

**DESIGN OF CONTROL SYSTEMS
BASED ON
Q(YOULA) -PARAMETRIZATION**

A thesis submitted to the Department of Electrical
and Electronic Engineering, University of Cape
Town, in partial fulfilment of the requirements
for the MSc Degree in Engineering

by

F C JACOBS

September 1992

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.

ACKNOWLEDGEMENTS

I would like to thank the following:

My Supervisor, Prof M Braae for his patience and dedicated guidance throughout the duration of this project,

and my fellow students, B J Olivier, W H Trossbach and G A Crosley for their assistance in clarifying some ideas, in general, on Multivariable controller design techniques in the initial stages of this project. In addition, special thanks to B J Olivier and W H Trossbach for letting me use two of their software routines.

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.

ABSTRACT

This thesis is concerned with the application of a method for control system design based on Youla(Q)-Parametrization. The fundamental concepts of the parametrization method have been used by Newton et al, Franklin and Raggazzini [Maciejowski, 1989] as early as 1957. The importance of its implications were not noticed during those years, but Youla and Kucera renewed it in the seventies [Maciejowski, 1989]. For the special case of an open loop stable plant, the implementation can be realized in the Internal Model Control(IMC)-Structure [Morari, 1987]. The IMC-Structure is essentially a open loop design method for closed loop controllers, thereby simplifying the design procedure.

A Multivariable controller design method is proposed that unifies the advantages offered by three design approaches/techniques, namely, The Q-Parametrization, The IMC-Structure and The Characteristic Loci Technique. The proposed method is based on controller parametrization. In addition, it retains the engineering insight and simplicity of the IMC-Structure. This is particularly useful in solving complex multivariable problems.

The design method was evaluated by its application to the design of a control law for a FLOTATION PROCESS SIMULATOR RIG(FLOTIRIG) built by Mr Fisher[Fisher, 1988]. The design of

the Multivariable controller was based on a Transfer Function Matrix, $G(s)$, which is a Linear Time Invariant(LTI) model of the FLOTRIG.

The final control system was shown to maintain the plant outputs at set values. The extent of interaction on the FLOTRIG was also reduced by the controller. The closed-loop experimental results were comparable with the results predicted by theory and a digital simulation study. The design approach greatly simplifies the search for a controller and gives considerable engineering insight to the designer at the design stage.

TABLE OF CONTENTS

	<i>Page</i>
Title Page	
Acknowledgements	i
Abstract	ii
Table of Contents	iv
List of Illustrations	ix
Nomenclature	xiv
 Chapter One: Introduction.....	 1
 Chapter Two: Multivariable Control Design Methods	 8
2.1 The Q(Youla)-Parametrization Method	9
2.1.1 Coprime MFDs & Fractional Representation	10
2.1.2 Relevant TFM Parametrization Relationships ...	17
2.1.3 Comments on the Q(Youla)-Parametrization Method	20
2.2 The IMC Design Method	20
2.2.1 The IMC-Structure	21
2.2.2 The IMC condition(s) for Internal and Closed Loop Stability	23
2.2.3 The IMC-Structure Design Method	25
2.2.4 Comments on the IMC design method	29
2.3 The Characteristic Loci Design Method	30
2.3.1 Interaction Minimization	31
2.3.2 Matrix PI-Controller	31

2.3.3 Comments on the Characteristic Loci Method ...	33
--	----

Chapter Three: Proposed Simplifying Control System Design

Approach' based on Q-Parametrization	35
3.1 Simplifying Design Approach(SISO-case)	37
3.1.1 Matrix Conditions for System Type Numbers	37
3.1.2 Modification to include Dynamic Compensation..	43
3.1.3 Alteration of Speed of Response	45
3.2 Simplifying Design Approach(MIMO-case)	47
3.2.1 Decoupling Controller Design	47
3.3 Setpoint Tracking Conditions for the Q-Parameter ...	51

Chapter Four: Flotation Rig(FLOTRIG) Identification58

4.1 Description of a typical Flotation Process	58
4.2 Description of the Flotation Rig	60
4.3 Flotation Cell Simulation	61
4.4 Instrumentation and Lay-out of the FLOTRIG	62
4.5 Identification of the FLOTRIG model	63
4.6 Open Loop Step Test on the FLOTRIG	63
4.6.1 FLOTRIG Linear Operating Region	64
4.6.2 Experimental Open Loop Step Tests	65
4.6.3 Final Open Loop Step Tests	68
4.7 Development of the FLOTRIG Model	69
4.7.1 Calculation of Transfer Function Models based on FLOTRIG Data	70
4.7.2 Generation of the FLOTRIG Transfer Function Matrix	70

Chapter Five: Controller Design for the FLOTRIG using The	
Proposed Design Approach based on Q(Youla)-	
Parametrization	73
5.1 Design of the Control System	74
5.2 Q-Parameter Design for Step Setpoint Tracking	74
5.3 Q-Parameter Design for Ramp Setpoint Tracking	82
 Chapter Six: Control Scheme Implementation and Comparison of	
PC-Simulation and Experimental Results	89
6.1 Comparison of Results for Step Setpoints	90
6.1.1 Comparison of Process Outputs(STEP)	91
6.1.2 Comparison of Process I/Ps and Model	
O/Ps(STEP)	96
6.2 Comparison of Results for Ramp Setpoints	101
6.2.1 Comparison of Process Outputs(RAMP)	104
6.2.2 Comparison of Process I/Ps and Model	
O/Ps(RAMP)	108
6.3 Comparison of Results for non-tracked Ramp	
Setpoints	114
6.3.1 Comparison of Process Outputs(non RAMP	
tracking)	114
6.3.2 Comparison of Process I/Ps and Model	
O/Ps(no RAMP tracking)	118
6.4 Observations on the results presented above	124
6.4.1 Comments on results obtained from step	
setpoints	124

6.4.2 Comments on results obtained from <i>ramp</i> setpoints	125
6.4.3 Comments on results obtained from <i>ramp</i> setpoints where $Q(s)$ was designed for <i>step</i> setpoint tracking.....	126
6.4.4 Comparison of results to previous INA and CL designs on the FLOTRIG	127
Chapter Seven: Conclusions.....	132
7.1 Summarizing conclusion on the literature survey	132
7.2 Multivariable Control System Design Methods	132
7.3 Proposed Simplifying Control System Design Approach	133
7.4 FLOTRIG Identification	133
7.5 Control Scheme Implementation and comparison of results	134
References	136
Bibliography	141

Appendix A: FLOTRIG Open Loop Step Tests	146
A.1 Open Loop Step Test - Rougher	147
A.2 Open Loop Step Test - Scavenger	148
A.3 Open Loop Step Test - Cleaner	149
A.4 Open Loop Step Test - Recleaner	150
 Appendix B: Data Analysis and FLOTRIG model estimates ...	151
B.1 Rougher level plot-Rougher tail valve stepped	152
B.2 Scavenger level plot-Rougher tail valve stepped ..	153
B.3 Cleaner level plot-Rougher tail valve stepped	154
B.4 Rougher level plot-Scavenger tail valve stepped ..	155
B.5 Scavenger level plot-Scavenger tail valve stepped	156
B.6 Cleaner level plot-Scavenger tail valve stepped ..	157
B.7 Rougher level plot-Cleaner tail valve stepped	158
B.8 Scavenger level plot-Cleaner tail valve stepped ..	159
B.9 Cleaner level plot-Cleaner tail valve stepped	160
B.10 Cleaner level plot-Recleaner tail valve stepped ..	161
B.11 Recleaner level plot-Recleaner tail valve stepped	162
B.12 Comments on Data and their estimates	163
 Appendix C: Design of a Q-Parameterized Controller	
implemented in the IMC-Structure for a Ramp Tracking	
Servomotor system	166
C.1 The Open Loop Servomotor System	167
C.2 Internal Model Control Design Method	168
C.3 Type numbers in IMC	172

Appendix D: Transmittance and Disturbance rejection	
analysis	178
D.1 Plant-Model mismatch Bode plots	181
D.2 Plant-Model match Bode plots	185
 Appendix E: FLOTRIG-Computer Interface Routines	189
E.1 ADC Interface Routines	189
E.2 DAC Interface Routines	191
 Appendix F: Listing of Step Test Routines	193
F.1 Program Listing(Step)	194
 Appendix G: Listing of Ramp Test Routines	221
G.1 Program Listing(Ramp)	222

LIST OF ILLUSTRATIONS

Figures	Page
 Chapter Two	
2.1 Positive feedback Closed-Loop Configuration	10
2.2 Classical feedback configuration $L(G,K)$	16
2.3 Controller structure for a stable plant	17
2.4 The Basic IMC-Structure	21
2.5 The Classical unity feedback structure	22
2.6 The Classical structure with $K(Q,M)$	22
2.7 Block diagram for derivation of IS conditions	23
 Chapter Four	
4.1 Flotation Rig Circuit	58
4.2 Simulated Flotation Plant Cell	61
4.3 Flotation Rig(FLOTRIG)	62
4.4 Flotation Cell Linear Operation Region.....	64
 Chapter Five	
5.1 The IMC-Structure.....	80
 Chapter Six	
6.1 O/Ps, $y_R[r_1(s)$ stepped +10%]	93
6.2 O/Ps, $y_S[r_1(s)$ stepped +10%]	93
6.3 O/Ps, $y_R[r_2(s)$ stepped +10%]	94
6.4 O/Ps, $y_S[r_2(s)$ stepped +10%]	94
6.5 O/Ps, $y_R[r_3(s)$ stepped + 5%]	95
6.6 O/Ps, $y_S[r_3(s)$ stepped +10%]	95

6.7 O/Ps, $y_R[r_4(s)$ stepped -10%]	96
6.8 O/Ps, $y_S[r_4(s)$ stepped +10%]	96
6.9 u, $y_m[r_1(s)$ stepped +10%]-R	98
6.10 u, $y_m[r_1(s)$ stepped +10%]-S	98
6.11 u, $y_m[r_2(s)$ stepped +10%]-R	99
6.12 u, $y_m[r_2(s)$ stepped +10%]-S	99
6.13 u, $y_m[r_3(s)$ stepped +5%]-R	100
6.14 u, $y_m[r_3(s)$ stepped +10%]-S	100
6.15 u, $y_m[r_4(s)$ stepped -10%]-R	101
6.16 u, $y_m[r_4(s)$ stepped +10%]-S	101
6.17 O/Ps, $y_R[r_1(s)$ ramp $\pm$ 4%]	105
6.18 O/Ps, $y_S[r_1(s)$ ramp $\pm$ 4%]	105
6.19 O/Ps, $y_R[r_2(s)$ ramp $\pm$ 4%]	106
6.20 O/Ps, $y_S[r_2(s)$ ramp $\pm$ 4%]	106
6.21 O/Ps, $y_R[r_3(s)$ ramp $\pm$ 4%]	107
6.22 O/Ps, $y_S[r_3(s)$ ramp $\pm$ 4%]	107
6.23 O/Ps, $y_R[r_4(s)$ ramp $\pm$ 4%]	108
6.24 O/Ps, $y_R[r_4(s)$ ramp $\pm$ 4%]	108
6.25 u & $y_m[r_1(s)$ ramp $\pm$ 4%]-R	110
6.26 u & $y_m[r_1(s)$ ramp $\pm$ 4%]-S	110
6.27 u & $y_m[r_2(s)$ ramp $\pm$ 4%]-R	111
6.28 u & $y_m[r_2(s)$ ramp $\pm$ 4%]-S	111
6.29 u & $y_m[r_3(s)$ ramp $\pm$ 4%]-R	112
6.30 u & $y_m[r_3(s)$ ramp $\pm$ 4%]-S	112
6.31 u & $y_m[r_4(s)$ ramp $\pm$ 4%]-R	113
6.32 u & $y_m[r_4(s)$ ramp $\pm$ 4%]-S	113

6.33 O/Ps, $y_R[r_1(s) \text{ ramp} \pm 4\%]$	115
6.34 O/Ps, $y_R[r_1(s) \text{ ramp} \pm 4\%]$	115
6.35 O/Ps, $y_R[r_2(s) \text{ ramp} \pm 4\%]$	116
6.36 O/Ps, $y_R[r_2(s) \text{ ramp} \pm 4\%]$	116
6.37 O/Ps, $y_R[r_3(s) \text{ ramp} \pm 4\%]$	117
6.38 O/Ps, $y_R[r_3(s) \text{ ramp} \pm 4\%]$	117
6.39 O/Ps, $y_R[r_4(s) \text{ ramp} \pm 4\%]$	118
6.40 O/Ps, $y_R[r_4(s) \text{ ramp} \pm 4\%]$	118
6.41 u & $y_m[r_1(s) \text{ ramp} \pm 4\%]$ -R	120
6.42 u & $y_m[r_1(s) \text{ ramp} \pm 4\%]$ -S	120
6.43 u & $y_m[r_2(s) \text{ ramp} \pm 4\%]$ -R	121
6.44 u & $y_m[r_2(s) \text{ ramp} \pm 4\%]$ -S	121
6.45 u & $y_m[r_3(s) \text{ ramp} \pm 4\%]$ -R	122
6.46 u & $y_m[r_3(s) \text{ ramp} \pm 4\%]$ -S	122
6.47 u & $y_m[r_4(s) \text{ ramp} \pm 4\%]$ -R	123
6.48 u & $y_m[r_4(s) \text{ ramp} \pm 4\%]$ -S	123

Appendix A

A.1 Rougher cell tail valve stepped +10%	146
A.2 Scavenger cell tail valve stepped +5%	147
A.3 Cleaner cell tail valve stepped +5%	148
A.4 Recleaner cell tail valve stepped +10%	149

Appendix B

B.1 $g_{11}(s)$ data & estimate	152
B.2 $g_{21}(s)$ data & estimate	153
B.3 $g_{31}(s)$ data & estimate	154
B.4 $g_{12}(s)$ data & estimate	155

B.5	$g_{22}(s)$ data & estimate	156
B.6	$g_{32}(s)$ data & estimate	157
B.7	$g_{13}(s)$ data & estimate	158
B.8	$g_{23}(s)$ data & estimate	159
B.9	$g_{33}(s)$ data & estimate	160
B.10	$g_{34}(s)$ data & estimate	161
B.11	$g_{44}(s)$ data & estimate	162

Appendix C

C.1	Block Diagram of the Servomotor system	167
C.2	Open Loop Step Test Results	168
C.3	IMC Step Response(Type I)	170
C.4	IMC triangular wave(Ramp) response $q(s)$ designed for step tracking	171
C.5	IMC ramp response $q(s)$ designed for ramp tracking	176
C.6	IMC step response $q(s)$ designed for ramp tracking	177

Appendix D

D	O/L IMC-Structure for Transmittance Analysis	178
D.1	Column 1: $T(s)$ - not matched	181
D.2	Column 2: $T(s)$ - not matched	181
D.3	Column 3: $T(s)$ - not matched	182
D.4	Column 4: $T(s)$ - not matched	182
D.5	Column 1: $T(s)$ - matched	185
D.6	Column 2: $T(s)$ - matched	185
D.7	Column 2: $T(s)$ - matched	186
D.8	Column 3: $T(s)$ - matched	186

Chapter Six

6.1 Simulated and Actual FLOTRIG results - *CL method* ..127

6.2 Simulated and Actual FLOTRIG results - *INA method*..127

6.3 Simulated and Actual FLOTRIG results - *PSD method*..128

6.4 Comparison of discrepancies in [%]129

NOMENCLATURE

abs(X)	matrix with (i,j) element $ x_{ij} $
ADC	Analog to Digital Converter
CL	Characteristic Loci
C/L	Closed-Loop
coeff	coefficient of an independent variable in a polynomial
$c_{ii}(s)$	diagonal elements of the matrix $M(s)Q_0$
D_0	tuning matrix for K_0
D_∞	tuning matrix for K_∞
DAC	Digital to Analog Converter
dB	decibels: $x \text{ dB} = 10(x/20)$
dc	direct current: also used for steady state
deg	degree(of polynomial)
det(of Matrix)	determinant(of matrix)
diag(x_{ii})	matrix with elements x_{ii} on the principal diagonal, also used as diag(x_i)
DNA	Direct Nyquist Array
$d(s)$, $D(s)$	mostly used to denote denominator polynomials also used to denote the disturbance vector
$e(s)$	error vector $\Rightarrow$ setpoint vector-process output
eigenvalue $_j(X)$	or e-value $_j(X)$ denotes the j th eigenvalue of the Matrix X obtained from $\det[X-sI]$
eqn	equation

$F(s)$	augmenting matrix for the decoupling matrix $L(s)$
$G(A, B, C, D)$	State-Space realization of the matrix $G(s)$
$G(s)$	plant/process system matrix
g_{ij}	element in row i , col j of $G(s)$
$H(s)$	Closed-Loop TFM
Hz	unit for linear frequency, Hertz
I	Identity matrix of unspecified dimension
I_n	Identity matrix with dimension n
INA	Inverse Nyquist Array
IS	Internal Stability
I/P	Input, also used to denote current(I) to pressure(P) converters
i	used to indicate the i th row of a matrix
j	used to indicate the j th col of a matrix also used for $(-1)^{\frac{1}{2}}$
$k(s)$	SISO controller transfer function
$K(s)$	MIMO matrix controller
K_0	Matrix to diagonalize the system at dc
K_∞	Matrix to diagonalize the system at high frequency
LHP	Left half plane
LOR	Linear Operating Region
LTI	Linear Time Invariant
$L(s)$	Decoupling matrix given by $L(s) = sQ_\infty + Q_0$
mA	units for current, milliamperes
MFD	Matrix Fraction Description

m	used to denote system type numbers also used to refer to the model
$m(s)$	SISO system model transfer function
$M(s)$	MIMO system model TFM
MIMO	multi-input multi-output
$n(s), N(s)$	mostly used to denote numerator polynomials
$P(s)$	is the diagonal matrix used to add required conditions to the Q_0 matrix to ensure setpoint tracking for all input of the form R/s^m . NB: $P(s) = F(s)R(s)$
PSD	proposed design method
$q(s)$	SISO Q-parameter transfer function
$Q(s)$	Q(Youla)-Parameter TFM [= $L(s)F(s)$]
Q_0	Matrix to diagonalize the system at dc
Q_∞	Matrix to diagonalize the system at high frequency
R & S	used as subscripts to indicate Rig or Digital Simulations, mostly for graphs
$\text{rank}\{X\}$	then maximum number of linearly independent rows/column of a matrix X
$\text{Re}\{z\}$	Real part of complex number z
$r(s)$	setpoint vector
$R(s)$	diagonal matrix added to satisfy higher order setpoint conditions in the matrix $P(s) = F(s)R(s)$
RHP	Right half plane
s	Laplace variable (= jw)

SISO	single-input single-output
TFM	Transfer Function Matrix
$T(s)$	used to denote Transmittance = $G(s)Q(s)$ in the case of an O/L structured system
$u(s)$	controller output vector
ω	angular frequency ($= j2\pi f$)
$Xh(z)$	Z-transform of the matrix $X(s)$ using a Zero order sample and hold (ZOH)
X^T	used to denote the transpose of matrix X
$y(s)$	process, $G(s)$ output vector
$y_m(s)$	model, $M(s)$ output vector
α	tuning parameter in the z-domain, also used to denote change speed of response in another context
β	tuning parameter in the s-domain
σ	used to denote the degree of a polynomial
μ	used to denote the degree of a polynomial
τ	parameter selected during the design of $Q(s)$ for higher order setpoint
δ	used to denote step change
$>$	greater than
$<$	less than
$<>$	not equal to
$\rightarrow$	tends to (mostly used with limits)

Chapter One

Introduction

During the past decades various methods were presented for the design of linear feedback control systems [Morari, 1987]. Examples of these are the multivariable frequency domain techniques introduced by Rosenbrock [Rosenbrock, 1974] and MacFarlane [MacFarlane, 1973]. These methods are more 'engineering' oriented approaches.

Some other proposed and studied techniques such as Q(Youla)-Parametrization [Youla et al, 1976] and Internal Model Control(IMC)-Structure [Garcia, 1982] design methods have been presented. The Q(Youla)-Parametrization has its roots in the Classical Treatment of the Analytic Linear Feedback design presented by Newton [Newton, 1957], [Youla et al, 1976]. The approach presented by Newton is an inherently open loop approach for designing closed loop systems. It should be noted that the IMC-Structure mentioned above is not the same as the Internal Model Principle presented by Wonham [Wonham, 1979].

The IMC-Structure Design Approach is a special case of the Q(Youla)-Parametrization [Morari, 1987]. The developments of these methods have introduced new approaches of great importance to the subject of linear feedback systems.

The Q-parametrization aims to minimize the search for all stabilizing controllers [Maciejowski, 1989] whereas the IMC-Structure allows a rational controller design procedure such that control quality and robustness are affected in a direct manner [Garcia, 1982].

These approaches are more direct alternatives, since they allow the design of closed loop systems by simply specifying an open loop precompensating parameter, $Q(s)$. The classical design of control systems would require the use of convoluted complex variable diagrams that show the open loop transfer function, $Q(s) \{= G(s)K(s)\}$ on say Inverse Nyquist Array(INA) or Direct Nyquist Array(DNA) plots. In order to ensure a satisfactory design with INA and DNA plots, considerable experience is essential [Venzke, 1989], [Fisher, 1988], [Braae, 1991].

The Q(Youla)-Parametrization of controller characterization was presented using fractional factorizations over the ring of polynomial matrices [Youla, 1976]. The main shortcoming of the original formulation as given by the Q(Youla)-Parametrization was that the stabilizing controller may not be proper [Chang, 1990]. This shortcoming was removed by Desoer [Chang, 1990], [Desoer, 1981], who also generalized it to over a general ring of matrices.

Various presentations and approaches on the implementation of the proper stabilizing controller parametrization have been made [Chang, 1990], [Moore, 1990]. These approaches have mostly been using State-Space since a procedure for Matrix Fractional Representation was introduced by Nett et al via State-Space [Maciejowski, 1989].

All these approaches, based on the Q(Youla)-Parametrization, aim to yield a means of solving some general feedback optimization problems. Most commonly the method optimizes some closed-loop properties, with the assurance that the controller does not attempt to cancel some RHP-poles or zeroes of the open-loop system in the 's'-plane [Youla, 1976].

The advantage offered by implementing the Q(Youla)-Parametrization in the IMC-Structure, for an open-loop(O/L) stable plant, is that more information is available to the designer at the design stage. For example, both the closed-loop(C/L) performance and the required input variations in the plant inputs are easily observed, especially when implemented in the IMC-structure. This is something which is missing in the classical structure where digital simulation is often needed to evaluate the design [Braae, 1991].

The Q(Youla)-Parametrization ensures asymptotic stability of the C/L system, whether the plant or process being

controlled is O/L unstable or non-minimum phase(NMP) or both.

The control system design which is presented in this thesis was based on Q-Parametrization which was implemented in the IMC-Structure for a process, $G(s)$, which is O/L stable. If the process is not O/L stable, then it should first be stabilized through classical feedback means, though many of the practical advantages of the Q-Parametrization are then lost.

In this thesis techniques from older methods like Characteristic Loci (CL) were found useful. This thesis proposes that the PI-controller design approach from the CL-method be exploited in the design of controller parameter, $Q(s)$.

The Q-Parametrization and its special case, The IMC-structure, have proven themselves to be significant developments in the control theory. This conclusion is based on the increasing number of presentations that utilize the approach in the design of control systems. For example [Desoer, 1981] did the theory presentation, [Lin, 1991] applied it to decoupling controller design, [Chang, 1990], described it from the point of 'all stabilizing controllers as frequency shaped state estimate feedback', [Meyer, 1990] to multivariate controller parametrization, [Chen, 1988],

CAD system based on Q-parametrization, [Garcia, 1982,1985] the special case IMC theory and simulations and [Boyd et al, 1988] did CAD system and simulation studies.

Although there are increasing presentations of the Parametrization, they are mostly based on theoretical and digital simulation studies. Not much has been presented on experimental study of the Parametrization. In order for the Controller Parametrization to be useful for industry, an increasing number of experimental studies are needed to evaluate its performance in practical situations. These studies will deal with real-world plant or process characteristics like:- non-linearities, valve stickiness or hysteresis and unavoidable disturbances such as for example feedback pump cavitations which may cause oscillations etc..

In the process of searching specifically for literature which dealt with some experimental studies, two papers have been found in which some studies which use the special case of the Q(Youla)-Parametrization, the IMC-Structure Design Approach are presented. The paper by [Wang, 1991] presented some study on an experimental SISO Heat Exchanger and the one by [Arkun, 1986] on a SISO as well as a 2-input 2-output MIMO system. This is insignificant in comparison to the more than twenty literature presentations found that utilized or mentioned the Controller Parametrization since the late seventies.

In this thesis the control system design based on the parametrization was implemented in a pilot study on a SISO servo motor system and then on a much bigger four-input four-output experimental Flotation Rig(FLOTRIG) system.

The FLOTRIG is a 4-input 4-output MIMO system and exhibits more comprehensive real-world control characteristics, which are not manifested in digital simulations studies.

The proposed Simplifying Design Approach based on Q(Youla)-Parametrization method was implemented in the IMC-structure. The parameterized controller was applied to the SISO Servo motor system and the FLOTRIG. But, since the FLOTRIG is a much bigger system, more detailed description, modelling and design is required. The FLOTRIG was modelled as one MIMO lumped parameter model.

The aim of this thesis is to unify the advantages offered by the IMC-Structure and the CL-Method to propose a simplifying controller design approach for the design of control systems based on Q(Youla)-Parametrization. The design is to be implemented in the IMC-Structure.

The objectives of this thesis are thus:-

- 1) to design control systems based on Q-Parametrization.
- 2) to perform a FLOTRIG identification to obtain models.

- 3) to apply the design based on the O/L step test models to the MIMO FLOTRIG system.
- 4) to evaluate the design control system using the digital simulation and FLOTRIG results.
- 5) to draw conclusions from the implementation of the control system which was based on Q-Parametrization.

The thesis is presented in the form of chapters. Each chapter covers the relevant aspects of the objectives as given above with some chapters added where necessary.

CHAPTER TWO

Multivariable Control System Design Methods

In this chapter a background and overview of three control system design methods are presented. They are:-

- * Q(Youla)-Parametrization
- * IMC-Structure Design Method [a special case of the Q(Youla)-Parametrization]
- * Characteristic Loci Design Method: Matrix PI-Controller

The chapter starts with the Q(Youla)-Parametrization [Youla et al, 1976], since it is the foundation upon which the thesis has been put together. Thereafter, the IMC-Structure design method [Garcia et al, 1982], which is a special case of the above Q(Youla)-Parametrization [Morari, 1987] is presented. Finally, the Characteristic Loci Design Method [MacFarlane et al, 1979,1982] is discussed. These methods are then combined to form a proposed technique for controller design, which is described in chapter 3. Each method is followed by general comments.

2.1 The Q(Youla)-Parametrization Method

The design of control systems using Q-parametrization is a fairly new method. However, the parametrization itself has been used in Optimal control in the fifties [Maciejowski, 1989]. The parametrization has significance in the theory and design of LTI feedback systems. It was reintroduced in the seventies by researchers such as Youla et al. [Youla, 1976]. The name was taken from Youla Dante C. The latter applied the parametrization to quadratic optimization and Zames applied it to H_∞ problems [Maciejowski, 1989]. In using model reference transformations, Zames [Zames, 1981] proposed a simple parametrization for both closed-loop stability and output-disturbance sensitivity, if the plant is O/L stable [Desoer et al, 1981].

The Parametrization is a method based on the frequency domain. The approaches to the original Parametrization use mainly, the state-space method for coprime matrix-fractional representations and the I/O rational transfer function matrix approach.

Firstly, the State-Space approach with coprime matrix fractional representations relations are presented since Q-Parametrization was first presented in Coprime matrix fractional representations. Thereafter, the more direct Rational Transfer Function Matrix relations will be discussed, which will be used in the rest of this thesis.

2.1.1 Coprime MFDs and Fractional Representations

This section assumes a positive feedback convention as in figure 2.1 below

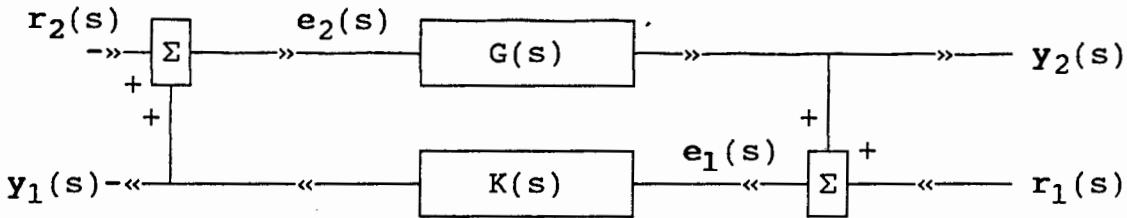


Figure 2.1: Positive feedback Closed-Loop Configuration.

The background and theory of fractional representation is briefly introduced after which the formulations of the theorems leading to the parametrization of all stabilizing controllers are presented.

The Axiomatic Fractional Representation approach to feedback system design was first presented in [Desoer et al, 1980]. The use of the theory of fractional representation for MIMO systems was presented in [Rosenbrock, 1970], when it was applied to polynomial matrix fractions, also called Matrix-Fraction Descriptions(MFD).

Fractional representation theory and the appropriate coprimeness concept are almost always identified together, since the main aspect of the fractional representation theory is the algebraic coprimeness concept [Desoer et al,

1980]. A useful representation of coprimeness as given by the Bezout's theorem is stated below [Maciejowski, 1989]:

Theorem 2.1.1:(Bezout's theorem)

U and V are right coprime if and only if there exist stable X and Y such that

$$XU + YV = I \quad \dots 2-1$$

where:-

I is a square identity matrix

U, V are stable transfer matrices with U and V right coprime which can be written in the form $G(s) = U(s)V^{-1}(s)$

Left Coprimeness can also be defined in a corresponding manner.

Right coprimeness is can also be defined as:-

Let U and V have a common stable right factor such that:

$$U = WX \quad \text{and} \quad V = ZX \quad \dots 2-2$$

then

X^{-1} is stable (or X , has all its zeroes in the LHP)

similarly for Left Coprimeness.

In [Nett et al, 1984], a connection between State-Space and Coprime Fractional Representations is presented. Coprime fractional representations of the transfer function of LTI

systems is given explicitly in terms of observable and controllable minimal state-space realizations of the transfer function.

Coprime fractional representations are by no means unique. However, an example of a procedure for obtaining a right coprime factorization for G , which has a controllable and stabilizable minimal realization $G(A, B, C, D)$, is given below [Maciejowski, 1989], where:-

$$G(s) = C(sI - A)^{-1}B + D \quad 2-3$$

$$u(s) = Kx \quad 2-4$$

$u(s)$ - (process/plant) input vector of G

$x(s)$ - state vector of G

$y(s)$ - output vector of G

K - any matrix such that $(A + BK)$ is asymptotically stable.

$$\text{Define } v = u - Kx \quad \dots 2-5$$

$$\text{Then } \dot{x} = (A + BK)x + Bv \quad \dots 2-6$$

$$u = Kx + v \quad \dots 2-7$$

$$y = (C + DK)x + Dv \quad \dots 2-8$$

which gives

$$u(s) = \{ K[sI - (A + BK)]^{-1} + I \} v(s) = M(s)v(s)$$

$$\begin{aligned} y(s) &= \{ (C + DK)(sI - A - BK)^{-1}B + D \} v(s) \\ &= N(s)v(s) \end{aligned} \quad \dots 2-9$$

and M and N stable so that

$$y(s) = N(s)M^{-1}(s)u(s) = G(s)u(s) \quad \dots 2-10$$

A similar alternative procedure is given in [Maciejowski, 1989] for left coprime fractional representation.

In view of the above coprime fractional representation theory the following theorems which related it to the parametrization of controllers are stated. The proofs are given in [Youla et al, 1976], [Desoer et al, 1980], [Maciejowski, 1989], [Chang et al, 1990], [Moore et al, 1990]. The argument 's' of matrices and vectors was omitted.

Theorem 2.1.2:

Let G and K have fractional representations,

$$G = NM^{-1} = \underline{M}^{-1}\underline{N} \quad (\text{right and left coprime respectively})$$

and

$$K = UV^{-1} = \underline{V}^{-1}\underline{U} \quad (\text{right and left coprime respectively})$$

..2-11

Then the feedback loop is stable if and only if both

$$\begin{bmatrix} \underline{V} & -\underline{U} \\ -\underline{N} & \underline{M} \end{bmatrix}^{-1} \quad \text{and} \quad \begin{bmatrix} \underline{M} & \underline{U} \\ \underline{N} & \underline{V} \end{bmatrix}^{-1} \quad \text{..2-12}$$

are stable.

Now the following theorem which was originally formulated in [Youla et al, 1976] is given below [Maciejowski, 1989]. The theorem was modified and generalized in [Desoer et al, 1980].

Theorem 2.1.3: (Youla's controller parametrization).

Let G and K have fractional representations,

$$G = NM^{-1} = \underline{M}^{-1}\underline{N} \quad \text{and} \quad K = UV^{-1} = \underline{V}^{-1}\underline{U} \quad \dots 2-13$$

If the feedback loop is stabilizable, then

$$M, N, U, V, \underline{M}, \underline{N}, \underline{U}, \underline{V}$$

(all matrices are of compatible dimensions) can be chosen such that

$$\begin{bmatrix} \underline{V} & -\underline{U} \\ -\underline{N} & \underline{M} \end{bmatrix} \begin{bmatrix} M & U \\ N & V \end{bmatrix} = \begin{bmatrix} I & 0 \\ 0 & I \end{bmatrix} \quad \dots 2-14$$

and $K = UV^{-1} = \underline{V}^{-1}\underline{U}$ is a stabilizing controller

Theorem 2.1.4:

Let $K_0 = RS^{-1} = \underline{S}^{-1}\underline{R}$, such that

$$\begin{bmatrix} \underline{S} & -\underline{R} \\ -\underline{N} & \underline{M} \end{bmatrix} \begin{bmatrix} M & R \\ N & S \end{bmatrix} = \begin{bmatrix} I & 0 \\ 0 & I \end{bmatrix} \quad \dots 2-15$$

holds. Let

$$U = R + MQ, \quad V = S + NQ \quad \dots 2-16$$

$$\underline{U} = \underline{R} + \underline{Q}\underline{M}, \quad \underline{V} = \underline{S} + \underline{Q}\underline{N} \quad \dots 2-17$$

where:-

Q - is any realizable matrix of compatible dimensions

Then

1) $K = UV^{-1} = \underline{V}^{-1}\underline{U}$ is a stabilizing controller of

$$G = NM^{-1} = \underline{M}^{-1}\underline{N} \quad \text{and}$$

2) Fractional representations given by equations 2-16 and 2-17 above hold for any stabilizing controller.

From the above theorem, it follows that

$$K = UV^{-1} = (R + MQ)(S + NQ)^{-1} \quad \text{..2-18}$$

which simplifies to the expression, using the matrix inversion lemma(MIL):

$$K = K_0 + \underline{S}^{-1}Q(I + SNQ)^{-1}\underline{S}^{-1} \quad \text{..2-19}$$

$$[\text{MIL: } (A + BCD)^{-1} = A^{-1} - A^{-1}B(DA^{-1}B + C^{-1})DA^{-1}] \quad \text{..2-20}$$

If G is stable and we choose

$$N = G$$

$$M = I$$

$$K_0 = 0 \quad (\text{for example, } R = \underline{R} = 0 \text{ and } S = \underline{S} = I)$$

then the equation given by (c) becomes:-

$$K = Q(I + GQ)^{-1} \quad \text{..2-21}$$

by re-arranging, it gives the expression:-

$$Q = K(I - GK)^{-1} \quad \text{..2-22}$$

which is the 'Youla' or Q-Parameter if the sign change is made to return to negative feedback.

From using equation 2-21 above for the classical unity feedback configuration the structure in figure 2.2 below can be obtained.

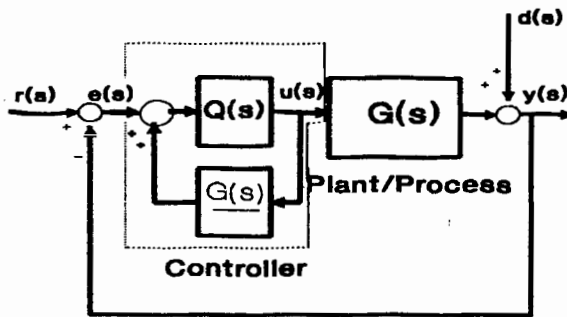


Figure 2.2: Controller structure for a stable plant

2.2.1 Relevant TFM Parametrization Relationships

Consider the classical feedback configuration given below in figure 2.3.

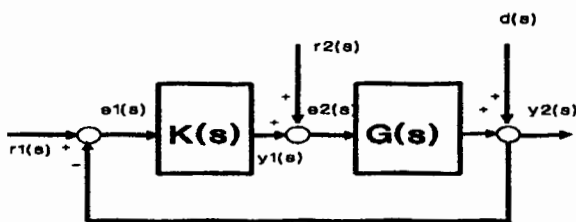


Figure 2.3: Classical feedback configuration $L(G, K)$.

All variables and matrices are functions of the Laplace variable, 's' (= $j2\pi f$), which has been omitted from the relations for brevity.

The closed-loop I/O transfer matrix that maps the vector r to the output vector y , for the $L(G, K)$ system shown above is given by

$$H_{yr} = \begin{bmatrix} H_{y1r1} & H_{y1r2} \\ H_{y2r1} & H_{y2r2} \end{bmatrix} \quad \dots 2-23$$

$$= \begin{bmatrix} K(I + GK)^{-1} & -KG(I + KG)^{-1} \\ GK(I + GK)^{-1} & G(I + KG)^{-1} \end{bmatrix} \quad \dots 2-24$$

Similarly the transfer matrix that maps r to e is given by :

$$H_{er} = \begin{bmatrix} H_{e1r1} & H_{e1r2} \\ H_{e2r1} & H_{e2r2} \end{bmatrix} \quad \dots 2-25$$

$$= \begin{bmatrix} (I + GK)^{-1} & -G(I + KG)^{-1} \\ K(I + GK)^{-1} & (I + KG)^{-1} \end{bmatrix} \quad \dots 2-26$$

where:-

$$r = \begin{bmatrix} r_1 \\ r_2 \end{bmatrix}$$

$$y = \begin{bmatrix} y_1 \\ y_2 \end{bmatrix}$$

$$e = \begin{bmatrix} e_1 \\ e_2 \end{bmatrix}$$

r, e, y are an $(2n \times 1)$ vectors for in n -input n -output process or plant and the transfer matrices

d is the $(n \times 1)$ output disturbance

H_{yr}, H_{er} are $(2n \times 2n)$ matrices for G, Q and K $n \times n$

Following an important observation a TFM $Q(s)$ was defined as

$$Q = K(I + GK)^{-1} \quad \dots 2-27$$

which can be shown to be the same as

$$Q = (I + KG)^{-1}K \quad \dots 2-28$$

If K is made the subject of the relations above, we see that the controller TFM is given by

$$K = (I - QG)^{-1}Q \quad \dots 2-29$$

where:-

Q , is the *Youla parameter* that ranges over all proper, stable transfer functions.

Now using this definition, it can be shown that the Rational Transfer Matrices, H_{yr} and H_{er} are given by

$$H_{yr} = \begin{bmatrix} Q & -QG \\ GQ & G(I - QG) \end{bmatrix} \quad \dots 2-30$$

and

$$H_{er} = \begin{bmatrix} (I - GQ) & -G(I - QG) \\ Q & (I - QG) \end{bmatrix} \quad .2-31$$

For the closed-loop system to be internally stable, matrix elements of the Transfer Matrices, H_{yr} and H_{er} , must be stable. It can be noted in the last two Transfer Matrices that Q appears in a simple manner of matrix sums and products. Subsequently, if we assume that G is stable, then it is true that the closed-loop is stable if and only if Q is stable. It is also observable from H_{y1r1} that K will be realizable only if Q is realizable.

Other observations are that:-

- * The input r_1 to output y_2 map of the system $L(G, K)$ has the simple form

$$H_{y2r1} = GQ \quad ..2-32$$

also called the *Complementary sensitivity matrix* or *transmittance*.

- * The map from the output disturbance d to the output y_2 is given by

$$H_{y2d} = I - GQ \quad ..2-33$$

also called the *Sensitivity Matrix*.

Other results on Input-Output Transfer Matrix Maps and algebraic manipulations in matrix rings are presented in [Desoer et al, 1981].

2.1.3 Comments on the Q(Youla)-Parametrization

The Q(Youla)-parametrization provides us with a set of all stabilizing controller which can be parameterized in terms of given matrices [Youla et al, 1976]. It has been shown to be a useful approach despite the complexity of the formulae. Simplifying parametrization formulae for C/L stability that apply to O/L stable systems, have been presented by Zames [Zames, 1981]. This has simplified the C/L transfer matrix, H_{y2u1} , from $GK(I + GK)^{-1}$ to GQ .

The next design method, the IMC-Structure Design Method [Garcia et al, 1982, 1985], is a special case of the Q(Youla)-Parametrization of all stabilizing controllers.

2.2 The IMC Design Method

The IMC-Structure Design Method for controller design is an approach which was first presented in [Garcia et al, 1982, 1] for the SISO case. The extension to MIMO systems was presented in [Garcia et al, 1985, 2]. It has also been covered extensively in [Garcia et al, 1985, 3], [Rivera et al, 1985], [Economou et al, 1986, 1], [Economou et al, 1986, 2] and [Zafiriu, 1989]. It is based on an open-loop design approach for closed-loop systems provided the process is open-loop stable.

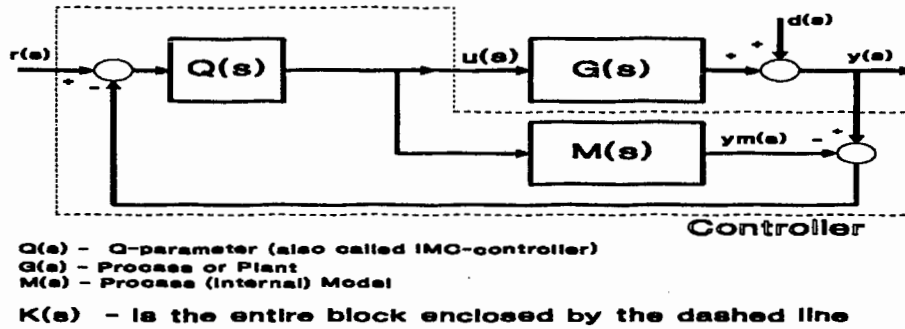


Figure 2.4: The basic IMC-Structure

In the IMC and related referenced literature the 'IMC-controller', $G_c(s)$, is used. In this thesis it will be referred to as 'Q(Youla)-Parameter', $Q(s)$, in view of the fact that the IMC-Structure Design Approach is a special case of the Q(Youla)-Parametrization as presented by Youla [Morari, 1987].

2.2.1 The IMC-Structure

The IMC-Structure is an alternative representation of the classical unity feedback structure for an open-loop stable plant or process. They are related through the equations

$$K = (I - QM)^{-1}Q \quad \text{..2-34}$$

$$Q = K(I + MK)^{-1} \quad \text{..2-35}$$

which are exactly the relations for the Q(Youla)-Parametrization given above if

$$G(\text{process/plant}) = M(\text{model}). \quad \text{..2-36}$$

The closed-loop structures given below show how the classical unity feedback scheme and the IMC-Structure relate to each other.

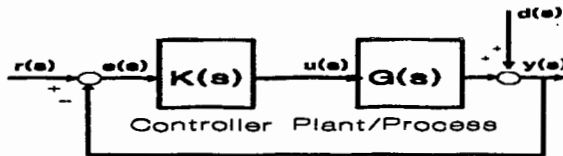


Figure 2.5: The Classical unity feedback structure

If the controller parametrization is applied to K as

$$K = (I - QM)^{-1}Q \quad \dots 2-37$$

then the following control structure is obtained

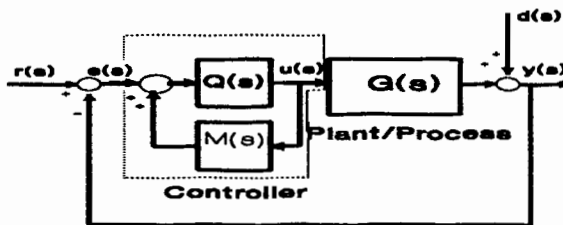


Figure 2.6: Classical structure with $K(Q,M)$

After re-arrangement using block-diagram algebra, the IMC-Structure results as given in figure 2.4 above. The

advantages of using the IMC-Structure are [Garcia et al, 1982, 1]

- * The Q-parameter is much easier to design than $K(s)$.
- * The IMC-Structure allows the inclusion of robustness as a design objective explicitly.

2.2.2 The IMC condition(s) for Internal and C/L Stability

Consider figure 2.7 given below. Firstly, consider the criteria for closed-loop stability. Using the Transfer Function Matrices

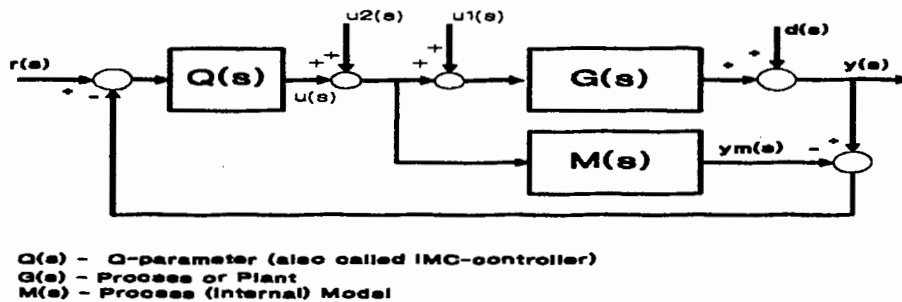


Figure 2.7: Block diagram for derivation of IS conditions

in the figure above the following IMC property is stated [Garcia, 1985, 2].

Property 1: Dual Stability Criterion.

If

$$G(s) = M(s)$$

..2-38

then the stability of both $G(s)$ and $Q(s)$ ensure overall closed-loop stability.

Proof: Consider the following input and output TFM relations. The arguments of the matrices and vectors are omitted.

$$u = \{I + Q(G - M)\}^{-1}Q(r - d) \quad \text{..2-39}$$

and

$$y = d + G\{I + Q(G - M)\}^{-1}Q(r - d) \quad \text{..2-40}$$

which under the assumption that $G = M$ become

$$u = Q(r - d) \quad \text{..2-41}$$

and

$$y = d + GQ(r - d) \quad \text{..2-42}$$

Therefore the system, G and Q -parameter poles must be in the LHP of the 's'-plane for C/L stability. **QED**

Now, in order to ensure internal stability the elements of the following transfer matrix that relate all possible system inputs and outputs must be stable [Zafiriou, 1989].

$$\begin{bmatrix} y \\ u \\ y_m \end{bmatrix} = \begin{bmatrix} GQ & (I - GQ)G & G \\ Q & -QG & 0 \\ GQ & -GQG & G \end{bmatrix} \begin{bmatrix} r \\ u_1 \\ u_2 \end{bmatrix} \quad \text{..2-43}$$

where:-

y, y_m, u, r, u_1, u_2 are output and input vectors
of compatible dimensions respectively
 Q is the Q (Youla)-Parameter

G is the Plant/Process

Theorem 2.2.1:

Assume that $G = M$. Then the IMC system given in figure 2.7 above is internally stable if and only if both Q and G are stable. [Zafiriou, 1989].

The theorem follows from inspections similar to those given in property 1 above.

The similarity between the IMC-Structure and the Classical-Structure is easily observable by also noting that the $2n \times 2n$ minor of the IMC internal stability matrix in equation 2-43, which is

$$\begin{bmatrix} GQ & (I - GQ)G \\ Q & -QG \end{bmatrix} \quad \dots 2-44$$

is precisely the matrix H_{yr} of equation 2-30 given in section 2.2.1 above if we interchange rows and the corresponding inputs and outputs.

2.2.3 The IMC-Structure Design Method

The IMC-Structure Design method assumes that the following conditions are satisfied [Garcia et al, 1982, 1]:

- * that the system, $G(s)$ is open-loop stable and
- * that the Model, $M(s) = G(s)$

Two more properties of the IMC-Structure Design Method as presented in [Garcia et al, 1985, 2] are given below.

An observation of a property of *theoretical interest*, which followed by inspection from the IMC-structure is given below.

Property 2: (Perfect Controller).

Assume that the parameter

$$Q(s) = M(s)^{-1} \quad \text{..2-45}$$

gives a C/L stable IMC-structure. Then, perfect step setpoint tracking is achieved despite any disturbances, $d(s)$ and M-G (model-plant) mismatch.

However this parameter obtained in the above manner is not generally realizable. Therefore, it is augmented with a detuning [Zafiriu et al, 1989] diagonal lowpass filter of the form

$$F(s) = \text{diag}\{ f_{11}(s), \dots, f_{nn}(s) \} \quad \text{..2-46}$$

where:-

$$f_{ii}(s) = (\beta_{ii}s + 1)^{-k} \quad \text{..2-47}$$

and

β_{ii} is a tuning parameter. (all $\beta_i > 0$)

$i = 1, 2, \dots, n$

k is chosen to make the resulting Q-parameter at least proper.

The resulting Q is of the form

$$Q(s) = M(s)^{-1}F(s) \quad \text{..2-48}$$

Thereby allowing for satisfaction of both increased robust stability and robust performance [Zafiriou et al, 1989]. The latter reference also discusses in detail the filter design and proposes a minimization procedure for the robust performance condition as a function of the filter. A discrete-time theorem as given and proven in [Garcia et al, 1985] is stated(proof omitted) below because of its relevance to robust filter design in the presence of plant-model mismatch.

Theorem 2.2.3: Assume that the diagonal TFM filter $F_h(z)$ is of the type [Garcia, 1985]

$$F_h(z) = \text{diag}\{ (1 - \alpha_i)(1 - \alpha_i z^{-1})^{-1} \} \quad \dots 2-49$$

for

$$\alpha_i \text{ in the interval } 0 \leq \alpha_i < 1$$

$$i = 1, 2, \dots, n$$

and that $Q_h^*(z)$ satisfy all the conditions as given in [Garcia, 1985], such that $Q_h(z) = Q_h^*(z)F_h(z)$ then there exists a α^* such that the system is C/L stable for all α_i in the interval $\alpha^* \leq \alpha_i < 1$ ($i = 1, \dots, n$) if and only if $G_h(z)$ and $M_h(z)$ satisfy

$$\text{Re}\{ \text{e-value}_j[G_h(1)M_h(1)^{-1}] \} > 0 \quad j = 1, \dots, n \quad \dots 2-50$$

where:-

$\text{e-value}_j[A]$ = the j th eigenvalue of A

$Q_h(z)$ = the Z-transform of $Q(s)h(s)$ and

$$h(s) = s^{-1}(1 - e^{-sT}) \text{ is a (Sample \& Hold)}$$

$$T = \text{sampling time}$$

Reasons for why the property 2 above is only of theoretical interest are because:[Garcia et al, 1982, 1985]

- * the resulting Q can be unrealizable
- * of nonminimal phase systems(NMP), for example can contain time delays and right half plane transmission zeroes of the system.

Property 3: (Zero Offset). Any parameter, $Q(s)$ such that

$$Q(j0) = M(j0)^{-1} \quad \dots 2-51$$

and the C/L is stable, will yield setpoint tracking for a step input. This also implies that asymptotic properties are obtained almost by default if the Q -parameter is the steady state inverse of the model [Garcia et al, 1982,1985].

Property 3, which is presented above is extended to higher 'system types' in [Zafiriou et al, 1989] to ensure asymptotic properties for closed loop responses. The definition is given below.

Definition 2.2 1: (System Types).

Let $G(s)$ be an $n \times n$ open-loop TFM and let m be the largest integer for which

$$\text{rank}\{ \lim_{s \rightarrow 0} s^m G(s) \} = n \quad \dots 2-52$$

Then the system $G(s)$ is said to be of type m .

For a C/L system to track setpoints of the form

$$r(s) = R/s^m \quad \text{..2-53}$$

it can be shown that the final error value in the IMC-Structure will be zero if the following criteria are satisfied simultaneously [Zafirou et al, 1989]:

$$\lim_{s \rightarrow 0} G(s)Q(s) = I \quad \text{..2-54}$$

and

$$\lim_{s \rightarrow 0} s[e(s)] = 0 \quad \text{..2-55}$$

or

$$\lim_{s \rightarrow 0} s^{-(m-1)}[I - GQ] = 0 \quad \text{..2-56}$$

or using L'Hospitals's Rule, the above equation becomes:

$$\lim_{s \rightarrow 0} \frac{d^{(m-1)}}{ds^{(m-1)}} [I - GQ] = 0 \quad \text{..2-57}$$

where, m is the resulting system type number.

2.2.4 Comments on the IMC-Structure Design Method

Assuming that a given system is open-loop stable and modelled exactly, the IMC-Structure is basically open-loop and stable. With this O/L stable structure, feedback is only used under conditions of model uncertainties. In addition, the IMC-Structure guarantees stability under actuator failure in the presence of reset(integral)-action thereby eliminating the possibility of anti-reset wind up [Zafirou,

1989]. The IMC-Structure Design approach provides the designer with direct intuitive feel for robust stability and controller performance through on-line tuning parameters, α .

The IMC-Structure therefore provides a useful, practically oriented and simplifying approach to control system design.

2.3 The Characteristic Loci Design Method

Characteristic Loci(CL)-Method is a multivariable frequency domain controller design technique. It was introduced by MacFarlane et al [MacFarlane, 1973] and developed further by Kouvaritakis [Kouvaritakis, 1979].

The method utilizes the theory of linear spaces from linear algebra. The vector spaces are defined over base fields which are functions of a complex variable. The CL-method provides amongst features, a means of assessing system stability and interaction of a MIMO feedback system. The topic of interest, is that of interaction in a multivariable system.

This section gives an overview of the CL method and discusses fundamentals of the theory that will be used in the control system design based on Q(Youla)-Parametrization which was discussed in chapter two. In the following sections the CL-method approach for minimizing the interaction in a multivariable system is presented.

2.3.1 Interaction Minimization

A general problem in multivariable feedback control systems is the design of a controller, $K(s)$, such that the characteristic loci space (or eigen-space) of the matrix $G(s)K(s)$ have certain desired properties. One of this properties is:-

The balancing of the characteristic loci gains at low frequencies and alignment of the characteristic directions at high frequencies in order to minimize interaction which is the same for both the open-loop and closed-loop system.

Presented in the following section is one CL-method of controller design to obtain the desired properties of accuracy, integrity and minimizing interaction.

2.3.2 Matrix PI-Controller

Consider a $n \times n$ square plant, $G(s)$, and controller, $K(s)$, which will be connected in a unity feedback control loop $L(G, K)$ as shown in figure 2.3. Then the Matrix PI-Controller [Fisher, 1988] based on the characteristic loci design method is given by:-

$$K(s) = K_{\infty}D_{\infty} + K_0D_0/s \quad \dots 2-58$$

where $K_0 D_0$ and $K_\infty D_\infty$ are $n \times n$ non-singular matrices of constants such that $K(s)$ is the multivariable equivalent of the scalar PI controller. Matrices, D_∞ and D_0 are scaling matrices, that reduce controller output to achieve acceptable performance.

The controller is designed using the information in the plant TFM as follows [Fisher, 1988]:-

Firstly, K_0 diagonalizes the plant at zero frequency, that is

$$K_0 = G^{-1}(j0) \quad \dots 2-59$$

which gives the desired design effects of:

Aligning the plant characteristic vector directions with standard basis at zero frequency, $s = j0$. This ensures that there is no steady state interaction in the O/L and C/L system.

Secondly, to find K_∞ the procedure can be given as follows:-

- a) Multiply row i of $G(s)$ by s^{p_i} where the integer p_i is chosen such that no element of s^{p_i} {row i of $G(s)$ } tends to infinity or do all tend to zero as $s (= j\omega)$ tends to $j\infty$.

Define the row vector, b^i , as:

$$b_i = \lim_{s \rightarrow \infty} s^{p_i} \{ g_i(s) \} \quad \dots 2-60$$

where:-

$$i = 1, 2, 3 \dots n$$

g_i = is a row vector of $G(s)$

b) Repeat the step a) for all the remaining rows of $G(s)$.

c) Combine the row vectors, b_i , in a matrix, B .

Then the matrix K_∞ is given by

$$K_\infty = B^{-1} \quad \dots 2-61$$

The effect of the matrix K_∞ is to ensure that the off diagonal elements of the matrix, $G(s)K_\infty$ become very small relative to the diagonal elements as $s(= j\omega)$ tends to $j\infty$.

2.3.4 Comments on the Characteristic Loci Design Method

The above discussion dealt mainly with the matrix PI-method of the Characteristic Loci-method.

The CL-method feature of minimizing interaction in the frequency domain over a linear vector space can be useful in the design of the Q-Parameter.

The PI-matrix controller design approach offers useful properties of steady state and high frequency reduction of interaction, since it ensures that the matrix, $G(s)K(s)$, approaches diagonal forms as $s(= j\omega)$ tends to either $j\infty$ or $j0$. This implies that $K(s)$ is an approximate inverse of $G(s)$ as s tends to $j\infty$ and to $j0$. This property, as discussed in

the next chapter is exploited in designing the controller based on the Q (Youla)-Parametrization.

Chapter Three

Proposed Simplifying Control System Design Approach based on Q(Youla)-Parametrization

The concept of using a simple open-loop method to design closed loop systems [Newton, 1957] is very appealing to the practicing control engineer. Many important practical aspects of the control scheme become visible during the design exercise itself and do not require additional post-design checking via digital simulation or on-line experimentation. For example the control action needed to give a particular closed loop performance as well as the performance itself are both easily observed while the closed loop transfer function is being designed.

Originating from the inherently open-loop approach which was described by Newton et al, a method which removes the restriction that requires the plant to be O/L stable [Newton, 1957] was proposed by Youla et al [Youla et al, 1976]. The inherently open-loop approach has been incorporated into the Internal Model Control(IMC)-Structure Design Approach [Zafiriou, 1989], which is similar in the sense of the open-loop approach to that described by Newton et al. The IMC-Structure Design approach is seen a special case of the Q(Youla)-Parametrization presented in [Youla et

al, 1976] and [Desoer et al, 1980] which utilizes the fractional representations of rational transfer functions or matrices.

