National Library of Canada
Canadian Theses Service
Ottawa, Canada
K1A 0N4

Bibliothèque nationale du Canada
Service des thèses canadiennes

NOTICE

The quality of this microform is heavily dependent upon the quality of the original thesis submitted for microfilming. Every effort has been made to ensure the highest quality of reproduction possible.

If pages are missing, contact the university which granted the degree.

Some pages may have indistinct print especially if the original pages were typed with a poor typewriter ribbon or if the university sent us an inferior photocopy.

Reproduction in full or in part of this microform is governed by the Canadian Copyright Act, R.S.C. 1970, c. C-30, and subsequent amondments.

AVIS

La qualité de cette microforme dépend grandement de la qualité de la thèse soumise au microfilmage. Nous avons tout fait pour assurer une qualité supérieure de reproduction.

S'il manque des pages, veuillez communiquer avec l'université qui a conféré le grade.

La qualité d'impression de certaines pages peut laisser à désirer, surtout si les pages originales ont été dactylographiées à l'aide d'un ruban usé ou si l'université nous a fait parvenir une photocopie de qualité inférieure.

La reproduction, même partielle, de cette microforme est soumise à la Loi canadienne sur le droit d'auteur, SRC 1970, c. C-30, et ses amendements subséquents.

Canada

NL-339 (r. 88/04) c

UNIVERSITY OF ALBERTA

Predictive Control Of Processes With Time Delays

by

Eric K.C. Lau ©

A THESIS

SUBMITTED TO THE FACULTY OF GRADUATE STUDIES AND RESEARCH

IN PARTIAL FULFILLMENT OF THE REQUIREMENTS FOR THE DEGREE

OF MASTER OF SCIENCE

IN

PROCESS CONTROL

THE DEPARTMENT OF CHEMICAL ENGINEERING

EDMONTON, ALBERTA

FALL, 1990

National Library
of Canada

Bibliothèque nationale
du Canada

Canadian Theses Service

Service des thèses canadiennes

Ottawa, Canada
K1A 0N4

The author has granted an irrevocable non-exclusive licence allowing the National Library of Canada to reproduce, loan, distribute or sell copies of his/her thesis by any means and in any form or format, making this thesis available to interested persons.

L'auteur a accordé une licence irrévocable et non exclusive permettant à la Bibliothèque nationale du Canada de reproduire, prêter, distribuer ou vendre des copies de sa thèse de quelque manière et sous quelque forme que ce soit pour mettre des exemplaires de cette thèse à la disposition des personnes intéressées.

The author retains ownership of the copyright in his/her thesis. Neither the thesis nor substantial extracts from it may be printed or otherwise reproduced without his/her permission.

L'auteur conserve la propriété du droit d'auteur qui protège sa thèse. Ni la thèse ni des extraits substantiels de celle-ci ne doivent être imprimés ou autrement reproduits sans son autorisation.

ISBN 0-315-65111-3

UNIVERSITY OF ALBERTA

RELEASE FORM

Name of author	Eric K. C. Lau
Title of thesis	Predictive Control Of Processes With Time Delays
Degree	Master of Science
Year this degree granted	FALL, 1990

Permission is hereby granted to the University of Alberta library to reproduce single copies of this thesis and to lead or sell such copies for private, scholarly of scientific research purposes only.

The author reserves other publication rights, and neither the thesis nor extensive extracts from it may be reprinted or otherwise reproduce without the author's written permission.

(SIGNED) __________

PERMANENT ADDRESS :

1419, Wah Chun House,
Wah Fu Estate,
Hong Kong

DATED August 1 1990

UNIVERSITY OF ALBERTA

FACULTY OF GRADUATE STUDIES AND RESEARCH

The undersigned certify that they have read, and recommend to the Faculty of Graduate Studies and Research for acceptance, a thesis entitled **Predictive Control Of Processes With Time Delays** submitted by **Eric K. C. Lau** in partial fulfillment of the requirements for the degree of **Master of Science in Process Control.**

Dr. S.L. Shah
Supervisor

Dr. D.G. Fisher

Dr. K. Bollinger

DATE August 1/1990

To my parents,

brothers and sisters

ABSTRACT

This thesis involves the implementation and evaluation of three predictive control schemes: (1) the PID control with Smith predictor, (2) the Robust Adaptive Control (RAC), and (3) the Multi-step Adaptive Predictive Control (MAPC). These three model-based predictive control schemes are capable of handling processes with time delay but suffer from the effect of model process mismatch (MPM). Therefore, attention was paid especially to the effect of time delay mismatch on these control schemes and ways of getting around the problem. It was found both from simulations and experimental evaluations (done on a Continuous Stirred Tank Heater) that filtering plays an important role in rendering the control schemes robust to time delay mismatch.

When the MPM is not too far off, the Smith predictor gives excellent control prediction and allows the closed loop to use a higher gain. With the introduction of an exponential residual filter, the Smith predictor can achieve similar performance even when the time delay is severely under or over estimated. For servo control, it is best to have a strong residual filter when the time delay of the process is not known or time-varying. For regulatory control, on the other hand, the filtering action must be reduced so as to make the controller more alert to disturbance rejection. The filter constant can therefore be used as an on-line tuning parameter for closed loop performance.

The RAC is a direct adaptive controller similar to the minimum variance controller. It features an augmented process that ensures stable control of non-minimum phase (NMP) processes and a parameter estimation scheme that uses regressor normalization and a dead zone to ensure robustness. The controller demonstrated its ability to give offset free servo and regulatory control to processes with variable time delay. Output weighting (or filtering) is found to be essential to its success. However, the choice of the output weighting requires some knowledge of the process, which renders the controller somewhat unfriendly to use. The MAPC, which is based on a multi-stage controller design, showed itself to be superior to the RAC in dealing with time delay mismatch both on performance and ease of use. The predictor used for the MAPC (called the Modified Kalman Filter Predictor) is structurally similar to the Smith predictor with a residual filter, except that the filter is chosen 'optimally' according to the knowledge of the ratio of the process and measurement noise (R_2/R_1). Results show that it is best to have a high ratio to commence the controller for robustness and progressively tune it down to increase servo and regulatory performances. Since the controller's 'default settings' can result in satisfactory results in most applications, the performance of MAPC can be tuned using R_2/R_1 alone.

This thesis also introduces the structure of the QNX operating system and Multicon. Together, they form a unified multitasking shell for developing Advanced Control Tasks (ACT) which relieves the control engineer from many computer software burdens. The three above mentioned control schemes are developed as ACTs. User manuals are written (in Appendix 2) so that these ACTs can be used for educational and experimental purposes. Although Multicon has successfully provided services to a number of graduate students for their applications, the present version has much to be improved on. Therefore, suggestions for enhancing its power and user friendliness are presented.

Acknowledgement

I would like to thank Dr. S.L. Shah for his guidance and supervision during the course of this research. My gratitude is also extended to Mr. Walter Bodez, Mr. Don Sutherland, Mr. Henry Sit, Mr. Bob Barton and other staffs of the electronics and machine shops and the DACS center for their prompt assistance in keeping the computer and experimental apparatus functional.

The work on this thesis was made more enjoyable by other graduate students. In particular, I owe much to discussions with Dr. Weiping Lu, Aaron Yiu, Kun-yu Kwok , Mike Foley and David Shook. Appreciation is also extended to Dr. Coorus Mohtadi of Oxford University for his valuable suggestions during his visit in the Department of Chemical Engineering. The financial support from the Department of Chemical Engineering is gratefully acknowledged.

Many thanks are for my parents, brothers and sisters in Hong Kong for their constant support and encouragement over so many years. Furthermore, great appreciations are credited to Miss Mei Yee Li, who has given me constant spiritual support through many days.

Lastly, I must take this opportunity to express my deepest praise to my Lord God, Jesus Christ, who has upheld me and strengthened me through good times and bad. May His name be glorified.

Table of Contents

Chapter Page

1 **Introduction** 1
  1.1 Motivation 1
  1.2 Organization of the thesis 3

2 **PID control with Smith Predictor** 4
  2.1 Introduction 4
  2.2 Smith Predictor 4
  2.3 Smith Predictor with a Residual Filter 6
  2.4 Proportional-Integral-Derivative Control 7
  2.5 Tuning of the PID controller 8
  2.6 Time delay analysis using frequency domain interpretation 10
  2.7 Experimental Evaluation of a Predictive control scheme 12
  2.8 Conclusions 13

3 **Robust Adaptive Control (RAC).** 20
  3.1 Introduction 20
  3.2 Derivation of RAC 21
  3.3 Parameter estimation in RAC 24
  3.4 Control Weightings used in RAC 25

  3.5 Reconstruction of RAC

    3.5.1 Introduction 27
    3.5.2 Choice of process model 27
    3.5.3 RAC with CARIMA model. 28
    3.5.4 Removing Offset 32
    3.5.5 Parameter estimation for the modified RAC . . 33
    3.5.6 Some Interpretations on P and Q weightings of RAC 34

3.6 Test run of RAC by simulation

3.6.1 Description of process 38
3.6.2 Discussion of results 39

3.7 Test run of RAC by experiment

3.7.1 Introduction 41
3.7.2 Discussion on the choice of dead-zone . . . 42
3.7.3 Results and discussions. 43
3.8 Conclusions 45

4 **Multistep Adaptive Predictive Control (MAPC).** 68
4.1 Introduction 68

4.2 The Modified Kalman Filter Predictor (MKFP)

4.2.1 Basic formulation of the Kalman Filter Predictor 69
4.2.2 Modified Kalman Filter Predictor: disturbance modelling 73
4.2.3 The role of $Gr(z^{-1})$ 78
4.2.4 Simulation on MKFP in the case of time delay mismatch 83
4.2.5 Summary to the simulation results 86

4.3 Adaptive Modified Kalman Filter Predictor (AMKFP)

4.3.1 Problem formulation and requirements 86
4.3.2 Recursive Least Square algorithm 88
4.3.3 Problems relating the Recursive Least Square algorithm 89

4.4 Multistep Controller Design

4.4.1 Basic development of the multistep controller . 90
4.4.2 Feedforward control 94
4.4.3 Some tuning guidelines for MAPC in the presence of time delay mismatch 96

4.5 Experimental results

4.5.1 Process and Program description 97
4.5.2 Results: with no time delay mismatch 97
4.5.3 Results: with time delay mismatch 100

4.6 Conclusions 101

5 QNX plus MULTICON. 124
5.1 Introduction 124

5.2 QNX Operating System

5.2.1 Introduction 124
5.2.2 Structure of the QNX OS 125
5.2.3 Inter-task communication in QNX 128
5.2.4 Task scheduling in QNX 128
5.2.5 Computer system 129

5.3 MULTICON

5.3.1 Introduction 130
5.3.2 Structure of MULTICON 131
5.4 Use of MULTICON 135

5.5 CSTH and the Opto-22 Interface

5.5.1 The CSTH 136
5.5.2 The Opto-22 Interface 138

6 Conclusions and Recommendations 145
6.1 Conclusions 145
6.2 Recommendations 147

7 References 150

Appendices

1 Innovations analysis for MKFP 156
2A User reference to PID, Smith predictor, and Feedforward control 158
2B User reference to RAC 164
2C User reference to MAPC 174

List of Figures

Figure Page

2.1 PID Control on the Continuous Stirred Tank Heater . . 14

2.2 Block Diagram of a Smith Predictor 14

2.3 Block Diagram of a Predictive Control Scheme . . . 15

2.4 Bode Plot for the Open Loop Transfer Function of the feedback System without Smith Predictor 15

2.5 An Alternate Block Diagram of a PID + Smith Predictor Control Scheme 16

2.6 Bode Plot of the Open Loop Transfer Function of the Feedback System with the Smith Predictor . . . 17

2.7 Bode Plot of $G_m^*(z^{-1})$ 18

2.8 PI Control with Smith Predictor and an exponential residual filter under time delay mismatch . . . 19

3.1 Block diagram of an adaptive controller 47

3.2 Performance of a d-step ahead self-tuning control on a varying time delay process 47

3.3 Root locus of the numerator of the augmented plant . 48

3.4 Output trajectories of RAC with dead-zone ON . . . 49

3.5 Output trajectories of RAC with dead-zone OFF . . 50

3.6 Block diagram of the augmented plant structure used in RAC 51

3.7 Simulation of RAC on 3rd order process with process order mismatch. (P=1, Q=0, R=1, T=1, Δ_b=0.0243) . . . 52

3.8 Simulation of RAC on 3rd order process with process order mismatch. (P=1, Q=0, R=1, T=1, Δ_b=0.01) 53

3.9 Simulation of RAC on 3rd order process with time delay mismatch. (P=1, Q=0, R=1, T=1, Δ_b=0.0243) . . . 54

3.10 Root locus of augmented process zeros with increasing value of λ 55

3.11 Simulation of RAC on 3rd order process with time delay mismatch. ($P=1$, $Q=5\Delta$, $R=1$, $T=1$, $\Delta b=0.0243$) . . . 56

3.12 Simulation of RAC on 3rd order process with time delay mismatch. ($P=1$, $Q=5\Delta/(1-0.7z^{-1})$, $R=1$, $T=1$, $\Delta b=0.0243$) 57

3.13 Simulation of RAC on a 3rd order non-minimum phase process with time delay and process order mismatch. ($P=1$, $Q=5\Delta$, $R=1$, $T=1$, $\Delta b=0.0243$) 58

3.14 Simulation of RAC on a 3rd order process with time delay mismatch. ($P=1$, $Q=3\Delta/(1-0.7z^{-1})$, $R=1$, $T=1$, $\Delta b=0.0243$) 59

3.15 Simulation of RAC on a 3rd order process with time delay mismatch. ($P=5(1-0.8z^{-1})$, $Q=3\Delta/(1-0.7z^{-1})$, $R=1$, $T=1$, $\Delta b=0.0243$) 60

3.16 RAC servo control on CSTH with no time delay mismatch. ($P=1$, $Q=0$, $R=1$, $T=1$, $\Delta b=0.05$, TC#2, $\hat{d}=5$) 61

3.17 RAC servo control on CSTH with time delay mismatch. ($P=1$, $Q=0.5\Delta$, $R=1$, $T=1$, $\Delta b=0.05$, TC#3, $\hat{d}=5$) 62

3.18 RAC servo control on CSTH with time delay mismatch. ($P=1$, $Q=0.95\Delta/(1-0.7z^{-1})$, $R=1$, $T=1$, $\Delta b=0.05$, TC#3, $\hat{d}=5$) . . 63

3.19 RAC servo control on CSTH with time delay mismatch. ($P=1$, $Q=0.95\Delta/(1.5-0.297z^{-1})$, $R=1$, $T=1$, $\Delta b=0.05$, TC#3, $\hat{d}=5$) 64

3.20 RAC servo control on CSTH with time delay mismatch. ($P=2(1-0.5z^{-1})$, $Q=0.95\Delta/(1.5-0.297z^{-1})$, $R=1$, $T=1$, $\Delta b=0.05$, TC#3, $\hat{d}=5$) 65

3.21 RAC servo control on CSTH with time delay mismatch. (P and Q are tuned on-line, $R=1$, $T=1$, TC#3, $\hat{d}=5$) . . 66

3.22 RAC servo and regulatory control on CSTH with time delay mismatch. ($P=3(1-0.66z^{-1})$, $Q=3\Delta/(1-0.7z^{-1})$, $R=1$, $T=1$, $\Delta b=0.05$, TC#3, $\hat{d}=5$) 67

4.1a Magnitude plot of the frequency response of $Gf(z^{-1})$ using a first order T-filter 103

4.1b Magnitude plot of the frequency response of $Gf(z^{-1})$

using a second order T-filter 103

4.2 Open loop response of the simulation process . . . 104

4.3a Open loop prediction from the MKFP with no time delay mismatch (R_2=0.3, R_1=0.125) 105

4.3b Open loop prediction from the MKFP with no time delay mismatch (R_2=0.3, R_1=0.0025) 105

4.3c Open loop prediction from the MKFP with no time delay mismatch (R_2=15.0, R_1=0.0025) 106

4.4a Open loop prediction from the MKFP with time delay mismatch (R_2=0.3, R_1=0.125) 106

4.4b Open loop prediction from the MKFP with time delay mismatch (R_2=0.3, R_1=0.0025) 107

4.4c Open loop prediction from the MKFP with time delay mismatch (R_2=15.0, R_1=0.0025) 107

4.5a Open loop prediction from the MKFP with model time delay varies from 1 to 12 (R2=0.3, R1=0.125) 108

4.5b Open loop prediction from the MKFP with model time delay varies from 1 to 12 (R2=0.3, R1=0.0025) 108

4.5c Open loop prediction from the MKFP with model time delay varies from 1 to 12 (R2=15.0, R1=0.0025) 109

4.6 Variance of the difference of the predicted output and the process undelayed output with respect to the choice of the model time delay 110

4.7 Root locus plot for the characteristic equation of Gr(z^{-1}) for no time delay mismatch case with respect to the choice of R_2/R_1 111

4.8 Root locus plot for the characteristic equation of Gr(z^{-1}) for time delay mismatch case with respect to the choice of R_2/R_1 111

4.9 Magnitude plot of the frequency response of Gr(z^{-1}) with no time delay mismatch using different values of R_2/R_1 112

4.10 Magnitude plot of the frequency response of Gr(z^{-1}) with time delay mismatch using different values of

R2/R1 112

4.11 Structure of the Adaptive Modified Kalman Filter Predictor 113

4.12 A network of parallel, optimal predictors obtained from the MKFP 113

4.13a Closed-loop response of MAPC on CSTH without time delay mismatch (TC#1) using different values of R2/R1 . . 114

4.13b Parameter trajectories for MAPC on CSTH. 115

4.14 Kalman gain trajectories of MAPC on CSTH without time delay mismatch (TC#1) using different values of R2/R1 116

4.15 Magnitude plot of the frequency response of $Gf(z^{-1})$ without time delay mismatch (TC#1) using different values of R2/R1 117

4.16 Comparison of $Gf(z^{-1})$ from the MKFP and from a T-filter 118

4.17 Regulatory control of MAPC on CSTH without time delay mismatch (TC#1) using different values of R2/R1 . . 119

4.18 Servo and regulatory control of MAPC on CSTH with varying time delay (TC#2, TC#3) using different values of R2/R1 120

4.19 Kalman gain trajectories of MAPC on CSTH with varying time delay (TC#2, TC#3) using different values of R2/R1 121

4.20 Magnitude plot of the frequency response of $Gf(z^{-1})$ with varying time delay (TC#1) using different values of R2/R1 122

4.21 Comparison of $Gf(z^{-1})$ obtained for time delay mismatch case and that without time delay using different values of R2/R1 123

5.1 Block diagram of a Real-Time Software Package for Process Control 139

5.2 General block diagram of an Operating System 139

5.3 The Client-Server model of QNX 140
5.4 The Scheduling Algorithm of QNX 140
5.5 Schematic diagram of the structure of MULTICON . . 141
5.6 Pseudo-code of the multicon core program . . . 142
5.7 Schematic diagram of the CSTH 141
5.8 Open loop response curve of the CSTH 144

List of Tables

Table Page

3.1 RAC Controller Settings for Simulation Runs . . . 39

3.2 RAC Controller Settings for Experimental Runs . . 43

5.1 Open Loop Process Model of the Continuous Stirred Tank Heater 137

Nomenclature and Notation

Alphabetical

$A(z^{-1})$	Polynomial corresponding to the process output.
$\hat{A}(z^{-1})$	Estimate of $A(z^{-1})$.
a_i	Coefficient of $A(z^{-1})$ polynomial.
$B(z^{-1})$	Polynomial corresponding to the process input.
$\hat{B}(z^{-1})$	Estimate of $B(z^{-1})$.
b_i	Coefficient of $B(z^{-1})$ polynomial.
$C(z^{-1})$	Polynomial corresponding to the process noise.
$\hat{C}(z^{-1})$	Estimate of $C(z^{-1})$.
d	Process time delay in sampling period.
$\hat{d}$	Process model time delay in sampling period.
$\bar{d}$	Upper bound for the time delay in sampling period.
E	Expected value of.
$E(z^{-1})$	Quotient polynomial from the Diophantine equation.
$e(t\|t)$	*a posteriori* estimation error.
$e^n(t\|t)$	Normalized $e(t\|t)$.
$F(z^{-1})$	Remainder polynomial from the Diophantine equation.
$G(z^{-1})$	Transfer function representing the process.
$G_c(z^{-1})$	Transfer function representing the controller.
$G_d(z^{-1})$	Transfer function for the load disturbance.
$G_f(z^{-1})$	Transfer function representing the residual filter.
$G_{ff}(z^{-1})$	Transfer function for the feedforward controller.
$G_m(z^{-1})$	Transfer function representing the process model.
$G_m^*(z^{-1})$	Transfer function representing the feedback error block in the Smith predictor design due to time delay mismatch.
$G_p(z^{-1})$	Transfer function representing the delay free process model.
K_c	Proportional constant for the PID controller.
K_d	Derivative constant for the PID controller.
K_i	Integral constant for the PID controller.
k_f	Filter constant used in the exponential filter.
$L(t)$	Kalman gain.

l_i	Coefficients of L(t).
N1	First point of the prediction horizon.
N2	Last point of the prediction horizon.
Nu	Control horizon.
n(t)	Normalization factor.
n_1	Order of $A(z^{-1})$ polynomial.
$\overline{n_1}$	Upper bound for n_1.
n_2	Order of $B(z^{-1})$ polynomial.
$\overline{n_2}$	Upper bound for n_2.
$P(z^{-1})$	Process output weighting transfer function.
P(t)	Covariance matrix.
$Q(z^{-1})$	Controller output weighting transfer function.
$R(z^{-1})$	Setpoint filter transfer function.
R_1	Variance of process noise.
R_2	Variance of measurement noise.
$T(z^{-1})$	T-filter.
T_s	Sampling period.
U(t)	Process input.
X(t)	State vector.
$\hat{X}(t)$	State estimates.
x(t)	Normalized regressor vector.
$x_r(t)$	Reduced order x(t).
Y(t)	Process output.
$\hat{Y}(t+\hat{d}+1\|t)$	$\hat{d}$ step ahead prediction of the process output.
$Y_{sp}(t)$	Setpoint of the process output.
$Y^n(t)$	Normalized process output.
Z(t)	Augmented process output.

Greek

β	Filter constant.
ε	Filter constant (used in Chapter 2).
$\varepsilon(t)$	Difference between the setpoint and the process output.
Δ	Integrator ($\Delta=1-z^{-1}$, different from $\Delta(t)$).
$\Delta(t)$	Effect of unmeasured disturbance and noise.

Δ_b	Dead-zone.
$\Delta^m(t)$	Effect of $\Delta(t)$ and unmodelled dynamics.
$\Delta^n(t)$	Normalized $\Delta^m(t)$.
δ	Time delay error in seconds (used in Chapter 2).
δ	Stands for degree of (used in Chapter 3 and 4).
$\phi(t)$	Regressor vector.
Φ	State Matrix.
$\Phi(t)$	Time varying state matrix.
$\phi_r(t)$	Reduced order regressor vector.
$\phi_u(t)$	Unmodelled regressor vector.
$\underline{\underline{\Gamma}}_Y$	Weighting matrix for the prediction trajectory.
$\underline{\underline{\Gamma}}_U$	Weighting matrix for the control output trajectory.
γ_{Yj}	Weighting for the prediction trajectory.
γ_{Uj}	Weighting for the control output trajectory.
Λ	Input matrix.
$\Lambda(t)$	Time varying input matrix.
λ	Controller output weighting.
$\lambda(t)$	Forgetting factor.
H	Output matrix.
$\eta_1(t)$	Process noise.
$\eta_2(t)$	Measurement noise.
θ	Fractional delay error, θ/τ_d (used in Chapter 2).
$\theta(t)$	Process model parameter vector.
$\theta_r(t)$	Reduced order process model parameter vector.
$\theta_u(t)$	Unmodelled process model parameter vector.
τ_d	Time delay in seconds.
$\xi(t)$	White noise.
ξ_i	Coefficients of the $C(z^{-1})$ polynomial.
ω_{co}	Cross over frequency in rad/s.
$\omega(t)$	Innovations (or residual) sequence.
Ξ	Measurement noise matrix.
$\zeta(t)$	Variable weighting factor and on/off switch.

Subscripts

$\underline{Y}$	Y vector.
$\underline{\underline{Y}}$	Y matrix.

Superscripts

$Y^f(t)$	Filtered value of $Y(t)$.
θ^t	Transpose of θ.
$\hat{\theta}$	Estimate of θ.

Abbreviations

AMKFP	Adaptive Modified Kalman Filter Predictor.
APCS	Adaptive Predictive Control System.
ARE	Algebraic Riccati Equation.
ARMA	Auto Regressive Moving Average.
CARIMA	Controller AutoRegressive Integrated Moving Average.
CSTH	Continuous Stirred Tank Heater.
DMC	Dymamic Matrix Control.
EHAC	Extended Horizon Adaptive Control.
EPSAC	Extended Prediction Self-Adaptive Control..
GMV	Generalized Minimum Variance Control.
GPC	Generalized Predictive Control.
LRPC	Long Range Predictive Control.
LSP	Least Square Predictor.
MAPC	Multistep Adaptive Predictive Control.
MKFP	Modified Kalman Filter Predictor.
MPM	Model Plant Mismatch.
MRAC	Model Reference Adaptive Control.
NMP	Non Minimum Phase.
PM	Phase Margin.
RAC	Robust Adaptive Control.
RAPC	Robust Adaptive Predictive Control.
RLS	Recursive Least Square.
SISO	Single Input Single Output system.

SP	Smith Predictor.
PID	Proportional-Integral-Derivative Control.

1 INTRODUCTION

1.1 Motivations

Adaptive control of processes with variable time delays is of considerable practical interest in the process industry. The Smith predictor pioneered by Smith (1957) plays an important role in time delay compensation and control. Like the Smith Predictor, many adaptive predictive control schemes also require an exact knowledge of the process time delay (Åström and Wittenmark, 1973; Ljung and Söderström, 1983). Since this requirement greatly restricts the performance and applicability of adaptive controllers, special attention has been paid to the on-line estimation of the time delay and controller design that is more robust to the wrong choice of the time delay.

On-line estimation of the time delay appears to be one of the solutions to the time delay problem. This method has widely been used in the area of signal processing (Hassab et al, 1979; Friedlander et al., 1984). However, it still needs to be tailored for adaptive control applications. The idea of using a rational approximation for the time delay has been examined by many authors (Robinson et al, 1970; Rao, 1983; Gawthrop et al., 1985). This philosophy remains controversial because of the uncertainty of the time delay on the performance of the system. de Souza et al. (1988) follow this method and use it in the design of an adaptive controller. The authors report promising results in that the order of the approximation can be as low as second order and the approximation still allows a very good fit to the system frequency response over the bandwidth relevant to control system design. Another approach is to use least squares to estimate the parameters of the system for each value of time delay, and then optimize the delay using a spline curve fitting method (Pearson et al., 1984). This statistical method is promising but computationally expensive. It appears that estimation of the time delay for adaptive control remains an open field of research. More reliable and efficient methods must be developed before it can be applied in actual applications.

Many adaptive controller designs have been configured to alleviate the problem of the dependence of the control algorithm on a good estimate of the time delay. For example, pole-placement algorithms (Wellstead et al., 1979; Åström and Wittenmark, 1980) and the generalized predictive control algorithms (Clarke et al., 1987) estimate a model that admits a range of possible delay values by over parameterizing the numerator dynamics. Cluett and Shah, (1989) proposed a direct adaptive control scheme which makes use of an augmented process representation to reduce the excessive control action due to time delay underestimation and unmodelled dynamics. Dumont and Zervos (1988) proposed an alternative process representation using the Laguerre series which is thought to be less sensitive to the time delay estimation. These methods have been demonstrated to be very successful in various applications provided that some a *priori* knowledge of the process time delay is available (e.g. the upper bound of the delay).

The implementation and evaluation of three different control schemes to deal with variable time-delay problem is the primary objective of this thesis. The schemes are based on the following categories: (1) Non-adaptive predictive control: PID control with Smith predictor; (2) a single step adaptive predictive controller: the Robust Adaptive Control (RAC, Cluett and Shah , 1989); and (3) multi-step adaptive predictive control: Multistep Adaptive Predictive Control (MAPC, Sripada, 1988). The features pertaining to their robustness against varying time delay are examined and experimentally tested on the pilot scale Continuous Stirred Tank Heater.

To evaluate these control schemes, a lot of computer software has been developed, including the real-time control programs for each scheme as well as utility programs. Since the effort spent on computer programming is enormous, the question of how to organize these programs in a reusable fashion quickly becomes a big concern. This becomes the secondary subject of the thesis: to organize the programs so that they can be used for continuing studies in an educational environment. This lead to the use of a PC based multiuser, multitasking operating system and the development of an application oriented shell so that organization and use of the user-written control schemes can be simplified.

1.2 Organization of the thesis

This thesis is organized as follows: Chapters 2, 3, and 4 are dedicated towards the discussions of three control schemes. They are the non-adaptive predictive control scheme in Chapter 2; the adaptive single-step predictive control scheme in Chapter 3; and the multi-step adaptive predictive control scheme in Chapter 4. The development and evaluation of the control properties in the presence of varying time delay are the focuses of these chapters. Simulations and experiments are used for illustration and evaluation whenever necessary. Figures are graphed at the end of each chapter to facilitate comparison.

Chapter 5 introduces the QNX operating system and MULTICON software package under which the control schemes were developed and run. Evaluation on the MULTICON is done by examining its internal structure. Its shortcomings and some suggestions on possible improvements are included in the recommendation section of Chapter 6.

Chapter 6 concludes the thesis with the inclusion of some recommendations possible for future work followed by a listing of the references. All the user manuals for the control schemes in the previous chapters are included in the Appendix section.

2 PID control and the Smith Predictor

2.1 Introduction

Proportional-Integral-Derivative controllers have dominated the process control industry for almost half a century. Even with the introduction of digital controllers which make the realization of many advance control techniques possible, the 3-term controller remains a standard mode of control in industry because it is both well understood, flexible, and works for over 90% of process applications.

Time delays are very common in the process industry due to reasons such as transportation lag and instrumentation analysis and response time. In the presence of time delay, the closed loop is more sensitive to low frequency periodic disturbances and therefore is susceptible to stability problems. Figure [illegible] shows the performance of PID control on the Continuous Stirred Tank Heater (see Chapter 5 for a description of the apparatus). The tightly tuned controller gives very good control performance up to time ≈ 130 sample periods, after which the time delay of the process suddenly increases (from ≈ 4 sampling periods to ≈ 7 sampling periods). At that point, the control becomes unstable. One common strategy to deal with this problem is to detune the PID controller which makes the response sluggish. Another strategy is to use time delay compensation to improve the control performance. O.J.M. Smith proposed a time delay compensation scheme called the Smith Predictor (1957) which effectively removes the time delay element from the characteristic equation. This improves the stability margin of the closed-loop system and therefore allowing higher gains. The Smith Predictor coupled with the PID controller forms the basis of classical predictive control. It has proven to be very effective in numerous industrial applications.

This chapter reviews the mathematical formulation of the Smith Predictor, the basic PID control algorithm, and its variations.

2.2 Smith Predictor

The block diagram of the Smith Predictor is presented in Figure 2.2. Let

$G(z^{-1}) = \frac{B(z^{-1})}{A(z^{-1})} z^{-d}$ represent the actual process including delay,

$G_m(z^{-1}) = \frac{\hat{B}(z^{-1})}{\hat{A}(z^{-1})} z^{-\hat{d}}$ represent the model including delay,

$G_p(z^{-1}) = \frac{\hat{B}(z^{-1})}{\hat{A}(z^{-1})}$ represent the delay free model , and

$G_f(z^{-1}) = 1$ representing the filter transfer function.

The prediction is given by

$$\hat{Y}(t+d\,|\,t) = \frac{\hat{B}(z^{-1})}{\hat{A}(z^{-1})}U(t) + G_f(z^{-1})\left(Y(t) - \frac{\hat{B}(z^{-1})}{\hat{A}(z^{-1})} z^{-\hat{d}}\right) \tag{2.2.1}$$

When there is no model-plant-mismatch (MPM), the residual term (i.e. $Y(t) - \frac{\hat{B}(z^{-1})}{\hat{A}(z^{-1})} z^{-\hat{d}}$) in the absence of disturbances becomes zero and the prediction is simply the undelayed output of the process. The predicted output $\hat{Y}(t+d\,|\,t)$ corresponding to the estimate of the undelayed output is used as the feedback signal instead of the true process output. The block diagram of the closed loop system with the Smith Predictor in place is presented in Figure 2.3. $G_c(z^{-1})$ denotes the controller transfer function. The closed loop transfer function described as in Figure 2.3 is given by:

$$Y(t) = \frac{G(z^{-1})G_c(z^{-1})}{1+G_p(z^{-1})G_c(z^{-1})+G_c(z^{-1})(G(z^{-1})-G_m(z^{-1}))} Y_{sp}(t) \tag{2.2.2}$$

When $G(z^{-1})=G_m(z^{-1})$, i.e. no MPM, Equation (2.2.2) becomes

$$Y(t) = \left(\frac{B(z^{-1})}{A(z^{-1})}G_c(z^{-1})z^{-d}\right) \Big/ \left(1+\frac{\hat{B}(z^{-1})}{\hat{A}(z^{-1})}G_c(z^{-1})\right) Y_{sp}(t) \tag{2.2.3}$$

where the time delay term has been completely removed from the characteristic equation.

2.3 Smith Predictor with a Residual Filter

Even though it has been over 40 years since the Smith predictor was first proposed, and even though simulation and experimental results demonstrated successful applications, the Smith predictor is still not being widely used. Detuning the controller for robustness is preferred instead. One of the main reasons is that a good model for representing the process is difficult to obtain. As long as there is no MPM or disturbance, the Smith Predictor gives excellent prediction. However, the performance of the predictor deteriorates in the presence of MPM and/or disturbances. These problems can be alleviated by introducing a residual filter, $Gr(z^{-1})$, in the feedback path (see Figure 2.3, also see section 2.6). An exponential filter is often found to give 'good' results (Walgama, 1986):

$$Gr(z^{-1}) = \frac{1-kr}{1-kr z^{-1}} \qquad 0 \le kr < 1 \tag{2.3.1}$$

The amount of filtering can be adjusted by kr: the higher the value of kr, the stronger the filtering will be. In the presences of large residuals due to MPM, it is desired to have strong filtering action so that the predictor will not be overly sensitive to MPM. On the other hand, kr cannot be too large because heavy filtering of $Gr(z^{-1})$ leads to slow disturbance rejection. This is because there is no disturbance modelling used and disturbance rejection depends solely on feedback (one modification called the 'Smith Predictor with load prediction' sets $Gr(z^{-1})$ as a first order lead filter so that it acts as an extrapolator). As a result, the selection of kr is usually a compromise between performance, e.g. speed, and robustness. For most chemical processes, where the setpoint is not frequently changed, kr should be small (say 0.75 instead of 0.95) so that the disturbance information can be passed quickly for feedback control.

The steady-state value of $Gr(z^{-1})$ is unity. This is important for the Smith predictor to result in unbiased prediction.

Many advanced predictive control schemes can be represented by the same block diagram as shown in Figure 2.3, only with a different value of

$G_f(z^{-1})$. The RAC and MAPC, which have a similar structure, are the subjects of detailed discussion and evaluation in the later chapters.

2.4 Proportional-Integral-Derivative Control

The positional form of the basic PID control scheme is given by:

$$U(t) = K_c e(k) + K_I T_s \sum e(i) + K_d/T_s (e(t)-e(t-1)) \tag{2.4.1}$$

The velocity form is given by differencing Equation (2.4.1). It is usually preferred over the positional form because it provides bumpless transfer and avoids reset windup. The velocity form is written as

$$U(t) = U(t-1) + (K_c+T_sK_I+K_d/T_s)e(t) - (1+2K_d/T_s)e(k-1) + K_d/T_s e(k-2) \tag{2.4.2}$$

Equation (2.4.2) is also known as the 'setpoint on PID control' structure since the proportional, integral, and derivative actions are taken on the error between the setpoint and the output. Åström and Wittenmark have (1984) mentioned and discussed several modifications to the basic PID control algorithm to improve its performance. Two of the common variations are outlined below.

Setpoint on PI only control:

When a setpoint change is requested in a setpoint on PID controller, a large kick in the output often results. This occurs because the derivative of the error at the time of the setpoint change is very large. This action is known as a derivative kick. A PID control algorithm that avoids this 'kick' can be realized by making the derivative action act only on the actual process output instead of the feedback error. The resulting algorithm is as follows (Vermeer, 1987):

$$U(t) = U(t-1) + K_c(e(t) - e(t-1)) + K_I T_s e(t) + K_d/T_s(-Y(t) + 2Y(t-1) - Y(t-2)) \tag{2.4.3}$$

Setpoint on I only control:

This structure is designed to remove the setpoint from the proportional and the derivative terms:

$$U(t) = U(t-1) + K_c(-Y(t) + Y(t-1)) + K_I T_s e(t) + K_d/T_s(-Y(t) + 2Y(t-1) - Y(t-2)) \qquad (2.4.4)$$

These three PID control algorithms have identical characteristic equations and therefore the stability margins are identical. However, the controllers' zeros are different, which contribute to the difference in control performance. In general, the setpoint on PID control algorithm is the most aggressive and the setpoint on I only control algorithm is the least aggressive among the three. A detailed analysis of these algorithms has been done by Vermeer (1987).

2.5 Tuning of the PID controller

Although there are only three tuning parameters in a PID controller, they allow one to adjust the low, middle, and high frequency behaviours of the process. There are numerous schemes suggested for getting 'good' estimates for the three parameters. Some commonly used off-line tuning methods are listed as follows (Seborg et al., 1989):

Off-line Methods based on frequency response analysis:

One commonly known method under this category is the Ziegler-Nichols method (1942). The process is set into closed loop proportional control and the tuning constants are estimated from the proportional gain required for a sustained oscillation. Another similar method is suggested by Åström and Hägglund (1983) where the proportional control is replaced by a relay. If a limit cycle exists under relay control, this method locates a single point on the Nyquist curve based on the describing function approximation. The PID tuning constants can then be designed for specific phase and gain margin. The amplitude of the oscillation can be controlled by controlling the relay amplitude, which is not possible for the Ziegler-Nichols method.

Model-based off-line tuning methods:

The Cohen and Coon method is perhaps the most well known tuning method under this category (1953). The authors realized that most of the process responses in chemical industry have a sigmoidal shape and can be approximated using a first order plus time delay model. The controller parameters are then determined according to this model to give one-quarter decay ratio and minimum ISE performance. Another model-based tuning method is based on Internal Model Control (IMC, Garcia and Morari, 1982). Rivera et al. (1986) borrowed the IMC framework for PID controller design. They took into account closed-loop performance and robustness in a very systematic fashion and provided a complete guideline for PI(D) design. Robustness to MPM (and also time delay error and disturbances) is achieved by an imbedded filter in the controller. It was found that many commonly used industrial models (in particular a first order plus time delay model) lead to PI(D) controller design, with the tuning constants closely tied with the model parameters and the filter constant. The method uses the Padé approximation for systems with time delay.

Another method that uses both frequency response method and process model method is due to Seborg and Yuwana (1982). From an underdamped process response, the ultimate gain and period are obtained. The PID constants can then be designed according to the desired phase and gain margins.

On-line Methods:

Many auto-tuning methods for the PID controller have been proposed and commenced for the PID controller. The relay method by Hägglund and Åström (1985) has been extended to an on-line method. A commercial controller built on this method called the Satt Control Autotuner has been evaluated by Goberdhansingh and Cluett (1987). The Foxboro EXACT is another class of auto-tuning PID controller developed recently based on the idea of pattern recognition of the process (Bristol, 1987). It has been evaluated by Minter and Fisher (1987). Model based self tuners such as the Self Tuning Regulator (Clarke and Gawthrop, 1975) can also achieve a 3-term PID control structure when the regulator parameters are adjusted accordingly. The Turnbull TCS

6355 is a commercial product based on the model based self tuner scheme. The Toshiba Tosdic 211D8 is another commercial product based on this scheme. It is a two degrees of freedom auto-tuning PID controller which claims to have better process control quality for both servo and regulatory performances (Shigemasa et. al., 1987).

2.6 Time delay analysis using frequency domain interpretation

Frequency domain analysis is a useful tool in examining stability of systems. In this section, this technique is applied to analyze: (1) the effect of time delay in closed loop control; (2) the effect of time delay mismatch on the Smith predictor; and (3) the importance of the residual filter in the presence of time delay mismatch.

Consider the following process in the s-domain:

$$G(s) = \frac{2.16}{48s+1} e^{-(30+\delta)s} \qquad (2.6.1)$$

where δ represents the time delay fluctuations in seconds.

A PID controller is applied to the process for feedback control. The controller is tuned using the Cohen-Coon (C-C) method, with δ assumed to be 0 ($K_c = 1.103$, $K_I = 0.019\ s^{-1}$, $K_d = 10.810$ s). For ease of comparison, these tuning constants will not be altered throughout the analysis.

Figure 2.4 shows the Bode plots of the open loop transfer function of the feedback system. When $\delta=0$, the phase margin (PM) $\approx 35^{\circ}$. Since

$$PM = \delta_{max}\,\omega_{co}, \qquad (2.6.2)$$

the maximum tolerable time delay error $\approx$13.5 s. This means that with the C-C settings, the loop allows a maximum increase of process time delay of the order of 13.5 seconds before it becomes unstable (which is a 45% increase in time delay). This effect has already been demonstrated clearly in Figure 2.1.

Assuming there is no MPM except for time delay mismatch, Figure 2.3 can be refigured to Figure 2.5, where

$G_c(z^{-1})$ = PID controller,

$G_p(z^{-1}) = 2.16/(48s+1)$, and

$$G_m^{*}(z^{-1}) = 1+G_f(z^{-1})e^{-\tau_d s}(e^{-\tau_d \theta s}-1) \qquad (2.6.3)$$

where $G_f(z^{-1}) = 1$ for ordinary Smith predictor, and

$\theta = \delta/\tau_d$ = fractional time delay error.

The Smith Predictor effectively removes the time delay term from the closed loop and thereby alleviates time delay problems. When there is no time delay mismatch (i.e. $\theta=0$), $G_m^{*}(z^{-1}) = 1$. Since $G_p(z^{-1})$ is first order, $G_c(z^{-1})$ can be of infinitely large gain without rendering the closed loop system unstable. When $\theta \neq 0$, however, the performance of the Smith predictor declines because the time delay is 'reintroduced' in the closed loop. Figure 2.6 is the Bode plot using the Smith predictor with different values of θ. Several points are worth noting:

(1) For perfect prediction, the system is always stable,

(2) Time delay mismatch will not alter the loop gain,

(3) There exists a value of θ where the loop becomes unstable.

Intuitively, if $G_m^{*}(z^{-1})$ is kept with gain equal to unity and the phase shift never exceeds -180° at all frequencies, then the closed loop will always be stable. It can be seen from Equation (2.6.3) that $G_f(z^{-1})$ plays a very important role in achieving this result. As mentioned in section 2.3, a first order low pass filter is usually used:

$$G_f(z^{-1}) = 1/(\varepsilon s+1) \qquad (2.6.4)$$

Figure 2.7 demonstrates that for the same amount of θ ($\theta=1$), the phase shift and the gain of $G_m^{*}(z^{-1})$ are better maintained at the desired values of 0° and 1 respectively as ε increases. This means that the closed loop is now 'insensitive' to time delay mismatch and of course the loop gain can be tuned up to achieve tighter and faster performance.

In summary, when there is no MPM except for time delay mismatch, the performance of the Smith predictor improves in terms of performance and robustness with the introduction of a residual filter. Whenever the delay mismatch is large, or the time delay varies greatly over time, stronger filtering is recommended. A word of caution on using the Padé approximation for the time delay: A third order approximation was used to derived the above plots. First and second order approximations were often poor.

2.7 Experimental Evaluation of a Predictive Control Scheme

A Smith Predictor cascaded to a PID controller was applied experimentally to the Continuous Stirred Tank Heater (CSTH) for testing (Refer to Chapter 5 for a detailed description on the apparatus). The purpose of the experiment is to demonstrate that control performance improves with the use of predictive control. Also the effects of time delay mismatch are examined.

Results and discussions

If the model time delay is incorrect, the performance suffers. This is demonstrated in Figure 2.8. The settings of the experiment is as follows:

$$Gm(z^{-1}) = \frac{\hat{B}(z^{-1})}{\hat{A}(z^{-1})} z^{-\hat{d}}$$

where $\hat{B}(z^{-1}) = 0.082z^{-1}$

$\hat{A}(z^{-1}) = 1-0.892z^{-1}$

$\hat{d} = 1$

The PI gains ($Kp=3.0$, $Ki=0.019$) are tuned for thermo-couple #1 (i.e. $d \simeq 1$).

Without $Gr(z^{-1})$: (from period 0 to 450)

With the time delay progressively increased by changing from thermo-couple #1 to #2 to #3, the performance of the servo control worsen. For thermo-couple #3 in particular, the response is only marginally stable.

With $Gr(z^{-1})$: (from period 450 onward)

Rather than detuning the controller, a residual filter $Gr(z^{-1})$ in the form of an exponential filter is used. kr is used as the tuning knob for the time delay mismatch case. Notice the mark improvement for the thermo-couple #3 case with $kr=0$ and $kr=0.95$. It can be concluded from the figure that when the knowledge of the process time delay (or even process model) is uncertain, a larger value of kr (heavier filtering) should be used. A more in-depth discussion on the role of $Gr(z^{-1})$ is presented in section 4.2.3.

2.8 Conclusions

This chapter introduces predictive control by reviewing the Smith Predictor and the PID controller. Through experimental runs, it has been demonstrated that time delay compensation can result in improved control performance if the time delay can be exactly modelled. However, when the time delay of the process is not known or when it is time varying, predictive control suffers from the same problem. In this case, a residual filter plays an important role in giving a more acceptable predictive control performance.

The following algorithms are implemented as an Advanced Control Task that runs under MULTICON. They are useful for experimental evaluation of the control algorithms. User references are included in Appendix 2A.

(1) PID control with variations (section 2.4),

(2) Smith predictor with an exponential residual filter.

A fixed gain feedforward control is also implemented to accommodate for a secondary measurable process input. A user reference manual for the controller is included in Appendix 2A.

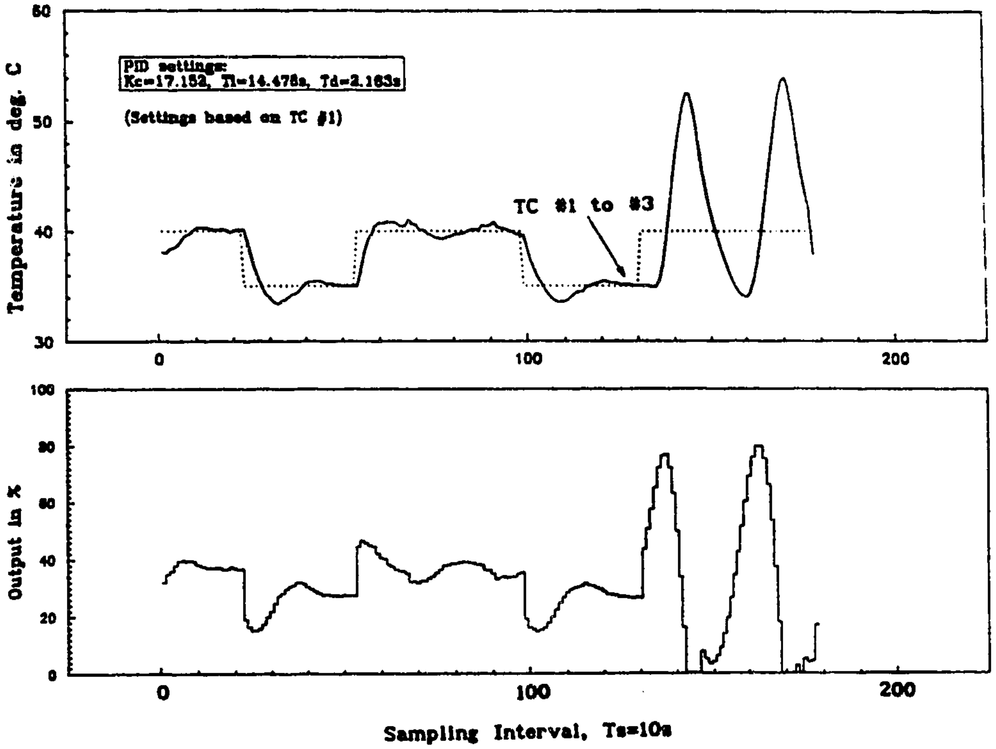


Figure 2.1, PID Control on the Continuous Stirred Tank Heater

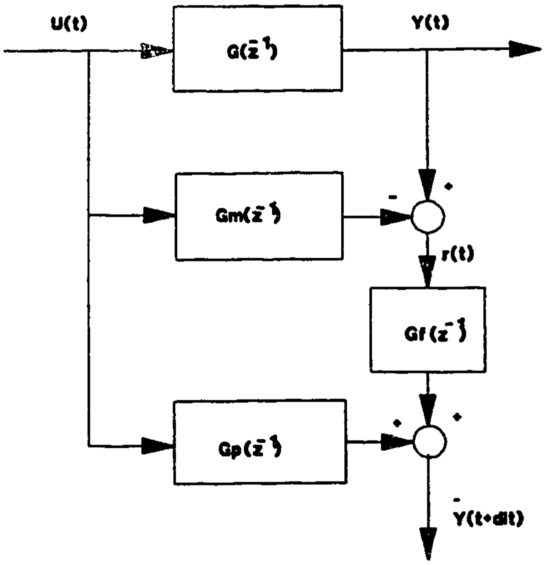


Figure 2.2. Block Diagram of a Smith Predictor

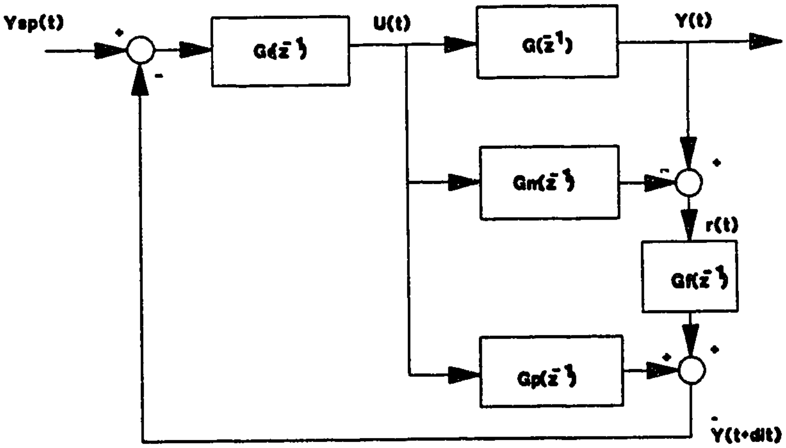


Figure 2.3. Block Diagram of a Predictive Control Scheme

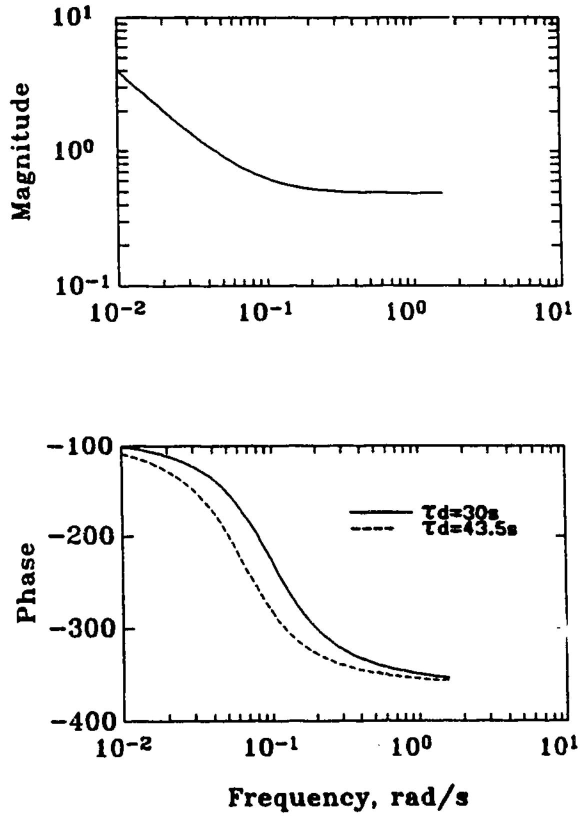


Figure 2.4, Bode Plot for the Open Loop Transfer Function of the Feedback System without the Smith Predictor

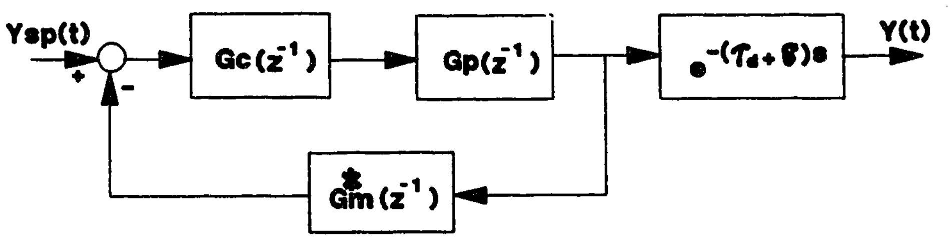


Figure 2.5, An Alternate Block Diagram of a PID + Smith Predictor Control Scheme

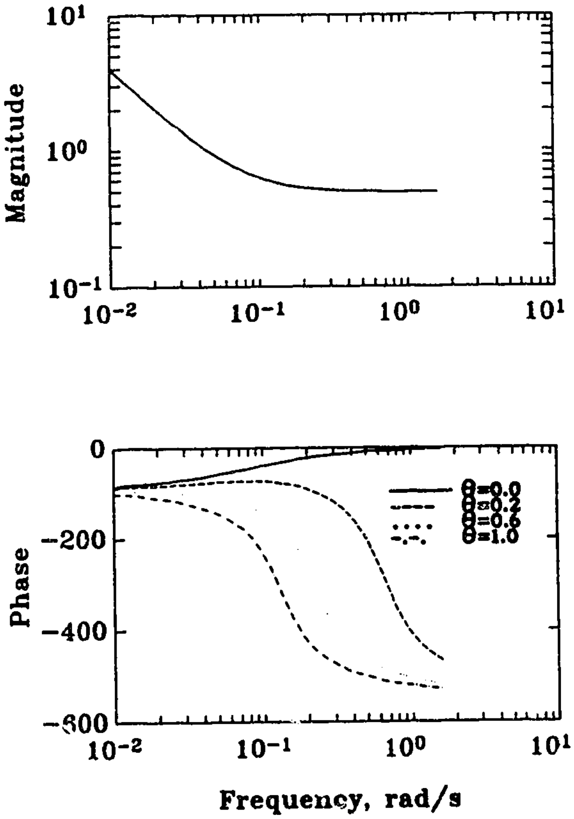


Figure 2.6. Bode Plot of the Open Loop Transfer Function of the Feedback System with the Smith Predictor

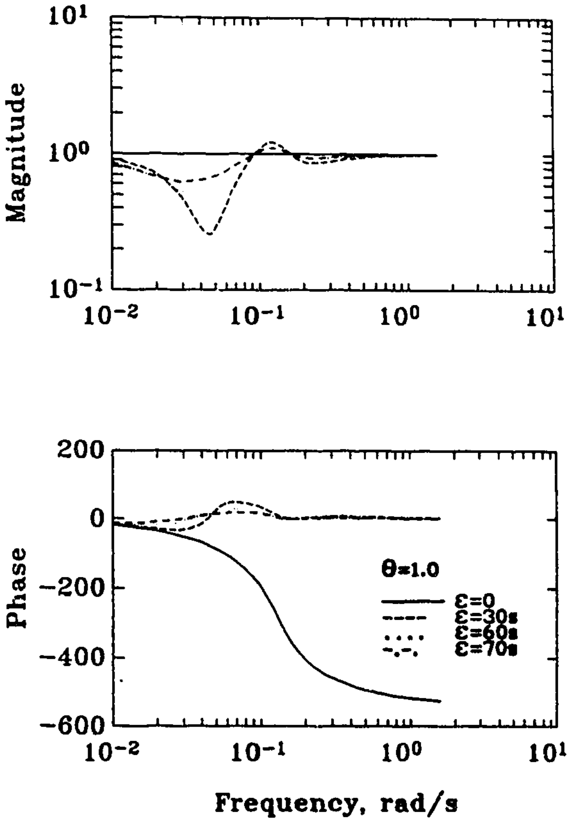


Figure 2.7, Bode Plot of $G\overset{\bullet}{m}(z^{-1})$

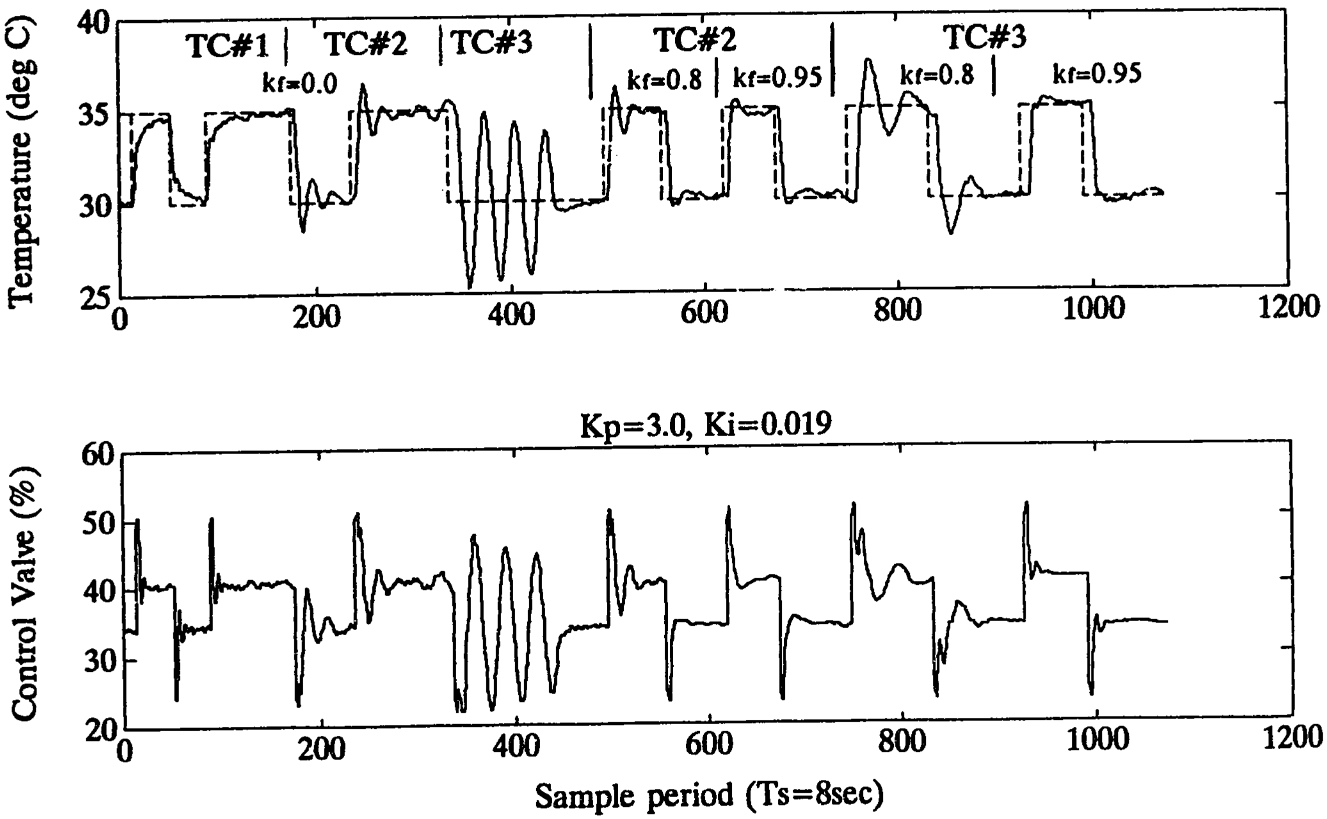


Figure 2.8. PI Control with Smith Predictor and an exponential residual filter under time delay mismatch

3 Robust Adaptive Control (RAC)

3.1 Introduction

Most adaptive predictive control schemes can be represented as in Figure 3.1. The adaptive control loop consists of three main components, namely (1) the parameter estimator which approximates the real process with a fixed order model through identification based on input and output data, (2) an optimal predictor that projects the undelayed output based on the model, and (3) the controller which generates the control signal for the process based on a controller design scheme (usually a minimization of a quadratic cost function) and the model parameters supplied by the parameter estimator. In order for the control scheme to work properly, both components have to be robust individually and collectively.

