achieved. If this gain is made very small, say below -60db, then it ensures that $S(s)$ is below this value and since $S(s) + T(s) = I$ this guarantees $T(s) = I$ at low frequencies. If W_1^{-1} is not made sufficiently small at low frequencies, then $S(s)$ does not become sufficiently small enough to ensure setpoint tracking.

5.6.2 : DISTURBANCE REJECTION

The rejection of low frequency disturbances is achieved in the same manner as setpoint tracking. Ensuring W_1^{-1} is very small at low frequencies ensures that $S(s)$ is small at those frequencies, thus achieving a dual gain of setpoint tracking and disturbance rejection.

The H_∞ synthesis method automatically eliminates interaction if the design specifications are chosen correctly. Choosing W_2^{-1} as a small constant value, results in a noticeable amount of interaction. Increasing its value decreases the amount of interaction. This is as a result of a relaxing of the constraint placed on the input action. More input action is allowed and thus the interaction can be eliminated.

5.6.6 : INPUT ACTION

It was found that in achieving the optimal solution using the Γ iteration procedure, the inputs for time simulations were large. In an attempt to eliminate these large discontinuous inputs, the weight W_2 was designed as a high pass filter, thus eliminating the high frequency discontinuities. This proved successful in reducing the inputs in magnitude, as well as having smooth input function in time.

The next chapter will present the design of a controller for the Heat Exchanger Rig. The design objectives that will be examined are those outlined in this section.

REFERENCES

1. ZAMES G., Feedback and Optimal Sensitivity : Model reference transformations, Multiplicative seminorms and Approximate inverses, *IEEE Trans. on Auto. Control*, Vol 26, No 1, pp.301-320, Feb 1981.
2. SAFONOV M.G., LAUB A.J. and HARTMAN G.L., Feedback Properties of Multivariable Systems : The Role and Use of the Return Difference Matrix, *IEEE Trans. on Auto. Control*, Vol 26, No 1, pp.47-65, Feb 1981.
3. MACIEJOWSKI J.M., Multivariable Feedback Design, Reading MA : Addison-Wesley, pp.75-80, 1989.

4. MACIEJOWSKI J.M., Multivariable Feedback Design, Reading MA : Addison-Wesley, pp.99-101, 1989.
5. VIDYASAGAR M., Control Systems Synthesis : A Factorisation Approach, Boston MA : MIT Press, 1985.
6. BRAAE M., Private Communication.
7. DOYLE J.C., GLOVER K., KHARGONEKAR P.P and FRANCIS B.A., State-Space Solutions to Standard H_2 and H_∞ Control Problems, *IEEE Trans. on Auto. Control*, Vol. 34, No.8, pp.831-847, Aug 1989.
8. CHIANG R.Y. and SAFONOV M.G., Robust Control Toolbox : for use with MATLAB, The MathWorks, pp.R10-R11, Jun 1988.
9. MACIEJOWSKI J.M., Multivariable Feedback Design, Reading MA : Addison-Wesley, pp.301-306, 1989.
10. CHIANG R.Y. and SAFONOV M.G., Robust Control Toolbox : for use with MATLAB, The MathWorks, pp.RR47-RR49, Jun 1988.
11. MACIEJOWSKI J.M., Multivariable Feedback Design, Reading MA : Addison-Wesley, pp.87-91, 1989.
12. CHIANG R.Y. and SAFONOV M.G., Robust Control Toolbox : for use with MATLAB, The MathWorks, pp.R6-R10, Jun 1988.
13. BRAAE M., Private Communication.
14. DARYANANI G., Principles of Active Network Synthesis and Design, Singapore : Wiley, pp.97-107, 1976.
15. CHIANG R.Y. and SAFONOV M.G., Robust Control Toolbox : for use with MATLAB, The MathWorks, pp.RR8-RR9, Jun 1988.
16. SAFONOV M.G., Simplifying the H_∞ Theory via Loop Shifting, *Proc. IEEE Conf. on Decision and Control*, Austin TX, pp.1399-1404, 1988.
17. MACIEJOWSKI J.M., Multivariable Feedback Design, Reading MA : Addison-Wesley, pp.101-102, 1989.
18. DARYANANI G., Principles of Active Network Synthesis and Design, Singapore : Wiley, pp.126-129, 1976.

19. CRAIG I.K., Sensitivity of H_{∞} Controller Designs to Structured Uncertainty, M.Sc. Dissertation, Department of Aeronautics and Astronautics, Massachusetts Institute of Technology, pp.19-42, May 1989.
20. BRAAE M., Private Communication.
21. CRAIG I.K., Sensitivity of H_{∞} Controller Designs to Structured Uncertainty, M.Sc. Dissertation, Department of Aeronautics and Astronautics, Massachusetts Institute of Technology, p.36, May 1989.
22. HULBERT D.G., MINTEK, Private Communication.

CHAPTER 6 : HERIG TEMPERATURE CONTROL USING H_{∞} DESIGN

In this chapter the design methodology of Chapter 5 will be put into practice to design a controller for an actual process, a Counter Current Heat Exchanger Rig.

This HERIG has been studied in depth in previous chapters and an open loop model has already been found for the process. This model will be reduced in order and the design will be done using the reduced order model. It will be shown that the reduced order model is a good approximation of the full order model.

The design will involve various stages before a final controller is chosen to be implemented on the HERIG. Once the final controller has been found, it will be implemented on the HERIG and the results obtained will be evaluated.

6.1 : SOFTWARE USED TO PERFORM DESIGN

The software needed to solve the synthesis problem was written by Safonov and Chiang¹ and is available in the MATLAB Robust Control Toolbox. Further software for MATLAB needed to be developed to implement the loading and saving of multivariable matrices, as well as writing a programme with which a controller for any plant can be designed. A listing of all the software written by the author is presented in Appendix G.

A PASCAL programme was also written to enable a user to input a multivariable transfer function matrix. MATLAB software written by the author then loads this data and converts it into a state space form. Finally software was also written to load and save the multivariable state space matrix forms.

6.2 : MODEL REDUCTION

The derivation of a multivariable transfer function matrix for the temperature response of the HERIG is detailed in Chapter 4. The matrix $G_T(s)$, when converted to a state space form, becomes a 28 state model. The reduction of this model was done using MATLAB. A routine for the Optimal Hankel minimum degree approximation was used².

The reduced order model needs to be evaluated by examining the frequency as well as the time responses of both systems. The reduced order was iteratively decreased until, by examining both sets of plots, it was felt that the minimum order model was found. The minimum order found was an 6TH order model.

The frequency response of the original 28TH order model and the reduced 6TH order model are compared in the following figure. Since each model has four inputs and four outputs there will be four principle gains plotted for each model.

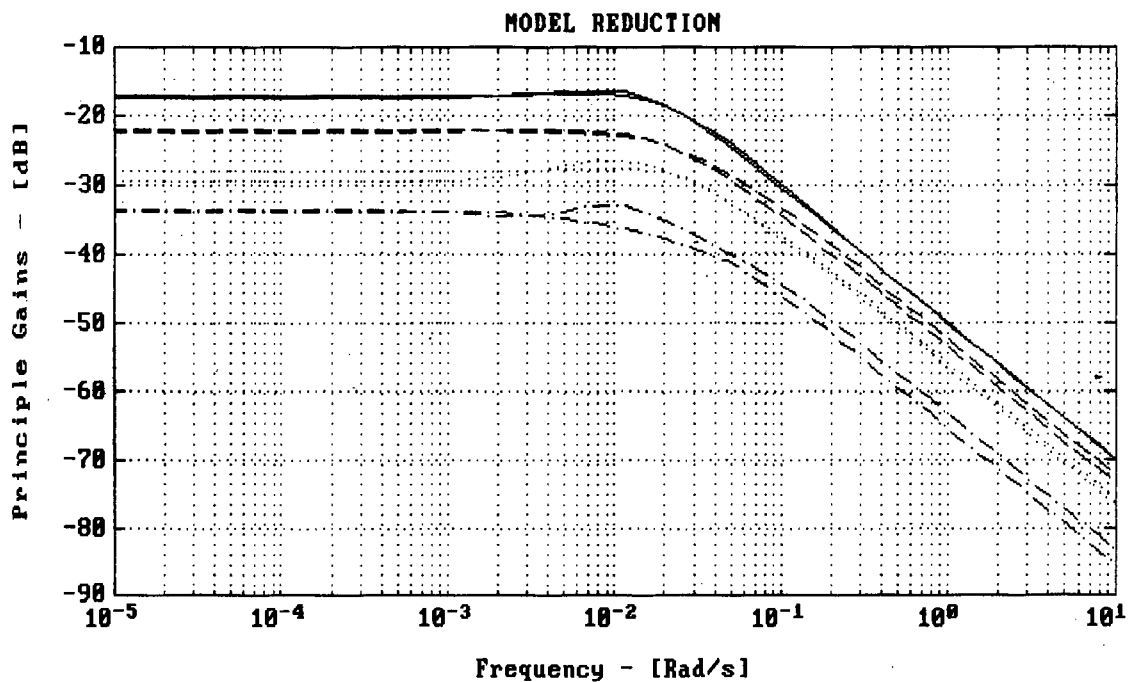


Figure 6.2.1 : COMPARATIVE FREQUENCY RESPONSE

The responses of the two systems are now compared for a step in the four inputs.

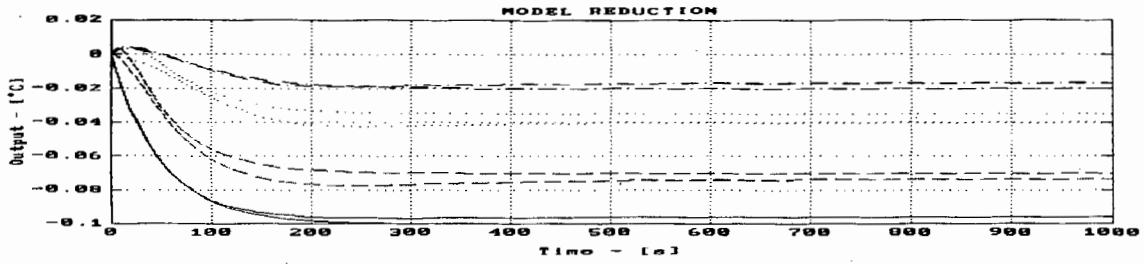


Figure 6.2.2 : COMPARATIVE STEP RESPONSE (STEP INPUT 1)

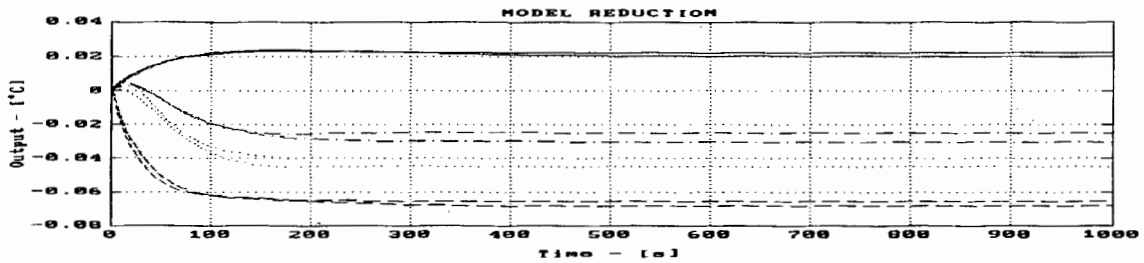


Figure 6.2.3 : COMPARATIVE STEP RESPONSE (STEP INPUT 2)

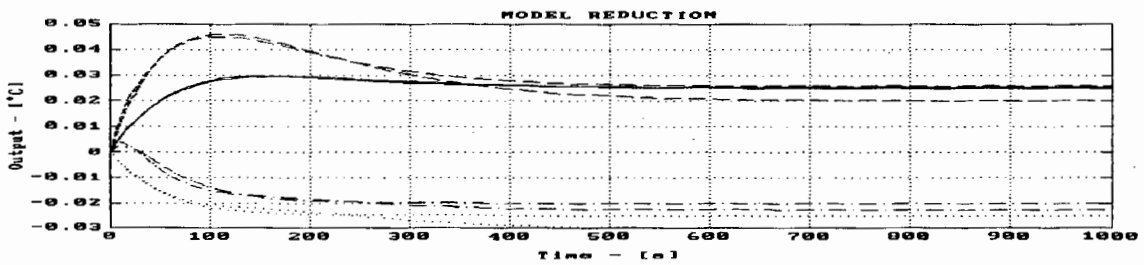


Figure 6.2.4 : COMPARATIVE STEP RESPONSE (STEP INPUT 3)

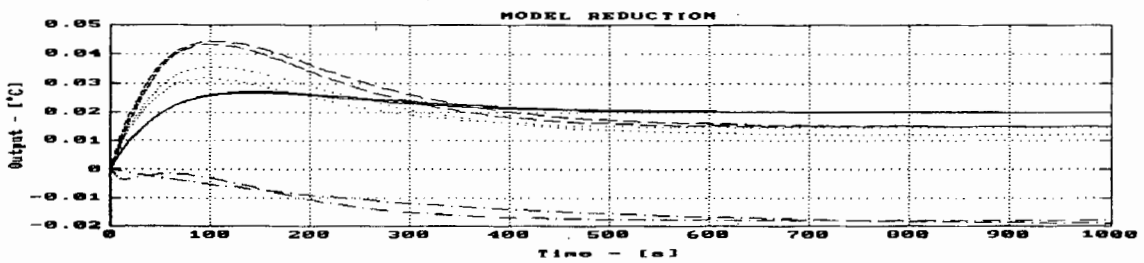


Figure 6.2.5 : COMPARATIVE STEP RESPONSE (STEP INPUT 4)

The responses show that the reduced order model is sufficiently similar to the original. The reduced order model has a state space realisation $G_T(s) : [A_T \ B_T; C_T \ D_T]$.

$$A_T = \begin{bmatrix} -0.11237 & 0.04249 & -0.02730 & -0.81698 & -0.50692 & -0.21676 \\ 0 & -0.03346 & 0.03701 & 0.20319 & -0.02755 & -0.91587 \\ 0 & 0 & -0.01945 & 0.19856 & 0.05059 & -0.30513 \\ 0 & 0 & -0.00147 & -0.01925 & 0.01134 & 0.01571 \\ 0 & 0 & 0 & 0 & -0.01606 & -0.05495 \\ 0 & 0 & 0 & 0 & 0.00065 & -0.00655 \end{bmatrix}$$

(6.1)

$$B_T = \begin{bmatrix} -0.05915 & 0.00106 & -0.13061 & 0.08681 \\ 0.02010 & -0.03872 & 0.03221 & 0.02388 \\ -0.02331 & 0.02644 & 0.02003 & 0.01249 \\ 0.00240 & 0.00031 & -0.00102 & 0.00007 \\ -0.00283 & -0.00033 & 0.00115 & 0.00177 \\ 0.00008 & 0.00005 & 0.00015 & 0.00022 \end{bmatrix}$$

(6.2)

$$C_T = \begin{bmatrix} 0.00006 & -0.01285 & 0.01144 & -0.25317 & 0.35220 & 0.68954 \\ -0.00700 & 0.02520 & -0.04804 & -0.10234 & 0.67748 & 0.90750 \\ 0.00168 & -0.00854 & -0.00382 & 0.61062 & 0.53456 & 0.30306 \\ -0.01030 & -0.00845 & -0.00050 & 0.61062 & 0.27354 & -0.40409 \end{bmatrix}$$

(6.3)

$$D_T = \begin{bmatrix} 0 & 0 & 0 & 0 \\ 0 & 0 & 0 & 0 \\ 0 & 0 & 0 & 0 \\ 0 & 0 & 0 & 0 \end{bmatrix}$$

(6.4)

This reduced order model has poles that are determined by the eigenvalues of the matrix A_T .

Poles of $G_T(s)$:

$$-0.1124$$

$$-0.0335$$

$$-0.0194 \pm j 0.0171$$

$$-0.0113 \pm j 0.0036$$

The zeros are computed as the transmission zeros of the state space system.

Zeros of $G_T(s)$:

$$0.0551$$

$$-0.0097$$

The transfer functions in Chapter 4 show that most of the full order system poles lie in the region between -0.01 and -0.02 . From the poles of the reduced order system it can be seen that this real value of pole position has been retained but that an imaginary component has been added.

The transfer function matrix has dead times in it, which are realised in state space via a first order Pade Approximation. This leads to non-minimum phase zeros being introduced. This right half plane zero characteristic has been retained by a zero at $+0.0551$ in the reduced order model.

6.3 : DESIGN NUMBER 1

The initial design is to make the closed loop system as fast as the open loop system. From Figure 6.2.1 it is seen that the open loop bandwidth is around 0.02 rad/s. Since the closed loop bandwidth is determined by the frequency at which the weights W_1^{-1} and W_3^{-1} cross over, these weights are chosen to cross over around this frequency. Both weights are chosen as first order functions so as to keep the order of the controller down.

To achieve disturbance rejection, setpoint tracking and robustness the low and high frequency gains of W_1^{-1} and W_3^{-1} respectively were chosen to be below -60db. The weight W_2^{-1} was chosen as a constant 10db, to keep the controller order as low as possible.

Table 6.1 : SPECIFICATIONS FOR WEIGHTS IN DESIGN 1

	Gain [dB]	wp [rad/s]	ws [rad/s]	A _{MAX} [dB]	A _{MIN} [dB]
(W1 ⁻¹) ₁₁	-60	1.0E-5	1.0E-2	4	60
(W1 ⁻¹) ₂₂	-60	1.0E-5	1.0E-2	4	60
(W1 ⁻¹) ₃₃	-60	1.0E-5	1.0E-2	4	60
(W1 ⁻¹) ₄₄	-60	1.0E-5	1.0E-2	4	60
(W2 ⁻¹) ₁₁	10	—	—	—	—
(W2 ⁻¹) ₂₂	10	—	—	—	—
(W2 ⁻¹) ₃₃	10	—	—	—	—
(W2 ⁻¹) ₄₄	10	—	—	—	—
(W3 ⁻¹) ₁₁	-60	2.0E+1	2.0E-2	4	60
(W3 ⁻¹) ₂₂	-60	2.0E+1	2.0E-2	4	60
(W3 ⁻¹) ₃₃	-60	2.0E+1	2.0E-2	4	60
(W3 ⁻¹) ₄₄	-60	2.0E+1	2.0E-2	4	60

6.3.1 : SYNTHESIS STAGE 1 ($\Gamma = 1E6$)

An initial value of $1E6$ was chosen for Γ . This large value of Γ results a controller that has 14 states, the same number of states that of the augmented plant $P(s)$.

The principle gains of weighting functions and the performance functions that result from the controller designed are displayed below.

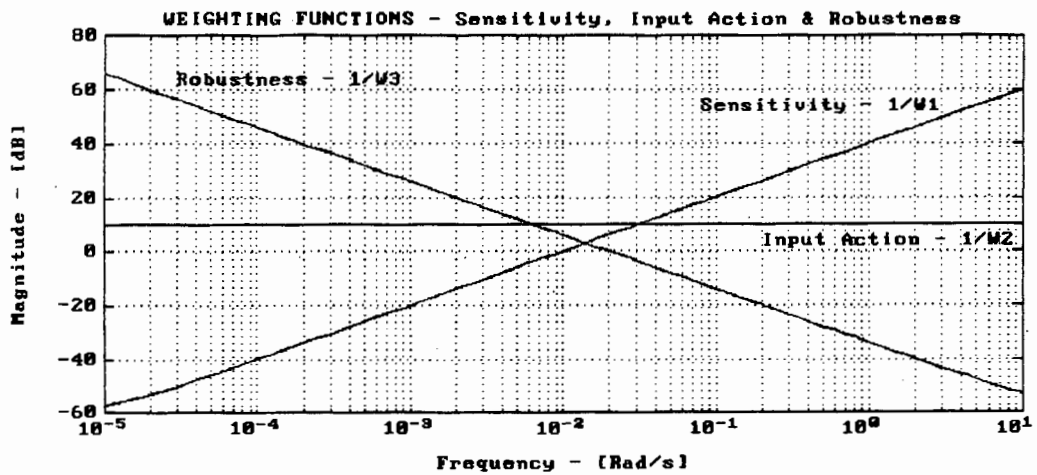


Figure 6.3.1 : WEIGHTING FUNCTIONS (DESIGN 1.1)

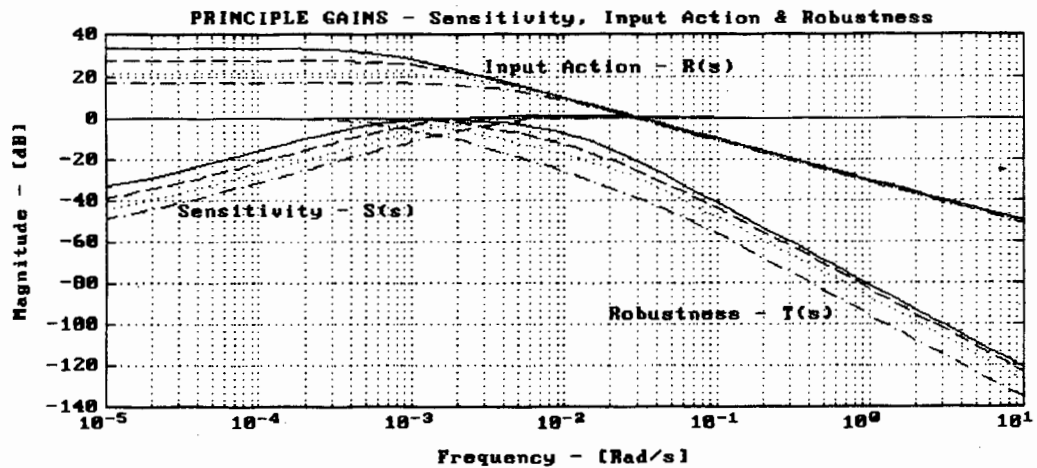


Figure 6.3.2 : PERFORMANCE FUNCTIONS (DESIGN 1.1)

The output and input functions for steps in the four setpoints are displayed in the figures that follow.

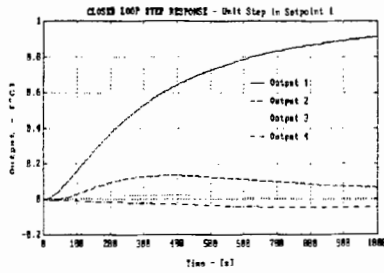


Figure 6.3.3 : OUTPUT
(Step Setpoint 1)

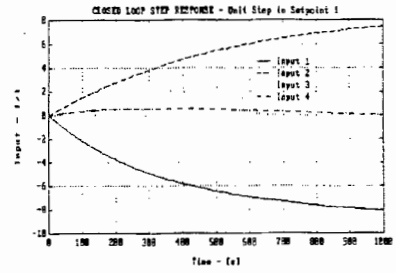


Figure 6.3.7 : INPUT
(Step Setpoint 1)

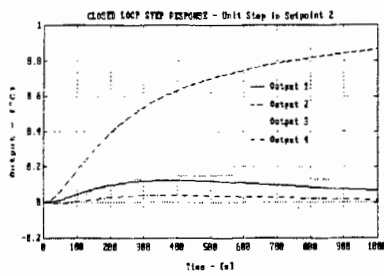


Figure 6.3.4 : OUTPUT
(Step Setpoint 2)

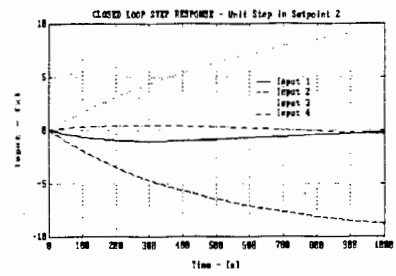


Figure 6.3.8 : INPUT
(Step Setpoint 2)

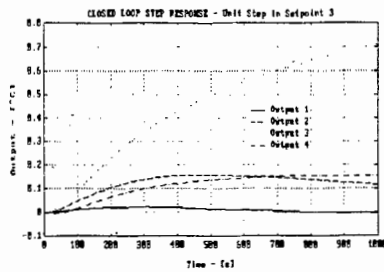


Figure 6.3.5 : OUTPUT
(Step Setpoint 3)

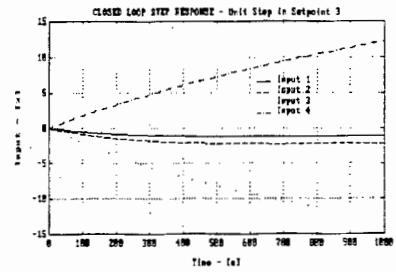


Figure 6.3.9 : INPUT
(Step Setpoint 3)

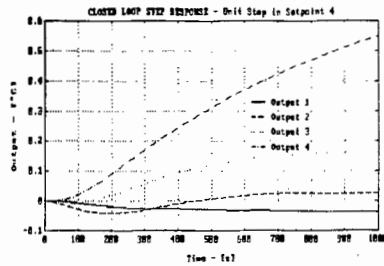


Figure 6.3.6 : OUTPUT
(Step Setpoint 4)

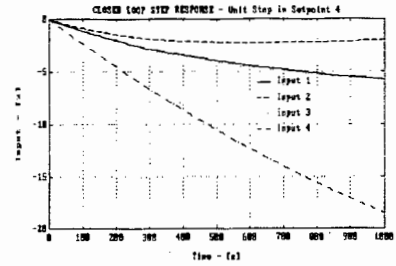


Figure 6.3.10 : INPUT
(Step Setpoint 4)

6.3.2 : SYNTHESIS STAGE 2 ($\Gamma = 16$)

The value of Γ was reduced until the minimum value was found, $\Gamma = 16$. This gives a controller with 10 states, the optimal number of states for this choice of weighting functions.

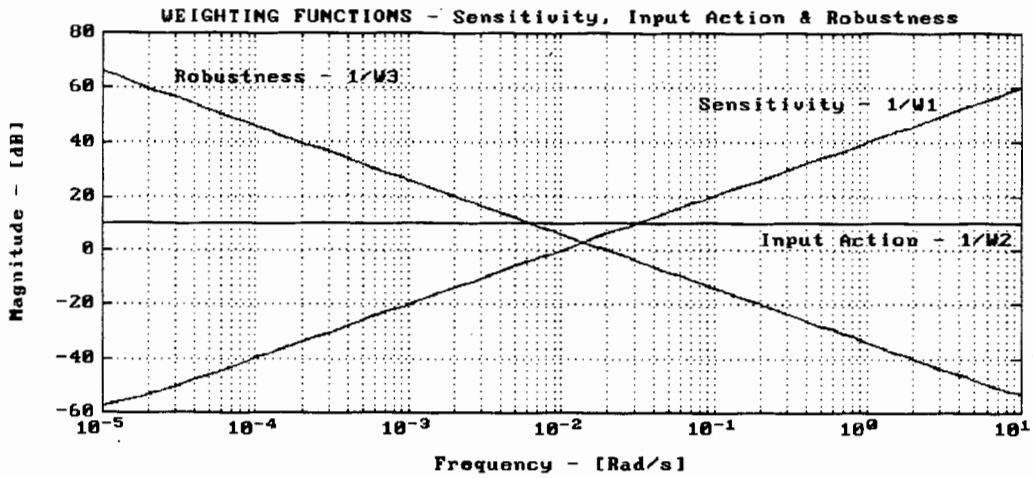


Figure 6.3.11 : WEIGHTING FUNCTIONS (DESIGN 1.2)

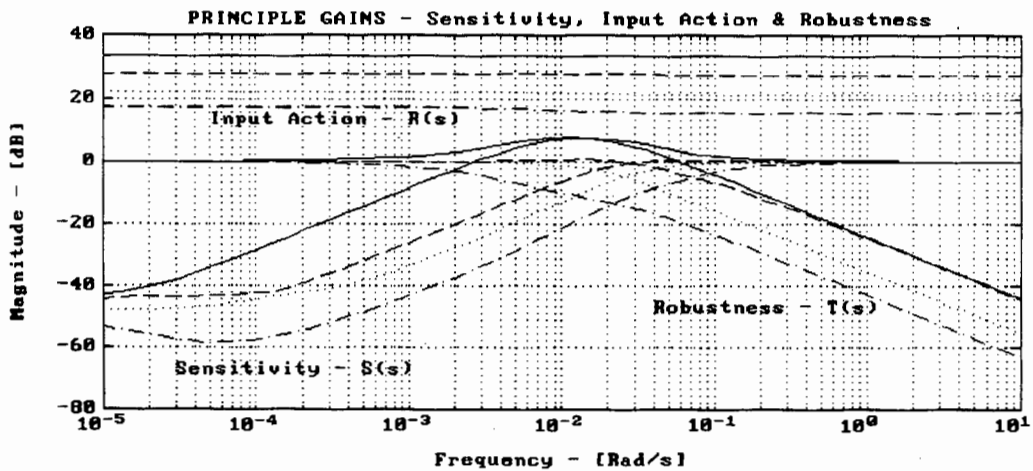


Figure 6.3.12 : PERFORMANCE FUNCTIONS (DESIGN 1.2)

The output and input functions for steps in the four setpoints are displayed in the figures that follow.

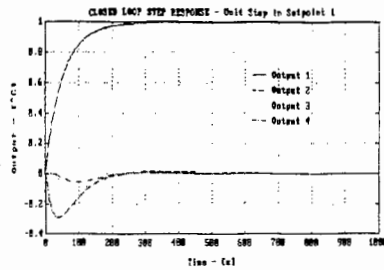


Figure 6.3.13 : OUTPUT
(Step Setpoint 1)

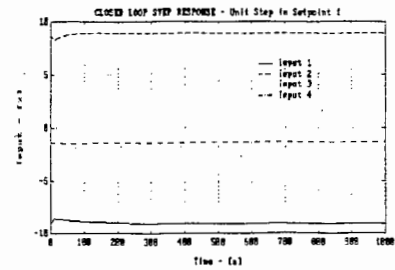


Figure 6.3.17 : INPUT
(Step Setpoint 1)

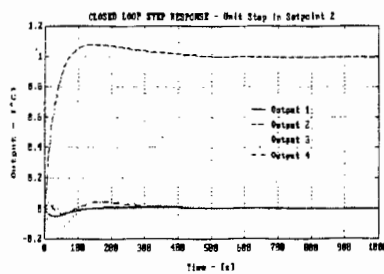


Figure 6.3.14 : OUTPUT
(Step Setpoint 2)

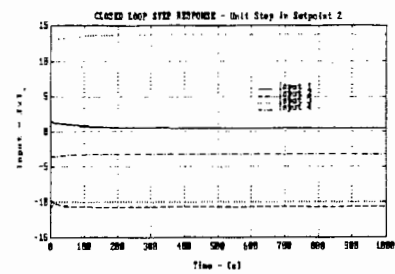


Figure 6.3.18 : INPUT
(Step Setpoint 2)

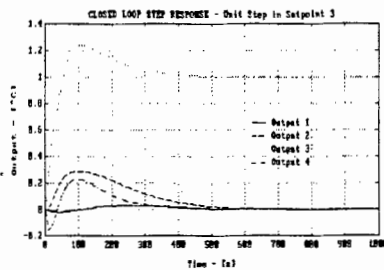


Figure 6.3.15 : OUTPUT
(Step Setpoint 3)

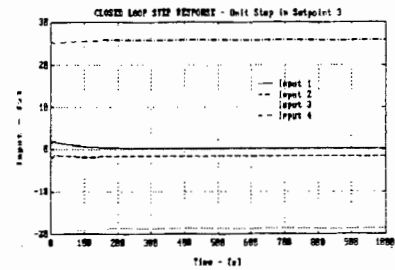


Figure 6.3.19 : INPUT
(Step Setpoint 3)

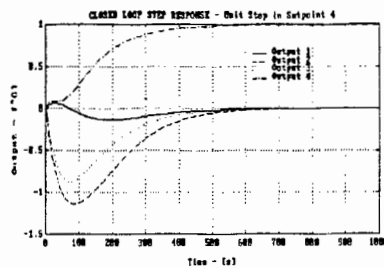


Figure 6.3.16 : OUTPUT
(Step Setpoint 4)

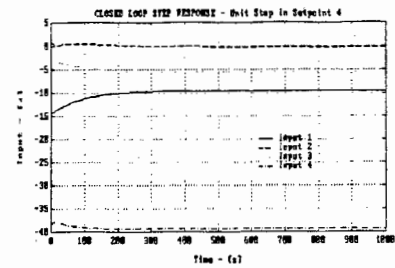


Figure 6.3.20 : INPUT
(Step Setpoint 4)

The principle gain plot of $T(s)$ for both synthesis cases are not close together. This indicates that the loops will not behave like independent single variable loops and interaction can thus be expected. This unacceptably high amount of interaction is confirmed when the time simulations are plotted.

This interaction is attributed to the low constraint (10 dB) placed on the input function. It should also be noted that for this low value of W_2^{-1} the functions $S(s)$ and $T(s)$ are not less than their weighting functions, even for the optimal solution. This phenomenon cannot be explained. The weighting function on the input will be relaxed in the next design in an attempt to rectify this and the interaction problems.

The synthesis technique of finding the minimum Γ value results in a controller of minimum order being found. A side effect of this optimal controller is that the inputs become discontinuous functions, they have large spikes as initial values. This high frequency action is due to the high frequencies of $R(s)$ not being attenuated (see Figure 6.3.12). The spikes do not as yet pose a problem since they remain less than $\pm 50\%$, the limiting region of input space.

Finally finding the Optimal controller also ensures that the speed of response of the on-diagonal elements are the same as the designed values. In synthesis stage 1 the bandwidth is a decade lower than expected and consequently the speed of response is much slower than anticipated. In stage 2 the bandwidth is around its expected value of 0.02 rad/s and response times around 50s.

6.4 : DESIGN NUMBER 2

The initial design of section 6.3 achieved setpoint tracking, due to the low gains chosen for W_1^{-1} and W_3^{-1} , and the desired response times for the loops were also achieved. There was however a large amount of interaction between the loops. This second design will attempt to eliminate this interaction but still retain the setpoint tracking, disturbance rejection, robustness and response times attained in the previous section.

To maintain the response the cross-over frequency of the first and last weights must be maintained. Their gain must be retained at the low value of -60db to ensure the other specifications still remain.

The initial value of 10db on W_2^{-1} is relaxed to 60db. It should be noted that there is still no frequency related constraint, since a constant value is still used.

Table 6.2 : SPECIFICATIONS FOR WEIGHTS IN DESIGN 2

	Gain [dB]	wp [rad/s]	ws [rad/s]	A _{MAX} [dB]	A _{MIN} [dB]
(W ₁ ⁻¹) ₁₁	-60	1.0E-5	1.0E-2	4	60
(W ₁ ⁻¹) ₂₂	-60	1.0E-5	1.0E-2	4	60
(W ₁ ⁻¹) ₃₃	-60	1.0E-5	1.0E-2	4	60
(W ₁ ⁻¹) ₄₄	-60	1.0E-5	1.0E-2	4	60
(W ₂ ⁻¹) ₁₁	60	—	—	—	—
(W ₂ ⁻¹) ₂₂	60	—	—	—	—
(W ₂ ⁻¹) ₃₃	60	—	—	—	—
(W ₂ ⁻¹) ₄₄	60	—	—	—	—
(W ₃ ⁻¹) ₁₁	-60	2.0E+1	2.0E-2	4	60
(W ₃ ⁻¹) ₂₂	-60	2.0E+1	2.0E-2	4	60
(W ₃ ⁻¹) ₃₃	-60	2.0E+1	2.0E-2	4	60
(W ₃ ⁻¹) ₄₄	-60	2.0E+1	2.0E-2	4	60

6.4.1 : SYNTHESIS STAGE 1 ($\Gamma = 1E6$)

An initial value of $1E6$ was chosen for Γ . This high value results in a controller that has 14 states, the same number of states that the augmented plant $P(s)$ has.

The principle gains of weighting functions and the performance functions that result from the controller designed are displayed below.

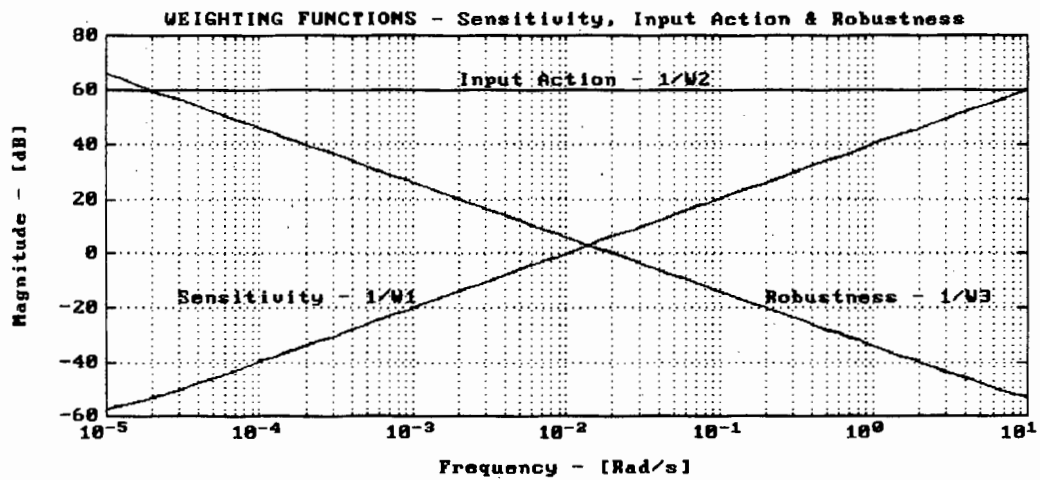


Figure 6.4.1 : WEIGHTING FUNCTIONS (DESIGN 2.1)

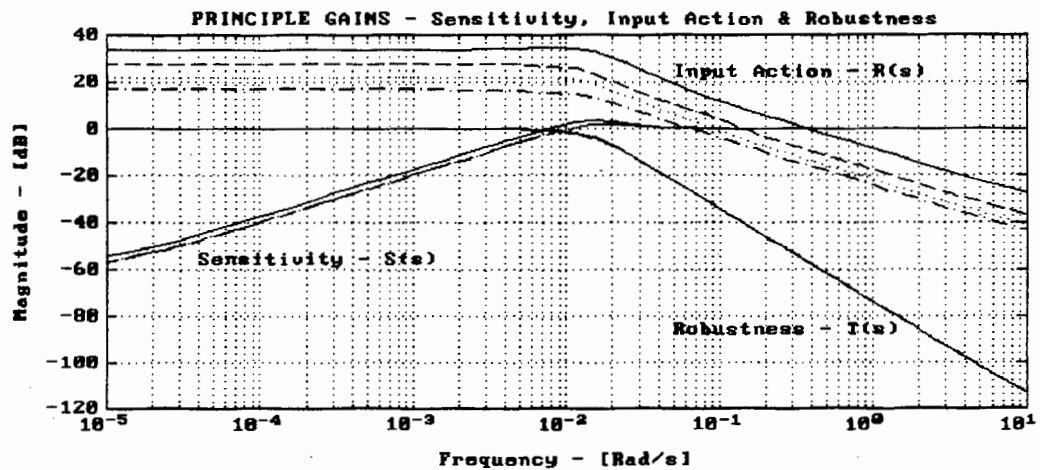


Figure 6.4.2 : PERFORMANCE FUNCTIONS (DESIGN 2.1)

The output and input functions for steps in the four setpoints are displayed in the figures that follow.

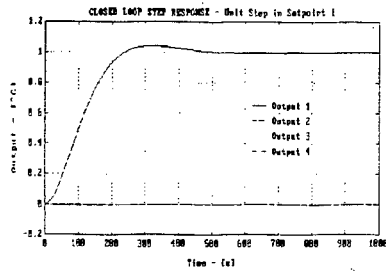


Figure 6.4.3 : OUTPUT
(Step Setpoint 1)

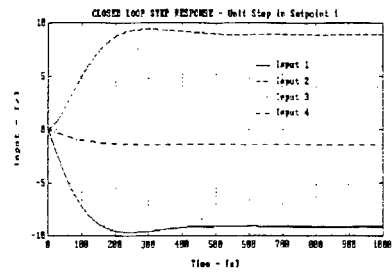


Figure 6.4.7 : INPUT
(Step Setpoint 1)

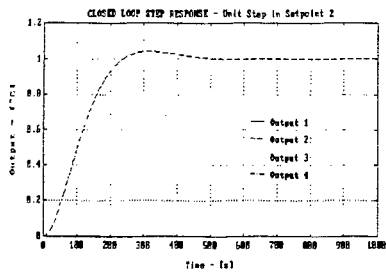


Figure 6.4.4 : OUTPUT
(Step Setpoint 2)

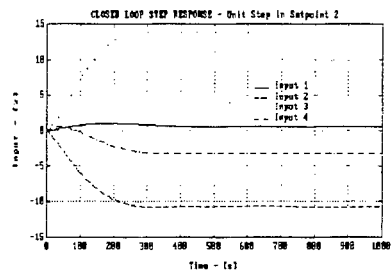


Figure 6.4.8 : INPUT
(Step Setpoint 2)

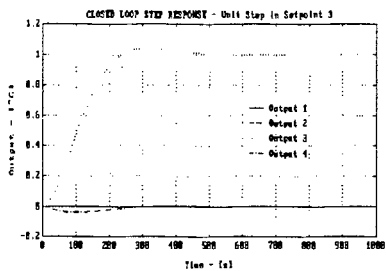


Figure 6.4.5 : OUTPUT
(Step Setpoint 3)

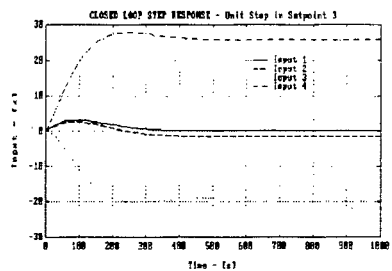


Figure 6.4.9 : INPUT
(Step Setpoint 3)

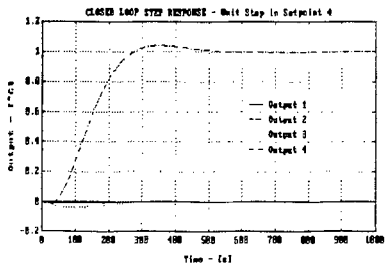


Figure 6.4.6 : OUTPUT
(Step Setpoint 4)

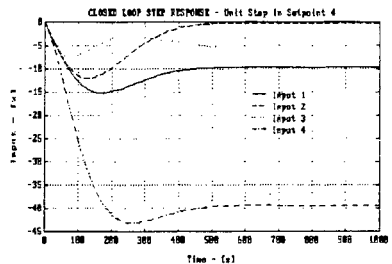


Figure 6.4.10 : INPUT
(Step Setpoint 4)

6.4.2 : SYNTHESIS STAGE 2 ($\Gamma = 0.95$)

The value of Γ was reduced until the minimum value was found, $\Gamma = 0.95$ (note that Doyle's algorithm only requires $\Gamma > 0$ and allows fractional values less than one). This gives a controller with 10 states, the optimal number of states for this choice of weighting functions.

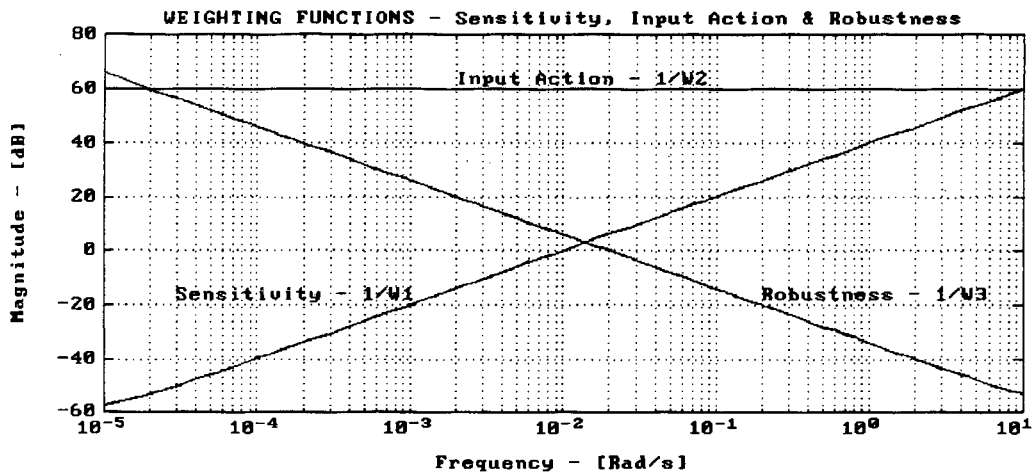


Figure 6.4.11 : WEIGHTING FUNCTIONS (DESIGN 2.2)

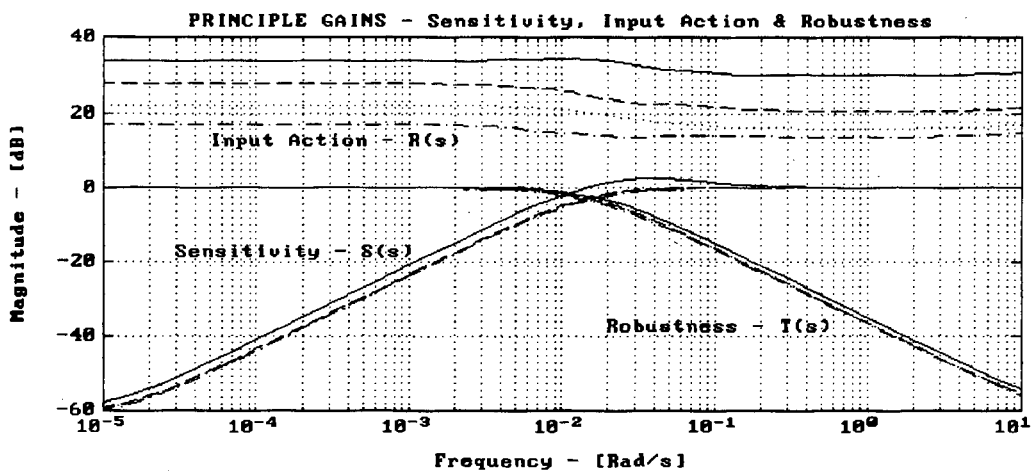


Figure 6.4.12 : PERFORMANCE FUNCTIONS (DESIGN 2.2)

The output and input functions for steps in the four setpoints are displayed in the figures that follow.