The IMC-Structure represents an explicitly open-loop format and hence simplifies the design of the Q-parameter utilizing O/L information. In IMC-Structure systems, asymptotic tracking of setpoints in regulation problems is accomplished almost by default using the standard internal model control concept, provided the designer specifies that the process output must equal the reference input. However asymptotic tracking of more complex setpoints like ramp (or parabolic) functions leads to a more difficult problem in that a whole set of criteria need to be satisfied to ensure that the final system has the correct Type Number. These conditions are based on derivatives of the transfer function $[I-G(s)Q(s)]$ which relates the error to the reference input [Zafiriou, 1989].

This chapter presents the problem of designing a Q-Parameterized controller and proposes a simplifying design approach for $Q(s)$, which will then be implemented in the IMC-Structure. This approach will be evaluated by firstly applying it to a SISO servo-motor system in which a step and triangular setpoint functions have to be tracked with zero error, see Appendix C. In traditional terminology, 'Type One' and 'Type Two' Systems have to be designed for the

servo-motor control system. Matrix formula, that do not require differentiation, for ensuring adequate Type Numbers when designing $Q(s)$ are also developed. This approach is then applied to an experimental MIMO flotation process rig(FLOTRIG) [Fisher, 1988] in order to evaluate its performance on a real-world experimental system.

3.1. Simplified Design Approach(SISO-case)

The proposition below in 3.1.1 is for a SISO system. The conditions under which it holds are given, and are followed by a verification based on a set of matrix equations using the Q(Youla)-Parametrization for $k(s) = q(s)[1 - g(s)q(s)]^{-1}$ [Youla et al, 1976], [Maciejowski, 1989]

3.1.1 Matrix Conditions for System Type Numbers

For an open loop stable SISO system the following proposition is made.

Proposition 3.1:

Consider the open-loop stable system given by

$$g(s) = \frac{N_0 + N_1s + N_2s^2 + \dots + N_ms^\sigma}{1 + D_1s + D_2s^2 + \dots + D_\mu s^\mu} \quad \dots 3-1$$

$$= N(s) / D(s) \quad \dots 3-2$$

where:- $N(s)$, $D(s)$ are real coeff. polynomials in s , the Laplace variable in units of [radians/second]

such that

$$[\deg\{N(s)\} = \sigma] \leq [\deg\{D(s)\} = \mu]$$

and the Q-parameter

$$q(s) = \frac{n_0 + n_1s + n_2s^2 + \dots}{1 + d_1s + d_2s^2 + \dots} \quad \dots 3-3$$

$$= n(s) / d_q(s) \quad \dots 3-4$$

similarly

$n(s)$, $d_q(s)$ are real coeff. polynomials in s ,

such that

$$[\deg\{n(s)\}] \leq [\deg\{d_q(s)\}]$$

to be implemented in the IMC-Structure [Garcia, 1982]. Then assuming that the model, $m(s) = g(s)$, the C/L system is given by

$$h(s) = g(s)q(s) \quad \dots 3-5$$

Then the following matrix condition for the numerator polynomial of $q(s)$, chosen such that $\deg\{n(s)\} \leq \deg\{d_q(s)\}$, simplifies the setpoint tracking conditions for setpoints of the form

$$r(s) = R/s^m \quad \dots 3-6$$

Type No:

$$\begin{bmatrix} n_0 \\ n_1 \\ n_2 \\ \dots \end{bmatrix} = \begin{bmatrix} N_0 & 0 & 0 & : \\ N_1 & N_0 & 0 & : \\ N_2 & N_1 & N_0 & : \\ \dots & \dots & \dots & : \end{bmatrix}^{-1} \begin{bmatrix} 1 & 0 & 0 & : \\ D_1 & 1 & 0 & : \\ D_2 & D_1 & 1 & : \\ \dots & \dots & \dots & : \end{bmatrix} \begin{bmatrix} 1 \\ d_1 \\ d_2 \\ \dots \end{bmatrix} \begin{matrix} \text{Type 1} \\ \text{Type 2} \\ \text{Type 3} \\ \end{matrix}$$

\dots 3-7

which sets the minimum degree allowable for $d_q(s)$. Note also that the matrices are nonsingular.

Equation 3-7 above can be verified in two ways as shown in verification 3.1a and 3.1b below.

Verification 3.1a: Using the Classical Approach.

Let $k(s)$ and $q(s)$ be given by the relations below

$$k(s) = q(s)[1 - g(s)q(s)]^{-1} \quad \dots 3-8$$

and

$$q(s) = k(s)[1 + g(s)k(s)]^{-1} \quad \dots 3-9$$

and

$$g(s) = m(s) \quad \dots 3-10$$

from the Q(Youla)-Parametrization [Maciejowski, 1989]

Together with the rational transfer functions of equations 3-2 and 3-4 an important observation is made to ensure the correct system type number as follows:

Let $k(s)$ be given by (using eqns. 3-2, 3-4 and 3-8). Then

$$k(s) = \left[\frac{n(s)D(s)}{d_q(s)D(s) - n(s)N(s)} \right] \quad \dots 3-11$$

can be written as

$$= \frac{(n_0 + n_1 s + \dots)(1 + D_1 s + \dots + D_n s^n)}{(1 + d_1 s + \dots)(1 + D_1 s + \dots + D_n s^n) - (n_0 + n_1 s + \dots)(N_0 + N_1 s + \dots + N_m s^m)} \quad \dots 3-12$$

For the final controlled system to be a Type m system; the coefficients of the powers of $s^0, s^1, s^2, \dots, s^{m-1}$ of the controller $k(s)$ must all be zero. From Equation 3-11 the denominator polynomial for $k(s)$ is given by

$$\begin{aligned} & (1 - n_0 N_0) + \\ & (d_1 + D_1 - n_0 N_1 - n_1 N_0) s + \\ & (D_2 + d_1 D_1 + d_2 - n_0 N_2 - n_1 N_1 - n_2 N_0) s^2 + \\ & \dots \end{aligned} \quad \dots 3-13$$

In these m constraints for ensuring the correct system Type Number the values of N_i and D_i are known from the system, transfer function, $g(s)$ or model, $m(s)$ while the values of d_i are set by the desired closed loop pole positions.

Thus the constraints of Equation 8 for system Type Number effectively impose conditions only on the numerator polynomial of module $q(s)$.

Now, by setting the required number of coefficients of powers of ' s ' from, ' s^0 ' to ' s^{m-1} ', in Equation 3-13 to zero it follows that, for a particular Type Number, the designer

is required to solve the following matrix equation for the unknown coefficients $n_0, n_1, n_2, \dots$

Type No:

$$\begin{bmatrix} n_0 \\ n_1 \\ n_2 \\ \vdots \end{bmatrix} = \begin{bmatrix} N_0 & 0 & 0 & : \\ N_1 & N_0 & 0 & : \\ N_2 & N_1 & N_0 & : \\ \dots & \dots & \dots & \dots \end{bmatrix}^{-1} \begin{bmatrix} 1 & 0 & 0 & : \\ D_1 & 1 & 0 & : \\ D_2 & D_1 & 1 & : \\ \dots & \dots & \dots & \dots \end{bmatrix} \begin{bmatrix} 1 \\ d_1 \\ d_2 \\ \vdots \end{bmatrix} \begin{matrix} \text{Type 1} \\ \text{Type 2} \\ \text{Type 3} \end{matrix}$$

..3-14

where the number of rows in the matrices and vectors is set by the desired Type Number as indicated, and is equal to the number of columns used in the matrices. To ensure that module $q(s)$ is realizable the order of polynomial $d_q(s)$ must equal or exceed that of polynomial $n(s)$. From Equation 3-14 it is clear that the required system Type Number sets the minimum order allowable for the numerator polynomial of module $q(s)$, which in turn sets the minimum order allowable for its denominator polynomial $d_q(s)$. QED

This is a much simpler condition to implement than the m derivative condition given in [Zafiriou, 1989]. This condition is also given below. It is also shown that, using the parametrization of the controller as

$$k(s) = q(s)[1 - g(s)q(s)]^{-1}$$

which becomes

$$k(s) = q(s)[1 - m(s)q(s)]^{-1}$$

for the IMC Design Method, a special case of the Q-Parametrization and which assumes that $m(s) = g(s)$ the same

result as equation 3-14 is obtained by using the observation made for higher order setpoint tracking.

Verification 3.1b: Using the IMC-conditions.

For tracking setpoints of the form $r(s) = R/s^m$ the following IMC-conditions are given

$$\lim_{s \rightarrow 0} [g(s)q(s)] = 1 \quad \dots 3-15$$

$$\lim_{s \rightarrow 0} [s\{e(s)\}] = 0 \quad \dots 3-16$$

or, in terms of the transfer functions in the IMC loop using L'Hospital's Rule for indeterminate forms. In this case, $0/0$

$$\lim_{s \rightarrow 0} \frac{d^{m-1}}{ds^{m-1}} [1 - g(s)q(s)] = 0 \quad \dots 3-17$$

where constant m is the Type Number for the resulting C/L IMC system.

Using these conditions asymptotic setpoint tracking is also achieved for setpoint of the R/s^m .

Now, by setting the required number of coefficients of powers of 's' in Equation 3-17 to zero after using the

relations in equations 3-2, 3-4 and 3-8 it is can also be shown that, for a particular Type Number, the designer is required to solve the following matrix equation for the unknown coefficients $n_0, n_1, n_2, \dots$. But obtain the derivatives is not trivial.

Type No:

$$\begin{bmatrix} n_0 \\ n_1 \\ n_2 \\ \vdots \end{bmatrix} = \begin{bmatrix} N_0 & 0 & 0 & : \\ N_1 & N_0 & 0 & : \\ N_2 & N_1 & N_0 & : \\ \dots & \dots & \dots & : \end{bmatrix}^{-1} \begin{bmatrix} 1 & 0 & 0 & : \\ D_1 & 1 & 0 & : \\ D_2 & D_1 & 1 & : \\ \dots & \dots & \dots & : \end{bmatrix} \begin{bmatrix} 1 \\ d_1 \\ d_2 \\ \vdots \end{bmatrix} \begin{matrix} \text{Type 1} \\ \text{Type 2} \\ \text{Type 3} \\ \end{matrix}$$

..3-18

which is the same as equation 3-14 above.

3.1.2 Modification to include Dynamic Compensation.

In the Newton/IMC approach, the design of feedback control systems module, $q(s)$, is usually chosen to cancel the undesirable poles of the process $g(s)$, provided of course that these are stable and reasonably damped [Astrom, 1984], and then to substitute in the poles desired for the closed loop system.

This condition on the dynamics of the closed loop system leads to a more restrictive form for numerator polynomial of module $q(s)$ than that used in Equation 2 in that

$$q(s) = \frac{(n_o^* + n_1^* s + \dots)(1 + D_1 s + \dots + D_n s^n)}{1 + d_1 s + d_2 s^2 + \dots}$$

$$= n^*(s) D(s) / d(s) \quad \dots 3-19$$

Once again the polynomials, $n^*(s)$ and $d(s)$ are not explicitly limited to any particular order provided that $q(s)$ is realizable in practice, though it is common to keep them as simple as possible. The designer is free to choose the closed loop pole positions to give the required dynamics while keeping the input within allowable ranges.

It is easy to show that the matrix conditions on polynomial $n(s)$ of equation 13 can be converted to the following matrix conditions on polynomial $n^*(s)$, since

$$n(s) = n^*(s) D(s) \quad \dots 3-20$$

and

$$n^*(s) = D^{-1}(s) n(s) \quad \dots 3-21$$

which leads to the matrix condition given below

Type No:

$$\begin{bmatrix} n_o^* \\ n_1^* \\ n_2^* \\ \dots \end{bmatrix} = \begin{bmatrix} 1 & 0 & 0 & : \\ D_1 & 1 & 0 & : \\ D_2 & D_1 & 1 & : \\ \dots & \dots & \dots & \dots \end{bmatrix}^{-1} \begin{bmatrix} n_o \\ n_1 \\ n_2 \\ \dots \end{bmatrix}$$

Type 1
Type 2
Type 3

$\dots 3-22$

where the dimensions of the matrices and vectors are once again set by the desired Type Number, as indicated. It should be noted that the order of polynomial $d_q(s)$ in module $q(s)$ must now equal or exceed the sum of the orders of polynomial $n^*(s)$ of module $q(s)$ and polynomial $D(s)$ of process model $g(s)$.

3.1.3 Alteration of speed of response.

A less obvious alternative observation is made for first order systems, whereby the response time is decreased by reducing the model output by a factor, say μ and increasing the gain of the $q(s)$ parameter by the same factor, μ . The gain reduction and corresponding gain increase in $q(s)$ ensures asymptotic setpoint tracking with zero offset. For example to double the speed or response of the system given below

$$g(s) = m(s) = \left[\frac{A}{(1 + sT)} \right]$$

and for

$$q(s) = 1/A$$

the response is doubled by making

$$m_{\text{new}}(s) = m(s) / 2 \quad \text{and}$$

$$q_{\text{new}}(s) = 2 * q(s)$$

The above observation can be very useful in changing the dynamics of more complex MIMO systems.

Another experimental approach to improve controller performance is to remove the zero(es) from the model, $m(s)$, but retaining the system poles and steady state gain if satisfactory.

This can be explained in the following quantitative manner: Suppose, $g(s) = m(s)$, and that $g(s)$ has a dominant zero which causes a large overshoot, then $g(s)u(s) = m(s)u(s)$ implies that the feedback signal is zero. Hence the Q-parameter, $q(s)$ does not see the overshoot if no dynamic compensation was made although setpoint tracking is achieved. On the contrary, if $m(s)$ not equal to $g(s)$, say the zero is removed from model, then, during the dynamic period, $\text{magnitude}[y(t)] > \text{magnitude}[y_m(t)]$, hence there is a (feedback signal) > 0 . Thus, $q(s)[r(s) - f/\text{back signal}]$ is less than the original $q(s)[r(s) - 0]$ signal, which implies that the process input is reduced, hence less or no overshoot. This remark can be very useful in MIMO systems since dynamic compensation can be complex in multivariable systems.

The application of the above approach to the design of $q(s)$ for a SISO servo-motor system is described in Appendix C. The experimental results verify that the theory is correct.

3.2 Simplified Design Approach(MIMO-case)

The SISO design approach presented above is extended to the multivariable case for open-loop stable systems. Issues relating to MIMO systems such as for example minimization of interaction and improvement of C/L performance are discussed within the context of the simplified design approach based on Q(Youla)-Parametrization as described in section 3.1 for the SISO-case. The designed parameter, $Q(s)$ is implemented in the IMC-Structure.

3.2.1 Decoupling Controller Design

Decoupling Controller design for linear multivariable plants has been presented in [Lin & Hsieh, 1991] based on the Q(Youla)-Parametrization. The approach is presented for both stable and unstable systems. It requires symbolic inversion of the plant, the computation of the Vandermonde matrix and the computation of the classical controller, $K(s)$. Another procedure for the design of multivariable feedback system with stable plants based on the Q-Parametrization was presented in [Desoer & Chen, 1981]. This approach requires coprime fractional representation of the plant which in turn requires inversion of polynomial matrices that are easier to compute than rational transfer function matrices.

A simplifying design approach for MIMO O/L stable or stabilized systems implemented in the IMC-Structure is presented, based on the Q-parametrization. The proposed

approach yields sufficient decoupling and setpoint tracking without attempting to symbolically invert matrices or cancel poles and zeroes of the system, since exact decoupling is not achievable, except in theory. The approach is given below.

Proposition 3.2: Approximate MIMO system decoupling.

Let the O/L system, $G(s)$ and its model, $M(s)$, be asymptotically stable, such that for C/L stability the IMC condition given below [Garcia, 1985] is satisfied in the case of plant-model mismatch

$$\operatorname{Re}\{\text{eigenvalue}_j[G(0)M(0)^{-1}]\} > 0 \quad \dots 3-23$$

where: $j = 1..n$ for an square $n \times n$ system $G(s)$

$\operatorname{Re}\{\text{eigenvalue}_j[A]\}$ denotes the real part of the j th eigenvalue of square matrix A .

Let a stable $Q(s)$ be given

$$Q(s) = (sQ_\infty + Q_0)F(s) = L(s)F(s) \quad \dots 3-24$$

(Adapted and modified from the Characteristic Loci approach given in [Fisher, 1988] as mentioned in chapter two.)

$$L(s) = sQ_\infty + Q_0 = sB^{-1} + K_0 \quad \dots 3-25$$

for

$$b_i = \lim_{s \rightarrow \infty} s^r g_i(s) \quad \dots 3-26$$

such that

g_i are row vectors of $G(s)$ for $i = 1, \dots, n$

r is an integer chosen such that no $b_i = 0$ as s tends to $j\infty$

$$B = [b_1, \dots, b_i, \dots, b_n]^T \quad \dots 3-27$$

$$K_O = G(j0)^{-1} \quad \dots 3-28$$

$$F(s) = \text{diag}\{(\beta_{jj}s + 1)^{-1}\} \quad \dots 3-29$$

for $j = 1..n$

$\beta_{jj} > 0$, is chosen such that the largest magnitude input-action for the j th column is reduced as required.

then, the resulting C/L system is stable, decoupled exactly at steady state(low frequency) and approximately at high frequency.

Verification 3.2:

Firstly, using the equation IMC closed-loop equation as given in chapter 2, property 1, that is:

$$y(s) = d(s) + G(s)Q(s)[r(s) - d(s)] \quad \dots 3-30$$

and since $M(j0) = G(j0)$..3-31

$y(j0)$ must tend to $r(j0)$

which is only true if

$$Q(j0) = G(j0)^{-1} \text{ and } F(j0) = I \quad \text{..3-32}$$

which are exactly the results of the definitions for Q_0 and $F(s)$ respectively.

QED

Similarly the approximate high frequency decoupling can be shown using the definition for Q_∞ with $G(s)$ as 's' tends to infinity.

Remark 1: $F(s)$ makes the $Q(s)$ proper.

Remark 2: The approach merely uses inverse computation of constant matrices and decouples the system using zeroes.

Remark 3: The constant, β_{jj} , gives the designer the choice of reducing input action as required but can also affect controller performance.

Remark 4: It can be shown by inspection that, the decoupling method given above is sufficient for step setpoint tracking with zero steady state error.

Remark 5: Step setpoint-tracking, interaction minimization, disturbance rejection and noise attenuation can be observed and analyzed satisfactorily by using Bode plots for the matrix $G(s)Q(s)$, since the C/L system is implemented in an explicitly O/L IMC-Structure.

The frequency-domain method that is based on approximate system decoupling simplifies the computation from symbolic inversion of rational transfer function or polynomial matrices to inverting easily obtained nonsingular constant matrices.

3.3 Setpoint Tracking Conditions for the Q-Parameter

Based on the implications of propositions 3.1 and 3.2, the following conditions are added to the Q-parameter for asymptotic tracking of setpoints of the form, $r(s) = R/s^m$.

Proposition 3.3: Consider the MIMO open-loop stable system given by

$$g_{ij}(s) = \frac{N_0 + N_1s + N_2s^2 + \dots + N_{\sigma_{ij}}s^{\sigma_{ij}}}{1 + D_1s + D_2s^2 + \dots + D_{\mu_{ij}}s^{\mu_{ij}}} \quad \dots 3-33$$

$$= N_{ij}(s) / D_{ij}(s) \quad \dots 3-34$$

where:-

$i, j = 1..n$, for a square $n \times n$ system, $G(s)$

$N_{ij}(s)$, $D_{ij}(s)$ are polynomials in s , the Laplace variable in units of [radians/second] such that

$$[\deg\{N_{ij}(s)\} = \sigma_{ij}] \leq [\deg\{D_{ij}(s)\} = \mu_{ij}] \quad \text{..3-35}$$

Let the square $n \times n$ Q-parameter be given by

$$Q(s) = Q_0 P(s) \quad \text{..3-36}$$

where:-

$$i = 1..n$$

$$Q_0 = G(j0)^{-1} \quad \text{..3-37}$$

$$P(s) = \text{diag}\{p_{ii}\} = F(s)R(s) \quad \text{..3-38}$$

$F(s)$ as defined in proposition 3.2 above

$$R(s) = \text{diag}\{r_{ii}(s)\} = \text{diag}\{n_{rii}(s) / d_{rii}(s)\}$$

$$p_{ii}(s) = \frac{n_0 + n_1 s + n_2 s^2 + \dots}{1 + d_1 s + d_2 s^2 + \dots} \quad \text{..3-39}$$

$$= n_{pii}(s) / d_{pii}(s) \quad \text{..3-40}$$

$$\deg\{n_{pii}(s)\} \leq \deg\{d_{pii}(s)\} \quad \text{..3-41}$$

to be implemented in the IMC-Structure [Garcia, 1982]. Then let the model, $M(s) = G^\#(s)$, where $G^\#(s)$ has all its zeroes removed (for each element) and only retaining the poles and steady state gains. Let the condition [Garcia, 1985]

$$\text{Re}\{\text{eigenvalue}_j[G(0)M(0)^{-1}]\} > 0$$

where:-

$j = 1..n$ for an square $n \times n$ system $G(s)$

$\text{Re}\{\text{eigenvalue}_j[A]\}$ denotes the real part of the j th eigenvalue of square matrix A .

still hold to ensure C/L system stability whenever, $Q(s)$ is stable.

Thus, the setpoint tracking conditions can be imposed on stable $p_{ii}(s)$, based on the diagonal elements of $M(s)$ with their steady state gains taken as unity, that is, $m_{ii}(j0) = 1$, then the design steps in proposition 3.1 hold for tracking setpoints of the form R/s^m .

Verification 3.3:

Using the Q-Parametrization of $K(s)$, that is

$$K(s) = Q(s)[I - M(s)Q(s)]^{-1} \quad \text{..3-42}$$

and letting

$$c_{ij}(s) = \text{elements of } \{MQ_0\} \quad \text{..3-43}$$

then it is sufficient to show that

$$\lim_{s \rightarrow 0} K(s) \text{ tends to } \lim_{s \rightarrow 0} \{k_{ij}/s^m\} \quad \text{..3-44}$$

$$i, j = 1..n$$

for the C/L system to track inputs of the form, $r(s) = R/s^m$

as follows:

$$\lim_{s \rightarrow 0} K(s) = \lim_{s \rightarrow 0} Q(s)[I - M(s)Q(s)]^{-1} \quad \text{..3-45}$$

$$= \lim_{s \rightarrow 0} Q_0 P(s)[I - \{M(s)Q_0\}P(s)]^{-1} \quad \text{..3-46}$$

noting¹ that Q_0 diagonalises $M(s)$ as s tends to 0, thus the off-diagonal elements tend to 0, that is

$$\lim_{s \rightarrow 0} M(s)Q_0 \rightarrow \lim_{s \rightarrow 0} \text{diag}[c_{ii}(s)] \quad \dots 3-47$$

such that

$$\lim_{s \rightarrow 0} K(s) \rightarrow \lim_{s \rightarrow 0} Q_0 P(s) [I - \text{diag}[c_{ii}(s)] P(s)]^{-1} \quad \dots 3-48$$

$$\rightarrow \lim_{s \rightarrow 0} Q_0 \cdot \text{diag} \left[\frac{p_{ii}(s)}{1 - c_{ii}(s)p_{ii}(s)} \right] \quad \dots 3-49$$

and since

$$p_{ii}(s) = n_{pii}(s) / d_{pii}(s)$$

and using equations 3-43 and 3-47

$$c_{ii}(s) = \left[\frac{N_{mqii}}{D_{ii}(s)} + \sum_{j < i}^n \frac{N_{mqij}}{D_{ij}(s)} \right] \quad \dots 3-50$$

where:-

D_{ii}, D_{ij} are polynomials of 's'

N_{mqii}, N_{mqij} are constants

The $c_{ii}(s)$ are obtained from diagonal elements of $M(s)Q_0$ hence the N_{mqij} numerator notation for $c_{ii}(s)$ since Q_0 only affects the constant numerators of $M(s)$.

1. Also using the property of limits: $\lim_{s \rightarrow 0} A(s)B(s)C(s)^{-1} =$

$$\lim_{s \rightarrow 0} [A(s)] \lim_{s \rightarrow 0} [B(s)] \lim_{s \rightarrow 0} [C(s)^{-1}] = \lim_{s \rightarrow 0} [A(s)B(s)] \lim_{s \rightarrow 0} [C(s)^{-1}]$$

Thus

$$\lim_{s \rightarrow 0} K(s) \rightarrow \lim_{s \rightarrow 0} Q_o \cdot \text{diag} \left[\frac{n_{pii} D_{ii}}{d_{pii} D_{ii} - (D_{ii} c_{ii}) n_{pii}} \right]$$

..3-51

where:-

$$D_{ii} c_{ii} = \left[N_{mqii} + \sum_{j < i}^n N_{mqij} \begin{bmatrix} D_{ii} \\ D_{ij} \end{bmatrix} \right]$$

thus equation 3-51 simplifies to equation 3-52 given below, after keeping the D_{ii} required for the setpoint tracking condition

$$\lim_{s \rightarrow 0} K(s) \rightarrow \lim_{s \rightarrow 0} Q_o \cdot \text{diag} \left[\frac{n_{pii} D_{ii}}{d_{pii} D_{ii} - (1) n_{pii}} \right]$$

..3-52

since

$$\lim_{s \rightarrow 0} D_{ii}(s) c_{ii}(s) = \lim_{s \rightarrow 0} \left[N_{mqii} + \sum_{j < i}^n N_{mqij} \begin{bmatrix} D_{ii}(s) \\ D_{ij}(s) \end{bmatrix} \right]$$

and using the notation for all $D_{ij}(s)$, which is

$$D_{ij}(s) = 1 + D_1 s + D_2 s^2 + \dots$$

the above equation simplifies to

$$\lim_{s \rightarrow 0} D_{ii}(s) c_{ii}(s) = \left[N_{mqii} + \sum_{j < i}^n N_{mqij}(1) \right] = 1$$

using the definition Q_0 and $M(j0)Q_0 = I$

Finally, note that equations 3-51 and 3-52 above contain a constant matrix multiplied by a diagonal matrix which has the elements with exactly the same form as equation 3-11 in proposition 3.1. The polynomials, $n_{pii}(s)$, were chosen based on the known polynomials, $D_{ii}(s)$ and $d_{pii}(s)$ in order satisfy the required system type number as in proposition 3.1.

Hence, by definition of the conditions on $n_{pii}(s)$

$$\lim_{s \rightarrow 0} K(s) \text{ tends to } \lim_{s \rightarrow 0} (k_{ij}/s^m) \quad QED$$

The Proposed Simplifying Approach based on Q(Youla)-Parametrization provides significant design insight and simplification for the design of SISO and MIMO systems.

The algebraic matrix conditions on the numerator of control module $q(s)$ or $Q(s)$ that ensure asymptotic tracking of setpoints are easier to impose than derivative conditions on the error signal.

The combination of the intuitive insight as to what the approximate desired C/L response should be and the frequency domain properties of the matrices have simplified the design approach of the Q-parameter and at large the design of $K(s)$ using $Q(s)$ and $M(s)$. The frequency domain can also be useful for analyzing setpoint tracking, decoupling, disturbance

rejection and noise attenuation. The time domain considerations were aimed at controller performance, which were then included by removing zeroes from the model and introducing tuning parameters which affect the controller performance in a simplistic manner.

The above proposed method for controller design was evaluated by applying it to the Flotation Rig(FLOTRIG) built by Mr Fisher [Fisher, 1988]. The FLOTRIG description and identification are given in the next chapter.

CHAPTER FOUR

The Flotation Rig Identification

In this chapter the background of the Flotation Process will be presented. This will be followed by a description of how a typical Industrial Flotation Process has been realized as a 'real-world' experimental model in the control laboratory [Fisher, 1988]. The final part of this chapter will mainly be concerned with the identification of the experimental model.

4.1 Description of typical a Flotation Process

Flotation is a widely used mineral process for separation of valuable mineral from the ore [Perry, 1963]. The process is based on the property that some particles adhere more strongly than others to air bubbles formed by an air flow in for example a slurry. A schematic of the flotation circuit is shown in figure 4.1 below.

The flotation process consists of a number of interconnected tanks, called flotation cells. The refining of minerals can be achieved by building up a network of these flotation cells. In this case four interconnected cells. The slurry first enters the circuit at the so-called Rougher cell. An air flow creates the bubbles to which the mineral particles stick in each of the tanks, called flotation cells. The mineral loaded bubbles rise to the surface, forming what is

called the 'froth' or concentrate which overflows into other cells [Koivo, 1977].

NB: Manual valves are indicated differently from the control valves, as can be seen in the figure below.

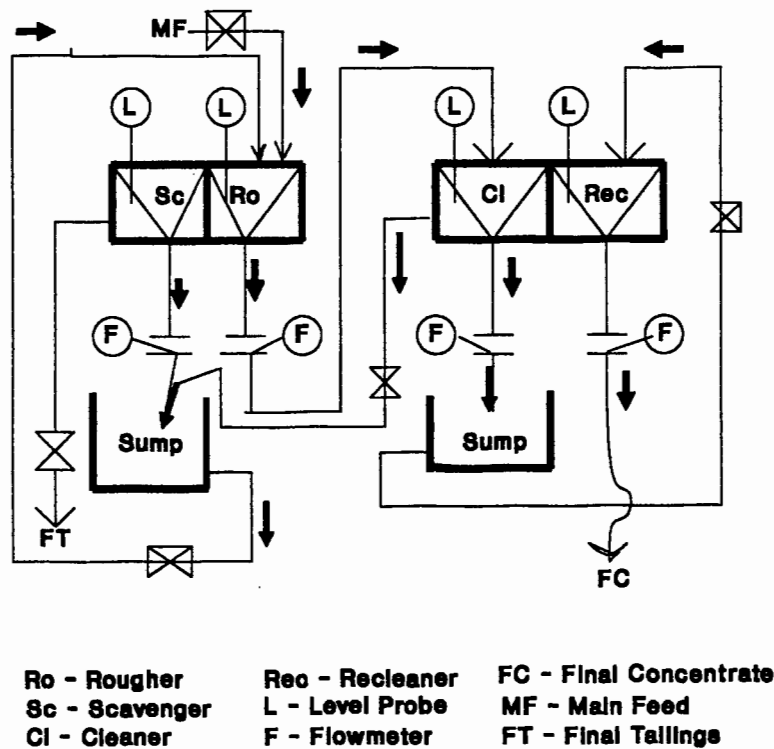


Figure 4.1: Flotation Rig Circuit

The froth overflows of the various cells are linked either directly from one tank into the other or through a sump tank. Similarly, the tailings outputs of each cell are also linked.

4.2 Description of the Flotation Rig

A flotation plant simulator was built and used by Mr Ian Fisher, in a control study [Fisher, 1988]. The flotation plant will be referred to as the FLOTRIG in this thesis.

The FLOTRIG is an experimental simulation of the fluid dynamics of a typical flotation plant used in the mineral extraction industry. It serves as an important test site in the application, evaluation and investigation of various control system design methodologies [Fisher 1988], [Venzke, 1989], [Trossbach, 1992].

The FLOTRIG exhibits various real-world engineering characteristics which are not manifested in pure computer simulation studies. Examples of these problems are nonlinearities, noise, disturbances etc. The design aim in such an environment would be, for example, to produce a LTI control scheme that is insensitive to these factors.

4.3 Flotation Cell Simulation

The FLOTRIG circuit was designed to be similar to a typical industrial flotation plant. The cells that make up the circuit are similar in design, as shown in figure 4.2 below. The product in the circuit is water. Each cell has a froth overflow (also called concentrate output), consisting of an inverted U-shaped piece at a certain height that simulates the overflow of the froth stream. The rates of the tail outflows are controlled by pneumatic valves, so that the slurry level in each cell is controlled by regulating the tail out flow from that cell.

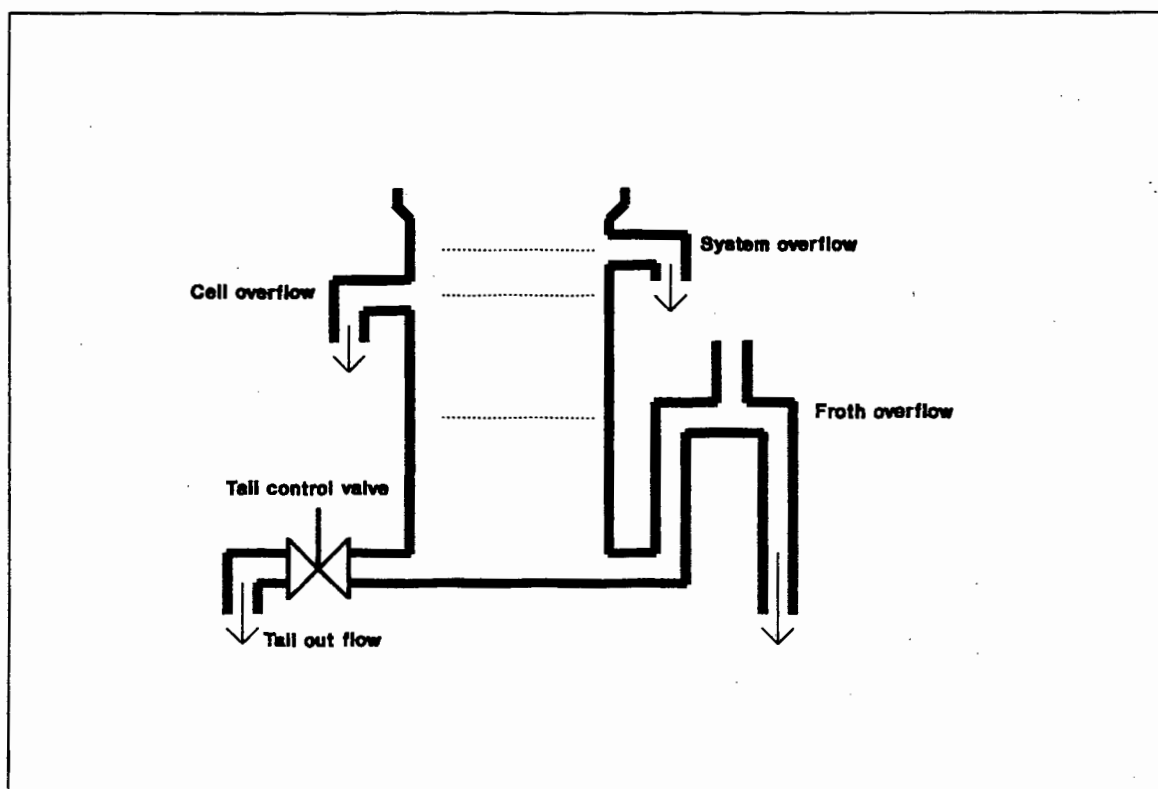


Figure 4.2: Simulated Flotation Plant Cell

4.4 Instrumentation and lay-out of the FLOTIRIG

The instruments on the FLOTIRIG consist of level probes, flow sensors and pneumatic actuators. The signals for the pneumatic actuators range from 4-20 mA whereas the level probe signals range from 0-10 V. These are connected to the PC via ADC/DAC interface cards. The setup of the FLOTIRIG is given in figure 4.3 below.

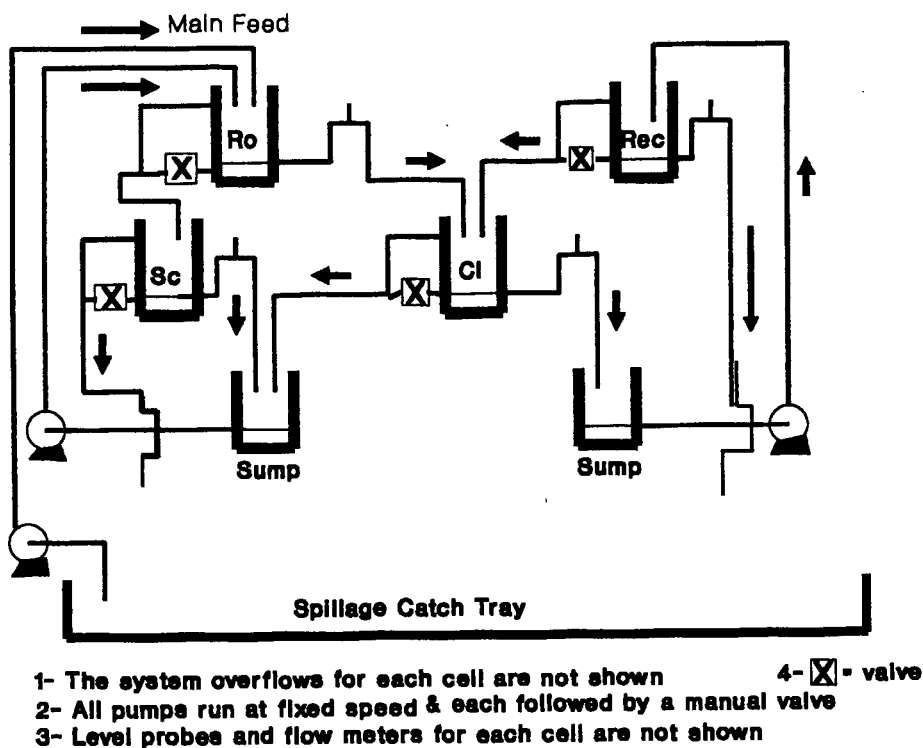


Figure 4.3: Flotation Rig (FLOTIRIG)

In the rest of the sections, the procedures and tests performed to identify the FLOTIRIG model are presented.

4.5 Identification of the FLOTRIG Model

System or Model identification forms an integral part of controller design. Such models are essential for any theoretical analysis or multivariable controller designs, if applicable. In this case a model is derived for the FLOTRIG system. The model takes the form of transfer functions required for the design method used in the thesis.

The following subsections describe the tests performed on the FLOTRIG, analysis of the data and the development of a final model.

4.6 Open Loop Step Tests on FLOTRIG

A model of the FLOTRIG is to be derived from an analysis of step test response data. The process input variables are the four pneumatic actuators and the process outputs are the four flotation cell levels. The Transfer Function Matrix(TFM) to be generated has the form given below. All matrix elements, inputs and outputs are functions of the argument 's'(Laplace domain variable):-

$$y = G u$$

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

where:-

y_1 = Rougher output	u_1 = Rougher input
y_2 = Scavenger output	u_2 = Scavenger input
y_3 = Cleaner output	u_3 = Cleaner input
y_4 = Recleaner output	u_4 = Recleaner input

The input and output units were in ADC/DAC bit counts but were converted to percentages. The output range is from 0% to 100% and the input from -100% to 100%. The ADC/DAC converters are 12-bit computer interface devices used with Data Translation Cards.

The transfer function time constants and delays are in seconds.

4.6.1 Flotation Rig Linear Operating Region

The FLOTTRIG is considered to be in the **Linear Operating Region(LOR)** when the level of each cell is between the froth overflow level and the cell overflow [Fisher, 1988]. The region which is the same for each cell, is shown in the figure 4.4 below.

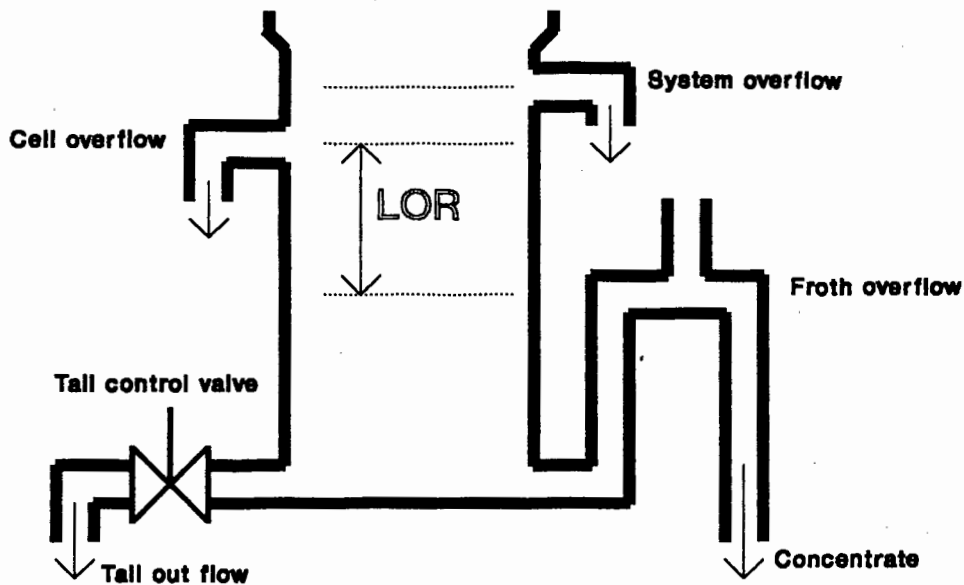


Figure 4.4: Flotation Cell linear operating region

The models obtained for the various cells will therefore only be valid in the linear region. Thus all levels must remain in this region while collecting any open loop data for modeling purposes.

4.6.2 Experimental Open Loop Step Tests

The experimental open loop step tests were done to establish

- i) the correct actuator setting for the linear region
- ii) the approximate time constants of the cell in this region.

The following procedure which was carried out to establish the above:

- a) Systematically closed the tail valve until the level steady state remains in the linear operating region. This must be done for each cell, and repeated, since the system is expected to have interaction.
- b) Wait until all levels are at steady state within the LOR.
- c) Start recording all actuator and level data, since these are the variables of concern.
- d) Step one (e.g. +10%) of the valves, (100% = close, 100% = open) such that the new steady state to be reached still remains in the LOR while continuing to record data.
- e) Once the new steady state has been attained, stop recording data.

Repeat the above procedure for all the cells. The most representative results from the step tests are given in Appendix A.

The following information was gathered from the preliminary step tests:-

- a) The fastest time constant, in order to establish a suitable sampling period. In this case it was about 33 seconds for a sampling time of one second.

- b) The time duration over which the FLOTRIG reaches attains steady state was observed, in order to obtain the length of time for data recording. The minimum time length required is about 540 seconds.
- c) The variables which are in direct relation to the product flow in each cell were noted, in order to determine which variable to used as the input to the controller and as output. In this case, water level in each cell and actuator positions respectively.
- d) The modelling approach for the FLOTRIG components. The rig was model as one lumped element. That is, the relationship from the actuator position to the water level for each cell, in units of DAC/ADC bit counts is modelled as one element (that includes I/P converters, valves , level probes, etc). The bit counts were then converted to percentage of range.
- e) The visible oscillations in the Cleaner and Rougher cells were caused by the Rougher feedback sump pump. That is, the pump drains the sump faster than the water enters the sump. Thus, while the sump level is very low (below the pump feed pipe), the pump cavitates until the sump water level rises to above the pump feed pipe level. The oscillations are very fast and are not considered as part of the fluid dynamics of the FLOTRIG. Therefore they

were modelled as disturbances and were excluded from the transfer function model, $G(s)$ for the FLOTRIG.

- f) The actuator positions that allow the steady state levels of each cell to settle in the LOR were noted. The respective positions obtained from the preliminary tests are:-

(100% = fully closed, -100% = fully open)

Rougher : -50 %

Scavenger : -50 %

Cleaner : -50 %

Recleaner : 0 %

4.6.3 Final Open Loop Step Tests

The same procedure for the preliminary open loop step tests was followed to obtain data for the final test. The step tests were done so that none of the levels in the four cells must enter the non-linear region of the FLOTRIG. The data obtained from the step tests are shown in Appendix A.

4.7 Development of the FLOTRIG model

The data obtained from the final open loop tests were used to develop a TFM model for the FLOTRIG.

The level data and step input changes obtained for each cell during the open loop step tests were used to fit the model as follows:-

- a) Select an order of the model transfer function that will estimate the data best.
- b) Choose dummy pole and zero positions to start with.
(done automatically in software)
- c) Scale the model to the step size of the respective input. (done automatically in software)
- d) Repetitively choose pole and zero positions that will closely correspond to the response of the data plotted.
- e) Select the gain of the estimate that will closely match the steady state of the data in together with step d).

Repeat steps d) and e) until the model estimate is satisfactory.

Each entry of the transfer function matrix was modelled individually. Therefore, the procedure given below was performed for each cell's response for every open loop step test input to the FLOTRIG. The steps are as follows:-

1. Read the data from the respective file which contains the corresponding step input and level values.

2. Perform steps a) to e) as given above to obtain the model estimate.
3. If the model transfer function fits the level data then go to step 1 to read next data file, else go to step 3.

4.7.1 Calculation of Transfer Function Models based on FLOTRIG data

The estimates for the data was found by performing steps 1) to 3) given above. The offset was not subtracted, but the dc-offset present before the step input was added to the function which approximated the corresponding data.

dc-offset = the average value before the step

Estimate transfer function models obtained in the above manner were plotted together with the respective data. The estimation of the transfer functions were done by plotting and comparing them with the respective data. If the estimate is not acceptable the corresponding parameter(e.g. gain, dead-time or pole positions, etc) of the estimate are changed until the estimate is acceptable.

4.7.2 Generating the Flotation Rig Transfer Function Matrix

The TFM for the FLOTRIG was developed one column at a time. That is, each column of the matrix was generated from a single step input to a particular cell. The elements of the matrix are estimate transfer functions of the corresponding element data.

The estimates for the respective transfer functions are presented in Appendix B.

The generated Flotation Rig model TFM is given below. All matrix elements, inputs and outputs are functions of the argument 's' (Laplace domain variable):-

$$y = G u$$

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

where:-

$$\begin{array}{ll} y_1 = \text{Rougher level} & u_1 = \text{Rougher valve} \\ y_2 = \text{Scavenger level} & u_2 = \text{Scavenger valve} \\ y_3 = \text{Cleaner level} & u_3 = \text{Cleaner valve} \\ y_4 = \text{Recleaner level} & u_4 = \text{Recleaner valve} \end{array}$$

An alternative presentation, poles and zeroes of the TFM are given in Appendix B.

The TFM elements for the FLOTRIG model are given below:-

$\frac{0.009(s + 0.005)}{s^2 + 0.05s + 0.0006}$	$\frac{0.00625}{(s + 0.0125)}$	$\frac{-0.016e^{-30s}}{(s + 0.02)}$	0
$\frac{-0.00813}{(s + 0.025)}$	$\frac{0.00744}{(s + 0.012)}$	$\frac{-0.0011}{(s + 0.005)}$	0
$\frac{0.00833(s + 0.0009)e^{-15s}}{(s^2 + 0.035s + 0.003)}$	$\frac{0.0182e^{-45s}}{(s + 0.028)}$	$\frac{0.036(s+0.0005)}{(s^2 + 0.05s + 0.0006)}$	$\frac{-0.0045e^{-20s}}{(s + 0.03)}$
0	0	0	$\frac{0.00378e^{-5s}}{(s + 0.0105)}$

A nominal TFM model, $G(s)$ was derived for the FLOTRIG from the step test data. This model will be used in the controller designs for the FLOTRIG. All analysis for the FLOTRIG will be done using this model.

Chapter 5

Controller Design for the FLOTRIG using the Proposed Simplifying Control System Design Approach based on Q-Parametrization

This chapter presents the multivariable controller design procedure of the proposed approach. The control scheme was applied to digital simulations and then implemented on the FLOTRIG. The results are given in the next chapter.

The method is frequency domain based and requires a LTI 's'-domain model of the FLOTRIG. This model, $G(s)$ was determined as given in chapter 4. All design steps will be based on this model.

Two controllers were designed. One for step setpoint tracking and the other one for ramp tracking. Their respective designs are presented below.

5.1 Design of the Control System

The open-loop process matrix, $G(s)$, which is given in Appendix B, had no open loop RHP poles in the s -plane. This implies that the O/L system characteristic polynomial will have no roots in the RHP and is thus open loop stable. Hence, the open-loop stability requirements in chapters 2 and 3 are satisfied. Therefore, if the model, $M(s)$ and the Q-parameter, $Q(s)$ are stable, then the system closed loop stable provided the plant-model mismatch satisfies the condition

$$\text{Re}(\text{eigen-value}_j[G(0)M(0)^{-1}]) > 0 \quad j = 1..n$$

using equation 3-23 as in chapter 3. The system is also internally stable if $Q(s)$, $M(s)$ and $G(s)$ are stable.

In the next two sub-sections the control system designs for both step and ramp setpoint tracking are presented.

By inspecting the open-loop TFM, $G(s)$, it can be seen that the off-diagonal elements of the system are non-zero, hence interaction is present. In order to ensure satisfactory C/L performance, the interaction should be minimized.

5.2 Q-Parameter Design for Step Setpoint-Tracking

In order to eliminate interaction, the process is decoupled at low and high frequencies as proposed in chapter three, using equations 3-25 and 3-28 which yields the following matrices:

$$Q_0 = G(j0)^{-1}$$

The matrix, Q_0 given below ensures steady state decoupling and step setpoint-tracking with zero error when implemented in the IMC-structure with the model, $M(s)$, such that $M(j0)Q_0 = I$, which is a requirement for integral-action.

$$Q_0 = \begin{bmatrix} 0.075 & 0.5 & -0.8 & 0 \\ -0.325 & 0.62 & -0.22 & 0 \\ 0.00833 & 0.65 & 0.03 & -0.15 \\ 0 & 0 & 0 & 0.36 \end{bmatrix}^{-1} \quad \text{..5-1}$$

The $Q_0 = G(j0)^{-1}$ simplifies to

$$= \begin{bmatrix} 0.826 & -2.7345 & 1.973 & 0.8221 \\ 0.0217 & 0.1137 & 1.4133 & -0.5889 \\ -1.159 & -0.1853 & 1.0683 & 0.4451 \\ 0 & 0 & 0 & 2.7778 \end{bmatrix} \quad \text{..5-2}$$

The corresponding matrix, Q_∞ , that diagonalizes the process at high frequencies, which is also obtained from $G(s)$ is given below, using equations 3-25 to 3-27:

$$Q_\infty = B^{-1} \quad \text{..5-3}$$

$$\text{and } B = \begin{bmatrix} 0.009 & 0.00625 & -0.016 & 0 \\ -0.008125 & 0.00744 & -0.0011 & 0 \\ 0.00833 & 0.0182 & 0.036 & -0.0045 \\ 0 & 0 & 0 & 0.00378 \end{bmatrix}$$

where, Q_∞ simplifies to

$$Q_\infty = \begin{bmatrix} 37.2914 & -66.8722 & 14.5307 & 17.2984 \\ 36.7055 & 59.2393 & 18.1236 & 21.5758 \\ -27.1855 & -14.4753 & 15.2530 & 18.1584 \\ 0 & 0 & 0 & 264.5503 \end{bmatrix} \quad \text{..5-4}$$

The Q -parameter, $Q(s)$, with the required effects of decoupling at high and low frequencies, is given by:

$$Q(s) = [Q_\infty s + Q_0]F(s)$$

using the notation of equation 3-24. The above equation shows that

$$\lim_{s \rightarrow 0} G(s)Q(s) \rightarrow \lim_{s \rightarrow 0} G(s)Q_0 \quad \text{..5-5}$$

and

$$\lim_{s \rightarrow \infty} G(s)Q(s) \text{ tends to } \lim_{s \rightarrow \infty} G(s)Q_\infty \cdot \text{diag}\{1/\beta_{ii}\} \quad \text{..5-6}$$

which ensure decoupling and the β_{ii} reduce control action, predominantly at high frequencies where the transmittance should be kept small.

The $Q(s)$ designed above and the intentionally mismatched model, $M(s)$, which is given below together form the controller. The controller is implemented as shown in figure 2.4 of chapter 2.

The TFM of the model, $M(s)$ with rational transfer functions as elements is given below as:-

(The argument 's' was omitted for brevity.)

$$Y_m = M u$$

Y_{m1}		m_{11}	m_{12}	m_{13}	m_{14}	u_1
Y_{m2}		m_{21}	m_{22}	m_{23}	m_{24}	u_2
Y_{m3}		m_{31}	m_{32}	m_{33}	m_{34}	u_3
Y_{m4}		m_{41}	m_{42}	m_{43}	m_{44}	u_4

where:-

Y_{mi} are model output that correspond to the respective process outputs as given below

u_i are model and process inputs

- | | |
|----------------------------------|-------------------------|
| Y_{m1} = Rougher m_{level} | u_1 = Rougher valve |
| Y_{m2} = Scavenger m_{level} | u_2 = Scavenger valve |
| Y_{m3} = Cleaner m_{level} | u_3 = Cleaner valve |
| Y_{m4} = Recleaner m_{level} | u_4 = Recleaner valve |

The matrix $M(s)$ is given on the next page

$\frac{(0.075)(0.0006)}{s^2 + 0.05s + 0.0006}$	$\frac{0.00625}{(s + 0.0125)}$	$\frac{-0.016e^{-30s}}{(s + 0.02)}$	0
$\frac{-0.00813}{(s + 0.025)}$	$\frac{0.00744}{(s + 0.012)}$	$\frac{-0.0011}{(s + 0.005)}$	0
$\frac{(0.025)(0.003)e^{-15s}}{(s^2 + 0.035s + 0.003)}$	$\frac{0.0182e^{-45s}}{(s + 0.028)}$	$\frac{(0.03)(0.0006)}{(s^2 + 0.05s + 0.0006)}$	$\frac{-0.0045e^{-20s}}{(s + 0.03)}$
0	0	0	$\frac{0.00378e^{-5s}}{(s + 0.0105)}$

A TFM model, $M(s)$ was obtained by removing the zeroes of the nominal model matrix, $G(s)$ which was derived for the FLOTRIG from the step test data. This model will also be used in the controller designs for ramp setpoints.

The selection of the values for β_{ii} can be done either on-line, or off-line if the process input upper and lower saturation limits as well as the maximum step setpoint changes are known.

The off-line procedure for determining the choices for β_{jj} is as follows:-

Let the maximum step setpoint change be $\delta r = 10\%$, the upper and lower process saturation limits be $u^+ = 100\%$ and $u^- = -100\%$ respectively and the Q_∞ matrix be as determined above. Then choose β_{jj} , such that the largest column j -entry, $q_{\infty ij}$ in the matrix gives

$$|(q_{\infty ij}/\beta_{jj})\delta r| \leq \pm \frac{1}{2} |u^+| \quad \dots 5-7$$

The choice of the factor $\frac{1}{2}$, is nominal but can be whatever is suitable to avoid process input saturation. These calculations are done based on the fact that the largest control action occurs at the time or just after the step change is made. Thus, the above calculations will ensure that control action is reduced to avoid frequent process input saturation which could lead to limits cycles in the classical designs. The effect of the choice of β_{ii} is visible in the process output and hence also the controller performance.

The β_{ii} choices are: $\beta_{11} = 100$, $\beta_{22} = 100$, $\beta_{33} = 100$, $\beta_{44} = 150$

The on-line procedure is as follows:-

Choose nominal $\beta_{ii} > 0$, such that $(1 + s\beta_{ii})$ have about the same positions as the slowest pole in the corresponding column. Then implement the controller with the process in closed-loop. Adjust β_{ii} for every input that is not satisfactorily reduced by observing the process inputs just

after a step change is made for the corresponding setpoint. A compromise should be made between required input action and controller performance.

Now, consider the IMC-Structure given below:

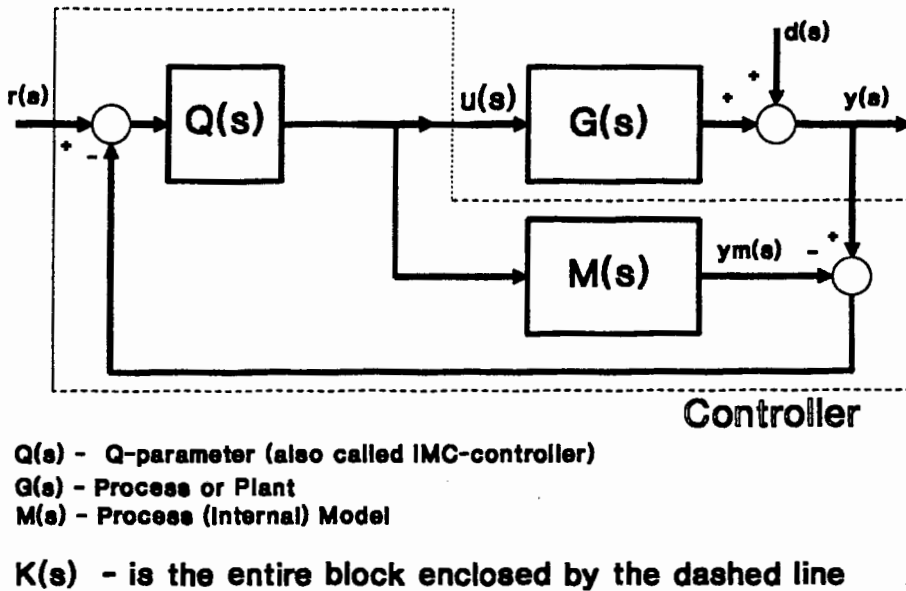


Figure 5.1: The IMC-Structure

Then the system response time can also be decreased by multiplying the model, $M(s)$, output vector magnitude by a factor, $1/\alpha$ and increasing the Q-parameter output vector by the same factor, α OR vice versa if the response is required to be slower, where α is greater than 1 in this case. This follows from the relations given below: (all vectors and matrices are functions of 's', except the constant, α)

Let the model, $M(s)$, output be, y_m

the plant, $G(s)$, output be, y

the Q-parameter output be, $Q[r-(y-y_m)]$

then to show the above statement we can write the process output as in terms of setpoint, r and model output, y_m as

$$y = GQ[r - (y - y_m)]$$

which follows from the above IMC-structure. Now, by applying the scaling, the above equation becomes

$$y = \alpha GQ[r - (y - y_m/\alpha)]$$

which can be shown to reduce to

$$y = [I + \alpha GQ]^{-1}[\alpha GQ][r + y_m/\alpha] \quad \text{..5-8}$$

where:- $\alpha > 1$ for reducing response time

$\alpha < 1$ for increasing response time

$\alpha \neq 0$

From the above relation it can be seen the C/L loop system response can be altered by the choice of α and still maintaining setpoint tracking, since

$$\begin{aligned} \lim_{s \rightarrow 0} sy(s) &= \lim_{s \rightarrow 0} s[I + \alpha G(s)Q(s)]^{-1}[\alpha G(s)Q(s)][r(s) + y_m(s)/\alpha] \\ &= [I + \alpha I]^{-1}[\alpha I][r + r/\alpha] \\ &= r \quad \text{QED} \end{aligned}$$

using the Final Value Theorem.

