

https://dspace.org/

The implementation of a generalised robust adaptive controller

Bergesen, Michel Ludvic

2015-01-08
http://hdl.handle.net/10539/16246

Downloaded from a DSpace repository. https://dspace.org/

THE IMPLEMENTATION OF A GENERALISED ROBUST ADAPTIVE CONTROLLER

Michel Ludvic Bergesen
Department of Electrical Engineering

A dissertation submitted to the Faculty of Engineering, University of the Witwatersrand, Johannesburg, in fulfilment of the requirements for the Degree of Master of Science in Engineering.

Johannesburg, 1988

DECLARATION

I declare that this dissertation is my own unaided work. It is being submitted for the Degree of Master of Science in Engineering at the University of the Witwatersrand, Johannesburg. It has not been submitted before for any degree or examination at any other university.

M. L. Bergesen

5th day of May 1988

ACKNOWLEDGEMENTS

I would like to express my thanks to:

- Prof. I.M. Macleod in the capacity of supervisor for this research, for his continued support and interest as well as contributing considerable time to clarify the underlying theory.

- The University of the Witwatersrand, Johannesburg, and the Council for Scientific and Industrial Research's Foundation for Research and Development (FRD) for financial assistance during the course of this work.

- The Johannesburg Consolidated Investment Company Limited, and in particular the Group Education Committee, for their continued interest and financial support of this research, as well as for granting me sufficient time to complete this work.

ABSTRACT

An adaptive controller is developed, comprising a robust parameter estimator and an explicit pole assignment controller design. The controller is reformulated to have a standard PID structure. A practical implementation is facilitated on a digital microcomputer, connected to a physical process. Test results are presented for this real process subject to variable dead-time and an external disturbance. Simulation results are also presented, for a nominally nonminimum-phase process subject to variable dead-time and large open-loop gain changes. Robust performance is demonstrated under all of these circumstances. Recommendations are given for the choices and considerations required in a robust practical implementation.

SYNOPSIS

Much research has been done in the field of adaptive control over the past few decades. However, a lot needs to be learned about the **robustness** of adaptive control algorithms. This research investigates the implementation of a practical adaptive control algorithm, with numerous features incorporated to improve the robust performance of such a controller. Parameter estimation is performed using Recursive Least Squares (RLS), with various signal conditioning filters to reduce estimator sensitivity to noise and modelling errors. The control design is based on closed-loop pole assignment, with adaptive feedforward compensation included. Further, provision is made in both the estimation model and the feedback control structure to eliminate deterministic unmeasurable disturbances, and to track deterministic setpoint variations. This is based on the Internal Model Principle. Measured random disturbance signals are included in the estimation model, for which "transfer function" polynomial coefficients are estimated and then used in the feedforward control design. A new shift- operator, namely the δ-operator, is used in all controller and estimator formulations. This has been shown to have better numerical properties and to correspond more closely to continuous-time control, than the traditional q^{-1} operator of z-domain discrete control. A practical implementation on a digital computer is investigated, applied to a real plant typical of an industrial application. Simulation results are also obtained for plant with nonminimum-phase zeros and variable dead-time.

TABLE OF CONTENTS

THE IMPLEMENTATION OF A GENERALISED ROBUST ADAPTIVE CONTROLLER. 1

1.0 INTRODUCTION 3
1.1 Research Goal 6
1.2 Research Approach 7
1.2.1 Real Plant 7
1.2.2 Simulation Tests 7
1.3 Structure of Dissertation 8

2.0 CURRENT APPROACHES TO ADAPTIVE ESTIMATION AND CONTROL 9
2.1 Introduction 9
2.2 Parameter Estimation 11
2.2.1 Linear Difference Equation Models 11
2.2.2 Online Estimation Schemes 12
2.2.3 Equation Error Estimation Methods 13
2.2.4 Projection Algorithm 13
2.2.5 Recursive Least Squares Algorithm 14
2.2.6 Modifications to RLS Estimation 16
2.2.6.1 Exponential Data Weighting 16
2.2.6.2 Covariance Resetting 17
2.2.6.3 Covariance Modification 17
2.2.6.4 Regularised Constant Trace Algorithms 18
2.2.7 Output Error Methods 19
2.2.8 Estimation in the Presence of Bounded Noise 20
2.2.9 Constrained Estimation 20
2.2.10 Numerical Sensitivity 21
2.2.11 Modified Estimation Schemes 22
2.3 Control Design Strategies 23
2.3.1 Minimum Prediction Error Controllers 23
2.3.1.1 One-Step-Ahead Control 24

2.3.1.2 Weighted One-Step-Ahead Control 25
2.3.1.3 Model-Reference Control 26
2.3.2 Pole Assignment Controllers 29
2.3.2.1 Rapprochement with Minimum Prediction Error Control . 30
2.4 Conclusion 31

3.0 DELTA-OPERATOR FORMULATION OF DISCRETE-TIME CONTROL 33

4.0 ROBUST ESTIMATOR STRUCTURE 41
4.1 Model structure 41
4.2 Relative Deadzone 48
4.3 The Recursive Least Squares Algorithm 51
4.4 Numerical Implementation 52
4.5 Covariance modification 52

5.0 ROBUST CONTROLLER STRUCTURE 53
5.1 Introduction 53
5.2 Control Synthesis 54
5.3 Conclusion 58

6.0 PRACTICAL IMPLEMENTATION CONSIDERATIONS 61
6.1 Introduction 61
6.2 General Choices 61
6.2.1 Selection of Sampling Period 61
6.2.2 Selection of Model Order 63
6.3 Implementation of the Robust Estimator 63
6.3.1 Input-Output Signal Normalisation 63
6.3.2 Signal Conditioning 64
6.3.2.1 Low-Pass Filtering 64
6.3.2.2 High-Pass Filtering 66
6.3.2.3 Initialisation of Discrete Filters 70
6.3.3 Deadzone Parameter Selection 70
6.3.4 Other Parameters 74
6.3.4.1 Choice of Forgetting Factor 74
6.3.4.2 Choice of Initial Values 75

6.4 Implementation of the Robust Controller 76
6.4.1 Pole Placement Design 76
6.4.2 Interpretation as a Digital PID Controller 80
6.4.3 Choice of Closed-loop Characteristic Polynomial 87
6.5 Software Considerations 92
6.5.1 Sequencing 92
6.5.2 Algorithm Implementation 94
6.6 Conclusion 94

7.0 EXPERIMENTAL RESULTS 95
7.1 Introduction 95
7.2 Physical Plant Configuration 95
7.3 Experiment 1 97
7.4 Experiment 2 105
7.5 Experiment 3 119
7.6 Experiment 4 124
7.7 Experiment 5 133
7.8 Experiment 6 143

8.0 CONCLUSION AND RECOMMENDATIONS 155
8.1 Towards an Industrial Implementation 158

9.0 APPENDIX A : CONVERGENCE OF RLS ESTIMATOR WITH DEADZONE 161

10.0 APPENDIX B : BIERMAN'S UDU COVARIANCE FACTORIZATION 169

11.0 APPENDIX C : POLE ASSIGNMENT FOR HIGHER-ORDER CONTROLLERS 177

12.0 APPENDIX D : PID APPROXIMATIONS FROM POLE ASSIGNMENT 189

13.0 APPENDIX E : SOFTWARE LISTING 193

14.0 REFERENCES 211

LIST OF ILLUSTRATIONS

Figure 1. Block Diagram of a Generalised Adaptive Controller. . . 10
Figure 2. Block Diagram of Model Reference Control System. . . . 28
Figure 3. Closed-Loop System Structure 30
Figure 4. The Inverse Delta Operator as an Integrator 36
Figure 5. Block Diagram implementation of the Delta-Operator . . 37
Figure 6. The Stability region in the Gamma-Plane 38
Figure 7. Frequency Response of High pass Filter for Robust Estimation 45
Figure 8. The Basic Relative Deadzone Function 49
Figure 9. Closed-Loop System Structure 56
Figure 10. Practical Implementation of the 1/E Filter 65
Figure 11. Frequency "Window" for Method of Differencing 68
Figure 12. Frequency "Window" for High-Pass Filter 69
Figure 13. Block Diagram of Discrete High-Pass Filter 71
Figure 14. Relative Deadzone Function showing scaling 74
Figure 15. Closed-Loop System with Controller 80
Figure 16. Frequency Response of PID Controller 83
Figure 17. Z-Plane Mappings for Forward and Backward Difference Derivative Approximations 85
Figure 18. State Feedback Interpretation of Pole Assignment . . . 91
Figure 19. Digital Control Algorithm Sequence 93
Figure 20. Physical Process and Digital Controller 96
Figure 21. Pole Locations of Closed Loop Polynomial 100
Figure 22. Setpoint Sequence for Experiment 1 102
Figure 23. System Output y(t) for Experiment 1 102
Figure 24. Control Input u(t) for Experiment 1 103
Figure 25. Deadzone Size for Experiment 1 103
Figure 26. Estimation Gain Sequence a(k) for Experiment 1 . . . 104
Figure 27. Prediction Error and Deadzone Size for Experiment 1 . 104
Figure 28. Prediction Error e(t) for Experiment 1 105
Figure 29. PID Gain Constant CK1 for Experiment 1 105
Figure 30. PID Integral Time Constant CK2 for Experiment 1 . . . 106

Figure 31. PID Derivative Time Constant CK3 for Experiment 1 . . 106
Figure 32. Estimated Parameter a0 for Experiment 1 107
Figure 33. Estimated Parameter b0 for Experiment 1 107
Figure 34. Estimated Parameter a1 for Experiment 1 108
Figure 35. Estimated Parameter b1 for Experiment 1 108
Figure 36. Setpoint Sequence for Experiment 2 112
Figure 37. System Output y(t) for Experiment 2 112
Figure 38. Control Input u(t) for Experiment 2 113
Figure 39. Deadzone Size for Experiment 2 113
Figure 40. Estimation Gain Sequence a(k) for Experiment 2 . . . 114
Figure 41. Prediction Error and Deadzone Size for Experiment 2 . 114
Figure 42. Prediction Error e(t) for Experiment 2 115
Figure 43. PID Gain Constant CK1 for Experiment 2 115
Figure 44. PID Integral Time Constant CK2 for Experiment 2 . . . 116
Figure 45. PID Derivative Time Constant CK3 for Experiment 2 . . 116
Figure 46. Estimated Parameter a0 for Experiment 2 117
Figure 47. Estimated Parameter b0 for Experiment 2 117
Figure 48. Estimated Parameter a1 for Experiment 2 118
Figure 49. Estimated Parameter b1 for Experiment 2 118
Figure 50. Setpoint Sequence for Experiment 3 121
Figure 51. System Output y(t) for Experiment 3 121
Figure 52. Control Input u(t) for Experiment 3 122
Figure 53. PID Gain Constant CK1 for Experiment 3 122
Figure 54. PID Integral Time Constant CK2 for Experiment 3 . . . 123
Figure 55. PID Derivative Time Constant CK3 for Experiment 3 . . 123
Figure 56. Setpoint Sequence for Experiment 4 126
Figure 57. System Output y(t) for Experiment 4 126
Figure 58. Control Input u(t) for Experiment 4 127
Figure 59. Deadzone Size for Experiment 4 127
Figure 60. Estimation Gain Sequence a(k) for Experiment 4 . . . 128
Figure 61. Prediction Error and Deadzone Size for Experiment 4 . 128
Figure 62. Prediction Error e(t) for Experiment 4 129
Figure 63. PID Gain Constant CK1 for Experiment 4 129
Figure 64. PID Integral Time Constant CK2 for Experiment 4 . . . 130
Figure 65. PID Derivative Time Constant CK3 for Experiment 4 . . 130
Figure 66. Estimated Parameter a0 for Experiment 4 131

Figure 67. Estimated Parameter b0 for Experiment 4 131
Figure 68. Estimated Parameter a1 for Experiment 4 132
Figure 69. Estimated Parameter b1 for Experiment 4 132
Figure 70. Setpoint Sequence for Experiment 5 136
Figure 71. System Output y(t) for Experiment 5 136
Figure 72. Control Input u(t) for Experiment 5 137
Figure 73. Deadzone Size for Experiment 5 137
Figure 74. Estimation Gain Sequence a(k) for Experiment 5 . . . 138
Figure 75. Prediction Error and Deadzone Size for Experiment 5 . 138
Figure 76. Prediction Error e(t) for Experiment 5 139
Figure 77. PID Gain Constant CK1 for Experiment 5 139
Figure 78. PID Integral Time Constant CK2 for Experiment 5 . . . 140
Figure 79. PID Derivative Time Constant CK3 for Experiment 5 . . 140
Figure 80. Estimated Parameter a0 for Experiment 5 141
Figure 81. Estimated Parameter b0 for Experiment 5 141
Figure 82. Estimated Parameter a1 for Experiment 5 142
Figure 83. Estimated Parameter b1 for Experiment 5 142
Figure 84. Setpoint Sequence for Experiment 6 147
Figure 85. System Output y(t) for Experiment 6 147
Figure 86. Control Input u(t) for Experiment 6 148
Figure 87. Deadzone Size for Experiment 6 148
Figure 88. Estimation Gain Sequence a(k) for Experiment 6 . . . 149
Figure 89. Prediction Error and Deadzone Size for Experiment 6 . 149
Figure 90. Prediction Error e(t) for Experiment 6 150
Figure 91. PID Gain Constant CK1 for Experiment 6 150
Figure 92. PID Integral Time Constant CK2 for Experiment 6 . . . 151
Figure 93. PID Derivative Time Constant CK3 for Experiment 6 . . 151
Figure 94. Estimated Parameter a0 for Experiment 6 152
Figure 95. Estimated Parameter b0 for Experiment 6 152
Figure 96. Estimated Parameter a1 for Experiment 6 153
Figure 97. Estimated Parameter b1 for Experiment 6 153
Figure 98. Controller Form Implementation 184
Figure 99. Controller Form with Cascaded Integrator 185
Figure 100. Observer Form 186
Figure 101. ARMA Block Diagram Implementation of Controller . . 188

THE IMPLEMENTATION OF A GENERALISED ROBUST ADAPTIVE CONTROLLER.

1.0 INTRODUCTION

In simple terms, to "adapt" means to change behaviour to conform to new circumstances. In the last two decades, there has been much interest in control systems *which automatically adapt* or change themselves in response to variations in process dynamics or disturbance conditions. Since ordinary feedback control is intended to achieve the same purpose, a stricter definition of adaptive control is desirable. While there is consensus that constant-gain feedback does not constitute an adaptive system, a clear definition has not been *given*. *Seborg, Edgar and* Shah (1986) simply state that an adaptive controller is one for which a satisfactory fixed model of the process to be controlled is not available *in advance*, due to uncertain behaviour, nonlinearities or time-varying system dynamics. Astrom (1987) takes the pragmatic view that adaptive control can be considered a specialised form of nonlinear feedback control, where the process states are seen in two categories, which change at different rates. The more slowly varying states are viewed as model parameters.

It is worthwhile to study the practical motivation for adaptive systems in more detail. Fixed servomechanism controller design depends on reasonably good system modelling. Since these models are difficult to generate in practice, a *generalised three-term* (PID) controller is frequently used in industrial applications, which can give aceptable performance when the corresponding controller coefficients are correctly set or "tuned" (Clarke). However, accurate tuning is often difficult or impossible due to the following factors :

- Many chemical processes have very complex dynamic characteristics, due to large phase lags or dead time. High-order mathematical descriptions of such processes are often necessary for accurate modelling.

- Nonlinear characteristics may alter the plant behaviour dramatically with large setpoint variations, eg. in pH control the incremental gain can change by many decades over a given pH range. Furthermore, actuators such as valves exhibit nonlinear effects, viz. saturation, hysteresis and dead-bands.

- The process dynamics themselves may be time-varying, for example during the decay of a catalytic reaction, or with the gradual fouling of physical plant such as pipes, ducts, filters or heat exchangers.

- When interacting control loops are tuned independantly, coupling effects may be such that tight control performance is not possible.

- Large disturbances are often present, such as variations in the material input to a process, as with ore milling and crushing. Environmental disturbances are also present, such as changes in ambient air temperature or coolant fluid temperature.

Adaptive control formulation goes some way towards a solution in these difficult circumstances. A procedure is employed whereby an estimated model of the process is generated online, and a design procedure followed to produce a suitable set of controller coefficients for some prespecified controller structure. Thus the system adapts itself to unknown or time-varying circumstances, incorporating many of the problems mentioned above.

This concept is intuitively appealing because of its close kinship with human capabilities for adaptation, and its connotations of "artificially intelligent" systems associated with the mystique of automatic computer control. The advent of inexpensive computer hardware possessing substantial processing power has enabled the implementation of complex online control algorithms, using supervisory real-time minicomputers. The emergence of sophisticated programmable logic controllers (PLC's) and other dedicated microprocessor-based devices has also allowed for flexible, rapid control system implementation. These factors possibly account for the widespread interest and research in the field of adaptive process

control. However, in the application of such a powerful and appealing concept, many problems have arisen. The vast literature on the subject addresses problems such as estimator convergence, parameter accuracy, long-term closed-loop stability, disturbance rejection characteristics and application to generalised systems with arbitrary structure, in particular nonminimum-phase processes.

Many constraints are imposed in the literature on the types of processes to be controlled, the nature of system excitation, knowledge of disturbance and noise characteristics, and restrictions on model classes. The large amount of prior process knowledge required by some formulations negates many of the advantages hoped for with adaptive systems.

Certain control design procedures have also been shown to be non-robust under general circumstances. An example of this is the well known minimum-variance controller, which minimises a quadratic cost function involving the system input, output and disturbances to achieve minimal variance of the controlled variable. Unfortunately, this attempt to minimize output variance often results in excessive control effort, which can represent a problem from an engineering point of view, due to actuator limitations. Another very popular technique used in adaptive control is that of model reference or model- following control. Part of the model reference design procedure is to assign the open-loop system zeros to the closed loop system poles, thereby cancelling these zeros. This has the implicit constraint that the open-loop zeros must lie in the left-hand half plane, for assymptotic stability of the closed loop system. Thus nonminimum phase systems cannot be controlled by this type of controller, which is a severe limitation.

A need has thus arisen for a generalised robust theory, giving reliable sytem identification under a wide range of process conditions, as well as flexible and stable closed-loop control for a large class of industrial processes. Furthermore, such a theory should be self-contained and simple enough to give satisfactory performance with a minimum of user-supplied information. Although this research has not provided a complete solution,

many of the robustness problems and stringent process requirements observed in the literature have been removed.

1.1 RESEARCH GOAL

This research has attempted to formulate a robust algorithm for the control of general time-varying systems, allowing for a broader class of open loop transfer functions. The algorithm consists of a parameter estimator, a controller design procedure and a controller. The parameter estimator is *made* robust to modelling error, process noise and deterministic disturbance inputs. The control design is based on closed-loop pole assignment, which is shown by *many authors* to be more robust than other strategies. (See "Current Approaches to Adaptive Estimation and Control" on page 9)

More specifically, this research investigates the practical implementation of such a controller, involving the combination of the following concepts:

- A new transform domain for control formulation.
- Signal conditioning (prefiltering) to improve parameter estimation robustness.
- A generalised explicit process model for parameter *estimation*, incorporating both unmeasured deterministic disturbances and measurable load disturbances.
- A robust pole assignment technique for control design.
- Explicit feedforward compensator design for measurable disturbance *rejection*.

1.2 RESEARCH APPROACH

The experimental implementation was evaluated using both a real physical process and a computer simulated plant model.

1.2.1 REAL PLANT

The physical implementation is facilitated using a microcomputer linked through an analog/digital converter to a real plant typical of an industrial situation. The adaptive controller algorithm performance is evaluated in this environment in terms of:

- Estimator robustness, measured by parameter convergence and/or stability under various conditions.
- Closed-loop control performance under both setpoint and load disturbance changes, as well as plant parameter variations such as open-loop gain.
- Deterministic disturbance rejection and variable dead-time compensation under closed-loop control.

1.2.2 SIMULATION TESTS

The algorithms are also tested using simulated plant models, to allow control testing for nonminimum-phase plants and those with variable dead-time.

1.3 STRUCTURE OF DISSERTATION

The layout of this report is as follows :

- Section 2 discusses the current approaches to adaptive estimation and control.
- Section 3 reviews some of the mathematical preliminaries necessary to the development of the algorithm, notably those relating continuous-time to discrete-time system representation.
- Section 4 develops the structure for the robust estimator.
- Section 5 develops the theory behind the robust pole assignment controller.
- Section 6 discusses the robust implementation requirements of the algorithm on a real plant, as regards estimator parameter selection, signal conditioning and robust controller design.
- Section 7 presents experimental results for the real plant under closed-loop control, as well as simulation results for a few test cases.
- Section 8 concludes the report and presents some recommendations for further work.

2.0 CURRENT APPROACHES TO ADAPTIVE ESTIMATION AND CONTROL

2.1 INTRODUCTION

Adaptive controllers were motivated initially by autopilot design requirements for high performance aircraft and rockets (Seborg et al, 1986). A simple form of adaptation is known as **gain scheduling** whereby the controller gain K_c is varied in response to changes in process gain K_p, so as to keep the product K_pK_c constant. This has found successful application particularly in pH control, and in fact standard industrial controllers with gain scheduling options are commercially available (Andreiev, 1981). However, gain scheduling is not always effective in the presence of varying plant dynamics or unknown time delays. Thus current approaches are developed to cope with unpredictable process changes, by relating the controller design to plant behaviour as indicated by online process variable measurements.

Adaptive controllers are typically constituted of a parameter estimator in combination with some feedback control design procedure. This is shown in Figure 1 on page 10. Model parameters are derived by examining the input-output characteristics of the process. The approach of using the model estimates thus derived as if they were the true plant parameters for the purposes of control design, is known as **certainty equivalence** adaptive control. A wide variety of such schemes have been proposed in the theory, by combining one of the many estimation techniques with a particular controller design method.

There are two possible formulations of the adaptive controller shown in Figure 1 on page 10. As it is detailed in the block diagram, it is an **explicit** scheme, since an explicit process model is estimated and these parameters used in the control design. This technique is also called **indirect**, since the control law is indirectly obtained from a system model. The other formulation is in cases when the system can be direcly

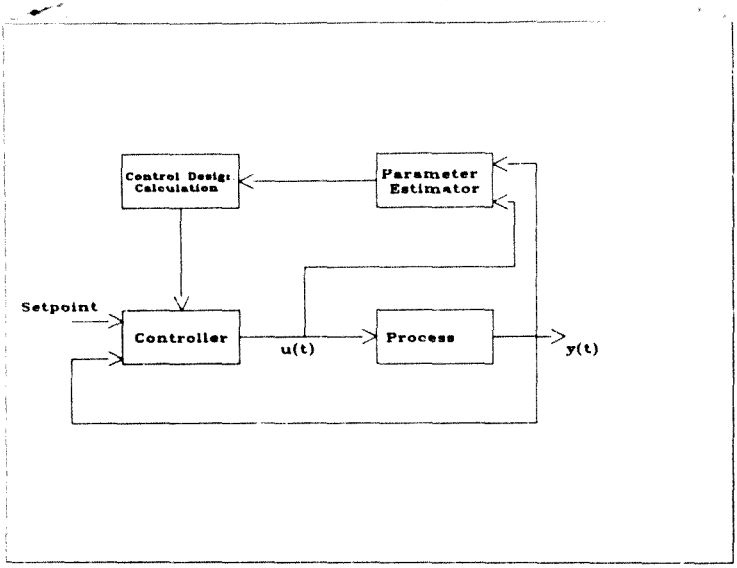

Figure 1. Block Diagram of a Generalised Adaptive Controller.

parameterised in terms of the control law parameters. This greatly simplifies the design calculations required for the controller. Such algorithms are known as **direct**, since the controller coefficients are directly estimated. These schemes are also called **implicit**, because the design includes an implicit process model.

This section first considers some parameter estimation schemes, and then details various types of control law design approaches with which the estimators can be used.

2.2 PARAMETER ESTIMATION

These procedures are employed to identify a system model which adequately describes process behaviour in terms of dynamic and steady-state response. Two main groups of algorithms can be used, namely **online** and **offline** techniques. Offline algorithms consider measured historical plant data in "batches", and a one-pass solution for model parameters is obtained (Junkins, 1981 : 7). The larger the number of parameters to be estimated, the bigger "blocks" of plant data need to be given to the estimator for reliable identification. The second group of **online** algorithms is of greater interest in this work, since the plant model is estimated using past and present process variable measurements at each sample. The estimates are thus a function of the process "history", described as a regression vector, and are continually updated online. This allows the model to track process behaviour, which is relevant in cases with time-varying plant dynamics.

2.2.1 LINEAR DIFFERENCE EQUATION MODELS

A typical form of process model for the purposes of adaptive control is known as the ARMAX (Autoregressive Moving Average with Auxiliary Input) model, given by Seborg et al (1986) as

$$y(t) + a_1 y(t-1) + \ldots + a_n y(t-n) = b_0 u(t-k) + b_1 u(t-k-1) + \ldots + b_m u(t-k-m)$$

$$+ c_0 \xi(t) + c_1 \xi(t-1) + \ldots + c_n \xi(t-n) + d(t) \qquad (2.1)$$

where y is the plant output, u is the input, ξ is a noise variable and d is the load disturbance variable. The sampling time instant is denoted by t, n and m are positive integers giving the order of the model, and k is the time delay as an integral multiple of the sampling period. The model parameters a_i, b_i and c_i are generally unknown, and need to be estimated.

The model is more compactly expressed using shift operator notation, thus:

$$A(q^{-1})y(t) = B(q^{-1})u(t-k) + C(q^{-1})\xi(t) + d(t) \qquad (2.2)$$

where $A(q^{-1}) = 1 + \sum_{i=1}^{n} a_i q^{-i}$

$$B(q^{-1}) = \sum_{i=0}^{m} b_i q^{-i}$$

$$C(q^{-1}) = \sum_{i=0}^{n} c_i q^{-i}$$

The coefficients c_i denote a discrete filter which models the process noise.

These linear discrete models are suitable for adaptive control because they yield algorithms easily implemented on a digital computer.

2.2.2 ONLINE ESTIMATION SCHEMES

The general form for an online parameter estimation scheme is given in Goodwin and Sin (1984 : 49) as

$$\theta(t) = \theta(t-1) + M(t-1)\phi(t-d)e(t) \qquad (2.3)$$

Where $\theta(t)$ is the parameter estimate at sample instant t, comprising a vector of a_i, b_i and c_i coefficients.

$M(t-1)$ is the estimation algorithm gain, possibly a matrix.

$\phi(t-d)$ is a regression vector of historical process measurements.

$e(t)$ is the modelling error (often the prediction error

resulting from $\theta(t-1)$).

2.2.3 EQUATION ERROR ESTIMATION METHODS

These are based on the predictive model

$$y(t) = \phi(t-1)\theta(t-1) \qquad (2.4)$$

where $y(t)$ is system output
$\phi(t-1)$ is a history vector of plant measurements
$\theta(t-1)$ is the estimated parameter vector

Such algorithms are generally used for deterministic adaptive estimation.

2.2.4 PROJECTION ALGORITHM

The first algorithm considered is also the simplest. The projection or gradient algorithm is given by

$$\theta(t) = \theta(t-1) + \frac{a\phi(t-1)}{\phi(t-1)^T\phi(t-1) + c} \cdot e(k)$$

This is of the form of 2.3, where

$$M(t-1) = \frac{a}{\phi(t-1)^T\phi(t-1) + c}$$

$$e(t) = y(t) - \phi(t-1)^T\theta(t-1)$$

and sample delay d = 1.

This is motivated geometrically, by minimising a quadratic error criterion (Goodwin and Sin 1984 : 51) :

$$J = \tfrac{1}{2}|\theta(t) - \theta(t-1)|^2 \qquad (2.6)$$

The symbols all have the same meaning as before, with constant $c > 0$ to prevent possible division by zero in $M(t-1)$, and gain factor a, typically chosen in the range $0 < a \leq 1$.

The algorithm is also sometimes called **Normalised Least Mean Squares (NLMS).** Although convergence is assured to a set of parameters θ, these may not be the correct process model parameters.

2.2.5 RECURSIVE LEAST SQUARES ALGORITHM

This develops from the projection algorithm, by ensuring that each successive estimate projects in a direction **orthogonal** to previous $\phi(.)$ vectors, which is shown to improve estimates (Goodwin and Sin, 1984 : 54). The algorithm resulting dates from the time of Gauss, and is given by

$$\theta(t) = \theta(t-1) + \frac{P(t-2)\phi(t-1)}{\phi(t-1)^T P(t-2)\phi(t-1) + 1}\,\varepsilon(t)$$

$$P(t-1) = P(t-2) - \frac{P(t-2)\phi(t-1)\phi(t-1)^T P(t-2)}{\phi(t-1)^T P(t-2)\phi(t-1) + 1} \qquad (2.7)$$

with some initial $\theta(0)$ given and $P(-1)$ any positive definite matrix P_0.

Goodwin and Sin (1984 : 59) show that this algorithm minimises the quadratic cost function

$$J_N(\theta) = \tfrac{1}{2}\sum_{t=1}^{n}(y(t) - \phi(t-1)^T\theta)^2 + \tfrac{1}{2}(\theta-\theta(0))^T P_0^{-1}(\theta-\theta(0)) \qquad (2.8)$$

or $J = \sum_{i=1}^{n} e_i(t)^2$ (Seborg et al 1986)

where $e_i(t) = y_i(t) - \phi_i(t-1)^T\theta_i(t-1)$

The form 2.7 contains the so-called **covariance matrix** P(t-1), which gives a measure of the estimation error. The diagonal elements of P(t-1) have a direct bearing on the convergence rate of the algorithm, and the off-diagonal elements should be small compared with the diagonal terms. As the estimates converge and the parameter error decreases, the elements of P become smaller.

The choices of $\theta(0)$ and P_0 are interdependant, since a large P_0 implies rapidly changing parameter estimates initially and hence a poor confidence in $\theta(0)$. On the other hand, if $\theta(0)$ is known to be a good estimate, smaller values for P_0 are usually chosen, to "desensitize" the updates earlier.

A significant problem with RLS algorithms is that in the long-term, the P-matrix will become very small after the parameters θ have converged. For processes with time-varying dynamics, the estimator is then insensitive to changing plant parameters and the estimates do not "track" the process as desired. This phenomenon is also known as "falling asleep" (Seborg et al 1986), and requires that the covariance matrix be modified to maintain algorithm sensitivity.

2.2.6 MODIFICATIONS TO RLS ESTIMATION

2.2.6.1 Exponential Data Weighting

This method weights new data more heavily than old data. An exponential forgetting factor or discounting factor is used, giving the cost function

$$J = \sum_{i=1}^{t} \lambda^{t-i} |\phi(i-1)^T \theta(i) - y(i)|^2 \qquad (2.9)$$

where $0 < \lambda \leq 1$. For $\lambda = 1$, we have the standard RLS form given in 2.7, as seen in the exponentially weighted update sequence

$$\theta(t) = \theta(t-1) + \frac{P(t-2)\phi(t-1)}{\phi(t-1)^T P(t-2)\phi(t-1) + \lambda(t-1)} \cdot e(t)$$

$$P(t-1) = \frac{1}{\lambda}\left[P(t-2) - \frac{P(t-2)\phi(t-1)\phi(t-1)^T P(t-2)}{\phi(t-1)^T P(t-2)\phi(t-1) + \lambda(t-1)}\right] \qquad (2.10)$$

The P-matrix is prevented from becoming too small for $\lambda < 1$, which maintains algorithm sensitivity. The smaller the value of λ, the faster the tracking speed of the algorithm.

Note in 2.10 that $\lambda(t-1)$ is time-dependant. Frequently a sequence is used such as

$$\lambda(t) = \lambda_0 \lambda(t-1) + (1 - \lambda_0) \qquad (2.11)$$

with $\lambda(0) = 0.95$ and $\lambda_0 = 0.99$ typically.

This has the effect of discounting old data during *initial estimation*, and then tending to normal recursive least squares ($\lambda = 1$). This is often applied in nonlinear estimation problems (Goodwin and Sin 1984 : 64).

Another technique for varying the forgetting factor is given by Fortescue et al (1981). They suggest that λ be kept near unity unless the prediction error becomes large, and then be decreased for several steps when larger error is detected. This would increase algorithm sensitivity to new data only when substantial model-process mismatch was present requiring parameter changes.

2.2.6.2 Covariance Resetting

This involves the *periodic resetting* of the covariance matrix P to some value P_i. Typically the resetting takes the form

$$P(t_i - 1) = k_i I \quad (2.12)$$

where I is the identity matrix and k_i some constant.

This is especially useful for fast tracking of rapidly time-varying processes.

2.2.6.3 Covariance Modification

An additional term is added to the P-matrix when modelling error is detected, thus :

$$P(t-1) = P(t-1) + Q(t-1) \quad (2.13)$$

with $0 \leq Q(t-1) < \infty$

Checking should be performed to ensure that P(t-1) stays bounded. This algorithm has a very similar effect to covariance resetting.

2.2.6.4 Regularised Constant Trace Algorithms

Another method of preventing the covariance matrix from becoming too small after parameter convergence is the constant trace algorithm. This technique is based on the knowledge that the diagonal elements of P determine the algorithm sensitivity, and hence the trace of the P-matrix is maintained at some predetermined value. This is facilitated by the following recursion :

$$P'(t-1) = k_0 P(t-1)/\text{trace}(P(t-1)) + k_1 I \qquad (2.14)$$

Where k_0 and k_1 are positive constants and I is the $n \times n$ identity matrix. (After Goodwin et al, 1986).

The choices of k_1 and k_0 can broadly be made as follows :

For an n-dimensional system model, we have after one step of the iteration that

$$P' = (k_0/(nP(0)))*P(0) + k_1$$

and $$\text{trace}(P') = n(k_0/n + k_1)$$
$$= k_0 + nk_1$$

then the constants should be chosen such that

$$k_0 + nk_1 = \text{the desired trace value.}$$

Choosing this value as nP(0), say, we have

$$k_0 + nk_1 = nP(0)$$

and suitable values for the constants are

$$k_0 = (n/2)P(0) \quad \text{and} \quad k_1 = \tfrac{1}{2}P(0) \qquad (2.15)$$

The value used for P(0) will determine the algorithm gain at each iteration, as with other methods. However, caution must be exercised in choosing this value, since a large P(0) will maintain large covariance matrix elements, and estimation may be very sensitive to process noise. An alternative approach has been suggested, that a large value of P(0) be used initially without any covariance modification, to allow rapid initial convergence. During this time trace(P) will become very small, and then the constant trace algorithm can be switched on with some value P'(0) which is smaller than P(0), to maintain some threshold of sensitivity in the algorithm.

2.2.7 OUTPUT ERROR METHODS

These algorithms are widely applied in stochastic process identification. They use a similar structure as that given in 2.3, namely

$$\theta(t) = \theta(t-1) + M(t-1)\phi(t-1)e'(t) \qquad (2.16)$$

The error is given by $e'(t) = y(t) - y'(t)$

$$= y(t) - \phi'(t-1)\theta(t)$$

where $y'(t)$ is the estimated process output, but $\phi'(t-1)$ contains previous **estimates** of $y'(t)$, rather than the actual plant measurement $y(t)$ (Goodwin and Sin, 1984 : 82). All of the preceeding estimation schemes can be developed in exactly the same way for the stochastic case, simply using $e'(t)$ in place of $e(t)$.

2.2.8 ESTIMATION IN THE PRESENCE OF BOUNDED NOISE

The "persistency of excitation" requirement for parameter convergence is documented by many authors, eg. Goodwin and Sin (1984 : 68), Anderson et al (1985), Anderson (1985), Boyd and Sastry (1986), Seborg et al (1986). This basically means that the input signal to a process must be "sufficiently rich" in spectral content to facilitate system identification. Anderson (1985) discusses in depth the phenomenon known as "bursting", in which the estimation algorithm gain grows rapidly in the absence of persistent excitation, causing large deviations in parameter estimates. To prevent this, many authors suggest the use of a **deadzone**, in which the algorithm update is switched off when the equation error signal e(t) drops below a certain threshold. Goodwin and Sin (1984 : 89) define a simple fixed deadzone as follows :

$$\theta(t) = \theta(t-1) + a(t-1).\frac{P(t-2)\phi(t-1)}{\phi(t-1)^T P(t-2)\phi(t-1) + 1}.\ e(t)$$

$$\text{with } a(t-1) = \begin{cases} 1 & \text{if } |e(t)| = |y(t) - \phi(t-1)^T\theta(t-1)| > 2\Delta \\ 0 & \text{otherwise} \end{cases} \qquad (2.17)$$

for some constant threshold Δ.

More sophisticated "relative deadzone functions" are proposed by Kreisselmeier (1986) and Goodwin et al (1986). These will be developed in more detail in a later section.

2.2.9 CONSTRAINED ESTIMATION

Frequently, the model parameter estimates can be constrained *a priori* to lie within a given region in parameter space. The estimation algorithms are then modified such that if the parameters move out of the predefined

region, they are forced back by some linear projection. This is accomplished generally by a linear coordinate transformation, effected by modifying P(t-1) to change the update direction. (See Goodwin and Sin 1984 : 92). De Larminat (1984) also suggests a type of parameter correction procedure, whereby the persistency of excitation requirement is removed, and stability of estimation is retained.

2.2.10 NUMERICAL SENSITIVITY

The implementation of RLS-type algorithms on finite-wordlength digital systems is susceptible to the problem of roundoff error. The P-matrix may become indefinite under such conditions, causing the algorithm to become unstable. One solution to the problem is called square-root filtering, where the covariance matrix is factored as

$$P(t) = S(t)S^{T}(t) \qquad (2.18)$$

(Seborg et al. 1986; Strejc, 1980) Then by adjusting S(t) at each iteration, the P-matrix will remain positive definite.

Another approach is to decompose P into upper triangular and diagonal matrices, thus :

$$P(t) = U(t)D(t)U^{T}(t) \qquad (2.19)$$

(Bierman, 1977). By updating factors of P, stability is retained. Such techniques are essential in certain circumstances, as detailed by Ljung and Soderstrom (1982), Bierman (1977). Astrom (1983) recommends the use of a factorization algorithm particularly if estimation is performed on data with a high d.c. offset level.

2.2.11 MODIFIED ESTIMATION SCHEMES

For stochastic estimation models, the *RLS algorithms detailed in* preceeding sections have been shown to produce good parameter estimates only if the noise sequence $\xi(t)$ and the equation error $e(t)$ are independant random variables. Howewver, if these two sequences are correlated, as in the case when $\xi(t)$ is a coloured noise sequence, biased estimates result. Modified schemes have been developed to overcome this problem, such as Extended Least Squares (ELS), Generalised Least Squares (GLS), and Instrumental Variable (IV) methods. These will not be described here, but a good survey of modified estimation techniques can be found in Strejc (1980).

2.3 CONTROL DESIGN STRATEGIES

A number of possible control design techniques are presented in the literature. Most of the unique controller structures cited are special cases of fairly general design procedures, which will be described in this section. Broadly, the design techniques may be divided into two categories, namely minimum prediction error designs, and closed-loop pole assignment. The former attempt to minimise the output error given by a predictive system description, often at the cost of large control effort. The latter approach stems from closed-loop stability considerations, and the overall transient response of the feedback system is significant in the design.

2.3.1 MINIMUM PREDICTION ERROR CONTROLLERS

The control strategies to be developed in these sections will be based on the well-known DARMA model (*Goodwin and Sin 1984 : 120*) :

$$A(q^{-1})y(t) = B(q^{-1})u(t) \qquad (2.20)$$

with $A(q^{-1}) = 1 + a_1q^{-1} + \ldots + a_nq^{-n}$

$$B(q^{-1}) = q^{-d}(b_0 + b_1q^{-1} + \ldots + b_mq^{-m})$$
$$= q^{-d}B'(q^{-1})$$

where d = time delay

This can be expressed as a d-step ahead predictor, thus :

$$y(t+d) = \alpha(q^{-1})y(t) + B(q^{-1})u(t) \qquad (2.21)$$

where $\alpha(q^{-1}) = \alpha_0 + \alpha_1q^{-1} + \ldots + \alpha_{n-1}q^{-(n-1)} = G(q^{-1})$

$$\beta(q^{-1}) = \beta_0 + \beta_1 q^{-1} + \ldots + \beta_{m+d-1} q^{-(m+d-1)}$$
$$= F(q^{-1})B'(q^{-1})$$

where $F(q^{-1})$ and $G(q^{-1})$ satisfy

$$1 = F(q^{-1})A(q^{-1}) + q^{-d}G(q^{-1}) \qquad (2.22)$$

(Goodwin and Sin 1984 : 107)

Note : The following control formulations use a **deterministic** process model. Minimum prediction error control designs based on **stocha** c predictive models are known as **minimum variance** control laws. ese are fairly widely documented in the literature on contol of stochastic processes. For the purposes of this research, all control formulations will be restricted to the deterministic case.

2.3.1.1 One-Step-Ahead Control

Using the predictor form given above, we develop a feedback control law which brings the system output y at time t+d, ie. y(t+d), to a reference value $y^*(t+d)$ in a si.,le step. The name "output deadbeat" control is also sometimes used to describe such controllers. The control law is characterised by

$$\beta(q^{-1})u(t) = y^*(t+d) - \alpha(q^{-1})y(t) \qquad (2.23)$$

This minimises a quadratic cost function comprising the squared prediction error

$$J(t+d) = \tfrac{1}{2}|y(t+d) - y^*(t+d)|^2$$

In closed-loop, the output response obeys

$$y(t) = y^*(t) \qquad t \geq d$$

and the control signal is characterised by

$$B(q^{-1})u(t) = A(q^{-1})y^*(t) \qquad t \geq d + n$$

Note that the control signal is generated by a transfer function whose poles are the zeros of $B(q^{-1})$. The *implicit constraint* for bounded inputs and outputs in closed-loop is that $B(q^{-1})$ is stable, ie. that the plant is minimum-phase.