For a known process, one should have no difficulty in designing a control scheme that meets the desired control performance. However, since the orders of most chemical processes are unknown and the parameters are usually time-varying, modelling the process often presents a difficult problem for the adaptive control application. This is also true for modelling the time-delays of the processes. Most adaptive predictive control schemes presented in the literature require an exact knowledge of the process time-delay, which is not possible if the delay is time-varying. When the controller is applied to an unknown process with an approximated process order and an approximated process time-delay, unmodelled dynamics may lead to unexpected instability. Figure 3.2 shows the performance of a d-step ahead minimum variance predictive controller applied to a first order process with variable time delay. When the time-delay of the process is increased at $k>5300$ second, which corresponds to a time-delay under-estimation case, the controller becomes unstable. Therefore, a control system that is more robust to such variation is required, especially a system which is backed by theoretical proof of stability.

Cluett et al. (1988) have presented a general approach for the stability analysis of the adaptive predictive control system (APCS). This approach is used to prove global stability for a class of stable-inverse processes in the presence of bounded disturbances and unmodelled dynamics due to the process-model order mismatch. Based on the theorem, a new Robust Adaptive Predictive Controller (RAPC) was developed. The key features of this controller is a normalization factor introduced to allow formal proof of the stability of a controller, with the modelling error treated as a bounded disturbance. The controller was further extended by Cluett and Shah (1987) to include time-delay mismatch, which is treated as a form of plant-order mismatch. They presented sufficient conditions for the RAPC to ensure global stability in the presence of model-plant time-delay mismatch. The controller will, from hereon, be denoted by the acronym RAC which stands for Robust Adaptive Controller.

Since the RAC is designed to deal with time-delay mismatch, Chapter 3 is designated to apply RAC to such a process for evaluation. The RAC is also extended in this Chapter so that it can handle undeterministic, non-zero mean disturbances.

3.2 Derivation of RAC

The RAC uses a discrete, SISO, time-variant ARMA (auto-regressive moving average) model to represent the process. The derivation is as follows:

Let the process be represented by

$$A(z^{-1})Y(t) = B(z^{-1})z^{-d}U(t) + \Delta(t) \qquad (3.2.1)$$

where the order of $A(z^{-1})$ is n_1 and order of $B(z^{-1})$ is n_2.

Rewriting Equation (3.2.1) in vector notation gives:

$$Y(t) = \theta(t)^t\phi(t-d) + \Delta(t) \qquad (3.2.2)$$

where

$$\phi(t-d)^t = [Y(t-d)\ Y(t-d-1)\ldots\ U(t-d)\ U(t-d-1)\ldots]$$

$\phi(t-d)$ is the regressor vector containing the past input U and the past output Y. d is the process delay including the unit sample delay. θ is a vector containing the time-variant unknown process parameters. The dimensions of ϕ and θ are determined by the process order and the time delay. $\Delta(t)$ represents the effect of unmeasured disturbances and noise on the output at time t.

Let $\hat{d}$ be an estimate of the true delay d. The process model can be reexpressed as follows:

$$Y(t) = \theta_r(t)^t \phi_r(t-\hat{d}) + \Delta^m(t) \tag{3.2.3}$$

where

$$\Delta^m(t) = \theta_u(t)^t \phi_u(t-\hat{d}) + \Delta(t)$$

$\Delta^m(t)$ includes the unmodelled dynamics and the perturbation term. Subscript r stands for reduced order and subscript u stands for unmodelled terms. The order of θ is larger than or equal to the order of the parameter estimates $\hat{\theta}_r(t)$ in the predictive adaptive model estimation of $\hat{Y}(t|t)$ of the process output at time t:

$$\hat{Y}(t|t) = \hat{\theta}_r(t)^t \phi_r(t-\hat{d}) \tag{3.2.4}$$

The corresponding *a posteriori* estimation error is defined as

$$\begin{aligned} e(t|t) &= Y(t) - \hat{Y}(t|t) \\ &= Y(t) - \hat{\theta}_r(t)^t \phi_r(t-\hat{d}) \end{aligned} \tag{3.2.5}$$

where the estimated parameter vector is generated by an appropriate adaptive law which guarantees parameter convergence. The control input U(t) can be calculated by setting the predictive output equal to the setpoint.

$$Y_{sp}(t+\hat{d}) = \hat{\theta}_r(t)^t \phi_r(t-\hat{d}) \tag{3.2.6}$$

The RAC derived above requires the following assumptions to establish global stability:

A1: The upper bound $\bar{d}$ for d is known.

A2: The upper bounds $\overline{n_1}$ and $\overline{n_2}$ for n_1 and n_2 are known.

A3: The sequence $\{\|\Phi(k)\|\}$ is unbounded only if there is a subsequence $\{k_s\}$ such that

(a) $\lim_{k_s \to \infty} \|\Phi(k_s-1)\| = \infty$, and

(b) $|Y(k_s)| > \alpha_1 \|\Phi(k_s-1)\| - \alpha_2, \quad \forall k_s$

where

$0 < \alpha_1 < \infty$ and $0 \leq \alpha_2 < \infty$ and

$\Phi(t-1)^t = [Y(t-1) \ldots Y(t-\bar{d}-\overline{n_1}+1),\ U(t-1) \ldots U(t-2\bar{d}-\overline{n_2}+1)]$

Φ is an I/O vector which contains all the elements included in $\phi_r(t-\hat{d})$ and $\phi_u(t-\hat{d})$. This assumption is standard for process with stable-inverse (Martin-Sanchez, 1984).

A4: The upper bound on $\|\theta_u\|$, the norm of the unmodelled part of the process parameter vector, and an upper bound on the norm of the perturbation signal $|\Delta(k)|$ are known. This allows for an estimation on the upper bound of the dead-zone used in the parameter estimator.

To formally guarantee the boundedness of the unmodelled term in Equation (3.2.3), Cluett et al. (1988) introduced a normalization scheme to the process represented by Equation (3.2.2). It is defined as follows:

$$Y^n(t) = Y(t)/n(t)$$

$$x(t-d) = \phi(t-d)/n(t)$$

$$n(t) = \max\left(\max_{1<=i<=m}(|\Phi_i(t-d)|, c\right) \tag{3.2.7}$$

where Φ is the I/O regressor vector of dimension m which defines an upper bound of the plant order (see A3). The normalized system of Equation (3.2.2) becomes

$$Y^n(t) = \theta^t x(t-d) + \Delta(t)/n(t) \tag{3.2.8}$$

Note that over-estimation of the plant order does not cause any practical problem on the normalization scheme even if the choice of m is very large. For the reduced model case, where the model order and the model delay are chosen to be less than the process delay, the normalized process may be represented by

$$Y^n(t) = \theta_r^t x_r(t-\hat{d}) + \Delta^n(t) \tag{3.2.9}$$

where $\Delta^n(t)$ contains the unmodelled dynamics and the perturbation signal sequence $\{\Delta(t)\}$. Any term that is due to time-delay mismatch is also lumped into $\Delta^n(t)$. It was proven by Cluett et al. (1988) that for a bounded $\{\Delta(t)\}$, the $\{\Delta^n(t)\}$ is also bounded, ie. the boundedness of $\{\Delta^n(t)\}$ only requires the boundedness of the unmeasured noise and disturbance.

The a priori estimation error for the normalized system is

$$e^n(t|t-1) = Y^n(t) - \hat{\theta}_r(t-1)^t x_r(t-\hat{d}) \tag{3.2.10}$$

3.3 Parameter estimation in RAC

A normalized parameter estimation scheme with a dead-zone defined by Cluett et al. (1988) can be used for the RAC to update $\hat{\theta}_r$. It is a recursive projection type estimator and has been proven to result in a globally stable controller when coupled with a d-step ahead minimum variance controller under the assumptions A1 to A4 in section 3.2.

The algorithm for updating the parameter estimates is as follows:

$$|e^n(t|t-1)| = Y^n(t) - \hat{\theta}(t-1)^t x(t-\hat{d}) \tag{3.3.1}$$

$$\hat{\theta}(t) = \hat{\theta}(t-1) + \frac{\zeta(t)^2 e^n(t|t-1)x(t-\hat{d})}{1 + \zeta(t)^2 x(t-\hat{d})^t x(t-\hat{d})} \tag{3.3.2}$$

$\zeta(t)$ acts as a variable weighting factor to the parameter update and a "switch" for shutting estimation on or off as per the following criterion:

(1) $\zeta(t)^2 = 0$ iff

$$|e^n(t|t-1)| \leq \Delta_b'(\zeta_1,\Delta_b,t) < 2\Delta_b < \infty \tag{3.3.3}$$

where

$$\Delta_b'(\zeta_1,\Delta_b,t) = \frac{2+2\zeta_1^2\ x(t-\hat{d})^t x(t-\hat{d})}{2 + \zeta_1^2\ x(t-\hat{d})^t x(t-\hat{d})}\ \Delta_b \tag{3.3.4}$$

with

$$0<\zeta_1^2<\infty \quad \text{and} \quad \Delta_b > \Delta_m = \max_{0<t<=\infty} |\Delta^n(t)|$$

Δ_b is an estimate of an upper bound on the absolute value of the perturbation signal $\Delta^n(t)$ and Δ_m is the least upper bound. Estimation of Δ_b requires assumption A4.

(2) $\zeta_l^2<\zeta(t)^2\le\zeta_b(t)^2\le\zeta_u^2<\infty$ iff $|e^n(t|t-1)|>\Delta_b'(\zeta_l,\Delta_b,t)\ge\Delta_b$ (3.3.5)

where

(i) $\zeta_b(t)^2=\zeta_u^2$ if $|e^n(t|t-1)|>\Delta_b'(\zeta_u,\Delta_b,t)$ (3.3.6)

where

$$\Delta_b'(\zeta_u,\Delta_b,t)=\frac{2+2\zeta_u^2\,x(t-\hat{d})^t x(t-\hat{d})}{2+\zeta_u^2\,x(t-\hat{d})^t x(t-\hat{d})}\,\Delta_b \tag{3.3.7}$$

$$\text{(ii) }\zeta_b(t)^2=\frac{2(|e^n(t|t-1)|-\Delta_b)}{(2\Delta_b-|e^n(t|t-1)|)x(t-\hat{d})^t x(t-\hat{d})}\quad\text{if} \tag{3.3.8}$$

$$\Delta_b'(\zeta_l,\Delta_b,t)<|e^n(t|t-1)|\le\Delta_b'(\zeta_u,\Delta_b,t) \tag{3.3.9}$$

Therefore, the on/off scheme can be summarized as follows: Adaptation will stop if $|e^n(t|t-1)|\le\Delta_b'(\zeta_l,\Delta_b,t)$ (ie. case (1)). Otherwise, $\zeta(t)^2$ will be chosen in the interval $(\zeta_l^2,\zeta_b(t)^2]$ (ie. case (2)). (ζ_l,ζ_u) are used to ensure nonsingularity of the leading coefficient associated with U(t).

3.4 Control Weightings used in RAC

When the process itself contains unstable zeros (i.e. NMP), and/or when the time-delay is under-estimated, the inverse stable assumption A3 is violated. The resulting controller is unstable. In order to overcome this problem, Cluett et al. (1988) proposed augmentation of the process with a process input weighting Q as defined in Equation (3.4.1).

$$Z(t)=Y(t)+z^{-\hat{d}}Q(z^{-1})U(t) \tag{3.4.1}$$

where the steady state value of $Q(z^{-1})=0$.

Minimum variance control is then applied to the augmented process. With a proper choice of Q weighting, all the zeros of the augmented process will be in the stable region. This satisfies the stability requirement and therefore results in a stable controller. In general, if the process is represented as $Y(t)=\dfrac{B(z^{-1})}{A(z^{-1})}U(t-d)$, the augmented process will be given as

$$Z(t) = \frac{B(z^{-1})}{A(z^{-1})} U(t-d) + z^{-\hat{d}}Q(z^{-1})U(t-\hat{d})$$

$$= \frac{B(z^{-1})z^{-(d-\hat{d})} + Q(z^{-1})A(z^{-1})}{A(z^{-1})} U(t-\hat{d})$$

Notice that the zeros of the augmented process get closer to the open loop poles as Q gets larger. This means that if the process is open loop stable, the zeros of the augmented process will be in the stable region when a large enough Q is chosen. The following example illustrates the effect of the augmented process for a NMP process and when the time delay is under-estimated.

Example:

Case I: NMP process

Let the NMP, open loop stable process be

$$Y(t) = \frac{0.9+1.5z^{-1}}{1-0.7z^{-1}} U(t-3)$$

Choosing $Q=\lambda\Delta$, the augmented process is given by

$$Z(t) = \frac{0.9+1.5z^{-1}}{1-0.7z^{-1}} U(t-3) + \lambda(1-z^{-1})U(t-3)$$

$$= \frac{(0.9+\lambda)+(1.5-\lambda)z^{-1}}{1-0.7z^{-1}} U(t-3)$$

Therefore the process zero will be in the stable region when λ is greater than 0.3.

Case II: time delay under-estimation

Let the estimated time delay $\hat{d}=1$. The augmented plant is given by:

$$Z(t) = \frac{0.9+1.5z^{-1}}{1-0.7z^{-1}} U(t-3) + \lambda\Delta U(t-1)$$

$$= \frac{\lambda-1.7\lambda z^{-1}+(0.7\lambda+0.9)z^{-2}+1.5z^{-3}}{1-0.7z^{-1}} U(t-1)$$

Figure 3.3 shows the roots of the augmented plant zeros as λ varies. It can be seen that all the roots are in the stable region when $\lambda > 12$. Notice that two extra zeros are introduced when the delay is under-estimated by two sample periods.

λ affects the leading coefficients of the numerator polynomials in both cases, especially in the second case where the leading coefficient is λ. Since the inverse of this value is essentially the controller gain, adjusting λ will have a direct effect on the performance. The higher the value, the slower the response speed.

3.5 Reconstruction of RAC

3.5.1 Introduction

The parameter estimator in RAC uses a dead-zone to prevent the destablizing parameter drift associated in the presence of unmodelled dynamics and disturbances (Rohrs et al., 1984). However, the drawback of using a dead-zone is that when the parameter estimation stops, prediction is carried out using the 'unconverged' set of parameters, which leads to offset in prediction. This prediction offset shows up in the plant output (referred to a controller offset) as shown in Figure 3.4. (See Figure 3.5 where the offset is completely eliminated when the dead-zone is removed.) This suggests that the predictor has to be modified so that it will give unbiased prediction, especially when the dead-zone is in effect. Steady-state bias and non zero-mean disturbances, which are not sufficiently represented by an ARMA model, are other sources contributing to the offset problem. This requires the choice of a model that is able to represent these effects.

3.5.2 Choice of process model

The CARIMA (Controlled AutoRegressive Integrated Moving Average) model is chosen to represent the process. This incremental model is a more 'realistic model' that accounts for undeterministic and non-zero mean disturbances than other positional models such as the ARMA or the CARMA models (Tuffs and Clarke, 1985). It is because the model provides an integrator in the forward path to remove controller offset (like the integration mode in a PID controller). In fact, this model has widely been discussed and utilized by other authors such as Fessl and Karny (1979), Harris et al. (1980), Belanger (1983), Gawthrop (1982), and Clarke et al. (1983) for tackling offset problems in the design of adaptive regulators.

3.5.3 RAC with the CARIMA model

The 'integrating form' of the Clarke-Gawthrop Self tuner (1979) forms the backbone of the reconstruction, but the unique features of the RAC are kept, namely (1) the idea of using an augmented plant to ensure the inverse stable criteria, (2) the use of normalization and a dead-zone in the parameter estimator to guarantee robustness in the on-line process model adaptation. Furthermore, a P weighting is added as the process output weighting in the augmented process and a R weighting is added as a setpoint weighting. All P, Q, and R weightings are extended to rational transfer function weightings in the backward shift operator. The resulting RAC has two integrators incorporated to remove offset: (1) an integrator in the forward path to ensure that there is no setpoint tracking offset, and (2) an integrator in the predictor to remove the prediction offset.

Consider the CARIMA model:

$$A(z^{-1})Y(t) = B(z^{-1})U(t-\partial) + C(z^{-1})\xi(t)/\Delta \qquad (3.5.1)$$

The augmented process is given by

$$Z(t) = P(z^{-1})Y(t) + Q(z^{-1})U(t-\partial) \qquad (3.5.2)$$

where

$$P(z^{-1}) = \frac{P_n(z^{-1})}{P_d(z^{-1})}$$

and

$$Q(z^{-1}) = \frac{Q_n(z^{-1})}{Q_d(z^{-1})}$$

Figure 3.6 is a block diagram showing the structure of the augmented process. In order that the process output, Y(t), tracks the augmented process output, $Z(t)$, $P(z^{-1})$ and $Q(z^{-1})$ have to be selected such that $P(1)=1$ and $Q(1)=0$.

Multiplying Equation (3.5.1) with $P(z^{-1})$ and rearranging, we have

$$P(z^{-1})Y(t) = \frac{P_n(z^{-1})B(z^{-1})}{P_d(z^{-1})A(z^{-1})} U(t-\hat{d}) + \frac{P_n(z^{-1})C(z^{-1})\xi(t)}{P_d(z^{-1})A(z^{-1})\,\Delta} \tag{3.5.3}$$

After substituting Equation (3.5.2) to (3.5.3), the augmented plant CARIMA model is obtained:

$$\therefore \quad Z(t) = \frac{B'(z^{-1})}{A'(z^{-1})} U(t-\hat{d}) + \frac{C'(z^{-1})}{A'(z^{-1})\Delta} \xi(t) \tag{3.5.4}$$

where

$$A'(z^{-1}) = P_d(z^{-1})Q_d(z^{-1})A(z^{-1})$$

$$B'(z^{-1}) = z^{\hat{d}-d}P_n(z^{-1})Q_d(z^{-1})B(z^{-1})+P_d(z^{-1})Q_n(z^{-1})A(z^{-1})$$

$$C'(z^{-1}) = P_n(z^{-1})Q_d(z^{-1})C(z^{-1})$$

The second term on the RHS of Equation (3.5.4) involves knowledge of future noise. This can be resolved by defining the following identity:

$$\frac{C'(z^{-1})}{A'(z^{-1})\Delta} = E(z^{-1}) + \frac{F(z^{-1})}{\Delta A'(z^{-1})} z^{-\hat{d}} \tag{3.5.5}$$

where

$$\delta E = \hat{d}-1$$

$$\begin{aligned} \delta F &= \max(\delta C'-1,\ \delta A') \\ &= \max(\delta P_n+\delta Q_d-1,\ \delta P_d+\delta Q_d+n) \end{aligned}$$

Substituting Equation (3.5.5) into (3.5.4) and shifting $\hat{d}$ step forward, the augmented process model becomes

$$Z(t+\hat{d}) = \frac{B'(z^{-1})}{A'(z^{-1})} U(t) + \frac{F(z^{-1})}{A'(z^{-1})\Delta} \xi(t) + E(z^{-1})\xi(t+\hat{d}) \tag{3.5.6}$$

The past noise term can be reconstructed from the process model. From Equation (3.5.4),

$$\xi(t) = \frac{A'(z^{-1})\Delta Z(t) - B'(z^{-1})\Delta U(t-\hat{d})}{C'(z^{-1})} \tag{3.5.7}$$

Therefore, substituting Equation (3.5.7) into (3.5.6), and combining with Equation (3.5.5) gives the following structure for the regression model:

$$Z(t+\hat{d}) = \frac{F(z^{-1})}{C'(z^{-1})} Z(t) + \frac{G(z^{-1})}{C'(z^{-1})} \Delta U(t) + E(z^{-1})\xi(t+\hat{d}) \tag{3.5.8}$$

where $G(z^{-1}) = E(z^{-1})B'(z^{-1})$

The optimal prediction of $Z(t+\hat{d})$, $Z^{*}(t+\hat{d})$, is now given by

$$Z^{*}(t+\hat{d}) = \frac{F(z^{-1})}{C'(z^{-1})} Z(t) + \frac{G(z^{-1})}{C'(z^{-1})} \Delta U(t) \tag{3.5.9}$$

with the estimation error equals to

$$E\xi(t+\hat{d}) = Z(t) - Z^{*}(t) \tag{3.5.10}$$

To determine the control law that gives minimum variance control to the augmented process, the following cost functional is minimized with respect to U(t):

$$J_{RAC} = [\, R(z^{-1})Y_{sp}(t+\hat{d}) - Z^{*}(t+\hat{d}) \,]^2 \tag{3.5.11}$$

where $Z^{*}(t+\hat{d}) = E\{Z(t+\hat{d})\}$

The minimization, which is obtained by setting $\partial J/\partial U(t)=0$, is equivalent to setting the prediction output to the desired setpoint. Therefore, the control law is given by

$$R(z^{-1})Y_{sp}(t+\hat{d}) = \frac{F(z^{-1})}{C'(z^{-1})} Z(t) + \frac{G(z^{-1})}{C'(z^{-1})} \Delta U(t) \tag{3.5.12}$$

$$\text{or} \quad U(t) = \frac{\left[R(z^{-1})Y_{sp}(t+\hat{d}) - \frac{F(z^{-1})}{C'(z^{-1})}Z(t)\right]C'(z^{-1})}{G(z^{-1})\Delta} \tag{3.5.13}$$

Discussion:

There are four points worth noting about the controller defined in Equation (3.5.12):

(1) It has an integrator in the forward path that arises from the CARIMA model. This integrator eliminates offset between the weighted setpoint, $R(z^{-1})Y_{sp}(t+\hat{d})$, and the weighted augmented process output, $\frac{F(z^{-1})}{C'(z^{-1})}Z(t)$.

(2) It requires that $G(z^{-1})$ be stable. This is equivalent to the inverse stable criterion stated in assumption A3 since G=EB'. As can be seen from Equation (3.5.8), this can be achieved by proper choice of $P(z^{-1})$ and $Q(z^{-1})$ weightings.

(3) The order of the controller is fixed by the degree of $F(z^{-1})$ and $G(z^{-1})$ polynomials. According to Equations (3.5.5) and (3.5.9), their degrees are affected by the process model order, delay, and the degrees of the weightings chosen. Since it is undesirable to have a high order controller in general, the degree of $F(z^{-1})$ and $G(z^{-1})$ can be deliberately chosen to be less than the design values from Equations (3.5.5) and (3.5.9) such that a reduced order controller is obtained. This is possible because the RAC is designed to handle unmodelled dynamics as long as the assumption A4 is satisfied by choosing a sufficiently large dead-zone.

(4) $\frac{F(z^{-1})}{C'(z^{-1})}$ can be interpreted as a residue filter. Substituting the Diophantine identity defined in Equation (3.5.5) into the optimal prediction model in Equation (3.5.9) gives

$$Z^*(t+\hat{d}) = \frac{B'(z^{-1})}{A'(z^{-1})}U(t) + \frac{F(z^{-1})}{C'(z^{-1})}\left(Z(t) - \frac{B'(z^{-1})}{A'(z^{-1})}z^{-\hat{d}}U(t)\right) \tag{3.5.14}$$

Equation (3.5.14) can be represented by the block diagram in

Figure 2.2. $\frac{F(z^{-1})}{C'(z^{-1})}$ is the disturbance filter which must satisfy the following two criteria: (1) $C'(z^{-1})$ must be stable in order that the predictor is stable. According to Equation (3.5.4), if the $C(z^{-1})$ polynomial of the CARIMA model is chosen to be unity, the poles of the filter are uniquely defined by the user chosen $P_n(z^{-1})$ and $Q_d(z^{-1})$ polynomials. Choosing $C(z^{-1})$ to be unity implies that the CARIMA model uses a step model as the disturbance model. This is a fair choice as long as the major disturbances to be rejected are step type disturbances. If such is the choice in the controller design, the disturbance filter in Equation (3.5.14) is bound to be stable. (2) The steady-state value of the filter must be unity in order that the prediction is not biased (see Equation (3.5.14)). This, however, is difficult to guarantee and this prediction bias eventually shows up as an offset between the process output and the setpoint. The next section will focus on the method of enforcing this requirement by incorporating an integrator to the predictor.

3.5.4 Removing Offset

The problem that causes RAC to have offset lies in the fact that the parameters of F and G obtained from the estimator are biased during periods of low excitation in the input signal. This will invariably occur when the parameters are frozen due to the dead-zone, which leads to a prediction offset. To ensure that there is no prediction offset when the parameter estimation is turned off, an integrator can be incorporated in the prediction step by splitting the F polynomial in the following manner:

$$F(z^{-1}) = F'(z^{-1})\Delta + C'(1) \qquad (3.5.15)$$

where $\delta F' = \delta F - 1$

Substitute Equation (3.5.15) into (3.5.8) to obtain the following new regression model:

$$Z(t+\hat{d}) = \frac{F'(z^{-1})}{C'(z^{-1})}\Delta Z(t) + \frac{C'(1)}{C'(z^{-1})} Z(t) + \frac{G(z^{-1})}{C'(z^{-1})}\Delta U(t) +$$

$$E(z^{-1})\xi(t+\hat{d}) \qquad (3.5.16)$$

It is well known that the $C(z^{-1})$ polynomial is not easy to identify on-line especially because of the different rate of convergence of the process model and the disturbance model. In order to get around this problem, a fixed polynomial $T(z^{-1})$ is used as a proxy for the $C(z^{-1})$ polynomial. Experience has shown that for a reasonably chosen sampling period, a value of $T(z^{-1})$ typically chosen to be $(1-0.8z^{-1})$ suffices. This method is referred to as the T-filter by authors such as Clarke et al. (1984). Using the certainty equivalence principle to replace the unknown F' and G polynomial, the adaptive optimal prediction of the $\hat{d}$-step ahead augmented process output $Z^{*}(t+\hat{d})$ is given by

$$Z^{*}(t+\hat{d}) = \frac{\hat{F}'(z^{-1})}{T'(z^{-1})}\Delta Z(t)+\frac{C'(1)}{T'(z^{-1})}Z(t)+\frac{\hat{G}(z^{-1})}{T'(z^{-1})}\Delta U(t) \qquad (3.5.17)$$

where $T'(z^{-1}) = T(z^{-1})P_n(z^{-1})Q_d(z^{-1})$

The control law for the modified RAC is (the superscript f stands for a signal filtered by T')

$$R(z^{-1})Y_{sp}(t+\hat{d}) = \hat{F}'(z^{-1})\Delta Z^{f}(t)+\hat{G}(z^{-1})\Delta U^{f}(t)+C'(1)Z^{f}(t) \qquad (3.5.18)$$

3.5.5 Parameter Estimation for the modified RAC

The prediction model for the reduced order controller in Equation (3.5.16) can be rewritten in vector form for parameter estimation:

$$\psi(t) = \theta_r(t)^{t}\phi_r(t-\hat{d}) + \Delta^{m}(t) \qquad (3.5.19)$$

where

$$\Delta^{m}(t) = \theta_u(t)^{t}\phi_u(t-\hat{d}) + E(Z^{-1})\xi(t)$$

$$\psi(T) = Z(t) - T'(1)Z^{f}(t-\hat{d})$$

$$\phi_r(t-\hat{d})^{t} = [\Delta Z^{f}(t-\hat{d})\ \Delta Z^{f}(t-\hat{d}-1)\ \ldots\ \Delta U^{f}(t-\hat{d})\ \Delta U^{f}(t-\hat{d}-1)\ \ldots\ ]$$

$$\theta_r(t)^{t} = [f_0\ f_1\ \ldots\ g_0\ g_1\ \ldots\ ]$$

The parameter estimation scheme from the original RAC defined in Equations (3.3.4) to (3.3.12) can be used for the modified RAC with slight modification to the normalization.

$$\psi^{n}(t) = \psi(t)/n(t) \tag{3.5.20}$$

$$x(t-\hat{d}) = \phi(t-\hat{d})/n(t) \tag{3.5.21}$$

$$n(t) = \max(\max_{1<=l<=m}(|\Phi_{l}(t-\hat{d})|,c)) \tag{3.5.22}$$

where Φ is a vector containing positional I/O data for the augmented process with dimension m. c is a constant to avoid division by zero. The nominal value of c is one.

3.5.6 Some interpretation on P and Q weighting of RAC

The control law of the RAC is obtained by minimizing the following cost functional with respect to U(t):

$$J = E\{[Z(t+\hat{d}) - R(z^{-1})Y_{sp}(t+\hat{d})]^{2}\} \tag{3.5.23}$$

where $Z(t+\hat{d})$ is defined as

$$Z(t+\hat{d}) = P(z^{-1})Y(t+\hat{d})+Q(z^{-1})U(t) \tag{3.5.24}$$

$P(z^{-1})$, $Q(z^{-1})$, and $R(z^{-1})$ weightings are user specified rational transfer functions in z^{-1} which make the RAC control algorithm flexible and practical. Since $R(z^{-1})$ is not involved in the closed-loop path (see Equation (3.5.17)), it can simply be interpreted as a setpoint filter. However, $P(z^{-1})$ and $Q(z^{-1})$ weightings deserves a closer look because they affect process modelling, estimation, as well as controller design.

$P(z^{-1})$ weighting

Consider the augmented process in Equation (3.5.4):

$$Z(t) = \frac{B'(z^{-1})}{A'(z^{-1})}U(t-\hat{d}) + \frac{C'(z^{-1})}{A'(z^{-1})\Delta}\xi(t) \tag{3.5.4}$$

where

$$A'(z^{-1}) = P_{d}(z^{-1})Q_{d}(z^{-1})A(z^{-1})$$

$$B'(z^{-1}) = Pn(z^{-1})Qd(z^{-1})B(z^{-1})+Pd(z^{-1})Qn(z^{-1})A(z^{-1})$$

$$C'(z^{-1}) = Pn(z^{-1})Qd(z^{-1})C(z^{-1})$$

If $Qn(z^{-1})=0$ and $Qd(z^{-1})=1$, the augmented process becomes

$$\begin{aligned} Z(t) &= P(z^{-1})\frac{B(z^{-1})}{A(z^{-1})} U(t-\hat{d}) + P(z^{-1})\frac{C(z^{-1})\xi(t)}{A(z^{-1})\ \Delta} \\ &= P(z^{-1})Y(t) \end{aligned} \tag{3.5.25}$$

Apart from adding zeros and poles to the augmented process, the most prominent effect of the $P(z^{-1})$ is to shape the disturbance as can be seen from Equation (3.5.25). This effect is more obvious when the closed loop system is examined. Substituting Equation (3.5.10) into Equation (3.5.11), and setting $Q(z^{-1})=0$, the control law is given by

$$R(z^{-1})Y_{sp}(t+\hat{d}) = P(z^{-1})Y(t+\hat{d}) - E(z^{-1})\xi(t+\hat{d}) \tag{3.5.26}$$

Rearranging Equation (3.5.26) gives the closed-loop model:

$$Y(t) = \frac{R(z^{-1})}{P(z^{-1})} Y_{sp}(t) + \frac{E(z^{-1})}{P(z^{-1})} \xi(t) \tag{3.5.27}$$

This leads to two important interpretations of the $P(z^{-1})$ weighting:

(1) The inverse $P(z^{-1})$ presented as an internal model for model reference control. The process output tracks the desired setpoint through the $R(z^{-1})/P(z^{-1})$ transfer function. It is suggested that the reference model $P(z^{-1})^{-1}$ be chosen as a discretized first order model of unity gain and time constant faster than the process time constant. The similarities between this design of model reference control and the model reference adaptive control (MRAC) are broadly discussed by Egardt (1978).

(2) The controller rejects disturbances according to the choice of the $P(z^{-1})$. For the special case where $P(z^{-1})=1$, the controller rejects step load disturbances in only $\hat{d}$ sample periods. As discussed by Tuffs

and Clarke (1985), this controller rejects step type disturbances with minimum time compared with other minimum variance type controller designs.

$Q(z^{-1})$ weighting

$Q(z^{-1})$ weighting has many roles: It reduces the excessive control activity, it shapes the control action according to its structure, and it places the augmented process zeros so that the controller can handle some non-minimum phase systems or systems with poorly damped zeros. Consider the augmented process in Equation (3.5.4):

$$Z(t) = \frac{B'(z^{-1})}{A'(z^{-1})} U(t-\hat{d}) + \frac{C'(z^{-1})}{A'(z^{-1})\Delta} \xi(t) \tag{3.5.4}$$

where

$$A'(z^{-1}) = P_d(z^{-1})Q_d(z^{-1})A(z^{-1})$$

$$B'(z^{-1}) = P_n(z^{-1})Q_d(z^{-1})B(z^{-1})+P_d(z^{-1})Q_n(z^{-1})A(z^{-1})$$

$$C'(z^{-1}) = P_n(z^{-1})Q_d(z^{-1})C(z^{-1})$$

As mentioned before, $Q(z^{-1})$ affects the positions of the zeros of the augmented process. By proper choice of $Q_n(z^{-1})$ and $Q_d(z^{-1})$, $B'(z^{-1})$ will have all its roots in the stable region. Therefore, assumption A3 can be satisfied. When $P(z^{-1})=1$, the control law resulted from the cost function J is

$$U(t) = \frac{R(z^{-1})Y_{sp}(t+\hat{d})-Y(t+\hat{d}|t)}{Q(z^{-1})} \tag{3.5.28}$$

Minimum variance control can be realized by setting $Q(z^{-1})=1$ and $R(z^{-1})=1$. This is known to give fastest closed-loop response in regulatory control (dead-beat control). However, without the filtering action of $Q(z^{-1})$, the control action calculated this way is very sensitive to the variance of the parameter estimates and the accuracy of the predicted output. The role of $Q(z^{-1})$ is to detune the controller to give conservative control. Three common choices of $Q(z^{-1})$ are outlined as follows:

(1) $Q = \lambda$

This is equivalent to a proportional control on the predicted error on setpoint where the proportional constant is equal to $1/\lambda$. According to classical control theory, offset will occur if proportional control is applied to an open-loop stable system except when the proportional gain is infinitely large. Therefore, it is preferred that λ is as small as possible to avoid offset problem. However, this is not practical because λ plays another role of shifting the augmented process zeros to the stable region. Its value is therefore not completely user chosen and it is bounded by some lower limit beyond which the augmented process zeros are unstable. This was demonstrated in the example in section 3.4.

(2) $Q = \lambda\Delta$

This is equivalent to the Integral only control. λ resembles the integral time constant τ_I. The selection of $Q(z^{-1})$ is able to eliminate controller offset, but the closed loop response is expected to be more sluggish and oscillatory. Integral only control is therefore seldom used alone. PI or PID control structure can be realized by using a more complicated structure of $Q(z^{-1})$.

(3) Q = PI structure

The conventional discrete PI controller takes the following structure:

$$L(z^{-1}) = \frac{(K_p+K_IT_s) - (K_p)z^{-1}}{1-z^{-1}} \tag{3.5.29}$$

Therefore, the PI structure of $Q(z^{-1})$ is

$$Q(z^{-1}) = \frac{\lambda\Delta}{q_0 + q_1z^{-1}} \tag{3.5.30}$$

where $q_0 = \lambda(K_p+K_IT_s)$ and

$q_1 = \lambda K_p$

The proportional action speeds up the response rate while the integral action eliminates controller offset. This adds flexibility to the controller but increases the complexity. Instead of 1 tuning parameter, the PI structure requires 2 tuning parameters.

3.6 Test Run on the RAC by Simulation

3.6.1 Description of process

The modified RAC algorithm defined by Equations (3.5.4) and (3.5.18) was implemented and tested using computer simulation. The process is a benchmark example commonly used for analyzing robustness of adaptive controllers (Rohrs et al., 1984). It is a third order process which contains second order, high frequency dynamics as defined in Equation (3.6.1). A time-delay term is added for time-delay mismatch case study.

$$Y(s) = \frac{2}{s+1}\,\frac{229}{s^2+30s+229}\,e^{-0.3s}U(s) \qquad (3.6.1)$$

The discretized version of the process is minimum-phase when sampled faster then 0.2 second. Two different sampling periods of 0.1 second and 0.3 second were chosen for controller testing:

For sampling at Ts=0.3s, the discretization of Equation (3.6.1) gives model I for simulation:

$$Y(t) = \frac{0.313+0.193z^{-1}+0.00292z^{-2}}{1-0.759z^{-1}+0.0137z^{-2}-0.0000914z^{-3}}\,U(t-1) \qquad (3.6.2)$$

For sampling at Ts=0.1s, the discretization of Equation (3.6.1) gives model II for simulation:

$$Y(t) = \frac{0.037+0.0717z^{-1}+0.0078z^{-2}}{1-1.3422z^{-1}+0.4455z^{-2}-0.045z^{-3}}\,U(t-4) \qquad (3.6.3)$$

The dead-zone is chosen by the following equation (Cluett, 1988):

$$\begin{aligned}\Delta_b &= \sqrt{(n_p-n_r)}\,\|\theta_u\| \qquad (3.6.4)\\ &= 0.0243\end{aligned}$$

where n_p is the process regressor length, n_r is the reduced order regressor length, and $\|\theta_u\|$ is the norm of the unmodelled terms.

The simulation settings are tabulated in the following Table 3.1.

Table 3.1, Controller settings for simulation runs

Run#	Figure	order	delay	Ts	$P(z^{-1})$	$Q(z^{-1})$	$R(z^{-1})$	$T(z^{-1})$	Δ_b
1	3.7	3/1	0/0	0.3	1	0	1	1	0.0243
2	3.8	3/1	0/0	0.3	1	0	1	1	0.01
3	3.9	3/3	3/0	0.3	1	0	1	1	0.0243
4	3.11	3/3	3/0	0.3	1	5Δ	1	1	0.0243
5	3.12	3/3	3/0	0.3	1	$\frac{5\Delta}{(1-.7z^{-1})}$	1	1	0.0243
6	3.13	3/3	3/0	0.1	1	5Δ	1	1	0.0243
7	3.14	3/3	3/0	0.3	1	$\frac{3\Delta}{(1-.7z^{-1})}$	1	1	0.0243
8	3.15	3/3	3/0	0.3	$5(1-.8z^{-1})$	$\frac{3\Delta}{(1-.7z^{-1})}$	1	1	0.0243

(order= process/model order; delay= process/model delay;)
(Ts= sampling time in second)

3.7.2 Discussion of Results

The results of applying RAC to model I is shown in Figure 3.7. The controller is stable, but the controller output contains high frequency oscillation. This is referred to as 'ringing' in the control literature, and is caused by the controller trying to cancel poorly damped negative controller poles (notice from the parameter trajectories that the controller pole is equal to -0.8). These poles are normally not desired because they make the controller change sign rapidly and cause excessive wear in the final control element. The ringing phenomenon can be alleviated by either increasing the sampling period or introducing controller weighting to obtain desired zero locations for the augmented process (this will be demonstrated later). The control variable tracks the setpoint exactly even when a dead-zone, Δ_b, is used. This is a big improvement compared with the original RAC design. The dead-zone represents an upper bound to the unmodelled dynamics. If this value is reasonably reduced so that the parameter estimator can be run more often while still maintaining the function of the dead-zone, the performance of the controller is expected to increase. This is demonstrated in Figure 3.8 where Δ_b=0.01. There is a mild improvement performance for both the input and output trajectories.

Figure 3.9 demonstrates the effect of time delay mismatch (no process order mismatch). The controller 'blows up' because the parameter associated with U(t), go, monotonically converges to a very small value. $Q(z^{-1})=\lambda\Delta$ is chosen. The value of λ is chosen such that the augmented process has all zeros in the stable region. Figure 3.10 shows the roots locus of the augmented process zeros with λ increasing from 1 to 6 (all the roots are stable when $\lambda \geq 5$). These zeros migrate towards the center as λ increases. Notice that these negative zeros are damped out rapidly so that no pronounce effect of ringing can be seen. The result is shown in Figure 3.11. The closed-loop response is stable if not slightly oscillatory. Notice that the go parameter in Figure 3.11 is fairly large compared with that in Figure 3.9. As mentioned in section 3.5, this structure of $Q(z^{-1})$ is equivalent to an Integral only controller. Whereas the Integral only controller can eliminate offset when applied to an open-loop stable process and increase the response speed, it is seldom used alone because of its overshoot and oscillatory behavior. Figure 3.12 shows the effect of using an inverse PI structure for $Q(z^{-1})$ weighting. The corresponding PI constants are chosen to be Kp=0.158 and Ki=0.14 repeats per second. The performance of the controller is very much improved. As shown in Figure 3.14, when λ is decreased to 3, the go parameter converges to a smaller number. This explains the more oscillatory behavior. Normally, λ is thought to be the proportional band and is used for tuning the controller.

The closeness of model-following is traded against control activity when $P(z^{-1})$ weighting is used with conjunction with $Q(z^{-1})$ weighting (Clarke and Gawthrop, 1979). It was recommended that to design for a certain performance, choose $P(z^{-1})$ as the inverse of the desired closed loop model and $Q(z^{-1})$ in the form of an inverse PI structure. Then λ can be used as a tuning knob for the controller performance (Clarke, 1984). $P(z^{-1})$ weighting in the form of a 1st order lead filter is applied to the process and the result is shown in Figure 3.15. The performance is improved compared with

Figure 3.14. The control is tighter, overshoot is less, and oscillation is reduced. As mentioned in section 3.5, $P(z^{-1})$ can be thought as the inverse model for model following control when $Q(z^{-1})=0$. Therefore, the choice of $P=5(1-0.8z^{-1})$ places the closed loop pole to $z=0.8$. This slows down the response and stabilizes the closed loop performance.

Finally, the controller is tested on model II which contains NMP zeros. The process model is first order and the time-delay is underestimated to be zero. Q is chosen as 5Δ. The result is shown in Figure 3.13. When the parameter estimates converge to a set of values, the output responses are sluggish but stable. This demonstrates that the RAC performs satisfactorily on a NMP process under process order mismatch and time-delay mismatch. The responses can again be improved by using an inverse PI equivalent $Q(z^{-1})$ weighting.

The simulation results concluded that the modified RAC performs satisfactorily in the presence of time delay mismatch. It is robust against offset that results from poor prediction. With a proper chosen $Q(z^{-1})$ weighting, RAC is able to control MP and NMP processes with little knowledge on the model order and model delay. The $P(z^{-1})$ weighting provides model following feature to the RAC design. However, a *priori* knowledge of the process is required to obtain good tuning parameters for $P(z^{-1})$ and $Q(z^{-1})$ weightings. This presents as one drawback of the RAC.

3.7 Test of RAC by Experimental Runs

3.7.1 Introduction

The RAC was tested on a pilot scale Continuous Stirred Tank Heater (CSTH). Chapter 5 contains a detail description of the apparatus. The purpose of the experiment is to demonstrate the robustness of RAC in the presence of time delay mismatch. This is studied by setting the model delay $\hat{d}$ to 5 (which is roughly the delay of thermo-couple #2) and using thermo-couple #3 to measure the temperature. Different $P(z^{-1})$ and $Q(z^{-1})$ weightings are tested for servo and regulatory control.

Choice of model

Fractional delay results with the above choice of sampling period. The modified z-transform of a 1st order model is used to represent the system (the process is sampled at 10 seconds and measurement is taken from thermo-couple #2):

$$Y(t) = \frac{b_0+b_1z^{-1}}{1-a_0z^{-1}} z^{-5}U(t-1)$$

Initialize RAC

The parameter vector is initialized to one, ie $\theta(0)=[1\ 1\ \ldots\ 1]$. This is a bad choice of initial parameter since these values are far from the true values. Therefore a sequence of step changes is introduced at the first 20 sampling intervals for open loop identification.

3.7.2 Discussion on the choice of the dead-zone

The algorithm for choosing the dead-zone for RAC is in Equations (3.3.3) to (3.3.9). In practice, a small dead-zone renders the parameter estimator more susceptible to parameter drifting during the periods of low excitation. On the other hand, a large dead-zone results in poor parameter estimates because the parameters may not be updated as frequently as necessary. The choice of the size of the dead-zone is really a compromise between these two cases.

Choosing the size of the dead-zone actually requires knowledge of the process as well as disturbances. Equation (3.6.4) is for the estimation of the dead-zone for the deterministic case. If perturbation $\Delta(t)$ is present, the dead-zone should be chosen to be the maximum of that from Equation (3.6.4) and the largest size of $\{\Delta(t)\}$. However, this knowledge is usually difficult, if not impossible, to obtain in practice. From the simulation results (Runs 1 and 2), a smaller dead-zone gives only mild improvement to the output performance and parameter update when there is no disturbance. The user has to choose a dead-zone that can accommodate the biggest possible disturbance. One practical method is the choose the dead-zone according to the maximum perceivable prediction error:

From Equations (3.3.1) and (3.3.7), the maximum value of $Y^n(t)=1$ and

$|Y^n-\hat{Y}^n|>2\Delta_b$. The dead-zone can therefore be chosen to be the fractional prediction error divided by 2:

$$\Delta_b = (err)/2 \qquad 0\le err\le 1$$

For all the experimental runs, the dead-zone is fixed at Δ_b=0.05 which means that a maximum of 10% in the prediction error is allowed.

3.7.3 Results and Discussions

The settings of the experimental runs are tabulated in Table 3.2.

Table 3.2, Settings for experimental runs

Run#	Figure	TC #	delay	Ts	$P(z^{-1})$	$Q(z^{-1})$	$R(z^{-1})$	$T(z^{-1})$	Δ_b
1	3.16	2	5	10	1	0	1	1	0.05
2	3.17	3	5	10	1	0.5Δ	1	1	0.05
3	3.18	3	5	10	1	$\frac{0.95\Delta}{(1-.7z^{-1})}$	1	1	0.05
4	3.19	3	5	10	1	$\frac{0.95\Delta}{(1.5-.297z^{-1})}$	1	1	0.05
5	3.20	3	5	10	$2(1-.5z^{-1})$	$\frac{3\Delta}{(1-.7z^{-1})}$	1	1	0.05
6	3.21	3	5	10	(changed	on-line)	1	1	0.05
7	3.22	3	5	10	$3(1-.66z^{-1})$	$\frac{3\Delta}{(1-.7z^{-1})}$	1	1	0.05

(delay=model delay in sample period; Ts=sampling time in second)

Figure 3.16 shows a 5-step ahead minimum variance predictive control ($P(z^{-1})$=1, $Q(z^{-1})$=1, $R(z^{-1})$=1) on the process. After the open loop identification and 2 setpoint changes, the parameters converge to some good values and the controller is able to drive the output to the setpoint satisfactorily. However, the control action is too vigorous (which is common for minimum variance control). With the chosen sampling time ($\hat{d}$=5), RAC may not be able to compensate for the time delay even when 2 b parameters are used in the model. The fractional time delay mismatch causes instability because the controller poles continuously migrates in and out of the stability region. Furthermore, it results in a small value of go which makes the controller very active. This can be seen from Figure 3.16 that the controller shows signs of ringing from period 150 to 250. This is caused by a [illegible] damped negative controller pole. After 250 sampling periods, go converges to about 0.913. The control action becomes very vigorous and shows sign of instability. Notice that there is no instability in the process output because the controller cancels the unstable zero with an unstable controller pole. This should be avoided because the unstable mode will eventually show up in the process output when cancellation is no longer exact due to finite word length of the digital computer. $Q(z^{-1})$ weighting can help alleviate this problem.

Figure 3.17 shows that RAC controls the process successfully under time delay mismatch by using $Q(z^{-1})$ weighting. Thermo-couple #3 is used to measure the temperature which adds about 75 seconds time delay to the process (the time delay is underestimated by 2 to 3 sample periods). After 100 sample periods from start, the parameters converges to some meaningful values and the controller starts to perform fairly satisfactorily. Notice that go is maintained at a fairly large number compared with that in Figure 3.16. There are still small traces of ringing in the control output. This can be removed by specifying a larger value of λ. However, the response speed is expected to slow down when doing so. The performance of the servo response at period 250 is fairly acceptable. The response is slow and not oscillatory, with about 30% overshoot.

An inverse PI structure for $Q(z^{-1})$ weighting is used. The result is shown in Figure 3.18. The parameters for $Q(z^{-1})$ are arbitrarily chosen and they correspond to Kp=0.316 and Ki=0.074. The responses are faster than the previous run but more oscillatory. Figure 3.19 shows another setting for the $Q(z^{-1})$ weighting with the same inverse PI structure. They correspond to Kp=0.85 and Ki=0.0313. It can be seen that the responses are sluggish but overshoot and oscillation are reduced. In general, the PI structure provides more flexibility (and therefore better performance can be achieved) to the controller output shaping but it makes the controller harder to tune.

The response speed is increased when a lead filter $P(z^{-1})$ weighting is added. This is shown in Figure 3.20. The controller performs well after several setpoint changes. Although overshoot is high (over 50%), the oscillation dies down rapidly and the output tracks the setpoint quickly. Notice that the parameters converge much faster when $P(z^{-1})$ is added when compared with Figure 3.19.

In order to obtain a 'tuned' set of $P(z^{-1})$ and $Q(z^{-1})$ weightings, they are adjusted on-line as shown in Figure 3.21. $P=3(1-0.66z^{-1})$ and $Q=3\Delta/(1-0.7z^{-1})$ appear to be acceptable settings. These values are used for examining the regulatory control of RAC. Two step disturbances are introduced: at 300 sampling period, water inlet flow is increased to 45% and at 350 sampling period, water inlet flow is dropped back to 35%. The results are shown in Figure 3.22. The disturbance rejection is reasonably fast compared with the speed for servo control.

3.8 Conclusion

The modified RAC demonstrates it ability to control both minimum and NMP stable processes with model order or time delay mismatch. The robust parameter estimator is equipped with regressor normalization and a dead-zone for stopping the estimation. The size of the dead-zone can be easily chosen to be the fractional prediction error divided by 2. Offset free performance is achieved, which is a big improvement compared with the original RAC design. It is also extended to include $P(z^{-1})$, $Q(z^{-1})$, $R(z^{-1})$, and $T(z^{-1})$ weightings in rational transfer functions.

$Q(z^{-1})$ weighting is very important for the success of the control algorithm when time delay mismatch exists. It detunes the controller and it places the augmented process zeros so that the controller can handle some non-minimum phase systems or systems with poorly damped zeros. The choice of $Q(z^{-1})$, however, is not easy and it requires some a *priori* knowledge of the process. $Q(z^{-1}) = \lambda\Delta$ is found to be a successful choice where λ can be used as an on-line tuning parameter. It is recommended that when the time delay is not known, or when it is time varying, a large λ should be used at the start and slowly reduced to obtain faster control response. There is a lower bound to the value of λ for which the closed-loop will be unstable. The user is advised to use a more conservative λ whenever the knowledge of the time delay is limited. An inverse-PI structure of $Q(z^{-1})$ adds more flexibilities to the controller and the controller performance is seen to improve.

$P(z^{-1})$ weighting is useful for output shaping. It is recommended to choose $P(z^{-1})^{-1}$ to be a first order filter with unity gain and time constant faster than the process time constant. With proper choice of $P(z^{-1})$ and $Q(z^{-1})$ weightings, the RAC can handle open-loop stable processes satisfactorily.

The modified RAC algorithm is implemented as an 'Advanced Control Task' under the QNX operating system using C programming language. This implementation is useful for educational purposes. Details of the program can be found in Chapter 5 and the user reference to the program can be found in Appendix 2B.