An analysis has been carried out to check the extent to which:

- * accurate asymptotic setpoint tracking is achieved
- * low and high frequency decoupling is minimized

* disturbance and noise attenuation is improved.

Appendix D, contains the Bode plots for the Transmittance matrix, $T(s) = G(s)Q(s)$ which approximates the closed loop system very closely, since the C/L implementation is based on an explicitly open-loop IMC-Structure. $T(s)$ provides information on design aims such as the ones given above, which are in direct relation to the design of the parameter, $Q(s)$.

5.3 Q-Parameter Design for Ramp Setpoint-Tracking

The design for the Q-Parameter for Ramp tracking uses the starting point of the model, $M(s)$ and Q_0 as given in section 5.2 above, such that the following equation is defined.

$$Q(s) = Q_0 F(s) R(s) \quad \dots 5-9$$

From equation 5-9 it can be seen that $F(s)R(s)$ needs to be designed, for $Q(s)$ to satisfy the ramp-tracking conditions using equations 3-33 to 3-41 from chapter 3 for tracking inputs of the form $r(s) = R/s^2$.

The designs for $R(s)$ will be given in a general form for both first and second order polynomials, since 1st and 2nd orders are the only denominator orders of elements on the diagonal of model, $M(s)$. The procedure is given below:

Firstly: First Order elements

Consider the first order diagonal elements of model, $M(s)$ without delays(if any), which are written as

$$m_{ii}(s) = \frac{a}{c + s}$$

which can be written as

$$m_{ii}(s) = \frac{a/c}{1 + (1/c)s}$$

Then make the numerator of $m_{ii}(s)$ unity, since

$$\begin{aligned} \lim_{s \rightarrow 0} M(s)Q(s) &= \lim_{s \rightarrow 0} [M(s)Q_0]F(s)R(s) && \text{..5-10} \\ &= \lim_{s \rightarrow 0} [I]F(s)R(s) \end{aligned}$$

from which it can be seen that Q_0 decouples $M(s)$ to unity as s tends to zero. It also implies that the diagonal elements have unity gain as s tends to zero. Thus, we can write

$$m_{ii}(s) = \frac{1}{1 + (1/c)s} \quad \text{..5-11}$$

without loss of design goals and information. $F(s)$ can be selected either separately or together with $R(s)$ such that they correspond to $P(s) = F(s)R(s)$ from equation 3-38.

The f_{ii} were selected to be

$$f_{ii}(s) = \frac{1}{1 + \beta_{ii}s}$$

such that: $\beta_{11} = 30$, $\beta_{22} = 40$, $\beta_{33} = 30$, $\beta_{44} = 45$

Following the selection of β_{ii} above, it was noted that only $R(s)$ had to be added to satisfy the requirements for ramp tracking, where $R(s)$ contains $r_{ii}(s)$, such that

$$r_{ii}(s) = \frac{n_{rii}(s)}{d_{rii}(s)} \quad \dots 5-12$$

which together with $F(s)$ gives

$$p_{ii}(s) = fr_{ii}(s) = \frac{n_{rii}(s)}{d_{rii}(s)(1 + \beta_{ii}s)} \equiv \frac{n_{frii}(s)}{d_{frii}(s)} \quad \dots 5-13$$

where the denominator polynomial, $d_{frii}(s)$, corresponds to $d_q(s)$ given in chapter three. This also shows that the denominators resulting from $F(s)R(s)$ could have been used to select the required pole positions for ramp tracking. From equations 3-11 to 3-14, it follows that for ramp-tracking, $\deg\{d_{frii}(s)\}$ must be at least 2. Thus, choosing the least order for $d_{rii}(s)$, $d_{frii}(s)$ becomes

$$d_{frii}(s) = (1 + \tau_{ii}s)(1 + \beta_{ii}s) \quad \dots 5-14$$

Therefore we have

$$\begin{aligned} d_{frii}(s) &= (1 + \tau_{ii}s)(1 + \beta_{ii}s) \\ &= 1 + (\tau_{ii} + \beta_{ii})s + \tau_{ii}\beta_{ii}s^2 \end{aligned} \quad \dots 5-15$$

$$n_{frii}(s) = n_0 + n_1s + \dots \text{ which is unknown and } \quad \dots 5-16$$

$$m_{ii}(s) = \frac{1}{1 + (1/c)s} \quad \dots 5-17$$

which correspond to $d_q(s)$, $n(s)$ and $\boxed{\frac{N(s)}{D(s)}}$ in equations

3-11 to 3-14 of chapter three.

Using equation 3-14 which is also given below as

$$\begin{bmatrix} n_0 \\ n_1 \\ n_2 \\ \dots \end{bmatrix} = \begin{bmatrix} N_0 & 0 & 0 & : \\ N_1 & N_0 & 0 & : \\ N_2 & N_1 & N_0 & : \\ \dots & \dots & \dots & \dots \end{bmatrix}^{-1} \begin{bmatrix} 1 & 0 & 0 & : \\ D_1 & 1 & 0 & : \\ D_2 & D_1 & 1 & : \\ \dots & \dots & \dots & \dots \end{bmatrix} \begin{bmatrix} 1 \\ d_1 \\ d_2 \\ \dots \end{bmatrix} \quad \begin{array}{l} \text{Type No:} \\ \text{Type 1} \\ \text{Type 2} \\ \text{Type 3} \end{array}$$

and equations 5-15 to 5-17 to determine $n_{frii}(s)$ it becomes

$$\begin{bmatrix} n_0 \\ n_1 \end{bmatrix} = \begin{bmatrix} 1 & 0 \\ 0 & 1 \end{bmatrix}^{-1} \begin{bmatrix} 1 & 0 \\ 1/c & 1 \end{bmatrix} \begin{bmatrix} 1 \\ \tau_{ii} + \beta_{ii} \end{bmatrix} \quad \begin{array}{l} \text{Type no:} \\ 1 \\ 2 \end{array}$$

which gives the required constraint on $n_{frii}(s)$ of

$$\begin{bmatrix} n_0 \\ n_1 \end{bmatrix} = \begin{bmatrix} 1 \\ 1/c + \tau_{ii} + \beta_{ii} \end{bmatrix} \quad \begin{array}{l} \text{Type no:} \\ 1 \\ 2 \end{array}$$

Secondly: Second Order elements of $M(s)$

Consider the second order diagonal elements of model, $M(s)$ without delays(if any), which are written as

$$m_{ij}(s) = \frac{a}{c + bs + s^2}$$

which can be written as

$$m_{ii}(s) = \frac{a/c}{1 + (b/c)s + (1/c)s^2}$$

from which the numerator can be made unity, in view of equation 5-10 . It also implies that the diagonal elements have unity gain as s tends to zero. Thus, we can write

$$m_{ii}(s) = \frac{1}{1 + (b/c)s + (1/c)s^2}$$

without loss of design goals and information. $F(s)$ is also known from above.

Thus only $R(s)$ has to be added to satisfy the requirements for ramp tracking, where $R(s)$ contains $r_{ii}(s)$, such that

$$r_{ii}(s) = \frac{n_{rii}(s)}{d_{rii}(s)}$$

which together with $F(s)$ gives

$$\begin{aligned} f_{rii}(s) &= \frac{n_{rii}(s)}{d_{rii}(s)(1 + \beta_{ii}s)} \\ &\equiv \frac{n_{frii}(s)}{d_{frii}(s)} \end{aligned}$$

where the denominator polynomial, $d_{frii}(s)$ can be associated with $d_q(s)$ given equation 3-4. From equations 3-11 to 3-14 it follows that for ramp-tracking, $\deg(d_{frii}(s))$ must be at least 2. Thus, choosing the least order for $d_{rii}(s)$, $d_{frii}(s)$ becomes

$$d_{frii}(s) = (1 + \tau_{ii}s)(1 + \beta_{ii}s)$$

Therefore we have

$$\begin{aligned} d_{frii}(s) &= (1 + \tau_{ii}s)(1 + \beta_{ii}s) \\ &= 1 + (\tau_{ii} + \beta_{ii})s + \tau_{ii}\beta_{ii}s^2 \end{aligned}$$

$n_{frii}(s) = n_0 + n_1s + \dots$ can be any order as long as $F(s)R(s)$ is realizable

$$m_{ii}(s) = \frac{1}{1 + (b/c)s + (1/c)s^2}$$

which correspond to $d_q(s)$, $n(s)$ and $\begin{bmatrix} N(s) \\ D(s) \end{bmatrix}$ respectively.

Using the equations 3-11 to 3-14 to determine $n_{frii}(s)$ which are also given above for first order systems we obtain

$$\begin{bmatrix} n_0 \\ n_1 \end{bmatrix} = \begin{bmatrix} 1 & 0 \\ 0 & 1 \end{bmatrix}^{-1} \begin{bmatrix} 1 & 0 \\ b/c & 1 \end{bmatrix} \begin{bmatrix} 1 \\ \tau_{ii} + \beta_{ii} \end{bmatrix} \quad \begin{matrix} \text{Type no:} \\ 1 \\ 2 \end{matrix}$$

which gives the required constraint on $n_{frii}(s)$ of

$$\begin{bmatrix} n_0 \\ n_1 \end{bmatrix} = \begin{bmatrix} 1 \\ b/c + \tau_{ii} + \beta_{ii} \end{bmatrix} \quad \begin{matrix} \text{Type no:} \\ 1 \\ 2 \end{matrix}$$

The final design for complete diagonal matrix is given below in the form of coefficient vectors similar to the above for the polynomial $n_{frii}(s)$ and $d_{frii}(s)$ as:

$$\begin{array}{ll} \text{coeffs } n_{frii}(s): & \text{coeff } d_{frii}(s): \\ n_0 \begin{bmatrix} 1 \\ \tau_{11} + \beta_{11} + (1 \text{ or } b)/c \end{bmatrix} & d_0 \begin{bmatrix} 1 \\ \tau_{11} + \beta_{11} \end{bmatrix} \\ n_1 & d_1 \begin{bmatrix} \tau_{11} \beta_{11} \end{bmatrix} \\ & d_2 \end{array}$$

The diagonal elements of $F(s)R(s)$ are thus

element 11

$$\begin{array}{ll} \text{coeffs } n_{fr11}(s): & \text{coeff } d_{fr11}(s): \\ n_0 \begin{bmatrix} 1 \\ 50 + 30 + (83.333) \end{bmatrix} & d_0 \begin{bmatrix} 1 \\ 50 + 30 \end{bmatrix} \\ n_1 & d_1 \begin{bmatrix} (50)(30) \end{bmatrix} \\ & d_2 \end{array}$$

element 22coeffs $n_{fr22}(s)$:

$$\begin{matrix} n_0 \\ n_1 \\ n_2 \end{matrix} \begin{bmatrix} 1 \\ 123.33 + 83.33 \\ (123.33)(83.33) \end{bmatrix}$$

coeff $d_{fr22}(s)$:

$$\begin{matrix} d_0 \\ d_1 \\ d_2 \end{matrix} \begin{bmatrix} 1 \\ 83.333 + 40 \\ 3333.32 \end{bmatrix}$$

element 33coeffs $n_{fr33}(s)$:

$$\begin{matrix} n_0 \\ n_1 \end{matrix} \begin{bmatrix} 1 \\ 50 + 30 + (83.333) \end{bmatrix}$$

coeff $d_{fr33}(s)$:

$$\begin{matrix} d_0 \\ d_1 \\ d_2 \end{matrix} \begin{bmatrix} 1 \\ 50 + 30 \\ (50)(30) \end{bmatrix}$$

element 44coeffs $n_{fr44}(s)$:

$$\begin{matrix} n_0 \\ n_1 \end{matrix} \begin{bmatrix} 1 \\ 50 + 45 + 95.238 \end{bmatrix}$$

coeff $d_{fr44}(s)$:

$$\begin{matrix} d_0 \\ d_1 \\ d_2 \end{matrix} \begin{bmatrix} 1 \\ 50 + 45 \\ (50)(45) \end{bmatrix}$$

The above design was tested using digital simulation and then applied to the FLOTRIG. The results obtained from the digital simulations and the experimental FLOTRIG are discussed and compared in the next chapter to evaluate the above Proposed Simplifying Design Approach. The method is based on Q(Youla)-Parametrization for a control system implemented in the IMC-structure.

for the same C/L matrix as given below:

$$\begin{bmatrix} Y_{n1}(s) \\ Y_{n2}(s) \\ Y_{n3}(s) \\ Y_{n4}(s) \end{bmatrix} = \begin{bmatrix} h_{n11}(s) & h_{n12}(s) & h_{n13}(s) & h_{n14}(s) \\ h_{n21}(s) & h_{n22}(s) & h_{n23}(s) & h_{n24}(s) \\ h_{n31}(s) & h_{n32}(s) & h_{n33}(s) & h_{n34}(s) \\ h_{n41}(s) & h_{n42}(s) & h_{n43}(s) & h_{n44}(s) \end{bmatrix} \begin{bmatrix} r_1(s) \\ r_2(s) \\ r_3(s) \\ r_4(s) \end{bmatrix}$$

where n stands for either (R) $\equiv$ FLOTRIG

or (S) $\equiv$ PC-Simulation

6.1 Comparison of Results for Step Setpoints

In the following two subsections, the results obtained from the FLOTRIG and the PC-Simulation are presented. They are:-

- Process outputs
- Setpoints
- Process inputs
- Model outputs

for a particular step setpoint change from $r_1(s)$ to $r_4(s)$.

The TFM matrices that form the controller designed for step setpoint tracking are given below as:

$$Q(s) = [sQ_\infty + Q_0]F(s)$$

$Q(s)=$

$\frac{37.2914s - 0.0826}{100s + 1}$	$\frac{-66.8722s - 2.7345}{100s + 1}$	$\frac{14.5307s + 1.973}{100s + 1}$	$\frac{17.2984s + 0.8221}{150s + 1}$
$\frac{36.7055s + 0.0217}{100s + 1}$	$\frac{59.2393s + 0.1137}{100s + 1}$	$\frac{18.1236s + 1.4133}{100s + 1}$	$\frac{21.5758s - 0.5889}{150s + 1}$
$\frac{-27.1855 - 1.159}{100s + 1}$	$\frac{-14.4753s - 0.1853}{100s + 1}$	$\frac{15.2530s + 1.0683}{100s + 1}$	$\frac{18.1584s + 0.4451}{150s + 1}$
0	0	0	$\frac{264.5503s - 2.7778}{150s + 1}$

and the model, $M(s)$ which is given as

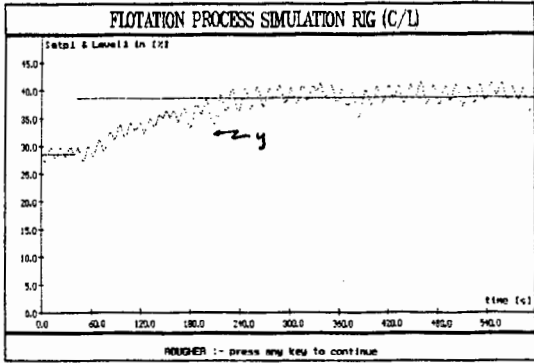
$\frac{(0.075)(0.0006)}{s^2 + 0.05s + 0.0006}$	$\frac{0.00625}{(s + 0.0125)}$	$\frac{-0.016e^{-30s}}{(s + 0.02)}$	0
$\frac{-0.00813}{(s + 0.025)}$	$\frac{0.00744}{(s + 0.012)}$	$\frac{-0.0011}{(s + 0.005)}$	0
$\frac{(0.025)(0.003)e^{-15s}}{(s^2 + 0.035s + 0.003)}$	$\frac{0.0182e^{-45s}}{(s + 0.028)}$	$\frac{(0.03)(0.0006)}{(s^2 + 0.05s + 0.0006)}$	$\frac{-0.0045e^{-20s}}{(s + 0.03)}$
0	0	0	$\frac{0.00378e^{-5s}}{(s + 0.0105)}$

6.1.1 Comparison of Process Outputs (STEP)

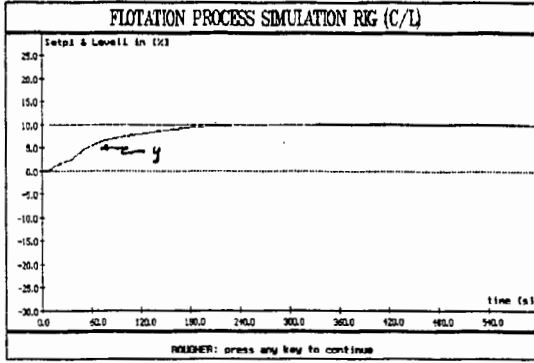
The graphs on the next four pages are given in matrix form. Every page contains two columns. The graphs in column one, show the responses of the FLOTRIG process outputs and column two, the PC-Simulated process outputs responses.

The process outputs and setpoints are distinguishable as:-

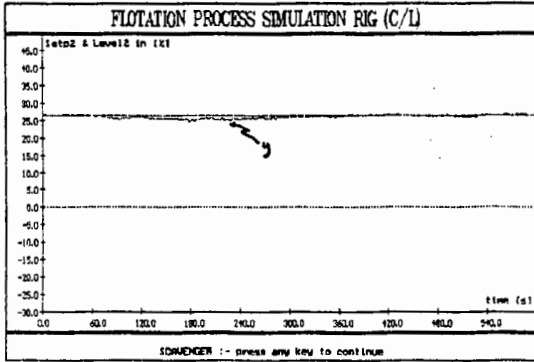
- Process outputs (noisy line for FLOTRIG outputs)
(smooth line for Simulation outputs)
- Setpoints (straight solid line) on one graph respectively.



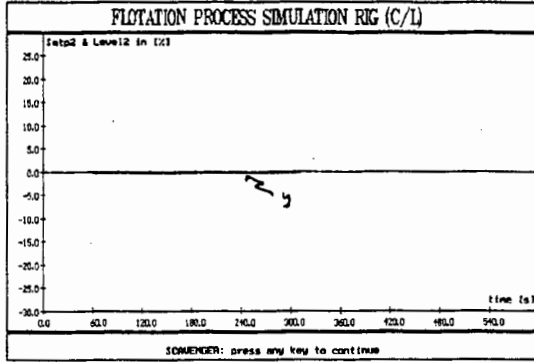
$Y_{R1}(s)$



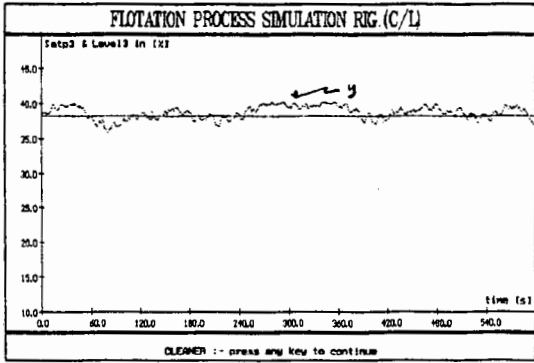
$Y_{S1}(s)$



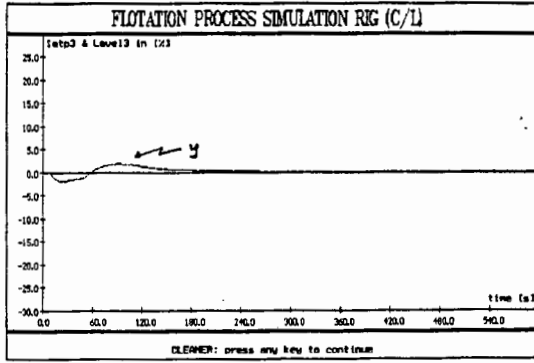
$Y_{R2}(s)$



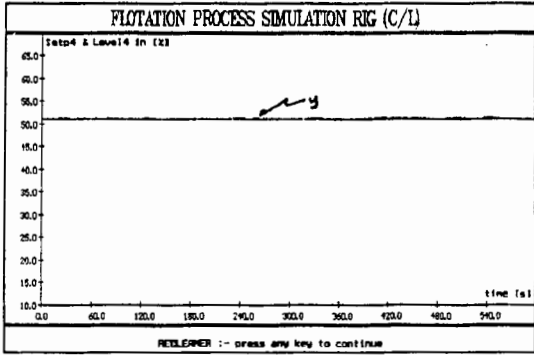
$Y_{S2}(s)$



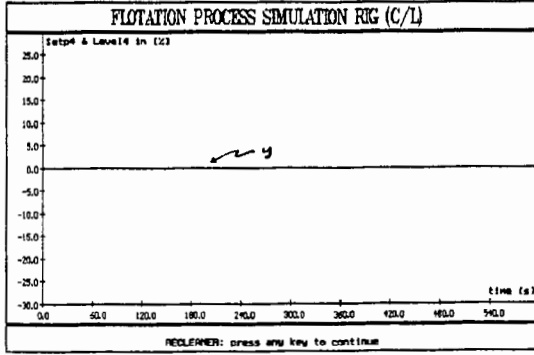
$Y_{R3}(s)$



$Y_{S3}(s)$



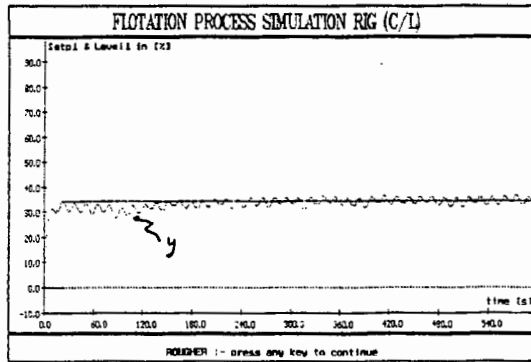
$Y_{R4}(s)$



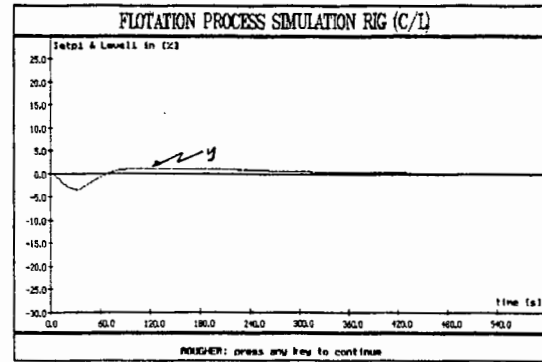
$Y_{S4}(s)$

Figure 6.1: O/p s, y_R [$r_1(s)$ stepped +10%]

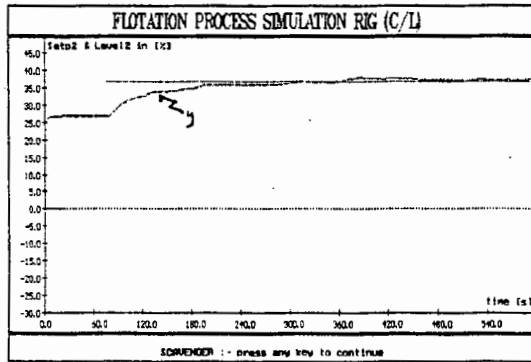
Figure 6.2: O/p s, y_S [$r_1(s)$ stepped +10%]



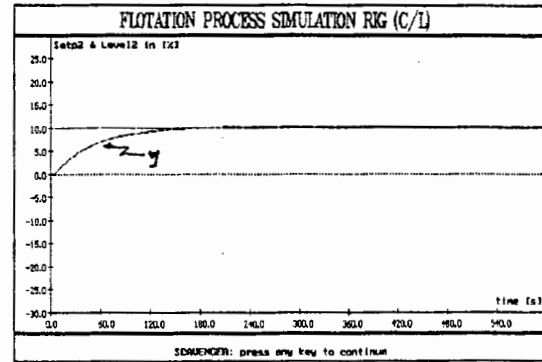
$Y_{R1}(s)$



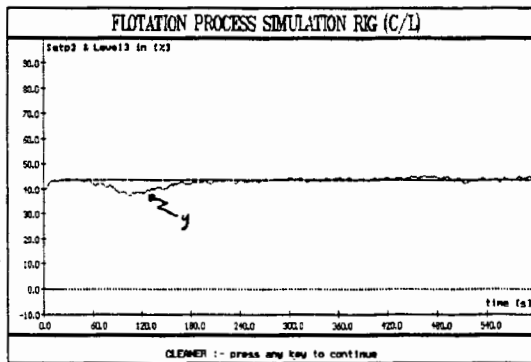
$Y_{S1}(s)$



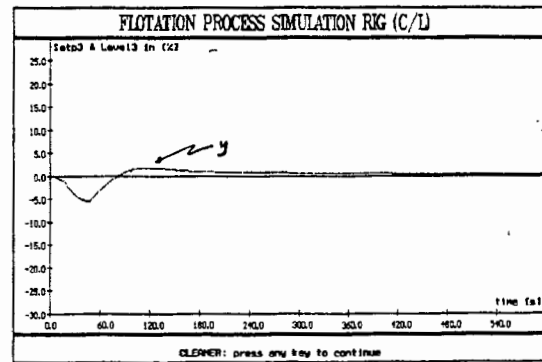
$Y_{R2}(s)$



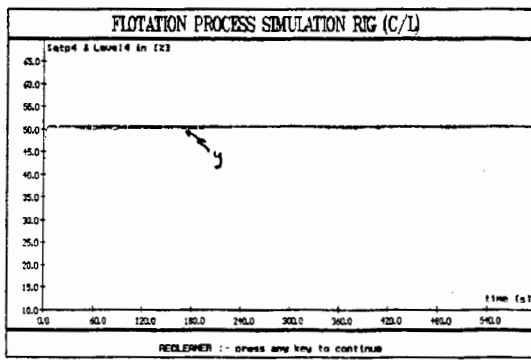
$Y_{S2}(s)$



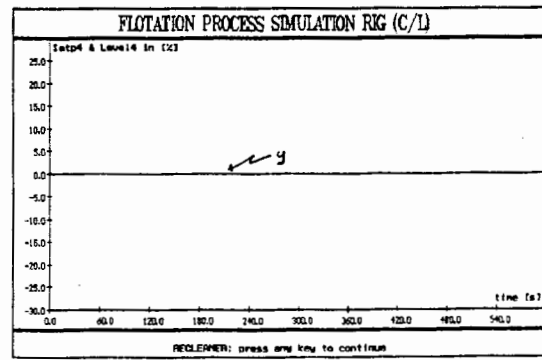
$Y_{R3}(s)$



$Y_{S3}(s)$



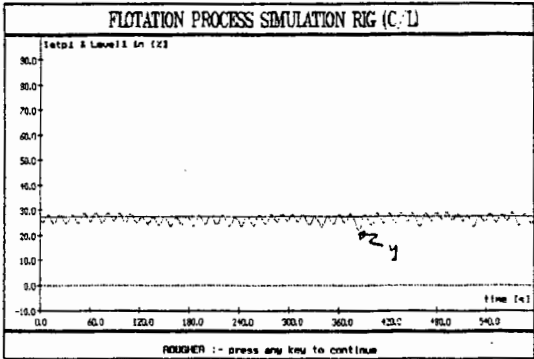
$Y_{R4}(s)$



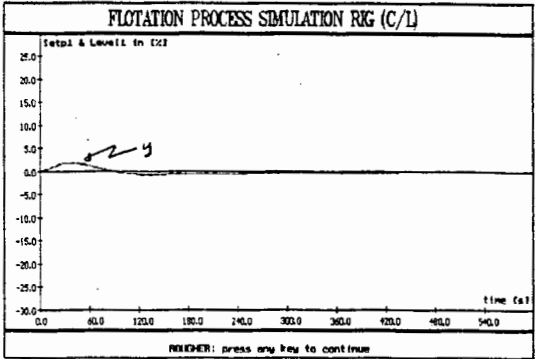
$Y_{S4}(s)$

Figure 6.3 O/p's, $y_R [r_2(s)]$ stepped +10%

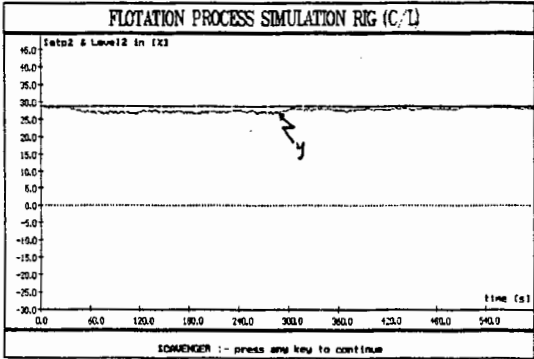
Figure 6.4 O/p's, $y_S [r_2(s)]$ stepped +10%



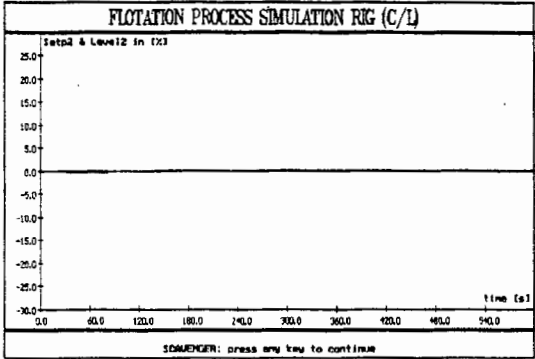
YR1(s)



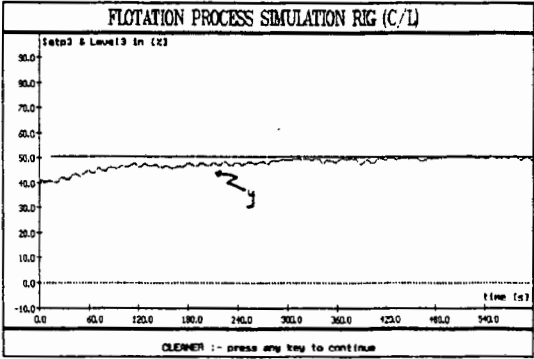
YS1(s)



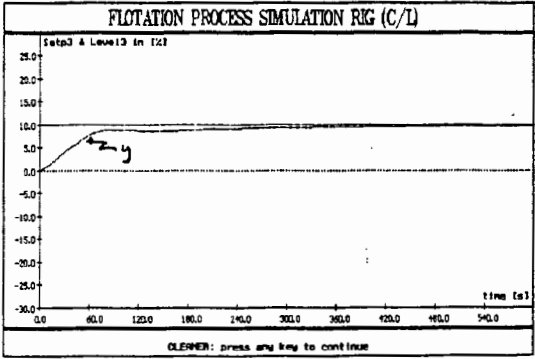
YR2(s)



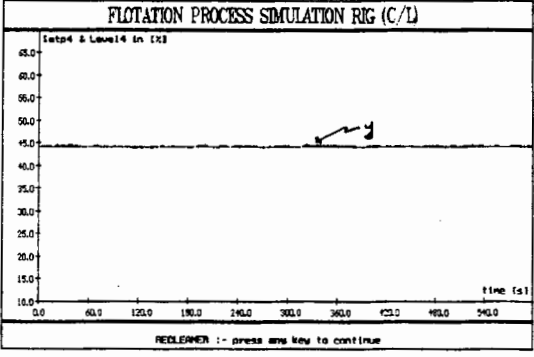
YS2(s)



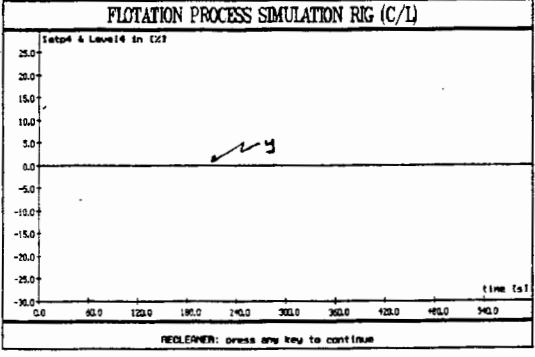
YR3(s)



YS3(s)



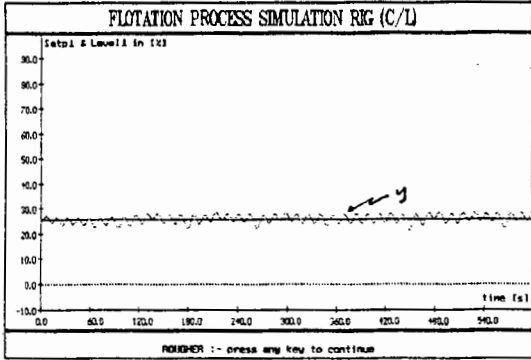
YR4(s)



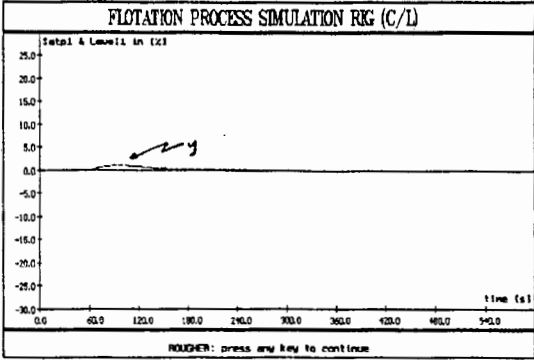
YS4(s)

Figure 6.5 O/ps, $y_R [r_3(s) \text{ stepped } +5\%]$

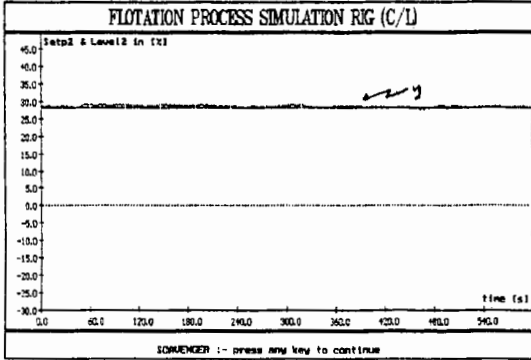
Figure 6.6 O/ps, $y_S [r_3(s) \text{ stepped } +10\%]$



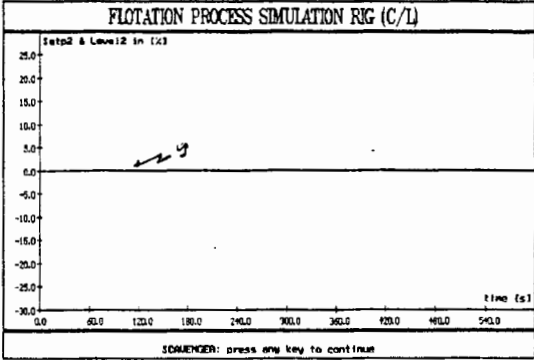
$Y_{R1}(s)$



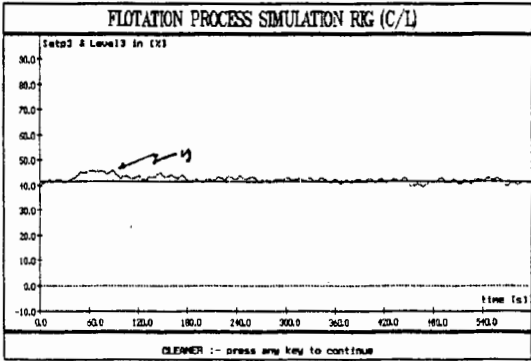
$Y_{S1}(s)$



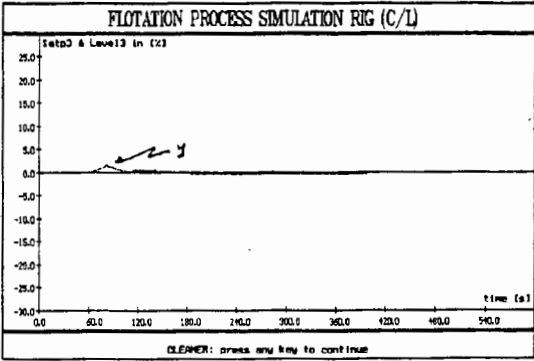
$Y_{R2}(s)$



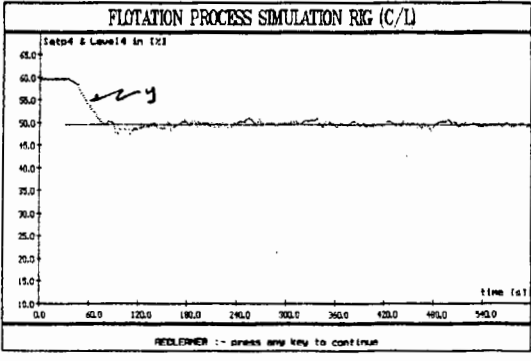
$Y_{S2}(s)$



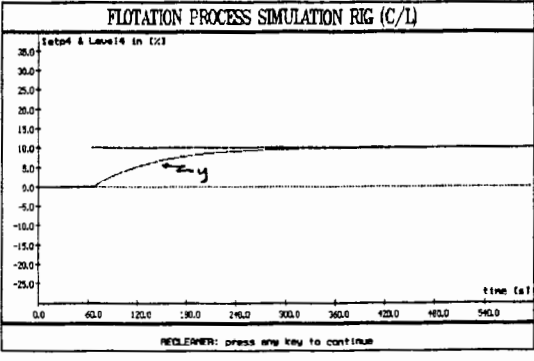
$Y_{R3}(s)$



$Y_{S3}(s)$



$Y_{R4}(s)$



$Y_{S4}(s)$

Figure 6.7 O/ps, $y_R[r_4(s)$ stepped -10%]

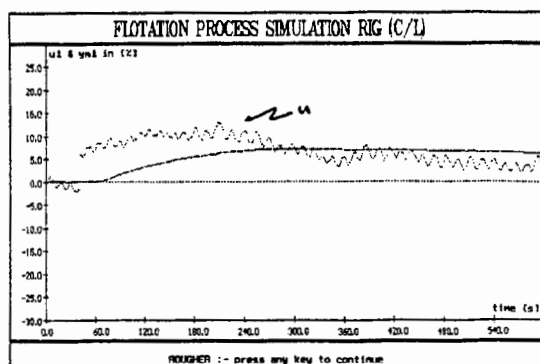
Figure 6.8 O/ps, $y_S[r_4(s)$ stepped +10%]

6.1.2 Comparison of Process I/P and Model O/P (STEP)

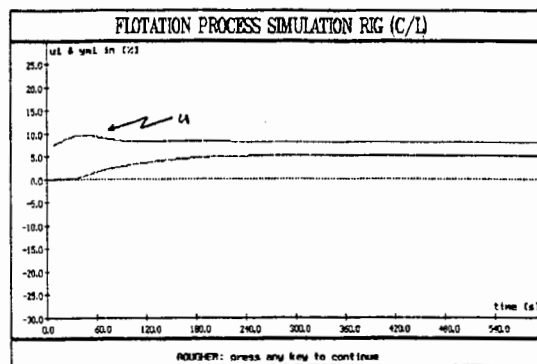
The graphs on the next four pages are given in matrix form. Every page contains two columns. The graphs in column one, show the responses of the FLOTRIG process inputs and model outputs. Column two, shows the PC-Simulated process inputs and model outputs.

They process inputs and model outputs are

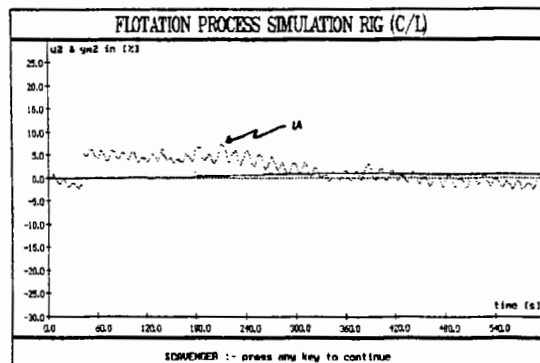
- Process inputs (the lines that show initial jumps at the start that correspond to the time at which the step change occurred)
 - Model outputs (smooth continuous lines)
- respectively.



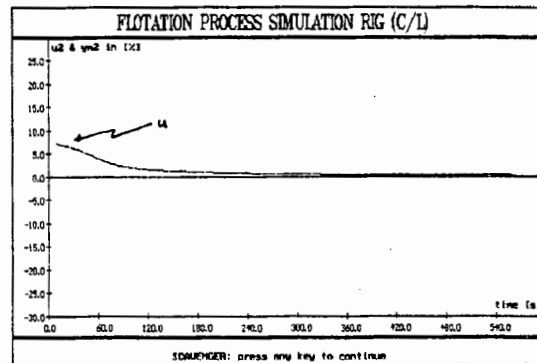
u_{11} & y_{m11} (R)



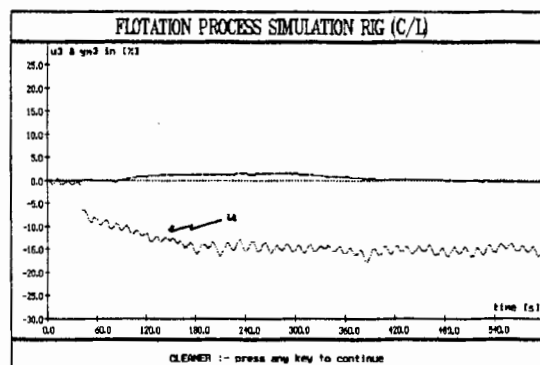
u_{11} & y_{m11} (S)



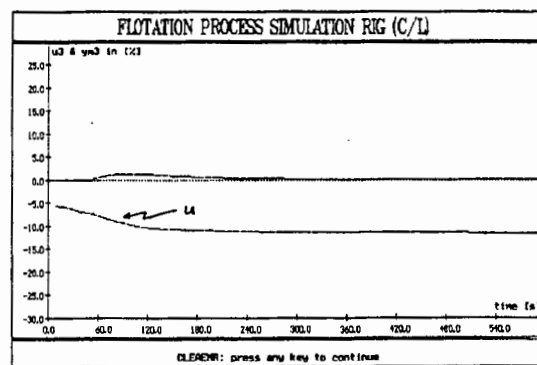
u_{21} & y_{m21} (R)



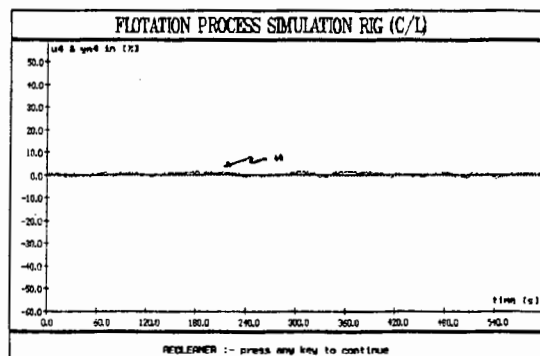
u_{21} & y_{m21} (S)



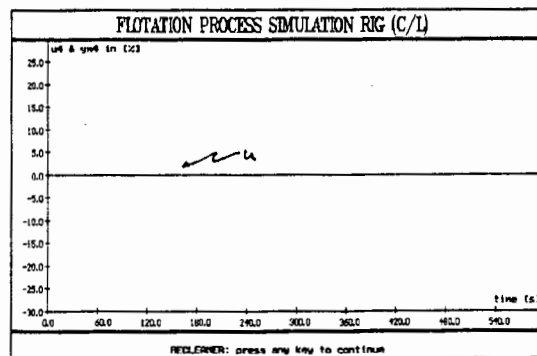
u_{31} & y_{m31} (R)



u_{31} & y_{m31} (S)



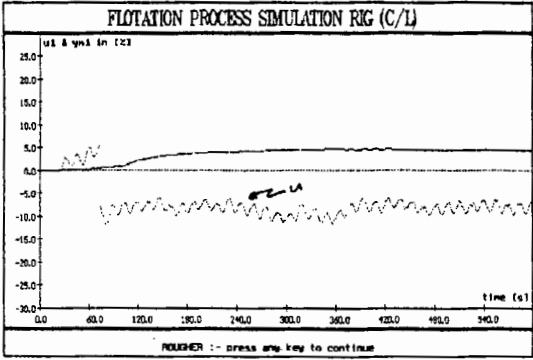
u_{41} & y_{m41} (R)



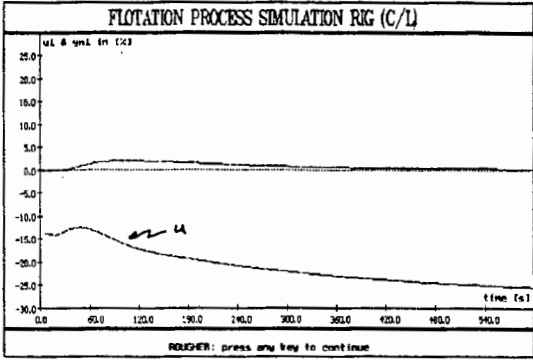
u_{41} & y_{m41} (S)

Figure 6.9 u & $y_m[r_1(s)$ stepped +10%]-R

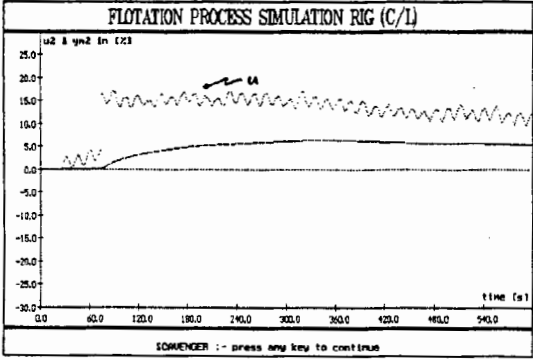
Figure 6.10 u & $y_m[r_1(s)$ stepped +10%]-S



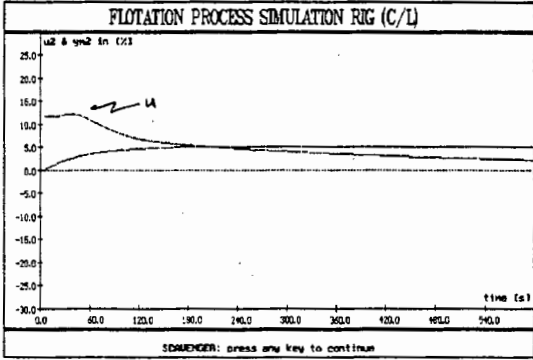
u_{12} & y_{m12} (R)



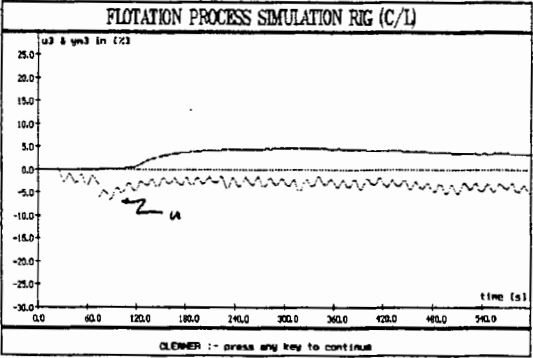
u_{12} & y_{m12} (S)



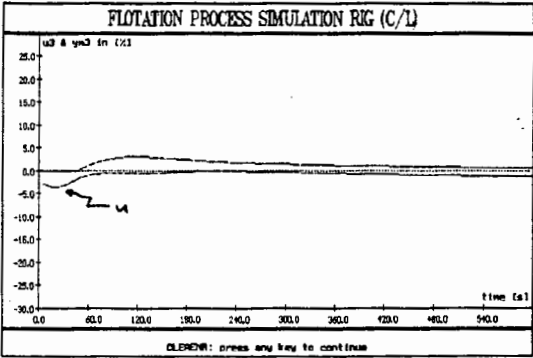
u_{22} & y_{m22} (R)



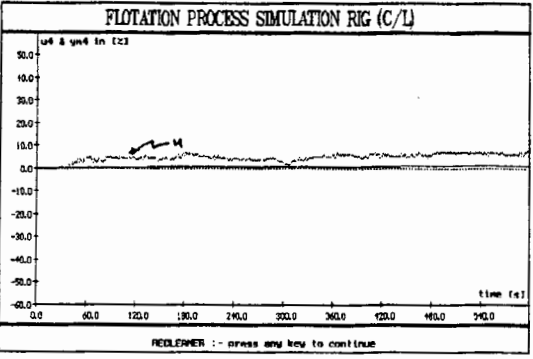
u_{22} & y_{m22} (S)



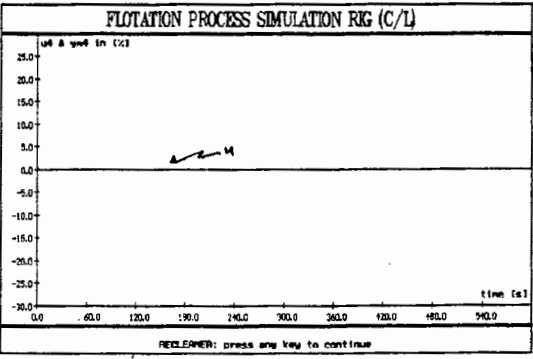
u_{32} & y_{m32} (R)



u_{32} & y_{m32} (S)



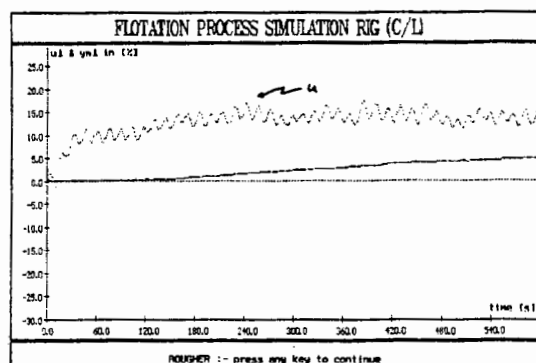
u_{42} & y_{m42} (R)



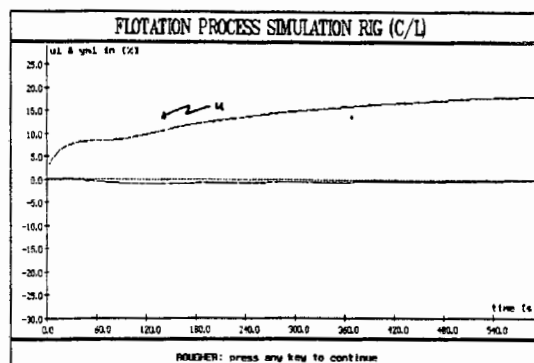
u_{42} & y_{m42} (S)

Figure 6.11 u & $y_m[r_2(s)$ stepped +10%]-R

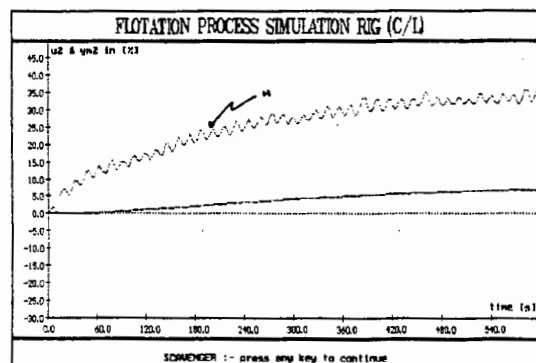
Figure 6.12 u & $y_m[r_2(s)$ stepped +10%]-S



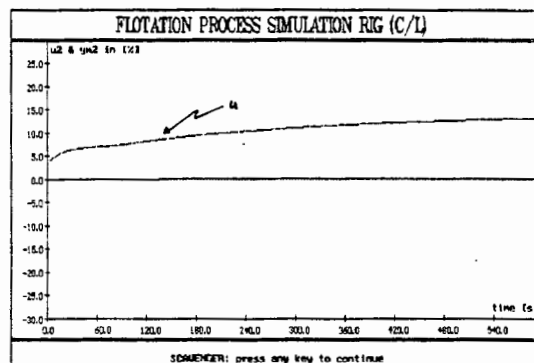
u_{13} & y_{m13} (R)



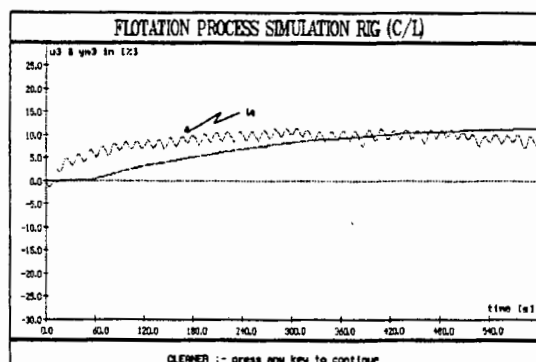
u_{13} & y_{m13} (S)



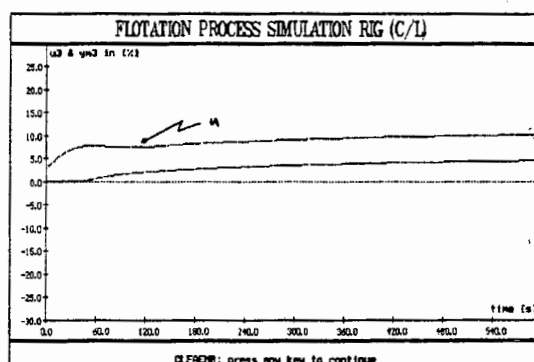
u_{23} & y_{m23} (R)



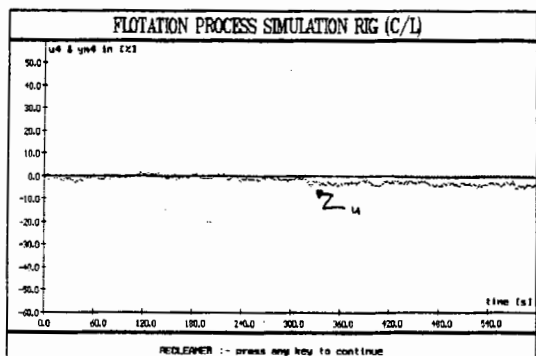
u_{23} & y_{m23} (S)



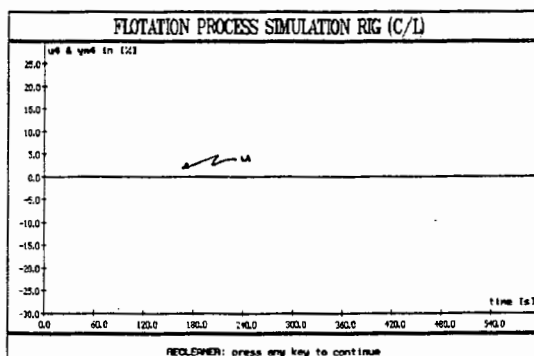
u_{33} & y_{m33} (R)



u_{33} & y_{m33} (S)



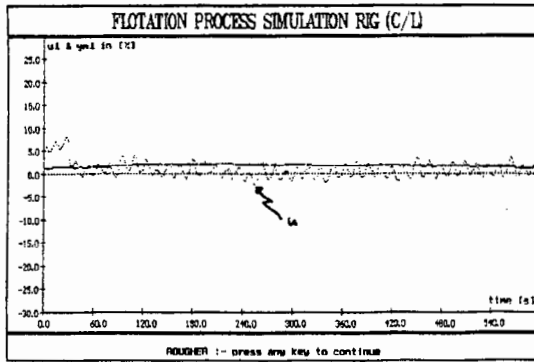
u_{43} & y_{m43} (R)



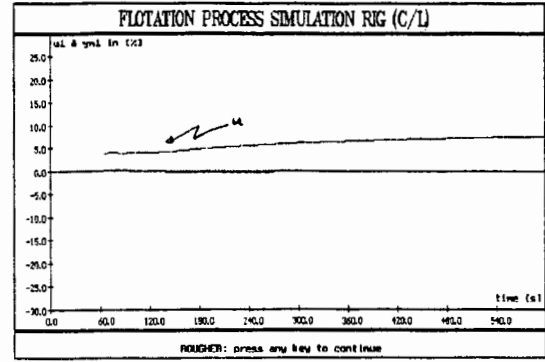
u_{43} & y_{m43} (S)

Figure 6.13 u & $y_m[r_3(s)]$ stepped +5%-R

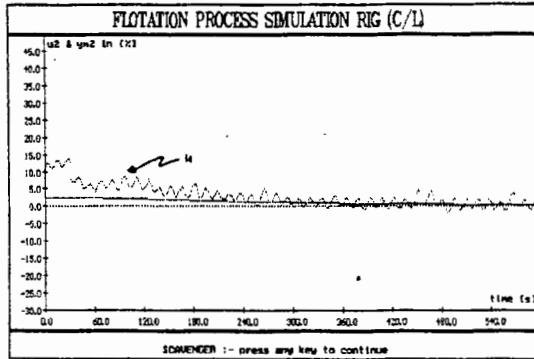
Figure 6.14 u & $y_m[r_3(s)]$ stepped +10%-S



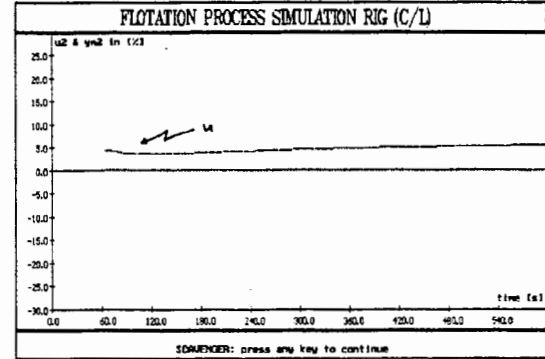
u_{14} & y_{m14} (R)



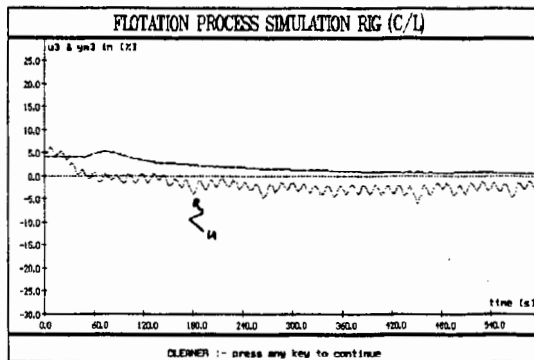
u_{14} & y_{m14} (S)



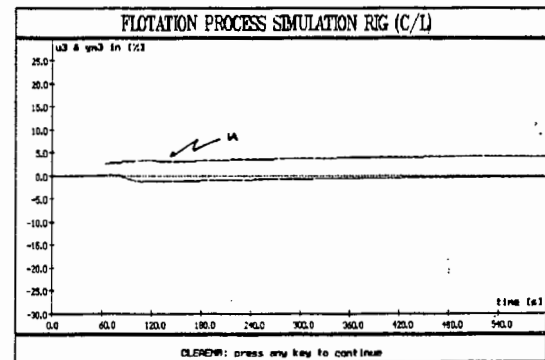
u_{24} & y_{m24} (R)



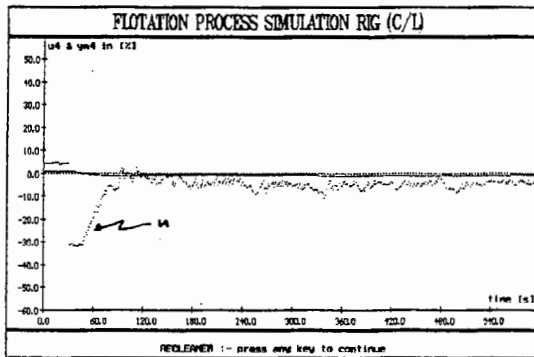
u_{24} & y_{m24} (S)



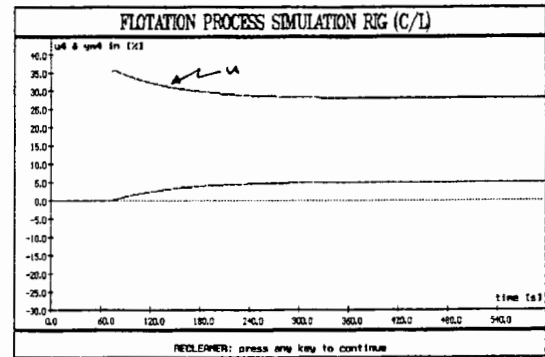
u_{34} & y_{m34} (R)



u_{34} & y_{m34} (S)



u_{44} & y_{m44} (R)



u_{44} & y_{m44} (S)

Figure 6.15 u & $y_m[r_4(s) \text{ stepped } +10\%]$ -R

Figure 6.16 u & $y_m[r_4(s) \text{ stepped } +10\%]$ -S

6.2 Comparison of Results for Ramp Setpoints

In the following two subsections, the results obtained from the FLOTRIG and the PC-Simulation are presented.