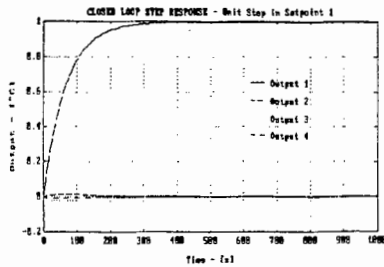


Figure 6.4.13 : OUTPUT
(Step Setpoint 1)

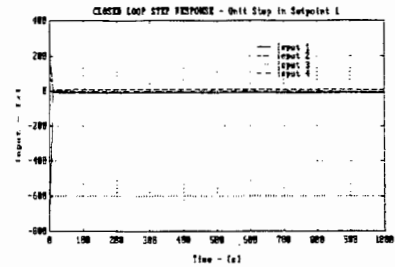


Figure 6.4.17 : INPUT
(Step Setpoint 1)

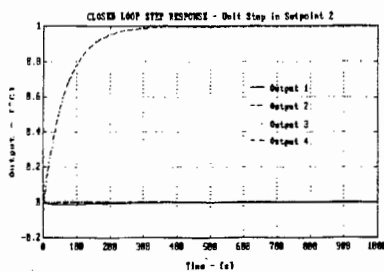


Figure 6.4.14 : OUTPUT
(Step Setpoint 2)

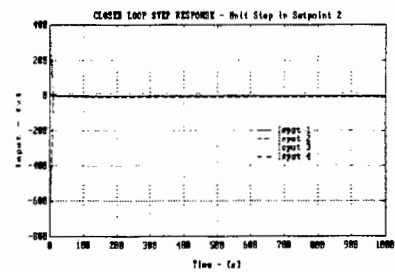


Figure 6.4.18 : INPUT
(Step Setpoint 2)

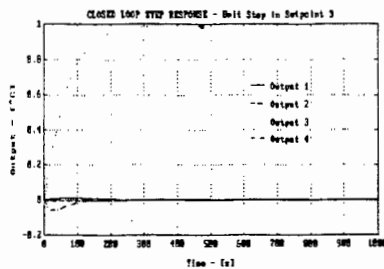


Figure 6.4.15 : OUTPUT
(Step Setpoint 3)

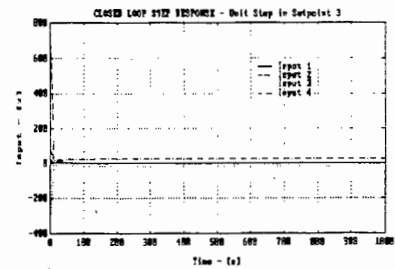


Figure 6.4.19 : INPUT
(Step Setpoint 3)

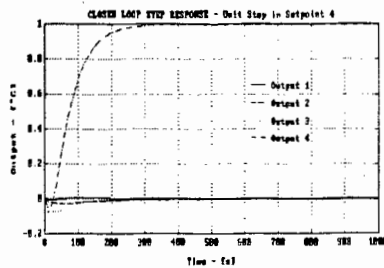


Figure 6.4.16 : OUTPUT
(Step Setpoint 4)

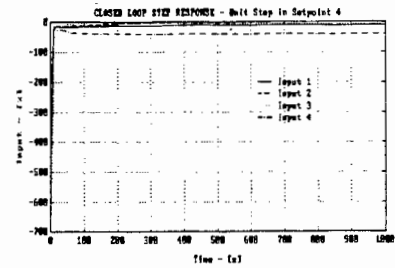


Figure 6.4.20 : INPUT
(Step Setpoint 4)

It might seem curious as to why a Γ value less than one is chosen (see equation 5.9). If a limiting value greater than one is found it means that the weights are too restrictive for the solution of equation 5.9, whereas if a positive fractional value less than one is the optimal case then it indicates that the weights are too relaxed and a solution can be found for a more constrained case.

From the principle gain plots of $T(s)$ it can be seen that increasing the value of W_2^{-1} has resulted in the principle gains being much closer together. This means that there is very little variation of the loops and each behaves as if they were single variable loops³. The lack of interaction is also evident in the time simulations.

The performance advantages gain from the first design have not been lost and the response time has remained the same for the case of the minimum value of Γ .

The input action has now become a problem. The initial spikes in input (see Figures 6.4.17 to 6.4.20) have become larger than for the first design. These values, greater than $\pm 100\%$, are outside the allowable input space, which is an unacceptable design.

The next design will downgrade the performance, speed of response, in an attempt to have a closed loop design with acceptable input action.

6.5 : DESIGN NUMBER 3

Once again the advantages gained from the previous two designs must not be lost when improving on them. This section will slow the response time down in an attempt to have the input functions as more acceptable continuous function.

This was done by choosing the closed loop bandwidth to be about 0.004 rad/s, thus giving a closed loop time constant in the region of 250s. It is felt that any further downgrading of the closed loop response time will result in a system that is unacceptably slow when compared to the open loop time constants.

Table 6.3 : SPECIFICATIONS FOR WEIGHTS IN DESIGN 3

	Gain [dB]	wp [rad/s]	ws [rad/s]	A _{MAX} [dB]	A _{MIN} [dB]
(W1 ⁻¹) ₁₁	-60	3.0E-6	3.0E-3	4	60
(W1 ⁻¹) ₂₂	-60	3.0E-6	3.0E-3	4	60
(W1 ⁻¹) ₃₃	-60	3.0E-6	3.0E-3	4	60
(W1 ⁻¹) ₄₄	-60	3.0E-6	3.0E-3	4	60
(W2 ⁻¹) ₁₁	60	—	—	—	—
(W2 ⁻¹) ₂₂	60	—	—	—	—
(W2 ⁻¹) ₃₃	60	—	—	—	—
(W2 ⁻¹) ₄₄	60	—	—	—	—
(W3 ⁻¹) ₁₁	-60	5.0E+0	5.0E-3	4	60
(W3 ⁻¹) ₂₂	-60	5.0E+0	5.0E-3	4	60
(W3 ⁻¹) ₃₃	-60	5.0E+0	5.0E-3	4	60
(W3 ⁻¹) ₄₄	-60	5.0E+0	5.0E-3	4	60

6.5.1 : SYNTHESIS STAGE 1 ($\Gamma = 1E6$)

An initial value of $1E6$ was chosen for Γ . This high value results a controller that has 14 states, the same number of states that the augmented plant $P(s)$ has.

The principle gains of weighting functions and the performance functions that result from the controller designed are displayed below.

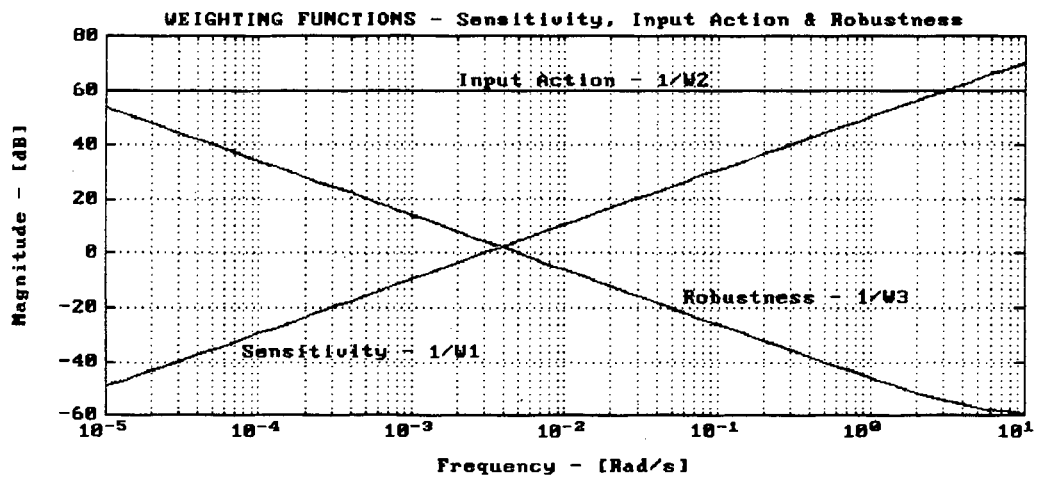


Figure 6.5.1 : WEIGHTING FUNCTIONS (DESIGN 3.1)

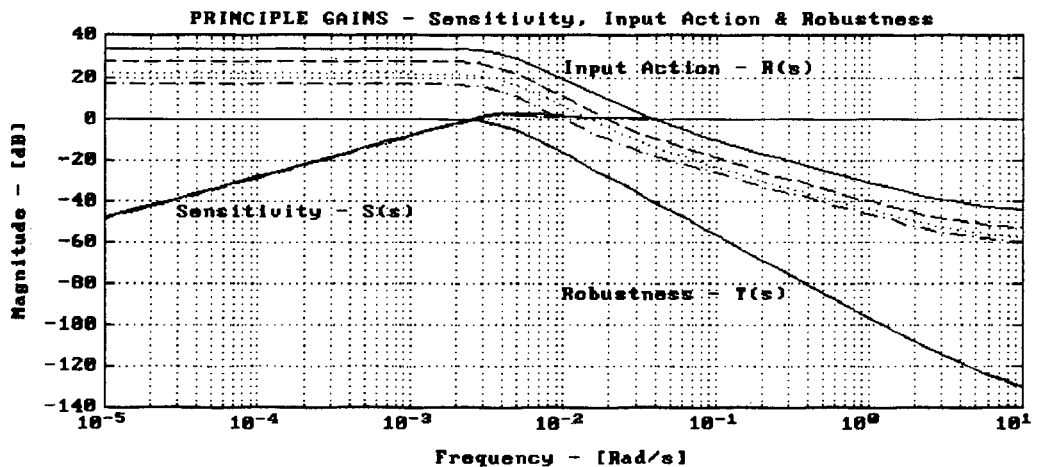


Figure 6.5.2 : PERFORMANCE FUNCTIONS (DESIGN 3.1)

The output and input functions for steps in the four setpoints are displayed in the figures that follow.

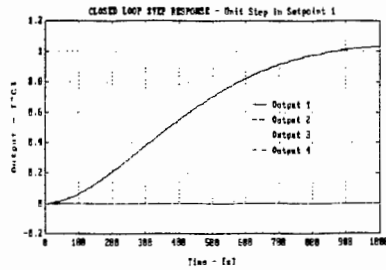


Figure 6.5.3 : OUTPUT
(Step Setpoint 1)

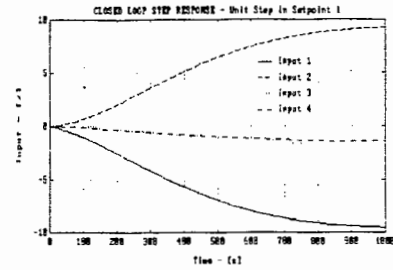


Figure 6.5.7 : INPUT
(Step Setpoint 1)

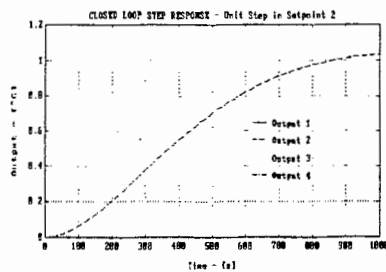


Figure 6.5.4 : OUTPUT
(Step Setpoint 2)

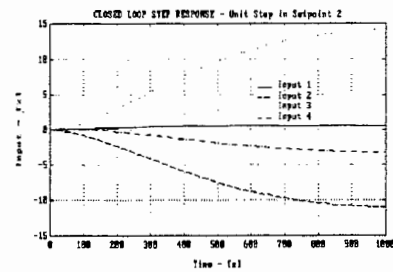


Figure 6.5.8 : INPUT
(Step Setpoint 2)

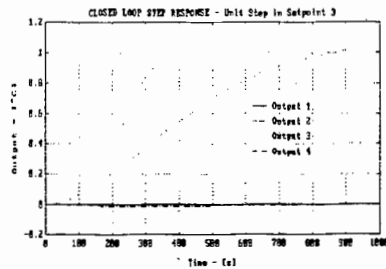


Figure 6.5.5 : OUTPUT
(Step Setpoint 3)

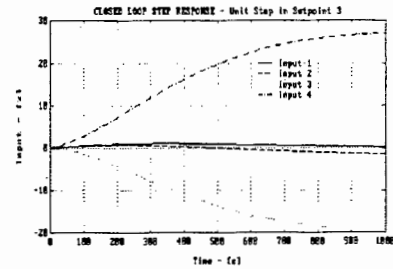


Figure 6.5.9 : INPUT
(Step Setpoint 3)

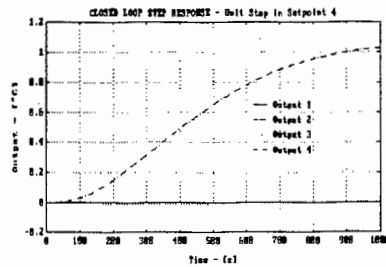


Figure 6.5.6 : OUTPUT
(Step Setpoint 4)

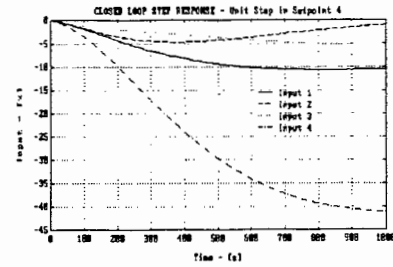


Figure 6.5.10 : INPUT
(Step Setpoint 4)

6.5.2 : SYNTHESIS STAGE 2 ($\Gamma = 0.85$)

The value of Γ was reduced until the minimum value was found, $\Gamma = 0.85$. This gives a controller with 10 states, the optimal number of states for this choice of weighting functions.

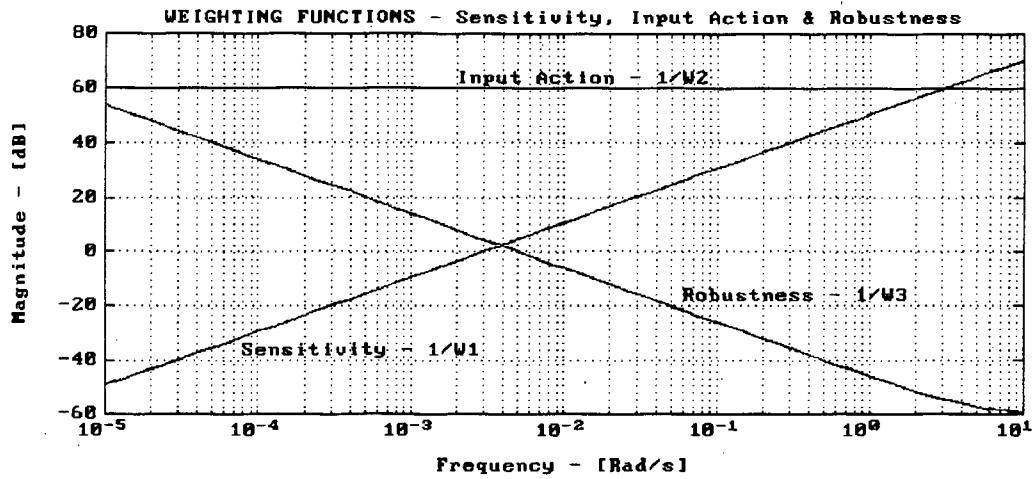


Figure 6.5.11 : WEIGHTING FUNCTIONS (DESIGN 3.2)

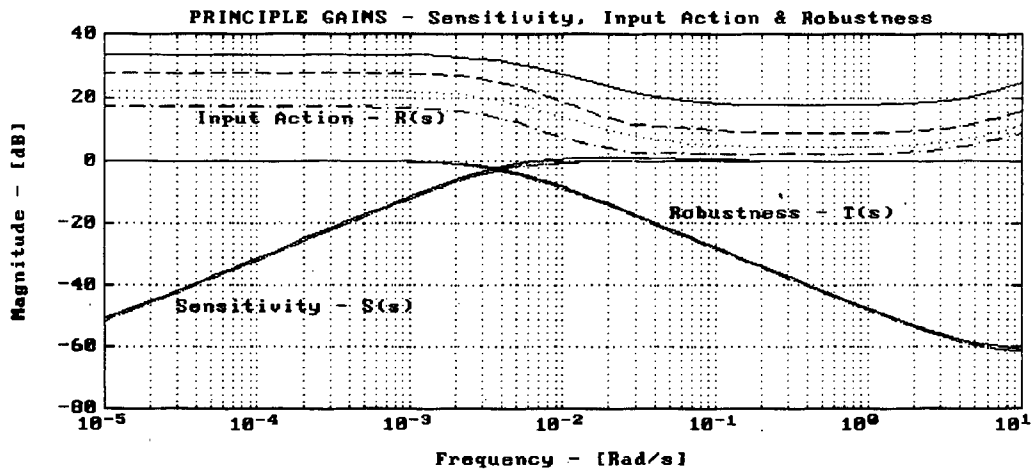


Figure 6.5.12 : PERFORMANCE FUNCTIONS (DESIGN 3.2)

The output and input functions for steps in the four setpoints are displayed in the figures that follow.

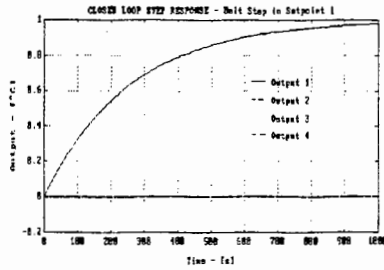


Figure 6.5.13 : OUTPUT
(Step Setpoint 1)

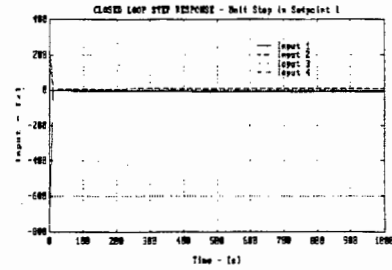


Figure 6.5.17 : INPUT
(Step Setpoint 1)

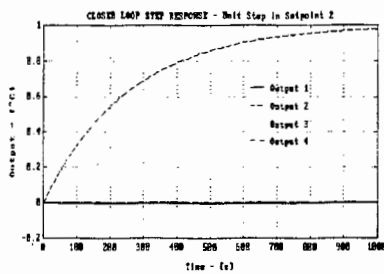


Figure 6.5.14 : OUTPUT
(Step Setpoint 2)

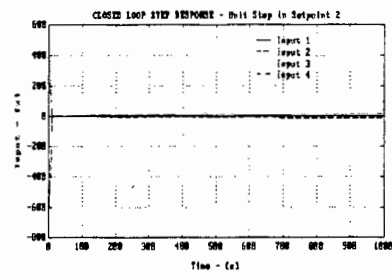


Figure 6.5.18 : INPUT
(Step Setpoint 2)

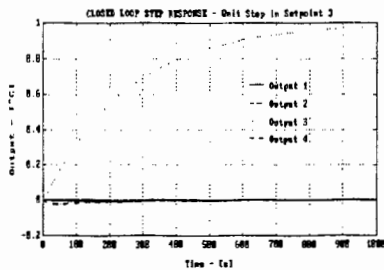


Figure 6.5.15 : OUTPUT
(Step Setpoint 3)

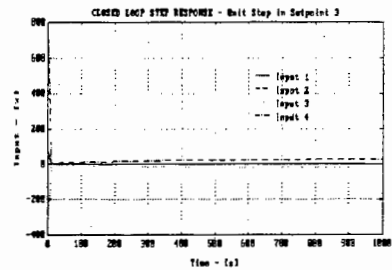


Figure 6.5.19 : INPUT
(Step Setpoint 3)

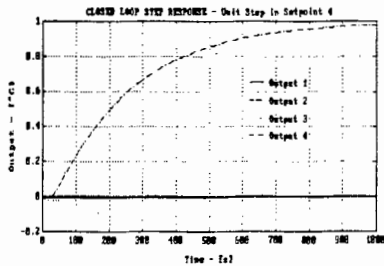


Figure 6.5.16 : OUTPUT
(Step Setpoint 4)

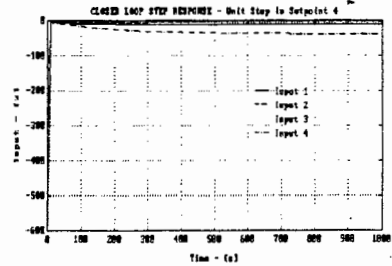


Figure 6.5.20 : INPUT
(Step Setpoint 4)

The design in synthesis stage 1 shows a very much slower response than was designed. But as is expected when the optimal value of Γ is found the system responds with the desired speed (time constant of around 250s). The principle gain plots show that the previously achieved levels of disturbance rejection and robustness have not been lost. Setpoint tracking has also been retained.

From the time responses of the inputs it is evident that slowing the response time down has not affected the input action at all. This is due to the high frequency action (see Figure 6.5.12) of $R(s)$ not being limited. The weight W_2^{-1} is a constant and thus does not have a state space realisation (matrices A, B and C are all null). This results in no frequency dependent characteristics being imposed on the input function $R(s)$.

The final design will use a frequency dependant weight W_2^{-1} to eliminate the high frequency spikes.

6.6 : DESIGN NUMBER 4

Another problem generated by the controller spikes are large numerical values in the state space matrices of the controller. This creates a system whose solution is dependent on a very small sampling interval. Thus it is reasoned that by eliminating the violent input action, the numerical values in the controller will become acceptably small to solve in real time.

From the previous graphs of input action, it is seen that the predominating feature is a high frequency spike at the time of a step. Thus if a frequency limitation can be placed on the input action, these high frequency spikes can be eliminated.

This can be done by designing the weighting function w_2^{-1} with the same frequency dependent characteristics as weight w_3^{-1} has, then the weight w_2^{-1} will attenuate the high frequencies of the input signal and a smoother input time response should result.

Table 6.4 : SPECIFICATIONS FOR WEIGHTS IN DESIGN 4

	Gain [dB]	w_p [rad/s]	w_s [rad/s]	A_{MAX} [dB]	A_{MIN} [dB]
$(w_1^{-1})_{11}$	-60	3.0E-6	3.0E-3	4	60
$(w_1^{-1})_{22}$	-60	3.0E-6	3.0E-3	4	60
$(w_1^{-1})_{33}$	-60	3.0E-6	3.0E-3	4	60
$(w_1^{-1})_{44}$	-60	3.0E-6	3.0E-3	4	60
$(w_2^{-1})_{11}$	-60	5.0E+2	5.0E-1	4	60
$(w_2^{-1})_{22}$	-60	5.0E+2	5.0E-1	4	60
$(w_2^{-1})_{33}$	-60	5.0E+2	5.0E-1	4	60
$(w_2^{-1})_{44}$	-60	5.0E+2	5.0E-1	4	60
$(w_3^{-1})_{11}$	-60	5.0E+0	5.0E-3	4	60
$(w_3^{-1})_{22}$	-60	5.0E+0	5.0E-3	4	60
$(w_3^{-1})_{33}$	-60	5.0E+0	5.0E-3	4	60
$(w_3^{-1})_{44}$	-60	5.0E+0	5.0E-3	4	60

6.6.1 : SYNTHESIS STAGE 1 ($\Gamma = 1E6$)

An initial value of $1E6$ was chosen for Γ . This high value results a controller that has 18 states, the same number of states that the augmented plant $P(s)$ has.

The principle gains of weighting functions and the performance functions that result from the controller designed are displayed below.

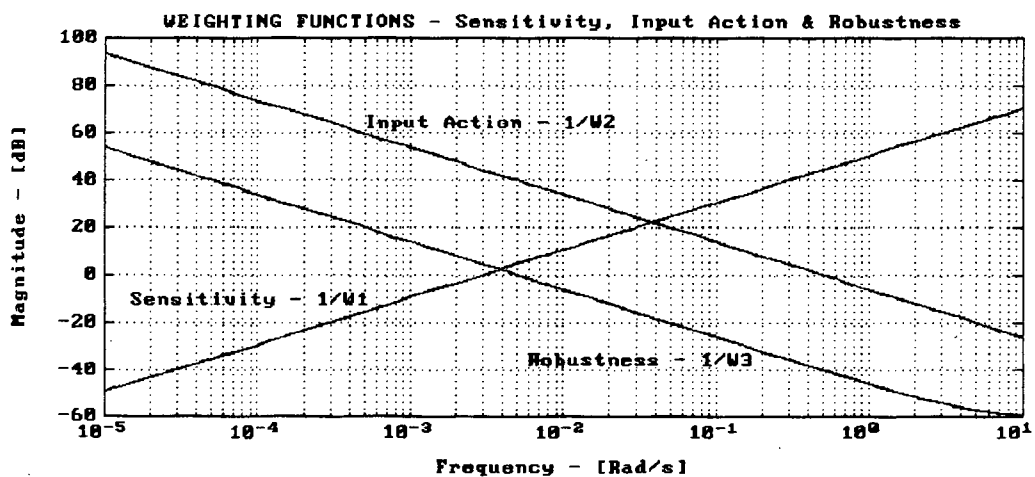


Figure 6.6.1 : WEIGHTING FUNCTIONS (DESIGN 4.1)

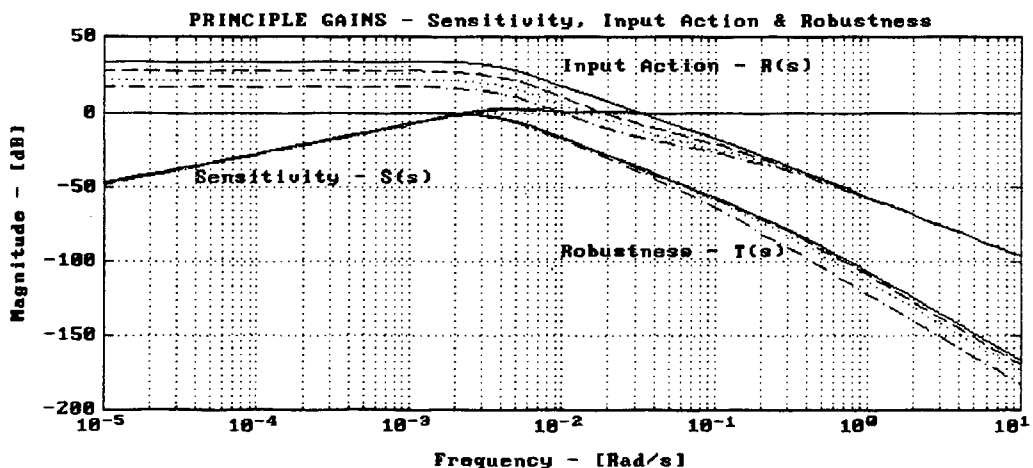


Figure 6.6.2 : PERFORMANCE FUNCTIONS (DESIGN 4.1)

The output and input functions for steps in the four setpoints are displayed in the figures that follow.

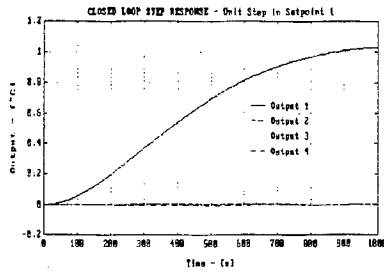


Figure 6.6.3 : OUTPUT
(Step Setpoint 1)

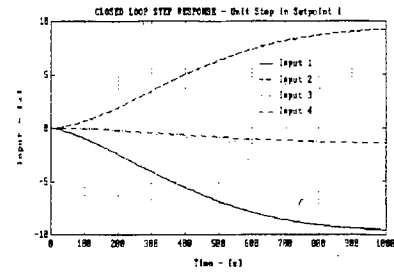


Figure 6.6.7 : INPUT
(Step Setpoint 1)

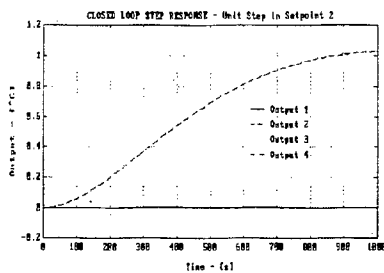


Figure 6.6.4 : OUTPUT
(Step Setpoint 2)

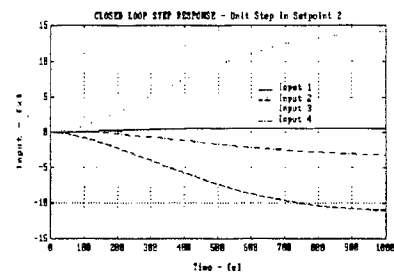


Figure 6.6.8 : INPUT
(Step Setpoint 2)

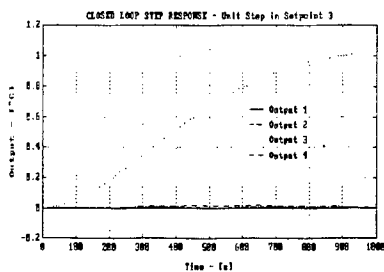


Figure 6.6.5 : OUTPUT
(Step Setpoint 3)

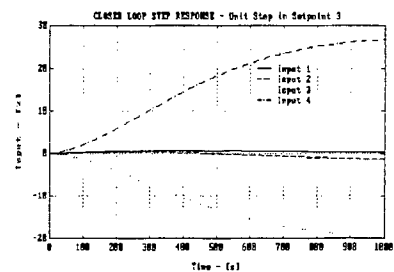


Figure 6.6.9 : INPUT
(Step Setpoint 3)

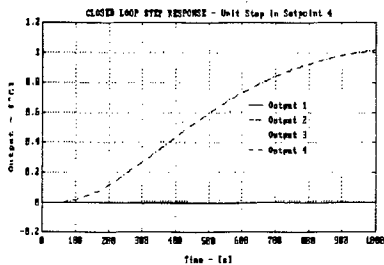


Figure 6.6.6 : OUTPUT
(Step Setpoint 4)

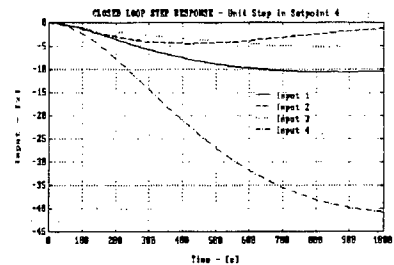


Figure 6.6.10 : INPUT
(Step Setpoint 4)

6.6.2 : SYNTHESIS STAGE 2 ($\Gamma = 0.9$)

The value of Γ was reduced until the minimum value was found, $\Gamma = 0.9$. This gives a controller with 14 states, the optimal number of states for this choice of weighting functions.

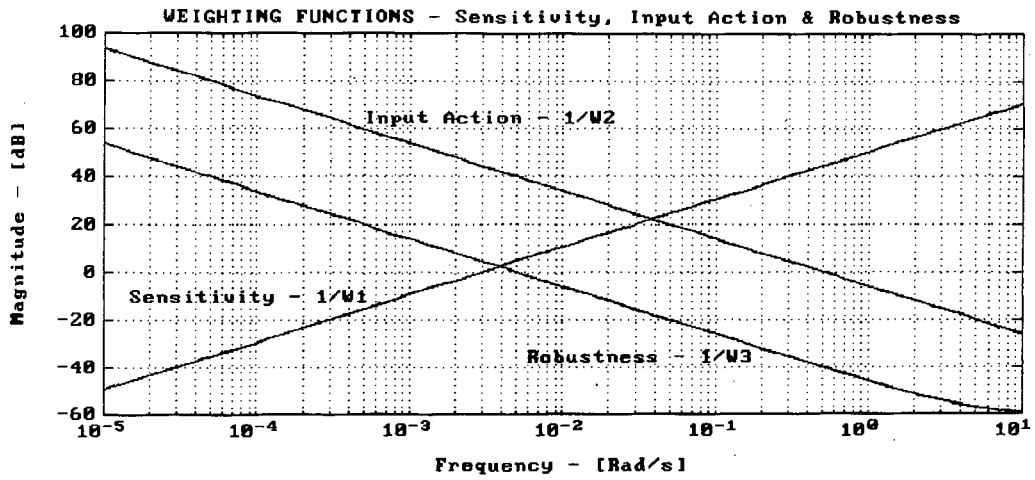


Figure 6.6.11 : WEIGHTING FUNCTIONS (DESIGN 4.2)

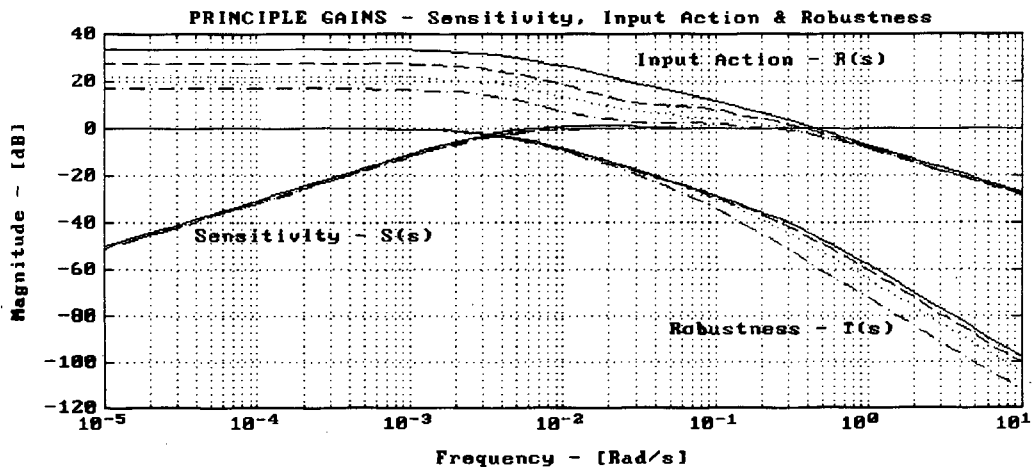


Figure 6.6.12 : PERFORMANCE FUNCTIONS (DESIGN 4.2)

The output and input functions for steps in the four setpoints are displayed in the figures that follow.

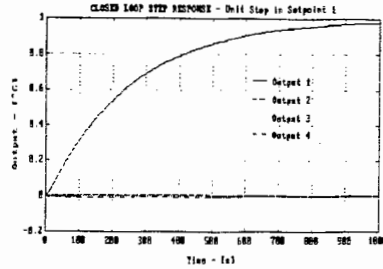


Figure 6.6.13 : OUTPUT
(Step Setpoint 1)

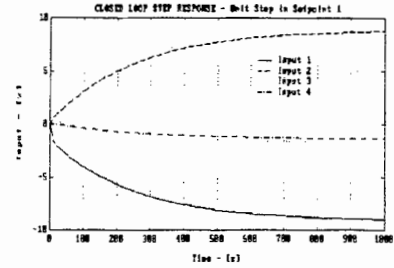


Figure 6.6.17 : INPUT
(Step Setpoint 1)

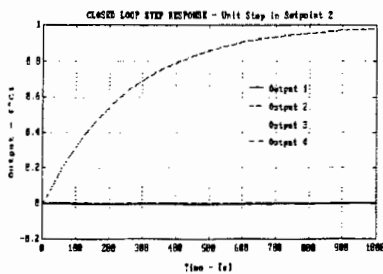


Figure 6.6.14 : OUTPUT
(Step Setpoint 2)

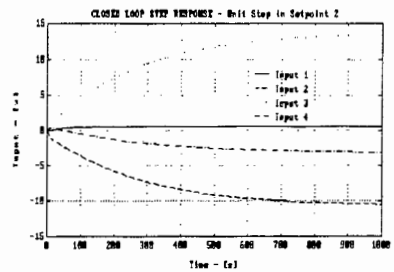


Figure 6.6.18 : INPUT
(Step Setpoint 2)

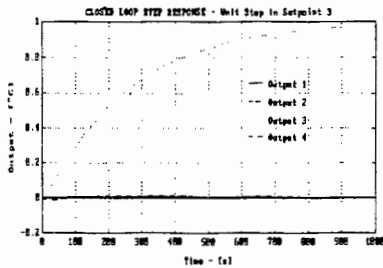


Figure 6.6.15 : OUTPUT
(Step Setpoint 3)

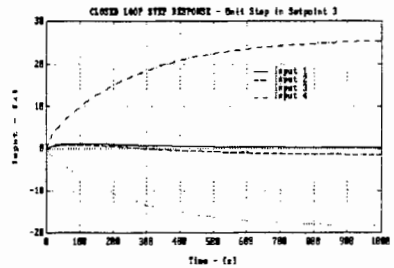


Figure 6.6.19 : INPUT
(Step Setpoint 3)

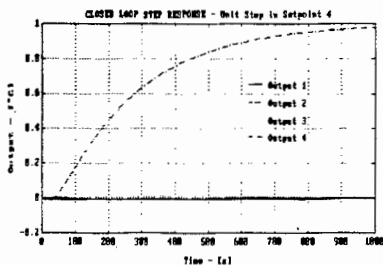


Figure 6.6.16 : OUTPUT
(Step Setpoint 4)

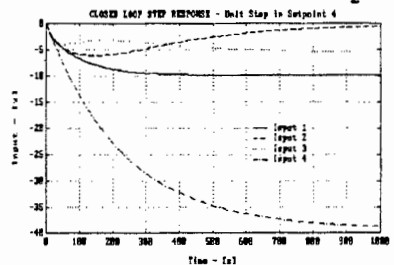


Figure 6.6.20 : INPUT
(Step Setpoint 4)

The time responses show that the input action is now a smooth function. The principle gain plot of $S(s)$, $R(s)$ and $T(s)$ all below the designed frequency weighted functions, thus eliminating the unexplained problem of the first design. The input action $R(s)$ also shows the expected attenuation of high frequency input action. The trade-off that has resulted from this smooth function is that the augmented plant now has four extra states, which results in a controller of 14 states.

The design procedure could continue to make one or more of the loops faster, but this was not done since it was found that making the loops faster resulted in a controller that could once more not be implemented due to an unachievable real time sampling interval. The controller as designed in this section was tested and found to work with the time step available on the computer dedicated to the HERIG. The results of closed loop step tests performed using this controller will be presented in the next section of this chapter.

This design facet of limiting input action has not been discussed in any of the literature to date. The limitations of input action is a typical problem encountered on industrial plants and this method of designing for such input action limitation is seen as a further step in the acceptance of H_{∞} as a method of designing multivariable controllers for industrial applications.

6.7 : IMPLEMENTATION OF THE CONTROLLER

The controller, $K_T(s)$, from synthesis phase 2 of design number 4 was chosen to be implemented on the HERIG. It has 14 states and the H_∞ synthesis guarantees it to be stable.

As was discussed by Craig⁴ the H_∞ synthesis places zeros at the augmented plant pole positions and poles at the augmented plant zeros (not right half plane zeros though). It then places poles in the controller to achieve the desired performance conditions.

Poles of $K_T(s)$:

-5.0
-5.0
-5.0
-5.0
-0.325
-0.250
-0.165
-0.060 $\pm j$ 0.0185
-0.0094
-0.000003
-0.000003
-0.000003 $\pm j$ 0.00000003

As expected there is a controller pole near the augmented plant zero position of -0.0097. There is also no controller pole at the non-minimum phase zero position, else this would give an unstable controller.

Zeros of $K_T(s)$:

-500.0
-500.0
-500.0
-500.0
-5.0
-5.0
-5.0
-5.0
-0.1124
-0.0335
-0.0194 $\pm j$ 0.0171
-0.0113 $\pm j$ 0.0036

The full order augmented plant has 18 states. The reduced order model eliminates four of them and a system with 14 states is used. As it turns out the states that are removed are those relating to weight W_1 . The controller is now expected to have zeros in the pole positions of the reduced augmented plant, now comprising poles from $G_T(s)$, W_2 and W_3 .

The four controller zeros at -500 cancel out the plant poles resulting from weight W_2 . Similarly the zeros at -5 cancel out the poles from weight W_3 . The last six controller zeros cancel the poles of the reduced order plant $G_T(s)$.

Thus it can be seen that the H_∞ synthesis works as expected. It is now left to verify that the implementation of the controller on the HERIG is as expected.

The results of closed loop step tests on the HERIG are now presented.