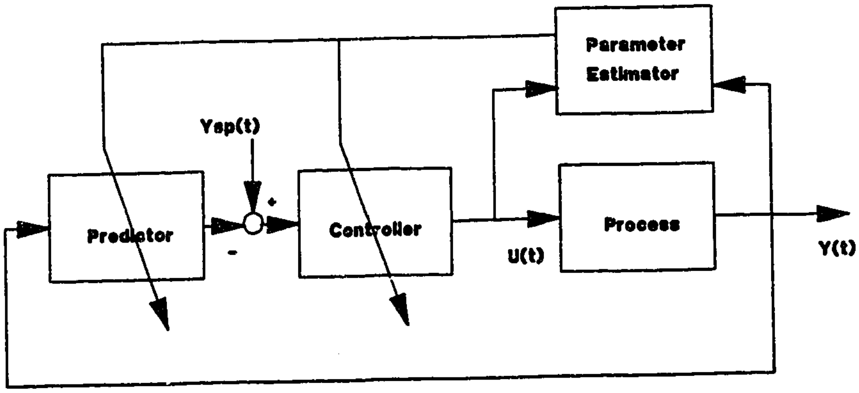


Figure 3.1. Block diagram of an adaptive controller

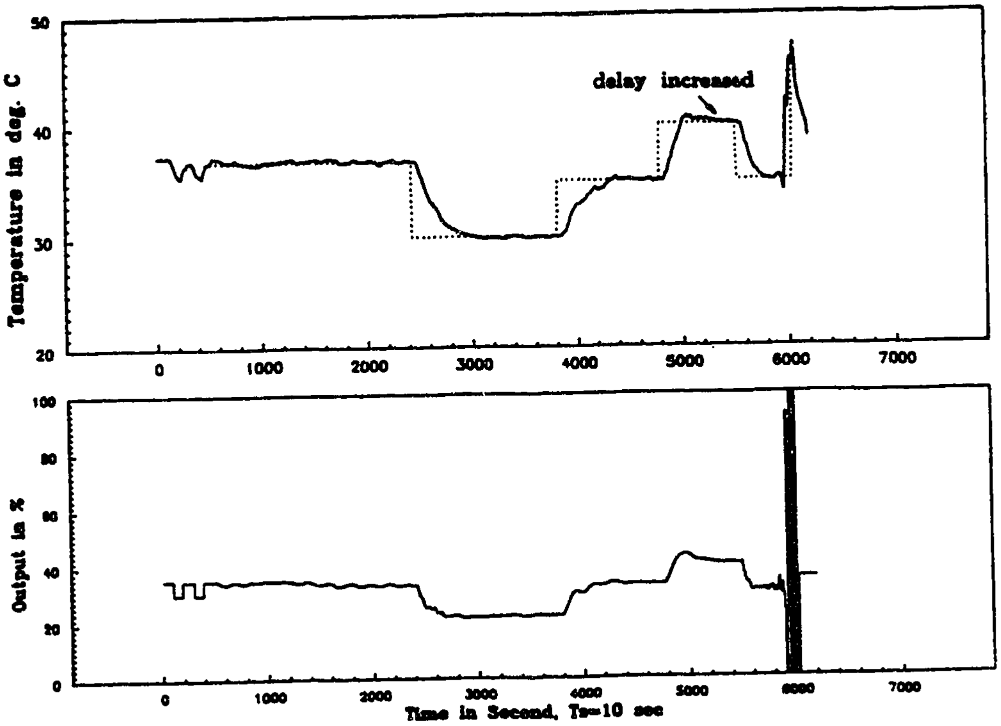


Figure 3.2. Performance of a d-step ahead self-tuning control on a varying time delay process

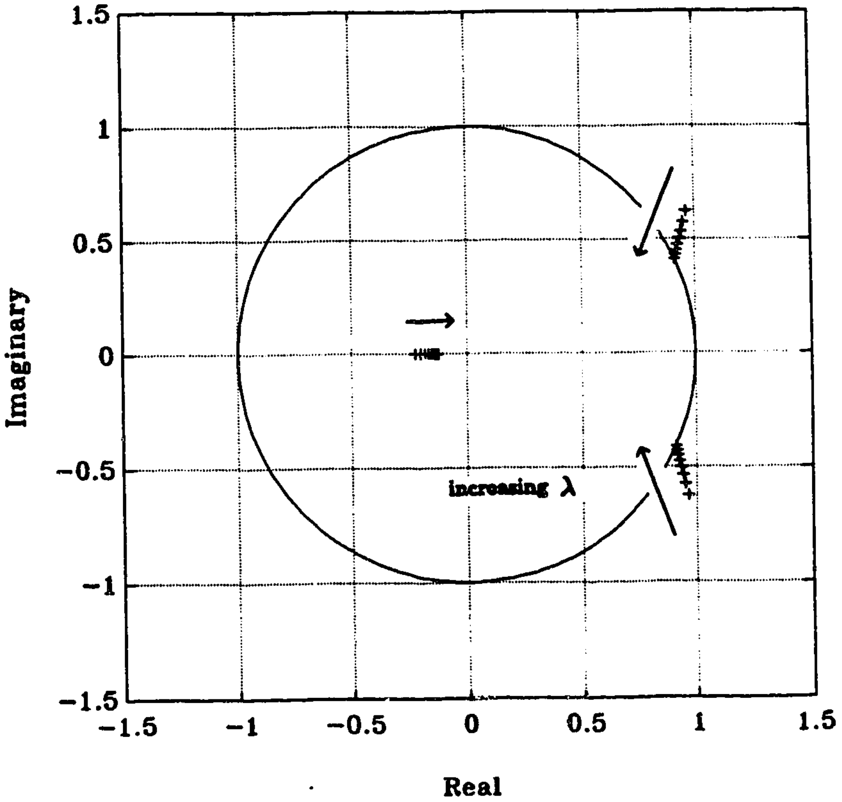


Figure 3.3. Root locus of the numerator of the augmented plant

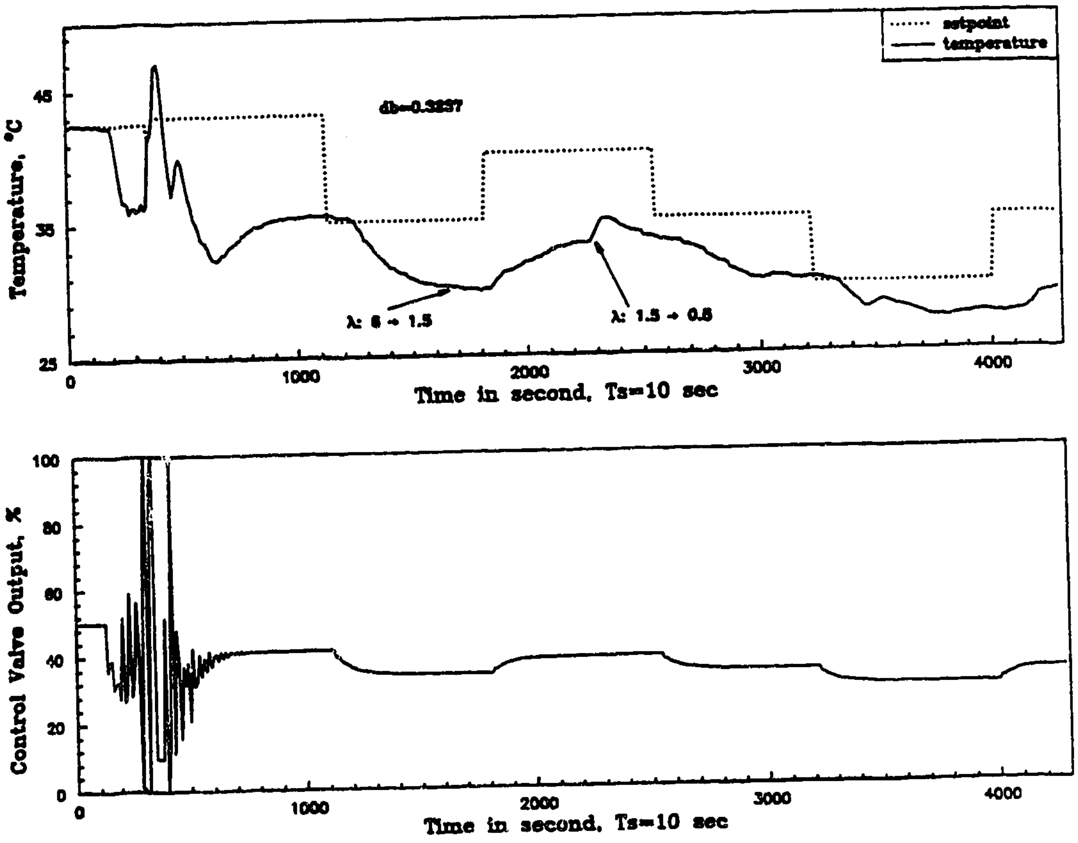


Figure 3.4. Output trajectories of RAC with dead-zone ON

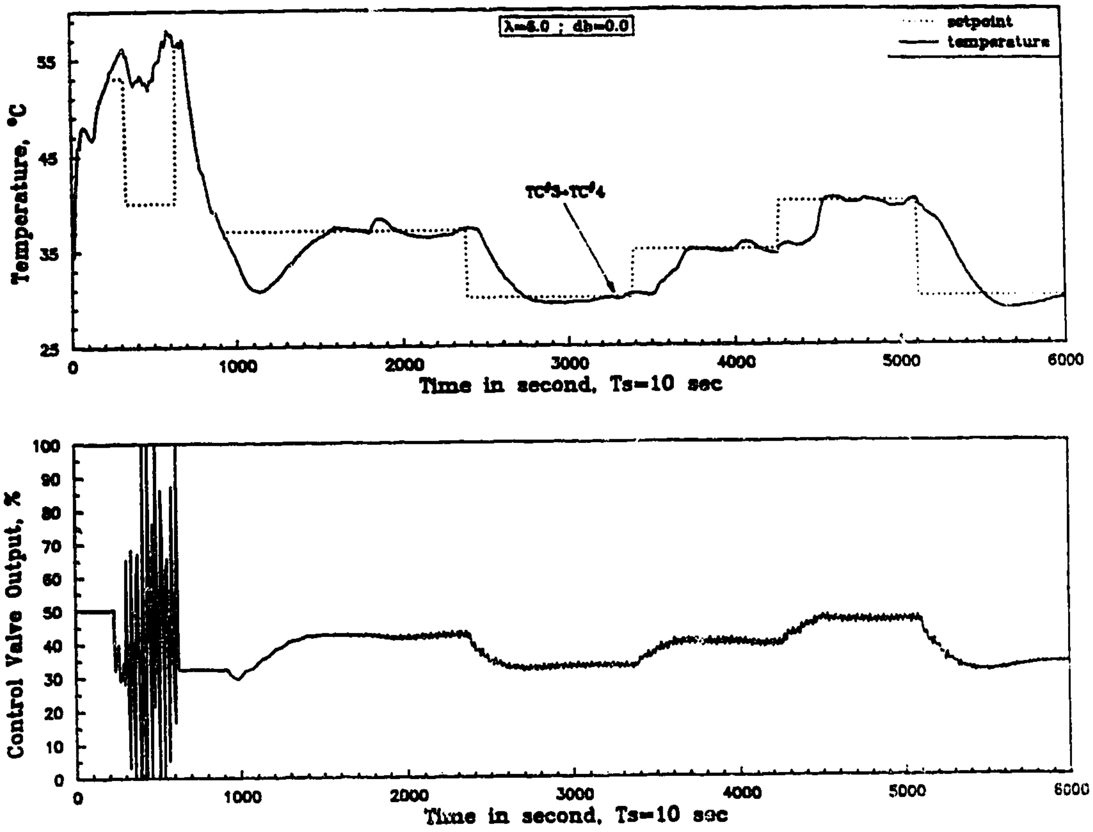


Figure 3.5. Output trajectories of RAC with dead-zone OFF

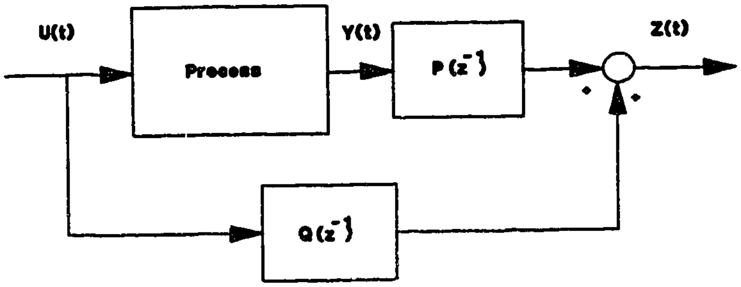


Figure 3.6. Block diagram of the augmented plant structure used in RAC

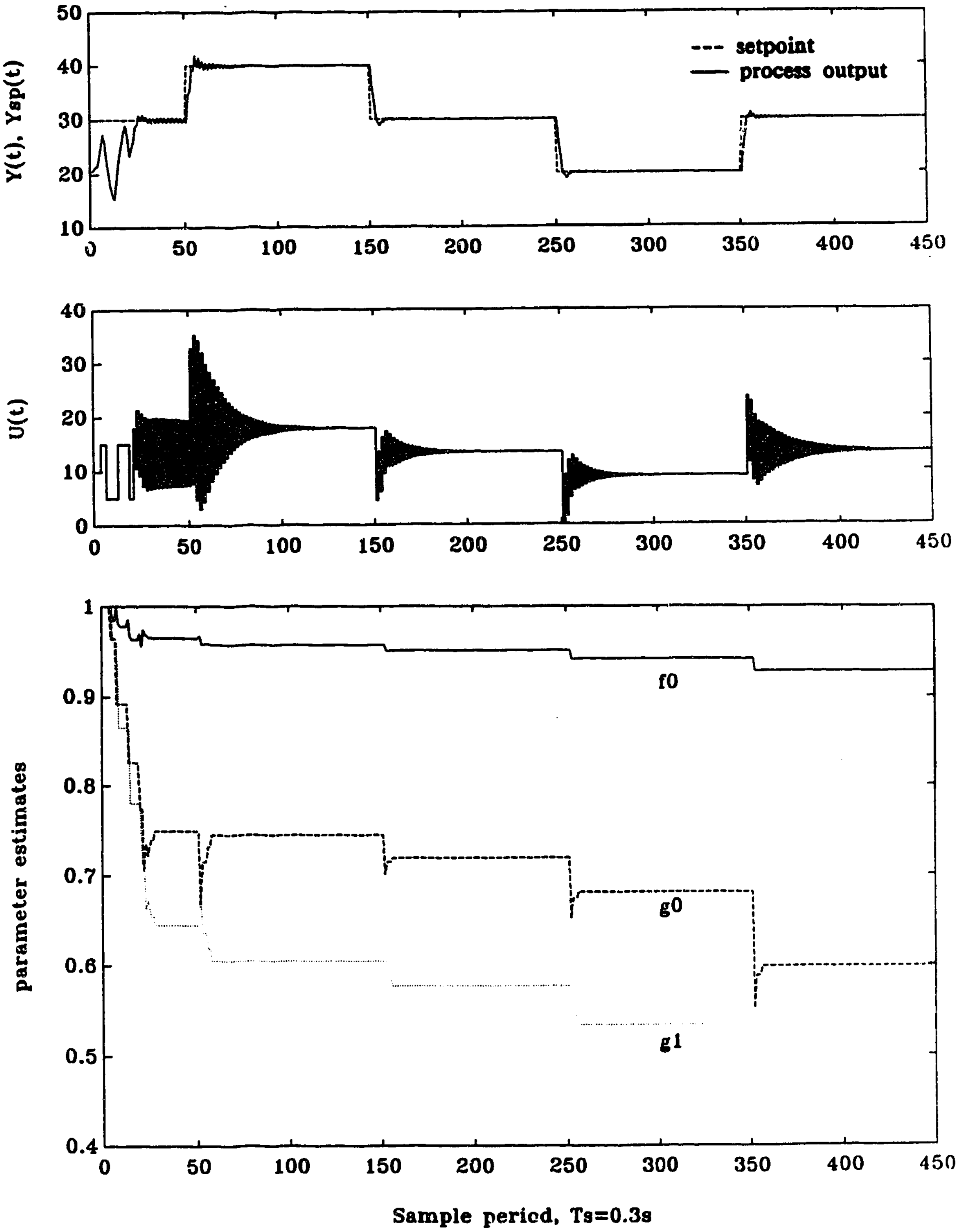


Figure 3.7. Simulation of RAC on 3rd order process with process order mismatch. (P=1, Q=0, R=1, T=1, Δb=0.0243)

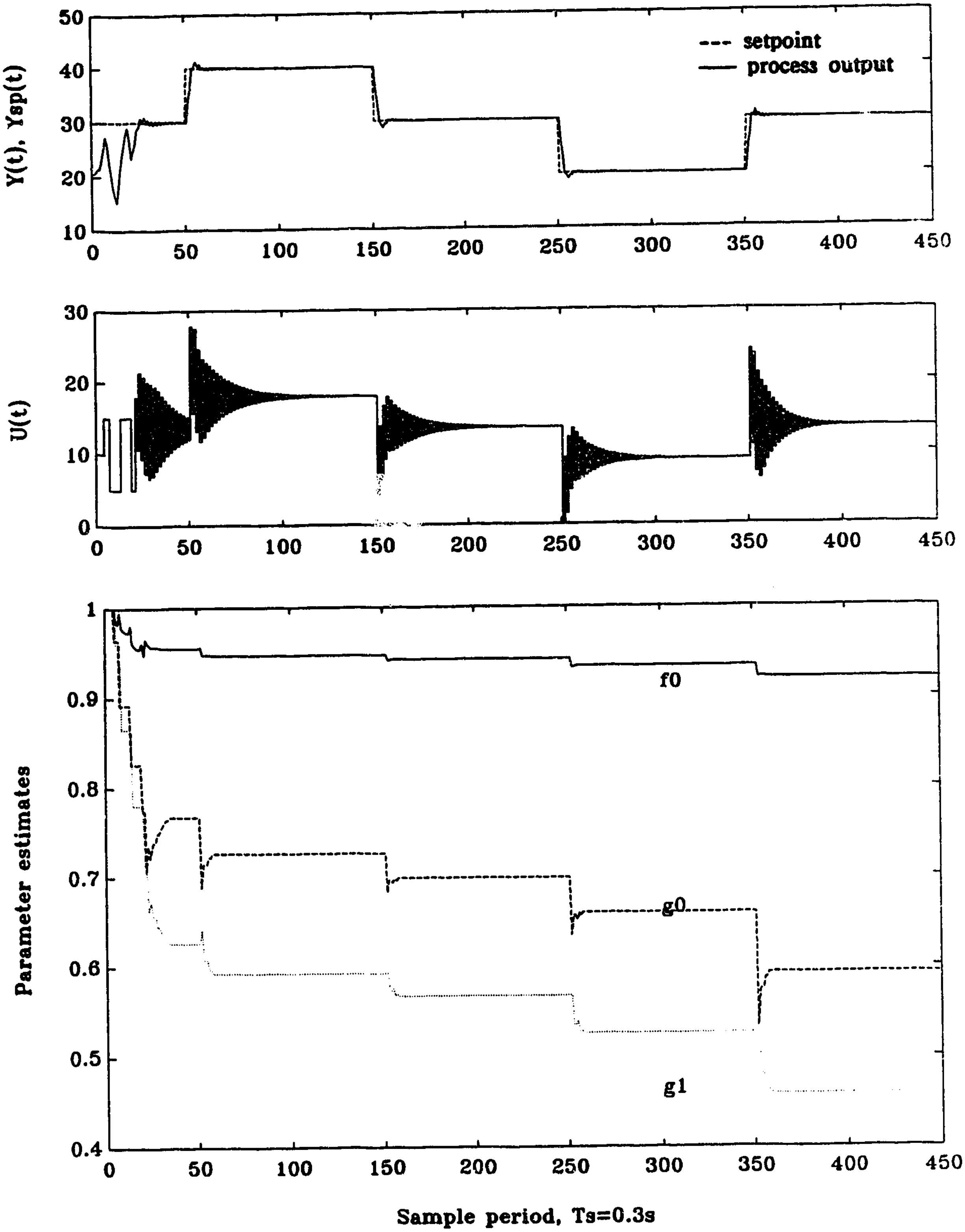


Figure 3.8. Simulation of RAC on 3rd order process with process order mismatch. (P=1, Q=0, R=1, T=1, Δ_b=0.01)

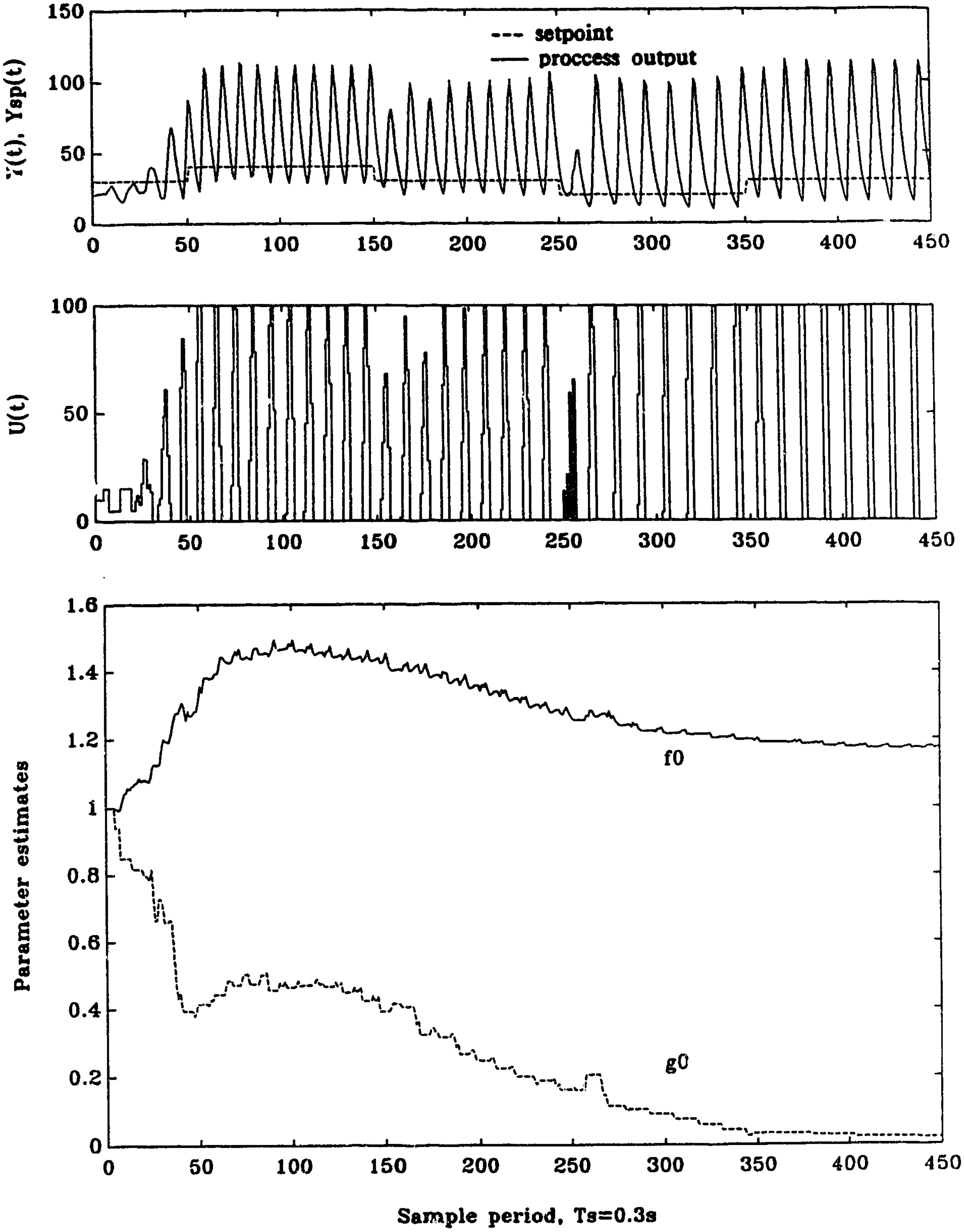


Figure 3.9, Simulation of RAC on 3rd order process with time delay mismatch. (P=1, Q=0, R=1, T=1, Δb=0.0243)

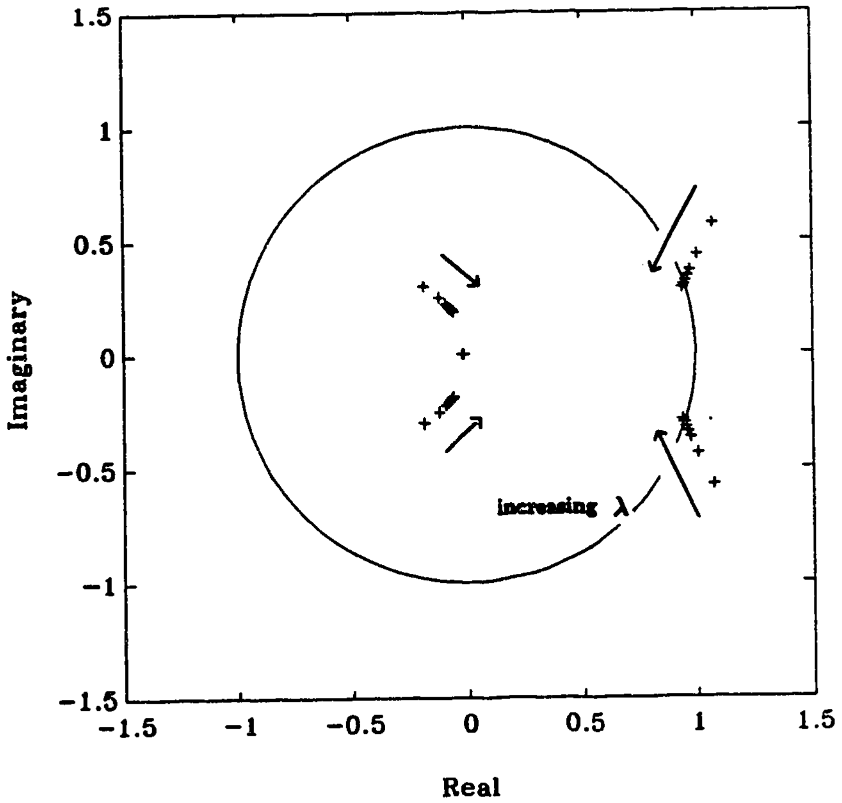


Figure 3.10, Root locus of augmented process zeros with increasing value of λ

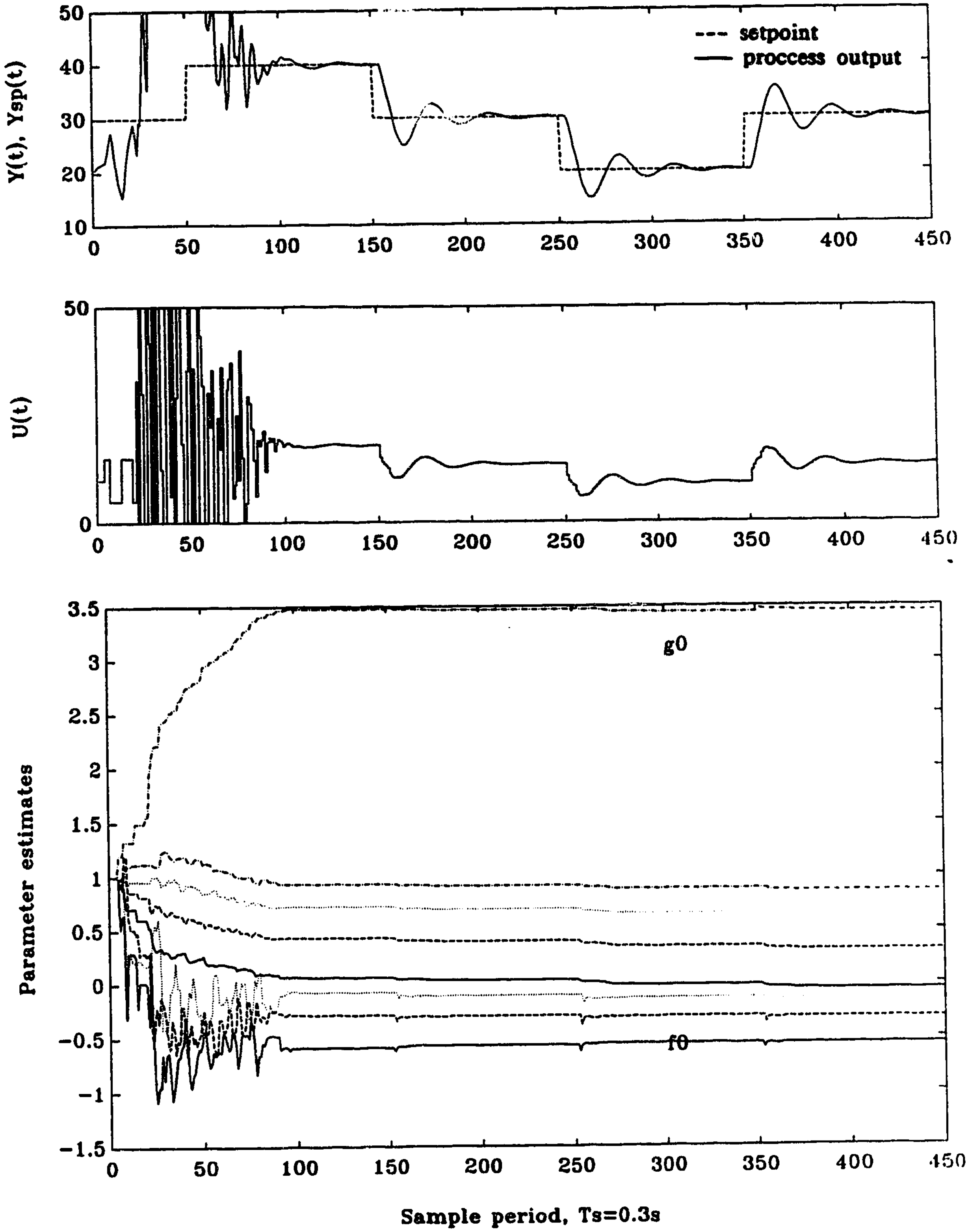


Figure 3.11, Simulation of RAC on 3rd order process with time delay mismatch. (P=1, Q=5Δ, R=1, T=1, Δ_b=0.0243)

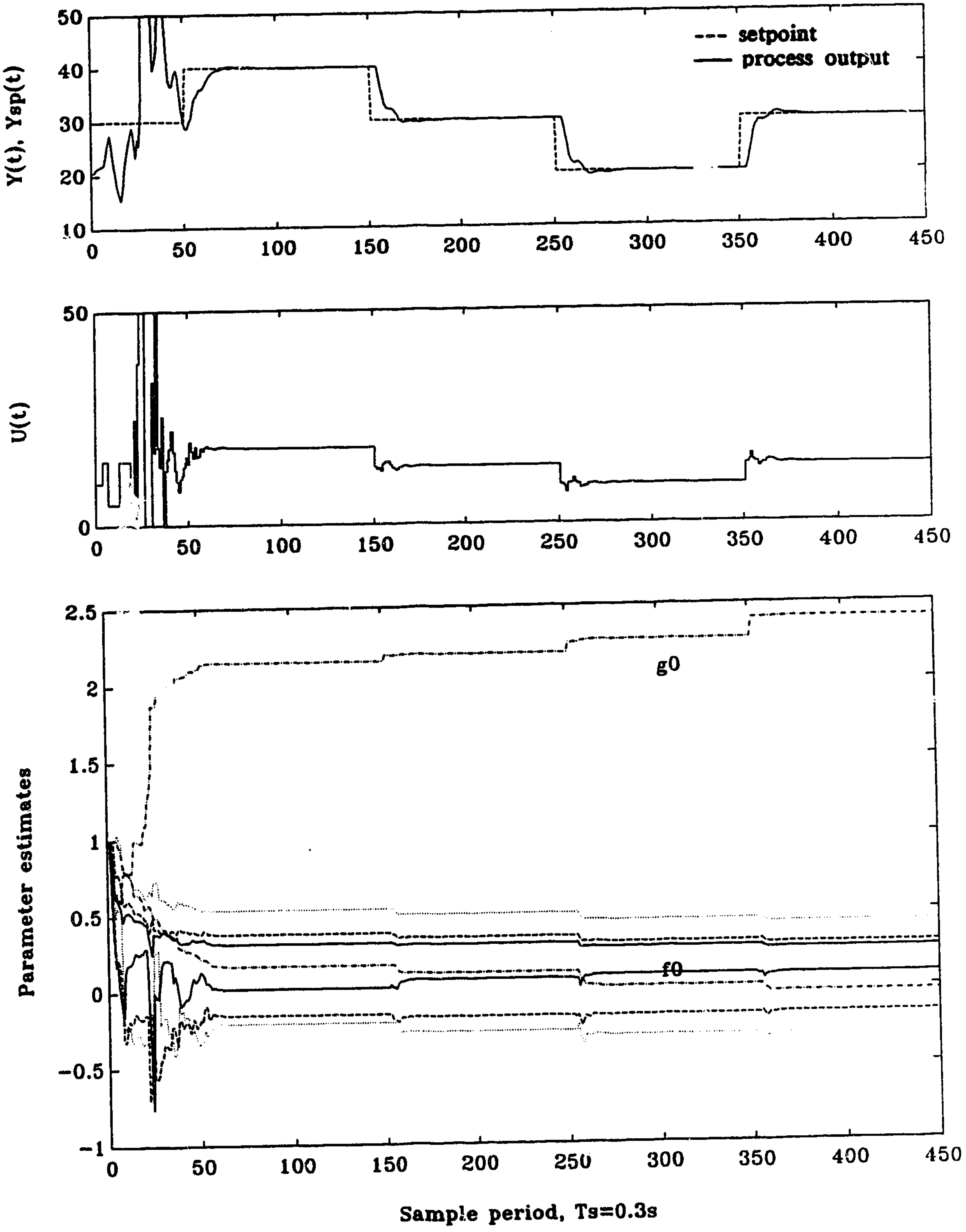


Figure 3.12. Simultaion of RAC on 3rd order process with time delay mismatch. ($P=1$, $Q=5\Delta/(1-0.7z^{-1})$, $R=1$, $T=1$, $\Delta b=0.0243$)

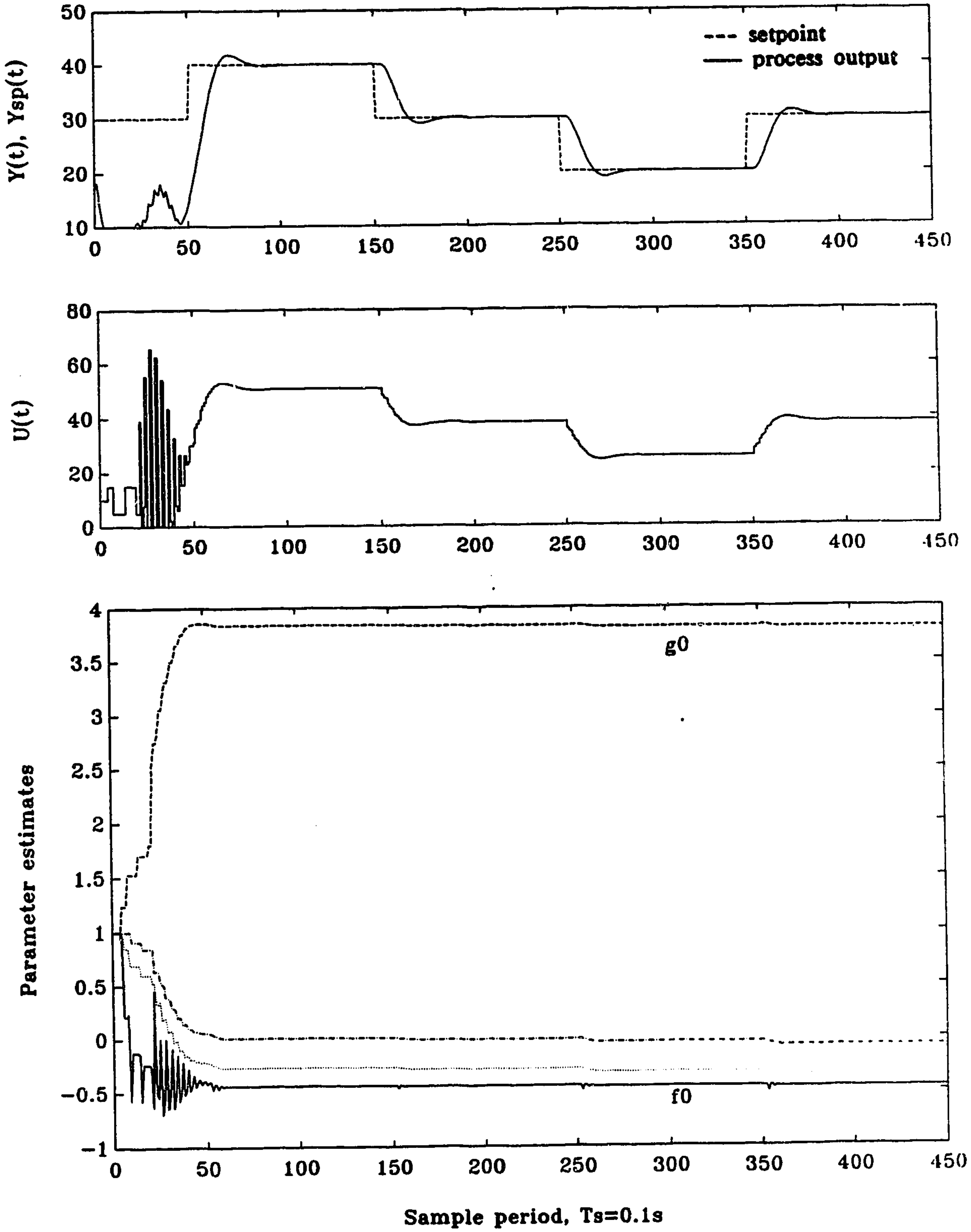


Figure 3.13. Simulation of RAC on a 3rd order non-minimum phase process with time delay and process order mismatch. (P=1, Q=5Δ, R=1, T=1, Δb=0.0243)

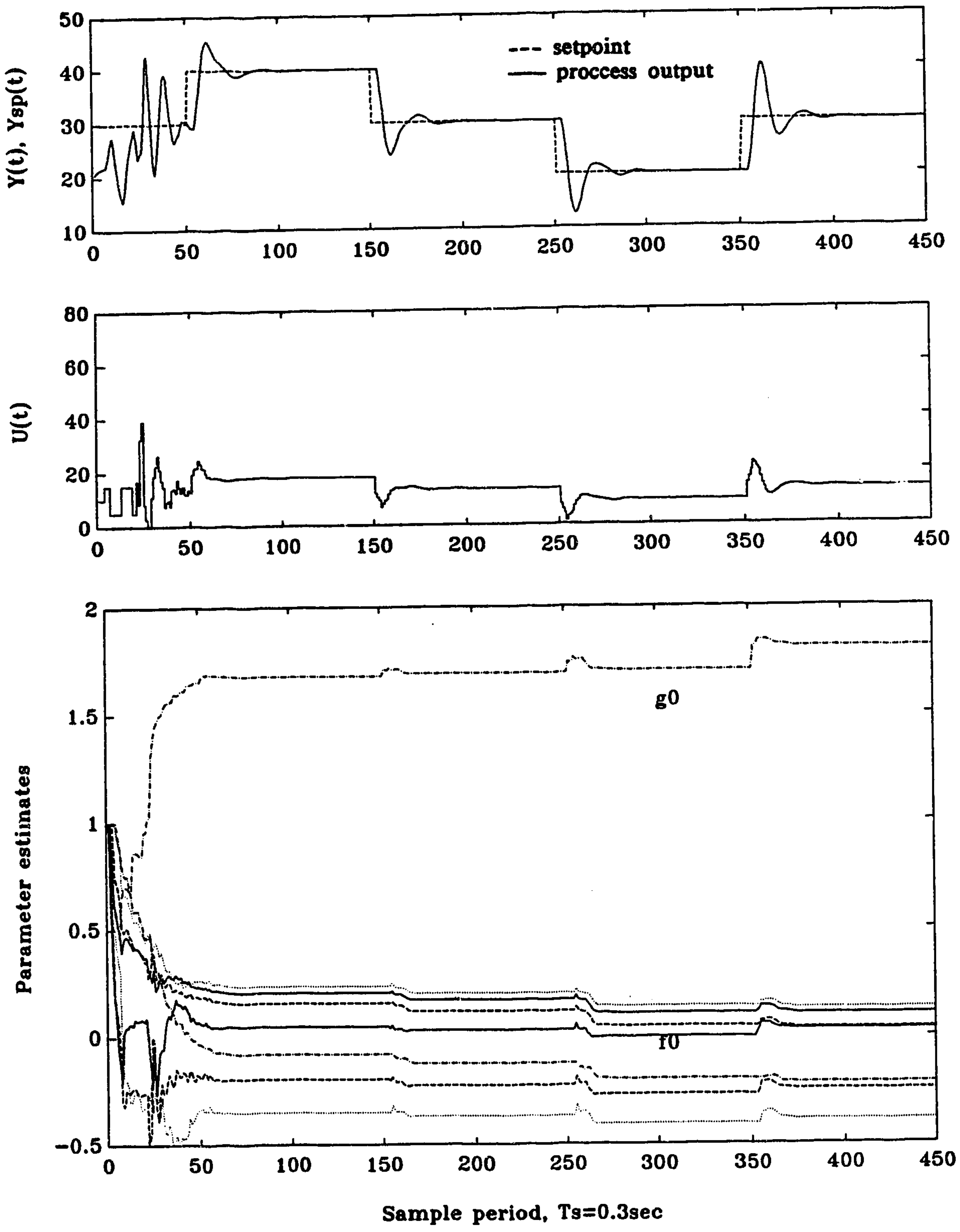


Figure 3.14, Simulation of RAC on a 3rd order process with time delay mismatch. (P=1, $Q=3\Delta/(1-0.7z^{-1})$, R=1, T=1, $\Delta b=0.0243$)

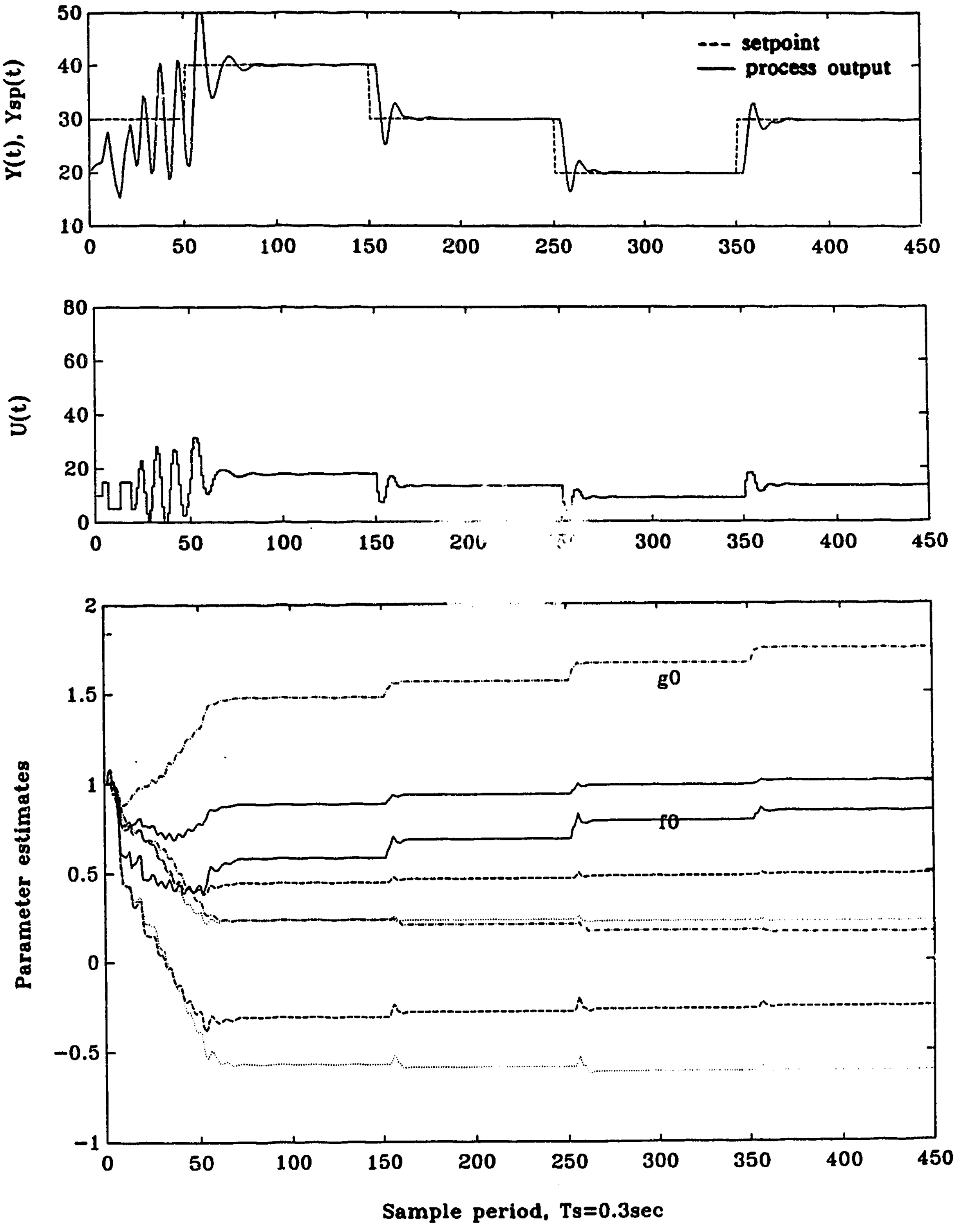


Figure 3.15, Simulation of RAC on a 3rd order process with time delay mismatch. ($P=5(1-0.8z^{-1})$, $Q=3\Delta/(1-0.7z^{-1})$, $R=1$, $T=1$, $\Delta b=0.0243$)

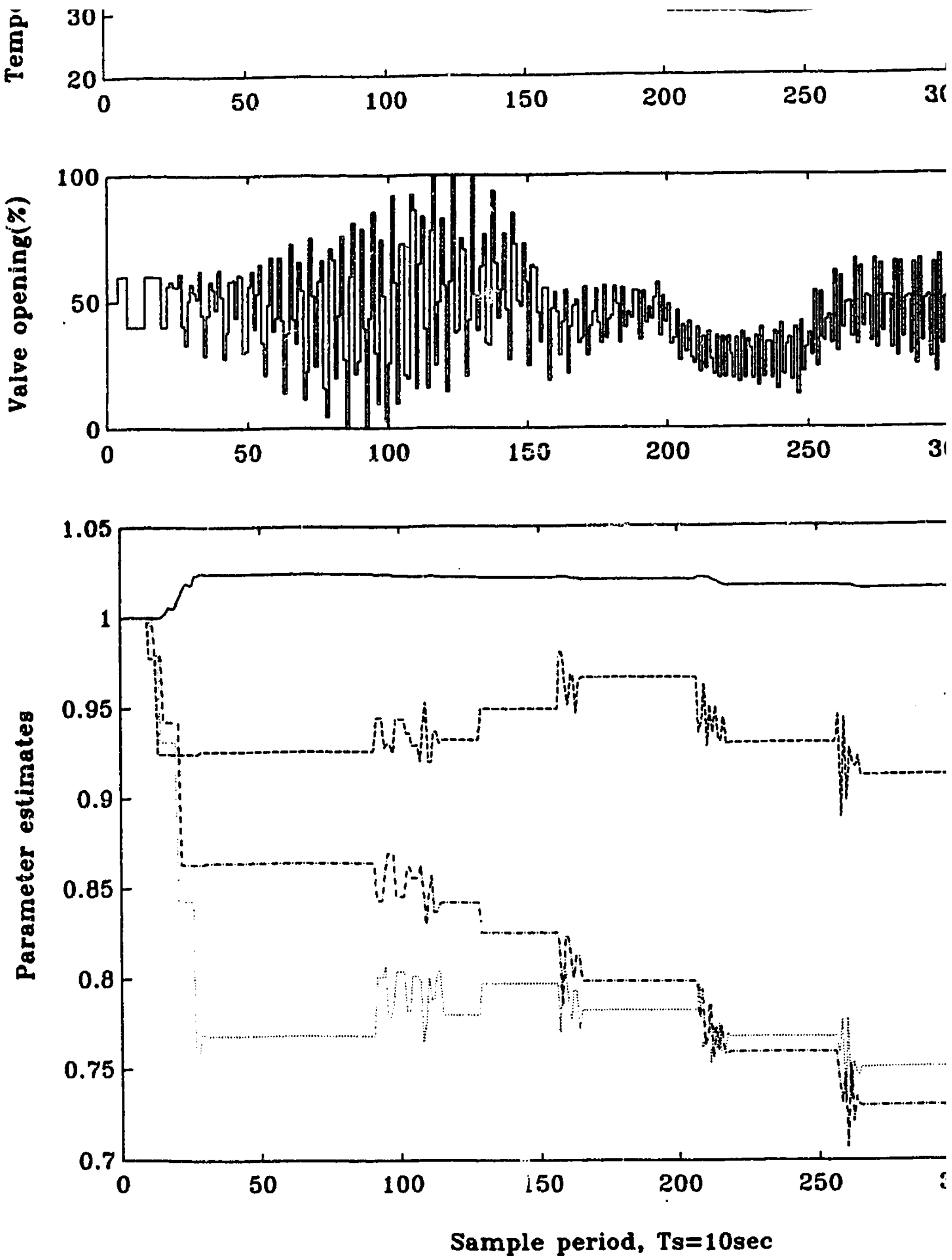


Figure 3.16, RAC servo control on CSTH with no time delay mis
Q=0, R=1, T=1, Δb=0.05, TC#2, $\hat{d}$=5)

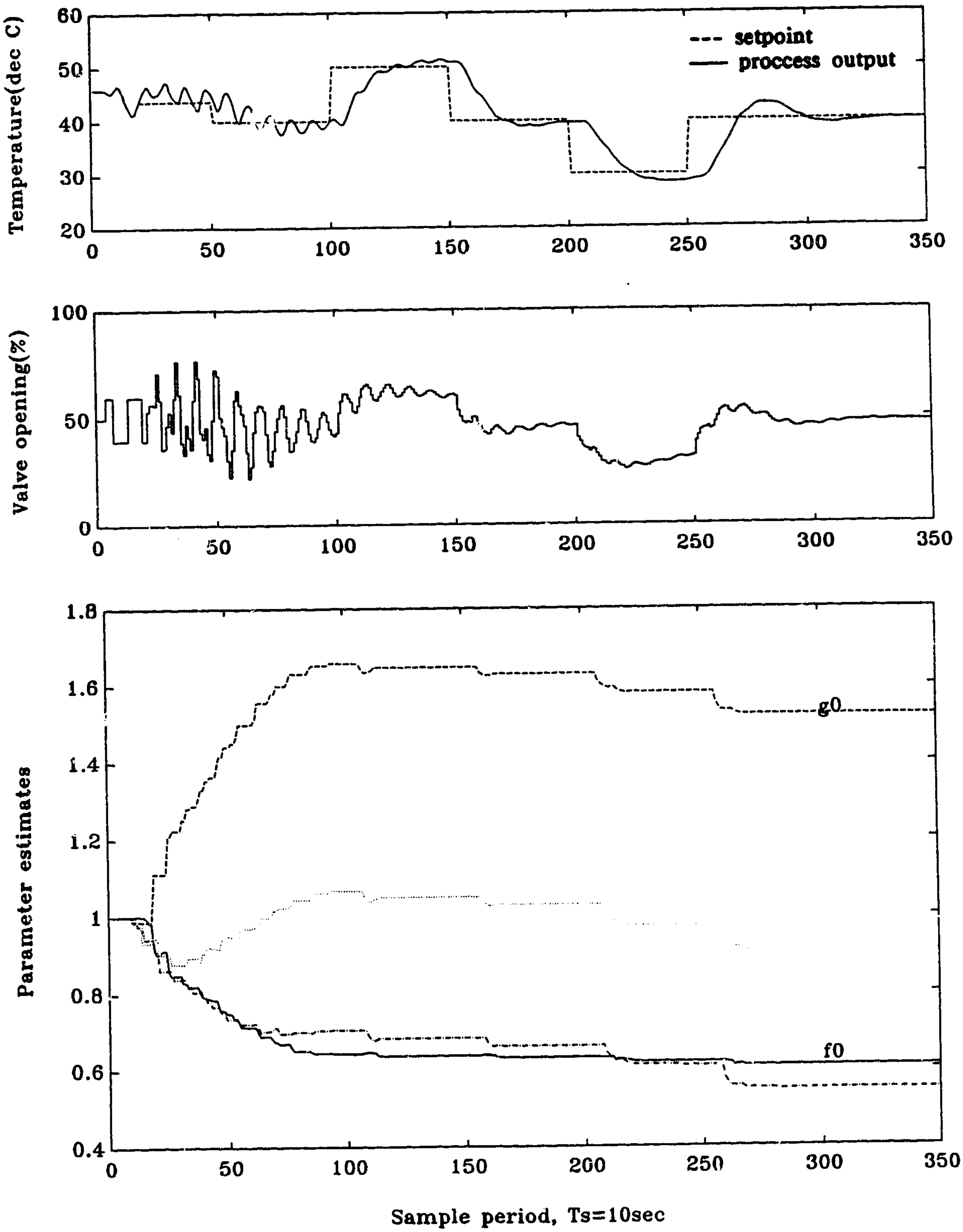


Figure 3.17, RAC servo control on CSTH with time delay mismatch. (P=1, Q=0.5Δ, R=1, T=1, Δb=0.05, TC#3, $\hat{d}$=5)

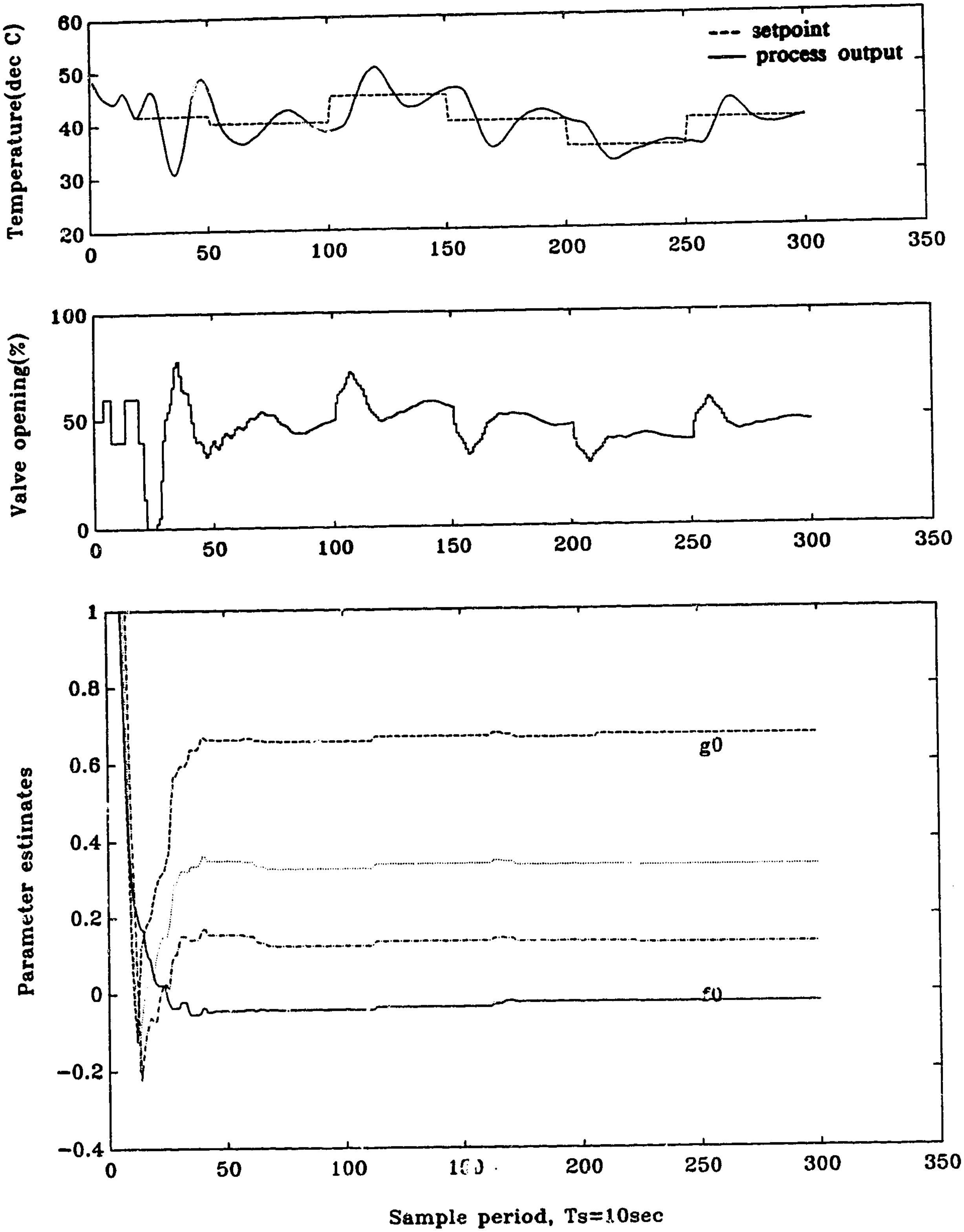


Figure 3.18, RAC servo control on CSTH with time delay mismatch. (P=1, $Q=0.95\Delta/(1-0.7z^{-1})$, R=1, T=1, $\Delta b=0.05$, TC#3, $\hat{d}=5$)