They are:-

- Process outputs
- Process inputs
- Model output

for a particular step setpoint change from $r_1(s)$ to $r_4(s)$.

The TFM matrices that form the controller are given below. The square brackets show how they were implemented in the actual software. They are

$$Q(s) = Q_0 P(s) = [Q_0 F(s)] R(s)$$

$Q_0 F =$

$\frac{-0.0826}{30s + 1}$	-2.7345	$\frac{1.973}{30s + 1}$	$\frac{0.8221}{45s + 1}$
$\frac{0.0217}{30s + 1}$	0.1137	$\frac{1.4133}{30s + 1}$	$\frac{-0.5889}{45s + 1}$
$\frac{-1.159}{30s + 1}$	-0.1853	$\frac{1.0683}{30s + 1}$	$\frac{0.4451}{45s + 1}$
0	0	0	$\frac{-2.7778}{45s + 1}$

and $R(s)$ is given as

$R(s) =$

$\frac{163.33s + 1}{50s + 1}$	0	0	0
0	$\frac{10277s^2 + 206.66s + 1}{3333.3s^2 + 123.3s + 1}$	0	0
0	0	$\frac{163.3s + 1}{50s + 1}$	0
0	0	0	$\frac{190.238s + 1}{50s + 1}$

and the model, $M(s)$ which is given as

$\frac{(0.075)(0.0006)}{s^2 + 0.05s + 0.0006}$	$\frac{0.00625}{(s + 0.0125)}$	$\frac{-0.016e^{-30s}}{(s + 0.02)}$	0
$\frac{-0.00813}{(s + 0.025)}$	$\frac{0.00744}{(s + 0.012)}$	$\frac{-0.0011}{(s + 0.005)}$	0
$\frac{(0.025)(0.003)e^{-15s}}{(s^2 + 0.035s + 0.003)}$	$\frac{0.0182e^{-45s}}{(s + 0.028)}$	$\frac{(0.03)(0.0006)}{(s^2 + 0.05s + 0.0006)}$	$\frac{-0.0045e^{-20s}}{(s + 0.03)}$
0	0	0	$\frac{0.00378e^{-5s}}{(s + 0.0105)}$

Note that the implementation of element $r_{22}(s)$ is different from the others since dynamic compensation was applied

although it was not required. In addition, if the $\deg(\text{denominator}) < \deg(\text{numerator})$ then $r_{ii}(s)$ would be unrealizable. This would have resulted from splitting the denominator as for the others to obtain a separate denominator for element $f_{22}(s)$.

6.2.1 Comparison of Process Outputs (RAMP)

The graphs on the next four pages are given in matrix form. Every page contains two columns. The graphs in column one, show the responses of the FLOTRIG process outputs. Column two, shows the PC-Simulated process outputs.

In the following graphs for ramp setpoints, the notation below is used:

ramp +4% OR ramp -4%

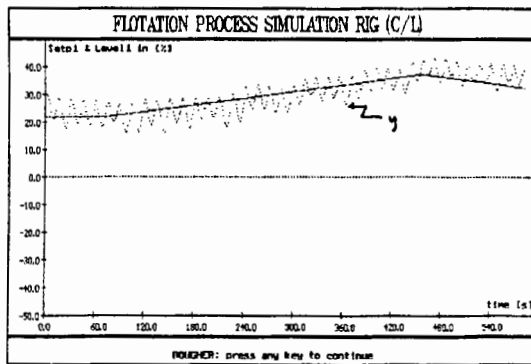
This denotes a slope of 4 percent obtained as follows:

$$\begin{aligned} \text{ramp slope} &= \frac{+ 20[\%]}{500[\text{sec}]} = + 0.04 [\% / \text{sec}] \\ &= + 4\% \end{aligned}$$

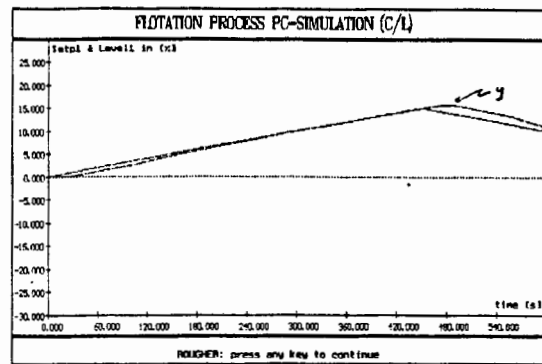
similarly for -4%

The process outputs and ramp setpoints are distinguished as:-

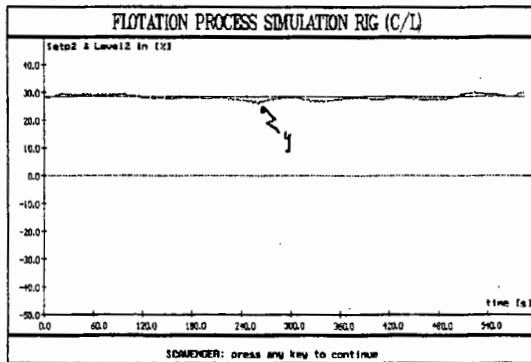
- Process outputs (noisy line for FLOTRIG results and smooth line for Simulation results)
- Setpoints (straight solid line) on one graph



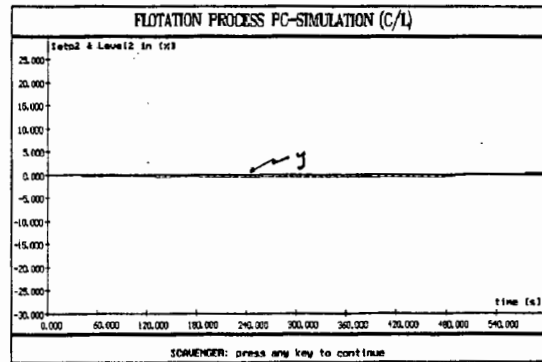
YR1



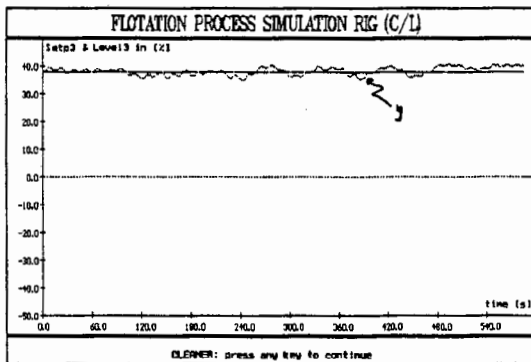
YS1



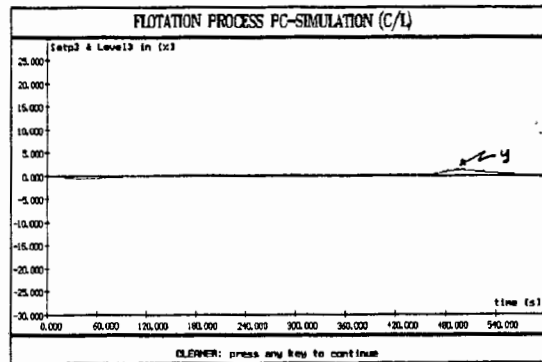
YR2



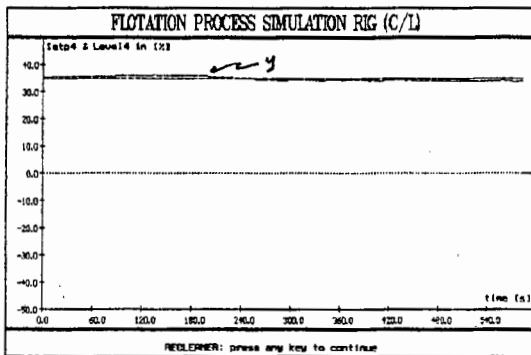
YS2



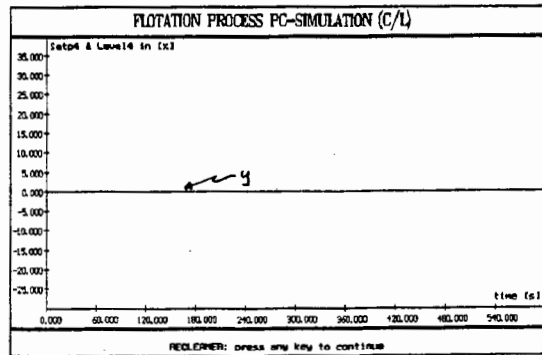
YR3



YS3

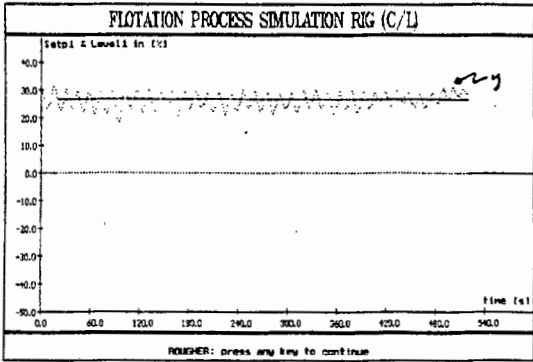


YR4

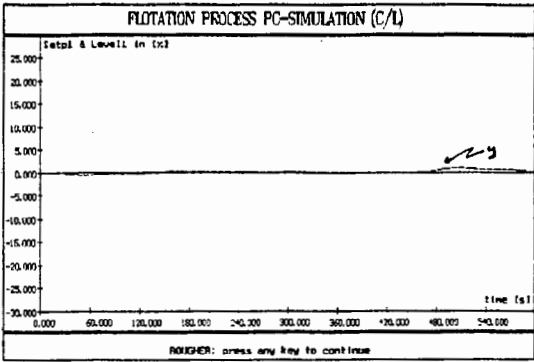


YS4

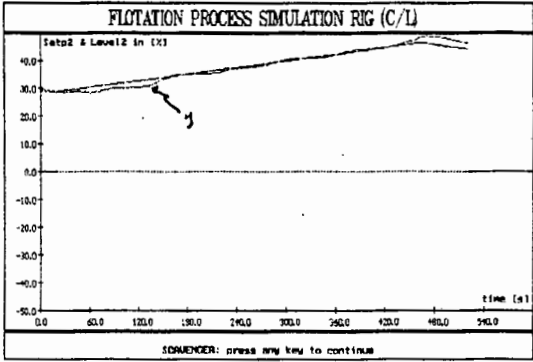
Figure 6.17 O/ps, y_R [$r_1(s)$ ramp +4%]Figure 6.18 O/ps, y_S [$r_1(s)$ ramp +4%]



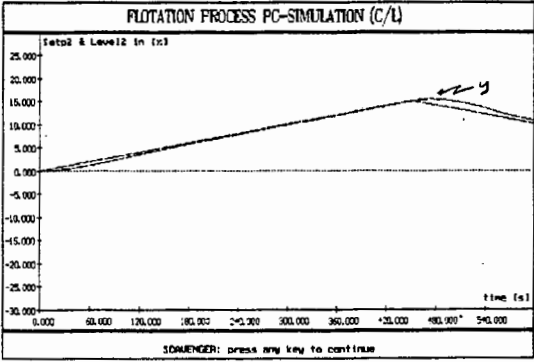
YR1



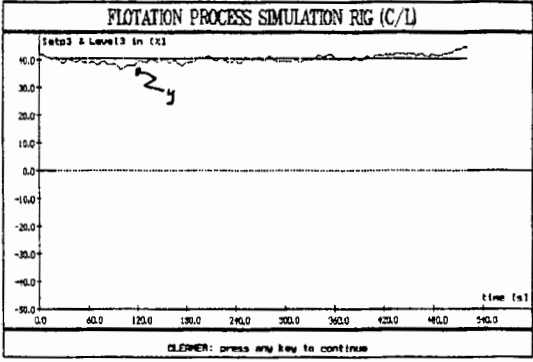
YS1



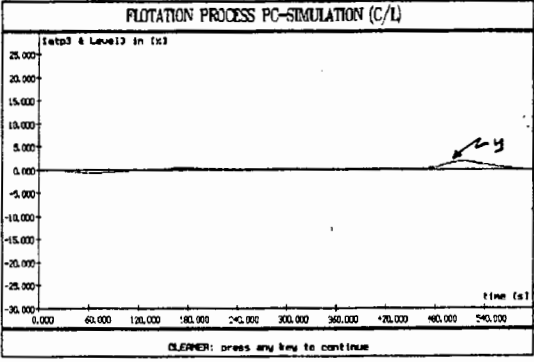
YR2



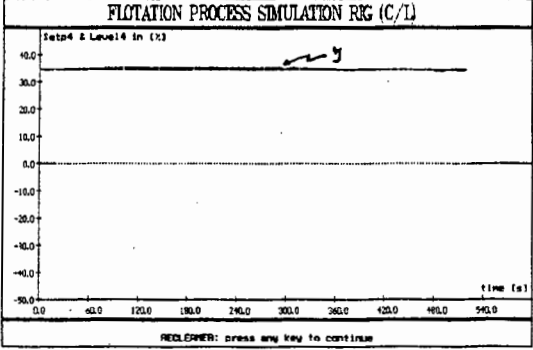
YS2



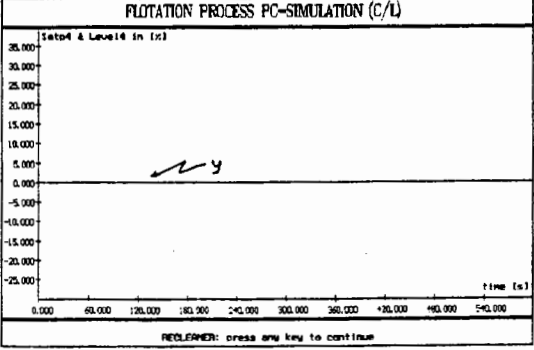
YR3



YS3



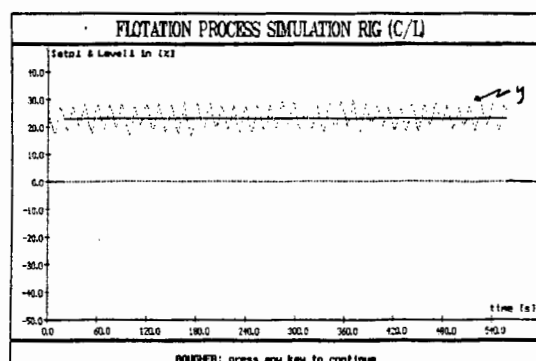
YR4



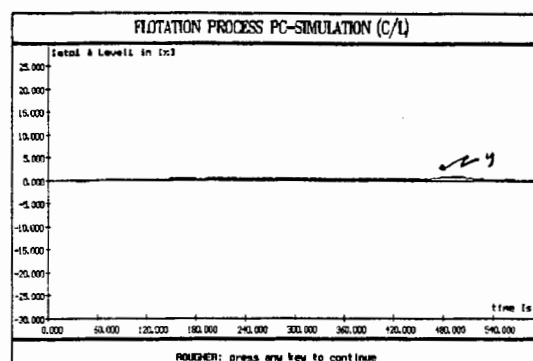
YS4

Figure 6.19 0/ps, y_R [$r_2(s)$ ramp +4%]

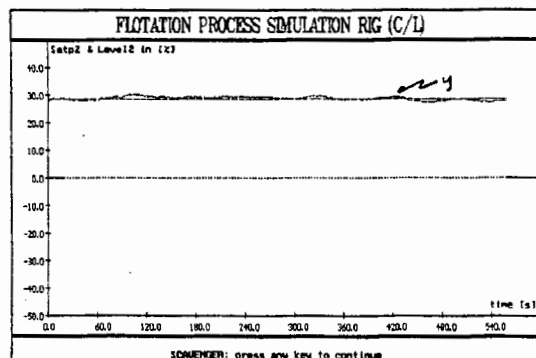
Figure 6.20 0/ps, y_S [$r_2(s)$ ramp +4%]



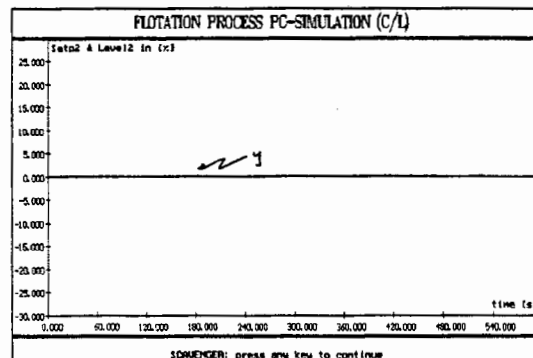
YR1



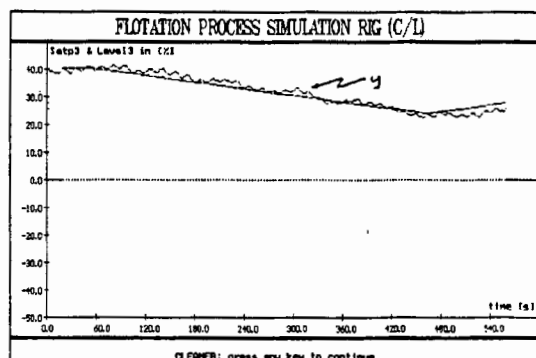
YS1



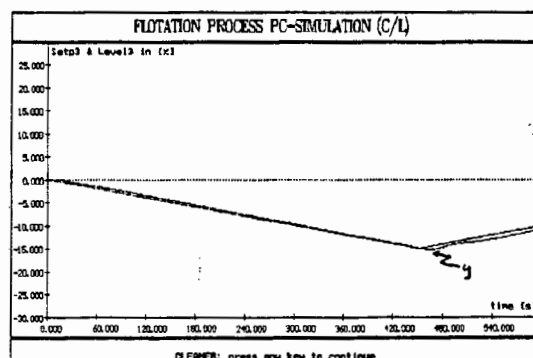
YR2



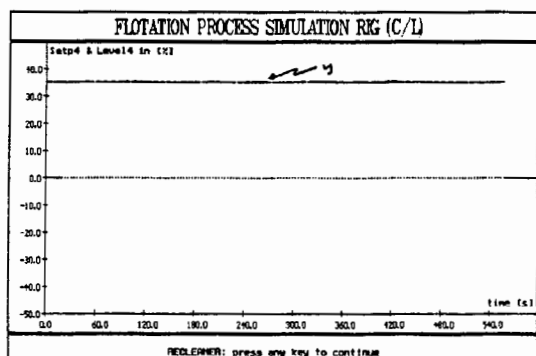
YS2



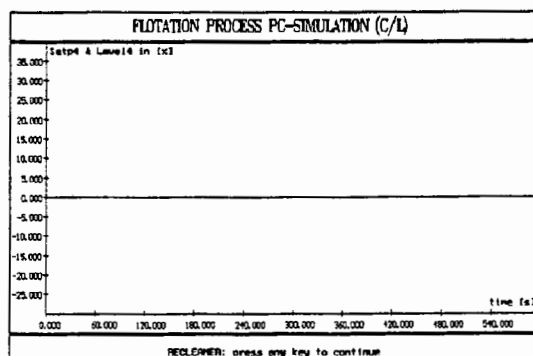
YR3



YS3

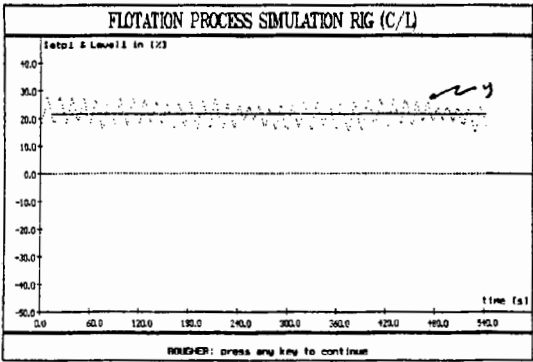


YR4

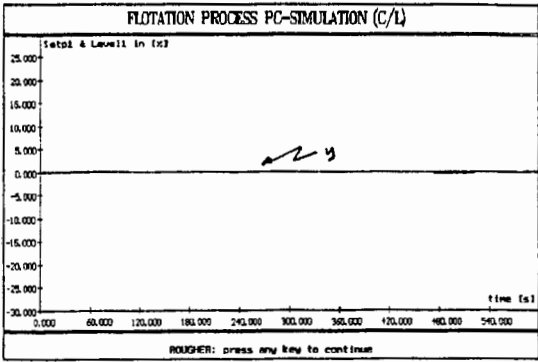


YS4

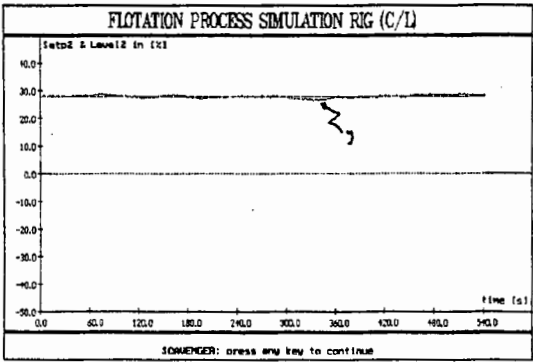
Figure 6.21 O/ps, y_R [$r_3(s)$ ramp -4%]Figure 6.22 O/ps, y_S [$r_3(s)$ ramp -4%]



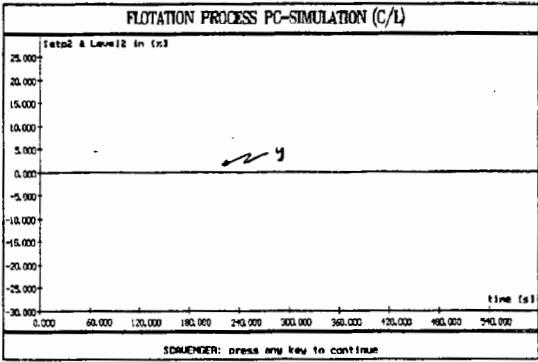
YR1



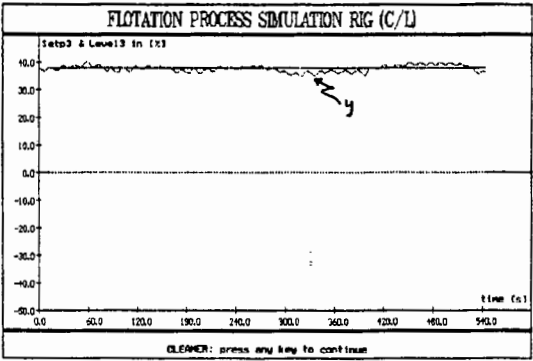
YS1



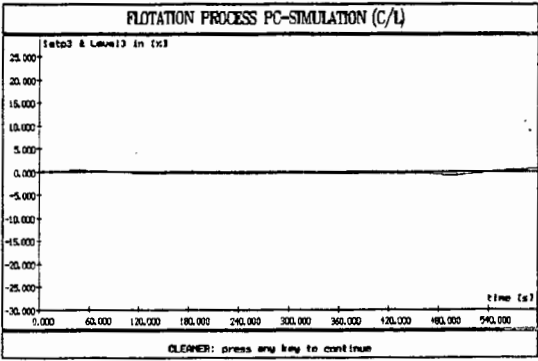
YR2



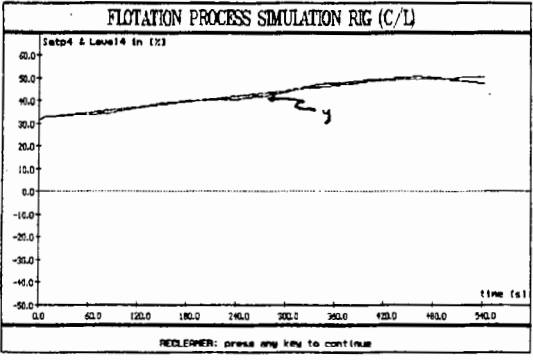
YS2



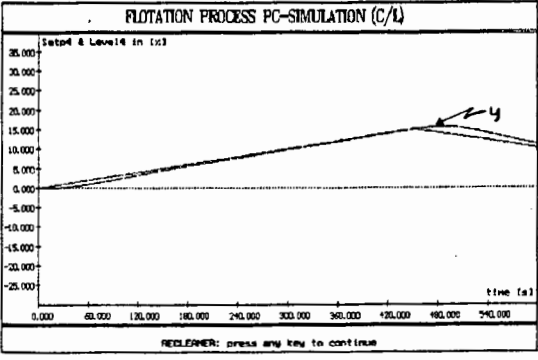
YR3



YS3



YR4



YS4

Figure 6.23 O/ps, y_R [$r_4(s)$ ramp +4%]

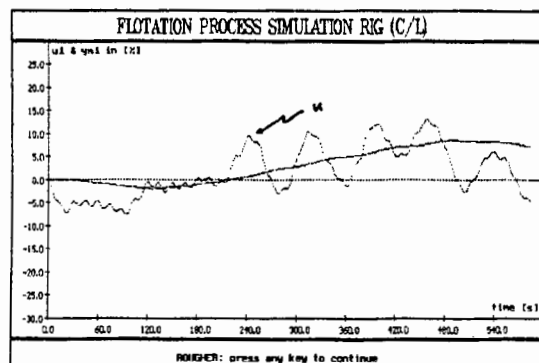
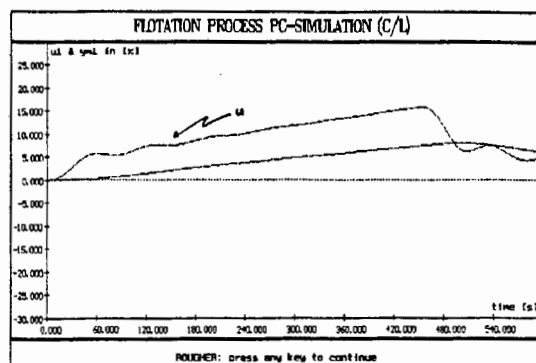
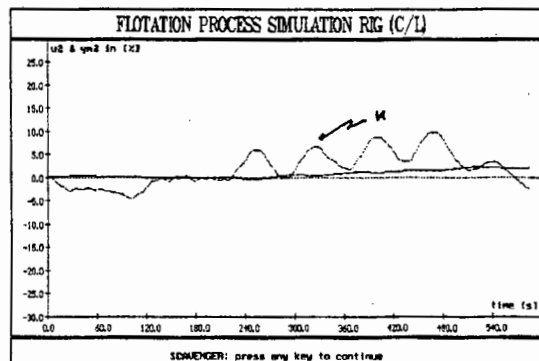
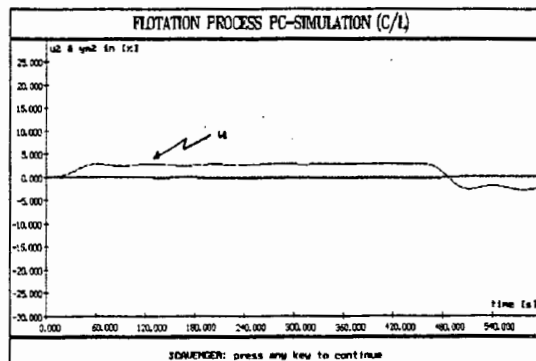
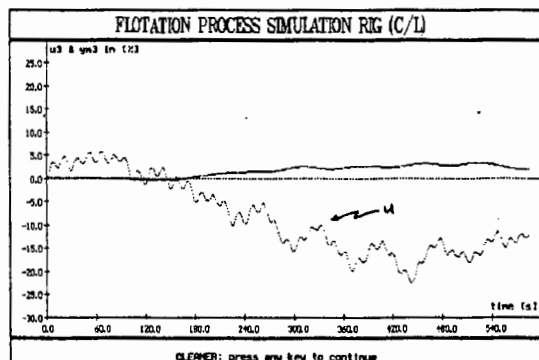
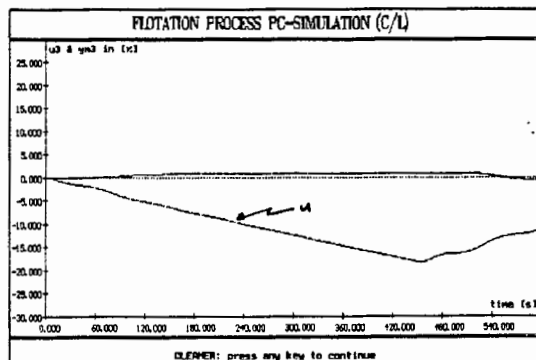
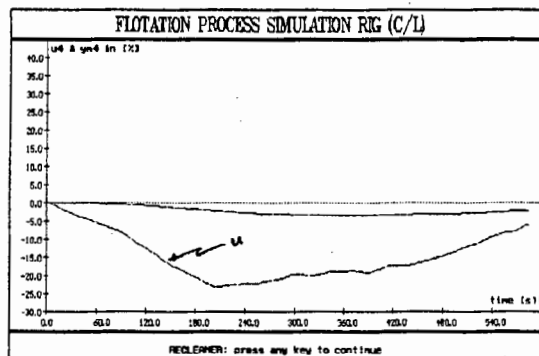
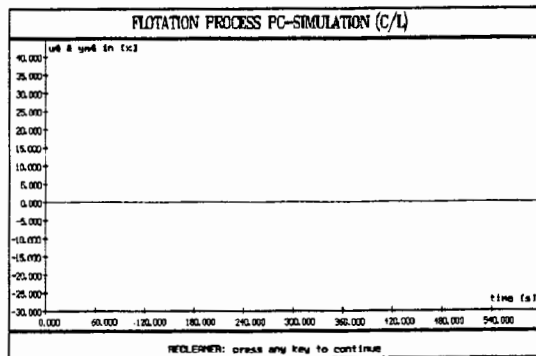
Figure 6.24 O/ps, y_S [$r_4(s)$ ramp +4%]

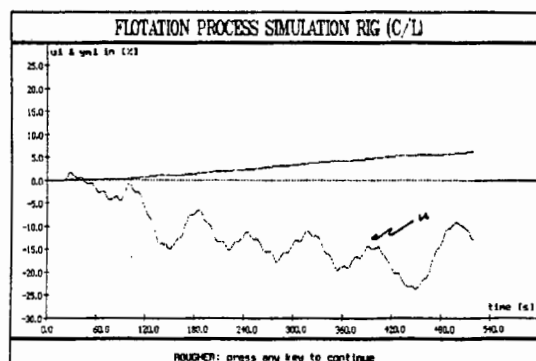
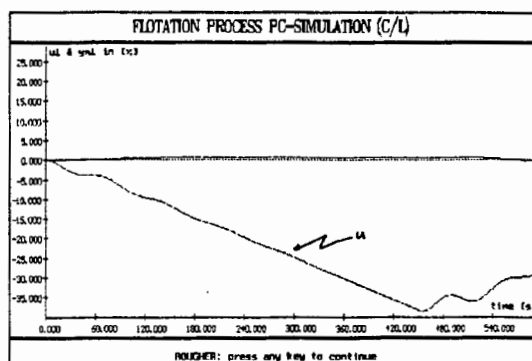
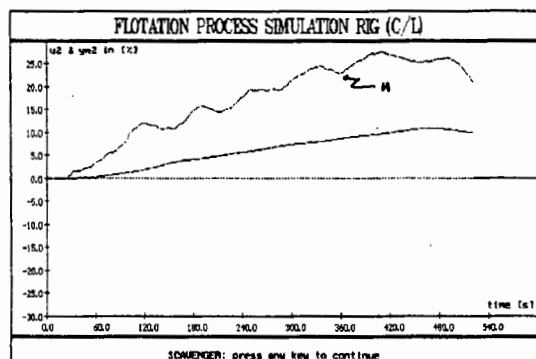
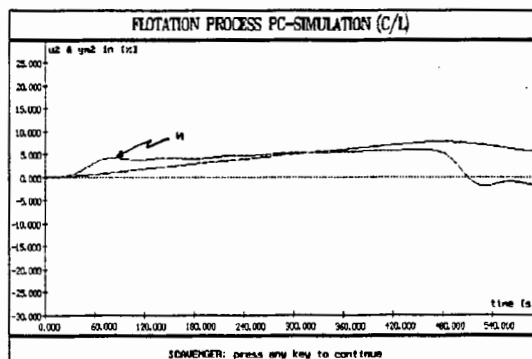
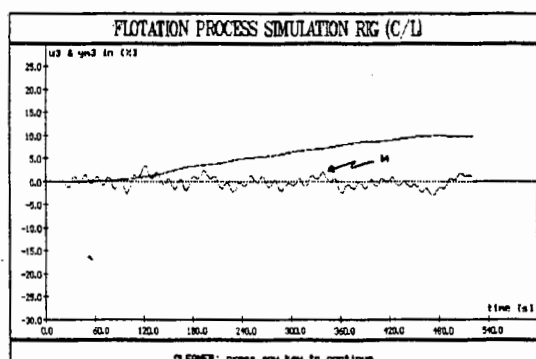
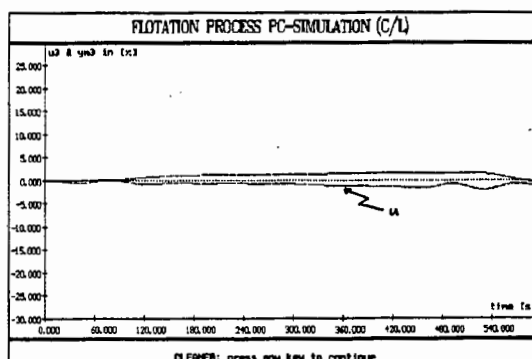
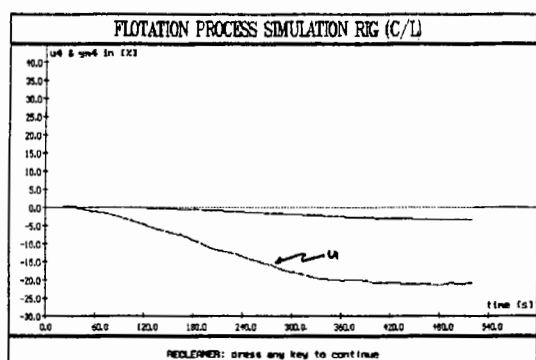
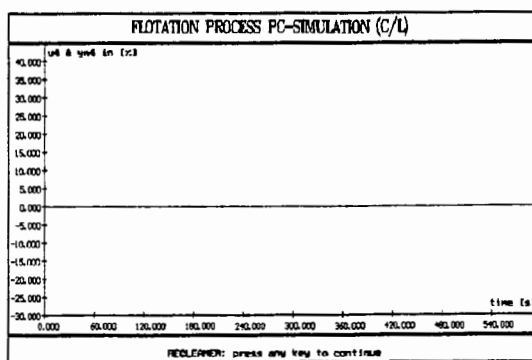
6.2.2 Comparison of Process I/P and Model O/P (RAMP)

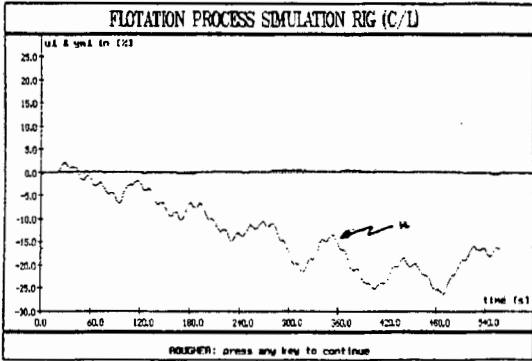
The graphs on the next four pages are given in matrix form. Every page contains two columns. The graphs in column one, show the responses of the FLOTRIG process inputs and model outputs. Column two, shows the PC-Simulated process inputs and model outputs.

They are:-

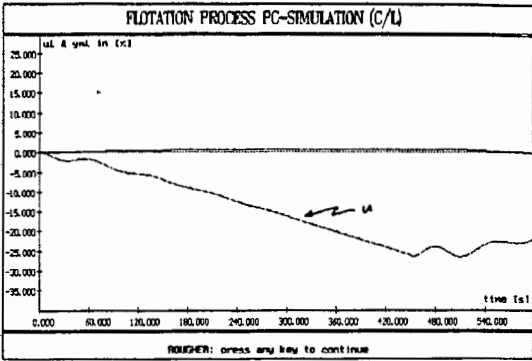
- Process inputs (the lines that show rapid changes compared other lines on the same plot)
- Model outputs (smooth continuous lines)

 $u_{11} \text{ \& } y_{m11} \text{ (R)}$  $u_{11} \text{ \& } y_{m11} \text{ (S)}$  $u_{21} \text{ \& } y_{m21} \text{ (R)}$  $u_{21} \text{ \& } y_{m21} \text{ (S)}$  $u_{31} \text{ \& } y_{m31} \text{ (R)}$  $u_{31} \text{ \& } y_{m31} \text{ (S)}$  $u_{41} \text{ \& } y_{m41} \text{ (R)}$  $u_{41} \text{ \& } y_{m41} \text{ (S)}$ Figure 6.25 $u \text{ \& } y_m [r_1(s) \text{ ramp } +4\%]$ Figure 6.26 $u \text{ \& } y_m [r_1(s) \text{ ramp } +4\%]$

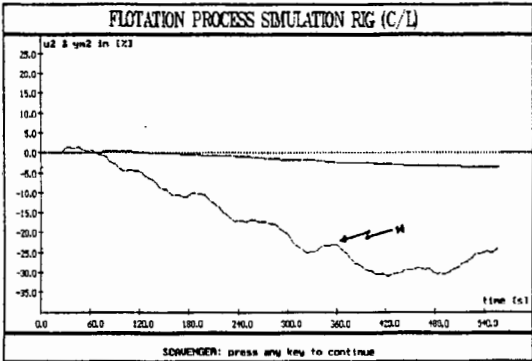
 $u_{12} & y_{m12} (R)$  $u_{12} & y_{m12} (S)$  $u_{22} & y_{m22} (R)$  $u_{22} & y_{m22} (S)$  $u_{32} & y_{m32} (R)$  $u_{32} & y_{m32} (S)$  $u_{42} & y_{m42} (R)$  $u_{42} & y_{m42} (S)$ Figure 6.27 $u & y_m [r_2(s) \text{ ramp } +4\%]$ Figure 6.28 $u & y_m [r_2(s) \text{ ramp } +4\%]$



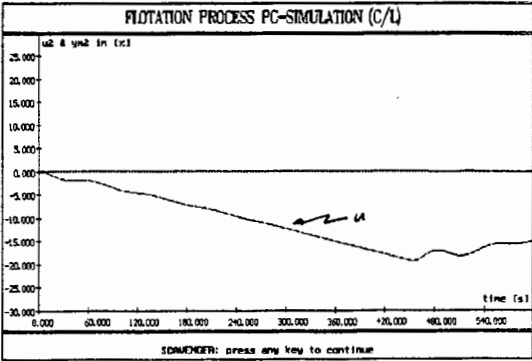
u_{13} & y_{m13} (R)



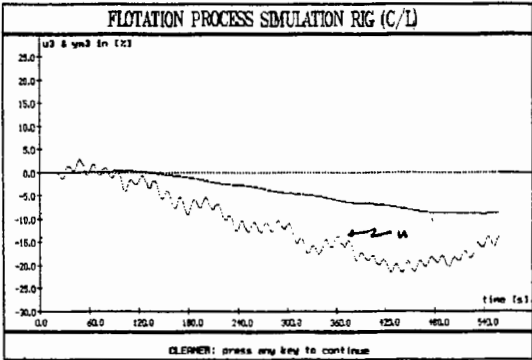
u_{13} & y_{m13} (S)



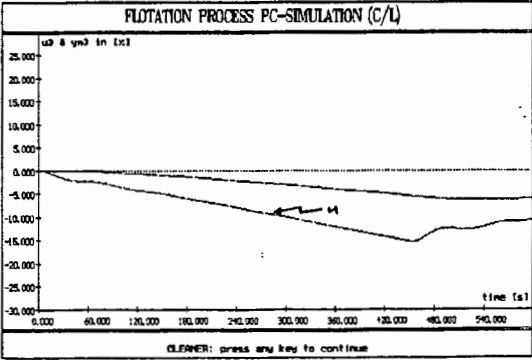
u_{23} & y_{m23} (R)



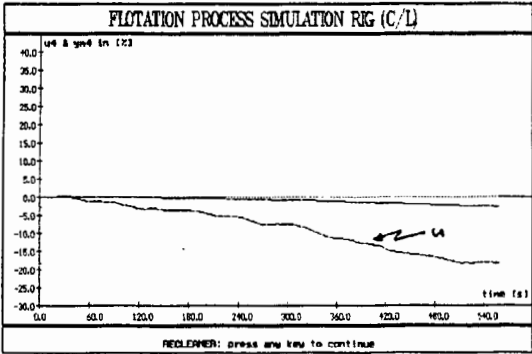
u_{23} & y_{m23} (S)



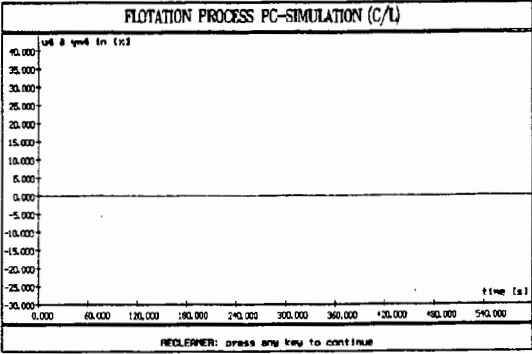
u_{33} & y_{m33} (R)



u_{33} & y_{m33} (S)



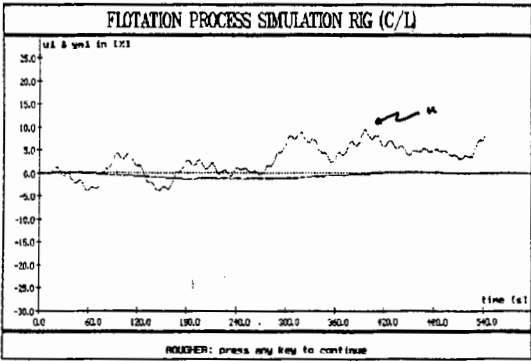
u_{43} & y_{m43} (R)



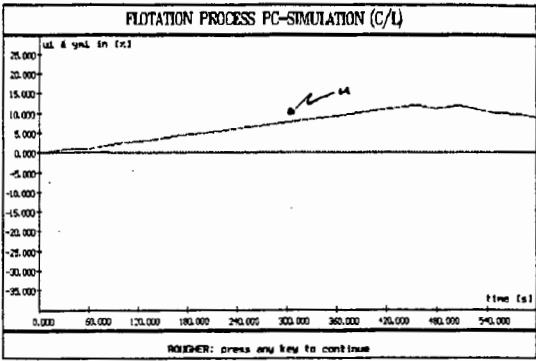
u_{43} & y_{m43} (S)

Figure 6.29 u & y_m [$r_3(s)$ ramp -4%]

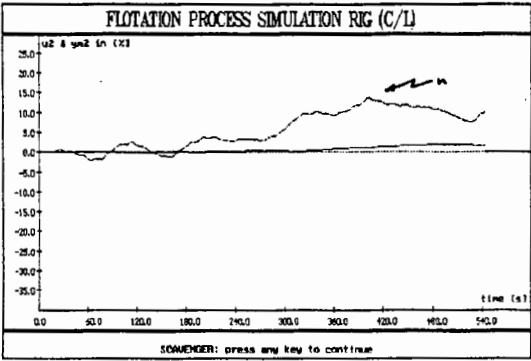
Figure 6.30 u & y_m [$r_3(s)$ ramp -4%]



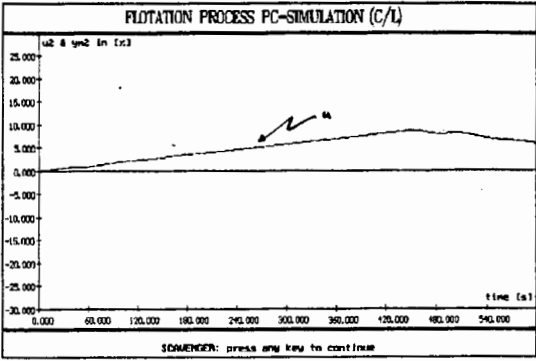
u_{14} & y_{m14} (R)



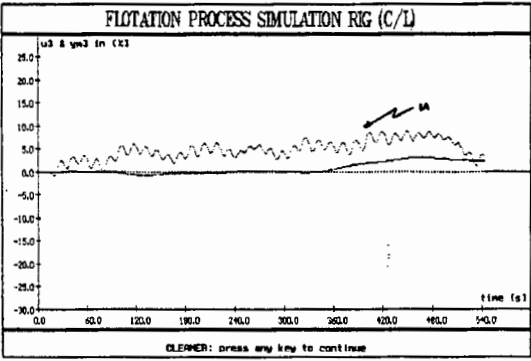
u_{14} & y_{m14} (S)



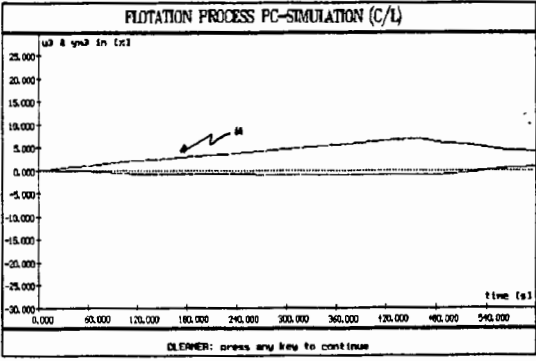
u_{24} & y_{m24} (R)



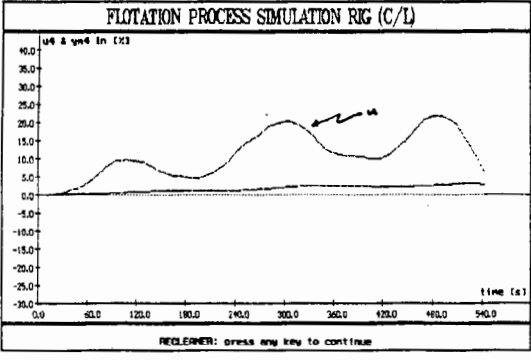
u_{24} & y_{m24} (S)



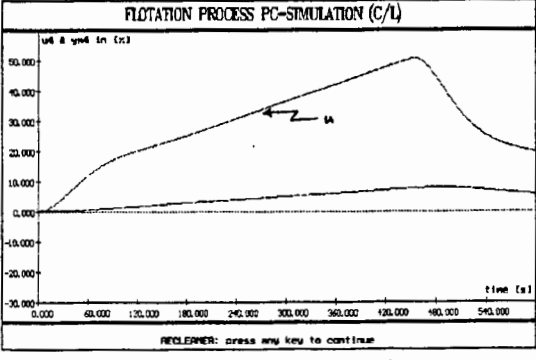
u_{34} & y_{m34} (R)



u_{34} & y_{m34} (S)



u_{44} & y_{m44} (R)



u_{44} & y_{m44} (S)

Figure 6.31 u & y_m [$r_4(s)$ ramp +4%]

Figure 6.32 u & y_m [$r_4(s)$ ramp +4%]

6.3 Comparison of Results for non-tracked Ramp Setpoints

In the following two subsections, the results obtained from the FLOTRIG and the PC-Simulation are presented.

6.3.1 Comparison of Process Outputs (NO-RAMP)

The graphs on the next four pages are given in matrix form. Every page contains two columns. The graphs in column one, show the responses of the FLOTRIG process outputs. Column two, shows the PC-Simulated process outputs.

For this case, the Transfer Function Matrices that form the controller are the same as the matrices designed for step setpoint tracking.

In the following graphs for ramp setpoints, the notation below is used:

ramp +4% OR ramp -4%

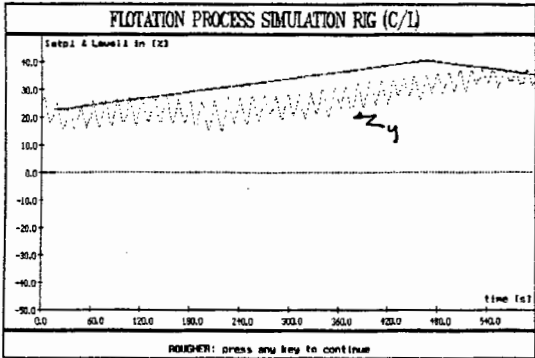
This denotes a slope of 4 percent obtained as follows:

$$\begin{aligned} \text{ramp slope} &= \frac{+ 20[\%]}{500[\text{sec}]} = + 0.04 [\% / \text{sec}] \\ &= + 4\% \end{aligned}$$

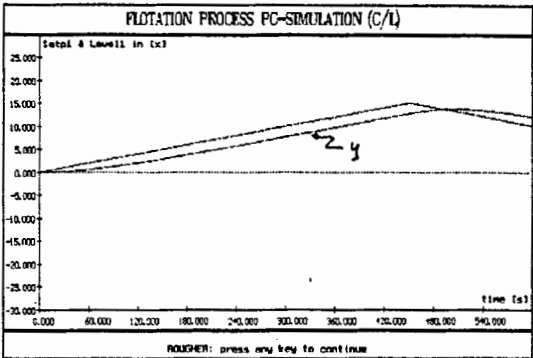
similarly for -4%

They process outputs and setpoints are distinguishable as:-

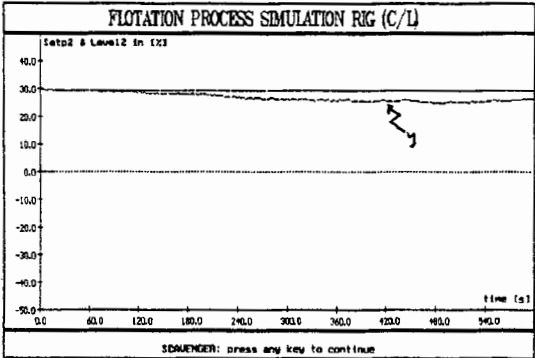
- Process outputs (noisy line for FLOTRIG results and smooth line for Simulation results)
- Setpoints (straight solid lines)



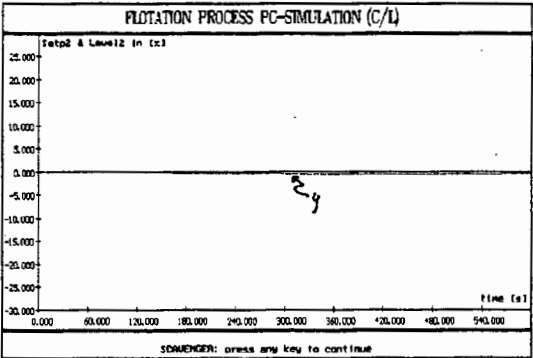
YR1



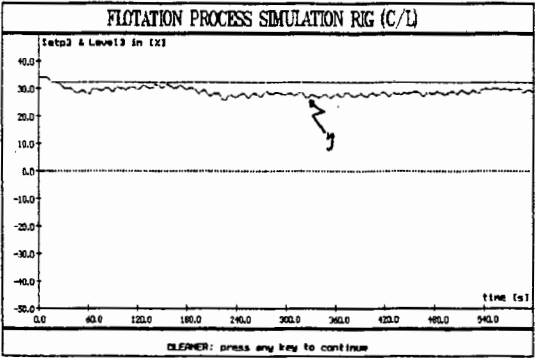
YS1



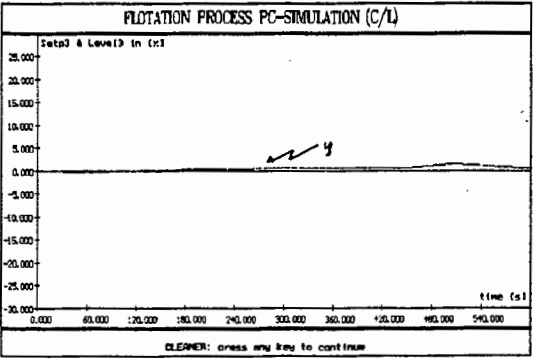
YR2



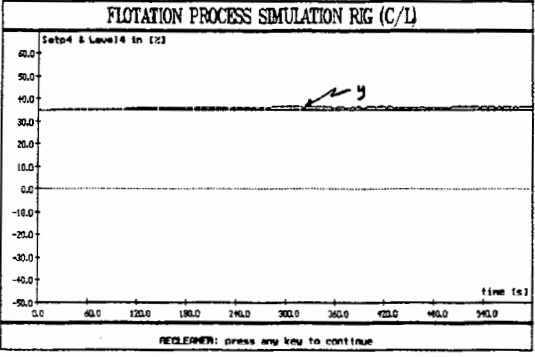
YS2



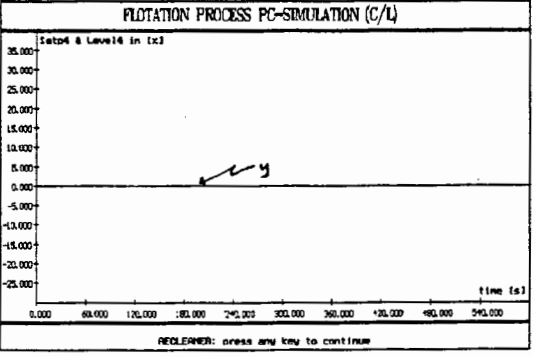
YR3



YS3



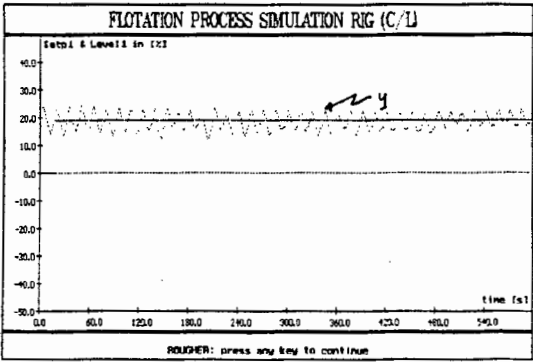
YR4



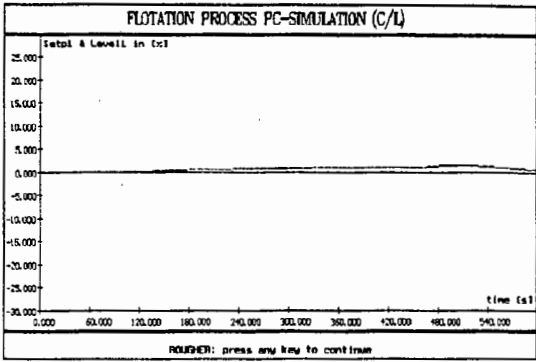
YS4

Figure 6.33 O/ps, $y_R [r_1(s) \text{ ramp } +4\%]$

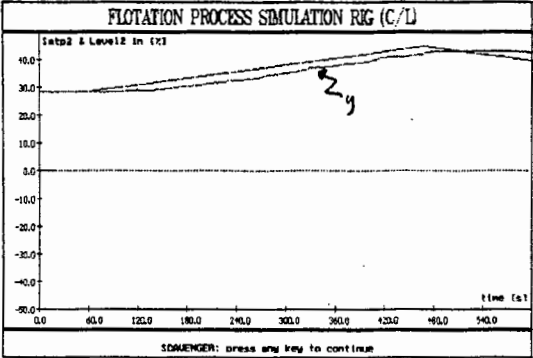
Figure 6.34 O/ps, $y_S [r_1(s) \text{ ramp } +4\%]$