A characteristic of one-step-ahead control is that excessive control effort may be required to force the process output to the reference signal at each sample. This can result in unpredictable or oscillatory inter-sample behaviour.

2.3.1.2 Weighted One-Step-Ahead Control

To reduce the excessive control effort required in one-step-ahead control, a **cost** or weight is placed on the control signal u(t), giving the modified cost function

$$J_2(t+d) = \tfrac{1}{2}[y(t+d) - y^*(t+d)]^2 + \tfrac{1}{2}\lambda u(t)^2$$

The control law becomes

$$u(t) = \frac{\beta_0 y^*(t+d) - \alpha(q^{-1})y(t) - \beta'(q^{-1})u(t-1)}{(\beta_0^2 + \lambda)} \qquad (2.24)$$

where

$$\beta'(q^{-1}) = q[B(q^{-1}) - \beta_0]$$

$$= \beta_1 + \beta_2 q^{-1} + \ldots + \beta_{m+d-1} q^{-(m+d-2)}$$

(Goodwin and Sin 1984 : 122).

In closed-loop, the system response is described by

$$\left[B'(q^{-1}) + (\lambda/B_0)A(q^{-1})\right]y(t+d) = B'(q^{-1})y^*(t+d)$$

and the control signal by

$$\left[B'(q^{-1}) + (\lambda/B_0)A(q^{-1})\right]u(t) = A(q^{-1})y^*(t+d)$$

Thus it is clear that the control effort has been "softened" by the inclusion of the weighting factor λ, at the cost of a relaxed output response to setpoint variations.

This control law can be made effective for all open-loop stable, minimum-phase processes, as well as certain nonminimum-phase and open-loop unstable plant, with suitable choices of λ (Goodwin and Sin 1984 : 123). The use of λ in a one-step-ahead control law is also known as "detuning". Other more sophisticated techniques for detuning (such as polynomial weighting factors) are given in Goodwin and Sin (1984 : 124).

2.3.1.3 Model-Reference Control

The system description of 2.20 is used, and a control law is designed to enable the process output y(t) to track a setpoint $y^*(t)$. However, the desired output response is generated by a prespecified reference model with a known transfer function, which is driven by a reference signal r(t). Thus $y^*(t)$ obeys

$$E(q^{-1})y^*(t) = q^{-q'}H(q^{-1})r(t) \qquad (2.25)$$

Then r(t) is the input to the reference model, which has transfer function

$$G(z^{-1}) = z^{-d'}H(z^{-1})/E(z^{-1})$$

The polynomials $H(z^{-1})$ and $E(z^{-1})$ are selected to give desired performance and stability. A clear constraint is that $E(z^{-1})$ is stable.

The model-reference system is detailed in Figure 2 on page 28. It can be seen that the tracking error is defined as

$$e(t) = y^{*}(t) - y(t)$$

As before, we derive the model reference controller first by developing a prediction of the output, and then choosing a control law which sets it equal to some function of the reference signal r(t). In this case, however, we predict $E(q^{-1})y(t)$ and choose control to set it equal to $q^{-d}H(q^{-1})r(t)$, to satisfy the objective

$$E(q^{-1})y(t) = q^{-d}H(q^{-1})r(t) \qquad (2.26)$$

(Note : this is the same as 2.25, with $y(t) = y^{*}(t)$).

The predictor form is given as (Goodwin and Sin 1984 : 131) :

$$E(q^{-1})y(t+d) = \alpha(q^{-1})y(t) + \beta(q^{-1})u(t) \qquad (2.27)$$

where

$$\alpha(q^{-1}) = G(q^{-1})$$
$$\beta(q^{-1}) = F(q^{-1})B'(q^{-1})$$

and $F(q^{-1})$, $G(q^{-1})$ are polynomials of order d-1, n-1 respectively, which satisfy the Generalised Prediction Equality

$$E(q^{-1}) = F(q^{-1})A(q^{-1}) + q^{-d}G(q^{-1}) \qquad (2.28)$$

(Macleod, 1987)

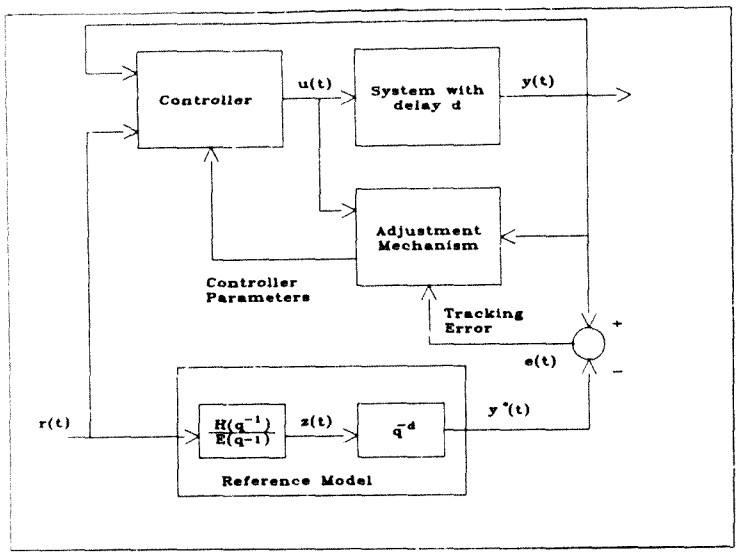

Figure 2. Block Diagram of Model Reference Control System.

This can be proved by multiplying 2.20 by $F(q^{-1})$ and using 2.28.

The control signal is generated using

$$\beta(q^{-1})u(t) = H(q^{-1})r(t) - \alpha(q^{-1})y(t) \qquad (2.29)$$

which gives the desired objective of 2.26.

Thus it is clear that model-reference control is a generalisation of one-step-ahead control, with a different approach to "softening" the transient response and hence the control effort. This technique is also only applicable to minimum-phase processes. A comprehensive review of adaptive model-reference control is given by Landau (1979).

2.3.2 POLE ASSIGNMENT CONTROLLERS

These controllers are designed by examining the closed-loop system, and placing the closed-loop poles at desired locations. From Figure 3 *on page* 30, it is clear that the closed-loop system "transfer function" can be written as

$$G(q^{-1}) = \frac{P(q^{-1})B(q^{-1})}{P(q^{-1})B(q^{-1}) + A(q^{-1})L(q^{-1})}$$

or $\left[P(q^{-1})B(q^{-1}) + A(q^{-1})L(q^{-1})\right]y(t) = P(q^{-1})B(q^{-1})y^{*}(t)$

(Macleod, 1987)

By setting the denominator of $G(q^{-1})$ equal to a desired closed-loop polynomial $A^{*}(q^{-1})$, the poles of the closed-loop system can be arbitrarily positioned, thus :

Solve $A(q^{-1})L(q^{-1}) + B(q^{-1})P(q^{-1}) = A^{*}(q^{-1})$ (2.30)

to give the controller polynomials $P(q^{-1})$ and $L(q^{-1})$.

the expression 2.30 is known as the Diophantine equation. and its solution involves the solving of a set of linear equations.

A condition on the solution is that $A(q^{-1})$ and $B(q^{-1})$ are relatively prime. However, since no attempt is made to cancel $B(q^{-1})$ in the closed loop, nonminimum-phase processes can easily be controlled by such designs.

The control law is then simply given by

$$L(q^{-1})u(t) = P(q^{-1})[y^{*}(t) - y(t)]$$

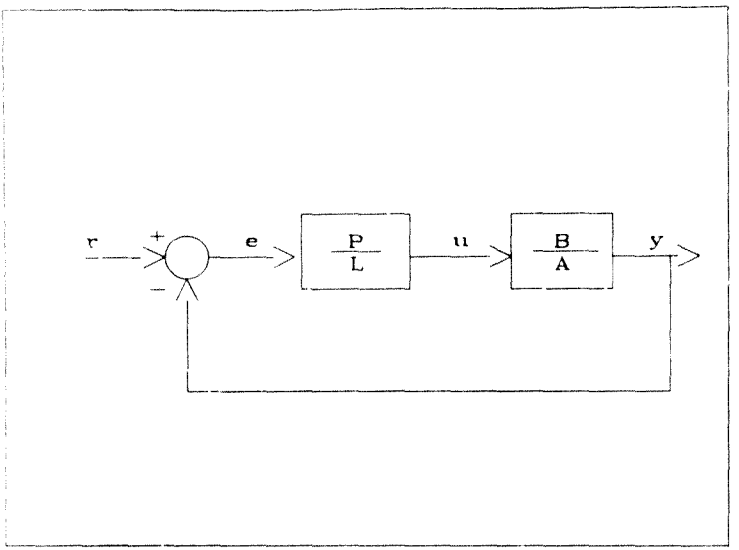

Figure 3. Closed-Loop System Structure

The selection of a suitable $A^*(q^{-1})$ is discussed in later sections.

2.3.2.1 Rapprochement with *Minimum Prediction Error Control*

It can be shown that the control design approaches of earlier sections are merely special cases of the pole assignment procedure.

Considering the Diophantine equation 2.30, and setting

$$A^*(q^{-1}) = B'(q^{-1}), \qquad 2.30 \text{ gives}$$

$$A(q^{-1})L(q^{-1}) + B(q^{-1})P(q^{-1}) = B'(q^{-1}) \qquad (2.31)$$

Clearly $B'(q^{-1})$ must be a factor of $L(q^{-1})$, so we set

$$L(q^{-1}) = F(q^{-1})B'(q^{-1})$$

which gives from 2.31

$$F(q^{-1})A(q^{-1}) + q^{-d}P(q^{-1}) = I$$

This is simply the d-step ahead prediction equality 2.22 used in one-step-ahead control, with $P(q^{-1}) = G(q^{-1})$. Thus the one-step-ahead approach can be interpreted as pole assignment, where the closed-loop poles are assigned the values of the open-loop zeros (Goodwin and Sin 1984 : 154).

Similarly, model-reference control can be achieved by choosing

$$A^{*}(q^{-1}) = B'(q^{-1})E(q^{-1}).$$

Substitution in 2.30 results in

$$F(q^{-1})A(q^{-1}) + q^{-d}P(q^{-1}) = E(q^{-1})$$

Which is the same as the Generalised Prediction Equality (2.28) as used in model-reference controller design. (Goodwin and Sin 1984 : 155).

2.4 CONCLUSION

This section has reviewed some of the current techniques in adaptive estimation and control for deterministic sampled systems. It must be noted that many variants are documented in the literature, but these are almost all based on the methods discussed here. The most popular and reliable estimation technique is the Recursive Least Squares (RLS) with appropri-

ate modifications to improve robustness. The controller design procedures discussed have been shown to be special forms of a more general case, namely closed-loop pole assignment.

3.0 DELTA-OPERATOR FORMULATION OF DISCRETE-TIME CONTROL

Astrom, Hagander & Sternby (1984) have documented the relationship between continuous time transfer functions and their discrete-time counterparts. In particular, they showed that for continuous-time transfer functions with stable zeros of relative degree greater than two, the corresponding discrete-time transfer function will contain a number of additional zeros equal to the pole excess, which migrate from z=0 to outside the unit circle, as the sampling rate increases. Thus a continuous-time plant which has no unstable zeros can become nonminimum-phase during sampling. In addition, for plants with delays which are fractional multiples of the sampling period Δ, at least one of the zeros of the sampled system will become nonminimum-phase (Clarke, 1984).

Thus it would not be sensible to use discrete control strategies which attempt to cancel all the open-loop plant zeros (see "Current Approaches to Adaptive Estimation and Control" on page 9), since this would imply an unstable set of controller poles. This could cause closed-loop instability since pole-zero cancellation will not be exact in discrete time.

Some authors have suggested very subtle approaches to the problem. Gawthrop (1980) has proposed a "hybrid controller" which ensures that continuous-time plants with stable zeros do not become nonminimum-phase during sampling. This is effected by inserting a "bandlimited differentiator" after the plant output measurement in the closed loop, which replaces the unstable discrete zeros predicted by Astrom et al (1984) by zeros near the origin. The plant model is formulated in continuous-time at a fast sampling rate, with a discrete-time control law at a slower rate. The overall adaptive controller then contains both discrete and continuous components, whose closed-loop stability properties tend to those of a continuous-time system as the sampling rate increases.

3.0 DELTA-OPERATOR FORMULATION OF DISCRETE-TIME CONTROL

Åstrom, Hagander & Sternby (1984) have documented the relationship between continuous time transfer functions and their discrete-time counterparts. In particular, they showed that for continuous-time transfer functions with stable zeros of relative degree greater than two, the corresponding discrete-time transfer function will contain a number of additional zeros equal to the pole excess, which migrate from $z=0$ to outside the unit circle, as the sampling rate increases. Thus a continuous-time plant which has no unstable zeros can become nonminimum-phase during sampling. In addition, for plants with delays which are fractional multiples of the sampling period h, at least one of the zeros of the sampled system will become nonminimum-phase (Clarke, 1984).

Thus it would not be sensible to use discrete control strategies which attempt to cancel all the open-loop plant zeros (see "Current Approaches to Adaptive Estimation and Control" on page 9), since this would imply an unstable set of controller poles. This could cause closed-loop instability since pole-zero cancellation will not be exact in discrete time.

Some authors have suggested very subtle approaches to the problem. Gawthrop (1980) has proposed a "hybrid controller" which ensures that continuous-time plants with stable zeros do not become nonminimum-phase during sampling. This is effected by inserting a "bandlimited *differentiator*" *after the plant output measurement* in the closed loop, which replaces the unstable discrete zeros predicted by Åstrom et al (1984) by zeros near the origin. The plant model is formulated in continuous-time at a fast sampling rate, with a discrete-time control law at a slower rate. The overall adaptive controller then contains both discrete and continuous components, whose closed-loop stability properties tend to those of a continuous-time system as the sampling rate increases.

However, more typically digital controllers are implemented with constraints on the sampling rate. Astrom, Hagander and Sternby (1984) showed that there is a critical sampling period h above which all zeros of the sampled system are inside the unit circle in the z-domain. Thus many authors advocate the use of lower sampling rates for discrete-time control systems. This has the disadvantage of yielding a low-bandwidth control law, and consequently a low-bandwidth closed-loop system.

By the very definition of the sampling operation, one would intuitively expect better approximation to continuous-time systems as the sampling became more rapid, since the discrete signals would be more accurate representations of the continuous ones. This is in direct contrast with the preceeding observations.

This apparent paradox has been resolved by Goodwin et al (1986), by introducing a different shift operator to replace the usual inverse shift operator. This has become known as the Delta-operator, and is defined as

$$\delta = \frac{q - 1}{\Delta} \qquad (3.1)$$

where q is the usual shift operator and Δ is the sampling interval. This has the advantage of giving a close connection between exact continuous and exact discrete transfer function plant models at higher sampling rates.

The δ-operator is not in itself a new mathematical concept. It has been used in the control literature as a way of motivating z-transforms, and is suggested in digital filtering applications to improve numerical behaviour (see Goodwin et al, 1986).

We examine the physical realisation of the δ-operator as follows:

$$\delta = \frac{\Delta}{q - 1} \qquad (3.2)$$

and $\delta^{-1}y(k) = y'(k)$, say.

Then $$y'(k) = \frac{\Delta}{q - 1}\, y(k)$$

$$= \frac{\Delta q^{-1}}{1 - q^{-1}}\, y(k)$$

So $$(1 - q^{-1})y'(k) = \Delta q^{-1}y(k)$$

or $$y'(k) = \Delta y(k-1) + y'(k-1) \qquad (3.3)$$

Equation 3.3 shows that we may interpret the δ^{-1}-operator as a discrete integrator, where y'(k) is the area under the function y(k) up to sample time k. Figure 4 on page 36 illustrates this concept.

The delta-operator may be implemented in practice as shown in Figure 5 on page 37.

This implementation is coded in pseudo-code as

```
YOUT:= DELTA*YOLD + YOUT;
YOLD:= Y;
```

We define the transform corresponding to the δ -operator as the Υ-transform. Further interesting properties emerge by examining this Υ-transform domain. It is easily shown (Middleton & Goodwin, 1986) that the stability region in the Υ-plane is a circle of radius $1/\Delta$, positioned at $(-1/\Delta,0)$. This region, as the sampling interval Δ decreases, approximates the Laplace-domain stability region, viz. the whole left-hand half-plane. This is illustrated in Figure 6 on page 38.

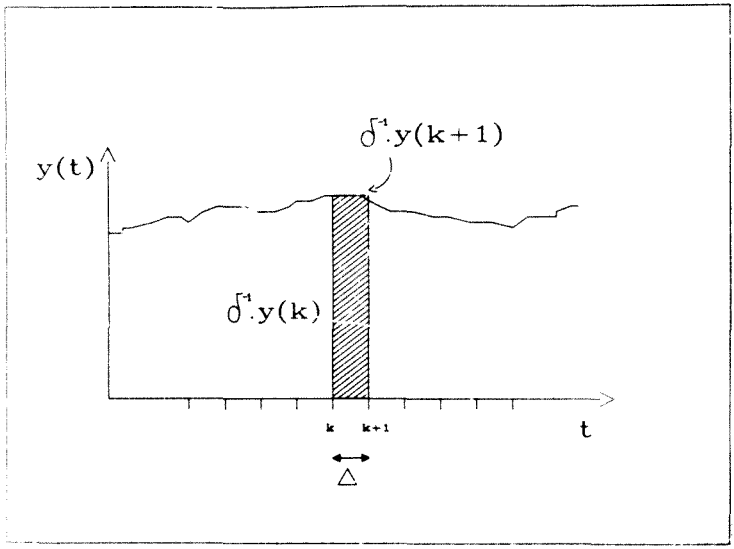

Figure 4. The Inverse Delta Operator as an Integrator

Middleton and Goodwir (1986) define the γ-transform as follows:

$$F_\gamma(x) = \Delta F_Z(1+\Delta x) = \Delta \sum_{k=0}^{\infty} f(k\Delta)(1+\Delta x)^{-k} \tag{3.4}$$

where F_Z is the normal Z-transform. Since z has been replaced by $(1+\Delta x)$, it is clear from 3.1 that 3.4 gives the natural transform for delta models where $q = (1 + \Delta\delta)$.

It is further shown that for the Laplace transform $F_s(s)$

$$\lim_{\Delta \to 0} \{F_\gamma(x)\} = F_s(s)|_{s=x}$$

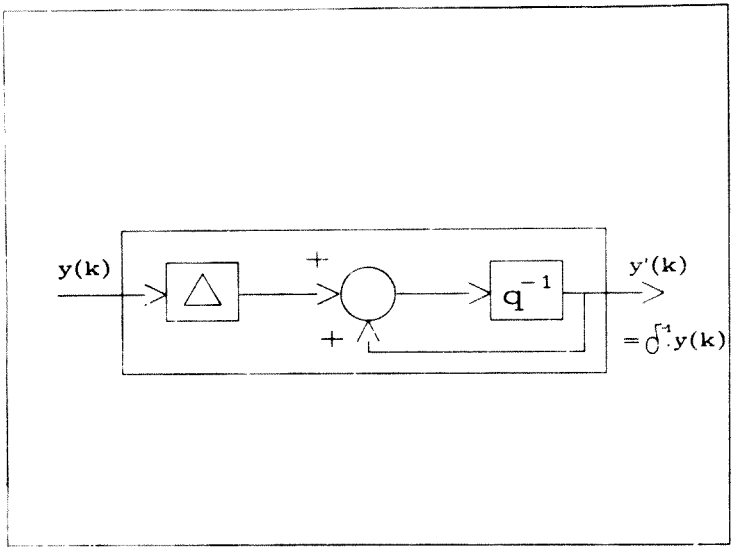

Figure 5. Block Diagram implementation of the Delta-Operator

$$= \int_0^{\infty} f(t)e^{-xt}\,dt \qquad (3.5)$$

where the integral in 3.5 is a Riemann integral. Thus a close correlation between the continuous time Laplace transform and the discrete-time Y-transform is established for fast sampling.

Middleton and Goodwin (1986) further demonstrate that the Y-transform has many similar properties to the Laplace transform, including linearity, final/initial value theorems, complex translation and real convolution.

Standard Laplace-domain transfer functions are expressed in the new domain as follows:

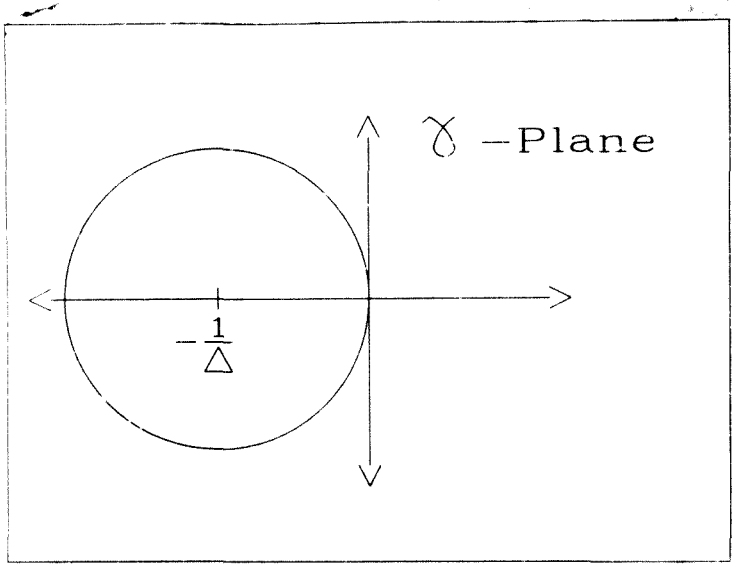

Figure 6. The Stability region in the Gamma-Plane

If the continous plant is

$$A'(D)y(t) = B'(D)u(t) \qquad \text{where } D \equiv d/dt$$

with Laplace-transformed model

$$A'(s)Y(s) = B'(s)U(s)$$

then the discrete-time δ-operator model is

$$A(\delta)y(k\Delta) = B(\delta)u(k\Delta)$$

and its Z-transformed model is

$$A_\gamma(x)Y_\gamma(x) = B_\gamma(x)U_\gamma(x)$$

with the transformation defined as

$$A(\delta) = \frac{1}{\Delta^n} A'(\delta\Delta+1)$$

$$B(\delta) = \frac{1}{\Delta^n} B'(\delta\Delta+1) \qquad (3.6)$$

It is important to note that the δ-operator system model will have the same order and relative degree as the inverse shift-operator model. This will not be true of many other possible substitutions.

Also, Middleton and Goodwin (1986) demonstrate that the δ-model has superior finite word-length characteristics as related to coefficient representation, round-off noise and online control synthesis. This last is a consequence of reduced eigenvalue sensitivity for finite word-length systems using δ-operators. In an adaptive control application using a numerically sensitive control design procedure (such as the solution of a Diophantine equation during pole assignment), this is a significant property

Furthermore, the plant model derived by an online parameter estimator will be closely related to the continuous-time plant. Feedback controller design can thus be performed using classical control engineering techniques, and the closed-loop numerical values obtained in practice will be far easier to interpret.

Goodwin et al (1986) developed and implemented a model-reference adaptive controller using the δ-operator representation. Excellent control performance was obtained for a wide range of disturbances, including large changes in open-loop system gain. This demonstrated the retention of continuous-time properties in the sampled system, ensuring stable con-

trol. A global convergence proof for the model reference adaptive case using δ-operators is also presented by the authors.

The discussion of this section has demonstrated that a new transform domain bridges the gap between discrete- and continuous-time theory. This means that all system transfer functions, especially with regard to pole and zero positions, can be examined in the Laplace-domain or frequency-domain. As long as the condition of fast sampling holds, the approximation $s \approx \delta$ can be made. This simplifies the analysis and design of the adaptive control system considerably.

The constraint of slow sampling normally advocated is now removed, allowing for the design of larger bandwidth closed-loop systems. This is desirable for better disturbance rejection and tracking performance. Faster sampling also avoids two practical control problems, viz. for slow sampling rates, control is open-loop between samples. This can yield intersample ripples, which implies that the sampled output is a poor representation of the actual plant response. Secondly, to yield a desired closed-loop bandwidth, the control signal for slow sampling rates will often be a series of large step changes. This generates significant high frequency energy in the actuators, which can be undesirable. Faster sampling will give a smoother sequence of small step changes for the same bandwidth.

All of the ensuing discussions of the estimator and controller are formulated in terms of δ-operators.

4.0 ROBUST ESTIMATOR STRUCTURE

This section develops the structure for a robust parameter estimator. The Recursive Least Squares method is used with some modifications, and the process model formulation is given in terms of δ-operators.

4.1 MODEL STRUCTURE

A linear difference equation system description is used, as described in "Linear Difference Equation Models" on page 11. This is conveniently expressed as a CARMA model (See "Minimum Prediction Error Controllers" on page 23), t

$$A(\delta)y(k\Delta) = B(\delta)u(k\Delta) + \xi(k\Delta) \qquad (4.1)$$

For compactness of notation we drop the arguments, giving

$$Ay = Bu + \xi$$

The structure of equation 4.1 is generalised further to include both unmeasurable deterministic disturbances and measurable random disturbance signals, thus:

$$Ay = Bu + Fz + d + \xi \qquad (4.2)$$

where the symbols have the following meanings:

A,B and F are monic polynomial operators, defined as

$$A = a_0 + a_1\delta + a_2\delta^2 + \ldots\ldots + \delta^n$$
$$B = b_0 + b_1\delta + b_2\delta^2 + \ldots\ldots + \delta^m$$

$$F = f_0 + f_1\delta + f_2\delta^2 + \ldots\ldots + \delta^r$$

z : measurable random disturbance signal
d : unmeasurable deterministic disturbance signal
ξ : modelling error and noise

The measured disturbance signal z(kΔ) will be used to develop an adaptive feedforward control block. This could be useful in many industrial process applications, since the disturbance signal measurements are often readily available online for monitoring purposes. One of the well-known problems with the use of feedforward control is the necessity for a good dynamic model of the disturbance-output relationship, for effective disturbance compensation. Adaptive estimation would appear to be ideally suited to such a situation, allowing for the generation of a good model in the presence of unknown or possibly time-varying disturbance behaviour.

The unmeasurable deterministic disturbance signal d(kΔ) is included to model known waveforms such as d.c. offsets, ramps, sinusoidal and other periodically varying disturbances which are introduced in the process inputs and outputs from unmeasurable sources. This situation may typically emerge for example when contiguous processes are coupled through material flow, and a periodic variation upstream causes observable deterministic variations in the downstream process.

The deterministic disturbance d is characterised by

$$Dd = 0 \qquad (4.3)$$

(Goodwin and Sin 1984 : 156)

where D is also a monic polynomial of the form

$$D = d_0 + d_1\delta + d_2\delta^2 + \ldots\ldots + \delta^k$$

D is known as a "deterministic signal nulling polynomial", since it eliminates the signal d when operating on it. Some examples of

deterministic signals and their nulling polynomials are (after Macleod, 1987) :

1. $d(k\Delta)$ = constant

   ie. $d(k\Delta) = d[(k+1)\Delta]$ for all k

   taking $d[(k+1)\Delta] - d(k\Delta) = 0$

   $$(q - 1)d(k\Delta) = 0$$

   Then $D(q) = q - 1$

   and in δ-operator form

   $$D(\delta) = \delta \quad \text{using 3 1}$$

(See "Delta-operator formulation of Discrete-Time Control" on page 33)

2. $d(k\Delta) = A.\sin(\rho k\Delta + \phi)$ (sinusoid)

   Then $D(q) = (q^2 - 2\cos\rho q + 1)$

   and $D(\delta) = \delta^2 + \frac{2(1-\cos\rho)}{\Delta}\delta + \frac{2(1-\cos\rho)}{\Delta^2}$

3. $d(k\Delta) = A_1\sin(\rho_1 k\Delta + \phi_1) + A_2\sin(\rho k\Delta + \phi_2)$
   (Sum of sinusoids)

   Then $D(q) = (q^2 - 2\cos\rho_1 q + 1)(q^2 - 2\cos\rho_2 q + 1)$

   and $D(\delta) = \left[\delta^2 + \frac{2(1-\cos\rho_1)}{\Delta}\delta + \frac{2(1-\cos\rho_1)}{\Delta^2}\right]\left[\delta^2 + \frac{2(1-\cos\rho_2)}{\Delta}\delta + \frac{2(1-\cos\rho_2)}{\Delta^2}\right]$

Note: Since D represents some deterministic function, it will have periodic components. Thus the roots of D lie on the stability boundary of the δ-plane.

To eliminate the deterministic disturbance d, we multiply both sides of equation 4.2 by D. This is in accordance with the Internal Model Principle (Goodwin and Sin 1984 : 156). Thus :

$$ADy = BDu + FDz + D\xi \qquad (4.4)$$

Now since D has roots on the stability boundary, the noise term ξ may be amplified at high frequencies. Thus we introduce a stable polynomial operator D' which is "near to" D.

For example, if the disturbance d is a constant (d.c. offset disturbance signal) then as before

$$D = \delta,$$

and we choose

$$D' = \delta + \varepsilon \qquad (\varepsilon \text{ is some small positive value}).$$

We divide 4.4 by D' to give

$$\frac{AD}{D'} y = \frac{BD}{D'} u + \frac{FD}{D'} z + \frac{D}{D'} \xi \qquad (4.5)$$

The "transfer function" specified by

$$\frac{D(\delta)}{D'(\delta)} = \frac{\delta}{\delta + \varepsilon}$$

is effectively a high-pass filter as we would expect to null a constant disturbance sig. The frequency response is determined by "how close" the corner frequency ε is to zero, as shown in Figure 7 on page 45, with

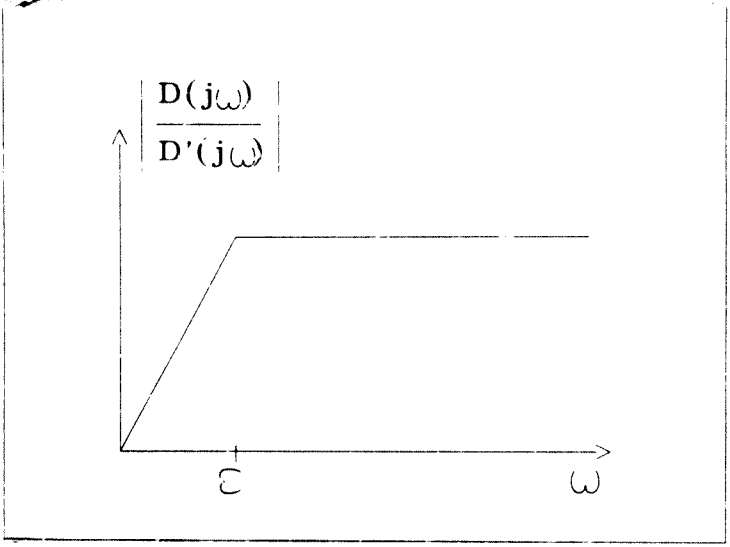

Figure 7. Frequency Response of High pass Filter for Robust Estimation

the approximation $\delta = j\omega$. For other deterministic disturbances, D/D' would yield a "band reject" filter with the stopband simply a notch or notches at the periodic frequency components of d(t).

We also bandlimit the estimated model by introducing a low-pass filter described by a monic stable polynomial operator E:

$$E = e_0 + e_1\delta + e_2\delta^2 + \ldots\ldots + \delta^n$$

This filtering will effectively be applied to the u,y and z signals measured, and will ensure that the signals used for estimation do not contain high frequency components. Thus the low-order model being estimated is not influenced by the unaccounted-for high-frequency system dynamics. Equation 4.5 becomes

$$\frac{AD}{D'E}\,y = \frac{BD}{D'E}\,u + \frac{FD}{D'E}\,z + \frac{D}{D'E}\,\xi \qquad (4.6)$$

We rearrange the model, defining

$$y' = \frac{D}{D'}\,y$$

Note: y' is the output of the high pass filter operating on the measured input y, where D and D' are defined as above for d = constant.

Also, define

$$y_f = \frac{1}{E}\,y'$$

Where y_f is the filtered output signal.

We can then rewrite equation 4.6 as

$$Ay_f = Bu_f + Fz_f + \eta_f \qquad (4.7)$$

With the following quantities defined:

$$y_f = \frac{D}{D'E}\,y$$

$$u_f = \frac{D}{D'E}\,u$$

$$z_f = \frac{D}{D'E}\,z$$

$$\eta_f = \frac{D}{D'E}\,\xi$$

This "filtered model" is in agreement with the signal conditioning or prefiltering described in Goodwin and Mayne (1987). Thus a bandlinited modal appears to be essential in adaptive estimation for unknown, possibly high-order or nonlinear processes.

Now, adding Ey_f to both sides of 4.7 yields

$$Ey_f = Ey_f - Ay_f + Bu_f + Fz_f + \eta_f \qquad (4.8)$$

Since $y' = Ey_f$ by definition, we have the regression form

$$y' = (E - A)y_f + Bu_f + Fz_f + \eta_f \qquad (4.9)$$

If we neglect the bounded noise term η_f, we can formulate a standard regression for $y'(k)$:

$$y'(k) = \phi(k-1)^T \cdot \theta(k)$$

where

$$\phi(k-1)^T = [y_f(k), \ldots \delta^{n-1}y_f(k), u_f(k), \ldots \delta^m u_f(k), z_f(k), \ldots \delta^r z_f(k)] \qquad (4.10)$$

$$\theta(k) = [e_0 - a_0, e_1 - a_1 \ldots e_{n-1} - a_{n-1}, b_0 \ldots b_m, f_0 \ldots f_r] \qquad (4.11)$$

This is compatible with the recursive least squares algorithm described in "Current Approaches to Adaptive Estimation and Control" on page 9.

This "filtered model" is in agreement with the signal conditioning or prefiltering described in Goodwin and Mayne (1987). Thus a bandlinited model appears to be essential in adaptive estimation for unknown, possibly high-order or nonlinear processes.

Now, adding Ey_f to both sides of 4.7 yields

$$Ey_f = Ey_f - Ay_f + Bu_f + Fz_f + \eta_f \tag{4.8}$$

Since $y' = Ey_f$ by definition, we have the regression form

$$y' = (E - A)y_f + Bu_f + Fz_f + \eta_f \tag{4.9}$$

If we neglect the bounded noise term η_f, we can formulate a standard regression for $y'(k)$:

$$y'(k) = \phi(k-1)^T \cdot \theta(k)$$

where

$$\phi(k-1)^T = [y_f(k), \ldots \delta^{n-1}y_f(k), u_f(k), \ldots \delta^m u_f(k), z_f(k), \ldots \delta^r z_f(k)] \tag{4.10}$$

$$\theta(k) = [e_0 - a_0, e_1 - a_1 \cdots e_{n-1} - a_{n-1}, b_0 \cdots b_m, f_0 \cdots f_r] \tag{4.11}$$

This is compatible with the recursive least squares algorithm described in "Current Approaches to Adaptive Estimation and Control" on page 9.

4.2 RELATIVE DEADZONE

As described in "Estimation in the Presence of Bounded Noise" on page 20, robustness in recursive estimation is dependant on "persistency of excitation", which means that the input signal u must be "sufficiently rich" in frequency content. Anderson (1985) has investigated the mechanism behind the so-called "bursting phenomenon", in which the parameter estimates (and consequently the estimated output y), exhibit "bursts" or intervals of unbounded growth. This is due to the estimator attempting to update the model estimates without any useful signal exciting the system. Consequently, due to the low order of the model used, high frequency noise causes erroneous parameter updates. Various other mechanisms for parameter divergence are explored by Anderson (1985). The robust estimator developed in this section uses the ***relative deadzone function*** described by Goodwin et al (1986), Kreisselmeier (1986), and Ortega and Lozano-Leal (1987). This function turns off the estimator when the equation error e(t) is smaller than some threshold. The threshold is dynamically varied according to the magnitudes of the input and output signals. The relative deadzone arises from the reasoning that modelling error is effectively a disturbance in the estimation system, which is directly correlated with the magnitudes of the plant input and output signals. Thus the deadzone is increased in the presence of large process signal measurements, since larger error terms are anticipated, and reduced when the plant inputs and outputs are small.

The relative deadzone used is illustrated in Figure 8 on page 49, and is defined as

$$f(g,e) = \begin{cases} e - g & \text{if } e > g \\ 0 & \text{if } |e| \leq g \\ e + g & \text{if } e \leq -g \end{cases} \qquad (4.12)$$

The relative deadzone is now developed to vary the size of g.

We define the function m(k) as:

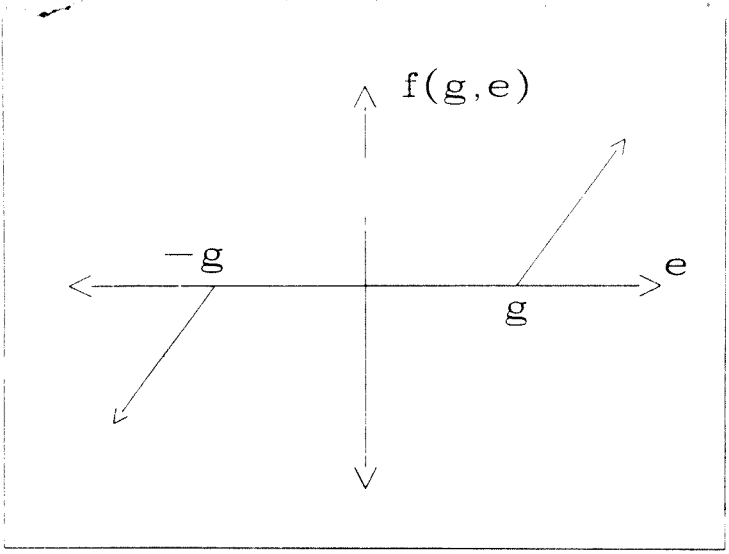

Figure 8. The Basic Relative Deadzone Function

$$m(k) = \sigma_0 m(k-1) + \varepsilon_0 + \varepsilon_1|u'(k-1)| + \varepsilon_2|y'(k-1)| + \varepsilon_3|z'(k-1)| \qquad (4.13)$$

Where the constants are chosen as

$\sigma_0' \in (0,1)$;

$\varepsilon_0 \geq 0$;

$\varepsilon_1 \geq 0$;

$\varepsilon_2 \geq 0$;

$\varepsilon_3 \geq 0$,

$m_0 \geq 0$;

and we choose

$\sigma_0 \in (\sigma_0', 1)$, and $m(0) = m_0$

so that

$$|\eta_f(k)| \leq m(k) \quad \text{for all } k.$$

m(k) is effectively a low-pass filtered function of the process input and output, which must **overbound** the filtered noise term $\eta_f(k)$. This term represents the filtered process noise and noise due to modelling error.

Note: For a "unity-gain" deadzone filter the function 4.13 should be modified as

$$m(k) = \sigma_0 m(k-1) + (1-\sigma_0)|\varepsilon_0 + \varepsilon_1|u'(k-1)| + \varepsilon_2|y'(k-1)| + \varepsilon_3|z'(k-1)||$$

Then choose $\varepsilon_0 > 0$ and $\alpha \in (0,1)$ to implement the deadzone as follows:

$$\beta = \sqrt{\varepsilon_0 + 1/(1-\alpha)} \tag{4.14}$$

Finally, define the magnitude function a(k) as:

$$a(k) = \begin{cases} 0 & \text{if } |e(k)| < \beta m(k) \\ \alpha \cdot f(e(k); \beta m(k))/e(k) & \text{otherwise} \end{cases} \tag{4.15}$$

The relationships 4.14 and 4.15 emerge from convergence analysis of the estimator with the deadzone, as performed by Goodwin et al (1986), Kreisselmeier (1986), Ortega and Lozano-Leal (1987). For completeness a

simplified convergence analysis is included in Appendix A. The choice of suitable constants are discussed in a later section.

4.3 THE RECURSIVE LEAST SQUARES ALGORITHM

The Actual parameter update is performed at each iteration using a modified RLS algorithm. In δ-operator notation, this is given by the following two stages :

$$\delta\theta'(k-1) = \frac{a(k)}{\Delta} \cdot \frac{P(k-2)\phi(k-1)}{\phi(k-1)^T P(k-2)\phi(k-1) + 1} \cdot e(k)$$

$$\delta P(k-2) = \frac{-a(k)}{\Delta} \cdot \frac{P(k-2)\phi(k-1)\phi(k-1)^T P(k-2)}{\phi(k-1)^T P(k-2)\phi(k-1) + 1} \qquad (4.16)$$

The equations 3.16 can be rewritten in recursion form as follows.

$$\theta(k) = \theta(k-1) + a(k) \cdot \frac{P(k-2)\phi(k-1)}{\phi(k-1)^T P(k-2)\phi(k-1) + 1} \cdot e(k)$$

$$P(k-1) = P(k-2) - a(k) \cdot \frac{P(k-2)\phi(k-1)\phi(k-1)^T P(k-2)}{\phi(k-1)^T P(k-2)\phi(k-1) + 1} \qquad (4.17)$$

The most important modification is the inclusion of the factor a(k), which turns off the update when the prediction error e(k) falls below the bound βm(k), as described above.

simplified convergence analysis is included in Appendix A. The choice of suitable constants are discussed in a later section.

4.3 THE RECURSIVE LEAST SQUARES ALGORITHM

The Actual parameter update is performed at each iteration using a modified RLS algorithm. In δ-operator notation, this is given by the following two stages :

$$\delta\theta'(k-1) = \frac{a(k)}{\Delta} \cdot \frac{P(k-2)\phi(k-1)}{\phi(k-1)^T P(k-2)\phi(k-1) + 1} \cdot e(k)$$

$$\delta P(k-2) = \frac{-a(k)}{\Delta} \cdot \frac{P(k-2)\phi(k-1)\phi(k-1)^T P(k-2)}{\phi(k-1)^T P(k-2)\phi(k-1) + 1} \qquad (4.16)$$

The equations 3.16 can be rewritten in recursion form as follows:

$$\theta(k) = \theta(k-1) + a(k) \cdot \frac{P(k-2)\phi(k-1)}{\phi(k-1)^T P(k-2)\phi(k-1) + 1} \cdot e(k)$$

$$P(k-1) = P(k-2) - a(k) \cdot \frac{P(k-2)\phi(k-1)\phi(k-1)^T P(k-2)}{\phi(k-1)^T P(k-2)\phi(k-1) + 1} \qquad (4.17)$$

The most important modification is the inclusion of the factor a(k), which turns off the update when the prediction error e(k) falls below the bound βm(k), as described above.