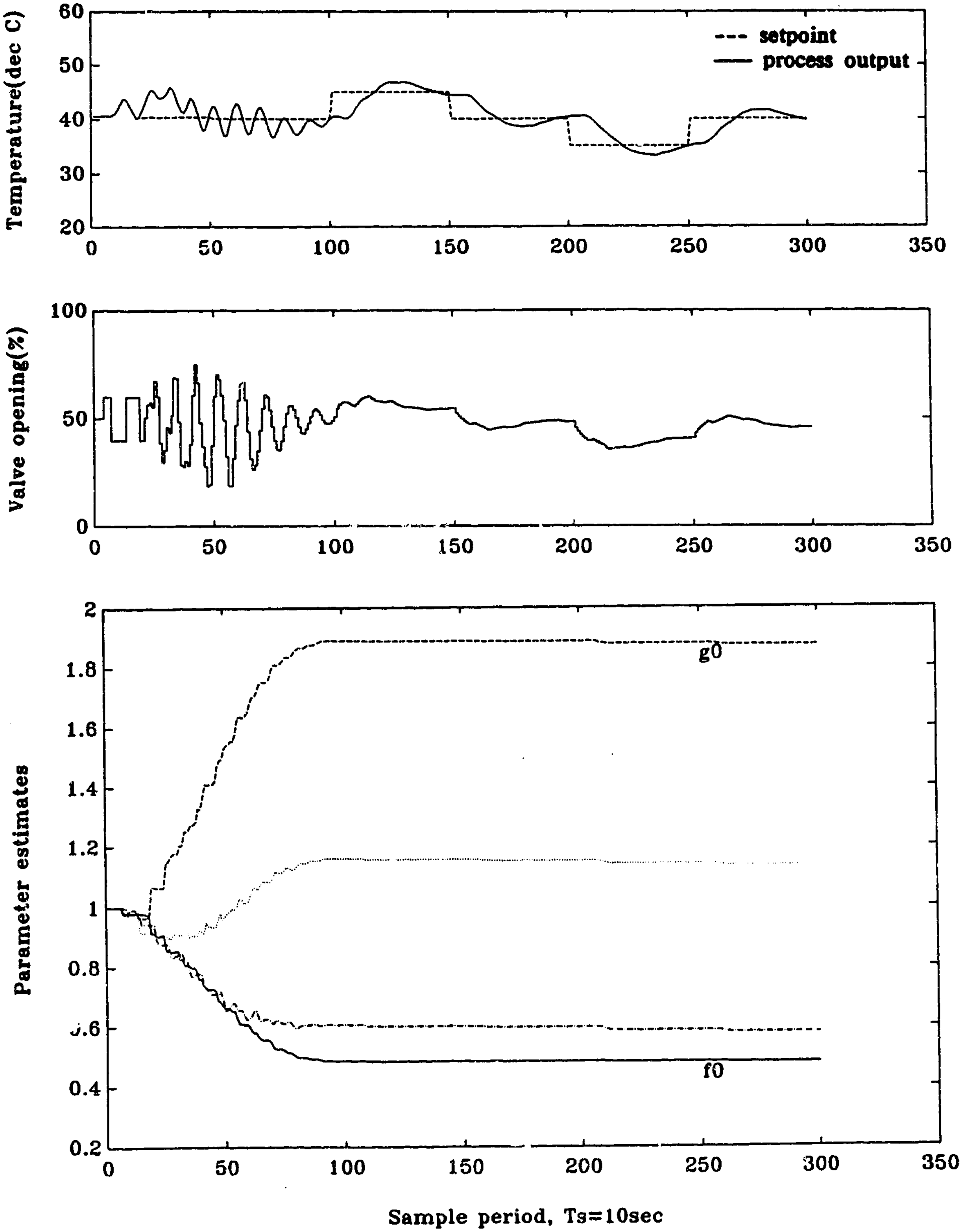


Figure 3.19, RAC servo control on CSTH with time delay mismatch. (P=1, $Q=0.95\Delta/(1.5-0.297z^{-1})$, R=1, T=1, $\Delta b=0.05$, TC#3, $\hat{d}=5$)

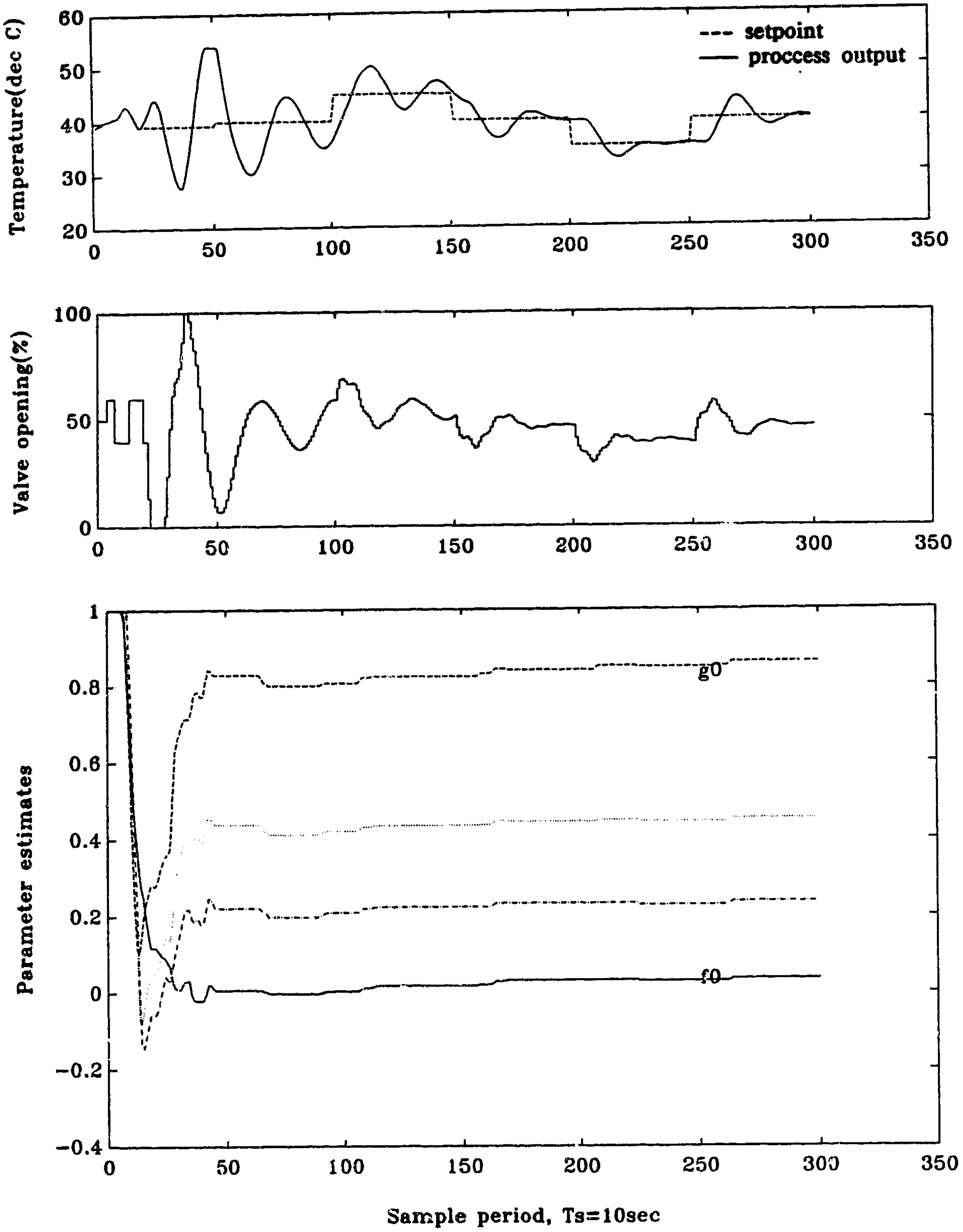


Figure 3.20, RAC servo control on CSTH with time delay mismatch. ($P=2(1-0.5z^{-1})$, $Q=0.95\Delta/(1.5-0.297z^{-1})$, $R=1$, $T=1$, $\Delta b=0.05$, TC#3, $\hat{d}=5$)

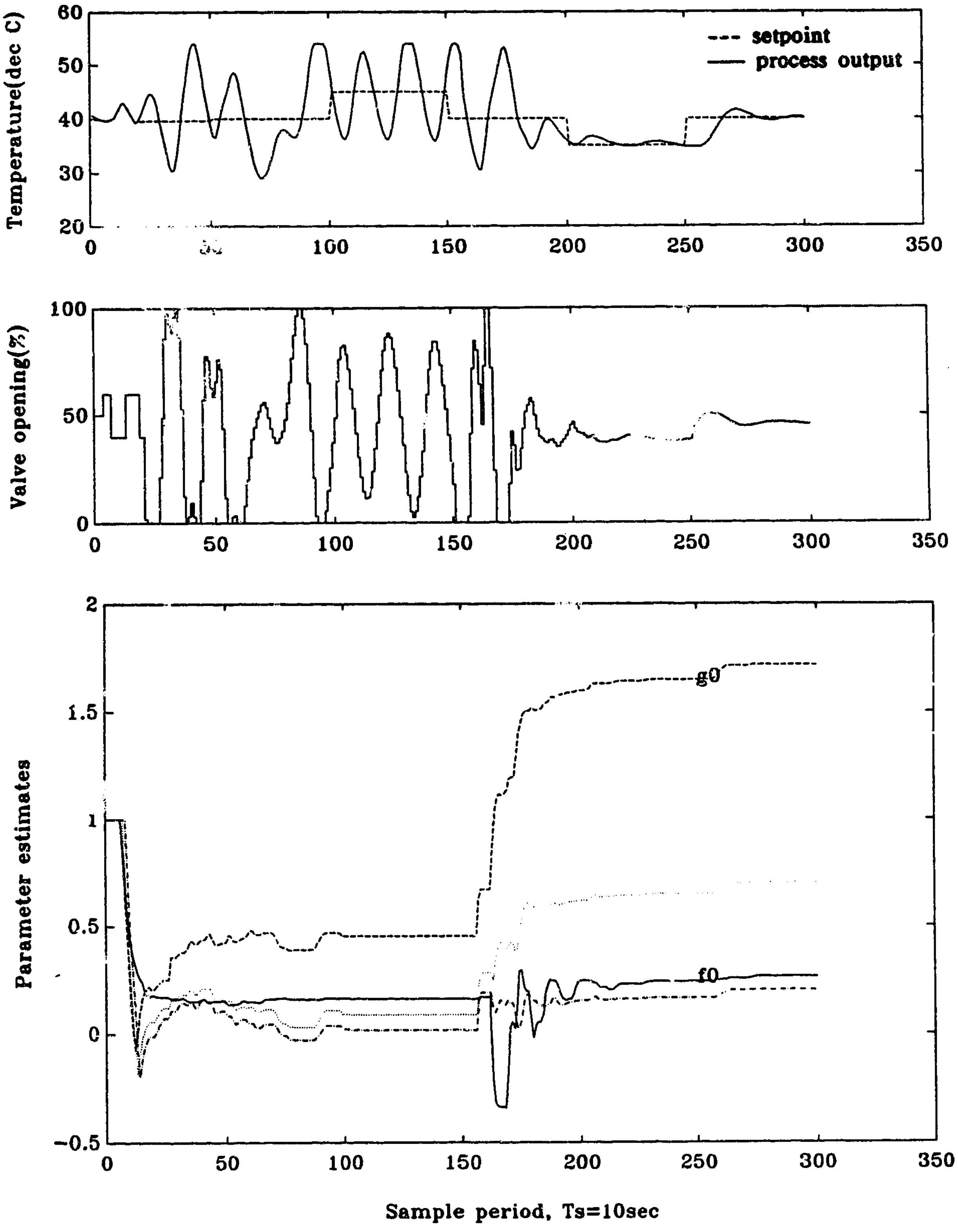


Figure 3.21, RAC servo control on CSTH with time delay mismatch. (P and Q are tuned on-line, R=1, T=1, TC#3, $\hat{d}$=5)

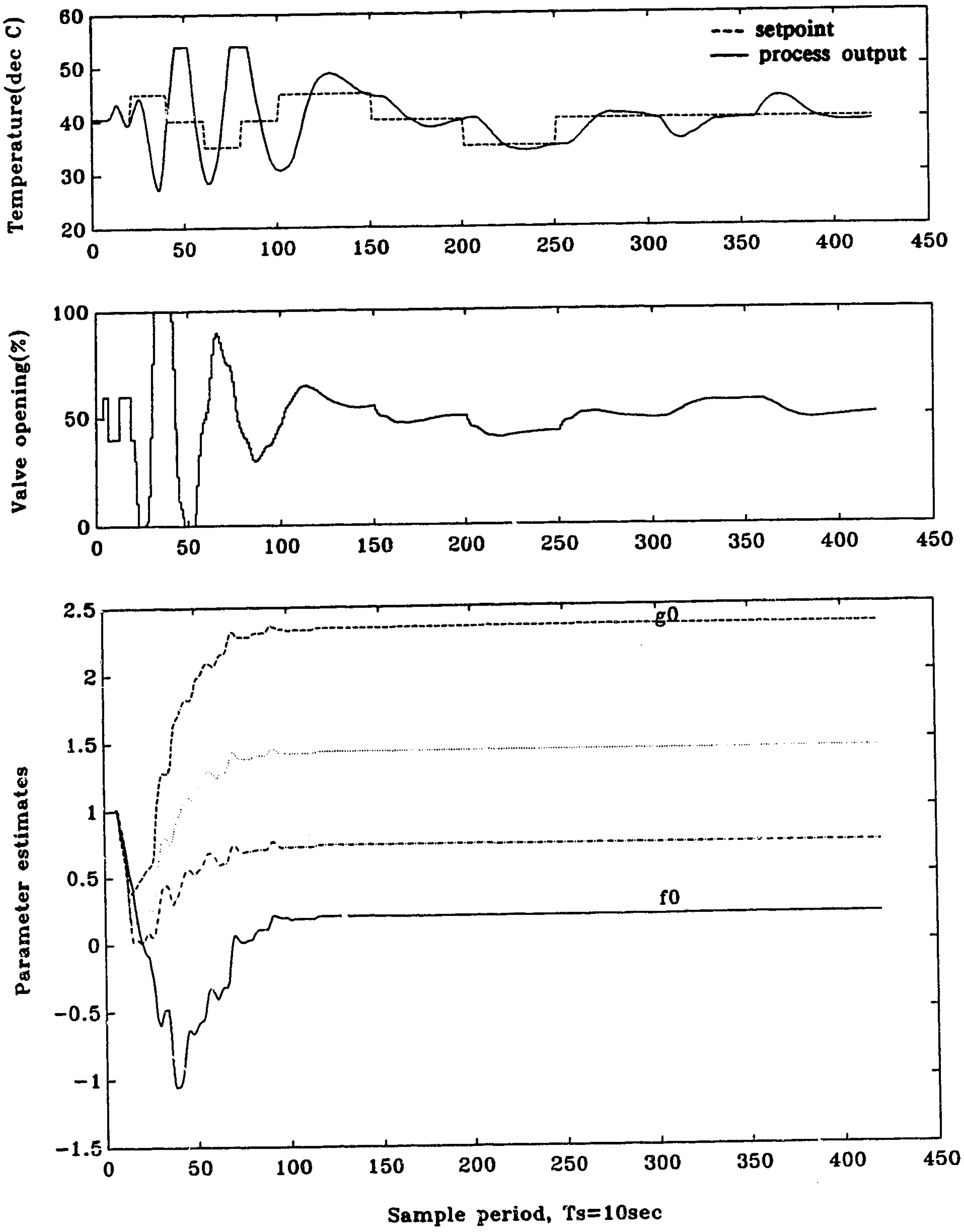


Figure 3.22. RAC servo and regulatory control on CSTH with time delay mismatch. ($P=3(1-0.66z^{-1})$, $Q=3\Delta/(1-0.7z^{-1})$, $R=1$, $T=1$, $\Delta b=0.05$, TC#3, $\hat{d}=5$)

4 Multistep-Adaptive-Predictive-Control (MAPC)

4.1 Introduction

Inverse-model adaptive schemes developed using a single point optimization, such as the GMV (Generalized Minimum Variance control) by Clarke (1984) and RAC (Robust Adaptive Control) by Cluett (1987), are sensitive to *a priori* choice of the time delay. If the time delay is underestimated, or if the process time delay increases at some point in time, this class of controllers can become unstable. The usual strategy to get around this problem is to detune the controller by the introduction of controller weightings. For example, Q weighting is absolutely essential to the success of the RAC as explained in Chapter 3. This method guarantees stable control by trading off response speed. However, the major drawback is that a successful choice of Q weighting is not easy and depends greatly on some *a priori* knowledge of the process. Since Q is non-adaptive (Minter, 1988), a user selected value of Q may not be able to stabilize the process if the process is highly time-varying. This prompts one to consider another class of controllers that are based on the minimization of a multi-stage cost function and can handle time-varying delays relatively easily.

Long range predictive control (LRPC) has recently received a lot of attention. This is because this class of control strategy has been found to be more robust to model-plant-mismatch (MPM), especially in the mismatch between the process and the model time delay, and can easily handle non-minimum-phase (NMP) processes. Although the formulation of the LRPC varies from one design to another, they are all based on the idea that control action is determined based on a long term prediction of the process output. At every sample period, a forecast of the process over a long range time horizon is generated based on a model of the process that is adjusted on-line with a recursive parameter estimator. The 'best' control path to bring the predicted trajectory to the setpoint is then determined by a series of control moves calculated such that a quadratic cost function is minimized. The control action is implemented in a receding horizon fashion, ie. only the first element of the predicted control path is implemented. The whole procedure is repeated at the next sample period.

Among the many LRPC strategies, EHAC (Extended Horizon Adaptive Control) by Ydstie et al. (1985), DMC (Dynamic Matrix Control) by Cutler and Ramaker (1980), EPSAC (Extended Prediction Self-Adaptive Control) by De Keyser et al. (1979), and GPC (Generalized Predictive Control) by Clarke et al. (1986) have received special attention in industry and in fact a number of them have been successfully implemented. In particular, DMC is being used as the main control platform by Shell at its oil refineries and other petrochemical units.

The Multistep-Adaptive-Predictive-Controller (MAPC) developed at the University of Alberta by Sripada (1988) and modified by Foley (1988) falls into the same category as LRPC. The most prominent feature of MAPC is that the process model is formulated using a state-space realization and the well established Kalman Filter (Kalman, 1960) is used for state estimation and prediction (called the Modified Kalman Filter Predictor, MKFP). Foley (1988) has compared MAPC with GPC and claimed that they are asymptotically equal provided that the disturbance model in MAPC is chosen to be the same as that in GPC.

In this study, the MAPC is successfully implemented as an Advanced Control Task in C programming language running under MULTICON and the QNX operating system (see Chapter 5). This chapter outlines the structure and the features of the MKFP and the MAPC. Special attention is paid on the performance of the MKFP in the presence of time delay mismatch. The MAPC is experimentally evaluated on the pilot scale Continuous Stirred Tank Heater (CSTH, see Chapter 5). Since LRPC based controllers are known to be more robust to model-plant delay mismatch, MAPC is especially tested under this condition.

4.2 The Modified Kalman Filter Predictor (MKFP)

4.2.1 Basic formulation of the Kalman Filter Predictor

Time delay compensation, which was pioneered by Smith (1957), plays an important role in process control especially when the time delay is significant. Chapter 2 discussed the classical predictive control using the

Smith predictor and the PID controller. However, the Smith predictor was originally developed to tackle deterministic process only. When stochastic noise or non-stationary disturbance terms are present, it was found that the predictive control can be improved by adding a noise filter $Gf(z^{-1})$ as shown in Figure 2.3. This is similar to the configuration of the Internal Model Control (IMC) by Garcia and Morari (1982). Rivera et al. (1986) analyzed IMC and suggested ways of choosing the filter (see Chapter 2). Bialkowski (1978) developed an optimal k-step-ahead predictor based on the Kalman Filter, called the Kalman Filter Predictor. The resulting predictor has the same structure as the Smith predictor shown in Figure 2.2, but the noise filter $Gf(z^{-1})$ is chosen optimally with respect to the a *priori* knowledge of the stochastic disturbances and the noise characteristics.

Consider the discrete state-space representation of a SISO system

$$X_1(t+1) = \Phi_1 X_1(t) + \Lambda_1 U(t) + \Xi_1 \eta 1(t) \tag{4.2.1}$$

$$Y(t) = H_1 X(t) + \eta 2(t) \tag{4.2.2}$$

where for $\hat{d}=k-1$

$$X_1(t) = [x_1(t)\ x_2(t)\ \dots\ x_n(t)\ x_{n+1}(t)\ \dots\ x_{n+\hat{d}}(t)]^T$$

$$\Phi_1 = \begin{bmatrix} 0 & 0 & \dots & 0 & -a_n & \dots & 0 & 0 \\ 1 & 0 & \dots & 0 & -a_{n-1} & \dots & 0 & 0 \\ \cdot & & & & & & & \\ 0 & & & 1 & -a_1 & \dots & 0 & 0 \\ \cdot & & & & 1 & & \cdot & \\ \cdot & & & & & & \cdot & \\ \cdot & & & & & & \cdot & \\ 0 & 0 & & \dots & & & 1 & 0 \end{bmatrix}_{(n+\hat{d})x(n+\hat{d})}$$

$$\Lambda_1^t = [b_n\ b_{n-1}\ \dots\ b_1\ 0\ \dots\ 0]_{1x(n+\hat{d})}$$

$$H_1 = [0\ \dots\ 0\ 1]_{1x(n+\hat{d})}$$

$$\Xi_1 = [\xi_1\ \xi_2\ \dots\ \xi_n\ 0\ \dots\ 0]_{1x(n+\hat{d})}$$

Notice that the state variables (except those associated with the delayed states) do not carry any physical meaning in this state space observable canonical form. This makes the interpretation of $\eta_1(t)$ and $\eta_2(t)$ very difficult. Previous authors interpreted $\eta_1(t)$ as process noise or state excitation. Equations (4.2.1) and (4.2.2) can be expressed in an input-output form by doing forward substitution of the states to give:

$$Y(t) = \frac{B(z^{-1})}{A(z^{-1})} z^{-\hat{d}} U(t-1) + \frac{C(z^{-1})}{A(z^{-1})} z^{-\hat{d}-1} \eta_1(t) + \eta_2(t) \tag{4.2.3}$$

where

$$A(z^{-1}) = 1 + a_1 z^{-1} + \dots + a_n z^{-n}$$

$$B(z^{-1}) = b_1 + b_2 z^{-1} + \dots + b_n z^{-n+1}$$

$$C(z^{-1}) = \xi_n z^{-1} + \xi_{n-1} z^{-2} + \dots + \xi_1 z^{-n}$$

Equation (4.2.3) is an ARMAX model with $\eta_1(t)$ as a zero mean stochastic disturbance which passes through the dynamic structure $C(z^{-1})z^{-\hat{d}-1}/A(z^{-1})$ to generate the deterministic disturbance, and $\eta_2(t)$ as a zero mean measurement noise.

The optimal state estimates $\hat{X}_1(t)$ at each sample instant can be obtained by a Kalman Filter. The standard Kalman Filter algorithm is a one step algorithm and its variations can be found in Aström and Wittenmark (1984). A two-step formulation of the time-varying Kalman Filter is used here (Franklin and Powell, 1980). It consists of the following steps:

(1) Gain Calculation:

$$L(t) = M(t)H^t(HM(t)H^t + R_2)^{-1} \tag{4.2.4}$$

(2) Measurement Update:

(a) *A Posteriori* State update

$$\hat{X}(t) = \bar{X}(t) + L(t)(Y(t) - H\bar{X}(t)) \tag{4.2.5}$$

(b) *A Posteriori* Covariance update

$$P(t) = M(t) - L(t)HM(t) \tag{4.2.6}$$

(3) Time Update:

(a) *A Priori* State update

$$\overline{X}(t+1) = \Phi\hat{X}(t) + \Lambda U(t) \qquad (4.2.7)$$

(b) *A Priori* Covariance update

$$M(t+1) = \Phi P(t)\Phi^t + \Xi R_1 \Xi^t \qquad (4.2.8)$$

where R_1 and R_2 are variances of $\eta_1(t)$ and $\eta_2(t)$ respectively, ie.

$E\{\eta_1(t)\eta_1(t)^t\} = R_1,$

$E\{\eta_2(t)\eta_2(t)^t\} = R_2,$

and

$E\{\eta_1(t)\eta_2(t)^t\} = 0.$

Notice that in Equations (4.2.4), (4.2.6), and (4.2.8), the Kalman gain update L(t), the a *posteriori* covariance update P(t), and the a *priori* covariance update do not depend on the process measurement. Therefore, the Kalman gain does not need to be calculated on-line. Furthermore, since L(t), P(t), and M(t) will finally converge to their steady state values, rather than calculating the Kalman gain at every time step, the steady state Kalman Filter can be used instead (Pappas et al, 1980):

$$M - \Phi M\Phi^t + \Phi LHM\Phi^t - R_1 = 0 \qquad (4.2.9)$$

$$L = MH^t(HMH^t + R_2)^{-1} \qquad (4.2.10)$$

where M and L are the steady state values of M(t) and L(t) respectively. Equation (4.2.9) is known as the Algebraic Riccati Equation (ARE).

The iterative Kalman Filter algorithm defined in Equations (4.2.4) to (4.2.8) can therefore be reduced from an iterative problem to an algebraic problem by solving the ARE just once. This method saves a lot of real-time computing power because the ARE can be solved off-line. Since the Kalman Filter will be extended to an adaptive version later, the iterative method is used instead. However, the control algorithm can be written such that the steady-state Kalman Filter would kick in to replace the recursive Kalman Filter whenever the user decides to switch off the parameter adaptation.

The $\hat{d}+1$ step ahead prediction of the output $Y(t)$ is given by forward shifting the state estimates:

$$\hat{Y}(t+\hat{d}+1|t) = H_1\hat{X}_1(n+\hat{d}+1|t) \tag{4.2.11}$$

Since

$$\hat{X}_1(t+\hat{d}+1|t) = \Phi_1^{\hat{d}+1}\hat{X}_1(t) + \sum_{j=t}^{t+\hat{d}} \Phi_1^{t+\hat{d}+j}\Lambda_1 U(j) \tag{4.2.12}$$

$$\therefore \quad \hat{Y}(t+\hat{d}+1|t) = H_1\Phi_1^{\hat{d}+1}\hat{X}_1(t) + \sum_{j=t}^{t+\hat{d}} H_1\Phi_1^{t+\hat{d}+j}\Lambda_1 U(j) \tag{4.2.13}$$

4.2.2 Modified Kalman Filter Predictor: disturbance modelling

The future prediction of the process output $\hat{Y}(t+\hat{d}+1|t)$ can be used as the feedback signal to replace the real process signal. When $\eta_1(t)$ and $\eta_2(t)$ are white noise, i.e. random noise with zero mean, this predictor will result in zero-offset prediction (provided there is no MPM). However, in the presence of coloured noises and/or in the presence of non-zero mean disturbances, the predictor will give biased results. To account for non zero mean noise/disturbance, Walgama (1986) proposed the Modified Kalman Filter Predictor(MKFP). An integrator is embedded into the state space model to represent the deterministic non-stationary noise dynamics. The state matrix and the input/output matrices in Equations (4.2.14) and (4.2.15) are redefined as follows:

$$X(t+1) = \Phi\ X(t) + \Lambda\ U(t) + \Xi\ \eta_1(t) \tag{4.2.14}$$

$$Y(t) = H\ X(t) + \eta_2(t) \tag{4.2.15}$$

where for $\hat{d}=k-1$

$$X(t) = [x_0(t)\ x_1(t)\ x_2(t)\ \ldots\ x_n(t)\ x_{n+1}(t)\ \ldots\ x_{n+\hat{d}}(t)]^T$$

$$\Phi = \begin{bmatrix} 1 & 0 & \cdots & 0 & 0 & \cdots & 0 & 0 \\ \xi_1 & 0 & \cdots & 0 & -a_n & \cdots & 0 & 0 \\ \xi_2 & 1 & \cdots & 0 & -a_{n-1} & \cdots & 0 & 0 \\ \cdot & & & & & & & \\ \xi_n & & & 1 & -a_1 & \cdots & 0 & 0 \\ 0 & & & & 1 & & & \cdot \\ \cdot & & & & & & & \cdot \\ \cdot & & & & & & & \cdot \\ 0 & 0 & & \cdots & & & 1 & 0 \end{bmatrix}_{(n+\hat{d}+1)\times(n+\hat{d}+1)}$$

$$\Lambda^t = [0 \ b_n \ b_{n-1} \ \ldots \ b_1 \ 0 \ \ldots \ 0]_{1\times(n+\hat{d}+1)}$$

$$H = [0 \ \ldots \ 0 \ 1]_{1\times(n+\hat{d}+1)}$$

$$\Xi^t = [1 \ 0 \ \ldots \ 0]_{1\times(n+\hat{d}+1)}$$

An extra state xo(t) is augmented to the original state-space model. It monstrated that the state-space system defined in Equations (4.2.15) is equivalent to the following input-output CARIMA

$$= \frac{B(z^{-1})}{A(z^{-1})} z^{-\hat{d}} U(t-1) + \frac{C(z^{-1})}{A(z^{-1})} z^{-\hat{d}-1} \frac{\eta_1(t)}{\Delta} + \eta_2(t) \qquad (4.2.16)$$

$$A(z^{-1}) = 1 + a_1 z^{-1} + a_2 z^{-2} + \ldots + a_n z^{-n}$$

$$B(z^{-1}) = b_1 + b_2 z^{-1} + \ldots + b_n z^{-n+1}$$

$$C(z^{-1}) = \xi_n z^{-1} + \xi_{n-1} z^{-2} + \ldots + \xi_1 z^{-n}$$

Notice that the only difference between Equation (4.2.3) and Equation (4.2.16) is the presence of the Δ term. Whereas the ARMAX model requires $\eta_1(t)$ and $\eta_2(t)$ to be zero mean for unbiased process representation, Δ in the CARIMA model relaxes this requirement. $\eta_2(t)$ can now represent non-zero mean disturbances because $\Delta\eta_2(t)=0$ at steady-state. Therefore, $\eta_1(t)$ can be interpreted as zero mean noise and $\eta_2(t)$ can be interpreted as non-zero mean random disturbances (eg. the steady-state bias and step disturbances).

Equation (4.2.16) provides an insight into the physical interpretation of $\eta_1(t)$ and $\eta_2(t)$ so that choosing R_1 and R_2 for the Kalman Filter predictor becomes physically meaningful. Experience has shown that it is the ratio of the noise variance R_2/R_1 and not the absolute value of each noise variance that affects the performance of the MKFP. Therefore, the common practice is to choose one (usually R_2 because it is more physically interpretable) and vary the other.

Innovations analysis is used to examine the structure of the Modified Kalman Filter Predictor and thereby understand how it rejects disturbances. The details of the analysis are in Appendix 1 and the results are listed as follows:

From the innovations analysis of the MKFP,

$$Y(t) = \frac{B(z^{-1})}{A(z^{-1})} z^{-\hat{d}} U(t-1) + \frac{C'(z^{-1})}{A(z^{-1})} \frac{\omega(t)}{\Delta} \tag{A1.7}$$

where $\omega(t)$ is the innovations sequence which is defined as

$$\omega(t) = Y(t) - \hat{Y}(t|t-1) \tag{A1.2}$$

$\omega(t)$ contains new information that $\hat{Y}(t|t-1)$ does not have, ie. it includes the current effects from process noise, measurement noise, disturbances, as well as unmodelled dynamics.

As shown in Appendix 1, the observer polynomial $C'(z^{-1})$ is chosen optimally according to the variances of the process and measurement noise, R_1 and R_2. With R_1 and R_2 specified, the Kalman gain $L(t)$ can be calculated and the $C'(z^{-1})$ is given by Equation (A1.8):

$$C'(z^{-1}) = [A(z^{-1})(1+K_2(z^{-1}))+z^{-\hat{d}}K_1(z^{-1})]\Delta+z^{-\hat{d}}D(z^{-1}) \tag{A1.8}$$

where

$$K_1(z^{-1}) = l_n + l_{n-1}z^{-1} + \ldots + l_1 z^{-n+1}$$

$$K_2(z^{-1}) = l_{n+d-1}z^{-1} + \ldots + l_{n+1}z^{-\hat{d}+1}$$

$$D(z^{-1}) = l_0[\xi_n z^{-1} + \ldots + \xi_1 z^{-n}]$$

l_i, $i \in [0,\ n+\hat{d}-1]$ are the elements of the Kalman gain vector obtained from Equations (4.2.4) to (4.2.8).

As can be seen from Equation (A1.7), the MKFP utilizes a step type model as the internal model for the disturbances $\omega(t)$. In theory, each type of disturbance (eg. a ramp type disturbance or a sinusoidal type disturbance) requires a correct model in order to get good disturbance rejection. However, since the most common type of disturbances encountered in the process industry are persistent (step) type functions, this internal model is often sufficient for handling a majority of them.

The original development of MAPC by Sripada (1988) uses a residual model as the disturbance model. This residual model is identified on-line by fitting a moving average model to the residual sequence (Single series forecasting, Man, 1984). Apart from complicating the formulation of the controller and requiring more computational effort, the main difficulty is to obtain on-line the process model and the disturbance model totally independent of each other from the input signals alone. Although Sripada suggested ways of switching on and off the model estimators under special conditions so as to obtain the models, Foley (1988) mentioned that such a system is basically non-realizable if the dynamics of the process are time-varying and unknown. Therefore, the MKFP appears to be a much simpler and realizable predictor.

The $\hat{d}+1$ step ahead prediction $\hat{Y}(t+\hat{d}+1|t)$ can again be obtained by forward shifting the state estimates:

$$\hat{Y}(t+\hat{d}+1|t) = H\Phi^{\hat{d}+1}\hat{X}(t) + \sum_{j=t}^{t+\hat{d}} H\Phi^{t+\hat{d}-j}\Lambda U(j) \tag{4.2.16}$$

To further investigate the internal structure of the MKFP so that it can be compared with the Smith Predictor (Chapter 2) and the optimal least squares predictor (Chapter 3), the prediction $\hat{Y}(t+\hat{d}+1|t)$ is reexpressed using the model in Equation (A1.7). Forward shifting Equation (A1.7) $\hat{d}+1$ steps ahead gives:

$$Y(t+\hat{d}+1) = \frac{B(z^{-1})}{A(z^{-1})} U(t) + \frac{C'(z^{-1})}{A(z^{-1})\Delta}\omega(t+\hat{d}+1) \qquad (4.2.17)$$

The following Diophantine Identity is then used to separate past and future noise terms:

$$\frac{C'(z^{-1})}{A(z^{-1})\Delta} = E(z^{-1}) + \frac{F(z^{-1})}{A(z^{-1})\Delta} z^{-\hat{d}-1} \qquad (4.2.18)$$

From Equation (A1.8),

$$E(z^{-1}) = 1 + K_2(z^{-1}) \qquad (4.2.19)$$

$$F(z^{-1}) = K_1(z^{-1})\Delta + D(z^{-1}) \qquad (4.2.20)$$

Substituting Equations (4.2.19) and (4.2.20) to Equation (4.2.17) gives

$$Y(t+\hat{d}+1) = \frac{B(z^{-1})}{A(z^{-1})} U(t) + \frac{K_1(z^{-1})\Delta+D(z^{-1})}{A(z^{-1})\Delta}\omega(t) + (1+K_2(z^{-1}))\omega(t+\hat{d}+1) \qquad (4.2.21)$$

Reconstructing $\omega(t)$ from Equation (A1.7) and substituting into Equation (4.2.21) gives

$$\begin{aligned} Y(t+\hat{d}+1) = {} & \frac{B(z^{-1})}{A(z^{-1})} U(t) + (1+K_2(z^{-1}))\omega(t+\hat{d}+1) + \\ & \frac{K_1(z^{-1})\Delta+D(z^{-1})}{C'(z^{-1})} Y(t) - \\ & \frac{K_1(z^{-1})\Delta+D(z^{-1})}{A(z^{-1})\Delta} \frac{B(z^{-1})}{C'(z^{-1})} z^{-\hat{d}}\Delta U(t) \end{aligned} \qquad (4.2.22)$$

Since $\frac{K_1(z^{-1})\Delta+D(z^{-1})}{A(z^{-1})\Delta} z^{-\hat{d}-1} = \frac{C'(z^{-1})}{A(z^{-1})\Delta} - (1+K_2(z^{-1}))$, Equation (4.2.22) becomes

$$Y(t+\hat{d}+1) = \frac{K_1(z^{-1})\Delta+D(z^{-1})}{C'(z^{-1})} Y(t) + \frac{(1+K_2(z^{-1}))}{C'(z^{-1})} B(z^{-1})\Delta U(t) +$$

$$(1+K_2(z^{-1}))\omega(t+\hat{d}+1) \qquad (4.2.23)$$

Taking the expected values of Equation (4.2.23) gives the d+1 step ahead prediction:

$$\hat{Y}(t+\hat{d}+1|t) = \frac{K_1(z^{-1})\Delta+D(z^{-1})}{C'(z^{-1})}Y(t)+\frac{(1+K_2(z^{-1}))}{C'(z^{-1})}B(z^{-1})\Delta U(t) \qquad (4.2.24)$$

Therefore, Equation (4.2.24) presents an alternative way to obtain the $\hat{d}+1$ step ahead prediction rather than by Equation (4.2.16). This method is computationally more efficient especially when the time delay is large. Notice that this predictor is identical to the prediction in Chapter 3 (see Equation (3.5.9)) (Foley, 1988). Equation (4.2.24) can further be expressed in the following manner using the Diophantine Identity in Equation (4.2.18):

$$\hat{Y}(t+\hat{d}+1|t) = \frac{B(z^{-1})}{A(z^{-1})}U(t) + Gf(z^{-1})\left(Y(t) - \frac{B(z^{-1})}{A(z^{-1})}z^{-\hat{d}-1}U(t)\right) \qquad (4.2.25)$$

Equation (4.2.25) can again be represented by the block diagram in Figure 2.2 where $Gf(z^{-1})$ is obtained optimally through the MKFP:

$$Gf(z^{-1}) = \frac{K_1(z^{-1})\Delta+D(z^{-1})}{C'(z^{-1})} \qquad (4.2.26)$$

Notice that the steady state value of the filter is unity, ie. $Gf(1) = D(1)/C'(1) = 1$. This is important for the MKFP to give offset free prediction.

4.2.3 The role of $Gf(z^{-1})$

Smith Predictor(SP):

In light of Equation (4.2.25) and Figure 2.2, the MKFP can be interpreted as a SP with an optimal time varying residual filter $Gf(z^{-1})$. In fact, $Gf(z^{-1})$ is the only difference between the MKFP and the SP. For the SP, $Gf(z^{-1})$ is unity. This means that the future effect of the residuals is predicted to be the same as the current effect (see Equation (4.2.25)). The controller will take immediate action to regulate this effect. This method is found to be robust if the model can closely represent the process. In fact, the Dynamic Matrix Control (Cutler and Ramaker, 1980) uses a multistep SP to generate the future trajectories and is proven to be very robust in industrial application. However, high frequency noise, unmodelled dynamics, time delay mismatch may cause problems. For example, the SP requires the exact knowledge of the process time delay. If time delay mismatch is big, the time delay term is reintroduced into the closed loop characteristic equation which makes the closed loop stability difficult to guarantee without detuning $G_c(z^{-1})$. However, an exact knowledge of the process time delay is generally not available because the delay is often a function of the process operating conditions. The specification of $G_f(z^{-1})$ becomes a necessity to guarantee robustness in the presence of noise and MPM.

Least-square predictor(LSP):

The LSP is found to have a similar structure to the MKFP except for $G_f(z^{-1})$. It is derived in section 3.5.2 and is reproduced here for convenience. The process model is

$$A(z^{-1})Y(t) = B(z^{-1})z^{-\hat{d}}U(t-1) + \frac{C(z^{-1})}{\Delta}\xi(t) \tag{4.2.27}$$

Multiplying Equation (4.2.27) by $E(z^{-1})\Delta/T(z^{-1})$ and rearranging gives

$$C(z^{-1})\frac{Y(t+\hat{d}+1)}{T(z^{-1})} = F(z^{-1})\frac{Y(t)}{T(z^{-1})} + E(z^{-1})B(z^{-1})\frac{\Delta U(t)}{T(z^{-1})} + E(z^{-1})\frac{C(z^{-1})}{T(z^{-1})}\xi(t+\hat{d}+1) \tag{4.2.28}$$

where $E(z^{-1})$ and $F(z^{-1})$ are obtained from the Diophantine identity

$$C(z^{-1}) = E(z^{-1})A(z^{-1})\Delta + F(z^{-1})z^{-\hat{d}-1} \tag{4.2.29}$$

The least-square prediction is given by

$$C(z^{-1})\frac{\hat{Y}(t+\hat{d}+1|t)}{T(z^{-1})} = F(z^{-1})\frac{Y(t)}{T(z^{-1})} + E(z^{-1})B(z^{-1})\frac{\Delta U(t)}{T(z^{-1})}$$

$$= F(z^{-1})\frac{Y(t)}{T(z^{-1})} + \frac{C(z^{-1})B(z^{-1})}{A(z^{-1})}\frac{U(t)}{T(z^{-1})} -$$

$$\frac{F(z^{-1})B(z^{-1})}{A(z^{-1})}z^{-\hat{d}-1}\frac{U(t)}{T(z^{-1})}$$

$$\therefore \quad C(z^{-1})\frac{\hat{Y}(t+\hat{d}+1|t)}{T(z^{-1})} = \frac{C(z^{-1})}{T(z^{-1})}\frac{B(z^{-1})}{A(z^{-1})}U(t) + \frac{F(z^{-1})}{T(z^{-1})}\left(Y(t)-\frac{B(z^{-1})}{A(z^{-1})}U(t-\hat{d}+1)\right) \tag{4.2.30}$$

Since identifying $C(z^{-1})$ on-line is known to be a difficult problem, a user chosen polynomial $T(z^{-1})$ known as the T-filter (Clarke, 1985) is assumed to be approximately equal to $C(z^{-1})$. In such case, the $\hat{d}+1$ step ahead prediction is given by

$$\hat{Y}(t+\hat{d}+1|t) = \frac{B(z^{-1})}{A(z^{-1})}U(t) + Gf(z^{-1})\left(Y(t) - \frac{B(z^{-1})}{A(z^{-1})}U(t-\hat{d}+1)\right) \tag{4.2.31}$$

where

$$Gf(z^{-1}) = \frac{F(z^{-1})}{T(z^{-1})} \tag{4.2.32}$$

Notice that $F(z^{-1})$ is uniquely determined by $T(z^{-1})$ through Equation (4.2.29) when $A(z^{-1})$ is known (ie. $T(z^{-1}) = E(z^{-1})A(z^{-1})\Delta + F(z^{-1})z^{-\hat{d}-1}$). The problem of identifying $C(z^{-1})$ on-line reduces to a problem of selecting a suitable $T(z^{-1})$ polynomial. However, choosing $T(z^{-1})$ is a complicated matter since the order of $T(z^{-1})$ and the locations of its roots can affect the robustness of the closed-loop as well as disturbance rejection characteristic. It is generally desired that $Gf(z^{-1})$ be low pass so that residuals resulting from noise plus high frequency unmodelled dynamics can be attenuated.

Figures 4.1a and 4.1b are magnitude plots of the frequency response of $Gf(z^{-1})=F(z^{-1})/T(z^{-1})$ (where $A(z^{-1})=1-0.901z^{-1}$, and $T(z^{-1})=(1-\beta z^{-1})^n$). For $n=1$, $Gf(z^{-1})$ is low pass only when the time constant of $T(z^{-1})$ is larger than the time constant of $A(z^{-1})$ (see Figure 4.1a). For $n=2$, $Gf(z^{-1})$ has a much better low pass characteristic as β increases, except the case where $\beta=0.99$. With $\beta=0.99$, the magnitude of the filter is greater than one at some low frequencies. This choice of β is not as good as the others as far as robustness of the filter is concern. The loss of performance is due to the non-minimum phase zero in $Gr(z^{-1})$ as explained by Mohtadi (1988). With a higher order $T(z^{-1})$, this phenomenon is more likely to happen. Therefore, the above simulations lead to the following guide-lines for the choice of $T(z^{-1})$ as suggested by Mohtadi (1988) and McIntosh (1988):

1) $T(z^{-1})$ is recommended to be used all the time,
2) Since $\delta F(z^{-1})=\delta A(z^{-1})$ (from Equation (4.2.29)), $\delta T(z^{-1})\geq\delta A(z^{-1})$ is a criterion for ensuring reasonable high frequency attenuation properties. In particular, $T(z^{-1})$ is chosen as $(1-\beta z^{-1})^{\delta A}$ with $\beta\approx$the dominant time constant of $A(z^{-1})$. This will ensure low pass characteristic of $Gr(z^{-1})$.

Modified Kalman Filter Predictor(MKFP):

The MKFP weights the residual sequence $\omega(t)$ against an 'optimally' chosen $Gr(z^{-1})$. This enhances the performance of the predictor because the predictor is now less sensitive to the effects of the residual. There is also one tuning knob for the MKFP, R_2/R_1 (similar to the β in the LSP). By adjusting R_2/R_1, $Gr(z^{-1})$ can be tailored to filter different types of residuals. It appears that a large R_2/R_1 suffices to give 'good' prediction regardless of the noise and unmodelled dynamics present. This makes the tuning of the MKFP easier than the LSP. Section 4.2.4 shows simulations on how R_2/R_1 affects the characteristic of $Gr(z^{-1})$.

The residual filter $Gr(z^{-1})$ is designed by the MKFP. Notice from Equation (4.2.26) that the degrees of the numerator and denominator polynomials of $Gr(z^{-1})$ depend only on the process model (model order and time delay) and their coefficients depend only on the model parameters and the kalman gain. R_2/R_1 alters the value of the kalman gain and thereby adjusts the poles and zeros of $Gr(z^{-1})$. This makes the MKFP very easy to use compare with the T-filter.

Closed-loop performance:

Before going into simulations, the closed loop transfer function is first examined to investigate how $Gr(z^{-1})$ affects the closed loop stability and behavior in two special cases. Consider the closed loop transfer function described as in Figure 2.3 (The argument (z^{-1}) is omitted for simplicity).

$$Y(t) = G_s(z^{-1})Y_{sp}(t) + G_r(z^{-1})\omega(t) \qquad (4.2.33)$$

where

$$G_s(z^{-1}) = \frac{GG_c}{1+G_pG_c+G_fG_c(G-G_m)}$$

$$G_r(z^{-1}) = \frac{1+G_mG_c-G_fG_pG_c}{1+G_pG_c+G_fG_c(G-G_m)}$$

Case 1: $G(z^{-1}) \neq G_m(z^{-1})$

Whenever there is MPM, whether the model order is not sufficient to represent the process dynamics or the time delay is over or under-estimated, the denominator (or characteristic equation) of the filter transfer function $G_f(z^{-1})$ enters the characteristic equations of both the closed loop servo transfer function $G_s(z^{-1})$ and the regulatory transfer function $G_r(z^{-1})$. Therefore, the stability and the performance of the filter affects both 'modes' of the closed loop.

Case 2: $G(z^{-1}) = G_m(z^{-1})$

When there is no MPM, the servo and regulatory models reduce to

$$G_s(z^{-1}) = \frac{GG_c}{1+G_pG_c}$$

$$G_r(z^{-1}) = \frac{1+G_pG_c-G_fGG_c}{1+G_pG_c} = 1-G_fG_s$$

Notice that for this ideal case, $G_f(z^{-1})$ only affects the characteristics of the disturbance rejection. This enables the controller to perform servo and regulatory control independently (a two degrees of freedom controller).

Since R_2/R_1 affects the performance of $G_f(z^{-1})$, it becomes the main tuning knob for the closed loop performance. In particular, when the MPM is significant, a large R_2/R_1 ratio desensitizes the residual effect and therefore making the closed loop more robust to MPM (see the characteristic equation in Equation (4.2.33)). When the MPM is insignificant, on the other hand, R_2/R_1 becomes solely the tuning knob for disturbance rejection. Since there is bound to be some MPM in any model based control algorithm, adjusting R_2/R_1 affects both modes of operation. This may not be too disadvantageous because complete decoupling of the two modes are usually not a crucial requirement in many process control applications.

4.2.4 Simulation on MKFP in the case of time delay mismatch

The following set of simulations shows the performance of the MKFP when time delay mismatch occurs. The characteristics of the filter $G_f(z^{-1})$ are also investigated. The model used for simulation is as follows:

$$Y(s) = \frac{2.16}{48s+1} e^{-30s}\, U(s) \qquad (4.2.34)$$

The process is sampled at 5 seconds and the corresponding discrete input-output ARMA model is:

$$Y(t) = \frac{0.045}{1-0.901z^{-1}}\, z^{-6}U(t-1) \qquad (4.2.35)$$

This is converted to the following state space model with process and measurement noise added:

$$X(t+1) = \Phi\, X(t) + \Lambda\, U(t) + \Xi\, \eta_1(t) \qquad (4.2.36)$$

$$Y(t) = H\, X(t) + \eta_2(t)$$

where for $\hat{d}=6$, $n=1$,

$$X(t) = [x_0(t)\ x_1(t)\ x_2(t)\ x_3(t)\ x_4(t)\ x_5(t)\ x_6(t)\ x_7(t)]^T$$

$$\Phi = \begin{bmatrix} 1 & 0 & 0 & 0 & 0 & 0 & 0 & 0 \\ 1 & -.901 & 0 & 0 & 0 & 0 & 0 & 0 \\ 0 & 1 & 0 & 0 & 0 & 0 & 0 & 0 \\ 0 & 0 & 1 & 0 & 0 & 0 & 0 & 0 \\ 0 & 0 & 0 & 1 & 0 & 0 & 0 & 0 \\ 0 & 0 & 0 & 0 & 1 & 0 & 0 & 0 \\ 0 & 0 & 0 & 0 & 0 & 1 & 0 & 0 \\ 0 & 0 & 0 & 0 & 0 & 0 & 1 & 0 \end{bmatrix}_{8\times8}$$

$\Lambda^t = [0\ 0.045\ 0\ 0\ 0\ 0\ 0\ 0]_{1x8}$

$H = [0\ 0\ 0\ 0\ 0\ 0\ 0\ 1]_{1x8}$

$\Xi^t = [1\ 0\ 0\ 0\ 0\ 0\ 0\ 0]_{1x8}$

$\eta_1(t)$ = Gaussian noise with variance = 0.0025

$\eta_2(t)$ = Gaussian noise with variance = 0.3

Figure 4.2 shows the open loop step response of the process. PRBS of magnitude one is injected into the steps to enhance excitation. The optimal state estimates $\hat{X}(t)$ are obtained by the MKFP and the $\hat{d}+1$ step ahead prediction is calculated by Equation (4.2.16). The performance of the MKFP using different R_2/R_1 is first examined.

Case 1: No time delay mismatch

When there is no time delay mismatch ($d=\hat{d}=6$), the MKFP generally gives 'good' offset free $\hat{d}+1$ step ahead prediction (see Figure 4.3a, b, c). Increasing R_2/R_1 improves the filtering property of the MKFP but worsens the tracking property slightly. With R_2/R_1 set to the correct value (Figure 4.3b, $R_2/R_1=120$), the MKFP gives 'optimal' prediction in a sense that the estimation error is minimum. The performance is seen to be 'best' among the three settings of R_2/R_1.

Case 2: With time delay mismatch

The MKFP is tested under time delay underestimated case ($d=6$, $\hat{d}=1$). The results are shown in Figure 4.4a, b, c. In this case, the MKFP is only predicting 1 step ahead while the process has a delay of 6 steps. It is desired that the predictor can give 6 step ahead prediction regardless of the time delay underestimation. With small R_2/R_1 ($R_2/R_1=2.4$), or even with the correct R_2/R_1 setting ($R_2/R_1=120$) , the prediction tracks the process output very closely. The effect due to the delay mismatch is treated as disturbance and is rejected quickly. With a large R_2/R_1 (Figure 4.4c, $R_2/R_1=6000$), the MKFP 'averages' out the error. As a result, the 1 step ahead prediction resembles the 6 step ahead prediction (a pseudo-prediction). Figures 4.5a, b, and c overlays the prediction with $\hat{d}$ varying from 1 to 12 ($d=6$). From these results, it can be seen that the performance is worse when the delay is overestimated (Figure 4.5a and b).

Figure 4.6 summarizes the above results by showing the variance of the difference between the predicted output and the process undelayed output (ie. $\mathrm{var}\{Y(t+\hat{d}+1|t)-Y(t+d+1)\}$) with respect to the model time delay $\hat{d}$. The variance is least when both R_2/R_1 and $\hat{d}$ are set to the correct values. As $\hat{d}$ departs from d, a large R_2/R_1 is necessary to reduce the variance. This leads to the conclusion that a large R_2/R_1 is necessary for good filtering and predicting actions especially when the time delay is not exactly known.

As mentioned in section 4.2.3, the performance of the MKFP is very much related to the performance of the filter $G_f(z^{-1})$. The characteristics of $G_f(z^{-1})$ in relation to R_2/R_1 is therefore examined. With no time delay mismatch ($d=\hat{d}=6$), the locations of the roots with different R_2/R_1 settings is shown in Figure 4.7. As R_2/R_1 increases, 2 conjugate pairs migrate towards the center of the unit circle while the other pair migrates towards a number close to unity. In the limiting case, the characteristic equation reduces to a n-th order equation $(1-\beta z^{-1})^n$ (where β is a number smaller than but close to unity). This justifies the choice of the T-filter as mentioned in section 4.2.3. A simulation plot is done for a time delay underestimation case in Figure 4.8. The order of the characteristic equation is now reduced since it is affected by $z^{-\hat{d}}$. A similar trend is observed: as R_2/R_1 increases, one root migrates towards the center while the other conjugate pair migrates towards a number smaller than but close to unity. Notice however that there is one unstable root at about -1.8 when $R_2/R_1=2.4$. This is a caution to the user that the MKFP can become unstable in the presence of delay mismatch if R_2/R_1 is not chosen appropriately large.