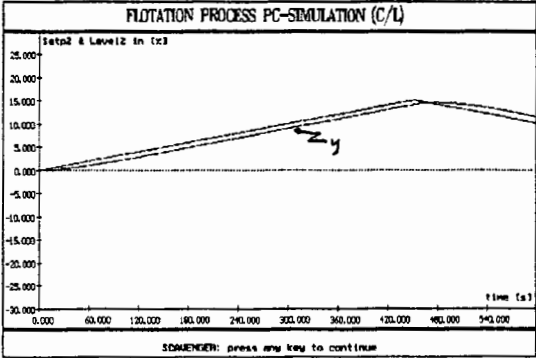
y_{R1}



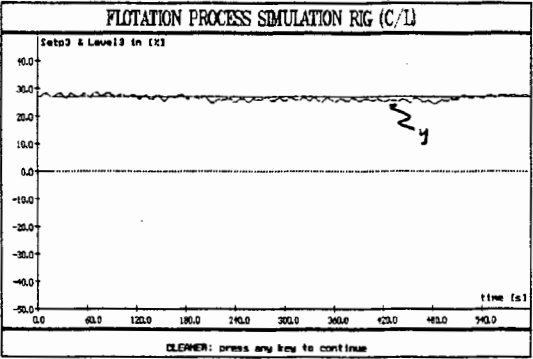
y_{S1}



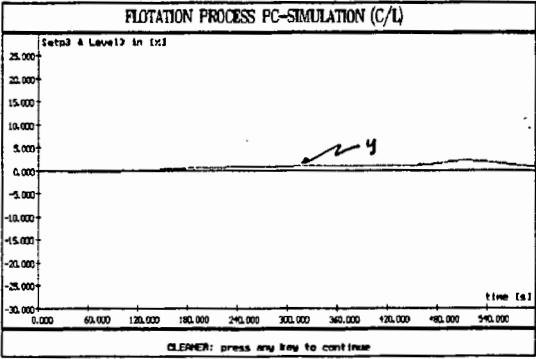
y_{R2}



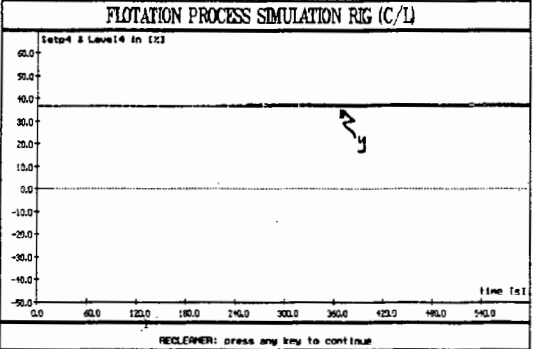
y_{S2}



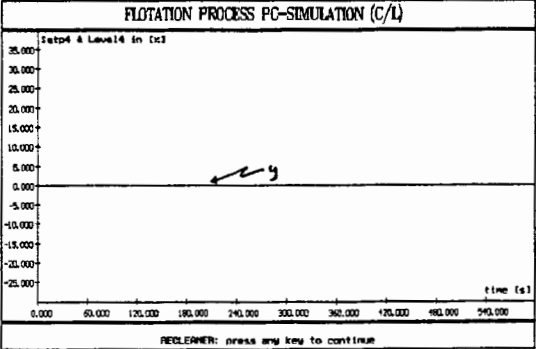
y_{R3}



y_{S3}



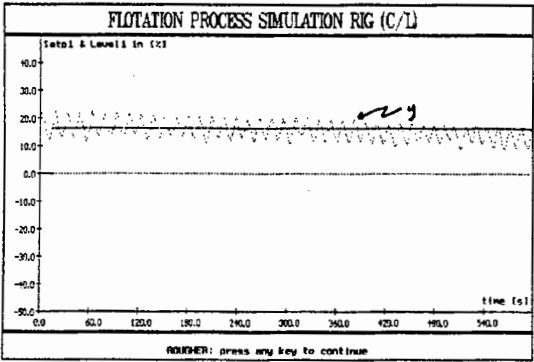
y_{R4}



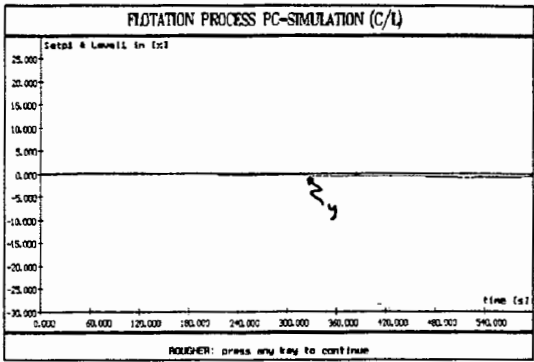
y_{S4}

Figure 6.35 0/ps, y_R [$r_2(s)$ ramp +4%]

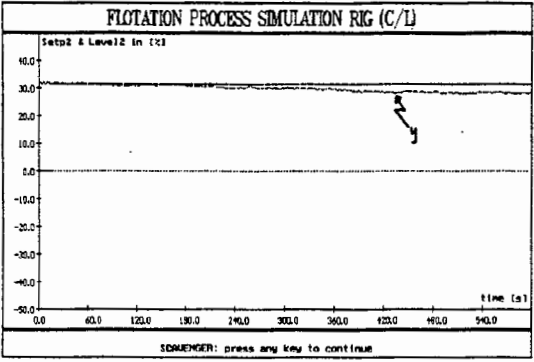
Figure 6.36 0/ps, y_S [$r_2(s)$ ramp +4%]



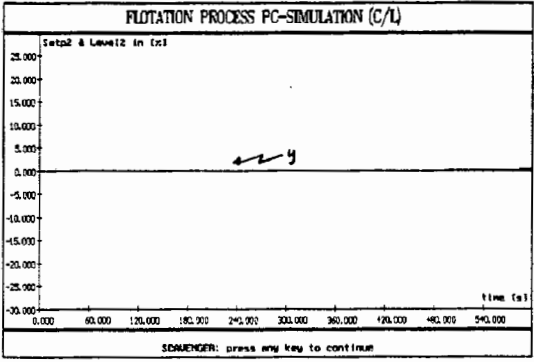
YR1



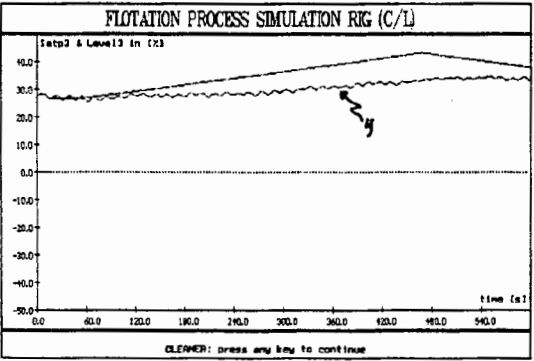
YS1



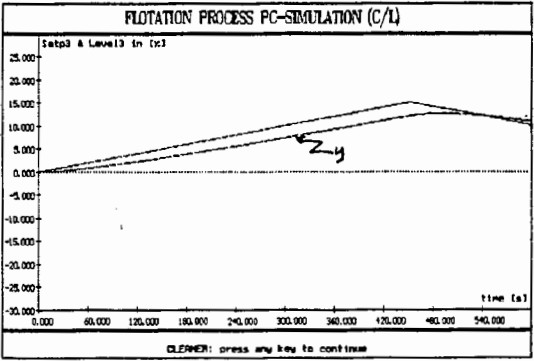
YR2



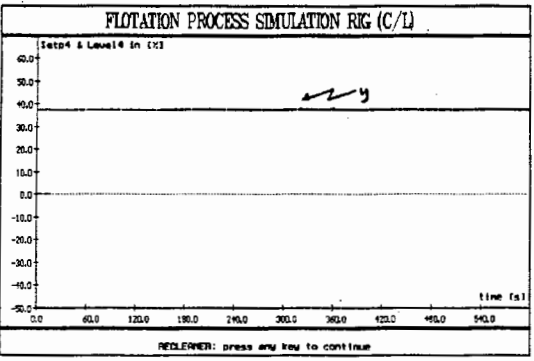
YS2



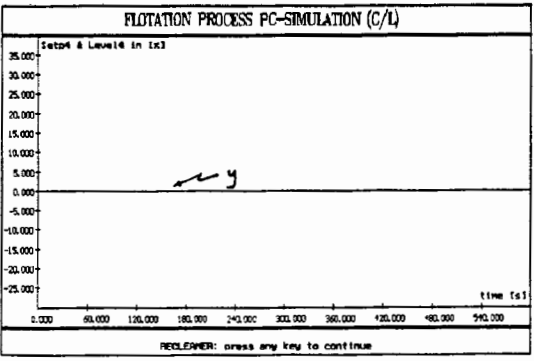
YR3



YS3



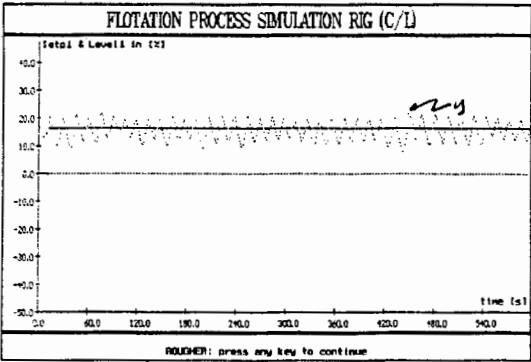
YR4



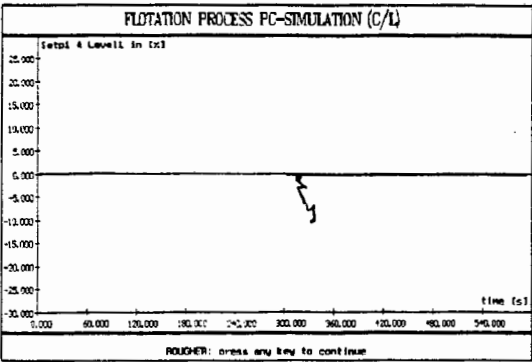
YS4

Figure 6.37 O/ps, y_R [$r_3(s)$ ramp -4%]

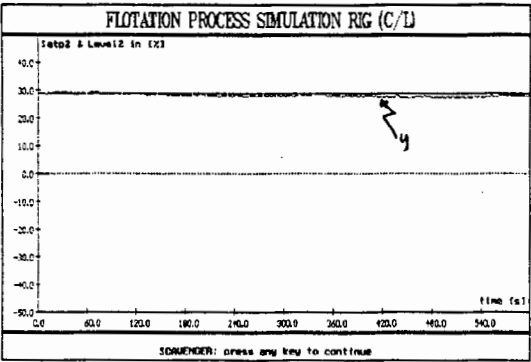
Figure 6.38 O/ps, y_S [$r_3(s)$ ramp -4%]



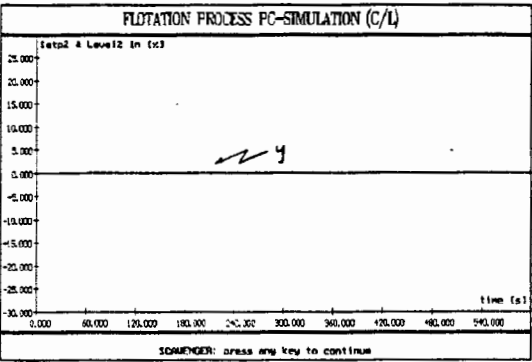
y_{R1}



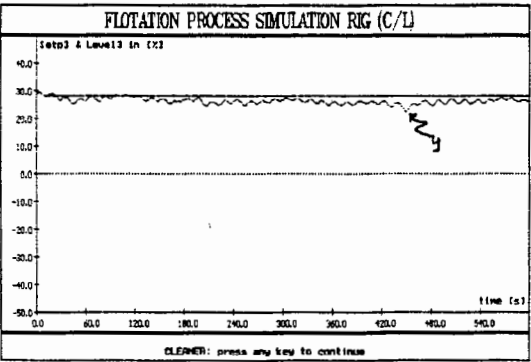
y_{S1}



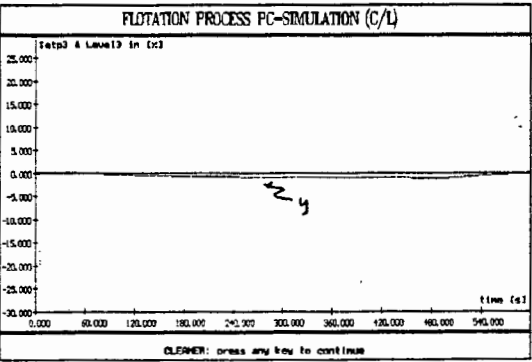
y_{R2}



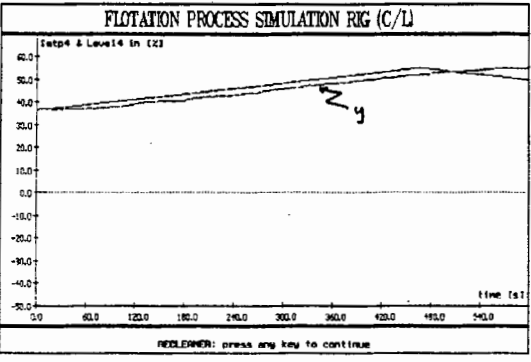
y_{S2}



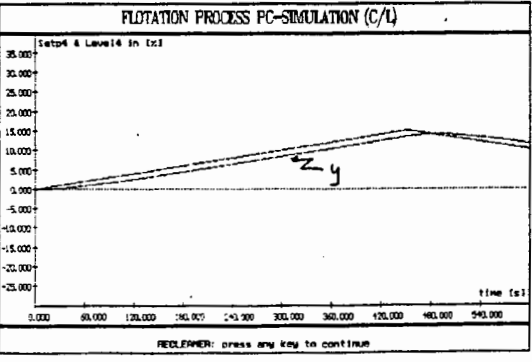
y_{R3}



y_{S3}



y_{R4}



y_{S4}

Figure 6.39 0/ps, y_R [$r_4(s)$ ramp +4%]

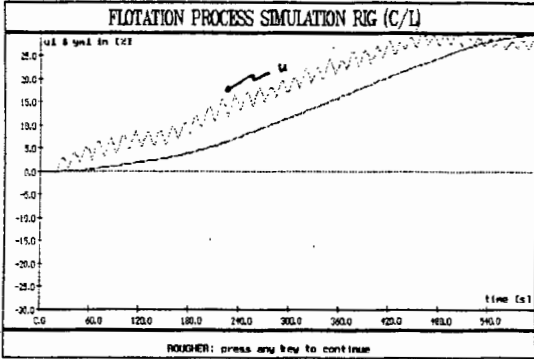
Figure 6.40 0/ps, y_S [$r_4(s)$ ramp +4%]

6.3.2 Comparison of Process I/P and Model O/P (NO-RAMP)

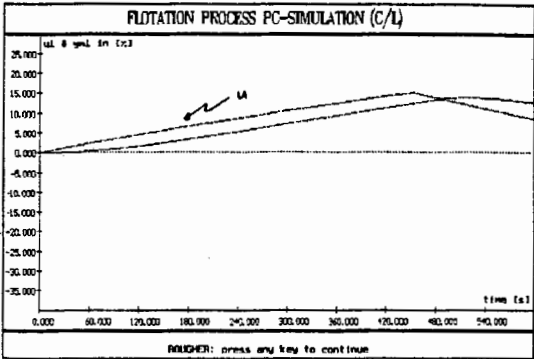
The graphs on the next four pages are given in matrix form. Every page contains two columns. The graphs in column one, show the responses of the FLOTRIG process inputs and model outputs. Column two, shows the PC-Simulated process inputs and model outputs.

They process inputs and model outputs are distinguishable as:-

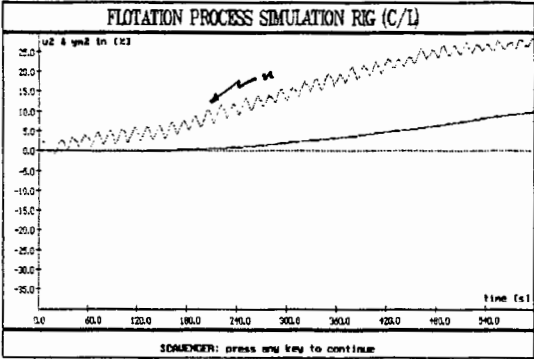
- Process inputs (the lines that show more rapid change than the other lines which are on the same plot)
- Model outputs (smooth continuous lines)



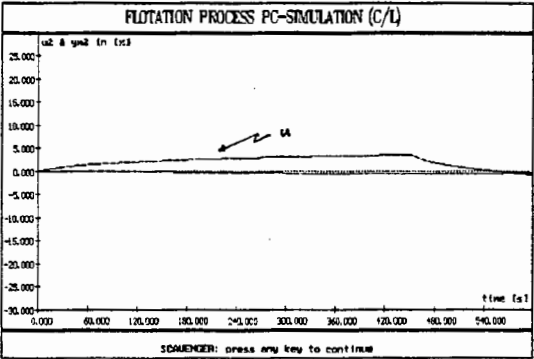
u_{11} & y_{m11} (R)



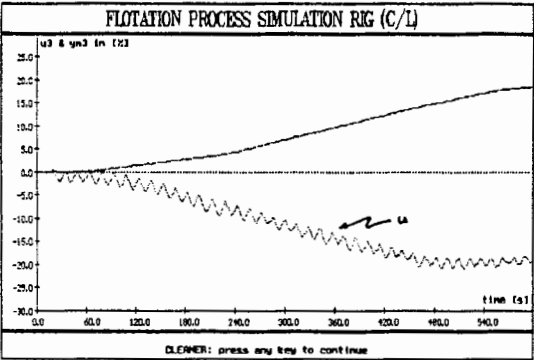
u_{11} & y_{m11} (S)



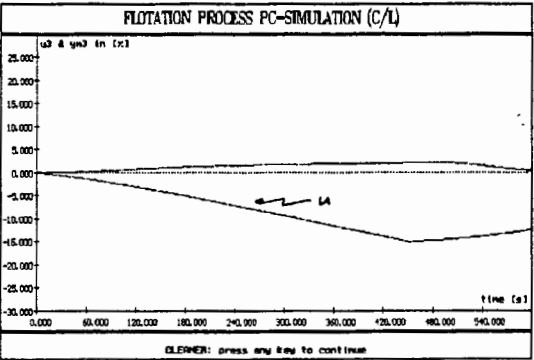
u_{21} & y_{m21} (R)



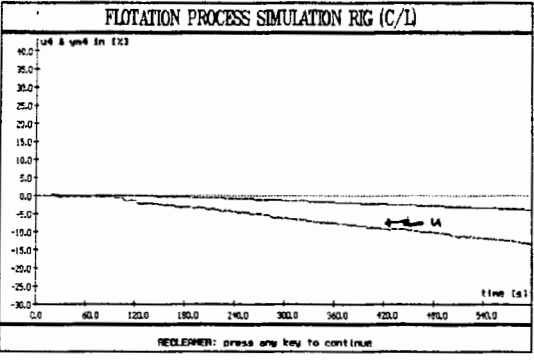
u_{21} & y_{m21} (S)



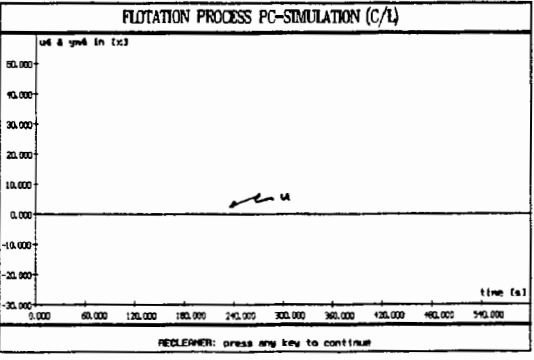
u_{31} & y_{m31} (R)



u_{31} & y_{m31} (S)



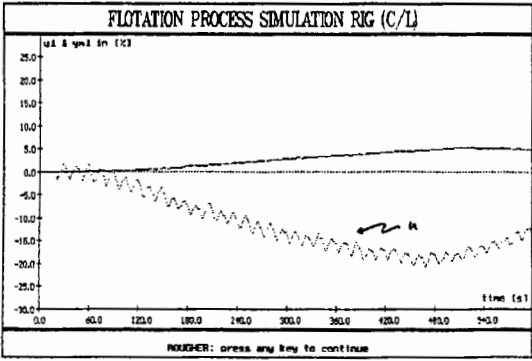
u_{41} & y_{m41} (R)



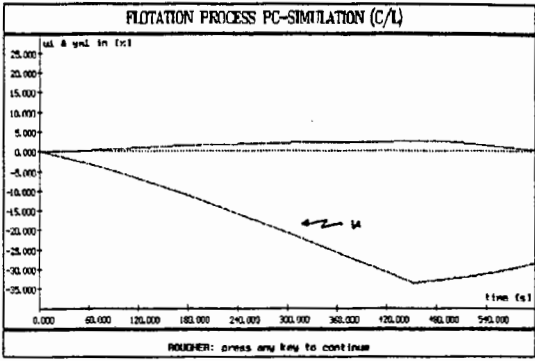
u_{41} & y_{m41} (S)

Figure 6.41 u & y_m [$r_1(s)$ ramp +4%]

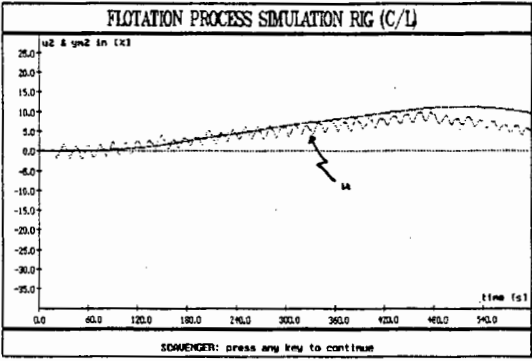
Figure 6.42 u & y_m [$r_1(s)$ ramp +4%]



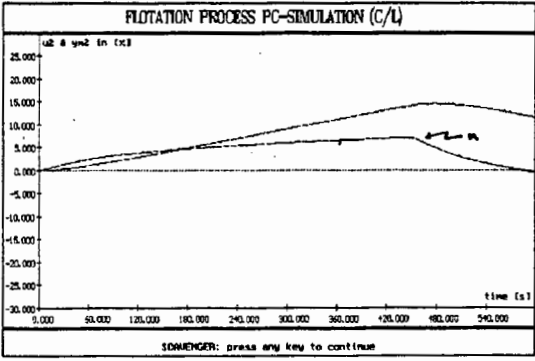
u_{12} & y_{m12} (R)



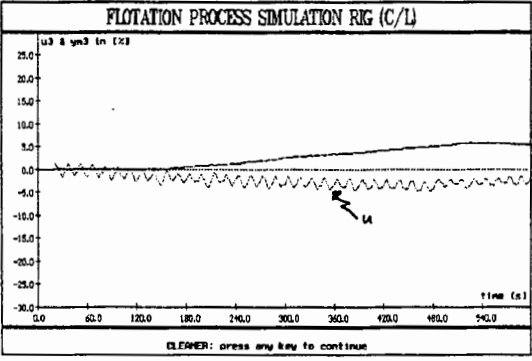
u_{12} & y_{m12} (S)



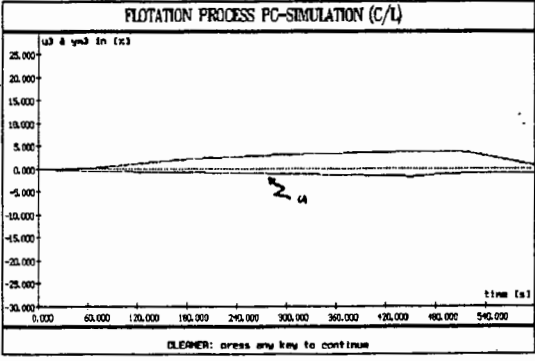
u_{22} & y_{m22} (R)



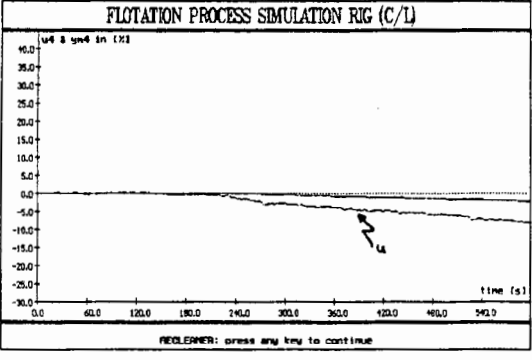
u_{22} & y_{m22} (S)



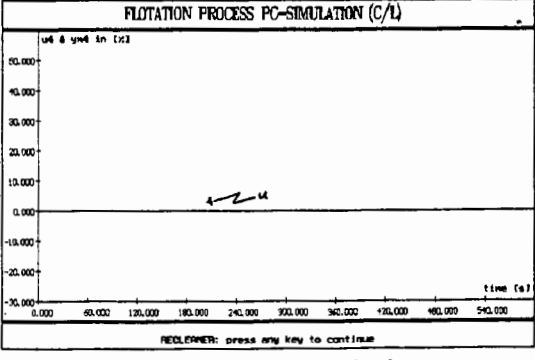
u_{32} & y_{m32} (R)



u_{32} & y_{m32} (S)



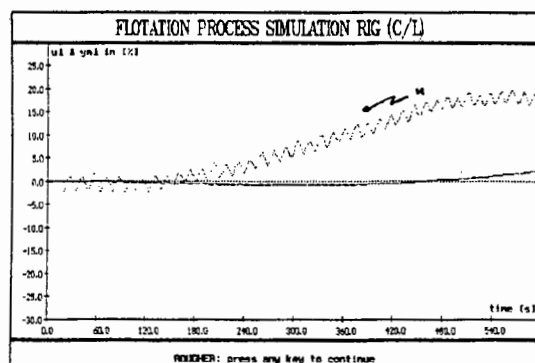
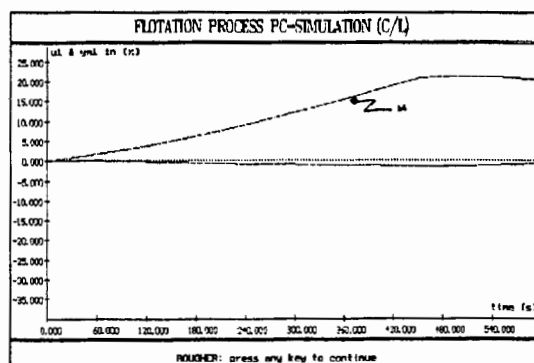
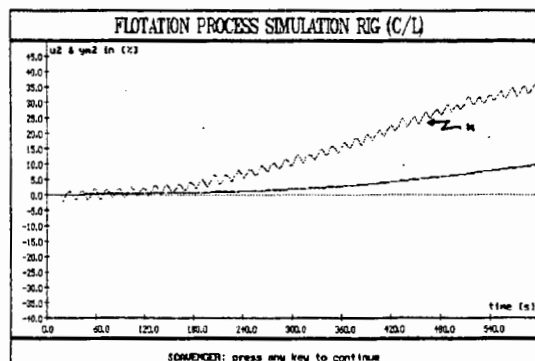
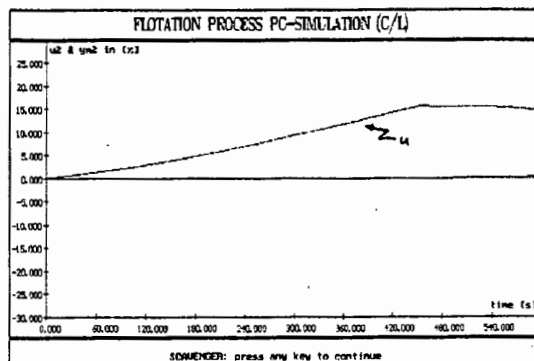
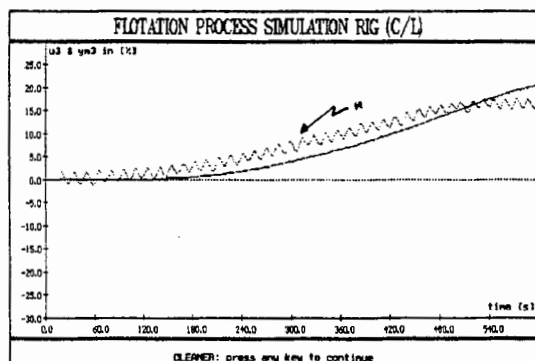
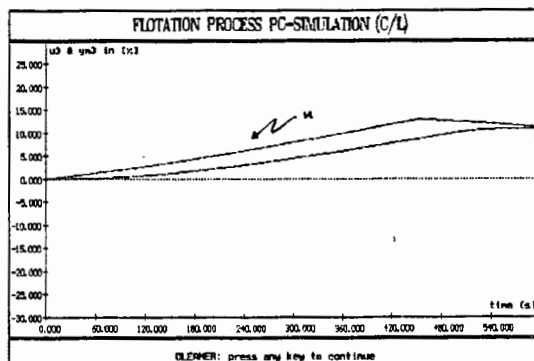
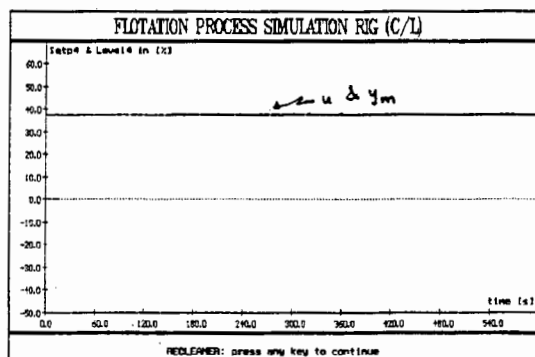
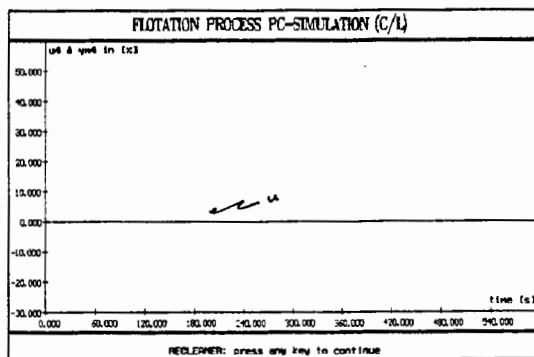
u_{42} & y_{m42} (R)

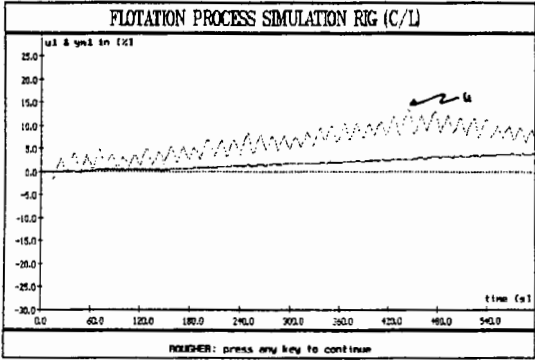


u_{42} & y_{m42} (S)

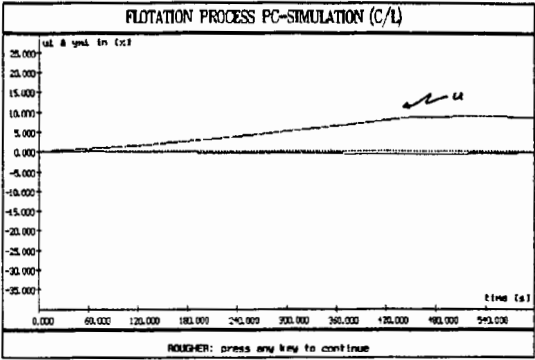
Figure 6.43 u & y_m [$r_2(s)$ ramp +4%]

Figure 6.44 u & y_m [$r_2(s)$ ramp +4%]

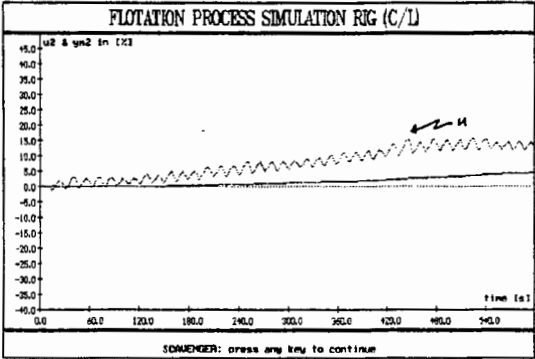
 u_{13} & y_{m13} (R) u_{13} & y_{m13} (S) u_{23} & y_{m23} (R) u_{23} & y_{m23} (S) u_{33} & y_{m33} (R) u_{33} & y_{m33} (S) u_{43} & y_{m43} (R) u_{43} & y_{m43} (S)Figure 6.45 u & y_m [$r_3(s)$ ramp +4%]Figure 6.46 u & y_m [$r_3(s)$ ramp +4%]



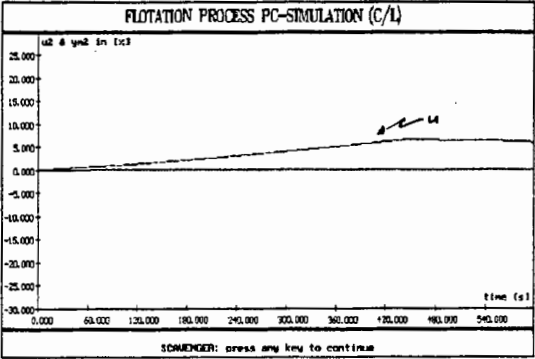
u_{14} & y_{m14} (R)



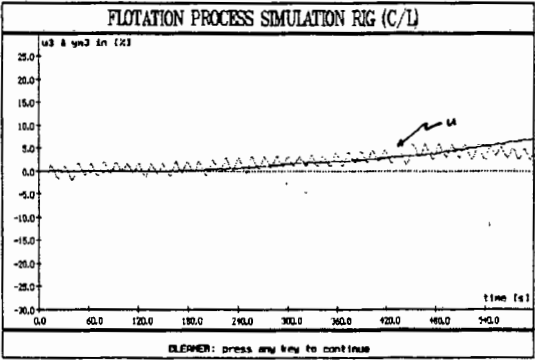
u_{14} & y_{m14} (S)



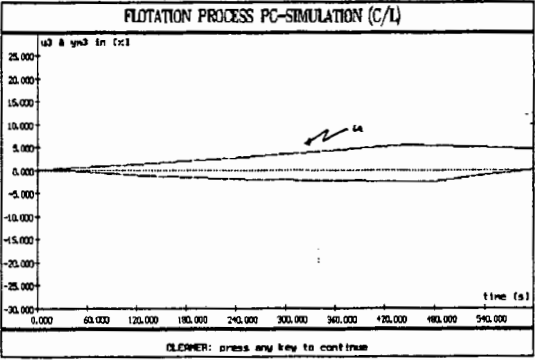
u_{24} & y_{m24} (R)



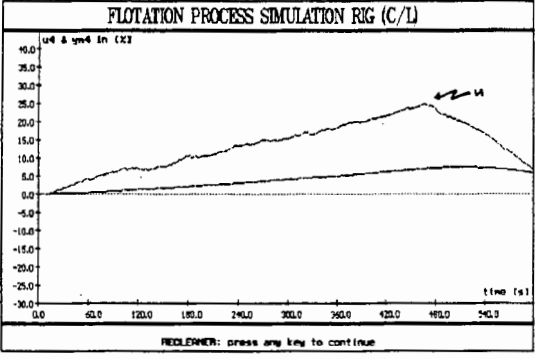
u_{24} & y_{m24} (S)



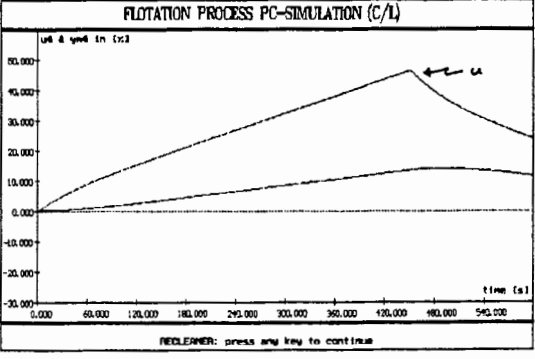
u_{34} & y_{m34} (R)



u_{34} & y_{m34} (S)



u_{44} & y_{m44} (R)



u_{44} & y_{m44} (S)

Figure 6.47 u & y_m [$r_4(s)$ ramp +4%]

Figure 6.48 u & y_m [$r_4(s)$ ramp +4%]

6.4 Observations on the results presented above

Generally, stability of the closed-loop system is noted in all the graphs. Also that the overall magnitude of the oscillations due to the feed back pump cavitations have been reduced by the presence of the controller.

6.4.1 Comments on results obtained from step setpoints

By comparison with the original open loop step tests given in Appendix A the extent to which the presence of the controller affects the original open loop system can be observed. It is easily noted that:-

- (a) the closed loop system remained stable. This is in confirms the requirements of the following conditions which will guarantee C/L stability:

- * Open Loop stability of the process
- * Stability of $Q(s)$
- * and the Plant-Model mismatch must satisfy the condition

$$\text{Re}(\text{eigen-value}_j[G(j0)M(j0)^{-1}]) > 0 \quad j = 1..n$$

This is proven in [Garcia, 1985].

- (b) the levels track their setpoints exactly. This was ensured by making $Q_0 = M(j0)^{-1}$. Note that Q_0 also decouples the process at steady state.
- (c) the overall amount of interaction is reduced even though the Q-parameter design predominantly decoupled the process at low and high frequencies.

The PC-Simulations and the FLOTRIG results agree closely with each other by considering the response magnitudes and wave shapes. The offsets which were present in the FLOTRIG process inputs have been removed for easier comparison of FLOTRIG and PC-Simulation inputs. The reason being that the PC-Simulations were done with all state initialized to zero as compared to the FLOTRIG results which had a different zero reference.

It is also noticeable that overall closed-loop responses given in 6.1 are faster than the open loop system in Appendices A and B.

The three process inputs of the FLOTRIG have offsets which are equal to the respective initial open loop valve positions for each flotation cell. It should be noted that the model, $M(s)$ inputs have no offset. The FLOTRIG offsets are:-

$$u_{1\text{plant}} = u_{1\text{model}} - 50 \quad [\%]$$

$$u_{2\text{plant}} = u_{2\text{model}} - 50 \quad [\%]$$

$$u_{3\text{plant}} = u_{3\text{model}} - 50 \quad [\%]$$

$$u_{4\text{plant}} = u_{4\text{model}} \quad [\%]$$

It is also noted that the control action is smooth except for the superimposed effects of the feedback pump cavitations which are more visible on the cell that are in direct connection with the Rougher cell. In addition the

process inputs do not hit limits (+100% and -100%) even if the initial valve position offsets are added

6.4.2 Comments on the results obtained from Ramp Setpoints

The comments given above for results obtained from step setpoints also hold for ramp setpoints. An additional observation is that, the $Q(s)$ designed for ramp tracking does not contain high frequency decoupling, but the amount of high frequency interaction is also minimal. The steady state errors on all the PC-Simulation and FLOTRIG results are zero. The FLOTRIG and PC-simulation results also compare closely.

6.4.3 Comments on results obtained from Ramp Setpoints where $Q(s)$ was designed for Step Setpoint Tracking

As expected, the process outputs do not track the ramp setpoints, which agrees with the theoretical expectations of the design of $Q(s)$.

Even in the presence of steady state errors for ramp setpoints, the closed-loop system still remained stable.

Although the interaction is minimal, the responses in the off-diagonal elements are not zero, since they are also

affected by the on-diagonal elements which do not track the ramp setpoints.

Once more, the FLOTRIG results closely correlated with the PC-Simulation results.

6.4.4 Comparison of results to previous INA and CL designs on the FLOTRIG

In this subsection the simulation and FLOTRIG results obtained in section 6.1 are compared to those obtained using INA and CL control system design methods for the FLOTRIG [Fisher, 1988].

The comparisons are done using INA and CL-Method tables given in [Fisher, 1988] and the table compiled from step test results in 6.1.

Table 6.1 for results obtained using the Characteristic Loci Method is given below.

Table 6.1: Simulated and actual FLOTRIG results - CL method

Cell stepped	Simulated Response (SR) [sec]	Actual Response (AR) [sec]	AR/SR ratio
Rougher	18	20	1.11
Scavenger	10	100	10.00
Cleaner	10	25	2.50
Recleaner	25	25	1.00

The second comparisons for results obtained using the INA-Method on the FLOTIG are given below in table 6.2.

Table 6.2: Simulated and actual FLOTIG results - INA-Method

Cell stepped	Simulated Response (SR) [sec]	Actual Response (AR) [sec]	AR/SR ratio
Rougher	15	20	1.33
Scavenger	17	100	5.88
Cleaner	10	20	2.00
Recleaner	30	20	0.66

Table 6.3 below contains results obtained from the FLOTIG and simulations using the proposed simplifying design(PSD) method.

Table 6.3: Simulated and actual FLOTIG results - PSD method

Cell stepped	Simulated Response (SR) [sec]	Actual Response (AR) [sec]	AR/SR ratio
Rougher	160	150	0.94
Scavenger	140	130	0.93
Cleaner	240	250	1.04
Recleaner	180	60	0.33

It is noted that the AR/SR ratios for the simplified design approach show that the actual results correspond closely to those expected from simulations. The only large discrepancy is that of the Recleaner cell ratio. It shows that the

actual response time is less than expected. That is because of the nonlinearity in the Recleaner *tail out flow* actuator which opens faster than it closes for the same step down setpoint change than for a step up.

Finally, in order to show the correlation between simulated and actual responses using the three design methods, their respective discrepancies are compared in table 6.4 below in percentage using the AR/SR ratios determined in tables 6.1 to 6.3 above. The discrepancies were calculated as follows

$$[\%] \text{ discrepancy} = \left[\frac{(\text{AR/SR}) - 1}{1} \right] * 100\%$$

Thus, negative discrepancies are for responses that were faster than expected and positive ones for results slower than expected.

Table 6.4: Comparison of discrepancies in [%]

Cell name	PSD [%]	INA [%]	CL [%]
Rougher	-6	33	11
Scavenger	-7	488	900
Cleaner	4	100	150
Recleaner	-67	-34	0

From the table above it is clear that 75% of the proposed design(PSD) method results correlate closely to those simulated except for the Recleaner cell.

By comparing the percentage discrepancies of the INA and CL methods to those of the PSD discrepancies it is clear that the proposed design method predicts the actual responses better than the INA and CL methods. This can be explained in the as follows:

The proposed method, which is based on Q-Parametrization, provides the designer with sufficient information at the design stages in order to achieve design objectives in a simplistic manner using open loop information inherent in the IMC-Structure.

The Characteristic Loci(CL) method does not provide the designer with information on the origin of a problem, although it provides important information on assessment of system stability, because the designer generally is unable to visualize what the loci represent in the plant [Fisher, 1988].

The Inverse Nyquist Array(INA) method is very useful for achieving diagonal dominance in a systematic manner although considerable experience is required to design a close to desired precompensator. Similar to the CL method, the INA

technique does not provide direct information to the designer on how the design affects the process [Fisher, 1988].

Chapter Seven

Conclusions

In view of the preceding chapters the following conclusions can be drawn:

7.1 Summarizing conclusion on the literature survey

After consulting the literature on Q(Youla)-Parametrization and its special case, the IMC Design method, it was found that two out of more than twenty publications that relate to the controller parametrization covered a SISO and 2x2 MIMO system for experimental studies. This thesis provides considerable insight into the design of a control system, based on Q(Youla)-Parametrization, for a practical 4x4 MIMO process.

7.2 The Multivariable Control System Design Methods

The Q(Youla)-Parametrization has introduced a significant theory for the design of linear feedback systems. The Q-Parametrization approach showed that any stabilizing controller can be characterized in terms of given matrices, although the resulting formulae are relatively complex. The formulae make use of coprime matrix fractional representations obtained most conveniently using the state-space approach presented by Nett et al [Nett et al, 1984].

The IMC Design Method which is a special case of the above parametrization, provides a design approach whereby the

designer has more direct information at the design stages. The method uses open loop information for the design of closed-loop control systems. In addition, it has many advantages over the traditional approaches that are most important in practical applications. However, the design theory tends to shift from the elegant simplicity that the IMC-Structure offers in the SISO case to more complex less obvious multivariable control law computations and tuning guidelines. In fact these tend to be more like model predictive controller designs [Garcia, 1985].

Useful insight was obtained from the diagonalization method offered by the classical Characteristic Loci Design Method via the Matrix-PI controller synthesis. A similar approach to that of synthesis of the Matrix-PI controller has been combined with the Q-parametrization for the implementation in the IMC-structure. This has lead to the formulation of a simpler, more direct approach to multivariable control system design.

7.3 The Proposed Simplifying Approach to Control System Design

The proposed design method, retains the design advantages offered by Q-Parametrization and the IMC-Structure in the SISO case. In addition, it simplifies the overall multivariable controller design. The algebraic matrix conditions on the numerators of the Q-parameter that ensure

asymptotic tracking of setpoints are simpler and easier to impose than derivative conditions on the error signal as in the IMC Design Method.

7.4 Flotation Rig(FLOTRIG) identification

Judging by the close correlation between the FLOTRIG and PC-Simulation results of the closed-loop system in chapter 6, it is reasonable to say that the model obtained for the FLOTRIG via the **FLOTRIG Identification** procedure was adequately close to the actual FLOTRIG system dynamics. The presence of non-linearities such as, actuator hysteresis and nonlinearities as well as system dynamic nonlinearities (e.g. the gain changes) did not pose significant problems. This is emphasized more by considering the LOR constrained in which the model was expected to be linear and the oscillations caused by the feedback pump cavitations etc.

7.5 Control scheme Implementation and Comparison of results

After comparisons have been done on the controller performance for the closed-loop system presented in chapter 6 and the original open loop system described in chapter 4, the following conclusions are drawn:

The presence of the controller has

- * improved the overall C/L system response
- * reduced the high frequency interaction in the C/L system
- * eliminated steady interaction and error in the C/L system

- * reduced the amplitude of oscillations caused by the feedback pump cavitations

The system responses corresponded closely to those expected from theory. In addition, the expected intermediate frequency interaction, which was not compensated for in the design, was also reduced.

The proposed design method results were compared to previous design results that used the INA and Characteristic Loci methods [Fisher, 1988], which were obtained from the FLOTRIG. It can be concluded that the simpler proposed design method, which also takes considerably less time to complete the design, showed better correlation between practical and digital simulation results than those given by the INA and CL methods.

Therefore, the controller designed using the simplifying control system design approach based on Q (Youla)-Parametrization can be considered satisfactory.

In conclusion, this thesis presented and evaluated a practical application of the proposed simplifying design method based on Q (Youla)-Parametrization method. The controller was applied in the IMC-Structure. Most literature described theoretical aspects and simulation results.

References

- Arkun Y, Hollet, J, Canney W M, Morari M, Experimental Study of Internal Model Control, *Ind. Eng. Chem. Process Des. Dev.* 1986,25, 102-108.
- Astrom K J, Wittenmark B, Computer controlled systems:Theory and design. Prentice-Hall, Englewood Cliffs, 1984.
- Braae M, Private Communication, 1991
- Chang B C, Yousuf A, Pole Placements and the Observer-Based Controller Parametrization, *IEEE Trans. on Auto. Control*, Vol AC-35, no 6, 726-729, June 1990.
- Chen S H, Lin C-A, Computer-aided controller design using O-Parametrization method, *Proc. 12th Nat. Symp. Automat. Control*, Taiwan, Republic of China, 247-260, 1988
- Desoer C A, Liu R, Murray J, Saeks R, Feedback System Design: The Fractional Representation Approach to Analysis and Synthesis, *IEEE Trans. on Auto. Control*, Vol AC-25, no. 3, 399-412, June 1980.
- Desoer C A, Chen M J, Design of Multivariable Feedback Systems with Stable Plant, *IEEE Trans. on Auto. Control*, Vol AC-26, no 2, 408-415, April 1981.
- Desoer C A, Chen M J, Extension of the "Design of Multivariable Feedback Systems with Stable Plant" to the Tracking Problem, *IEEE Trans. on Auto. Control*, Vol AC-26, no 2, 526-527, April 1981.

- Lin C-A, Hsieh T-F, Decoupling Controller Design for Multivariable Plants, *IEEE Trans. on Auto. Control*, Vol 36, no 4, 485-489, April 1991.
- MacFarlane A G J, Belleruttiti, J J, The Characteristic Locus Design Method, *Automatica*, Vol 9, 575-588, 1973.
- Maciejowski J M, Multivariable Feedback Design, Reading MA : Addison Wesley, 1989.
- Moore J B, Glover K, Telford A, All Stabilizing Controllers as Frequency-Shaped State Estimate Feedback, *IEEE Trans. on Auto. Control*, Vol AC-35, no 2, 203-208, February, 1990.
- Morari M, Robust Process Control, *Chem. Eng. Res. Des*, 65, 462-479, November 1987.
- Nett C N, Jacobson C A, Balas M J, A Connection Between State-Space and Doubly Coprime Fractional Representations, *IEEE Trans. on Auto. Control*, Vol AC-29, no. 9, 831-832, September 1984.
- Newton G C, Gould L A, Kaiser J F, Analytical design of linear feedback controls, Wiley, New York, 1957.
- Olivier B J, H_{∞} Optimal Control of a Counter-Current Process, *MSc Dissertation*, Department of Electrical and Electronic Engineering, Des 1991.
- Perry J H, Chemical Engineers Handbook, 4th edition, McGraw-Hill, New York, 1963.
- Ricker N L, Use of Quadratic Programming for Constrained Internal Model Control, *Ind. Eng. Chem. Process Des. Dev.*, 24, 925-936, 1985.

Rivera D, Morari M, Skogestad S, Internal Model Control.4. PID controller Design. *Ind. Eng. Chem. Process Des. Dev.*, 25, 252-265, 1985.

Rosenbrock H H, State-Space and Multivariable Theory. London:Wesley, 1970.

Rosenbrock H H, Computer Aided Control System Design, Academic press: London, 1974.

Venzke R H E, Comparison of the INA technique to the Pole Assignment Technique, *MSc Dissertation*, Department of Electrical and Electronic Engineering, December 1989.

Wang T, Shah B, Rangel D, Sribaugam D, Zhou G Y, Horn R, Weber C, Control of Laboratory Heat Exchanger Using Robust Internal Model Control, *American Control Conference*, Houston, 145-150, July 1991.

Wonham W, Linear Multivariable Control: A geometric Approach, 2nd ed., New York: Springer-Verlag, 1979.

Youla DC, Bongiorno J J, Jabr H A, Modern Wiener-Hopf Design of Optimal Controllers. Part I: The Single-Input-Output Case, *IEEE Trans on Auto. Control*, AC-21,, 3-13, February 1976.

Youla DC, Jabr H A, Bongiorno J J, Modern Wiener-Hopf Design of Optimal Controllers. Part II: The Multivariable Case, *IEEE Trans on Auto. Control*, AC-21,, 319-338, June 1976.

Zafiriou E, Morari M, Robust Process control. Prentice-Hall, Inc. New Jersey, 1989.

Zames G, Feedback and Optimal Sensitivity:Model Reference Transformation, Multiplicative Seminorms, and Approximate Inverses, *IEEE Trans. on Auto. Control*,26, no.1, 301-320, February 1981.

Bibliography

- Arkun Y, Hollet, J, Canney W M, Morari M, Experimental Study of Internal Model Control, *Ind. Eng. Chem. Process Des. Dev.* 1986,25, 102-108.
- Astrom K J, Wittenmark B, Computer controlled systems:Theory and design. Prentice-Hall, Englewood Cliffs, 1984.
- Boyd S P, Balakrishnan V, Barrett C, Khraishi N, Li X, Meyer D G, Norman S, A New CAD Method and Associated Architectures for Linear Controllers, *IEEE Trans. on Auto. Control*, Vol AC-33, no 3, 268-283, March 1988.
- Chang B C, Yousuf A, Pole Placements and the Observer-Based Controller Parametrization, *IEEE Trans. on Auto. Control*, Vol AC-35, no 6, 726-729, June 1990.
- Chen S H, Lin C-A, Computer-aided controller design using Q-Parametrization method, *Proc. 12th Nat. Symp. Automat. Control*, Taiwan, Republic of China, 247-260, 1988
- Da Silveira M A, Correa G O, On the Parametrization of Solutions to the Servomechanism Problem, *Automatica*, Vol 27, no 6, 1035-1038, 1991.
- Desoer C A, Liu R, Murray J, Saeks R, Feedback System Design: The Fractional Representation Approach to Analysis and Synthesis, *IEEE Trans. on Auto. Control*, Vol AC-25, no. 3, 399-412, June 1980.
- Desoer C A, Chen M J, Design of Multivariable Feedback Systems with Stable Plant, *IEEE Trans. on Auto. Control*, Vol AC-26, no 2, 408-415, April 1981.

Desoer C A, Chen M J, Extension of the "Design of Multivariable Feedback Systems with Stable Plant" to the Tracking Problem, *IEEE Trans. on Auto. Control*, Vol AC-26, no 2, 526-527, April 1981.

Economou E, Morari M, Internal Model Control, 6. Multiloop Design, *Ind. Eng. Chem. Process Des. Dev.*, 25, 411-419, 1986.

Economou E, Morari M, Palsson B O, Internal Model Control, 5. Extension to Nonlinear Systems, *Ind. Eng. Chem. Process Des. Dev.*, 25, 403-411, 1986.

Fisher I, Multivariable Control of a Flotation Plant Simulator, *MSc Dissertation*, Department of Electrical and Electronic Engineering, December 1988.

Garcia C, Morari M, Internal Model Control.1. A unifying review and some new results., *Ind. Eng. Chem. Process Des. Dev.*, 21, 308-323, 1982.

Garcia C, Morari M, Internal Model Control.2. Design Procedure for multivariable systems. *Ind. Eng. Chem. Process Des. Dev.*, 24, 472-484, 1985.

Garcia C, Morari M, Internal Model Control.3. Multivariable control law computation. *Ind. Eng. Chem. Process Des. Dev.*, 24, 484-494, 1985.

Koivo H N, Cojocariu R, An Optimal Control for a Flotation Circuit, *Automatica*, Vol 13, 37-45, 1977

Koivo H N, Pohjolainen S, Tuning of Multivariable PI-Controllers for Unknown Systems with Input Delay, *Automatica*, Vol 21, 81-91, 1985

Kouvaritakis B A, Theory and Practice of the Characteristic Locus Design Method, *IEE Proc.*, Vol 126, no 6, 542-548, June 1979.

Kouvaritakis B A, Complex Align: A Technique for the Characteristic Locus Method of Design, *IEE Proc.*, Vol 129, no 1, 1-5, January 1982.

Leyva-Ramos J, A New Method for Block Partial Fraction Expansion of Matrix Fraction Descriptions, *IEEE Trans. on Auto. Control*, Vol 36, no 12, 1482-1485, December 1991.

Lin C-A, Hsieh T-F, Decoupling Controller Design for Multivariable Plants, *IEEE Trans. on Auto. Control*, Vol 36, no 4, 485-489, April 1991.

MacFarlane A G J, Belleruttiti J J, The Characteristic Locus Design Method, *Automatica*, Vol 9, 575-588, 1973.

Maciejowski J M, Multivariable Feedback Design, Reading MA : Addison Wesley, 1989.

Meyer D G, A Parametrization of Stabilizing Controller for Multirate Sampled-Data Systems, *IEEE Trans. on Auto. Control*, Vol AC-35, no 2, 233-236, February 1990.

Moore J B, Glover K, Telford A, All Stabilizing Controllers as Frequency-Shaped State Estimate Feedback, *IEEE Trans. on Auto. Control*, Vol AC-35, no 2, 203-208, February, 1990.

Morari M, Robust Process Control, *Chem. Eng. Res. Des*, 65, 462-479, November 1987.

Nett C N, Jacobson C A, Balas M J, A Connection Between State-Space and Doubly Coprime Fractional Representations, *IEEE Trans. on Auto. Control*, Vol AC-29, no. 9, 831-832, September 1984.

Newton G C, Gould L A, Kaiser J F, Analytical design of linear feedback controls, Wiley, New York, 1957

Olivier B J, H_{∞} Optimal Control of a Counter-Current Process, *MSc Dissertation*, Department of Electrical and Electronic Engineering, Des 1991.

Raven F H, Automatic Control Engineering, 4th ed., McGraw-Hill, 1987.

Ricker N L, Use of Quadratic Programming for Constrained Internal Model Control, *Ind. Eng. Chem. Process Des. Dev.*, 24, 925-936, 1985.

Rivera D, Morari M, Skogestad S, Internal Model Control.4. PID controller Design. *Ind. Eng. Chem. Process Des. Dev.*, 25, 252-265, 1985.

Rosenbrock H H, State-Space and Multivariable Theory. London:Wesley, 1970.

Rosenbrock H H, Computer Aided Control System Design, Academic press: London, 1974.

Tebbutt C D, An Expert system approach to controller design, *IEE Proc.*, Vol 137, Pt. D, no 6, 367-373, November, 1990.

Venzke R H E, Comparison of the INA technique to the Pole Assignment Technique, *MSc Dissertation*, Department of Electrical and Electronic Engineering, December 1989.

Vidyasagar M, Conditions for a Feedback Transfer Function Matrix to be Proper, *IEEE Trans. on Auto. Control*, Technical Note, 570-571, August, 1975.

Wang T, Shah B, Rangel D, Stribaugam D, Zhou G Y, Horn R, Weber C, Control of Laboratory Heat Exchanger Using Robust Internal Model Control, *American Control Conference*, Houston, 145-150, July 1991.

Wonham W, On Pole Assignment in Multi-Input Controllable Linear Systems, *IEEE Trans. on Auto. Control*, Vol AC-12, no 6, 660-665, December 1967.

Wonham W, Linear Multivariable Control: A geometric Approach, 2nd ed., New York: Springer-Verlag, 1979.

Wylie C R, Barrett L, Advanced Engineering Mathematics, 5th ed, McGraw-Hill, 1985.

Youla DC, Bongiorno J J, Jabr H A, Modern Wiener-Hopf Design of Optimal Controllers. Part I: The Single-Input-Output Case, *IEEE Trans on Auto. Control*, AC-21,, 3-13, February 1976.

Youla DC, Jabr H A, Bongiorno J J, Modern Wiener-Hopf Design of Optimal Controllers. Part II: The Multivariable Case, *IEEE Trans on Auto. Control*, AC-21,, 319-338, June 1976.

Zafiriou E, Morari M, Robust Process control. Prentice-Hall, Inc. New Jersey, 1989.

Zames G, Feedback and Optimal Sensitivity: Model Reference Transformation, Multiplicative Seminorms, and Approximate Inverses, *IEEE Trans. on Auto. Control*, 26, no.1, 301-320, February 1981.

APPENDIX A

Flotation Rig(FLOTRIG) Open Loop Step Tests

A set of the many performed open loop step tests that is most representative of the FLOTRIG is presented in this appendix.

The data represent the fluid dynamics of the FLOTRIG for each of the four cells in the LOR. The data are not normalized but are given in percentages for both the process input and output.