4.4 NUMERICAL IMPLEMENTATION

As discussed in "Numerical Sensitivity" on page 21, for finite-wordlength implementations of the RLS algorithm a covariance factorization technique is more robust than the standard form. In this research, the UDU^T factorization proposed by Bierman (1977) is used to improve numerical behaviour of the algorithm. The algorithm is described in Appendix B, modified to include the relative deadzone and a forgetting factor.

4.5 COVARIANCE MODIFICATION

Many possible schemes for modifying the covariance matrix P were proposed in "Modifications to RLS Estimation" on page 16, thus maintaining algorithm sensitivity to dynamic plant variations. This estimator uses the exponential forgetting factor approach, since it is a well-proven procedure and allows for many possible alternatives, eg. Goodwin and Sin (1984 : 64), Fortescue et al (1981). Although the regularised constant trace algorithm was considered, its implementation is extremely difficult in conjunction with Bierman's covariance factorization method. The numerical advantages of the factorization approach were deemed more significant than the type of covariance modification used.

4.4 NUMERICAL IMPLEMENTATION

As discussed in "Numerical Sensitivity" on page 21, for finite-wordlength implementations of the RLS algorithm a covariance factorization technique is more robust than the standard form. In this research, the UDU^T factorization proposed by Bierman (1977) is used to improve numerical behaviour of the algorithm. The algorithm is described in Appendix B, modified to include the relative deadzone and a forgetting factor.

4.5 COVARIANCE MODIFICATION

Many possible schemes for modifying the covariance matrix P were proposed in "Modifications to RLS Estimation" on page 16, thus maintaining algorithm sensitivity to dynamic plant variations. This estimator uses the exponential forgetting factor approach, since it is a well-proven procedure and allows for many possible alternatives, eg. Goodwin and Sin (1984 : 64), Fortescue et al (1981). Although the regularised constant trace algorithm was considered, its implementation is extremely difficult in conjunction with Bierman's covariance factorization method. The numerical advantages of the factorization approach were deemed more significant than the type of covariance modification used.

5.0 ROBUST CONTROLLER STRUCTURE

5.1 INTRODUCTION

Many different controller design procedures are possible. However, it has been shown that the most general method is that of closed-loop pole assignment (See "Pole Assignment Controllers" on page 29). The other methods described are all special forms of this technique, and lose generality in that they constrain the nature of the process to be controlled. Extensive research has been done on the adaptive control of nonminimum-phase systems in particular, eg. M'Saad, Ortega and Landau (1985), Clarke (1984), Goodwin and Sin (1981), Kumar and Moore (1983), Elliott (1982). In each of these cases, the control design approach used can be interpreted as a form of pole-placement.

The closed loop robustness of adaptive pole assignment algorithms has also been well-established, as in Astrom (1980), Anderson and Johnstone (1985), Ortega et al (1985). Practical implementations of adaptive controllers with pole-placement are documented by Wellstead et al (1979), Astrom and Wittenmark (1980), Wellstead and Sanoff (1981), Allidina et al (1981), Lin and Chen (1986).

Thus pole assignment is deemed the most general and robust approach available, and is used in this research. An **explicit** formulation is developed in this section, ie. the plant parameter estimator does not directly estimate the controller coefficients, but supplies them to a separate control design procedure.

The control philosophy is based on **certainty equivalence**, in that the plant parameter estimates are taken to describe the true plant, and thus the closed-loop pole placement assumes the real plant poles are the same as those of the estimated model. The design procedure is made robust in the following ways:

- No attempt is made to cancel the open-loop zeros (whether stable or unstable) in the closed-loop structure. This allows plants which are open-loop unstable or have unstable inverses to be effectively stabilized and controlled.

- Provision is made for deterministic disturbance nulling by the controller, as well as deterministic setpoint tracking. This is facilitated by incorporating the monic zero polynomial descriptions of the deterministic signals in the controller denominator, to ensure high feedback gain at critical frequencies. It is well known from classical control that high gain feedback at a given frequency will eliminate that component in the plant output response, as in the Internal Model Principle (Goodwin and Sin 1984 : 156). Such approaches are also documented by Puthenpura and MacGregor (1987), and Tuffs and Clarke (1985), for the elimination of deterministic disturbances in the closed loop.

- Checking is performed to ascertain whether any of the estimated plant parameters are zero, which would not allow a solution of the pole-placement equations. If this is the case, alternative solutions are used.

5.2 CONTROL SYNTHESIS

The controller structure is developed as follows:

Assume that the setpoint y^* satisfies a model of the form

$$Sy^* = 0 \qquad (5.1)$$

where S is a monic polynomial of degree δ_S and has roots on the stability boundary.

By the Internal Model Principle (Goodwin & Sin, 1984 : 156) the control law is

$$LDS.u = Py^{*} - Gy \qquad (5.2)$$

where $L = l_0 + l_1\delta + ... + \delta^t$

D = deterministic disturbance nulling polynomial as defined in the previous section. The inclusion of D and S in the controller renders deterministic modes in disturbances or in the setpoint unobservable at the system output (Goodwin and Sin 1984 : 157)

$P = p_0 + p_1\delta + ... + \delta^a$

$G = g_0 + g_1\delta + ... + \delta^c$

If we set $L' = LDS$ and $P = G$ (as is often done for convenience), we get

$$L'u = Pe \qquad (5.3)$$

where $e = (y^{*} - y)$

The closed-loop system is detailed in Figure 9 on page 56. This shows the polynomials $P(\delta)$, $L'(\delta)$, $B(\delta)$ and $A(\delta)$ in transfer function form, which is not strictly accurate. However, this representation lends itself to block diagram forms, and will be used in the ensuing development.

The closed-loop "transfer function" is

$$\frac{y}{y^{*}} = G_{CL}(\delta)$$

where

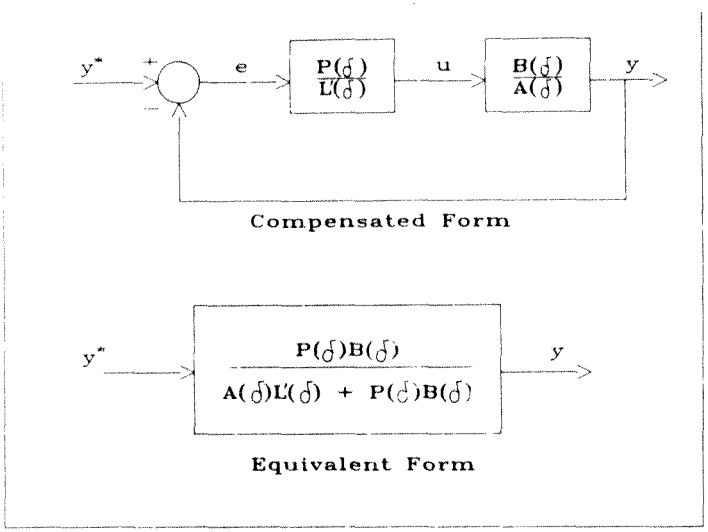

Figure 9. Closed-Loop System Structure

$$G_{CL}(\delta) = \frac{PB/L'A}{1 + PB/L'A}$$

$$= \frac{PB}{L'A + PB} \qquad (5.4)$$

The closed-loop poles are clearly given by

$$L'A + PB = A^* \qquad (5.5)$$

where A^* is some desired closed loop characteristic polynomial. The feedback controller design is accomplished by solving 5.5 for a prespecified set of closed-loop poles, viz. for some defined A^*.

As discussed earlier, the inclusion of both the D and the S polynomials in the closed-loop identity 5.5 allows perfect setpoint tracking for the class of deterministic disturbance and/or setpoint signals described by the polynomials.

We now include a feedforward component in the control signal, thus:

set $u = u' - Hz$ (5.6)

where H is some designed feedforward transfer function
z is the measured plant disturbance signal
u' is the control input signal derived fromfeedback

Then taking equation 4.4 (see "Model structure" on page 41)

$$ADy = BDu + FDz + AD\xi$$

and multiplying through by LS:

$$ADLSy = BDLSu + FDLSz + ADLS\xi \quad (5.7)$$

Now, since $ADLS = A^{*} - BP$ (from 5.5)

and $DLSt = P(y^{*} - y)$ (from 5.3)

We have (using 5.6)

$$A^{*}y = BPy^{*} + DLS(F - BH)z + ADLS\xi \quad (5.8)$$

Now to null the measured disturbance signal through feedforward, it is appropriate (from 5.8) to choose

$$H = \frac{F}{BD_1} \quad (5.9)$$

where D_1 is some stable polynomial chosen to make H proper.

However, this would constrain the polynomial $B(\delta)$ to be stable, ie. the process would have to be minimum-phase to ensure boundedness of the control signal.

Thus we factor B as

$$B = B^+B^-$$

where all the zeros of B^+ are well inside the stability region, and all the zeros of B^- are either outside the stability region or correspond to poorly damped or oscillatory modes. This idea is used frequently in the literature, eg. Astrom and Wittenmark (1980).

Then the feedforward transfer function is designed as

$$H = \frac{F}{B^+D_1} \qquad (5.10)$$

which ensures a stable control signal response. It is clear that if $B^- \neq 1$, ie. there are some poorly damped or unstable zeros in $B(\delta)$, perfect disturbance rejection through feedforward is not possible, since the polynomial term

$$(F - BH)z$$

is not exactly cancelled. However, general process models (both minimum- and nonminimum-phase) can be handled by this procedure.

5.3 CONCLUSION

This section has developed the overall structure for the pole placement controller. The feedback control coefficients are derived by solving

equation 5.5, to give the coefficents l_i and p_i in the L and P polynomials. A closed-loop characteristic polynomial A^* must be determined before the adaptive control calculations are performed, to place the poles at desired locations. Deterministic disturbance nulling is performed in closed loop, by including nulling polynomials in the controller. The feedforward transfer function H is determined by using 5.10, for some prespecified D_i. This control design would be performed each time the parameters were updated, to give the new control coefficients corresponding to the new plant model estimated. The selection of a suitable A^*, and the practical solution of the Diophantine equation is given in the next section.

6.0 PRACTICAL IMPLEMENTATION CONSIDERATIONS

6.1 INTRODUCTION

This section discusses aspects of the practical *implementation* of the robust estimator and controller. This *encompasses* the choice of prefiltering, *estimator parameter* selection, and other user choices such as sampling frequency and model structure. A practical discrete algorithm for the controller is given which corresponds to the familiar PID form. The rationale for selecting a suitable closed-loop polynomial A^* is also developed. Finally, some software considerations are described.

6.2 GENERAL CHOICES

6.2.1 SELECTION OF SAMPLING PERIOD

Classically, the sampling frequency should be chosen according to the Nyquist criterion, namely

$$\omega_s \geq 2\omega_p$$

where ω_s is the sampling frequncy
ω_p is the bandwidth of the plant model

However, Ljung (1987 : 382) suggests that if data acquisition is costless, then sampling should be performed as fast as possible. It is clear that slow sampling yields data sets which are less informative, but it must

also be realised that the cost effectiveness of new information will decrease as one samples faster and faster.

Furthermore, with the delta-operator formulation used in this implementation a higher sampling rate will not result in a nonminimum-phase discrete model (See "Delta-operator formulation of Discrete-Time Control" on page 33). Thus fast sampling would be desirable, but a point is reached above which the advantages are negligible for faster sampling.

Ljung (1987 : 382) describes a further tradeoff, in that the faster the sampling rate the more noise susceptible the measured signals. Thus one must balance noise reduction against relevance to plant dynamics. The same author further reminds us that the same sampling interval should be used for control as is used to build the discrete model. Katz (1981 : 229) suggests that this control output signal should use a fast sampling rate, to provide a "smoother" control action. He describes the nature of the zero-order-hold control signal in terms of a "roughness function", to quantify its effect on the continuous-time system and actuators.

Finally, Seborg et al (1986) recommend the selection of the sampling rate such that the dead time t_d is an integral multiple of the sampling interval, ie.

$$t_d = kT_s \qquad k \in (0;1;2; \ldots.)$$

This is motivated to avoid nonminimum-phase discrete models (Goodwin and Sin 1984 : 229), and is clearly only possible when the plant dead-time is known *a priori*. It would be advantageous only in circumstances where t_d is not expected to be variable over a large range. This requirement also becomes insignificant when using a δ-operator process description (See "Delta-operator formulation of Discrete-Time Control" on page 33).

6.2.2 SELECTION OF MODEL ORDER

The estimator uses a fixed model structure, but this is extensible to any order. Although real systems are infinite-dimensional, the advantages of higher-order models can be questioned. Ljung and Soderstrom (1983 : 264) describe a tradeoff between model complexity and adequate system description, involving the comparison of models of different orders to ascertain whether higher-order descriptions are worthwhile. Generally, such a facility for parallel identification is not available.

A further consideration arises from the adaptive control law design. Using closed-loop pole assignment, the model order directly affects the order of the controller design. This gives increased computational overhead, as well as resulting in a complex controller structure. In this research, a simple second-order model has been used, to yield a controller which has a PID-related structure.

6.3 IMPLEMENTATION OF THE ROBUST ESTIMATOR

6.3.1 INPUT-OUTPUT SIGNAL NORMALISATION

Wittenmark and Astrom (1984) suggest that the inputs and outputs of the process should be scaled to lie within the same magnitude range. This will improve numerical behaviour of the estimator and control law. Astrom (1987) also uses a variable normalising factor on the measured plant variables to discuss estimator convergence.

In this work, the input and output signals are shifted and scaled such that they lie in the range

$$0 \leq (y(k\Delta), u(k\Delta)) \leq 1$$

This has the additional advantage of allowing estimator parameter settings to remain fixed for a fairly wide range of real systems. The signal spans and zeros used by the algorithm can then simply be adjusted to give the range used above for different plants.

6.3.2 SIGNAL CONDITIONING

6.3.2.1 Low-Pass Filtering

It has been noted that in the implementation of the filters used to realise the 1/E transfer functions, while a standard controller form is numerically well-behaved, the filters should be unity-gain. If this is not the case, and the filters are implemented as

$$\frac{1}{E} = \frac{1}{\delta^n + e_{n-1}\delta^{n-1} + \ldots\ldots + e_0}$$

then the filter would have an effective d.c. gain of $1/e_0$. This scaling would render the internal variables of the estimator numerically sensitive to finite-wordlength errors.

For unity-gain, the filters are implemented as shown in Figure 10 on page 65, to ensure that the signals are in "reasonable" ranges.

Phillips and Nagle (1984 : 471) discuss the implications of finite wordlengths in digital filter implementation, and suggest the use of scaling factors to ensure boundedness of the internal filter signals. Various techniques for determining suitable factors are discussed. In

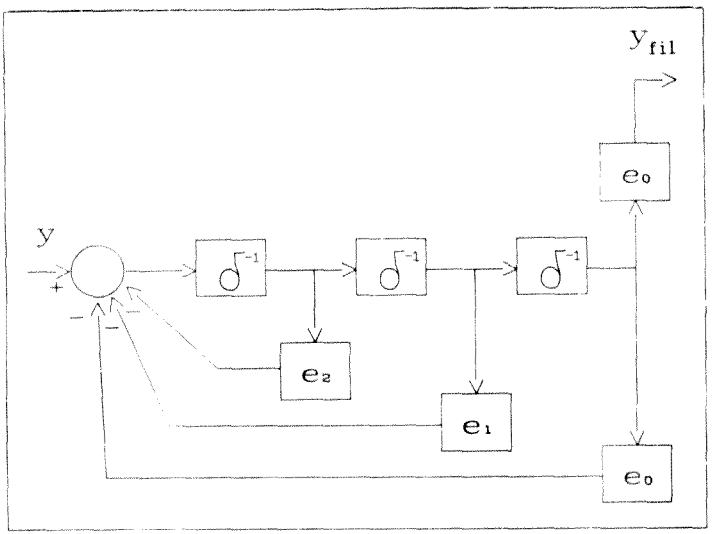

Figure 10. Practical Implementation of the I.E Filter

this work it is considered sufficient to use unity-gain filtering in conjunction with the input-output normalisation discussed earlier.

The selection of the cutoff frequency ω_E for the low-pass filter is made by examining the desired bandwidth range of the closed-loop system. The E-filter dictates the bandwidth limit for the validity of the estimated model, and thus ω_E should at least overbound the dominant plant dynamics to be modelled. On the other hand, the effects of unmodelled high frequency dynamics and noise must be eliminated from the estimator, thus ω_E should not be made too high. A fundamental constraint is that ω_E should obey the Nyquist Criterion, viz

$$\omega_E \leq \tfrac{1}{2}\omega_s$$

with ω_s = sampling frequency

Under normal operating conditions, ω_E would be selected in relation to the closed-loop characteristic polynomial A^*. The fastest pole of A^* should not be faster than ω_E, since the plant model may not be valid beyond this range.

6.3.2.2 High-Pass Filtering

This filter emerged from the deterministic nulling concept detailed in "Model structure" on page 41, giving

$$G_{HP}(\delta) = \frac{\delta}{\delta + \varepsilon}$$

to null a d.c offset or constant disturbance. This has a frequency response as detailed earlier in Figure 7 on page 45. We briefly examine the validity of this approach by comparing it with some of the other techniques for eliminating d.c. values.

Isermann (1982) proposes three approaches :

1. Averaging :

The d.c. value is given by

$$y_{dc} = \frac{1}{N} \sum_{k=1}^{N} y(k)$$

or recursively

$$y_{dc}(k) = y_{dc}(k-1) + [y(k) - y_{dc}(k-1)]/k \qquad (6.1)$$

This value can be subtracted from y(k) to give only dynamic variations about zero.

2. Estimation of a Constant

A constant value could be explicitly included in the DARMA model, thus :

$$Ay = Bu + \xi + y_{dc} \tag{6.2}$$

y_{dc} would simply become another parameter to be estimated by the algorithm.

3. Differencing

The variations in u(k) and y(k) are calculated at each sample by

$$\begin{aligned} \Delta u(k) &= u(k) - u(k-1) \\ \Delta y(k) &= y(k) - y(k-1) \end{aligned} \tag{6.3}$$

These values are then used by the estimator.

The first two methods require additional computation in the estimator algorithm, and will only give reliable d.c. estimates over a long period. The third approach is a type of signal conditioning, similar to that of high-pass filtering suggested here. However, the method of *differencing* corresponds to a function

$$\Delta y(k) = (1 - q^{-1})y(k)$$

which is a discrete differentiator. The *overall estimator frequency* "window" in conjunction with the 1/E filter would then be as shown in Figure 11 on page 68. On the other hand, the frequency "window" of the estimator using the high pass filter model, ie.

$$W(\delta) = \frac{D(\delta)}{D'(\delta)E(\delta)} \tag{6.4}$$

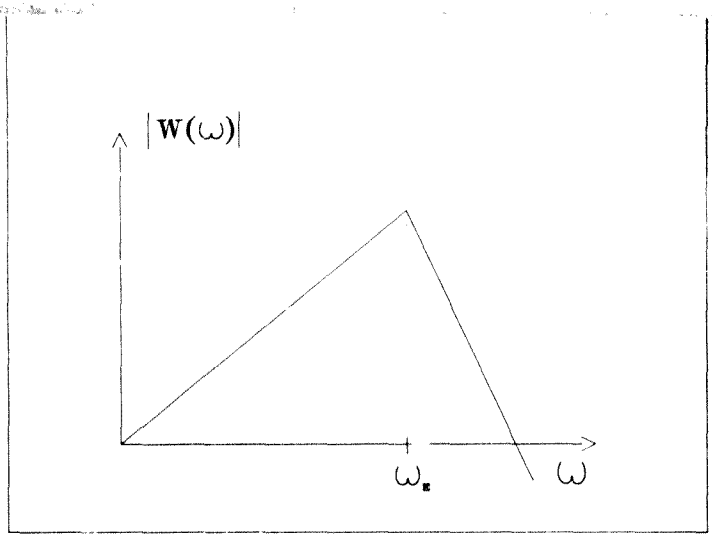

Figure 11. Frequency "Window" for Method of Differencing

is shown in Figure 12 on page 69. This latter is clearly more desirable, since it avoids the passband distortion which would be present with the method of differencing.

The choice of a suitable high-pass frequency ε is not very critical. This can be observed in that the estimator generates a "transfer function"-type model, of the form

$$G(\delta) = \frac{y(k\Delta)}{u(k\Delta)}$$

and we are operating on both the input and the output by the same window $W(\delta)$, thus

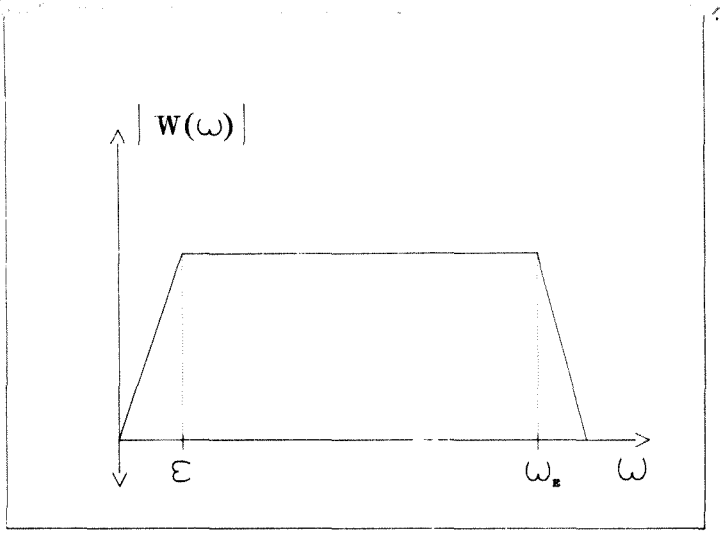

Figure 12. Frequency "Window" for High-Pass Filter

$$\frac{W(\delta)y(k\Delta)}{W(\delta)u(k\Delta)} = \frac{y(k\Delta)}{u(k\Delta)} = G(\delta)$$

A general guideline is that ε should not be made so high as to become comparable with the slowest plant dynamics. On the other hand, ε should not be too low since this would allow for very slowly varying components to be passed to the estimator, which would be seen as d.c. offsets. As previously discussed, this is known to cause problems in estimation (Astrom 1983).

6.3.2.3 Initialisation of Discrete Filters

The pseudo-code implementation of the discrete high-pass filter is simply

```
YOUT = YIN - EPS*XHP
XHP = XHP + DELTA*YOUT
```

This corresponds to the block diagram of Figure 13 on page 71. XHP is an internal state variable of the filter. On startup, if XHP is initialised to zero, the input value to the filter will immediately be transferred to the output. If the estimator is started up when the plant is at rest, any d.c. value on the signals will be passed through to the estimation algorithm giving incorrect estimates. It would then take some time before the filter state XHP had "charged up" and canceled the d.c. value. To eliminate this phenomenon, the filter is initialised using the first signal value read, such that YOUT is zero on startup. This is accomplished by setting

```
XHP = YINITIAL/EPS                                    (6.5)
```

where YINITIAL is the first signal value read.

6.3.3 DEADZONE PARAMETER SELECTION

The choice of the constant values used by the relative deadzone is difficult. However, some general guidelines can be given:

1. σ_s : This value affects the rate of tracking of the error e, by the deadzone magnitude function Bm(k). This can be seen by examining equation 4.13, which indicates that σ_s is effectively the pole position of a first-order response on m(k). This is a digital filter, and thus we may approximate a value for σ_s by

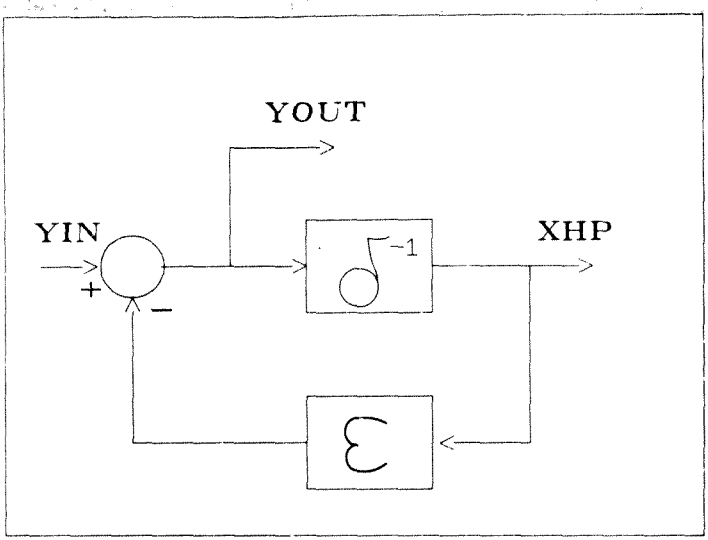

Figure 13. Block Diagram o. Discrete High-Pass Filter

$$\sigma_\epsilon = z_i = e^{s\Delta}$$

where z_i = the discrete first-order pole
s = the continuous-time desired pole position
Δ = sampling interval

The value for s may be chosen to be twice the fastest plant pole, say, which would ensure fast tracking of the error by the deadzone.

2. m_ϵ : It has been found that the initial value for m(k) does not affect the long-term performance of the deadzone. This value can be chosen as $m_\epsilon = 0$, and m(k) will rapidly track the error (for fairly fast σ_ϵ).

3. ε_0 : This is the minimum value for the deadzone. It should be chosen with regard to the lowest expected noise threshold, as well as allowing a minimum margin for plant-model mismatch. The choice of this value may be governed by the maximum resolution of the analog/digital conversion process, since this would limit the smallest noise level able to be discerned. On the other hand, if a process known to be of high order or to be nonlinear, is approximated by a low-order linear model, a larger value for ε_0 is indicated since "modelling noise" is large. Similarly in practical circumstances where Signal/Noise ratios are low, a larger ε_0 should be chosen. For average conditions, typically ε_0 may be 0,5 % of the input signal range.

   For $0 \leq u(k\Delta) \leq 1$ one may use $\varepsilon_0 = 0{,}005$, say.

4. ε_1, ε_2 and ε_3 : These values weight the measured signals. The input signal $u(k\Delta)$ is seen as most significant in determining the size of the deadzone, since "persistent excitation" implies a constantly exciting input. Thus the coefficient ε_1 is chosen to give roughly the same effect as ε_0 when $u(k\Delta)$ is a maximum. This is found by solving $\varepsilon_1 u(k\Delta)_{max} = \varepsilon_0$. For a normalised input signal, $u(k\Delta)_{max} = 1$, and a suitable value is given by $\varepsilon_1 = \varepsilon_0$. The values for ε_2 and ε_3 are selected to be much smaller than ε_1, typically

$$0{,}1\varepsilon_1 \leq (\varepsilon_2 \; ; \; \varepsilon_3) \leq 0{,}5\varepsilon_1 .$$

5. α, ε_4 : The value chosen for α strongly affects the convergence rate of the estimator, as can be seen from equation 4.15 and 4.17. (α is the maximum value of the magnitude function $a(k)$, which is in turn the gain value for both the parameter vector and P-matrix update). Thus it would seem that a value for α fairly close to unity is appropriate. However, the relationship 4.14 shows that for large α, a large value for β results. Typically, if we take ε_4 to be negligibly small, we have

$$\beta = \frac{1}{\sqrt{1 - \alpha}}$$

Some approximate values are given below :

$\alpha = 0,5$	$\beta = 1,4$
$\alpha = 0,75$	$\beta = 2$
$\alpha = 0,9$	$\beta = 3,2$
$\alpha = 0,95$	$\beta = 4,5$
$\alpha = 0,99$	$\beta = 10$

Thus it is clear that β increases dramatically as α approaches unity. From equation 4.15, it is clear that the algorithm update is turned off when

$$e(k) < \beta m(k)$$

where $m(k)$ is already calculated such that it overbounds the filtered error term $e(k)$.

This is illustrated in Figure 14 on page 74. Since the ε_i coefficients have been selected in $m(k)$ to account for process noise and modelling error, we do not wish to scale this deadzone up much more.

A tradeoff is clearly necessary between higher algorithm gain (large α) and large deadzone size (large β). This relationship is required by convergence (see Appendix A), and means broadly that the larger the RLS gain, the more the estimator must be protected from erroneous updating by increasing the deadzone.

Typically a smaller value for α is preferred, since although the algorithm gain is smaller, more updates will be performed and convergence is still assured. This effect is illustrated in the practical results of the next section.

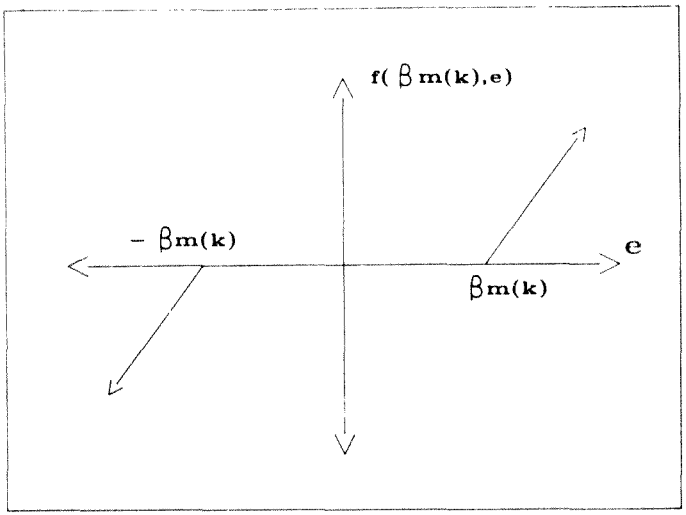

Figure 14. Relative Deadzone Function showing scaling

6.3.4 OTHER PARAMETERS

6.3.4.1 Choice of Forgetting Factor

It will be recalled that the forgetting factor approach minimises a cost function

$$J = \sum_{i=1}^{t} \lambda^{t-i} e(i)^2$$

For λ fairly close to unity, which is generally the case in practice, we can write

$$\lambda^i = e^{i.\ln(\lambda)} \approx e^{i.\ln(\lambda-1+1)} \approx e^{i(\lambda-1)}$$

This gives an exponential decay time constant of

$$T_\theta = 1/(1 - \lambda) \qquad (6.6)$$

(Ljung and Soderstrom 1983 : 274)

T_θ can be seen to represent the number of historical samples which are significant to the present estimate $\theta(k)$.

Thus for $\lambda = 0,99$ $T_\theta = 100$

$\lambda = 0,95$ $T_\theta = 20$

A value for T_θ between 20 and 100 samples appears to be satisfactory for *general applications*.

Alternatively, one of the variable forgetting factor approaches can be used, as discussed in "Exponential Data Weighting" on page 16.

6.3.4.2 Choice of Initial Values

As discussed in "Recursive Least Squares Algorithm" on page 14, the choices of the values of $\theta(0)$ and $P(0)$ are interdependant. If a good estimate of $\theta(0)$ is available, a smaller initial covariance matrix $P(0)$ may be used. On the other hand, a large $P(0)$ indicates poor confidence in $\theta(0)$, since high gain estimation updates will be performed initially.

Typically, for input and output signals normalised as

$$0 \leq (u(k\Delta) \; ; \; y(k\Delta)) \leq 1$$

and an unknown plant, we simply set $\theta(0) = 0$ and choose

$$1 \leq P(0) \leq 100$$

Ljung and Soderstrom (1983 : 299) give a statistical interpretation of these two choices, and document some simulation results to illustrate different selections.

6.4 IMPLEMENTATION OF THE ROBUST CONTROLLER

6.4.1 POLE PLACEMENT DESIGN

The pole placement problem is solved here for a specific case, using a second-order plant model. This is a desirable approach since it allows adequate modelling of many industrial processes, as well as resulting in a second-order controller which is easily interpreted in PID-form.

The following polynomial orders are used :

degree(A) = 2
degree(B) = 1
degree(F) = 2
degree(E) = 2
degree(D) = 1
degree(S) = 0
degree(P) = 2
degree(L) = 1

where all polynomials have the same meaning as used in "Model structure" on page 41 and "Robust Controller Structure" on page 53.

D and S are simply

$D = \delta$ (constant d.c. offset disturbance)

$S = 1$ (no deterministic component in the setpoint)

The structure for the controller is given by

$$G_c(\delta) = \frac{p_2\delta^2 + p_1\delta + p_0}{\delta(l_1\delta + l_0)} \qquad (6.7)$$

Note the emergence of an integral term in the controller. This is a natural consequence of the inclusion of the disturbance nulling polynomial D in the controller denominator, as described by Tuffs and Clarke (1985) and Allidina and Yin (1985) for elimination of steady-state error.

The controller is designed by solving the Diophantine equation 5.5, thus:

$$ALDS + BP = A^* \qquad (6.8)$$

A^* is the desired closed-loop characteristic polynomial, and it emerges from 6.8 that degree(A^*) = 4 in this case. A general rule is that

$$\text{degree}(A^*) = 2.\text{degree}(A)$$

Thus $A^* = \delta^4 + \alpha_3\delta^3 + \alpha_2\delta^2 + \alpha_1\delta + \alpha_0$

Equation 6.8 expanded gives

$$(\delta^2 + a_1\delta + a_0)\delta(l_1\delta + l_0) + (b_1\delta + b_0)(p_2\delta^2 + p_1\delta + p_0)$$
$$= \delta^4 + \alpha_3\delta^3 + \alpha_2\delta^2 + \alpha_1\delta + \alpha_0 \qquad (6.9)$$

In matrix form, this is as follows :

$D = \delta$ (constant d.c. offset disturbance)

$S = 1$ (no deterministic component in the setpoint)

The structure for the controller is given by

$$G_c(\delta) = \frac{p_2\delta^2 + p_1\delta + p_0}{\delta(l_1\delta + l_0)} \qquad (6.7)$$

Note the emergence of an integral term in the controller. This is a natural consequence of the inclusion of the disturbance nulling polynomial D in the controller denominator, as described by Tuffs and Clarke (1985) and Allidina and Yin (1985) for elimination of steady-state error.

The controller is designed by solving the Diophantine equation 5.5, thus:

$$ALDS + BP = A^* \qquad (6.8)$$

A^* is the desired closed-loop characteristic polynomial, and it emerges from 6.8 that degree(A^*) = 4 in this case. A general rule is that

$$\text{degree}(A^*) = 2.\text{degree}(A)$$

Thus $A^* = \delta^4 + \alpha_3\delta^3 + \alpha_2\delta^2 + \alpha_1\delta + \alpha_0$

Equation 6.8 expanded gives

$$(\delta^2 + a_1\delta + a_0)\delta(l_1\delta + l_0) + (b_1\delta + b_0)(p_2\delta^2 + p_1\delta + p_0)$$

$$= \delta^4 + \alpha_3\delta^3 + \alpha_2\delta^2 + \alpha_1\delta + \alpha_0 \qquad (6.9)$$

In matrix form, this is as follows :

$$\begin{bmatrix} 1 & 0 & 0 & 0 & 0 \\ a_1 & 1 & b_1 & 0 & 0 \\ a_2 & a_1 & b_2 & b_1 & 0 \\ 0 & a_2 & 0 & b_2 & b_1 \\ 0 & 0 & 0 & 0 & b_2 \end{bmatrix} \begin{bmatrix} l_1 \\ l_2 \\ p_2 \\ p_1 \\ p_0 \end{bmatrix} = \begin{bmatrix} 1 \\ \alpha_1 \\ \alpha_2 \\ \alpha_1 \\ \alpha_0 \end{bmatrix} \qquad (6.10)$$

A unique sequential solution is obtained for p_2, p_1, p_0, l_1 and l_0 as follows :

1. $l_1 = 1$

2. $p_0 = \alpha_0 / b_0$

3. $l_0 = X/Y$ where

   - $X = \alpha_2 - (b_0/b_1)\alpha_1 - (b_1/b_0)\alpha_1 - a_0 - (b_0/b_1)a_1)l_1 + (b_1^2/b_0)p_0$
   - $Y = a_1 - b_0/b_1 - (b_1/b_0)a_0$

4. $p_1 = (\alpha_1 - a_0 l_0 - b_1 p_0)/b_0$

5. $p_2 = (\alpha_2 - a_1 - l_0)/b_1$

It is possibla, however, that $b_1 = 0$ ie. there is no zero in the B-polynomial. In this case the solution above would give a division by zero error, and an *alternative* solution is required :

1. $l_1 = 1$

2. $p_0 = \alpha_0 / b_0$

3. $l_0 = \alpha_1 - a_1$

4. $p_1 = (\alpha_1 - a_0 l_0)/b_0$

5. $p_2 = (a_2 - a_0 - a_1 l_0)/b_0$

In practice, the controller design would be poorly conditioned if b_1 were very small, thus a test is required of the nature of

```
IF |b0/b1| < 2wE THEN
    ( Algorithm 1 )
ELSE
    ( Algorithm 2 )
ENDIF
```

The feedforward block design would be relatively simple in this case, since the root of $B(\delta)$ is simply at

$$\delta = -b_0/b_1$$

Thus if both b_1 and b_0 have the same sign, we can use

$$B^+(\delta) = B(\delta)$$
$$B^-(\delta) = 1 \qquad \text{(all zeros stable)}$$

and solve equation 5.10 thus :

$$H(\delta) = \frac{F(\delta)}{B^+(\delta)D_1(\delta)}$$

Note : Since degree$(F) = 2$, we need to choose D_1 such that

$$\text{degree}(D_1) = 2 - \text{degree}(B^+) \qquad (6.11)$$

to make $H(\delta)$ realisable.

The system with controller is detailed in Figure 15 on page 80. Solutions for higher order systems and controllers are given for completeness in Appendix C.

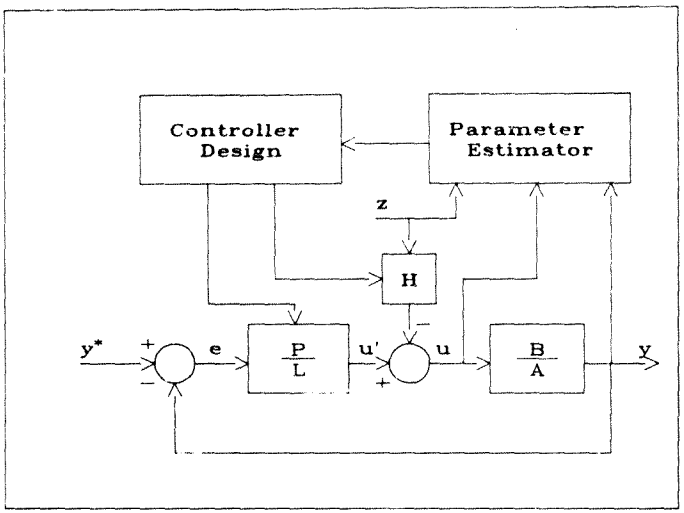

Figure 15. Closed-Loop System with Controller

6.4.2 INTERPRETATION AS A DIGITAL PID CONTROLLER

The pole assignment solution of the previous section yields a controller structure compatible with traditional PID designs. This is appealing from an engineering point of view, due to the widespread usage of fixed-parameter PID controllers. As stated by Ortega and Kelly (1984) :

> "An overwhelming majority of the regulators used in industry are PID, because when properly tuned they generally achieve satisfactory performance."

Digitally implemented PID controllers have been documented by many authors, as in Astrom and Wittenmark (1984 : 180), Kuo (1980 : 509),

Phillips and Nagle (1984 : 254), Jacquot (1981 : 77), and Phillips and Parr (1984).

Practical **adaptive** controllers with PID structure are also described in the literature, as by Cameron and Seborg (1983), Seborg et al (1986), Ortega and Kelly (1984), Andreiev (1981), and Kraus and Myron (1984). Most of these use techniques other than that of closed-loop pole assignment for controller design, however Ortega and Kelly (1984) develop both an implicit and explicit formulation based on pole placement.

There are many different descriptions of an analog PID controller. The standard "textbook" form is

$$m(t) = K_c\left[e(t) + (1/T_i)\int e(t)dt + T_d(de/dt)\right]$$

or in Laplace domain

$$M(s) = [P(s) + I(s) + D(s)]E(s)$$

where $P(s) = K_c$

$$I(s) = \frac{K_c}{T_i s}$$

$$D(s) = \frac{K_c T_d s}{aT_d s + 1} \qquad 1/20 \leq a \leq 1/5$$

(After Macleod, 1987) (6.12)

The inclusion of the factor $1/(aT_d s+1)$ in the derivative term is to make it realisable, since a pure differentiator is non-causal. The constant a is usually fixed by the controller manufacturer and represents some low-pass filtering characteristic to bandlimit the overall controller frequency response.

Alternative formulations of the controller are :

$$M(s) = \frac{K_c}{aT_d s + 1}(1 + 1/(T_i s) + T_d s)E(s) \tag{6.13}$$

This corresponds to all terms filtered by the fixed low-pass filter.

$$M(s) = K_c[1 + 1/(T_i s)]\left[\frac{1 + T_d s}{aT_d s + 1}\right]E(s) \tag{6.14}$$

This is a type of "cascaded" realisation, which is equivalent to the others if $T_i > 4T_d$, which is generally true in practice.

It is instructive to examine the frequency response of a PID controller. This is shown in Figure 16 on page 83, and indicates a special type of lag-lead compensation. It can be seen that T_i and T_d both determine zero positions of the controller, with T_i governing the low-frequency response and T_d the higher frequencies. A PI controller is in fact of phase-lag type, with high gain at low frequencies to reduce steady-state error, and an increase in stability. The PD form is a phase-lead controller, which advances the phase at higher frequencies to increase system bandwidth, thus speeding up the closed-loop response. Since the gain-crossover frequency is shifted up, the stability margin is also improved. Noise amplification at high frequencies is limited by the pole at $1/(aT_d)$.