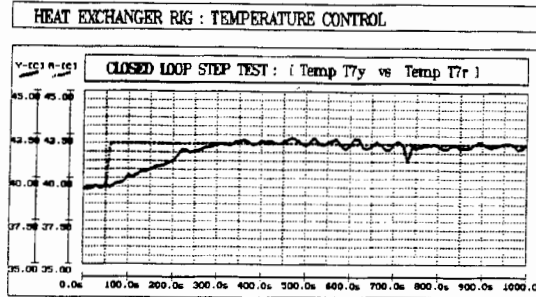


Figure 6.7.1 : TEMPERATURE
H11(S)

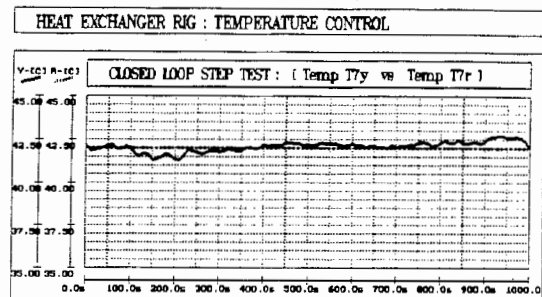


Figure 6.7.5 : TEMPERATURE
H12(S)

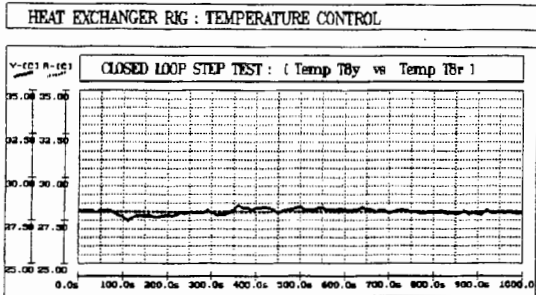


Figure 6.7.2 : TEMPERATURE
H21(S)

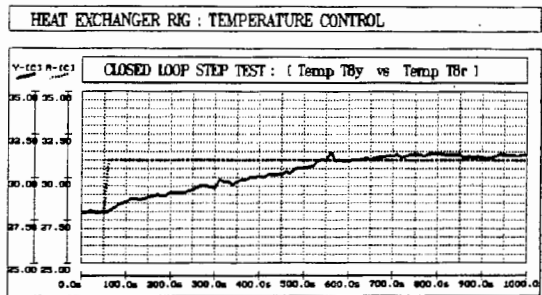


Figure 6.7.6 : TEMPERATURE
H22(S)

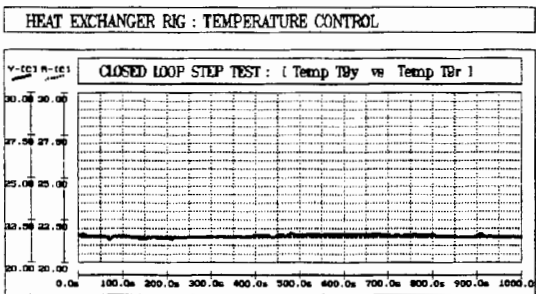


Figure 6.7.3 : TEMPERATURE
H31(S)

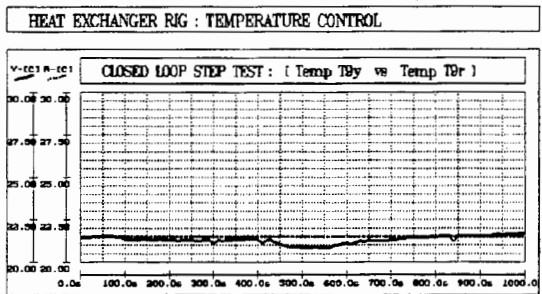


Figure 6.7.7 : TEMPERATURE
H32(S)

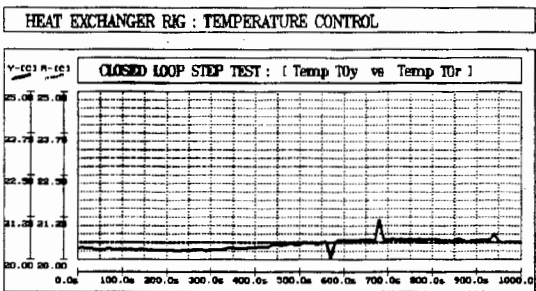


Figure 6.7.4 : TEMPERATURE
H41(S)

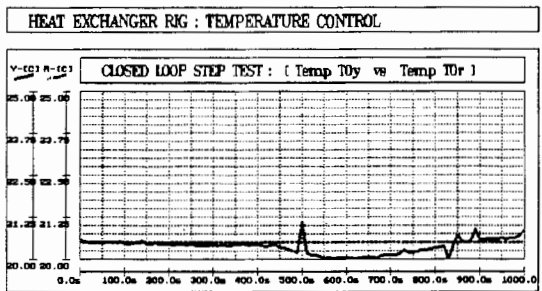


Figure 6.7.8 : TEMPERATURE
H42(S)

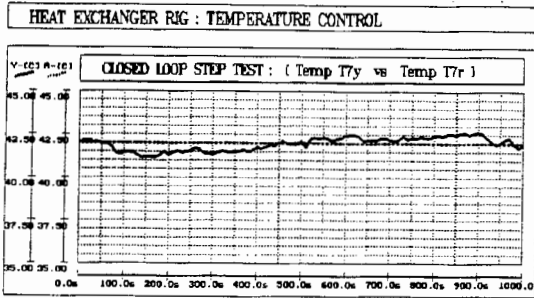


Figure 6.7.9 : TEMPERATURE
H13(S)

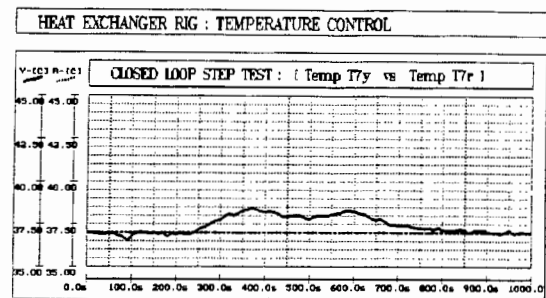


Figure 6.7.13 : TEMPERATURE
H14(S)

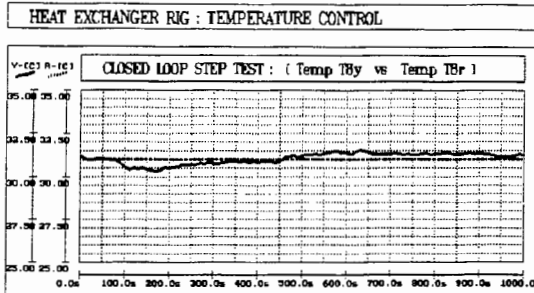


Figure 6.7.10 : TEMPERATURE
H23(S)

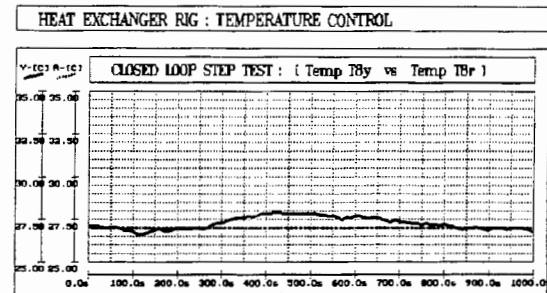


Figure 6.7.14 : TEMPERATURE
H24(S)

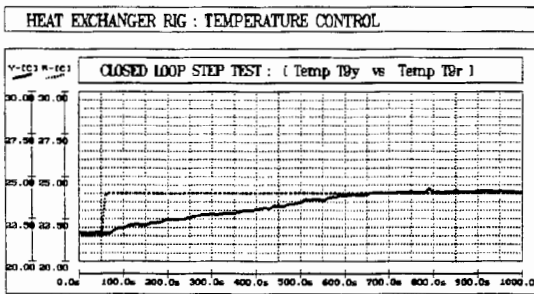


Figure 6.7.11 : TEMPERATURE
H33(S)

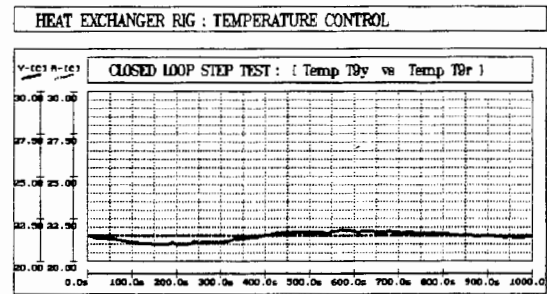


Figure 6.7.15 : TEMPERATURE
H34(S)

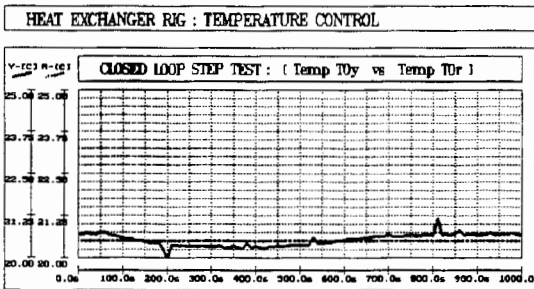


Figure 6.7.12 : TEMPERATURE
H43(S)

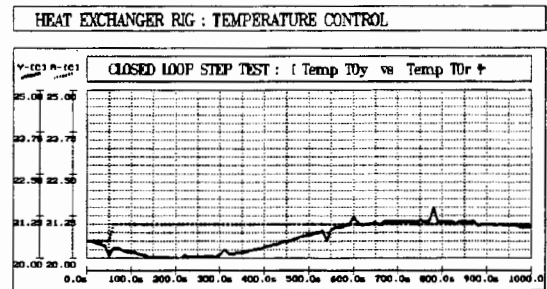


Figure 6.7.16 : TEMPERATURE
H44(S)

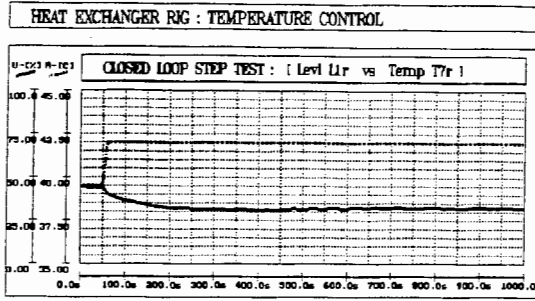


Figure 6.7.17 : TEMPERATURE
U11(S)

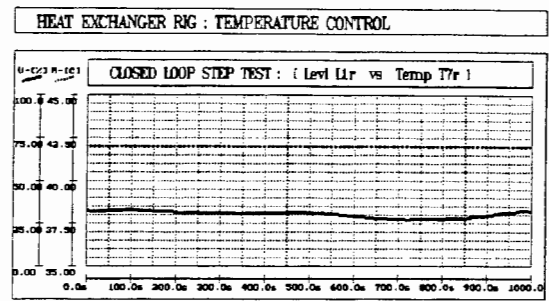


Figure 6.7.21 : TEMPERATURE
U12(S)

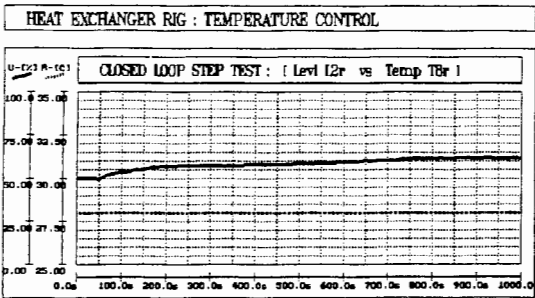


Figure 6.7.18 : TEMPERATURE
U21(S)

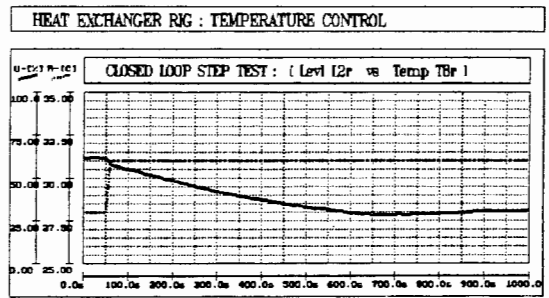


Figure 6.7.22 : TEMPERATURE
U22(S)

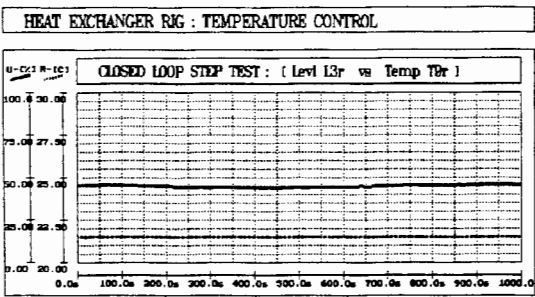


Figure 6.7.19 : TEMPERATURE
U31(S)

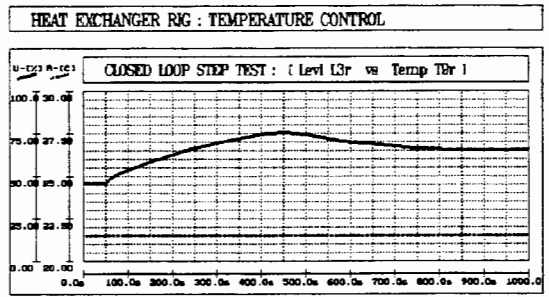


Figure 6.7.23 : TEMPERATURE
U32(S)

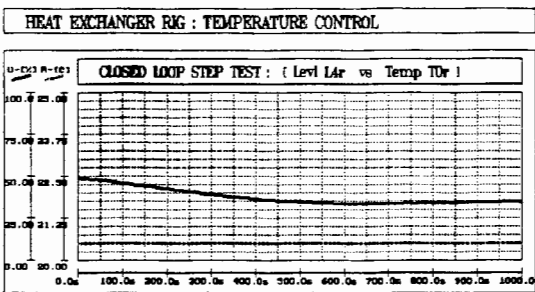


Figure 6.7.20 : TEMPERATURE
U41(S)

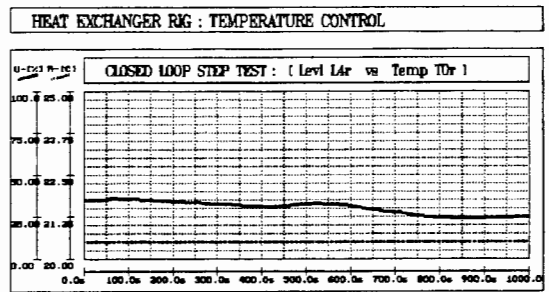


Figure 6.7.24 : TEMPERATURE
U42(S)

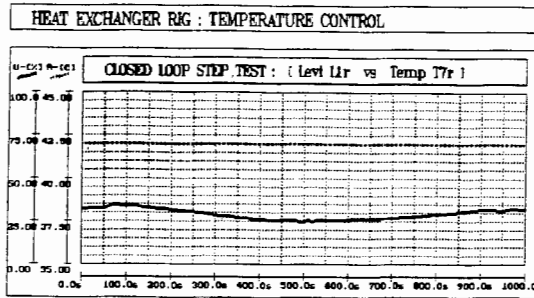


Figure 6.7.25 : TEMPERATURE
U13(S)

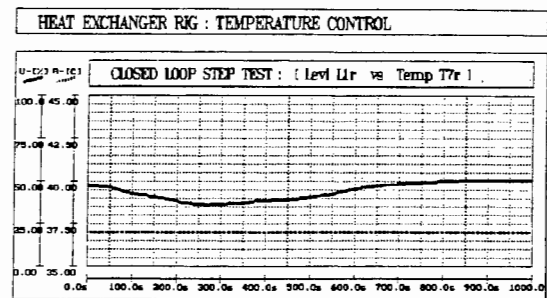


Figure 6.7.29 : TEMPERATURE
U14(S)

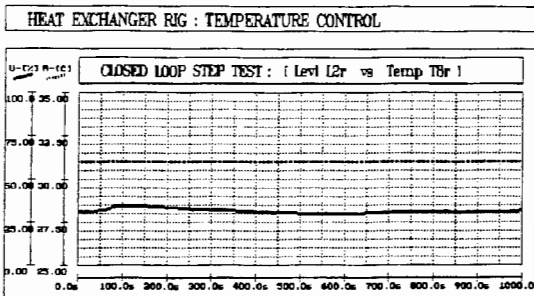


Figure 6.7.26 : TEMPERATURE
U23(S)

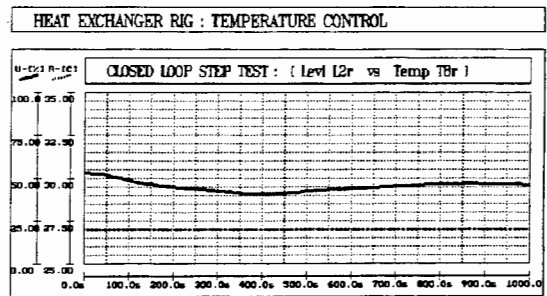


Figure 6.7.30 : TEMPERATURE
U24(S)

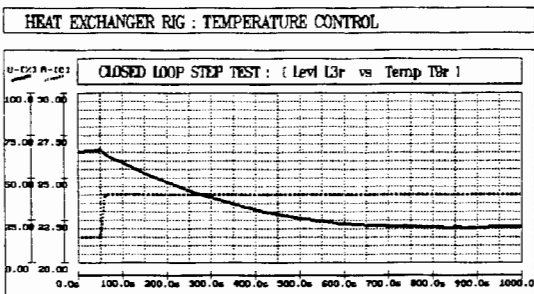


Figure 6.7.27 : TEMPERATURE
U33(S)

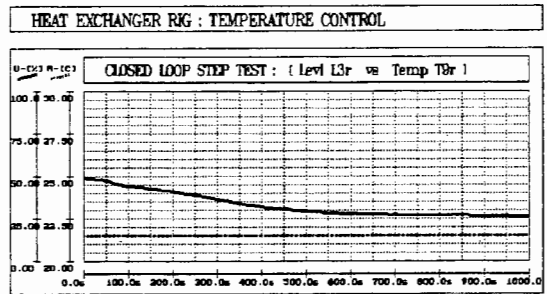


Figure 6.7.31 : TEMPERATURE
U34(S)

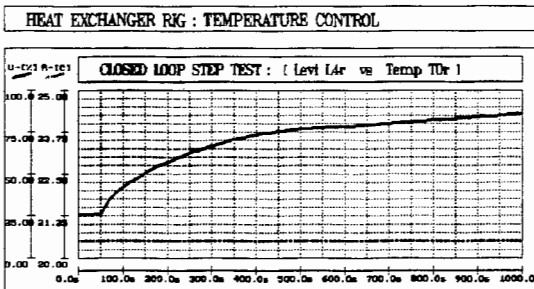


Figure 6.7.28 : TEMPERATURE
U43(S)

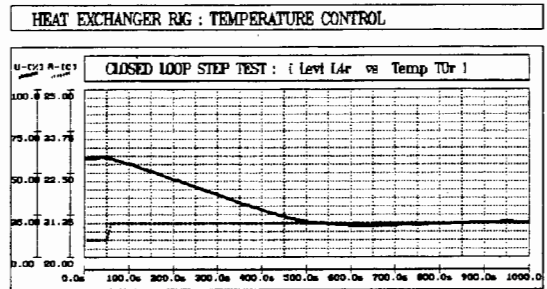


Figure 6.7.32 : TEMPERATURE
U44(S)

These step responses show that all the required stepped setpoints are tracked. In addition the setpoints of the other outputs are maintained when step tests are performed on each loop. The first three step tests were performed sequentially, but the step test on the fourth tank had to be performed separately due to limitations of the input space.

A full examination of the time responses, inputs and outputs will be discussed in the next chapter.

REFERENCES

1. CHIANG R.Y. and SAFONOV M.G., Robust Control Toolbox : for use with MATLAB, The Mathworks, pp.RR28-RR32, Jun 1988.
2. CHIANG R.Y. and SAFONOV M.G., Robust Control Toolbox : for use with MATLAB, The Mathworks, pp.RR47-RR49, Jun 1988.
3. MACIEJOWSKI J.M., Multivariable Feedback Design, Reading MA : Addison-Wesley, pp.81-87, 1989.
4. CRAIG I.K., Sensitivity of H_{∞} Controller Designs to Structured Uncertainty, M.Sc. Dissertation, Department of Aeronautics and Astronautics, Massachusetts Institute of Technology, May 1988.

CHAPTER 7 : COMPARISON OF RESULTS

The previous chapter presented sets of simulated closed loop responses for both the outputs and the inputs. Each set of graphs was for a particular controller design. Then finally a controller was chosen and implemented on the HERIG. The closed loop step test results were also presented in this chapter.

These two different sets of graphs need to be compared against each other to determine how well the designed controller behaves on the HERIG when compared to the simulated results.

7.1 : COMPARISON OF OUTPUTS

When presenting the output responses, they will be plotted against the respective setpoints which will indicate if setpoint tracking has been achieved. Further the outputs will indicate the amount of interaction there is between the loops. Finally the rise times of the on-diagonal elements of the two outputs can be compared on the same graph.

The simulated outputs from Chapter 6 needed to be level shifted to the steady state values before the setpoints were stepped. Further the simulated results needed to be scaled for the size of step performed on the HERIG, since the simulated results were all for unit steps in setpoints.

7.1.1 : STEP IN SETPOINT 1

In the following four graphs the setpoints and outputs for both the simulated and observed responses are plotted.

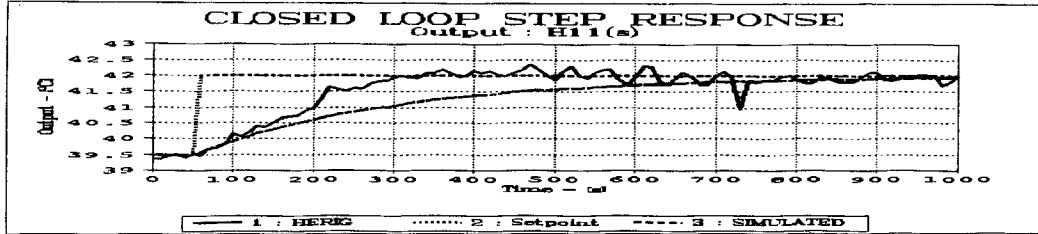


Figure 7.1.1 : OBSERVED AND SIMULATED H11(s)

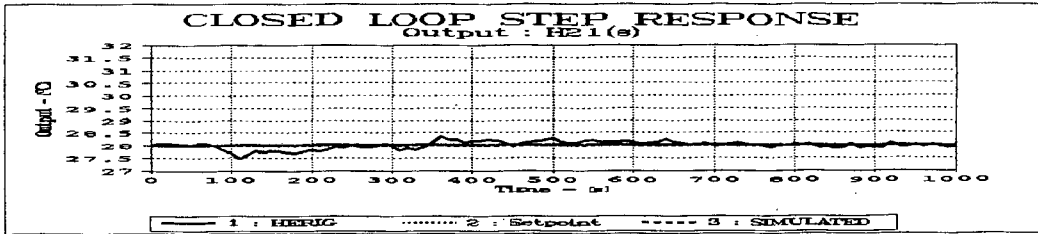


Figure 7.1.2 : OBSERVED AND SIMULATED H21(s)

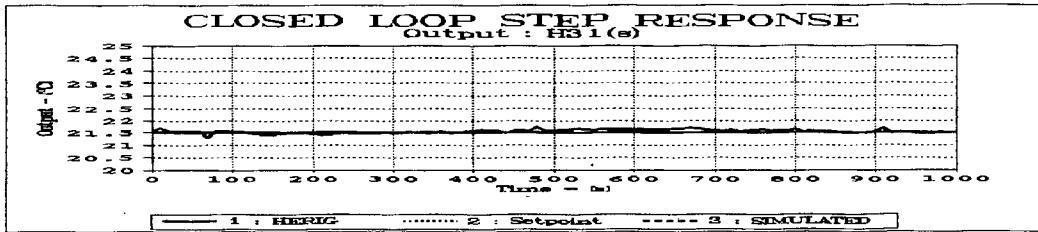


Figure 7.1.3 : OBSERVED AND SIMULATED H31(s)

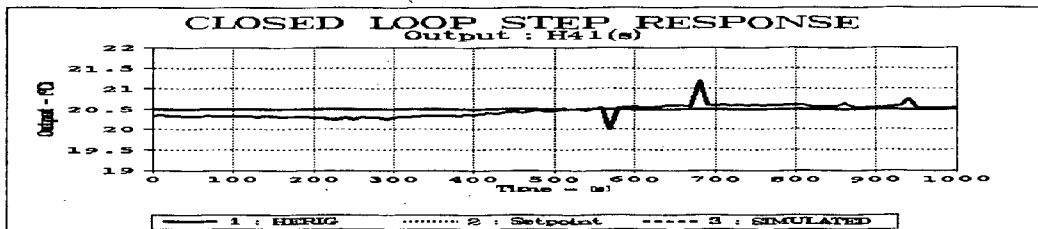


Figure 7.1.4 : OBSERVED AND SIMULATED H41(s)

7.1.2 : STEP IN SETPOINT 2

In the following four graphs the setpoints and outputs for both the simulated and observed responses are plotted.

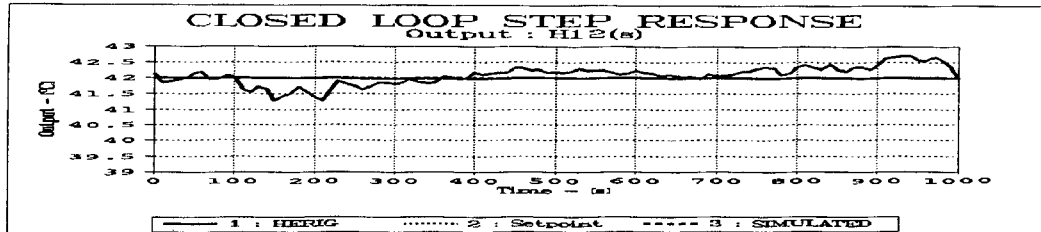


Figure 7.1.5 : OBSERVED AND SIMULATED H12(s)

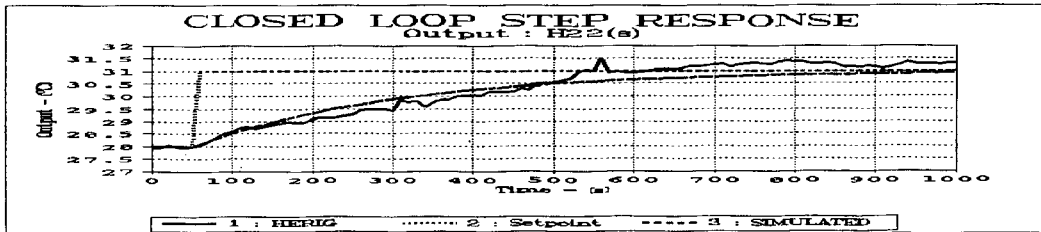


Figure 7.1.6 : OBSERVED AND SIMULATED H22(s)

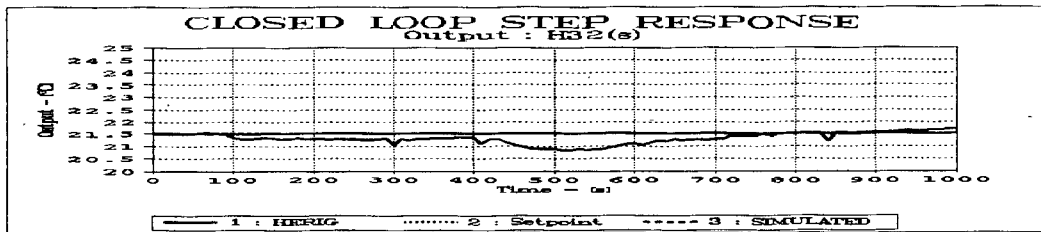


Figure 7.1.7 : OBSERVED AND SIMULATED H32(s)

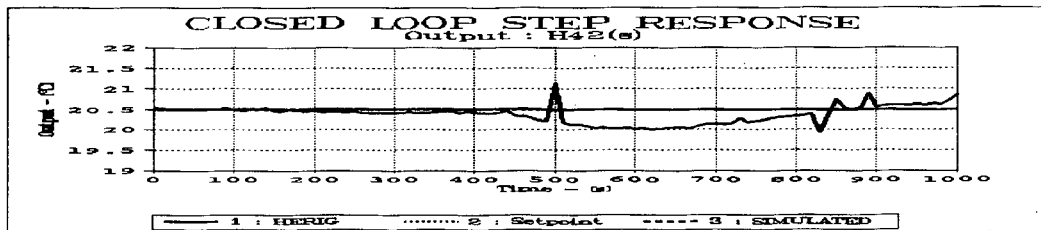


Figure 7.1.8 : OBSERVED AND SIMULATED H42(s)

7.1.3 : STEP IN SETPOINT 3

In the following four graphs the setpoints and outputs for both the simulated and observed responses are plotted.

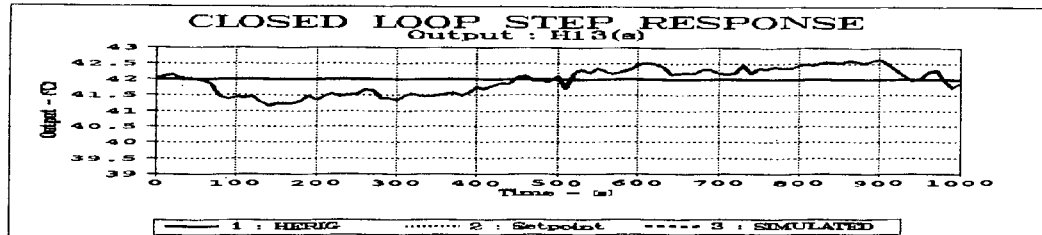


Figure 7.1.9 : OBSERVED AND SIMULATED H13(s)

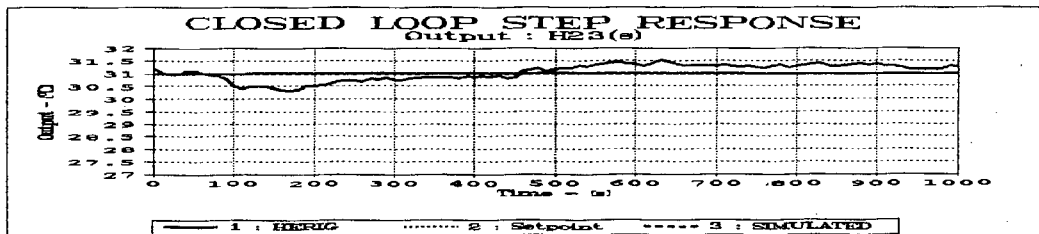


Figure 7.1.10 : OBSERVED AND SIMULATED H23(s)

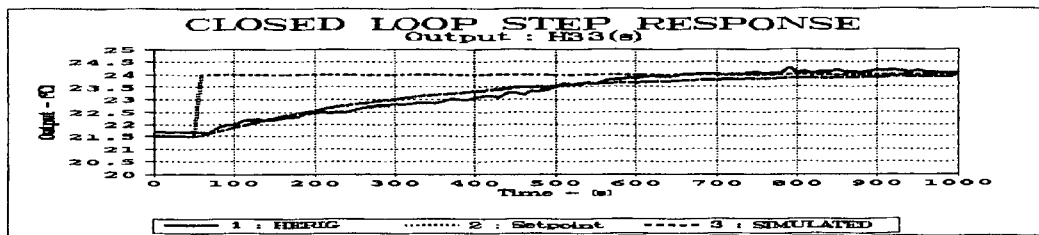


Figure 7.1.11 : OBSERVED AND SIMULATED H33(s)

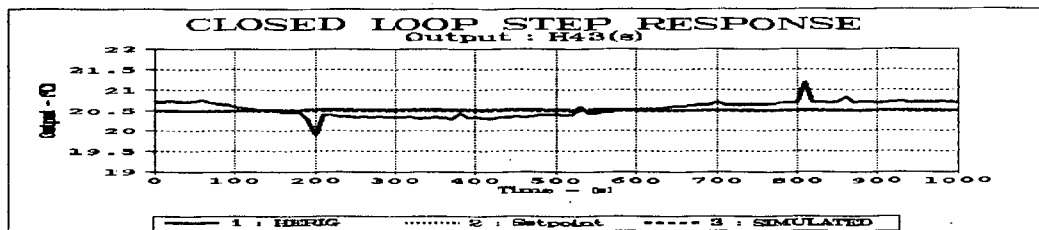


Figure 7.1.12 : OBSERVED AND SIMULATED H43(s)

7.1.4 : STEP IN SETPOINT 4

In the following four graphs the setpoints and outputs for both the simulated and observed responses are plotted.

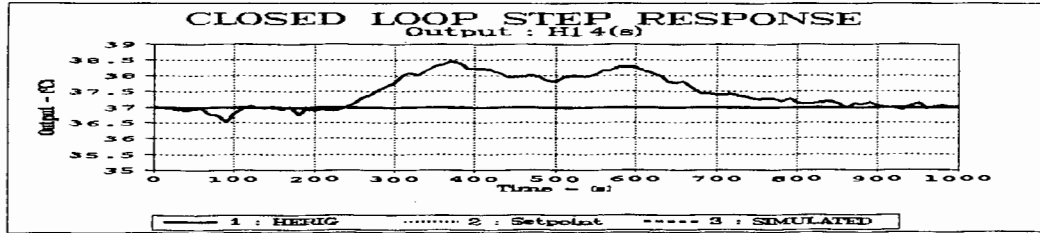


Figure 7.1.13 : OBSERVED AND SIMULATED H14(s)

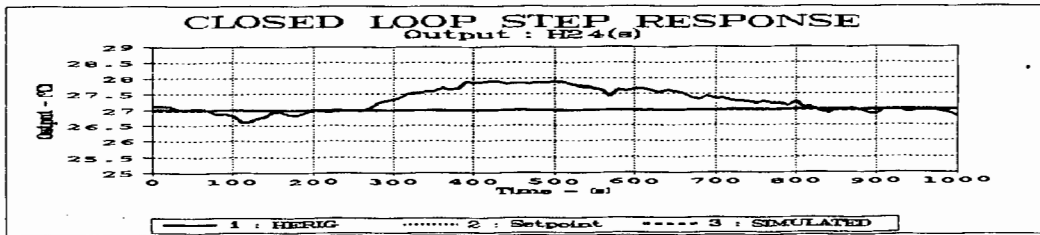


Figure 7.1.14 : OBSERVED AND SIMULATED H24(s)

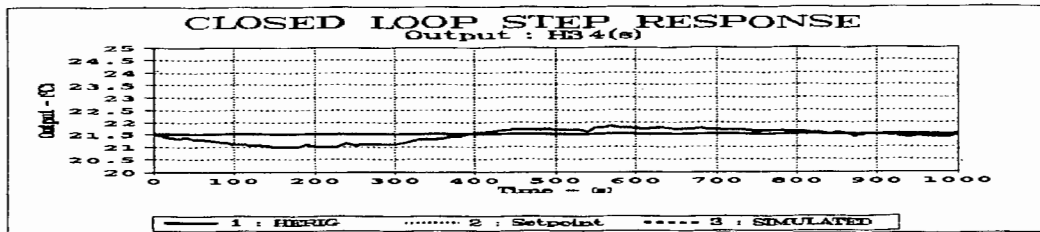


Figure 7.1.15 : OBSERVED AND SIMULATED H34(s)

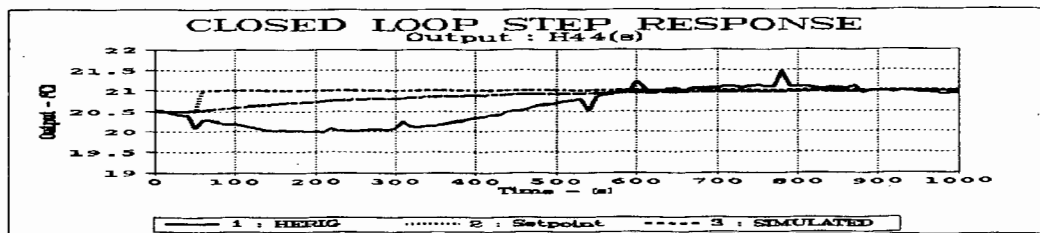


Figure 7.1.16 : OBSERVED AND SIMULATED H44(s)

The graphs show that all the on-diagonals track the setpoints, for the HERIG results even in the face of some noisy readings. The interaction between the loops was greater than expected in some cases (see section 7.1.3). The interaction appear at the same time in all four graphs and has the same characteristic and is attributed to being an external disturbance.

The response times for the second and third loops are excellent. It is seen that the first loop responds quicker than expected. This is due to the drift of the HOT reserve increasing the temperature of the water fed to the first tank, thus this is a false indication of the risetime. The fourth loop was very troublesome, due to a very small temperature drop across tank 4. Thus the response for the fourth step are not very good.

Finally it should be noted that even though the controller was implemented on a real process, that has many non-linearities in it, steady state disturbance rejection and setpoint tracking was achieved.

7.2 : COMPARISON OF INPUTS

It is not sufficient to judge a controllers performance with outputs alone. The inputs need to be checked to ensure that none of them hit any limits in achieving the required setpoints. The input functions themselves also need to be compared to check that they behave in the same manner.

Finally the inputs needs to be checked to verify that they settle down to final values and thus ensure that there are no long term closed loop instabilities.

7.2.1 : STEP IN SETPOINT 1

In the following four graphs the setpoints and outputs for both the simulated and observed responses are plotted.

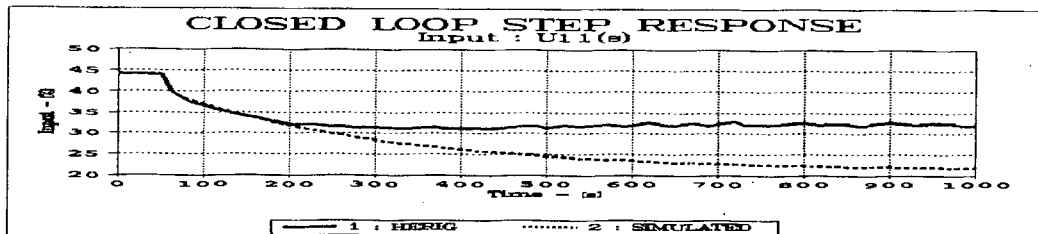


Figure 7.2.1 : OBSERVED AND SIMULATED U11(s)

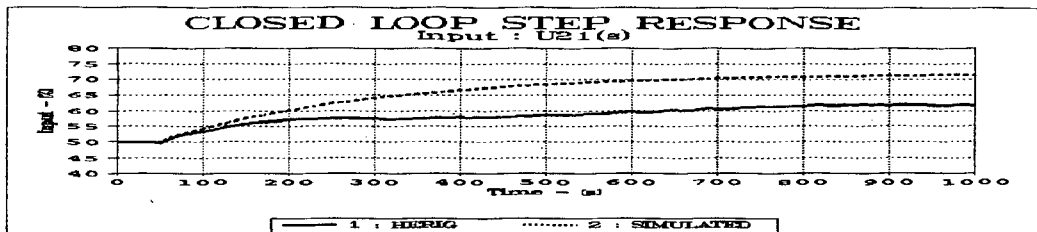


Figure 7.2.2 : OBSERVED AND SIMULATED U21(s)

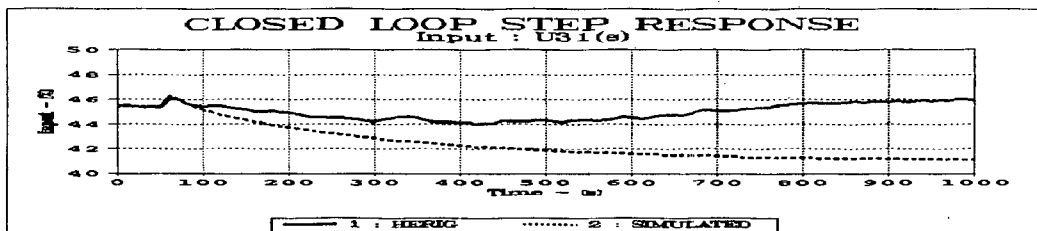


Figure 7.2.3 : OBSERVED AND SIMULATED U31(s)

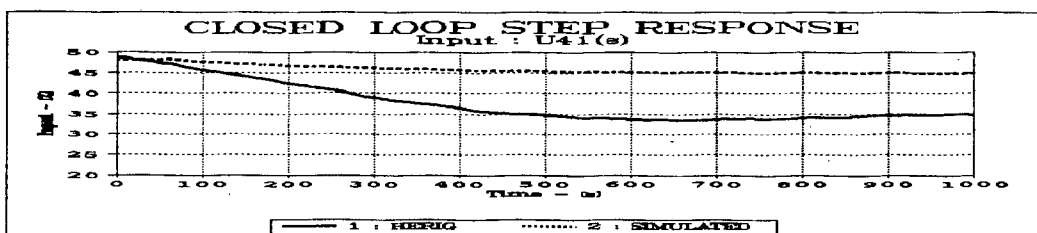


Figure 7.2.4 : OBSERVED AND SIMULATED U41(s)

7.2.2 : STEP IN SETPOINT 2

In the following four graphs the setpoints and outputs for both the simulated and observed responses are plotted.

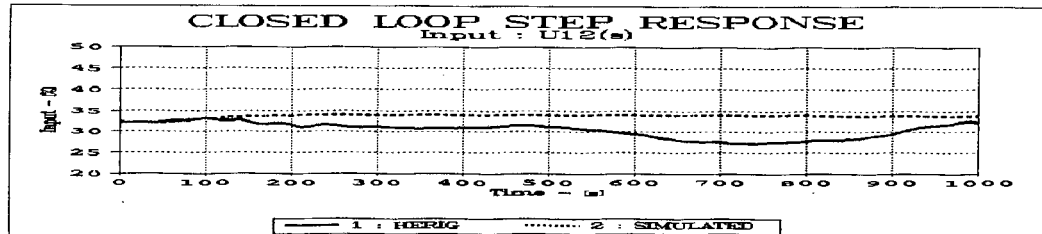


Figure 7.2.5 : OBSERVED AND SIMULATED U12(s)

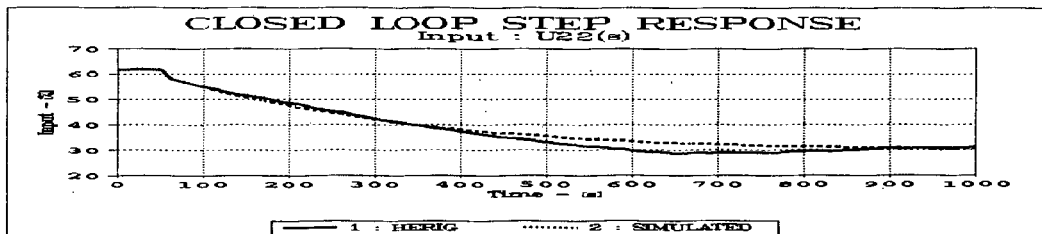


Figure 7.2.6 : OBSERVED AND SIMULATED U22(s)

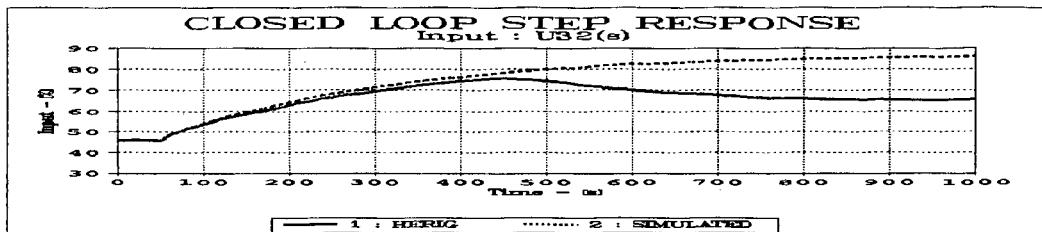


Figure 7.2.7 : OBSERVED AND SIMULATED U32(s)

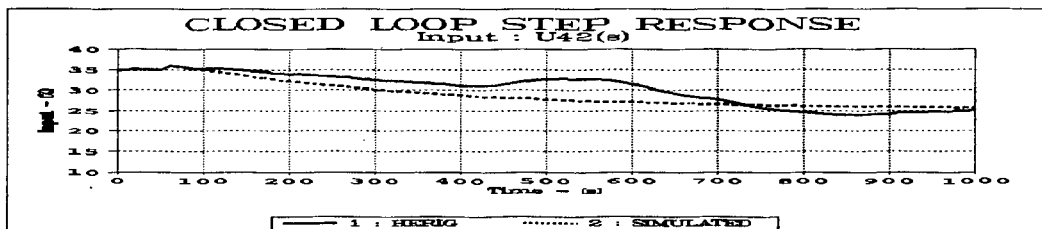


Figure 7.2.8 : OBSERVED AND SIMULATED U42(s)

7.2.3 : STEP IN SETPOINT 3

In the following four graphs the setpoints and outputs for both the simulated and observed responses are plotted.

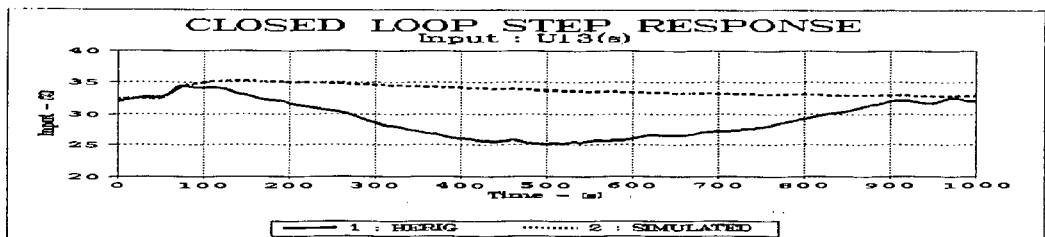


Figure 7.2.9 : OBSERVED AND SIMULATED U13(s)

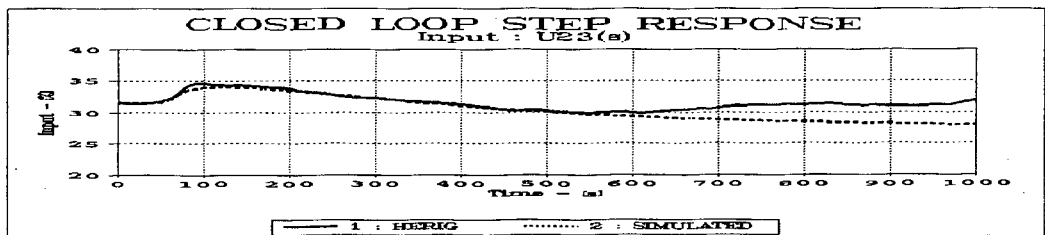


Figure 7.2.10 : OBSERVED AND SIMULATED U23(s)

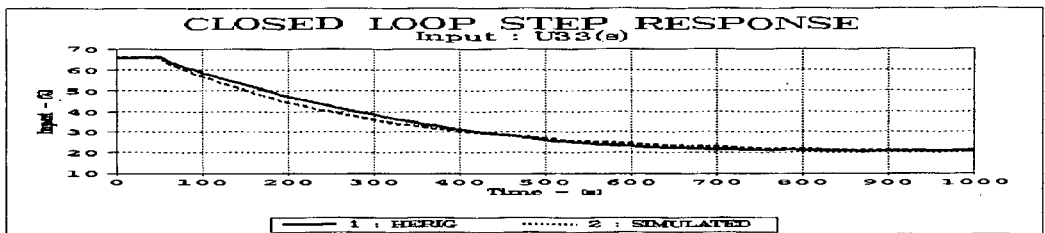


Figure 7.2.11 : OBSERVED AND SIMULATED U33(s)

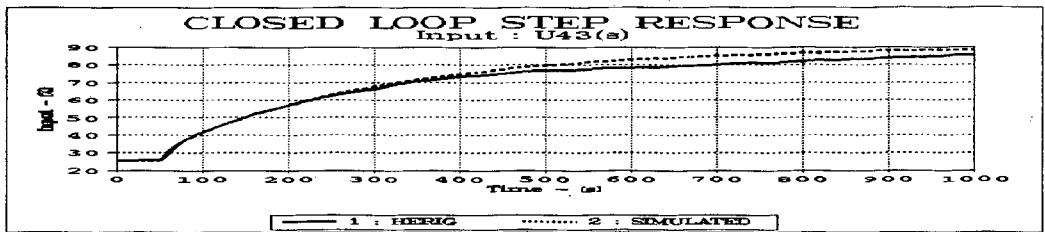


Figure 7.2.12 : OBSERVED AND SIMULATED U43(s)

7.2.4 : STEP IN SETPOINT 4

In the following four graphs the setpoints and outputs for both the simulated and observed responses are plotted.

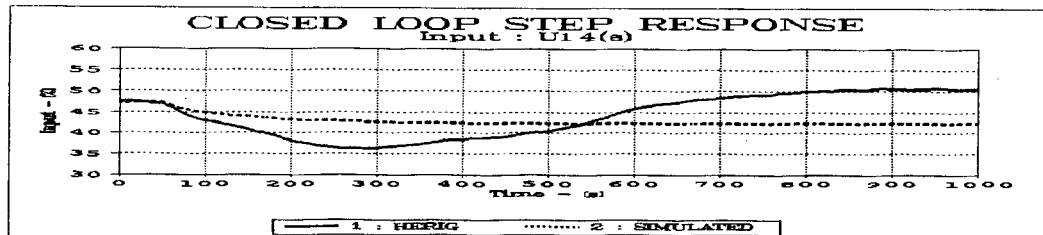


Figure 7.2.13 : OBSERVED AND SIMULATED U14(s)

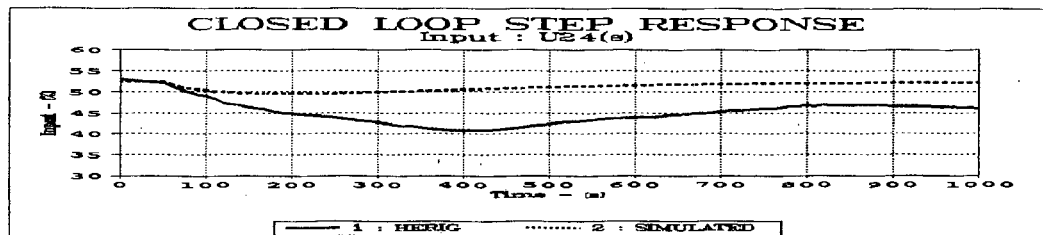


Figure 7.2.14 : OBSERVED AND SIMULATED U24(s)

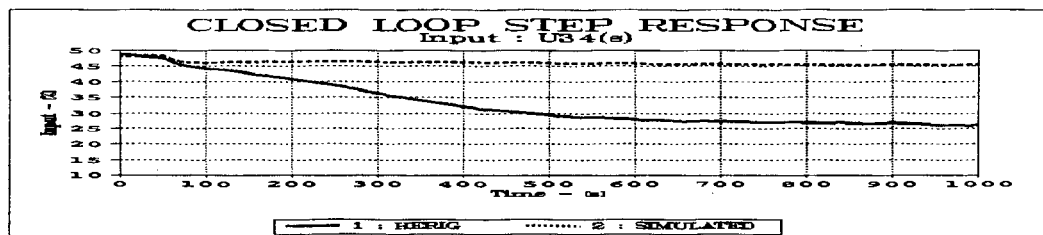


Figure 7.2.15 : OBSERVED AND SIMULATED U34(s)

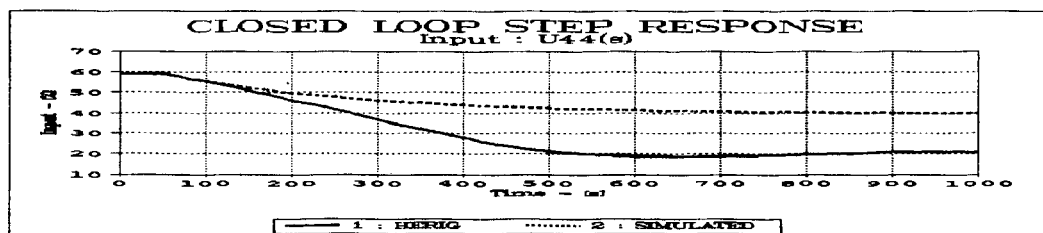


Figure 7.2.16 : OBSERVED AND SIMULATED U44(s)

By checking the inputs it can be said that all of them seem to settle to a final value, and thus stability is ensured. None of the inputs hit a limit during the step tests (care was taken with the step sizes to ensure this).