Figures 4.9 and 4.10 show the frequency responses of the filters for no delay mismatch case and delay mismatch case, respectively. In both cases, the MKFP can attenuate high frequency noise more effectively with large R_2/R_1 (ie. the larger R_2/R_1, the stronger the high frequency attenuation of $G_f(z^{-1})$). The frequency responses obtained from the least-square predictor (where $T(z^{-1})=(1-0.9z^{-1})^2$) are overlaid in both figures for comparison.

Notice the close match between the $Gr(z^{-1})$ obtained from the T-filter and that from the MKFP using a large R_2/R_1 especially at the high frequency region. As mentioned before, many authors have stated that the selection of $T(z^{-1})=1-\beta z^{-1}$ with $\beta\approx0.8$ to 0.95 is a good choice. Because of the similarities, the MKFP provides a channel for the explanation of the success of the T-filter in GMV and GPC applications.

4.2.5 Summary to the simulation results

The rule of thumb in using the MKFP is: estimate a value of R_2 (the variance of the measurement noise) and then specify a value of R_1 10^{-4} to 10^{-5} order of magnitude smaller. This will ensure robust performance against effects due to undeterministic disturbances, time delay mismatch, and MPM. Having said these, however, it is important to mention that in feedback control using the MKFP as the predictor, heavy filtering makes disturbance rejection slow because of the extra phase lag introduced to the feedback signal (consider $Gr(z^{-1})=0$ in the limiting case). Furthermore, the predicted trajectory becomes more sluggish as R_2/R_1 increases. This slows down the tracking ability of the controller as well. In other words, the controller performance is highest only at some values of R_2/R_1, which depends on the degree of MPM. This is a classical trade of in robustness and performance. Therefore, in closed-loop control where the time delay is uncertain, it is recommended to start with a large value of R_2/R_1 and progressive tune it down to increase performance.

4.3 Adaptive Modified Kalman Filter Predictor

4.3.1 Problem formulation and requirements

For a time varying process, Equations (4.2.14) and (4.2.15) can be rewritten as

$$X(t+1) = \Phi(t)\ X(t) + \Lambda(t)\ U(t) + \Xi\ \eta_1(t) \tag{4.3.1}$$

$$Y(t) = H(t)\ X(t) + \eta_2(t) \tag{4.3.2}$$

where for $\hat{d}=k-1$

$$X(t) = [x_0(t)\ x_1(t)\ x_2(t)\ \ldots\ x_n(t)\ x_{n+1}(t)\ \ldots\ x_{n+\hat{d}}(t)]^T$$

$$\Phi = \begin{bmatrix} 1 & 0 & \dots & 0 & 0 & \dots & 0 & 0 \\ \xi_1 & 0 & \dots & 0 & -a_n & \dots & 0 & 0 \\ \xi_2 & 1 & \dots & 0 & -a_{n-1} & \dots & 0 & 0 \\ \cdot & & & & & & & \\ \xi_n & & & 1 & -a_1 & \dots & 0 & 0 \\ 0 & & & & 1 & & & \cdot \\ \cdot & & & & & & & \cdot \\ \cdot & & & & & & & \cdot \\ 0 & 0 & & \dots & & & 1 & 0 \end{bmatrix}_{(n+\hat{d}+1)x(n+\hat{d}+1)}$$

$$\Lambda^t = [0 \; b_n \; b_{n-1} \; \dots \; b_1 \; 0 \; \dots \; 0]_{1x(n+\hat{d}+1)}$$

$$H = [0 \; \dots \; 0 \; 1]_{1x(n+\hat{d}+1)}$$

$$\Xi^t = [1 \; 0 \; \dots \; 0]_{1x(n+\hat{d}+1)}$$

As in Equation (4.2.3), the time varying state-space system can be expressed in the following input-output CARIMA model:

$$Y(t) = \frac{B(z^{-1},t)}{A(z^{-1},t)} z^{-\hat{d}} U(t-1) + \frac{C(z^{-1},t)}{A(z^{-1},t)} z^{-\hat{d}-1} \frac{\eta_1(t)}{\Delta} + \eta_2(t) \qquad (4.3.2)$$

where

$$A(z^{-1},t) = 1 + a_1(t)z^{-1} + \dots + a_n(t)z^{-n}$$

$$B(z^{-1},t) = b_1(t) + b_2(t)z^{-1} + \dots + b_n(t)z^{-n+1}$$

$$C(z^{-1},t) = \xi_n(t)z^{-1} + \xi_{n-1}(t)z^{-2} + \dots + \xi_1(t)z^{-n}$$

Since the choices of ξ_i affect the dynamics of the MKFP as much as R_2/R_1, they can be used as the tuning parameters as well. Since they are difficult to estimate on-line, as explained earlier, tuning of the MKFP will be easier if ξ_i are user-prechosen and only R_2/R_1 is used as the tuning knob. Thus even though $C(z^{-1},t)$ is time variant, it is not estimated on line. $A(z^{-1},t)$ and $B(z^{-1},t)$ can be estimated on line using a RLS based parameter estimator. Figure 4.11 shows the structure of the AMKFP. In general, any parameter estimator that has the following properties can be used:

1) Must retain alertness when changing from one operating condition to another operating condition.
2) Must be able to avoid parameter drift in a noisy environment.
3) Must be able to provide a model that can closely represent the process even in the presence of noise and disturbance.
4) Must be reasonably fast in parameter convergence and must be numerically stable.

A fast converging parameter estimator is crucial to the success of the Kalman Filter Predictor since the convergence of the Kalman Filter Predictor depends on the process model supplied from the parameter estimator, (see Figure 4.11). Recursive Least Square (RLS) algorithm provides fast convergence compared with other estimation schemes (eg. projection algorithm). Therefore it is chosen as the basic parameter estimator for the AMKFP.

4.3.2 Recursive Least Square Algorithm

The RLS algorithm is based on the minimization of the following cost function with respect to $\hat{\theta}$:

$$J_{RLS} = \sum_{i=0}^{t} \lambda(i)^{t-1}(Y(t) - \phi(i)^{t}\hat{\theta})^{2}, \quad 0<\lambda<1 \tag{4.3.4}$$

where

λ(i) is the data discounting factor or the forgetting factor.

The resulting algorithm is as follows:

Parameter update:

$$\hat{\theta}(t) = \hat{\theta}(t-1) + K(t-1)\phi(t-i)[y(t)-\phi(t-1)^{t}\hat{\theta}(t-1)] \tag{4.3.5}$$

Gain update:

$$K(t-1) = P(t-2)/[\lambda(t)+\phi(t-1)^{t}P(t-2)\phi(t-1)] \tag{4.3.6}$$

Covariance update:

$$P(t-1) = [P(t-2) - \frac{P(t-2)\phi(t-1)\phi(t-1)^{t}P(t-2)}{\lambda(t) + \phi(t-1)^{t}P(t-2)\phi(t-1)}]/\lambda(t) \tag{4.3.7}$$

where $$\phi(t-1)^t = [\Delta Y(t-1) \;\; \Delta Y(t-2) \; \ldots \; \Delta Y(t-n), \;\; \Delta U(t-\hat{d}-1) \;\; \Delta U(t-\hat{d}-2) \; \ldots \; \Delta U(t-\hat{d}-n)]$$

and $$\hat{\theta}^t(t-1) = [\hat{a}_1 \;\; \hat{a}_2 \ldots \hat{a}_n, \; \hat{b}_1 \;\; \hat{b}_2 \ldots \hat{b}_n]$$

The properties of the RLS algorithm have been widely discussed by Goodwin and Sin (1984). There are several draw backs which make it impractical for on-line identification: it encounters problems when the input signals are either too rich or too poor, resulting in covariance turn-off and covariance blow-up problems respectively. The following section briefly discusses the problems and provides appropriate solutions to these problems.

4.3.3 Problems relating to the Recursive Least Square Algorithm

The basic RLS algorithm with a unity constant forgetting factor is known to have the problem of "falling asleep". This occurs because the covariance matrix P decreases rapidly as the parameter estimates converge to some values. The rate of convergence is especially fast when the input signals are rich, ie. persistently exciting. Since P(t) basically defines the alertness of the algorithm, this will shut off the parameter update and the estimator "falls asleep" forever. Many ad hoc schemes have been proposed to fix this problem. Two most prominent ones are "covariance resetting" (Goodwin and Sin, 1988) and "constant trace variable forgetting factor" as proposed by Sripada (1987). The covariance resetting scheme will reset the covariance matrix P(t) to P(0) if the trace of P(t) becomes smaller than a predefined value. The constant trace variable forgetting factor scheme calculates at every sample instant a forgetting factor $\lambda(t)$ such that the trace of P(t) is maintained. Both of the schemes try to retain the alertness of the RLS algorithm by preventing the "shrinkage" of the covariance matrix.

When the input signals are poor, i.e. due to insufficient excitation, the RLS algorithm will encounter covariance blowup. This leads to a very erratic behavior of the estimator and the parameter estimates become very oscillatory. Furthermore, noisy input signals will result in parameter drift because the large covariance matrix magnifies the effect of the noise. Ad hoc fixes have been proposed to deal with this problems such as selective discounting on the input signals using a variable forgetting factor and using a dead zone to stop parameter estimation when the signals are 'silent' (Ydstie, 1985).

In order to obtain an 'unbiased' input/output representation of the process, very high frequency disturbances have to be filtered (low-passed) from the I/O data before being used for process identification. The band-width of the filter must be large enough to permit accurate estimation of the actual variations in the process parameters. McIntosh (1988) discussed in detail the choice of this low-pass filter ($1/T(z^{-1})$, where $T(z^{-1})$ is usually chosen as a first or second order polynomial). He also suggested to use a bandpass filter of the form $\Delta/T(z^{-1})$. By differencing the signals, very low frequency disturbances occur from dc bias or process non-linearity can be removed.

The parameter estimator chosen for this work utilizes the following features to ensure proper on-line working conditions:

1) Ydstie's (Ydstie, 1985) variable forgetting factor for old data discounting which is based on the idea of keeping the weighted sum of the residual constant.
2) Cluett's (Cluett, 1988) normalization factor and estimator on/off scheme to ensure robustness to unmodelled dynamics.
3) UDU factorization on the covariance update to increase numerical stability.
4) Data filtering by a band pass filter (McIntosh, 1988).

4.4 Multistep Controller Design

4.4.1 Basic development of the Multistep controller

The main reason for the robustness of MAPC towards time delay mismatch lies partly in the robustness of the MKFP but mainly in its multistep controller design. The MAPC calculates the set of the future input sequence $\{U(t+i),\ i\in[1,N_u]\}$ based on the minimization of a multistep cost functional.

It minimizes the setpoint error and the control effort squared over the prespecified trajectory. The objective function is given by:

$$J_{MAPC} = \sum_{j=N1}^{N2}\{Y_{sp}(t+j)-\hat{Y}(t+j|t)\}^2\gamma_{Yj}(t)+\sum_{j=1}^{Nu}\{\Delta U(t-j-1)\}^2\gamma_{Uj}(t) \qquad (4.4.1)$$

where $\hat{Y}(t+j|t)$ is the predicted output trajectory generated from the AMKFP, the $Y_{sp}(t+j)$ is the preset future setpoint trajectory and $\Delta U(t-j-1)$ is the future control sequence. $\gamma_{Yj}(t)$ and $\gamma_{Uj}(t)$ are weighting functions which provide individual weightings on both input and output trajectories. N_1 and N_2 are the output horizons ($N_2 \geq N_1 \geq 1$) and N_u is the controlled horizon ($1 \leq N_u < N_2$).

At every sample instant, a sequence of estimated future process output $\{\hat{Y}(t+j|t),\ j\in[N_1,N_2]\}$ is generated by the AMKFP simply by forward shifting the state-space model j number of times:

$$\begin{aligned}\{\hat{Y}(t+j|t),\ j\in[N_1,N_2]\} &= \{H(t)\Phi(t)^j\hat{X}(t|t),\ j\in[N_1,N_2]\} + \\ &\{\sum_{i=t}^{t+j-1} H(t)\Phi(t)^{t+j-i-1}\Lambda(t)U(i),\ j\in[N_1,N_2]\}\end{aligned} \qquad (4.4.2)$$

It can be shown that this is equivalent to the block diagram in Figure 4.12 (Foley, 1988). The predictive structure of Equation (4.4.2) is equivalent to a network of parallel, optimal Smith predictor (see sections 4.2.2 and 4.2.3).

Rearranging Equation (4.4.2) gives

$$\begin{aligned}\{Y(t+j|t),\ j\in[N_1,N_2]\} &= \{H(t)\Phi(t)^j\hat{X}(t|t),\ j\in[N_1,N_2]\} + \\ &h(t)\{U(t+j),\ j\in[N_1,N_2]\}\end{aligned} \qquad (4.4.3)$$

where h(t) is finite impulse response coefficient sequence of the process model and is given

$$h(t) = [H(t)\Phi(t)^{j-1}\Lambda(t) \quad H(t)\Phi(t)^{j-2}\Lambda(t)\ \dots\ H(t)\Lambda(t)]$$

The future input sequence $\{U(t+j),\ j\in[N_1,N_2]\}$ can be written in incremental form with respect to the input U(t-1), ie.

$$\Delta U(t+i) = U(t+i) - U(t-1),\ i=0,\ 1,\ 2\dots$$

Therefore, the prediction equation (4.4.2) can be written as

$$\begin{aligned}\langle Y(t+j|t),\ j\in[N_1,N_2]\rangle = {} & \langle H(t)\Phi(t)^j\hat{X}(t|t),\ j\in[N_1,N_2]\rangle + \\ & h(t)\underline{U}(t-1) + \\ & h(t)\langle \Delta U(t+i-1),\ i\in[1,N_u]\rangle \end{aligned} \qquad (4.4.4)$$

where $\underline{U}(t-1)^t = [U(t-1)\ U(t-1)\ \ldots\ U(t-1)]$

and $\Delta U(t+i-1) = 0 \quad \text{for } i>Nu$

The future predicted trajectory is constructed by three components as shown in Equation (4.4.4). The first term on the RHS (ie. $\langle H(t)\Phi(t)^j\hat{X}(t|t), j\in[N_1, N_2]\rangle$) defines the prediction from the model (state projection). The second term (ie. $h(t)\underline{U}(t-1)$) defines the adjustment to the prediction due to the most previous actual measurement of the input U(t-1). The third term (ie. $h(t)\langle \Delta U(t+i-1), i\in[1,N_u]\rangle$) is the prediction adjustment due to the future input increments. Notice that the future input incremental sequence is unknown and the objective of the controller is to determine at every sampling instant the 'best' sequence to satisfy the controller design requirement in Equation (4.4.1). Furthermore, $\Delta U(t+i+1)$ is set to zero whenever i is greater than the control horizon N_u. This means that the controller is only allowed N_u number of moves to achieve the control objective.

The multistep prediction defined in Equation (4.4.4) can be reexpressed in vector notation:

$$\underline{Y}(t) = \underline{Y}^*(t) + G(t)\Delta\underline{U}(t) \qquad (4.4.5)$$

where

$$\begin{aligned}\underline{Y}(t) &= \langle Y(t+j|t),\ j\in[N_1,N_2]\rangle, \\ \underline{Y}^*(t) &= \langle Y^*(t+j|t),\ j\in[N_1,N_2]\rangle \\ &= \langle H(t)\Phi(t)^j\hat{X}(t|t),\ j\in[N_1,N_2]\rangle + \end{aligned}$$

$$\begin{bmatrix} H(t)\Phi(t)^{N1-1}\Lambda(t) + \ldots + H(t)\Lambda(t) & 0 \\ \vdots & \ddots \\ H(t)\Phi(t)^{N2-1}\Lambda(t) + \ldots + \ldots + H(t)\Lambda(t) \end{bmatrix}_{(N2-N1+1)x1} U(t-1), \qquad (4.4.6)$$

$$G(t) = G'(t)S \qquad (4.4.7)$$

where

$$G'(t) = \begin{bmatrix} H\Phi^{N1-1}\Lambda & H\Phi^{N1-2}\Lambda & \ldots & H\Lambda & 0 & \ldots & 0 \\ H\Phi^{N1}\Lambda & H\Phi^{N1-1}\Lambda & \ldots & & H\Lambda & 0 \ldots & 0 \\ \vdots & \vdots & & & & \ddots & \vdots \\ H\Phi^{N2-1}\Lambda & H\Phi^{N2-2}\Lambda & & \ldots & & & H\Lambda \end{bmatrix}_{(N2-N1+1)xN2} \qquad (4.4.8)$$

and

$$S = \begin{bmatrix} 1 & 0 & \ldots & 0 \\ 1 & 1 & \ldots & \\ 1 & 1 & & 0 \\ \vdots & & \ddots & \vdots \\ 1 & 1 & \ldots & 1 \end{bmatrix}_{N2xNu} \qquad (4.4.9)$$

and

$\Delta\underline{U}(t) = \{\Delta U(t+i-1),\ i\in[1,Nu]\}.$

To obtain the multistep control law, the cost functional in Equation (4.4.1) is rewritten in vector notation and is minimized with respect to $\Delta\underline{U}(t)$:

$$J_{MAPC} = [\underline{Y}_{sp}(t)-\underline{Y}(t)]^t\underline{\underline{\Gamma}}_Y[\underline{Y}_{sp}(t)-\underline{Y}(t)] + \Delta\underline{U}(t)^t\underline{\underline{\Gamma}}_U\Delta\underline{U}(t) \qquad (4.4.10)$$

where $\underline{\underline{\Gamma}}_Y = \begin{bmatrix} \gamma_{YN1} & \cdots & 0 \\ & \ddots & \\ 0 & \cdots & \gamma_{YN2} \end{bmatrix}$

$$\underline{\underline{\Gamma}}_U = \begin{bmatrix} \gamma_{UN1} & \cdots & 0 \\ & \ddots & \\ 0 & \cdots & \gamma_{UN2} \end{bmatrix}$$

$$\frac{\partial J}{\partial \underline{U}} = \left(-\frac{\partial \underline{Y}(t)}{\partial \Delta \underline{U}(t)}\right)^t \underline{\underline{\Gamma}}_Y [\underline{Y}_{sp}(t) - \underline{Y}(t)] + [\underline{Y}_{sp}(t) - \underline{Y}(t)]^t \underline{\underline{\Gamma}}_Y \left(-\frac{\partial \underline{Y}(t)}{\partial \Delta \underline{U}(t)}\right) +$$

$$\left(\frac{\partial \Delta \underline{U}(t)}{\partial \Delta \underline{U}(t)}\right)^t \underline{\underline{\Gamma}}_U \Delta \underline{U}(t) + \Delta \underline{U}(t)^t \underline{\underline{\Gamma}}_U \left(\frac{\partial \Delta \underline{U}(t)}{\partial \Delta \underline{U}(t)}\right) \qquad (4.4.11)$$

Since from Equation (4.4.5), $\dfrac{\partial \underline{Y}(t)}{\partial \Delta \underline{U}(t)} = G(t)$, the controller law is obtained by setting Equation (4.4.11) to zero:

$$-G(t)\underline{\underline{\Gamma}}_Y [\underline{Y}_{sp}(t) - \underline{Y}^*(t) - G(t)\Delta \underline{U}(t)] + \underline{\underline{\Gamma}}_U \Delta \underline{U}(t) = 0$$

or

$$\Delta \underline{U}(t) = \left(G(t)^t \underline{\underline{\Gamma}}_Y G(t) + \underline{\underline{\Gamma}}_U\right)^{-1} G(t)\underline{\underline{\Gamma}}_Y [\underline{Y}_{sp}(t) - \underline{Y}^*(t)] \qquad (4.4.12)$$

At every sampling instant, the future trajectory of the process is obtained from Equation (4.4.5). The control action can then be obtained from Equation (4.4.12). Since the MAPC is implemented in a receding horizon strategy, only the first row of $(G(t)^t \underline{\underline{\Gamma}}_Y G(t) + \underline{\underline{\Gamma}}_U)^{-1} G(t)\underline{\underline{\Gamma}}_Y$ is required.

4.4.2 Feedforward Control

Feedforward control can be easily implemented by augmenting the state-space model with other inputs, ie. Equations (4.2.1) and (4.2.2) becomes

$$X(t+1) = \Phi\, X(t) + \Lambda(t)\, U(t) + \Xi\, \eta\iota(t) \qquad (4.4.13)$$

$$Y(t) = H\,X(t) + \eta_2(t) \qquad (4.4.14)$$

where

$$\Lambda(t)\ U(t) = \Lambda_1(t)U_1(t)+\Lambda_2(t)U_2(t)+\ldots+\Lambda_m(t)U_m(t)$$

$\Lambda(t)$ is the input matrix of dimension $(n+d)\times m$, and $U(t)$ is the input vector of dimension $m\times1$. $\Lambda_1(t)$, $\Lambda_2(t)$ ..., $\Lambda_m(t)$ are updated on-line by the parameter estimator. The state vector $X(t)$ is updated by the Kalman Filter defined in Equations (4.2.5) to (4.2.9).

The control law is identical to Equation (4.4.12) except that the $\underline{Y}^{*}(t)$ in Equation (4.4.6) is updated with the multidimensional input matrix B and the input vector $U(t)$. $G(t)$ remains unchanged because it defines the control scheme in the feedback path.

It is well known from classical control design that for a feedforward controller to be realizable, the time delay of the process model has to be equal or smaller than the time delay of the load transfer function.

$$G_{ff}(s)G(s)e^{-ps} = -G_d(s)e^{-ds} \qquad (4.4.15)$$

where

$G_{ff}(s)$ is the feedforward controller,
$G(s)\ e^{-ps}$ is the process model with p as the model time-delay,
$G_d(s)\ e^{-ds}$ is the disturbance model with d as the model time delay.

Rearranging Equation (4.4.15) gives the feedforward controller:

$$G_{ff}(s) = -\frac{G(s)}{G_p(s)}\,e^{-(d-p)s} \qquad (4.4.16)$$

It is clear from Equation (4.5.2) that in order for $G_{ff}(s)$ to be realizable, the process model time-delay should be at least as big as the disturbance model time-delay. Therefore, this serves as a design criterion for the feedforward controller.

4.4.3 Some tuning guidelines for MAPC in the presence of time delay mismatch

The multistep control scheme appears to be very flexible and can be tailored to tackle different control applications. However, flexibility always implies complication. There are essentially five controller parameters that can be used for tuning: N_1, N_2, N_u, $\underline{\Gamma}_Y$, and $\underline{\Gamma}_U$. Many authors have given detailed suggestions on how these parameters should be selected for different applications (Clarke et al. 1987, McIntosh, 1988). For example, the 'default settings' suggested by Clarke et al. (1987) that can handle various situations greatly simplify the use of the controller.

It is of great interest here to converge quickly to the selection of those parameters that are able to best handle the time delay mismatch cases. The output horizon N_2 is a good candidate for such application. When the process time delay is uncertain, a large value of N_2 is recommended. It was mentioned by McIntosh (1988) that as $N_2 \to \infty$ and $N_1=1$, $N_u=1$, $\underline{\Gamma}_U=0$, and $\underline{\Gamma}_Y=1$, the controller will provide a single control step so as to reduce the error at steady state to zero. This is called the 'mean-level' controller and it behaves conservatively. Usually N_2 is chosen such that $d_{max}+1<N_2 \leq t_s$, where d_{max} is the upper bound for the process time delay and t_s is the settling time of the process in sampling intervals (including time delay), while $N_1=1$, $N_u=1$, $\underline{\Gamma}_U=0$, and $\underline{\Gamma}_Y=1$.

Another effective tuning parameter is R_2/R_1 in the MKFP. When the process time delay is uncertain, a suitably large value of R_2/R_1 guarantees performance and robustness of the MKFP (see section 4.2). However, when the MKFP is cascaded to a controller for feedback control, R_2/R_1 cannot be unlimitedly high because of two reasons: (1) the presence of a low-pass filter in the feedback path 'desensitizes' the controller towards disturbances. A smaller R_2/R_1 reduces the low-pass characteristic of the filter (the band-width of the filter increases) and makes the controller more alert to disturbance. (2) the sluggish predicted trajectory resulted from an over size R_2/R_1 makes servo control slow. Therefore, it is recommended to use a large R_2/R_1 (choose R_1 10^{-4} times of R_2) when commencing the controller for robustness and tune it down for better servo and regulatory control performance.

4.5 Experimental results

4.5.1 Process and Program description

Experiments were conducted to demonstrate the performance of MAPC on a process with unknown or time varying time delay. The experimental apparatus used is the pilot scale Continuous Stirred Tank Heater (CSTH). Details of the process are described in section 5.5.1.

MAPC is implemented as an Advanced Control Task (ACT) running under MULTICON, which is a task scheduling shell that runs under the QNX operating system (see chapter 5 for details on MULTICON and QNX OS). It consists of three tasks (initial settings for each task are listed in {}):

(1) parest :the on-line parameter estimator (see section 4.3).
{$P(0)=10I$; $\sigma=100.0$; $db_l=0.1$; $db_h=1000$; $a(0)$, and $b(0)$ are obtained from the user start-up file}

(2) kfp :the Modified Kalman Filter Predictor task that does the on-line state estimation.
{$P(0)=3I$; $L(0)=[0 \ \dots \ 0]$}

(3) mapc :the controller signal calculation step.
{N1=1, N2=10, Nu=1; $\Gamma_{\underline{Y}}=1$, $\Gamma_{\underline{U}}=0$}

The sampling time is chosen to be 8 seconds. The CSTH can be represented by a first order model plus time delay. However, the model order is chosen as six and the model time delay is chosen as one to allow enough b parameters for accommodating time delay variation between TC#1 to TC#3 (see Table 5.1 for different time delays associated with each thermocouple).

The parameters of the controller are set to their 'default' values (ie. N1=1, N2=10, Nu=1, $\Gamma_{\underline{Y}}=1$, and $\Gamma_{\underline{U}}=0$). The upper and lower deadband and σ used in the parameter estimator will not be changed throughout the experiment. The only tuning parameter used to achieve performance and maintain robustness under time delay mismatch is R_2/R_1 in the MKFP. Results for these runs are summarized in the next section.

4.5.2 Results: with no time delay mismatch

Servo Control

Thermocouple #1 is used for this set of experiments. With the chosen sampling period, the model delay exactly represents the process time delay (see Table 5.1 for the time delays associated with each thermocouple).

The time responses for servo control are shown in Figure 4.13a. After the parameters converge to some meaningful values (Figure 4.13b), the performance of the controller becomes very satisfactory. (Notice that it takes more than 70 cycles before the parameters of the process converge. The interaction between the Kalman Filter Predictor and the parameter estimator adds difficulties to the convergence. It is expected that convergence will be faster if good initial values are provided). With a small value of R_2/R_1 (300), the controller achieved almost deadbeat control (see period from 100 to 170). The R_2/R_1 ratio is stepped up to 3,000 and then to 30,000 by keeping $R_1=0.01$ and varying R_2 (from period 170 to 340). With a larger ratio, the servo response becomes more sluggish with more overshoot. The control action is very conservative compared with that of a small ratio. This is related to the sluggishness of the trajectory predicted by the MKFP as mentioned in section 4.2.4.

The R_2/R_1 ratio is dropped back to 3,000 and then 300 but this time by keeping $R_2=300$ and varying R_1 (from period 340 onward). This demonstrates that it is the ratio R_2/R_1 rather than the absolute values of R_2 and R_1 that determines the performance of the controller. As can be seen, the responses from period 100 to 170 are almost identical with that from 400 to 470 ($R_2/R_1=300$) and the responses from period 170 to 240 are almost identical with that from 330 to 420 ($R_2/R_1=3{,}000$). The plot of the Kalman gain in Figure 4.14 illustrates this point more clearly. The gain approaches similar values when the ratios are the same. However, the convergence rate of the gain appears to be affected by the absolute values of R_2 and R_1: a big R_2 seems to slow convergence down (compare the Kalman gain from 150 to 250 and from 320 to 380). Another effect of using a large R_2/R_1 ratio is that the values of the Kalman gain become very small. This explains the filtering effect of the MKFP because the prediction update is more conservative when the gain gets smaller. This can be seen from the frequency response of the residual filter $G_f(z^{-1})$ in Figure 4.15.

Generally speaking, the filter is predominantly a low-order filter with the order equals 1.5. The low pass characteristic of the filter increases with the R_2/R_1 ratio. As the ratio increases, the band-width of the filter decreases which means more cut-off on lower frequency signals. This explains the sluggishness of the output response: some wanted frequency characteristics of the process are attenuated.

Figure 4.16 shows a comparison of the frequency responses between the $G_f(z^{-1})$ obtained from a second order T-filter and $G_f(z^{-1})$ by various R_2/R_1 of the MKFP. They exhibit similar low frequency characteristics (unity gain) but deviate greatly in the mid and high frequency domains. More importantly, the T-filter does not appear to be able to cut out high frequency noise. The difference is due to the fact that a high order A polynomial is being used ($\delta A=6$). To achieve similar performance, the order of the T-filter must be chosen such that $\delta T > \delta A$ and the roots of the filter chosen appropriately (see section 4.2.3 for some suggested guide-lines on how to choose $T(z^{-1})$). This may be difficult. The MKFP selects the filter directly.

Figure 4.13b suggests that the process can be sufficiently represented by a second order model since the tailing parameters of the $A(z^{-1})$ polynomial are near zero. The reason for using a high model order is due to the fact that the state space model in Equations (4.2.14, 15) requires the degree of $A(z^{-1})$ equals the degree of $B(z^{-1})$, and the $B(z^{-1})$ polynomial has to be over-parameterized to handle different values of the time delay. This adds strain to both the parameter estimator as well as the MKFP. It also makes the controller more prone to numerical errors. To alleviate this problem, the parameters from $\hat{a}_{na+1}$ to $\hat{a}_{nb}$ in $A(z^{-1})$ can be set to zero, that is, to use a lower order $A(z^{-1})$ polynomial and a high order $B(z^{-1})$ polynomial in the state space formulation. Walgama (1986) took advantage of the sparseness of the state matrix and proposed a 'speed-up' version of the MKFP. This approach cuts calculation steps down by an order of 10 times compared with the algorithmic approach. This implementation helps to reduce computation effort as well as numerical errors.

Regulatory Control

The experiment is repeated for regulatory control using the inlet water as disturbance. The output responses are shown in Figure 4.17. Disturbance rejection becomes more active as the R_2/R_1 ratio decreases. This result suggests that R_2/R_1 should not be chosen to be too large when the controller is predominantly used for regulatory control.

As mentioned in section 4.3.2, both servo and regulatory controls are affected by $G_f(z^{-1})$ only when there are MPM. If there is no MPM, $G_f(z^{-1})$ will affect regulatory control only. It is clear that the process model is only an approximation to the process since R_2/R_1 affects both modes of operation. Since some degree of MPM is bound to exist, the residual filter should be used all the time. However, it is seen that the output responses using large R_2/R_1 are sluggish for both servo and regulatory control. The controller becomes more alert when R_2/R_1 decreases. It can be concluded that whenever the user is confident about the time delay, i.e. the time delay mismatch is small, R_2/R_1 should be set to a small number.

4.5.3 Results: With Time Delay Mismatch

A sixth order model and a unit time delay is again used to represent the process. Time delay mismatch is introduced by switching to thermocouple #2 and #3 at some point in time (the time delays associated with these two thermocouples can be seen from Table 5.1). This introduces a mismatch (underestimation) of five to six sample periods.

The time response is shown in Figure 4.18 (the start-up portion is omitted in the plot). MAPC demonstrates good servo and regulatory control in spite of the drastic variation of the time delay. (1) Compare the responses from 100 to 250 and from 350 to 550: the performance of the control improves when a smaller R_2/R_1 ratio is used. For this amount of time delay mismatch, $R_2/R_1=30,000$ gives a better performance. (2) Compare the responses from 350 to 550 and from 550 to 800: when time delay mismatch is more severe, a larger R_2/R_1 ratio is used and the servo and regulatory responses are comparable to the less time delay case. These results suggest that whenever the time delay is uncertain, a large R_2/R_1 is always preferred.

Figure 4.19 plots the Kalman gain trajectories. Notice that at period

50, a setpoint change causes the gain to converge to a set of 'good' values. From then on, the gain update only responses to the change in the R_2/R_1 ratio.

Figure 4.20 shows the frequency response of the residual filter $G_f(z^{-1})$. Although it behaves as a low pass filter, its dynamic behavior is different from that for no time delay mismatch case in several aspects (for ease of referencing, let filter 1 be the filter for no time delay mismatch and filter 2 be the filter with time delay mismatch). Figure 4.21 shows a comparison of the two filters. Both have low pass characteristics and their dominant orders are very similar (about 1.2). However, their dynamic behaviors are very different. In particular, when $R_2/R_1=30,000$, the filter 1 shows a sharper cut off characteristics and its band-width is larger (at ≈ 0.3 rad $\sec^{-1}$ compared with ≈ 0.09 rad $\sec^{-1}$ for filter 2). Therefore, filter 1 retains middle to low frequency information, which may be essential for characterizing the process, better than filter 2. In fact, the frequency response curve of filter 2 with $R_2/R_1=30,000$ matches better with that of filter 1 with $R_2/R_1=3,000$. These observations lead to the conclusion that whenever the time delay mismatch is small, R_2/R_1 should be set to a small value so as to avoid over filtering.

4.6 Conclusion

The Multistep Adaptive Predictive Control (MAPC) consists of three major components that cooperate with each other to form a powerful control scheme: (1) an on-line parameter estimator; (2) the Modified Kalman Filter Predictor (MKFP) which predicts the process output trajectory into the future; and (3) a multistep controller design that calculates the control action based on a long term prediction (from the MKFP) of the process output.

The MAPC has been applied to a process with varying time delay and is found to perform very well. Experiments shows that with reasonable choice of sampling time and the process model order, the default controller settings ($N1=1$, $N2=10$, $NU=1$, $\underline{\underline{\Gamma}}_Y=1$, $\underline{\underline{\Gamma}}_U=0$) are sufficient for good control performance. This makes the controller very easy to use.

The robustness against time delay mismatch can be attributed to the MKFP, which gives 'good' future prediction regardless of the uncertain specification of time delay in the process model. The noise covariance ratio R_2/R_1 is a very important tuning knob to the predictor. It adjusts the properties of the residual filter to compensate for error due to time delay mismatch (disturbances and noises as well). A large value of R_2/R_1 should be used whenever the time delay of the process is uncertain. This way, the MKFP puts heavier filtering on the error and gives a smoother (but more sluggish) future output trajectory. Because the residual filter is designed 'optimally' by the MKFP (the order and the coefficients of the filter depend on the model parameters and the kalman gain), the user is left only with the choice of R_2/R_1 to alter the properties of the filter. This makes the MKFP very easy to use compared with the T-filter where the order and the coefficients of the filter have to be chosen.

Since tuning the predictor directly affect the closed-loop performance, the ratio, R_2/R_1, is an important performance tuning knob for the whole MAPC control scheme. From the experimental results, a large R_2/R_1 is needed whenever the time delay mismatch is large. This ensures the robustness of the controller and also gives 'good' servo and regulatory performances. From the experiment, R_2/R_1 in the order of 10^3 to 10^4 are found to give satisfactory performance when the mismatch is big (i.e.6 sample units). If R_2/R_1 is too small (e.g. 10^2), the controller becomes unstable. However, an over-sized R_2/R_1 worsen both modes of operation because of the sluggishness of the predicted output and the heavy filtering on the feedback signal. It is recommended that a large R_2/R_1 should be chosen at least in the order of 10^3 at the start for robustness and tuned down progressively for tighter performances.

The MAPC is implemented as an Advanced Control Task that runs under MULTICON. It is developed so that further investigation on the control scheme can be made. Full user documentation on the tasks can be found in Appendix 2C.

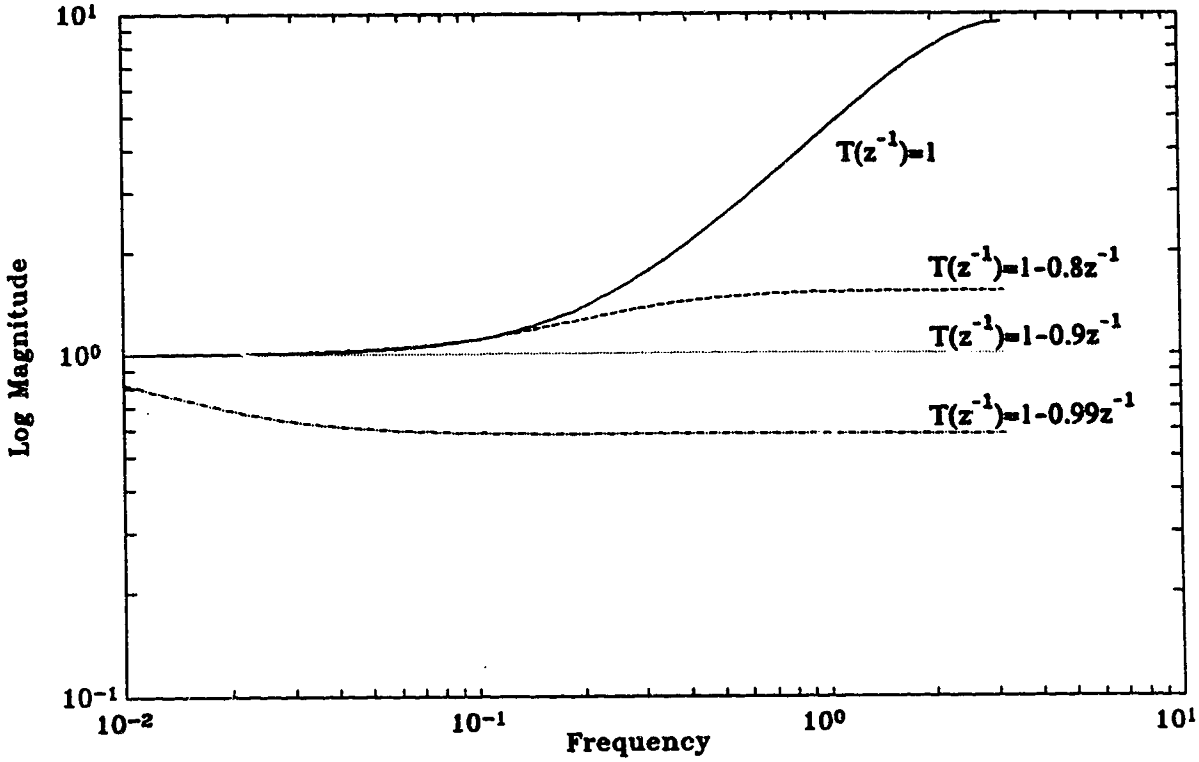


Figure 4.1a. Magnitude plot of the frequency response of $G_f(z^{-1})$ using a first order T-filter

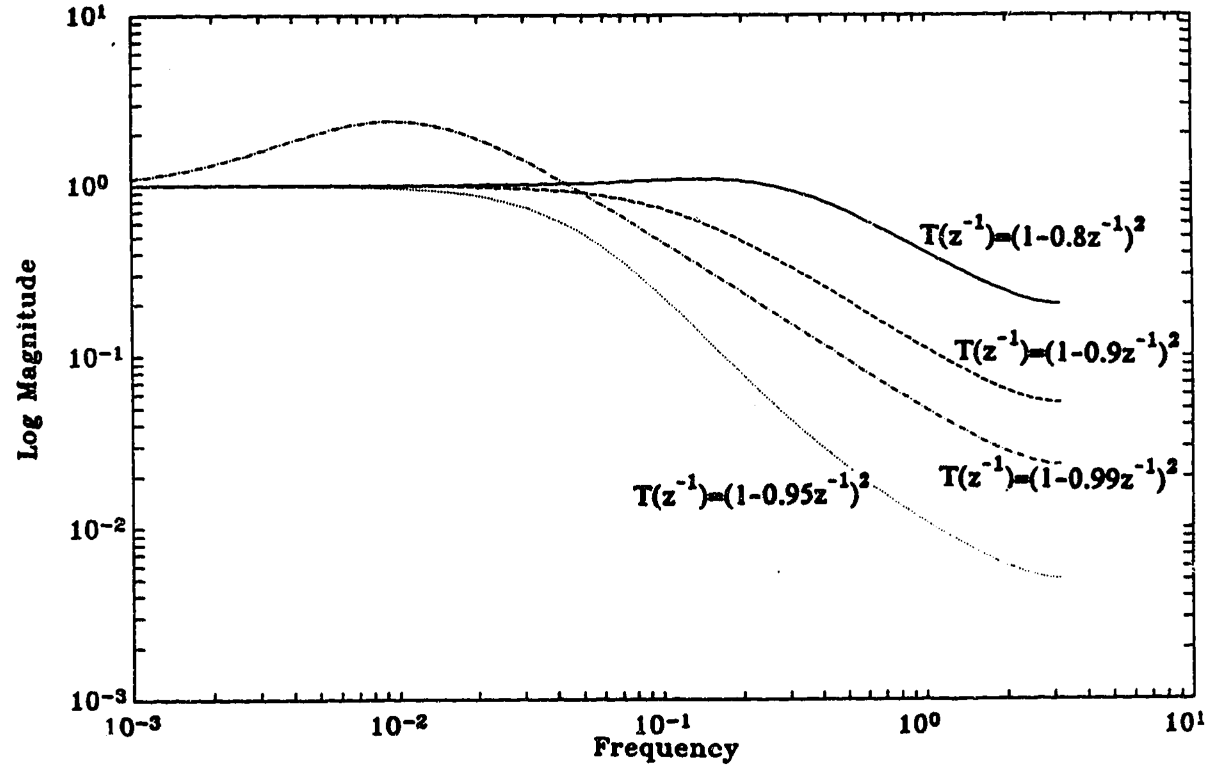


Figure 4.1b. Magnitude plot of the frequency response of $G_f(z^{-1})$ using a second order T-filter

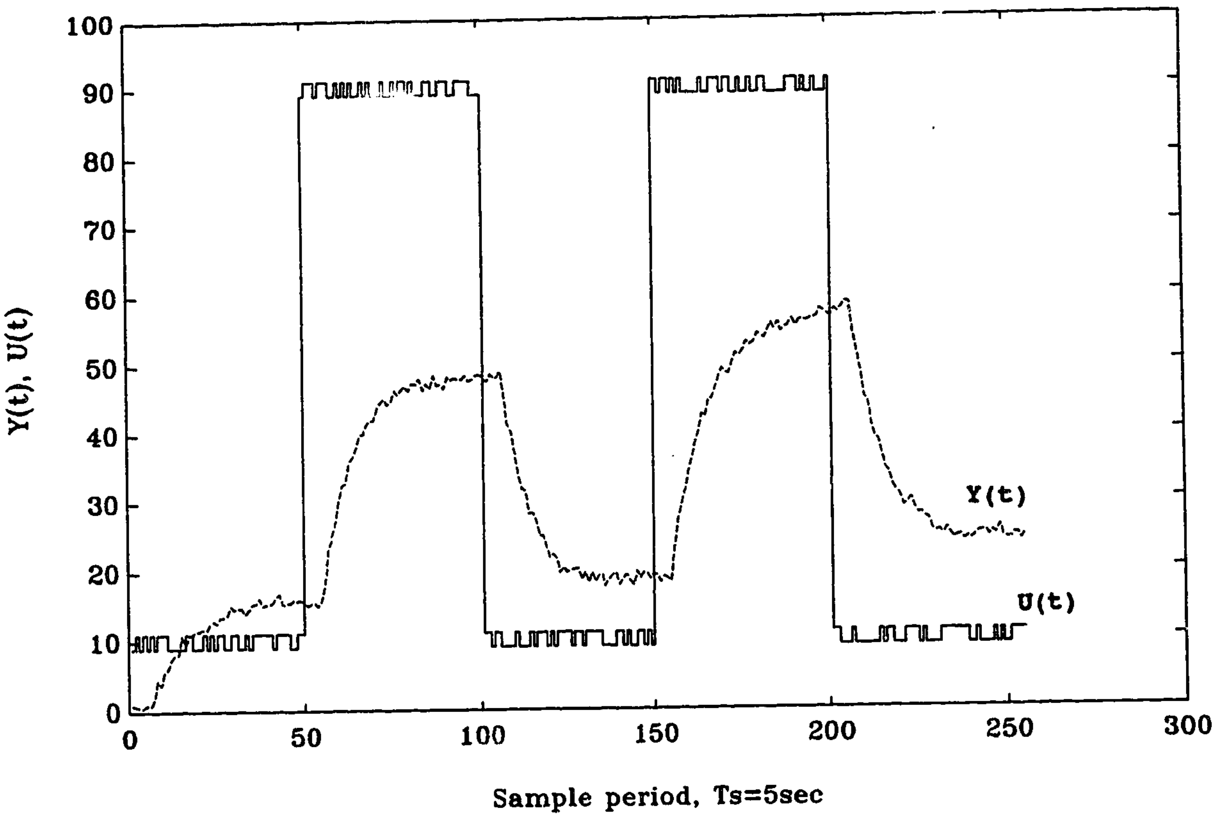


Figure 4.2. Open loop response of the simulation process

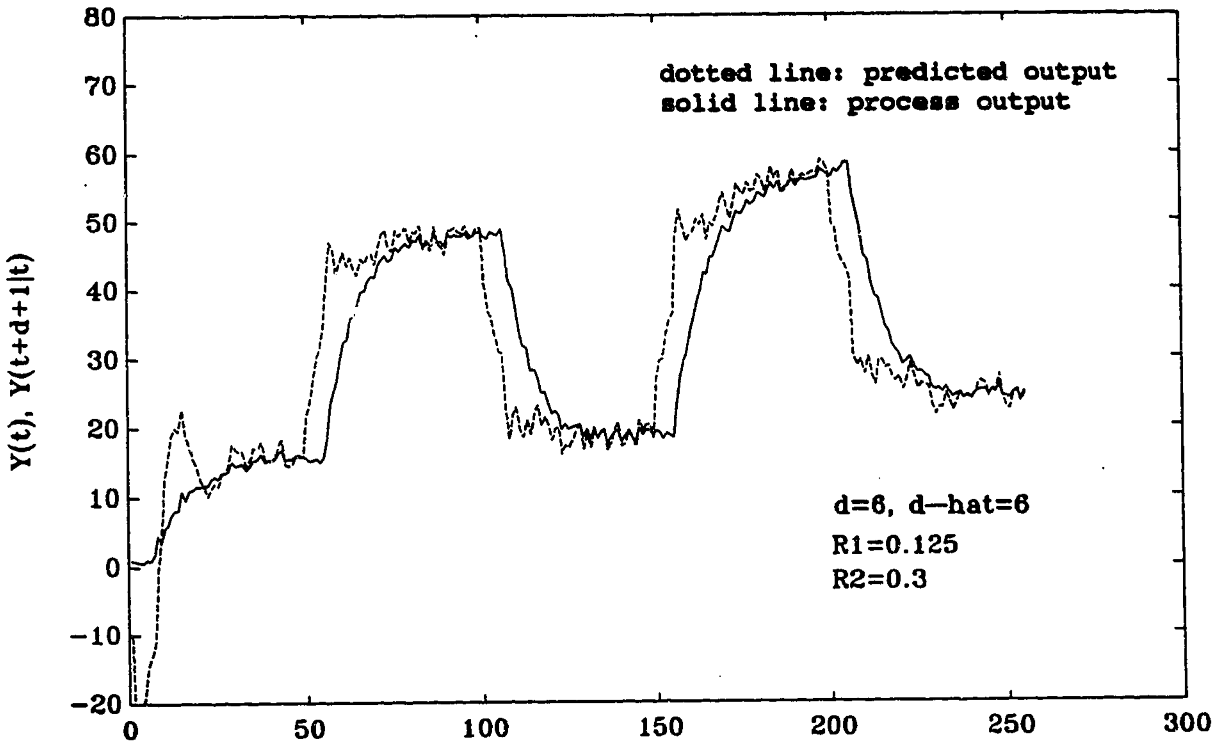


Figure 4.3a. Open loop prediction from the MKFP with no time delay mismatch (R_2=0.3, R_1=0.125)

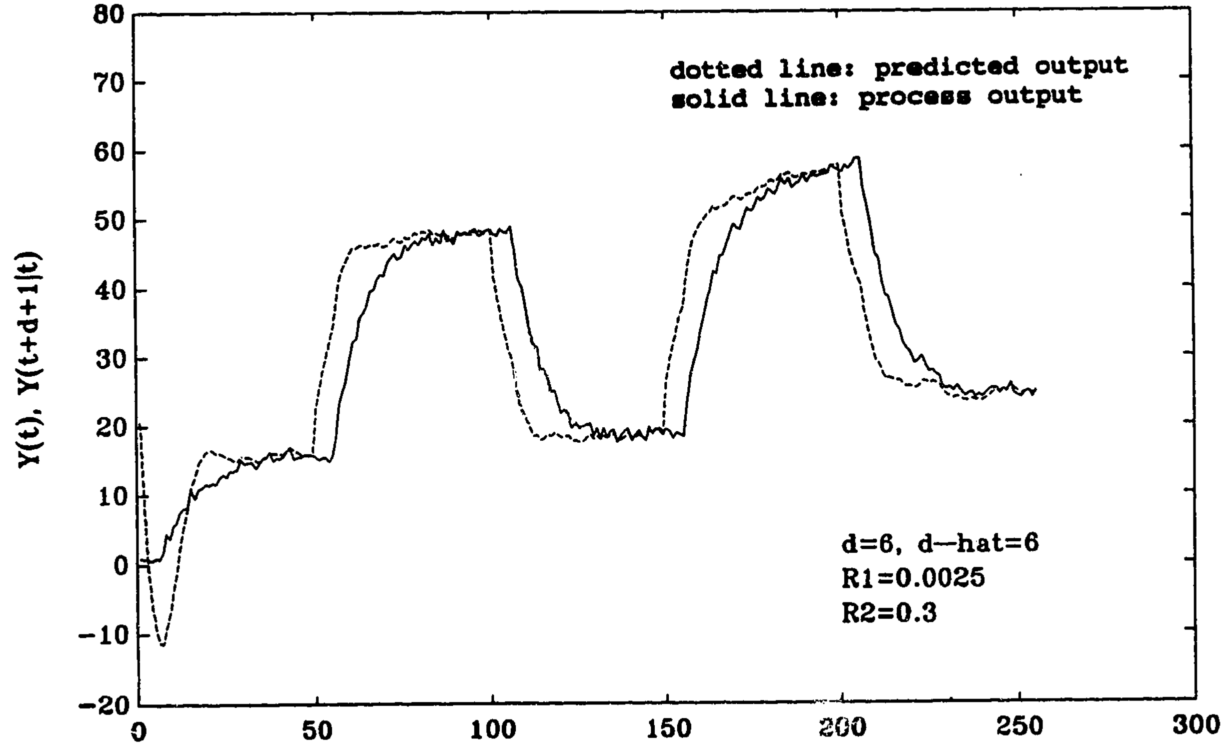


Figure 4.3b. Open loop prediction from the MKFP with no time delay mismatch (R_2=0.3, R_1=0.0025)

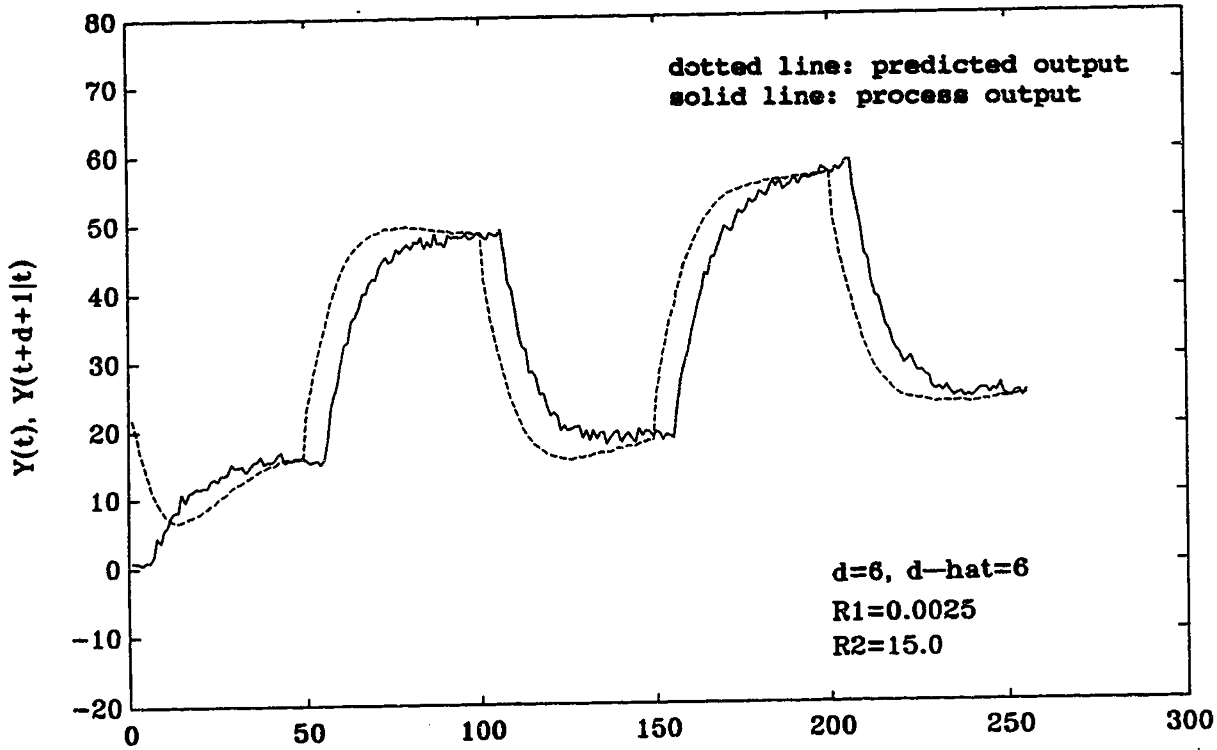


Figure 4.3c, Open loop prediction from the MKFP with no time delay mismatch (R_2=15.0, R_1=0.0025)

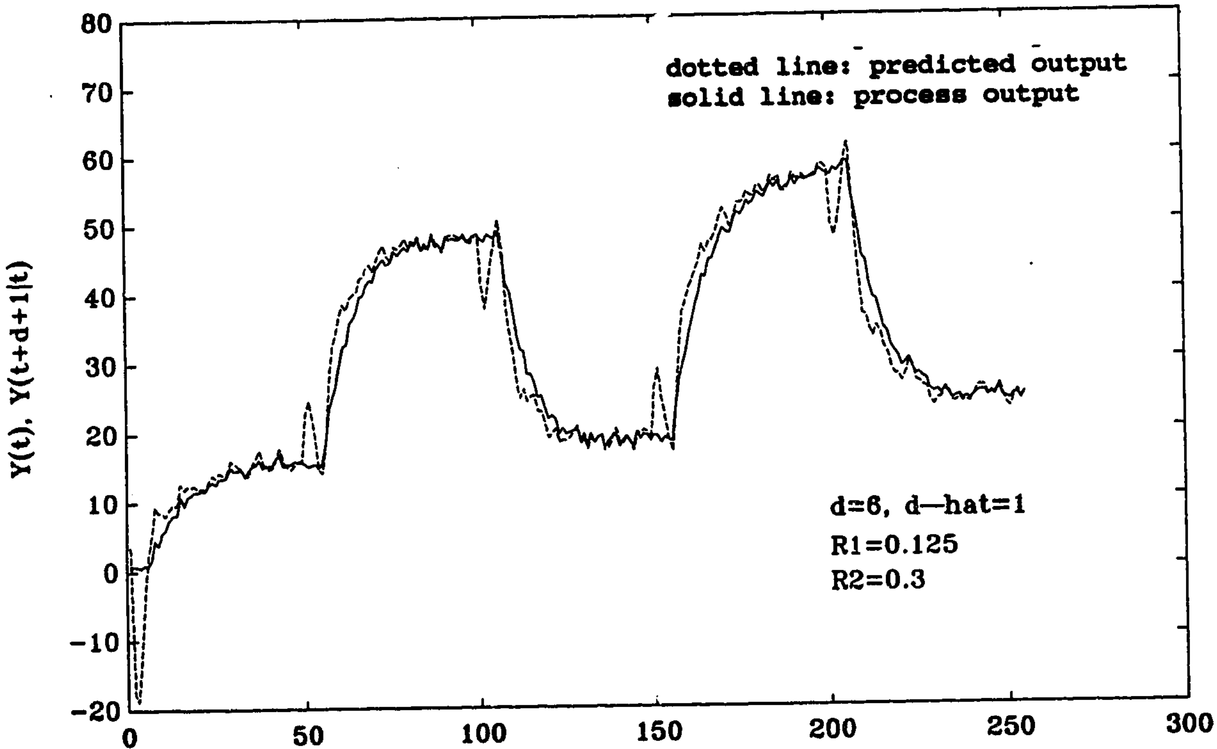


Figure 4.4a, Open loop prediction from the MKFP with time delay mismatch (R_2=0.3, R_1=0.125)