The data plotted have a sampling frequency of 1 Hz and have the following format:

For a step change in one of the process inputs, all other data(levels) were recorded for the corresponding FLOTRIG cells. These graphs are plotted on the same set of axes. Thus, the graphs on one set of axes represent the responses of the four cells when a particular cell tail valve was stepped a certain percentage. The initial tail valve positions were more or kept the same as follows:

Initial Valve settings: (100% = fully closed, -100% = fully open)

Rougher tail valve: -50 %

Scavenger tail valve: -50 %

Cleaner tail valve: -50 %

Recleaner tail valve: 0 %

The order in which the above valve positions was also the sequence in which they were stepped. Similarly they are also presented that sequence.

The recorded data were used to determine estimated transfer functions for each cell. The graphs are presented on the next four pages.

A.1 Open Loop Step Test - Rougher

The open loop responses for a step change in the Rougher tail valve are given below in figure A.1.

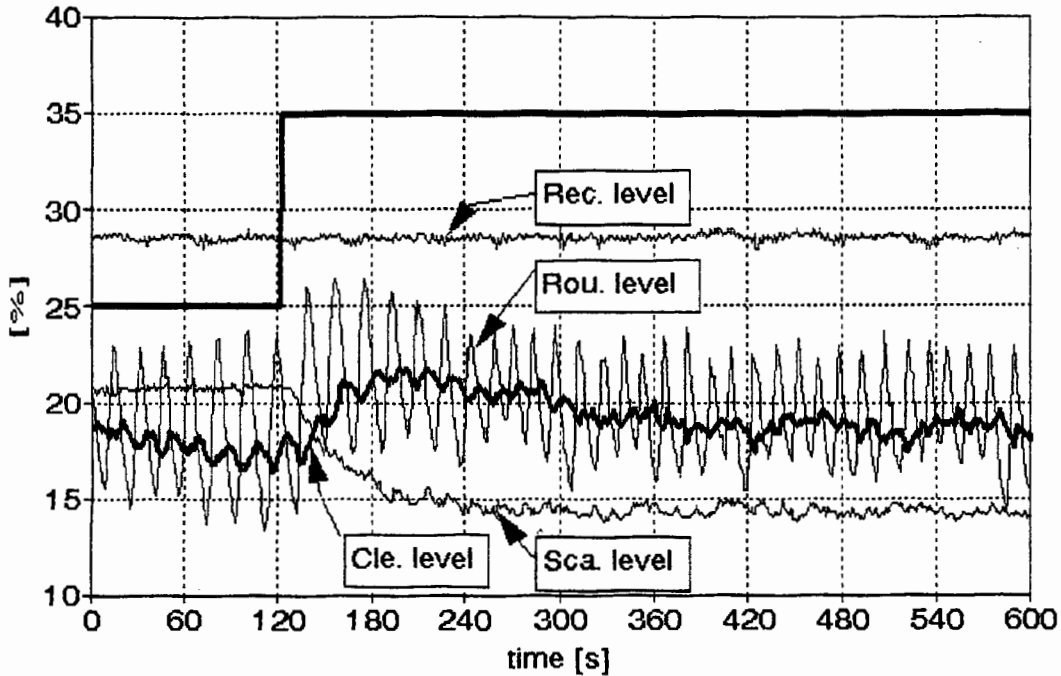


Figure A.1: ROUGHER CELL tail valve stepped +10%

Valve position before step change: (100% = fully closed, -100% fully open)

Rougher(Rou)	tail valve: -50%
Scavenger(Sca)	tail valve: -50%
Cleaner(Cle)	tail valve: -50%
Recleaner(Rec)	tail valve: 0%

Observations:

- no setpoint tracking
- interaction between Rougher, Cleaner and Scavenger
- disturbance present on Rougher and Cleaner cell outputs
- noise present on all outputs

A.2 Open Loop Step Test - Scavenger

The open loop responses to a step change in the Scavenger tail valve are given below in figure A.1.

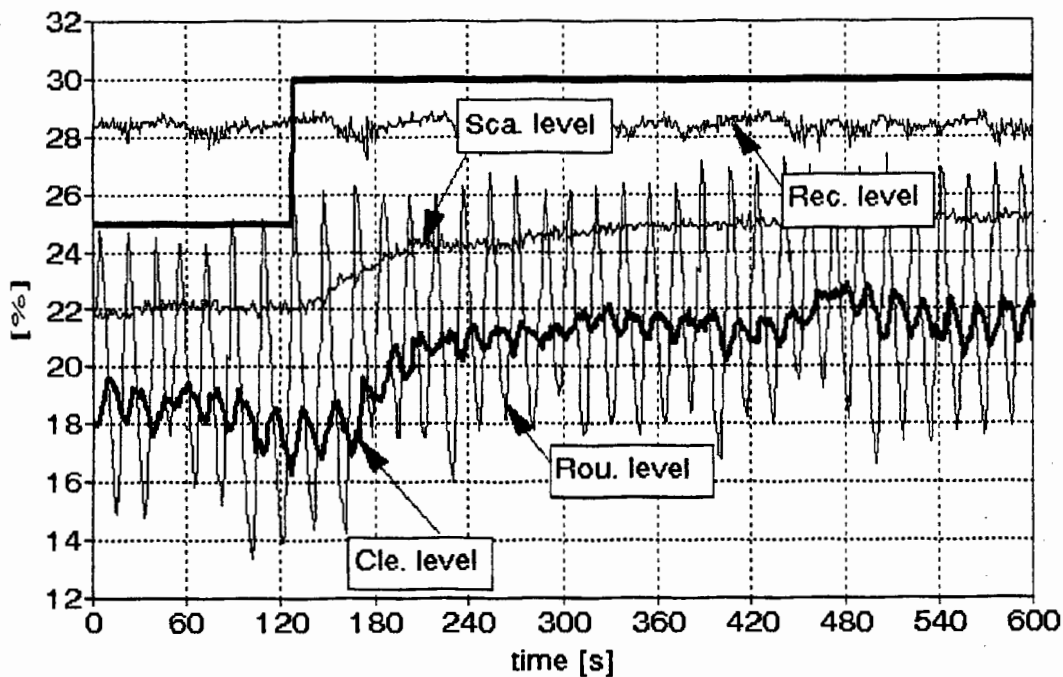


Figure A.2: SCAVENGER CELL tail valve stepped +5%

Valve position before step change: (100% = fully closed, -100% fully open)

Rougher(Rou)	tail valve: -50%
Scavenger(Sca)	tail valve: -50%
Cleaner(Cle)	tail valve: -50%
Recleaner(Rec)	tail valve: 0%

Observations:

- no setpoint tracking
- interaction between Rougher, Cleaner and Scavenger
- disturbance present on Rougher and Cleaner cell outputs
- noise present on all outputs

A.3 Open Loop Step Test - Cleaner

The open loop responses for a step change in the Cleaner tail valve are given below in figure A.3.

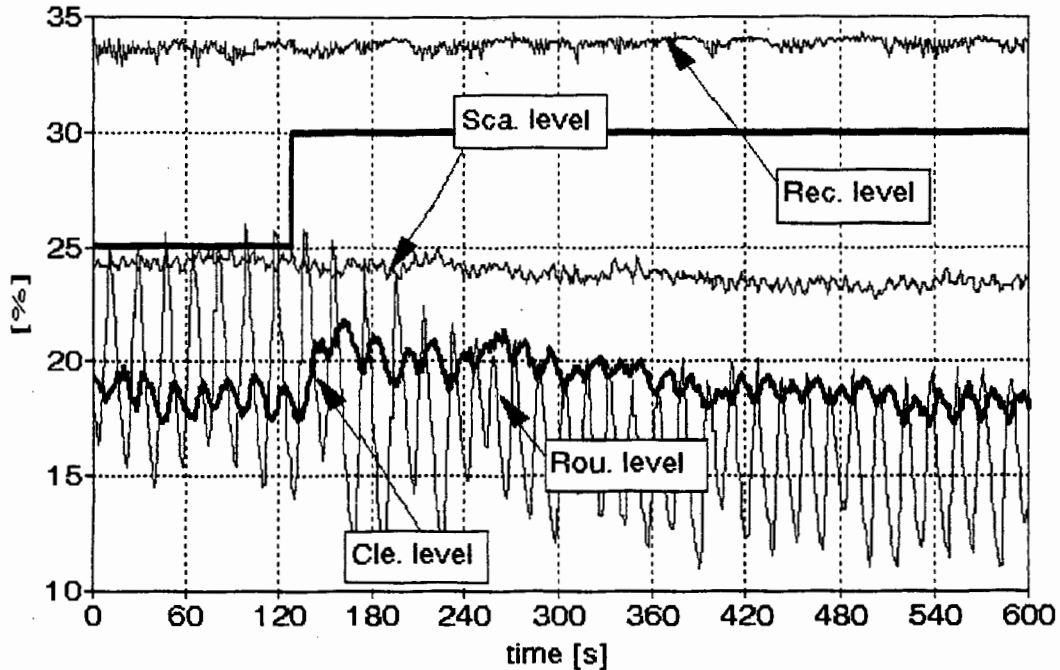


Figure A.3: CLEANER CELL tail valve stepped +5%

Valve position before step change: (100% = fully closed, -100% fully open)

Rougher(Rou)	tail valve: -50%
Scavenger(Sca)	tail valve: -50%
Cleaner(Cle)	tail valve: -50%
Recleaner(Rec)	tail valve: 0%

Observations:

- no setpoint tracking
- interaction between Rougher, Cleaner and Scavenger
- disturbance present on Rougher and Cleaner cell outputs
- noise present on all outputs

A.4 Open Loop Step Test - Recleaner

The open loop responses for a step change in the Recleaner tail valve are given below in figure A.4.

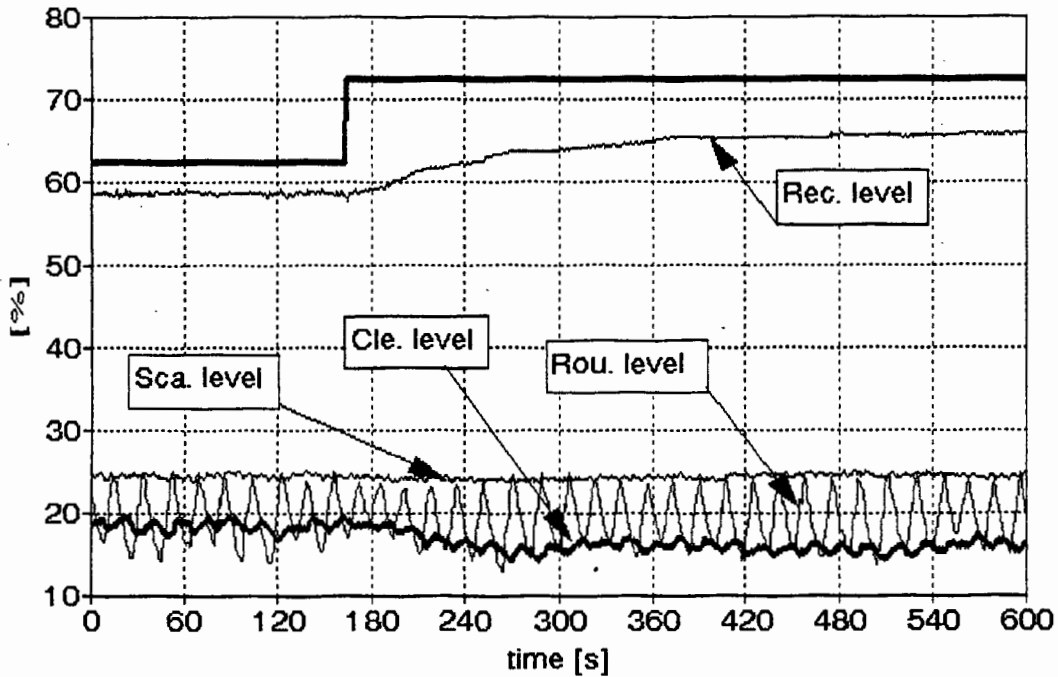


Figure A.4: RECLENER CELL tail valve stepped +10%

Valve position before step change: (100% = fully closed, -100% fully open)

Rougher(Rou)	tail valve: -50%
Scavenger(Sca)	tail valve: -50%
Cleaner(Cle)	tail valve: -50%
Recleaner(Rec)	tail valve: 0%

Observations:

- no setpoint tracking
- interaction between Recleaner and Cleaner
- disturbance present on Rougher and Cleaner cell outputs
- noise present on all outputs

Appendix B

Data Analysis and FLOTRIG Model Estimates

This appendix contains the estimates fitted to the data obtained from the open loop step test performed on the FLOTRIG. All the data presented have been normalized to the respective step sizes. The estimates were fitted with the Identification software given in [Olivier, 1991].

The plots on the next four pages show how the data obtained from the FLOTRIG(non-smooth) and the estimate(smooth) fit the respective cell responses. Every page represents an element of the open loop system matrix. Below every graph, the corresponding transfer functions(in the 's'-domain) that estimate the data are given in factorized form. The elements of the matrix which were regarded as zero, as noted after obtaining the open loop step test results, are represented by a zero. This can also be confirmed from the O/L step tests graphs which are shown in Appendix A.

The graphs follow on the next pages. The respective steps are:-

Rougher tail valve stepped +10%

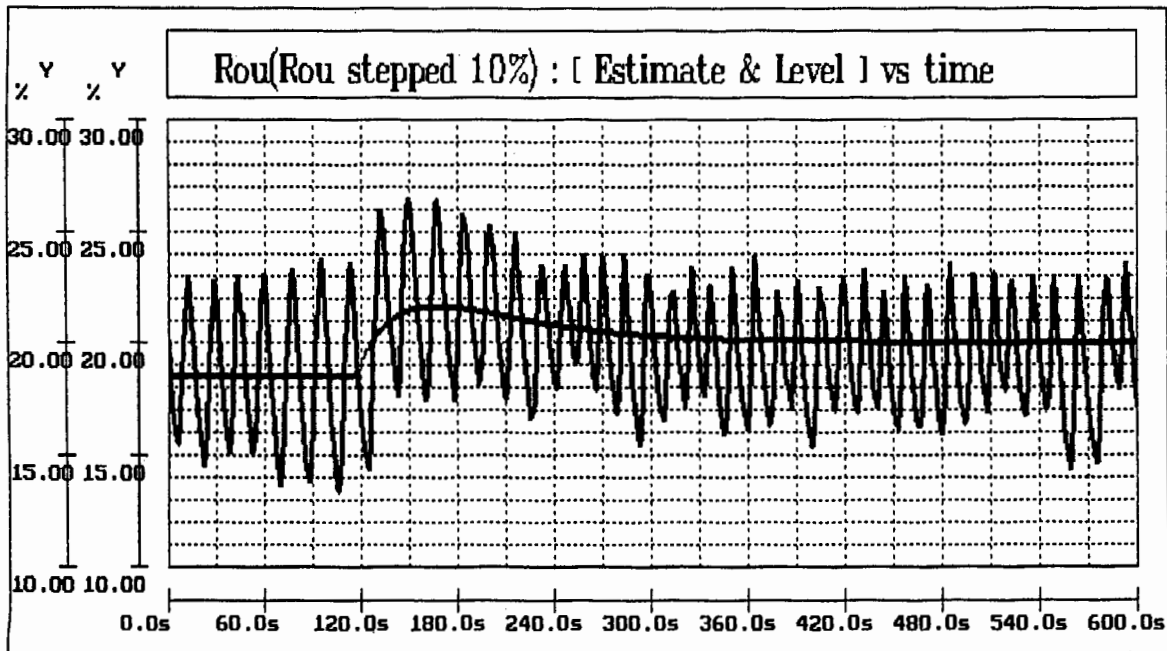
Scavenger tail valve stepped +10%

Cleaner tail valve stepped +10%

Recleaner tail valve stepped +10%

B.1 Rougher level plot - Rougher tail valve stepped

FLOTATION PROCESS RIG : SECOND ORDER DATA FIT

Figure B.1: $g_{11}(s)$ & estimate

$$g_{11}(s) = 0.075 \left[\frac{(0.03)(0.02)}{0.005} \right] \left[\frac{(s + 0.005)}{(s + 0.03)(s + 0.02)} \right]$$

$$g_{11}(s) = 0.075 \left[\frac{(200s + 1)}{(33.33s + 1)(50s + 1)} \right]$$

Gain : $K = 0.075$

Zero(es) at: $s = -0.005$ [rad/s]

Pole(s) at: $s = -0.03$ and -0.02 [rad/s]

Delay : 0 [secs]

B.2 Scavenger level plot - Rougher tail valve stepped

FLOTATION PROCESS RIG : FIRST ORDER DATA FIT

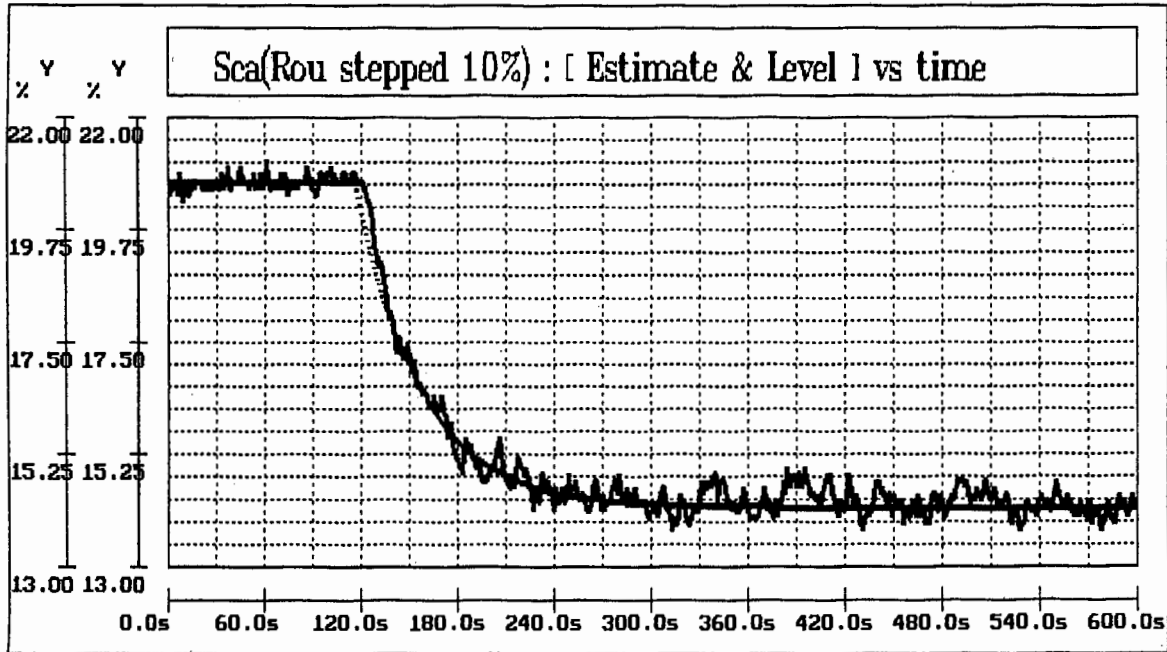


Figure B.2: g21(s) & estimate

$$g_{21}(s) = -0.325 \left[\frac{(0.025)}{(s + 0.025)} \right] e^{-15s}$$

$$g_{21}(s) = -0.325 \left[\frac{1}{(40s + 1)} \right] e^{-15s}$$

Gain : $K = -0.325$

Zero(es) at: non

Pole(s) at: $s = -0.025$ [rad/s]

Delay : 15 [secs]

B.3 Cleaner level plot - Rougher tail valve stepped

FLOTATION PROCESS RIG : SECOND ORDER DATA FIT

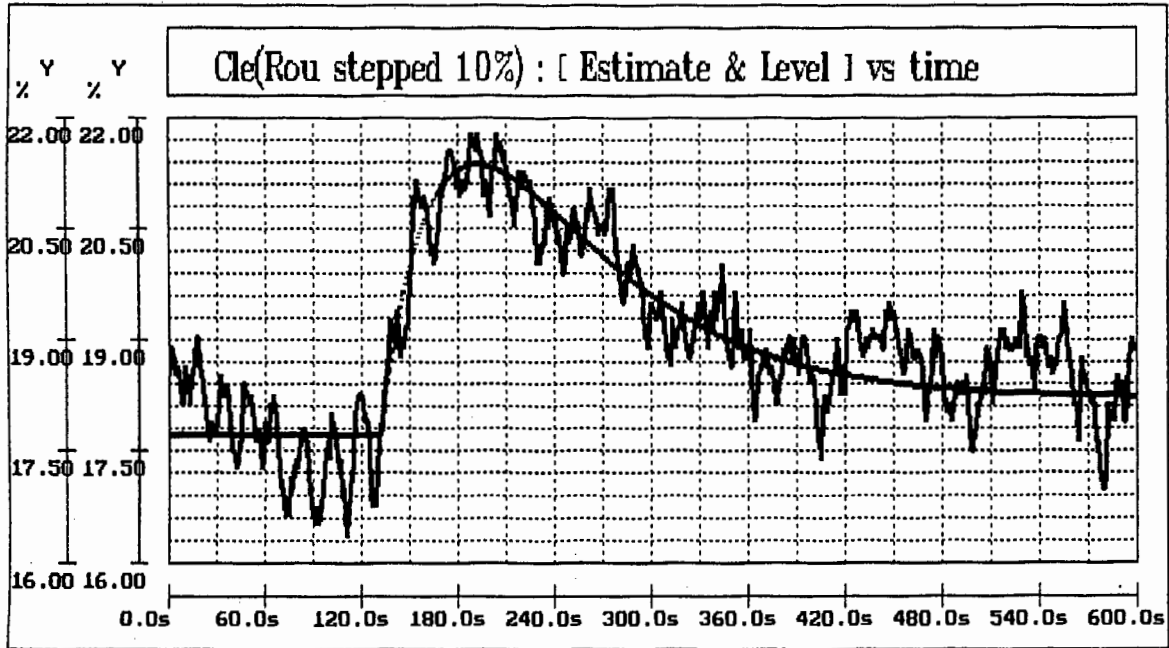


Figure B.3: g31(s) & estimate

$$g_{31}(s) = \frac{0.05}{2} \left[\frac{(0.015)(0.02)}{0.0009} \right] \left[\frac{(s + 0.0009)}{(s + 0.02)(s + 0.015)} \right]$$

$$g_{31}(s) = \frac{0.05}{2} \left[\frac{(1 \ 111.11s + 1)}{(50s + 1)(66.66s + 1)} \right]$$

Gain : K = 0.05

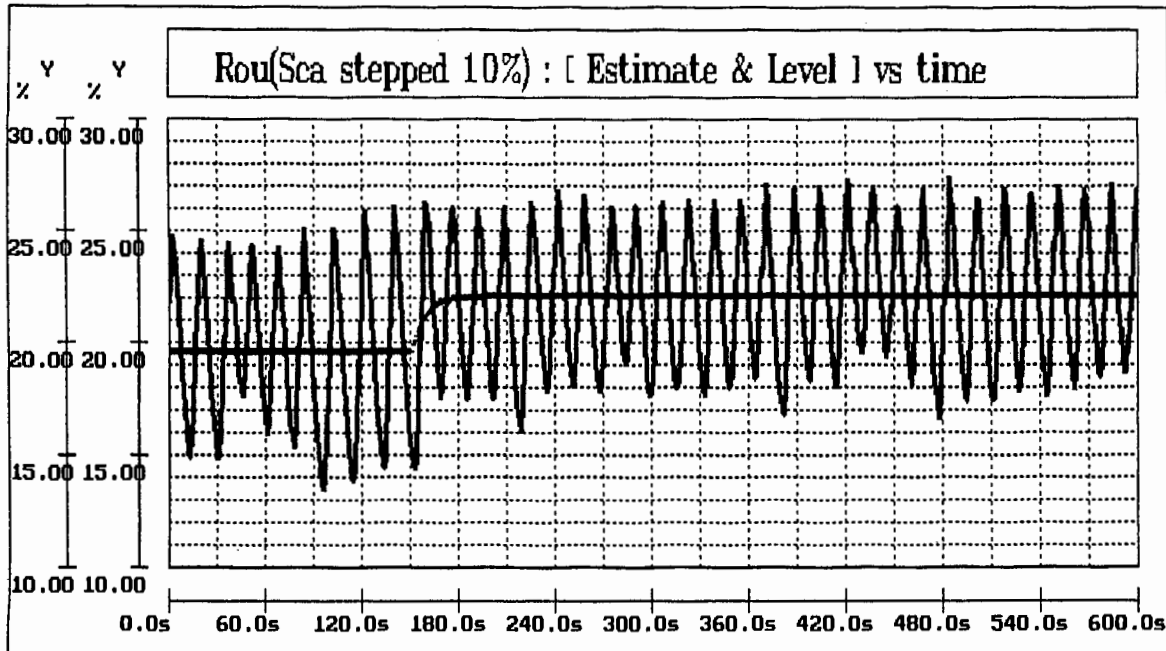
Zero(es) at: s = -0.0009 [rad/s]

Pole(s) at: s = -0.02 and -0.015 [rad/s]

Delay : 0 [secs]

B.4 Rougher level plot - Scavenger tail valve stepped

FLOTATION PROCESS RIG : FIRST ORDER DATA FIT

Figure B.4: $g_{12}(s)$ & estimate

$$g_{12}(s) = 0.5 \left[\frac{0.0125}{(s + 0.0125)} \right] e^{-30s}$$

$$g_{12}(s) = 0.5 \left[\frac{1}{(80s + 1)} \right] e^{-30s}$$

Gain : $K = 0.5$

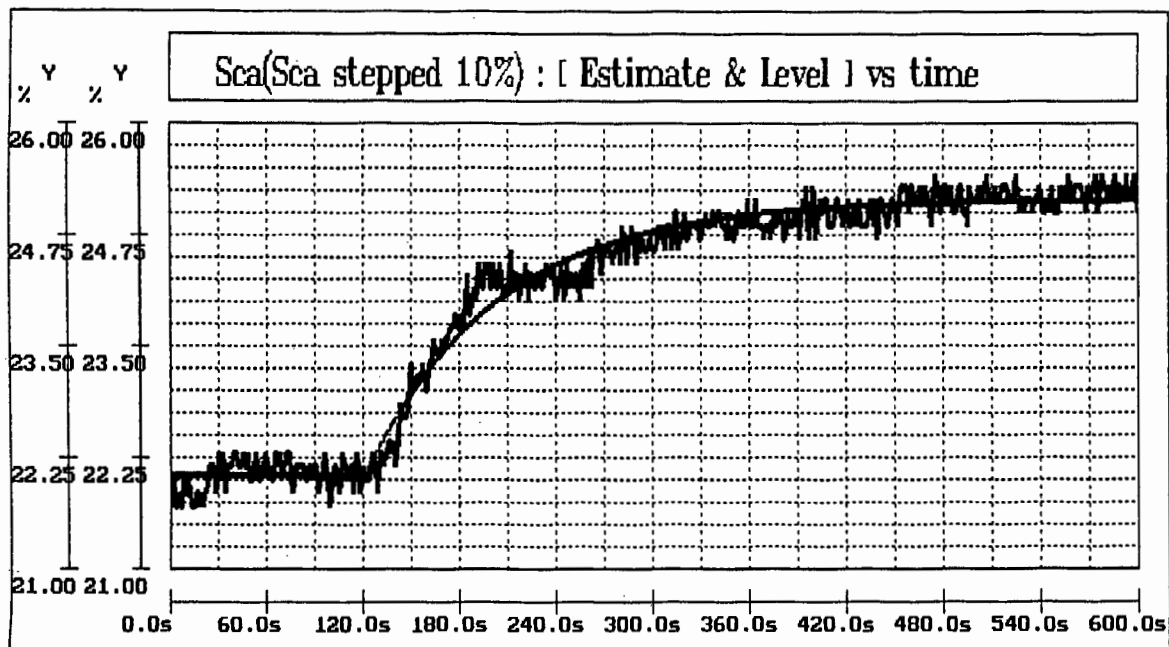
Zero(es) at: non

Pole(s) at: $s = -0.0125$ [rad/s]

Delay : 30 [secs]

B.5 Scavenger level plot - Scavenger tail valve stepped

FLOTATION PROCESS RIG : FIRST ORDER DATA FIT

Figure B.5: $g_{22}(s)$ & estimate

$$g_{22}(s) = 0.62 \left[\frac{0.012}{(s + 0.012)} \right]$$

$$g_{22}(s) = 0.62 \left[\frac{1}{(83.33s + 1)} \right]$$

Gain : $K = 0.62$

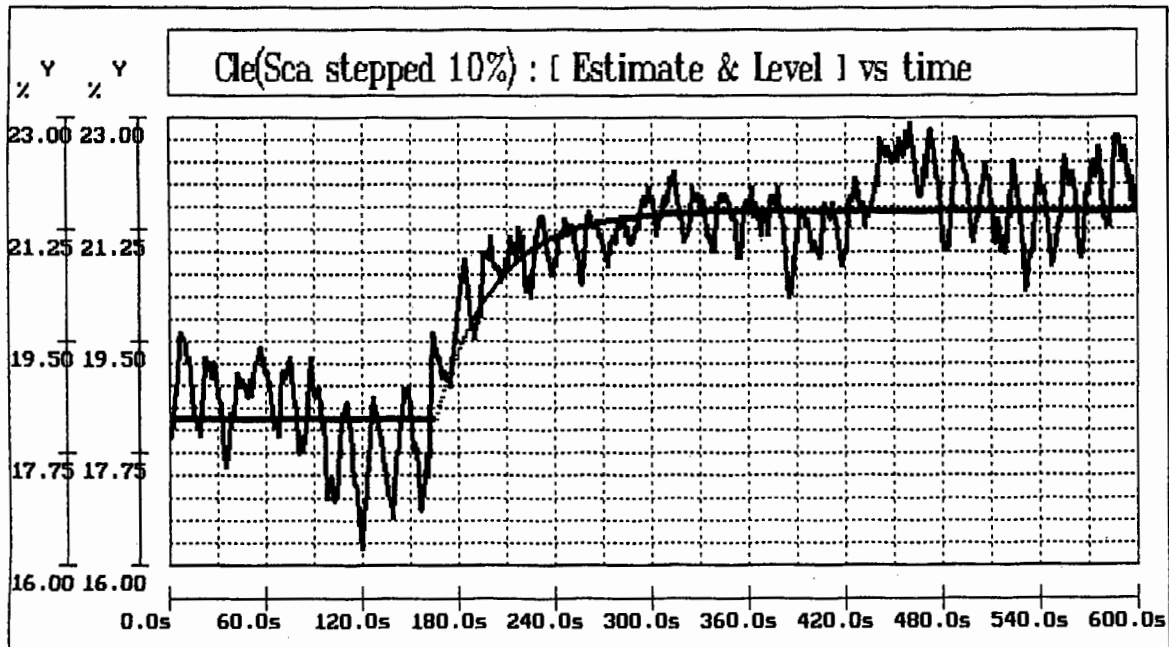
Zero(es) at: non

Pole(s) at: $s = -0.012$ [rad/s]

Delay : 0 [secs]

B.6 Cleaner level plot - Scavenger tail valve stepped

FLOTATION PROCESS RIG : FIRST ORDER DATA FIT

Figure B.6: $g_{32}(s)$ & estimate

$$g_{32}(s) = 0.65 \left[\frac{0.028}{(s + 0.028)} \right] e^{-45s}$$

$$g_{32}(s) = 0.65 \left[\frac{1}{(35.71s + 1)} \right] e^{-45s}$$

Gain : $K = 0.65$

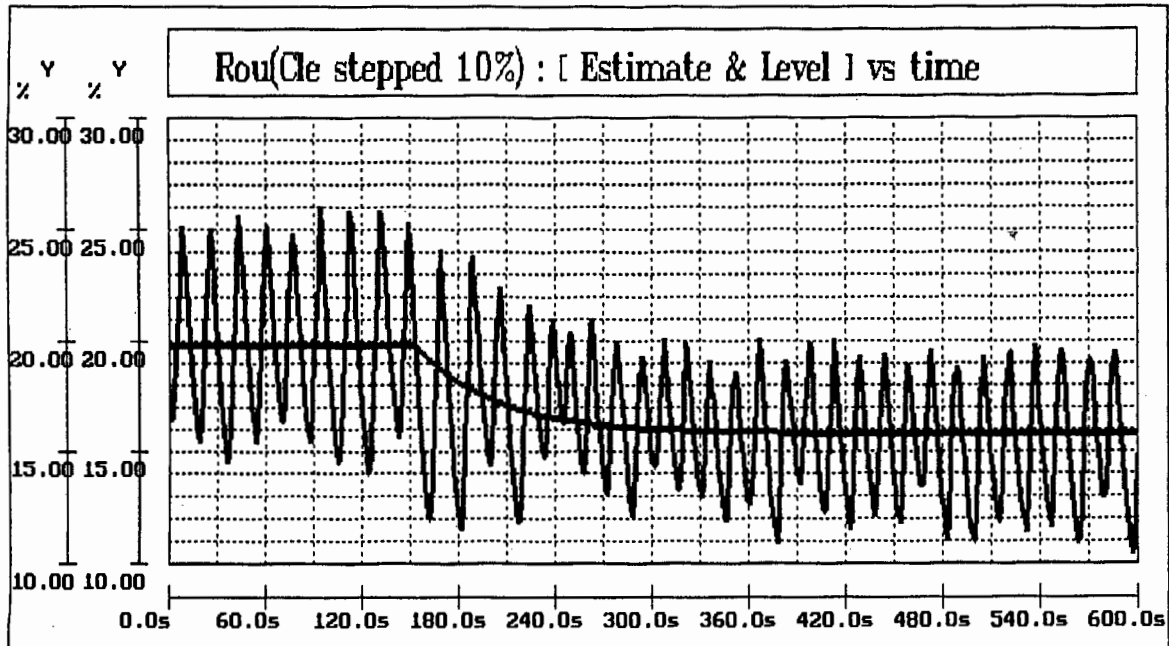
Zero(es) at: non

Pole(s) at: $s = -0.028$ [rad/s]

Delay : 45 [secs]

B.7 Rougher level plot - Cleaner tail valve stepped

FLOTATION PROCESS RIG : FIRST ORDER DATA FIT

Figure B.7: $g_{13}(s)$ & estimate

$$g_{13}(s) = -0.8 \left[\frac{0.02}{(s + 0.02)} \right] e^{-30s}$$

$$g_{13}(s) = -0.8 \left[\frac{1}{(50s + 1)} \right] e^{-30s}$$

Gain : $K = -0.8$

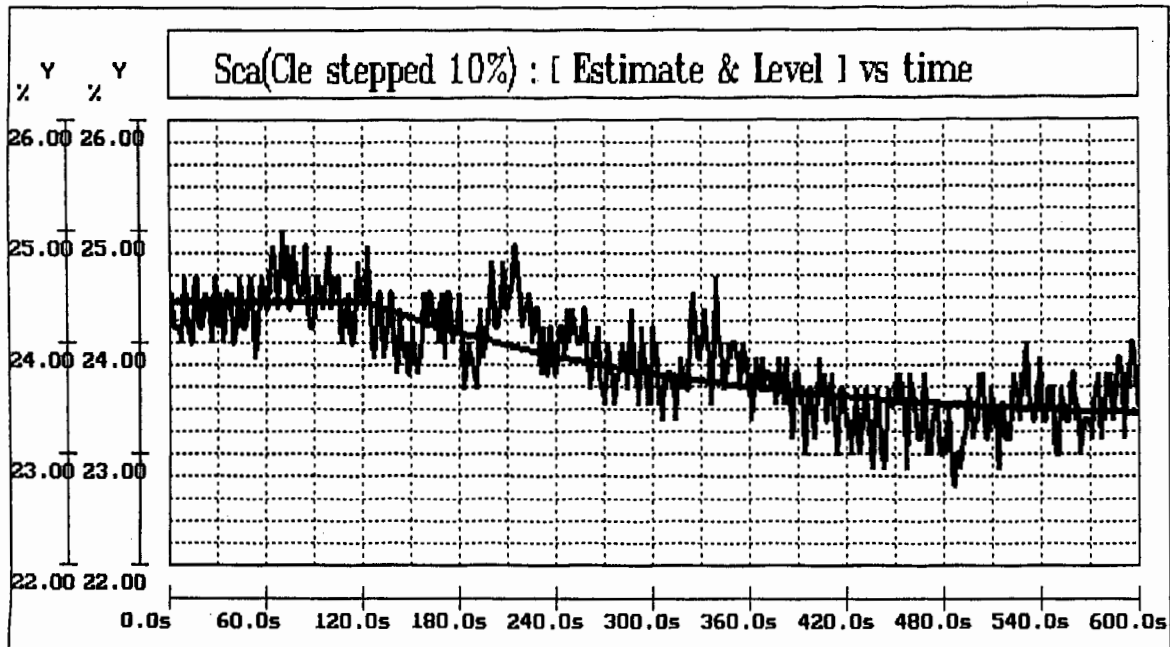
Zero(es) at: non

Pole(s) at: $s = -0.02$ [rad/s]

Delay : 30 [secs]

B.8 Scavenger level plot - Cleaner tail valve stepped

FLOTATION PROCESS RIG : FIRST ORDER DATA FIT

Figure B.8: $g_{23}(s)$ & estimate

$$g_{23}(s) = -0.22 \left[\frac{0.005}{(s + 0.005)} \right]$$

$$g_{23}(s) = -0.22 \left[\frac{1}{(200s + 1)} \right]$$

Gain : $K = -0.22$

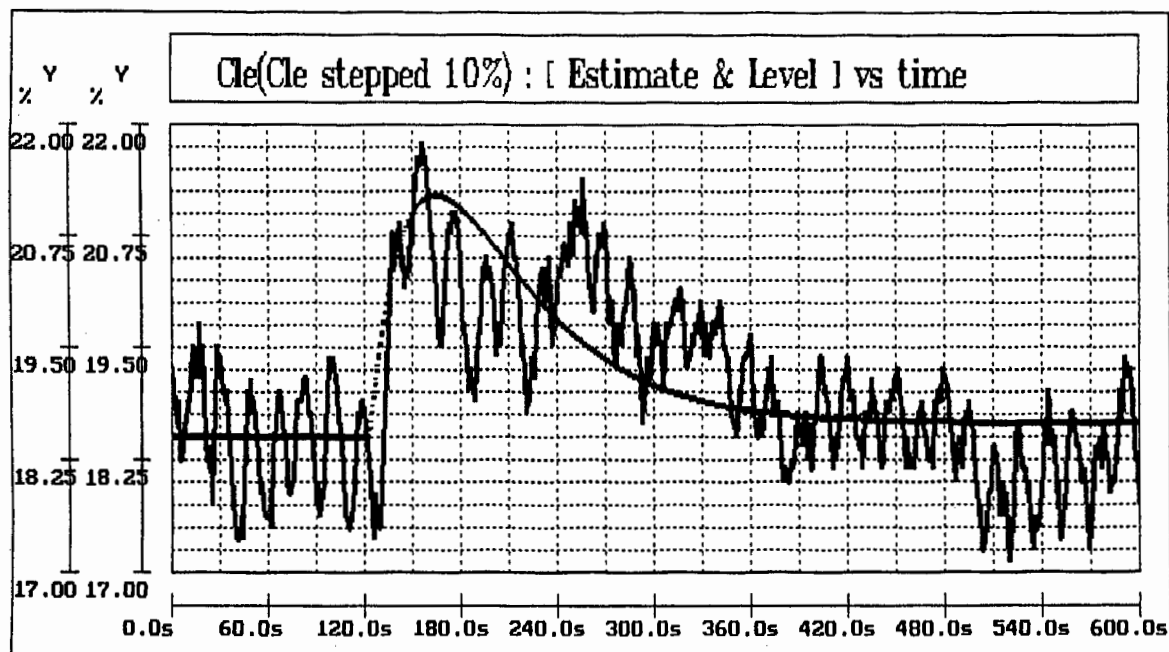
Zero(es) at: non

Pole(s) at: $s = -0.005$ [rad/s]

Delay : 0 [secs]

B.9 Cleaner level plot - Cleaner tail valve stepped

FLOTATION PROCESS RIG : SECOND ORDER DATA FIT

Figure B.9: $g_{33}(s)$ & estimate

$$g_{33}(s) = 0.03 \left[\frac{(0.03)(0.02)}{0.0005} \right] \left[\frac{(s + 0.0005)}{(s + 0.03)(s + 0.02)} \right]$$

$$g_{33}(s) = 0.03 \left[\frac{(2000s + 1)}{(33.33s + 1)(50s + 1)} \right]$$

Gain : $K = 0.03$

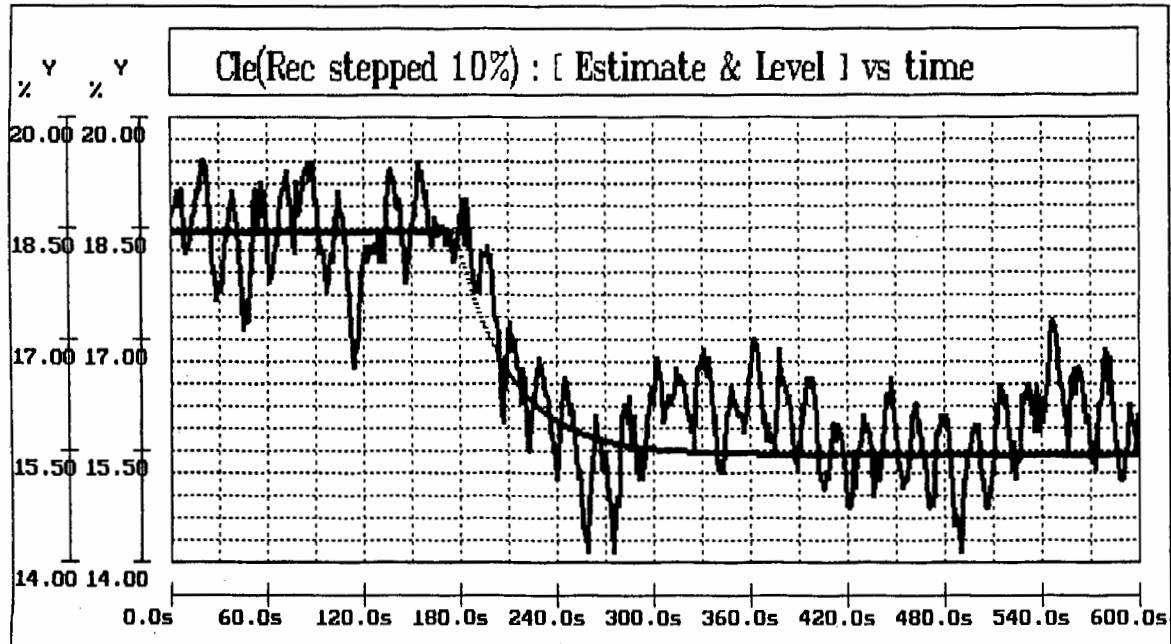
Zero(es) at: $s = -0.0005$ [rad/s]

Pole(s) at: $s = -0.03$ and -0.02 [rad/s]

Delay : 0 [secs]

B.10 Cleaner level plot - Recleaner tail valve stepped

FLOTATION PROCESS RIG : FIRST ORDER DATA FIT

Figure B.10: $g_{34}(s)$ & estimate

$$g_{34}(s) = -0.15 \left[\frac{0.03}{(s + 0.03)} \right] e^{-20s}$$

$$g_{34}(s) = -0.15 \left[\frac{1}{(33.33s + 1)} \right] e^{-20s}$$

Gain : $K = -0.15$

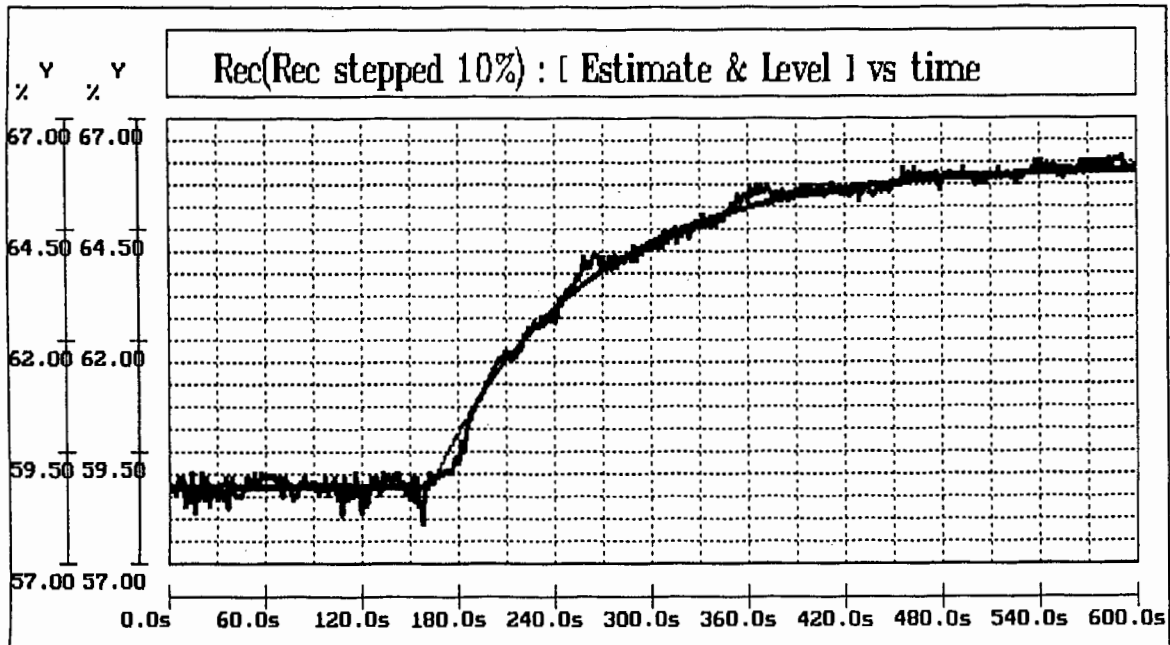
Zero(es) at: non

Pole(s) at: $s = -0.03$ [rad/s]

Delay : 20 [secs]

B.11 Recleaner level plot - Recleaner tail valve stepped

FLOTATION PROCESS RIG : FIRST ORDER DATA FIT

Figure B.11: $g_{44}(s)$ & estimate

$$g_{44}(s) = 0.36 \left[\frac{0.0105}{(s + 0.0105)} \right] e^{-5s}$$

$$g_{44}(s) = 0.36 \left[\frac{1}{(95.24s + 1)} \right] e^{-5s}$$

Gain : $K = 0.36$

Zero(es) at: non

Pole(s) at: $s = -0.0105$ [rad/s]

Deadtime : 5 [secs]

B.12 Comments on the Data and Estimates

- a) The orders of the estimates to the data were kept as low as possible. However, most of the estimates seem to have approximately the right order.
- b) The high frequency components which are superimposed on most of the data were not modelled. For example the oscillations caused by the surging of the sump feedback pumps. The pump 'surge-cavitation' cycle is explained in chapter four.
- c) All elements have been modelled from initial steady conditions to final their respective steady state.
- d) Elements, $g_{31}(s)$, $g_{22}(s)$, $g_{32}(s)$, $g_{23}(s)$, $g_{33}(s)$ and $g_{34}(s)$ show unexpected changes in the data. These changes were not common and were easily associated with irrelevant signal changes in the FLOTRIG instrumentation.

Appendix C

Design of a Q-Parameterized controller implemented in the IMC-Structure for a Ramp tracking Servo Motor

In this appendix the design approach for $q(s)$ is presented for a SISO servo motor system after having considered the Internal Model Control(IMC) Design Method, a special case of the Q(Youla)-Parametrization. From the IMC design theory it was noted that asymptotic tracking of setpoints when implemented in the (IMC) structures can be ensured by imposing a set of criteria based on derivatives of the transfer function $[1-q(s)g(s)]$ relating error to setpoint. In addition it should be noted that the Internal Model referred to here is not the same as the "Internal Model Principle" mentioned presented by Wonham as discussed in chapter 2. Simpler algebraic constraints on the numerator polynomial of module $q(s)$ that give a particular Type Number for the final IMC system were developed and applied to the servomotor system for that is to track a triangular setpoint with zero steady-state error.

The SISO design procedure as given in chapter three was applied to the servomotor. The application is presented next.

C.1. THE OPEN LOOP SERVO MOTOR SYSTEM

A block diagram for the servo motor system is shown in Fig C.1. The motor is connected to an IBM PC via a 12-bit, 10-Volt bipolar Data Translation analog interface card. A LTI transfer function model for the dominant dynamic behaviour of the motor was derived from step test data collected and plotted using the PC.

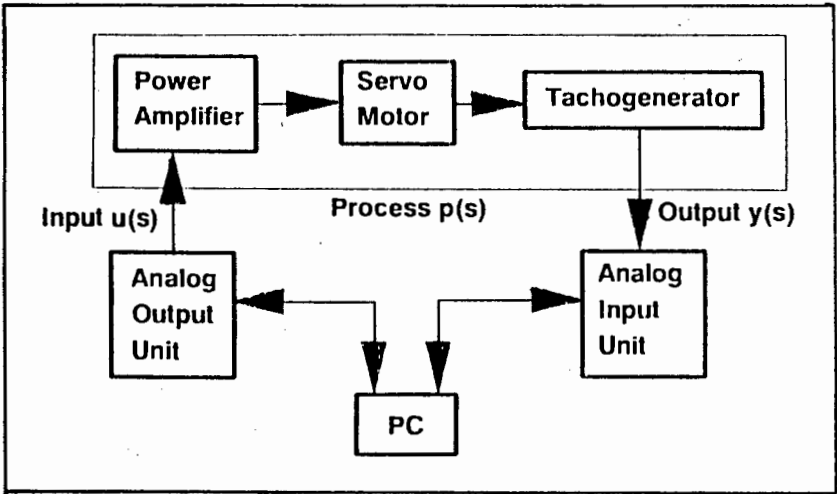


Figure C.1: Block diagram of servo-motor system

The nominal transfer function model relating motor speed (ie. tachogenerator voltage) to armature voltage is

$$\begin{aligned} g(s) &= y(s) / u(s) \\ &= A / (1 + \tau s) \\ &= 11.4 / (1+1.3s) \quad [\text{Count}]/[\text{Count}] \\ &= \frac{N_0 + N_1s + N_2s^2 + \dots + N_ms^m}{1 + D_1s + D_2s^2 + \dots + D_ns^n} \\ &= N(s) / D(s) \end{aligned} \quad \dots c-1$$

where $N(s)$ and $D(s)$ are polynomials in 's'. The $\deg\{N(s)\} \leq \deg\{D(s)\}$.

This system is open loop stable as required by the IMC method. It has a gain of 11.4 counts/count and a time constant of 1.3 seconds.

Correlation between the model and the experimental data is adequate as illustrated by Fig C.2 below.

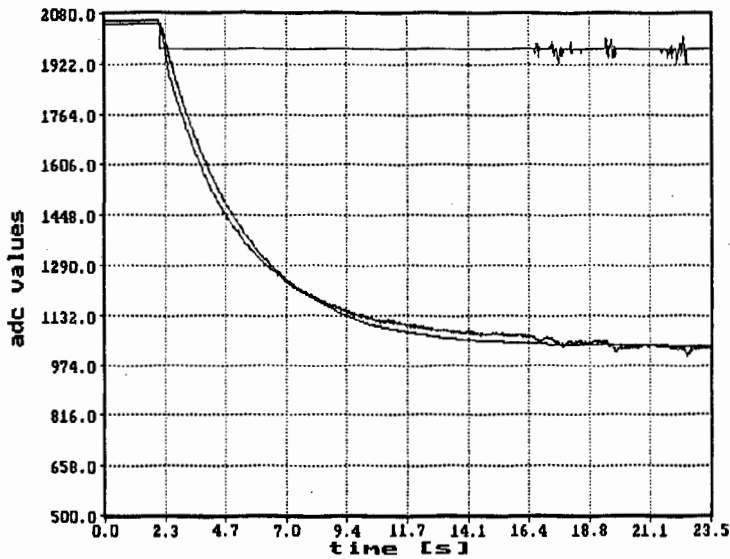


Figure C.2: Open loop step test results

C.2. INTERNAL MODEL CONTROL

The internal model control loop is shown chapter two. The design of the closed loop system reduces very simply to the open loop design of the Newton (or IMC) control module $q(s)$ so that

$$h(s) = q(s) g(s) \quad \text{....c-2}$$

is the transfer function of the desired closed loop system. The control action needed to produce the designed closed loop performance can easily be gauged directly from module $q(s)$ which is the transfer function from the reference input, $r(s)$, to the process input, $u(s)$.

The standard IMC is used to design a feedback control scheme for the servo-motor system. The specifications for the closed loop system are that the output must equal the setpoint and it must be twice as fast as the open loop system.

The control module $q(s)$ needed to achieve these specifications is given by

$$\begin{aligned} q(s) &= u(s) / r(s) \\ &= h(s)/g(s) \\ &= (1 + s\tau) / A(1 + \tau s/2) \\ &= (1+1.3s) / 11.4(1+0.65s) \text{ [Count]/[Count]} \\ &= \frac{n_0 + n_1s + n_2s^2 + \dots}{1 + d_1s + d_2s^2 + \dots} \\ &= n(s) / d(s) \quad \text{....c-3} \end{aligned}$$

The relationship between $k(s)$ and $q(s)$ is

$$\begin{aligned} k(s) &= \frac{q(s)}{1 - q(s)g(s)} \\ q(s) &= \frac{k(s)}{1 + g(s)k(s)} \end{aligned}$$

It should be noted that asymptotic setpoint tracking is ensured since the steady state gain of $q(s)$ is the inverse of the process gain. Also the slow pole of the servo motor is cancelled by the zero in $q(s)$ and replaced by a faster pole as required in the design specifications. Finally the high frequency gain of this module is easily seen to be $(2/11.4)$ [Counts/Counts].

The trade-off between input movement and speed of response is very clearly indicated during the design of $q(s)$, and is a major practical advantage noted in the application of internal model control methods to the design of feedback control systems. Once the design specifications have been met excessively fast and large excursions in input $u(s)$ due to setpoint changes can be reduced very easily by including suitable low pass filters in module $q(s)$.

The closed loop step response is plotted in Fig C.3 below, showing that the design objectives have been attained.

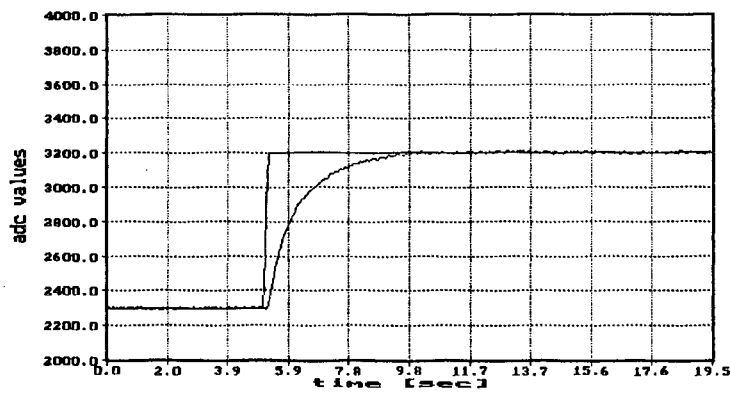


Figure C.3: IMC step response (Type I)

The controller, $k(s)$, that is produced by this IMC design method is

$$\begin{aligned} k(s) &= (1 + \tau s) / A(\tau/2)s \\ &= (1+1.3s) / 11.4(0.65s) \end{aligned} \quad \text{....C-4}$$

which is clearly Type 1.

Thus the standard IMC system will track step changes in setpoint exactly. However the design is not suitable for tracking ramp setpoints as expected from theory and as illustrated by the closed loop responses in Fig C.4 below.

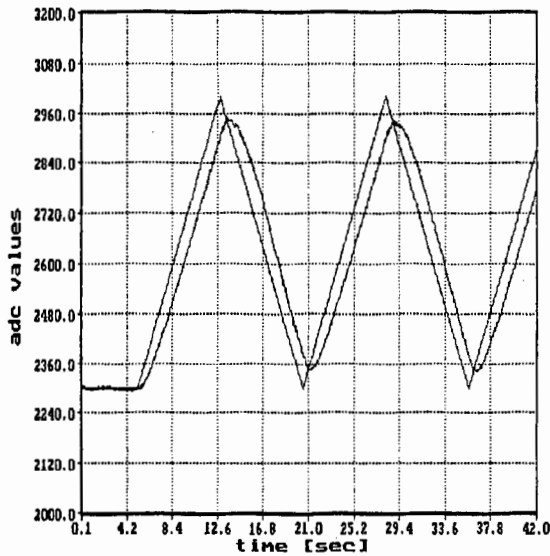


Figure C.4: IMC triangular wave (ramp) response $q(s)$ designed for step tracking

C 3. TYPE NUMBERS IN IMC

In traditional feedback configurations System Type Number is an important criterion that determines whether or not the given closed loop system will track step, ramp, parabolic or

higher-order setpoints asymptotically. Details on system type numbers are given chapters two and three.

An alternative approach, that avoids the computation of these derivatives, considers the traditional controller $k(s)$ that is equivalent to the IMC control configuration. Using the transfer functions defined in Equations c-1 and c-3 the formulation as given in chapter three was used for the design of $q(s)$. This approach will be applied to the servomotor to design a $q(s)$ that will ensure zero steady state error for a triangular-wave setpoint and to double its speed of response compared to the open loop system.

From traditional feedback theory it can be concluded that the system Type Number must be at least 2. For the IMC controller(= Q-parameter) the design of the numerator, $n(s)$, of control module, $q(s)$, is given by the simple matrix equation

$$\begin{aligned} \begin{bmatrix} n_0 \\ n_1 \end{bmatrix} &= \begin{bmatrix} N_0 & 0 \\ N_1 & N_0 \end{bmatrix}^{-1} \begin{bmatrix} 1 & 0 \\ D_1 & 1 \end{bmatrix} \begin{bmatrix} 1 \\ d_1 \end{bmatrix} \begin{matrix} \text{Type 1} \\ \text{Type 2} \end{matrix} \\ &= \begin{bmatrix} 11.4 & 0 \\ 0 & 11.4 \end{bmatrix}^{-1} \begin{bmatrix} 1 & 0 \\ 1.3 & 1 \end{bmatrix} \begin{bmatrix} 1 \\ 0.65 \end{bmatrix} \\ &= \begin{bmatrix} 0.0877 \\ 0.1711 \end{bmatrix} \end{aligned}$$

....c-5

Thus the control module $q(s)$ is given by the transfer function

$$q(s) = \frac{0.0877 + 0.1711s}{1 + 0.65s} \quad \dots c-6$$

In traditional terms the controller $k(s)$ derived from this design is

$$k(s) = \frac{0.0877 + 0.2851s + 0.2224s^2}{0.8455s^2} \quad \dots c-7$$

which is Type 2 as designed. However the response of the resulting closed loop system has not been taken into account.