To find a suitable approximation to the continuous PID form, we use 6.13 with all terms filtered. This is rearranged as

$$M(s) = \frac{K_c}{T_i T_d a}\cdot\frac{(T_i T_d s^2 + T_i s + 1)}{s(s + 1/(aT_d))}$$

and approximating with $s = \delta$

$$\frac{K_c}{T_i T_d a}\cdot\frac{(T_i T_d \delta^2 + T_i \delta + 1)}{\delta(\delta + 1/(aT_d))} = \frac{p_2\delta^2 + p_1\delta + p_0}{\delta(\delta + 1_0)}$$

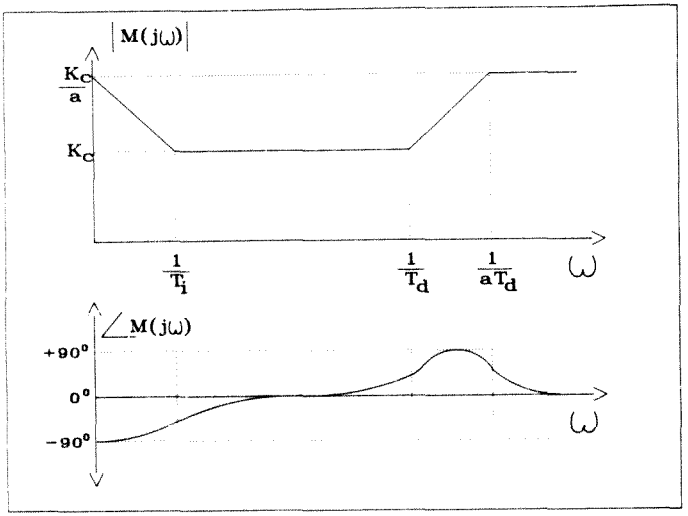

Figure 16 Frequency Response of PID Controller

Then $p_2 = K_c/a$

$p_1 = K_c/(T_d a)$

$p_0 = K_c/(T_i T_d a)$

$l_0 = 1/(T_d a)$

This gives the solution

$K_c = p_1/l_0$

$T_i = p_1/p_0$

$$T_d = p_2/p_1 \qquad (6.15)$$

This solution appears reasonable, since the gain is determined by a ratio of $p_1/1_1$, and the integral and derivative terms by the controller zeros given by the p_i coefficients. The fixed constant a does not affect the PID values, which seems acceptable since this value will vary from manufacturer to manufacturer as previously discussed. Many other alternative solutions can be found using the other continuous PID descriptions. Some of these are provided in Appendix D.

Examining the discrete PID form, a further modification is needed to suitably implement the integral and derivative terms. The integral term is approximated simply using an Euler forward difference. However, as in Astrom and Wittenmark (1984 : 180) and Jacquot (1981 : 79) a backward difference must be used for the derivative term. This is because a forward difference implementation would lead to an unstable discrete controller for small values of T_d.

Figure 17 on page 85 illustrates the mappings of the stable region of the s-plane onto the z-plane using both forward and backward derivative approximations (After Macleod, 1987).

Our definition of the δ-operator is derived using a forward difference (See "Delta-operator formulation of Discrete-Time Control" on page 33). Hence we need to define a new **backward difference** δ-operator, thus :

$$\delta_1 = \frac{1 - q^{-1}}{\Delta} \qquad (6.16)$$

The relationship with the normal δ-operator is as follows :

$$\delta_1 = \frac{q - 1}{q\Delta} = \frac{\delta}{q} = \frac{\delta}{\delta\Delta + 1} \qquad (6.17)$$

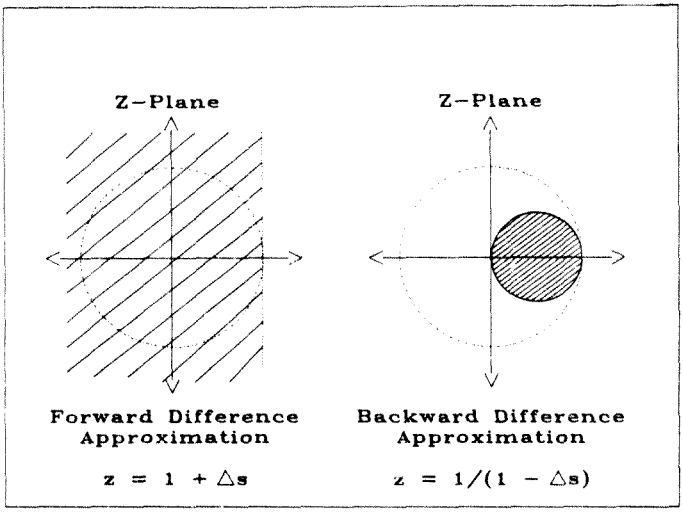

Figure 17. Z-Plane Mappings for Forward and Backward Difference Derivative Approximations

The discrete PID controller is then derived using δ for the integral term, and δ_1 for the derivative term and fixed low-pass filter.

Then in 6.13 we have

$$M(\delta) = K_c(1 + 1/(T_i\delta) + T_d\delta_1)/(aT_d\delta_1 + 1)$$

Substituting back from 3.1 and 6.16 to get shift-operator form :

$$M(q) = K_c\left[1 + \frac{\Delta}{T_i(q-1)} + \frac{T_d}{\Delta}(1 - q^{-1})\right]\left[\frac{q\Delta}{q(\Delta + aT_d) - aT_d}\right] \tag{6.18}$$

A further refinement is now introduced, in accordance with Astrom and Wittenmark (1984 : 183) to implement **bumpless transfer**. This means that the algorithm must ensure that the internal states of the regulator are such, that the control output does not undergo a step change on switchover between manual and automatic control. Such a step change could easily occur if the algorithm were implemented as in 6.18, since the value of the integrator may not give the same error-driven output as the manual control signal applied to the actuator. To overcome this phenomenon, the algorithm is implemented in an **incremental** form, where M(q) only generates the change in control signal Δu. This will guarantee a more gradual change if the calculated output differs from the manual setting on switchover.

Then $\Delta u = (1 - q^{-1})u = u_k - u_{k-1}$

$= (1 - q^{-1})M(q)e(k)$

We implement

$$M'(q) = (1 - q^{-1})M(q)$$

$$= K_c\left[1 - q^{-1} + q^{-1}\Delta/T_i + (T_d/\Delta)(1 - 2q^{-1} + q^{-2})\right].\left[\frac{q\Delta}{q(\Delta + aT_d) - aT_d}\right] \qquad (6.19)$$

A problem in 6.19 is apparent, in that the integral term has a delay of on sample relative to the other two terms. This is overcome simply by using the backward difference operator δ_1 for this term as well in 6.13, giving the incremental form

$$M'(q) = (1 - q^{-1})M(q)$$

$$= K_c\left[1 - q^{-1} + \Delta/T_i + (T_d/\Delta)(1 - 2q^{-1} + q^{-2})\right].\left[\frac{q\Delta}{q(\Delta + aT_d) - aT_d}\right] \qquad (6.20)$$

The low pass filter on all terms is implemented by rearranging as

$$H_{LP}(q) = \frac{q\Delta}{q(\Delta + aT_d) - aT_d}$$

$$= \frac{\Delta q/(\Delta + aT_d)}{q - aT_d/(\Delta + aT_d)}$$

$$= \frac{a_1}{(1 - a_2 q^{-1})} \qquad (6.21)$$

where $a_1 = \Delta/(\Delta + aT_d)$

$$a_2 = aT_d/(\Delta + aT_d) = (1 - a_1)$$

In industrial PID controllers, the cutoff frequency of this filter is fixed as mentioned earlier. Thus constant values for a_1 and a_2 are used in the implementation. The algorithm can be seen in the software listing of Appendix E.

6.4.3 CHOICE OF CLOSED-LOOP CHARACTERISTIC POLYNOMIAL

Pole assignment attempts to shift the open-loop system poles to desired locations. This may often be done to increase bandwidth by "speeding up" the system poles, or improve stability by moving plant poles which are close to the stability boundary further away. For underdamped modes, pole assignment may also be used to alter the transient response. If the closed loop bandwidth is increased, caution should be exercised. If the shifted poles are made too fast, actuator saturation may result. Furthermore, since the controller design is based on a bandlimited estimated process model, nonlinearities may be exhibited if the bandwidth is exceeded for certain plants.

We then factor A^* as

$$A^* = A'A_s$$

where A' are the shifted plant poles to give the desired closed-loop response

A_s are the remaining closed-loop poles

(After Goodwin and Sin, 1984 : 148)

and A_s, A' are of order n-1 and n respectively, to give

$$\text{degree}(A^*) = 2n$$

The rationale for selecting the other closed-loop pole locations A_s emerges from considering pole assignment in state-space terms. The control law is then formulated as a state observer plus state-variable feedback (Goodwin and Sin 1984 : 148).

To facilitate this, factorise $A'(q^{-1})$ further as

$$A'(q^{-1}) = A(q^{-1}) + K(q^{-1})$$

where $K(q^{-1}) = k_1q^{-1} + \ldots + k_nq^{-n}$

$A(q^{-1})$ = process model denominator polynomial as before

Then

$$A^*(q^{-1}) = A_s(q^{-1})\left[A(q^{-1}) + K(q^{-1})\right] \qquad (6.22)$$

Define the filter polynomial $R(q^{-1})$ as

$$R(q^{-1}) = L(q^{-1}) - A_s(q^{-1}) \qquad (6.23)$$

where $L(q^{-1})$ is the control law polynomial used in the previous formulation.

The generalised control law

$$L(q^{-1})u(t) = M(q^{-1})y^{*}(t+d) - P(q^{-1})y(t)$$

becomes

$$A_0(q^{-1})u(t) = M(q^{-1})y^{*}(t+d) - \left[R(q^{-1})u(t) + P(q^{-1})y(t)\right] \qquad (6.24)$$

The second terr on the right hand side of 6.24 can be expressed as a function of a "state vector", by describing the system as

$$A(q^{-1})z(t) = u(t)$$

$$y(t) = B(q^{-1})z(t)$$

Using this form, 6.24 becomes

$$A_0(q^{-1})u(t) = M(q^{-1})y^{*}(t+d) - \left[R(q^{-1})A(q^{-1}) + P(q^{-1})B(q^{-1})\right]z(t) \qquad (6.25)$$

Using the Diophantine identity

$$L(q^{-1})A(q^{-1}) + P(q^{-1})B(q^{-1}) = A^{*}(q^{-1})$$

in conjunction with 6.22 and 6.23 gives

$$R(q^{-1})A(q^{-1}) + P(q^{-1})B(q^{-1}) = A_0(q^{-1})K(q^{-1}) \qquad (6.26)$$

and 6.25 becomes

$$A_0(q^{-1})u(t) = M(q^{-1})y^{*}(t+d) - A_0(q^{-1})K(q^{-1})z(t)$$

This is expressed as a "transfer function" (for stable A_0q^{-1}) thus :

$$u(t) = \frac{M(q^{-1})}{A_0(q^{-1})}y^{*}(t+d) - K(q^{-1})z(t) \qquad (6.27)$$

The dynamics represented by $A_s(q^{-1})$ allow estimation of the state vector $z(t)$. These are effectively **observer dynamics.**

The state feedback interpretation is detailed in Figure 18 on page 91. This is a physically realisable system, and is equivalent to the previous formulation (O'Reilly 1983 : 185). However, implementing the pole placement as in "Robust Controller Structure" on page 53, we need to incorporate the dynamics into the control law design. This can also be seen in the fact that the closed-loop polynomial $A^*(q^{-1})$ is twice the dimension of the plant polynomial $A(q^{-1})$, ie. we are designing a state observer and feedback controller together during pole assignment. The observer dynamics then emerge in the closed loop, seen by

$$\frac{y(t)}{y^*(t+d)} = \frac{P(q^{-1})B(q^{-1})}{A'(q^{-1})A_s(q^{-1})} \qquad (6.28)$$

where $P(q^{-1}) = M(q^{-1})$ for simplicity in our case.

This is made clearer by re-examining the Diophantine identity :

$$L(q^{-1})A(q^{-1}) + P(q^{-1})B'(q^{-1}) = A^*(q^{-1}) = A'(q^{-1})A_s(q^{-1})$$

The inclusion of observer poles in the closed-loop polynomial is described frequently in the literature on adaptive pole placement, eg. Samson and Fuchs (1981), Kraft (1979), Elliott and Wolovich (1979), Tsay and Shieh (1981), Astrom (1983).

Now, $L(q^{-1})$ includes polynomial descriptions of deterministic signals, to null possible deterministic modes in disturbances or in the setpoint (See "Control Synthesis" on page 54). In "Pole Placement Design" on page 76, this resulted in integral action for nulling constant signals. Thus $A_s(q^{-1})$ will contain observer dynamics to estimate these deterministic signal nulling modes. This is also described by Kailath (1980 : 276).

However, the observer dynamics must be assymptotically stable for the observation errors to decay. Taking D and S to describe the deterministic

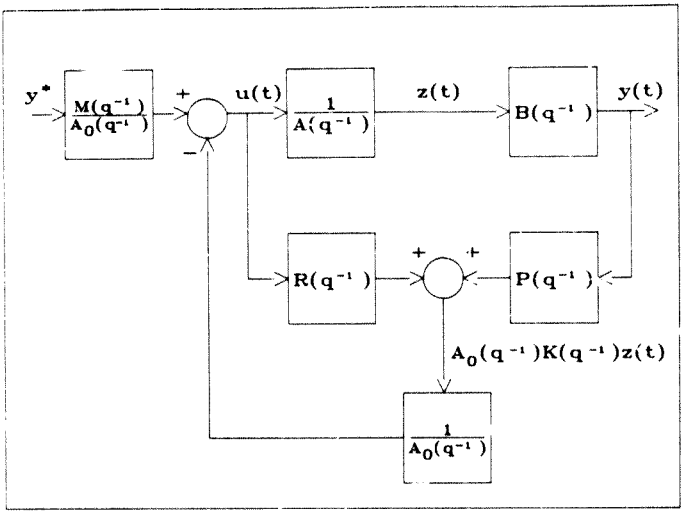

Figure 18. State Feedback Interpretation of Pole Assignment

signals, a suitable choice for observer poles may then be D' and S'. These are stable polynomial operators "near to" D and S respectively, as described in "Model structure" on page 41. Thus

$$A_s = A_s'D'S'$$

where A_s' are the remaining observer dynamics corresponding to the plant.

Caution must be observed in the case of $D = \delta$ and $D' = \delta + \varepsilon$ for some small ε. This D' corresponds to an "observer pole" near the origin. Such a mode would be unobservable in the system output under normal closed-loop control, thus it can be placed fairly flexibly. However, if made too slow,

the mode could be excited by an initial condition. This would then exhibit a long decay time before the integral action eliminated the error.

Typically choose $0{,}1\omega_p \leq \varepsilon \leq 0{,}2\omega_p$ where ω_p is the speed of the shifted plant poles.

The choice for A_0' (observer poles for the process itself) is made by considering the objective of state observation. The observer dynamics must be faster than those of the process, to ensure assymptotic tracking of the system states (O'Reilly 1983 : 201). Thus it seems reasonable to place these observer poles to correspond to ω_E, *the upper bandwidth limit* for model validity. In the PID case this corresponds to a single low-pass filter pole, as described in "Interpretation as a Digital PID Controller" on page 80. If this pole is fixed as for an industrial PID, this aspect of pole placement design is ignored.

6.5 SOFTWARE CONSIDERATIONS

6.5.1 SEQUENCING

For any adaptive real-time digital algorithm, an additional **computational delay** is introduced into the closed-loop system. This is due to the finite time requirements of the model estimation and control law calculations.

Thus the sequencing should be performed as shown in Figure 19 on page 93 (After Astrom and Wittenmark 1984 : 362). The critical part of the software cycle is in the model estimation and control law calculations, *prior* to the generation of a new control signal. This should be minimised through suitable refinement of the coding. The remaining updates such as the recalculation of gain matrices, filter output values, etc can then be performed in the less critical part of the cycle. Execution will be

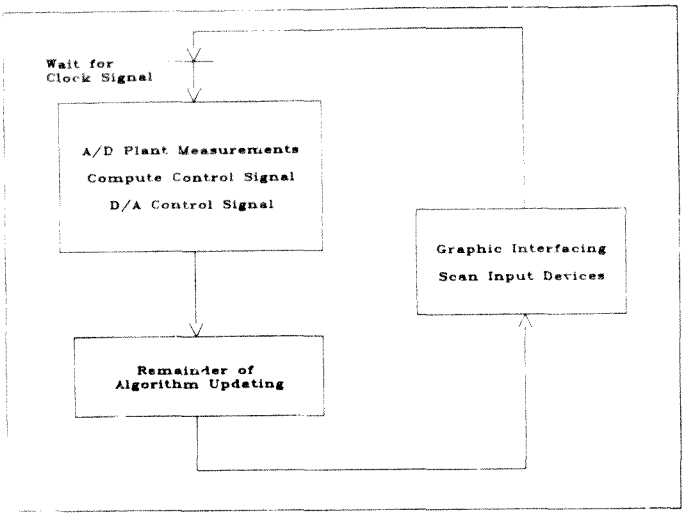

Figure 19. Digital Control Algorithm Sequence

halted after this until another clock signal is received to repeat the sampling process. The idle part of the loop can also be used to perform user interfacing, such as interactive graphic updates or scanning of input devices eg. keypads.

It is clear that if the sampling interval is long and the processing power of the digital system substantial, the computational delay will be negligible. However, for faster sampling and particularly when processes with short time constants are controlled, careful software sequencing and optimisation is desirable.

This sequencing can be seen in the software listing of Appendix E. The iteration is described in pseudo-code as

```
REPEAT
   IF SAMPLETIME THEN
   BEGIN
      READVALUE(Y)          {A/D plant output signal}
      ADAP1                 {estimate model and calculate control}
      WRITEVALUE(U)         {D/A control signal}
      ADAP2                 {perform rest of updating}
   END
{perform graphic interfacing, read keyboard}
< FOREVER >
```

6.5.2 ALGORITHM IMPLEMENTATION

The most complex calculations involve the RLS estimation algorithm. This is implemented using Bierman's UDU^T covariance factorization, and uses upper triangular and diagonal matrices. These are reformulated as **vectors**, to eliminate matrix operations for sparse matrices. In this way, optimization of both processing time and variable storage is facilitated.

6.6 CONCLUSION

This section has reviewed practical implementation requirements for the adaptive algorithm. The theory has been developed where necessary to support the arguments, and an attempt has been made to rationalise the implementation choices taken.

7.0 EXPERIMENTAL RESULTS

7.1 INTRODUCTION

This section presents a series of results produced both by simulation and during tests on a physical plant. The adaptive controller is evaluated with regard to

- Estimator robustness in terms of parameter behaviour and prediction error reduction.

- Closed-loop control performance as observed by setpoint tracking, overall stability and the nature of the control signals.

These measures of performance are examined under normal plant conditions, during which the signals are subject to both noise and bias. Robustness to plant variations is also investigated, with variable dead-time introduced artificially during closed-loop operation. Furthermore, the plant is subjected to an external disturbance, allowing observation of the closed-loop disturbance rejection of the adaptive system.

Simulation results are discussed for nonminimum-phase plant with significant variable dead-time, and with forward-path gain changes introduced during closed-loop control.

7.2 PHYSICAL PLANT CONFIGURATION

The process and controlling microcomputer is configured as in Figure 20 on page 96. The plant consists of a metal strip which can be heated at

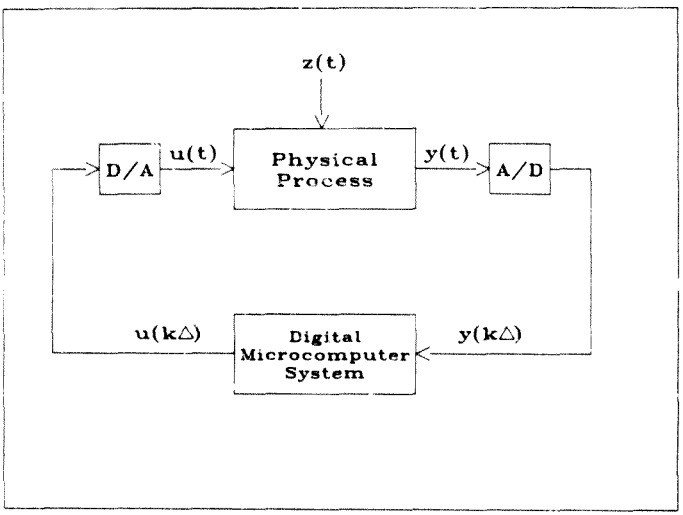

Figure 20. Physical Process and Digital Controller

one end. The control objective is to maintain the temperature at a point on the bar at some desired reference value. The temperature measurement y(t) is effected using a platinum resistance thermometer via a resistance measurement brige, and the control signal u(t) is the voltage applied to the heater. The process disturbance z(t) is introduced by switching on a fan to increase air flow over the bar at the point of measurement, and thus alter the process dynamics. This disturbance signal was not directly measured, and thus the feedforward control block has not been tested during these experiments. However, the **feedback** loop disturbance rejection was observed for a fairly severe external disturbance.

7.3 EXPERIMENT 1

The constant parameter values used by the estimator are as follows for this experiment:

ε = 0,0008	(high pass filter pole)
ω_E = 0,025	(low pass filter cutoff frequency)
P(0) = 10	(initial value for P-matrix)
λ = 0,97	(forgetting factor)
Δ = 20 seconds	(sampling interval)
σ = 0,98	(first-order pole in deadzone function m(k))
ε_0 = 0,0008	(deadzone threshold in m(k))
ε_1 = 0,0008	(coefficient of u(kΔ) in m(k))
ε_2 = 0,00008	(coefficient of y(kΔ) in m(k))
α = 0,95	(maximum value of estimator update gain a(k))
β = 3,5	(scaling factor to increase deadzone size)

These choices are all as motivated in "Implementation of the Robust Estimator" on page 63. However, a large value for α (α = 0,95 selected here) was used to facilitate high-gain updating, which resulted in a large deadzone scaling coefficient β = 3,5. This deadzone was found to be too large if the ε_i scaling coefficients were chosen as suggested in "Implementation of the Robust Estimator" on page 63, severely inhibiting parameter updating. Thus for this experiment much smaller values for ε_0, ε_1 and ε_2 were used to give an overall deadzone compatible with expected prediction error magnitudes.

The setpoint sequence was generated as shown in Figure 22 on page 102. Switchover from the "manual tuning phase" to automatic control was done at time TAUTO = 4000 seconds. Prior to this, a rapidly varying input signal was generated simply by setting the input equal to the setpoint. This is illustrated in Figure 24 on page 103. This is to ensure parameter convergence during initial identification, in accordance with persistent excitation requirements.

The control input signal during closed-loop operation shows large peaks at setpoint changes, decaying to a steady state value. There are also large overshoots before the settling to a constant value occurs, which is a consequence of the controller design. It is believed that the closed-loop objective may be a little too ambitious, namely in the choice of A^*. Consequently a fairly high-gain PID controller has resulted, giving overshoots on the control signal. The plant output y(t) (See Figure 23 on page 102) also shows small deviations initially on rising and falling setpoint response. However, the control signal stabilises rapidly and the process output responds smoothly to the setpoint.

A further point to note is the small variations about the steady value of the control signal between setpoint changes. This is evidence of the derivative control tarm in the PID, responding to small changes in process conditions such as external disturbances.

Further examining the system output y(t) in Figure 23 on page 102, it is clear that fairly good setpoint tracking is achieved. There is a significant d.c. level on the measured output signal, clearly visible in the initial value of y(t). This level corresponds to the ambient temperature during the experiment, and has to be treated as an unknown constant disturbance in the adaptive system. Consequently the A/D measurement **zero** cannot simply be adjusted to correspond to this value, since in practice it would vary slowly as the ambient temperature changed during long-term operation. This is a good test of the signal conditioning elements included in the estimator for nulling constant disturbances.

The closed-loop characteristic polynomial A^* is set using

$$\alpha_3 = 3.8E\text{-}02$$
$$\alpha_2 = 5.2E\text{-}04$$
$$\alpha_1 = 3.5E\text{-}06$$
$$\alpha_0 = 5.1E\text{-}09$$

in $\quad A^* = \delta^4 + \alpha_3\delta^3 + \alpha_2\delta^2 + \alpha_1\delta + \alpha_0$

which corresponds to an s-plane diagram as shown in Figure 21 on page 100.

The roots are $s_1 = -1{,}942E\text{-}03$
$s_2 = (-7{,}912 + j8{,}197)E\text{-}03$
$s_3 = (-7{,}912 - j8{,}197)E\text{-}03$
$s_4 = -2{,}023E\text{-}02$

s_1 is the "observer pole" near the origin corresponding to the deterministic mode $D = \delta$ in the controller denominator polynomial. s_2 and s_3 are the "shifted process poles" which govern the dominant closed loop performance. These correspond to a damped second-order response, with damping factor

$$\xi = \cos(45^\circ) = 0{,}7$$

s_4 is the remaining "observer pole" for the process itself, and is placed fairly close to ω_E. These choices are made in accordance with the rationale given in "Choice of Closed-loop Characteristic Polynomial" on page 87. The closed loop objective detailed here was used for all experiments discussed in this section.

Examining Figure 32 on page 107, Figure 33 on page 107, Figure 34 on page 108 and Figure 35 on page 108, it is clear that very rapid parameter convergence occurred initially. In fact, almost no further parameter activity is apparent after the switchover from manual tuning to automatic control.

This is further seen in Figure 26 on page 104, which details the function a(k) against time. Since a(k) obeys the relationship of 4.15, the maximum possible magnitude for it is given by α. In this case a(k) will always be less than unity. Furthermore, a(k) only has a nonzero value when parameter updating occurs. Thus it is clear that fairly extensive activity occurred initially, as seen by the large a(k) values. This implies high-gain parameter updating. During automatic control a few updates are per-

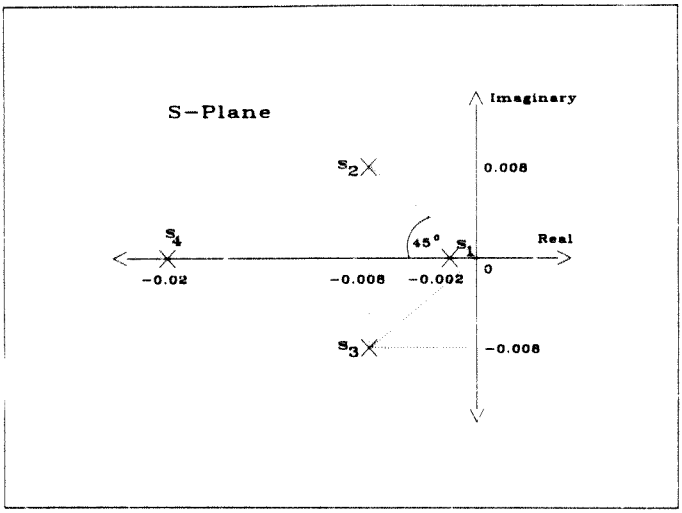

Figure 21. Pole Locations of Closed Loop Polynomial

formed, corresponding to setpoint changes and consequent changes in input signal. This is in accordance with persistent excitation considerations.

The prediction error signal e(t), shown in Figure 28 on page 105, further illuminates the parameter updating process. When this error is greater than the deadzone, the update gain a(k) is nonzero and a parameter change occurs. Larger prediction errors are present during initial identification, with the error exhibiting smaller peaks during setpoint changes. The deadzone size itself is detailed in Figure 25 on page 103. The magnitude function m(k) is as defined in 4.13, and gives a first-order filtered function of the process input and output. The input is weighted far more strongly than the output, since ϵ_1 is much larger than ϵ_2. The ... id value ϵ_0 is also clearly visible. Thus the deadzone size $\delta m(k)$ ...es proportionally to the input signal primarily, with a larger deadzone initially when the input signal is large.

Combining the deadzone and the absolute prediction error |e(k)| (See Figure 27 on page 104), the updating of the parameters and the behaviour of a(k) (Figure 26 on page 104) becomes obvious.

The controller design parameters are given in Figure 29 on page 105, Figure 30 on page 106 and Figure 31 on page 106. These values are only calculated after switchover to automatic control. The PID coefficients are fairly constant throughout the experiment, indicating stable closed-loop operation and robust controller design. Some minor control parameter changes occur, corresponding to estimated model updates after a set point change (as discussed earlier).

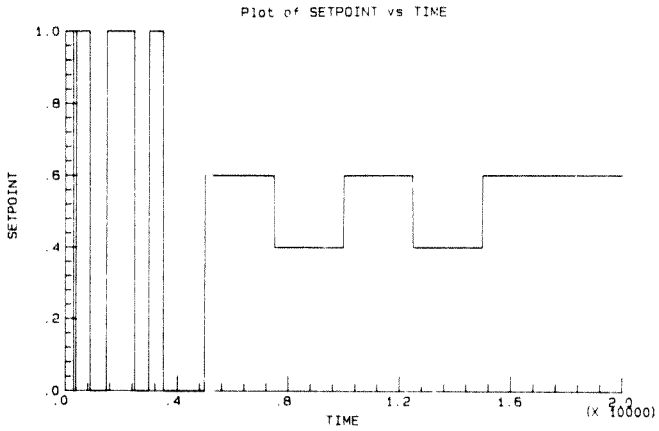

Figure 22. Setpoint Sequence for Experiment 1

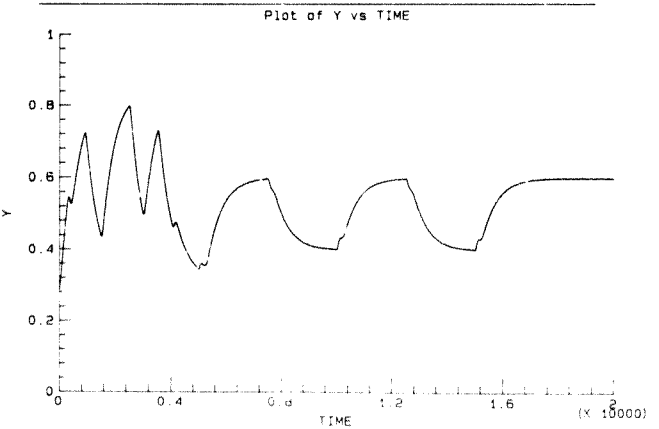

Figure 23. System Output y(t) for Experiment 1

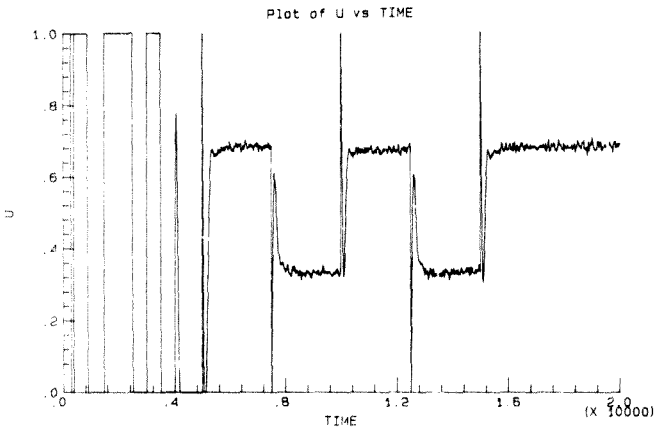

Figure 24. Control Input u(t) for Experiment 1

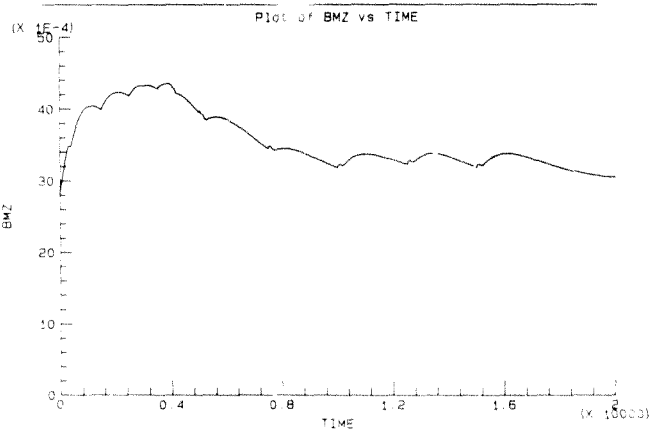

Figure 25. Deadzone Size for Experiment 1

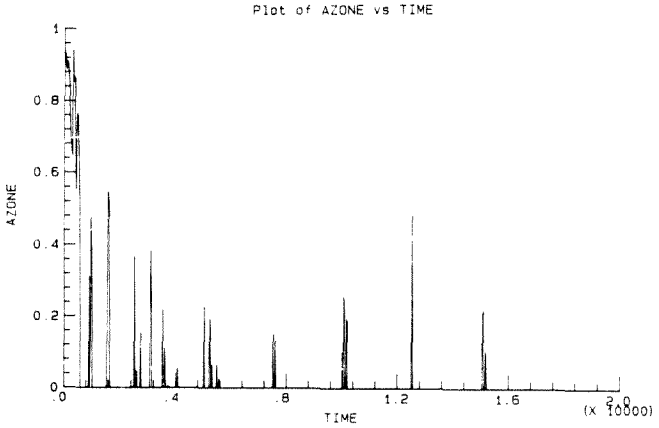

Figure 26. Estimation Gain Sequence a(k) for Experiment 1

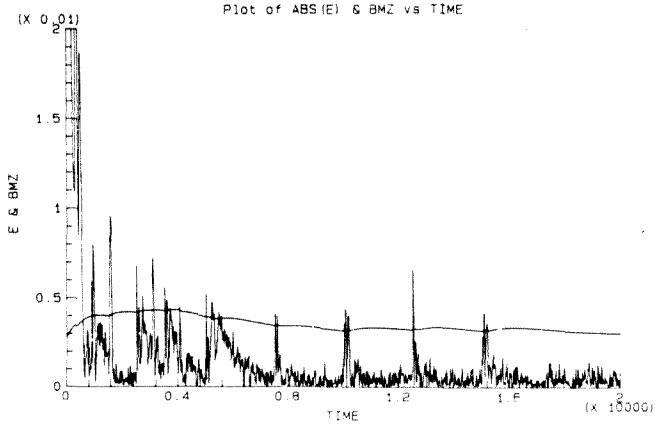

Figure 27. Prediction Error and Deadzone Size for Experiment 1

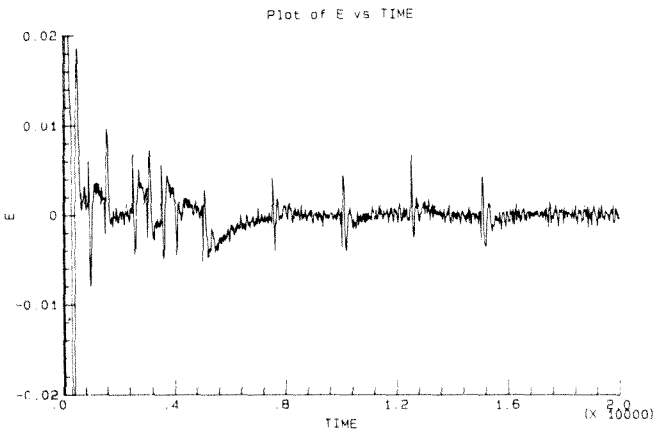

Figure 28. Prediction Error e(t) for Experiment 1

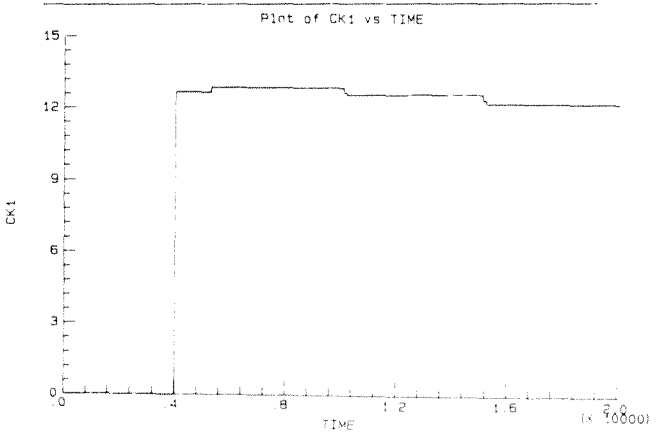

Figure 29. PID Gain Constant CK1 for Experiment 1

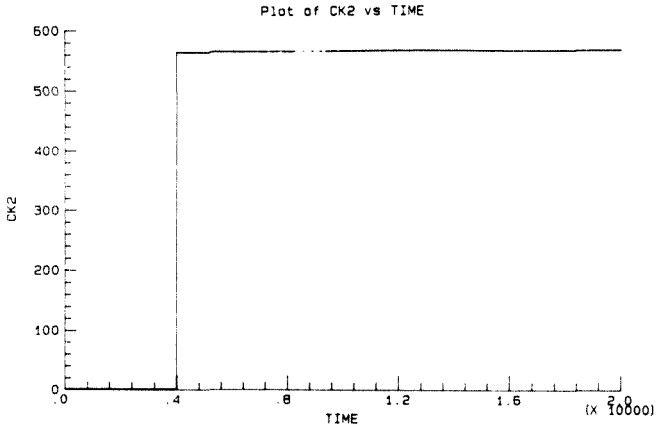

Figure 30. PID Integral Time Constant CK2 for Experiment 1

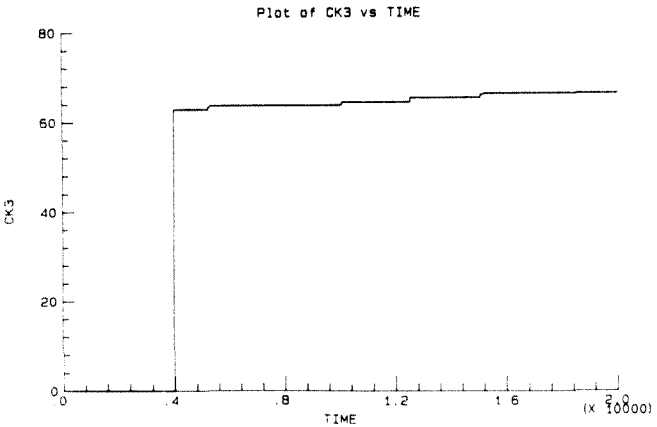

Figure 31. PID Derivative Time Constant CK3 for Experiment 1

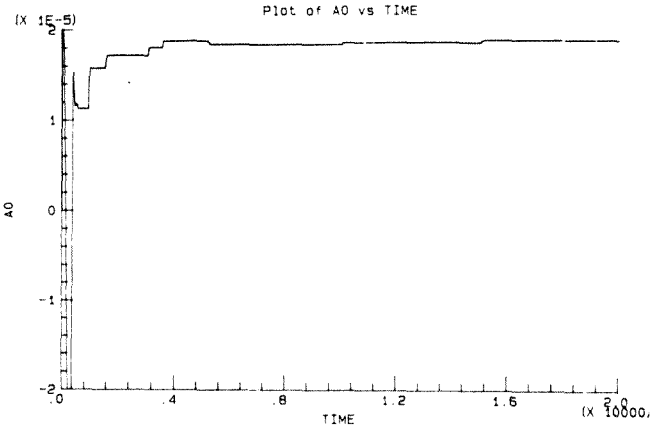

Figure 32. Estimated Parameter a0 for Experiment 1

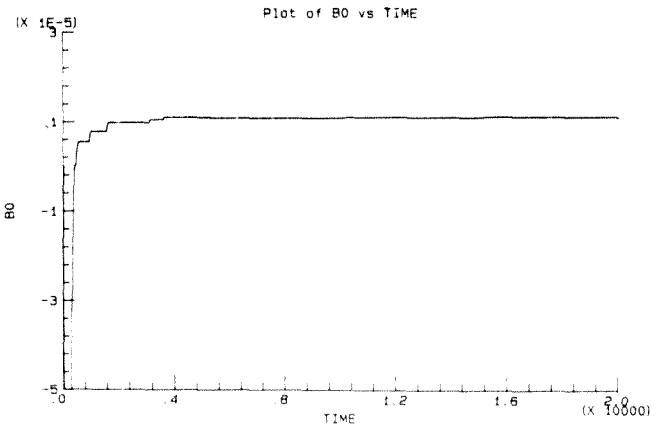

Figure 33. Estimated Parameter b0 for Experiment 1

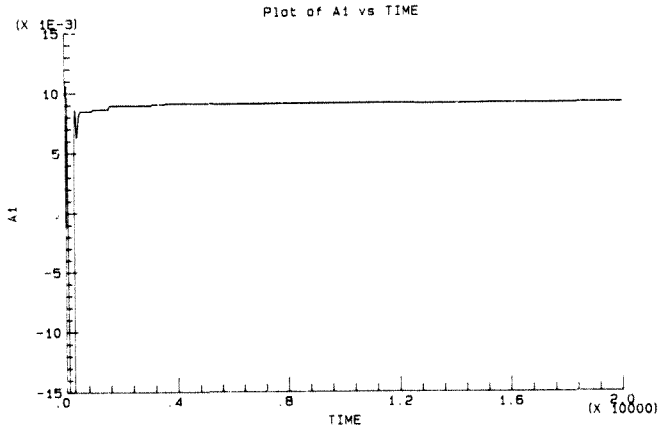

Figure 34. Estimated Parameter $a1$ for Experiment 1

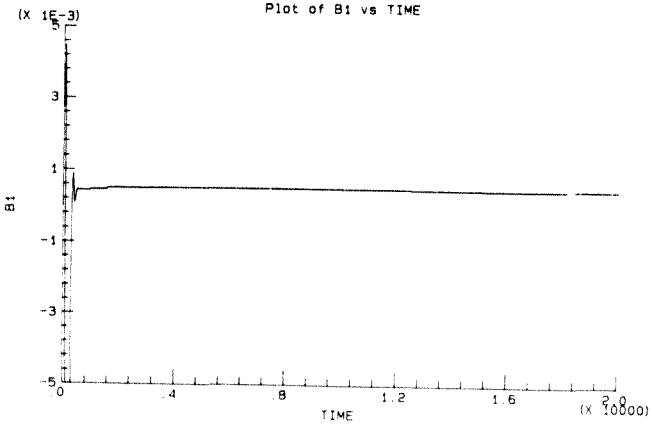

Figure 35. Estimated Parameter b1 for Experiment 1

7.4 EXPERIMENT 2

The constant parameters used by the estimator are:

$\varepsilon = 0{,}0008$	(high pass filter pole)
$\omega_E = 0{,}025$	(low pass filter cutoff frequency)
$P(0) = 10$	(initial value for P-matrix)
$\lambda = 0{,}97$	(forgetting factor)
$\Delta = 20$ seconds	(sampling interval)
$\sigma = 0{,}98$	(first-order pole in deadzone function m(k))
$\varepsilon_0 = 0{,}002$	(deadzone threshold in m(k))
$\varepsilon_1 = 0{,}002$	(coefficient of u(kΔ) in m(k))
$\varepsilon_2 = 0{,}0002$	(coefficient of y(kΔ) in m(k))
$\alpha = 0{,}5$	(maximum value of estimator update gain a(k))
$\beta = 1{,}4$	(scaling factor to increase deadzone size)

Effectively the only differences from Experiment 1 are the choices of α, β and the ε_i coefficients. The same setpoint sequence was used as in Experiment 1, shown in Figure 36 on page 112. Also, the same manual tuning phase was used as in Experiment 1 with switchover to automatic control at TAUTO = 4000 seconds.