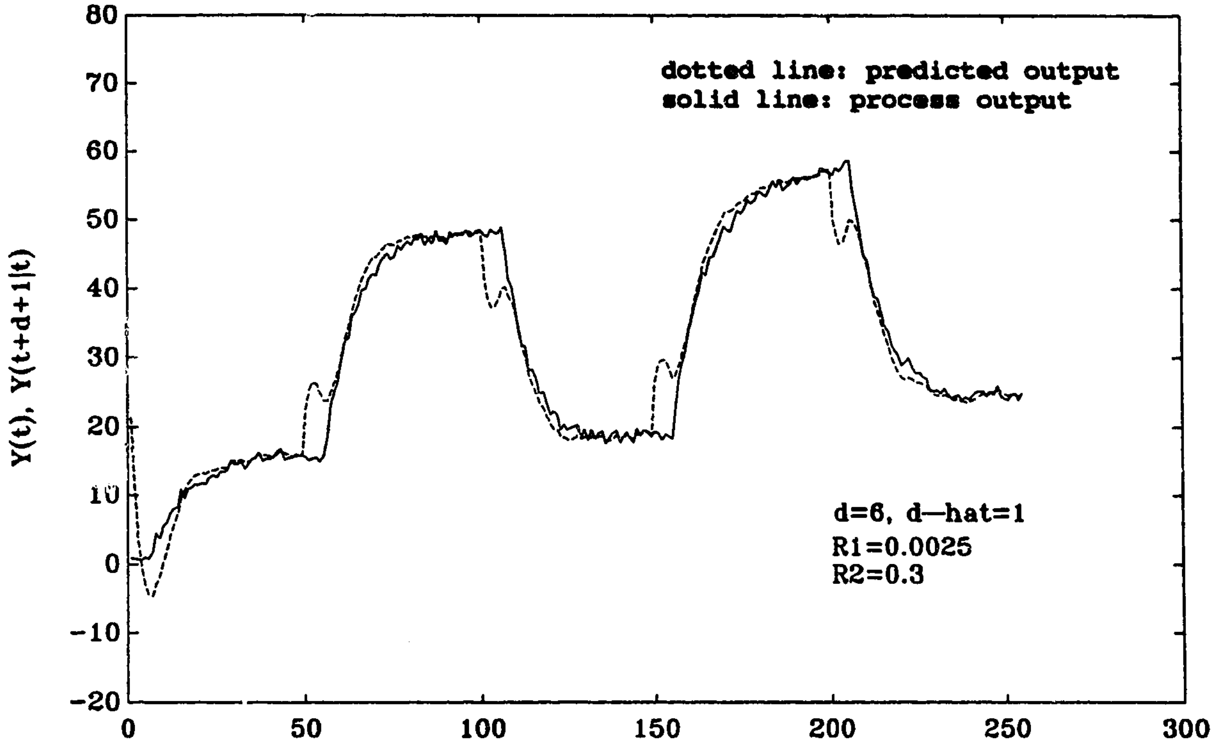


Figure 4.4b. Open loop prediction from the MKFP with time delay mismatch (R_2=0.3, R_1=0.0025)

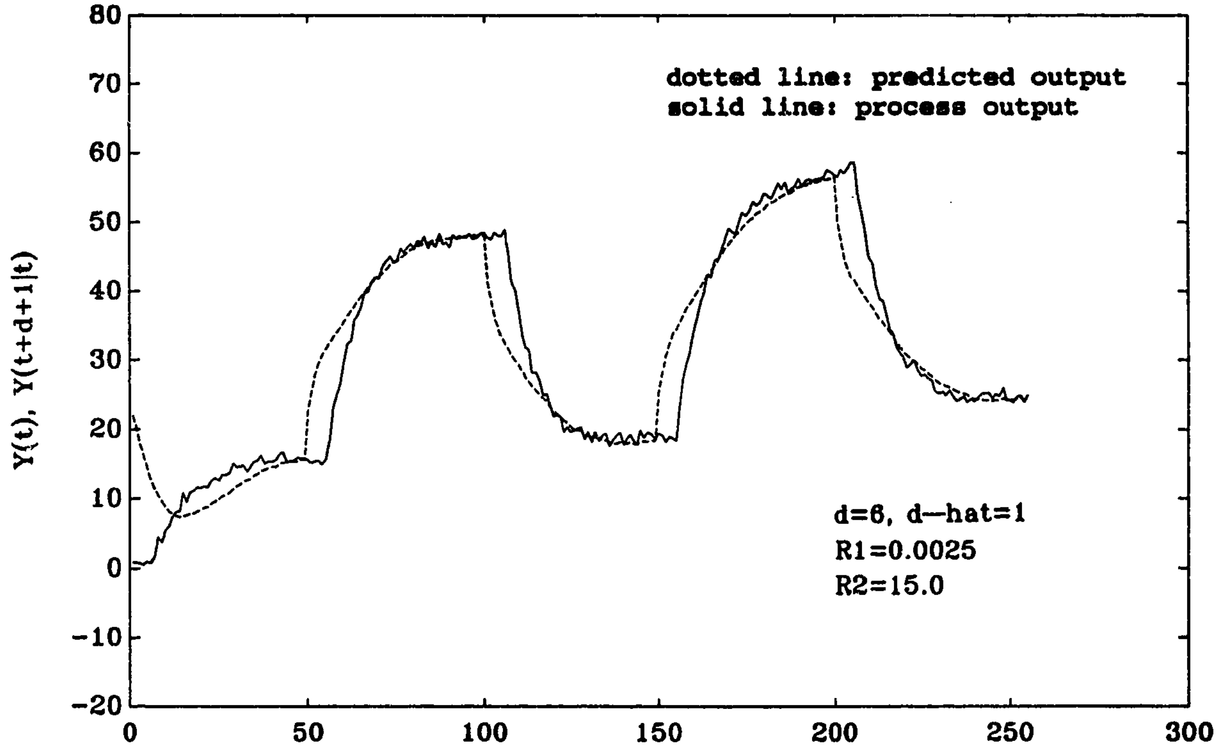


Figure 4.4c. Open loop prediction from the MKFP with time delay mismatch (R_2=15.0, R_1=0.0025)

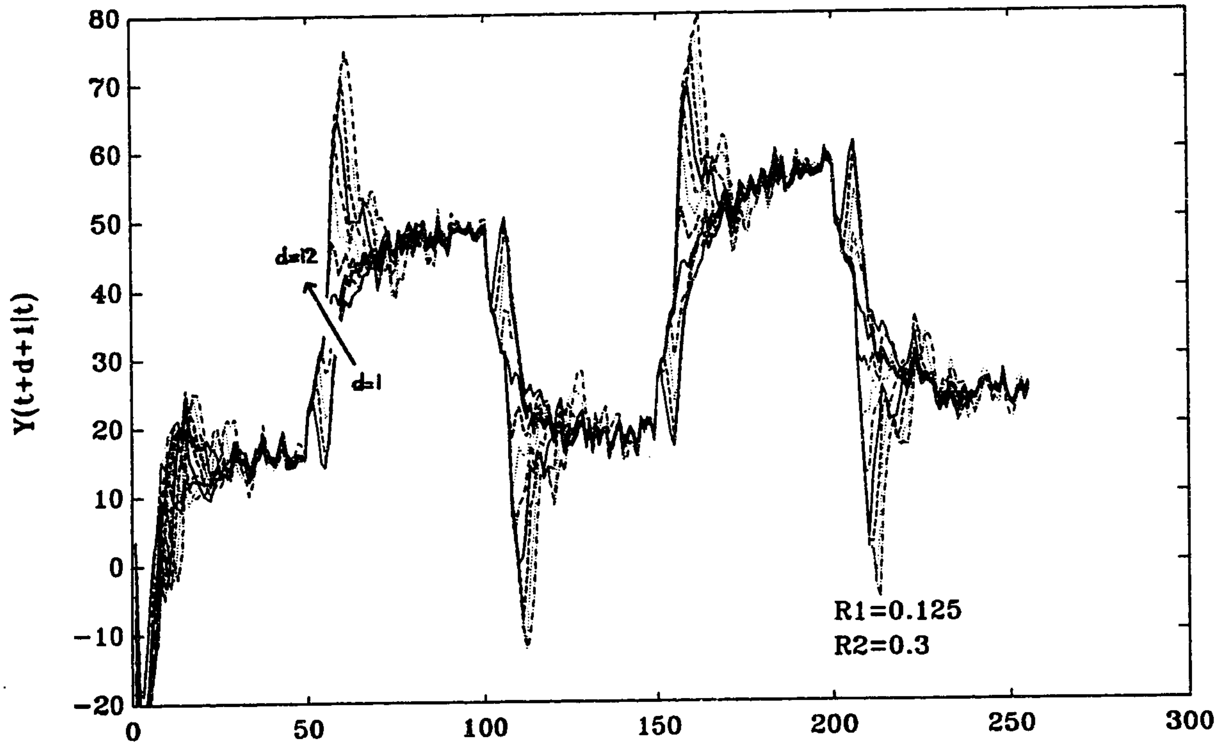


Figure 4.5a. Open loop prediction from the MKFP with model time delay varies from 1 to 12 (R2=0.3, R1=0.125)

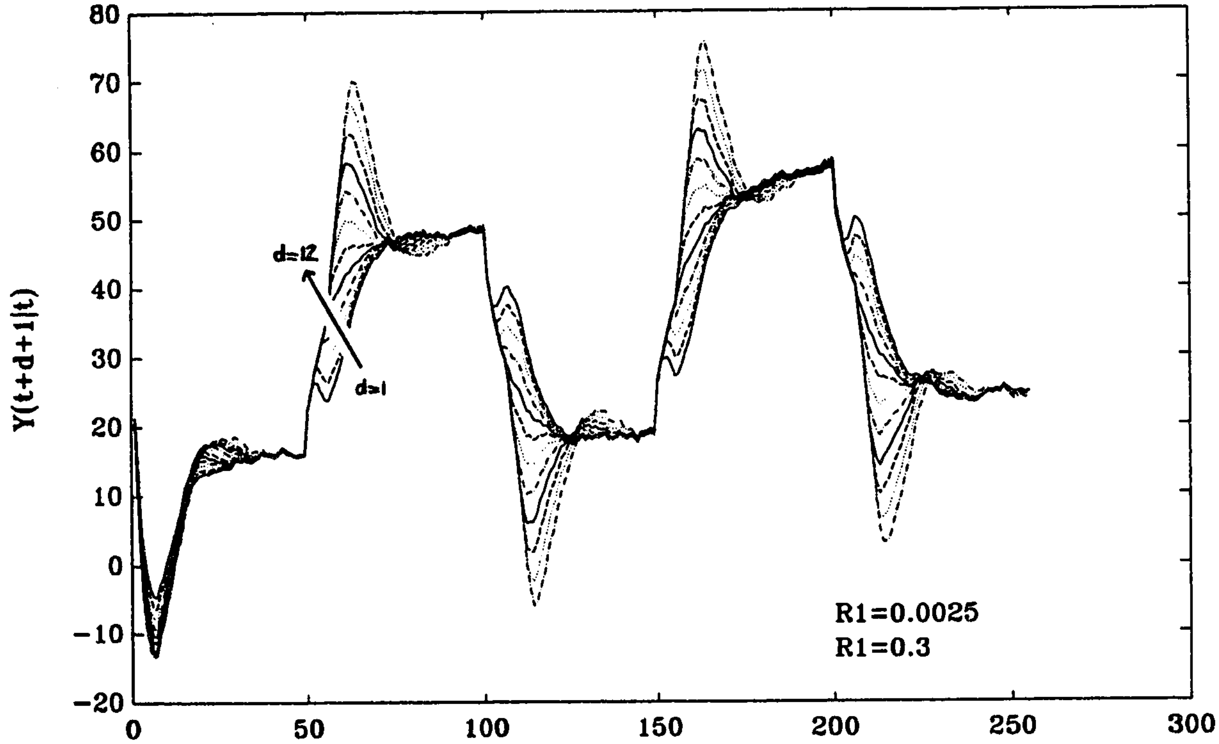


Figure 4.5b. Open loop prediction from the MKFP with model time delay varies from 1 to 12 (R2=0.3, R1=0.0025)

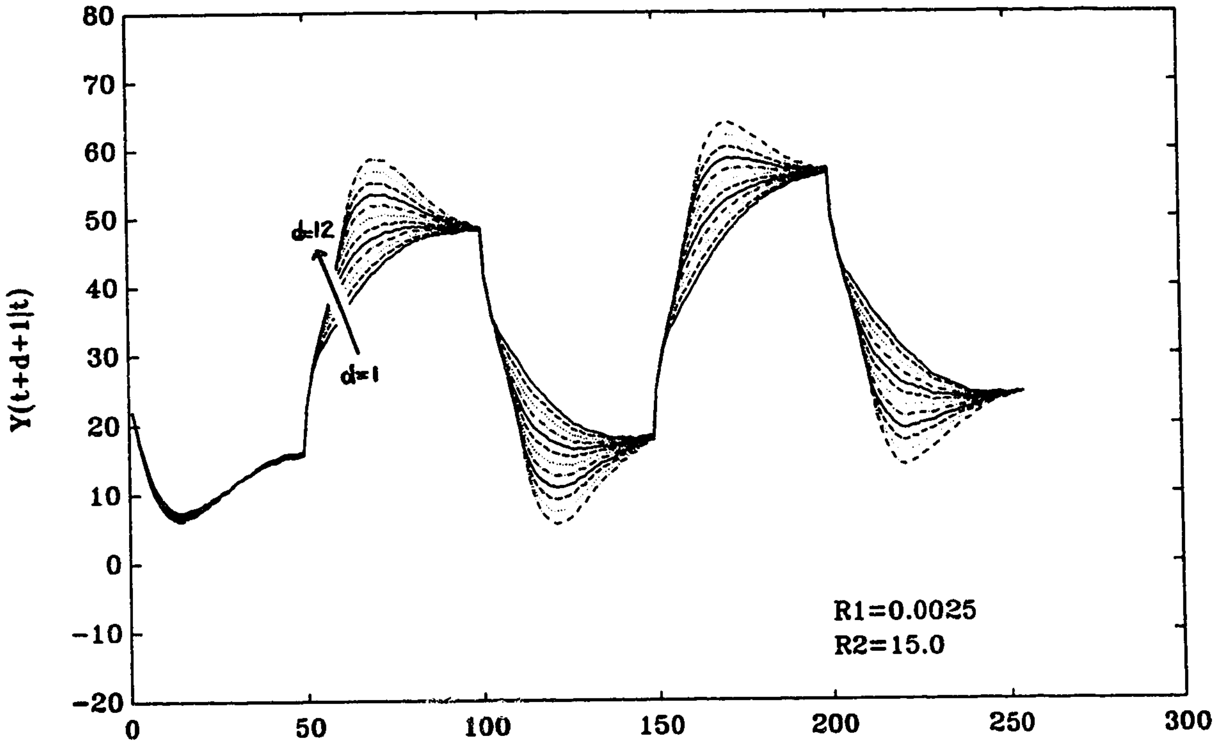


Figure 4.5c. Open loop prediction from the MKFP with model time delay varies from 1 to 12 (R2=15.0, R1=0.0025)

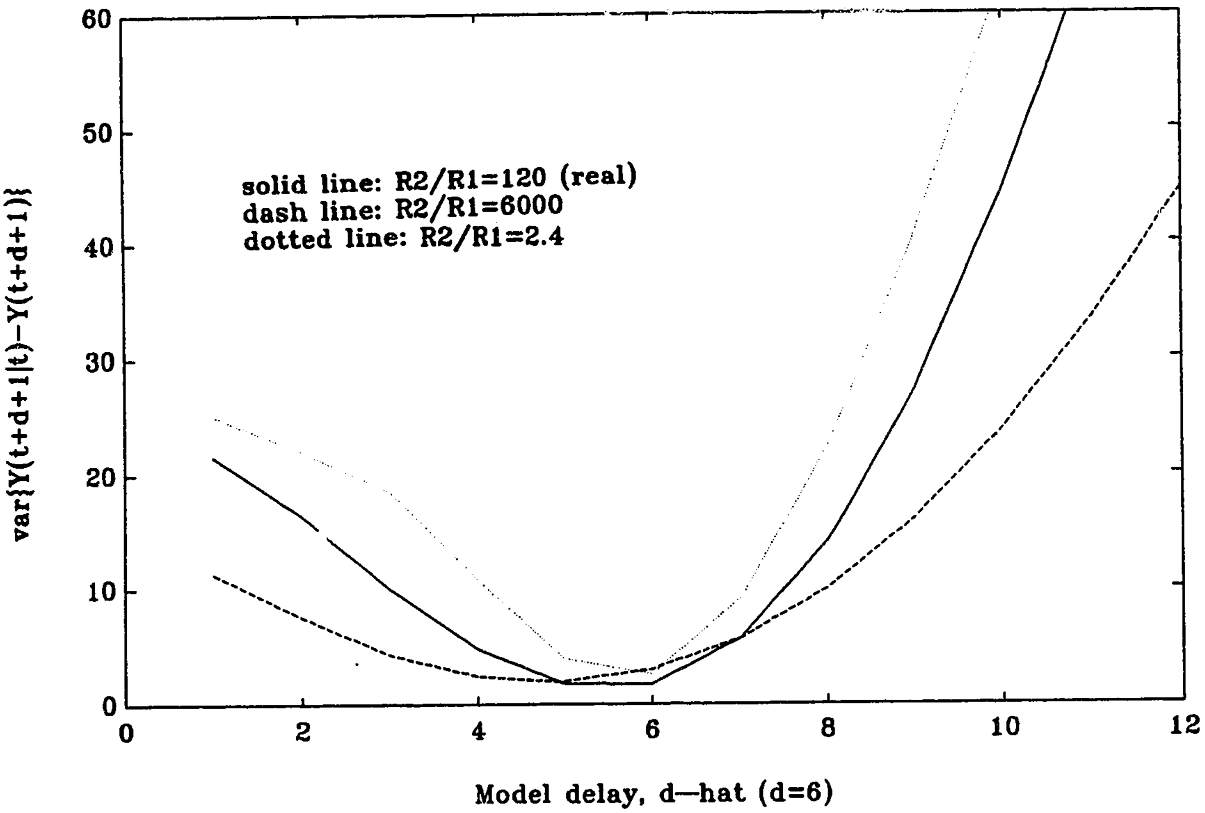


Figure 4.6. Variance of the difference of the predicted output and the process undelayed output with respect to the choice of the model time delay

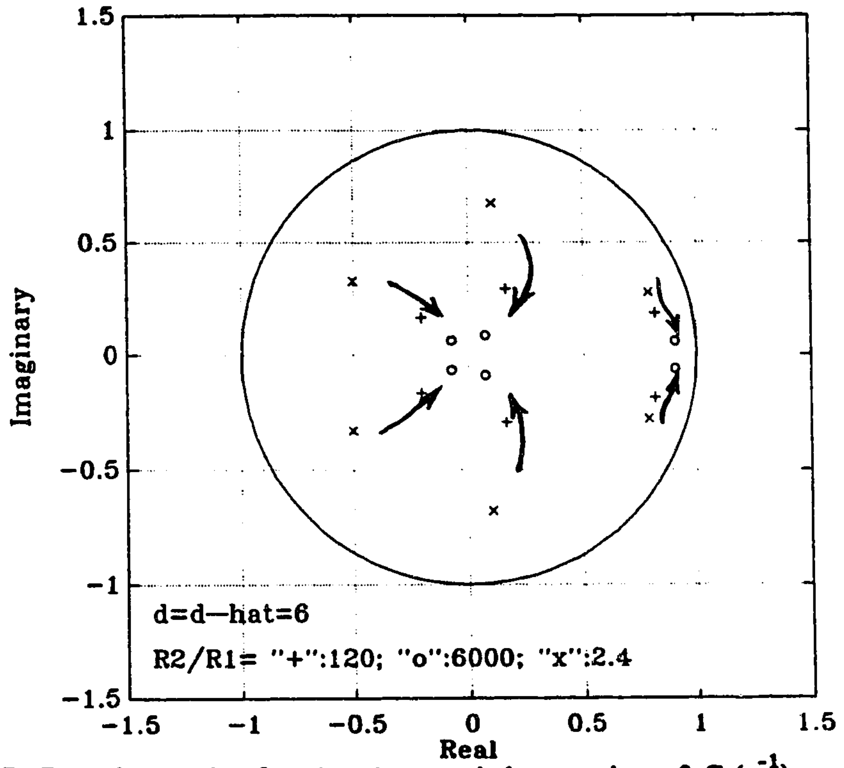


Figure 4.7. Root locus plot for the characteristic equation of $G_f(z^{-1})$ for no time delay mismatch case with respect to the choice of R_2/R_1

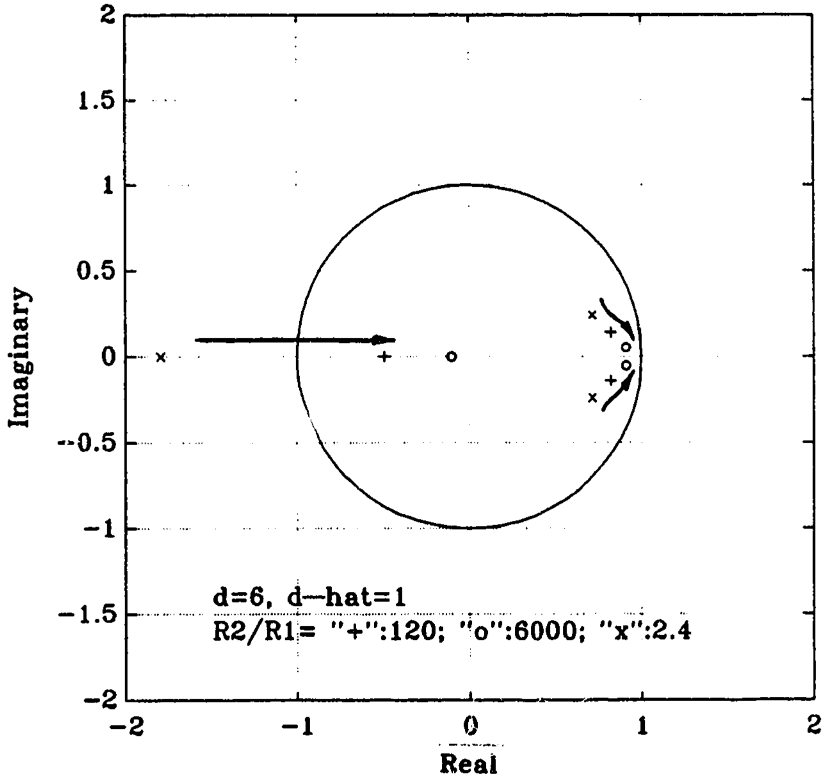


Figure 4.8. Root locus plot for the characteristic equation of $G_f(z^{-1})$ for time delay mismatch case with respect to the choice of R_2/R_1

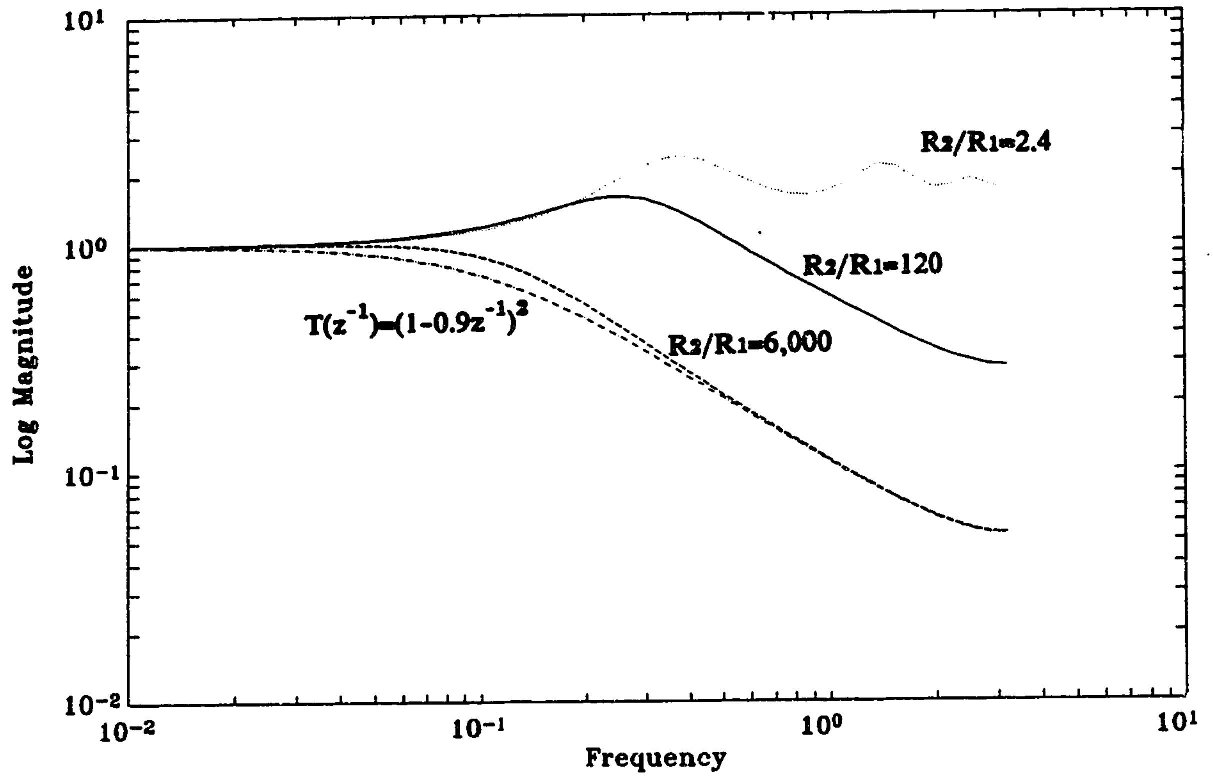


Figure 4.9. Magnitude plot of the frequency response of $G_f(z^{-1})$ with no time delay mismatch using different values of R_2/R_1

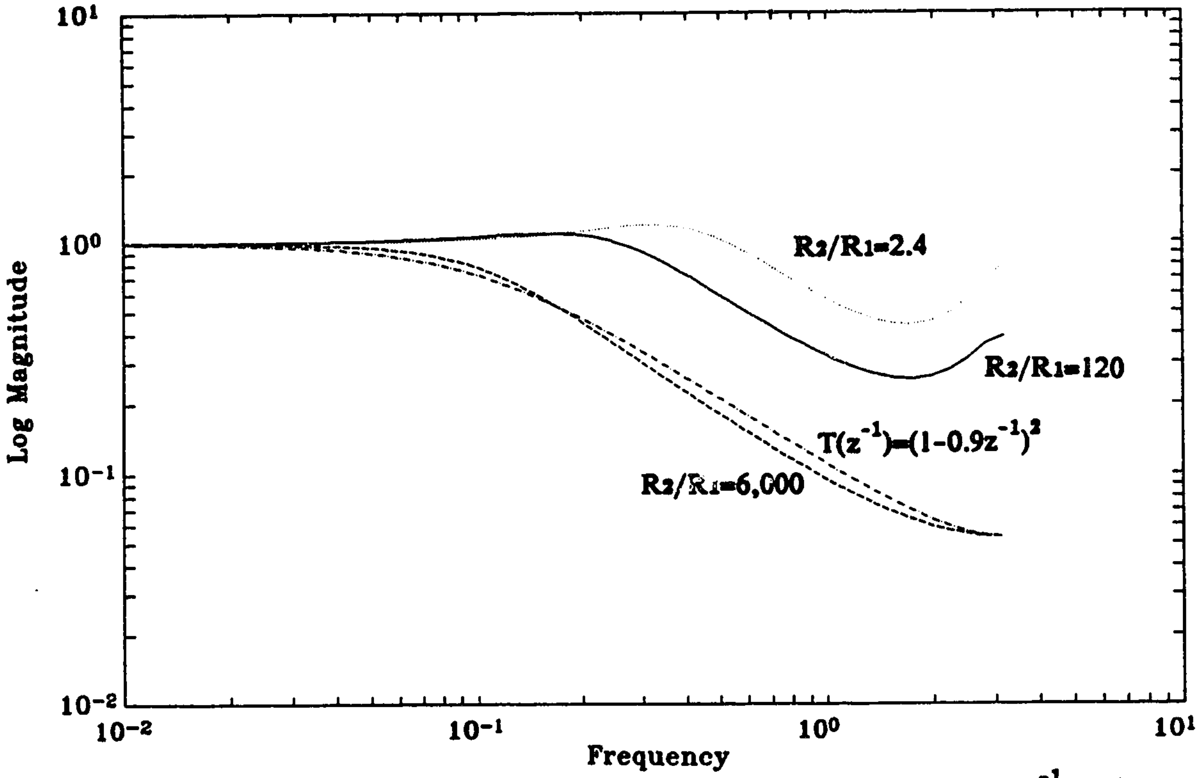


Figure 4.10. Magnitude plot of the frequency response of $G_f(z^{-1})$ with time delay mismatch using different values of R_2/R_1

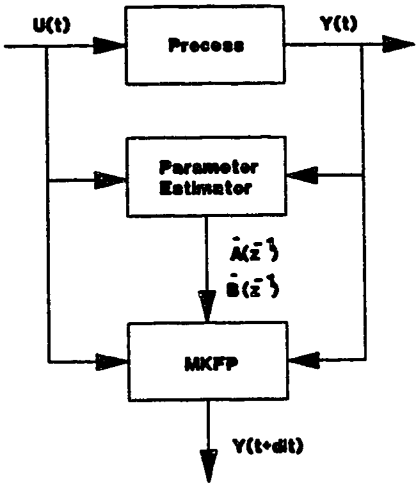


Figure 4.11. Structure of the Adaptive Modified Kalman Filter Predictor

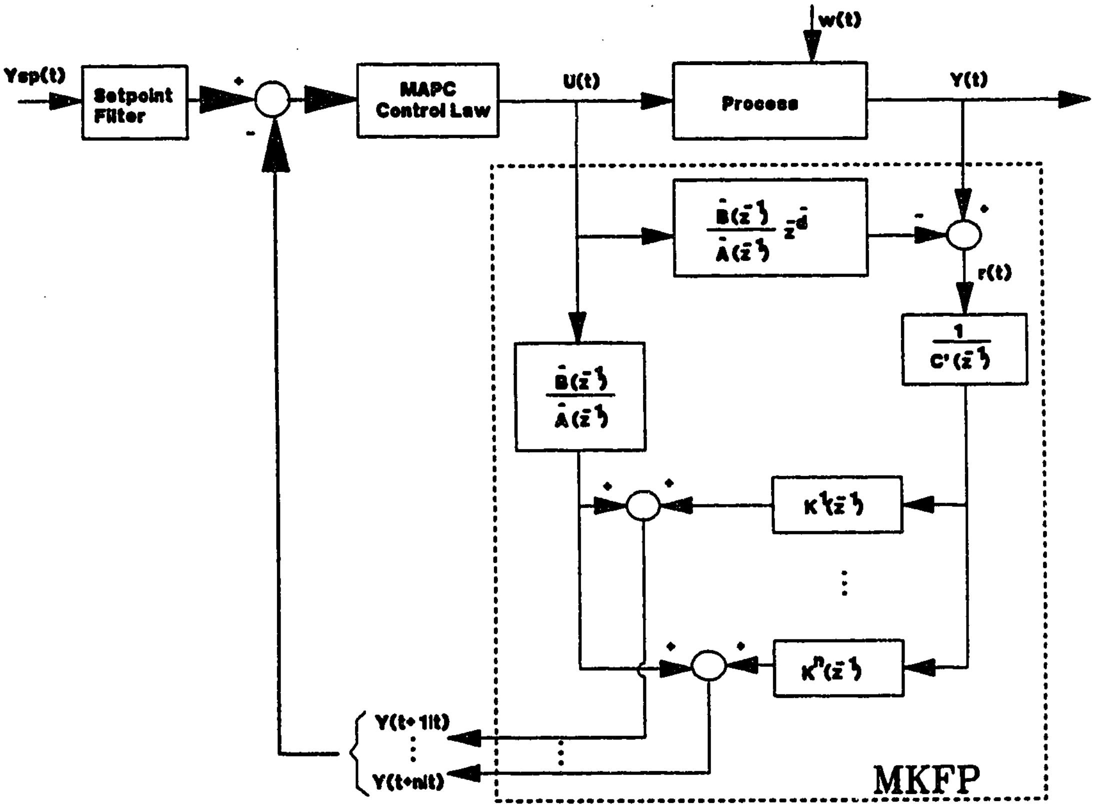


Figure 4.12. A network of parallel, optimal predictors obtained from the MKFP

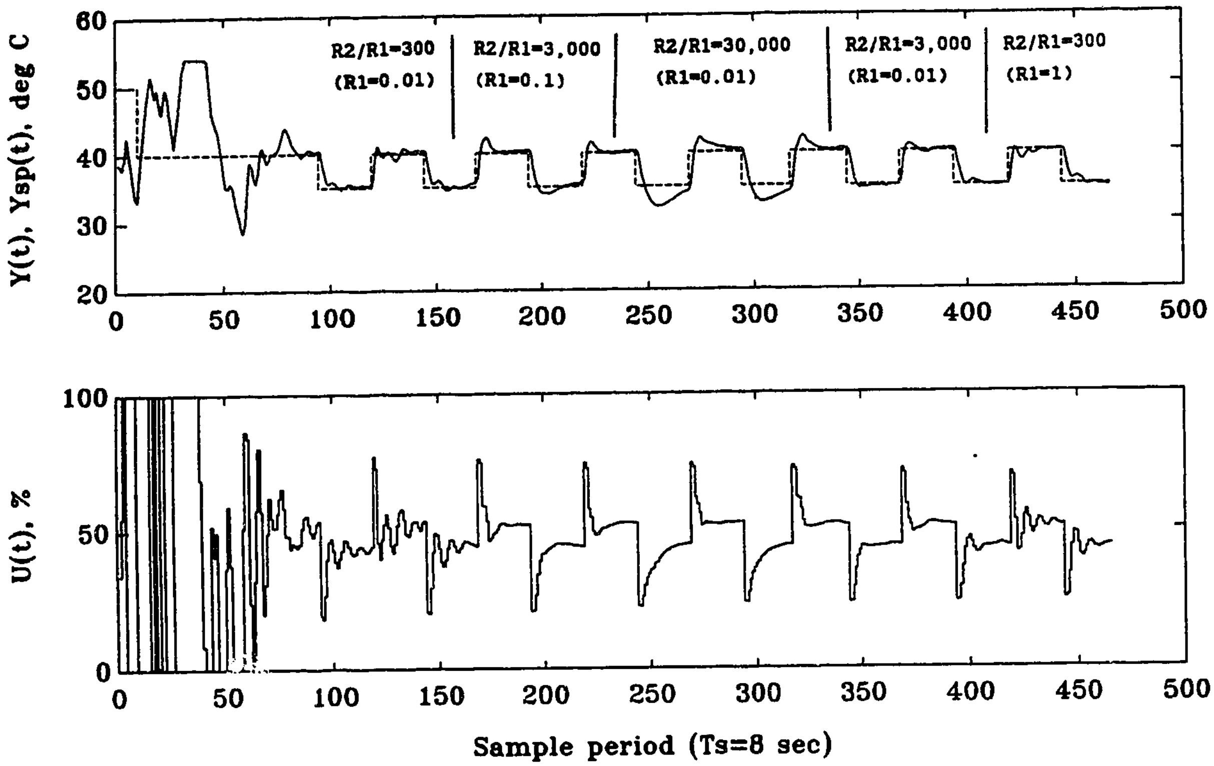


Figure 4.13a. Closed-loop response of MAPC on CSTH without time delay mismatch (TC#1) using different values of R_2/R_1

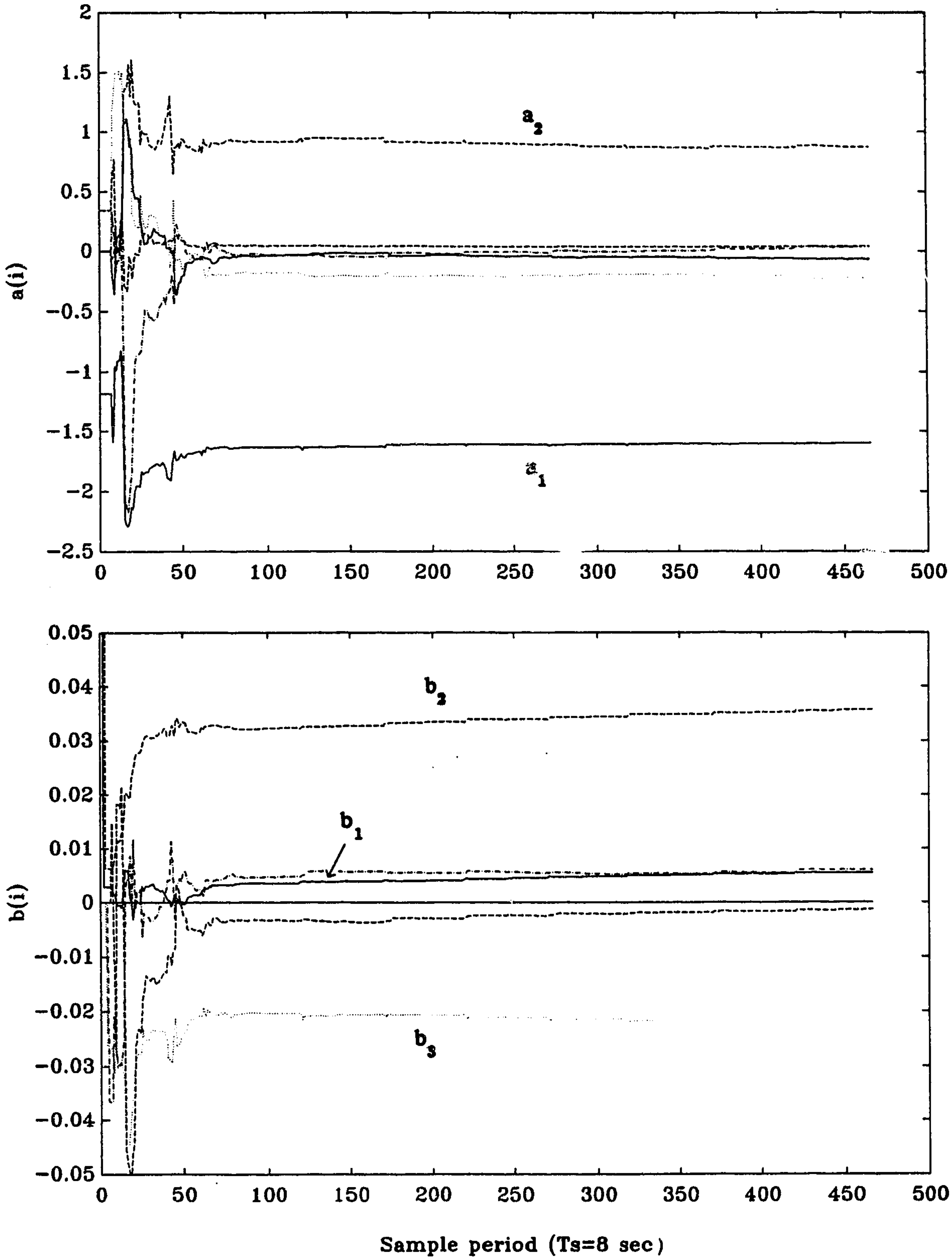


Figure 4.13b, Parameter trajectories for MAPC on CSTH

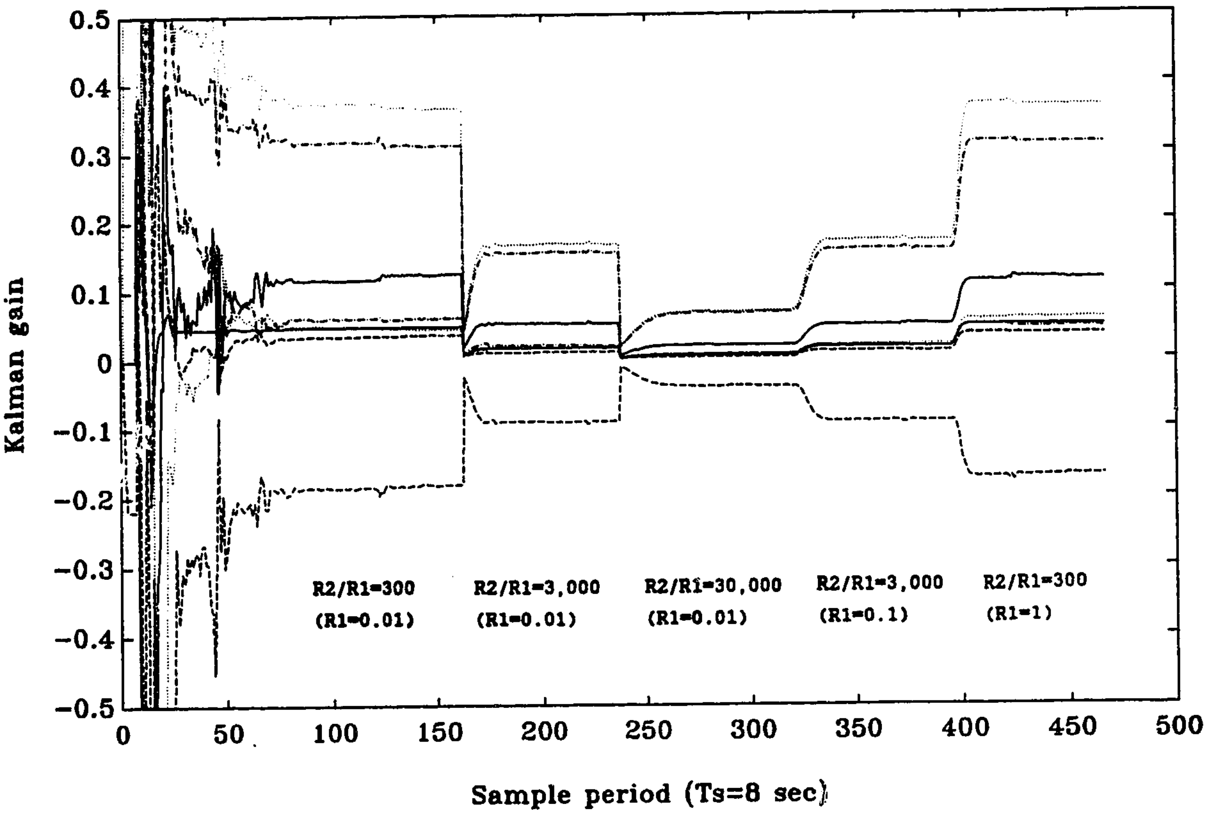


Figure 4.14. Kalman gain trajectories of MAPC on CSTH without time delay mismatch (TC#1) using different values of R_2/R_1

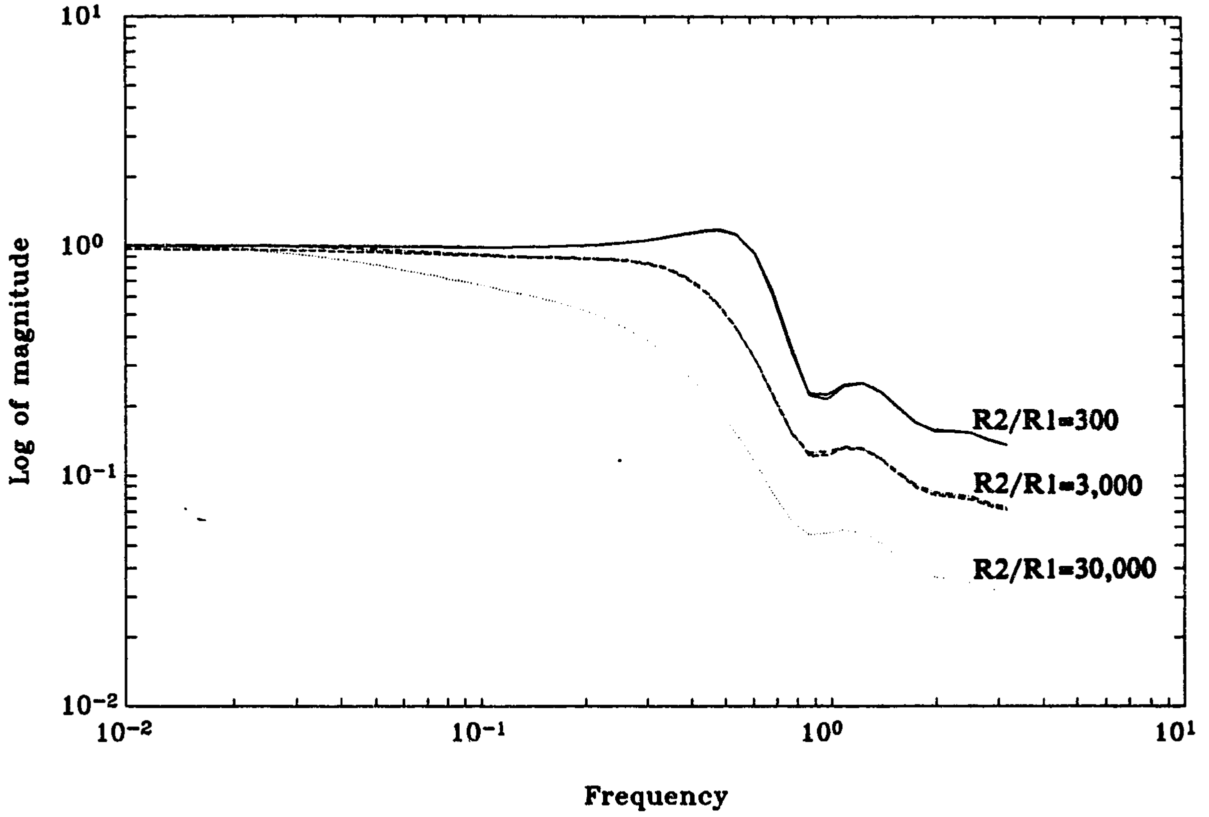


Figure 4.15. Magnitude plot of the frequency response of $G_f(z^{-1})$ without time delay mismatch (TC#1) using different values of R_2/R_1

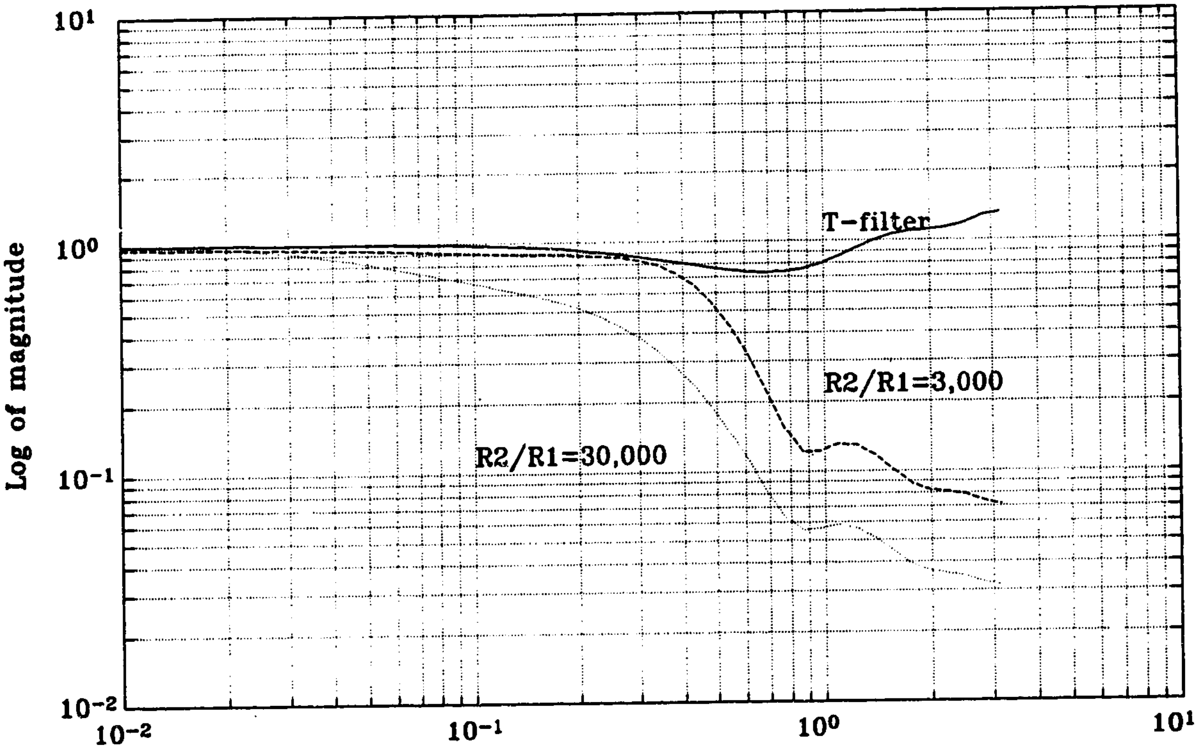


Figure 4.16. Comparison of $G_f(z^{-1})$ from the MKFP and from a T-filter

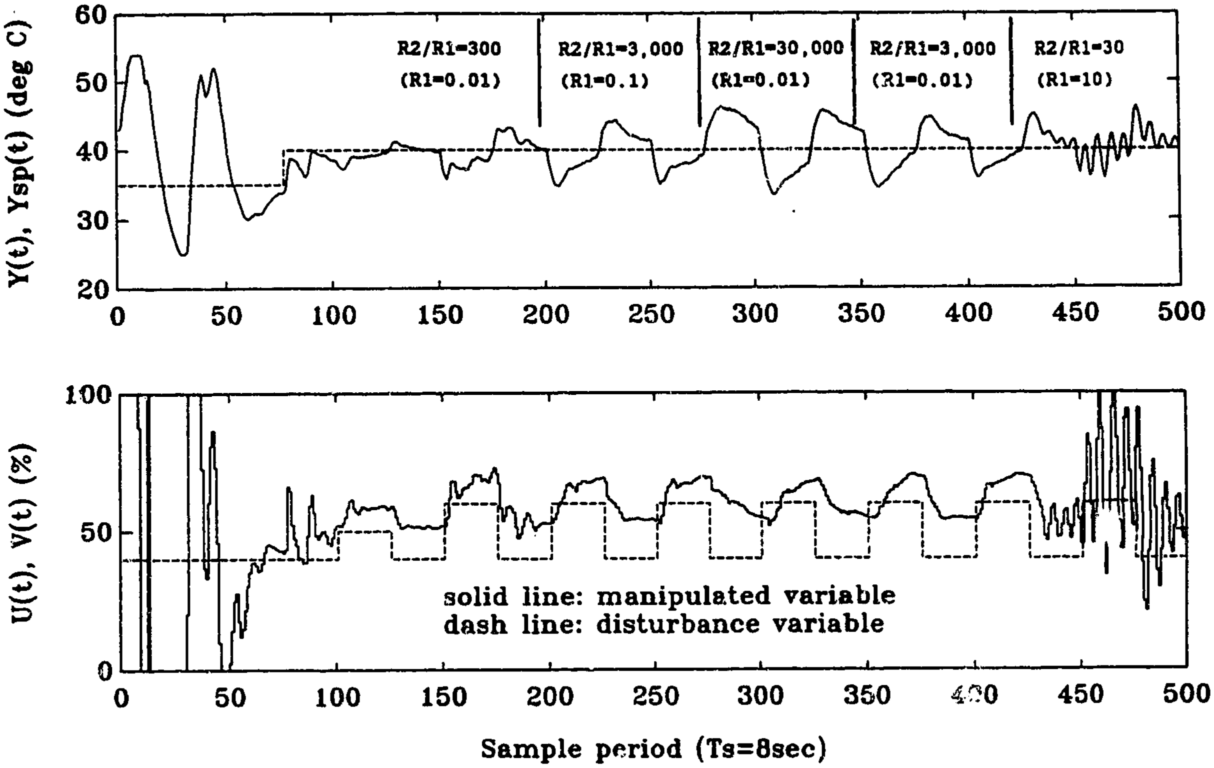


Figure 4.17. Regulatory control of MAPC on CSTH without time delay mismatch (TC#1) using different values of R_2/R_1

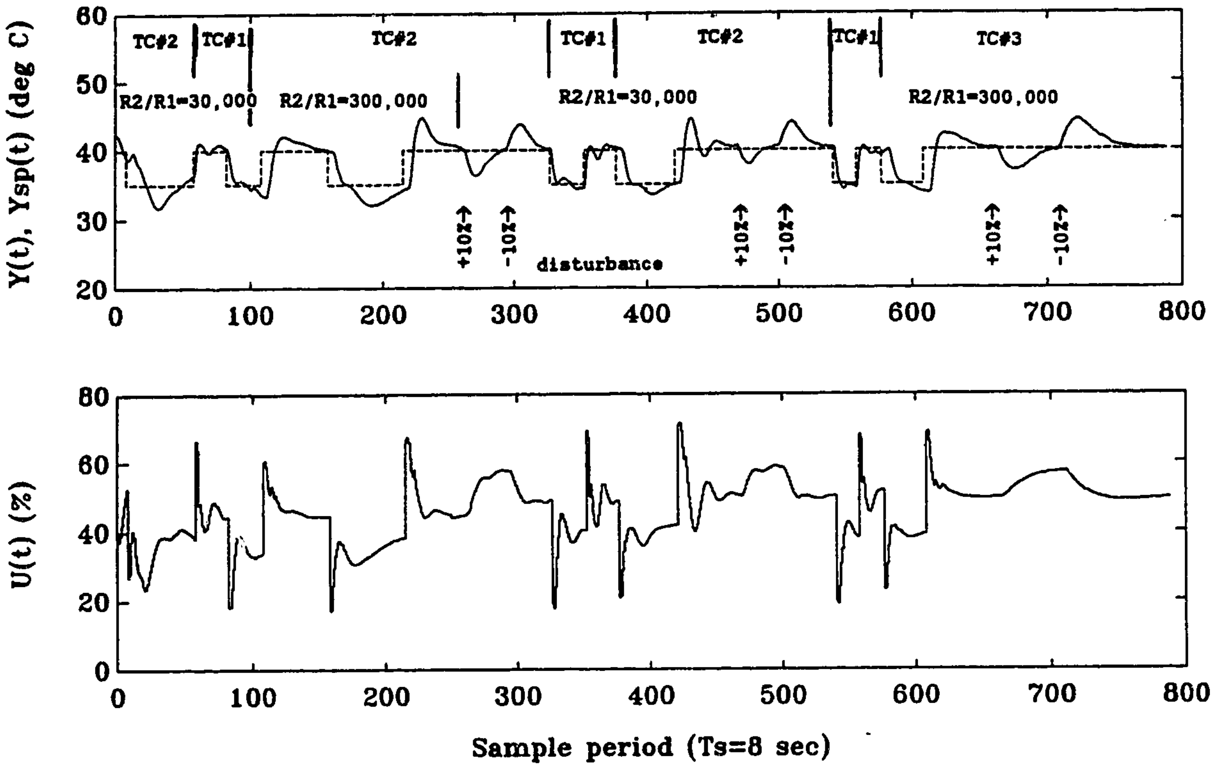


Figure 4.18. Servo and regulatory control of MAPC on CSTH with varying time delay (TC#2, TC#3) using different values of R_2/R_1