The design will now be modified to include Dynamic Compensation as follows:-

In the IMC approach to designing feedback control systems module $q(s)$ is usually chosen to cancel the undesirable poles of the process $g(s)$, provided of course that these are stable and reasonably damped [Astrom, 1980], and then to substitute in the poles desired for the closed loop system.

This condition on the dynamics of the closed loop system leads to a more restrictive form for numerator polynomial of module $q(s)$ than that used in Equation c-3 in that

$$\begin{aligned} q(s) &= \frac{(n_o^* + n_1^*s + \dots)(1 + D_1s + \dots + D_ns^n)}{1 + d_1s + d_2s^2 + \dots} \\ &= n^*(s) D(s) / d(s) \quad \dots c-8 \end{aligned}$$

Once again the polynomials, $n^*(s)$ and $d(s)$ are not explicitly limited to any particular order provided that $q(s)$ is realizable in practice, though it is common to keep them as simple as possible. The designer is free to choose the closed loop pole positions to give the required dynamics while keeping the input within allowable ranges.

It is easy to show that the matrix conditions on polynomial $n(s)$ can be converted to the following matrix conditions on polynomial $n^*(s)$

$$\begin{bmatrix} n_o^* \\ n_1^* \\ n_2^* \\ \dots \end{bmatrix} = \begin{bmatrix} 1 & 0 & 0 & : \\ D_1 & 1 & 0 & : \\ D_2 & D_1 & 1 & : \\ \dots & \dots & \dots & \dots \end{bmatrix}^{-1} \begin{bmatrix} n_o \\ n_1 \\ n_2 \\ \dots \end{bmatrix}$$

Type No:

Type 1

Type 2

Type 3

....c-9

where the dimensions of the matrices and vectors are once again set by the desired Type Number, as indicated. It should be noted that the order of polynomial $d(s)$ in module $q(s)$ must now equal or exceed the sum of the orders of polynomial $n^*(s)$ of module $q(s)$ and polynomial $D(s)$ of process model $g(s)$.

For the servo-motor system equation c-9 aims to cancel the slow servo-motor pole and to substitute in a faster pole for the closed loop dynamics. At the same time the resulting closed loop system must be Type 2 in order to track the triangular (ramp) variation in the reference input.

It is clear from the constraints imposed by type number equations given in chapter three that the numerator polynomial $n(s)$ of control module $q(s)$ is at least order 1. Since the process denominator $D(s)$ is order 1 then final order of the numerator for module $q(s)$ will be at least 2.

Thus, the denominator polynomial of module $q(s)$, which is also the denominator of the final closed loop system, has to be at least order 2. Also the system specifications call for a doubling of the speed of response so the closed loop poles are set to be at $s = -0.65 \pm j0.65$.

The corresponding polynomial for the closed loop characteristic function is then

$$d(s) = 1 + d_1s + d_2s^2 = 1 + 1.5385s + 1.1834s^2 \quad \dots c-10$$

Using this and type number constraining equations it is possible to compute the numerator polynomial factor $n^*(s)$ of control module $q(s)$ from the matrix equations

$$\begin{aligned} \begin{bmatrix} n_0 \\ n_1 \end{bmatrix} &= \begin{bmatrix} N_0 & 0 \\ N_1 & N_0 \end{bmatrix}^{-1} \begin{bmatrix} 1 & 0 \\ D_1 & 1 \end{bmatrix} \begin{bmatrix} 1 \\ d_1 \end{bmatrix} \begin{matrix} \text{Type 1} \\ \text{Type 2} \end{matrix} \\ &= \begin{bmatrix} 11.4 & 0 \\ 0 & 11.4 \end{bmatrix}^{-1} \begin{bmatrix} 1 & 0 \\ 1.3 & 1 \end{bmatrix} \begin{bmatrix} 1 \\ 1.5385 \end{bmatrix} \\ &= \begin{bmatrix} 0.0877 \\ 0.2490 \end{bmatrix} \quad \dots c-11 \end{aligned}$$

and finally

$$\begin{aligned} \begin{bmatrix} n^*_{0} \\ n^*_{1} \end{bmatrix} &= \begin{bmatrix} 1 & 0 \\ 1.3 & 1 \end{bmatrix}^{-1} \begin{bmatrix} 0.0877 \\ 0.2490 \end{bmatrix} \\ &= \begin{bmatrix} 0.0877 \\ 0.1350 \end{bmatrix} \end{aligned}$$

....c-12

The final control module, $q(s)$, is then given by

$$\begin{aligned} q(s) &= \frac{(0.0877 + 0.1350s)(1 + 1.3s)}{1 + 1.5385s + 1.1834s^2} \\ &= \frac{0.0877 + 0.2490s + 0.1755s^2}{1 + 1.5385s + 1.1834s^2} \end{aligned}$$

This control module ensure that the C/L system tracks ramp setpoints with zero error and has a transient decay rate twice that of the open loop system as shown in Figure c-5 given below.

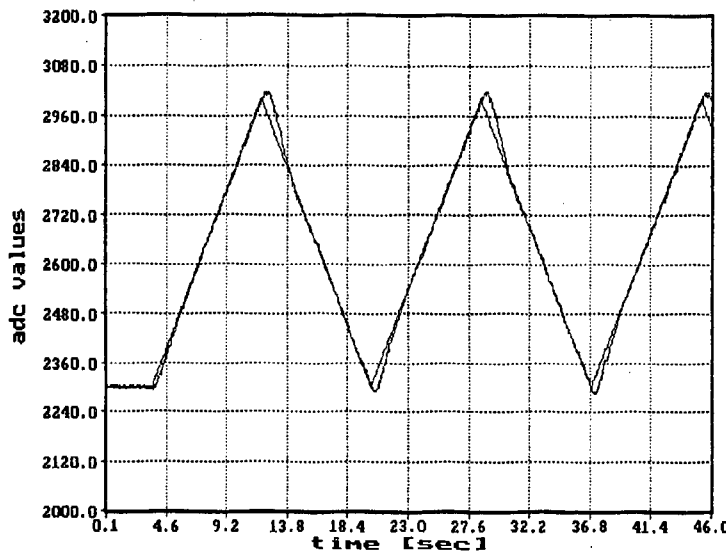


Figure C.5: IMC ramp response (Type II)
 $q(s)$ designed for ramp tracking

Its performance for step inputs is given in Figure C.6 below, from which it can be seen that the increase in type number has resulted in a less stable closed loop system than that needed for tracking step inputs.

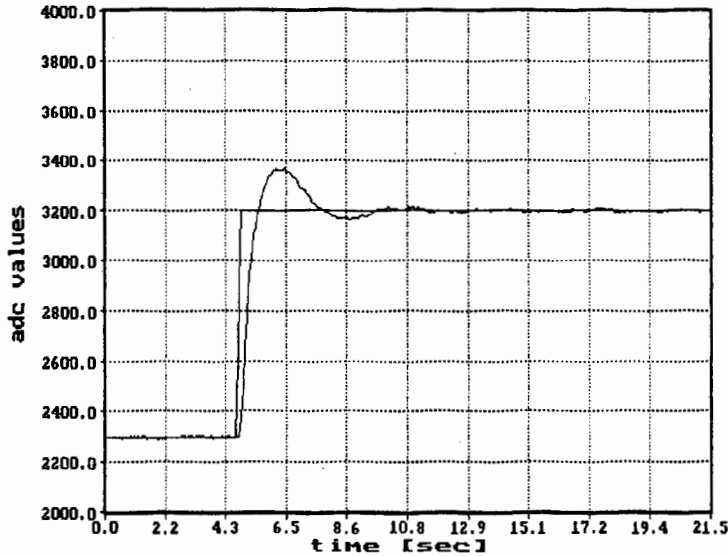


Figure C.6: IMC step response (Type II)
 $q(s)$ designed for ramp tracking

The traditional controller, $k(s)$, corresponding the control module $q(s)$ for the IMC design is

$$k(s) = \frac{0.0877 + 0.3630s + 0.4992s^2 + 0.2282s^3}{s^2 (1.1828 + 1.5384s)}$$

which is Type 2 as expected.

Hence, the design aims have been achieved and are much simpler than the derivative conditions on the error signal.

Appendix D

Transmittance & Disturbance Rejection Analysis

This appendix presents the Bode plots of the transmittance matrices obtained from considering the open-loop configuration of the system based on Q(Youla)-Parametrization which was implemented in the IMC-Structure. The figure D.1 given below shows the respective blocks of matrices which are useful for analyzing the extent to which the C/L system will reject disturbances and attenuate noise.

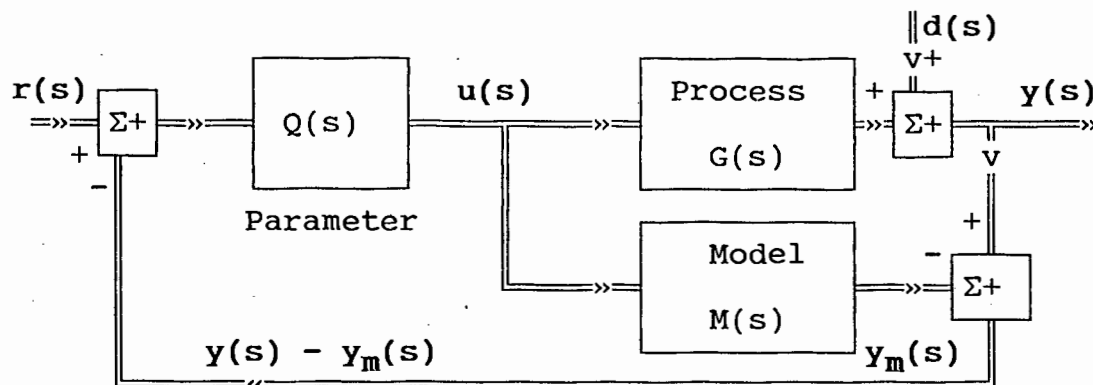


Figure D: C/L IMC-structure for Transmittance Analysis

Two matrices are presented, which are, the GQ-Matrix where $M^*(s)$ is assumed to be an exact model of $G(s)$, whence called, 'G- M^* matching' and the MQ-Matrix where, $M(s)$ is as used in the controller design of chapter five, which is assumed to be not equal to $G(s)$, whence called, 'G-M mismatch'. In this analysis, the unaltered model obtained from the fitting estimates to the recorded open loop step test data will be regarded as a close approximation of $G(s)$

such that they are interchangeable in the analysis for $G-M^*$ matching. The altered $M(s)$, as in chapter five will be used in the $G-M$ mismatch analysis. The reason for the above open loop based considerations of transmittance analysis is because of the C/L TFM simplification brought by the Q(Youla)-Parametrization of all stabilizing controllers which is implemented in the IMC-Structure. That is:-

$$H(s) = [I + G(s)K(s)]^{-1}G(s)K(s)$$

which becomes

$$H(s) = G(s)Q(s)$$

by defining the Q-parameter as

$$Q(s) = K(s)[I + G(s)K(s)]^{-1}$$

where $H(s)$ is the closed loop TFM. In reality, the design will be on a model $M(s)$ obtained from some data recorded from the actual process, as in this case. The $Q(s)$ must be defined as ($M(s)$ in equation below refers to both M^* and M)

$$Q(s) = K(s)[I + M(s)K(s)]^{-1}$$

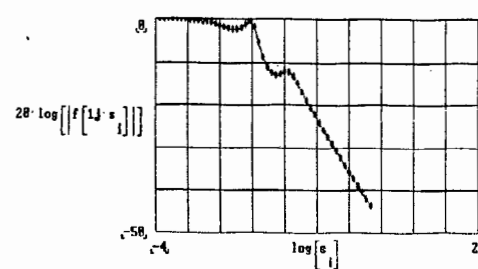
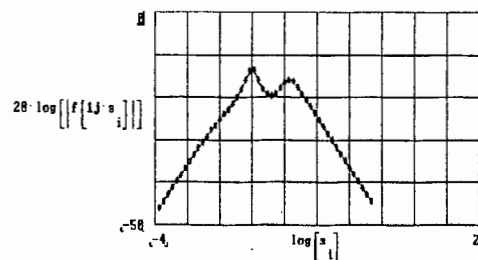
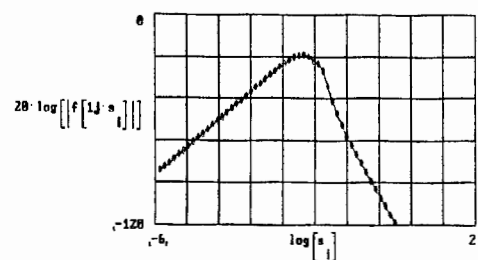
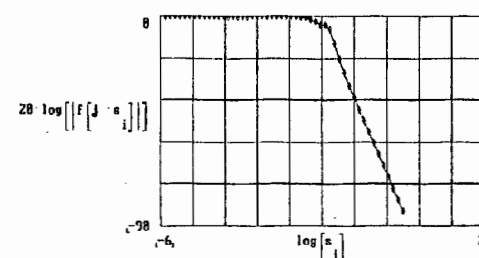
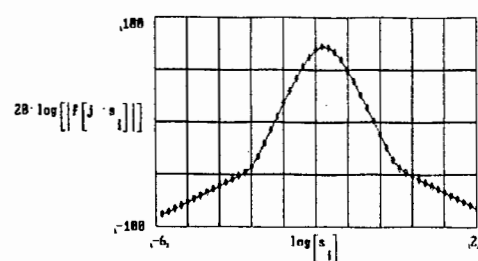
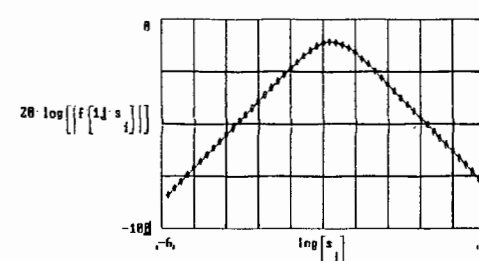
such that

$$K(s) = [I - Q(s)M(s)]^{-1}Q(s)$$

Hence, the analysis of Transmittance on the C/L TFM, $H(s)$ which is directly affected by $M(s)$ and $Q(s)$ using an exact $M(s)$ and an intentionally mismatched $M(s)$.

The Bode Plots were done using MATHCAD. Note that when the package was used, 's' was defined as $2\pi f$, but was multiplied by $1j$ to obtain $j2\pi f(j\omega)$. The plots are given below, where

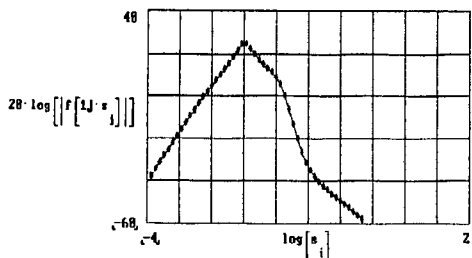
the first case is for 'G-M mismatch' and the second for 'G-M* match'.

D.1 Plant-Model mismatch Bode Plots $T_{11}(j2\pi f)$  $T_{12}(j2\pi f)$  $T_{21}(j2\pi f)$  $T_{22}(j2\pi f)$  $T_{31}(j2\pi f)$  $T_{32}(j2\pi f)$

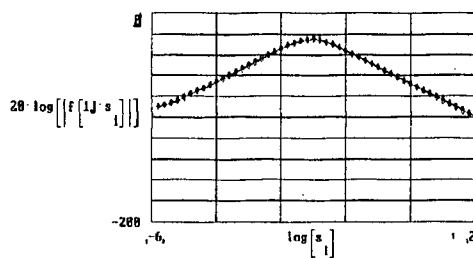
$$T_{41}(j2\pi f) = 0$$

$$T_{42}(j2\pi f) = 0$$

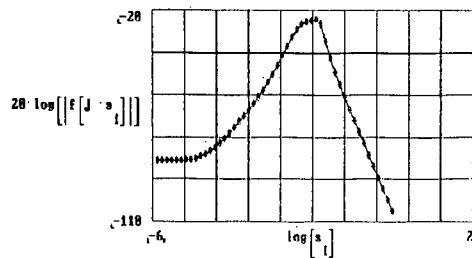
Figure D.1: Column 1: $T(s)$ - not MatchedFigure D.2: Column 2: $T(s)$ - not Matched



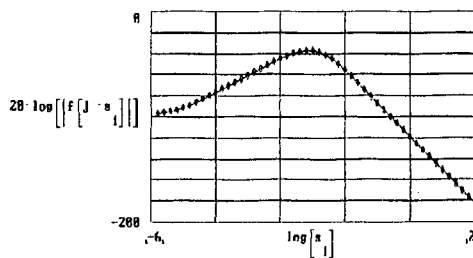
$T_{13}(j2\pi f)$



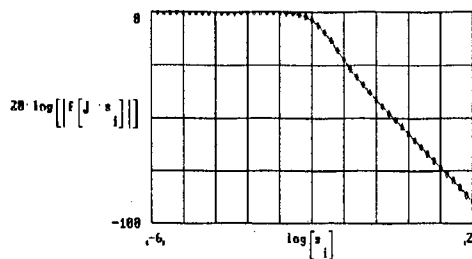
$T_{14}(j2\pi f)$



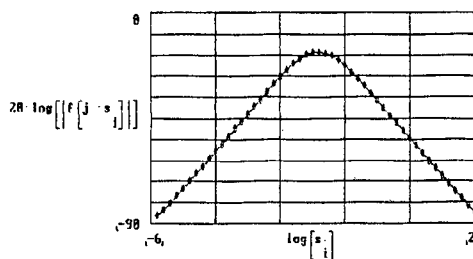
$T_{23}(j2\pi f)$



$T_{24}(j2\pi f)$

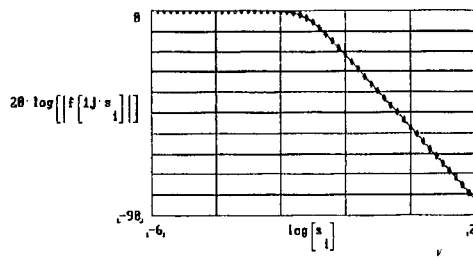


$T_{33}(j2\pi f)$



$T_{34}(j2\pi f)$

$T_{43}(j2\pi f) = 0$



$T_{44}(j2\pi f)$

Figure D.3: Column 3: $T(s)$ - not Matched

Figure D.4: Column 4: $T(s)$ - not Matched

The model, $M(s)$ was altered to maintain nominal performance and thereby keeps the complexity of Q minimal. The idea here originated from considering what the feedback signals are in the IMC-structure. This G-M intentional mismatch has an added advantage as compared to the G-M match. The reasons are:-

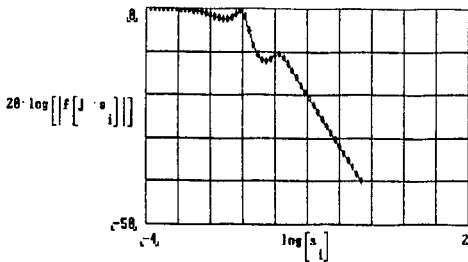
1. The Q -parameter is designed based on the exact model for the process, $G(s)$. This gives the GQ -path the required Transmittance for robustness.
2. The altered $M(s)$ was still implemented with the Q -parameter based on $G(s)$, and thus the controller forces the output response of the Plant to be closed to the desired response, otherwise:

if $M = G$ then, the response will be decoupled at low and high frequency but will still contain a significant amount of the unwanted characteristics of the open loop system, in view of the fact that, for $M = G$, the system will be open-loop (zero feedback).

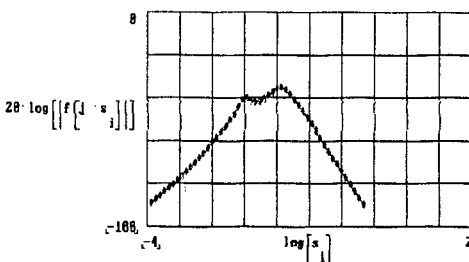
The MQ -matrix Bode plots which are given above have two-fold reasons. *Firstly*, to show how the disturbance and noise inputs would affect the system in the frequency domain, if the actual Plant, $G(s)$ would change to $M(s)$, while the same Q -parameter is used. *Secondly*, how closed-loop setpoint tracking and transient decoupling are affected if $G(s)$ in

the controller is intentionally changed from the $G(s)$ -model to $M(s)$.

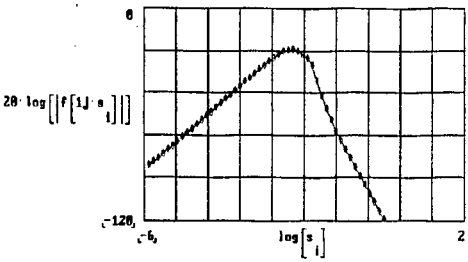
D.2 Plant-Model match Bode Plots



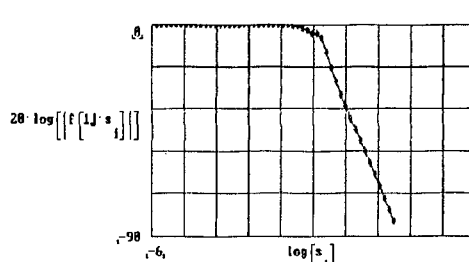
$T_{011}(j2\pi f)$



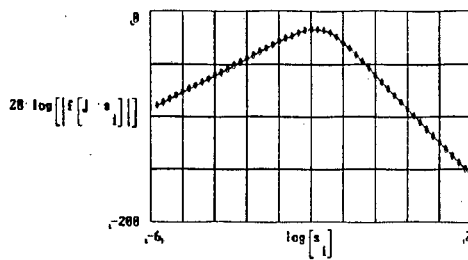
$T_{012}(j2\pi f)$



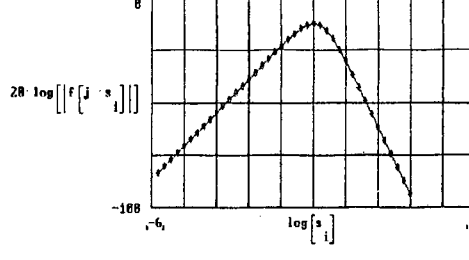
$T_{021}(j2\pi f)$



$T_{022}(j2\pi f)$



$T_{031}(j2\pi f)$



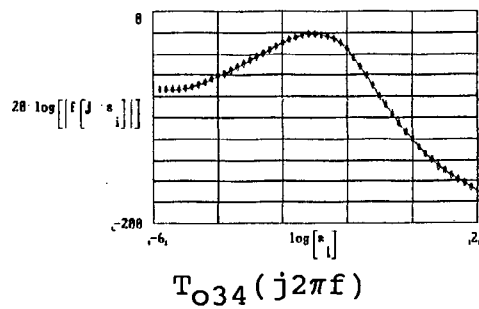
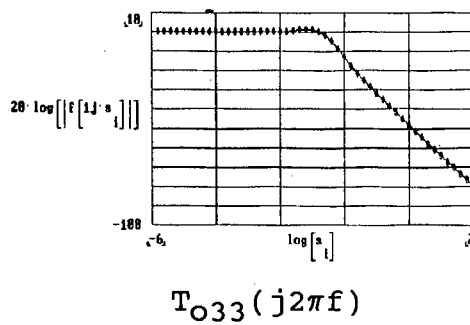
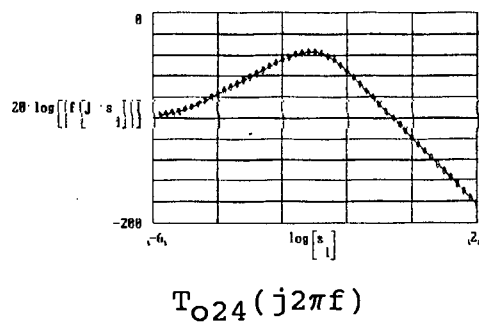
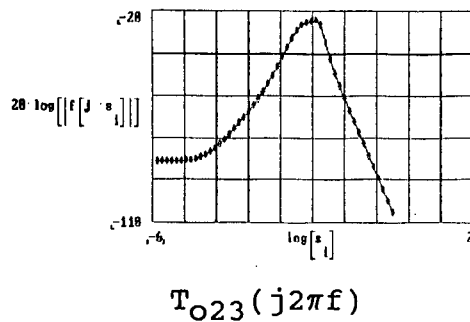
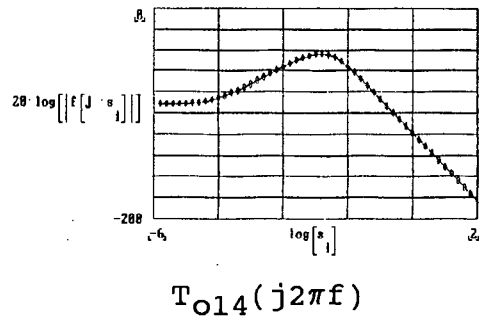
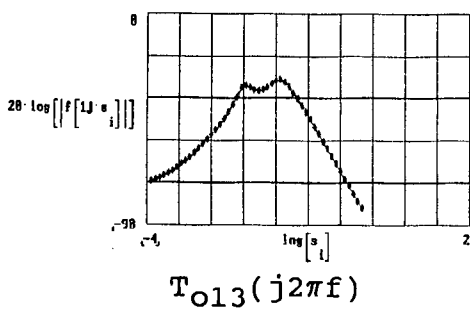
$T_{032}(j2\pi f)$

$T_{041}(j2\pi f) = 0$

$T_{042}(j2\pi f) = 0$

Figure D.4: Column 1: $T_0(s)$ - Matched

Figure D.6: Column 2: $T_0(s)$ - Matched



$$T_{O43}(j2\pi f) = 0$$

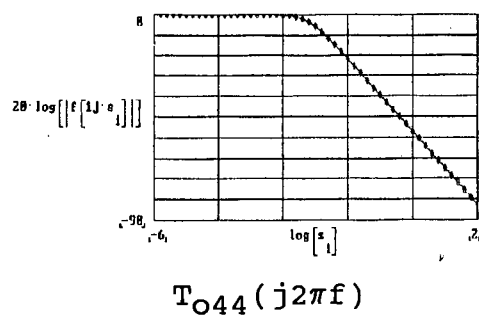


Figure D.7: Column 3: $T_O(s)$ - Matched

Figure D.8: Column 4: $T_O(s)$ - Matched

Bode plots are not given for those elements which from the original $G(s)$ -model were regarded as zero since their magnitudes on the Bode plots would be at $-\infty$ dB.

The diagonal plots show setpoint tracking at low frequencies. The frequency plot range either from 10^{-6} or 10^{-4} rad/s to 10^2 rad/s. It is also observable from the plots that all disturbances and noise with frequencies above 10^{-4} rad/s are greatly attenuated. The off diagonals show transient and steady state decoupling, since their magnitudes are low at both high and low frequencies. The highest peak is generally also below 0 dB, and is thus attenuated. It is also noted that the mid-band frequency amplitudes are higher relative to the high and low frequencies magnitudes because they were not compensated for in the proposed Q-parameter design method.

In conclusion, the following comparative remarks can be made.

a) Disturbance & Noise rejection:

Both the mismatched and matched transmittance matrices reflect the high and low decoupling which also attenuates disturbances and noise those frequencies. However, the mismatched Transmittance matrix shows mid-band frequency disturbance and noise amplification as can be seen from the off-diagonal elements, but NOT in the diagonal

elements. This corresponds to the expectation of interaction at intermediate frequencies.

b) Setpoint tracking & Decoupling. The following observations can be made:

The diagonal elements will track their respective setpoint. The off-diagonal elements will tend to zero at high and low frequencies, since their setpoints are not stepped but are at zero, thereby showing that the amount of decoupling is still acceptable.

What these plots don't show is that the responses of elements of the closed-loop system have been improved by intentionally mismatching the Plant, $G(s)$ and the model, $M(s)$. By using the Q (based on G)- $G(s)$ - $M(s)$ combination we achieve:-

- *- The design simplicity of the Q -parameter
- *- Improved response
- *- Acceptable decoupling and exact setpoint tracking
- *- Acceptable Disturbance and noise attenuation

Appendix E

FLOTRIG-Computer Interface Routines

The routines given in this appendix use 12-bit analogue-to-digital, DT2801 and digital-to-analogue, DT2815 interface cards. The index for the routines is given below.

Name of Procedure	Source Filename	Description of Procedure
adc_init	dt2801_r.pas	initialize the ADC and the DT2801 interface card
adc_read	dt2801_r.pas	reads the analog signal in the form of a low and high byte and convert it to one real value
dac_init	dt2815_r.pas	initializes the DAC and the DT2815 card
dac_write	dt2815_r.pas	writes the computed digital value to the analog process

E.1 The ADC interface routine

```
{*****}
{Procedure library for the Data Translation Card DT2801 }
{*****}
{Ref DT2801 Manual}
```

```
UNIT DT2801_R;
```

```
{*****}
      INTERFACE
{*****}
```

```
CONST
```

```
  datareg  = $02EC; {base address}
  statreg  = $02ED;
  cclear   = $0001;
  comm_wait = $4;
  read_wait = $5;
  write_wait = $2;
```

```

cstop    = $000F;
cerror   = $2;
cadin    = $000C;

```

```

PROCEDURE adc_init;
PROCEDURE adc_read(chan_no:integer;var value:real);

```

```

(*****)
      IMPLEMENTATION
(*****)

```

```

PROCEDURE adc_init;

```

```

{This procedure ensures that the analog-to-digital converter
is initialized before any further reading from the ADC can
take place}

```

```

VAR
  state : integer;

```

```

BEGIN
  port[statreg] := cstop;
  state := port[datareg];
  REPEAT
    state := port[statreg];
  UNTIL ((state and comm_wait) > 0);
  port[datareg] := cclear;
END;

```

```

PROCEDURE adc_read(chan_no:integer;var value:real);

```

```

{This procedure reads the analog input for the corresponding
signal required which is connected to that channel}

```

```

VAR
  high,low : integer;

```

```

BEGIN
  REPEAT UNTIL ((port[statreg] and comm_wait) > 0);
  port[statreg] := cadin;

```

```

  REPEAT UNTIL (((port[statreg] xor write_wait) and write_wait) > 0);
  port [datareg] := 0;

```

```

  REPEAT UNTIL (((port[statreg] xor write_wait) and write_wait) > 0);
  port[datareg] := chan_no;

```

```

  REPEAT UNTIL ((port[statreg] AND read_wait) > 0); {poll and wait for low byte}
  low := port[datareg];

```

```

  REPEAT UNTIL ((port[statreg] AND read_wait) > 0); {poll and wait for high byte}
  high := port[datareg];

```

```

  value := high*256.0 + low;    {determine the counts of the analog signal}

```

END; (adc_read)

```
{*****}
{***   INITIALIZATION SECTION   ***}
{*****}
```

BEGIN

END.

+

```
{*****}
```

```
PROCEDURE dac_write (chan_no:integer;value:integer);
```

{In this procedure, the digital counts are clamped between minimum and maximum values. Then the value is splitted in a low and high byte. The status register for the DAC is then polled until it is ready to receive the low and high bytes. Thereafter, the bytes are written to the port one after the other, but are interpreted as one signal}

```
VAR
```

```
new_val, high, low, ready_low : integer;
```

```
BEGIN
```

```
IF (value > 4095.0) THEN
```

```
new_val := 4095
```

```
ELSE IF (value < 0.0) THEN
```

```
new_val := 0
```

```
ELSE
```

```
new_val := (value);
```

```
high := new_val div 16;
```

```
low := 16*(new_val - 16*high) + 2*chan_no;
```

```
ready_low := port[dac_statreg];
```

```
REPEAT
```

```
ready_low := port[dac_statreg];
```

```
UNTIL ((ready_low AND write_wait) = $0);
```

```
IF (ready_low = 0) then
```

```
begin
```

```
port[dac_datareg] := low;
```

```
REPEAT UNTIL ((port[dac_statreg]) = $10);      (check H_byte & IBF)
```

```
port[dac_datareg] := high;
```

```
end;
```

```
END;
```

```
{*****}
```

```
{      INITIALIZATION      }
```

```
{*****}
```

```
BEGIN
```

```
END.
```

Appendix F

Listing of Routines for Step Tests

The listing given in this appendix can be used for open loop step tests, C/L FLORTIG control and PC-simulations. It can also be used to check whether the $Q(s)$ parameter designed for step setpoint tracking does not track ramp setpoints.

An index is given below for all the routines used in the program.

Name of Procedure	Source Filename	Description of Procedure
clamp	rig_qp5.pas	clamps the number which is out of bounds to within the min. & max limits given
clock	rig_qp5.pas	determines the real-time using the dos clock
graphdim	rig_qp5.pas	defines the graph min & max dimensions and labels
initialize	rig_qp5.pas	initializes most of the variables used
initpointers	rig_qp5.pas	initializes all pointers
initstates	rig_qp5.pas	initializes all the pointer states for computations
initdelays	rig_qp5.pas	initializes all delays
qpar	rig_qp5.pas	computes the $Q(s)$ parameter
plant	rig_qp5.pas	computes the nominal model $G(s)$
model	rig_qp5.pas	computes the model $M(s)$

Continues on the next page...

flotrig	rig_qp5.pas	outputs u(s) values to the FLOTRIG and returns y(s) values
servkeysim	rig_qp5.pas	handles keyboard requests and does C/L control plots and calculations
main program	rig_qp5.pas	runs all routines and plots the respective variables as required on separate graphs

F.1 Programme listing(Step)

```
program steps(input,output);
```

```
uses dos,
    crt,
    GRAPH,
    IGRAPH,
    RKUTTA,
    DT2801_R,DT2815_R;
```

```
type pvarr = array[1..4] of real;
```

```
var
    key,KeySimRig           :char;
    DACchNo, ADCchNo, ADCstep,chno :integer;
    rep0,rep1,rep2,rep3      : integer;
    SetPoint,valueR,valueW   :integer;
    time, Told, Tnew, dT     :real;
    Output_R, Value         :real;
    hour, min, sec, sec100   :word;
    freq, count             :word;
    Recfil,Clefil,Scafil,Roufil :TEXT;
    FADCname                 :STRING[32];
    LOG, RampOn, FileOpen    :BOOLEAN;
    t                        :real;
    checkfil                : integer;
    gs11,gs12,gs13          : ptr;
    gs21,gs22,gs23          : ptr;
    gs31,gs32,gs33,gs34 : ptr;
                        gs44 : ptr;

    ms11,ms12,ms13          : ptr;      {11=Rou, 22=Sca, 33=Cle, 44=Rec}
    ms21,ms22,ms23          : ptr;
    ms31,ms32,ms33,ms34 : ptr;
                        ms44 : ptr;

    qs11,qs12,qs13,qs14 : ptr;      {11=Rou, 22=Sca, 33=Cle, 44=Rec}
```

```

qs21,qs22,qs23,qs24 : ptr;
qs31,qs32,qs33,qs34 : ptr;
      qs44 : ptr;

      inv11 : ptr;
      inv22 : ptr;
      inv33 : ptr;
      inv44 : ptr;

      filt1 : ptr;
      filt2 : ptr;
      filt3 : ptr;
      filt4 : ptr;

mdly12 : array[1..30] of real;
mdly13 : array[1..30] of real;
mdly31 : array[1..15] of real;
mdly32 : array[1..45] of real;
mdly34 : array[1..20] of real;
mdly44 : array[1..5] of real;

gdly12 : array[1..30] of real;
gdly13 : array[1..30] of real;
gdly31 : array[1..15] of real;
gdly32 : array[1..45] of real;
gdly34 : array[1..20] of real;
gdly44 : array[1..5] of real;

setp  : array[1..4,1..600] of real;
u_ctrl : array[1..4,1..600] of real;
ym_out : array[1..4,1..600] of real;
yp_out : array[1..4,1..600] of real;

cell,point : integer;

{*****}

Procedure Clamp(var val:real; min, max:real);

{This procedure clamps the computed controller output to
the minimum(min) and maximum(max) as specified}

begin
  if val < min then val := min;
  if val > max then val := max;
end;

{*****}

Procedure clock(var hrs, min, sec, sec100:word; var time:real);

```

```
{ Uses the PC clock timing to calculate real-time}
```

```
begin
  time := (3600*hrs + 60*min + sec + 0.01*sec100);
end;

{*****}
```

```
Procedure graphdim;
```

```
{ Defines the graph dimensions for the initial plot}
```

```
begin
  ymax := 100;
  ymin := -30;
  xmax := 600;
  xmin := 0;
  yint := 10;
  xint := 60;
  xlabl := 'time (s)';
  ylabl := 'Level (%)';
  head := 'FLOTATION PROCESS SIMULATION RIG (C/L)';
end;
```

```
{*****}
```

```
Procedure Initialize;
```

```
{Initialises almost all used variables}
```

```
var inp0,inp1,inp2,inp3 :integer;
```

```
begin
  dac_init;
  adc_init;
  RampOn := FALSE;
  LOG := FALSE;
  inp0 := 2048; {2048=0%, 2560=25%, 3072=50%, 3584=75%, 4095=100%}
  dac_write(0,inp0);
  inp1 := 1024;
  dac_write(1,inp1);
  inp2 := 1024;
  dac_write(2,inp2);
  inp3 := 1024;
  dac_write(3,inp3);
  key := ' ';
end; {initialize}
```

```
{*****}
```

```
Procedure initpointers;
```

{Initialises all pointers used in the program}

```
begin
  new(qs11);new(qs12);new(qs13);
  new(qs21);new(qs22);new(qs23);
  new(qs31);new(qs32);new(qs33);new(qs34);
                                new(qs44);
  new(ms11);new(ms12);new(ms13);
  new(ms21);new(ms22);new(ms23);
  new(ms31);new(ms32);new(ms33);new(ms34);
                                new(ms44);

  new(qs11);new(qs12);new(qs13);new(qs14);
  new(qs21);new(qs22);new(qs23);new(qs24);
  new(qs31);new(qs32);new(qs33);new(qs34);
                                new(qs44);

  new(filt1);
  new(filt2);
  new(filt3);
  new(filt4);
```

end;

{*****}

Procedure initstates;

{Initialises all states of the pointers used to zero}

```
begin
  Clearstates(qs11); Clearstates(qs12); Clearstates(qs13);
  Clearstates(qs21); Clearstates(qs22); Clearstates(qs23);
  Clearstates(qs31); Clearstates(qs32); Clearstates(qs33);Clearstates(qs34);
                                Clearstates(qs44);
  Clearstates(ms11); Clearstates(ms12); Clearstates(ms13);
  Clearstates(ms21); Clearstates(ms22); Clearstates(ms23);
  Clearstates(ms31); Clearstates(ms32); Clearstates(ms33);Clearstates(ms34);
                                Clearstates(ms44);

  Clearstates(qs11); Clearstates(qs12); Clearstates(qs13);Clearstates(qs14);
  Clearstates(qs21); Clearstates(qs22); Clearstates(qs23);Clearstates(qs24);
  Clearstates(qs31); Clearstates(qs32); Clearstates(qs33);Clearstates(qs34);
                                Clearstates(qs44);

  Clearstates(filt1);
  Clearstates(filt2);
  Clearstates(filt3);
  Clearstates(filt4);
end;
```

{*****}

Procedure initdelays;

{Initialises all the arrays for the delays to zero}

```

var z : integer;
    c : integer;
begin
  for z := 1 to 30 do
    begin
      mdly12[z] := 0;
      gdly12[z] := 0;
      mdly13[z] := 0;
      gdly13[z] := 0;
    end;

    for z := 1 to 15 do
      begin
        mdly31[z] := 0;
        gdly31[z] := 0;
      end;

      for z := 1 to 45 do
        begin
          mdly32[z] := 0;
          gdly32[z] := 0;
        end;

        for z := 1 to 20 do
          begin
            mdly34[z] := 0;
            gdly34[z] := 0;
          end;

          for z := 1 to 5 do
            begin
              mdly44[z] := 0;
              gdly44[z] := 0;
            end;

            for z := 1 to 4 do
              for c := 1 to 600 do
                begin
                  setp[z,c] := 0;
                  u_ctrl[z,c] := 0;
                  ym_out[z,c] := 0;
                  yp_out[z,c] := 0;
                end;
              end;
            end;
          end;
        end;
      end;
    end;
  end;
end;

```

{*****}

```
Procedure Qpar(aq:pvarr; var cq: pvarr);
```

{This procedure is for the Q-Parameter which is implemented as

$$Q(s) = [sQ_{\infty} + Q_0]F(s)$$

where $Q(s)$ is as defined in the main body of the thesis. This Qpar procedure and the Model procedure given further below form the controller which is implemented in the IMC-Structure. The output of this procedure is the exactly the model input but is adjusted in the FLOTTRIG procedure because the process zero reference is not the same as that of the Model.

The specifications and computations in this procedure are as follows:

```
'ordN'      means order of the numerator
'ordD'      means order of the denominator
'coeffN[i]' means real coefficient of  $s^i$  for the numerator
              polynomial, similarly for the denominator}
```

```
var
```

```
  cqa,cqb,cqc,cqd : real;
  i                : integer;
```

```
begin
```

```
with qs11^ do
begin
  deadtime := 0;
  ordN := 1;
  ordD := 1;
  coeffN[0] := 0.826;
  coeffN[1] := 37.2914;
  coeffD[0] := 1;
  coeffD[1] := 100;
end;
```

```
with qs12^ do
begin
  deadtime := 0;
  ordN := 1;
  ordD := 1;
  coeffN[0] := -2.7345;
  coeffN[1] := -66.8722;
  coeffD[0] := 1;
  coeffD[1] := 100;
end;
```

```
with qs13^ do
begin
  deadtime := 0;
  ordN := 1;
  ordD := 1;
  coeffN[0] := 1.9730;
  coeffN[1] := 14.5307;
  coeffD[0] := 1;
  coeffD[1] := 100;
end;
```

```
with qs14^ do
begin
  deadtime := 0;
  ordN := 1;
  ordD := 1;
  coeffN[0] := 0.8221;
  coeffN[1] := 17.2984;
  coeffD[0] := 1;
  coeffD[1] := 100;
end;
```

```
with qs21^ do
begin
  deadtime := 0;
  ordN := 1;
  ordD := 1;
  coeffN[0] := 0.0217;
  coeffN[1] := 36.7055;
  coeffD[0] := 1;
  coeffD[1] := 100;
end;
```

```
with qs22^ do
begin
  deadtime := 0;
  ordN := 1;
  ordD := 1;
  coeffN[0] := 0.1137;
  coeffN[1] := 59.2393;
  coeffD[0] := 1;
  coeffD[1] := 100;
end;
```

```
with qs23^ do
begin
  deadtime := 0;
  ordN := 1;
  ordD := 1;
  coeffN[0] := 1.4133;
  coeffN[1] := 18.1236;
  coeffD[0] := 1;
  coeffD[1] := 100;
end;
```

```
with qs24^ do
begin
  deadtime := 0;
  ordN := 1;
  ordD := 1;
  coeffN[0] := 0.5889;
  coeffN[1] := 21.5758;
  coeffD[0] := 1;
  coeffD[1] := 100;
```

```
end;

with qs31^ do
begin
  deadtime := 0;
  ordN := 1;
  ordD := 1;
  coeffN[0] := -1.159;
  coeffN[1] := -27.1855;
  coeffD[0] := 1;
  coeffD[1] := 100;
end;

with qs32^ do
begin
  deadtime := 0;
  ordN := 1;
  ordD := 1;
  coeffN[0] := -0.1853;
  coeffN[1] := -14.4753;
  coeffD[0] := 1;
  coeffD[1] := 100;
end;

with qs33^ do
begin
  deadtime := 0;
  ordN := 1;
  ordD := 1;
  coeffN[0] := 1.0683;
  coeffN[1] := 15.2530;
  coeffD[0] := 1;
  coeffD[1] := 100;
end;

with qs34^ do
begin
  deadtime := 0;
  ordN := 1;
  ordD := 1;
  coeffN[0] := 0.4451;
  coeffN[1] := 18.1584;
  coeffD[0] := 1;
  coeffD[1] := 150;
end;

with qs44^ do
begin
  deadtime := 0;
  ordN := 1;
  ordD := 1;
  coeffN[0] := 2.7778;
  coeffN[1] := 264.5503;
  coeffD[0] := 1;
```

```

    coeffD[1] := 150;
end;

rk(qs11,aq[1],dT,cqa); {Given the element  $q_{11}(s)$ , the
                        present input and sampling interval
                        calculate the output}
rk(qs12,aq[2],dT,cqb);
rk(qs13,aq[3],dT,cqc);
rk(qs14,aq[4],dT,cqd);
cq[1] := (cqa + cqb + cqc + cqd); {Sum the respective outputs of each row element}

rk(qs21,aq[1],dT,cqa);
rk(qs22,aq[2],dT,cqb);
rk(qs23,aq[3],dT,cqc);
rk(qs24,aq[4],dT,cqd);
cq[2] := (cqa + cqb + cqc + cqd);

rk(qs31,aq[1],dT,cqa);
rk(qs32,aq[2],dT,cqb);
rk(qs33,aq[3],dT,cqc);
rk(qs34,aq[4],dT,cqd);
cq[3] := (cqa + cqb + cqc + cqd);

rk(qs44,aq[4],dT,cqd);
cq[4] := (cqd);

end; { end of proc Qpar}

{*****}

Procedure Plant(up:pvarr; var yp:pvarr);

{This procedure is represents the nominal model of the actual Plant,  $G(s)$ . This Plant procedure, the
Qpar procedure given above and the Model procedure given further below are used to perform a digital
simulation of the actual which is implemented in the IMC-Structure.

The specifications and computations in this procedure are as follows:
' ordN'      means order of the numerator
' ordD'      means order of the denominator
'coeffN[i]' means real coefficient of  $s^i$  for the numerator
              polynomial, similarly for the denominator}

var
    ypa,ypb,ypc,ypd : real;
    i                 : integer;

begin
    with qs11^ do
        begin
            deadtime := 0;
            ordN := 1;
            ordD := 2;
            coeffN[0] := 0.005*0.009;
            coeffN[1] := 0.009;

```

```
    coeffD[0] := 0.0006;
    coeffD[1] := 0.05;
    coeffD[2] := 1;
end;

with gs12^ do
begin
    deadtime := 30;
    ordN := 0;
    ordD := 1;
    coeffN[0] := 0.00625;
    coeffD[0] := 0.0125;
    coeffD[1] := 1;
end;

with gs13^ do
begin
    deadtime := 30;
    ordN := 0;
    ordD := 1;
    coeffN[0] := -0.016;
    coeffD[0] := 0.02;
    coeffD[1] := 1;
end;

with gs21^ do
begin
    deadtime := 0;
    ordN := 0;
    ordD := 1;
    coeffN[0] := -0.01625/2;
    coeffD[0] := 0.025;
    coeffD[1] := 1;
end;

with gs22^ do
begin
    deadtime := 0;
    ordN := 0;
    ordD := 1;
    coeffN[0] := 0.00744;
    coeffD[0] := 0.012;
    coeffD[1] := 1;
end;

with gs23^ do
begin
    deadtime := 0;
    ordN := 0;
    ordD := 1;
    coeffN[0] := -0.0011;
    coeffD[0] := 0.005;
    coeffD[1] := 1;
end;
```

```
with gs31^ do
begin
  deadtime := 15;
  ordN := 1;
  ordD := 2;
  coeffN[0] := 0.0009*0.016667/2;
  coeffN[1] := 0.016667/2;
  coeffD[0] := 0.0003;
  coeffD[1] := 0.035;
  coeffD[2] := 1;
end;
```

```
with gs32^ do
begin
  deadtime := 45;
  ordN := 0;
  ordD := 1;
  coeffN[0] := 0.0182;
  coeffD[0] := 0.028;
  coeffD[1] := 1;
end;
```

```
with gs33^ do
begin
  deadtime := 0;
  ordN := 1;
  ordD := 2;
  coeffN[0] := 0.0005*0.036;
  coeffN[1] := 0.036;
  coeffD[0] := 0.0006;
  coeffD[1] := 0.05;
  coeffD[2] := 1;
end;
```

```
with gs34^ do
begin
  deadtime := 20;
  ordN := 0;
  ordD := 1;
  coeffN[0] := -0.009/2;
  coeffD[0] := 0.03;
  coeffD[1] := 1;
end;
```

```
with gs44^ do
begin
  deadtime := 5;
  ordN := 0;
  ordD := 1;
  coeffN[0] := 0.00756/2;
  coeffD[0] := 0.0105;
  coeffD[1] := 1;
end;
```

```

gdly12[1] := up[2];
gdly13[1] := up[3];
gdly31[1] := up[1];
gdly32[1] := up[2];
gdly34[1] := up[4];
gdly44[1] := up[4];

for i := 30 downto 2 do gdly12[i] := gdly12[i-1];
for i := 30 downto 2 do gdly13[i] := gdly13[i-1];
for i := 15 downto 2 do gdly31[i] := gdly31[i-1];
for i := 45 downto 2 do gdly32[i] := gdly32[i-1];
for i := 20 downto 2 do gdly34[i] := gdly34[i-1];
for i := 5 downto 2 do gdly44[i] := gdly44[i-1];

rk(qs11,up[1],dT,ypa);
rk(qs12,gdly12[30],dT,ypb);
rk(qs13,gdly13[30],dT,ypc);
yp[1] := (ypa + ypb + ypc);

rk(qs21,up[1],dT,ypa);
rk(qs22,up[2],dT,ypb);
rk(qs23,up[3],dT,ypc);
yp[2] := (ypa + ypb + ypc);

rk(qs31,gdly31[15],dT,ypa);
rk(qs32,gdly32[45],dT,ypb);
rk(qs33,up[3],dT,ypc);
rk(qs34,gdly34[20],dT,ypd);
yp[3] := (ypa + ypb + ypc + ypd);

rk(qs44,gdly44[5],dT,ypd);
yp[4] := (ypd);

end; { end of proc Plant}

{*****}

Procedure Model(um:pvarr; var ymm:pvarr);

{This procedure is represents the mismatched model of the nominal model, G(s).

The specifications and computations in this procedure are as follows:
' ordN'      means order of the numerator
' ordD'      means order of the denominator
'coeffN[i]'  means real coefficient of  $s^{-1}$  for the numerator
              polynomial, similarly for the denominator}

var
  yma,ymb,ymc,ymd : real;
  i                : integer;

begin

```

```
with ms11^ do
begin
  deadtime := 0;
  ordN := 0;
  ordD := 2;
  coeffN[0] := 0.15*0.0006/2;
  coeffN[1] := 0.018/2;
  coeffD[0] := 0.0006;
  coeffD[1] := 0.05;
  coeffD[2] := 1;
end;
```

```
with ms12^ do
begin
  deadtime := 30;
  ordN := 0;
  ordD := 1;
  coeffN[0] := 0.00625;
  coeffD[0] := 0.0125;
  coeffD[1] := 1;
end;
```

```
with ms13^ do
begin
  deadtime := 30;
  ordN := 0;
  ordD := 1;
  coeffN[0] := -0.016;
  coeffD[0] := 0.02;
  coeffD[1] := 1;
end;
```

```
with ms21^ do
begin
  deadtime := 0;
  ordN := 0;
  ordD := 1;
  coeffN[0] := -0.01625/2;
  coeffD[0] := 0.025;
  coeffD[1] := 1;
end;
```

```
with ms22^ do
begin
  deadtime := 0;
  ordN := 0;
  ordD := 1;
  coeffN[0] := 0.00744;
  coeffD[0] := 0.012;
  coeffD[1] := 1;
end;
```

```
with ms23^ do
begin
```

```
    deadtime := 0;
    ordN := 0;
    ordD := 1;
    coeffN[0] := -0.0011;
    coeffD[0] := 0.005;
    coeffD[1] := 1;
end;

with ms31^ do
begin
    deadtime := 15;
    ordN := 0;
    ordD := 2;
    coeffN[0] := 0.05*0.0003/2;
    coeffN[1] := 0.016667/2;
    coeffD[0] := 0.0003;
    coeffD[1] := 0.035;
    coeffD[2] := 1;
end;

with ms32^ do
begin
    deadtime := 45;
    ordN := 0;
    ordD := 1;
    coeffN[0] := 0.0182;
    coeffD[0] := 0.028;
    coeffD[1] := 1;
end;

with ms33^ do
begin
    deadtime := 0;
    ordN := 0;
    ordD := 2;
    coeffN[0] := 0.0006*0.03;
    coeffN[1] := 0.036;
    coeffD[0] := 0.0006;
    coeffD[1] := 0.05;
    coeffD[2] := 1;
end;

with ms34^ do
begin
    deadtime := 20;
    ordN := 0;
    ordD := 1;
    coeffN[0] := -0.009/2;
    coeffD[0] := 0.03;
    coeffD[1] := 1;
end;

with ms44^ do
begin
```

```

    deadtime := 5;
    ordN := 0;
    ordD := 1;
    coeffN[0] := 0.00756/2;
    coeffD[0] := 0.0105;
    coeffD[1] := 1;
end;

mdly12[1] := um[2];
mdly13[1] := um[3];
mdly31[1] := um[1];
mdly32[1] := um[2];
mdly34[1] := um[4];
mdly44[1] := um[4];

for i := 30 downto 2 do mdly12[i] := mdly12[i-1];
for i := 30 downto 2 do mdly13[i] := mdly13[i-1];
for i := 15 downto 2 do mdly31[i] := mdly31[i-1];
for i := 45 downto 2 do mdly32[i] := mdly32[i-1];
for i := 20 downto 2 do mdly34[i] := mdly34[i-1];
for i := 5 downto 2 do mdly44[i] := mdly44[i-1];

rk(ms11,um[1],dT,ya);
rk(ms12,mdly12[30],dT,yb);
rk(ms13,mdly13[30],dT,yc);
ymm[1] := (ya + yb + yc);

rk(ms21,um[1],dT,ya);
rk(ms22,um[2],dT,yb);
rk(ms23,um[3],dT,yc);
ymm[2] := (ya + yb + yc);

rk(ms31,mdly31[15],dT,ya);
rk(ms32,mdly32[45],dT,yb);
rk(ms33,um[3],dT,yc);
rk(ms34,mdly34[20],dT,yd);
ymm[3] := (ya + yb + yc + yd);

rk(ms44,mdly44[5],dT,yd);
ymm[4] := (yd);

end; { end of proc model}

{*****}

Procedure PlotRig(urig : pvarr; var yrig: pvarr);

{This procedure receives the Q-Parameter output as computes by the Qpar procedure. It then adjusts
the respective u(s) values,('urig[i]')', to the zero references of the actual plant and outputs the
values to the FLOTIRIG}

var chnoV,j      : integer;
    level        : real;

```

```

position      : integer;
offs          : array[1..4] of integer;
factr         : array[1..4] of real;
percoff       : array[1..4] of real;

```

```
{0=Rec, 1=Cle, 2=Sca, 3=Rou for valves=V}
```

```
{4=Rec, 5=Cle, 6=Sca, 7=Rou for levels=L}
```

```
begin
```

```
  offs[4] := 2000; {1=Rou, 2=Sca, 3=Cle, 4=Rec}
```

```
  offs[3] := 2185;
```

```
  offs[2] := 2550;
```

```
  offs[1] := 2550;
```

```
  factr[4] := 6.6;
```

```
  factr[3] := 7.94;
```

```
  factr[2] := 7;
```

```
  factr[1] := 8;
```

```
  percoff[4] := 0;
```

```
  percoff[3] := 0;
```

```
  percoff[2] := 0;
```

```
  percoff[1] := 0;
```

```
for chnoV := 3 downto 0 do
```

```
  begin
```

```
    position := trunc(2048 + 20.48*urig[4-chnoV]);
```

```
    dac_write(chnoV, position);
```

```
    adc_read(chnoV + 4, level);
```

```
    yrig[4-chnoV] := (level - offs[4-chnoV])/factr[4-chnoV];
```

```
    with filt1^ do
```

```
      begin
```

```
        deadtime := 0;
```

```
        ordN := 2;
```

```
        ordD := 2;
```

```
        coeffN[2] := 1;
```

```
        coeffN[1] := 0;
```

```
        coeffN[0] := sqr(2*3.1415*7/240);
```

```
        coeffD[0] := sqr(2*3.1415*7/240);
```

```
        coeffD[1] := 2*0.707*(2*3.1415*7/240);
```

```
        coeffD[2] := 1;
```

```
      end;
```

```
    with filt2^ do
```

```
      begin
```

```
        deadtime := 0;
```

```
        ordN := 0;
```

```
        ordD := 1;
```

```
        coeffN[0] := 1;
```

```
        coeffD[0] := 1;
```

```
        coeffD[1] := 1;
```

```
      end;
```

```
    with filt3^ do
```

```
      begin
```

```

    deadtime := 0;
    ordN := 0;
    ordD := 1;
    coeffN[0] := 1;
    coeffD[0] := 1;
    coeffD[1] := 1;
end;

with filt4^ do
begin
    deadtime := 0;
    ordN := 0;
    ordD := 1;
    coeffN[0] := 1;
    coeffD[0] := 1;
    coeffD[1] := 1;
end;
rk(filt1,yrig[1],dT,yrig[1]);
rk(filt2,yrig[2],dT,yrig[2]);
rk(filt3,yrig[3],dT,yrig[3]);
rk(filt4,yrig[4],dT,yrig[4]);
end;
end;

{*****}

```

Procedure ServKeySim;

{This procedure deals with most of the requests obtained from the keyboardsuch as

- To select a PLOTTRIG cell to step
- To perform open loop step tests
- To switch the controller in (Closed Loop)
- To switch the controller out (Open Loop)
- To plot data for the required cell
- To perform closed loop step tests}
- To check the system for ramp tracking with a Q(s) designed for step setpoint tracking}

```

var chno, j, ctr_in      : integer;
    choice, Key_pressed : char;
    exit                : char;
    level,step          : real;
    i, ex,wy           : integer;
    tex                 : array[1..4] of string;
    a,b,c,y,ym,r,u,aold : pvarr;
    uold, ain, uflot    : pvarr;

    factr               : array[1..4] of real;
    percoff             : array[1..4] of real;
    offs                : array[1..4] of integer;
    Qcontrol            : char;
    row,col             : integer;