All the input functions from the HERIG had the same polarity as their corresponding simulated functions. The final values that they settled to were in a few cases very similar, but in general they were different. This is due to variations in the gains, from the modelled values, of the actual plant, which then requires more (or less) input action to achieve the desired setpoints.

The shapes of the input responses are the same for many of the plots. This indicates that not only does the H_{∞} synthesis and design predict the correct responses, but that the controller has been implemented on the HERIG successfully.

Thus in conclusion it can be said that the H_{∞} designed controller performed as expected on the actual HERIG, despite drifts, sensor noise, input disturbances and variations in the actual process parameters.

CHAPTER 8 : CONCLUSIONS

This dissertation has presented a discussion and modelling of the Carbon-in-pulp gold extraction process as well as a Heat Exchanger Rig that simulates the CIP process. A new method of controller design, H_{∞} optimal Control, has been formally presented and used to design a multivariable feedback controller for the HERIG. Based on the contents of this dissertation, the following conclusions can be drawn.

8.1 : MODELLING OF CIP

The CIP process is one which is being studied extensively in the literature. Much attention is being paid to the accurate modelling of this process, as it is chemically a very involved process. A simple model, based on fundamental rate exchange expressions, was found to be easy to derive.

This simple model for the CIP process was found to simulate the adsorption of gold onto activated carbon accurately enough. The simulation provided a basic understanding of the process involved and also gave a indication as to the gains and time constants that can be expected on an operational plant that transfers the carbon in a continuous manner.

The simulations showed that there is a noticeable amount of interaction between the gold concentrations in solution in the different tanks due to changes in the amount of carbon in the tanks. This has led to the conclusion that to operate the process in closed loop a multivariable controller should be considered because of this interaction.

8.2 : MODELLING OF THE HERIG

The Heat Exchanger Rig was found to be a highly involved real process. The simulation of such a process involved many thermal calculations of rates of heat transfer. It was found that the approximation of a lumped parameter model was acceptable as the system was broken up into a large number of small sections that were lumped together to form the model.

The results of the temperature responses of the HERIG to a change in water levels were found to be good. All steady state values agreed with the data obtained from steady state calculations. The transient behaviour was explained in detail and the responses observed correlated well with those observed on the HERIG itself.

Thus it was concluded that the modelling of the HERIG provided a good understanding of the process involved and that the predicted results of the simulator were very similar to those noted on the rig itself.

8.3 : SIMILARITIES BETWEEN CIP AND THE HERIG

When comparing the results of the simulators it was found that the gains of the two systems were very different, but this was expected as the means of heat and carbon transfer are different. By the same reasoning the time constants are also different but expected.

The two processes show the same types of response to changes in the respective inputs. The fact that the gains are different does not pose any problem as scaling of the inputs and outputs can eliminate this difference. The assumptions made about which variables are to be related on the two processes turned out to be correct.

The conclusions drawn about the similarities in the two processes are that there are enough variables that behave similarly to say that the two systems can be compared. The similarities in the dynamic response curves allows the conclusion to be drawn that a study of the closed loop control of the HERIG will allow relevant comments to be made about the closed loop control of the CIP process.

8.4 : H_{∞} DESIGN METHODOLOGY

The mathematical complexities of the solution to the synthesis problem posed have been solved. This allows the H_{∞} method of controller design to be considered much more readily as a design tool. The method involves frequency domain specification to be placed on a plant model, after which a controller may be able to be found that satisfies these frequency criteria.

The method of obtaining a controller can be seen to involve a synthesis as well as a design phase. The synthesis is an easily solved problem, provided certain matrix conditions are obeyed. The design section is one that was studied in depth in this dissertation and the following conclusions were drawn as to the objectives, time and frequency domain, that could be achieved with the weighting functions.

The frequency domain aims of robustness, disturbance rejection and measurement noise attenuation are easily achieved as the weights are specified in this domain. Setpoint tracking is also easily achievable since this merely requires that the closed loop transfer function matrix, $T(s)$, be equal to the identity matrix I at low frequencies.

It was found that interaction between the loops could be eliminated easily. In the frequency domain interaction can be seen when the principle gains of the closed loop transfer function $T(s)$ are not close together. If they are found to be far apart, an adjustment of the weight w_2^{-1} on the input action was found to cure this problem.

Finally the problem of excessive input action was addressed. Violent input action results when an optimal controller is found using the synthesis technique of Glover and Doyle. This will show up on the principle gain plot of the input action function $R(s)$. Obtaining a smooth input action function was achievable by choosing the weight w_2^{-1} such that $R(s)$ is attenuated severely at high frequencies.

Finally the speed of response of a system has a direct bearing on the magnitude of the input action. If the magnitudes are too big then the closed loop system can easily be slowed down and the magnitude of the input action will decrease. The speed of response is determined by the crossover frequency of the sensitivity function $S(s)$ and the robustness function $T(s)$.

Thus it can be concluded that the design objectives in both the time and frequency domain can be achieved using frequency domain weighting functions. It has been shown that using the principle gain plot, much insight can be gained as to the design of weights that will accomplish the design objectives. The time domain objectives can be seen in the principle gain plots, but need to be verified using time domain simulations.

8.5 : DESIGN OF A CONTROLLER FOR HERIG TEMPERATURE CONTROL

An algorithm was developed to implement the design and the synthesis phases in obtaining a controller for the HERIG. This can be achieved using a suitable computer package.

It was found that a reduction of an original model proved useful for two different reasons. Firstly it reduces the order of a controller that achieves the desired performance objectives and secondly it saves greatly on computational time. This is useful as a full multivariable design is a lengthy process.

The design of a controller that worked for simulations proved easy, but when it came to implementation of the controllers problems were encountered. These were solved by ensuring that the input actions generated were smooth functions. Thus a design that met all of the time and frequency domain specifications was achieved on the HERIG.

8.6 : CONTROL OF THE HEAT EXCHANGER RIG

The controller implemented on the HERIG tracked all the required setpoints, even though the interaction between some of the loops was greater than expected. The response times of the loops compared very well with those expected from the simulations.

The input functions never hit limits and all settled down to some or other final value. Many of the inputs showed exactly the same responses as the simulator predicted, which validated the controller implementation.

Thus in conclusion the designed controller was closed loop stable and most functions performed as expected. The implementation of the H_{∞} designed controller was thus considered totally successful.

8.7 : CONTROL OF THE CIP PROCESS

Since the models of the CIP and HERIG processes were shown to be similar, it is concluded that a multivariable H_{∞} based controller can be designed for an actual CIP process. It would require that the carbon on a CIP process be transferred continuously. The concentrations of gold in solution in each tank can then be controlled to specific setpoints without affecting the concentrations in other tanks.

The amount of carbon used is critical, as the activated carbon is expensive. It has been shown that using the H_{∞} method that the amount of input action used to achieve the required dynamic responses, carbon used in this case, can be controlled in the design of a controller. This is very important economical conclusion when considering the implementation on an operational plant.

Thus finally this dissertation concludes that the H_{∞} Optimal method of controller design can successfully be used on counter current processes, in particular it has application to the industrial process of carbon-in-pulp gold extraction.

BIBLIOGRAPHY

- ADAMS M.D. and NICOL M.J., The Kinetics of the Elution of Gold from Activated Carbon, *GOLD 100 : Proceedings of the International Conference on Gold. Vol 2 : Extractive Metallurgy of Gold*, Johannesburg, SAIMM, pp. 111-121, 1986.
- ANONYMOUS, The Development of an Integrated Carbon-in-Pulp process for South African Conditions, Application Report No. 3, MINTEK, pp.1-12, 1987.
- BRAAE M., Lecture notes : Advanced Control Systems, 1990.
- BRAAE M., Lecture notes : Control Engineering I, 1988.
- CARRIER C., HODOUIN D. and COURCHESNE, Dynamic Simulation of the CIP Gold Recovery Process, *Proceedings of the International Symposium on Gold Metallurgy*, Winipeg, pp.309-325, 1987.
- CHIANG R.Y. and SAFONOV M.G., Robust Control Toolbox : for use with MATLAB, The MathWorks, Jun 1988.
- CRAIG I.K., Sensitivity of H_{∞} Controller Designs to Structured Uncertainty, M.Sc. Dissertation, Department of Aeronautics and Astronautics, Massachusetts Institute of Technology, May 1989.
- DARYANANI G., Principles of Active Network Synthesis and Design, Singapore : Wiley, pp.126-129, 1976.
- DE WAAL A., Performance Index Analysis of Control Systems Synthesised for a Counter-Current Process, M.Sc. Dissertation, Department of Electrical and Electronic Engineering, University of Cape Town, April 1990.
- DIXON S., CHO E.H. and PITT C.H., The Interaction between Gold cyanide, Silver cyanide and High Surface Area Charcoal, *AIChE Symposium Series 74*, Vol. 74, no. 173, pp.75-83, 1978.

- DOYLE J.C., GLOVER K., KHARGONEKAR P.P and FRANCIS B.A., State-Space Solutions to Standard H_2 and H_∞ Control Problems, *IEEE Trans. on Auto. Control*, Vol. 34, No.8, pp.831-847, Aug 1989.
- FAST J.L., Carbon-in-Pulp Pioneering at the Carlton Mill, *E&MJ*, pp.56-57, June 1990.
- FLEMING C.A. and NICOL M.J., The Absorption of Gold Cyanide onto Activated Carbon. III. Factors Influencing the Rate of Loading and the equilibrium Capacity, *J. S. Afr. Inst. Min. Metall.*, Vol. 84, no. 4, pp.85-93, Apr 1984.
- FLEMING C.E., NICOL M.J. and NICOL D.I., The Optimisation of a Carbon-in-Pulp Adsorption Circuit based on the Kinetics of Extraction of Aurocyanide by Activated Carbon, Presented at MINTEK Meeting "Ion Exchange and Solvent Extraction in Mineral Processing", MINTEK, Feb. 1980.
- GEAR A.B.J., The Design of Decentralized Controllers for Large Scale Systems, M.Sc. Dissertation, Department of Electrical and Electronic Engineering, University of Cape Town, Feb 1988.
- HALL K.B., Homestake uses carbon-in-pulp process to recover gold from slimes, *World Mining*, vol.27 no. 12, pp.44-49, 1974.
- HIMMELBLAU D.M. and BISCHOFF K.B., Process Analysis and Simulation, Wiley, pp.222-223, 1968.
- HOLMAN J.P., Heat Transfer, Tokyo : McGraw-Hill, 1976.
- JAMES H.E., DUNNE R.C., DE WAAL S.A. and LAXEN P.A., Established Procedures and New Advances in the Extraction of Gold, Uranium and Pyrite from Witwatersrand Ores, *Proceedings 12TH CMMI Congress*, Johannesburg, S. Afr. Inst. Min. Metall., pp.515-525, 1982.
- KERN D.Q., Process Heat Transfer, Tokyo : McGraw-Hill, 1950.

- KREYSZIG E., Advanced Engineering Mathematics, New York : Wiley, pp.661-672, 1972.
- LAXEN P.A., FLEMING C.A., HOLTRUM D.A. and RUBIN R. A Review of Pilot-Plant Testwork Conducted on the Carbon-in-Pulp Process for the Recovery of Gold, *Proceedings 12TH CMMI Congress*, Johannesburg, S. Afr. Inst. Min. Metall., pp.551-561, 1982.
- MACIEJOWSKI J.M., Multivariable Feedback Design, Reading MA : Addison-Wesley, 1989.
- MENNE D., Predicting and Assessing Carbon-in-Pulp Performance, *CIM XIV, 1ST Mineral Processing Conference*, Toronto, 1982.
- NICOL M.J., FLEMING C.A. and CROMBERGE G., The Absorption of Gold Cyanide onto Activated Carbon. I. The Kinetics of Absorption from pulps, *J. S. Afr. Inst. Min. Metall.*, Vol. 84, no. 2, pp.50-54, Feb 1984.
- NICOL M.J., FLEMING C.A. and CROMBERGE G., The Absorption of Gold Cyanide onto Activated Carbon. II. Application of the Kinetic Model to Multistage Absorption Circuits, *J. S. Afr. Inst. Min. Metall.*, Vol. 84, no. 3, pp.70-78, Mar 1984.
- POHLANDT-WATSON C., Historical, Analytical and Environmental Aspects of the Cyanide Process, *Cyanide in Gold Processing : The Next 100 Years*, Johannesburg, The South African Chemical Institute, 1990.
- ROSENBROCK H.H., State-Space and Multivariable Theory, London : Nelson, 1970.
- SAFONOV M.G., LAUB A.J. and HARTMAN G.L., Feedback Properties of Multivariable Systems : The Role and Use of the Return Difference Matrix, *IEEE Trans. on Auto. Control*, Vol 26, No 1, pp.47-65, Feb 1981.
- SAFONOV M.G., Simplifying the H_{∞} Theory via Loop Shifting, *Proc. IEEE Conf. on Decision and Control*, Austin TX, pp.1399-1404, 1988.

- SEINFELD J.H. and LAPIDUS L., Mathematical Modeling in Chemical Engineering Volume 3 Process Modeling, Estimation and Identification, New Jersey : Prentice-Hall, pp.134-138, 1974.
- STANGE W., The Simulation and Optimisation of the CIP Process, *Cyanide in Gold Processing : The Next 100 Years*, Johannesburg, The South African Chemical Institute, 1990.
- STANGE W., KING R.P. and WOOLLACOTT L.C., Towards more Effective Simulation of CIP-CIL Processes. Part II : A Population Balance Based Simulation Approach, To be published in *J. S. Afr. Inst. Min. Metall.*, 1990.
- STANGE W., WOOLLACOTT L.C. and KING R.P., Towards more Effective Simulation of CIP-CIL Processes. Part III : Validation and Use of a New Simulator, To be published in *J. S. Afr. Inst. Min. Metall.*, 1990.
- STANLEY G.G., The Extractive Metallurgy of Gold in South Africa : Volume 1, Johannesburg : Kelvin House, 1987.
- TODTENHAUPT P., FORSCHNER P., GRIMSLEY D. and LASES A.B.M., Design of Agitators for the ERGO Carbon-in-Leach Plant, *GOLD 100 : Proceedings of the International Conference on Gold. Vol 2 : Extractive Metallurgy of Gold*, Johannesburg, SAIMM, pp. 195-207, 1986.
- VAN DER WALT T.J., Parametric Sensitivity Analysis of Abnormal Behaviour in the Dynamic Simulation of Carbon-in-Pulp Plants, B.sc. Dissertation, Department of Metallurgical Engineering, University of Stellenbosch, Nov 1989.
- VAN DEVENTER J.S.J., Factors Influencing the Efficiency of a Carbon-in-Pulp Adsorption Circuit for the Recovery of Gold from Cyanided Pulps, *GOLD 100 : Proceedings of the International Conference on Gold. Vol 2 : Extractive Metallurgy of Gold*, Johannesburg, SAIMM, pp. 85-95, 1986.

- VIDYASAGAR M., Control Systems Synthesis : A Factorisation Approach, Boston MA : MIT Press, 1985.
- WELTY J.R., WICKS C.E. and WILSON R.E., Fundamentals of Momentum, Heat and Mass Transfer, New York : Wiley, pp. 169-170, 1976.
- WILLIAMS D.F. and GLASSER D., The Modelling and Simulation of Processes for the absorption of Gold by Activated Charcoal, *J. S. Afr. Inst. Min. Metall.*, Vol. 85, no. 8, pp.237-243, Aug 1985.
- WOOLLACOTT L.C., STANGE W. and KING R.P., Towards more Effective Simulation of CIP-CIL Processes. Part I : The Modelling of Adsorption and Leaching, To be published in *J. S. Afr. Inst. Min. Metall.*, 1990.
- YOUNG G.J.C., DOUGLAS W.D. and HAMPSHIRE M.J., Carbon in Pulp Process for Recovering Gold from Acid Plant Calcines at President Brand, *Mining Engineering*, pp.257-264, March 1984.
- ZAMES G., Feedback and Optimal Sensitivity : Model reference transformations, Multiplicative seminorms and Approximate inverses, *IEEE Trans. on Auto. Control*, Vol 26, No 1, pp.301-320, Feb 1981.

APPENDIX A : CARBON-IN-PULP SIMULATION SOFTWARE

A.1 : MAIN PROGRAMME

```
PROGRAM carbon_in_pulp_simulator;
```

```
{ $M 65520,0,655360 }
```

```
USES
```

```
    plant,matrix,data,  
    crt;
```

```
VAR
```

```
    inpt : vect_ptr;  
    oupt : vect_ptr;
```

```
    stat : stat_ptr;
```

```
    logs : data_ptr;
```

```
    pts : integer;
```

```
    time : real;
```

```
    ch : char;
```

```
    row : integer;
```

```
PROCEDURE setup_vect;
```

```
BEGIN
```

```
    new (inpt);vect_initl (inpt);
```

```
    new (oupt);vect_initl (oupt);
```

```
END;
```

```
PROCEDURE setup_stat;
```

```
BEGIN
```

```
    new (stat);stat_initl (stat);
```

```
END;
```

```
PROCEDURE setup_data;
```

```
BEGIN
```

```
    new (logs);data_init (logs);
```

```
END;
```

```
PROCEDURE setup_stst;
```

```
VAR
```

```
    i,j : integer;
```

```
    z : double;
```

```
BEGIN
```

```
    clrscr;
```

```
    gotoxy (1, 1);write ('_____');
```

```

gotoxy (1, 2);write ('
gotoxy (1, 3);write ('
gotoxy (1, 4);write ('
gotoxy (1, 5);write ('
gotoxy (1, 6);write ('
gotoxy (1, 7);write ('
gotoxy (1, 8);write ('
gotoxy (1, 9);write ('
gotoxy (1,10);write ('
gotoxy (1,11);write ('
gotoxy (1,12);write ('
gotoxy (1,13);write ('
gotoxy (1,14);write ('
gotoxy (1,15);write ('
gotoxy (1,16);write ('
gotoxy (1,17);write ('
gotoxy (1,18);write ('
gotoxy (1,19);write ('
gotoxy (1,20);write ('
gotoxy (1,21);write ('

```

CALCULATION OF THE STEADY STATE VALUES			
SPECIFIED :			
Fs =		Cs0 =	
Fc =		Cc5 =	
inpt[T]	Cs [g/T]	Cc [g/T]	oupt[g/T]
[]	[]	[]	[]
[]	[]	[]	[]
[]	[]	[]	[]
[]	[]	[]	[]
[]	[]	[]	[]
CALCULATED :			
Cc5 =			
Iter =			

```
FOR i := 1 to max_cntrl do
```

```
BEGIN
```

```
inpt^.o^[i] := Mass[i];
```

```
inpt^.n^[i] := Mass[i];
```

```
gotoxy (32-24,10+i);write (inpt^.n^[i]:7:3);
```

```
END;
```

```
gotoxy (56-24,6);write (Fs:6:2,' [T/hr]');
```

```
gotoxy (56-24,7);write (Fc:6:2,' [T/hr]');
```

```
gotoxy (79-24,6);write (Cs0:5:2,' [g/T]');
```

```
gotoxy (79-24,7);write (Cc5:5:2,' [g/T]');
```

```
stat^.o^[5] := 10*(Cc5);
```

```
j := 0;
```

```
REPEAT
```

```
inc (j);
```

```
stat^.o^[1] := (Fs*Cs0 + inpt^.o^[1]*k2*stat^.o^[5])/(Fs + inpt^.n^[1]*k1*(ye-stat^.o^[5]));
```

```
stat^.o^[6] := stat^.o^[5] - inpt^.n^[1]/Fc * (k1*stat^.o^[1]*(ye-stat^.o^[5]) - k2*stat^.o^[5]);
```

```
stat^.o^[2] := (Fs*stat^.o^[1] + inpt^.o^[2]*k2*stat^.o^[6])/(Fs + inpt^.n^[2]*k1*(ye-stat^.o^[6]));
```

```
stat^.o^[7] := stat^.o^[6] - inpt^.n^[2]/Fc * (k1*stat^.o^[2]*(ye-stat^.o^[6]) - k2*stat^.o^[6]);
```

```
stat^.o^[3] := (Fs*stat^.o^[2] + inpt^.o^[3]*k2*stat^.o^[7])/(Fs + inpt^.n^[3]*k1*(ye-stat^.o^[7]));
```

```
stat^.o^[8] := stat^.o^[7] - inpt^.n^[3]/Fc * (k1*stat^.o^[3]*(ye-stat^.o^[7]) - k2*stat^.o^[7]);
```

```
stat^.o^[4] := (Fs*stat^.o^[3] + inpt^.o^[4]*k2*stat^.o^[8])/(Fs + inpt^.n^[4]*k1*(ye-stat^.o^[8]));
```

```
z := stat^.o^[8] - inpt^.n^[4]/Fc * (k1*stat^.o^[4]*(ye-stat^.o^[8]) - k2*stat^.o^[8]);
```

```
FOR i := 1 to max_cntrl do
```

```
BEGIN
```

```
gotoxy (47-24,10+i);write (stat^.o^[i+0]:10:6);
```

```
gotoxy (70-24,10+i);write (stat^.o^[i+4]:10:3);
```

```
oupt^.o^[i] := dydt (i,stat^.o^,inpt^.n^);
```

```
oupt^.n^[i] := dydt (i,stat^.o^,inpt^.n^);
```

```
gotoxy (91-24,10+i);write (oupt^.n^[i]:7:3);
```

```

END;
gotoxy (64-24,18);write (z:12:6);
gotoxy (64-24,19);write (j:3);

stat^.o^[5] := stat^.o^[5] + (Cc5 - z)/2;

UNTIL (abs(Cc5 - z)/Cc5 < 1E-14);

gotoxy (1,22);write ('
gotoxy (1,23);write ('
gotoxy (1,24);write ('
REPEAT
UNTIL keypressed;
ch := readkey;

clrscr;
gotoxy (1, 1);write ('
gotoxy (1, 2);write ('
gotoxy (1, 3);write ('
gotoxy (1, 4);write ('
gotoxy (1, 5);write ('
gotoxy (1, 6);write ('
gotoxy (1, 7);write ('
gotoxy (1, 8);write ('
gotoxy (1, 9);write ('
gotoxy (1,10);write ('
gotoxy (1,11);write ('
gotoxy (1,12);write ('
gotoxy (1,13);write ('
gotoxy (1,14);write ('
gotoxy (1,15);write ('
gotoxy (1,16);write ('
gotoxy (1,17);write ('
gotoxy (1,18);write ('
gotoxy (1,19);write ('
gotoxy (1,20);write ('
gotoxy (1,21);write ('
gotoxy (1,22);write ('
gotoxy (1,23);write ('
gotoxy (1,24);write ('

<<< PRESS ANY KEY TO CONTINUE >>>

FOR i := 1 to max_state do
  stat^.n^[i] := dXdt (i,stat^.o^,inpt^.n^,stat^.o^,0);
FOR i := 1 to 4 do
  BEGIN
    gotoxy (25,05+i);write ('dCs',i:1,'dt = ',stat^.n^[i+0]);
    gotoxy (25,15+i);write ('dCc',i:1,'dt = ',stat^.n^[i+4]);
  END;

REPEAT
UNTIL keypressed;
ch := readkey;
END;

```

PROCEDURE open_loop_simulation;

PROCEDURE setup_scrn;

BEGIN

clrscr;

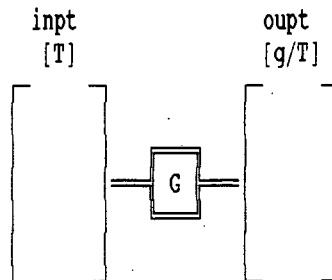
```

gotoxy (1, 1);write ('');
gotoxy (1, 2);write ('');
gotoxy (1, 3);write ('');
gotoxy (1, 4);write ('');
gotoxy (1, 5);write ('');
gotoxy (1, 6);write ('');
gotoxy (1, 7);write ('');
gotoxy (1, 8);write ('');
gotoxy (1, 9);write ('');
gotoxy (1,10);write ('');
gotoxy (1,11);write ('');
gotoxy (1,12);write ('');
gotoxy (1,13);write ('');
gotoxy (1,14);write ('');
gotoxy (1,15);write ('');
gotoxy (1,16);write ('');
gotoxy (1,17);write ('');
gotoxy (1,18);write ('');
gotoxy (1,19);write ('');
gotoxy (1,20);write ('');
gotoxy (1,21);write ('');
gotoxy (1,22);write ('');
gotoxy (1,23);write ('');
gotoxy (1,24);write ('');

```

CARBON-IN-PULP SIMULATION
: OPEN LOOP SYSTEM

hr



END;

PROCEDURE updat_scrn;

BEGIN

```

gotoxy (90-27,4);write (time:6:3);

gotoxy (57-27,15);write (inpt^.n^[1]:7:3);
gotoxy (57-27,16);write (inpt^.n^[2]:7:3);
gotoxy (57-27,18);write (inpt^.n^[3]:7:3);
gotoxy (57-27,19);write (inpt^.n^[4]:7:3);

```

```

gotoxy (77-27,15);write (oupt^.n^[1]:7:3);
gotoxy (77-27,16);write (oupt^.n^[2]:7:3);
gotoxy (77-27,18);write (oupt^.n^[3]:7:3);
gotoxy (77-27,19);write (oupt^.n^[4]:7:3);

```

END;

PROCEDURE loggs_data;

BEGIN

```

IF (pts <= max_point) and (abs(time - pts*dx) < dt/2) then
    BEGIN
        data_logg (logs,oupt^.n,inpt^.n,time,pts);
        inc (pts);
    END;

```

END;

```
PROCEDURE simulate;
BEGIN
  time := time + dt;

  IF abs(time - 0.05*dx*max_point) < dt/2 THEN
    inpt^.n^[row] := inpt^.n^[row] * Step[row];

    stat_ip2op (inpt,stat,oupt);
  END;

BEGIN
  REPEAT
    clrscr;
    write ('WHICH INPUT IS TO BE STEPPED (1/2/3/4) ? ');
    row := ord(upcase(readkey)) - ord('0');
  UNTIL (row > 0) AND (row <= max_cntrl);

  setup_scrn;
  loggs_data;
  REPEAT
    simulate;
    loggs_data;
    updat_scrn;
  UNTIL (time > dx*max_point);

  data_save (logs,'cip_g'+chr(ord('0')+row)+'.dat');
END;

{*****}
{      MAIN      PROGRAMME      }
{*****}
BEGIN
  setup_vect;
  setup_stat;
  setup_data;
  setup_stst;

  pts := 0;
  time := 0;

  open_loop_simulation;
END.
```

A.2 : PLANT DESCRIPTION UNIT

UNIT plant;

{*****}

INTERFACE

{*****}

CONST

{***** GENERAL CONSTANTS *****}

max_state = 8; { number of states in system }
max_cntrl = 4; { number of inputs/outputs in system }

max_point = 100; { number of points to be logged }
dt = 0.5; { time interval between increments [hr] }
dx = 0.5; { time interval between logging [hr] }

{***** SPECIFIC CONSTANTS *****}

Cs0 = 1.0; { concentration of Au in slurry entering Tank 1 [g Au/Ton slurry] }
Cc5 = 50.0; { concentration of Au on carbon entering Tank 5 [g Au/Ton carbon] }
Ps = 700.0; { flowrate of slurry [Ton slurry/hr] }
Fc = 0.6; { flowrate of carbon [Ton carbon/hr] }
Ms = 2592.0; { mass of slurry in each tanks [Ton slurry] }

k1 = 0.017; { adsorption rate constant [(g Au/Ton slurry)⁻¹ * (hr)⁻¹] }
k2 = 0.005; { adsorption rate constant [hr⁻¹] }
ye = 11650; { adsorption parameter [g Au/Ton carbon] }

TYPE

v_var = array [1..max_cntrl] of double;
v_ptr = ^v_var;
vect_var = record
o : v_ptr;
n : v_ptr;
END;
vect_ptr = ^vect_var;

x_var = array [1..max_state] of double;
x_ptr = ^x_var;
stat_var = record
o : x_ptr;
n : x_ptr;
END;
stat_ptr = ^stat_var;

VAR

Step : array [1..max_cntrl] of double; { factor to step input by }
Mass : array [1..max_cntrl] of double; { initial masses in tanks [Tons carbon] }

FUNCTION dxdt (i:integer;x:x_var;u:v_var;z:x_var;f:double) : double;
FUNCTION dydt (i:integer;x:x_var;u:v_var) : double;

{*****}

IMPLEMENTATION

```
{*****}
FUNCTION dXdt (i:integer;x:x_var;u:v_var;z:x_var;f:double) : double;
BEGIN
  CASE i of
    1 : dXdt := Fs/Ms * (Cs0 - x[1]) - u[1]/Ms * (k1*x[1]*(ye - x[5]) - k2*x[5]);
    2 : dXdt := Fs/Ms * (x[1] - x[2]) - u[2]/Ms * (k1*x[2]*(ye - x[6]) - k2*x[6]);
    3 : dXdt := Fs/Ms * (x[2] - x[3]) - u[3]/Ms * (k1*x[3]*(ye - x[7]) - k2*x[7]);
    4 : dXdt := Fs/Ms * (x[3] - x[4]) - u[4]/Ms * (k1*x[4]*(ye - x[8]) - k2*x[8]);
    5 : dXdt := Fc/u[1] * (x[6] - x[5]) + (k1*x[1]*(ye - x[5]) - k2*x[5]);
    6 : dXdt := Fc/u[2] * (x[7] - x[6]) + (k1*x[2]*(ye - x[6]) - k2*x[6]);
    7 : dXdt := Fc/u[3] * (x[8] - x[7]) + (k1*x[3]*(ye - x[7]) - k2*x[7]);
    8 : dXdt := Fc/u[4] * (Cc5 - x[8]) + (k1*x[4]*(ye - x[8]) - k2*x[8]);
  END;
END;

FUNCTION dYdt (i:integer;x:x_var;u:v_var) : double;
BEGIN
  CASE i of
    1 : dydt := x[1];
    2 : dydt := x[2];
    3 : dydt := x[3];
    4 : dydt := x[4];
  END;
END;

{*****}
{***  INITIALIZATION SECTION  ***}
{*****}
BEGIN
  Step[1] := 1.5;
  Step[2] := 1.5;
  Step[3] := 1.5;
  Step[4] := 1.5;

  Mass[1] := 14.4;
  Mass[2] := 14.4;
  Mass[3] := 14.4;
  Mass[4] := 14.4;
END.
```

A.3 : MATRIX HANDLING UNIT

```
UNIT matrix;

{*****}
      INTERFACE
{*****}
USES
  plant;

  PROCEDURE vect_initl (v:vect_ptr);
  PROCEDURE stat_initl (x:stat_ptr);
  PROCEDURE stat_ip2op (u:vect_ptr;x:stat_ptr;y:vect_ptr);

{*****}
      IMPLEMENTATION
{*****}

PROCEDURE vect_initl (v:vect_ptr);
VAR
  r : integer;
BEGIN
  new (v^.o);
  new (v^.n);
  FOR r := 1 to max_cntrl do
    BEGIN
      v^.o^[r] := 0;
      v^.n^[r] := 0;
    END;
  END;

PROCEDURE stat_initl (x:stat_ptr);
VAR
  j : integer;
BEGIN
  new (x^.o);
  new (x^.n);
  FOR j := 1 to max_state DO
    BEGIN
      x^.o^[j] := 0;
      x^.n^[j] := 0;
    END;
  END;

PROCEDURE stat_ip2op (u:vect_ptr;x:stat_ptr;y:vect_ptr);
VAR
  j : integer;
  m : array [0..4] of x_var;
BEGIN
  FOR j := 1 to max_state DO m[0][j] := 0;
  FOR j := 1 to max_state DO m[1][j] := 0;
  FOR j := 1 to max_state DO m[2][j] := 0;
  FOR j := 1 to max_state DO m[3][j] := 0;
```

```
FOR j := 1 to max_state DO m[4][j] := 0;

FOR j := 1 to max_state DO
  m[1][j] := dt * dXdt (j,x^.o^,u^.n^,m[0],0.0);
FOR j := 1 to max_state DO
  m[2][j] := dt * dXdt (j,x^.o^,u^.n^,m[1],0.5);
FOR j := 1 to max_state DO
  m[3][j] := dt * dXdt (j,x^.o^,u^.n^,m[2],0.5);
FOR j := 1 to max_state DO
  m[4][j] := dt * dXdt (j,x^.o^,u^.n^,m[3],1.0);

FOR j := 1 to max_state DO
  x^.n^[j] := x^.o^[j] + (1/6)*(m[1][j] + 2*m[2][j] + 2*m[3][j] + m[4][j]);

FOR j := 1 to max_cntrl DO
  y^.n^[j] := dYdt (j,x^.n^,u^.n^);

FOR j := 1 to max_state DO
  x^.o^[j] := x^.n^[j];
END;

{*****}
{***   INITIALIZATION SECTION   ***}
{*****}
BEGIN
END.
```

A.4 : DATA HANDLING UNIT

```
UNIT data;

{*****}
      INTERFACE
{*****}
USES
  plant;

TYPE
  data_var = record
    y : array [0..max_point] of v_ptr;
    u : array [0..max_point] of v_ptr;
    x : array [0..max_point] of double;
  END;
  data_ptr = ^data_var;

  PROCEDURE data_init (d:data_ptr);
  PROCEDURE data_logg (d:data_ptr;y,u:v_ptr;x:double;z:integer);
  PROCEDURE data_save (d:data_ptr;f:string);

{*****}
      IMPLEMENTATION
{*****}
USES
  crt,graph;

  PROCEDURE data_init (d:data_ptr);
  VAR
    i : integer;
  BEGIN
    FOR i := 0 to max_point do
      BEGIN
        new (d^.y[i]);
        new (d^.u[i]);
      END;
    END;

  PROCEDURE data_logg (d:data_ptr;y,u:v_ptr;x:double;z:integer);
  VAR
    r,c : integer;
  BEGIN
    FOR r := 1 to max_cntrl do
      BEGIN
        d^.y[z]^r := y^r;
        d^.u[z]^r := u^r;
      END;
      d^.x[z] := x;
    END;

  PROCEDURE data_save (d:data_ptr;f:string);
  VAR
```

```

    i,j    : integer;
    outfile : text;
BEGIN
  assign (outfile,f);
  rewrite (outfile);
  FOR i := 0 to max_point DO
    BEGIN
      FOR j := 1 to max_cntrl DO write (outfile,d^.y[i]^[j]);
      FOR j := 1 to max_cntrl DO write (outfile,d^.u[i]^[j]);
      write  (outfile,d^.x[i]);
      writeln (outfile);
    END;
  close  (outfile);
END;

{*****}
{***  INITIALIZATION SECTION  ***}
{*****}
BEGIN
END.
```

APPENDIX B : CALCULATION OF CONSTANTS FOR THE HERIG

The calculation of heat transfer coefficients for the HERIG require properties of water and air to be used. These properties, together with data acquired from De Waal¹ concerning the physical layout of the rig will be presented before the coefficients are calculated.

B.1 : PROPERTIES OF AIR AND WATER

The properties of water are all obtained for an operating temperature of 40°C, which was chosen as the mean of the maximum and minimum temperatures on the HERIG. They are as follows² :

Mean wall temperature	$T_{WL} = 313$	[K]
Density	$\rho_W = 992.2$	[kg/m ³]
Viscosity	$\mu_W = 6.532E-4$	[kg/ms]
Thermal conductivity	$k_W = 0.6305$	[J/sm ² K]
Heat capacity	$C_W = 4178.5$	[J/kgK]

The pipe sections not submerged lose heat to air at room temperature, which is assumed to be at 23°C. The properties used are³ :

Mean air temperature	$T_{RM} = 296$	[K]
Density	$\rho_A = 1.1774$	[kg/m ³]
Viscosity	$\mu_A = 1.983E-5$	[kg/ms]
Thermal conductivity	$k_A = 0.02624$	[J/sm ² K]
Heat capacity	$C_A = 1005.7$	[J/kgK]

B.2 : PHYSICAL LAYOUT OF THE HERIG

The lengths of pipe in each tank are of different lengths, but same inner and outer diameters. Dividing the pipes in each into 20 sections gives :

Inner diameter of pipe	D_I	= 0.0120	[m]
Outer diameter of pipe	D_O	= 0.0127	[m]
Pipe section volumes in Tank1	V_{P1}	= 2.07E-5	[m ³]
Pipe section volumes in Tank2	V_{P2}	= 3.58E-5	[m ³]
Pipe section volumes in Tank3	V_{P3}	= 5.71E-5	[m ³]
Pipe section volumes in Tank4	V_{P4}	= 2.74E-5	[m ³]
Pipe area of contact in Tank1	A_1	= 7.38E-3	[m ²]
Pipe area of contact in Tank2	A_2	= 12.8E-3	[m ²]
Pipe area of contact in Tank3	A_3	= 20.3E-3	[m ²]
Pipe area of contact in Tank4	A_4	= 9.78E-3	[m ²]

The tank dimensions and stirrer specifications are as follows (the tank volumes refer to the maximum volume of water in a tank) :

Volume of Tank 1	V_{T1}	= 21.53E-3	[m ³]
Volume of Tank 2	V_{T2}	= 21.19E-3	[m ³]
Volume of Tank 3	V_{T3}	= 20.71E-3	[m ³]
Volume of Tank 4	V_{T4}	= 21.38E-3	[m ³]
Diameter of the tanks	D_T	= 0.3	[m]
Length of stirrers	L_S	= 0.04	[m]
Speed of stirrers	N_S	= 200	[rpm]

B.3 : OVERALL HEAT TRANSFER COEFFICIENT TO WATER

Using equations 3.6 to 3.8, the Reynolds, Prandtl and Nusseldt numbers are :

$$Re_W = 6750$$

$$Pr_W = 4.33$$

$$Nu_W = 47.83$$

By equations 3.5 and 3.9, the inner and outer transfer coefficients are :

$$H_{iW} = 2526 \quad [J/sm^2^{\circ}C]$$

$$H_{oW} = 1908 \quad [J/sm^2^{\circ}C]$$

Using equation 3.4, the overall heat transfer coefficient then becomes :

$$U_W = 1058 \quad [J/sm^2^{\circ}C]$$

B.4 : OVERALL HEAT TRANSFER COEFFICIENT TO AIR

Using equations 3.13 to 3.15, the Grashof, Prandtl and Nusseldt numbers are :

$$Gr_A = 3951$$

$$Pr_A = 0.76$$

$$Nu_A = 3.92$$

By equations 3.11 and 3.12, the inner and outer transfer coefficients are :

$$H_{iA} = 2526 \quad [J/sm^2^{\circ}C]$$

$$H_{oA} = 8.11 \quad [J/sm^2^{\circ}C]$$

Using equation 3.10, the overall heat transfer coefficient then becomes :

$$U_A = 8.08 \quad [J/sm^2^{\circ}C]$$

REFERENCES

1. DE WAAL A., Performance Index Analysis of Control Systems Synthesised for a Counter-Current Process, M.Sc. Dissertation, Department of Electrical and Electronic Engineering, University of Cape Town, April 1990.
2. LIDE D.R., Handbook of Chemistry and Physics, Boston : CRC Press, p. G-8, 1990.
3. HOLMAN J.P., Heat Transfer, Tokyo : McGraw-Hill, p. 503, 1976.