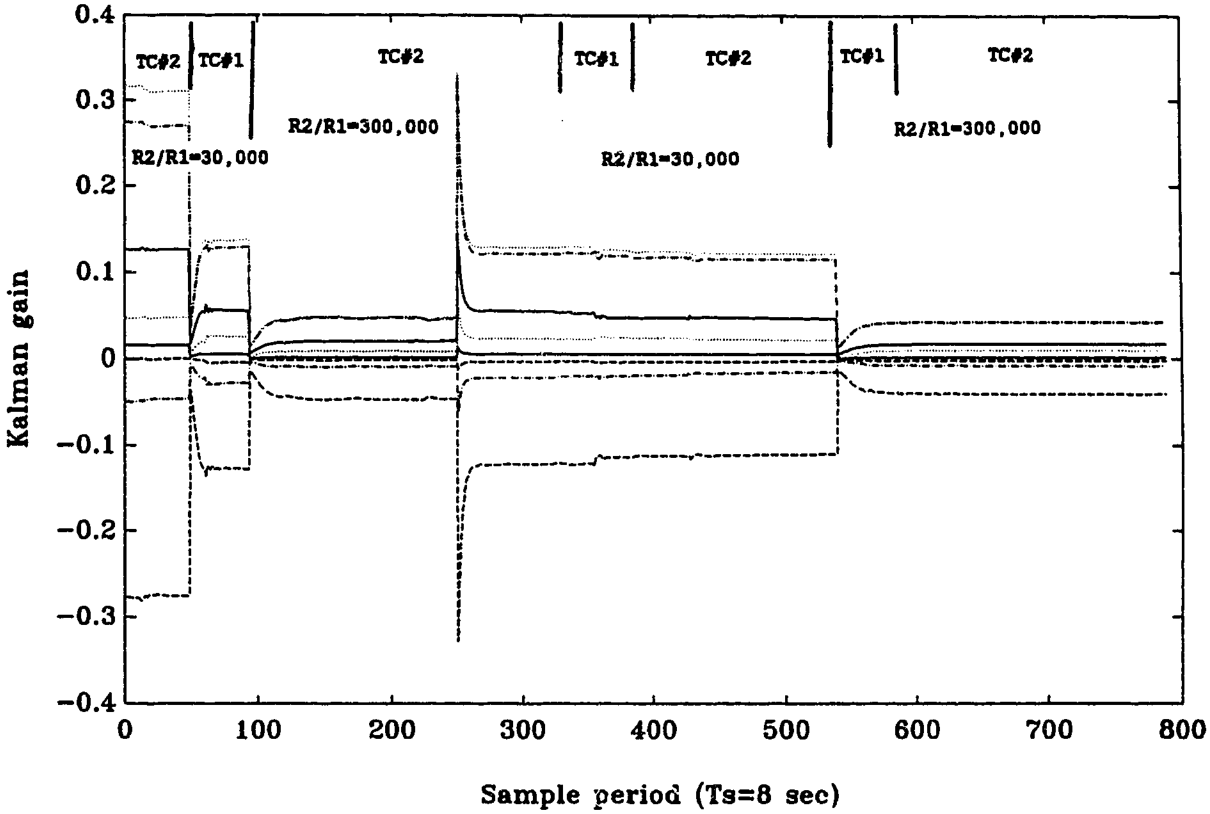


Figure 4.19. Kalman gain trajectories of MAPC on CSTH with varying time delay (TC#2, TC#3) using different values of R_2/R_1

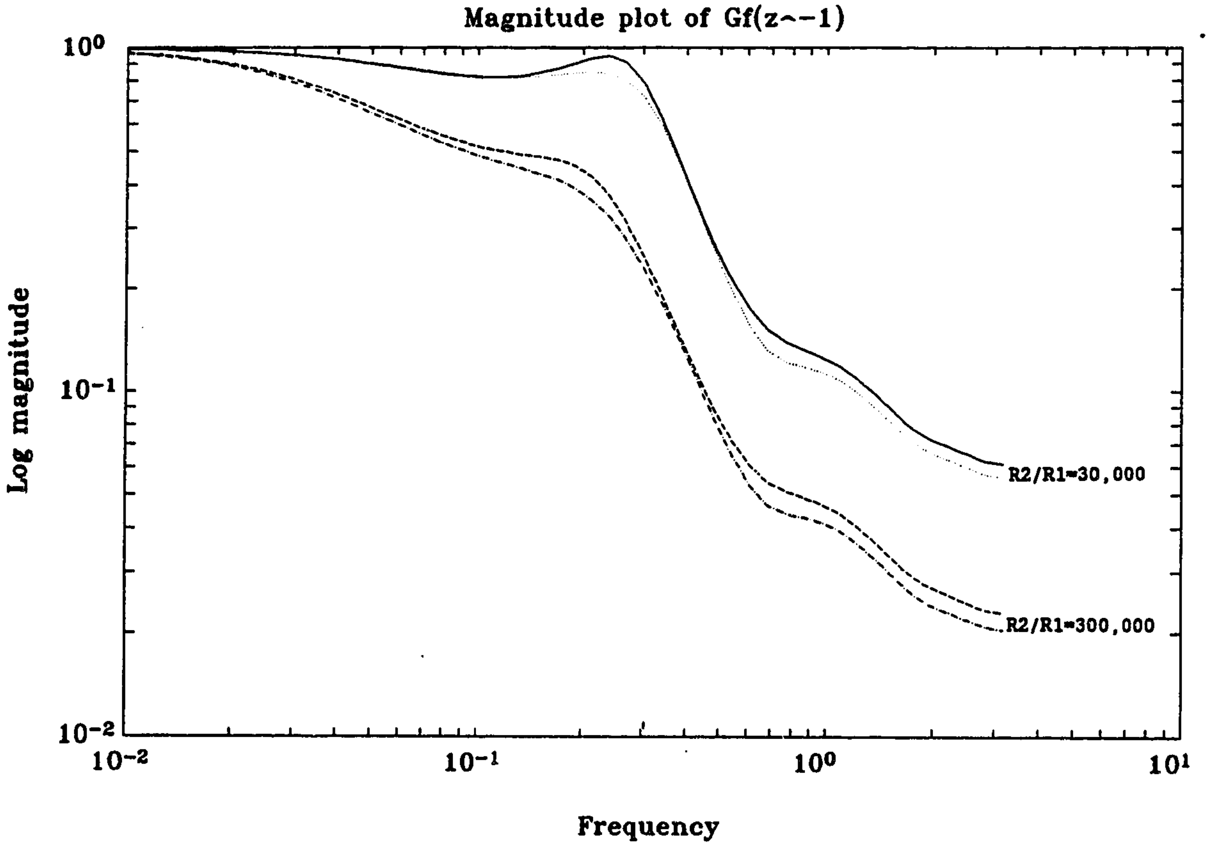


Figure 4.20. Magnitude plot of the frequency response of $G_f(z^{-1})$ with varying time delay (TC#1) using different values of R_2/R_1

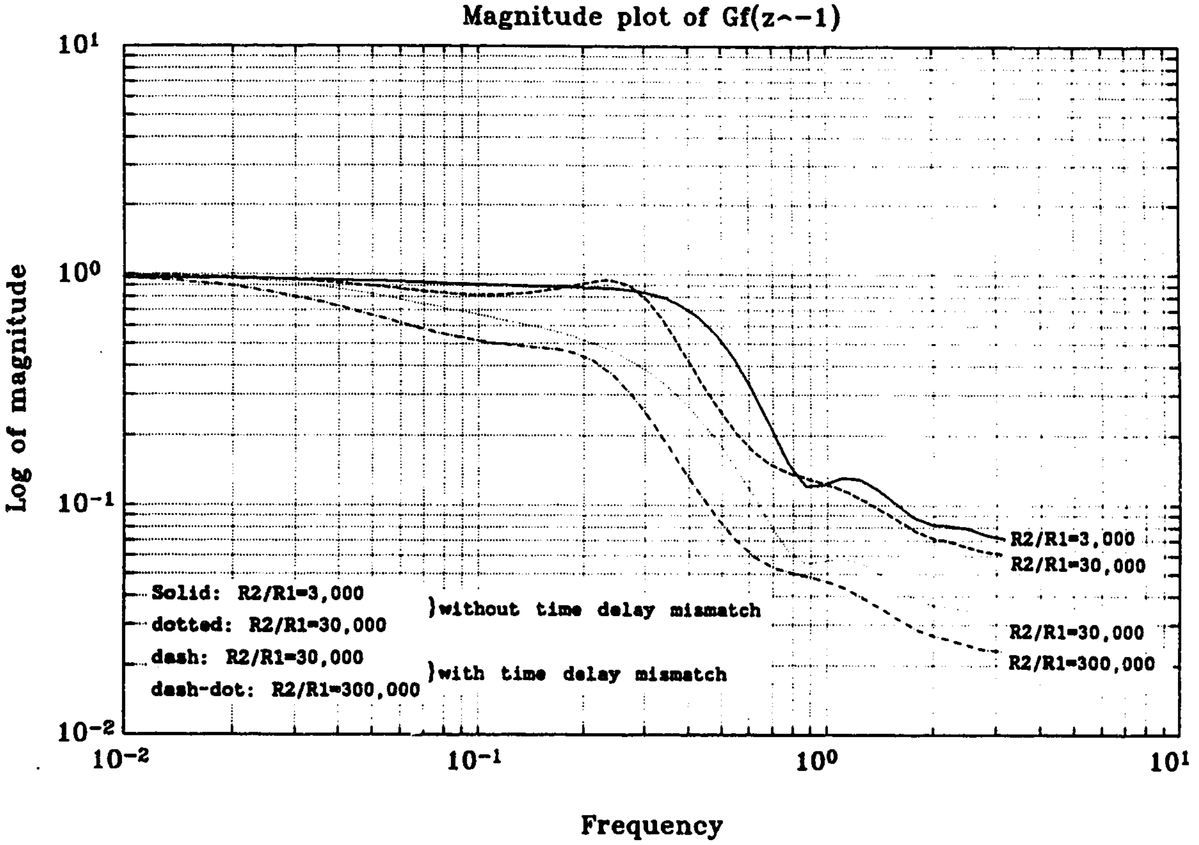


Figure 4.21. Comparison of $G_f(z^{-1})$ obtained for time delay mismatch case and that without time delay using different values of R_2/R_1

5 QNX plus MULTICON

5.1 Introduction

A typical real time software package for computer process control can be described by Figure 5.1. It is usually characterized by four goal-oriented modules labelled as modules 1, 2, 4, and 5 in the figure. Modules 1 and 2 handle information in real time. They control data exchange between the operator and the computer and between the process and the computer i.e. the I/O stages of process control. Module 4 contains the mathematical formulation to process the measured data (eg. digital filter, control algorithm etc) and decision algorithms to produce qualitative and quantitative information to send back to the process. Module 5 contains programs which modify the control algorithm or operating conditions in module 4 in an on-line manner based on current results.

Module 3 is the real time executive module that is under direct control of the operating system. It schedules and coordinates the other four modules so that the whole real time control software package runs in proper order. In this chapter, the real time executive program, MULTIpurpose CONtrol package (or MULTICON), developed at the University of Alberta (Qiu, 1988) for educational and experimental purposes is introduced. The main features are reviewed and some possible extensions and expansions are suggested (some of the proposals are already in progress). Since MULTICON is supported by and created under QNX, a brief introduction to the operating system is given first.

5.2 QNX Operating System

5.2.1 Introduction

The job of an operating system (referred to as OS herein) is to provide for an orderly and controlled allocation of the processor, memories, and I/O devices among the various programs competing for them. With the advent of personal computers, many vendors have been developing OS that are small (in terms of memory requirement) and yet powerful for them. Two OS that are currently dominating the PC world are the MS-DOS written by Microsoft, Inc.

for the IBM PC and compatibles using the Intel 8088 CPU and its successors; and UNIX, which is dominant on the larger PCs using the Motorola 68000 CPU family (and recently on workstations that use the Intel 80386 CPU). Whereas later versions of the MS-DOS are powerful and have been well received by many PC users, it is a 'single minded' OS since only one application program can be run at one time until it is finished. UNIX offers a multitasking, multiuser alternative to MS-DOS, but this system is usually larger in size and is usually not used on PC based (Intel 8088/80286) machines. Microsoft Inc. offers a Unix solution to the PC called Xenix. However, none of these OS supports real-time applications.

Since 1982, Quantum Software System Limited of Ottawa, Ontario has been marketing a PC based real-time, multiuser, multitasking OS called QNX (was called QUNIX at that time). It was designed and developed as an outgrowth of research done at the University of Waterloo. Instead of building on a monolithic or layer model (which are common OS architectures), QNX uses a modular architecture that is based on message passing. This design provides many modular advantages to the OS. Furthermore, it does not require much memory (a lean 150Kbytes) and is found to be very flexible in terms of real-time control applications. Figure 5.2 shows the layer model of an OS. Although this is not the structure used for QNX, the figure suffices in describing the many facets that an OS must handle. The following section contains a brief introduction to the structure and some important features of the QNX OS version 2.15.

5.2.2 Structure of the QNX OS

QNX is built on a client-server model. The approach is to implement most of the OS functions in server tasks (called administration tasks in QNX) and link them up through a message system. To request a service, a user task (called client task) sends the request to a server task, which then does the work and sends back the answer. This model is shown in Figure 5.3, where all the kernel does is to handle the communication between the client and the servers. This way the OS is split into small cooperating parts where each part handles one group of system function. This modular structure provides strict benefits to ease of maintenance and expandability. For instance, when a bug is triggered in one of the server tasks (e.g. the file server), the server task may crash, but the whole system will remain intact. Also, since all the server tasks run as user-mode tasks and not in kernel mode, user tasks are able to become server tasks. This means the OS can easily be extended by connecting these user-defined server tasks to the existing OS through messages. The MULTICON presented in section 5.3 is an example of such task. Another advantage of this message sending model is that it provides the OS a correct platform to work in a distributed (or network) system. The unified message passing system works for communication between either local tasks or remote tasks. This is because the client need not know whether the message is handled locally in its own machine, or some where across the network. As far as the client is concerned, a request is sent and a reply comes back.

All server and user tasks are held together by the QNX kernel (which is represented by a lean 10Kbytes of highly optimized code). It performs message passing and task synchronization functions. The task scheduler in the kernel makes sure that all the tasks get a fair share of the CPU time (on a 8MHz 80286 PC, the kernel performs well over 3000 task switches per second). This message system and the task scheduler algorithm will be described in section 5.2.3 and 5.2.4 respectively. At startup, five server tasks are created. Each of these server tasks provides a set of system calls for the user to use the hardware of the computer. They are listed in descending priority as follows: (For reference purposes, priority level is from 1 to 15, 1 being the top priority. The default priority of a user task is 8):

(1) Task administrator (*task*): Priority 1. Responsible for creating and destroying tasks. It is also responsible for allocating memory for the tasks. This is the most important task in the whole system. The 8088 version (or the 80286 real mode version) supports 64 task while the 80286 protected mode version supports 150 tasks.

(2) Device administrator (*dev*): Priority 2. Responsible for handling

all requests to open, close, read, and write to character devices such as the terminal, keyboard, line printer etc.

(3) File system administrator (*fsys*): Priority 3. Responsible for handling all requests to open, close, read, and write to files. It implements a tree-structured file system that supports disks up to 1 Terabyte in size with 0.5Kbyte unit of allocation.

(4) Network administrator (*net*): Priority 3. Responsible for handling all data requests which must be transmitted over the network. This task exists only in the network version of QNX.

(5) Idle administrator (*idle*): Priority 15. Responsible for consuming any spare processor time.

In addition to the above five tasks, two other important QNX tasks are:

The Timer

The Timer Administration task (*timer*) is not created at boot, but may be created after the system is running. It adds complex timing capabilities to the system and is responsible for providing 'wake up' service for real-time applications. Usually the timer task is run as a background task at a priority of 8.

The Shell

A task called the command interpreter, or shell, is created at log in. Although the shell is not part of the QNX OS, just as the text editor and the compiler are not part of the OS, it serves as a primary interface between the user and QNX. The shell has the terminal as the standard output and the keyboard as the standard input. It sits most of the time idling to wait for a user input. When a user types in a command to start a user program, the shell creates a son task and runs the user program as its son. It waits for the son to finish running and then reclaims the resources allocated to the son. The reader is referred to the QNX user manual for the many features that is supported by the QNX shell.

5.2.3 Inter-task communication in QNX

As mentioned earlier, message passing forms the basic structure of the QNX OS and it is handled by the kernel. Three basic system primitives are SEND, RECEIVE, and REPLY. When a task sends a message to another task, it will block (i.e. the task is held at its current state) until the receiving task receives the message and replies to it. When a task is waiting for a message to receive, it will block until another task sends to it (CRECEIVE is a non-blocking form of RECEIVE, i.e. whenever there is no in coming message, the task carries on with the next line of execution). When there are many tasks sending to one task, the messages are placed onto a stack and are processed on a first come first serve basis. Usually, the sending task and the receiving task agree on a certain communication protocol and the OS does not check the content of the message. The OS simply does a byte-copy from the source to the destination. The maximum message size in QNX is 64k bytes.

The message system not only provides the advantages mentioned above, the block send and block receive features also help to coordinate and synchronize the tasks. However, one common problem associated with messages is that it might get lost, especially when it is sent across a network. This could be a serious problem because if there is no external message coming in, both the sender and the receiver will be blocked. This may result in a locked situation. Some complex OS guard against this problem by enforcing the receiver to reply the sender with an acknowledgement message. If the sender does not receive any acknowledgement within a certain period of time, it sends the message again. QNX does not provide this feature on the OS level. It only protects the sender from sending to 'dead' tasks that of course do not respond to messages. However, the programmer can always make use of the system timer to accomplish similar protection in the user software.

5.2.4 Task scheduling in QNX

For a single processor computer to accomplish multitasking, the OS must be able to switch the CPU and other resources for the use of several time-competing tasks. This scheduling is done by the kernel in QNX. It uses the 'round robin' algorithm to schedule the tasks at a single priority level. The rules are summarized as follows:

(1) When a task is waiting for input or output (eg. accessing external devices such as a disk file), it will not occupy any CPU time.

(2) Each task can only occupy the CPU for a certain period of time (50 millisecond by default) if there is another task waiting to run. All tasks queue in a round robin fashion. The size of the time slice can be adjusted by the user.

QNX also supports priority scheduling. Each task is assigned a priority, and the runnable task with the highest priority is allowed to run. This leads to rule number three for task scheduling in QNX:

(3) The higher priority task will never relinquish the CPU to the lower priority tasks.

This seems to be a drawback to the QNX operating system because high priority task may run indefinitely. Some complex OS prevent this by decreasing the priority of the running task at each time slice. When the priority level drops below the next highest task, the task switch occurs. However, since most tasks require I/O, by rule number (1), the above problem seldom occurs. Figure 5.4 shows the scheduling algorithm used in QNX.

5.2.5 Computer System

Here is a description of the computer system that the QNX OS is currently mounted. It is running on an IBM PC model 30 (a Intel 8086 based computer) with a clock rate of 20MHz. The computer system includes the following:

(1) Memory : 640k bytes on board memory @ 15μsec response time.

(2) Hard Disk Drive : 20 Mbytes Winchester. Contains a 3 Mbyte DOS partition and a 17 Mbyte QNX partition.

(3) Floppy Disk drive : One 760kbytes $3\frac{1}{2}$" floppy drive.

(4) Coprocessor : Intel 8087 floating point coprocessor.

(5) Ports : 2 RS-422 serial ports and 1 parallel port.
(6) Cards : QNX network card, VGA graphic card.
(7) Display : IBM VGA colour monitor.
(8) Printer : IBM Proprinter.

The real mode version of QNX is used. Therefore, only 64 tasks are supported at one time and the memory is restricted to 604Kbytes. For higher end computers such as the 80286 and 80386 micro-computers, QNX can run in the protected mode. Up to 150 tasks can be supported and the system memory restriction is relaxed from a maximum of 640 kbyte to 16 Mbytes.

5.3 MULTICON

5.3.1 Introduction

MULTICON (MULTI-purpose CONtrol system package) is a real time executive program run under the QNX operating system (in QNX, an executive task is also referred to as an administration task). It is created to support the existence and control the activity of user tasks used for real time control application (referred to as Advanced Control Tasks (ACT) herein) and other system tasks such as the Process Operator Communication (POC) interface task and the on-line graphics task. The following are a list of the functionalities of MULTICON:

(1) It maintains an on-line data base and does house keeping on it.
(2) It performs task creation, termination, and task-scheduling.
(3) It sends and receives analog signals from the process (outside world) and places them in the on-line data base.
(4) It allows ACTs to communicate with each other (and also with itself) through the on-line data base.

Since MULTICON is built on the QNX OS, true multitasking is possible. This means that more than one tasks (control tasks or other supportive tasks such as an on-line parameter estimator task) can be run concurrently. All the user has to do is to fill out a table on how the tasks are to be scheduled and MULTICON will do the scheduling properly. Furthermore, MULTICON greatly simplifies the communication process between ACTs, i.e. the ACT developers no longer have to know about the message system of QNX. The ACTs can therefore be developed in a true modular form. By using the communication features in MULTICON, several functional blocks can be connected to form the desired control algorithm. With all these features and capabilities, MULTICON is a flexible shell for a control engineer to develop, test, and compare different control algorithms.

5.3.2 Structure of MULTICON

The MULTICON package is made up of five independent tasks which coordinate with each other by message sending:

(1) *multicon* : the main administrative task (core).
(2) *opto_drv* : the opto-22 interface driver.
(3) *timing* : a software interrupt to *multicon* core task.
(4) *sys_poc* : the user interface to *multicon*.
(5) *graph* : the on-line graphics task.

The *multicon* core (task 1) is responsible for creating the other four system tasks as well as other user defined ACTs at the start of the session, and terminates them when the session ends. Figure 5.5 shows the relationship of these tasks. A number indicating their priorities is included — the smaller the number, the higher the priority of the task. The functions and features of each task are described qualitatively as follows:

timing

The *timing* task is responsible for giving soft interrupts (or clock ticks) to the core *multicon* task at a resolution of one second. This is used by the core to check if there is any task that needs a 'wake up' call.

multicon

The *multicon* core program follows a typical structure of an administration task. Its function is to coordinate the activities of all tasks in the MULTICON package. The pseudo-code in Figure 5.6 describes its structure and the following is a short description to its operation.

The system tasks (*timing*, *sys_poc*, *opto_drv*, and *graph*) and the user defined ACTs send messages to the *multicon* core regularly. The *multicon* core services the tasks on a first-come first-serve basis. Upon receiving a message, the *multicon* core first discovers the identity of the task that is sending the message. It processes the request according to the five different cases (see Figure 5.6) and then replies to the task. Take case 5 as an example: when an ACT requires a sample from the process, it sends a message to *multicon* core to request the reading. Multicon identifies that the message is from an ACT, so it processes it according to case 5. It takes the sample reading from its data base (to be mentioned later) and sends it back (reply) to the ACT. When there is no incoming message, the *multicon* core sits idling.

The *multicon* core is capable of managing single-loop, multiloop, as well as concurrent tasks within the loop. The reader should refer to the user manual of MULTICON for details about how to set up these configurations (Qiu, 1988). This function is provided by the 'wake-up' service of *timing* (case 1) followed by case 5: whenever an ACT tells the *multicon* core that it is finished with one pass (by sending a wait signal to *multicon*), case 5 checks if there are subsequent and concurrent tasks to be activated.

When the user requests to terminate the session, *multicon* will kill all ACTs gracefully using the system *break* command, which means that the ACTs will not be terminated in the middle of the program. This is important for taking care of devices and files which would otherwise be left in an indeterminate state.

There are two features of the *multicon* core that are worth noting.

(1) The *timing* task sends an interrupt to *multicon* once every second which gives the *multicon* core its real-time properties. At these instances, *multicon* scans the task table to determine if there are ACTs (or loops) due for activation (case 1). It starts those tasks (or loops) whose sample times are up and those tasks that are waiting to be created. Therefore, the default time resolution for any discrete event running under *multicon* is one second. This also implies that the minimum sampling time for ACTs is one second. For those ACTs that specify zero sampling time, their schedulings are

totally under the control of the OS and *multicon* will not intervene with their operations.

(2) The *multicon* core maintains a pool (data base) for communications among the whole software package, especially for communication between different ACTs. The pool is an array of 100 elements which can store a variable name, a floating point value, and an explanation of the variable. This means that 100 numeric values can be passed between different ACTs. The editing and modification to the data base is done through *sys_poc*.

opto_drv

This task establishes a link between the *multicon* core and the 'outside world' through the serial port. It is created by the *multicon* core but run as a background task, i.e. it is not a son of *multicon* (see Figure 5.5). At startup, the task initializes the opto-22 interface and configures the I/O channels. Then it sends messages to *multicon* regularly to initiate accesses to the process, i.e. it asks the *multicon* core whether there is someone who wants to access the process. After a message is sent, it is blocked until the *multicon* core replies. There are two situations where the *multicon* core will reply to *opto_drv*:

(1) When it is in 'RUN' mode, the *multicon* core replies whenever (i) there is an ACT due for activation. Before the *multicon* core 'wakes up' the ACT, it requests *opto-drv* for a sample of the process. In other words, the process is sampled at every sampling interval set by the fastest loop. (ii) An ACT wants to send information to the process.

(2) When it is in 'PAUSE' mode, the *multicon* core replies whenever the *timing* task interrupts. In this case, the process is sampled every one second.

sys_poc

This is the user interface of MULTICON. Its main functions are to do maintenance on the data base and to interface the user with the *multicon* core. It is created by the *multicon* core as a concurrent son running at the same priority as its father (see Figure 5.5). It scans the keyboard in a continuous fashion for a key character and performs accordingly upon receiving a character that is on the menu. There are two possible requests from the user:

(1) the user may ask for a change in operating mode, eg. change from automatic control to manual control or quiting MULTICON. In this case, *sys_poc* is only responsible for sending the command to the *multicon* core and letting it accomplish the request.

(2) The user may want to alter the data base. In this case, *sys_poc* receives from the user all information about one particular item (the mnemonic number, the mnemonic name, a floating point value, and an explanatory note about that variable) into a buffer and then sends to the *multicon* core. Upon receiving the message, the *multicon* core modifies the data base accordingly. Therefore, *sys_poc* lets the user modify the pool in the *multicon* core one at a time. During the user input, the *multicon* core continues with its normal computational activity with the old data in the pool.

graph

This task equips MULTICON with an on-line graphics feature. It is created by the *multicon* core to run on another tty (see section 5.2 on QNX) so that the user can switch easily between the graphics window and the main control panel. After *graph* is created, it creates a son (a grandson of the *multicon* core) *graph_poc* as an user interface (see Figure 5.5). *Graph* sends a message to the *multicon* core every one second to get values from the pool for plotting. It also replies to *graph_poc* every second to see if the user has requested any changes. Up to twelve variables can be plotted simultaneously in an on-line manner.

ACT

All user written Advanced Control Tasks (ACTs) are created by the *multicon* core as concurrent sons. They all have the same priority, one lower than *multicon*. The usual structure of an ACT is as follows:

```
#include "r_w.h"          /* header file that include the three
                             three utility program */
     .
     .
main()    {
     .
     .
loop :    {
                read from pool;
                calculation block;
                write to pool;
          goto loop if not exiting;
          }
}
```

MULTICON makes communication between ACTs simple. Three utility programs (*read*, *write*, and *access*) and the communication protocol are provided for the user. They are in the *r_w.h* header file. To use the utilities, the user has to 'include' *r_w.h* at the beginning of the source file. The *read* and *write* can access up to 20 elements from the pool and multiple *read* and *write* are allowed in the ACTs. Each of the 100 elements in the pool are therefore available to all ACTs at any time through the use of the utility programs. Furthermore, the *read* and *write* statements provide a simple way to send a wait signal to the *multicon* core for task swapping. This is important for ACTs that are to be run in discrete fashion. The reader should refer to the MULTICON user manual for details about the *r_w.h* file.

5.4 Use of MULTICON

Qiu (1988) wrote a detailed user manual for MULTICON. The purpose of this write-up is not to duplicate those efforts. Rather, this section is to summarize the ACTs that have been developed by the author to run on MULTICON.

The detailed theory and properties of the following control algorithms have been discussed in Chapter 2, 3, and 4. They are listed according to their functions:

Non Adaptive ACTs

(1) *pid* : PID control task.

(2) *feedfw* : Fixed gain feedforward control task.

(3) *smp* : Smith Predictor task.

Adaptive ACTs

(1) *rac* : Robust Adaptive Controller task (Single step Adaptive controller, Chapter 3).

(2) *mapc*: Multistep Adaptive Predictive Control task (Chapter 4).

(3) *kfp* : Adaptive Modified Kalman Filter Predictor task (Chapter 4).

Parameter Estimation ACTs

(1) *ils* : Improved Least Square task.

These ACTs can be combined in different ways to form different control schemes. For example, the *kfp* or the *smp* can be combined with the *pid* to form a predictive control scheme. The control tasks mentioned in Chapters 2, 3, and 4 are constructed by linking particular ACTs together (see section 5.3.3, point number 2):

(1) Non-Adaptive Predictive Control: *pid* + *smp* + *feedfw*;

(2) Single Step Adaptive Predictive Control: *rac*;

(3) Multi-Step Adaptive Predictive Control: *kfp* + *mapc* + *ils*;

The user manuals for these three control schemes are in Appendices 2A, 2B, and 2C respectively. They were all connected to the Continuous Stirred Tank Heater for experimental evaluation and comparison. Other ACTs have been written by other developers, eg.:

(1) SISO GPC: *gpc* + *ils*;

(2) MIMO GPC with control output constraints:

5.5 CSTH and the Opto-22 Interface

5.5.1 The Continuous Stirred Tank Heater (CSTH)

The pilot scale Continuous Stirred Tank Heater (CSTH) is described schematically in Figure 5.7. Cold water flows into the glass tank, which is heated up with steam, and exits the tank through a long copper pipe. A stirrer is used to ensure good mixing in the tank. The steam and water flowrates are independently controlled by pneumatic valves. Water level in the tank is maintained by a pneumatic proportional regulator at the exit. The inlet water flowrate is measured by a magnetic flow meter and the water exit temperature is measured by one of the four thermocouples located along the copper pipe.

The following are the nominal operating conditions for the CSTH:

(1)	Inlet water temperature	5°C
(2)	Inlet water flowrate	50 cm^3/s
(3)	Water level	25 cm
(4)	Steam flowrate (% valve opening)	50%
(5)	Exit water temperature	35°C

Various feedback and feedforward control schemes can be configured. The following is the usual set up for experimental evaluation of control algorithms:

(1) The steam flowrate is the manipulated variable.
(2) The exit water temperature is the measured variable.
(3) The inlet water flowrate is the disturbance variable.

The open loop response curves of the CSTH are shown in Figure 5.8. They can be approximated by a first order model plus a time delay. The models obtained from the first three thermo-couples are listed in Table 5.1:

Table 5.1 Open loop process model of CSTH

Thermocouple	Gain(°C/%)	Time constant(s)	Time delay(s)
#1	0.76	70	8
#2	0.76	70	46
#3	0.75	68	78

Notice that the model parameters vary with operation conditions. The water level changes when inlet water flowrate changes. This is because of the offset due to the proportional regulator used for level control of the tank. The time constant and the time delay also vary with the water flowrate and the water tank level. Therefore, the models in Table 5.1 are accurate only at the nominal operating point.

Sampling Time

The sampling time for the process is chosen as 8 to 10 seconds, which is the usual recommendation for discrete sampling: choose a sampling rate 0.1 to 0.25 of the dominant time constant or the time delay (whichever is larger) (Stephanopoulos, 1984).

5.5.2 The Opto-22 Interface

The Opto-22 behaves as a low end computer to the host computer (the IBM model 30) connected through a RS-422 serial line. Its responsibility includes: (1) Optically isolating the process with the computer. (2) Continuously scanning the I/O channels and providing signal conversions for inputs and outputs (eg. Analog to digital conversion). (3) Responding to requests from the host computer.

There are two input variables (water temperature and water flowrate) and two output variables (steam control valve position and water control valve position) for the CSTH. Whenever the host computer (MULTICON) needs to access these variables, say to sample the water temperature, it sends a command to Opto-22 to ask for the data. Opto-22 gets the most recent digital data from the channel connected to the thermocouple and loads it onto the serial line to send the data back to the host computer. A very complete set of communication protocol is provided by the Opto which makes the Opto-22 very easy to talk with.

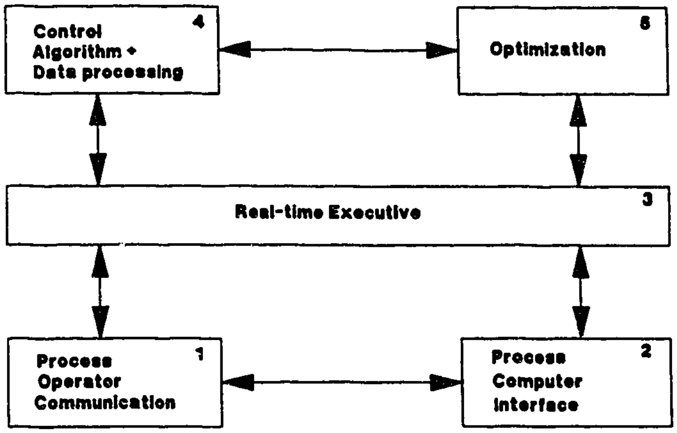

Control Algorithm + Data processing
4
Optimization
5
Real-time Executive
3
Process Operator Communication
1
Process Computer Interface
2

Figure 5.1, Block diagram of a Real-Time Software Package for Process Control

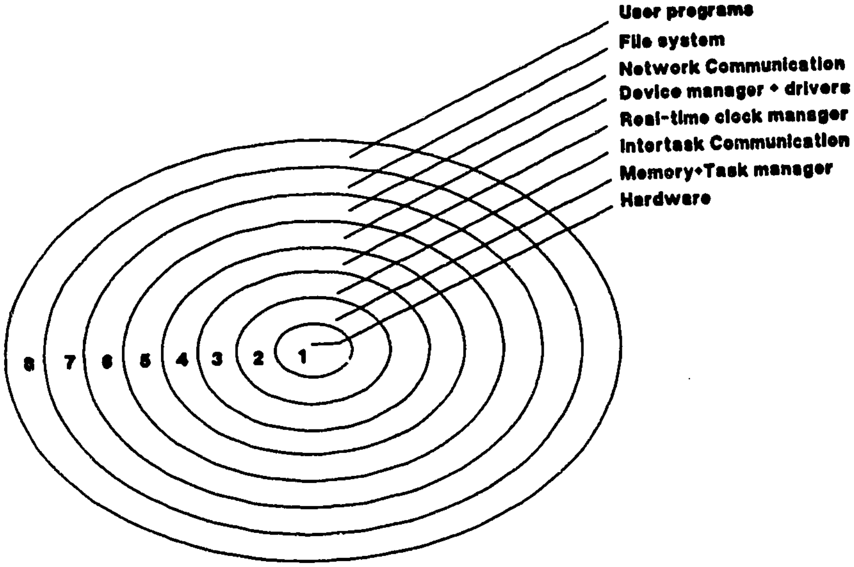


Figure 5.2, General block diagram of an Operating System

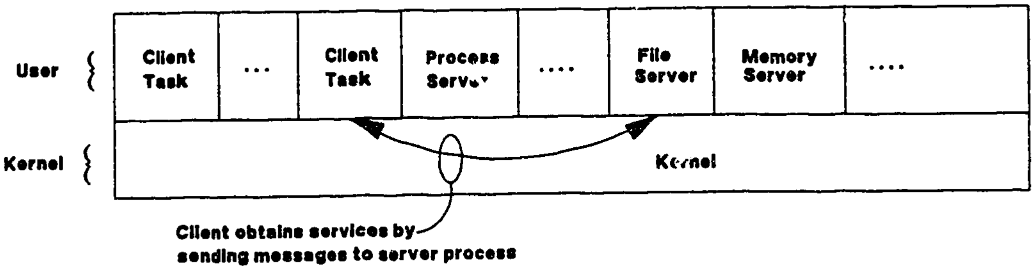


Figure 5.3. The Client-Server model of QNX Operating System

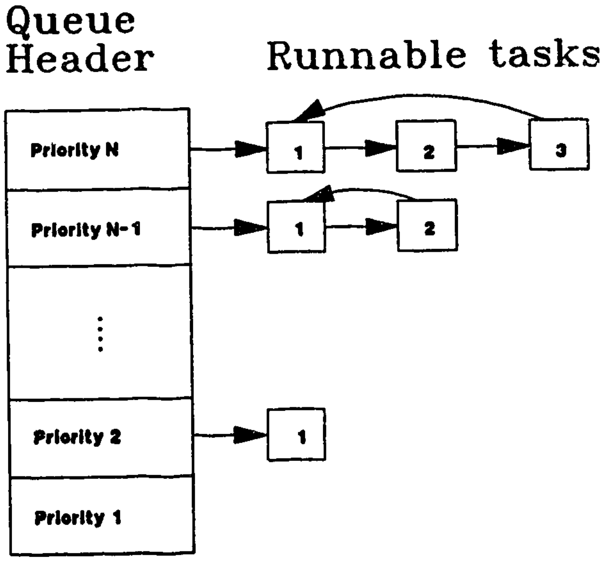


Figure 5.4. The Scheduling Algorithm of QNX Operating System

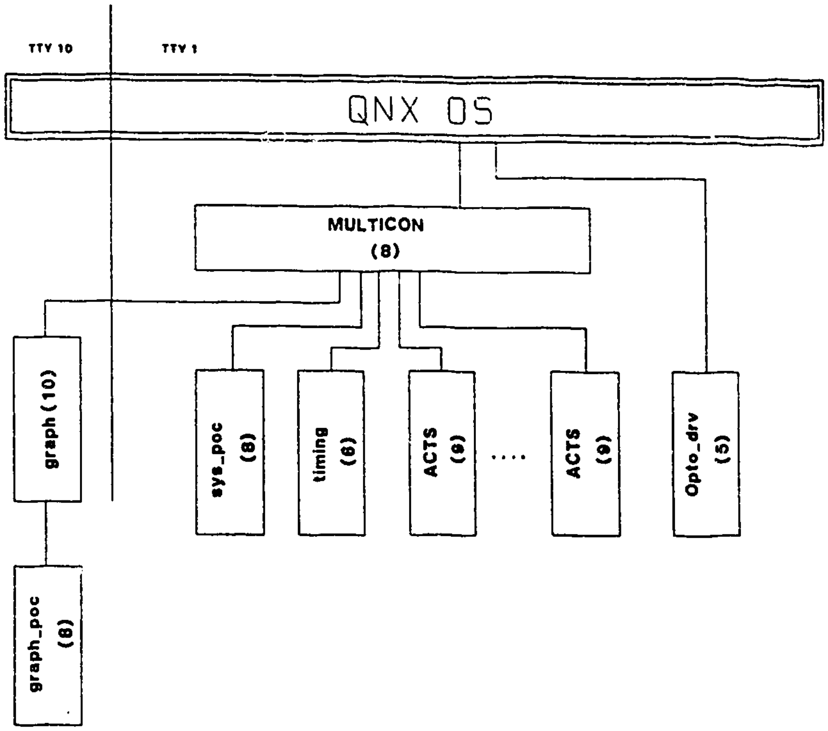


Figure 5.5. Schematic diagram of the structure of MULTICON

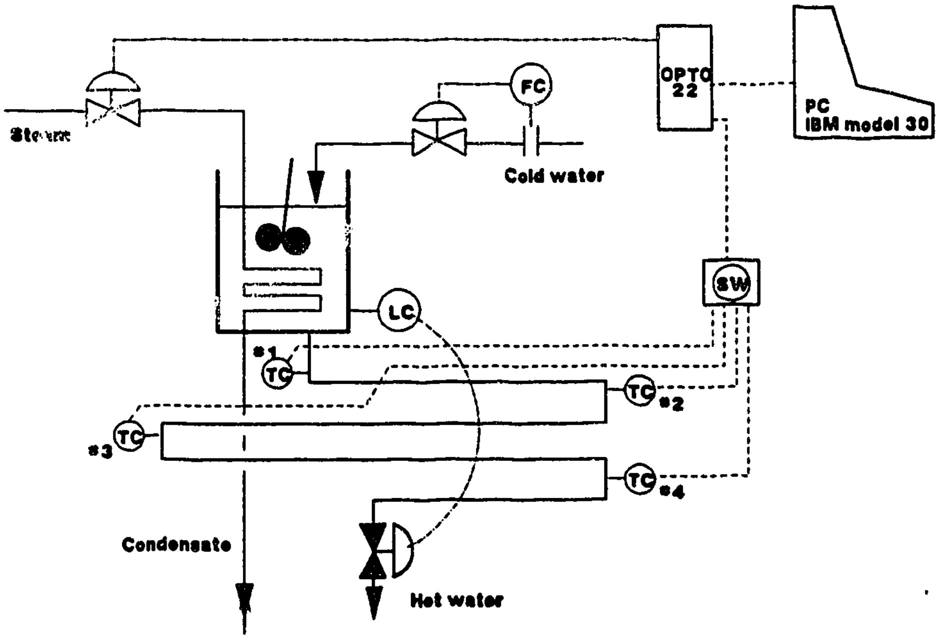


Figure 5.7. Schematic diagram of the CSTH

```
create all system tasks;
loop: {
     receive message from any task;
     process message immediately according to cases: {
          case 1 : message from timing {
               do : {
                    check if any task is waiting to be created :
                    if so create the task;
                    check if there is any task (or loop) whose
                    sampling period is due : {
                         sample process and reply to opto_drv;
                         activate the task (or loop);
                         }
                    }
               }
          case 2 : message from opto_drv {
               do : {
                    check if any task requests sampling the
                    process :
                         update data pool if so;
                    /* reply is done through case 1 */
               }
          case 3 : message from sys_poc {
               do : {
                    case : for modifying operation mode {
                         /* modes: auto/manual/run/pause/quit */
                         set flags according to modes;
                         reply to sys_poc;
                         }
                    cases : for modifying data pool {
                         modification includes: append,
                         modify, insert, and cancel to task
                         table and system data table;
                         reply to sys_poc;
                         }
                    case : for terminating session {
                         exit the loop;
                         }
          case 4 : message from graph {
               do : {
                    reply graph with data it needs for on-line
                    plotting;
                    }
          case 5 : message from other ACTS {
               do : {
                    check if : ACTs request to wait {
```

```
                    if so : {
                         reply with data requested and hold
                              the task;
                         check if subsequent task should be
                              activated; activate if so;
                         check if concurrent task should be
                              activated; activate if so;
                         }
                    if not: reply with data requested only;
                    }
               }
          }
     goto loop;
     write data to file;
     kill system tasks and ACTs;
     }
```

Figure 5.6, Pseudo-core for the *multicon* core program.

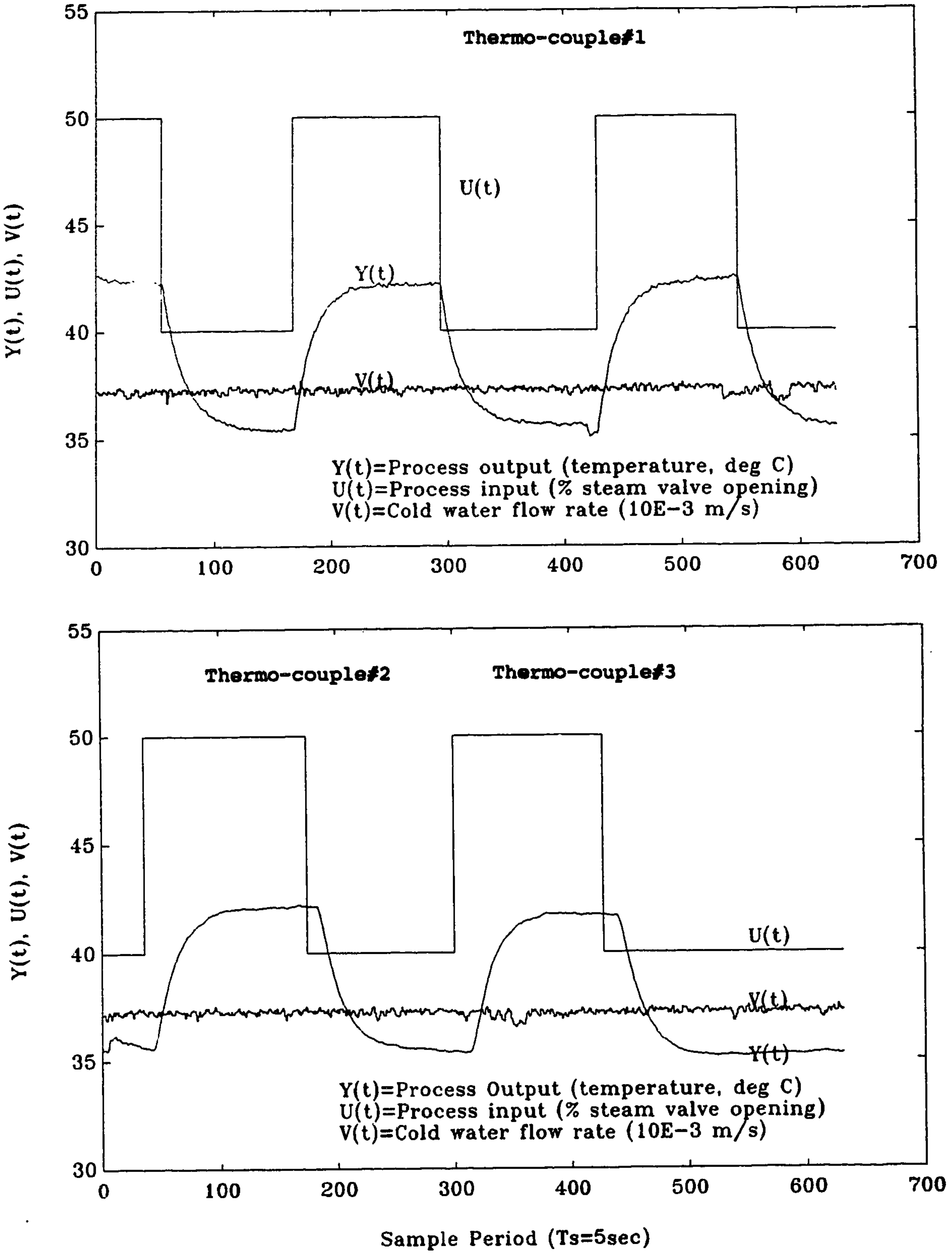


Figure 5.8. Open loop response curve of the CSTH

6 Conclusions and Recommendations

6.1 Conclusions

The main contributions of this thesis are the implementation and evaluation of three PC based Advanced Control algorithms that have the capability of handling processes with varying time delay.

(1) The PID+Smith predictor offers a simple but effective solution for predictive control as long as the process model can represent the process sufficiently well. When the time delay is known, the closed-loop system can afford high gains that result in faster and tighter control performance with the Smith predictor in place . In the presence of MPM, or in particular time delay mismatch, the residual filter $Gf(z^{-1})$ plays an important role in attenuating the residual terms due to the mismatch. An exponential filter is used as the residual filter and is found to be very effective. It was found that when the delay mismatch increases, stronger filtering action is needed to improve on servo performance. On the other hand, the filtering action should not be too strong for regulatory control. The user must select the filter according to the mode of application.

(2) The RAC is an adaptive predictive controller based on a single stage controller design criterion. It gives robust servo and regulatory control in the presence of time delay mismatch. The dead-zone is very important for the robustness of the parameter estimator and it can be chosen as 1/2 of the maximum allowable prediction error. The control weighting $Q(z^{-1})$ is very important for the success of the control scheme because (1) it ensures inverse-stable criterion, so that the controller can handle NMP processes and (2) it detunes the controller whenever its gain becomes excessively high due to time delay mismatch. The choice of $Q(z^{-1})$ is not easy but can be made simple by choosing $Q(z^{-1})=\lambda\Delta$. When the time delay is not known, start off with a large value of λ for robustness and progressively tune it down to gain speed and alertness. A more complex form of $Q(z^{-1})$, such as an inverse PI form, can be used to get better performance. However, a more complex

$Q(z^{-1})$ is more difficult to use without some *a priori* knowledge of the process.

Due to offset problems with the original RAC design, the RAC was reconstructed using a CARIMA model for process representation and an integrator in the predictor to remove offset. The resulting controller achieved offset free performance. In addition, to enhance flexibility to the control scheme, $P(z^{-1})$, $R(z^{-1})$, and $T(z^{-1})$ weightings are added.

(3) The MAPC is a powerful control scheme that belongs to the LRPC family. It includes a robust parameter estimator for process identification, a MKFP for providing the j-step ahead prediction trajectory, and a multistep controller.

The MKFP resembles a Smith predictor with an optimally chosen residual filter. Tuning the MKFP is therefore the same as adjusting the characteristics of the filter. The variances of the process noise, R_1, and the measurement noise, R_2, affect the performance of the MKFP. It was found that it is not the individual values of R_2 or R_1 that affects the performance but their ratio, R_2/R_1. In the presence of time delay mismatch, a large R_2/R_1 usually gives stronger residual filtering (a low-pass filter that attenuates high frequency residuals) and smoother (or more sluggish) prediction trajectory. On the other hand, if R_2/R_1 is too small, the filter (and therefore the predictor) may become unstable. It is therefore a precaution to the user that R_2/R_1 should not be set too small whenever the delay is not known or is time varying.

In terms of the overall performance of MAPC, the R_2/R_1 serves as a major tuning parameter as well. It was found that a large (or an over sized) R_2/R_1 slows down both servo and regulatory controls because of the sluggishness of the prediction trajectory and the strong filtering on the residuals. This is a classical trade-off between robustness and performance. The rule of thumb is to choose R_1 10^{-4} times smaller than R_2 (which is the variance of the measurement noise) when commencing the controller, and progressively tuning R_2/R_1 down to achieve tighter controller performances. With R_2/R_1 as the tuning knob and the rest of the tuning parameters set to default values, the controller is able to give excellent servo and regulatory control performances to the CSTH even under severe time delay mismatch.

Among the three predictive control schemes, the MAPC is the most complicated but versatile and powerful controller. When computer resources are cheap and available, this scheme is preferable over the others. Also, the controller is very simple to use thanks to the 'default' settings of the control parameters.

There are other contributions in terms of software development:

(1) The multitasking, multiuser real-time QNX operating system and the MULTICON executive program are examined by looking at their internal structures. The features supported by MULTICON are also mentioned. Together, they form a unified platform for the development of Advanced Control Tasks.

(2) Each of the above mentioned Advanced Control Tasks are implemented in C programming language and run under the MULTICON environment.

(3) Full user documentation for each Advanced Control Task is written for later reference and use of the programs.

(4) A software utility library for matrix and vector manipulation is developed with full user documentation.

Organizing these Advanced Control algorithms into a unified environment allows easy access and use of the software. This is useful when the control algorithms are to be evaluated by future users. For example, the PID, Smith predictor, and the Feedforward ACTs are being frequently used for educational purposes.

6.2 Recommendations

The recommendations are mainly for the upgrade of MULTICON. With all its present features, MULTICON can support up to the limit of 100 ACTs, only limited by the computer memory and speed and the operating system's mode of operation (for 8088 or 80286 real-mode version, QNX supports up to 64 tasks; for 80286 or 386 protected-mode version, QNX supports up to 150 tasks). It supports single loop, multi loop, and concurrent tasks within a loop. Each ACT or loop can communicate with any other ACT or loop through the pool. MULTICON is indeed a flexible software package for developing control tasks. However, there is still much room for improvement and some suggestions are listed as follow:

(1) The data pool can be improved on. There are two drawbacks in the present version of MULTICON. First, it can only contain up to 100 floating point elements and 16 of them are used as system variables. This means that only a limited number of variables can be passed among ACTs. Second, each of these elements is referred to by a mnemonic name. Accessing a variable with multiple values (eg. a matrix) is very cumbersome. Of course, to solve the first problem, the array of the pool can be expanded to a bigger size. However, the basic limitation is rooted in the basic design of the data structure.

Instead of storing a floating point value, MULTICON could store a pointer which points to a block of memory for storing the real data. This would allow MULTICON to store variables with multiple values. The block of memory can be dynamically allocated according to the size of the variable. This feature is important for MULTICON to support ACTs with variable size data structure. For example, in the Recursive Least Square algorithm, the size of the covariance matrix changes with the number of parameters estimated. If the RLS ACT wants to pass the covariance matrix to another ACT, MULTICON must use a dynamic data structure for its pool.

This modification would require a major overhaul on the whole MULTICON system including the data structure, communication protocol, the user interface, and also the data pool editor.

(2) The ACTs can be considered as functional blocks (eg. RLS, PID, GPC etc). Each has inputs, outputs, and parameters. All these variables must be in the data pool in order that they can be shared among different ACTs and/or modified by the operator. This means that all the

ACTs must use the same name to reference one particular variable. Future ACT developers must therefore be informed of the variable names of other previously developed ACTs in order to coordinate with them. This reduces the modular structure of the ACTs. The problem lies in the fact that MULTICON uses mnemonic name to refer to the variable. For example, if the RLS ACT wants to pass the parameter estimates to the CONTROL ACT, then both the RLS and CONTROL ACTs must use the same mnemonic names for the variables that represent the parameter estimates.

A utility program for equivalencing variables can be provided by MULTICON to alleviate this problem. Thus every time when MULTICON is invoked, the user has to define a set of variable names (or mnemonic names) in the pool. Then he is asked by the utility program to equate these mnemonic names to the input and output variables of the ACT to be used. This way, the user physically links the modular ACTs to form a complete control loop.

(3) In constructing a control loop, one particular ACT may be used many times. For example, the same filter ACT can be used for the regressor filter and the controller output filter. This requires that MULTICON to be able to create ACT with the same name more than once. This could be done because the QNX OS supports the creation of the same task multiple number of times (by using the *fork* command). With this modification, and modification number 2, MULTICON can become a true modular shell for ACTs.

(4) Other minor changes can be made: (i) The *sys_poc* can be improved on. (ii) A built in data acquisition task *historian* can be added. The structure of this task is similar to the *graph* except for the destination of the output. Whereas *graph* writes to the graphic terminal, *historian* writes to a disk file.

7 References

7.1 Chapter 1

Åström, K.J., Wittenmark, B., (1973) "On self-tuning regulators", Automatica, Vol. 9, pp.185-199.

Åström, K.J., Wittenmark, B., (1980) "Self-tuning Controllers based on pole-zero placement", Proc. IEE, Vol. 127, pp.120-130.

Clarke, D.W., Mohtadi, C., Tuffs, P.S., (1987) "Generalized Predictive Control", Automatica, Vol. 23, No. 2, pp.137-148.

Cluett, W.R., Shah, S.L., (1989) "A Robust Adaptive Controller with Time-Delay Mismatch", Adaptive control strategies for Industrial use - Proceedings of a workshop, Editors: Shah and Dumont, Springer Verlag LNCIS series', 1989.

de Souza, C.E., Goodwin, G.C., Mayne, D.Q, Palaniswami, M., (1988) "An Adaptive Control Algorithm for Linear Systems having Unknown Time Delay", Automatica, Vol. 24, No. 3, pp.327-341.

Dumont, G.A., Zervos, C., (1988) "Deterministic Adaptive Control based on Laguerne Series Representation", Int. J. Control, 48, pp.2333-2359.

Friedlander, B., Porat, B., (1984) "A parametric technique for time delay estimation", ASSP Conf.

Gawthrop, P.J., Nihtila, M.T., (1985) "Identification of time delays using a polynomial identification method", Syst. Control Lett., 5, pp.267-271.

Hassab, J.C., Boucher, R.E., (1979) "Optimum estimation of time delay by a generalized correlator", IEEE Trans. Acoust. Speech Sig. Process., ASSP-27, pp.373-380.