The process output y(t) is shown in Figure 37 on page 112. Very similar performance to that of the previous experiment is observed, with good setpoint tracking according to the closed-loop polynomial $\hat{A}^*$. Examining the control input signal (See Figure 38 on page 113), however, much higher amplitude variations are present during the controlled setpoint response. This is primarily due to poor tuning of the controller, and is clarified later.

The process parameter estimates are shown in Figure 46 on page 117, Figure 47 on page 117, Figure 48 on page 118, and Figure 49 on page 118. It is clear that the estimates converge much more slowly than for the previous experiment. This is apparent because the estimation gain constant α is set to a much smaller value than previously ($\alpha = 0{,}5$). However,

this has allowed the use of a much smaller deadzone scaling coeffient (ß = 1,4) in accordance with convergence requirements. Furthermore, the reduced ß-value enabled the choice of much more reasonable ε_0, ε_1 and ε_2 coefficients than in Experiment 1. These coefficients could be selected on engineering grounds to correspond to expected noise levels and signal ranges, and to overbound modelling error. Clearly, the price paid for these parameter choices is slower convergence of the estimator, and a correspondingly oscillatory control input signal u(t) while the controller is poorly tuned. However, it is believed that these choices would be more robust in "difficult" process environments or for "poorly behaved" plants.

Figure 40 on page 114 illustrates the behaviour of the function a(k). As before, a nonzero value of a(k) indicates that parameter updating is occurring. There is far more activity during the manual identification period than in Experiment 1, and more sustained parameter updating throughout the remainder of the experiment. This is borne out by the continued parameter changes discussed earlier. The peak magnitude of the function a(k) is now less than 0,5 as governed by the smaller α-value.

The deadzone magnitude function ßm(k) (Figure 39 on page 113) is very similar to that in Experiment 1, since the input signals are similar for the two experiments. Note that rapid fluctuations in u(t) do not affect ßm(k), since it is a low-pass filtered function of the process input and output.

The prediction error e(t) is shown in Figure 42 on page 115. As expected, the error is substantially larger early in the identification stage, and exhibits rapid fluctuations corresponding to the variations in u(t). The combination of e(t) and ßm(k) is given in Figure 41 on page 114 for completeness, to clarify the behaviour of a(k).

The PID coefficients shown in Figure 43 on page 115, Figure 44 on page 116 and Figure 45 on page 116 also show more variation than in the preceeding experiment. The calculated gain value and derivative time are in fact very poor at switchover to automatic control, to the extent of

applying **positive feedback** to the plant. However, the resultant incorrect control input signal facilitates rapid parameter convergence just after switchover, and more reasonable controller values are obtained. The corresponding step changes in estimated parameters are clearly visible after time T = 4000 seconds. Once again, a high PID gain constant value is obtained as well as a high derivative time constant. This is believed to be related to the choice of closed-loop polynomial A^*, as mentioned previously.

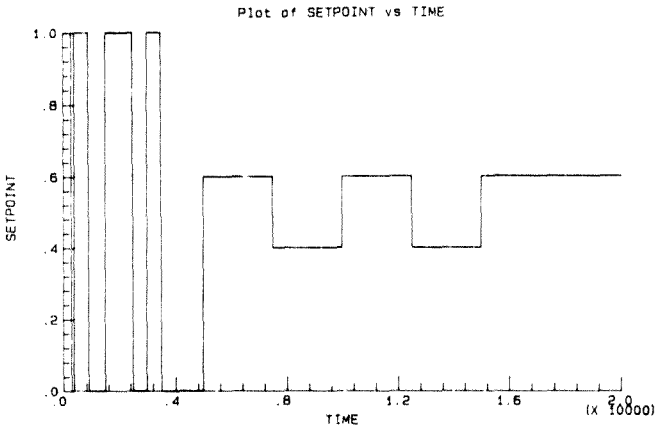

Figure 36. Setpoint Sequence for Experiment 2

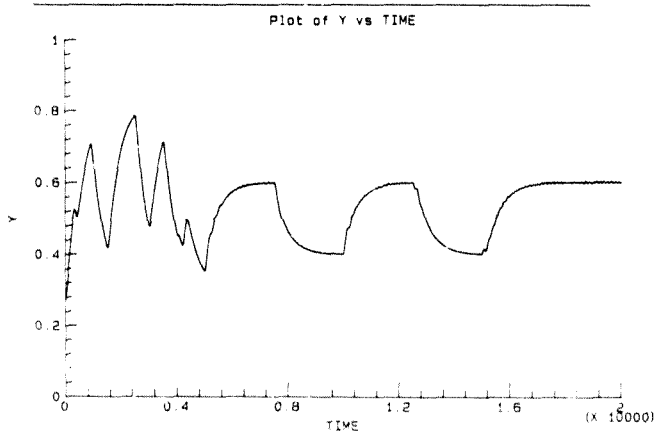

Figure 37. System Output y(t) for Experiment 2

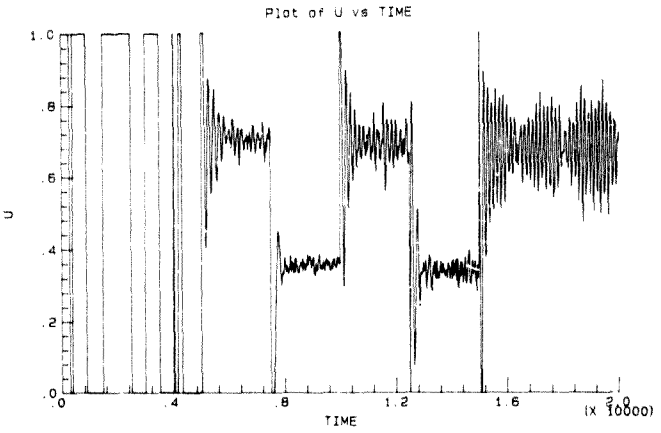

Figure 38. Control Input u(t) for Experiment 2

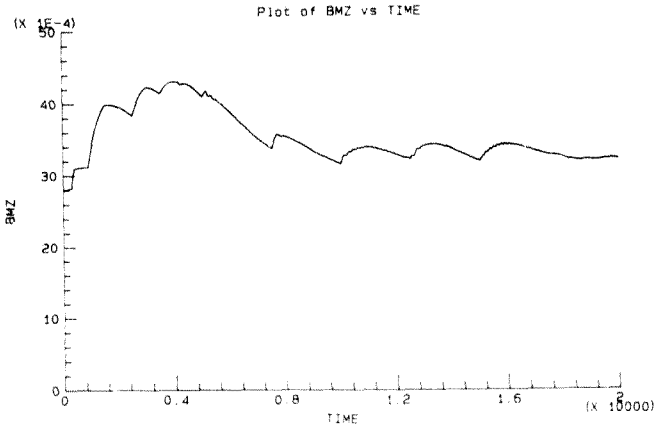

Figure 39. Deadzone Size for Experiment 2

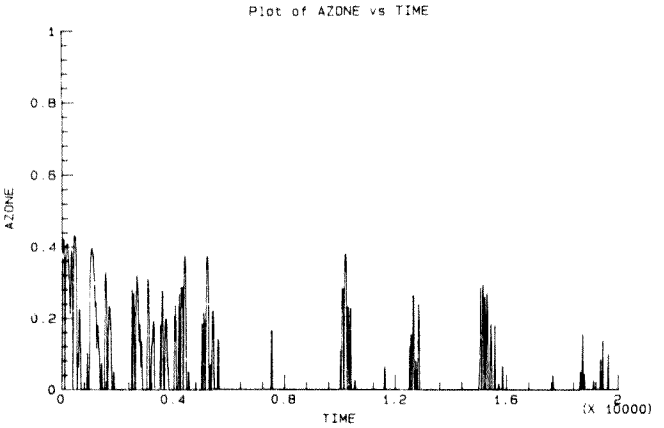

Figure 40. Estimation Gain Sequence a(k) for Experiment 2

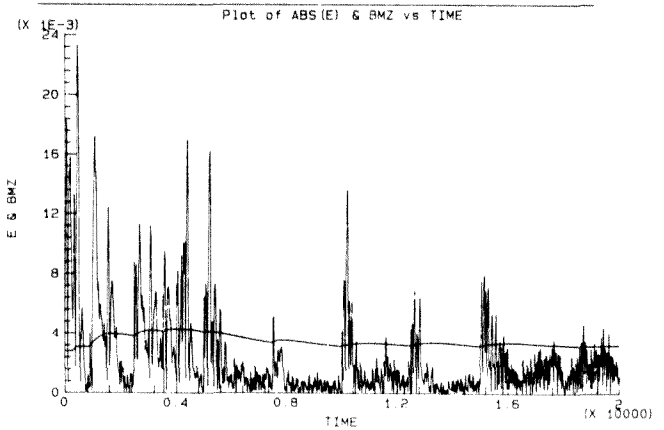

Figure 41. Prediction Error and Deadzone Size for Experiment 2

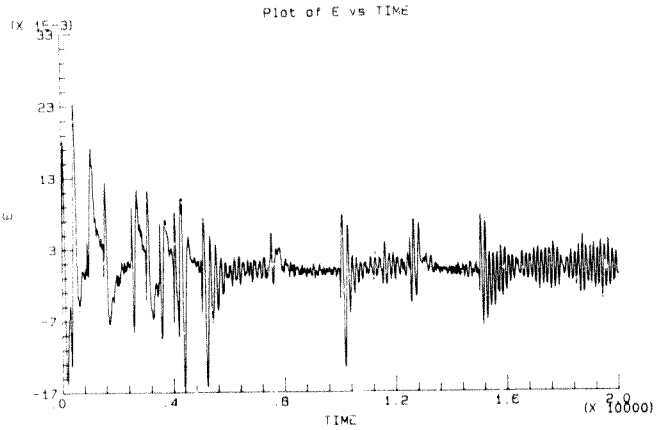

Figure 42. Prediction Error e(t) for Experiment 2

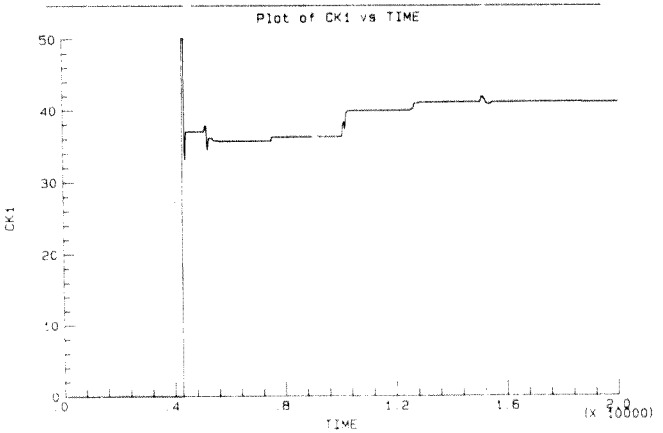

Figure 43. PID Gain Constant CK1 for Experiment 2

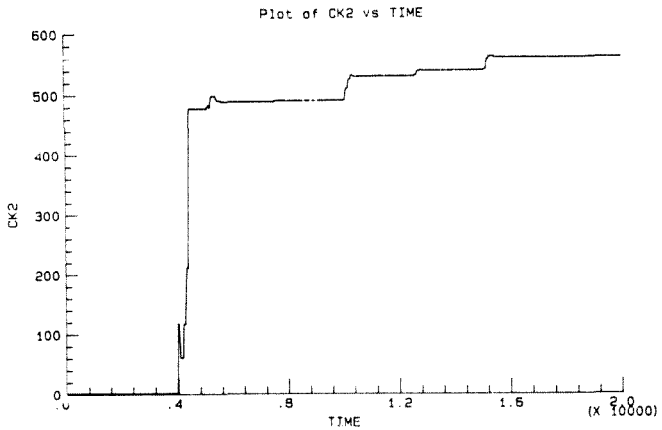

Figure 44. PID Integral Time Constant CK2 for Experiment 2

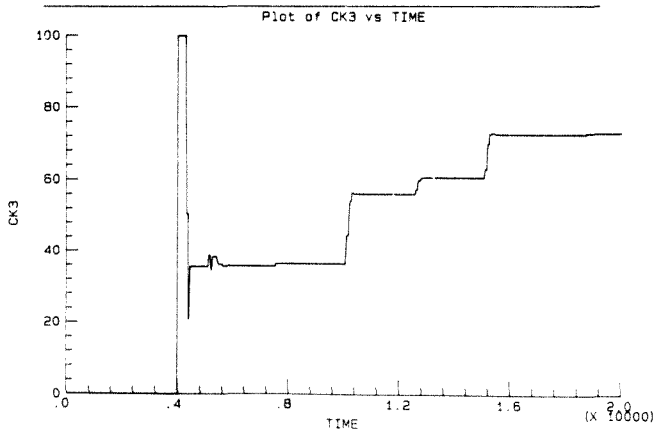

Figure 45. PID Derivative Time Constant CK3 for Experiment 2

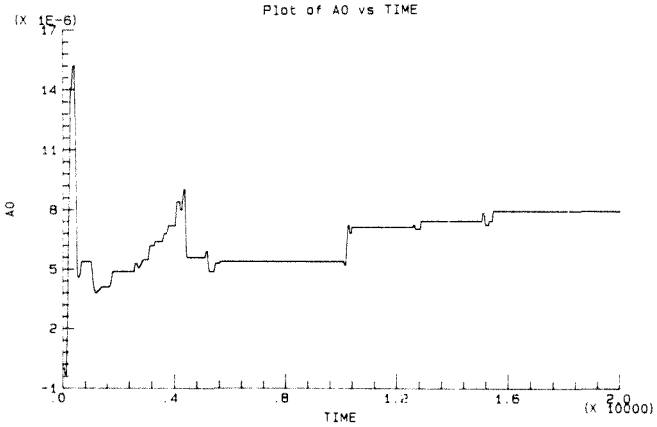

Figure 46. Estimated Parameter a0 for Experiment 2

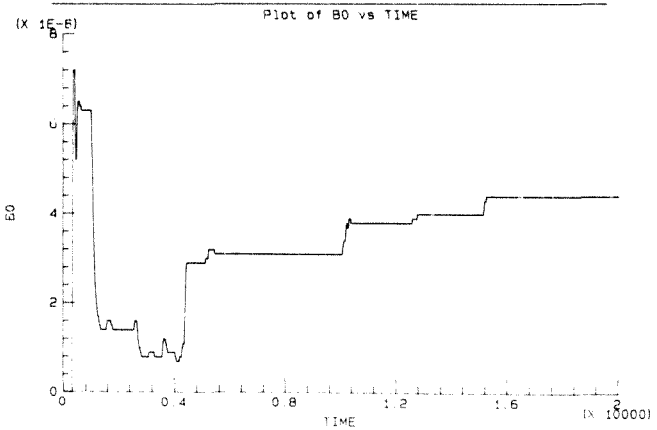

Figure 47. Estimated Parameter b0 for Experiment 2

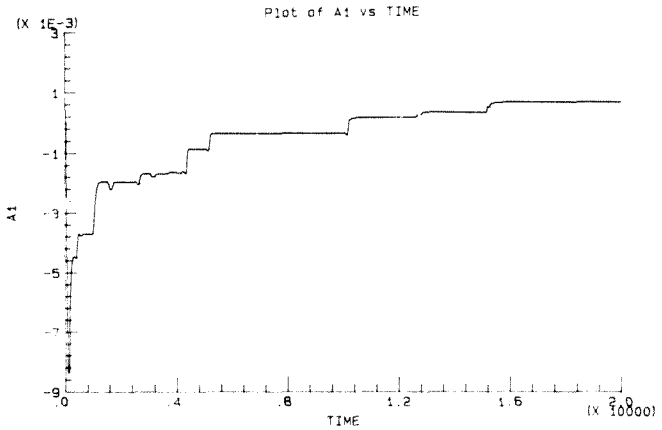

Figure 48. Estimated Parameter a1 for Experiment 2

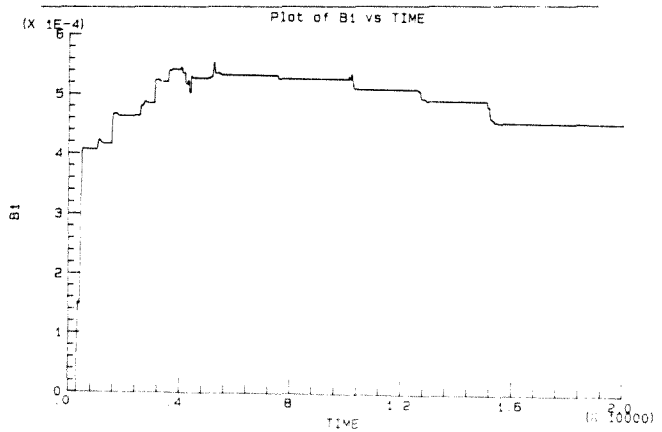

Figure 49. Estimated Parameter b1 for Experiment 2

7.5 EXPERIMENT 3

This is a short experiment illustrating the persistent excitation requirements of the estimator. The constant estimator parameters were the same as those used in Experiment 1, hence fairly rapid convergence is expected.

During this experiment, the setpoint sequence was generated as in Figure 50 on page 121, with switchover to automatic control at TAUTO = 1200 seconds. Thus a far shorter time period was available for parameter convergence before the loop was closed. The system output y(t) is shown in Figure 51 on page 121. It is clear that very poor setpoint response is obtained initially, but after two setpoint changes under automatic control the tracking is fairly good.

Examining the PID constant values in Figure 53 on page 122, Figure 54 on page 123 and Figure 55 on page 123 it is clear that there are fairly large controller coefficient changes after switchover. The effect of this is observed in the control input u(t) (Figure 52 on page 122). The control signal is oscillatory over its full range for a certain time period (3000 < T < 5500), during which the controller is poorly tuned. However, this "richness" of control signal and hence process output response enables rapid parameter convergence and good redesign of the controller.

This experiment thus illustrates the effect of switchover to automatic control before adequate system identification has occurred. The number of samples given for "pretuning" in this case was

$$N = 1200/20 = 60 \text{ samples}$$

for sampling interval, $\Delta = 20$ seconds.

However, as seen in the earlier experiments, it is more desirable to have a longer "pretuning" period with a rich input signal, to ensure a "good" controller on switchover. Nevertheless, the overall robustness of the

adaptive system is effectively demonstrated in this experiment, since process identification and controller tuning was performed successfully in closed loop. From a theoretical viewpoint this is true because estimation is independant of the controller design.

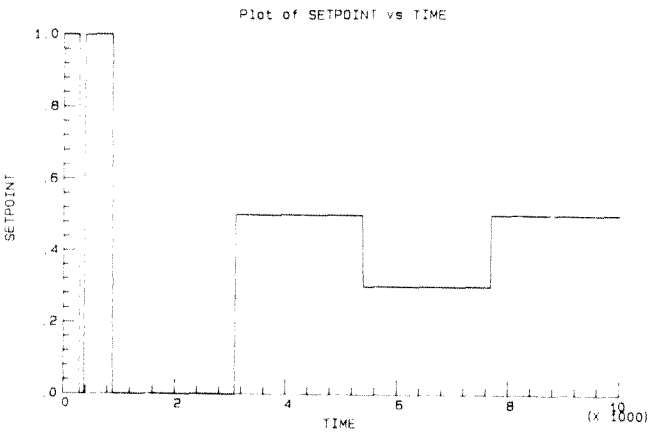

Figure 50. Setpoint Sequence for Experiment 3

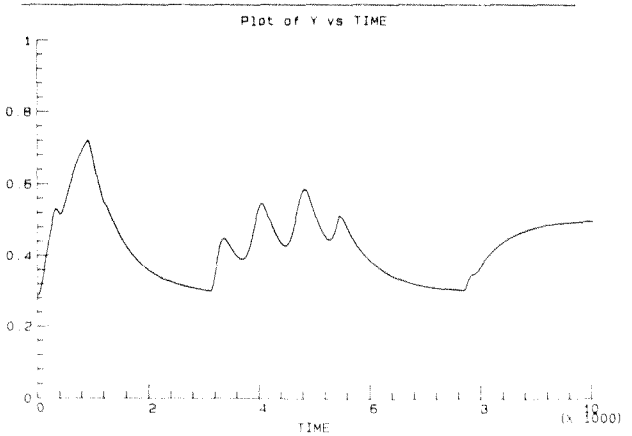

Figure 51. System Output y(t) for Experiment 3

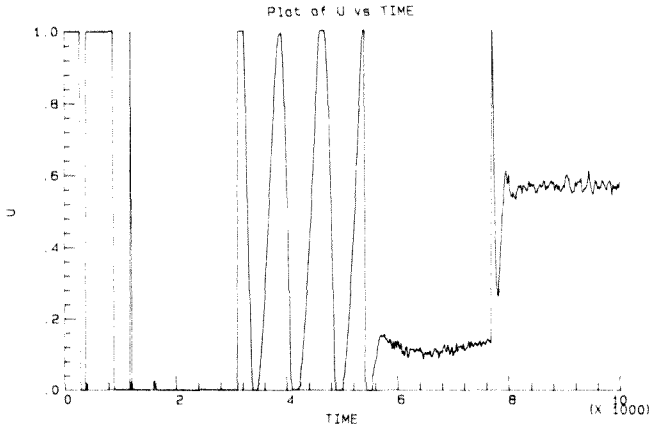

Figure 52. Control Input u(t) for Experiment 3

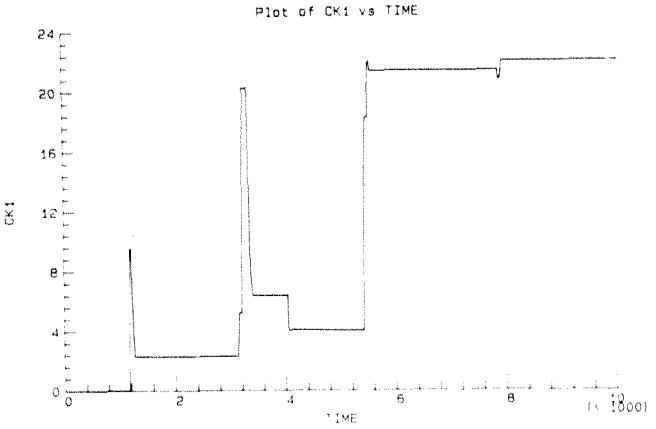

Figure 53. PID Gain Constant CK1 for Experiment 3

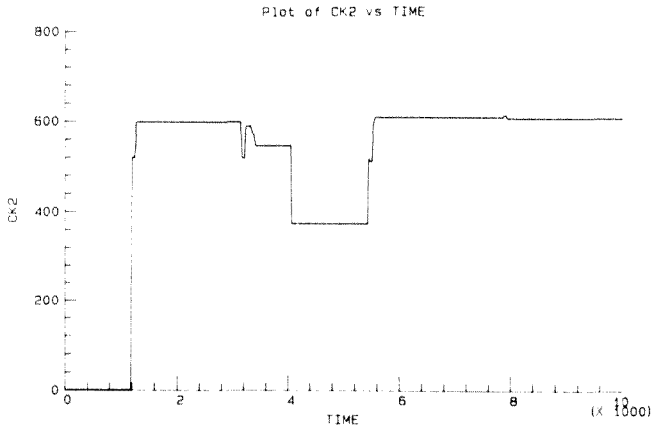

Figure 54. PID Integral Time Constant CK2 for Experiment 3

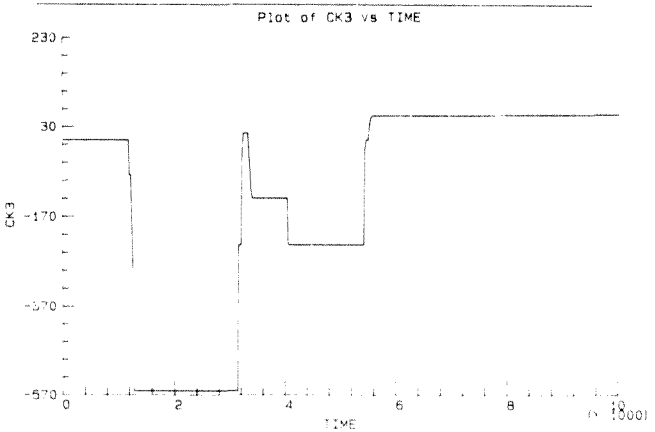

Figure 55. PID Derivative Time Constant CK3 for Experiment 3

7.6 EXPERIMENT 4

This experiment was performed over a longer time period, and investigates the ability of the adaptive controller to readapt to processes with variable time delay.

The estimator constant parameters used were as in Experiment 2, since these yield a reasonable tradeoff between robustness and convergence rate as discussed earlier. The setpoint sequence was generated as in Figure 56 on page 126, and a time delay t_d = 60 seconds was artificially introduced into the system at T = 17500 seconds through the software sampling in the microcomputer.

The control input signal is shown in Figure 58 on page 127, and takes a very similar form to that of earlier experiments.

The introduction of the time delay at T = 17500 seconds is manifested as a peak in the prediction error signal e(t) in Figure 62 on page 129. A corresponding large peak in the parameter update gain a(k) at this time is shown in Figure 60 on page 128. For clarity the prediction error e(t) and deadzone magnitude Bm(k) are combined again in Figure 61 on page 128. This last shows that the large prediction error exceeds the deadzone substantially when the time delay is introduced, resulting in a large parameter update gain a(k).

The deadzone function Bm(k) is given on its own in Figure 59 on page 127, and behaves much as discussed previously.

The estimated parameters a_0, b_0, a_1 and b_1 are shown in Figure 66 on page 131, Figure 67 on page 131, Figure 68 on page 132 and Figure 69 on page 132 respectively. A large change in all four parameter estimates is apparent after the introduction of the time delay. It is interesting to note that the b_1 coefficient which determines the position of the process model zero becomes negative. This is as expected, since the estimator approxi-

mates the delay time as in the well-known first-order Pade approximation to a delay term (Ralston 1965 : 278):

$$e^{-\theta s} = \frac{1 + \theta s/2}{1 - \theta s/2} \qquad (7.1)$$

which has an unstable zero, ie. the process becomes nonminumum phase.

The PID controller coefficients K_c, T_i and T_d are detailed in Figure 63 on page 129, Figure 64 on page 130 and Figure 65 on page 130 respectively. It is seen that while very little change occurs in the integral and derivative constants after the delay has been introduced, the gain changes substantially. This is due to the PID approximation used to derive the coefficients from the pole placement procedure. Although this controller design is difficult to interpret in terms of changing pole and zero positions, the overall closed-loop response y(t) (See Figure 57 on page 126) is not dramatically affected by the introduction of the delay.

These results demonstrate satisfactory performance of the adaptive controller for processes with variable dead-time.

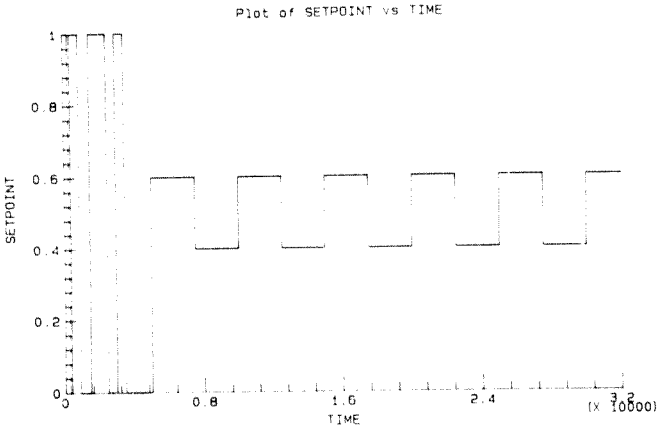

Figure 56. Setpoint Sequence for Experiment 4

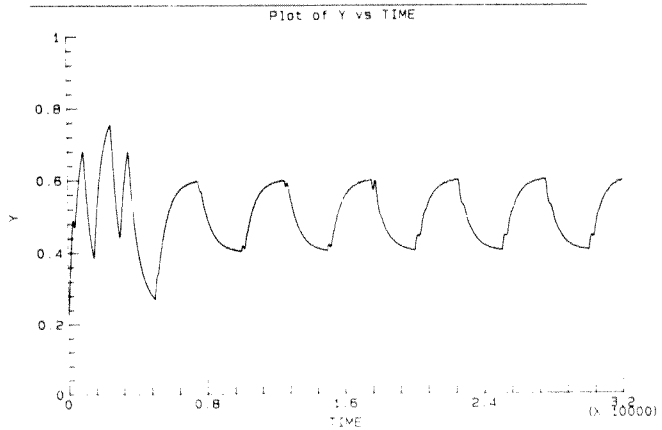

Figure 57. System Output y(t) for Experiment 4

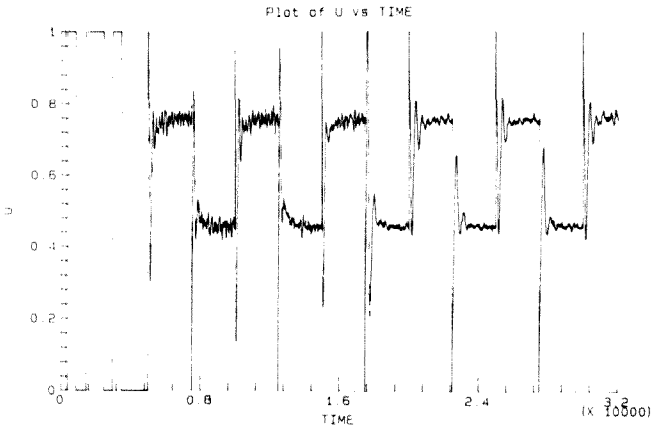

Figure 58. Control Input u(t) for Experiment 4

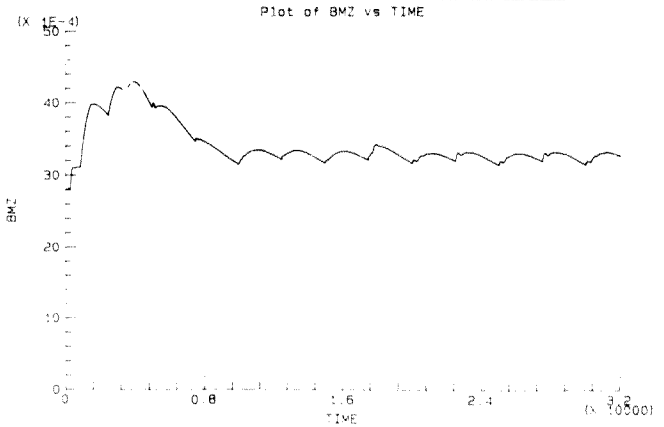

Figure 59. Deadzone Size for Experiment 4

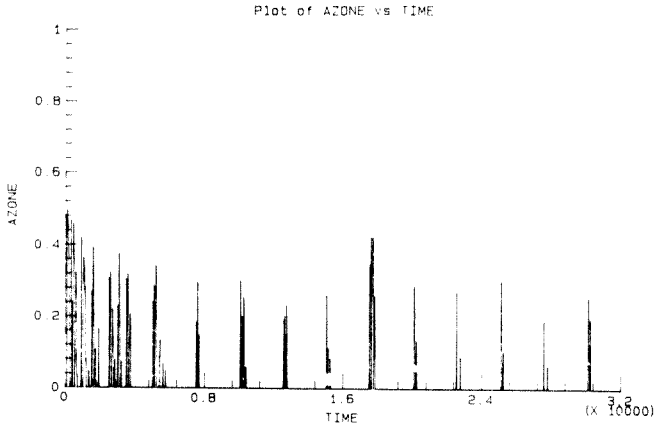

Figure 60. Estimation Gain Sequence a(k) for Experiment 4

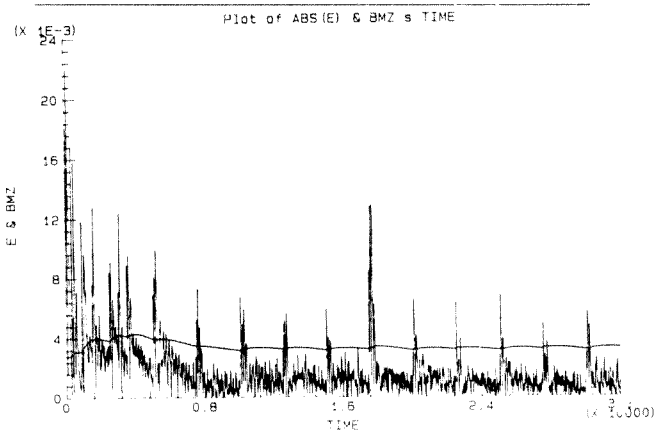

Figure 61. Prediction Error and Deadzone Size for Experiment 4

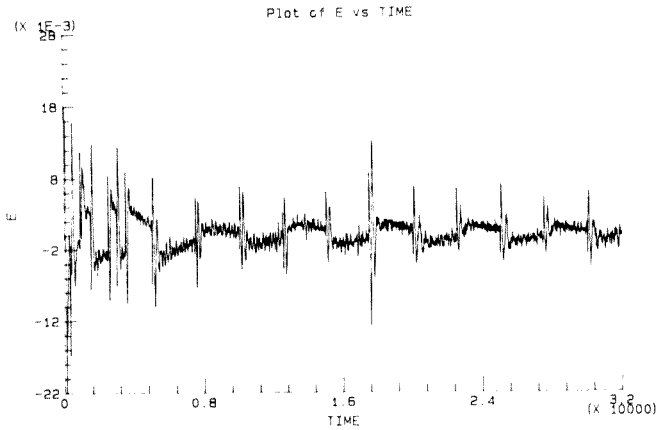

Figure 62. Prediction Error e(t) for Experiment 4

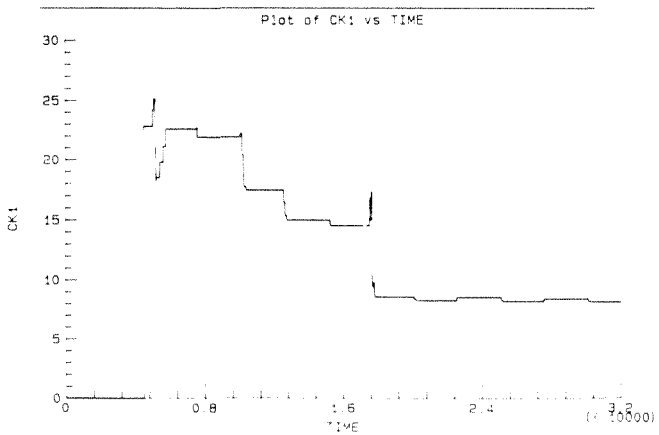

Figure 63. PID Gain Constant CK1 for Experiment 4

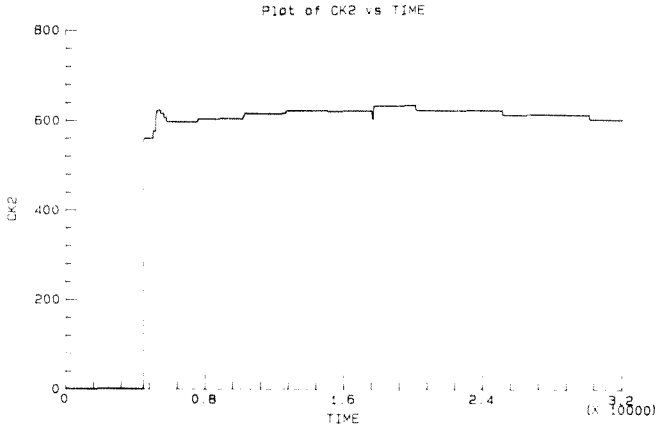

Figure 64. PID Integral Time Constant CK2 for Experiment 4

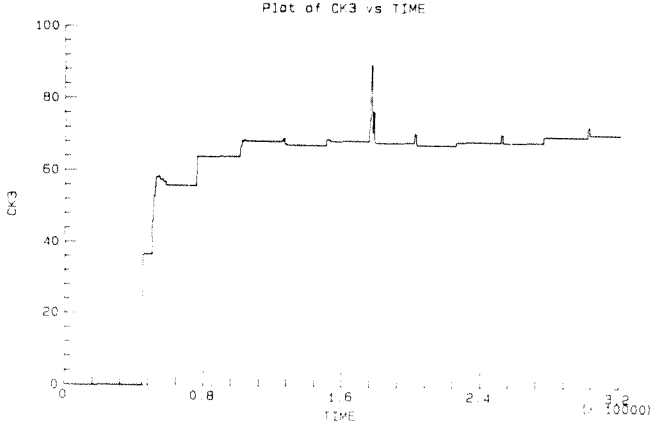

Figure 65. PID Derivative Time Constant CK3 for Experiment 4

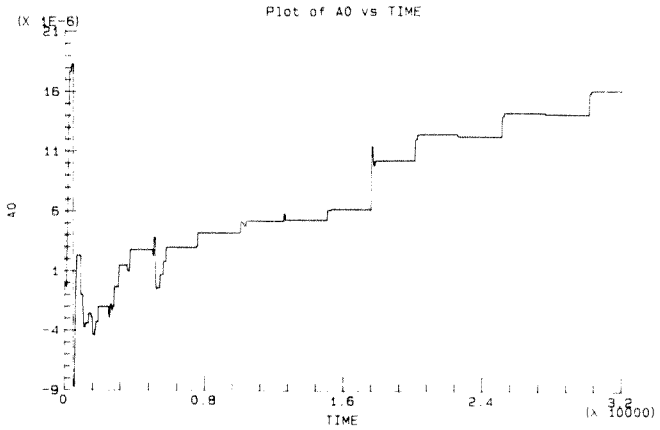

Figure 66. Estimated Parameter a0 for Experiment 4

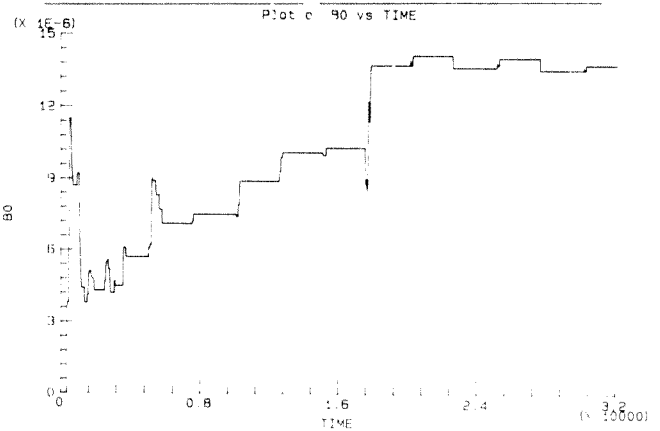

Figure 67. Estimated Parameter b0 for Experiment 4

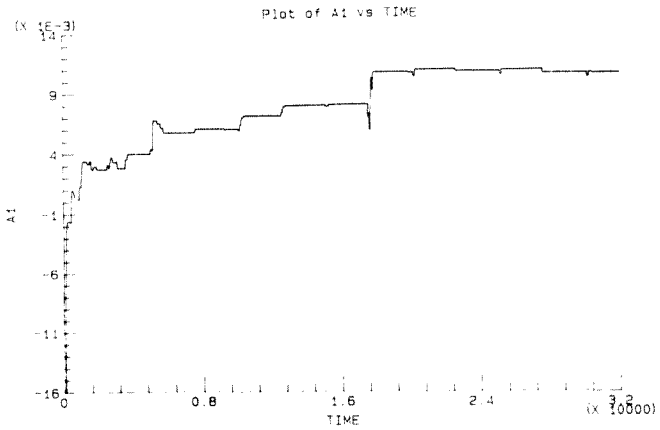

Figure 68. Estimated Parameter a1 for Experiment 4

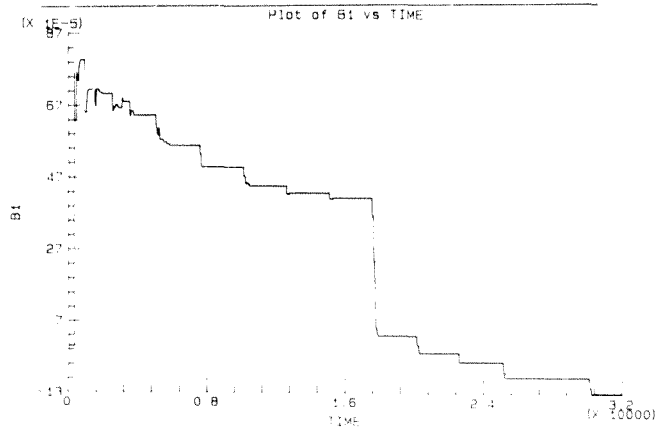

Figure 69. Estimated Parameter b1 for Experiment 4

7.7 EXPERIMENT 5

This experiment was performed to invesstigate the **feedback** system's robustness to external process disturbances. No disturbance measurement was made, and hence **feedforward** control testing was not performed.