APPENDIX C : HEAT EXCHANGER RIG SIMULATION SOFTWARE

C.1 : MAIN PROGRAMME

```
PROGRAM heat_exchanger_simulator;
```

```
{ $M 65520,0,655360 }
```

```
USES
```

```
    plant,matrix,data,  
    crt;
```

```
VAR
```

```
    inpt : vect_ptr;  
    oupt : vect_ptr;
```

```
    stat : stat_ptr;
```

```
    logs : data_ptr;
```

```
    pts : integer;
```

```
    time : real;
```

```
    ch : char;
```

```
    row : integer;
```

```
PROCEDURE setup_vect;
```

```
BEGIN
```

```
    new (inpt);vect_initl (inpt);
```

```
    new (oupt);vect_initl (oupt);
```

```
END;
```

```
PROCEDURE setup_stat;
```

```
BEGIN
```

```
    new (stat);stat_initl (stat);
```

```
END;
```

```
PROCEDURE setup_data;
```

```
BEGIN
```

```
    new (logs);data_init (logs);
```

```
END;
```

```
PROCEDURE setup_stst;
```

```
VAR
```

```
    i,j : integer;
```

```
    z : double;
```

```
BEGIN
```

```
    clrscr;
```

```
    gotoxy (1, 1);write ('_____');
```

```

gotoxy (1, 2);write ('
gotoxy (1, 3);write ('
gotoxy (1, 4);write ('
gotoxy (1, 5);write ('
gotoxy (1, 6);write ('
gotoxy (1, 7);write ('
gotoxy (1, 8);write ('
gotoxy (1, 9);write ('
gotoxy (1,10);write ('
gotoxy (1,11);write ('
gotoxy (1,12);write ('
gotoxy (1,13);write ('
gotoxy (1,14);write ('
gotoxy (1,15);write ('
gotoxy (1,16);write ('
gotoxy (1,17);write ('
gotoxy (1,18);write ('
gotoxy (1,19);write ('
gotoxy (1,20);write ('
gotoxy (1,21);write ('

```

CALCULATION OF THE STEADY STATE VALUES			
SPECIFIED :			
Fp =		Tp0 =	
Ft =		Tt5 =	
inpt[%]	TP [°C]	Tt [°C]	oupt[°C]
[]	[]	[]	[]
[]	[]	[]	[]
[]	[]	[]	[]
[]	[]	[]	[]
[]	[]	[]	[]
CALCULATED :			
Tt5 =			
Iter =			

```
FOR i := 1 to max_cntrl do
```

```
  BEGIN
```

```
    inpt^.o^[i] := Hght[i];
```

```
    inpt^.n^[i] := Hght[i];
```

```
    gotoxy (32-24,10+i);write (inpt^.n^[i]:7:3);
```

```
  END;
```

```
gotoxy (55-24,6);write (Fp*60E3:5:2,' [l/min]');
```

```
gotoxy (55-24,7);write (Ft*60E3:5:2,' [l/min]');
```

```
gotoxy (79-24,6);write (Tp0:5:2,' [°C]');
```

```
gotoxy (79-24,7);write (Tt5:5:2,' [°C]');
```

```
stat^.o^[Lvl*4+1] := (Tp0 + Tt5)/2.0;
```

```
j := 0;
```

```
REPEAT
```

```
  inc (j);
```

```
{ Tank 1 : Next Pipe Temps }
```

```
  stat^.o^[1] := (U1*Ao[1]/(pw*Cw*Fp+U1*Ao[1])) * stat^.o^[Lvl*4+1] +  
                (pw*Cw*Fp/(pw*Cw*Fp+U1*Ao[1])) * Tp0;
```

```
FOR i := Lvl*0+2 to Lvl*0+round(Lvl*inpt^.n^[1]/100) do
```

```
  stat^.o^[i] := (U1*Ao[1]/(pw*Cw*Fp+U1*Ao[1])) * stat^.o^[Lvl*4+1] +  
                (pw*Cw*Fp/(pw*Cw*Fp+U1*Ao[1])) * stat^.o^[i-1];
```

```
FOR i := Lvl*0+round(Lvl*inpt^.n^[1]/100)+1 to Lvl*1 do
```

```
  stat^.o^[i] := (U2*Ao[1]/(pw*Cw*Fp+U2*Ao[1])) * Ta +  
                (pw*Cw*Fp/(pw*Cw*Fp+U2*Ao[1])) * stat^.o^[i-1];
```

```
{ Tank 1 : Next Tank Temp }
```

```
stat^.o^[Lvl*4+2] := stat^.o^[Lvl*4+1];
```

```
FOR i := Lvl*0+1 to Lvl*0+round(Lvl*inpt^.n^[1]/100) do
```

```
  stat^.o^[Lvl*4+2] := stat^.o^[Lvl*4+2] - U1*Ao[1]/(pw*Cw*Ft) * (stat^.o^[i] - stat^.o^[Lvl*4+1]);
```

```
{ Tank 2 : Next Pipe Temps }
```

```

FOR i := Lvl*1+1 to Lvl*1+round(Lvl*inpt^.n^[2]/100) do
  stat^.o^[i] := (U1*Ao[2]/(pw*Cw*Fp+U1*Ao[2])) * stat^.o^[Lvl*4+2] +
    (pw*Cw*Fp/(pw*Cw*Fp+U1*Ao[2])) * stat^.o^[i-1];
FOR i := Lvl*1+round(Lvl*inpt^.n^[2]/100)+1 to Lvl*2 do
  stat^.o^[i] := (U2*Ao[2]/(pw*Cw*Fp+U2*Ao[2])) * Ta
    +
    (pw*Cw*Fp/(pw*Cw*Fp+U2*Ao[2])) * stat^.o^[i-1];

{ Tank 2 : Next Tank Temp }
stat^.o^[Lvl*4+3] := stat^.o^[Lvl*4+2];
FOR i := Lvl*1+1 to Lvl*1+round(Lvl*inpt^.n^[2]/100) do
  stat^.o^[Lvl*4+3] := stat^.o^[Lvl*4+3] - U1*Ao[2]/(pw*Cw*Ft) * (stat^.o^[i] - stat^.o^[Lvl*4+2]);

{ Tank 3 : Next Pipe Temps }
FOR i := Lvl*2+1 to Lvl*2+round(Lvl*inpt^.n^[3]/100) do
  stat^.o^[i] := (U1*Ao[3]/(pw*Cw*Fp+U1*Ao[3])) * stat^.o^[Lvl*4+3] +
    (pw*Cw*Fp/(pw*Cw*Fp+U1*Ao[3])) * stat^.o^[i-1];
FOR i := Lvl*2+round(Lvl*inpt^.n^[3]/100)+1 to Lvl*3 do
  stat^.o^[i] := (U2*Ao[3]/(pw*Cw*Fp+U2*Ao[3])) * Ta
    +
    (pw*Cw*Fp/(pw*Cw*Fp+U2*Ao[3])) * stat^.o^[i-1];

{ Tank 3 : Next Tank Temp }
stat^.o^[Lvl*4+4] := stat^.o^[Lvl*4+3];
FOR i := Lvl*2+1 to Lvl*2+round(Lvl*inpt^.n^[3]/100) do
  stat^.o^[Lvl*4+4] := stat^.o^[Lvl*4+4] - U1*Ao[3]/(pw*Cw*Ft) * (stat^.o^[i] - stat^.o^[Lvl*4+3]);

{ Tank 4 : Next Pipe Temps }
FOR i := Lvl*3+1 to Lvl*3+round(Lvl*inpt^.n^[4]/100) do
  stat^.o^[i] := (U1*Ao[4]/(pw*Cw*Fp+U1*Ao[4])) * stat^.o^[Lvl*4+4] +
    (pw*Cw*Fp/(pw*Cw*Fp+U1*Ao[4])) * stat^.o^[i-1];
FOR i := Lvl*3+round(Lvl*inpt^.n^[4]/100)+1 to Lvl*4 do
  stat^.o^[i] := (U2*Ao[4]/(pw*Cw*Fp+U2*Ao[4])) * Ta
    +
    (pw*Cw*Fp/(pw*Cw*Fp+U2*Ao[4])) * stat^.o^[i-1];

{ Tank 4 : Next Tank Temp }
z := stat^.o^[Lvl*4+4];
FOR i := Lvl*3+1 to Lvl*3+round(Lvl*inpt^.n^[4]/100) do
  z := z - U1*Ao[4]/(pw*Cw*Ft) * (stat^.o^[i] - stat^.o^[Lvl*4+4]);

FOR i := 1 to max_cntrl do
  BEGIN
    gotoxy (47-24,10+i);write (stat^.o^[Lvl*i]:10:6);
    gotoxy (70-24,10+i);write (stat^.o^[Lvl*4+i]:10:6);
    oupt^.o^[i] := dydt (i,stat^.o^,inpt^.n^);
    oupt^.n^[i] := dydt (i,stat^.o^,inpt^.n^);
    gotoxy (91-24,10+i);write (oupt^.n^[i]:7:3);
  END;
  gotoxy (64-24,18);write (z:10:6);
  gotoxy (64-24,19);write (j:3);

  stat^.o^[Lvl*4+1] := stat^.o^[Lvl*4+1] + (Tt5 - z)/2;

UNTIL (abs(Tt5 - z)/Tt5 < 1E-14);

gotoxy (1,22);write ('

```

```

gotoxy (1,23);write ('
gotoxy (1,24);write ('
REPEAT
UNTIL keypressed;
ch := readkey;

```

<<< PRESS ANY KEY TO CONTINUE >>>

```

clrscr;
gotoxy (1, 1);write ('
gotoxy (1, 2);write ('
gotoxy (1, 3);write ('
gotoxy (1, 4);write ('
gotoxy (1, 5);write ('
gotoxy (1, 6);write ('
gotoxy (1, 7);write ('
gotoxy (1, 8);write ('
gotoxy (1, 9);write ('
gotoxy (1,10);write ('
gotoxy (1,11);write ('
gotoxy (1,12);write ('
gotoxy (1,13);write ('
gotoxy (1,14);write ('
gotoxy (1,15);write ('
gotoxy (1,16);write ('
gotoxy (1,17);write ('
gotoxy (1,18);write ('
gotoxy (1,19);write ('
gotoxy (1,20);write ('
gotoxy (1,21);write ('
gotoxy (1,22);write ('
gotoxy (1,23);write ('
gotoxy (1,24);write ('

```

```

FOR i := 1 to max_state do
  stat^.n[i] := dxdt (i,stat^.o,inpt^.n,stat^.o,0);
FOR i := 1 to 4 do
  BEGIN
    FOR j := 1 to Lvl do
      BEGIN
        gotoxy ((10+20*i)-28,1+j);write ('Tp',(Lvl*(i-1)+j):2,' = ',stat^.n[Lvl*(i-1)+j]:10:6);
      END;
        gotoxy ((10+20*i)-28,3+Lvl);write ('Tt',i:1,' = ',stat^.n[Lvl*4+i]:10:6);
      END;

```

```

REPEAT
UNTIL keypressed;
ch := readkey;
END;

```

PROCEDURE open_loop_simulation;

```

PROCEDURE setup_scrn;
BEGIN
  clrscr;
  gotoxy (1, 1);write ('

```

```

gotoxy (1,2);write (' ');
gotoxy (1,3);write (' ');
gotoxy (1,4);write (' ');
gotoxy (1,5);write (' ');
gotoxy (1,6);write (' ');
gotoxy (1,7);write (' ');
gotoxy (1,8);write (' ');
gotoxy (1,9);write (' ');
gotoxy (1,10);write (' ');
gotoxy (1,11);write (' ');
gotoxy (1,12);write (' ');
gotoxy (1,13);write (' ');
gotoxy (1,14);write (' ');
gotoxy (1,15);write (' ');
gotoxy (1,16);write (' ');
gotoxy (1,17);write (' ');
gotoxy (1,18);write (' ');
gotoxy (1,19);write (' ');
gotoxy (1,20);write (' ');
gotoxy (1,21);write (' ');
gotoxy (1,22);write (' ');
gotoxy (1,23);write (' ');
gotoxy (1,24);write (' ');
END;

```

```

PROCEDURE updat_scrn;
BEGIN
  gotoxy (90-27,4);write (time:6:3);

  gotoxy (57-27,15);write (inpt^.n^[1]:7:3);
  gotoxy (57-27,16);write (inpt^.n^[2]:7:3);
  gotoxy (57-27,18);write (inpt^.n^[3]:7:3);
  gotoxy (57-27,19);write (inpt^.n^[4]:7:3);

  gotoxy (77-27,15);write (oupt^.n^[1]:7:3);
  gotoxy (77-27,16);write (oupt^.n^[2]:7:3);
  gotoxy (77-27,18);write (oupt^.n^[3]:7:3);
  gotoxy (77-27,19);write (oupt^.n^[4]:7:3);
END;

PROCEDURE loggs_data;
BEGIN
  IF (pts <= max_point) and (abs(time - pts*dx) < dt/2) then
    BEGIN
      data_logg (logs,oupt^.n,inpt^.n,time,pts);
      inc (pts);
    END;
END;

PROCEDURE simulate;
BEGIN
  time := time + dt;

  IF abs(time - 0.05*dx*max_point) < dt/2 THEN

```

```
inpt^.n^[row] := inpt^.n^[row] * Step[row];

stat_ip2op (inpt,stat,oupt);
END;

BEGIN
REPEAT
  clrscr;
  write ('WHICH INPUT IS TO BE STEPPED (1/2/3/4) ? ');
  row := ord(uppercase(readkey)) - ord('0');
UNTIL (row > 0) AND (row <= max_cntrl);

setup_scrn;
loggs_data;
REPEAT
  simulate;
  loggs_data;
  updat_scrn;
UNTIL (time > dx*max_point);

data_save (logs,'hes_g'+chr(ord('0')+row)+'.dat');
END;

{*****}
{      M A I N      P R O G R A M M E      }
{*****}
BEGIN
  setup_vect;
  setup_stat;
  setup_data;
  setup_stst;

  pts := 0;
  time := 0;

  open_loop_simulation;
END.
```

C.2 : PLANT DESCRIPTION UNIT

UNIT plant;

{*****}

INTERFACE

{*****}

CONST

{***** GENERAL CONSTANTS *****}

max_state = 84; { number of states in system }
max_cntrl = 4; { number of inputs/outputs in system }

max_point = 100; { number of points to be logged }
dt = 0.5; { time interval between increments [s] }
dx = 10.0; { time interval between logging [s] }

{***** SPECIFIC CONSTANTS *****}

Lvl = 20; { number of pipe levels }

Tp0 = 58.0; { temperature entering pipes [C] }
Tt5 = 23.0; { temperature entering tanks [C] }
Fp = 2.5/60E3; { flowrate entering pipes [m^3/s] }
Ft = 5.0/60E3; { flowrate entering tanks [m^3/s] }

Tw = 40.0; { water : operating temperature [C] }
pw = 992.2; { water : density [kg/m^3] }
Cw = 4178.5; { water : heat capacity [J/(kg.C)] }
kw = 0.6305; { water : thermal conductivity [J/(m.s.C)] }
uw = 0.0006532; { water : viscosity [kg/(m.s)] }

Ta = 23.0; { air : operating temperature [C] }
pa = 1.1774; { air : density [kg/m^3] }
Ca = 1005.7; { air : heat capacity [J/(kg.C)] }
ka = 0.02624; { air : thermal conductivity [J/(m.s.C)] }
ua = 1.983E-5; { air : viscosity [kg/(m.s)] }

Dp = 0.50*0.0254; { outer diameter of copper pipe [m] }
Di = 0.47*0.0254; { inner diameter of copper pipe [m] }

Ls = 0.04; { length of stirrer in reactor [m] }
St = 400/60; { stirring rate in tanks [1/s] }
Dr = 0.3; { diameter of empty reactor tanks [m] }

Vr : array [1..4] of double { volume of empty reactor tanks [m^3] }
= (22.0E-3, 22.0E-3, 22.0E-3, 22.0E-3);

Lp : array [1..4] of double { length of copper pipes in tanks [m] }
= (3.7, 6.4, 10.2, 4.9);

TYPE

v_var = array [1..max_cntrl] of double;
v_ptr = ^v_var;
vect_var = record
o : v_ptr;

```

    n : v_ptr;
    END;
    vect_ptr = ^vect_var;

    x_var = array [1..max_state] of double;
    x_ptr = ^x_var;
    stat_var = record
        o : x_ptr;
        n : x_ptr;
    END;
    stat_ptr = ^stat_var;

VAR
    { Constants for : water }
    Rew : double;          { Reynolds number [] }
    Prw : double;          { Prandtl number [] }
    Nuw : double;          { Nusselt number [] }
    Hiw : double;          { outer heat transfer coefficient [J/(s.m^2.C)] }
    How : double;          { innner heat transfer coefficient [J/(s.m^2.C)] }
    U1 : double;           { overall heat transfer coeff [J/(s.m^2.C)] }

    { Constants for : air }
    Gra : double;          { Grashof number [] }
    Pra : double;          { Prandtl number [] }
    Nua : double;          { Nusselt number [] }
    Hia : double;          { outer heat transfer coefficient [J/(s.m^2.C)] }
    Hoa : double;          { innner heat transfer coefficient [J/(s.m^2.C)] }
    U2 : double;           { overall heat transfer coeff [J/(s.m^2.C)] }

    Vp : array [1..4] of double;    { volumes of pipe sections [m^3] }
    Vt : array [1..4] of double;    { volumes of tanks [m^3] }
    Ao : array [1..4] of double;    { surface area of pipe sections [m^2] }

    Step : array [1..max_cntrl] of double;    { factor to step inputs by }
    Hght : array [1..max_cntrl] of double;    { initial heights of tanks [%] }

    FUNCTION dxdt (i:integer;x:x_var;u:v_var;z:x_var;f:double) : double;
    FUNCTION dydt (i:integer;x:x_var;u:v_var) : double;

{*****}
    IMPLEMENTATION
{*****}
    FUNCTION dxdt (i:integer;x:x_var;u:v_var;z:x_var;f:double) : double;
VAR
    temp : double;
    j : integer;
BEGIN
    CASE i of
        1 : BEGIN
            dxdt := Fp/Vp[1] * ((Tp0 - (x[1]+z[1]*f))
            - U1*Ao[1]/(pw*Cw*Vp[1]) * ((x[1] +z[1] *f)-(x[Lvl*4+1]+z[Lvl*4+1]*f)));
            END;
        Lvl*0+2..Lvl*1 : IF (i <= Lvl*0+round(Lvl*u[1]/100)) THEN
            BEGIN

```

```

    dXdt := Fp/Vp[1] * ((x[i-1]+z[i-1]*f)-(x[i]+z[i]*f))
    - U1*Ao[1]/(pw*Cw*Vp[1]) * ((x[i] +z[i] *f)-(x[Lvl*4+1]+z[Lvl*4+1]*f));
END
ELSE
BEGIN
    dXdt := Fp/Vp[1] * ((x[i-1]+z[i-1]*f)-(x[i]+z[i]*f))
    - U2*Ao[1]/(pw*Cw*Vp[1]) * ((x[i] +z[i] *f)-(Ta));
END;
Lvl*1+1..Lvl*2 : IF (i <= Lvl*1+round(Lvl*u[2]/100)) THEN
BEGIN
    dXdt := Fp/Vp[2] * ((x[i-1]+z[i-1]*f)-(x[i]+z[i]*f))
    - U1*Ao[2]/(pw*Cw*Vp[2]) * ((x[i] +z[i] *f)-(x[Lvl*4+2]+z[Lvl*4+2]*f));
END
ELSE
BEGIN
    dXdt := Fp/Vp[2] * ((x[i-1]+z[i-1]*f)-(x[i]+z[i]*f))
    - U2*Ao[2]/(pw*Cw*Vp[2]) * ((x[i] +z[i] *f)-(Ta));
END;
Lvl*2+1..Lvl*3 : IF (i <= Lvl*2+round(Lvl*u[3]/100)) THEN
BEGIN
    dXdt := Fp/Vp[3] * ((x[i-1]+z[i-1]*f)-(x[i]+z[i]*f))
    - U1*Ao[3]/(pw*Cw*Vp[3]) * ((x[i] +z[i] *f)-(x[Lvl*4+3]+z[Lvl*4+3]*f));
END
ELSE
BEGIN
    dXdt := Fp/Vp[3] * ((x[i-1]+z[i-1]*f)-(x[i]+z[i]*f))
    - U2*Ao[3]/(pw*Cw*Vp[3]) * ((x[i] +z[i] *f)-(Ta));
END;
Lvl*3+1..Lvl*4 : IF (i <= Lvl*3+round(Lvl*u[4]/100)) THEN
BEGIN
    dXdt := Fp/Vp[4] * ((x[i-1]+z[i-1]*f)-(x[i]+z[i]*f))
    - U1*Ao[4]/(pw*Cw*Vp[4]) * ((x[i] +z[i] *f)-(x[Lvl*4+4]+z[Lvl*4+4]*f));
END
ELSE
BEGIN
    dXdt := Fp/Vp[4] * ((x[i-1]+z[i-1]*f)-(x[i]+z[i]*f))
    - U2*Ao[4]/(pw*Cw*Vp[4]) * ((x[i] +z[i] *f)-(Ta));
END;
Lvl*4+1 : BEGIN
    temp := Ft/(Vt[1]*u[1]/100) * ((x[i+1]+z[i+1]*f)-(x[i]+z[i]*f));
    FOR j := 1 to round(Lvl*u[1]/100) do
        temp := temp + U1*Ao[1]/(pw*Cw*Vt[1]*u[1]/100) * ((x[Lvl*0+j]+z[Lvl*0+j]*f)-(x[i]+z[i]*f));
    dXdt := temp;
END;
Lvl*4+2 : BEGIN
    temp := Ft/(Vt[2]*u[2]/100) * ((x[i+1]+z[i+1]*f)-(x[i]+z[i]*f));
    FOR j := 1 to round(Lvl*u[2]/100) do
        temp := temp + U1*Ao[2]/(pw*Cw*Vt[2]*u[2]/100) * ((x[Lvl*1+j]+z[Lvl*1+j]*f)-(x[i]+z[i]*f));
    dXdt := temp;
END;
Lvl*4+3 : BEGIN
    temp := Ft/(Vt[3]*u[3]/100) * ((x[i+1]+z[i+1]*f)-(x[i]+z[i]*f));
    FOR j := 1 to round(Lvl*u[3]/100) do
        temp := temp + U1*Ao[3]/(pw*Cw*Vt[3]*u[3]/100) * ((x[Lvl*2+j]+z[Lvl*2+j]*f)-(x[i]+z[i]*f));

```

```

        dxdt := temp;
    END;
Lvl*4+4 : BEGIN
    temp := Ft/(Vt[4]*u[4]/100) * (Tt5          -(x[i]+z[i]*f));
    FOR j := 1 to round(Lvl*u[4]/100) do
        temp := temp + U1*Ao[4]/(pw*Cw*Vt[4]*u[4]/100) * ((x[Lvl*3+j]+z[Lvl*3+j]*f)-(x[i]+z[i]*f));
        dxdt := temp;
    END;
END;
END;

FUNCTION dYdt (i:integer;x:x_var;u:v_var) : double;
BEGIN
    CASE i of
        1 : dYdt := x[Lvl*1];
        2 : dYdt := x[Lvl*2];
        3 : dYdt := x[Lvl*3];
        4 : dYdt := x[Lvl*4];
    END;
END;

{*****}
{***  INITIALIZATION SECTION  ***}
{*****}
BEGIN
    { Heat transfer to water }
    Rew := pw*(Fp/(pi*sqr(Di/2)))*Di/uw;
    Prw := cw*uw/kw;
    Nuw := 0.023*exp(0.8*ln(Rew))*exp(0.4*ln(Prw));

    Hiw := Nuw*kw/Di;
    How := 0.87*kw/Dr*exp((2/3)*ln(Ls*Ts*St*pw/uw))*exp((1/3)*ln(cw*uw/kw));

    U1 := 1/((1/How)+(Dp/(Di*Hiw)));

    { Heat transfer to air }
    Pra := ca*ua/ka;
    Gra := 9.8*(1/(273+(Tw+Ta)/2))*(Tw-Ta)*exp(3*ln(Dp))/exp(2*ln(ua/pa));
    Nua := 0.53*exp((1/4)*ln(Gra*Pra));

    Hia := Nuw*kw/Di;
    Hoa := Nua*ka/Dp;

    U2 := 1/((1/Hoa)+(Dp/(Di*Hia)));

    { Volumes and surface areas of contact }
    Vp[1] := pi * sqr(Di/2) * Lp[1] / Lvl;
    Vp[2] := pi * sqr(Di/2) * Lp[2] / Lvl;
    Vp[3] := pi * sqr(Di/2) * Lp[3] / Lvl;
    Vp[4] := pi * sqr(Di/2) * Lp[4] / Lvl;

    Vt[1] := Vr[1] - pi * sqr(Dp/2) * Lp[1];
    Vt[2] := Vr[2] - pi * sqr(Dp/2) * Lp[2];
    Vt[3] := Vr[3] - pi * sqr(Dp/2) * Lp[3];

```

Vt[4] := Vr[4] - pi * sqr(Dp/2) * Lp[4];

Ao[1] := pi * Dp * Lp[1] / Lvl;

Ao[2] := pi * Dp * Lp[2] / Lvl;

Ao[3] := pi * Dp * Lp[3] / Lvl;

Ao[4] := pi * Dp * Lp[4] / Lvl;

Step[1] := 4.0;

Step[2] := 4.0;

Step[3] := 4.0;

Step[4] := 4.0;

Hght[1] := 20.0;

Hght[2] := 20.0;

Hght[3] := 20.0;

Hght[4] := 20.0;

END.

C.3 : MATRIX HANDLING UNIT

```
UNIT matrix;

{*****}
      INTERFACE
{*****}
USES
  plant;

  PROCEDURE vect_initl (v:vect_ptr);
  PROCEDURE stat_initl (x:stat_ptr);
  PROCEDURE stat_ip2op (u:vect_ptr;x:stat_ptr;y:vect_ptr);

{*****}
      IMPLEMENTATION
{*****}

PROCEDURE vect_initl (v:vect_ptr);
VAR
  r : integer;
BEGIN
  new (v^.o);
  new (v^.n);
  FOR r := 1 to max_cntrl do
    BEGIN
      v^.o^[r] := 0;
      v^.n^[r] := 0;
    END;
  END;

PROCEDURE stat_initl (x:stat_ptr);
VAR
  j : integer;
BEGIN
  new (x^.o);
  new (x^.n);
  FOR j := 1 to max_state DO
    BEGIN
      x^.o^[j] := 0;
      x^.n^[j] := 0;
    END;
  END;

PROCEDURE stat_ip2op (u:vect_ptr;x:stat_ptr;y:vect_ptr);
VAR
  j : integer;
  m : array [0..4] of x_var;
BEGIN
  FOR j := 1 to max_state DO m[0][j] := 0;
  FOR j := 1 to max_state DO m[1][j] := 0;
  FOR j := 1 to max_state DO m[2][j] := 0;
  FOR j := 1 to max_state DO m[3][j] := 0;
```

```
FOR j := 1 to max_state DO m[4][j] := 0;

FOR j := 1 to max_state DO
  m[1][j] := dt * dxdt (j,x^.o^,u^.n^,m[0],0.0);
FOR j := 1 to max_state DO
  m[2][j] := dt * dxdt (j,x^.o^,u^.n^,m[1],0.5);
FOR j := 1 to max_state DO
  m[3][j] := dt * dxdt (j,x^.o^,u^.n^,m[2],0.5);
FOR j := 1 to max_state DO
  m[4][j] := dt * dxdt (j,x^.o^,u^.n^,m[3],1.0);

FOR j := 1 to max_state DO
  x^.n^[j] := x^.o^[j] + (1/6)*(m[1][j] + 2*m[2][j] + 2*m[3][j] + m[4][j]);

FOR j := 1 to max_cntrl DO
  y^.n^[j] := dydt (j,x^.n^,u^.n^);

FOR j := 1 to max_state DO
  x^.o^[j] := x^.n^[j];
END;

{*****}
{***   INITIALIZATION SECTION   ***}
{*****}
BEGIN
END.
```

C.4 : DATA HANDLING UNIT

UNIT data;

{*****}

INTERFACE

{*****}

USES

plant;

TYPE

data_var = record

y : array [0..max_point] of v_ptr;

u : array [0..max_point] of v_ptr;

x : array [0..max_point] of double;

END;

data_ptr = ^data_var;

PROCEDURE data_init (d:data_ptr);

PROCEDURE data_logg (d:data_ptr;y,u:v_ptr;x:double;z:integer);

PROCEDURE data_save (d:data_ptr;f:string);

{*****}

IMPLEMENTATION

{*****}

USES

crt,graph;

PROCEDURE data_init (d:data_ptr);

VAR

i : integer;

BEGIN

FOR i := 0 to max_point do

BEGIN

new (d^.y[i]);

new (d^.u[i]);

END;

END;

PROCEDURE data_logg (d:data_ptr;y,u:v_ptr;x:double;z:integer);

VAR

r,c : integer;

BEGIN

FOR r := 1 to max_cntrl do

BEGIN

d^.y[z]^r := y^r;

d^.u[z]^r := u^r;

END;

d^.x[z] := x;

END;

PROCEDURE data_save (d:data_ptr;f:string);

VAR

```

    i,j    : integer;
    outfile : text;
BEGIN
    assign (outfile,f);
    rewrite (outfile);
    FOR i := 0 to max_point DO
        BEGIN
            FOR j := 1 to max_cntrl DO write (outfile,d^.y[i]^[j]);
            FOR j := 1 to max_cntrl DO write (outfile,d^.u[i]^[j]);
            write  (outfile,d^.x[i]);
            writeln (outfile);
        END;
    close  (outfile);
END;

{*****}
{***  INITIALIZATION  SECTION  ***}
{*****}
BEGIN
END.
```

APPENDIX D : HERIG CONTROLLER SOFTWARE

D.1 : MAIN PROGRAMME

PROGRAM herig;

USES

plant,matrix,data,intrface,
crt,dos;

VAR

filey,fileu : string[12];
outfile : text;

hrs,min,sec,hun : word;
time,told,tlog : real;
exit,ends,logg : boolean;
optn,chce,logs : integer;
loop1,loop2,pts : integer;
ch : char;

dac1,dac2 : array [0..7] of real;
adc1,adc2 : array [0..7] of real;

storey,storeu : data_ptr;
labley,lableu : labl_ptr;

tbeg,ttky,stir : vect_ptr;
flwr,flwe,flwu,flwy,flwf : vect_ptr;
lvlr,lvle,lvlu,lvly,lvlf : vect_ptr;
tppr,tppe,tppu,tppy,tppf : vect_ptr;

xflk,xflf : ssst_ptr;
mflk,mflf : ssmx_ptr;
xlvk,xlvf : ssst_ptr;
mlvk,mlvf : ssmx_ptr;
xtpk,xtpf : ssst_ptr;
mtpk,mtpf : ssmx_ptr;

PROCEDURE prog_varble_initl;

BEGIN

exit := false;

new (storey);data_init (storey);
new (labley);
new (storeu);data_init (storeu);
new (lableu);

```

new (tbeg);vect_initl (tbeg);
new (ttky);vect_initl (ttky);
new (stir);vect_initl (stir);
stir^.n^[1] := 20.0;
stir^.o^[1] := 20.0;
stir^.n^[2] := 20.0;
stir^.o^[2] := 20.0;
stir^.n^[3] := 20.0;
stir^.o^[3] := 20.0;
stir^.n^[4] := 20.0;
stir^.o^[4] := 20.0;

new (flwr);vect_initl (flwr);
new (flwe);vect_initl (flwe);
new (flwu);vect_initl (flwu);
new (flwy);vect_initl (flwy);
new (flwf);vect_initl (flwf);
flwr^.n^[1] := 50.0;flwe^.n^[1] := 0.0;flwu^.n^[1] := 0.0;flwy^.n^[1] := 50.0;flwf^.n^[1] := 50.0;
flwr^.o^[1] := 50.0;flwe^.o^[1] := 0.0;flwu^.o^[1] := 0.0;flwy^.o^[1] := 50.0;flwf^.o^[1] := 50.0;
flwr^.n^[2] := 50.0;flwe^.n^[2] := 0.0;flwu^.n^[2] := 0.0;flwy^.n^[2] := 50.0;flwf^.n^[2] := 50.0;
flwr^.o^[2] := 50.0;flwe^.o^[2] := 0.0;flwu^.o^[2] := 0.0;flwy^.o^[2] := 50.0;flwf^.o^[2] := 50.0;
flwr^.n^[3] := 0.0;flwe^.n^[3] := 0.0;flwu^.n^[3] := 0.0;flwy^.n^[3] := 0.0;flwf^.n^[3] := 0.0;
flwr^.o^[3] := 0.0;flwe^.o^[3] := 0.0;flwu^.o^[3] := 0.0;flwy^.o^[3] := 0.0;flwf^.o^[3] := 0.0;
flwr^.n^[4] := 0.0;flwe^.n^[4] := 0.0;flwu^.n^[4] := 0.0;flwy^.n^[4] := 0.0;flwf^.n^[4] := 0.0;
flwr^.o^[4] := 0.0;flwe^.o^[4] := 0.0;flwu^.o^[4] := 0.0;flwy^.o^[4] := 0.0;flwf^.o^[4] := 0.0;

new (lvlr);vect_initl (lvlr);
new (lvle);vect_initl (lvle);
new (lvlu);vect_initl (lvlu);
new (lvly);vect_initl (lvly);
new (lvlf);vect_initl (lvlf);
lvlr^.n^[1] := 40.0;lvle^.n^[1] := 0.0;lvlu^.n^[1] := 0.0;lvly^.n^[1] := 60.0;lvlf^.n^[1] := 60.0;
lvlr^.o^[1] := 40.0;lvle^.o^[1] := 0.0;lvlu^.o^[1] := 0.0;lvly^.o^[1] := 60.0;lvlf^.o^[1] := 60.0;
lvlr^.n^[2] := 40.0;lvle^.n^[2] := 0.0;lvlu^.n^[2] := 0.0;lvly^.n^[2] := 60.0;lvlf^.n^[2] := 60.0;
lvlr^.o^[2] := 40.0;lvle^.o^[2] := 0.0;lvlu^.o^[2] := 0.0;lvly^.o^[2] := 60.0;lvlf^.o^[2] := 60.0;
lvlr^.n^[3] := 40.0;lvle^.n^[3] := 0.0;lvlu^.n^[3] := 0.0;lvly^.n^[3] := 60.0;lvlf^.n^[3] := 60.0;
lvlr^.o^[3] := 40.0;lvle^.o^[3] := 0.0;lvlu^.o^[3] := 0.0;lvly^.o^[3] := 60.0;lvlf^.o^[3] := 60.0;
lvlr^.n^[4] := 40.0;lvle^.n^[4] := 0.0;lvlu^.n^[4] := 0.0;lvly^.n^[4] := 60.0;lvlf^.n^[4] := 60.0;
lvlr^.o^[4] := 40.0;lvle^.o^[4] := 0.0;lvlu^.o^[4] := 0.0;lvly^.o^[4] := 60.0;lvlf^.o^[4] := 60.0;

new (tppr);vect_initl (tppr);
new (tppe);vect_initl (tppe);
new (tppu);vect_initl (tppu);
new (tppy);vect_initl (tppy);
new (tppf);vect_initl (tppf);
tppr^.n^[1] := 40.0;tppe^.n^[1] := 0.0;tppu^.n^[1] := 40.0;tppy^.n^[1] := 40.0;tppf^.n^[1] := 40.0;
tppr^.o^[1] := 40.0;tppe^.o^[1] := 0.0;tppu^.o^[1] := 40.0;tppy^.o^[1] := 40.0;tppf^.o^[1] := 40.0;
tppr^.n^[2] := 32.0;tppe^.n^[2] := 0.0;tppu^.n^[2] := 40.0;tppy^.n^[2] := 32.0;tppf^.n^[2] := 32.0;
tppr^.o^[2] := 32.0;tppe^.o^[2] := 0.0;tppu^.o^[2] := 40.0;tppy^.o^[2] := 32.0;tppf^.o^[2] := 32.0;
tppr^.n^[3] := 27.0;tppe^.n^[3] := 0.0;tppu^.n^[3] := 40.0;tppy^.n^[3] := 27.0;tppf^.n^[3] := 27.0;
tppr^.o^[3] := 27.0;tppe^.o^[3] := 0.0;tppu^.o^[3] := 40.0;tppy^.o^[3] := 27.0;tppf^.o^[3] := 27.0;
tppr^.n^[4] := 24.0;tppe^.n^[4] := 0.0;tppu^.n^[4] := 40.0;tppy^.n^[4] := 24.0;tppf^.n^[4] := 24.0;
tppr^.o^[4] := 24.0;tppe^.o^[4] := 0.0;tppu^.o^[4] := 40.0;tppy^.o^[4] := 24.0;tppf^.o^[4] := 24.0;

```

```
new (xflk);ssst_initl (xflk);
new (mflk);ssmx_initl (mflk);
ssmx_loads ('kf.ssm',mflk);

new (xflf);ssst_initl (xflf);
new (mflf);ssmx_initl (mflf);
ssmx_loads ('ff.ssm',mflf);

new (xlvk);ssst_initl (xlvk);
new (mlvk);ssmx_initl (mlvk);
ssmx_loads ('kl.ssm',mlvk);

new (xlvf);ssst_initl (xlvf);
new (mlvf);ssmx_initl (mlvf);
ssmx_loads ('fl.ssm',mlvf);

new (xtpk);ssst_initl (xtpk);
new (mtpk);ssmx_initl (mtpk);
ssmx_loads ('kt.ssm',mtpk);

new (xtpf);ssst_initl (xtpf);
new (mtpf);ssmx_initl (mtpf);
ssmx_loads ('ft.ssm',mtpf);

dac_initl (1);
dac_initl (2);
adc_initl (1);
adc_initl (2);

ends := false;
logg := false;

chce := 1;
logs := 1;

told := 0;
settime (0,0,0,0);
END;

PROCEDURE prog_varble_reads;
BEGIN
  FOR loop1 := 0 to 7 DO
    BEGIN
      adc_reads (1,loop1,adc1[loop1]);
      CASE loop1 OF
        0   : tppy^.n^[loop1+4] := (adc1[loop1] + 212.8) / 51.55;
        1..4 : ttky^.n^[5-loop1] := (adc1[loop1] + 212.8) / 51.55;
        5   : tbeg^.n^[loop1-3] := (adc1[loop1] + 212.8) / 51.55;
        6..7 : tppy^.n^[loop1-5] := (adc1[loop1] + 212.8) / 51.55;
      END;
      adc_reads (2,loop1,adc2[loop1]);
      CASE loop1 OF
        0..1 : flwy^.n^[loop1+1] := 100 * (adc2[loop1] - 819) / (4095 - 819);
        2..5 : lvlv^.n^[loop1-1] := 100 * (adc2[loop1]/4095);
```

```

6   : tppy^.n^[loop1-3] := (adc2[loop1] + 212.8) / 51.55;
7   : tbeq^.n^[loop1-6] := (adc2[loop1] + 212.8) / 51.55;
END;
END;
END;

```

PROCEDURE prog_varble_write;

```

BEGIN
FOR loop1 := 0 to 7 DO
BEGIN
CASE loop1 OF
0..3 : dac1[loop1] := stir^.n^[loop1+1] * 40.95;
END;
dac_write (1,loop1,dac1[loop1]);
CASE loop1 OF
0 : dac2[loop1] := flwu^.n^[loop1+1] * 40.95;
1..4 : dac2[loop1] := lvlu^.n^[5-loop1] * 40.95;
5 : dac2[loop1] := flwu^.n^[loop1-3] * 40.95;
END;
dac_write (2,loop1,dac2[loop1]);
END;
END;

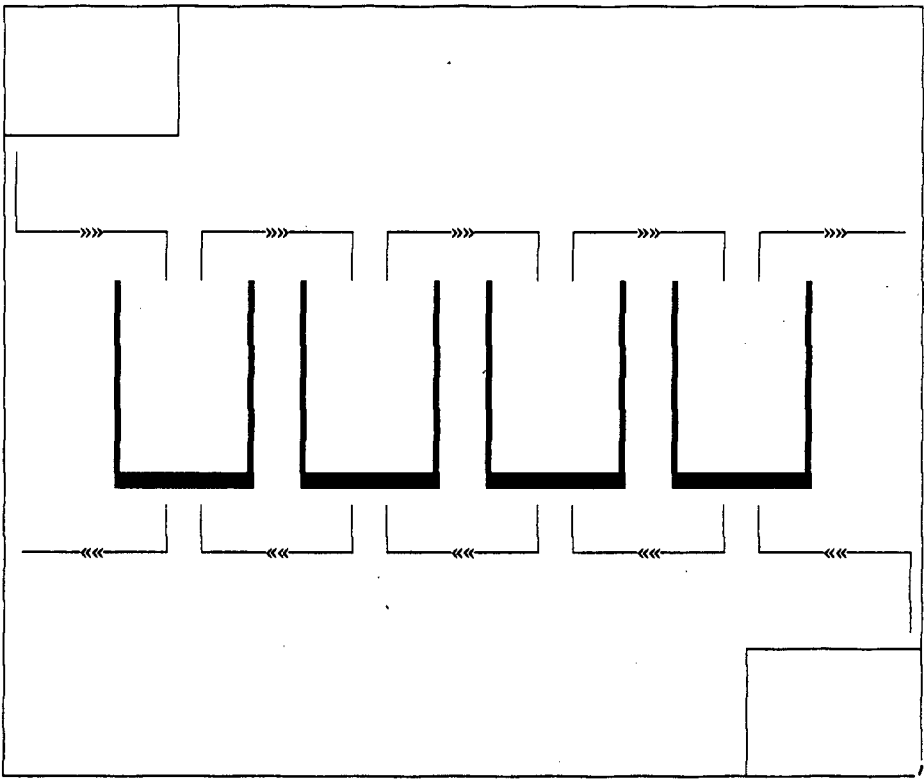
```

PROCEDURE prog_screen_init1;

```

BEGIN
clrscr;
gotoxy(1, 1);write ('');
gotoxy(1, 2);write ('');
gotoxy(1, 3);write ('');
gotoxy(1, 4);write ('');
gotoxy(1, 5);write ('');
gotoxy(1, 6);write ('');
gotoxy(1, 7);write ('');
gotoxy(1, 8);write ('');
gotoxy(1, 9);write ('');
gotoxy(1,10);write ('');
gotoxy(1,11);write ('');
gotoxy(1,12);write ('');
gotoxy(1,13);write ('');
gotoxy(1,14);write ('');
gotoxy(1,15);write ('');
gotoxy(1,16);write ('');
gotoxy(1,17);write ('');
gotoxy(1,18);write ('');
gotoxy(1,19);write ('');
gotoxy(1,20);write ('');
gotoxy(1,21);write ('');
gotoxy(1,22);write ('');
gotoxy(1,23);write ('');
gotoxy(1,24);write ('');
gotoxy(1,25);write ('');
END;

```



PROCEDURE check_keys;

```
PROCEDURE key_ShF1;  
BEGIN  
  IF (stir^.n^[1] <= 90) THEN  
    stir^.n^[1] := stir^.n^[1] + 10.0  
  ELSE  
    stir^.n^[1] := 100.0;  
END;
```

```
PROCEDURE key_CtF1;  
BEGIN  
  IF (stir^.n^[1] >= 10) THEN  
    stir^.n^[1] := stir^.n^[1] - 10.0  
  ELSE  
    stir^.n^[1] := 0.0;  
END;
```

```
PROCEDURE key_ShF2;  
BEGIN  
  IF (stir^.n^[2] <= 90) THEN  
    stir^.n^[2] := stir^.n^[2] + 10.0  
  ELSE  
    stir^.n^[2] := 100.0;  
END;
```

```
PROCEDURE key_CtF2;  
BEGIN  
  IF (stir^.n^[2] >= 10) THEN  
    stir^.n^[2] := stir^.n^[2] - 10.0  
  ELSE  
    stir^.n^[2] := 0.0;  
END;
```

```
PROCEDURE key_ShF3;  
BEGIN  
  IF (stir^.n^[3] <= 90) THEN  
    stir^.n^[3] := stir^.n^[3] + 10.0  
  ELSE  
    stir^.n^[3] := 100.0;  
END;
```

```
PROCEDURE key_CtF3;  
BEGIN  
  IF (stir^.n^[3] >= 10) THEN  
    stir^.n^[3] := stir^.n^[3] - 10.0  
  ELSE  
    stir^.n^[3] := 0.0;  
END;
```

```
PROCEDURE key_ShF4;  
BEGIN  
  IF (stir^.n^[4] <= 90) THEN  
    stir^.n^[4] := stir^.n^[4] + 10.0  
  ELSE
```

```
    stir^.n^[4] := 100.0;
END;

PROCEDURE key_CtF4;
BEGIN
    IF (stir^.n^[4] >= 10) THEN
        stir^.n^[4] := stir^.n^[4] - 10.0
    ELSE
        stir^.n^[4] := 0.0;
    END;
END;

PROCEDURE key_lfar;
BEGIN
    CASE optn OF
        1 : IF (chce > 1) THEN dec (chce);
        2 : IF (chce > 1) THEN dec (chce);
        3 : IF (chce > 1) THEN dec (chce);
        4 : IF (chce > 1) THEN dec (chce);
    END;
    logs := 1;
END;

PROCEDURE key_rgar;
BEGIN
    CASE optn OF
        1 : IF (chce < 2) THEN inc (chce);
        2 : IF (chce < 2) THEN inc (chce);
        3 : IF (chce < 1) THEN inc (chce);
        4 : IF (chce < 1) THEN inc (chce);
    END;
    logs := 1;
END;

PROCEDURE key_upar;
BEGIN
    CASE optn OF
        1 : CASE chce OF
            1 : IF (logs > 1) THEN dec (logs);
            2 : IF (logs > 1) THEN dec (logs);
            END;
        2 : CASE chce OF
            1 : IF (logs > 1) THEN dec (logs);
            2 : IF (logs > 1) THEN dec (logs);
            END;
        3 : CASE chce OF
            1 : IF (logs > 1) THEN dec (logs);
            END;
        4 : CASE chce OF
            1 : IF (logs > 1) THEN dec (logs);
            END;
    END;
END;

PROCEDURE key_dnar;
```

BEGIN

CASE optn OF

1 : CASE chce OF

1 : IF (logs < 2) THEN inc (logs);

2 : IF (logs < 4) THEN inc (logs);

END;

2 : CASE chce OF

1 : IF (logs < 2) THEN inc (logs);

2 : IF (logs < 4) THEN inc (logs);

END;

3 : CASE chce OF

1 : IF (logs < 4) THEN inc (logs);

END;

4 : CASE chce OF

1 : IF (logs < 4) THEN inc (logs);

END;

END;

END;

PROCEDURE key_pgup;

BEGIN

CASE optn OF

1 : CASE chce OF

1 : IF (flwu^.n^[logs] <= 90) THEN

flwu^.n^[logs] := flwu^.n^[logs] + 10.0

ELSE

flwu^.n^[logs] := 100.0;

2 : IF (lvlu^.n^[logs] <= 90) THEN

lvlu^.n^[logs] := lvlu^.n^[logs] + 10.0

ELSE

lvlu^.n^[logs] := 100.0;

END;

2 : CASE chce OF

1 : IF (flwr^.n^[logs] <= 90) THEN

flwr^.n^[logs] := flwr^.n^[logs] + 10.0

ELSE

flwr^.n^[logs] := 100.0;

2 : IF (lvlr^.n^[logs] <= 90) THEN

lvlr^.n^[logs] := lvlr^.n^[logs] + 10.0

ELSE

lvlr^.n^[logs] := 100.0;

END;

3 : CASE chce OF

1 : IF (tppu^.n^[logs] <= 90) THEN

tppu^.n^[logs] := tppu^.n^[logs] + 10.0

ELSE

tppu^.n^[logs] := 100.0;

END;

4 : CASE chce OF

1 : IF (tppr^.n^[logs] <= 59) THEN

tppr^.n^[logs] := tppr^.n^[logs] + 0.5

ELSE

tppr^.n^[logs] := 60.0;

END;

END;
END;

PROCEDURE key_pgdn;

BEGIN

CASE optn OF

1 : CASE chce OF

1 : IF (flwu^.n^[logs] >= 10) THEN

flwu^.n^[logs] := flwu^.n^[logs] - 10.0

ELSE

flwu^.n^[logs] := 0.0;

2: IF (lvlu^.n^[logs] >= 10) THEN

lvlu^.n^[logs] := lvlu^.n^[logs] - 10.0

ELSE

lvlu^.n^[logs] := 0.0;

END;

2 : CASE chce OF

1 : IF (flwr^.n^[logs] >= 20) THEN

flwr^.n^[logs] := flwr^.n^[logs] - 10.0

ELSE

flwr^.n^[logs] := 10.0;

2: IF (lvlr^.n^[logs] >= 10) THEN

lvlr^.n^[logs] := lvlr^.n^[logs] - 10.0

ELSE

lvlr^.n^[logs] := 0.0;

END;

3 : CASE chce OF

1 : IF (tppu^.n^[logs] >= 10) THEN

tppu^.n^[logs] := tppu^.n^[logs] - 10.0

ELSE

tppu^.n^[logs] := 0.0;

END;

4 : CASE chce OF

1 : IF (tppr^.n^[logs] >= 21) THEN

tppr^.n^[logs] := tppr^.n^[logs] - 0.5

ELSE

tppr^.n^[logs] := 20.0;

END;

END;

END;

PROCEDURE key_atl1;

BEGIN

IF NOT logg THEN

BEGIN

pts := 0;

logg := true;

gettime (hrs,min,sec,hun);

tlog := hrs*3600 + min*60 + sec*1 + hun*0.01;