```

```

begin

```

```

r[1] := 0; a[1] := 0; uold[1] := -50; b[1] := 0; c[1] := 0; u[1] := 0; ym[1] := 0; y[1] := 0;
uflot[1] := 0;
r[2] := 0; a[2] := 0; uold[2] := -50; b[2] := 0; c[2] := 0; u[2] := 0; ym[2] := 0; y[2] := 0;
uflot[2] := 0;
r[3] := 0; a[3] := 0; uold[3] := -50; b[3] := 0; c[3] := 0; u[3] := 0; ym[3] := 0; y[3] := 0;
uflot[3] := 0;
r[4] := 0; a[4] := 0; uold[4] := 0; b[4] := 0; c[4] := 0; u[4] := 0; ym[4] := 0; y[4] := 0;
uflot[4] := 0;

tex[4] := 'L4';
tex[3] := 'L3';
tex[2] := 'L2';
tex[1] := 'L1';
initstates; {clear the states}
initdelays; {initialise the arrays for delays}
clrscr;

repeat
  clrscr;
  choice := ' ';
  key_pressed := ' ';

  rep0 := 0; rep1 := 0; rep2 := 0; rep3 := 0;

  time := 0;
  gotoxy(15,5);
  write('_____');
  TextAttr := 112;
  if KeySimRig = 'S' then
    write(' MAIN MENU (Simulation) ')
  else
    write(' MAIN MENU (Plot. Rig ) ');
  TextAttr := 7;
  writeln('_____'); gotoxy(15,6);
  writeln(' Press <L> - for logging data '); gotoxy(15,7);
  writeln(' (toggles on / off ) '); gotoxy(15,8);
  writeln(' <0> - step test on RECLEANER '); gotoxy(15,9);
  writeln(' <1> - step test on CLEANER '); gotoxy(15,10);
  writeln(' <2> - step test on SCAVENGER '); gotoxy(15,11);
  writeln(' <3> - step test on ROUGHER '); gotoxy(15,12);
  writeln(' <A> - step tests on all cells '); gotoxy(15,13);
  writeln(' <U/D> - to step(u/d) a chosen cell '); gotoxy(15,14);
  writeln(' <R> - for ramp tests on cells '); gotoxy(15,15);
  writeln(' <E> - To Quit (EXIT) '); gotoxy(15,16);
  writeln(' '); gotoxy(15,17);
  writeln(' Enter choice : ');
  repeat
    gotoxy(39,17); clreol;
    readln(choice);
    choice := upcase(choice);
    if not(choice in ['0'..'3', 'E']) then
      begin
        for i:= 1 to 8 do
          begin

```

```

    Sound(1000+100*i);
    delay(100);
    NoSound;
end;
end;
until choice in ['0'..'3','E'];

if choice = 'E' then exit := choice;

if choice in ['0'..'3'] then
begin
    case choice of
        '0' : foot := 'L = Log : <RECLEANER> : Q = Quit , C = C/Loop';
        '1' : foot := 'L = Log : <CLEANER> : Q = Quit , C = C/Loop';
        '2' : foot := 'L = Log : <SCAVENGER> : Q = Quit , C = C/Loop';
        '3' : foot := 'L = Log : <ROUGHER> : Q = Quit , C = C/Loop';
    end;

    settime(0,0,0,0); {initial value of time}
    time := 0;
    Told := 0;
    ctr_in := 0;
    col := 0;
    Qcontrol := 'N';
    RampOn := false;

    InstGraph('c:');
    ScrSetUp;
    CalcScale;
    Axes(Show);

    repeat
        if col <= 600 then
            begin
                gettime(Hour,min,sec,sec100);
                clock(Hour,min,sec,sec100,time);
                delay(950);

                if keypressed then key_pressed := upcase(readkey);

                case key_pressed of
                    'C': Qcontrol := 'C';
                    'N': begin
                        Qcontrol := 'N';
                        ctr_in := 0;
                    end;

                    'L': LOG := not LOG;
                end; {case key_pressed}

                (* Insert controller and close the loop *)
            end
        end
    until key_pressed = 'E';
end;

```

```

if Qcontrol = 'C' then
begin
  for j := 1 to 4 do
  begin
    if ctr_in = 0 then
    begin
      r[j] := y[j];
      Told := y[52-ord(choice)];
    end;
    b[j] := (y[j] - ym[j]);
    a[j] := r[j] - b[j];
  end; { for loop}

  if ctr_in = 0 then ctr_in := ctr_in + 2;

  Qpar(a,u);
  for j := 1 to 4 do u[j] := u[j];
  for j := 1 to 4 do clamp(u[j],-100,100);
  if KeySimRig = 'S' then
  begin
    Plant(u,y);      {Simulation model of the rig}
    Model(u,ym);
    for j := 1 to 4 do ym[j] := ym[j];
  end;
  if KeySimRig = 'R' then
  begin
    for j := 1 to 4 do uflot[j] := u[j] + uold[j];
    FlotRig(uflot,y);      {Flotation process sim. rig}
    Model(u,ym);
    for j := 1 to 4 do ym[j] := ym[j];
  end;

end; {end 'C' if statement}

(* Remove the controller from the loop *)

if Qcontrol = 'N' then
begin
  if KeySimRig = 'S' then
    Plant(u,y);      {Simulation model of the rig}
  if KeySimRig = 'R' then
  begin
    FlotRig(uold,y);      {Flotation process sim. rig}
  end;
end; {end 'N' if statement}

DrawP(time,r[52-ord(choice)]);
if (time > 200*2) and (time < 201*2) then
begin
  ex := Trunc(xscl*(time-xmin)+50);
  wy := Trunc(300-yscl*(r[52-ord(choice)] - ymin));
  Settextstyle(0,horizdir,1);
  Outtextxy(ex,wy+5,'r');

```

```

end;

DrawP(time,ym[52-ord(choice)]);
if (time > 331*2) and (time < 332*2) then
begin
  ex := Trunc(xscl*(time-xmin)+50);
  wy := Trunc(300-yscl*(ym[52-ord(choice)] - ymin));
  Settextstyle(0,horizdir,1);
  Outtextxy(ex,wy+5,'ym');
end;

DrawP(time,u[52-ord(choice)]);
if (time > 461*2) and (time < 462*2) then
begin
  ex := Trunc(xscl*(time-xmin)+50);
  wy := Trunc(300-yscl*(u[52-ord(choice)] - ymin));
  Settextstyle(0,horizdir,1);
  Outtextxy(ex,wy+5,'uflot');
end;

DrawP(time,y[52-ord(choice)]);
if (time >= 592*2) and (time <= 593*2) then
begin
  ex := Trunc(xscl*(time-xmin)+50);
  wy := Trunc(300-yscl*(y[52-ord(choice)] - ymin));
  Settextstyle(0,horizdir,1);
  Outtextxy(ex,wy+5,'L'+choice);
end;

DrawP(time,y[1]);
if (time >= 30*40) and (time <= 30*40+2) then
begin
  ex := Trunc(xscl*(time-xmin)+50);
  wy := Trunc(300-yscl*(y[1] - ymin));
  Settextstyle(0,horizdir,1);
  Outtextxy(ex,wy+5,'L1');
end;

DrawP(time,y[2]);
if (time >= 610) and (time <= 612) then
begin
  ex := Trunc(xscl*(time-xmin)+50);
  wy := Trunc(300-yscl*(y[2] - ymin));
  Settextstyle(0,horizdir,1);
  Outtextxy(ex,wy+5,'L2');
end;

DrawP(time,y[3]);
if (time >= 920) and (time <= 922) then
begin
  ex := Trunc(xscl*(time-xmin)+50);
  wy := Trunc(300-yscl*(y[3] - ymin));
  Settextstyle(0,horizdir,1);
  Outtextxy(ex,wy+5,'L3');

```

```

end;

DrawP(time,y[4]);
if (time >= 1530) and (time <= 1532) then
begin
  ex := Trunc(xsc1*(time-xmin)+50);
  wy := Trunc(300-ysc1*(y[4] - ymin));
  Settextstyle(0,horizdir,1);
  Outtextxy(ex,wy+5,'L4');
end;

col := col + 1;
if col <= 600 then
begin
  for row := 1 to 4 do
  begin
    setp [row,col] := r[row];
    u_ctrl[row,col] := u[row];
    ym_out[row,col] := ym[row];
    yp_out[row,col] := y[row];
  end;
end; {end if col < 600}
end; {end if "col < 600" then}

if Qcontrol = 'N' then
begin
  case key_pressed of
    'U': begin
      if choice in ['0'..'3'] then
        u[52-ord(choice)] := u[52-ord(choice)] + 10;
        Key_pressed := ' ';
      end;
    'D': begin
      if choice in ['0'..'3'] then
        u[52-ord(choice)] := u[52-ord(choice)] - 10;
        Key_pressed := ' ';
      end;
    end;
end; {if Qcontrol 'N'.....}

if Qcontrol = 'C' then
begin
  case key_pressed of
    'U': begin
      case choice of
        '0': step := 10; {204.8*2;}
        '1': step := 10; {204.8;}
        '2': step := 10; {204.8;}
        '3': step := 10; {204.8*2;}
      end;
      r[52-ord(choice)] := r[52-ord(choice)] + step;
      For j := 0 to 50 do Drawp(time,2*j);
      Key_pressed := ' ';
    end;
  end;
end;

```

```

'D': begin
  case choice of
    '0': step := -10;
    '1': step := -10;
    '2': step := -10;
    '3': step := -10;
  end;
  r[52-ord(choice)] := r[52-ord(choice)] + step;
  For j := 0 to 50 do Drawp(time,j*2);
  Key_pressed := ' ';
end;
'R': begin
  RampOn := not RampOn;
  For j := 0 to 50 do Drawp(time,j*2);
  Key_pressed := ' ';
end;

end; {end case keypressed}

if (RampOn = true) then
begin
  if time <= 550 then
    r[52-ord(choice)] := r[52-ord(choice)] + 20/500;
  if time > 550 then
    r[52-ord(choice)] := r[52-ord(choice)] - 20/500;
  end;
end; {if Qcontrol 'C'.....}

if keypressed then Key_pressed := upcase(readkey);

until (key_pressed in ['Q']) or (key_pressed = #27);

if (key_pressed = 'Q') or (key_pressed = #27) then closegraph;
end;
until exit = 'E';
end;

{*****}

{-----MAIN PROGRAM-----}
begin
  initpointers;
  dT := 1.0;
  checkfil := 0;
  repeat
    clrscr;

    Log := false;
    KeySimRig := ' ';
    Gotoxy(15,5);
    Write(' ');
    TextAttr := 112;

```

```

writeln('Press the Corresponding Key'); gotoxy(15,7);
TextAttr := 7;
writeln(' <R> * Do C/Loop tests on the FLOTATION RIG'); gotoxy(15,8);
writeln(' <S> * Do C/Loop simulations of the RIG'); gotoxy(15,9);
writeln(' <P> * To return to the program '); gotoxy(15,11);
writeln('      GRAPHS for each cell '); gotoxy(15,12);
writeln('      ONLY after a simulation is made '); gotoxy(15,14);
writeln(' <0> * r & y of RECLENER GRAPHS '); gotoxy(15,15);
writeln(' <A> * u & ym of RECLANER GRAPHS '); gotoxy(15,16);
writeln(' <1> * r & y of CLEANER GRAPHS '); gotoxy(15,17);
writeln(' <B> * u & ym of CLEANER GRAPHS '); gotoxy(15,18);
writeln(' <2> * r & y of SCAVENEGR GRAPHS '); gotoxy(15,19);
writeln(' <C> * u & ym of SCAVENGER GRAPHS '); gotoxy(15,20);
writeln(' <3> * r & y of ROUGHER GRAPHS '); gotoxy(15,21);
writeln(' <D> * u & ym of ROUGHER GRAPHS '); gotoxy(15,22);
writeln(' KEY = '); gotoxy(22,22);
repeat until keypressed;

gotoxy(22,22); clreol;

if keypressed then KeySimRig := upcase(ReadKey);
if KeySimRig = 'R' then
  initialize; {***** ADC & DAC cards *****}

if KeySimRig in ['R','S'] then
begin
  graphdim; {***** graph dimensions *****}
  ServKeySim;
end;

if KeySimRig in ['0'..'3','A'..'D'] then
begin
  head := 'FLOTATION PROCESS SIMULATION RIG (C/L)';
  xmax := 600;
  xmin := 0;
  xint := 60;
  xlabel := 'time [s]';
  case KeySimRig of
    '0':begin
      cell := 4;
      ymax := 70;
      ymin := -50;
      yint := 10;
      ylabel := 'Setp4 & Level4 in [%]';
      foot := 'RECLENER: press any key to continue';
      InstGraph('c:');
      ScrSetUp;
      CalcScale;
      Axes(Show);
      for point := 1 to 600 do
        begin
          Drawp(point,Setp[cell,point]);
          Drawp(point,yp_out[cell,point]);
        end;
    end;
  end;

```

```

        repeat until keypressed;
    end;
'A':begin
    cell := 4;
    ymax := 45;
    ymin := -30;
    yint := 5;
    ylabl := 'u4 & ym4 in [%]';
    foot := 'RECLEANER: press any key to continue';
    InstGraph('c:');
    ScrSetUp;
    CalcScale;
    Axes(Show);
    for point := 1 to 600 do
        begin
            Drawp(point,u_ctrl[cell,point]);
            Drawp(point,ym_out[cell,point]);
        end;
        repeat until keypressed;
    end;
'1':begin
    cell := 3;
    ymax := 50;
    ymin := -50;
    yint := 10;
    ylabl:= 'Setp3 & Level3 in [%]';
    foot := 'CLEANER: press any key to continue';
    InstGraph('c:');
    ScrSetUp;
    CalcScale;
    Axes(Show);
    for point := 1 to 600 do
        begin
            Drawp(point,Setp[cell,point]);
            Drawp(point,yp_out[cell,point]);
        end;
        repeat until keypressed;
    end;
'B':begin
    cell := 3;
    ymax := 30;
    ymin := -30;
    yint := 5;
    ylabl:= 'u3 & ym3 in [%]';
    foot := 'CLEANER: press any key to continue';
    InstGraph('c:');
    ScrSetUp;
    CalcScale;
    Axes(Show);
    for point := 1 to 600 do
        begin
            Drawp(point,u_ctrl[cell,point]);
            Drawp(point,ym_out[cell,point]);
        end;

```

```

        repeat until keypressed;
    end;
'2':begin
    cell := 2;
    ymax := 50;
    ymin := -50;
    yint := 10;
    ylabl:= 'Setp2 & Level2 in [%]';
    foot := 'SCAVENGER: press any key to continue';
    InstGraph('c:');
    ScrSetUp;
    CalcScale;
    Axes>Show);
    for point := 1 to 600 do
        begin
            Drawp(point,Setp[cell,point]);
            Drawp(point,yp_out[cell,point]);
        end;
        repeat until keypressed;
    end;
'C':begin
    cell := 2;
    ymax := 50;
    ymin := -40;
    yint := 5;
    ylabl:= 'u2 & ym2 in [%]';
    foot := 'SCAVENGER: press any key to continue';
    InstGraph('c:');
    ScrSetUp;
    CalcScale;
    Axes>Show);
    for point := 1 to 600 do
        begin
            Drawp(point,u_ctrl[cell,point]);
            Drawp(point,ym_out[cell,point]);
        end;
        repeat until keypressed;
    end;
'3':begin
    cell := 1;
    ymax := 50;
    ymin := -50;
    yint := 10;
    ylabl:= 'Setp1 & Levell in [%]';
    foot := 'ROUGHHER: press any key to continue';
    InstGraph('c:');
    ScrSetUp;
    CalcScale;
    Axes>Show);
    for point := 1 to 600 do
        begin
            Drawp(point,setp[cell,point]);
            Drawp(point,yp_out[cell,point]);
        end;

```

```
        repeat until keypressed;
    end;
'D':begin
    cell := 1;
    ymax := 30;
    ymin := -30;
    yint := 5;
    ylabl:= 'u1 & ym1 in [%]';
    foot := 'ROUGHER: press any key to continue';
    InstGraph('c:');
    ScrSetUp;
    CalcScale;
    Axes(Show);
    for point := 1 to 600 do
        begin
            Drawp(point,u_ctrl[cell,point]);
            Drawp(point,ym_out[cell,point]);
        end;
        repeat until keypressed;
    end;
end; {end case KeySimRig}
closegraph;
end;
until KeySimRig = 'P';
end.
→
```

Appendix G

Listing of Routines for Ramp Tests

The listing given in this appendix can be used for open loop ramp tests, C/L FLORTIG control and PC-simulations. It can also be used to check whether the $Q(s)$ parameter designed for ramp setpoint tracking does in fact *track ramp setpoints*.

An index is given below for all the routines used in the program.

Name of Procedure	Source Filename	Description of Procedure
clamp	rigrmp5.pas	clamps the number which is out of bounds to within the min. & max limits given
clock	rigrmp5.pas	determines the real-time using the dos clock
graphdim	rigrmp5.pas	defines the graph min & max dimensions and labels
initialize	rigrmp5.pas	initializes most of the variables used
initpointers	rigrmp5.pas	initializes all pointers
initstates	rigrmp5.pas	initializes all the pointer states for computations
initdelays	rigrmp5.pas	initializes all delays
dynamics	rigrmp5.pas	computes $R(s)$ which is the condition matrix added to ensure ramp tracking with matrix $Q_0 F(s)$ since $Q(s)$ designed for ramp tracking = $Q_0(s)R(s)$

qpar	rigrmp5.pas	computes the $Q_O F(s)$ which is not the same as the $Q(s)$ designed for step setpoints
plant	rigrmp5.pas	computes the nominal model $G(s)$
model	rigrmp5.pas	computes the model $M(s)$
flotrig	rigrmp5.pas	outputs $u(s)$ values to the FLOTRIG and returns $y(s)$ values
servkeysim	rigrmp5.pas	handles keyboard requests and does C/L control plots and calculations
main program	rigrmp5.pas	runs all routines and plots the respective variables as required on separate graphs

G.1 Programme listing(Ramp)

```
program rig(input,output);
```

```
uses dos,
      crt,
      GRAPH,
      IGRAPH,
      RKUTTA,
      DT2801_R,DT2815_R;
```

```
type pvarr = array[1..4] of real;
```

```
var
  key,KeySimRig           :char;
  DACchNo, ADCchNo, ADCstep,chno :integer;
  rep0,rep1,rep2,rep3      :integer;
  SetPoint,valueR,valueW   :integer;
  time, Told, Tnew, dT     :real;
  Output_R, Value         :real;
  hour, min, sec, sec100   :word;
  freq, count             :word;
  Recfil,Clefil,Scafil,Roufil :TEXT;
  FADcname                :STRING[32];
  LOG, RampOn, FileOpen    :BOOLEAN;
```

```

t                                :real;
checkfil                        : integer;
gs11,gs12,gs13                 : ptr;
gs21,gs22,gs23                 : ptr;
gs31,gs32,gs33,gs34            : ptr;
                                gs44 : ptr;

ms11,ms12,ms13                 : ptr;      {11=Rou, 22=Sca, 33=Cle, 44=Rec}
ms21,ms22,ms23                 : ptr;
ms31,ms32,ms33,ms34            : ptr;
                                ms44 : ptr;

gs11,gs12,gs13,gs14            : ptr;      {11=Rou, 22=Sca, 33=Cle, 44=Rec}
gs21,gs22,gs23,gs24            : ptr;
gs31,gs32,gs33,gs34            : ptr;
                                gs44 : ptr;

inv11 : ptr;
inv22 : ptr;
inv33 : ptr;
inv44 : ptr;

filt1 : ptr;
filt2 : ptr;
filt3 : ptr;
filt4 : ptr;

mdly12 : array[1..30] of real;
mdly13 : array[1..30] of real;
mdly31 : array[1..15] of real;
mdly32 : array[1..45] of real;
mdly34 : array[1..20] of real;
mdly44 : array[1..5] of real;

gdly12 : array[1..30] of real;
gdly13 : array[1..30] of real;
gdly31 : array[1..15] of real;
gdly32 : array[1..45] of real;
gdly34 : array[1..20] of real;
gdly44 : array[1..5] of real;

setp : array[1..4,1..600] of real;
u_ctrl : array[1..4,1..600] of real;
ym_out : array[1..4,1..600] of real;
yp_out : array[1..4,1..600] of real;

cell,point : integer;

{*****}
Procedure Clamp(var val:real; min, max:real);

{Clamps the computed values as specified by the user}

```

```

begin
  if val < min then val := min;
  if val > max then val := max;
end;

```

```

{*****}

```

```

Procedure clock(var hrs, min, sec, sec100:word; var time:real);

```

```

{ computes real time using the PC clock}

```

```

begin
  time := (3600*hrs + 60*min + sec + 0.01*sec100);
end;

```

```

{*****}

```

```

Procedure graphdim;

```

```

{Specifies main initial graph dimensions}

```

```

begin
  ymax := 100;
  ymin := -30;
  xmax := 600;
  xmin := 0;
  yint := 10;
  xint := 60;
  xlabl := 'time (s)';
  ylabl := 'Level (%)';
  head := 'FLOTATION PROCESS SIMULATION RIG (C/L)';
end;

```

```

{*****}

```

```

Procedure Initialize;

```

```

{ Initialize most used variables}

```

```

var inp0,inp1,inp2,inp3 :integer;

```

```

begin
  dac_init;
  adc_init;
  RampOn := FALSE;
  LOG := FALSE;
  inp0 := 2048; {2048=0%, 2560=25%, 3072=50%, 3584=75%, 4095=100%}
  dac_write(0,inp0);
  inp1 := 1024;
  dac_write(1,inp1);
  inp2 := 1024;
  dac_write(2,inp2);
  inp3 := 1024;
  dac_write(3,inp3);

```

```

Clearstates(qs31); Clearstates(qs32); Clearstates(qs33); Clearstates(qs34);
                                Clearstates(qs44);

```

```

Clearstates(inv11);
Clearstates(inv22);
Clearstates(inv33);
Clearstates(inv44);

```

```

Clearstates(filt1);
Clearstates(filt2);
Clearstates(filt3);
Clearstates(filt4);
end;

```

```

{*****}

```

```

Procedure initdelays;

```

```

{Initialise all arrays for that mimic delays
by passing values from one cell to the other
during every sampling interval}

```

```

var z : integer;
    c : integer;
begin
  for z := 1 to 30 do
    begin
      mdly12[z] := 0;
      gdly12[z] := 0;
      mdly13[z] := 0;
      gdly13[z] := 0;
    end;

    for z := 1 to 15 do
      begin
        mdly31[z] := 0;
        gdly31[z] := 0;
      end;

      for z := 1 to 45 do
        begin
          mdly32[z] := 0;
          gdly32[z] := 0;
        end;

        for z := 1 to 20 do
          begin
            mdly34[z] := 0;
            gdly34[z] := 0;
          end;

          for z := 1 to 5 do
            begin
              mdly44[z] := 0;

```

```

    key := ' ';
end; {initialize}

{*****}

Procedure initpointers;

{Initialise all pointers}

begin
    new(qs11);new(qs12);new(qs13);
    new(qs21);new(qs22);new(qs23);
    new(qs31);new(qs32);new(qs33);new(qs34);
                                new(qs44);
    new(ms11);new(ms12);new(ms13);
    new(ms21);new(ms22);new(ms23);
    new(ms31);new(ms32);new(ms33);new(ms34);
                                new(ms44);

    new(qs11);new(qs12);new(qs13);new(qs14);
    new(qs21);new(qs22);new(qs23);new(qs24);
    new(qs31);new(qs32);new(qs33);new(qs34);
                                new(qs44);

    new(inv11);
    new(inv22);
    new(inv33);
    new(inv44);

    new(filt1);
    new(filt2);
    new(filt3);
    new(filt4);

end;

{*****}

Procedure initstates;

{Initialise all pointers states transfer functions
used with the Runge-Kutta Algorithm}

begin
    Clearstates(qs11); Clearstates(qs12); Clearstates(qs13);
    Clearstates(qs21); Clearstates(qs22); Clearstates(qs23);
    Clearstates(qs31); Clearstates(qs32); Clearstates(qs33);Clearstates(qs34);
                                Clearstates(qs44);
    Clearstates(ms11); Clearstates(ms12); Clearstates(ms13);
    Clearstates(ms21); Clearstates(ms22); Clearstates(ms23);
    Clearstates(ms31); Clearstates(ms32); Clearstates(ms33);Clearstates(ms34);
                                Clearstates(ms44);

    Clearstates(qs11); Clearstates(qs12); Clearstates(qs13);Clearstates(qs14);
    Clearstates(qs21); Clearstates(qs22); Clearstates(qs23);Clearstates(qs24);

```

```

    gdly44[z] := 0;
end;

for z := 1 to 4 do
  for c := 1 to 600 do
    begin
      setp[z,c] := 0;
      u_ctrl[z,c] := 0;
      ym_out[z,c] := 0;
      yp_out[z,c] := 0;
    end;
  end;
end;

{*****}

```

Procedure Dynamics(cd:pvarr; var ud: pvarr);

{ This procedure is for the matrix, $R(s)$ which was designed as the extra matrix to be added to satisfy the ramp tracking conditions.

That can be seen from the equation given below as:

$$Q(s) = Q_0 F(s) R(s)$$

where $Q(s)$ is the Q -Parameter as defined in the main body of this thesis}

begin

```

with inv11^ do
  begin
    deadtime := 0;
    ordN := 1;
    ordD := 1;
    coeffN[0] := 1;
    coeffN[1] := (50 + 100 + 83.333);
    coeffD[0] := 1;
    coeffD[1] := 50;
  end;

```

```

with inv22^ do
  begin
    deadtime := 0;
    ordN := 2;
    ordD := 2;
    coeffN[0] := 1;
    coeffN[1] := (50 + 100 + 83.333);
    coeffN[2] := 10277.75;
    coeffD[0] := 1;
    coeffD[1] := 50;
    coeffD[2] := 3333.32;
  end;

```

```

with inv33^ do
  begin
    deadtime := 0;

```

```

ordN := 1;
ordD := 1;
coeffN[0] := 1;
coeffN[1] := (50 + 100 + 83.333);
coeffD[0] := 1;
coeffD[1] := 50;
end;

with inv44^ do
begin
  deadtime := 0;
  ordN := 1;
  ordD := 1;
  coeffN[0] := 1;
  coeffN[1] := (50 + 100 + 95.238);
  coeffD[0] := 1;
  coeffD[1] := 50;
end;

rk(inv11,cd[1],dT,ud[1]);
rk(inv22,cd[2],dT,ud[2]);
rk(inv33,cd[3],dT,ud[3]);
rk(inv44,cd[4],dT,ud[4]);

end; {end Procedure Dynamics}

{*****}

Procedure Qpar(aq:pvarr; var cq: pvarr);

{This procedure computes the first part,  $Q_0F(s)$ , of the
complete matrix which forms the Q-Parameter as specified
for ramp tracking conditions in the main body of the
thesis.

The definitions of the variables are similar to the ones
given for the Qpar procedure in Appendix F}.

var
  cqa,cqb,cqc,cqd : real;
  i                : integer;

begin

with qsl1^ do
begin
  deadtime := 0;
  ordN := 0;
  ordD := 1;
  coeffN[0] := 0.826;
  coeffN[1] := 37.2914;
  coeffD[0] := 1;
  coeffD[1] := 100;
end;

```

```
with qs12^ do
begin
  deadtime := 0;
  ordN := 0;
  ordD := 1;
  coeffN[0] := -2.7345;
  coeffN[1] := -66.8722;
  coeffD[0] := 1;
  coeffD[1] := 100;
end;
```

```
with qs13^ do
begin
  deadtime := 0;
  ordN := 0;
  ordD := 1;
  coeffN[0] := 1.9730;
  coeffN[1] := 14.5307;
  coeffD[0] := 1;
  coeffD[1] := 100;
end;
```

```
with qs14^ do
begin
  deadtime := 0;
  ordN := 0;
  ordD := 1;
  coeffN[0] := 0.8221;
  coeffN[1] := 17.2984;
  coeffD[0] := 1;
  coeffD[1] := 100;
end;
```

```
with qs21^ do
begin
  deadtime := 0;
  ordN := 0;
  ordD := 1;
  coeffN[0] := 0.0217;
  coeffN[1] := 36.7055;
  coeffD[0] := 1;
  coeffD[1] := 100;
end;
```

```
with qs22^ do
begin
  deadtime := 0;
  ordN := 0;
  ordD := 1;
  coeffN[0] := 0.1137;
  coeffN[1] := 59.2393;
  coeffD[0] := 1;
  coeffD[1] := 100;
```

```

    coeffD[1] := 100;
end;

with qs34^ do
begin
    deadtime := 0;
    ordN := 0;
    ordD := 1;
    coeffN[0] := 0.4451;
    coeffN[1] := 18.1584;
    coeffD[0] := 1;
    coeffD[1] := 100;
end;

with qs44^ do
begin
    deadtime := 0;
    ordN := 0;
    ordD := 1;
    coeffN[0] := 2.7778;
    coeffN[1] := 264.5503;
    coeffD[0] := 1;
    coeffD[1] := 100;
end;

rk(qs11,aq[1],dT,cqa);
rk(qs12,aq[2],dT,cqb);
rk(qs13,aq[3],dT,cqc);
rk(qs14,aq[4],dT,cqd);
cq[1] := (cqa + cqb + cqc + cqd);

rk(qs21,aq[1],dT,cqa);
rk(qs22,aq[2],dT,cqb);
rk(qs23,aq[3],dT,cqc);
rk(qs24,aq[4],dT,cqd);
cq[2] := (cqa + cqb + cqc + cqd);

rk(qs31,aq[1],dT,cqa);
rk(qs32,aq[2],dT,cqb);
rk(qs33,aq[3],dT,cqc);
rk(qs34,aq[4],dT,cqd);
cq[3] := (cqa + cqb + cqc + cqd);

rk(qs44,aq[4],dT,cqd);
cq[4] := (cqd);

end; { end of proc Qpar}

{*****}

Procedure Plant(up:pvarr; var yp:pvarr);

{This procedure is similar to the one given in Appendix F
since no alterations have been made}

```

```
var
  ypa,ypb,ypc,ypd : real;
  i                : integer;

begin
  with gs11^ do
    begin
      deadtime := 0;
      ordN := 1;
      ordD := 2;
      coeffN[0] := 0.005*0.009;
      coeffN[1] := 0.009;
      coeffD[0] := 0.0006;
      coeffD[1] := 0.05;
      coeffD[2] := 1;
    end;

  with gs12^ do
    begin
      deadtime := 30;
      ordN := 0;
      ordD := 1;
      coeffN[0] := 0.00625;
      coeffD[0] := 0.0125;
      coeffD[1] := 1;
    end;

  with gs13^ do
    begin
      deadtime := 30;
      ordN := 0;
      ordD := 1;
      coeffN[0] := -0.016;
      coeffD[0] := 0.02;
      coeffD[1] := 1;
    end;

  with gs21^ do
    begin
      deadtime := 0;
      ordN := 0;
      ordD := 1;
      coeffN[0] := -0.01625/2;
      coeffD[0] := 0.025;
      coeffD[1] := 1;
    end;

  with gs22^ do
    begin
      deadtime := 0;
      ordN := 0;
      ordD := 1;
```

```
coeffN[0] := 0.00744;
coeffD[0] := 0.012;
coeffD[1] := 1;
end;

with gs23^ do
begin
  deadtime := 0;
  ordN := 0;
  ordD := 1;
  coeffN[0] := -0.0011;
  coeffD[0] := 0.005;
  coeffD[1] := 1;
end;

with gs31^ do
begin
  deadtime := 15;
  ordN := 1;
  ordD := 2;
  coeffN[0] := 0.0009*0.016667/2;
  coeffN[1] := 0.016667/2;
  coeffD[0] := 0.0003;
  coeffD[1] := 0.035;
  coeffD[2] := 1;
end;

with gs32^ do
begin
  deadtime := 45;
  ordN := 0;
  ordD := 1;
  coeffN[0] := 0.0182;
  coeffD[0] := 0.028;
  coeffD[1] := 1;
end;

with gs33^ do
begin
  deadtime := 0;
  ordN := 1;
  ordD := 2;
  coeffN[0] := 0.0005*0.036;
  coeffN[1] := 0.036;
  coeffD[0] := 0.0006;
  coeffD[1] := 0.05;
  coeffD[2] := 1;
end;

with gs34^ do
begin
  deadtime := 20;
  ordN := 0;
  ordD := 1;
```

```

    coeffN[0] := -0.009/2;
    coeffD[0] := 0.03;
    coeffD[1] := 1;
end;

with gs44^ do
begin
    deadtime := 5;
    ordN := 0;
    ordD := 1;
    coeffN[0] := 0.00756/2;
    coeffD[0] := 0.0105;
    coeffD[1] := 1;
end;

qdly12[1] := up[2];
qdly13[1] := up[3];
qdly31[1] := up[1];
qdly32[1] := up[2];
qdly34[1] := up[4];
qdly44[1] := up[4];

for i := 30 downto 2 do qdly12[i] := qdly12[i-1];
for i := 30 downto 2 do qdly13[i] := qdly13[i-1];
for i := 15 downto 2 do qdly31[i] := qdly31[i-1];
for i := 45 downto 2 do qdly32[i] := qdly32[i-1];
for i := 20 downto 2 do qdly34[i] := qdly34[i-1];
for i := 5 downto 2 do qdly44[i] := qdly44[i-1];

rk(qs11,up[1],dT,ya);
rk(qs12,qdly12[30],dT,ypb);
rk(qs13,qdly13[30],dT,ypc);
yp[1] := (ypa + ypb + ypc);

rk(qs21,up[1],dT,ya);
rk(qs22,up[2],dT,ypb);
rk(qs23,up[3],dT,ypc);
yp[2] := (ypa + ypb + ypc);

rk(qs31,qdly31[15],dT,ya);
rk(qs32,qdly32[45],dT,ypb);
rk(qs33,up[3],dT,ypc);
rk(qs34,qdly34[20],dT,ypd);
yp[3] := (ypa + ypb + ypc + ypd);

rk(qs44,qdly44[5],dT,ypd);
yp[4] := (ypd);

end; { end of proc Plant}

{*****}

```

```
Procedure Model(um:pvarr; var ymm:pvarr);
```

```
{This procedure computes the mismatched model,  $M(s)$ .
```

```
  The numerators, 'ordN', and
    denominators, 'ordD',
  are specified for each TFH element. The coefficients for
  numerator and denominator, which are, 'coeffN' and 'coeffD' are also given for each element of the
  matrix of transfer
  functions}
```

```
var
  yma,ymb,ymc,ymd : real;
  i                : integer;
```

```
begin
```

```
  with ms11^ do
  begin
    deadtime := 0;
    ordN := 0;
    ordD := 2;
    coeffN[0] := 0.15*0.0006/2;
    coeffN[1] := 0.018/2;
    coeffD[0] := 0.0006;
    coeffD[1] := 0.05;
    coeffD[2] := 1;
  end;
```

```
  with ms12^ do
  begin
    deadtime := 30;
    ordN := 0;
    ordD := 1;
    coeffN[0] := 0.00625;
    coeffD[0] := 0.0125;
    coeffD[1] := 1;
  end;
```

```
  with ms13^ do
  begin
    deadtime := 30;
    ordN := 0;
    ordD := 1;
    coeffN[0] := -0.016;
    coeffD[0] := 0.02;
    coeffD[1] := 1;
  end;
```

```
  with ms21^ do
  begin
    deadtime := 0;
    ordN := 0;
    ordD := 1;
```

```
coeffN[0] := -0.01625/2;  
coeffD[0] := 0.025;  
coeffD[1] := 1;  
end;
```

```
with ms22^ do  
begin  
  deadtime := 0;  
  ordN := 0;  
  ordD := 1;  
  coeffN[0] := 0.00744;  
  coeffD[0] := 0.012;  
  coeffD[1] := 1;  
end;
```

```
with ms23^ do  
begin  
  deadtime := 0;  
  ordN := 0;  
  ordD := 1;  
  coeffN[0] := -0.0011;  
  coeffD[0] := 0.005;  
  coeffD[1] := 1;  
end;
```

```
with ms31^ do  
begin  
  deadtime := 15;  
  ordN := 0;  
  ordD := 2;  
  coeffN[0] := 0.05*0.0003/2;  
  coeffN[1] := 0.016667/2;  
  coeffD[0] := 0.0003;  
  coeffD[1] := 0.035;  
  coeffD[2] := 1;  
end;
```

```
with ms32^ do  
begin  
  deadtime := 45;  
  ordN := 0;  
  ordD := 1;  
  coeffN[0] := 0.0182;  
  coeffD[0] := 0.028;  
  coeffD[1] := 1;  
end;
```

```
with ms33^ do  
begin  
  deadtime := 0;  
  ordN := 0;  
  ordD := 2;  
  coeffN[0] := 0.0006*0.03;  
  coeffN[1] := 0.036;
```

```

    coeffD[0] := 0.0006;
    coeffD[1] := 0.05;
    coeffD[2] := 1;
end;

with ms34^ do
begin
    deadtime := 20;
    ordN := 0;
    ordD := 1;
    coeffN[0] := -0.009/2;
    coeffD[0] := 0.03;
    coeffD[1] := 1;
end;

with ms44^ do
begin
    deadtime := 5;
    ordN := 0;
    ordD := 1;
    coeffN[0] := 0.00756/2;
    coeffD[0] := 0.0105;
    coeffD[1] := 1;
end;

mdly12[1] := um[2];
mdly13[1] := um[3];
mdly31[1] := um[1];
mdly32[1] := um[2];
mdly34[1] := um[4];
mdly44[1] := um[4];

for i := 30 downto 2 do mdly12[i] := mdly12[i-1];
for i := 30 downto 2 do mdly13[i] := mdly13[i-1];
for i := 15 downto 2 do mdly31[i] := mdly31[i-1];
for i := 45 downto 2 do mdly32[i] := mdly32[i-1];
for i := 20 downto 2 do mdly34[i] := mdly34[i-1];
for i := 5 downto 2 do mdly44[i] := mdly44[i-1];

rk(ms11,um[1],dT,ya);
rk(ms12,mdly12[30],dT,yb);
rk(ms13,mdly13[30],dT,yc);
ymm[1] := (ya + yb + yc);

rk(ms21,um[1],dT,ya);
rk(ms22,um[2],dT,yb);
rk(ms23,um[3],dT,yc);
ymm[2] := (ya + yb + yc);

rk(ms31,mdly31[15],dT,ya);
rk(ms32,mdly32[45],dT,yb);
rk(ms33,um[3],dT,yc);
rk(ms34,mdly34[20],dT,yd);
ymm[3] := (ya + yb + yc + yd);

```

```

rk(ms44,mdly44[5],dT,ymd);
ymm[4] := (ymd);

end; { end of proc model}

{*****}

Procedure FlotRig(urig : pvarr; var yrig: pvarr);

{This procedure receives the FLOTTRIG inputs,'urig[i]', for
each column and writes the respective values to the
appropriate valves. It also reads the respective levels of
the FLOTTRIG cells into 'yrig[i]'.}

var chnoV,j      : integer;
    level        : real;
    position      : integer;
    offs          : array[1..4] of integer;
    factr         : array[1..4] of real;
    percoeff      : array[1..4] of real;

{0=Rec, 1=Cle, 2=Sca, 3=Rou for valves=V}
{4=Rec, 5=Cle, 6=Sca, 7=Rou for levels=L}
begin
    offs[4] := 2000; {1915;} {1=Rou, 2=Sca, 3=Cle, 4=Rec}
    offs[3] := 2185; {2290;}
    offs[2] := 2550;
    offs[1] := 2550;

    factr[4]:= 6.6; {7.45}
    factr[3]:= 7.94; {7.94}
    factr[2]:= 7;   {7}
    factr[1]:= 8;   {8}

    percoeff[4]:= 0;
    percoeff[3]:= 0;
    percoeff[2]:= 0;
    percoeff[1]:= 0;

    for chnoV := 3 downto 0 do
        begin
            position := trunc(2048 + 20.48*urig[4-chnoV]);
            dac_write(chnoV,position);
            adc_read(chnoV + 4, level);
            yrig[4-chnoV] := (level - offs[4-chnoV])/factr[4-chnoV];
            with filt1^ do
                begin
                    deadtime := 0;
                    ordN := 0;
                    ordD := 1;
                    coeffN[0] := 1;
                    coeffD[0] := 1;

```

```

        coeffD[1] := 1;
    end;

    with filt2^ do
    begin
        deadtime := 0;
        ordN := 0;
        ordD := 1;
        coeffN[0] := 1;
        coeffD[0] := 1;
        coeffD[1] := 1;
    end;
    with filt3^ do
    begin
        deadtime := 0;
        ordN := 0;
        ordD := 1;
        coeffN[0] := 1;
        coeffD[0] := 1;
        coeffD[1] := 1;
    end;

    with filt4^ do
    begin
        deadtime := 0;
        ordN := 0;
        ordD := 1;
        coeffN[0] := 1;
        coeffD[0] := 1;
        coeffD[1] := 1;
    end;
    rk(filt1,yrig[1],dT,yrig[1]);
    rk(filt2,yrig[2],dT,yrig[2]);
    rk(filt3,yrig[3],dT,yrig[3]);
    rk(filt4,yrig[4],dT,yrig[4]);
end;
end;

{*****}

```

Procedure ServKeySim;

{ This procedure services all request obtained from the keyboard for functions required by the user. In this case for selecting cells, closing the loop etc.}

```

var chno, j, ctr_in    : integer;
    choice, Key_pressed : char;
    exit               : char;
    level,step         : real;
    i, ex,wy           : integer;
    tex                : array[1..4] of string;
    a,b,c,y,ym,r,u,aold : pvarr;
    uold, ain, uflot    : pvarr;

```

```

factr           : array[1..4] of real;
percoff         : array[1..4] of real;
offs            : array[1..4] of integer;
Qcontrol        : char;
row,col         : integer;

```

```

begin
  r[1] := 0; a[1] := 0; uold[1] := -50; b[1] := 0; c[1] := 0; u[1] := 0; ym[1] := 0; y[1] := 0;
  uflot[1] := 0;
  r[2] := 0; a[2] := 0; uold[2] := -50; b[2] := 0; c[2] := 0; u[2] := 0; ym[2] := 0; y[2] := 0;
  uflot[2] := 0;
  r[3] := 0; a[3] := 0; uold[3] := -50; b[3] := 0; c[3] := 0; u[3] := 0; ym[3] := 0; y[3] := 0;
  uflot[3] := 0;
  r[4] := 0; a[4] := 0; uold[4] := 0; b[4] := 0; c[4] := 0; u[4] := 0; ym[4] := 0; y[4] := 0;
  uflot[4] := 0;

  tex[4] := 'L4';
  tex[3] := 'L3';
  tex[2] := 'L2';
  tex[1] := 'L1';
  initstates; {clear the states}
  initdelays; {initialise the arrays for delays}
  clrscr;

  repeat
    clrscr;
    choice := ' ';
    key_pressed := ' ';

    rep0 := 0; rep1 := 0; rep2 := 0; rep3 := 0;

    time := 0;
    gotoxy(15,5);
    write('_____');
    TextAttr := 112;
    if KeySimRig = 'S' then
      write(' MAIN MENU (Simulation) ');
    else
      write(' MAIN MENU (Flot. Rig ) ');
    TextAttr := 7;
    writeln('_____'); gotoxy(15,6);
    writeln(' Press <L> - for logging data '); gotoxy(15,7);
    writeln(' (toggles on / off ) '); gotoxy(15,8);
    writeln(' <0> - step test on RECLENER '); gotoxy(15,9);
    writeln(' <1> - step test on CLEANER '); gotoxy(15,10);
    writeln(' <2> - step test on SCAVENGER '); gotoxy(15,11);
    writeln(' <3> - step test on ROUGHER '); gotoxy(15,12);
    writeln(' <A> - step tests on all cells '); gotoxy(15,13);
    writeln(' <U/D> - to step(u/d) a chosen cell '); gotoxy(15,14);
    writeln(' <R> - for ramp tests on cells '); gotoxy(15,15);
    writeln(' <E> - To Quit (EXIT) '); gotoxy(15,16);
    writeln(' '); gotoxy(15,17);

```

```

writeln('      Enter choice : ');
repeat
  gotoxy(39,17);clreol;
  readln(choice);
  choice := upcase(choice);
  if not(choice in ['0'..'3','E']) then
    begin
      for i:= 1 to 8 do
        begin
          Sound(1000+100*i);
          delay(100);
          NoSound;
        end;
      end;
    until choice in ['0'..'3','E'];

  if choice = 'E' then exit := choice;

  if choice in ['0'..'3'] then
    begin
      case choice of
        '0' : foot := 'L = Log : <RECLEANER> : Q = Quit , C = C/Loop';
        '1' : foot := 'L = Log : <CLEANER> : Q = Quit , C = C/Loop';
        '2' : foot := 'L = Log : <SCAVENGER> : Q = Quit , C = C/Loop';
        '3' : foot := 'L = Log : <ROUGHES> : Q = Quit , C = C/Loop';
      end;

      settime(0,0,0,0); {initial value of time}
      time := 0;
      Told := 0;
      ctr_in := 0;
      col := 0;
      Qcontrol := 'N';
      RampOn := false;

      InstGraph('c:');
      ScrSetUp;
      CalcScale;
      Axes(Show);

      repeat
        if col <= 600 then
          begin

            gettime(Hour,min,sec,sec100);
            clock(Hour,min,sec,sec100,time);
            delay(950);

            if keypressed then key_pressed := upcase(readkey);

            case key_pressed of
              'C': Qcontrol := 'C';
              'N': begin

```

```

        Qcontrol := 'N';
        ctr_in := 0;
    end;

    'L': LOG := not LOG;
end; {case key_pressed}

(* Insert controller and close the loop *)

if Qcontrol = 'C' then
begin
    for j := 1 to 4 do
    begin
        if ctr_in = 0 then
        begin
            r[j] := y[j];
            Told := y[52-ord(choice)];
        end;
        b[j] := (y[j] - ym[j]);
        a[j] := r[j] - b[j];
    end; { for loop}

    if ctr_in = 0 then ctr_in := ctr_in + 2;

    Dynamics(a,c);
    Qpar(c,u);
    for j := 1 to 4 do u[j] := 2*u[j];
    for j := 1 to 4 do clamp(u[j],-100,100);
    if KeySimRig = 'S' then
    begin
        Plant(u,y);    {Simulation model of the rig}
        Model(u,ym);
        for j := 1 to 4 do ym[j] := ym[j]/2;
    end;
    if KeySimRig = 'R' then
    begin
        for j := 1 to 4 do uflot[j] := u[j] + uold[j];
        FlotRig(uflot,y);    {Flotation process sim. rig}
        Model(u,ym);
        for j := 1 to 4 do ym[j] := ym[j]/2;
    end;

end; {end 'C' if statement}

(* Remove the controller from the loop *)

if Qcontrol = 'N' then
begin
    if KeySimRig = 'S' then
        Plant(u,y);    {Simulation model of the rig}
    if KeySimRig = 'R' then
    begin
        FlotRig(uold,y);    {Flotation process sim. rig}

```

```

    end;
end; {end 'N' if statement}

DrawP(time,r[52-ord(choice)]);
if (time > 200*2) and (time < 201*2) then
begin
    ex := Trunc(xscl*(time-xmin)+50);
    wy := Trunc(300-yscl*(r[52-ord(choice)] - ymin));
    Settextstyle(0,horizdir,1);
    Outtextxy(ex,wy+5,'r');
end;

DrawP(time,ym[52-ord(choice)]);
if (time > 331*2) and (time < 332*2) then
begin
    ex := Trunc(xscl*(time-xmin)+50);
    wy := Trunc(300-yscl*(ym[52-ord(choice)] - ymin));
    Settextstyle(0,horizdir,1);
    Outtextxy(ex,wy+5,'ym');
end;

DrawP(time,u[52-ord(choice)]);
if (time > 461*2) and (time < 462*2) then
begin
    ex := Trunc(xscl*(time-xmin)+50);
    wy := Trunc(300-yscl*(u[52-ord(choice)] - ymin));
    Settextstyle(0,horizdir,1);
    Outtextxy(ex,wy+5,'uflot');
end;

DrawP(time,y[52-ord(choice)]);
if (time >= 592*2) and (time <= 593*2) then
begin
    ex := Trunc(xscl*(time-xmin)+50);
    wy := Trunc(300-yscl*(y[52-ord(choice)] - ymin));
    Settextstyle(0,horizdir,1);
    Outtextxy(ex,wy+5,'L'+choice);
end;

DrawP(time,y[1]);
if (time >= 30*40) and (time <= 30*40+2) then
begin
    ex := Trunc(xscl*(time-xmin)+50);
    wy := Trunc(300-yscl*(y[1] - ymin));
    Settextstyle(0,horizdir,1);
    Outtextxy(ex,wy+5,'L1');
end;

DrawP(time,y[2]);
if (time >= 610) and (time <= 612) then
begin
    ex := Trunc(xscl*(time-xmin)+50);
    wy := Trunc(300-yscl*(y[2] - ymin));
    Settextstyle(0,horizdir,1);

```

```

    Outtextxy(ex,wy+5,'L2');
end;

DrawP(time,y[3]);
if (time >= 920) and (time <= 922) then
begin
    ex := Trunc(xscl*(time-xmin)+50);
    wy := Trunc(300-yscl*(y[3] - ymin));
    Settextstyle(0,horizdir,1);
    Outtextxy(ex,wy+5,'L3');
end;

DrawP(time,y[4]);
if (time >= 1530) and (time <= 1532) then
begin
    ex := Trunc(xscl*(time-xmin)+50);
    wy := Trunc(300-yscl*(y[4] - ymin));
    Settextstyle(0,horizdir,1);
    Outtextxy(ex,wy+5,'L4');
end;

col := col + 1;
if col <= 600 then
begin
    for row := 1 to 4 do
    begin
        setp [row,col] := r[row];
        u_ctrl[row,col] := u[row];
        ym_out[row,col] := ym[row];
        yp_out[row,col] := y[row];
    end;
end; {end if col < 600}
end; {end if "col < 600" then}

if Qcontrol = 'N' then
begin
    case key_pressed of
        'U': begin
            if choice in ['0'..'3'] then
                u[52-ord(choice)] := u[52-ord(choice)] + 10;
                Key_pressed := ' ';
            end;
        'D': begin
            if choice in ['0'..'3'] then
                u[52-ord(choice)] := u[52-ord(choice)] - 10;
                Key_pressed := ' ';
            end;
    end;
end; {if Qcontrol 'N'.....}

if Qcontrol = 'C' then
begin
    case key_pressed of
        'U': begin

```

```

        case choice of
          '0': step := 10; {204.8*2;}
          '1': step := 10; {204.8;}
          '2': step := 10; {204.8;}
          '3': step := 10; {204.8*2;}
        end;
        r[52-ord(choice)] := r[52-ord(choice)] + step;
        For j := 0 to 50 do Drawp(time,2*j);
        Key_pressed := ' ';
      end;
    'D': begin
      case choice of
        '0': step := -10;
        '1': step := -10;
        '2': step := -10;
        '3': step := -10;
      end;
      r[52-ord(choice)] := r[52-ord(choice)] + step;
      For j := 0 to 50 do Drawp(time,j*2);
      Key_pressed := ' ';
    end;
    'R': begin
      RampOn := not RampOn;
      For j := 0 to 50 do Drawp(time,j*2);
      Key_pressed := ' ';
    end;

  end; {end case keypressed}

  if (RampOn = true) then
    begin
      if time <= 450 then
        r[52-ord(choice)] := r[52-ord(choice)] + 20/500;
      if time > 450 then
        r[52-ord(choice)] := r[52-ord(choice)] - 20/500;
      end;
    end; {if Qcontrol 'C'.....}

    if keypressed then Key_pressed := upcase(readkey);

    until (key_pressed in ['Q']) or (key_pressed = #27);

    if (key_pressed = 'Q') or (key_pressed = #27) then closegraph;
  end;
  until exit = 'E';
end;

{*****}

{-----MAIN PROGRAM-----}
begin
  initpointers;

```

```

dT := 1.0;
checkfil := 0;
repeat
  clrscr;

  Log := false;
  KeySimRig := ' ';
  Gotoxy(15,5);
  Write(' ');
  TextAttr := 112;
  writeln('Press the Corresponding Key'); gotoxy(15,7);
  TextAttr := 7;
  Writeln(' <R> * Do C/Loop tests on the FLOTATION RIG');gotoxy(15,8);
  Writeln(' <S> * Do C/Loop simulations of the RIG');gotoxy(15,9);
  Writeln(' <P> * To return to the program ');gotoxy(15,11);
  Writeln('          GRAPHS for each cell ');gotoxy(15,12);
  Writeln('          ONLY after a simulation is made ');gotoxy(15,14);
  Writeln(' <O> * r & y of RECLEANER GRAPHS ');gotoxy(15,15);
  Writeln(' <A> * u & ym of RECLANER GRAPHS ');gotoxy(15,16);
  Writeln(' <I> * r & y of CLEANER GRAPHS ');gotoxy(15,17);
  Writeln(' <B> * u & ym of CLEANER GRAPHS ');gotoxy(15,18);
  Writeln(' <2> * r & y of SCAVENEGR GRAPHS ');gotoxy(15,19);
  Writeln(' <C> * u & ym of SCAVENGER GRAPHS ');gotoxy(15,20);
  Writeln(' <3> * r & y of ROUGHER GRAPHS ');gotoxy(15,21);
  Writeln(' <D> * u & ym of ROUGHER GRAPHS ');gotoxy(15,22);
  WRITELN(' KEY = ');gotoxy(22,22);
  repeat until keypressed;

  gotoxy(22,22);clreol;

  if keypressed then KeySimRig := upcase(ReadKey);
  if KeySimRig = 'R' then
    initialize; {***** ADC & DAC cards *****}

  if KeySimRig in ['R','S'] then
    begin
      graphdim; {***** graph dimensions *****}
      ServKeySim;
    end;

  if KeySimRig in ['0'..'3','A'..'D'] then
    begin
      head := 'FLOTATION PROCESS SIMULATION RIG (C/L)';
      xmax := 600;
      xmin := 0;
      xint := 60;
      xlabl := 'time [s]';
      case KeySimRig of
        '0':begin
          cell := 4;
          ymax := 70;
          ymin := -50;
          yint := 10;
          ylabl:= 'Setp4 & Level4 in [%]';

```

```

    foot := 'RECLENER: press any key to continue';
    InstGraph('c:');
    ScrSetUp;
    CalcScale;
    Axes(Show);
    for point := 1 to 600 do
        begin
            Drawp(point,Setp[cell,point]);
            Drawp(point,yp_out[cell,point]);
        end;
    repeat until keypressed;
end;
'A':begin
    cell := 4;
    ymax := 45;
    ymin := -30;
    yint := 5;
    ylabl := 'u4 & ym4 in [%]';
    foot := 'RECLENER: press any key to continue';
    InstGraph('c:');
    ScrSetUp;
    CalcScale;
    Axes(Show);
    for point := 1 to 600 do
        begin
            Drawp(point,u_ctrl[cell,point]);
            Drawp(point,ym_out[cell,point]);
        end;
    repeat until keypressed;
end;
'1':begin
    cell := 3;
    ymax := 50;
    ymin := -50;
    yint := 10;
    ylabl:= 'Setp3 & Level3 in [%]';
    foot := 'CLENER: press any key to continue';
    InstGraph('c:');
    ScrSetUp;
    CalcScale;
    Axes(Show);
    for point := 1 to 600 do
        begin
            Drawp(point,Setp[cell,point]);
            Drawp(point,yp_out[cell,point]);
        end;
    repeat until keypressed;
end;
'B':begin
    cell := 3;
    ymax := 30;
    ymin := -30;
    yint := 5;
    ylabl:= 'u3 & ym3 in [%]';

```

```

    foot := 'CLEANER: press any key to continue';
    InstGraph('c:');
    ScrSetUp;
    CalcScale;
    Axes(Show);
    for point := 1 to 600 do
        begin
            Drawp(point,u_ctrl[cell,point]);
            Drawp(point,ym_out[cell,point]);
        end;
    repeat until keypressed;
end;
'2':begin
    cell := 2;
    ymax := 50;
    ymin := -50;
    yint := 10;
    ylabl:= 'Setp2 & Level2 in [%]';
    foot := 'SCAVENGER: press any key to continue';
    InstGraph('c:');
    ScrSetUp;
    CalcScale;
    Axes(Show);
    for point := 1 to 600 do
        begin
            Drawp(point,Setp[cell,point]);
            Drawp(point,yp_out[cell,point]);
        end;
    repeat until keypressed;
end;
'C':begin
    cell := 2;
    ymax := 30;
    ymin := -40;
    yint := 5;
    ylabl:= 'u2 & ym2 in [%]';
    foot := 'SCAVENGER: press any key to continue';
    InstGraph('c:');
    ScrSetUp;
    CalcScale;
    Axes(Show);
    for point := 1 to 600 do
        begin
            Drawp(point,u_ctrl[cell,point]);
            Drawp(point,ym_out[cell,point]);
        end;
    repeat until keypressed;
end;
'3':begin
    cell := 1;
    ymax := 50;
    ymin := -50;
    yint := 10;
    ylabl:= 'Setp1 & Level1 in [%]';

```

```

    foot := 'ROUGHER: press any key to continue';
    InstGraph('c:');
    ScrSetUp;
    CalcScale;
    Axes>Show;
    for point := 1 to 600 do
        begin
            Drawp(point, setp[cell, point]);
            Drawp(point, yp_out[cell, point]);
        end;
    repeat until keypressed;
end;
'D':begin
    cell := 1;
    ymax := 30;
    ymin := -30;
    yint := 5;
    ylabl:= 'u1 & ym1 in [%]';
    foot := 'ROUGHER: press any key to continue';
    InstGraph('c:');
    ScrSetUp;
    CalcScale;
    Axes>Show;
    for point := 1 to 600 do
        begin
            Drawp(point, u_ctrl[cell, point]);
            Drawp(point, ym_out[cell, point]);
        end;
    repeat until keypressed;
end;
end; {end case KeySimRig}
closegraph;
end;
until KeySimRig = 'P';
end.

```