Once again, the estimator constants were set as in Experiments 2 and 4 to ensure a robust system. The setpoint signal used is shown in Figure 70 on page 136. This sequence was generated to examine the **regulatory** nature of the adaptive system, since a constant setpoint was applied after the pretuning phase and a few setpoint step changes under automatic control. These dynamic setpoint changes ensured good convergence and a fair controller design before disturbance testing was done. The disturbance was applied at time T=20000 seconds, by switching on a fan to increase airflow over the heated bar at the point of temperature measurement.

The effect of the disturbance is clearly visible in the system output y(t) (Figure 71 on page 136). However, the feedback system rapidly compensates for the disturbance and fairly good regulatory control is still achieved. Figure 72 on page 137 details the control signal u(t) during the experiment, which shows fairly similar behaviour to that of earlier experiments prior to the introduction of the disturbance (cf. Experiment 2).

However, after the fan is switched on large high frequency variations are present in the control signal. These are explained by examining the controller design, where the controller coefficients are given in Figure 77 on page 139, Figure 78 on page 140 and Figure 79 on page 140. The PID gain constant CK1 is fairly large at thetime of switching on the disturbance. This accounts for the large variations in control signal duringthe setpoint tracking response (for time T < 20000 seconds), as well as the even larger amplitude variations after the disturbance was introduced. However, the "richness" of signals supplied to the estimator facilitated rapid redesign of the controller as a better process model was found.

The robustness of the controller design is apparent here, since the gain of the controller CK1 is reduced in response to the disturbance and consequently the amplitude of the variations on the control signal are decreasing. At the same time, the derivative time constant CK3 increases, allowing the controller to respond rapidly to higher frequency disturbance variations. Similarly, the integral time CK2 approaches a value nearer that of earlier experiments.

Once again, the deadzone size $\beta m(k)$ is given in Figure 73 on page 137, and shows the low-pass filtered characteristic function dominated by u(t). As before, the high-frequency variations in control input do not substantially affect the deadzone, since it is bandlimited. This is particularly desirable in this case, since the high-frequency "noise" resulting from the disturbance should not be passed through to the estimator or deadzone.

The error signal e(t) *is given in Figure 76 on page* 139, and shows a very similar pattern to earlier experiments. The disturbance results in a fairly large error signal, facilitating substantial parameter updating. This is better seen in the combination of $|e(k)|$ and $\beta m(k)$ in figref refid=e5ebmz., as well as in the behaviour of the parameter update gain function a(k) (Figure 74 on page 138).

The parameters themselves behave much as in Experiment 2 prior to the application of the disturbance (*See Figure 80 on page* 141, *Figure* 81 *on* page 141, Figure 82 on page 142 and Figure 83 on page 142). They exhibit fairly slow convergence, with large changes at each setpoint step change. Once the disturbance is introduced, further parameter changes occur despite the setpoint remaining constant. As discussed earlier, this is a result of the rich control input signal in response to the disturbance. Notably the b_0 and b_1 coefficients are subject to large changes. The increase in b_0 implies a nett increase in forward-path plant gain, seen in the ratio of (b_0/a_0) at steady state. The corresponding reduction in PID controller gain thus seems correct, and is evidenced in the reduction of variations for both the control input signal u(t) and process output y(t). The decrease in b_1 implies a faster plant model zero, and is manifested

as an increase in both the derivative and integral time constants of the controller.

This experiment indicates that external disturbances in fact improve estimator performance in the adaptive system. Robustness and disturbance rejection are also clearly illustrated.

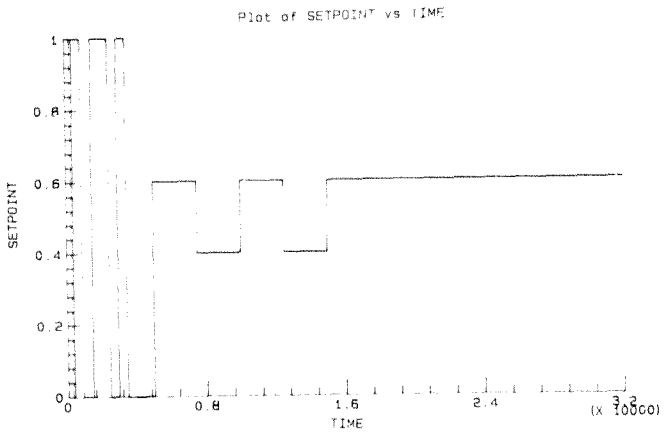

Figure 70. Setpoint Sequence for Experiment 5

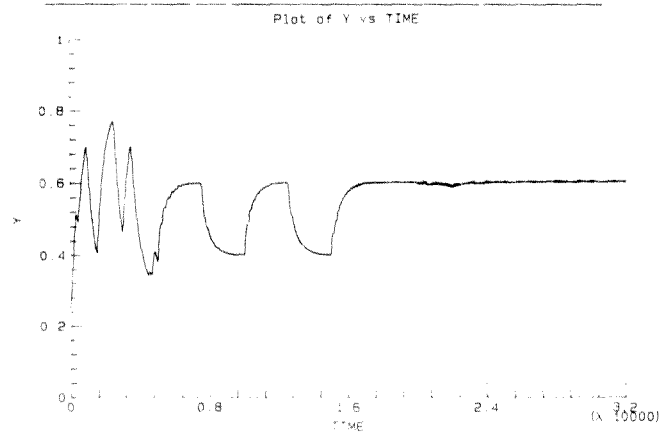

Figure 71. System Output y(t) for Experiment 5

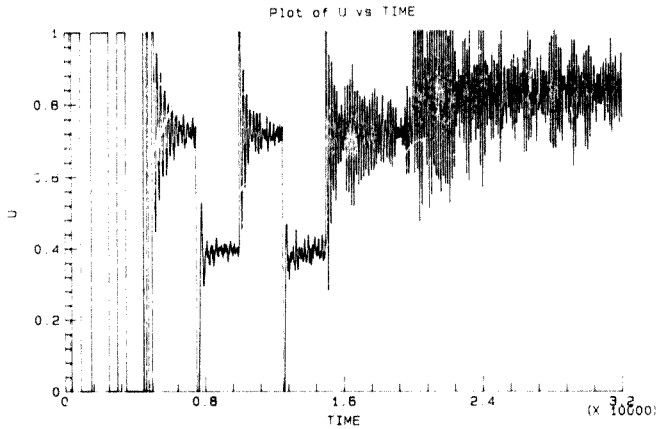

Figure 72. Control Input u(t) for Experiment 5

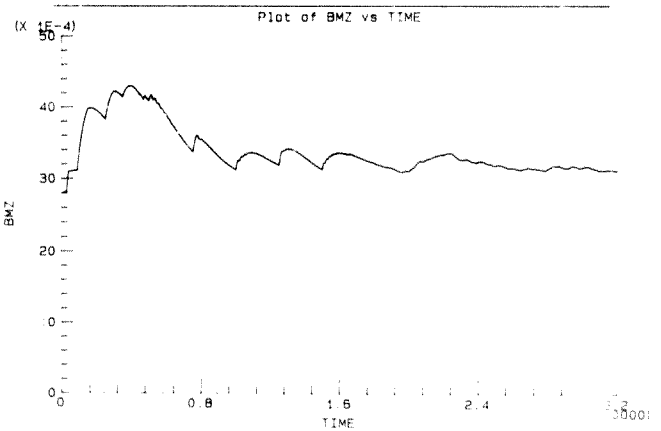

Figure 73. Deadzone Size for Experiment 5

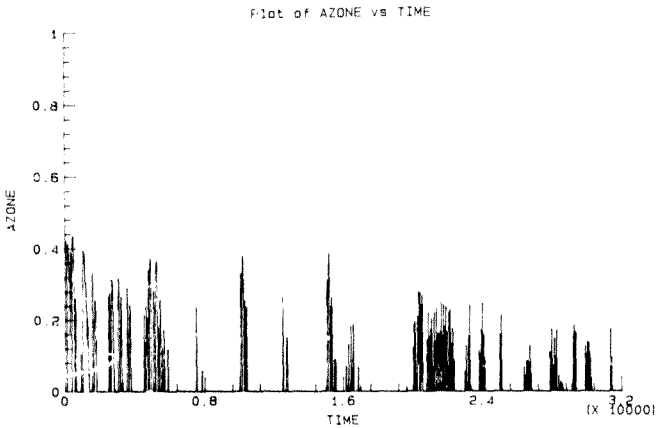

Figure 74. Estimation Gain Sequence a(k) for Experiment 5

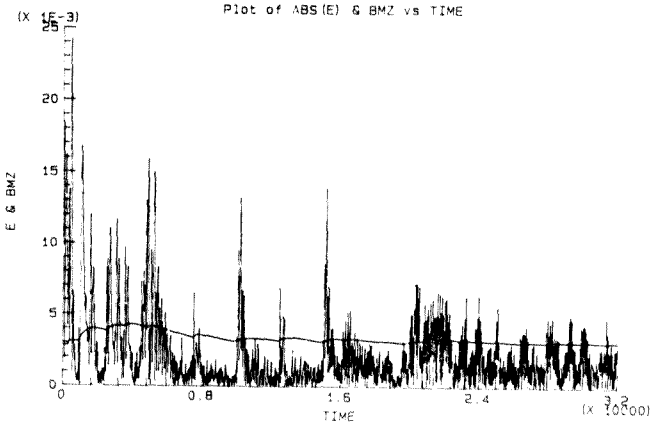

Figure 75. Prediction Error and Deadzone Size for Experiment 5

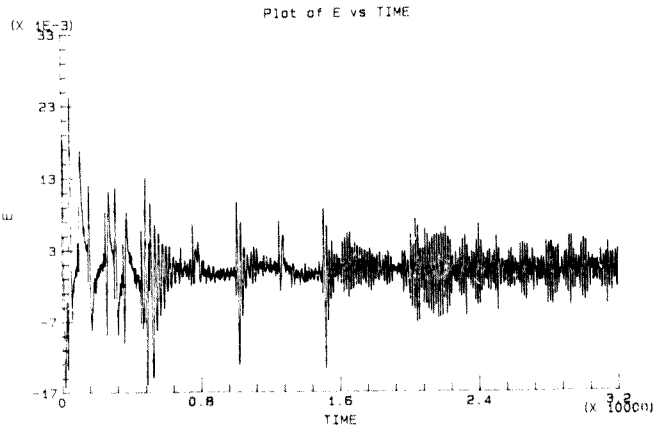

Figure 76. Prediction Error e(t) for Experiment 5

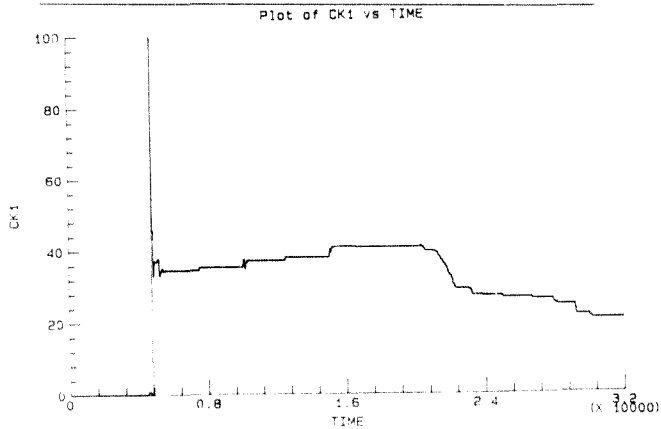

Figure 77. PID Gain Constant CK1 for Experiment 5

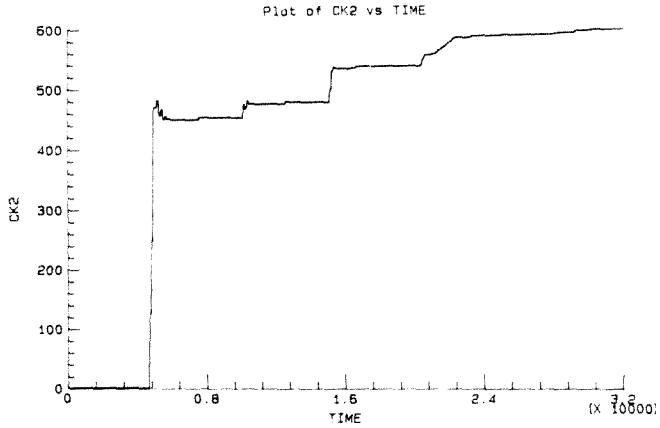

Figure 78. PID Integral Time Constant CK2 for Experiment 5

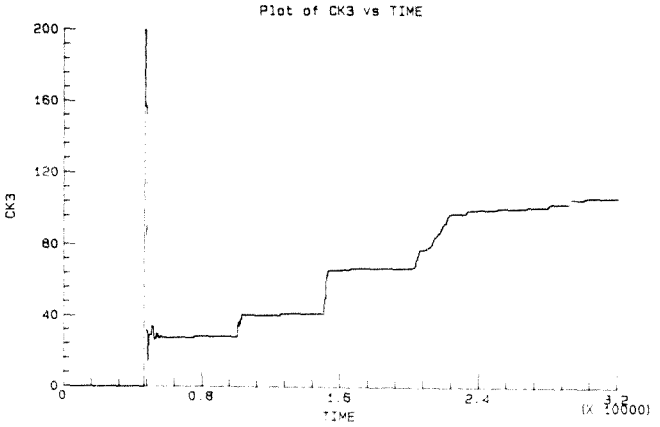

Figure 79. PID Derivative Time Constant CK3 for Experiment 5

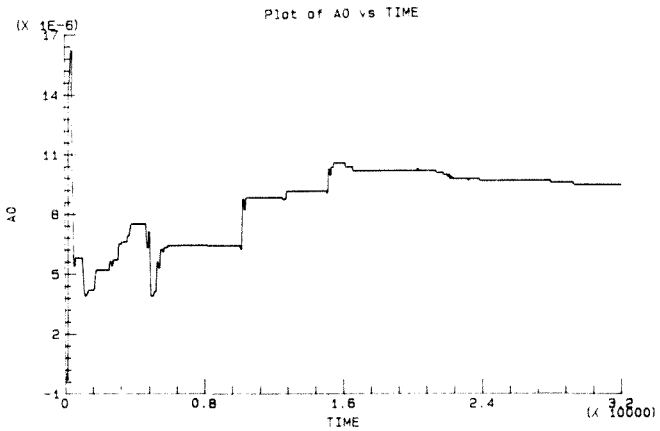

Figure 80. Estimated Parameter a0 for Experiment 5

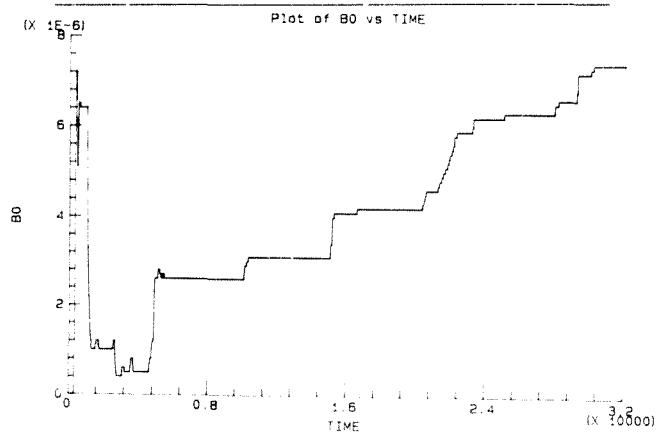

Figure 81. Estimated Parameter b0 for Experiment 5

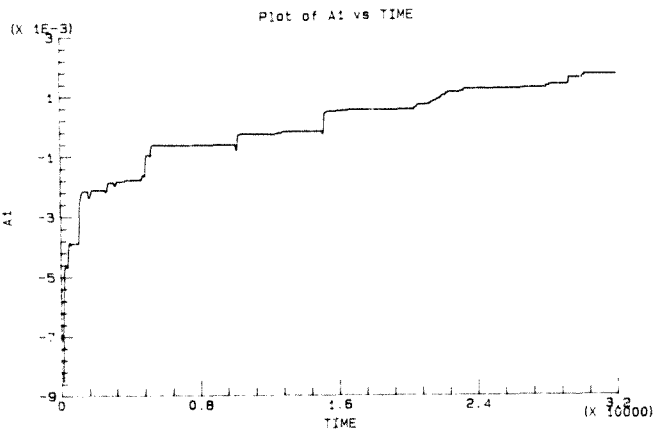

Figure 82. Estimated Parameter a1 for Experiment 5

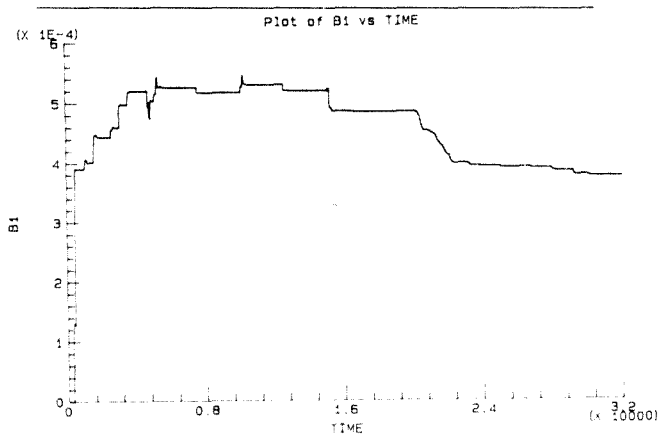

Figure 83. Estimated Parameter b1 for Experiment 5

7.8 EXPERIMENT 6

These results were produced during a simulation run, in which the process was represented by a software model. The experiment was carried out to *examine estimator performance on a true nonminimum-phase plant, with* variable dead-time and substantial parameter variations.

The estimator constants were retained as in Experiments 2, 4 and 5 to yield a robust adaptive system in the long term. The only change was that a sampling interval $\Delta = 5$ seconds was used. This faster sampling was found to affect parameter convergence, particularly the b_1 parameter governing the nonminimum-phase zero of the plant. The interval $\Delta = 5$ seconds is a *compromise between very fast sampling* which may cause numerical problems (since the change in measured signals may be very small from sample to sample), and poorer estimator convergence for slower sampling. Further, much smaller e_i coefficients were used to scale the deadzone function m(k), since the noise levels and expected process-model mismatch could effectively be reduced to zero in the simulation.

The setpoint sequence (Figure 84 on page 147) was exactly as for Experiment 4, thus testing *dynamic setpoint tracking*. The nominal process model used was

$a_0 = 4E\text{-}06$
$b_0 = 4E\text{-}06$
$a_1 = 5E\text{-}03$
$b_1 = -5E\text{-}05$

Thus the system is nonminimum-phase. At time $T = 12500$ seconds, a dead time of $\tau_d = 120$ seconds was introduced. Furthermore, at time $T = 20000$ seconds the a_0 coefficient was reduced by a factor of 10. This resulted in a nett increase in open loop gain of ten times. Such a situation may be encountered practically for example in pH control (Seborg et al, 1986), and is thus a good test for the adaptive system.

The signal ranges were normalised as in all previous experiments as

$$0 \leq (u(t);\ y(t)) \leq 1$$

and a significant initial value was placed on the system output y(t). This is seen in Figure 85 on page 147). The setpoint tracking is good, with the nonminimum-phase characteristic clearly demonstrated as a slight deviation of the output in the opposite direction at each setpoint change, before the tracking response is followed. However, neither the introduction of the time delay nor the decade change in open-loop gain affects the output response substantially. The nonminimum-phase characteristic is slightly more pronounced after the gain change, however this is to be expected since no attempt is made to modify the open-loop zero of the system.

Figure 86 on page 148 shows the control input signal for the experiment. The nonminimum-phase characteristic is manifested as a control signal "spike" in the opposite direction to that expected at a setpoint change, followed by the normal peak decaying to a constant input value. Note the absence of any fast oscillations on the control signal. This is a result of simulation, since although derivative control was applied no rapid external disturbances or fast process changes were present. After the decade change in gain, the steady-state value of the control signal is greatly reduced between setpoint changes. Thisis as expected, since the setpoint itself remained the same for a higher-gain plant.

The deadzone $\beta m(k)$ is described by Figure 87 on page 148. The large deviations in control input signal at setpoint changes result in small peaks on $\beta m(k)$, slightly desensitizing estimation during these periods. This is in accordance with the motivation for $\beta m(k)$, which is to overbound the process noise and modelling error which are correlated with the magnitude of the input.

The error signal (Figure 90 on page 150) decays rapidly, and becomes very small by time T = 5000 seconds. Small spikes on the error are visible when the delay is introduced, and then quite a substantial error is present

after the gain change occurs. The plot of $|e(k)|$ and $\delta m(k)$ combined (Figure 89 on page 149), and $a(k)$ (Figure 88 on page 149) are included for completeness to describe parameter update behaviour. The peak magnitude of $a(k)$ is again limited by α ($\alpha = 0,5$), and it is clear that larger update gain is present both initially during convergence and when the gain change occurs.

The estimated parameters a_0, b_0, a_1 and b_1 are given in Figure 94 on page 152, Figure 95 on page 152, Figure 96 on page 153 and Figure 97 on page 153 respectively. The actual parameter values are superimposed for clarity. It is seen for all parameters that they converge very closely to the real values in a short time. The b_1 coefficient shows a small steady-state error, related to the sampling interval Δ as discussed earlier. After the delay is introduced (T=12500 seconds) small changes occur in all the parameters. The b_1 becomes more negative, indicating a faster unstable zero which agrees with the Pade approximation to the delay term. The changes in the a_1 coefficients at this time are also due to the approximation of the delay by a pole-zero pair. The gain change (reduction of a_0 by a factor of 10) causes rapid convergence of a_0 to its new value, and short-term deviations in the other parameters. The a_1 and b_0 coefficients remain unchanged in the long run, however, while the b_1 coefficient reaches a new value after fairly substantial deviations. These adjustments are due to the strong excitation observed in u(t) at this time.

It is believed that convergence of the b_1 parameter for nonminimum-phase processes is strongly linked to the sampling interval Δ, since the δ-operator approximation relies on fast sampling to yield discrete models "close to" the continuous process. One of the effects discussed in "Delta-operator formulation of Discrete-Time Control" on page 33 is to alter the mapping of zeros from continuous to discrete-time, particularly ensuring that continuous minimum-phase zeros remain thus in the discrete equivalent. However, the approximation $s = \delta$ is dependant on the sampling rate, and thus steady-state error in b_1 was found to decrease for smaller Δ as discussed previously.

The controller coefficients behave as in Figure 91 on page 150, Figure 92 on page 151 and Figure 93 on page 151. Small adjustments in PID gain CK1 and derivative time CK3 are seen after the dead time is introduced. Slightly larger changes in all three coefficients are seen after the gain change occurs, but none are very significant. In contrast to Experiment 5 where a plant gain change (b_0/a_0) resulted in a noticeable adjustment to the PID gain, the change in a_0 here only slightly affects the PID gain. This is difficult to interpret, however it must be remembered that merely an approximation to a PID controller is developed here from a pole placement design. Thus the closed-loop performance is a better indicator of the success of the controller design, clearly demonstrated in the process output y(t). If better agreement with intuitive PID controller settings is desired, a more suitable approximation may have to be found (See Appendix D).

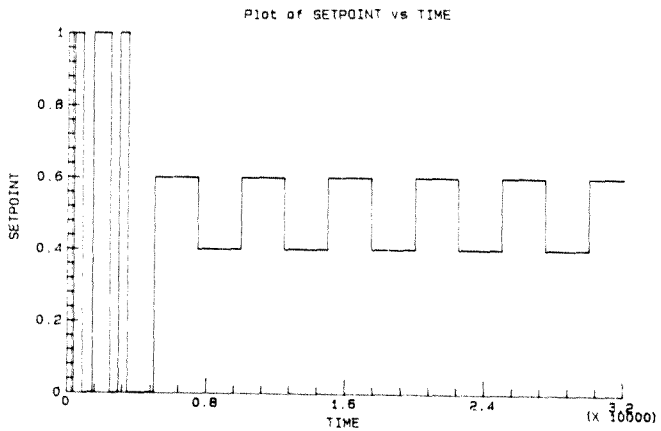

Figure 84. Setpoint Sequence for Experiment 6

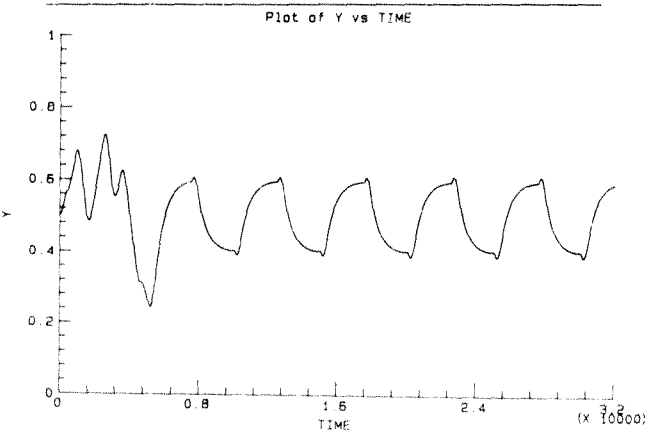

Figure 85. System Output y(t) for Experiment 6

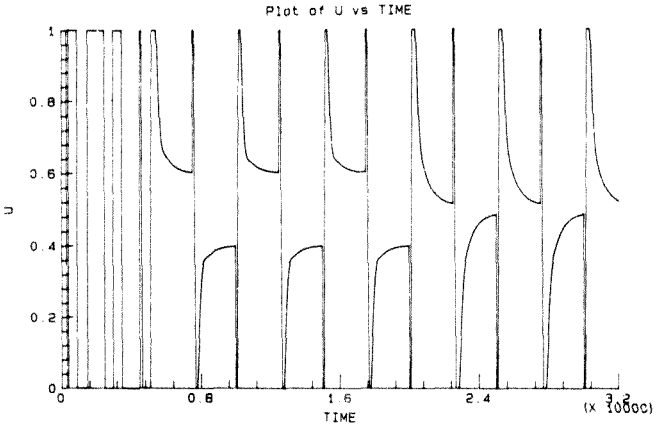

Figure 86. Control Input u(t) for Experiment 6

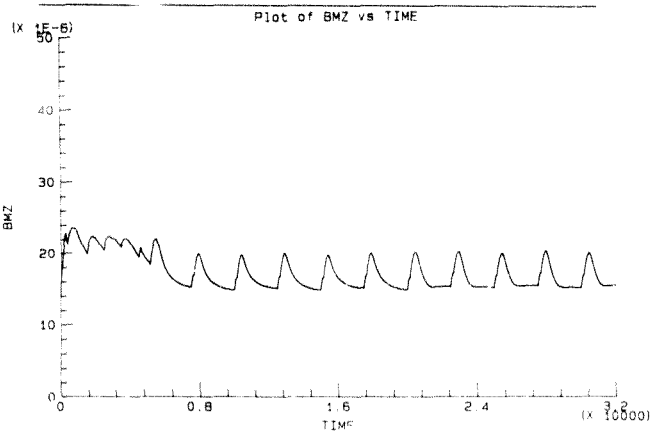

Figure 87. Deadzone Size for Experiment 6

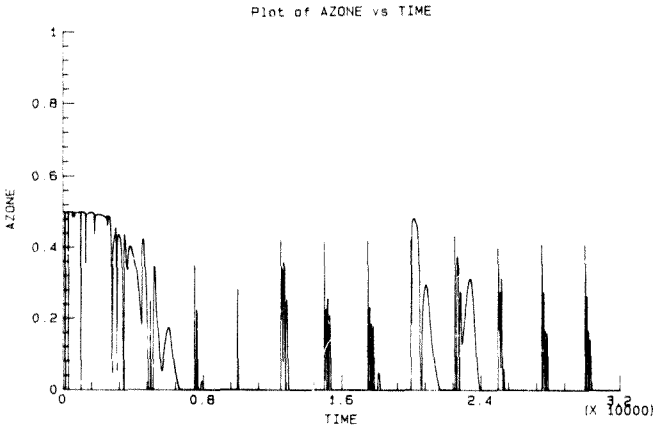

Figure 88. Estimation Gain Sequence a(k) for Experiment 6

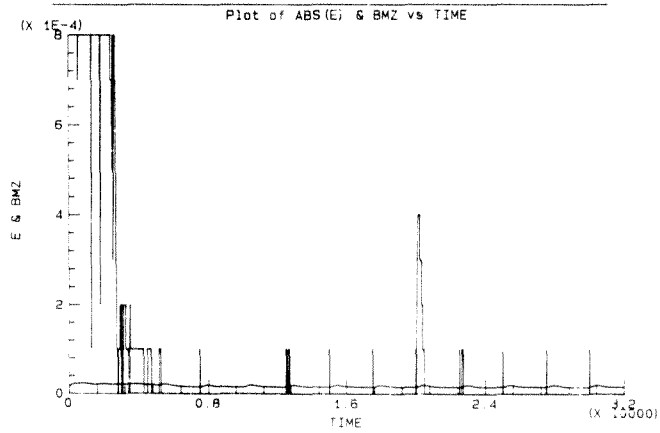

Figure 89. Prediction Error and Deadzone Size for Experiment 6

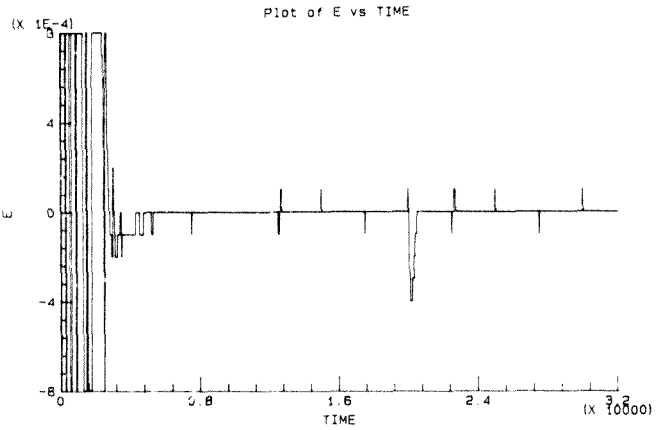

Figure 90. Prediction Error e(t) for Experiment 6

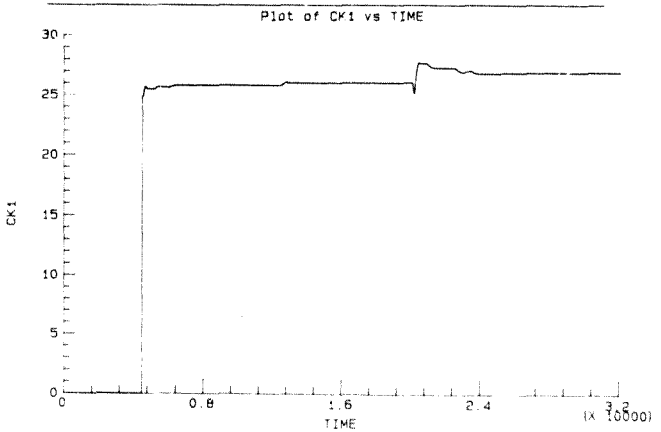

Figure 91. PID Gain Constant CK1 for Experiment 6

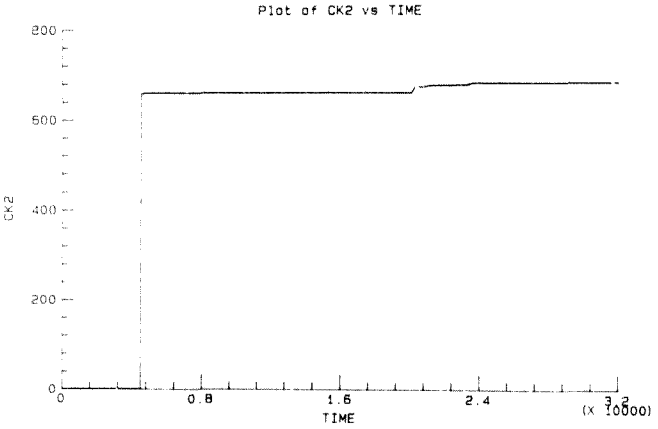

Figure 92. PID Integral Time Constant CK2 for Experiment 6

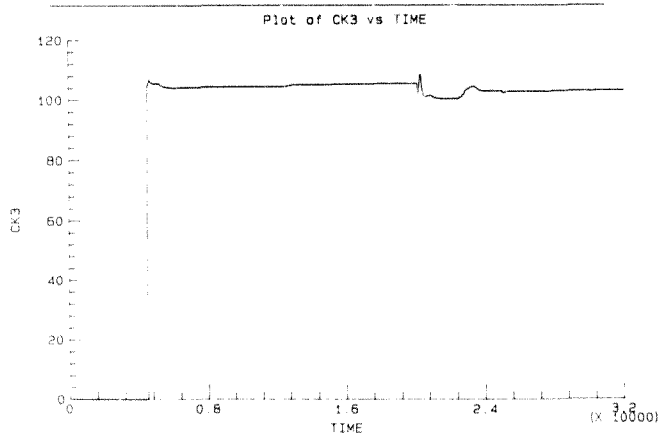

Figure 93. PID Derivative Time Constant CK3 for Experiment 6

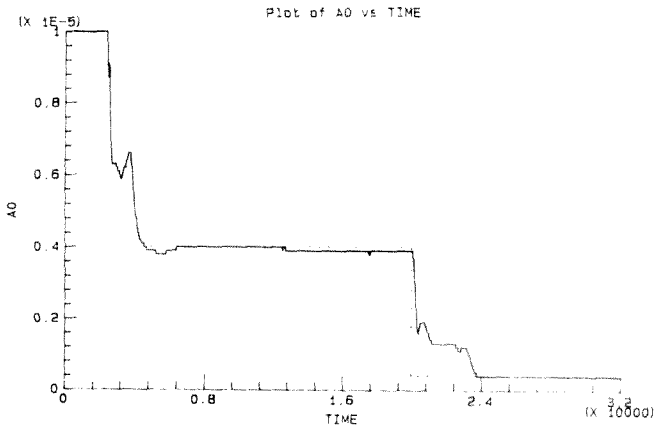

Figure 94. Estimated Parameter a0 for Experiment 6

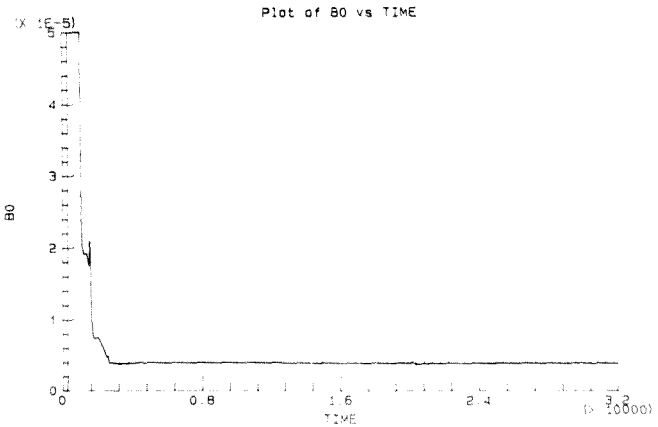

Figure 95. Estimated Parameter b0 for Experiment 6

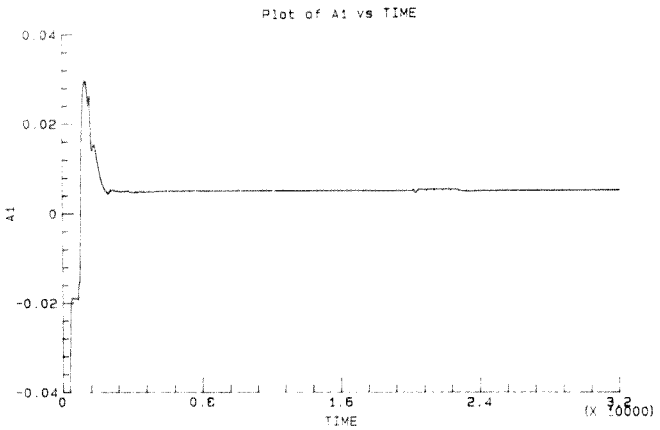

Figure 96. Estimated P rameter a1 for Experiment 6

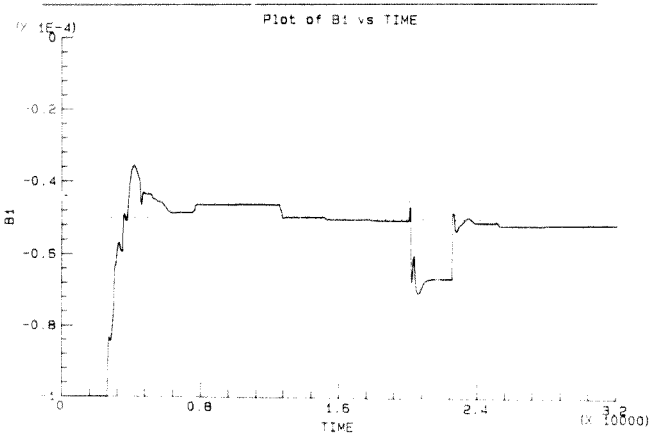

Figure 97. Estimated Parameter b1 for Experiment 6

8.0 CONCLUSION AND RECOMMENDATIONS

The implementation of a robust adaptive control system has been described. The need for adaptive systems has been discussed, and background to the problem of robust adaptive controllers has been examined in a review of current approaches to estimation and control.

A new transform domain has been developed, and a novel operator, viz. the δ-operator, used for all formulations of process models and controller structures. This has been shown to have good numerical properties, as well as exhibiting a close correlation between continuous-time system models and their discrete counterparts. Thus process models and controllers could be examined in a continuous-time framework, giving results which were intuitively easier to interpret. Furthermore, higher sampling rates were possible which removed many of the control problems inherent with slow sampling.

Robust system identification has been facilitated using a general system model, allowing for unmeasurable deterministic disturbances and measured random process disturbances. This resulted in signal conditioning elements to "bandlimit" estimation and hence the validity of the estimated model. A sophisticated relative deadzone was used to prevent parameter estimation on meaningless data. This was seen to improve robustness to process noise as well as "modelling error noise", in the absence of persistent excitation. Other techniques such as covariance modification (exponential data weighting) have been used to maintain algorithm sensitivity to process variations. The numerical robustness of the estimation algorithm was further enhanced by using a covariance factorization technique, namely Bierman's UDU^T factorization.

The robust controller design has been accomplished by closed-loop pole assignment. This has been shown to have good robustness properties, in particular allowing for the control of nonminimum-phase processes. The structure of the estimator has been extended to allow the estimation of

a "disturbance transfer function", from which a feedforward control block was designed. A second-order pole placement solution was developed to yield an equivalent PID-controller structure, providing an intuitively appealing form for industrial implementation.

A large part of this work has focused on real implementation issues, involving estimator parameter selection, signal conditioning filter settings, choice of closed-loop performance criteria and some software considerations. Fundamental choices such as model order and sampling rate have also been discussed.

Practical results have been presented from tests of the algorithm on a physical process, typical of an industrial temperature control application. These results have demonstrated very good performance of the controller under a wide range of conditions. Closed-loop control performance has been investigated for a real plant with varying dead-time, as well as under the influence of a strong external disturbance. The results indicated very good robustness of the closed-loop system for all cases tested.

Furthermore, estimator parameter selection issues were highlighted during practical testing. A tradeoff became apparent between long-term system robustness under possibly "difficult" process conditions, and accuracy of process model in the short term. This was also found to relate to the choice of closed-loop performance objective. It was seen that for tight performance specifications, an accurate process model was necessary to ensure "good" controller design and well-behaved control signals. When the process model accuracy was lower, poor controller design resulted and the control signal exhibited large variations to satisfy the same performance objective. Hence under these circumstances a further compromise may be necessary, in the relaxation of the closed-loop performance requirement.

A set of results produced by simulation further showed good performance on a true nonminimum-phase plant, with variable dead time and subject to a large change in open-loop gain. Thus the overall robustness of the

adaptive controller has been thoroughly examined, under many practical circumstances which may be encountered in industrial processes.

It is believed that this research is significant in that a number of untried theoretical concepts have been successfully combined to give a robust practical adaptive controller. These are summarised as:

- A new transform domain for control formulation, allowing better correspondence between continuous- and discrete-time models.
- Robust estimation involving a generalised process model to include external disturbances, deterministic signal nulling and other signal conditioning, and a sophisticated relative deadzone. A covariance factorization technique is used to improve numerical stability of the algorithm.
- Robust pole assignment controller design with a PID-equivalent form used to give a simple implementation, and the facility for feedforward control.

The limitations of this research are evident in that the controller is not yet fully configured for industrial implementation. Some areas which may require further work are

- Investigation of the suitability of the PID-equivalent form used, and experimentation with other possible approximations.
- A more complete understanding of the choice of closed-loop poles for pole placement.
- Testing of the feedforward compensation allowed for with the controller.
- Evaluation of performance for a wider range of real processes.

Some ideas are given in the following discussion concerning possible forms for an industrial working version of the adaptive controller.

8.1 TOWARDS AN INDUSTRIAL IMPLEMENTATION

This controller could be implemented on a dedicated microprocessor-based controller such as a PLC or other device. Alternatively, it could be incorporated into a larger direct digital control (DDC) computer system. This would allow one software module to serve many control loops, providing substantial gain at small cost.

The algorithm could be applied merely in an automatic tuning sense, to estimate process models and calculate controller coefficients which are downloaded to separate controllers, such as PID modules (either in software or remote hardware). On the other hand, the algorithm could be applied in a true adaptive sense, to continually track variable processes and adjust control law coefficients online. This latter application would require careful attention to the user interface as described by Astrom (1987), Wittenmark and Astrom (1981). They suggest that while a "black box" implementation is impractical and probably undesirable for general applications, one should reduce the number of user-specified parameters to a minimum.