Ljung, L., Söderström, T., (1983) Theory and Practice of Recursive Identification, M.I.T. Press, Cambridge, MA.

Pearson, A.E., Wu, C.Y., (1984) "Decoupled delay estimation in the identification of differential delay systems", Automatica, 20, pp.761-772.

Rao, G.P. (1983) Piecewise Constant Orthogonal Functions and their Application to Systems and Control, Springer, New York.

Robinson, W.R., Soudack, A.C., (1970) "A method for the identification of time delays in linear systems", IEEE Trans. Aut. Control, AC-15, pp.97-101.

Smith, O.J.M., (1957) "A Closer Control of Loops with Dead Time", Chem. Eng. Progr., 53, No.5, pp.217-219.

Sripada, R., (1988) "Multi-Step Adaptive Predictive Control", Ph.D. Thesis, Dept. of Chem. Eng., Univ. of Alberta, Edmonton.

Walgama, K.S., (1986) "Multivariable Adaptive Predictive Control for Stochastic System with Time Delays", M.Sc. Thesis, Dept. of Chem. Eng., Univ. of Alberta, Edmonton.

Wellstead, P.E., Prager, D., Zanker, P., (1979) "Pole assignment Self-tuning regulator", Proc. IEE, Vol. 126, pp.781-787.

7.2 Chapter 2

Åström, K.J., Hägglund, T., (1983) "Automatic tuning of simple regulators for phase and amplitude margins specifications", IFAC Workshop on Adaptive Systems in Control and Signal Processing, San Francisco.

Åström, K.J., Wittenmark, B., (1984) Computer Controlled Systems: Theory and Design, Prentice-Hall Inc., pp.181-183, New Jersey.

Bristol, E.H., (1987) "Process system adaptation beyond math models", Adaptive control strategies for Industrial use - Proceedings of a workshop, Editors: Shah and Dumont, Springer Verlag LNCIS series', 1989.

Clarke, D.W., Gawthrop, P.J., (1975) "Self-tuning controller", IEE Proc., 122(9), pp.929-934.

Cohen, G.H., Coon, G.A., (1953) "Theoretical investigation of retarded control", Trans. ASME, 75, p.827.

Garcia, C.E. and M. Morari (1982), "Internal Model Control. 1. A Unifying Review and Some New Results", Ind. Eng. Chem. Process Des. Dev., 21, pp.472-484.

Goberdhansingh, E., Cluett, W.R., (1987) "Evaluation of an industrial PID autotuner", Adaptive control strategies for Industrial use - Proceedings of a workshop, Editors: Shah and Dumont, Springer Verlag LNCIS series', 1989.

Minter, B.J., Fisher, D.G., (1987) "Self-tuning versus Adaptive-predictive controllers", Adaptive control strategies for Industrial use - Proceedings of a workshop, Editors: Shah and Dumont, Springer Verlag LNCIS series', 1989.

Rivera, D.E., Morari, M., Skogestad, S., (1986) "Internal Model

Control. 4. PID Controller Design", Ind. Eng. Chem. Process Des. Dev., Vol. 25, pp.252-265.

Seborg, D., Yuwana, M., (1982), "A new method for on-line controller tuning", AIChE J., 28, pp.434-440.

Seborg, D.E., Edgar, T.F., Mellichamp, D.A., (1989) Process Dynamics and Control, John Wiley & Sons.

Shigemasa, T., Iino, Y., Kanda, M., (1986) "Two Degrees of Freedom PID Auto-tuning Controller Based on Frequency Region Methods", Adaptive control strategies for Industrial use - Proceedings of a workshop, Editors: Shah and Dumont, Springer Verlag LNCIS series', 1989.

Smith, O.J.M., (1957) "A Closer Control of Loops with Dead Time", Chem. Eng. Prog., Vol. 53, No. 5, pp.217-219.

Vermeer, P.J., (1987) "Design and Evaluation of Practical Self-Tuning PID Controller", M.Sc. Thesis, University of Alberta, Dept. of Chemical Engineering.

Ziegler, J.G., Nichols, N.B., (1942) "Optimal settings for automatic controllers", Trans. ASME, 64, pp.759-768.

7.3 Chapter 3

Astrom, K.J., Wittenmark, B., Computer Controlled Systems: Theory and Design, Chapter 12, "Optimal Design Methods: Input-Output approach", pp. 294-296, Prentice-Hall International Editions, 1984.

Belanger, P.R. (1983), "On type 1 system and the Clarke-Gawthrop regulator", Automatica, Vol. 19, No. 1, pp. 91-94.

Clarke, D.W., Gawthrop, P.J. (1979), "Self-tuning control", Proc. IEE, 1979, 126, (6), pp.633-640.

Clarke, D.W., Hodgson, A.J.F., Tuffs, P.S. (1983), "Offset problem and k-incremental predictors in self-tuning control", IEE Proc. D, 130, (5), pp. 217-225.

Clarke, D.W. (1984), "Self-tuning control of Nonminimum-phase system:, Automatica, Vol. 20, No. 5, pp.501-517.

Cluett, W.R., Martin-Sanchez, .M., Shah, S.L., Fisher, D.G. (1988), "Stable Discrete-Time Adaptive Control in the Presence of Unmodeled Dynamics", IEEE Trans on Auto control, Vol. 33, No. 4.

Cluett, W.R., Shah, S.L. (1988), "A Robust Adaptive Controller with Time-Delay Mismatch", Adaptive control strategies for Industrial use - Proceedings of a workshop, Editors: Shah and Dumont, Springer Verlag LNCIS series', 1989.

Egardt, B. (1978), "Stability of model-reference adaptive and self-tuning regulators", Lund report LUFD2/(TFRT-1017)/1-163/(1978).

Fessl, J., Karny, M. (1979), "Choice of models for self-tuning regulators", IFAC Symposium on Identification and system parameter estimation, Darmstadt.

Martin-Sanchez, J.M., (1984), "A globally stable APCS in the presence of bounded noises, and disturbances", IEEE Trans. Automat. Contr., Vol. AC-29, pp.461-464.

Harris, T.J., MacGregor, J.F., Wright, J.D. (1980), "Self-tuning and adaptive controllers: an application to catalytic reactor control", Technometrics, 22, 153.

Praly, L., 1983 "Robustness of Model Reference Adaptive Control", in Proc. 3rd Yale Workshop, June, pp.224-226.

Rohrs, C.E., Athans, M., Valavani, L., Stein, G. (1984), "Some Design Guidelines for Discrete-time Adaptive Controllers", IFAC Automatica, Vol. 20, No. 5, pp. 653-660.

Seborg, D.E., Edgar, T.F., Mellichamp, D.A., Process Dynamics and Control, John Wiley & Sons, 1989.

Tuffs, P.S., Clarke, D.W. (1985), "Self-tuning control of offset: a unified approach", Proc. IEE, Vol. 132, No. 3, pp.100-110.

7.4 Chapter 4

Åström, K.J., Wittenmark, L.B. (1984), Computer Controlled Systems, Prentice-Hall, Englewood Cliffs.

Bialkowski (1978), "Application of Steady State Kalman Filter - Theory with Field Results", JACC, Vol. 1, pp.361-374.

Clarke, D.W., Mohtadi, C., Tuffs, P.S. (1987), "Generalized Predictive Control-Part I. The Basic Algorithm", IFAC, Automatica, Vol.23, No.2, pp.137-148.

Clarke, D.W., Mohtadi, C., Tuffs, P.S. (1987), "Generalized Predictive Control-Part II. Extensions and Interpretations", IFAC, Automatica, Vol.23, No.2, pp.137-148.

Clarke, D.W. (1984), "Self-tuning control pf Nonminimum-phase system:, Automatica, Vol. 20, No. 5, pp.501-517.

Cluett, W.R., Shah, S.L. (1988), "A Robust Adaptive Controller with Time-Delay Mismatch", Adaptive control strategies for Industrial use - Proceedings of a workshop, Editors: Shah and Dumo Springer Verlag LNCIS series', 1989.

Cutler, C.R., Ramaker, B.C. (1980) "Dynamic Matrix Control - a computer control algorithm", JACC, San Francisco, paper WP5-B.

De Keyser, R.M.C., Van Cauwenberghe, A. (1979), "A self-tuning multistep predictor application", Automatica, 17, pp.167-174.

De Keyser, R.M.C., Van de Velde, Ph.G.A., Dumortier, F.A.G. (1988),"A Comparative Study of Self-adaptive Long-Range Predictive Control Methods", IFAC, Automatica, Vol.24, No.2, pp.149-163.

Foley, M.W. (1988), "Adaptive Control using a Kalman Filter", M.Sc. Thesis, Dept. of Chem. Eng., Univ. of Alberta, Edmonton, Canada.

Franklin, G.F., Powell, J.D., Digital Control of Dynamics System, Addison - Wesley Publishing Co., 1980.

Garcia, C.E. and M. Morari (1982), "Internal Model Control. 1. A Unifying Review and Some New Results", Ind. Eng. Chem. Process Des. Dev., 21, pp.472-484.

Goodwin, G.C., Sin, K.S. (1984), Adaptive Filtering, Prediction and Control. Prentice-Hall, Englewood Cliffs, New Jersey.

Kalman, R.E. (1960), "A new approach to linear filtering and prediction problems", ASME Trans. D, 82, pp.35-45.

Pappas, T., Laud, A.J., Sandell, N.R.Jr. (1980), "On the Numerical Solution of the Discrete Time Riccati Equation", IEEE Trans. Autom. Contr., AC-25, August, pp.631-41.

Seborg, D.E., Edgar, T.F., Shah, S.L. (1986) "Adaptive Control Strategies for Process Control: A Survey", AIChE Journal, Vol. 32, No. 6, pp.881-913.

Smith, O.J.M. (1957), "A Closer Control of Loops with Dead Time", Chem. Eng. Progr., 53, No.5, pp.217-219.

Sripada, R. (1988), "Multi-Step Adaptive Predictive Control", Ph.D. Thesis, Dept. of Chem. Eng., Univ. of Alberta, Edmonton, Canada.

Sripada, R. and D.G. Fisher (1987). "Improved Least Squares

Identification", Int. J. Control, 46, pp.1889-1913.

Walgama, K.S. (1986), "Multivariable Adaptive Predictive Control for Stochastic System with Time Delays", M.Sc. Thesis, Dept. of Chem. Eng., Univ. of Alberta, Edmonton, Canada.

Ydstie, B.E., Kershenbaum, L.S., Sargent, R.W.H. (1985), "Theory and Application of an Extended Horizon Self-Tuning Controller", AIChE Journal, Vol.31, NO.11, pp.1771-1780.

7.5 Chapter 5

Allworth, ST., Introduction to Real-Time Software Design, Springer-Verlag New York Inc., New York, 1981.

Comer, Douglas, Operating System Design, The XINU Approach, Prentice-Hall, Inc., New Jersey, 1984.

Glass, Robert L., Real-time Software, Prentice-Hall Canada Inc., Toronto, 1983.

Hildebrand, Dan., "Message-Passing Operating Systems", Dr. Dobb's Journal, June, 1988.

"QNX reference guide", Version 2.1, Quantum Software System Ltd., 1988.

Qiu, Z., "MULTICON user manual", 1988, University of Alberta, Canada.

Stephanopoulos, G., Chemical Process Control, An Introduction to Theory and Practice, Prentice-Hall, Inc., New Jersey, 1984.

Tanenbaum, Andrew S., Operating Systems Design and Implementation, Prentice-Hall, Inc., New Jersey, 1987.

Appendix 1
Innovations analysis for MKFP

The process model is

$$X(t+1) = \Phi\ X(t) + \Lambda\ U(t) + \Xi\ \eta_1(t) \qquad (4.2.14)$$

$$Y(t) = H\ X(t) + \eta_2(t) \qquad (4.2.15)$$

Consider steady state Kalman gain for simplicity (ie. steady state value of L(t)=L). From the Kalman filter update in Equations (4.2.4) to (4.2.8), Equation (4.2.14) can be expressed as

$$\hat{X}(t+1) = \Phi\hat{X}(t) + \Lambda U(t) + L\omega(t+1) \qquad (A1.1)$$

where the innovation sequence $\omega(t)$ is defined as

$$\begin{aligned}\omega(t) &= Y(t) - \hat{Y}(t|t-1) \qquad (A1.2)\\ &= Y(t) - H\Phi\hat{X}(t-1) - H\Lambda U(t-1)\end{aligned}$$

By successive substitution, the n^{th} state is given by

$$\begin{aligned}\hat{X}_n(t) = {}& [1-A(z^{-1})]\hat{X}_n(t) + B(z^{-1})U(t-1) + \\ & K_1(z^{-1})\omega(t) + D(z^{-1})\omega(t)/\Delta \qquad (A1.3)\end{aligned}$$

where $K_1(z^{-1}) = l_n + l_{n-1}z^{-1} + \ldots + l_1 z^{-n+1}$

$D(z^{-1}) = l_0[\xi_n z^{-1} + \ldots + \xi_1 z^{-n}]$

The $n+\hat{d}-1^{th}$ state is obtained by further successive substitution:

$$\hat{X}_{n+\hat{d}-1}(t) = z^{-\hat{d}}\hat{X}_n(t) + K_2(z^{-1})\omega(t) \qquad (A1.4)$$

where $K_2(z^{-1}) = l_{n+\hat{d}-1}z^{-1} + \ldots + l_{n+1}z^{-\hat{d}+1}$

Substitute Equation (A1.3) into (A1.4) gives

$$\begin{aligned}\hat{X}_{n+\hat{d}-1}(t) = {}& z^{-\hat{d}}A(z^{-1})^{-1}B(z^{-1})U(t-1) + \\ & z^{-\hat{d}}A(z^{-1})^{-1}K_1(z^{-1})\omega(t) + K_2(z^{-1})\omega(t) + \\ & z^{-\hat{d}}A(z^{-1})^{-1}D(z^{-1})\omega(t)/\Delta \qquad (A1.5)\end{aligned}$$

From Equation (A1.2),

$$\begin{aligned}Y(t) &= \hat{Y}(t|t-1) + \omega(t)\\ &= \hat{X}_{n+\hat{d}-1}(t-1) + \omega(t) \qquad (A1.6)\end{aligned}$$

Substituting Equation (A1.5) into (A1.6) gives

$$\begin{aligned}
Y(t) &= z^{-\hat{d}}A(z^{-1})^{-1}B(z^{-1})U(t-1) + \\
&\quad z^{-\hat{d}}A(z^{-1})^{-1}K_1(z^{-1})\omega(t) + \\
&\quad A(z^{-1})^{-1}A(z^{-1})(1+K_2(z^{-1}))\omega(t) + \\
&\quad z^{-\hat{d}}A(z^{-1})^{-1}D(z^{-1})\omega(t)/\Delta \\
&= z^{-\hat{d}}A(z^{-1})^{-1}B(z^{-1})U(t-1) + \\
&\quad A(z^{-1})^{-1}\left\{A(z^{-1})(1+K_2(z^{-1}))\Delta+z^{-\hat{d}}(K_1(z^{-1})\Delta+D(z^{-1}))\right\}\omega(t)/\Delta \\
&= \frac{B(z^{-1})}{A(z^{-1})}U(t-\hat{d}-1) + \frac{C(z^{-1})}{A(z^{-1})}\omega(t)/\Delta
\end{aligned} \tag{A1.7}$$

$$\text{where } C(z^{-1}) = [A(z^{-1})(1+K_2(z^{-1}))+z^{-\hat{d}}K_1(z^{-1})]\Delta + z^{-\hat{d}}D(z^{-1}) \tag{A1.8}$$

Therefore the state space model can be rewritten in an input/output form as in Equation (A1.7).

APPENDIX 2A

PID+Smith Predictor+Feedforward Control

User Reference

---

1 Introduction

The references for the following three tasks are included in this manual:

(1) PID control with variations.

(2) Smith predictor with load prediction.

(3) Feedforward control.

1.1 PID control

(1) Basic control algorithm (or setpoint on PID):

$$U(t) = U(t-1) + K_c[(1+T_sK_I+K_p/T_s)e(t)-(1+2K_d/T_s)e(k-1)+K_d/T_se(k-2)] \tag{1}$$

U	control output
e	difference between process output and setpoint
K_c	Proportional gain
K_I	Integral gain $(\sec^{-1})$
K_d	Derivative gain (sec)
T_s	sampling time (sec)

(2) Setpoint on PI only control:

$$U(t) = U(t-1) + K_c[e(t)-e(t-1)+K_IT_se(t)+K_d/T_s(-Y(t)+2Y(t-1)-Y(t-2))] \tag{2}$$

Y	process output

(3) Setpoint on I only control:

$$U(t) = U(t-1) + K_c[-Y(t) + Y(t-1) + K_IT_se(t) + K_d/T_s(-Y(t) + 2Y(t-1) - Y(t-2))] \tag{3}$$

1.2 Smith predictor with load prediction

$$\hat{Y}(t+d|t) = \frac{\hat{B}(z^{-1})}{\hat{A}(z^{-1})}U(t) + \left(\frac{1-\alpha}{1-\alpha z^{-1}}\right)\left(Y(t)-\frac{\hat{B}(z^{-1})}{\hat{A}(z^{-1})}z^{-\hat{d}}\right) \tag{4}$$

$\hat{Y}$	predicted output

$\hat{A}$	process model of order one $(1-a_0 z^{-1})$
$\hat{B}$	process model of order one $(b_0+b_1 z^{-1})$
$\hat{d}$	estimated time delay in sample period
α	residual filter constant

1.3 Feedforward

$$U(t) = U(t) + K_{ff}(V(t)-\bar{V}) \qquad (5)$$

K_{ff}	feedforward gain
V	disturbance variable
$\bar{V}$	disturbance variable nominal value

The feedforward signal is added to the control output calculated from the feedback path.

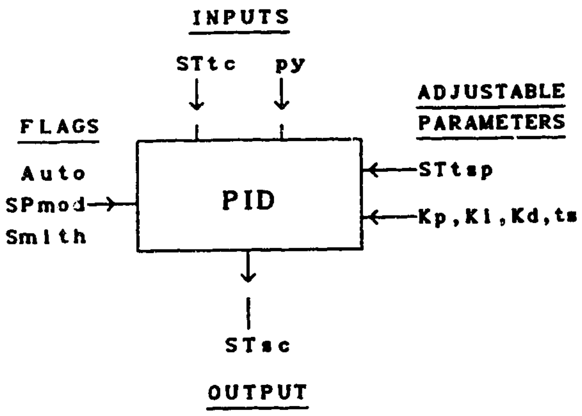
2 How to use PID+Smith Predictor+Feedforward

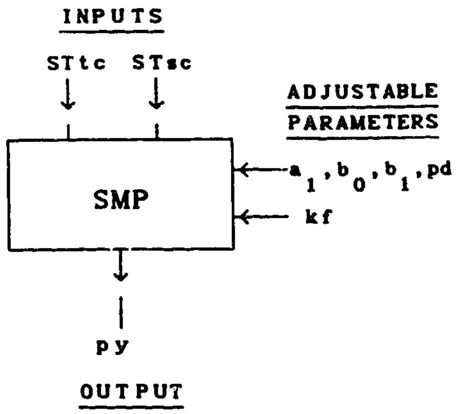
2.1 Program files

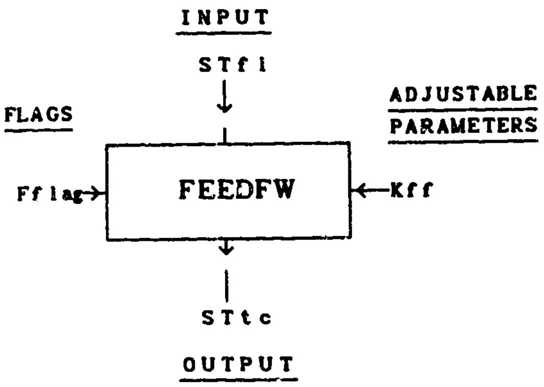
The following files are needed (these files are located in [1]3:/user/eric/pid/):

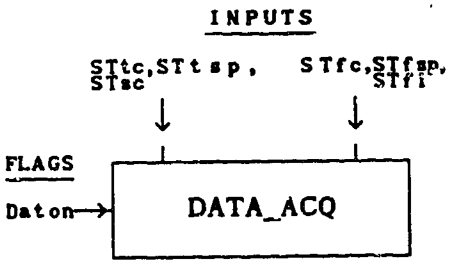
(1) pid : pid control with variations
(2) feedfw : feedforward control
(3) smp : Smith predictor with load prediction
(4) data_acq : data acquisition task
(5) pid2 : second pid control (if needed)
(6) startpi : MULTICON startup file

2.2 Block diagram for each task









2.3 Procedure for starting up

(!) Read MULTICON user manual.
(1) Startup MULTICON.
(2) Use *3:/user/eric/pid* as the working directory and *startpi* as the working file.

2.4 On-line variables

(A) System defined variables

0. STfl : Stirred Tank input water flow (% of 0-1m/s)
1. STtc : Stirred Tank thermo couple (°C)
2.
3.
4. STsc : Stirred Tank steam control valve opening (%)
5. STfc : Stirred Tank input water flow valve opening (%)
6.
7. cnth1: high limit for STsc
8. cnth2: high limit for STfc
9.
10. cntl1: low limit for STsc
11. cntl2: low limit for STfc
12.
13.
14. STtsp: Stirred Tank temperature setpoint (°C)
15. STfsp: Stirred Tank inlet flow setpoint (% on 0-1m/s)
16.

(B) User defined variables

17. Auto : Auto/Manual switch (0=auto; 1=manual)
18. SPmod: PID mode switch
    (0=SP on PID; 1=SP on PI; 2=SP on I only)
19. Smith: Smith Predictor (0=off; 1=on)
20. ts : sampling time in second
    (must match that in task table)
21. kp : proportional constant

22. ki : integral constnat (sec^{-1})
23. kd : derivative constant (sec)
24. a1 : proccess model parameter $(A(z^{-1})=1-a1z^{-1})$
25. b0 : process model parameter $(B(z^{-1})=b_0+b_1z^{-1})$
26. b1 : process model parameter
27. pd : process model delay (in sample period)
28. kf : filter constant (α)
29. py : smith predictor output (used as feedback signal)
30. Auto2: loop 2 Auto/Manual switch (0=auto; 1=manual)
31. kp2 : loop 2 proportional constant
32. ki2 : loop 2 integral constant (sec^{-1})
33. kd2 : loop 2 derivative constant (sec)
34. ts2 : loop 2 sampling time in second
(must match that in task table)
35. Daton: data acquisition flag (0=off; 1=on; 2=over)
36. Kff : feedforward gain
37. Fflag: feedforward flag (0=off; 1=on)

2.5 History files

The data acquisition task stores the following data:

(1) time (in second)
(2) process output
(3) process input (ie. control output)
(4) setpoint
(5) disturbance variable
(6) control action affect the disturbance variable
(7) setpoint to disturbance variable

These data are store to *outfile.log* in directory *3:/user/eric/pid/*.

2.6 Quit PID+Smith Predictor+Feedforward

Modify mnemonic #35 to "over" before issuing "*Goodby*" in MULTICON to quit. This will ensure the history data file is closed properly.

3 Modifying the tasks

3.1 Source files

Source files are residing in [1]3:/user/eric/pid/.

(1) pid.c : source file for task pid.
(2) smp.c : source file for Smith predictor.
(3) feedfw.c : source file for feedforward control.
(4) pid2.c : source file for second pid control.
(5) data_acq.c: source file for data acquisition task.

3.2 Compiling task

This is how a task (eg. pid) can be compiled and linked:

```
#cc pid.c s=1000 -T +8 c=pid
```

Command line explanation:

cc : compile command,
s : stack size,
-T : do not use ram disk for temperate storage,
+8 : generate 8088 co-processor code,
c=pid : name the core file to pid.

APPENDIX 2B

ROBUST ADAPTIVE CONTROLLER (RAC)

USER REFERENCE

---

1 Introduction

The robust adaptive controller is developed by W. Cluett and modified by E. Lau. It is implemented as an advanced control task on MULTICON for signal-input-signal-output control.

1.1 Process Model

$$A(z^{-1})Y(t) = B(z^{-1})z^{-d}U(t-1) + C(z^{-1})\xi(t)/\Delta \qquad (1)$$

Y	Process output
U	Process input
ξ	Disturbance (assumed white noise)
A	Monic polynomial with order Na
B	Polynomial with order Nb
C	Monic polynomial with order Nc
d	Integer time delay ($d \leq 0$)
t	Current sample time (integer)

1.2 Augmented Process

$$Z(t) = \frac{P_n(z^{-1})}{P_d(z^{-1})} Y(t) + \frac{Q_n(z^{-1})}{Q_d(z^{-1})} U(t-d) \qquad (2)$$

Z	Augmented process output
P_n	Weighting polynomial with order dP_n
P_d	Weighting polynomial with order dP_d
Q_n	Weighting polynomial with order dQ_n
Q_d	Weighting polynomial with order dQ_d

1.3 CARIMA model for the augmented process

$$Z(t) = \frac{B'(z^{-1})}{A'(z^{-1})}U(t-d) + \frac{C'(z^{-1})}{A'(z^{-1})\Delta}\xi(t) \qquad (3)$$

A'	$P_d(z^{-1})Q_d(z^{-1})A(z^{-1})$
B'	$P_n(z^{-1})Q_d(z^{-1})B(z^{-1})+P_d(z^{-1})Q_n(z^{-1})A(z^{-1})$
C'	$P_n(z^{-1})Q_d(z^{-1})C(z^{-1})$

1.4 Diophantine Equation

$$\frac{C'(z^{-1})}{A'(z^{-1})\Delta} = E(z^{-1}) + \frac{F(z^{-1})z^{-d}}{A'(z^{-1})\Delta} \qquad (4)$$

E	Monic polynomial with order d-1
F	Polynomial with order $\max(dP_n+dQ_d-1, dP_d+dQ_d+N_a)$

1.5 Cost Function

$$J = \left\{ \frac{R_n(z^{-1})}{R_d(z^{-1})} Y_{sp}(t+d) - Z^*(t+d) \right\}^2 \qquad (5)$$

Z^*	$\mathcal{E}\{Z(t+d \mid t)\}$
R_n	Weighting polynomial with order dR_n
R_d	Weighting polynomial with order dR_d
$\mathcal{E}$	Expectation operator

1.6 Control Law

$$R(z^{-1})Y_{sp}(z^{-1}) = \hat{F}'(z^{-1})\Delta Z^f(t) + \hat{G}(z^{-1})\Delta U^f(t) + C'(1)Z^f(t) \qquad (6)$$

F'	Polynomial with order dF-1 ($F=F'\Delta+C'(1)$)
^	Estimated value
f	Filtered by $T'(z^{-1})=T(z^{-1})P_n(z^{-1})Q_d(z^{-1})$

1.7 Estimator

Projection algorithm with normalization and deadzone.

2 Program Design

The RAC program is written in QNX C that runs on the real-time QNX operating system. The program uses the dynamic memory allocation feature provided by C so that memory is acquired only on request. This is important because on-line adjustments may vary the length of $P_n(z^{-1})$, $P_d(z^{-1})$, $Q_n(z^{-1})$, $Q_d(z^{-1})$, $R_n(z^{-1})$, $R_d(z^{-1})$, $T(z^{-1})$, $F'(z^{-1})$ and $G(z^{-1})$.

2.1 Program Description

The following pseudo-code describes the program operation.

```
#include <stdio.h>
#include "define.h"
#include "matrix.h"
#include "poly.h"
#include "rac_flag.h"
#include "rac_fcn.h"
```

```
main()
begin
    Declare variables;
    Dynamically allocate memory for variables;
    Read data file (initrac) to initialize variables;
    Open history data file;

    wait for the to-go signal;
    repeat for ever
        begin
            receive_from_multicon (all information);
            if ail_done then
                    close all files;
                    break out from infinite loop;
            augment the process variable with P and Q weightings;
            if setpoint_filter is needed
                    filter the setpoint with R weighting;
            form regressor;
            form the k-incremental output;
            normalize regressor and k-incremental output;
            if dimensions of P and Q weightings changed
                    form theta;

            estimate the model;
            if estimator==faulty then
                    switch to manual control;
            if initial_identification==ON then
                    control output= preset output sequence;
            else
                    if automatic==ON then
                            calculate minimum variance
                            incremental control output dU;
                            constraints the outputs;
                    else
                        setpoint tracking;

            send_to_multicon (control output, flags)
            store control signals;

            if history_on_off==ON then
                    write to data file;
        wait until a sample period is up;

        end;

    free all memory allocated to variables;
end;
```

2.2 The initrac file

After the task is started up and memories are allocated to the variables, the program reads from a data file "initrac" to do initialization of variables as well as the MULTICON data table. These are the variables that the user can prespecify for the RAC:

(1) Na, Nb, d
(2) Parameters for $P(z^{-1})$
(3) Parameters for $Q(z^{-1})$
(4) Parameters for $R(z^{-1})$
(5) Parameters for $T(z^{-1})$
(6) Deadzone used for the parameter estimator, Δb

2.3 The history file

Two history files in /user/eric/rac_new/ are open for data acquisition purposes:

(1) "out.m" is for accumulating the process variable, the process manipulated variable, and the setpoint variable.
(2) "the.m" is for accumulating the parameter estimates.

2.4 Initial Identification

For the first 20 sample periods, the controller gives a sequence of step changes around the steady state values of the process input for open loop identification. During this period, the setpoint variable tracks the process manipulated variable. When the 20 sample period is over, the program kicks into automatic mode.

3 How to use RAC

3.1 Program files required

The following files are required to startup and run the RAC task. They are residing in the directory [1]3:/user/eric/rac_new/.

(1) rac : the main RAC task,
(2) initrac : RAC initialization file,
(3) run_rac : MULTICON startup file for RAC.

3.2 Procedure of starting RAC

(!) Read MULTICON user manual.

(1) Edit and modify initrac if necessary.

(2) Change current directory to the multicon directory:

#cd /user/multicon <cr>

(3) Start up MULTICON:

#multicon <cr>

(4) Specify */user/eric/rac_new/* as the working directory and *run_rac* as the working file.

3.3 On-line variables

(A) System defined variables

0. STfl : Stirred Tank input water flow (% of 0-lm/s)
1. STtc : Stirred Tank thermo couple (°C)
2.
3.
4. STsc : Stirred Tank steam control valve opening (%)
5. STfc : Stirred Tank input water flow valve opening (%)
6.
7. cnthl: high limit for STsc
8. cnth2: high limit for STfc
9.
10. cntll: low limit for STsc
11. cntl2: low limit for STfc
12.
13.
14. STtsp: Stirred Tank temperature setpoint (°C)
15. STfsp: Stirred Tank inlet flow setpoint (% on 0-lm/s)
16.

(B) User defined variables

(B.1) Variables for process model:

17. na : degree of A polynomial in process model
18. nb : degree of B polynomial in process model

19. delay: process model time delay (>=0)
20. -----

(B.2) Variables for flags and Switches:

21. done : all finish flag
22. rac : status flag for rac
23. estim: status flag for estimator
24. adapt: adaptive/non-adaptive flag
25. auto : auto/manual flag
26. SPfil: setpoint filter flag
27. ffwd : feedforward flag (not used yet)
28. daton: data acquisition flag
29. -----

(B.3) Variables for P weightings:

30. dPn : degree of P numerator polynomial
31. Pn0 : 1st element of the P numerator polynomial
32. Pn1 : 2nd element of the P numerator polynomial
33. Pn2 : 3rd element of the P numerator polynomial
34. Pn3 : 4th element of the P numerator polynomial
35. dPd : degree of P denominator polynomial
36. Pd0 : 1st element of the P denominator polynomial
37. Pd1 : 2nd element of the P denominator polynomial
38. Pd2 : 3rd element of the P denominator polynomial
39. Pd3 : 4th element of the P denominator polynomial
40. Plag : delay of P weighting

(B.4) Variables for Q weightings:

41. dQn : degree of Q numerator polynomial
42. Qn0 : 1st element of the Q numerator polynomial
43. Qn1 : 2nd element of the Q numerator polynomial
44. Qn2 : 3rd element of the Q numerator polynomial
45. Qn3 : 4th element of the Q numerator polynomial
46. dQd : degree of Q denominator polynomial
47. Qd0 : 1st element of the Q denominator polynomial
48. Qd1 : 2nd element of the Q denominator polynomial

49. Qd2 : 3rd element of the Q denominator polynomial
50. Qd3 : 4th element of the Q denominator polynomial
51. Qlag : delay of Q weighting

(B.5) Variables for R weightings:

52. dRn : degree of R numerator polynomial
53. Rn0 : 1st element of the R numerator polynomial
54. Rn1 : 2nd element of the R numerator polynomial
55. Rn2 : 3rd element of the R numerator polynomial
56. Rn3 : 4th element of the R numerator polynomial
57. dRd : degree of R denominator polynomial
58. Rd0 : 1st element of the R denominator polynomial
59. Rd1 : 2nd element of the R denominator polynomial
60. Rd2 : 3rd element of the R denominator polynomial
61. Rd3 : 4th element of the R denominator polynomial
62. Rlag : delay of R weighting

(B.6) Variables for T weightings:

63. dTn : degree of T numerator polynomial
64. Tn0 : 1st element of the T numerator polynomial
65. Tn1 : 2nd element of the T numerator polynomial
66. Tn2 : 3rd element of the T numerator polynomial
67. Tn3 : 4th element of the T numerator polynomial
68. dTd : degree of T denominator polynomial
69. Td0 : 1st element of the T denominator polynomial
70. Td1 : 2nd element of the T denominator polynomial
71. Td2 : 3rd element of the T denominator polynomial
72. Td3 : 4th element of the T denominator polynomial
73. Tlag : delay of T weighting
74. -----

(B.7) Variables for parameter estimation and control:

75. db : deadband
76. phl : lower bound of deadband
77. phu : upper bound of deadband
78. phiHi: up bound for normalization is phiHi+delay

79. kgain: kalman gain for parameter update
80. chi : variable parameter update weighting
81. dF : degree of F controller polynomial
82. f0 : 1st element of the F polynomial
83. f1 : 2nd element of the F polynomial
84. f2 : 3rd element of the F polynomial
85. f3 : 4th element of the F polynomial
86. dG : degree of G controller polynomial
87. g0 : 1st element of the G polynomial
88. g1 : 2nd element of the G polynomial
89. g2 : 3rd element of the G polynomial
90. g3 : 4th element of the G polynomial

(B.8) Variables for others:

91. z(t) : current augmented plant output
92. yf(t): current filtered STtc output

3.4 Some notes about changing parameters on-line

All of the above variables except STfl, STtc, rac, and estim can be changed on-line by the user by modifying the MULTICON data table. When the user wishes to modify some of the P, Q, R, or T weightings however, it is recommended that the system should be paused before making the changes. This is because MULTICON takes the most update values in the data table to do data acquisition and control. If the user is not able to enter all the changes for the weighting within one sample period, MULTICON will be using part-new part-old data for the weightings. Issuing a pause before changing the weightings avoids this problem.

3.5 Procedure to quit RAC

To quit RAC, modify mnemonic #21 'done' to 'YES'. This will notify the rac task to close all the data files before MULTICON quit. Then issue the '*Goodby*' command in MULTICON to end the session gracefully.

4 Modifying RAC

4.1 Source files

Source files are residing in [1]3:/user/eric/rac_new/. The following files are required to modify and compile the rac task:

(1) rac.c : source file for task rac.
(2) rac_fcn.c : source file for all the subprograms used in rac.
(3) rac_fcn.h : include file for rac subprograms.
(4) rac_flag.h: include file for rac flag.
(5) matrix.h : include file for matrix subprograms.
(6) poly.h : include file for polynomial subprograms.
(7) matrix.o : object file for the matrix subprograms
(8) poly.h : object file for the polynomial subprograms.

The most updated version of the matrix and polynomial subprograms are residing in [1]3:/user/ctools/. The user is recommended to make a copy of the matrix.o, poly.o, matrix.h, and poly.h to the directory containing the rac task to be modified before compiling the task.

4.2 Compiling RAC

The following command line will compile and link the necessary files for the rac task:

```
#cc rac.c rac_fcn.c matrix.o poly.o +Wc,+u s=10000 -T +8 c=rac
```

Command line explanation:

cc : compile command,
+Wc,+u : use definite name for structure variables,
s : stack size,
-T : do not use ram disk for temperate storage,
+8 : generate 8088 co-processor code,
c=rac : name the core file to rac.

Note that the *+Wc,+u* option must be used because all the programs listed above uses structure variables extensively. The stack size must be reasonably large because memories are allocated dynamically to the program variables from this stack. The *-T* option is used whenever there is no ram disk is mounted. The compilation will be slowed down slightly when this option is used.

APPENDIX 2C

MULTISTEP-ADAPTIVE-PREDICTIVE-CONTROL USER MANUAL

---

1 Introduction

The MAPC was originally designed by R. Sripada (1988). It is an adaptive Long Range Predictive Controller which uses a Modified Kalman Filter Predictor (MKFP) for disturbance shaping and prediction, and a multistep controller design to calculate the 'best' control action at each sample interval.

1.1 Process Model

$$X(t+1) = \Phi\, X(t) + \Lambda\, U(t) + \Xi\, \eta_1(t) \tag{1}$$

$$Y(t) = H\, X(t) + \eta_2(t) \tag{2}$$

Y	Process output
U	Process input
X	State vector of order n+d+1
Φ	State matrix of dimension (n+d+1)X(n+d+1)
Λ	Input matrix of dimension (n+d+1)X2
H	Output matrix of dimension 1X(n+d+1)
Ξ	Process noise vector of order n+d+1
η_1	Process noise (assumed white)
η_2	Measurement noise (assumed white)
n	Model order
d	Time delay ($d \geq 0$)

$$X(t) = [x_0(t)\ x_1(t)\ x_2(t)\ \dots\ x_n(t)\ x_{n+1}(t)\ \dots\ x_{n+d}(t)]^t$$

$$\Phi = \begin{bmatrix} 1 & 0 & \dots & 0 & 0 & \dots & 0 & 0 \\ \xi_1 & 0 & \dots & 0 & -a_n & \dots & 0 & 0 \\ \xi_2 & 1 & \dots & 0 & -a_{n-1} & \dots & 0 & 0 \\ \cdot & & & & & & & \\ 0 & & & 1 & -a_1 & \dots & 0 & 0 \\ \cdot & & & & 1 & & & \cdot \\ \cdot & & & & & & & \cdot \\ \cdot & & & & & & & \cdot \\ 0 & 0 & & & \dots & & 1 & 0 \end{bmatrix}_{(n+d+1)x(n+d+1)}$$

$$\Lambda^t = \begin{bmatrix} 0 & b1_n & b1_{n-1} & \dots & b1_1 & 0 & \dots & 0 \\ 0 & b2_n & b2_{n-1} & \dots & b2_1 & 0 & \dots & 0 \end{bmatrix}_{2X(n+d+1)}$$

$H = [0 \dots 0 \; 1]_{1x(n+d+1)}$

$\Xi^t = [1 \; 0 \dots 0]_{1x(n+d+1)}$

a_i, b_i are obtained from parameter estimator, where

$$A(z^{-1}) = 1 + a_1 z^{-1} + a_2 z^{-2} + \dots + a_n z^{-n}$$

$$B(z^{-1}) = b_1 + b_2 z^{-1} + \dots + b_n z^{-n+1}$$

ξ

1.2 Estimation

Any estimator that provides unbiased $A(z^{-1})$ and $B(z^{-1})$.

1.3 Kalman Filter

$$L(t) = M(t)H^T(HM(t)H^T + R_2)^{-1} \tag{3}$$

$$\hat{X}(t) = \overline{X}(t) + L(t)(Y(t) - H\overline{X}(t)) \tag{4}$$

$$P(t) = M(t) - L(t)HM(t) \tag{5}$$

$$\overline{X}(t+1) = \Phi\hat{X}(t) + \Lambda U(t) \tag{6}$$

$$M(t+1) = \Phi P(t)\Phi^T + \Xi R_1 \Xi^T \tag{7}$$

L	Kalman gain vector of order n+d+1
$\hat{X}$	State estimation vector of order n+d+1
P	Covariance matrix of dimension (n+d+1)X(n+d+1)
R_1	$E\{\eta_1(t)\eta_1(t)^t\}$
R_2	$E\{\eta_2(t)\eta_2(t)^t\}$
E	expectation operator

1.4 Prediction

$$\{Y(t+j|t),\ j\in[N_1,N_2]\} = \{H(t)\Phi(t)^j\hat{X}(t|t),\ j\in[N_1,N_2]\} + \{\sum_{i=t}^{t+j-1} H(t)\Phi(t)^{t+j-i-1}\Lambda(t)U(i),\ j\in[N_1,N_2]\} \tag{8}$$

N_1	First point of the control horizon wrt time t
N_2	Last point of the control horizon wrt time t

1.5 Controller Cost Function

$$J = \sum_{j=N1}^{N2} \{Y_{sp}(t+j) - Y(t+j|t)\}^2 \gamma_{Yj}(t) + \sum_{j=1}^{Nu} \{\Delta U(t-j-1)\}^2 \gamma_{Uj}(t) \qquad (9)$$

Nu	Control horizon
Y_{sp}	Setpoint
γ_{Yj}	Weighting on prediction error
γ_{Uj}	Weighting on output

1.6 Control Law

$$\Delta\underline{U}(t) = \left(G(t)^T \underline{\underline{\Gamma}}_Y G(t) + \underline{\underline{\Gamma}}_U\right)^{-1} G(t) \underline{\underline{\Gamma}}_Y [\underline{Y}_{sp}(t) - \underline{Y}^*(t)] \qquad (10)$$

$$\Delta\underline{U}(t) = \{\Delta U(t+i-1),\ i \in [1, Nu]\}$$

$$\underline{\underline{\Gamma}}_Y = \begin{bmatrix} \gamma_{YN1} & \cdots & 0 \\ & \ddots & \\ 0 & \cdots & \gamma_{YN2} \end{bmatrix} \qquad \underline{\underline{\Gamma}}_U = \begin{bmatrix} \gamma_{UN1} & \cdots & 0 \\ & \ddots & \\ 0 & \cdots & \gamma_{UN2} \end{bmatrix}$$

$$\underline{Y}^*(t) = \{H(t)\Phi(t)^j \hat{X}(t|t),\ j \in [N_1, N_2]\} +$$

$$\begin{bmatrix} H(t)\Phi(t)^{N1-1}\Lambda(t) + \ldots + H(t)\Lambda(t) \\ \vdots \qquad\qquad \ddots \\ H(t)\Phi(t)^{N2-1}\Lambda(t) + \ldots + \ldots + H(t)\Lambda(t) \end{bmatrix}_{(N2-N1+1)X1} U(t-1)$$

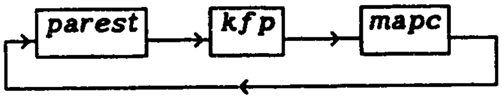
$$G(t) = G'(t)S$$

$$G'(t) = \begin{bmatrix} H\Phi^{N1-1}\Lambda & \ldots & H\Lambda & 0 & & \ldots & 0 \\ H\Phi^{N1}\Lambda & \ldots & & H\Lambda & 0 & \ldots & 0 \\ \vdots & & & & \ddots & & \vdots \\ H\Phi^{N2-1}\Lambda & & \ldots & & & & H\Lambda \end{bmatrix}_{(N2-N1+1)XN2}$$

$$S = \begin{bmatrix} 1 & 0 & \cdots & 0 \\ 1 & 1 & \cdots & \\ 1 & 1 & & 0 \\ \vdots & & \ddots & \vdots \\ 1 & 1 & \cdots & 1 \end{bmatrix}_{N2XNu}$$

2 Program Design

The MAPC is designed for taking 2 inputs from the process (process variable and disturbance variable) and giving 1 control signal. It comprises of three tasks that run in the following sequence:



(1) *parest*: parameter estimation task that gives the process representation in an ARMA model.

(2) *kfp*: the modified Kalman Filter Predictor that accepts the ARMA model from *parest* and gives the predicted state estimates.

(3) *mapc*: the multistep control design task that accepts from *parest* an ARMA model and *kfp* the state estimates to calculate the control output.

These tasks are written in QNX C that runs on the real-time QNX operating system. Dynamic memory allocation feature provided by C is used so that memory is acquired only on request.

2.1 Program description

kfp:

```
#include <stdio.h>
#include "define.h"
#include  "matrix.h"
#include   "kfp_fcn.h"
```

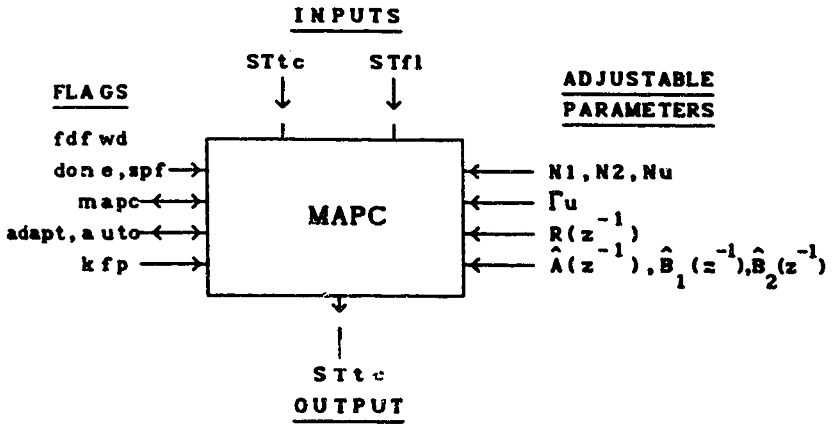
```
main()
begin
     declare variables;
     allocate memory for variables;
     initialize variable (read from initkfp);
     wait for the to-go signal;
     repeat for ever
     begin
          receive_from_multicon (process I/O, model, flags);
          if all_done
                    close history file;
                    go out of infinit loop;
          update state space model;
          Kalman filter;
          if kalman_filter==faulty
                    send_to_multicon (flag);
                    close history file;
                    go out of infinit loop;
          send_to_multicon (state update, flags);
          if history_on_off==ON
                    write_history (state, kalman gain);
          wait until sample period is up;
     end;
free memory allocated for variables;
end.
```

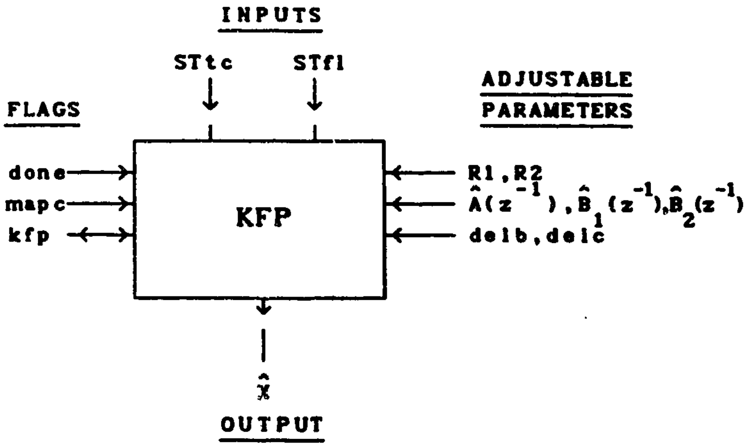
mapc:

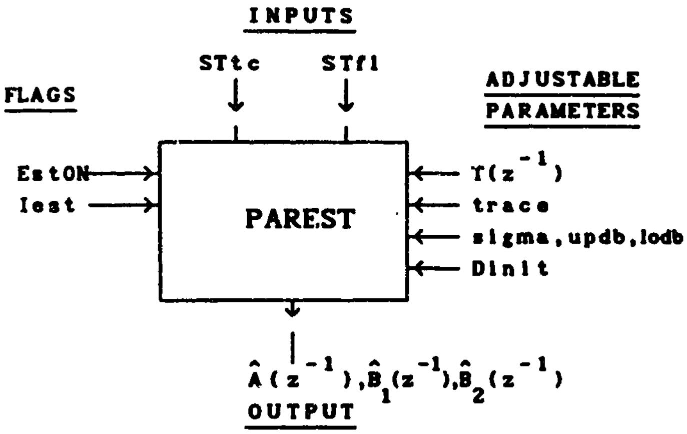
```
#include <stdio.h>
#include "define.h"
#include "matrix.h"
#include "mapc_fcn.h"
main()
begin
     declare  variables;
     allocate memory for variables;
     initialize variable (read from initmapc);
     wait for the to-go signal;
     repeat for ever
     begin
          receive_from_multicon (I/O,states,model,flags,parameters);
          if kalman_filter==faulty
                    switch to manual control;
                    switch to non-adaptive (model not updated);
          if all_done
                    close history file;
                    go out of infinit loop;
          if adaptive==yes
                    update model;
          if automatic==yes
```

```
                setpoint filtering;
                calculate setpoint trajectory;
                calculate control output;
                if calculation==faulty
                        switch to manual;
                        close history file;
                        send_to_multicon (flags);
                constraint control output;
            else
                setpoint tracking;
        send_to_multicon (control output, flags);
        if history_on_off==on
                write to file;
        wait until sample period is up;
    end;
free memory allocated for variables;
end.
```

2.2 Block diagram of each task







2.3 The Init files

The user can prespecify some initial settings for the both the kfp task and the mapc task.

kfp:

(1) Process order, Process delay for the 2 inputs
(2) Initial estimates of $A(z^{-1})$
(3) Initial estimates of $B(z^{-1})$, including $b1_1$ and $b2_1$
(4) Process noise vector
(5) Measurement noise vector
(6) Variance of process noise and measurement noise
(7) Initial state estimates
(8) Initial values for the diagonal elements of the covariance matrix

mapc:

(1) Process order, process delay for the 2 inputs
(2) Initial estimates of $A(z^{-1})$
(3) Initial estimates of $B(z^{-1})$, including $b1_1$ and $b2_1$
(4) Process noise vector
(5) Measurement noise vector
(6) Initial state estimates

2.4 History files

Both kfp task and mapc task opens their own history files which record the following data:

kfp:

(1) All values of the state update
(2) All values of the Kalman gain

The data file is "kfp_out" which is located in \user\eric\mapc_new.

mapc:

(1) Process output
(2) Setpoint
(3) Process input (ie. control output)

(4) parameters for $A(z^{-1})$

(5) parameters for $B(z^{-1})$, including $b1_1$ and $b2_1$

3 How to use MAPC

3.1 Program files

The following files, which are residing in /user/eric/mapc_new, are required to run MAPC:

(1) mapc : Multistep controller execuable image.

(2) kfp : MKFP execuable image.

(3) parest : parameter estimator execuable image.

(4) initkfp : kfp task user initialization file.

(5) initmapc: mapc task user initialization file.

(6) run_mapc: startup file for MAPC.

3.2 Procedure for starting MAPC

(!) Read user manual for MULTICON

(1) Initialize initkfp and initmapc if necessary.

(2) Fire up MULTICON.

(3) Use /user/eric/mapc_new as the working directory and run_mapc as the working data file.

3.3 On-Line variables

(A) System defined variables

0. STfl : Stirred Tank input water flow (% of 0-1m/s)
1. STtc : Stirred Tank thermo couple (°C)
2.
3.
4. STsc : Stirred Tank steam control valve opening (%)
5. STfc : Stirred Tank input water flow valve opening (%)
6.
7. cnth1: high limit for STsc
8. cnth2: high limit for STfc
9.

10. cntl1: low limit for STsc
11. cntl2: low limit for STfc
12.
13.
14. STtsp: Stirred Tank temperature setpoint (°C)
15. STfsp: Stirred Tank inlet flow setpoint (% on 0-1m/s)
16.

(B) User defined variables

17. done : all done flag (0=done; 1=not done)
18. mapc : mapc task status flag (0=ok; 1=not ok)
19. kfp : kfp task status flag (0=ok; 1=not ok)
20. adapt: adaptive flag (0=off; 1=on)
21. auto : automatic flag (0=manual; 1=automatic)
22. spf : setpoint filter flag (0=off; 1=on)
23. daton: data logging flag (0=off; 1=on)
24. fdfwd: feedforward control flag (0=off; 1=on)
25. N1 : first point of prediction horizon (>0)
26. N2 : last point of prediction horizon (>N1)
27. Nu : Control horizon (>0)
28. dRn : degree of $Rn(z^{-1})$
29. Rn0 :
30. Rn1 :
31. Rn2 :
32. dRd : degree of $Rd(z^{-1})$
33. Rd0 :
34. Rd1 :
35. Rd2 :
36. Rlag : setpoint filter delay
37. R1 : process noise variance
38. R2 : measurement noise variance
39. lmda1: γ_{U1}
40. lmda2: γ_{U2}
41. lmda3: γ_{U3}

42. X0 : states estimate from kalman filter
43. X1 :
44. X2 :
45. X3 :
46. X4 :
47. X5 :
48. X6 :
49. X7 :
50. X8 :
51. X9 :
52. X10 :
53. X11 :
54. a1 : $A(z^{-1}) = 1 - a_1 z^{-1} - a_2 z^{-1} - \ldots - a_n z^{-1}$
55. a2 :
56. a3 :
57. a4 :
58. a5 :
59. a6 :
60. a7 :
61. a8 :
62. a9 :
63. b0 : $B(z^{-1}) = b1_0 + b1_1 z^{-1} + \ldots + b1_n z^{-1}$
64. b1 :
65. b2 :
66. b3 :
67. b4 :
68. b5 :
69. b6 :
70. b7 :
71. b8 :
72. c0 : $C(z^{-1}) = b2_0 + b2_1 z^{-1} + \ldots + b2_n z^{-1}$
73. c1 :
74. c2 :

75. c3 :
76. c4 :
77. c5 :
78. c6 :
79. c7 :
80. c8 :
81. na : number of a parameters
82. nb : number of b parameters
83. nc : number of c parameters
84. delb : process time delay
85. delc : disturbance time delay
86. Tn0 : T-filter used in parest
87. Tn1 :
88. Tn2 :
89. Td0 :
90. Td1 :
91. Td2 :
92. trace: trace of covariance matrix for parest
93. Sigma: Ydstie's forgetting factor tuning knob
94. updb : upper bound for deadband
95. lodb : lower bound for deadband
96. Dinit: Initial diagonal element of UD matrix
97. EstON: Estimator flag (0=off; 1=on)
98. Iest : parest Initialization flag (0=default)

3.4 Procedure to quit MAPC

To quit MAPC, modify mnemonic #17 'done' to 0. This will notify both mapc task and kfp task to close all the data files before MULTICON quit. Then issue the '*Goodby*' command in MULTICON to end the session gracefully.

4 Modifying MAPC

4.1 Source files

Source files are residing in [1]3:/user/eric/mapc_new/. The following files are required to modify and compile the MAPC tasks:

(1) mapc.c : source file for task mapc.
(2) mapc_fcn.c: source file for all subprograms used in mapc.
(3) mapc_fcn.h: include file for mapc subprograms.
(4) kfp.c : source file for task rac.
(5) kfp_fcn.c : source file for all subprograms used in kfp.
(6) kfp_fcn.h : include file for kfp subprogram.
(7) flag.h : include file for mapc and kfp flags.
(8) matrix.h : include file for matrix subprograms.
(9) matrix.o : object file for the matrix subprograms

The most updated version of the matrix subprograms are residing in [1]3:/user/ctools/. The user is recommended to make a copy of the matrix.o and matrix.h to the directory containing the MAPC tasks before compiling the tasks.

4.2 Compiling MAPC

The following command line will compile and link the necessary files for the mapc task:

```
#cc mapc.c mapc_fcn.c matrix.o +Wc,+u s=10000 -T +8 c=mapc
```

Command line explanation:

cc : compile command,
+Wc,+u : use definite name for structure variables,
s : stack size,
-T : do not use ram disk for temperate storage,
+8 : generate 8088 co-processor code,
c=mapc : name the core file to mapc.

Note that the *+Wc,+u* option must be used because all the programs listed above uses structure variables extensively. The stack size must be reasonably large because memories are allocated dynamically to the program variables from this stack. The *-T* option is used whenever the ram disk is not mounted. The compilation will be slowed down slightly when this option is used.