END

ELSE IF logg THEN

BEGIN

pts := 0;

logg := false;

```
textbackground (lightgray);textcolor (black);
gotoxy (26,04);write ('AltL: LOGS');
textbackground (black);textcolor (green);
END;

CASE optn OF
1 : CASE chce OF
  1 : BEGIN
    filey := 'gf_0.dat';filey[4] := chr(48+logs);
    END;
  2 : BEGIN
    filey := 'gl_0.dat';filey[4] := chr(48+logs);
    END;
  END;
2 : CASE chce OF
  1 : BEGIN
    filey := 'hf_0.dat';filey[4] := chr(48+logs);
    fileu := 'uf_0.dat';fileu[4] := chr(48+logs);
    END;
  2 : BEGIN
    filey := 'hl_0.dat';filey[4] := chr(48+logs);
    fileu := 'ul_0.dat';fileu[4] := chr(48+logs);
    END;
  END;
3 : CASE chce OF
  1 : BEGIN
    filey := 'gt_0.dat';filey[4] := chr(48+logs);
    END;
  END;
4 : CASE chce OF
  1 : BEGIN
    filey := 'ht_0.dat';filey[4] := chr(48+logs);
    fileu := 'ut_0.dat';fileu[4] := chr(48+logs);
    END;
  END;
END;

CASE optn OF
1 : CASE chce OF
  1 : BEGIN
    labley^.head1 := 'HEAT EXCHANGER RIG : FLOW CONTROL';
    labley^.head2 := 'OPEN LOOP STEP TEST';
    labley^.yname[1] := 'Flow F1y';labley^.ylabl[1] := 'Y';labley^.yunit[1] := '%';
    labley^.yname[2] := 'Flow F2y';labley^.ylabl[2] := 'Y';labley^.yunit[2] := '%';
    labley^.yname[3] := ''; labley^.ylabl[3] := ''; labley^.yunit[3] := '';
    labley^.yname[4] := ''; labley^.ylabl[4] := ''; labley^.yunit[4] := '';

    labley^.uname[1] := 'Valv C1 ';labley^.ulabl[1] := 'U';labley^.uunit[1] := '%';
    labley^.uname[2] := 'Valv C6 ';labley^.ulabl[2] := 'U';labley^.uunit[2] := '%';
    labley^.uname[3] := ''; labley^.ulabl[3] := ''; labley^.uunit[3] := '';
    labley^.uname[4] := ''; labley^.ulabl[4] := ''; labley^.uunit[4] := '';

    labley^.xlabl := 's';
  END;
```

```

2 : BEGIN
  labley^.head1 := 'HEAT EXCHANGER RIG : LEVEL CONTROL';
  labley^.head2 := 'OPEN LOOP STEP TEST';
  labley^.yname[1] := 'Levl L1y'; labley^.ylabl[1] := 'Y'; labley^.yunit[1] := '%';
  labley^.yname[2] := 'Levl L2y'; labley^.ylabl[2] := 'Y'; labley^.yunit[2] := '%';
  labley^.yname[3] := 'Levl L3y'; labley^.ylabl[3] := 'Y'; labley^.yunit[3] := '%';
  labley^.yname[4] := 'Levl L4y'; labley^.ylabl[4] := 'Y'; labley^.yunit[4] := '%';

  labley^.uname[1] := 'Valv C5'; labley^.ulabl[1] := 'U'; labley^.uunit[1] := '%';
  labley^.uname[2] := 'Valv C4'; labley^.ulabl[2] := 'U'; labley^.uunit[2] := '%';
  labley^.uname[3] := 'Valv C3'; labley^.ulabl[3] := 'U'; labley^.uunit[3] := '%';
  labley^.uname[4] := 'Valv C2'; labley^.ulabl[4] := 'U'; labley^.uunit[4] := '%';

  labley^.xlabl := 's';
END;
END;
2 : CASE chce OF
1 : BEGIN
  labley^.head1 := 'HEAT EXCHANGER RIG : FLOW CONTROL';
  labley^.head2 := 'CLOSED LOOP STEP TEST';
  labley^.yname[1] := 'Flow F1y'; labley^.ylabl[1] := 'Y'; labley^.yunit[1] := '%';
  labley^.yname[2] := 'Flow F2y'; labley^.ylabl[2] := 'Y'; labley^.yunit[2] := '%';
  labley^.yname[3] := ''; labley^.ylabl[3] := ''; labley^.yunit[3] := '';
  labley^.yname[4] := ''; labley^.ylabl[4] := ''; labley^.yunit[4] := '';

  labley^.uname[1] := 'Flow F1r'; labley^.ulabl[1] := 'R'; labley^.uunit[1] := '%';
  labley^.uname[2] := 'Flow F2r'; labley^.ulabl[2] := 'R'; labley^.uunit[2] := '%';
  labley^.uname[3] := ''; labley^.ulabl[3] := ''; labley^.uunit[3] := '';
  labley^.uname[4] := ''; labley^.ulabl[4] := ''; labley^.uunit[4] := '';

  labley^.xlabl := 's';

  lableu^.head1 := 'HEAT EXCHANGER RIG : FLOW CONTROL';
  lableu^.head2 := 'CLOSED LOOP STEP TEST';
  lableu^.yname[1] := 'Valv C1'; lableu^.ylabl[1] := 'U'; lableu^.yunit[1] := '%';
  lableu^.yname[2] := 'Valv C6'; lableu^.ylabl[2] := 'U'; lableu^.yunit[2] := '%';
  lableu^.yname[3] := ''; lableu^.ylabl[3] := ''; lableu^.yunit[3] := '';
  lableu^.yname[4] := ''; lableu^.ylabl[4] := ''; lableu^.yunit[4] := '';

  lableu^.uname[1] := 'Flow F1r'; lableu^.ulabl[1] := 'R'; lableu^.uunit[1] := '%';
  lableu^.uname[2] := 'Flow F2r'; lableu^.ulabl[2] := 'R'; lableu^.uunit[2] := '%';
  lableu^.uname[3] := ''; lableu^.ulabl[3] := ''; lableu^.uunit[3] := '';
  lableu^.uname[4] := ''; lableu^.ulabl[4] := ''; lableu^.uunit[4] := '';

  lableu^.xlabl := 's';
END;
2 : BEGIN
  labley^.head1 := 'HEAT EXCHANGER RIG : LEVEL CONTROL';
  labley^.head2 := 'OPEN LOOP STEP TEST';
  labley^.yname[1] := 'Levl L1y'; labley^.ylabl[1] := 'Y'; labley^.yunit[1] := '%';
  labley^.yname[2] := 'Levl L2y'; labley^.ylabl[2] := 'Y'; labley^.yunit[2] := '%';
  labley^.yname[3] := 'Levl L3y'; labley^.ylabl[3] := 'Y'; labley^.yunit[3] := '%';
  labley^.yname[4] := 'Levl L4y'; labley^.ylabl[4] := 'Y'; labley^.yunit[4] := '%';

```

```

labley^.uname[1] := 'Levl L1r';labley^.ulabl[1] := 'R';labley^.uunit[1] := '%';
labley^.uname[2] := 'Levl L2r';labley^.ulabl[2] := 'R';labley^.uunit[2] := '%';
labley^.uname[3] := 'Levl L3r';labley^.ulabl[3] := 'R';labley^.uunit[3] := '%';
labley^.uname[4] := 'Levl L4r';labley^.ulabl[4] := 'R';labley^.uunit[4] := '%';

labley^.xlabl := 's';

lableu^.head1 := 'HEAT EXCHANGER RIG : LEVEL CONTROL';
lableu^.head2 := 'OPEN LOOP STEP TEST';
lableu^.yname[1] := 'Valv C5';lableu^.ylabl[1] := 'U';lableu^.yunit[1] := '%';
lableu^.yname[2] := 'Valv C4';lableu^.ylabl[2] := 'U';lableu^.yunit[2] := '%';
lableu^.yname[3] := 'Valv C3';lableu^.ylabl[3] := 'U';lableu^.yunit[3] := '%';
lableu^.yname[4] := 'Valv C2';lableu^.ylabl[4] := 'U';lableu^.yunit[4] := '%';

lableu^.uname[1] := 'Levl L1r';lableu^.ulabl[1] := 'R';lableu^.uunit[1] := '%';
lableu^.uname[2] := 'Levl L2r';lableu^.ulabl[2] := 'R';lableu^.uunit[2] := '%';
lableu^.uname[3] := 'Levl L3r';lableu^.ulabl[3] := 'R';lableu^.uunit[3] := '%';
lableu^.uname[4] := 'Levl L4r';lableu^.ulabl[4] := 'R';lableu^.uunit[4] := '%';

lableu^.xlabl := 's';
END;
END;
3 : CASE chce OF
1 : BEGIN
labley^.head1 := 'HEAT EXCHANGER RIG : TEMPERATURE CONTROL';
labley^.head2 := 'OPEN LOOP STEP TEST';
labley^.yname[1] := 'Temp T7y';labley^.ylabl[1] := 'Y';labley^.yunit[1] := 'C';
labley^.yname[2] := 'Temp T8y';labley^.ylabl[2] := 'Y';labley^.yunit[2] := 'C';
labley^.yname[3] := 'Temp T9y';labley^.ylabl[3] := 'Y';labley^.yunit[3] := 'C';
labley^.yname[4] := 'Temp T0y';labley^.ylabl[4] := 'Y';labley^.yunit[4] := 'C';

labley^.uname[1] := 'Levl L1r';labley^.ulabl[1] := 'U';labley^.uunit[1] := '%';
labley^.uname[2] := 'Levl L2r';labley^.ulabl[2] := 'U';labley^.uunit[2] := '%';
labley^.uname[3] := 'Levl L3r';labley^.ulabl[3] := 'U';labley^.uunit[3] := '%';
labley^.uname[4] := 'Levl L4r';labley^.ulabl[4] := 'U';labley^.uunit[4] := '%';

labley^.xlabl := 's';
END;
END;
4 : CASE chce OF
1 : BEGIN
labley^.head1 := 'HEAT EXCHANGER RIG : TEMPERATURE CONTROL';
labley^.head2 := 'CLOSED LOOP STEP TEST';
labley^.yname[1] := 'Temp T7y';labley^.ylabl[1] := 'Y';labley^.yunit[1] := 'C';
labley^.yname[2] := 'Temp T8y';labley^.ylabl[2] := 'Y';labley^.yunit[2] := 'C';
labley^.yname[3] := 'Temp T9y';labley^.ylabl[3] := 'Y';labley^.yunit[3] := 'C';
labley^.yname[4] := 'Temp T0y';labley^.ylabl[4] := 'Y';labley^.yunit[4] := 'C';

labley^.uname[1] := 'Temp T7r';labley^.ulabl[1] := 'R';labley^.uunit[1] := 'C';
labley^.uname[2] := 'Temp T8r';labley^.ulabl[2] := 'R';labley^.uunit[2] := 'C';
labley^.uname[3] := 'Temp T9r';labley^.ulabl[3] := 'R';labley^.uunit[3] := 'C';
labley^.uname[4] := 'Temp T0r';labley^.ulabl[4] := 'R';labley^.uunit[4] := 'C';

labley^.xlabl := 's';

```

```
    lableu^.head1 := 'HEAT EXCHANGER RIG : TEMPERATURE CONTROL';
    lableu^.head2 := 'CLOSED LOOP STEP TEST';
    lableu^.yname[1] := 'Levl L1r';lableu^.ylabl[1] := 'U';lableu^.yunit[1] := '%';
    lableu^.yname[2] := 'Levl L2r';lableu^.ylabl[2] := 'U';lableu^.yunit[2] := '%';
    lableu^.yname[3] := 'Levl L3r';lableu^.ylabl[3] := 'U';lableu^.yunit[3] := '%';
    lableu^.yname[4] := 'Levl L4r';lableu^.ylabl[4] := 'U';lableu^.yunit[4] := '%';

    lableu^.uname[1] := 'Temp T7r';lableu^.ulabl[1] := 'R';lableu^.uunit[1] := 'C';
    lableu^.uname[2] := 'Temp T8r';lableu^.ulabl[2] := 'R';lableu^.uunit[2] := 'C';
    lableu^.uname[3] := 'Temp T9r';lableu^.ulabl[3] := 'R';lableu^.uunit[3] := 'C';
    lableu^.uname[4] := 'Temp T0r';lableu^.ulabl[4] := 'R';lableu^.uunit[4] := 'C';

    lableu^.xlabl := 's';
  END;
END;
END;
END;

PROCEDURE key_altp;
BEGIN
  data_plot (storey,labley);
  CASE optn OF
    1 : CASE chce OF
      1 : ;
      2 : ;
    END;
    2 : CASE chce OF
      1 : data_plot (storeu,lableu);
      2 : data_plot (storeu,lableu);
    END;
    3 : CASE chce OF
      1 : ;
    END;
    4 : CASE chce OF
      1 : data_plot (storeu,lableu);
    END;
  END;
  prog_screen_initl;
END;

PROCEDURE key_alts;
BEGIN
  textbackground (lightgray);textcolor (black);
  gotoxy (50,04);write (' SAVING ');

  data_save (storey,labley,filey);
  CASE optn OF
    1 : CASE chce OF
      1 : ;
      2 : ;
    END;
    2 : CASE chce OF
      1 : data_save (storeu,lableu,fileu);
```

```
2 : data_save (storeu,lableu,fileu);
END;
3 : CASE chce OF
1 : ;
END;
4 : CASE chce OF
1 : data_save (storeu,lableu,fileu);
END;
END;

gotoxy (50,04);write ('Alts: SAVE');
textbackground (black);textcolor (green);
END;

PROCEDURE key_altx;
BEGIN
ends := true;
END;

BEGIN
ch := upcase(readkey);
IF (ord(ch) = 0) THEN ch := upcase(readkey);
CASE ch OF
chr(84) : key_ShF1;
chr(85) : key_ShF2;
chr(86) : key_ShF3;
chr(87) : key_ShF4;
chr(94) : key_CtF1;
chr(95) : key_CtF2;
chr(96) : key_CtF3;
chr(65) : key_CtF4;
chr(75) : key_lfar;
chr(77) : key_rgar;
chr(72) : key_upar;
chr(80) : key_dnar;
chr(73) : key_pgup;
chr(81) : key_pgdn;
chr(38) : key_altl;
chr(25) : key_altp;
chr(31) : key_alts;
chr(45) : key_altx;
END;
END;

PROCEDURE check_logg;
BEGIN
gettime (hrs,min,sec,hun);
time := hrs*3600 + min*60 + sec*1 + hun*0.01;
IF (logg) AND (pts <= max_point) AND (round((time-tlog)/dx) = pts) THEN
BEGIN
textbackground (lightgray);textcolor (black);
gotoxy (26,04);write ('PTS =',pts:5);
textbackground (black);textcolor (green);
```

```

CASE optn OF
1 : CASE chce OF
    1 : data_logg (pts,storey,flwy^.n,flwu^.n,pts*dx);
    2 : data_logg (pts,storey,lvly^.n,lvlu^.n,pts*dx);
    END;
    2 : CASE chce OF
    1 : data_logg (pts,storey,flwy^.n,flwr^.n,pts*dx);
    2 : data_logg (pts,storey,lvly^.n,lvlr^.n,pts*dx);
    END;
    3 : CASE chce OF
    1 : data_logg (pts,storey,tppy^.n,tppu^.n,pts*dx);
    END;
    4 : CASE chce OF
    1 : data_logg (pts,storey,tppy^.n,tppr^.n,pts*dx);
    END;
END;

CASE optn OF
1 : CASE chce OF
    1 : ;
    2 : ;
    END;
    2 : CASE chce OF
    1 : data_logg (pts,storeu,flwu^.n,flwr^.n,pts*dx);
    2 : data_logg (pts,storeu,lvlu^.n,lvlr^.n,pts*dx);
    END;
    3 : CASE chce OF
    1 : ;
    END;
    4 : CASE chce OF
    1 : data_logg (pts,storeu,tppu^.n,tppr^.n,pts*dx);
    END;
END;

inc (pts);
END
ELSE IF (logg) AND (pts > max_point) THEN
BEGIN
    textbackground (lightgray);textcolor (black);
    gotoxy (26,04);write ('AltL: LOGS');
    textbackground (black);textcolor (green);
    logg := false;
    pts := 0;
END;
END;

{*****}
{****  OPTION 1 : OPEN LOOP FLOW & LEVEL CONTROL  ****}
{*****}
PROCEDURE control_1;

PROCEDURE ctrl_screen_initl;
BEGIN
    textbackground (lightgray);textcolor (black);

```

```
gotoxy (20,02);write (' HEAT EXCHANGER RIG : [1] OPEN LOOP FLOW & LEVEL CONTROL ');
gotoxy (26,04);write ('AltL: LOGS');
gotoxy (38,04);write ('AltP: PLOT');
gotoxy (50,04);write ('AltS: SAVE');
gotoxy (62,04);write ('AltX: EXIT');
textbackground (black);textcolor (green);
```

```
highvideo;
gotoxy (03,02);write ('CHR : SLURRY');
normvideo;
gotoxy (04,03);
gotoxy (04,04);write ('F2y=  %');
gotoxy (04,07);write ('T6 =   C');
gotoxy (04,09);write ('C6 =  %');
```

```
FOR loop1 := 1 to 4 DO
```

```
  BEGIN
```

```
    highvideo;
```

```
    gotoxy (loop1*16- 3,10);write ('TANK no',loop1:1);
```

```
    normvideo;
```

```
    IF (loop1 <> 4) THEN
```

```
      BEGIN
```

```
        gotoxy (loop1*16+ 4,06);
```

```
        gotoxy (loop1*16+ 4,07);write ('T',(loop1+6):1,'y=   C');
```

```
      END
```

```
    ELSE
```

```
      BEGIN
```

```
        gotoxy (loop1*16+ 4,06);
```

```
        gotoxy (loop1*16+ 4,07);write ('T0',      'y=   C');
```

```
      END;
```

```
    gotoxy (loop1*16- 4,11);
```

```
    gotoxy (loop1*16- 4,12);write ('S',loop1:1  ,' =  %');
```

```
    gotoxy (loop1*16- 4,13);
```

```
    gotoxy (loop1*16- 4,14);
```

```
    gotoxy (loop1*16- 4,15);write ('L',loop1:1  ,'y=  %');
```

```
    gotoxy (loop1*16-12,17);write ('C',(6-loop1):1,' =  %');
```

```
    gotoxy (loop1*16-12,19);write ('T',(6-loop1):1,'y=  C');
```

```
    gotoxy (loop1*16-12,20);
```

```
  END;
```

```
gotoxy (68,17);write ('C1 =  %');
```

```
gotoxy (68,19);write ('T1 =   C');
```

```
gotoxy (68,22);write ('Fly=  %');
```

```
gotoxy (68,23);
```

```
highvideo;
```

```
gotoxy (67,24);write ('CCR : CARBON');
```

```
normvideo;
```

```
IF (chce = 1) THEN
```

```
  BEGIN
```

```
    textbackground (lightgray);textcolor (black);
```

```
    gotoxy (04,22);write ('1:O/L FLOW');
```

```
    gotoxy (04,23);write (' VLVE no',(5*logs-4):1,' ');
```

```
    textbackground (black);textcolor (green);
```

```

END
ELSE
BEGIN
  gotoxy (04,22);write ('1:O/L FLOW');
  gotoxy (04,23);write (' ');
END;

IF (chce = 2) THEN
BEGIN
  textbackground (lightgray);textcolor (black);
  gotoxy (16,22);write ('2:O/L LEVL');
  gotoxy (16,23);write (' VLVE no',(6-logs):1,' ');
  textbackground (black);textcolor (green);
END
ELSE
BEGIN
  gotoxy (16,22);write ('2:O/L LEVL');
  gotoxy (16,23);write (' ');
END;
END;

PROCEDURE ctrl_screen_updat;
BEGIN
  gotoxy (08,03);
  gotoxy (08,04);write (flwy^.n^[2]:5:1);
  gotoxy (08,07);write (tbeg^.n^[2]:5:1);
  gotoxy (08,09);write (flwu^.n^[2]:5:1);

  FOR loop1 := 1 to 4 DO
  BEGIN
    gotoxy (loop1*16+ 8,06);
    gotoxy (loop1*16+ 8,07);write (tppy^.n^[loop1]:5:1);
    gotoxy (loop1*16 ,11);
    gotoxy (loop1*16 ,12);write (stir^.n^[loop1]:5:1);
    gotoxy (loop1*16 ,13);
    gotoxy (loop1*16 ,14);
    gotoxy (loop1*16 ,15);write (lvly^.n^[loop1]:5:1);
    gotoxy (loop1*16- 8,17);write (lvlu^.n^[loop1]:5:1);
    gotoxy (loop1*16- 8,19);write (ttky^.n^[loop1]:5:1);
    gotoxy (loop1*16- 8,20);
  END;

  gotoxy (72,17);write (flwu^.n^[1]:5:1);
  gotoxy (72,19);write (tbeg^.n^[1]:5:1);
  gotoxy (72,22);write (flwy^.n^[1]:5:1);
  gotoxy (72,23);
END;

BEGIN
  prog_screen_initl;
  ctrl_screen_initl;
  REPEAT
    prog_varble_reads;

```

```
{#####}
{      IMPLEMENTING  CONTROL LOOP      }
{#####}
gettime (hrs,min,sec,hun);
time := hrs*3600 + min*60 + sec*1 + hun*0.01;
dt := time - told;
told := hrs*3600 + min*60 + sec*1 + hun*0.01;
{#####}

prog_varble_write;

ctrl_screen_updat;

IF keypressed THEN
  BEGIN
    check_keys;
    ctrl_screen_initl;
  END;
  check_logg;
UNTIL ends;
END;

{#####}
{*****  OPTION 2 : CLOSED LOOP FLOW & LEVEL CONTROL  *****}
{#####}
PROCEDURE control_2;

PROCEDURE ctr2_screen_initl;
BEGIN
  textbackground (lightgray);textcolor (black);
  gotoxy (20,02);write (' HEAT EXCHANGER RIG : [2] CLOSED LOOP FLOW & LEVEL CONTROL');
  gotoxy (26,04);write ('AltL: LOGS');
  gotoxy (38,04);write ('AltP: PLOT');
  gotoxy (50,04);write ('AltS: SAVE');
  gotoxy (62,04);write ('AltX: EXIT');
  textbackground (black);textcolor (green);

  highvideo;
  gotoxy (03,02);write ('CHR : SLURRY');
  normvideo;
  gotoxy (04,03);write ('F2r=   %');
  gotoxy (04,04);write ('F2y=   %');
  gotoxy (04,07);write ('T6 =   C');
  gotoxy (04,09);write ('C6 =   %');

  FOR loop1 := 1 to 4 DO
    BEGIN
      highvideo;
      gotoxy (loop1*16- 3,10);write ('TANK no',loop1:1);
      normvideo;
      IF (loop1 <> 4) THEN
        BEGIN
          gotoxy (loop1*16+ 4,06);
          gotoxy (loop1*16+ 4,07);write ('T',(loop1+6):1,'y=   C');
```

```
END
ELSE
BEGIN
  gotoxy (loop1*16+ 4,06);
  gotoxy (loop1*16+ 4,07);write ('T0',      'y=    C');
END;
gotoxy (loop1*16- 4,11);
gotoxy (loop1*16- 4,12);write ('S',loop1:1    , ' =    %');
gotoxy (loop1*16- 4,13);
gotoxy (loop1*16- 4,14);write ('L',loop1:1    , 'r=    %');
gotoxy (loop1*16- 4,15);write ('L',loop1:1    , 'y=    %');
gotoxy (loop1*16-12,17);write ('C',(6-loop1):1, ' =    %');
gotoxy (loop1*16-12,19);write ('T',(6-loop1):1, 'y=    C');
gotoxy (loop1*16-12,20);
END;

gotoxy (68,17);write ('C1 =    %');
gotoxy (68,19);write ('T1 =    C');
gotoxy (68,22);write ('Fly=    %');
gotoxy (68,23);write ('Flr=    %');
highvideo;
gotoxy (67,24);write ('CCR : CARBON');
normvideo;

IF (chce = 1) THEN
BEGIN
  textbackground (lightgray);textcolor (black);
  gotoxy (04,22);write ('1:C/L FLOW');
  gotoxy (04,23);write (' FLOW no',logs:1, ' ');
  textbackground (black);textcolor (green);
END
ELSE
BEGIN
  gotoxy (04,22);write ('1:C/L FLOW');
  gotoxy (04,23);write (' ');
END;

IF (chce = 2) THEN
BEGIN
  textbackground (lightgray);textcolor (black);
  gotoxy (16,22);write ('2:C/L LEVL');
  gotoxy (16,23);write (' LEVL no',logs:1, ' ');
  textbackground (black);textcolor (green);
END
ELSE
BEGIN
  gotoxy (16,22);write ('2:C/L LEVL');
  gotoxy (16,23);write (' ');
END;

PROCEDURE ctr2_screen_updat;
BEGIN
  gotoxy (08,03);write (flwr^.n^[2]:5:1);
```

```
gotoxy (08,04);write (flwy^.n^[2]:5:1);
gotoxy (08,07);write (tbeg^.n^[2]:5:1);
gotoxy (08,09);write (flwu^.n^[2]:5:1);

FOR loop1 := 1 to 4 DO
  BEGIN
    gotoxy (loop1*16+ 8,06);
    gotoxy (loop1*16+ 8,07);write (tppy^.n^[loop1]:5:1);
    gotoxy (loop1*16 ,11);
    gotoxy (loop1*16 ,12);write (stir^.n^[loop1]:5:1);
    gotoxy (loop1*16 ,13);
    gotoxy (loop1*16 ,14);write (lvlr^.n^[loop1]:5:1);
    gotoxy (loop1*16 ,15);write (lvly^.n^[loop1]:5:1);
    gotoxy (loop1*16- 8,17);write (lvlu^.n^[loop1]:5:1);
    gotoxy (loop1*16- 8,19);write (ttky^.n^[loop1]:5:1);
    gotoxy (loop1*16- 8,20);
  END;

gotoxy (72,17);write (flwu^.n^[1]:5:1);
gotoxy (72,19);write (tbeg^.n^[1]:5:1);
gotoxy (72,22);write (flwy^.n^[1]:5:1);
gotoxy (72,23);write (flwr^.n^[1]:5:1);
END;

BEGIN
  prog_screen_initl;
  ctr2_screen_initl;
  REPEAT
    prog_varble_reads;

    {#####}
    {      IMPLEMENTING CONTROL LOOP      }
    {#####}
    gettime (hrs,min,sec,hun);
    time := hrs*3600 + min*60 + sec*1 + hun*0.01;
    dt := time - told;
    told := hrs*3600 + min*60 + sec*1 + hun*0.01;

    stsp_ip2op (flwy,xflf,mflf,flwf);
    vect_subtr (flwr,flwy,flwe);
    stsp_ip2op (flwe,xflk,mflk,flwu);
    vect_limit (flwu,100.0,0.0);

    stsp_ip2op (lvly,xlvf,mlvf,lvlf);
    vect_subtr (lvlr,lvlf,lvle);
    stsp_ip2op (lvle,xlvk,mlvk,lvlu);
    vect_limit (lvlu,100.0,0.0);
    {#####}

  prog_varble_write;

  IF keypressed THEN
    BEGIN
      check_keys;
```

```
ctr2_screen_init1;
END;
check_logg;

ctr2_screen_updat;
UNTIL ends;
END;

{*****}
{**** OPTION 3 : OPEN LOOP PIPE TEMP CONTROL ****}
{*****}
PROCEDURE control_3;

PROCEDURE ctr3_screen_init1;
BEGIN
textbackground (lightgray);textcolor (black);
gotoxy (20,02);write (' HEAT EXCHANGER RIG : [3] OPEN LOOP PIPE TEMP CONTROL');
gotoxy (26,04);write ('AltL: LOGS');
gotoxy (38,04);write ('AltP: PLOT');
gotoxy (50,04);write ('AltS: SAVE');
gotoxy (62,04);write ('AltX: EXIT');
textbackground (black);textcolor (green);

highvideo;
gotoxy (03,02);write ('CHR : SLURRY');
normvideo;
gotoxy (04,03);write ('F2r=   %');
gotoxy (04,04);write ('F2y=   %');
gotoxy (04,07);write ('T6 =   C');
gotoxy (04,09);write ('C6 =   %');

FOR loop1 := 1 to 4 DO
BEGIN
highvideo;
gotoxy (loop1*16- 3,10);write ('TANK no',loop1:1);
normvideo;
IF (loop1 <> 4) THEN
BEGIN
gotoxy (loop1*16+ 4,06);
gotoxy (loop1*16+ 4,07);write ('T',(loop1+6):1,'y=   C');
END
ELSE
BEGIN
gotoxy (loop1*16+ 4,06);
gotoxy (loop1*16+ 4,07);write ('T0',      'y=   C');
END;
gotoxy (loop1*16- 4,11);
gotoxy (loop1*16- 4,12);write ('S',loop1:1  ,' =   %');
gotoxy (loop1*16- 4,13);
gotoxy (loop1*16- 4,14);write ('L',loop1:1  ,'r=   %');
gotoxy (loop1*16- 4,15);write ('L',loop1:1  ,'y=   %');
gotoxy (loop1*16-12,17);write ('C',(6-loop1):1,' =   %');
gotoxy (loop1*16-12,19);write ('T',(6-loop1):1,'y=   C');
gotoxy (loop1*16-12,20);
```

```
END;

gotoxy (68,17);write ('C1 =    %');
gotoxy (68,19);write ('T1 =    C');
gotoxy (68,22);write ('Fly=    %');
gotoxy (68,23);write ('Flr=    %');
highvideo;
gotoxy (67,24);write ('CCR : CARBON');
normvideo;

IF (chce = 1) THEN
  BEGIN
    textbackground (lightgray);textcolor (black);
    gotoxy (04,22);write ('1:O/L TEMP');
    gotoxy (04,23);write (' LEVL no',logs:1,' ');
    textbackground (black);textcolor (green);
  END
ELSE
  BEGIN
    gotoxy (04,22);write ('1:O/L TEMP');
    gotoxy (04,23);write (' ');
  END;
END;

PROCEDURE ctr3_screen_updat;
BEGIN
  gotoxy (08,03);write (flwr^.n^[2]:5:1);
  gotoxy (08,04);write (flwy^.n^[2]:5:1);
  gotoxy (08,07);write (tbeq^.n^[2]:5:1);
  gotoxy (08,09);write (flwu^.n^[2]:5:1);

  FOR loop1 := 1 to 4 DO
    BEGIN
      gotoxy (loop1*16+ 8,06);
      gotoxy (loop1*16+ 8,07);write (tppy^.n^[loop1]:5:1);
      gotoxy (loop1*16 ,11);
      gotoxy (loop1*16 ,12);write (stir^.n^[loop1]:5:1);
      gotoxy (loop1*16 ,13);
      gotoxy (loop1*16 ,14);write (tppu^.n^[loop1]:5:1);
      gotoxy (loop1*16 ,15);write (lvly^.n^[loop1]:5:1);
      gotoxy (loop1*16- 8,17);write (lvlu^.n^[loop1]:5:1);
      gotoxy (loop1*16- 8,19);write (ttky^.n^[loop1]:5:1);
      gotoxy (loop1*16- 8,20);
    END;

    gotoxy (72,17);write (flwu^.n^[1]:5:1);
    gotoxy (72,19);write (tbeq^.n^[1]:5:1);
    gotoxy (72,22);write (flwy^.n^[1]:5:1);
    gotoxy (72,23);write (flwr^.n^[1]:5:1);
  END;

  BEGIN
    proq_screen_initl;
    ctr3_screen_initl;
```

```

REPEAT
  prog_varble_reads;

  {#####}
  {      IMPLEMENTING CONTROL LOOP      }
  {#####}
  gettime (hrs,min,sec,hun);
  time := hrs*3600 + min*60 + sec*1 + hun*0.01;
  dt   := time - told;
  told := hrs*3600 + min*60 + sec*1 + hun*0.01;

  stsp_ip2op (flwy,xflf,mflf,flwf);
  vect_subtr (flwr,flwy,flwe);
  stsp_ip2op (flwe,xflk,mflk,flwu);
  vect_limit (flwu,100.0,0.0);

  stsp_ip2op (lvly,xlvf,mlvf,lvlf);
  vect_subtr (tppu,lvly,lvle);
  stsp_ip2op (lvle,xlvk,mlvk,lvlu);
  vect_limit (lvlu,100.0,0.0);
  {#####}

  prog_varble_write;

  IF keypressed THEN
  BEGIN
    check_keys;
    ctr3_screen_init1;
  END;
  check_logg;

  ctr3_screen_updat;
  UNTIL ends;
END;

{#####}
{*****  OPTION 4 : CLOSED LOOP PIPE TEMP CONTROL  *****}
{#####}
PROCEDURE control_4;

PROCEDURE ctr4_screen_init1;
BEGIN
  textbackground (lightgray);textcolor (black);
  gotoxy (20,02);write (' HEAT EXCHANGER RIG : [4] CLOSED LOOP PIPE TEMP CONTROL');
  gotoxy (26,04);write ('AltL: LOGS');
  gotoxy (38,04);write ('AltP: PLOT');
  gotoxy (50,04);write ('AltS: SAVE');
  gotoxy (62,04);write ('AltX: EXIT');
  textbackground (black);textcolor (green);

  highvideo;
  gotoxy (03,02);write ('CHR : SLURRY');
  normvideo;
  gotoxy (04,03);write ('F2r=   %');

```

```

gotoxy (04,04);write ('F2y=    %');
gotoxy (04,07);write ('T6 =    C');
gotoxy (04,09);write ('C6 =    %');

FOR loop1 := 1 to 4 DO
  BEGIN
    highvideo;
    gotoxy (loop1*16- 3,10);write ('TANK no',loop1:1);
    normvideo;
    IF (loop1 <> 4) THEN
      BEGIN
        gotoxy (loop1*16+ 4,06);write ('T',(loop1+6):1,'r=    C');
        gotoxy (loop1*16+ 4,07);write ('T',(loop1+6):1,'y=    C');
      END
    ELSE
      BEGIN
        gotoxy (loop1*16+ 4,06);write ('T',0:1      ,'r=    C');
        gotoxy (loop1*16+ 4,07);write ('T',0:1      ,'y=    C');
      END;
    gotoxy (loop1*16- 4,11);
    gotoxy (loop1*16- 4,12);write ('S',loop1:1    ,' =    %');
    gotoxy (loop1*16- 4,13);
    gotoxy (loop1*16- 4,14);write ('L',loop1:1    ,'r=    %');
    gotoxy (loop1*16- 4,15);write ('L',loop1:1    ,'y=    %');
    gotoxy (loop1*16-12,17);write ('C',(6-loop1):1,' =    %');
    gotoxy (loop1*16-12,19);write ('T',(6-loop1):1,'y=    C');
    gotoxy (loop1*16-12,20);
  END;

  gotoxy (68,17);write ('C1 =    %');
  gotoxy (68,19);write ('T1 =    C');
  gotoxy (68,22);write ('Fly=    %');
  gotoxy (68,23);write ('Flr=    %');
  highvideo;
  gotoxy (67,24);write ('CCR : CARBON');
  normvideo;

  IF (chce = 1) THEN
    BEGIN
      textbackground (lightgray);textcolor (black);
      gotoxy (04,22);write ('1:C/L TEMP');
      IF (logs <> 4) THEN
        BEGIN
          gotoxy (04,23);write (' TEMP no',(logs+6):1,' ');
        END
      ELSE
        BEGIN
          gotoxy (04,23);write (' TEMP no',(    0):1,' ');
        END;
      textbackground (black);textcolor (green);
    END
  ELSE
    BEGIN
      gotoxy (04,22);write ('1:C/L TEMP');
    END
  END

```

```

    gotoxy (04,23);write ('      ');
END;
END;

PROCEDURE ctr4_screen_updat;
BEGIN
    gotoxy (08,03);write (flwr^.n^[2]:5:1);
    gotoxy (08,04);write (flwy^.n^[2]:5:1);
    gotoxy (08,07);write (tbeg^.n^[2]:5:1);
    gotoxy (08,09);write (flwu^.n^[2]:5:1);

    FOR loop1 := 1 to 4 DO
        BEGIN
            gotoxy (loop1*16+ 8,06);write (tppr^.n^[loop1]:5:1);
            gotoxy (loop1*16+ 8,07);write (tppy^.n^[loop1]:5:1);
            gotoxy (loop1*16      ,11);
            gotoxy (loop1*16      ,12);write (stir^.n^[loop1]:5:1);
            gotoxy (loop1*16      ,13);
            gotoxy (loop1*16      ,14);write (tppu^.n^[loop1]:5:1);
            gotoxy (loop1*16      ,15);write (lvly^.n^[loop1]:5:1);
            gotoxy (loop1*16- 8,17);write (lvlu^.n^[loop1]:5:1);
            gotoxy (loop1*16- 8,19);write (ttky^.n^[loop1]:5:1);
            gotoxy (loop1*16- 8,20);
        END;

        gotoxy (72,17);write (flwu^.n^[1]:5:1);
        gotoxy (72,19);write (tbeg^.n^[1]:5:1);
        gotoxy (72,22);write (flwy^.n^[1]:5:1);
        gotoxy (72,23);write (flwr^.n^[1]:5:1);
    END;

    BEGIN
        prog_screen_initl;
        ctr4_screen_initl;
        REPEAT
            prog_varble_reads;

            {#####}
            {      IMPLEMENTING CONTROL LOOP      }
            {#####}
            gettime (hrs,min,sec,hun);
            time := hrs*3600 + min*60 + sec*1 + hun*0.01;
            dt := time - told;
            told := hrs*3600 + min*60 + sec*1 + hun*0.01;

            stsp_ip2op (flwy,xflf,mflf,flwf);
            vect_subtr (flwr,flwy,flwe);
            stsp_ip2op (flwe,xflk,mflk,flwu);
            vect_limit (flwu,100.0,0.0);

            stsp_ip2op (tppy,xtpf,mtpf,tppf);
            vect_subtr (tppr,tppe,tppe);
            stsp_ip2op (tppe,xtpk,mtpk,tppu);
            vect_limit (tppu,100.0,0.0);

```

```

stsp_ip2op (lvly,xlvf,mlvf,lvlf);
vect_subtr (tppu,lvly,lvle);
stsp_ip2op (lvle,xlvk,mlvk,lvlu);
vect_limit (lvlu,100.0,0.0);
{#####}

prog_varble_write;

IF keypressed THEN
  BEGIN
    check_keys;
    ctr4_screen_initl;
  END;
check_logg;

ctr4_screen_updat;
UNTIL ends;
END;

{#####}
{****    MAIN    PROGRAMME    ****}
{#####}
BEGIN
  prog_varble_initl;
  REPEAT
    clrscr;
    gotoxy(15,05);write (' ');
    gotoxy(15,06);write (' COUNTER CURRENT HEAT EXCHANGER RIG CONTROL ');
    gotoxy(15,07);write (' ');

    gotoxy(15,10);write (' ');
    gotoxy(15,11);write (' 1. OPEN LOOP FLOW & LEVEL CONTROL ');
    gotoxy(15,12);write (' 2. CLOSED LOOP FLOW & LEVEL CONTROL ');
    gotoxy(15,13);write (' 3. OPEN LOOP PIPE TEMPERATURE CONTROL ');
    gotoxy(15,14);write (' 4. CLOSED LOOP PIPE TEMPERATURE CONTROL ');
    gotoxy(15,15);write (' 0. EXIT ');
    gotoxy(15,16);write (' ');
  REPEAT
    optn := ord(readkey) - ord('0');
  UNTIL ((optn >= 0) AND (optn <= 4));

  ends := false;
  CASE optn OF
    1 : control_1;
    2 : control_2;
    3 : control_3;
    4 : control_4;
    0 : exit := true;
  END;
  UNTIL exit;
  clrscr;
END.

```

D.2 : PLANT DESCRIPTION UNIT

UNIT plant;

{*****}

INTERFACE

{*****}

CONST

```
max_cntrl = 4;      { number of inputs/outputs in system }
max_sttsp = 12;     { number of states in state space controllers }
max_point = 100;    { number of points to be logged }

dx = 10.0;          { time interval between logging }
```

TYPE

```
v_var = array [1..max_cntrl] of double;
v_ptr = ^v_var;
vect_var = record
  o : v_ptr;
  n : v_ptr;
END;
vect_ptr = ^vect_var;

ssmx_var = record
  x : double;
  u : double;
  a : array [1..max_sttsp,1..max_sttsp] of double;
  b : array [1..max_sttsp,1..max_cntrl] of double;
  c : array [1..max_cntrl,1..max_sttsp] of double;
  d : array [1..max_cntrl,1..max_cntrl] of double;
END;
ssmx_ptr = ^ssmx_var;

s_var = array [1..max_sttsp] of double;
s_ptr = ^s_var;
ssst_var = record
  o : s_ptr;
  n : s_ptr;
END;
ssst_ptr = ^ssst_var;
```

VAR

dt : double;

{*****}

IMPLEMENTATION

{*****}

```
{*****}
{***  INITIALIZATION  SECTION  ***}
{*****}
```

BEGIN

END.

D.3 : MATRIX HANDLING UNIT

UNIT matrix;

{*****}

INTERFACE

{*****}

USES

plant;

PROCEDURE vect_initl (v:vect_ptr);

PROCEDURE vect_subtr (v1,v2,v3:vect_ptr);

PROCEDURE vect_limit (u:vect_ptr;uppr,bott:real);

PROCEDURE ssmx_initl (g:ssmx_ptr);

PROCEDURE ssst_initl (s:ssst_ptr);

PROCEDURE ssmx_loads (filename:string;g:ssmx_ptr);

PROCEDURE stsp_ip2op (u:vect_ptr;s:ssst_ptr;g:ssmx_ptr;y:vect_ptr);

{*****}

IMPLEMENTATION

{*****}

PROCEDURE vect_initl (v:vect_ptr);

VAR

r : integer;

BEGIN

new (v^.o);

new (v^.n);

FOR r := 1 to max_cntrl do

BEGIN

v^.o^[r] := 0;

v^.n^[r] := 0;

END;

END;

PROCEDURE vect_subtr (v1,v2,v3:vect_ptr);

VAR

r : integer;

BEGIN

FOR r := 1 to max_cntrl do

BEGIN

v3^.n^[r] := v1^.n^[r] - v2^.n^[r];

END;

END;

PROCEDURE vect_limit (u:vect_ptr;uppr,bott:real);

VAR

r : integer;

BEGIN

FOR r := 1 to max_cntrl do

BEGIN

u^.n^[r] := u^.n^[r] + (uppr-bott)/2;

IF (u^.n^[r] > uppr) THEN

```
    u^.n^[r] := uppr
  ELSE IF (u^.n^[r] < bott) THEN
    u^.n^[r] := bott;
  END;
END;
```

```
PROCEDURE ssmx_initl (g:ssmx_ptr);
```

```
VAR
```

```
  i,j : integer;
```

```
BEGIN
```

```
  g^.x := max_sttsp;
```

```
  g^.u := max_cntrl;
```

```
  FOR i := 1 to max_sttsp do
```

```
    BEGIN
```

```
      FOR j := 1 to max_sttsp do
```

```
        g^.a[i,j] := 0;
```

```
      FOR j := 1 to max_cntrl do
```

```
        g^.b[i,j] := 0;
```

```
    END;
```

```
  FOR i := 1 to max_cntrl do
```

```
    BEGIN
```

```
      FOR j := 1 to max_sttsp do
```

```
        g^.c[i,j] := 0;
```

```
      FOR j := 1 to max_cntrl do
```

```
        g^.d[i,j] := 0;
```

```
    END;
```

```
END;
```

```
PROCEDURE ssst_initl (s:ssst_ptr);
```

```
VAR
```

```
  j : integer;
```

```
BEGIN
```

```
  new (s^.o);
```

```
  new (s^.n);
```

```
  FOR j := 1 to max_sttsp do
```

```
    BEGIN
```

```
      s^.o^[j] := 0;
```

```
      s^.n^[j] := 0;
```

```
    END;
```

```
END;
```

```
PROCEDURE ssmx_loads (filename:string;q:ssmx_ptr);
```

```
VAR
```

```
  inpfiler : text;
```

```
  r,c      : integer;
```

```
BEGIN
```

```
  assign (inpfiler,filename);
```

```
  reset (inpfiler);
```

```
  read (inpfiler,g^.x);
```

```
  read (inpfiler,g^.u);
```

```
  readln (inpfiler);
```

```
  FOR r := 1 to trunc(g^.x) DO
```

```

BEGIN
  FOR c := 1 to trunc(g^.x) DO read (inpfile,g^.a[r,c]);
  FOR c := 1 to trunc(g^.u) DO read (inpfile,g^.b[r,c]);
  readln (inpfile);
END;
FOR r := 1 to trunc(g^.u) DO
  BEGIN
    FOR c := 1 to trunc(g^.x) DO read (inpfile,g^.c[r,c]);
    FOR c := 1 to trunc(g^.u) DO read (inpfile,g^.d[r,c]);
    readln (inpfile);
  END;
close (inpfile);
END;

PROCEDURE stsp_ip2op (u:vect_ptr;s:ssst_ptr;q:ssmx_ptr;y:vect_ptr);
VAR
  i,j : integer;
  m1 : array[1..max_sttsp] of double;
  m2 : array[1..max_sttsp] of double;
  m3 : array[1..max_sttsp] of double;
  m4 : array[1..max_sttsp] of double;
BEGIN
  { Runge-Kutta : First iteration }
  FOR i := 1 to trunc(g^.x) do
    BEGIN
      m1[i] := 0;
      FOR j := 1 to trunc(g^.x) do
        m1[i] := m1[i] + dt*g^.a[i,j]*(s^.o^[j]);
      FOR j := 1 to trunc(g^.u) do
        m1[i] := m1[i] + dt*g^.b[i,j]*u^.n^[j];
      END;

      { Runge-Kutta : Second iteration }
      FOR i := 1 to trunc(g^.x) do
        BEGIN
          m2[i] := 0;
          FOR j := 1 to trunc(g^.x) do
            m2[i] := m2[i] + dt*g^.a[i,j]*(s^.o^[j]+m1[j]/2);
          FOR j := 1 to trunc(g^.u) do
            m2[i] := m2[i] + dt*g^.b[i,j]*u^.n^[j];
          END;

          { Runge-Kutta : Third iteration }
          FOR i := 1 to trunc(g^.x) do
            BEGIN
              m3[i] := 0;
              FOR j := 1 to trunc(g^.x) do
                m3[i] := m3[i] + dt*g^.a[i,j]*(s^.o^[j]+m2[j]/2);
              FOR j := 1 to trunc(g^.u) do
                m3[i] := m3[i] + dt*g^.b[i,j]*u^.n^[j];
              END;

              { Runge-Kutta : Fourth iteration }
              FOR i := 1 to trunc(g^.x) do

```

```
BEGIN
  m4[i] := 0;
  FOR j := 1 to trunc(g^.x) do
    m4[i] := m4[i] + dt*g^.a[i,j]*(s^.o^[j]+m3[j]);
  FOR j := 1 to trunc(g^.u) do
    m4[i] := m4[i] + dt*g^.b[i,j]*u^.n^[j];
  END;

  FOR i := 1 to trunc(g^.x) do
    s^.n^[i] := s^.o^[i] + (1/6)*(m1[i]+2*m2[i]+2*m3[i]+m4[i]);

  FOR i := 1 to trunc(g^.u) do
    BEGIN
      y^.n^[i] := 0;
      FOR j := 1 to trunc(g^.x) do
        y^.n^[i] := y^.n^[i] + g^.c[i,j]*s^.n^[j];
      FOR j := 1 to trunc(g^.u) do
        y^.n^[i] := y^.n^[i] + g^.d[i,j]*u^.n^[j];
      END;

    FOR i := 1 to trunc(g^.x) do
      s^.o^[i] := s^.n^[i];

    FOR j := 1 to trunc(g^.u) do
      u^.o^[j] := u^.n^[j];

    FOR j := 1 to trunc(g^.u) do
      y^.o^[j] := y^.n^[j];
    END;

  {*****}
  {***  INITIALIZATION SECTION  ***}
  {*****}
  BEGIN
  END.
```