One such possibility is to introduce **performance related dials**, ie. dials relating to the closed-loop performance of the system. This may typically be **closed-loop bandwidth**, or possibly **gain** or **phase margin**. The user would then select a desired performance objective and the adaptive controller parameters would be derived automatically. The constant parameters to be set *a priori* are easily related to closed-loop bandwidth, for example. The closed-loop polynomial A^* has been shown to be limited by closed-loop bandwidth, and the bandlimiting filter on the estimator also follows naturally. If the sampling interval is made variable, this too may be determined by the closed- loop bandwidth

Such concepts have been documented by McDermott and Mellichamp (1984) who propose automatic setting of the sampling rate and closed loop polynomial for pole placement control design. This is done based on an optimization of the sytem step response. Another application is given in Kraus and Myron (1984), of a self-tuning PID controller whose settings are based on the transient setpoint step response. This uses an implicit design method based on a type of "expert system" approach, in which the control engineer's expertise is utilised through knowledge-based rules built into the controller design. In our case, the explicit nature of the estimation and control requires a direct relation between the desired transient performance and the constant parameters used by the adaptive system.

Finally, it must be remembered that the sophistication of an adaptive controller does not imply global applicability. Prior process knowledge may indicate in many cases that a fixed-parameter controller will be very successful. The adaptive controller, being inherently nonlinear, demands careful application to real processes. As stated by Wittenmark and Astrom (1984):

> "The theory deals with idealized situations where all the conditions are under control. The theory thus gives the ultimate limit of what can be achieved and expected under idealized conditions. The practical situation is, however, such that there are all kinds of violations of the theory."

However, good performance has been achieved on a real process during this work. With careful extension of the theory, it is believed that a practical industrial version of this adaptive controller is realisable. Future work may also include testing of the feedforward compensation included with the controller, as well as application to nonlinear or even open-loop unstable processes. The suitability of other PID approximations from a pole assignment design could also be examined.

9.0 APPENDIX A : CONVERGENCE OF RLS ESTIMATOR WITH DEADZONE

CONVERGENCE OF RLS ESTIMATOR WITH RELATIVE DEADZONE

A brief convergence proof is presented for the RLS estimator with a deadzone as defined in this work. The δ-operator form is retained. For more detailed proofs, see Goodwin and Sin (1984 : 61), Goodwin et al (1986), Kreisselmeier and Anderson (1986). The treatment given here is based largely on informal notes provided by Prof. I.M. Macleod (1987).

A.1 Lemma 1 (preliminary lemma)

Given a process model defined by

$$Ay_f(k) = Bu_f(k) + n_f(k)$$

with $$y_f = \frac{D}{D'E}\, y$$

$$u_f = \frac{D}{D'E}\, u$$

$$n_f = \frac{D}{D'E}\, n$$

There exist constants $\sigma_0 \in (0,1)$
$\varepsilon_d \geq 0$
$\varepsilon_1 \geq 0$
$\varepsilon_2 \geq 0$
$m_0 \geq 0$

Such that $|n_f(k)| \leq m(k)$ for all k

where m(k) is the solution of

$$m(k) = \sigma_0 m(k-1) + \varepsilon_d + \varepsilon_1|\bar{\theta}'(k-1)| + \varepsilon_2|y(k-1)| \quad , \qquad m(0) = m_0 \tag{A.1}$$

For any $\sigma_0 \in (\sigma_0', 1)$.

Proof: Rewriting the process model as

$$y_f(k) = Hu_f(k)$$

where $H = H_0(1 + H') = \frac{B}{A}\left[1 + \frac{B'}{A'}\right]$

the "error transfer function" is given by

$$H_0H' = BB'/AA'$$

We can then describe the filtered error as

$$\eta_f = \left[\frac{1}{E}\frac{BB'}{A'}\right]\frac{D}{D'}u + \frac{AD}{ED'}\xi$$

$$= \left[\frac{1}{E}\frac{BB'}{A'}\right]\bar{u} + \frac{D}{ED'}\xi$$

where ξ is a bounded random dsiturbance term.

From Goodwin and Sin (1984 : 488), since E and A' are stable there exist constants K_1, K_2, $K_3 < \infty$ and $\lambda \in (0,1)$ such that

$$|\eta_f(k)| \leq K_1\lambda^k + \sum_{j=1}^{k} \lambda^{j-1}[K_2|\bar{u}(k-j)| + K_3]$$

A closed-form solution for the equation A.1 is

$$m(k) = m_0\sigma_0^k + \sum_{j=1}^{k} \sigma_0^{j-1}[\varepsilon_0 + \varepsilon_1|\bar{u}(k-j)| + \varepsilon_2|y(k-j)|]$$

Thus the result follows by choosing

$$\sigma_0 = \lambda$$
$$\varepsilon_1 \geq K_2$$
$$\varepsilon_2 \geq 0$$
$$\varepsilon_0 \geq K_3$$

A.2 Assumption 1

Constants ε_0, ε_1, ε_2, σ_0, $m(0)$ and a filter $1/E(\delta)$ are assumed known such that

$$|\eta_f(k)| \leq m(k) \text{ for all } k \qquad (A.2)$$

in accordance with the preliminary Lemma.

A.3 Lemma 2 (after Goodwin and Sin 1984 : 60)

The following properties hold for a RLS parameter estimator applied to any system described by

$$y = (B/A)u + d + \xi$$

subject to assumption 1.

(a) $$\lim_{k\to\infty} \frac{f(\delta m(k), e(k))^2}{1 + \phi(k-1)^T \phi(k-1)} = 0$$

(b) $$\lim_{k\to\infty} \|\theta(k) - \theta(k-1)\| = 0$$

(c) For all k, $\|\hat{\theta}(k) - \theta_0\| \leq \upsilon(P_0)\|\hat{\theta}(0) - \theta_0\|$

where $\upsilon(P_0)$ = condition number of matrix P_0

$$\triangleq \lambda_{max}(P_0)/\lambda_{min}(P_0)$$

and λ_{max} is the maximum eigenvalue of P_0

λ_{min} is the minimum eigenvalue of P_0

θ_0 denotes the parameter vector for a suitable nominal model (B/A).

Note: A key fact is that the above properties hold **irrespective** of the control law.

Proof: Consider the non-negative non-increasing Lyapunov function

$$V = \tfrac{1}{2}\tilde{\theta}^T P^{-1}\tilde{\theta} \tag{A.3}$$

where $\tilde{\theta} \triangleq \hat{\theta} - \theta_*$

Now, from the algorithm of RLS :

$$\delta P(k-2) = \frac{-a(k)}{\Delta}\,\frac{P(k-2)\phi(k-1)\phi(k-1)^T P(k-2)}{1 + \phi(k-1)^T P(k-2)\phi(k-1)} \tag{A.4}$$

Applying the matrix inversion lemma gives

$$\Delta\delta P(k-2)^{-1} = \frac{a(k)\phi(k-1)\phi(k-1)^T}{1 - (1-a(k))\phi(k-1)^T P(k-2)\phi(k-1)} \tag{A.5}$$

Also from the RLS algorithm

$$\delta\theta(k-1) = \frac{a(k)}{\Delta}\,\frac{P(k-2)\phi(k-1)}{1 + \phi(k-1)^T P(k-2)\phi(k-1)}\,.e(k) \tag{A.6}$$

where $e(k) = \bar{y}(k) - \phi(k-1)^T\hat{\theta}(k-1)$ (A.7)

Furthermore

$$y(k) = \phi(k-1)^T\theta_* + \eta_f \tag{A.8}$$

where θ_* is a vector of parameters satisfying the nominal process model as before.

Thus $e(k) = \eta_f(k) - \phi(k-1)^T\tilde{\theta}(k-1)$ (A.9)

with $\tilde{\theta}(k-1) = \hat{\theta}(k-1) - \theta_*$ by definition

Using A.5 with A.6, A.7 and A.8 gives (in A.3) :

$$\Delta\delta V = \frac{a}{1 + \phi^T P \phi}\left[\left(\frac{1 + \phi^T P \phi}{1 + (1-a)\phi^T P \phi}\right) \right. \qquad \left. \right] \qquad (A.10)$$

having dropped the argument k.

From the definition of the deadzone

$$a(k) = \begin{cases} 0 & \text{if } |e(k)| \leq \beta m(k) \\ \alpha f(\beta m(k), e(k))/e(k) & \text{otherwise} \end{cases}$$

It is clear that a(k) is bounded by $a \leq \alpha$ for all k.

Also, using the definition of β, viz.

$$\beta = +\sqrt{\varepsilon_4 + 1/(1-\alpha)} \qquad , \qquad \varepsilon_4 > 0$$

we see that for the coefficient term of η_f^2 in A.10

$$\frac{1 + \phi^T P \phi}{1 + (1-a)\phi^T P \phi} \leq \frac{1 + \phi^T P \phi}{1 + (1-\alpha)\phi^T P \phi}$$

$$\leq \beta^2 - \varepsilon_4 \qquad (A.11)$$

Substituting in A.10 gives

$$\Delta\delta V \leq \left[\frac{a}{1 + \phi^T P \phi}\right] \cdot \left[\eta_f^2(\beta^2 - \varepsilon_4) - e^2\right]$$

$$\leq \left[\frac{\alpha}{1 + \phi^T P \phi}\right] \cdot \left[\frac{\beta^2 m^2}{\beta^2}(\beta^2 - \varepsilon_4) - e^2\right]$$

since $|\eta_f| \leq m$ from A.2.

Also note that $|e| > \beta m$ for algorithm updating; deadzone definition prevents update otherwise.

Then $$\Delta\delta V \leq \left[\frac{\alpha}{1 + \phi^T P \phi}\right]\frac{e^2}{\beta^2}(\beta^2 - \varepsilon_4 - \beta^2)$$

$$= \frac{-\varepsilon_4 \alpha}{\beta^2}\left[\frac{1}{1 + \phi^T P \phi}\right] e^2$$

But $|f(\beta m, e)| \leq |e|$ for $\beta m \geq 0$

Thus $$\Delta\delta V \leq \frac{-\varepsilon_4 \alpha}{\beta^2}\left[\frac{1}{1 + \phi^T P \phi}\right] f(\beta m, e)^2 \qquad (A.12)$$

Which implies that the Lyapunov function V is decreasing.

Result (a) follows directly from A.12.

The remaining results follow as in Goodwin and Sin (1984 : 61).

10.0 APPENDIX B : BIERMAN'S UDU COVARIANCE FACTORIZATION

10.0 APPENDIX B : BIERMAN'S UDU COVARIANCE FACTORIZATION

BIERMAN'S UDU^T COVARIANCE FACTORIZATION

The algorithm used in this work was taken from Ljung and Soderstrom (1983 : 329). An analysis of part of the algorithm is presented to document the incorporation of a deadzone gain into the factorization sequence, which is not obvious at first glance.

The covariance matrix update is given by

$$P(t) = \frac{1}{\lambda}\left[P(t-1) - \frac{P(t-1)\phi(t)\phi^T(t)P(t-1)}{\lambda + \phi^T(t)P(t-1)\phi(t)}\right] \qquad (B.1)$$

(λ is a forgetting factor)

We factorize P(t) as

$$P(t) = U(t)D(t)U^T(t)$$

where U(t) is an upper triangular matrix with all diagonal elements equal to 1, and D(t) is a diagonal matrix.

Then B.1 becomes

$$U(t)D(t)U^T(t) = \left[U(t-1)D(t-1)U^T(t-1) - U(t-1)g(t)g^T(t)U(t-1)/\beta(t)\right]/\lambda$$

where $f(t) = U^T(t-1)\phi(t)$

$g(t) = D(t-1)f(t)$

$\beta(t) = \lambda + \phi^T(t)P(t-1)\phi(t)$.

$\quad = \lambda + f^T(t)g(t)$ (B.2)

which gives

$$U(t)D(t)U^T(t) = U(t-1)\left[D(t-1) - g(t)g^T(t)/\beta(t)\right]U^T(t-1)/\lambda \qquad (B.3)$$

Ljung and Soderstrom (1983 : 330) show that the term in brackets can be factorized as

$$D(t-1) - g(t,g^T(t)/\beta(t) = \bar{U}(t)\bar{D}(t)\bar{U}^T(t)$$

giving the relationships

$$U(t) = U(t-1)\bar{U}(t)$$
$$D(t) = \bar{D}(t)/\lambda$$

Their algorithm is summarised as follows :

Initialise U(0) and D(0) at time t=0 such that $U(0)D(0)U^T(0) = P(0)$.

At time t, compute vector L(t) and update U(t-1) and D(t-1) by performing steps 1-6 :

1. Compute $f = U^T(t-1)\phi(t)$;
   $g = D(t-1)f$;
   $\beta_0 = \lambda$

2. For j = 1 ... d repeat steps 3 - 5.

3. Compute $\beta_j = \beta_{j-1} + f_j g_j$

   $D(t)_{jj} = \beta_{j-1} D(t-1)_{jj}/\beta_j \lambda$

   $v_j = g_j$

   $u_j = -f_j/\beta_{j-1}$

4. For i = 1 ... j-1 repeat step 5 (if j = 1 skip step 5)

5. Compute $U(t)_{ij} = U(t-1)_{ij} + v_i u_j$

   $v_i = v_i + U(t-1)_{ij} v_j$

6. $\bar{L}(t) = \begin{bmatrix} v_1 \\ \vdots \\ v_d \end{bmatrix}$ $\quad L(t) = \bar{L}(t)/\beta_d$

A discussion is now given to include the deadzone update gain in the recursion. This is taken from the proof given in Bierman (1977 : 78), and is based on informal notes provided by Prof. I.M. Macleod (1987). To avoid confusion, we call the deadzone update gain b(k).

For convenience, the equivalence in notation with Ljung and Soderstrom (1983) is tabulated below.

Bierman	Ljung and Soderstrom
$\tilde{U}$	U
$\tilde{D}$	D
a	ϕ
v	g
f	f
α	β
λ	μ
K	v
r	λ

The equivalent algorithm is then as follows in Bierman's notation :

(i) $f = \tilde{U}^T a$ $\qquad f^T = (f_1 \ \ldots \ f_n)$

$v = \tilde{D} f,$ $\qquad v_i = \tilde{d}_i f_i \quad i = 1 \ \ldots \ n$

(ii) $\hat{d}_i = \tilde{d}_i r/\alpha_i$

$\alpha_i = r + v_i f_i$

$K_i^T = [v_i \ \underbrace{0 \ \ldots \ 0}_{n-i}]$

For j = 2 ... n cycle through steps (iii) - (vi) :

(iii) $\alpha_j = \alpha_{j-1} + v_j f_j$

(iv) $\hat{d}_j = \tilde{d}_j \alpha_{j-1}/\alpha_j$

(v) $\hat{u}_j = \tilde{u}_j + \lambda_j \tilde{K}_j$
$\lambda_j = -f_j/\alpha_{j-1}$

(vi) $K_{j-1} = K_j + v_j \tilde{u}_j$

where $\tilde{U} = |\tilde{u}_1 \ \ldots \ \tilde{u}_n|$, $\hat{U} = |\hat{u}_1 \ \ldots \ \hat{u}_n|$

(vii) Kalman gain K is $K = K_{n+1}/\alpha_n$

Further, in the notation of Bierman we then have

$$\hat{U}\hat{D}\hat{U}^T = \tilde{U}[\tilde{D} - (\tilde{D}\tilde{U}^T a)(\tilde{D}\tilde{U}^T a)^T/\alpha]\tilde{U}^T/r$$

$$= \tilde{U}[\tilde{D} - vv^T/\alpha]\tilde{U}^T/r \qquad (B.4)$$

where $\hat{U} = U(k-1)$
$\hat{D} = D(k-1)$
$\tilde{U} = U(k-2)$
$\tilde{D} = D(k-2)$

and $\alpha = r + \sum_{i=1}^{n} f_i v_i$

If the bracketed term in B.4 is factored as UDU^T, then we have

$$U = UU^T$$
$$D = D/r \qquad (B.5)$$

Bierman proceeds by using the Agee-Turner factorization update theorem (cf. Bierman 1977 : 44) to give the bracketed term in the above expressions, ending with a simple representation for U and D :

$$\hat{d}_j = \tilde{d}_j + c_j v_j^2 \qquad (B.6)$$

$$c_{j-1} = c_j \tilde{d}_j / \hat{d}_j \qquad (B.7)$$

$$\tilde{u}_{ij} = c_j v_i / \hat{d}_j \quad \text{for } i = 1 \ldots j-1 \qquad (B.8)$$

These expressions are backward recursive, from j = n down to j = 1. Now, c_n must include the deadzone update *gain* b(k), thus :

$$c_n = -b/a \qquad (B.9)$$

Bierman continues to develop a different representation for d_j, since at this point the recursions B.6 - B.8 update the diagonals d_j as **differences**. Such calculations are susceptible to loss of numerical accuracy, the prevention of which is the very motivation for covariance factorization. The equations B.6 and B.7 are rearranged as

$$\hat{d}_j = \tilde{d}_j + c_j(\tilde{d}_j f_j)^2 = \tilde{d}_j(1 + c_j \tilde{d}_j f_j^2) \qquad (B.10)$$

$$1/c_{j-1} = 1/c_j(\hat{d}_j/\tilde{d}_j) = 1/c_j + \tilde{d}_j f_j^2 \qquad (B.11)$$

Now, since

$$b/c_n = -a = -(r + \sum_{i=1}^{n} \tilde{d}_i f_i^2)$$

it follows that

$$1/c_j = -\alpha_j/b \qquad \text{from B 11.}$$

If we eliminate the c_j's from the algorithm, B.7 gives

$$\hat{d}_j = \tilde{d}_j c_j / c_{j-1} = \tilde{d}_j \alpha_{j-1}/\alpha_j$$

which is the same result as in step (iv) of the algorithm.

Similarly, using B.8 and realising that

$$k_j = v_j c_j / \hat{d}_j \qquad \text{(from the algorithm)}$$

we have

$$\lambda_j = v_j c_j / \hat{d}_j = -bv_j/(\alpha_j \hat{d}_j)$$

$$= -b\tilde{d}_j f_j \alpha_j/(\alpha_j \tilde{d}_j \alpha_{j-1}) \qquad \text{(substituting from the algorithm)}$$

$$= -bf_j/\alpha_{j-1} \qquad \text{(B.12)}$$

The remaining developments are as in Bierman (1977 : 80), with the modification

$$K = \tilde{U}\tilde{D}\tilde{U}^T a/\alpha_n$$

$$= b\tilde{U}v/\alpha_n$$

$$= bK_{n+1}/\alpha_n \qquad \text{(B.13)}$$

Thus the algorithm is modified to include a deadzone update gain b(k), simply by replacing steps (v) and (vii) by B.12 and B.13 respectively.

Going back to the notation of Ljung and Soderstrom (1983), we modify the original algorithm by substituting in step 3:

$$\mu_j = -f_j b/\beta_{j-1}$$

where b still represents the deadzone update gain.

Also add the following after step 5 :

$$\beta_d = \beta_d/b$$

which will effectively alter the normalised Kalman gain $\bar{L}$ to update L(t) in step 6 as :

$$L(t) = \bar{L}(t)b/\beta_d$$

11.0 APPENDIX C : POLE ASSIGNMENT FOR HIGHER-ORDER CONTROLLERS

POLE ASSIGNMENT FOR HIGHER-ORDER CONTROLLERS

C.1 Introduction

The implementation of the pole assignment controller developed in this research can be extended to controllers of any order. A limiting case is presented in this Appendix, using a third order estimator and a third order controller in both its proper and strictly proper forms.

This estimated model structure allows for effective parameterization of up to a third order plant with no dead time, or a second order plant with dead time. This result is arrived at by using the well-known first-order Pade approximation to the element $e^{-\theta\delta}$ in a system transfer function, to replace the delay element with a first-order pole-zero pair (Ralston 1965 : 278).

C.2 Strictly Proper Implementation

We develop the plant model and controller using the following polynomial orders:

degree(A) = 3
degree(B) = 2
degree(E) = 3
degree(D) = 1
degree(S) = 0
degree(P) = 2
degree(L) = 2

where all polynomials have the same meaning as used in the preceding discussions, and D and S are simply defined as

D = δ (for a constant d.c. offset disturbance)
S = 1 (no deterministic component in the setpoint)

POLE ASSIGNMENT FOR HIGHER-ORDER CONTROLLERS

C.1 Introduction

The implementation of the pole assignment controller developed in this research can be extended to controllers of any order. A limiting case is presented in this Appendix, using a third order estimator and a third order controller in both its proper and strictly proper forms.

This estimated model structure allows for effective parameterization of up to a third order plant with no dead time, or a second order plant with dead time. This result is arrived at by using the well-known first-order Pade approximation to the element $e^{-\theta\delta}$ in a system transfer function, to replace the delay element with a first-order pole-zero pair (Ralston 1965 : 278).

C.2 Strictly Proper Implementation

We develop the plant model and controller using the following polynomial orders:

degree(A) = 3
degree(B) = 2
degree(E) = 3
degree(D) = 1
degree(S) = 0
degree(P) = 2
degree(L) = 2

where all polynomials have the same meaning as used in the preceding discussions, and D and S are simply defined as

D = δ (for a constant d.c. offset disturbance)
S = 1 (no deterministic component in the setpoint)

The controller order emerges from the solution of the Diophantine equation, which is presented later in this section.

The structure for the strictly proper controller is

$$G_C(\delta) = \frac{p_2\delta^2 + p_1\delta + p_0}{\delta(\delta^2 + l_1\delta + l_0)}$$

$$= \frac{P}{LDS} \qquad (C.1)$$

Note the presence of a pure integral term in the denominator of $G_C(\delta)$.

To design the controller, the coefficients of the polynomials L and P must be found. This is done by solving the Diophantine equation (from 5.5)

$$ALDS + PB = A^* \qquad (C.2)$$

where A^* is the desired closed loop characteristic polynomial, given by

$$A^* = \delta^6 + \alpha_5\delta^5 + \ldots\ldots + \alpha_2\delta^2 + \alpha_1\delta + \alpha_0$$

Note : the order of A^* is twice the order of the plant denominator polynomial A. This is used as a general rule in all cases.

The set of simultaneous equations represented by equating coefficients in C.2 can be expressed in matrix form as follows:

$$\begin{bmatrix} 1 & 0 & 0 & 0 & 0 & 0 \\ a_2 & 1 & 0 & 0 & 0 & 0 \\ a_1 & a_2 & 1 & b_2 & 0 & 0 \\ a_0 & a_1 & a_2 & b_1 & b_2 & 0 \\ 0 & a_0 & a_1 & b_0 & b_1 & b_2 \\ 0 & 0 & a_0 & 0 & b_0 & b_1 \\ 0 & 0 & 0 & 0 & 0 & b_0 \end{bmatrix} \begin{bmatrix} l_2 \\ l_1 \\ l_0 \\ p_2 \\ p_1 \\ p_0 \end{bmatrix} = \begin{bmatrix} 1 \\ \alpha_5 \\ \alpha_4 \\ \alpha_3 \\ \alpha_2 \\ \alpha_1 \\ \alpha_0 \end{bmatrix} \qquad (C.3)$$

Note : the matrix is not square, in that an additional degree of freedom exists. This allows an easy closed-form solution, thus:

1. $l_2 = 1$

2. $p_0 = \alpha_0 / b_0$

3. $l_1 = \alpha_1 - a_2$

4. $p_2 = (X_1 + X_2 - X_3)/X_4$ where

   - $X_1 = b_2\alpha_2 - b_2a_1\alpha_4 - b_1\alpha_3 + b_1a_2\alpha_2$
   - $X_2 = b_1(a_1l_1 - a_2^2l_1 + a_2l_2 - a_2a_1l_2)$
   - $X_3 = b_2(a_0l_1 + b_2p_0 - a_1a_2l_1 - a_1^2l_2)$
   - $X_4 = b_2b_0 - b_2^2a_1 - b_1^2 + a_2b_1b_2$

5. $p_1 = (Y_1 + Y_2)/b_2$ where

   - $Y_1 = \alpha_3 - a_2\alpha_4 - a_1l_1 - a_2^2l_1$
   - $Y_2 = a_2a_1l_2 + a_2b_2p_2 - a_0l_2 - b_1p_2$

6. $l_0 = (\alpha_1 - b_1p_0 - b_0p_1)/a_0$

This solution assumes that none of the plant parameters are zero. However, if any of the b_i's are zero, simpler solutions can be derived, thus:

If $b_2 = 0$; $b_1 \neq 0$:

1. $l_2 = 1$

2. $p_0 = \alpha_0 / b_0$

3. $l_1 = a_3 - a_2$

4. $l_0 = a_4 - a_2 l_1 - a_1$

5. $p_1 = (a_1 - a_0 l_0 - b_1 p_0)/b_0$

6. $p_2 = (a_2 - a_1 l_0 - a_0 l_1 - b_1 p_1)/b_0$

If $b_1 = 0$; $b_2 \neq 0$:

1. $l_2 = 1$

2. $p_0 = a_0 / b_0$

3. $l_1 = a_3 - a_2$

4. $p_1 = (X_1 + X_2)/X_3$ where

   - $X_1 = b_2(a_1 - a_0 a_4 + a_0 a_2 l_1 + a_3 a_1)$
   - $X_2 = b_0(a_3 l_1 + a_0 - a_2^2 l_1 - a_1 a_2 - a_3 + a_2 a_4)$
   - $X_3 = b_2(b_0 a_2 - b_2 a_0)$

5. $p_1 = (a_1 - a_0 a_4 + a_0 a_2 l_1 + a_0 a_1 + a_0 b_2 p_2)/b_0$

6. $l_0 = (a_1 - b_1 p_0 - b_0 p_1)/a_0$

If $b_1 = 0$: $b_2 = 0$:

1. $l_2 = 1$

2. $p_0 = a_0 / b_0$

3. $l_1 = a_3 - a_2$

4. $l_0 = \alpha_4 - a_2 l_1 - a_1$

5. $p_1 = (\alpha_1 - a_0 l_0)/b_0$

6. $p_2 = (\alpha_2 - a_1 l_0 - a_0 l_1)/b_0$

C.3 Proper Controller Implementation

We develop the estimator model using the same polynomial orders, viz. :

degree(A) = 3
degree(B) = 2
degree(E) = 3
degree(D) = 1
degree(S) = 0
degree(P) = 3
degree(L) = 2

The only difference is the allowing of a third order numerator polynomial P in the controller.

The structure for the proper controller is

$$G_C(\delta) = \frac{p_3\delta^3 + p_2\delta^2 + p_1\delta + p_0}{\delta(\delta^2 + l_1\delta + l_0)} \qquad (C.4)$$

Again, we solve the Diophantine equation

$$ALDS + PB = A^*$$

The set of simultaneous equations represented by equating coefficients in 5.4 for this case can be expressed in matrix form as follows:

$$\begin{bmatrix} 1 & 0 & 0 & 0 & 0 & 0 & 0 \\ a_2 & 1 & 0 & b_2 & 0 & 0 & 0 \\ a_1 & a_2 & 1 & b_1 & b_2 & 0 & 0 \\ a_0 & a_1 & a_2 & b_0 & b_1 & b_2 & 0 \\ 0 & a_0 & a_1 & 0 & b_0 & b_1 & b_2 \\ 0 & 0 & a_0 & 0 & 0 & b_0 & b_1 \\ 0 & 0 & 0 & 0 & 0 & 0 & b_0 \end{bmatrix} \cdot \begin{bmatrix} l_2 \\ l_1 \\ l_0 \\ p_3 \\ p_2 \\ p_1 \\ p_0 \end{bmatrix} = \begin{bmatrix} 1 \\ \alpha_6 \\ \alpha_5 \\ \alpha_4 \\ \alpha_3 \\ \alpha_2 \\ \alpha_1 \\ \alpha_0 \end{bmatrix}$$

(C.5)

Unfortunately, a closed-form solution for this matrix is extremely difficult to obtain. Thus numerical matrix methods are used to solve this system for the controller coefficients.

This controller is the most general form possible with a third order system.

C.4 Software Realisation

The strictly proper and proper controllers may be implemented in many different forms. The numerical robustness of each of these implementations is similar, and depends on the relative magnitudes of the controller coefficients (p_i's and l_i's). The strictly proper structure is used to demonstrate three possible implementations, with the proper structure illustrated in a fourth implementation.

C.4.1 Controller Form

The block diagram for this form is given in Figure 98. The pseudo-code implementation is as follows:

```
CONTROLERR = SETPOINT - Y
U = P0*C2 + P1*C1 + P2*C0
C0D = CONTROLERR - L0*C1 - L1*C0
C1D = C0
C2D = C1
C0 = C0D*DELTA + C0
C1 = C1D*DELTA + C1
C2 = C2D*DELTA + C2
```

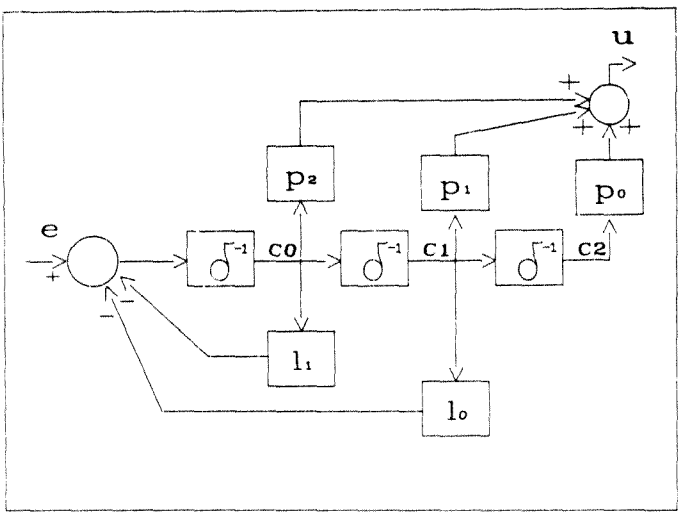

Figure 98. Controller Form Implementation

C.4.2 Controller Form with Cascaded Integrator

The block diagram for this form is given in Figure 99. It is an intuitively appealing implementation since the action of the integrator is clearly illustrated. In pseudo-code this is:

```
C1D = C0 - L0*C2 - L1*C1
U = P0*C2 + P1*C1 + P2*C1D
C0D = SETPOINT - Y
C2D = C1
C0 = C0D*DELTA + C0
C1 = C1D*DELTA + C1
C2 = C2D*DELTA + C2
```

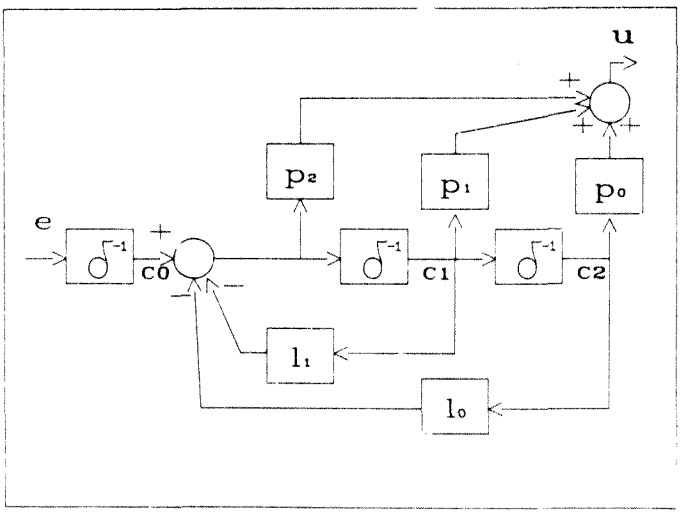

Figure 99. Controller Form with Cascaded Integrator

C.4.3 Observer Form

The block diagram for this form is given in Figure 100 In pseudo-code this becomes

```
U = C2
CONTROLERR = SETPOINT - Y
C0D = P0*CONTROLERR
C1D = C0 + P1*CONTROLERR - L0*U
C2D = C1 + P2*CONTROLERR - L1*U
C0 = C0D*DELTA + C0
C1 = C1D*DELTA + C1
C2 = C2D*DELTA + C2
```

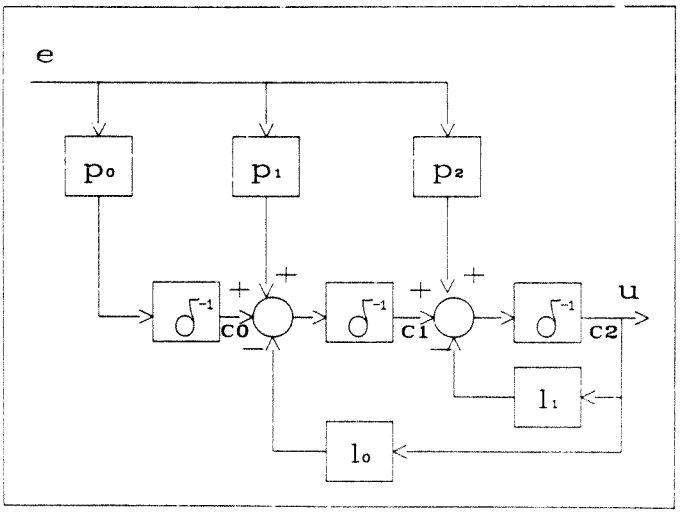

Figure 100. Observer Form

C.4.4 ARMA Form

The proper controller is illustrated in a form typically used for digital filtering applications, viz. the ARMA form. This is shown in Figure 101 on page 188, and the pseudeo-code for software realisation is given as:

```
CONTROLERR = SETPOINT - Y
U = P3*CONTROLERR + P2*C0 + P1*C1 + P0*C2 - L1*C3 - L0*C4
C0D = CONTROLERR
C1D = C0
C2D = C1
C3D = U
C4D = C3
C0 = C0D*DELTA + C0
C1 = C1D*DELTA + C1
C2 = C2D*DELTA + C2
C3 = C3D*DELTA + C3
C4 = C4D*DELTA + C4
```

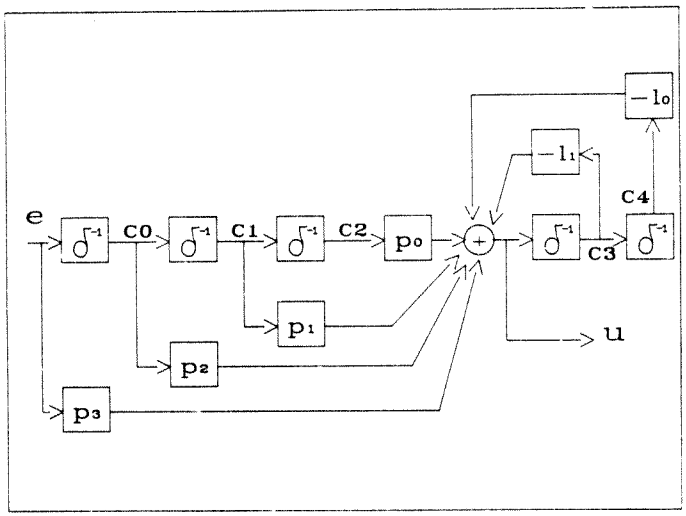

Figure 101. ARMA Block Diagram Implementation of Controller

12.0 APPENDIX D : PID APPROXIMATIONS FROM POLE ASSIGNMENT

PID Approximations from Pole Assignment

In addition to the PID approximation used in this work, many others are possible. Another two are presented here by way of illustration.

D.1 Direct Formula

This approximation is based on equation 6.12, which is the direct formula

$$M(s) = K_c\left[\, 1 + 1/(T_i s) + T_d s/(aT_d s+1)\right]$$

Rearranging, and substituting $s = \delta$ we have

$$M(\delta) = U(\delta)/E(\delta)$$

$$= \frac{K_c}{T_i T_d a}\left[\frac{(1+a)T_i T_d \delta^2 + (T_i + aT_d)\delta + 1}{\delta(\delta + 1/T_d a)}\right]$$

$$= \frac{p_2\delta^2 + p_1\delta + p_0}{\delta(\delta + l_0)}$$

Then $p_2 = K_c(1+a)/a$

$p_1 = K_c(T_i + aT_d)/(T_i T_d a)$

$p_0 = K_c/T_i T_d a$

$l_0 = 1/T_d a$

Giving the approximations

$T_d = 1/(a l_0)$

$K_c = p_2 a/(1+a)$

$T_i = p_1/p_0 - aT_d$

$$= p_1/p_0 - 1/l_0 \qquad (D.1)$$

It is significant that this solution depends on the fixed constant a, where

$$0{,}05 \leq a \leq 0{,}2 \qquad \text{typically.}$$

D.2 "Cascaded" Form

The second alternative approximation comes from equation 6.14, which corresponds to a "cascaded" realisation. This is equivalent to the other continuous PID formulae if $T_i > 4T_d$, as mentioned previously. Then

$$M(s) = K_c(1 + 1/(T_i s))\left[\frac{T_d s + 1}{aT_d s + 1}\right]$$

which is rearranged as follows (with the delta-operator approximation $s = \delta$):

$$M(\delta) = U(\delta)/E(\delta)$$

$$= \frac{K_c}{T_i T_d a}\left[\frac{(T_i T_d \delta^2 + (T_i + T_d)\delta + 1}{\delta(\delta + 1/T_d a)}\right]$$

$$= \frac{p_2\delta^2 + p_1\delta + p_0}{\delta(\delta + l_0)}$$

Then $p_2 = K_c/a$

$$p_1 = K_c(T_i + T_d)/(T_i T_d a)$$

$$p_0 = K_c/T_i T_d a$$

$$l_0 = 1/T_d a$$

Giving the approximations

$$K_c = p_2 a$$

$$T_d = 1/(a l_s)$$

$$\begin{aligned} T_i &= p_1/p_s - T_d \\ &= p_1/p_s - 1/a l_s \end{aligned} \qquad (D.2)$$

These approximations are again dependant on the fixed low-pass cutoff value a, which can vary from manufacturer to manufacturer in industrial PID's. Further experimentation is required to determine the suitability of these alternatives.