D.4 : DATA HANDLING UNIT

```
UNIT data;

{*****}
      INTERFACE
{*****}
USES
  plant;

TYPE
  data_var = record
    y : array [0..max_point] of v_ptr;
    u : array [0..max_point] of v_ptr;
    x : array [0..max_point] of double;
  END;
  data_ptr = ^data_var;

  labl_var = record
    head1 : string;
    head2 : string;
    yname : array [1..max_cntrl] of string;
    uname : array [1..max_cntrl] of string;
    ylabl : array [1..max_cntrl] of string;
    ulabl : array [1..max_cntrl] of string;
    yunit : array [1..max_cntrl] of string;
    uunit : array [1..max_cntrl] of string;
    xlabl : string;
  END;
  labl_ptr = ^labl_var;

  PROCEDURE data_init (d:data_ptr);
  PROCEDURE data_logg (z:integer;d:data_ptr;y:v_ptr;u:v_ptr;x:double);
  PROCEDURE data_plot (d:data_ptr;l:labl_ptr);
  PROCEDURE data_save (d:data_ptr;l:labl_ptr;f:string);

{*****}
      IMPLEMENTATION
{*****}
USES
  crt,graph;

PROCEDURE data_init (d:data_ptr);
VAR
  i : integer;
BEGIN
  FOR i := 0 to max_point do
    BEGIN
      new (d^.y[i]);
      new (d^.u[i]);
    END;
  END;
END;
```

```

PROCEDURE data_logg (z:integer;d:data_ptr;y:v_ptr;u:v_ptr;x:double);
VAR
  r,c : integer;
BEGIN
  FOR r := 1 to max_cntrl do
    BEGIN
      d^.y[z]^r := y^r;
      d^.u[z]^r := u^r;
    END;
  d^.x[z] := x;
END;

PROCEDURE data_plot (d:data_ptr;l:labl_ptr);
VAR
  y_maxi,y_mini : real;
  y_maxo,y_mino : real;
  x_pos,y_pos : integer;
  gd,gm : integer;
  i,row : integer;
  ch1,ch2 : char;
  exit_1 : boolean;
  s : string;
BEGIN
  gd := detect;
  initgraph (gd,gm,'');
  setviewport (0,0,getmaxx,getmaxy,clipon);clearviewport;

  setviewport (0,0,719,29,clipon);rectangle (0,0,719,29);
  setttextstyle (triplexfont,horizdir,2);
  outtextxy (30,5,l^.head1);
  setviewport (0,50,719,339,clipon);rectangle (0,0,719,289);

  row := 1;
  exit_1 := false;
  REPEAT
    { Heading }
    setlinestyle (solidln,1,normwidth);
    setviewport (100,60,700,90,clipon);clearviewport;rectangle (0,0,600,30);
    setttextstyle (triplexfont,horizdir,1);
    outtextxy (30,5,l^.head2+' : [ '+l^.yname[row]+' vs '+l^.uname[row]+' ]');

    y_maxo := 100;
    y_mino := 0;
    y_maxi := 100;
    y_mini := 0;

    { Plot x-axis }
    setviewport (100,305,718,340,clipoff);
    setttextstyle (defaultfont,horizdir,1);
    line (0,10,600,10);
    for i := 0 to 10 do
      BEGIN
        line (i*60,5,i*60,15);
        str (i/10*dx*max_point:3:1,s);outtextxy (i*60-30,17,s+l^.xlabl);
      END;
    end;
  UNTIL exit_1 = true;
END;

```

```

END;

{ Plot y-axis : output }
setviewport (1,60,45,320,clipon);clearviewport;
settextstyle (defaultfont,horizdir,1);
outtextxy (5,10,l^.ylabl[row]+'-['+l^.yunit[row]+'']);
setlinestyle (userbitln,$FFFF,thickwidth);
line (10,25,35,20);
setlinestyle (solidln,1,normwidth);
line (35,40,35,240);
for i := 0 to 4 do
  BEGIN
    line (30,40+i*50,40,40+i*50);
    str (y_maxo-i*0.25*(y_maxo-y_mino):4:2,s);
    outtextxy ( 0,45+i*50,s);
  END;

{ Plot y-axis : input }
setviewport (46,60,90,320,clipon);clearviewport;
settextstyle (defaultfont,horizdir,1);
outtextxy (5,10,l^.ulabl[row]+'-['+l^.uunit[row]+'']);
setlinestyle (userbitln,$9999,thickwidth);
line (10,25,35,20);
setlinestyle (solidln,1,normwidth);
line (35,40,35,240);
for i := 0 to 4 do
  BEGIN
    line (30,40+i*50,40,40+i*50);
    str (y_maxi-i*0.25*(y_maxi-y_mino):4:2,s);
    outtextxy ( 0,45+i*50,s);
  END;

{ Plot graph }
setviewport (100,100,700,300,clipon);clearviewport;
rectangle (0,0,600,200);

{ Plot grids }
setlinestyle (dottedln,1,normwidth);
for i := 1 to 19 do line (0,10*i,600,10*i);
for i := 1 to 19 do line (i*30,0,i*30,200);

{ Plot output }
x_pos := 0;
y_pos := 200 - round(200*((d^.y[0]^row)-y_mino)/(y_maxo-y_mino));
moveto (x_pos,y_pos);
setlinestyle (userbitln,$FFFF,thickwidth);
for i := 1 to max_point do
  BEGIN
    x_pos := i * (600 div max_point);
    y_pos := 200 - round(200*((d^.y[i]^row)-y_mino)/(y_maxo-y_mino));
    lineto (x_pos,y_pos);
  END;
setlinestyle (solidln,1,normwidth);

```

```
{ Plot input }
x_pos := 0;
y_pos := 200 - round(200*((d^.u[0]^row)-y_mini)/(y_maxi-y_mini));
moveto (x_pos,y_pos);
setlinestyle (userbitln,$9999,thickwidth);
for i := 1 to max_point do
  BEGIN
    x_pos := i * (600 div max_point);
    y_pos := 200 - round(200*((d^.u[i]^row)-y_mini)/(y_maxi-y_mini));
    lineto (x_pos,y_pos);
  END;
setlinestyle (solidln,1,normwidth);

REPEAT
  chl := upcase (readkey);
  row := ord(chl) - ord('0');
UNTIL (row >= 0) AND (row <= max_cntrl);
CASE chl of
  '0' : exit_1 := true;
END;
UNTIL exit_1;

setviewport (0,0,getmaxx,getmaxy,clipon);clearviewport;
closegraph;
END;

PROCEDURE data_save (d:data_ptr;l:labl_ptr;f:string);
VAR
  i,j    : integer;
  outfile : text;
BEGIN
  assign (outfile,f);
  rewrite (outfile);
  writeln (outfile,l^.head1);
  writeln (outfile,l^.head2);
  FOR i := 1 to max_cntrl DO
    BEGIN
      writeln (outfile,l^.yname[i]);
      writeln (outfile,l^.uname[i]);
      writeln (outfile,l^.ylabl[i]);
      writeln (outfile,l^.ulabl[i]);
      writeln (outfile,l^.yunit[i]);
      writeln (outfile,l^.uunit[i]);
    END;
  writeln (outfile,l^.xlabl);
  FOR i := 0 to max_point DO
    BEGIN
      FOR j := 1 to max_cntrl DO write (outfile,d^.y[i]^j);
      FOR j := 1 to max_cntrl DO write (outfile,d^.u[i]^j);
      write (outfile,d^.x[i]);
      writeln (outfile);
    END;
  close (outfile);
END;
```

```
{*****}  
{***  INITIALIZATION SECTION  ***}  
{*****}  
BEGIN  
END.
```

D.5 : INTERFACE UNIT

UNIT intrface;

{*****}

INTERFACE

{*****}

CONST

num_dac = 2;

dac_datareg : array [1..num_dac] of integer
= (\$0224,\$0226);

dac_statreg : array [1..num_dac] of integer
= (\$0225,\$0227);

dac_initial : array [1..num_dac] of integer
= (\$000B,\$000D);

num_adc = 2;

adc_datareg : array [1..num_adc] of integer
= (\$02EC,\$02EE);

adc_statreg : array [1..num_adc] of integer
= (\$02ED,\$02EF);

adc_initial : array [1..num_adc] of integer
= (\$0001,\$0001);

PROCEDURE dac_initl (i:integer);

PROCEDURE dac_write (i,j:integer;var z:real);

PROCEDURE adc_initl (i:integer);

PROCEDURE adc_reads (i,j:integer;var z:real);

{*****}

IMPLEMENTATION

{*****}

PROCEDURE dac_initl (i:integer);

VAR

x : integer;

BEGIN

REPEAT

port [dac_statreg[i]] := 0;

x := port [dac_statreg[i]];

UNTIL ((x and \$04) <> 0);

port [dac_datareg[i]] := dac_initial[i];

END;

PROCEDURE dac_write (i,j:integer;var z:real);

VAR

x,hgh,low : integer;

BEGIN

IF (z > 4095.0) THEN

x := 4095

ELSE IF (z < 0.0) THEN

x := 0

ELSE

```
x := round (z);

low := 16*(x - 16*(x div 16)) + 2*j;
hgh := x div 16;

REPEAT UNTIL ((port [dac_statreg[i]] AND $02) = 0);
port [dac_datareg[i]] := low;

REPEAT UNTIL ((port [dac_statreg[i]] AND $02) = 0);
port [dac_datareg[i]] := hgh;
END;

PROCEDURE adc_initl (i:integer);
VAR
  x : integer;
BEGIN
  port [adc_statreg[i]] := $000F;
  x := port [adc_datareg[i]];
  REPEAT
    x := port [adc_statreg[i]];
  UNTIL ((x and $04) <> 0);
  port [adc_datareg[i]] := adc_initial[i];
END;

PROCEDURE adc_reads (i,j:integer;var z:real);
VAR
  x,hgh,low : integer;
BEGIN
  REPEAT UNTIL ((port [adc_statreg[i]] AND $04) <> 0);
  port [adc_statreg[i]] := $0C;

  REPEAT UNTIL (((port [adc_statreg[i]] XOR $02) AND $02) <> 0);
  port [adc_datareg[i]] := $00;

  REPEAT UNTIL (((port [adc_statreg[i]] XOR $02) AND $02) <> 0);
  port [adc_datareg[i]] := j;

  REPEAT UNTIL ((port [adc_statreg[i]] AND $05) <> 0);
  low := port [adc_datareg[i]];

  REPEAT UNTIL ((port [adc_statreg[i]] AND $05) <> 0);
  hgh := port [adc_datareg[i]];

  z := hgh*256.0 + low;
END;

{*****}
{***   INITIALIZATION SECTION   ***}
{*****}
BEGIN
END.
```

APPENDIX E : SYSTEM IDENTIFICATION SOFTWARE

PROGRAM ident;

{ \$M 65520,0,655360 }

USES

dos,crt,graph;

CONST

max_point = 100;

max_cntrl = 4;

valid0 : set of '0'..'Z' = ['1','2','X'];

TYPE

v_var = array [1..max_cntrl] of double;

v_ptr = ^v_var;

data_var = record

u : array [0..max_point] of v_ptr;

y : array [0..max_point] of v_ptr;

x : array [0..max_point] of double;

END;

data_ptr = ^data_var;

labl_var = record

head1 : string;

head2 : string;

yname : array [0..max_cntrl] of string;

uname : array [0..max_cntrl] of string;

ylabl : array [0..max_cntrl] of string;

ulabl : array [0..max_cntrl] of string;

yunit : array [0..max_cntrl] of string;

uunit : array [0..max_cntrl] of string;

xlabl : string;

END;

labl_ptr = ^labl_var;

VAR

datclc : data_ptr;

datlog : data_ptr;

lables : labl_ptr;

search : pathstr;

inputname : string;

inputfile : text;

x0,x1 : real;

```

p1,p2,p3,p4 : real;
z1,z2,z3,z4 : real;

loop1 : integer;
loop2 : integer;
row,col : integer;
exit0 : boolean;
ch0 : char;

PROCEDURE data_plot (d:data_ptr;l:labl_ptr);
VAR
  y_maxi,y_mini : real;
  y_maxo,y_mino : real;
  i_fact,o_fact : real;
  i_diff,o_diff : real;
  x_pos,y_pos : integer;
  i,gm,gd : integer;
  s : string;
BEGIN
  gd := detect;
  initgraph (gd,gm,'');
  setviewport (0,0,getmaxx,getmaxy,clipon);clearviewport;

  setviewport (0,0,719,29,clipon);rectangle (0,0,719,29);
  settextstyle (triplexfont,horizdir,2);
  outtextxy (30,5,l^.head1);
  setviewport (0,50,719,339,clipon);rectangle (0,0,719,289);

  { Heading }
  setlinestyle (solidln,1,normwidth);
  setviewport (100,60,700,90,clipon);clearviewport;rectangle (0,0,600,30);
  settextstyle (triplexfont,horizdir,1);
  outtextxy (30,5,l^.head2+' : [ '+l^.yname[row]+' vs '+l^.uname[row]+' ]');

  { Determine y_maxi,y_mini,y_maxo,y_mino }
  y_maxi := -1E12;
  y_mini := +1E12;
  y_maxo := -1E12;
  y_mino := +1E12;
  for i := 0 to max_point do
    BEGIN
      if d^.u[i]^row > y_maxi then y_maxi := d^.u[i]^row;
      if d^.u[i]^row < y_mini then y_mini := d^.u[i]^row;
      if d^.y[i]^row > y_maxo then y_maxo := d^.y[i]^row;
      if d^.y[i]^row < y_mino then y_mino := d^.y[i]^row;
    END;

  i_diff := y_maxi - y_mini;
  i_fact := 1E-12;
  REPEAT i_fact := i_fact*10;
  UNTIL (i_diff*i_fact > 1);
  y_maxi := (trunc(y_maxi*i_fact)+1.0)/i_fact;
  y_mini := (trunc(y_mini*i_fact)+0.0)/i_fact;

```

```

o_diff := y_maxo - y_mino;
o_fact := 1E-12;
REPEAT o_fact := o_fact*10;
UNTIL (o_diff*o_fact > 1);
y_maxo := (trunc(y_maxo*o_fact)+1.0)/o_fact;
y_mino := (trunc(y_mino*o_fact)+0.0)/o_fact;

IF (l^.uname[row] = 'Estimate') THEN
  BEGIN
    y_maxi := y_maxo;
    y_mini := y_mino;
  END
ELSE
  BEGIN
    y_maxi := y_maxi;
    y_mini := y_mini;
  END;

{ Plot x-axis }
setviewport (100,305,718,340,clipoff);
settextstyle (defaultfont,horizdir,1);
line (0,10,600,10);
for i := 0 to 10 do
  BEGIN
    line (i*60,5,i*60,15);
    str (d^.x[trunc(i*max_point/10.0)]:3:1,s);outtextxy (i*60-30,17,s+l^.xlabl);
  END;

{ Plot y-axis : output }
setviewport (1,60,45,320,clipon);clearviewport;
settextstyle (defaultfont,horizdir,1);
outtextxy (5,10,l^.ylabl[row]+'-['+l^.yunit[row]+'']');
setlinestyle (userbitln,$FFFF,thickwidth);
line (10,25,35,20);
setlinestyle (solidln,1,normwidth);
line (35,40,35,240);
for i := 0 to 4 do
  BEGIN
    line (30,40+i*50,40,40+i*50);
    str (y_maxo-i*0.25*(y_maxo-y_mino):4:2,s);
    outtextxy (0,45+i*50,s);
  END;

{ Plot y-axis : input }
setviewport (46,60,90,320,clipon);clearviewport;
settextstyle (defaultfont,horizdir,1);
outtextxy (5,10,l^.ulabl[row]+'-['+l^.uunit[row]+'']');
setlinestyle (userbitln,$9999,thickwidth);
line (10,25,35,20);
setlinestyle (solidln,1,normwidth);
line (35,40,35,240);
for i := 0 to 4 do
  BEGIN
    line (30,40+i*50,40,40+i*50);
  END;

```

```

    str (y_maxi-i*0.25*(y_maxi-y_mini):4:2,s);
    outtextxy ( 0,45+i*50,s);
END;

{ Plot graph }
setviewport (100,100,700,300,clipon);clearviewport;
rectangle (0,0,600,200);

{ Plot grids }
setlinestyle (dottedln,1,normwidth);
for i := 1 to 19 do line (0,10*i,600,10*i);
for i := 1 to 19 do line (i*30,0,i*30,200);

{ Plot output }
x_pos := 0;
y_pos := 200 - round(200*((d^.y[0]^row)-y_min)/(y_maxo-y_min));
moveto (x_pos,y_pos);
setlinestyle (userbitln,$FFFF,thickwidth);
for i := 1 to max_point do
    BEGIN
        x_pos := i * (600 div max_point);
        y_pos := 200 - round(200*((d^.y[i]^row)-y_min)/(y_maxo-y_min));
        lineto (x_pos,y_pos);
    END;

{ Plot input }
x_pos := 0;
y_pos := 200 - round(200*((d^.u[0]^row)-y_min)/(y_maxi-y_min));
moveto (x_pos,y_pos);
setlinestyle (userbitln,$9999,thickwidth);
for i := 1 to max_point do
    BEGIN
        x_pos := i * (600 div max_point);
        y_pos := 200 - round(200*((d^.u[i]^row)-y_min)/(y_maxi-y_min));
        lineto (x_pos,y_pos);
    END;

REPEAT
UNTIL keypressed;

setviewport (0,0,getmaxx,getmaxy,clipon);clearviewport;
closegraph;
END;

{*****}
{****  PROCEDURE TO FIT DESIRED FUNCTION  ****}
{*****}
PROCEDURE func_fit (ordr,tipe:char);
VAR
    loop : integer;
    tstp : integer;
    tdel : integer;
    step : real;
    mean : real;

```

```

delt : real;

FUNCTION estimate (t:real) : real;
BEGIN
CASE ordr OF
'1' : CASE tipe OF
'1' : estimate := step * x1*(1-exp(-p1*(t)));
'2' : estimate := step * x1*(1-(1-p1/z1)*exp(-p1*(t)));
'3' : estimate := step * x1*(1-(1+p1/z1)*exp(-p1*(t)));
'4' : estimate := step * x1*(t);
END;
'2' : CASE tipe OF
'1' : estimate := step * x1*(1-(p2*exp(-p1*(t))-p1*exp(-p2*(t)))/(p2-p1));
'2' : estimate := step * x1*(1-sqrt(p1*p1+p2*p2*pi*pi)/(p2*pi)*exp(-p1*(t))*sin(p2*pi*(t)-arctan(-
p2*pi/p1)));
'3' : estimate := step * x1*(1-(1+p1*(t))*exp(-p1*(t)));
'4' : estimate := step * x1*(1-p2/z1*(p1-z1)/(p1-p2)*exp(-p1*(t))-p1/z1*(p2-z1)/(p2-p1)*exp(-
p2*(t)));
END;
END;
END;

BEGIN
step := (datlog^.u[max_point]^[col] - datlog^.u[0]^[col]);
delt := datlog^.x[max_point]/max_point;
tdel := round(x0/delt);

tstp := 0;
REPEAT
inc (tstp);
UNTIL (abs(datlog^.u[max_point]^[col]-datlog^.u[tstp+1]^[col]) < abs(step/3));

mean := 0;
FOR loop := 0 to tstp+tdel DO
BEGIN
mean := mean + datlog^.y[loop]^[row];
END;
mean := mean / (tstp+tdel+1);

FOR loop := 0 to tstp+tdel DO
BEGIN
datclc^.u[loop]^[row] := mean;
datclc^.y[loop]^[row] := datlog^.y[loop]^[row];
datclc^.x[loop] := datlog^.x[loop];
END;
FOR loop := tstp+tdel+1 to max_point DO
BEGIN
datclc^.u[loop]^[row] := mean + estimate (datlog^.x[loop]-datlog^.x[tstp+tdel]);
datclc^.y[loop]^[row] := datlog^.y[loop]^[row];
datclc^.x[loop] := datlog^.x[loop];
END;
END;

{*****}

```

```

{*****      1 : FIRST ORDER RESPONSE      *****)
{*****}
PROCEDURE order_1;
CONST
    valid1 : set of '0'..'Z' = ['1','2','3','4','X'];
VAR
    exit1 : boolean;
    chl   : char;

{*****}
{*****      1.1 : FIRST ORDER      *****)
{*****      Real Poles :  $\alpha$       *****)
{*****      Comp Poles : ---      *****)
{*****      LHP Zeros : ---      *****)
{*****      RHP Zeros : ---      *****)
{*****}
PROCEDURE selct_1;
CONST
    valid11 : set of '0'..'Z' = ['1','2','3','P','D','X'];
VAR
    exit11 : boolean;
    chl1   : char;
BEGIN
    x0 := 0;
    x1 := 1;
    p1 := 1;

    exit11 := false;
    REPEAT
        func_fit (ch0,chl);

        clrscr;
        gotoxy (20,01);write ('');
        gotoxy (20,02);write ('');
        gotoxy (20,03);write ('');
        gotoxy (20,04);write ('');
        gotoxy (20,05);write ('');
        gotoxy (20,06);write ('');
        gotoxy (20,07);write ('');
        gotoxy (20,09);write ('');
        gotoxy (20,10);write ('');
        gotoxy (20,11);write ('');
        gotoxy (20,12);write ('');
        gotoxy (20,13);write ('');
        gotoxy (20,14);write ('');
        gotoxy (20,15);write ('');
        gotoxy (20,16);write ('');
        gotoxy (20,17);write ('');
        gotoxy (20,23);write ('');
        gotoxy (20,24);write ('');
        gotoxy (20,25);write ('');
        gotoxy (44,10);write (x1:10:6);
        gotoxy (44,11);write (p1:10:6);
        gotoxy (44,12);write (x0:10:6);
    
```

1 : FIRST ORDER RESPONSE

$$1 : K \left[\frac{\alpha}{s + \alpha} \right] e^{-s\tau}$$

1 : K =

2 : α =

3 : τ =

P : Plot

D : Display

X : Exit

OPTION :

```

REPEAT
  gotoxy (44,24);
  ch11 := upcase(readkey);
  UNTIL ch11 in valid11;
  CASE ch11 OF
    '1' : BEGIN
      gotoxy (44,10);write ('      ');
      gotoxy (44,10);readln (x1);
      END;
    '2' : BEGIN
      gotoxy (44,11);write ('      ');
      gotoxy (44,11);readln (p1);
      END;
    '3' : BEGIN
      gotoxy (44,12);write ('      ');
      gotoxy (44,12);readln (x0);
      END;
    'P' : data_plot (datclc,lables);
    'D' : BEGIN
      clrscr;

      writeln ('NUMERATOR');
      IF (x0 = 0) THEN
        BEGIN
          writeln ('s^0 = ',(x1*p1):20:12);
          END
        ELSE
          BEGIN
            writeln ('s^0 = ',(2*x1*p1/x0):20:12);
            writeln ('s^1 = ',(-x1*p1):20:12);
            END;
          writeln;

          writeln ('DENOMINATOR');
          IF (x0 = 0) THEN
            BEGIN
              writeln ('s^0 = ',(p1):20:12);
              writeln ('s^1 = ',(1.0):20:12);
              END
            ELSE
              BEGIN
                writeln ('s^0 = ',(2*p1/x0):20:12);
                writeln ('s^1 = ',(p1+2/x0):20:12);
                writeln ('s^2 = ',(1.0):20:12);
                END;
              writeln;

              writeln ('<<< PRESS ANY KEY TO CONTINUE >>>');
              REPEAT UNTIL keypressed;
              END;
    'X' : exit11 := true;
  END;
UNTIL exit11;
END;

```

```

{*****}
{***** 1.2 : FIRST ORDER *****}
{***** Real Poles :  $\alpha$  *****}
{***** Comp Poles : --- *****}
{***** LHP Zeros : a *****}
{***** RHP Zeros : --- *****}
{*****}
PROCEDURE selct_2;
CONST
    valid12 : set of '0'..'Z' = ['1','2','3','4','P','D','X'];
VAR
    exit12 : boolean;
    ch12 : char;
BEGIN
    x0 := 0;
    x1 := 1;
    p1 := 1;
    z1 := 1;

    exit12 := false;
    REPEAT
        func_fit (ch0,ch1);

        clrscr;
        gotoxy (20,01);write ('');
        gotoxy (20,02);write ('');
        gotoxy (20,03);write ('');
        gotoxy (20,04);write ('');
        gotoxy (20,05);write ('');
        gotoxy (20,06);write ('');
        gotoxy (20,07);write ('');
        gotoxy (20,09);write ('');
        gotoxy (20,10);write ('');
        gotoxy (20,11);write ('');
        gotoxy (20,12);write ('');
        gotoxy (20,13);write ('');
        gotoxy (20,14);write ('');
        gotoxy (20,15);write ('');
        gotoxy (20,16);write ('');
        gotoxy (20,17);write ('');
        gotoxy (20,18);write ('');
        gotoxy (20,23);write ('');
        gotoxy (20,24);write ('');
        gotoxy (20,25);write ('');
        gotoxy (44,10);write (x1:10:6);
        gotoxy (44,11);write (p1:10:6);
        gotoxy (44,12);write (z1:10:6);
        gotoxy (44,13);write (x0:10:6);
        REPEAT
            gotoxy (44,24);
            ch12 := upcase(readkey);
        UNTIL ch12 in valid12;
        CASE ch12 OF

```

```

1 : FIRST ORDER RESPONSE

2 : K  $\begin{bmatrix} \alpha & s + a \\ - & s + \alpha \end{bmatrix} e^{-s\tau}$ 

```

```

1 : K =
2 :  $\alpha$  =
3 : a =
4 :  $\tau$  =

P : Plot
D : Display
X : Exit

```

```

OPTION :

```

```


```

```

'1' : BEGIN
    gotoxy (44,10);write (' ');
    gotoxy (44,10);readln (x1);
END;
'2' : BEGIN
    gotoxy (44,11);write (' ');
    gotoxy (44,11);readln (p1);
END;
'3' : BEGIN
    gotoxy (44,12);write (' ');
    gotoxy (44,12);readln (z1);
END;
'4' : BEGIN
    gotoxy (44,13);write (' ');
    gotoxy (44,13);readln (x0);
END;
'P' : data_plot (datclc,labes);
'D' : BEGIN
    clrscr;

    writeln ('NUMERATOR');
    IF (x0 = 0) THEN
        BEGIN
            writeln ('s^0 = ',(x1*p1):20:12);
            writeln ('s^1 = ',(x1*p1/z1):20:12);
        END
    ELSE
        BEGIN
            writeln ('s^0 = ',(2*x1*p1/x0):20:12);
            writeln ('s^1 = ',(x1*p1*(2/(z1*x0)-1)):20:12);
            writeln ('s^2 = ',(-x1*p1/z1):20:12);
        END;
    writeln;

    writeln ('DENOMINATOR');
    IF (x0 = 0) THEN
        BEGIN
            writeln ('s^0 = ',(p1):20:12);
            writeln ('s^1 = ',(1.0):20:12);
        END
    ELSE
        BEGIN
            writeln ('s^0 = ',(2*p1/x0):20:12);
            writeln ('s^1 = ',(p1+2/x0):20:12);
            writeln ('s^2 = ',(1.0):20:12);
        END;
    writeln;

    writeln ('<<< PRESS ANY KEY TO CONTINUE >>>');
    REPEAT UNTIL keypressed;
END;
'X' : exit12 := true;
END;
UNTIL exit12;

```

END;

```

{*****}
{***** 1.3 : FIRST ORDER *****}
{***** Real Poles :  $\alpha$  *****}
{***** Comp Poles : --- *****}
{***** LHP Zeros : --- *****}
{***** RHP Zeros :  $a$  *****}
{*****}
PROCEDURE selct_3;
CONST
    valid13 : set of '0'..'Z' = ['1','2','3','4','P','D','X'];
VAR
    exit13 : boolean;
    ch13 : char;
BEGIN
    x0 := 0;
    x1 := 1;
    p1 := 1;
    z1 := 1;

    exit13 := false;
    REPEAT
        func_fit (ch0,ch1);

        clrscr;
        gotoxy (20,01);write ('');
        gotoxy (20,02);write ('');
        gotoxy (20,03);write ('');
        gotoxy (20,04);write ('');
        gotoxy (20,05);write ('');
        gotoxy (20,06);write ('');
        gotoxy (20,07);write ('');
        gotoxy (20,09);write ('');
        gotoxy (20,10);write ('');
        gotoxy (20,11);write ('');
        gotoxy (20,12);write ('');
        gotoxy (20,13);write ('');
        gotoxy (20,14);write ('');
        gotoxy (20,15);write ('');
        gotoxy (20,16);write ('');
        gotoxy (20,17);write ('');
        gotoxy (20,18);write ('');
        gotoxy (20,23);write ('');
        gotoxy (20,24);write ('');
        gotoxy (20,25);write ('');
        gotoxy (44,10);write (x1:10:6);
        gotoxy (44,11);write (p1:10:6);
        gotoxy (44,12);write (z1:10:6);
        gotoxy (44,13);write (x0:10:6);
        REPEAT
            gotoxy (44,24);
            ch13 := upcase(readkey);
        UNTIL ch13 in valid13;
    
```

1 : FIRST ORDER RESPONSE

3 : $K \left[\begin{array}{cc} \alpha & s - a \\ - & \frac{s - a}{s + \alpha} \end{array} \right] e^{-s\tau}$

1 : K =

2 : α =

3 : a =

4 : τ =

P : Plot

D : Display

X : Exit

OPTION :

```

CASE ch13 OF
'1' : BEGIN
    gotoxy (44,10);write (' ');
    gotoxy (44,10);readln (x1);
END;
'2' : BEGIN
    gotoxy (44,11);write (' ');
    gotoxy (44,11);readln (p1);
END;
'3' : BEGIN
    gotoxy (44,12);write (' ');
    gotoxy (44,12);readln (z1);
END;
'4' : BEGIN
    gotoxy (44,13);write (' ');
    gotoxy (44,13);readln (x0);
END;
'P' : data_plot (datclc,lablles);
'D' : BEGIN
    clrscr;

    writeln ('NUMERATOR');
    IF (x0 = 0) THEN
        BEGIN
            writeln ('s^0 = ',(x1*p1):20:12);
            writeln ('s^1 = ',(-x1*p1/z1):20:12);
        END
    ELSE
        BEGIN
            writeln ('s^0 = ',(2*x1*p1/x0):20:12);
            writeln ('s^1 = ',(-x1*p1*(2/(z1*x0)+1)):20:12);
            writeln ('s^2 = ',(x1*p1/z1):20:12);
        END;
    writeln;

    writeln ('DENOMINATOR');
    IF (x0 = 0) THEN
        BEGIN
            writeln ('s^0 = ',(p1):20:12);
            writeln ('s^1 = ',(1.0):20:12);
        END
    ELSE
        BEGIN
            writeln ('s^0 = ',(2*p1/x0):20:12);
            writeln ('s^1 = ',(p1+2/x0):20:12);
            writeln ('s^2 = ',(1.0):20:12);
        END;
    writeln;

    writeln ('<<< PRESS ANY KEY TO CONTINUE >>>');
    REPEAT UNTIL keypressed;
END;
'X' : exit13 := true;
END;

```

```

UNTIL exit13;
END;

{*****}
{****  1.4 : FIRST ORDER      ****}
{****      Real Poles : 0      ****}
{****      Comp Poles : ---    ****}
{****      LHP Zeros : ---     ****}
{****      RHP Zeros : ---     ****}
{*****}
PROCEDURE selct_4;
CONST
    valid14 : set of '0'..'Z' = ['1','2','P','D','X'];
VAR
    exit14 : boolean;
    chl4   : char;
BEGIN
    x0 := 0;
    x1 := 1;

    exit14 := false;
    REPEAT
        func_fit (ch0,chl);

        clrscr;
        gotoxy (20,01);write ('');
        gotoxy (20,02);write ('');
        gotoxy (20,03);write ('');
        gotoxy (20,04);write ('');
        gotoxy (20,05);write ('');
        gotoxy (20,06);write ('');
        gotoxy (20,07);write ('');
        gotoxy (20,09);write ('');
        gotoxy (20,10);write ('');
        gotoxy (20,11);write ('');
        gotoxy (20,12);write ('');
        gotoxy (20,13);write ('');
        gotoxy (20,14);write ('');
        gotoxy (20,15);write ('');
        gotoxy (20,16);write ('');
        gotoxy (20,23);write ('');
        gotoxy (20,24);write ('');
        gotoxy (20,25);write ('');
        gotoxy (44,10);write (x1:10:6);
        gotoxy (44,11);write (x0:10:6);
        REPEAT
            gotoxy (44,24);
            chl4 := upcase(readkey);
            UNTIL chl4 in valid14;
            CASE chl4 OF
                '1' : BEGIN
                    gotoxy (44,10);write ('');
                    gotoxy (44,10);readln (x1);
                    END;

```

```

1 : FIRST ORDER RESPONSE
TETe |');
4 : K [ 1 -s\tau ] e |');
      [ - ] e |');
      [ s ] e |');

```

```

1 : K =
2 : \tau =

P : Plot
D : Display
X : Exit

```

```

OPTION :

```

```

'2' : BEGIN
    gotoxy (44,11);write (' ');
    gotoxy (44,11);readln (x0);
END;
'P' : data_plot (datclc,lablcs);
'D' : BEGIN
    clrscr;

    writeln ('NUMERATOR');
    IF (x0 = 0) THEN
        BEGIN
            writeln ('s^0 = ',(x1):20:12);
        END
    ELSE
        BEGIN
            writeln ('s^0 = ',(2*x1/x0):20:12);
            writeln ('s^1 = ',(-x1):20:12);
        END;
    writeln;

    writeln ('DENOMINATOR');
    IF (x0 = 0) THEN
        BEGIN
            writeln ('s^0 = ',(0.0):20:12);
            writeln ('s^1 = ',(1.0):20:12);
        END
    ELSE
        BEGIN
            writeln ('s^0 = ',(0.0):20:12);
            writeln ('s^1 = ',(2/x0):20:12);
            writeln ('s^2 = ',(1.0):20:12);
        END;
    writeln;

    writeln ('<<< PRESS ANY KEY TO CONTINUE >>>');
    REPEAT UNTIL keypressed;
END;
'X' : exit14 := true;
END;
UNTIL exit14;
END;

BEGIN
    exit1 := false;
    REPEAT
        clrscr;
        gotoxy (15,01);write (' ');
        gotoxy (15,02);write (' ');
        gotoxy (15,03);write (' ');
        gotoxy (15,06);write (' ');
        gotoxy (15,07);write (' ');
        gotoxy (15,08);write (' ');
        gotoxy (15,09);write (' ');
        gotoxy (15,10);write (' ');

```

1 : FIRST ORDER RESPONSE

1 : K $\left[\frac{\alpha}{s + \alpha} \right] e^{-st}$

$\left[\frac{\alpha}{\alpha} \quad s + \alpha \right]^{-st}$

```

    );
    );
    );
    );
    );
    );
    );
    );
    );

```

```

gotoxy (15,11);write ('      2      :      K      [      -      —      ]      e      ');
gotoxy (15,12);write ('      a      s      +      α      ');
gotoxy (15,13);write ('      α      s      -      a      -s†      ');
gotoxy (15,14);write ('      3      :      K      [      -      -      —      ]      e      ');
gotoxy (15,15);write ('      a      s      +      α      ');
gotoxy (15,16);write ('      1      -s†      ');
gotoxy (15,17);write ('      4      :      K      [      —      ]      e      ');
gotoxy (15,18);write ('      s      ');
gotoxy (15,19);write ('      ');
gotoxy (15,20);write ('      X      :      Exit      ');
gotoxy (15,21);write ('      ');
gotoxy (15,23);write ('      ');
gotoxy (15,24);write ('      ');
gotoxy (15,25);write ('      ');
REPEAT
  gotoxy (37,24);
  chl := upcase(readkey);
UNTIL chl in valid1;
CASE chl OF
  '1' : selct_1;
  '2' : selct_2;
  '3' : selct_3;
  '4' : selct_4;
  'X' : exit1 := true;
END;
UNTIL exit1;
END;

{*****}
{*****      2      :      FIRST ORDER RESPONSES      *****}
{*****}
PROCEDURE order_2;
CONST
  valid2 : set of '0'..'Z' = ['1','2','3','4','X'];
VAR
  exit2 : boolean;
  ch2   : char;

{*****}
{*****      2.1      :      SECOND ORDER      *****}
{*****      Real Poles : α,β      *****}
{*****      Comp Poles : ---      *****}
{*****      LHP Zeros : ---      *****}
{*****      RHP Zeros : ---      *****}
{*****}
PROCEDURE selct_1;
CONST
  valid21 : set of '0'..'Z' = ['1','2','3','4','P','D','X'];
VAR
  exit21 : boolean;
  ch21   : char;
BEGIN
  x0 := 0;
  x1 := 1;

```

```

p1 := 0.5;
p2 := 1.0;

exit21 := false;
REPEAT
  func_fit (ch0,ch2);

  clrscr;
  gotoxy (20,01);write (' ');
  gotoxy (20,02);write (' ');
  gotoxy (20,03);write (' ');
  gotoxy (20,04);write (' ');
  gotoxy (20,05);write (' ');
  gotoxy (20,06);write (' ');
  gotoxy (20,07);write (' ');
  gotoxy (20,09);write (' ');
  gotoxy (20,10);write (' ');
  gotoxy (20,11);write (' ');
  gotoxy (20,12);write (' ');
  gotoxy (20,13);write (' ');
  gotoxy (20,14);write (' ');
  gotoxy (20,15);write (' ');
  gotoxy (20,16);write (' ');
  gotoxy (20,17);write (' ');
  gotoxy (20,18);write (' ');
  gotoxy (20,23);write (' ');
  gotoxy (20,24);write (' ');
  gotoxy (20,25);write (' ');
  gotoxy (44,10);write (x1:10:6);
  gotoxy (44,11);write (p1:10:6);
  gotoxy (44,12);write (p2:10:6);
  gotoxy (44,13);write (x0:10:6);
  REPEAT
    gotoxy (44,24);
    ch21 := upcase(readkey);
  UNTIL ch21 in valid21;
  CASE ch21 OF
    '1' : BEGIN
      gotoxy (44,10);write (' ');
      gotoxy (44,10);readln (x1);
      END;
    '2' : BEGIN
      gotoxy (44,11);write (' ');
      gotoxy (44,11);readln (p1);
      END;
    '3' : BEGIN
      gotoxy (44,12);write (' ');
      gotoxy (44,12);readln (p2);
      END;
    '4' : BEGIN
      gotoxy (44,13);write (' ');
      gotoxy (44,13);readln (x0);
      END;
    'P' : data_plot (datclc,lables);

```

2 : SECOND ORDER RESPONSE

$$1 : K \left[\frac{\alpha\beta}{(s + \alpha)(s + \beta)} \right] e^{-s\tau}$$

1 : K =

2 : α =

3 : β =

4 : τ =

P : Plot

D : Display

X : Exit

OPTION :

```

'D' : BEGIN
    clrscr;

    writeln ('NUMERATOR');
    IF (x0 = 0) THEN
        BEGIN
            writeln ('s^0 = ',(x1*p1*p2):20:12);
        END
    ELSE
        BEGIN
            writeln ('s^0 = ',(2*x1*p1*p2/x0):20:12);
            writeln ('s^1 = ',(-x1*p1*p2):20:12);
        END;
    writeln;

    writeln ('DENOMINATOR');
    IF (x0 = 0) THEN
        BEGIN
            writeln ('s^0 = ',(p1*p2):20:12);
            writeln ('s^1 = ',(p1+p2):20:12);
            writeln ('s^2 = ',(1.0):20:12);
        END
    ELSE
        BEGIN
            writeln ('s^0 = ',(2*p1*p2/x0):20:12);
            writeln ('s^1 = ',(p1*p2+2*(p1+p2)/x0):20:12);
            writeln ('s^2 = ',(p1+p2+2/x0):20:12);
            writeln ('s^3 = ',(1.0):20:12);
        END;
    writeln;

    writeln ('<<< PRESS ANY KEY TO CONTINUE >>>');
    REPEAT UNTIL keypressed;
END;

'X' : exit21 := true;
END;
UNTIL exit21;
END;

{*****}
{***** 2.2 : SECOND ORDER *****}
{***** Real Poles :  $\alpha \pm j\beta$  *****}
{***** Comp Poles : --- *****}
{***** LHP Zeros : --- *****}
{***** RHP Zeros : --- *****}
{*****}
PROCEDURE selct_2;
CONST
    valid22 : set of '0'..'Z' = ['1','2','3','4','P','D','X'];
VAR
    exit22 : boolean;
    ch22 : char;
BEGIN
    x0 := 0;

```

```

x1 := 1;
p1 := 1;
p2 := 1;

exit22 := false;
REPEAT
  func_fit (ch0,ch2);

  clrscr;
  gotoxy (20,01);write ('');
  gotoxy (20,02);write ('');
  gotoxy (20,03);write ('');
  gotoxy (20,04);write ('');
  gotoxy (20,05);write ('');
  gotoxy (20,06);write ('');
  gotoxy (20,07);write ('');
  gotoxy (20,09);write ('');
  gotoxy (20,10);write ('');
  gotoxy (20,11);write ('');
  gotoxy (20,12);write ('');
  gotoxy (20,13);write ('');
  gotoxy (20,14);write ('');
  gotoxy (20,15);write ('');
  gotoxy (20,16);write ('');
  gotoxy (20,17);write ('');
  gotoxy (20,18);write ('');
  gotoxy (20,23);write ('');
  gotoxy (20,24);write ('');
  gotoxy (20,25);write ('');
  gotoxy (44,10);write (x1:10:6);
  gotoxy (44,11);write (p1:10:6);
  gotoxy (44,12);write (p2:10:6,'π');
  gotoxy (44,13);write (x0:10:6);
  REPEAT
    gotoxy (44,24);
    ch22 := upcase(readkey);
  UNTIL ch22 in valid22;
  CASE ch22 OF
    '1' : BEGIN
      gotoxy (44,10);write ('');
      gotoxy (44,10);readln (x1);
      END;
    '2' : BEGIN
      gotoxy (44,11);write ('');
      gotoxy (44,11);readln (p1);
      END;
    '3' : BEGIN
      gotoxy (44,12);write ('');
      gotoxy (44,12);readln (p2);
      END;
    '4' : BEGIN
      gotoxy (44,13);write ('');
      gotoxy (44,13);readln (x0);
      END;
  
```

2 : SECOND ORDER RESPONSE

$$2 : K \left[\frac{\alpha^2 + \beta^2}{(s + \alpha)^2 + \beta^2} \right] e^{-s\tau}$$

1 : K =

2 : α =

3 : β =

4 : τ =

P : Plot

D : Display

X : Exit

OPTION :

```

'P' : data_plot (datclc,lables);
'D' : BEGIN
    clrscr;

    writeln ('NUMERATOR');
    IF (x0 = 0) THEN
        BEGIN
            writeln ('s^0 = ',(x1*(p1*p1+p2*p2*pi*pi)):20:12);
        END
    ELSE
        BEGIN
            writeln ('s^0 = ',(2*x1*(p1*p1+p2*p2*pi*pi)/x0):20:12);
            writeln ('s^1 = ',(-x1*(p1*p1+p2*p2*pi*pi)):20:12);
        END;
    writeln;

    writeln ('DENOMINATOR');
    IF (x0 = 0) THEN
        BEGIN
            writeln ('s^0 = ',(p1*p1+p2*p2*pi*pi):20:12);
            writeln ('s^1 = ',(2*p1):20:12);
            writeln ('s^2 = ',(1.0):20:12);
        END
    ELSE
        BEGIN
            writeln ('s^0 = ',(2*(p1*p1+p2*p2*pi*pi)/x0):20:12);
            writeln ('s^1 = ',(p1*p1+p2*p2*pi*pi+4*p1/x0):20:12);
            writeln ('s^2 = ',(2*p1+2/x0):20:12);
            writeln ('s^3 = ',(1.0):20:12);
        END;
    writeln;

    writeln ('<<< PRESS ANY KEY TO CONTINUE >>>');
    REPEAT UNTIL keypressed;
END;
'X' : exit22 := true;
END;
UNTIL exit22;
END;

{*****}
{**** 2.3 : SECOND ORDER *****}
{**** Real Poles :  $\alpha$  ,  $\alpha$  *****}
{**** Comp Poles : --- *****}
{**** LHP Zeros : --- *****}
{**** RHP Zeros : --- *****}
{*****}
PROCEDURE selct_3;
CONST
    valid23 : set of '0'..'Z' = ['1','2','3','P','D','X'];
VAR
    exit23 : boolean;
    ch23 : char;
BEGIN

```

```

x0 := 0;
x1 := 1;
p1 := 1;

exit23 := false;
REPEAT
  func_fit (ch0,ch2);

  clrscr;
  gotoxy (20,01);write ('');
  gotoxy (20,02);write ('');
  gotoxy (20,03);write ('');
  gotoxy (20,04);write ('');
  gotoxy (20,05);write ('');
  gotoxy (20,06);write ('');
  gotoxy (20,07);write ('');
  gotoxy (20,09);write ('');
  gotoxy (20,10);write ('');
  gotoxy (20,11);write ('');
  gotoxy (20,12);write ('');
  gotoxy (20,13);write ('');
  gotoxy (20,14);write ('');
  gotoxy (20,15);write ('');
  gotoxy (20,16);write ('');
  gotoxy (20,17);write ('');
  gotoxy (20,23);write ('');
  gotoxy (20,24);write ('');
  gotoxy (20,25);write ('');
  gotoxy (44,10);write (x1:10:6);
  gotoxy (44,11);write (p1:10:6);
  gotoxy (44,12);write (x0:10:6);
  REPEAT
    gotoxy (44,24);
    ch23 := upcase(readkey);
  UNTIL ch23 in valid23;
  CASE ch23 OF
    '1' : BEGIN
      gotoxy (44,10);write ('');
      gotoxy (44,10);readln (x1);
    END;
    '2' : BEGIN
      gotoxy (44,11);write ('');
      gotoxy (44,11);readln (p1);
    END;
    '3' : BEGIN
      gotoxy (44,12);write ('');
      gotoxy (44,12);readln (x0);
    END;
    'P' : data_plot (datclc,lables);
    'D' : BEGIN
      clrscr;

      writeln ('NUMERATOR');
      IF (x0 = 0) THEN

```

2 : SECOND ORDER RESPONSE

3 : $K \left[\frac{\alpha^2}{(s + \alpha)^2} \right] e^{-s\tau}$

1 : K =

2 : α =

3 : τ =

P : Plot

D : Display

X : Exit

OPTION :

```

BEGIN
  writeln ('s^0 = ',(x1*p1*p1):20:12);
END
ELSE
  BEGIN
    writeln ('s^0 = ',(2*x1*p1*p1/x0):20:12);
    writeln ('s^1 = ',(-x1*p1*p1):20:12);
  END;
  writeln;

  writeln ('DENOMINATOR');
  IF (x0 = 0) THEN
    BEGIN
      writeln ('s^0 = ',(p1*p1):20:12);
      writeln ('s^1 = ',(2*p1):20:12);
      writeln ('s^2 = ',(1.0):20:12);
    END
  ELSE
    BEGIN
      writeln ('s^0 = ',(2*p1*p1/x0):20:12);
      writeln ('s^1 = ',(p1*p1+4*p1/x0):20:12);
      writeln ('s^2 = ',(2*p1+2/x0):20:12);
      writeln ('s^3 = ',(1.0):20:12);
    END;
    writeln;

    writeln ('<<< PRESS ANY KEY TO CONTINUE >>>');
    REPEAT UNTIL keypressed;
  END;
  'X' : exit23 := true;
END;
UNTIL exit23;
END;

{*****}
{**** 2.4 : SECOND ORDER ****}
{**** Real Poles :  $\alpha, \beta$  ****}
{**** Comp Poles : --- ****}
{**** LHP Zeros : a ****}
{**** RHP Zeros : --- ****}
{*****}
PROCEDURE selct_4;
CONST
  valid24 : set of '0'..'Z' = ['1','2','3','4','5','P','D','X'];
VAR
  exit24 : boolean;
  ch24 : char;
BEGIN
  x0 := 0;
  x1 := 1;
  p1 := 0.5;
  p2 := 1.0;
  z1 := 2.0;

```

```

exit24 := false;
REPEAT
  func_fit (ch0,ch2);

  clrscr;
  gotoxy (20,01);write ('');
  gotoxy (20,02);write ('');
  gotoxy (20,03);write ('');
  gotoxy (20,04);write ('');
  gotoxy (20,05);write ('');
  gotoxy (20,06);write ('');
  gotoxy (20,07);write ('');
  gotoxy (20,09);write ('');
  gotoxy (20,10);write ('');
  gotoxy (20,11);write ('');
  gotoxy (20,12);write ('');
  gotoxy (20,13);write ('');
  gotoxy (20,14);write ('');
  gotoxy (20,15);write ('');
  gotoxy (20,16);write ('');
  gotoxy (20,17);write ('');
  gotoxy (20,18);write ('');
  gotoxy (20,19);write ('');
  gotoxy (20,23);write ('');
  gotoxy (20,24);write ('');
  gotoxy (20,25);write ('');
  gotoxy (44,10);write (x1:10:6);
  gotoxy (44,11);write (p1:10:6);
  gotoxy (44,12);write (p2:10:6);
  gotoxy (44,13);write (z1:10:6);
  gotoxy (44,14);write (x0:10:6);
  REPEAT
    gotoxy (44,24);
    ch24 := upcase(readkey);
  UNTIL ch24 in valid24;
  CASE ch24 OF
    '1' : BEGIN
      gotoxy (44,10);write ('');
      gotoxy (44,10);readln (x1);
      END;
    '2' : BEGIN
      gotoxy (44,11);write ('');
      gotoxy (44,11);readln (p1);
      END;
    '3' : BEGIN
      gotoxy (44,12);write ('');
      gotoxy (44,12);readln (p2);
      END;
    '4' : BEGIN
      gotoxy (44,13);write ('');
      gotoxy (44,13);readln (z1);
      END;
    '5' : BEGIN
      gotoxy (44,14);write ('');

```

2 : SECOND ORDER RESPONSE

4 : $K \left[\frac{\alpha\beta}{a} \frac{(s + a)}{(s + \alpha)(s + \beta)} \right] e^{-s\tau}$

1 : K =

2 : α =

3 : β =

4 : a =

5 : τ =

P : Plot

D : Display

X : Exit

OPTION :


```

gotoxy (15,09);write ('
gotoxy (15,10);write ('
gotoxy (15,11);write ('
gotoxy (15,12);write ('
gotoxy (15,13);write ('
gotoxy (15,14);write ('
gotoxy (15,15);write ('
gotoxy (15,16);write ('
gotoxy (15,17);write ('
gotoxy (15,18);write ('
gotoxy (15,19);write ('
gotoxy (15,20);write ('
gotoxy (15,21);write ('
gotoxy (15,23);write ('
gotoxy (15,24);write ('
gotoxy (15,25);write ('
REPEAT
  gotoxy (37,24);
  ch2 := upcase(readkey);
UNTIL ch2 in valid2;
CASE ch2 OF
  '1' : selct_1;
  '2' : selct_2;
  '3' : selct_3;
  '4' : selct_4;
  'X' : exit2 := true;
END;
UNTIL exit2;
END;

( (s + α)(s + B) )
      α²+B²
2 : K [-----] e-sr
      (s + α)² + B²

      α²
3 : K [-----] e-sr
      (s + α)²

      αβ (s + a)
4 : K [---] e-sr
      a (s + α)(s + B)

X : Exit

OPTION :

')';
')';
')';
')';
')';
')';
')';
')';
')';
')';
')';
')';
')';
')';
')';
')';

{*****}
{***** MAIN PROGRAMME *****}
{*****}
BEGIN
  new (datclc);
  new (datlog);
  FOR loop1 := 0 to max_point DO
    BEGIN
      new (datclc^.y[loop1]);
      new (datclc^.u[loop1]);
      new (datlog^.y[loop1]);
      new (datlog^.u[loop1]);
    END;
  new (lables);

  REPEAT
    clrscr;
    write ('NAME OF INPUT FILE : ');
    readln (inputname);
    write ('ROW (Output) NUMBER : ');
    readln (row);
    write ('COL (Step) NUMBER : ');
    readln (col);
  
```

```

search := fsearch (inputname,'c:');
UNTIL (search <> '');
assign (inputfile,fexpand(inputname));
reset (inputfile);
readln (inputfile,labes^.head1);
readln (inputfile,labes^.head2);
FOR loop1 := 1 to max_cntrl DO
BEGIN
  readln (inputfile,labes^.yname[loop1]);
  readln (inputfile,labes^.uname[loop1]);
  readln (inputfile,labes^.ylabl[loop1]);
  readln (inputfile,labes^.ulabl[loop1]);
  readln (inputfile,labes^.yunit[loop1]);
  readln (inputfile,labes^.uunit[loop1]);
END;
readln (inputfile,labes^.xlabl);

FOR loop1 := 0 to max_point DO
BEGIN
  FOR loop2 := 1 to max_cntrl DO
    read (inputfile,datlog^.y[loop1]^[loop2]);
  FOR loop2 := 1 to max_cntrl DO
    read (inputfile,datlog^.u[loop1]^[loop2]);
  read (inputfile,datlog^.x[loop1]);
  readln (inputfile);
END;
close (inputfile);

data_plot (datlog,labes);

labes^.uname[row] := 'Estimate';
labes^.ulabl[row] := 'Z';
labes^.uunit[row] := labes^.yunit[row];

exit0 := false;
REPEAT
  clrscr;
  gotoxy (20,01);write ('');
  gotoxy (20,02);write (' CURVE FITTING TO LOGGED DATA ');
  gotoxy (20,03);write ('');
  gotoxy (20,08);write ('');
  gotoxy (20,09);write ('');
  gotoxy (20,10);write (' 1 : First Order ');
  gotoxy (20,11);write ('');
  gotoxy (20,12);write (' 2 : Second Order ');
  gotoxy (20,13);write ('');
  gotoxy (20,14);write ('');
  gotoxy (20,15);write (' X : Exit ');
  gotoxy (20,16);write ('');
  gotoxy (20,17);write ('');
  gotoxy (20,20);write ('');
  gotoxy (20,21);write (' OPTION : ');
  gotoxy (20,22);write ('');
  REPEAT

```

```
gotoxy (36,21);
ch0 := upcase(readkey);
UNTIL ch0 in valid0;
CASE ch0 OF
'1' : order_1;
'2' : order_2;
'X' : exit0 := true;
END;
UNTIL exit0;
clrscr;
END.
```

APPENDIX F : OPEN LOOP TEMPERATURE TRANSFER FUNCTIONS

Transfer functions fitted for the first step test :

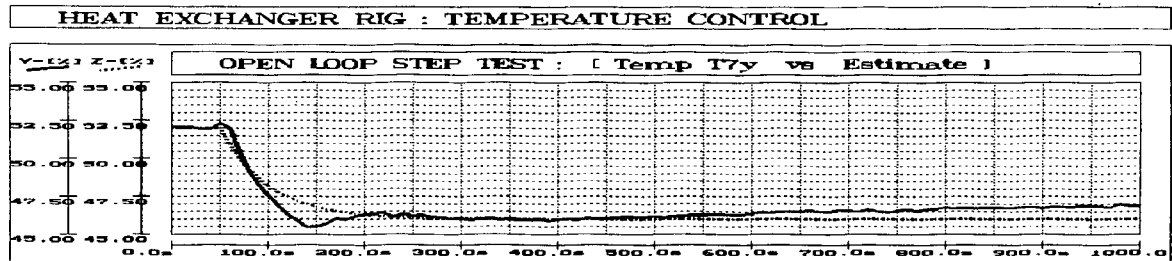


Figure F.1 : TEMPERATURE RESPONSE G11(s)

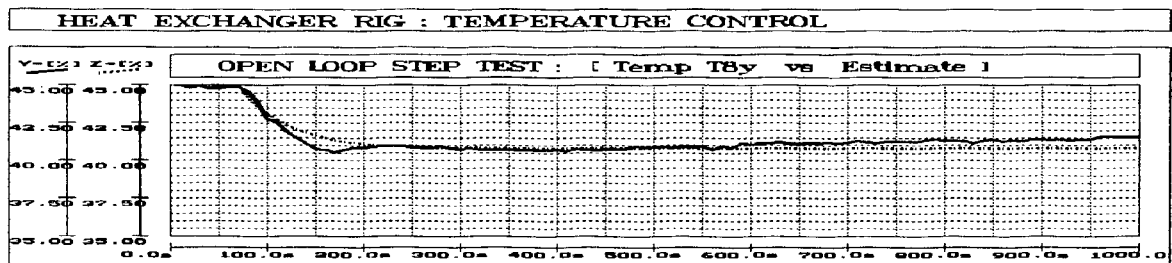


Figure F.2 : TEMPERATURE RESPONSE G21(s)

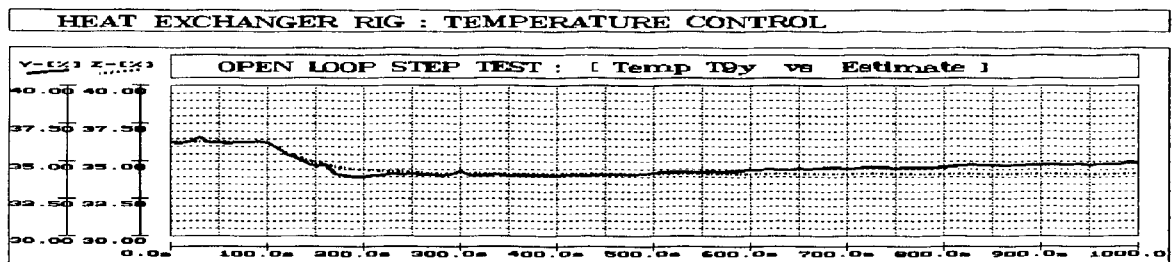


Figure F.3 : TEMPERATURE RESPONSE G31(s)

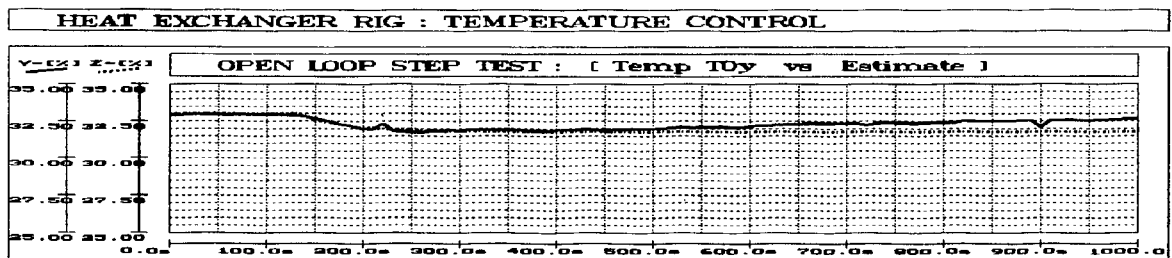


Figure F.4 : TEMPERATURE RESPONSE G41(s)

Transfer functions fitted for the second step test :

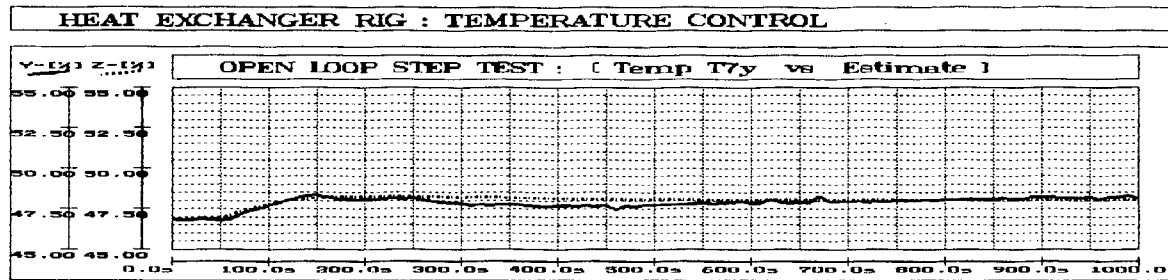


Figure F.5 : TEMPERATURE RESPONSE G12(s)

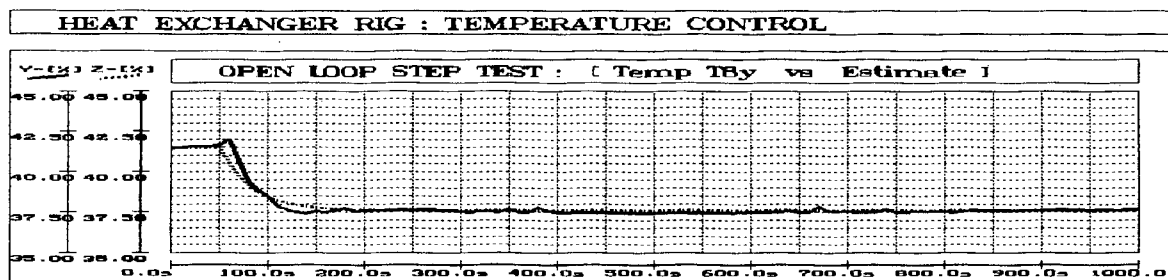


Figure F.6 : TEMPERATURE RESPONSE G22(s)

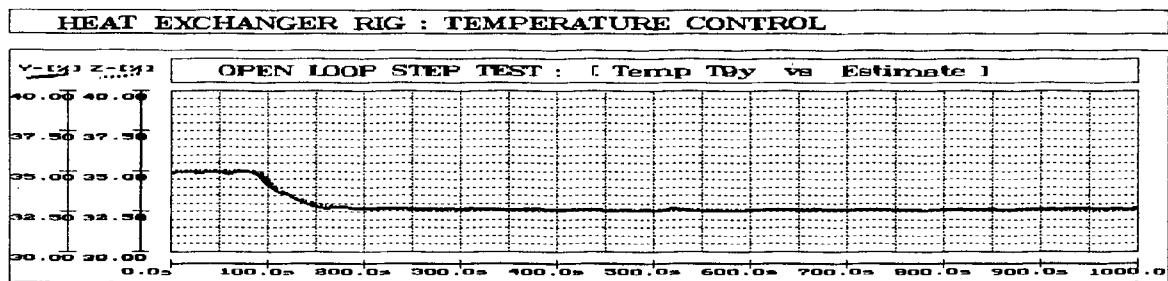


Figure F.7 : TEMPERATURE RESPONSE G32(s)

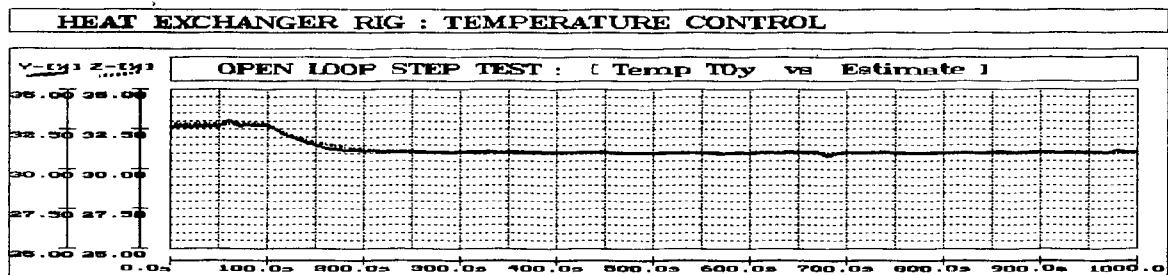


Figure F.8 : TEMPERATURE RESPONSE G42(s)

Transfer functions fitted for the third step test :

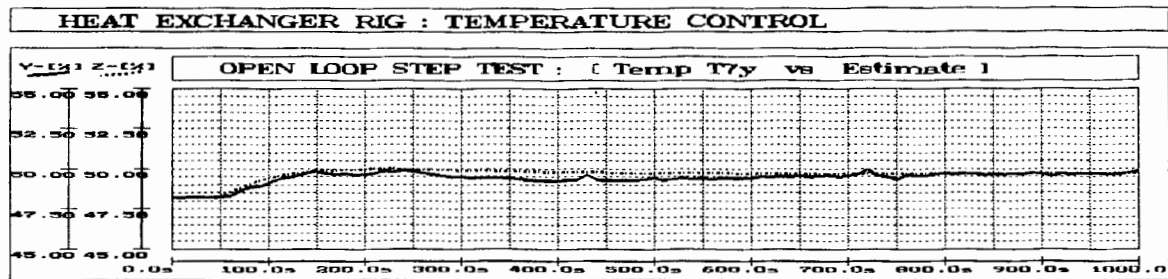


Figure F.9 : TEMPERATURE RESPONSE G13(s)

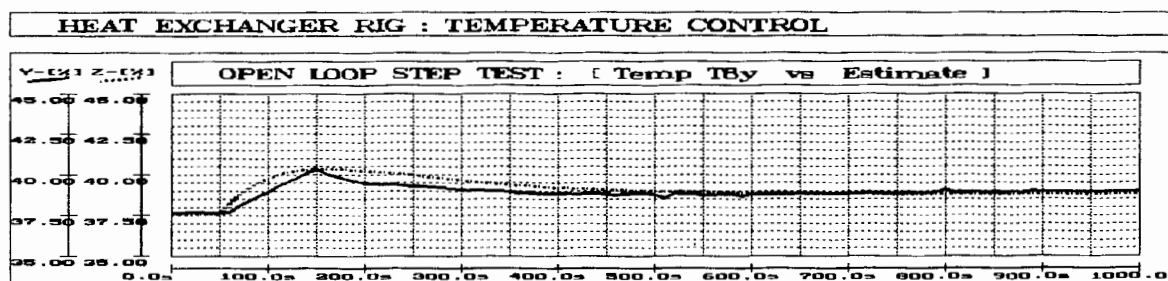


Figure F.10 : TEMPERATURE RESPONSE G23(s)

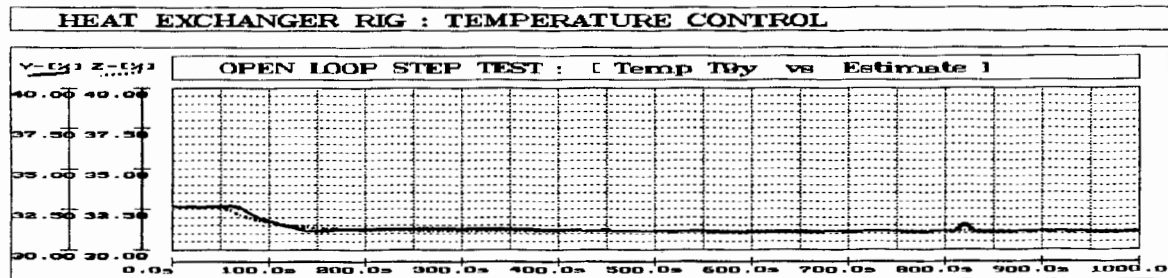


Figure F.11 : TEMPERATURE RESPONSE G33(s)

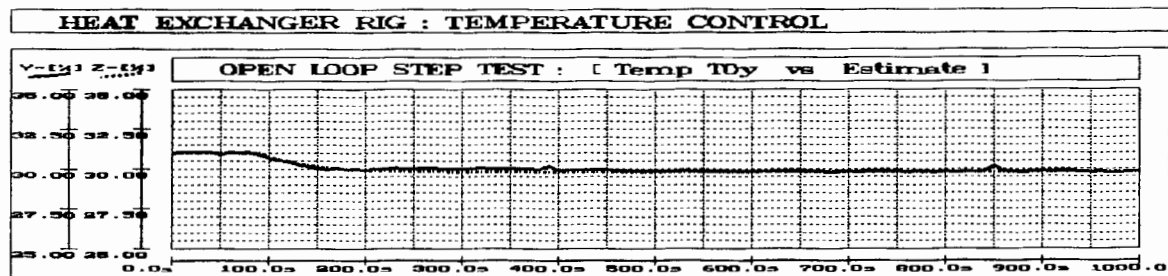


Figure F.12 : TEMPERATURE RESPONSE G43(s)

Transfer functions fitted for the fourth step test :

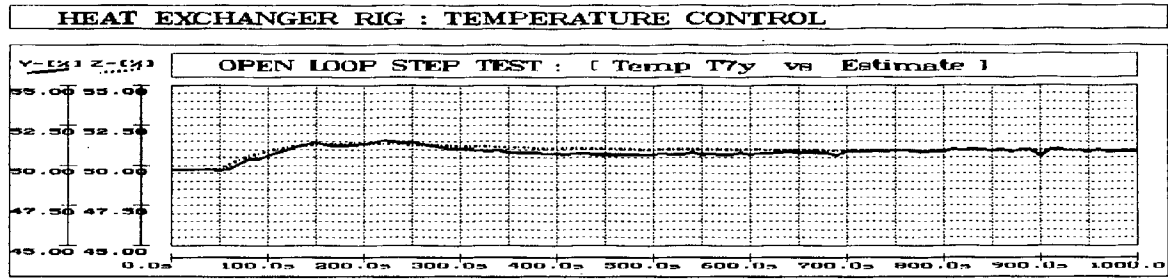


Figure F.13 : TEMPERATURE RESPONSE G14(s)

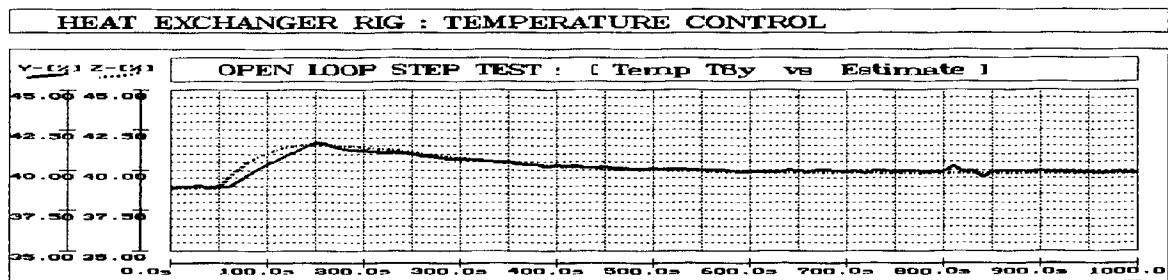


Figure F.14 : TEMPERATURE RESPONSE G24(s)

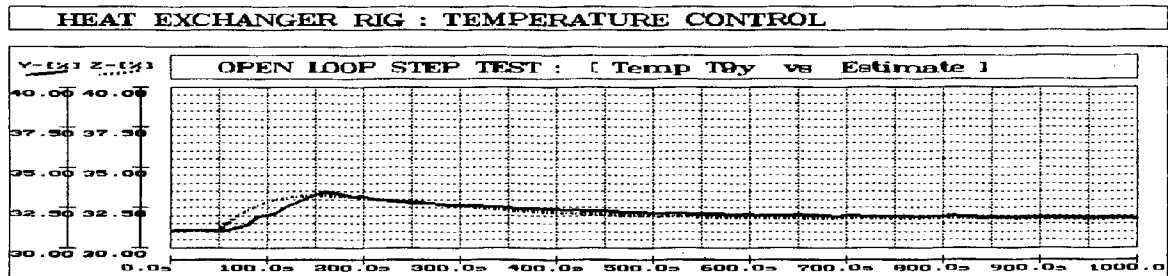


Figure F.15 : TEMPERATURE RESPONSE G34(s)

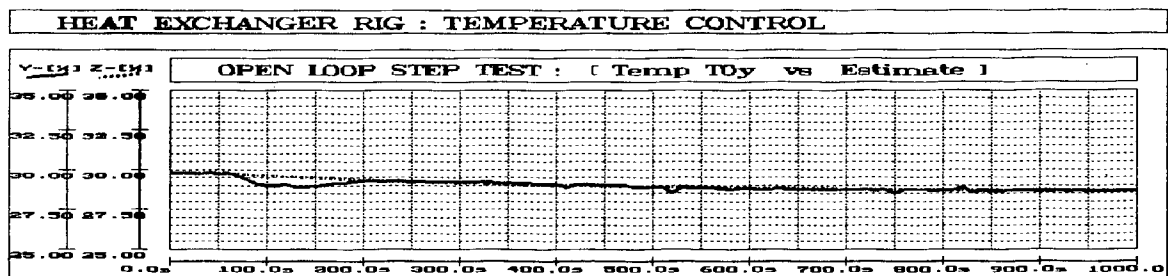


Figure F.16 : TEMPERATURE RESPONSE G44(s)

APPENDIX G : MATLAB SOFTWARE USED FOR H_∞ DESIGNS

G.1 : H_∞ DESIGN PROGRAMME

```
%
% HINFDSGN - Design of an H-infinity controller
%

% LOAD PLANT G(s) IN STATE SPACE FORM
%
clc
fileg=input('Name of plant = ','s');
[ag,bg,cg,dg]=loadssm(fileg);
xg=size(ag)*[1;0];
ng=size(cg)*[1;0];

% DESIGN OF WEIGHTING FUNCTIONS
%
clc
w=logspace(-5,1,50);

% ----- W1 : weighting on the Sensitivity function S(s)
aw1=[];
bw1=[];
cw1=[];
dw1=[];
gain(1,:)=[60];loss(1,:)=[4 60];freq(1,:)=[3.0E-6 3.0E-3];
gain(2,:)=[60];loss(2,:)=[4 60];freq(2,:)=[3.0E-6 3.0E-3];
gain(3,:)=[60];loss(3,:)=[4 60];freq(3,:)=[3.0E-6 3.0E-3];
gain(4,:)=[60];loss(4,:)=[4 60];freq(4,:)=[3.0E-6 3.0E-3];
disp('          ----- Design of : W1 -----');
for i=1:ng
    [a,b,c,d] = BUTTWRTH ('LP',gain(i),loss(i,:),freq(i,:));
    [aw1,bw1,cw1,dw1] = APPEND (aw1,bw1,cw1,dw1,a,b,c,d);
    [mw1(:,i),pw1(:,i)]=bode(a,b,c,d,1,w);
end
wlinv=-20*log10(mw1);

% ----- W2 : weighting on the Input action function R(s)
aw2=[];
bw2=[];
cw2=[];
dw2=[];
gain(1,:)=[60];loss(1,:)=[4 60];freq(1,:)=[5.0E+2 5.0E-1];
gain(2,:)=[60];loss(2,:)=[4 60];freq(2,:)=[5.0E+2 5.0E-1];
gain(3,:)=[60];loss(3,:)=[4 60];freq(3,:)=[5.0E+2 5.0E-1];
gain(4,:)=[60];loss(4,:)=[4 60];freq(4,:)=[5.0E+2 5.0E-1];
disp('          ----- Design of : W2 -----');
```

```

for i=1:ng
    [a,b,c,d] = BUTTWRTH ('HP',gain(i),loss(i,:),freq(i,:));
    [aw2,bw2,cw2,dw2] = APPEND (aw2,bw2,cw2,dw2,a,b,c,d);
    [mw2(:,i),pw2(:,i)]=bode(a,b,c,d,1,w);
end
w2inv=-20*log10(mw2);

% ----- W3 : weighting on the Closed loop transfer function T(s)
aw3=[];
bw3=[];
cw3=[];
dw3=[];
gain(1,:)=[60];loss(1,:)=[4 60];freq(1,:)=[5.0E+0 5.0E-3];
gain(2,:)=[60];loss(2,:)=[4 60];freq(2,:)=[5.0E+0 5.0E-3];
gain(3,:)=[60];loss(3,:)=[4 60];freq(3,:)=[5.0E+0 5.0E-3];
gain(4,:)=[60];loss(4,:)=[4 60];freq(4,:)=[5.0E+0 5.0E-3];
disp('----- Design of : W3 -----');
for i=1:ng
    [a,b,c,d] = BUTTWRTH ('HP',gain(i),loss(i,:),freq(i,:));
    [aw3,bw3,cw3,dw3] = APPEND (aw3,bw3,cw3,dw3,a,b,c,d);
    [mw3(:,i),pw3(:,i)]=bode(a,b,c,d,1,w);
end
w3inv=-20*log10(mw3);

semilogx (w,w1inv,w,w2inv,w,w3inv)
title ('WEIGHTING FUNCTIONS - 1/W1 & 1/W2 & 1/W3')
xlabel('Frequency - [Rad/s]')
ylabel('Magnitude - [db]')
grid
pause

% H-INFINITY SYNTHESIS
sysq=[aq bq;cq dq];
sysw1=[aw1 bw1;cw1 dw1];
sysw2=[aw2 bw2;cw2 dw2];
sysw3=[aw3 bw3;cw3 dw3];
dim=[xq size(aw1)*[1;0] size(aw2)*[1;0] size(aw3)*[1;0]];
[A,B1,B2,C1,C2,D11,D12,D21,D22]=AUGMENT(sysq,sysw1,sysw2,sysw3,dim);

% STEP 1 : ADJUST FOR LEVEL OF PERFORMANCE gamma
gamma=0.9;
A=A;
B1=B1/sqrt(gamma);
B2=B2/sqrt(gamma);
C1=C1/sqrt(gamma);
C2=C2/sqrt(gamma);
D11=D11/gamma;
D12=D12;
D21=D21;
D22=D22*gamma;

% STEP 2 : SCALE INPUTS AND OUTPUTS
Sunom=chol(D12.'*D12);
Syinv=chol(D21*D21.').';

```

```
Suinv=inv(Sunom);
Synom=inv(Syinv);
A=A;
B1=B1;
B2=B2*Suinv;
C1=C1;
C2=Synom*C2;
D11=D11;
D12=D12*Suinv;
D21=Synom*D21;
D22=Synom*D22*Suinv;
```

```
% STEP 3 : PERFORM SYNTHESIS OF H-INFINITY CONTROLLER
```

```
hinf
```

```
pause
```

```
% STEP 4 : INCORPORATE Gamma & SCALING INTO CONTROLLER
```

```
%
```

```
ak=acp;
```

```
bk=sqrt(gamma)*bcp*Synom;
```

```
ck=sqrt(gamma)*Suinv*ccp;
```

```
dk=gamma*Suinv*dcp*Synom;
```

```
[filek]=savessm(ak,bk,ck,dk);
```

G.2 : BUTTERWORTH FILTER DESIGN PROGRAMME

```
function [a,b,c,d] = BUTTWRTH (case,gain,loss,freq)

%
%
%

% Butterworth ALL PASS filter
if case=='AP'
    a=[];
    b=[];
    c=[];
    d=10^(gain/20);

% Butterworth LOW PASS filter
elseif case=='LP'
    Amax=loss(1);
    Amin=loss(2);
    wp=freq(1);
    ws=freq(2);

    n=ceil(log10((10^(0.1*Amin)-1)/(10^(0.1*Amax)-1))/log10((ws/wp)^2));
    [z,p,k] = BUTTAP (n);
    [a,b,c,d] = ZP2SS (z,p,k);
    [a,b,c,d] = LP2LP (a,b,c,d,wp);
    c=10^(gain/20)*c;
    d=10^(gain/20)*d;

% Butterworth HIGH PASS filter
elseif case=='HP'
    Amax=loss(1);
    Amin=loss(2);
    wp=freq(1);
    ws=freq(2);

    n=ceil(log10((10^(0.1*Amin)-1)/(10^(0.1*Amax)-1))/log10((wp/ws)^2));
    [z,p,k] = BUTTAP (n);
    [a,b,c,d] = ZP2SS (z,p,k);
    [a,b,c,d] = LP2HP (a,b,c,d,wp);
    c=10^(gain/20)*c;
    d=10^(gain/20)*d;

% Butterworth BAND PASS filter
elseif case=='BP'
    Amax=loss(1);
    Amin=loss(2);
    w1=freq(1);
    w2=freq(2);
    w3=freq(3);
    w4=freq(4);

    n=ceil(log10((10^(0.1*Amin)-1)/(10^(0.1*Amax)-1))/log10((w4-w3)/(w2-w1)^2));
```

```
[z,p,k] = BUTTAP (n);
[a,b,c,d] = ZP2SS (z,p,k);
[a,b,c,d] = LP2BP (a,b,c,d,sqrt(w2*w1),(w2-w1));
c=10^(gain/20)*c;
d=10^(gain/20)*d;

% Butterworth BAND REJECT filter
elseif case=='BR'
    Amax=loss(1);
    Amin=loss(2);
    w1=freq(1);
    w2=freq(2);
    w3=freq(3);
    w4=freq(4);

    n=ceil(log10((10^(0.1*Amin)-1)/(10^(0.1*Amax)-1))/log10((w2-w1)/(w4-w3)^2));
    [z,p,k] = BUTTAP (n);
    [a,b,c,d] = ZP2SS (z,p,k);
    [a,b,c,d] = LP2BS (a,b,c,d,sqrt(w2*w1),(w2-w1));
    c=10^(gain/20)*c;
    d=10^(gain/20)*d;

end
```

G.3 : SAVING MATRIX IN TRANSFER FUNCTION FORM

```

PROGRAM mtrx2dat;

USES
    crt;

CONST
    maxn = 8;
    maxo = 19;

TYPE
    functype = RECORD
        nsz : integer;
        num : array [0..maxo] of real;
        dsz : integer;
        den : array [0..maxo] of real;
        put : boolean;
    END;

VAR
    row,col : integer;
    ch : char;
    filename : string;
    outfile : text;
    exit : boolean;
    i,j : integer;
    n : integer;
    xo,yo : integer;
    func : array [1..maxn,1..maxn] of functype;

PROCEDURE screen;
BEGIN
    clrscr;
    yo := 1 + trunc((25-(3*n+1))/2);
    xo := 4 + trunc((73-(9*n+1))/2);
    FOR i := 1 to n DO
        FOR j := 1 to n DO
            BEGIN
                gotoxy(xo+(9*j-8),yo+(3*i-3));write('———');
                gotoxy(xo+(9*j-8),yo+(3*i-0));write('———');
                gotoxy(xo+(9*j-9),yo+(3*i-2));write(' ');
                gotoxy(xo+(9*j-9),yo+(3*i-1));write(' ');
                gotoxy(xo+(9*j-0),yo+(3*i-2));write(' ');
                gotoxy(xo+(9*j-0),yo+(3*i-1));write(' ');
                gotoxy(xo+(9*j-9),yo+(3*i-3));write(' ');
                gotoxy(xo+(9*j-0),yo+(3*i-3));write(' ');
                gotoxy(xo+(9*j-9),yo+(3*i-0));write(' ');
                gotoxy(xo+(9*j-0),yo+(3*i-0));write(' ');
                IF func[i,j].put THEN textattr := black + lightgray*16;
                gotoxy(xo+(9*j-7),yo+(3*i-2));write('G (s) ');
                gotoxy(xo+(9*j-7),yo+(3*i-1));write(' ',i:1,j:1,' ');
                textattr := black + blue*16;
            END
        END
    END

```

```

END;

FOR i := 1 to n DO
  IF (i=n) THEN
    BEGIN
      gotoxy(xo, yo); write('┌');
      gotoxy(xo+(9*i), yo); write('┐');
      gotoxy(xo, yo+(3*i)); write('└');
      gotoxy(xo+(9*i), yo+(3*i)); write('┘');
    END
  ELSE
    BEGIN
      gotoxy(xo+(9*i), yo); write('┌');
      gotoxy(xo+(9*i), yo+(3*n)); write('┐');
      gotoxy(xo, yo+(3*i)); write('└');
      gotoxy(xo+(9*n), yo+(3*i)); write('┘');
    END;
    gotoxy(xo+(9*col-3), yo+(3*row-1));
  END;
END;

PROCEDURE xferfunction;
BEGIN
  REPEAT
    clrscr;
    write('Numerator order = ');
    readln(func[row,col].nsz);
    writeln;
    FOR i := 0 to func[row,col].nsz DO
      BEGIN
        write('s^', i:2, ' = ');
        readln(func[row,col].num[i]);
      END;

    clrscr;
    write('Denominator = ');
    readln(func[row,col].dsz);
    writeln;
    FOR i := 0 to func[row,col].dsz DO
      BEGIN
        write('s^', i:2, ' = ');
        readln(func[row,col].den[i]);
      END;

    clrscr;
    gotoxy(1, 4); write (' ');
    gotoxy(1, 5); write (' ');
    gotoxy(1, 6); write (' ');
    gotoxy(1, 7); write (' ');
    gotoxy(1, 8); write (' ');
    gotoxy(1, 9); write (' ');
    gotoxy(1, 10); write (' ');
    gotoxy(1, 11); write (' ');
    gotoxy(1, 12); write (' ');
    gotoxy(1, 13); write (' ');
  
```

```

gotoxy(1,14);write ('
gotoxy(1,15);write ('
gotoxy(1,16);write ('
gotoxy(1,17);write ('
gotoxy(1,18);write ('
');
');
');
');
');

FOR i := func[row,col].nsz downto 0 do
BEGIN
    gotoxy(15+16*trunc(4.0*frac((func[row,col].nsz-i)/4.0)),
        10-trunc(int(func[row,col].nsz/4.0)-int((func[row,col].nsz-i)/4.0)));
    IF (func[row,col].num[i] < 1E1) THEN write(func[row,col].num[i]:9:6)
    ELSE IF (func[row,col].num[i] < 1E2) THEN write(func[row,col].num[i]:9:5)
    ELSE IF (func[row,col].num[i] < 1E3) THEN write(func[row,col].num[i]:9:4)
    ELSE IF (func[row,col].num[i] < 1E4) THEN write(func[row,col].num[i]:9:3)
    ELSE IF (func[row,col].num[i] < 1E5) THEN write(func[row,col].num[i]:9:2)
    ELSE
        write(func[row,col].num[i]:9:1);
    write('s^',i:2);
    IF (i <> 0) THEN write(' + ');
END;

gotoxy(4,11);write('G ',row:1,col:1,' (s) = _____');

FOR i := func[row,col].dsz downto 0 do
BEGIN
    gotoxy(15+16*trunc(4.0*frac((func[row,col].dsz-i)/4.0)),
        12+trunc(int((func[row,col].dsz-i)/4.0)));
    IF (func[row,col].den[i] < 1E1) THEN write(func[row,col].den[i]:9:6)
    ELSE IF (func[row,col].den[i] < 1E2) THEN write(func[row,col].den[i]:9:5)
    ELSE IF (func[row,col].den[i] < 1E3) THEN write(func[row,col].den[i]:9:4)
    ELSE IF (func[row,col].den[i] < 1E4) THEN write(func[row,col].den[i]:9:3)
    ELSE IF (func[row,col].den[i] < 1E5) THEN write(func[row,col].den[i]:9:2)
    ELSE
        write(func[row,col].den[i]:9:1);
    write('s^',i:2);
    IF (i <> 0) THEN write(' + ');
END;

gotoxy(1,23);write ('
gotoxy(1,24);write ('
gotoxy(1,25);write ('
gotoxy(5,24);write ('REPEAT (Y/N) ? ');
ch := upcase(readkey);
func[row,col].put := true;
UNTIL ch = 'N';
END;

BEGIN
REPEAT
    clrscr;
    write ('ORDER OF G(s) = ');
    n := ord(readkey) - ord('0');
    UNTIL (n <= 8) AND (n >= 1);

FOR i := 1 to maxn DO
FOR j := 1 to maxn DO

```

```
BEGIN
    func[i,j].nsz := 0;
    func[i,j].num[0] := 0;
    func[i,j].dsz := 0;
    func[i,j].den[0] := 1;
    func[i,j].put := false;
END;
row := 1;
col := 1;
exit := false;
REPEAT
    screen;
    ch := readkey;
    IF (ch=chr(0)) THEN ch := readkey;
    CASE ch OF
        chr(75) : IF (col > 1) THEN dec(col);
        chr(77) : IF (col < n) THEN inc(col);
        chr(72) : IF (row > 1) THEN dec(row);
        chr(80) : IF (row < n) THEN inc(row);
        chr(13) : xferfunction;
        chr(27) : exit := true;
    END;
UNTIL exit;

clrscr;
write ('Filename : ');
readln(filename);

IF (filename <> '') THEN
    BEGIN
        assign (outfile,filename+'.dat');
        rewrite(outfile);
        writeln (outfile,n);
        FOR col := 1 to n DO
            FOR row := 1 to n DO
                BEGIN
                    writeln (outfile,func[row,col].nsz);
                    FOR i := func[row,col].nsz downto 0 DO
                        writeln (outfile,func[row,col].num[i]);
                    writeln (outfile,func[row,col].dsz);
                    FOR i := func[row,col].dsz downto 0 DO
                        writeln (outfile,func[row,col].den[i]);
                    END;
                close (outfile);
            END;
        END.
    END.
```

G.4 : LOADING MATRIX IN TRANSFER FUNCTION FORM

```
function [a,b,c,d] = LOADTFM (filename)

%
% function [a,b,c,d] = LOADTFM (filename)
%
% Reads an nxn transfer function matrix into NUM and DEN
% from a file FILENAME. The polynomials are read one column
% at a time. The matrices NUM and DEN are then padded with
% zeros to fit the highest degree polynomial read in. The
% matrix is then returned in state space form as [A,B,C,D]

% Load the matrix from file
eval(['load ',filename,'.tfm;'])
eval(['data=',filename,',';'])
n=data(1);

% Calculate the maximum order polynomial in each column
nnd=1;
for i=1:n
    max(i)=0;
    for j=1:n
        beg=nnd+2;
        nnd=beg+data(beg-1);
        if (nnd-beg+1) > max(i), max(i)=(nnd-beg+1);, end

        beg=nnd+2;
        nnd=beg+data(beg-1);
        if (nnd-beg+1) > max(i), max(i)=(nnd-beg+1);, end
    end
end

% Obtain state space realization for each column
% Parallel these together to form overall A B C D model
a=[];b=[];c=[];d=[];
nnd=1;
for i=1:n
    num=[];
    den=[];
    numi=[];
    deni=[];

    for j=1:n
        beg=nnd+2;
        nnd=beg+data(beg-1);
        for z=beg:nnd, num(j,max(i)-(nnd-beg)+(z-beg))=data(z);,end

        beg=nnd+2;
        nnd=beg+data(beg-1);
        for z=beg:nnd, den(j,max(i)-(nnd-beg)+(z-beg))=data(z);,end
    end
end
```

```
for j=1:n
    x=1;
    for z=1:n
        if z==j
            x=conv(x,num(z,:));
        else
            x=conv(x,den(z,:));
        end
    end
    numi(j,:)=x;
end

y=1;
for z=1:n
    y=conv(y,den(z,:));
end
deni=y;

[ai,bi,ci,di]=tf2ss(numi,deni);
[a,b,c,d]=parallel(a,b,c,d,ai,bi,ci,di);
end
```

G.5 : SAVING MATRIX IN STATE SPACE FORM

```
function [filename] = SAVESSM (a,b,c,d)

%
% function [filename] = SAVESSM (a,b,c,d)
%
% Saves an nxn transfer function matrix in
% [A,B,C,D] form. A first line contains
% the number of states and inputs/outputs

clc
filename=input('Name of file = ','s');

data=[a b;c d];
rplc=size(data)*[1;0];
data=[size(a)*[1;0] size(d)*[0;1] 3:rplc;data];
eval(['save ',filename,'.ssm data /ascii /double'])
```

G.6 : LOADING MATRIX IN STATE SPACE FORM

```
function [a,b,c,d] = LOADSSM (filename)

%
% function [a,b,c,d] = LOADSSM (filename)
%
% Reads an nxn transfer function matrix, stored in
% state space form, into [A,B,C,D]. The first element
% of the last line contains the number of states,
% while the second has the number of inputs/outputs

% Load the matrix from file
eval(['load ',filename,'.ssm;'])
eval(['data=',filename,';'])
x=data(1,1);
u=data(1,2);

% Matrix : A
a=data(2:x+1,1:x);

% Matrix : B
b=data(2:x+1,x+1:x+u);

% Matrix : C
c=data(x+2:x+u+1,1:x);

% Matrix : D
d=data(x+2:x+u+1,x+1:x+u);
```