13.0 APPENDIX E : SOFTWARE LISTING

```
PROGRAM ADAPT;            {A Generalised Robust Adaptive Controller}

{                                          M.L. Bergesen
                                          30 March 1988
*****************************************************************}

{$R+}
{$I TYPEDEF.SYS}
{$I GRAPHIX.SYS}                {Include Files for Graphics}
{$I KERNEL.SYS}

CONST  ZERO = 0;
       EPSHP = 0.0008;          WE = 0.025;
       UMAX = 1.0;              UMIN = 0.0;
       CTR2 = 10.0;             GAMMA = 600;
       SIGMA = 0.98;            EPS1 = 0.002;
       EPS0 = 0.002;            EPS2 = 0.0002;
       ALPHA = 0.5;             BETA = 1.4;
       TAUTO = 4500;            TSTOP = 32000;
       SPS1 = 1.0;              SPMIN = 0.0;
       SPMAX = 1.0;             T1 = 300;        T13 = 17500;
       T2 = 400;                T3 = 900;        T14 = 20000;
       T4 = 1500;               T5 = 2500;       T15 = 22500;
       T6 = 3000;               T7 = 3500;       T16 = 25000;
       T8 = 5000;               T12 = 15000;     T17 = 27500;
       LAMK = 0.97;             T9 = 7500;       T18 = 30000;
       T10 = 10000;             T11 = 12500;
       CFAC = 0.3;              VSPAN = 10.0;
       MANU = 1;                USPAN = 8.7;
       TU1S = 0.0;              UZERO = 0.5402299;
       TU1T = 0.0;              DELTA = 20;
       XTE100 = 0.0;            SIZE = 4;
       TC00 = 0.0;              Y00 = 0.0;


TYPE   AVEC = ARRAY[1..5] OF REAL;
       THVEC = ARRAY[1..4] OF REAL;
       EAVECTOR = ARRAY[1..2] OF REAL;
       UVECTOR = ARRAY[1..6] OF REAL;
       VECTOR = ARRAY[1..7] OF REAL;
       INTVECT = ARRAY[1..SIZE] OF INTEGER;
       ST = STRING[8]

VAR
       {Variables for Estimator and Controller}

       A1, A0, B1, B0 : REAL;
       YHAT : REAL;
       XHP1, XHP2 : REAL;
       EXIT, AUTO, RFLAG : BOOLEAN;
       YEF, CDF, CDF1, CK1, CK2, CK3 : REAL;
       SIG1, MZ0, AZONE : REAL;
       Y,U,UOUT,YINIT : REAL;
       ASTAR : AVEC;
       YBAR, E, UBAR : REAL;
       MZ, BMZ : REAL;
       SP, YL : REAL;
       CP0, CL0, CP1, CP2 : REAL;
       THETA, K, GAM, YVEC : THVEC;
       YBAR1, BJ : REAL;
```

```
        UVEC : UVECTOR;
        EAVEC : EAVECTOR;
        PHIK1, D, ETH : THVEC;
        T, TOLD : INTEGER;

        {Variables for A/D, D/A Conversion}

        VOLTAGE : REAL;
        COUNT : INTEGER;
        BASE_ADDRESS, COMMAND_REGISTER, STATUS_REGISTER : INTEGER;
        DATA_REGISTER, COMMAND_WAIT, WRITE_WAIT, READ_WAIT : INTEGER;
        CSTOP, CCLEAR, CERROR, CADIN, GAIN_CODE : INTEGER;
        RRANGE, ROFFSET, RFACTOR : REAL;
        STATUS : INTEGER;
        HIGH, LOW : INTEGER;
        TEMP, DATA_VALUE, ERROR1, ERROR2 : REAL;
        ERRFLAG : BOOLEAN;
        CDAOUT, EXT_TRIGGER, INTVAL : INTEGER;
        RANGE, OFFSET, FACTOR : VECTOR;

        {Variables for Graphic Interface}

        DRAWCOUNT : INTEGER;
        NEWX,OLDX : INTEGER;
        NEWY,OLDY : INTVECT;
        OLDYVAL,NEWYVAL : INTVECT;
        OLDSP,NEWSP : INTEGER;
        FLASH : BOOLEAN;
        CONSTAT : STRING[6]

        {Variables for Real Time System Clock}

        SYSTIME : ST,
        INITTIME : REAL;
        CH : CHAR,
        HR,MIN,SEC, NEWTIME, OLDTIME : REAL;
        CD : INTEGER;

        {Files for Data Logging}

        OUTFILE : TEXT;
        OUTPARMS : TEXT;

{General Graphics Plotting Procedures}

PROCEDURE DRAWSP;    {Plots Setpoint Display}

VAR I : INTEGER;
    FILLING : BOOLEAN;

BEGIN
    SELECTSCREEN(2);
    CLEARSCREEN;
    SELECTSCREEN(1);
    COPYSCREEN;
    SELECTSCREEN(2);
    FILLING:= TRUE;
    GOTOXY(1,6);
    WRITE('SETPOINT');
```

```
    SELECTWINDOW(7);
    SELECTWORLD(2);
    SETCOLORBLACK;
    {DRAWSQUARE(0,OLDSP,ROUND(XMAXGLB/12),0,FILLING);}
    DRAWLINE(0,OLDSP,ROUND(XMAXGLB/12),OLDSP);
    NEWSP:= ROUND(SP*1000);
    OLDSP:= NEWSP;
    SETCOLORWHITE;
    DRAWBORDER;
    {DRAWSQUARE(0,NEWSP,ROUND(XMAXGLB/12),0,FILLING);}
    DRAWLINE(0,NEWSP,ROUND(XMAXGLB/12),NEWSP);
    GOTOXY(2,7);
    WRITE(SP:5:2);
    COPYSCREEN;
    SELECTSCREEN(1);
END;


PROCEDURE DRAWTIME;       {Plots System Clock Time}

BEGIN
   SELECTWINDOW(8);
   DRAWBORDER;
   GOTOXY(2,2);
   WRITE('TIME');
   GOTOXY(2,3);
   WRITE(T);
END;


PROCEDURE INITGRAF;       {Initialises Graphic Display}

VAR I : INTEGER;

BEGIN
  INITGRAPHIC;

  DEFINEWORLD(1,0,0,XMAXGLB,1000);
  DEFINEWINDOW(5,ROUND(XMAXGLB/9),0,XMAXGLB,(YMAXGLB DIV 2)-10);
  DEFINEWINDOW(6,ROUND(XMAXGLB/9),(YMAXGLB DIV 2)+10,XMAXGLB, YMAXGLB);

  DEFINEWORLD(2,0,0,ROUND(XMAXGLB/12),1000);
  DEFINEWINDOW(7,0,ROUND(YMAXGLB/4),ROUND(XMAXGLB/12),YMAXGLB);
  DEFINEWINDOW(8,0,4,ROUND(XMAXGLB/12),ROUND(YMAXGLB/8));

  SELECTWORLD(1);
  OLDX:= 0;
  OLDYVAL[1]:= 0;
  OLDYVAL[2]:= 0;
  FOR I:= 1 TO SIZE DO
     OLDY[I]:= 0;
  DRAWCOUNT:= 0;
  SELECTWINDOW(5);
  DRAWBORDER;
  SELECTWINDOW(6);
  DRAWBORDER;
END;

PROCEDURE DRAWOUTPUT(Y1, Y2 : REAL);     {Plots Traces of
```

```
                                                Variables Y1 and Y2)

VAR I : INTEGER;

BEGIN
    SELECTWORLD(1);
    IF ABS(Y1) > 1000 THEN
       NEWYVAL[1]:= ROUND(1000*Y1/ABS(Y1))
    ELSE
       NEWYVAL[1]:= ROUND(Y1);
    IF ABS(Y2) > 1000 THEN
       NEWYVAL[2]:= ROUND(1000*Y2/ABS(Y2))
    ELSE
       NEWYVAL[2]:= ROUND(Y2);

    IF DRAWCOUNT > XMAXGLB THEN
    BEGIN
      SELECTSCREEN(2);
      CLEARSCREEN;
      SELECTWINDOW(5);
      DRAWBORDER;
      SELECTWINDOW(6);
      DRAWBORDER;
      DRAWSP;
      COPYSCREEN;
      SELECTSCREEN(1);
      DRAWCOUNT:= 0;
      OLDX:= 0;
    END;

    NEWX:= DRAWCOUNT;
    SELECTWINDOW(5);
    DRAWLINE(OLDX,OLDYVAL[1],NEWX,NEWYVAL[1]);
    SELECTWINDOW(6);
    DRAWLINE(OLDX,OLDYVAL[2],NEWX,NEWYVAL[2]);
    OLDX:= NEWX;
    OLDYVAL[1]:= NEWYVAL[1]
    OLDYVAL[2]:= NEWYVAL[2]
    DRAWCOUNT:= DRAWCOUNT + 1;
END;


FUNCTION TIME : ST;       {Gets System Time from Real-Time Clock}
TYPE
  REGISTORS = RECORD
              AX,BX,CX,DX,BP,SI,DS,ES,FLAGS: INTEGER;
              END;
VAR
  REGISREC              : REGISTORS;
  HOUR , MINUTE , SECOND : STRING[2]
  CX , DX               : INTEGER;
BEGIN
  WITH REGISREC DO
  BEGIN
    AX := $2C SHL 8;
  END;
  MSDOS(REGISREC);
  WITH REGISREC DO
  BEGIN
    STR(CX SHR 8 , HOUR);
```

```
    STR(CX MOD 256 , MINUTE);
    STR(DX SHR 8 , SECOND);
  END;
  IF LENGTH(HOUR) = 1 THEN INSERT('0',HOUR,1);
  IF LENGTH(MINUTE) = 1 THEN INSERT('0',MINUTE,1);
  IF LENGTH(SECOND) = 1 THEN INSERT('0',SECOND,1);
  TIME := HOUR + ':' + MINUTE + ':' + SECOND;
END;


PROCEDURE RLS0;    {Initialisation of RLS Algorithm}

CONST NPAR = 4;
      NUVEC = 6;

VAR
    E0, E1 : REAL;
    J : INTEGER;

BEGIN
   YBAR1:= 0;
   BJ:= 1;
   FOR J:= 1 TO NPAR DO
   BEGIN
     K[J]:= 0;
     THETA[J]:= 0;
     PHIK1[J]:= 0;
     D[J]:= CTR2;
   END;
   FOR J:= 1 TO NUVEC DO
     UVEC[J]:= 0;
   E1:= 2*0.707*WE;
   E0:= WE*WE;
   EAVEC[1]:= -E1;
   EAVEC[2]:= -E0;
   GAM[1]:= GAMMA*E0;
   GAM[2]:= E0;
   GAM[3]:= GAMMA*E0;
   GAM[4]:= E0;
   ETH[1]:= E1/(GAMMA*E0);
   ETH[2]:= 1.0;
   ETH[3]:= 0.0;
   ETH[4]:= 0.0;
END;


PROCEDURE RLS1;   {Part 1 of RLS Algorithm to give Model Parameters}

CONST  NPAR = 4;

VAR  EE : REAL;
     J : INTEGER;

BEGIN
   EE:= E/BJ;
   FOR J:= 1 TO NPAR DO
      THETA[J]:= THETA[J] + EE*K[J]
   A1:= -THETA[1]*GAM[1]
   A0:= -THETA[2]*GAM[2]
   B1:= THETA[3]*GAM[3]
```

```
   BO:= THETA[4]*GAM[4]
END;

PROCEDURE RLS2(FLAG : BOOLEAN);  {Part 2 of RLS Algorithm to update
                                  Phi-vector, Covariance Matrix}

CONST  NPAR = 4;
       NA = 2;
       NUVEC = 6;

VAR  PHIK : THVEC;
     FJ, BJ1, GJ, MUJ, W, PY, PU : REAL;
     J, I, LF, LU : INTEGER;

BEGIN
   {Step 1 : Update PHIK1(K-1) to PHIK1(K), Becoming
      PHIK1(K-1) for next Iteration}

   PY:= YBAR;
   PU:= UBAR;
   FOR J:= 1 TO NA DO
   BEGIN
     PY:= PY + EAVEC[J]*PHIK1[J]
     PU:= PU + EAVEC[J]*PHIK1[NA+J]
   END;
   PY:= PHIK1[1] + DELTA*PY;
   PU:= PHIK1[NA+1] + DELTA*PU;
   FOR J:= NPAR DOWNTO 2 DO
     PHIK1[J]:= PHIK1[J] + DELTA*PHIK1[J-1]
   PHIK1[1]:= PY;
   PHIK1[NA+1]:= PU;

   {Step 2 : Apply Gamma Factors and store in "Local" Var PHIK}

   FOR J:= 1 TO NPAR DO
     PHIK[J]:= PHIK1[J]*GAM[J]

   {Step 3 : Calc YBAR1(K+1) (ie Predicted YBAR) for next Time -
       Assumes that THETA has Already been Updated to give THETA(K)}

   YBAR1:= 0;
   FOR J:= 1 TO NPAR DO
     YBAR1:= YBAR1 + PHIK[J]*(THETA[J] + ETH[J]);

   {Step 4 : RLS Algorithm., Uses PHIK[K] to Calculate the
      Estimation Gain Vector K[K] and BJ[K]. These are, However,
      only used in the next Iteration as K[K-1] and BJ[K-1].
      Bierman's U'*D*U Covariance Update Algorithm is used.}

   IF FLAG THEN
   BEGIN
     FJ:= PHIK[1]
     GJ:= D[1]*FJ;
     K[1]:= CJ;
     BJ:= (1 + GJ*FJ);
     D[1]:= D[1]/(BJ*LAMK);
     {Assume NPAR > 1 from here on}
     LF:= 0;
     LU:= 0;
```

```
      {F:= U'*PHI}

      FOR J:= 2 TO NPAR DO
      BEGIN
         FJ:= PHIK[J]
         FOR I:= 1 TO J-1 DO
         BEGIN
            LF:= LF + 1;
            FJ:= FJ + PHIK[I]*UVEC[LF]
         END,

         {G:= D*F}

         GJ:= FJ*D[J]
         K[J]:= GJ;
         BJ1:= BJ;
         BJ:= BJ + (GJ*FJ);
         D[J]:= (D[J]*BJ1)/(BJ*LAMK);
         MUJ:= -(FJ*AZONE)/BJ1;

         {Uj = Uj + MUj*Kj
          Kj+1 = Kj + GJ*UJ}

         FOR I:= 1 TO J-1 DO
         BEGIN
            LU:= LU + 1;
            W:= UVEC[LU] + K[I]*MUJ;
            K[I]:= K[I] + UVEC[LU]*GJ;
            UVEC[LU]:= W;
         END;
      END;
      BJ:= BJ/AZONE;
   END
   ELSE BEGIN
      BJ:= 1.0;
      FOR J:= 1 TO NPAR DO
         K[J]:= 0.0;
   END;
END;  {RLS2}


PROCEDURE WAIT(PORTNUM, BITVAL1, BITVAL2 : INTEGER);

{Waits for Signal at Input Port from A/D, D/A Hardware}

VAR PORTVAL : INTEGER;

BEGIN
  PORTVAL:= PORT[PORTNUM]
  PORTVAL:= (PORTVAL XOR BITVAL2) AND BITVAL1;
  WHILE PORTVAL = 0 DO
  BEGIN
     PORTVAL:= PORT[PORTNUM]
     PORTVAL:= (PORTVAL XOR BITVAL2) AND BITVAL1;
  END;
END;


PROCEDURE ERRORTRAP;
```

```
{Illegal Status Register on A/D ; D/A Hardware}

BEGIN
  WRITELN;
  WRITELN('FATAL ERROR - ILLEGAL STATUS REGISTER VALUE');
  WRITELN('STATUS REGISTER VALUE IS ',STATUS);
END;


PROCEDURE FATALERR;

{Fatal board error}

BEGIN
   WRITELN;
   WRITELN('FATAL BOARD ERROR');
   WRITELN('STATUS REGISTER VALUE IS ',STATUS);

   {Read the Error Register}

   PORT[COMMAND_REGISTER]:= CSTOP;
   TEMP:= PORT[DATA_REGISTER]

   WAIT(STATUS_REGISTER, COMMAND_WAIT, ZERO);
   PORT[COMMAND_REGISTER]:= CERROR;

   WAIT(STATUS_REGISTER, READ_WAIT, ZERO);
   ERROR1:= PORT[DATA_REGISTER]
   WAIT(STATUS_REGISTER, READ_WAIT, ZERO);
   ERROR2:= PORT[DATA_REGISTER]
   WRITELN('ERROR REGISTER VALUES ARE:');
   WRITELN('      BYTE 1 -   ',ERROR1);
   WRITELN('      BYTE 2 -   ',ERROR2);
END;


PROCEDURE INITBOARD;    {Initialise A/D; D/A Board}

BEGIN
   BASE_ADDRESS:= $2EC;
   COMMAND_REGISTER:= BASE_ADDRESS + 1;
   STATUS_REGISTER:= BASE_ADDRESS + 1;
   DATA_REGISTER:= BASE_ADDRESS;
   COMMAND_WAIT:= $4;
   WRITE_WAIT:= $2;
   READ_WAIT:= $5;
   CSTOP:= $F;
   CCLEAR:=$1;
   CERROR:= $2;
   CADIN:= $C;
   GAIN_CODE:= 0;
   RFACTOR:= 4096;
   RRANGE:= 20;
   ROFFSET:= 10;
   ERRFLAG:= FALSE;
   CDAOUT:= $8;
   EXT_TRIGGER:= $80;
   OFFSET[1]:= 5;       RANGE[1]:= 10;      FACTOR[1]:= 256;
   OFFSET[2]:= 0;       RANGE[2]:= 5;       FACTOR[2]:= 256;
```

```
   OFFSET[3]:= 10;      RANGE[3]:= 20;      FACTOR[3]:= 4096;
   OFFSET[4]:= 5;       RANGE[4]:= 10;      FACTOR[4]:= 4096;
   OFFSET[5]:= 2.5;     RANGE[5]:= 5;       FACTOR[5]:= 4096;
   OFFSET[6]:= 0;       RANGE[6]:= 10;      FACTOR[6]:= 4096;
   OFFSET[7]:= 0;       RANGE[7]:= 5;       FACTOR[7]:= 4096;
END;


PROCEDURE CLEARBOARD;

BEGIN
      {Stop and clear the DT2801}

      PORT[COMMAND_REGISTER]:= CSTOP;
      TEMP:= PORT[DATA_REGISTER]
      WAIT(STATUS_REGISTER, COMMAND_WAIT, ZERO);
      PORT[COMMAND_REGISTER]:= CCLEAR;
END;


PROCEDURE READVAL(VAR VOLTS : REAL; CHANNEL : INTEGER);

{Read a Voltage Value from A/D}

BEGIN

   {Check for legal Status Register}

   STATUS:= PORT[STATUS_REGISTER]
   IF NOT((STATUS AND $70) = 0) THEN
   BEGIN
     ERRORTRAP;
     ERRFLAG:= TRUE;
   END;

   IF NOT ERRFLAG THEN
   BEGIN

      { Stop and clear the DT2801 }

        {  Write READ A/D IMMEDIATE command }

        WAIT(STATUS_REGISTER, COMMAND_WAIT, ZERO);
        PORT[COMMAND_REGISTER]:= CADIN;

        {  Write A/D gain byte }

        WAIT(STATUS_REGISTER, WRITE_WAIT, WRITE_WAIT);
        PORT[DATA_REGISTER]:= GAIN_CODE;

        { Write A/D channel byte }

        WAIT(STATUS_REGISTER, WRITE_WAIT, WRITE_WAIT);
        PORT[DATA_REGISTER]:= CHANNEL;

        { Read two bytes of A/D data from the Data Out Register
          and combine the two bytes into one word }

        WAIT(STATUS_REGISTER, READ_WAIT, ZERO);
        LOW:= PORT[DATA_REGISTER]
```

```
        WAIT(STATUS_REGISTER, READ_WAIT, ZERO);
        HIGH:= PORT[DATA_REGISTER]
        DATA_VALUE:= HIGH * 256 + LOW;

        IF DATA_VALUE > 32767 THEN DATA_VALUE:= DATA_VALUE - 6.5536E+04;

        { Check for ERROR }

        WAIT(STATUS_REGISTER, COMMAND_WAIT, ZERO);
        STATUS:= PORT[STATUS_REGISTER]
        IF (STATUS AND $80) <> 0 THEN
        BEGIN
           FATALERR;
           ERRFLAG:= TRUE;
        END;

        { Calculate the A/D reading in volts }

        VOLTS:= ((RRANGE * DATA_VALUE/RFACTOR) - ROFFSET);
   END;
END;


PROCEDURE WRITEVAL(VOUT : REAL);

{Write a Voltage Value to D/A}

BEGIN

   { Check for legal Status Register }

   STATUS:= PORT[STATUS_REGISTER]
   IF NOT((STATUS AND $70) = 0) THEN
   BEGIN
     ERRORTRAP;
     ERRFLAG:= TRUE;
   END;

   IF NOT ERRFLAG THEN
   BEGIN
      VOUT:= VOUT + 0.03;

      IF ((VOUT > (RANGE[3]-OFFSET[3])) OR (VOUT < -OFFSET[3])) THEN
          VOUT:= (ABS(VOUT)/VOUT)*OFFSET[3]

      DATA_VALUE:= (VOUT + OFFSET[3])*FACTOR[3]
      INTVAL:= ROUND(DATA_VALUE / RANGE[3]);

      IF INTVAL > ROUND(FACTOR[3] - 1) THEN
           INTVAL:= ROUND(FACTOR[3] - 1);

        { Write WRITE D/A IMMEDIATE command }

        WAIT(STATUS_REGISTER, COMMAND_WAIT, ZERO);
        PORT[COMMAND_REGISTER]:= CDAOUT;

        { Write D/A SELECT byte }

        WAIT(STATUS_REGISTER, WRITE_WAIT, WRITE_WAIT);
        PORT[DATA_REGISTER]:= 1;
```

```
        { Write HIGH & LOW bytes of DATA_VALUE }

        HIGH:= INTVAL DIV 256;
        LOW:= INTVAL - HIGH*256;
        WAIT(STATUS_REGISTER, WRITE_WAIT, WRITE_WAIT);
        PORT[DATA_REGISTER]:= LOW;
        WAIT(STATUS_REGISTER, WRITE_WAIT, WRITE_WAIT);
        PORT[DATA_REGISTER]:= HIGH;

        {  Check for ERROR }

        WAIT(STATUS_REGISTER, COMMAND_WAIT, ZERO);
        STATUS:= PORT[STATUS_REGISTER]
        IF (STATUS AND $80) <> 0 THEN
        BEGIN
           FATALERR;
           ERRFLAG:= TRUE;
        END;
   END;
END;


PROCEDURE INITIAL;  {General System Initialisation}

BEGIN
   {Initialise Parameter Estimation}

   RLS0;
   A0:= 0;
   B0:= 0;
   A1:= 0;
   B1:= 0;
   YHAT:= 0;

   {Initialise Controller States, etc.}

   AUTO:= FALSE;
   YEF:= 0;
   CDF1:= 0;
   CK1:= 0;
   CK2:= 1;
   CK3:= 0;

   {Initialise Deadzone}

   SIG1:= 1 - SIGMA;
   MZ0:= EPS0;
   AZONE:= 1;

   {Initialise Pole Positioning Vector}

   ASTAR[1]:= 1.0;
   ASTAR[2]:= 3.8E-02;  {5.965E-02;}
   ASTAR[3]:= 5.2E-04;  {9.363E-04;}
   ASTAR[4]:= 3.5E-06;  {4.909E-06;}
   ASTAR[5]:= 5.1E-09;  {4.086E-09;}

   {Initialise Plant Signals and Discrete State Variables}
```

```
    U:= 0;
    READVAL(YINIT,0);
    YINIT:= -YINIT/YSPAN;
    XHP1:= YINIT/EPSHP;
    XHP2:= SPS1/EPSHP;

    {Initialise System Clock for A/D Timing}

    SYSTIME:= TIME;
    VAL(COPY(SYSTIME,1,2),HR,CD);
    VAL(COPY(SYSTIME,4,2),MIN,CD);
    VAL(COPY(SYSTIME,7,2),SEC,CD);
    NEWTIME:= 3600*HR + 60*MIN + SEC;
    OLDTIME:= NEWTIME;
    INITTIME:= NEWTIME;
    T:= 0;

    {Initialise Setpoint Display}

    SP:= 0;
    OLDSP:= 0;
    FLASH:= TRUE;
    DRAWSP;
    CONSTAT:= 'MANUAL';

END;


PROCEDURE ADAP1;

{First Part of Adaptive Controller Calculation to give
                    Control Signal U}

    FUNCTION DEAD(LOWER, UPPER, INVAL : REAL) : REAL;

    {Relative Deadzone Function}

    BEGIN
      IF INVAL <= LOWER THEN
         DEAD:= INVAL - LOWER
      ELSE BEGIN
         IF INVAL > UPPER THEN
            DEAD:= INVAL - UPPER
         ELSE
            DEAD:= 0.0;
      END;
    END;  {DEAD}

    FUNCTION STEP(TIMEVAL : INTEGER) : REAL;

    {Generates a Unit Step at the Specified Time}

    BEGIN
       IF T >= TIMEVAL THEN
          STEP:= 1
       ELSE
          STEP:= 0;
    END;  {STEP}

BEGIN
```

```
{High Pass Filter Measurement}

YBAR:= Y - EPSHP*XHP1;
XHP1:= XHP1 + DELTA*YBAR;

{Calculate Prediction Error}

YHAT:= YBAR1;
E:= YBAR - YHAT;

{Update the Parameter Vector}

MZ:= MZ0,
BMZ:= BETA*MZ;
RFLAG:= FALSE;
IF (ABS(E) > BMZ) THEN
BEGIN
  AZONE:= (ALPHA/E)*DEAD(-BMZ, BMZ, E);
  RLS1;
  RFLAG:= TRUE;
END;

{Generate Dr..ing Signals}

IF NOT AUTO THEN
BEGIN
  SP:= SI   *(1 - STEP(T1) + STEP(T2) - STEP(T3) + STEP(T4)
  - STEP(T5) + STEP(T6) - STEP(T7) + 0.6*STEP(T8)
  - 0.2*STEP(T9) + 0.2*STEP(T10) - 0.2*STEP(T11) + 0.2*STEP(T12)
  -0.2*STEP(T13) + 0.2*STEP(T14) - 0.2*STEP(T15) + 0.2*STEP(T16)
  -0.2*STEP(T17) + 0.2*STEP(T18));
  DRAWSP;
END;
YE:= (SP - Y);
IF T >= TAUTO THEN
BEGIN
   AUTO:= TRUE;
   CONSTAT:= 'AUTO   ';
END
ELSE
   CONSTAT:= 'MANUAL';
IF AUTO THEN
BEGIN
   {Controller on AUTO - First Synthesise Controller}

   CP0:= ASTAR[5]/B0;
   IF ABS(B0/B1) < 2*WE THEN
   BEGIN
      CL0:= (ASTAR[3]-(B0*ASTAR[2])/B1-(B1*ASTAR[4])/B0
            -(A0-(B0*A1)/B1)+(B1*CP0*B1)/B0)/(A1-B0/B1-(B1*A0)/B0);
      CP1:= (ASTAR[4]-A0*CL0-B1*CP0)/B0;
      CP2:= (ASTAR[2]-A1-CL0)/B1;
   END
   ELSE BEGIN
      CL0:= ASTAR[2] - A1;
      CP1:= (ASTAR[4] - A0*CL0)/B0;
      CP2:= (ASTAR[3] - A0 - A1*CL0)/B0;
   END;

   {Convert Controller to PID Form}
```

```
      CK1:= CP1/CL0;
      CK2:= CP1/CP0;
      CK3:= CP2/CP1;

      {Calc. Control Input to Plant, Using Incremental PID
       Algorithm With First Order L.P. Filter on all Terms.
       A Backward Difference Approximation to the Continuous-
       Time Equation is Used}

      CDF:= CFAC*(YE-YEF);     {Filtered (1-Q**-1)*YE}
      YEF:= YEF + CDF;         {Filtered YE}
      U:= U + CK1*(CDF + (DELTA*YEF)/CK2 + CK3*(CDF-CDF1)/DELTA);
      CDF1:= CDF;
      IF U > UMAX THEN
         U:= UMAX
      ELSE IF U < UMIN THEN
         U:= UMIN;
   END
   ELSE
      {Controller on Manual}

      U:= SP*MANU;      {Scaled to give Reasonable S-S Output}
END;  {ADAP1}


PROCEDURE ADAP2;

{Second Part of Adaptive Controller Calculation to Update
                  Remaining Variables}

BEGIN
   {High-Pass Filtering of U}

   UBAR:= -EPSHP*XHP2 + U;
   XHP2:= XHP2 + DELTA*UBAR;

   {Update Deadzone Function}

   MZO:= SIGMA*MZ+SIG1*(EPS0+EPS1*ABS(UBAR)+EPS2*ABS(Y));

   {Update Parameter Estimates}

   RLS2(RFLAG);
END;


PROCEDURE DELAYPLANT;

{Introduces a Delay of DELAYTIME Samples into Plant Output
                     Measurement}

CONST DELAYTIME = 3;

VAR I : INTEGER;
    DUMMY : REAL;

BEGIN
   IF T > T13 THEN
   BEGIN
```

```
      DUMMY:= Y;
      Y:= YVEC[DELAYTIME];
      FOR I:= 1 TO DELAYTIME - 1 DO
         YVEC[I+1]:= YVEC[I];
      YVEC[1]:= DUMMY;
   END
   ELSE BEGIN
      FOR I:= 1 TO DELAYTIME - 1 DO
         YVEC[I]:= Y;
   END;
END;


PROCEDURE SAMPLE;

{Main Adaptive Controller Calculation Performed at each Sampling
                              Time}

BEGIN
   {Read Plant Output from A/D}

   READVAL(Y,0);
   Y:= -Y/YSPAN;
   {DELAYPLANT;}

   {Perform First Part of Adaptive Calculation}

   ADAP1;

   {Write Control Signal to D/A}

   UOUT:= (U - UZERO)*USPAN;
   WRITEVAL(UOUT);

   {Graphic Updating, Remaining part of Adaptive Algorithm Update,
                     Data Logging to Diskfile}

   GOTOXY(12,13);
   WRITELN('A0 : ',A0:7,'    A1 : ',A1:7,'    B0 : ',B0:7,
                                      '    B1 : ',B1:7,'    ');
   DRAWOUTPUT(500*(SP + 0.5),1000*Y);
   ADAP2;
   OLDTIME:= NEWTIME;
   WRITELN(OUTFILE,T:5,' ',SP:4:1,' ',U:6:3,' ',Y:6:3,' ',AZONE:7:5,
           ' ',E:7:4,' ',CK1:7:3,' ',CK2:7:2,' ',CK3:7:2,' ',BMZ:9:7);
   WRITELN(OUTPARMS,A0:10:7,' ',B0:10:7,' ',A1:10:7,' ',B1:10:7);
END;


BEGIN  {Main Program}

   {Initialisation}

   INITGRAF;
   INITBOARD;
   CLEARBOARD;
   INITIAL;
   ASSIGN(OUTFILE,'ADAPT3.DAT');
   ASSIGN(OUTPARMS,'ADAPT3PARMS.DAT');
   REWRITE(OUTFILE);
```

```
REWRIT    TPARMS);
EXIT:= FALSE;
DRAWTIME;
SAMPLE;

REPEAT
   {Execute Adaptive Control at each Sampling Instant}

   IF (NEWTIME - OLDTIME = DELTA) THEN SAMPLE;

   {Get New System Time}

   SYSTIME:= TIME;
   VAL(COPY(SYSTIME,1,2),HR,CD);
   VAL(COPY(SYSTIME,4,2),MIN,CD);
   VAL(COPY(SYSTIME,7,2),SEC,CD);
   NEWTIME:= 3600*HR + 60*MIN + SEC;
   T:= ROUND(NEWTIME - INITTIME);
   IF T <> TOLD THEN

   {Update Graphic Display Every Second}

   BEGIN
      DRAWTIME;
      TOLD:= T;
      SELECTWINDOW(7);
      GOTOXY(2,5);
      IF FLASH THEN
      BEGIN
         FLASH:= NOT FLASH;
         WRITE(CU STAT);
      END
      ELSE BEGIN
         WRITE('       ');
         FLASH:= NOT FLASH;
      END;
   END;
   IF NEWTIME = 0 THEN OLDTIME:= -DELTA;

   {Read Keyboard for User Input}

   IF KEYPRESSED THEN
   BEGIN
      READ(KBD,CH);
      CASE CH OF
         '+' : BEGIN   (Increase Setpoint)
                  IF AUTO THEN
                  BEGIN
                    SP:= SP + 0.01;
                    IF SP > SPMAX THEN
                      SP:= SPMAX;
                    DRAWSP;
                  END;
               END;
         '-' : BEGIN   (Decrease Setpoint)
                  IF AUTO THEN
                  BEGIN
                    SP:= SP - 0.01;
                    IF SP < SPMIN THEN
                      SP:= SPMIN;
```

```
                    DRAWSP;
                  END;
                END;
            'E' : EXIT:= TRUE;   {Terminate Program}
            'e' : EXIT:= TRUE;   {Terminate Program}
          END;
      END;
  UNTIL (EXIT) OR (  > INT(TSTOP));
  CLOSE(OUTFILE);
  CLOSE(OUTPARMS);
  LEAVEGRAPHIC;
  WRITEVAL(-5);
  {WRITELN('A0: ',A0:7:5,'   B0: ',B0:7:5,'   A1: ',A1:7:5,'   B1: ',B1:7:5);}
  GOTOXY(16,10);
  WRITELN('RUN SUCCESSFULLY COMPLETED AT TIME = ',T:5,' SECONDS');
  GOTOXY(20,13);
  WRITELN('MACHINE MAY BE SWITCHED OFF AT THIS POINT');
  GOTOXY(1,22);
  WRITELN;
END.   {Adaptive Controller}
```

14.0 REFERENCES

1. Allidina A.Y. and Hughes F.M. *Generalised Self-Tuning Controller with Pole Assignment* IEE Proceedings Vol. 127 Part D No. 1, 13 - 18 (1980)

2. Allidina A.Y., Hughes F.M. and Tye C. *Self-Tuning Control for Systems employing Feedforward* IEE Proceedings Vol. 128 Part D No. 6, 283 - 291 (1981)

3. Allidina A.Y. and Yin H. *Explicit pole-assignment self-tuning algorithms* Int.J.Control Vol 42 No. 5, 1113 - 1130 (1985)

4. Anderson B.D.O. *Adaptive Systems, Lack of Persistency of Excitation & Bursting Phenomena* Automatica Vol. 21 No. 3, 247 - 258 (1985)

5. Anderson B.D.O. and Johnstone R. McG. *Global Adaptive Pole Positioning* IEEE Transactions on Automatic Control Vol AC-30 No. 1, 11 - 21 (1985)

6. Andreiev N. *A New Dimension : A Self-Tuning Controller that continually optimizes PID Constants* Control Engineering, 84 - 85 (1981)

7. Astrom K.J. *Robustness of a Design Method based on Assignment of Poles and Zeros* IEEE Transaction on Automatic Control Vol. AC-27 No. 3, 588 - 590 (1980)

8. Astrom K.J. *Theory and Applications of Adaptive Control - A Survey* Automatica Vol. 19 No. 5, 471 - 486 (1983)

9. Astrom K.J. *Adaptive Feedback Control* IEEE Proceedings Vol. 75 No. 2, 185 - 215 (1987)

10. Astrom K.J., Anton J.J. and Arzen K.E. *Expert Control* Automatica Vol. 22 No. 3, 277 - 286 (1986)

11. Astrom K.J., Borisson U., Ljung L. and Wittenmark B. *Theory and Applications of Self-Tuning Regulators* Automatica Vol. 13, 457 - 476 (1977)

12. Astrom K.J., Hagander P. and Sternby J. *Zeros of Sampled Systems* Automatica Vol. 20 No. 1, 31 - 38 (1984)

13. Astrom K.J. and Wittenmark B. *Self-Tuning Controllers based on Pole-Zero Placement* IEE Proceedings Vol. 127 Part D No. 3, 120 - 130 (1980)

14. Astrom K.J. and Wittenmark B. *Computer Controlled Systems - Theory and Design* Prentice-Hall (1984)

15. Bierman G.J. *Factorization Methods for Discrete Sequential Estimation* Academic Press (1977)

16. Boyd S. and Sastry S.S. *Necessary and Sufficient Conditions for Parameter Convergence in Adaptive Control* Automatica Vol. 22 No. 6, 629 - 639 (1986)

17. Clarke D.W. *Tutorial Lecture Notes - Adaptive Control* IEE - Department of Engineering Science, University of Oxford, U.K.

18. Clarke D.W. *Self-Tuning Control of Nonminimum-Phase Systems* Automatica Vol. 20 No. 5, 501 - 517 (1984)

19. Clarke D.W. and Gawthrop P.J. *Self-Tuning Control* IEE Proceedings Vol. 126 No. 6, 635 - 639 (1979)

20. Cameron F. and Seborg D.E. *A Self-Tuning Controller with a PID Structure* Int.J.Control Vol. 38 No. 2, 401 - 417 (1983)

21. Davies W.D.T. *System Identification for Self-Adaptive Control* John Wiley & Sons (1970)

22. De Larminat P.H. *On the Stabilizability condition in Indirect Adaptive Control* Automatica Vol. 20 No. 6, 793 - 795 (1984)

23. Elliott H. *Direct Adaptive Pole Placement with Application to Nonminimum-Phase Systems* IEEE Transactions on Automatic Control Vol AC-27 No. 3, 720 - 721 (1982)

24. Elliott H. and Wolovich W.A. *Parameter Adaptive Identification and Control* IEEE Transactions on Automatic Control Vol. AC-24 No. 4, 592 - 598 (1979)

25. Fjeld M. and Wilhelm R.G. *Self-Tuning Regulators - the Software Way* Control Engineering, 99 - 102 (1981)

26. Fortescue T.R., Kershenbaum L.S. and Ydstie B.E. *Implementation of Self-Tuning Regulators with Variable Forgetting Factors* Automatica Vol. 17 No. 6, 831 - 835 (1981)

27. Gawthrop P.J. *Some Interpretations of the Self-Tuning Controller* IEE Proceedings Vol. 124 No. 10, 889 - 894 (1977)

28. GawthropP.J. *Hybrid Self-Tuning Control* IEE Proceedings Vol. 127 Part D No. 5, 229 - 236 (1980)

29. Gawthrop P.J. and Lim K.W. *Robustness of Self-Tuning Controllers* IEE Proceed. Vol. 129 Part D No. 1, 21 - 29 (1982)

30. Goodwin G.C., Hill D.J. and Palaniswami M. *A Perspective on Convergence of Adaptive Control Algorithms* Automatica Vol. 20 No. 5, 519 - 531 (1984)

31. Goodwin G.C., Leal R.L., Maynes D.Q. and Middleton R.H. *Rapprochement between Continuous and Discrete Model Reference Adaptive Control* Automatica Vol. 22 No. 2, 199 - 207 (1986)

32. Goodwin G.C. and Mayne D.Q. *A Parameter Estimation Perspective of Continuous-Time Model Reference Adaptive Control* Automatica Vol. 23 No. 1, 57 - 70 (1987)

33. Goodwin G.C. and Sin K.S. *Adaptive Control of Nonminimum-Phase Systems* IEEE Transactions on Automatic Control Vol AC-26 No. 2, 478 - 485 (1981)

34. Goodwin G.C. and Sin K.S. *Adaptive Filtering, Prediction and Control* Prentice-Hall (1984)

35. Isermann R. *Digital Control Systems* Springer-Verlag (1981)

36. Isermann R. *Parameter-Adaptive Control Algorithms - A Tutorial* Automatica Vol. 18 No. 5, 513 - 528 (1982)

37. Jacquot R.G. *Modern Digital Control Systems* Marcel Dekker Inc. (1981)

38. Junkins J.L. *An Introduction to Optimal Estimation of Dynamical Systems* Sijthoff and Noordhoff (1978)

39. Kailath T. *Linear Systems* Prentice-Hall (1980)

40. Katz P. *Digital Control using Microprocessors* Prentice-Hall (1981)

41. Khalil H.K. *On the Robustness of Output Feedback Control Methods to Modelling Errors* IEEE Transactions on Automatic Control Vol. AC-26 No. 2, 524 - 526 (1981)

42. Kraft L.G. (III) *A Control Structure using an Adaptive Observer* IEEE Transactions on Automatic Control Vol. AC-24 No. 5, 804 - 806 (1979)

43. Kraus T.W. and Myron T.J. *Self Tuning Controller uses Pattern Recognition Approach* Control Engineering, 106 - 111 (1984)

44. Kreisselmeier G. *On Adaptive State Regulation* IEEE Transactions on Automatic Control Vol. AC-27 No. 1, 3 - 16 (1982)

45. Kreisselmeier G. and Anderson B.D.O. *Robust Model Reference Adaptive Control* IEEE Transactions on Automatic Control Vol. AC-31 No. 2, 127 - 132 (1986)

46. Kreisselmeier G. and Narendra K. *Stable Model Reference Adaptive Control in the Presence of Bounded Disturbances* IEEE Transactions on Automatic Control Vol. AC-27 No. 6, 1169 - 1175 (1982)

47. Kumar R. and Moore J.B. *Detection Techniques in Least Squares Identification* Automatica Vol. 17 No. 6, 805 - 818 (1981)

48. [illegible] R. and Moore J.B. *On Adaptive Minimum-Variance Regulation for [illegible]num-Phase Plants* Automatica Vol. 19 No. 4, 449 - 451 (1983)

49. Kuo B.C. *Digital Control Systems* Holt, Rinehart & Winston (1980)

50. Kurz H., Isermann R. and Schumann R. *Experimental Comparison and Application of Various Parameter-Adaptive Control Algorithms* Automatica Vol. 16, 117 - 133 (1980)

51. Landau Y.D. *Adaptive Control - The Model Reference Approach* Marcel Dekker (1979)

52. Latawiec K. and Chyra M. *On Low-Frequency and Long-Run Effects in Self-Tuning Control* Automatica Vol. 19 No. 4, 419 - 424 (1983)

53. Lin C.M. and Chen B.S. *Adaptive Controller with Desired Pole/Zero Assignment* IEE Proceedings Vol. 133 Part D No. 6, 301 - 3[illegible] (1986)

54. Ljung L. *System Identification : Theory for the User* Prentice-Hall (1987)

55. Ljung L. and Soderstrom T. *Theory and Practice of Recursive Identification* MIT Press (1983)

56. Macleod I.M. *State Space Modelling and Control - Course Notes* (1987)

57. Mc Dermott P.E. and Mellichamp D.A. *An Auto-Pole-Placement Self-Tuning Controller* Int.J.Control Vol. 40 No. 6, 1131 - 1147 (1984)

58. Middleton R.H. and Goodwin G.C. *Improved Finite Word Length Characteristics in Digital Control using Delta Operators* IEEE Transactions on Automatic Control Vol. AC-31 No. 11, 1015 - 1021 (1986)

59. Moore J.B. and Casalino G. *On Robustness to Noise of Least-Squares Based Adaptive Control* Automatica Vol. 23 No. 2, 203 - 208 (1987)

60. Morse A.S. *Global Stability of Parameter-Adaptive Control Systems* IEEE Transactions on Automatic Control Vol. AC-25 No. 3, 433 - 439 (1980)

61. M'Saad M., Ortega R. and Landau Y.D. *Adaptive Controllers for Discrete-Time Systems with Arbitrary Zeros: An Overview* Automatica Vol. 21 No. 4, 413 - 423 (1985)

62. O'Reilly J. *Observers for Linear Systems* Academic Press (1983)

63. Ortega R. and Kelly R. *PID Self-Tuners: some Theoretical and Practical Aspects* IEEE Transactions on Industrial Electronics Vol. IE-31 No. 4, 332 - 337 (1984)

64. Ortega R. and Lear R.L. *A Note on Direct Adaptive Control of Systems with Bounded Disturbances* Automatica Vol. 23 No. 2, 253 - 254 (1987)

65. Ortega R., Praly L. and Landau Y.D. *Robustness of Discrete-Time Direct Adaptive Controllers* IEEE Transactions on Automatic Control Vol. AC-30 No. 12, 1179 - 1187 (1985)

66. Phillips C.L. and Huang C.T. *Bode Design of Digital Controllers* IEEE Transactions on Education Vol. E-26 No. 2, 70 - 71 (1983)

67. Phillips C.L. and Nagle H.T. *Digital Control System Analysis & Design* Prentice-Hall (1984)

68. Phillips C.L. and Parr J.M. *Robust Design of a Digital PID Predictor-Controller* IEEE Transactions on Industrial Electronics Vol. IE-31 No. 4, 328 - 332 (1984)

69. Puthenpura S.C. and MacGregor J.F. *Pole-Zero Placement Controllers and Self-Tuning Regulators with better Set-Point Tracking* IEE Proceedings Vol. 134 Part D No. 1, 26 - 30 (1987)

70. Ralston A. *A First Course in Numerical Analysis* McGraw-Hill (1965)

71. Samson C. and Fuchs J-J *Discrete Adaptive Regulation of Not-Necessarily Minimum-Phase Systems* IEE Proceedings Vol. 128 Part D No. 3, 102 - 105 (1981)

72. Seborg D.E., Edgar T.F. and Shah S.L. *Adaptive Control Strategies for Process Control: A Survey* AIChE Journal Vol. 32 No. 6, 881 - 913 (1986)

73. Silveira H.M. and Doraiswami R. *New Structure for an Adaptive Servomechanism Controller* IEE Proceedings Vol. 131 Part D No. 2, 64 - 67 (1984)

74. Song H.K., Shah S.L. and Fisher D.G. *A Self-Tuning Robust Controller* Automatica Vol. 22 No. 5, 521 - 531 (1986)

75. Strejc V. *Least Squares Parameter Estimation* Automatica Vol. 16, 535 - 550 (1980)

76. Strejc V. *State-Space Theory of Discrete Linear Control* Academia (Prague, 1981)

77. Tan C.I. and McInnis B.C. *Adaptive Digital Control Implemented Using Residue Number Systems* IEEE Transactions on Automatic Control Vol. AC-27 No. 2, 499 - 502 (1982)

78 Tsay Y.T. and Shieh L.S. *State-Space Approach for Self-Tuning Feedback Control with Pole Assignment* IEE Proceedings Vol. 128 Part D No. 3, 93 - 101 (1981)

79. Tuffs P.S. and Clarke D.W. *Self-Tuning Control of Offset: A Unified Approach* IEE Proceedings Vol. 132 Part D No. 3, 100 - 109 (1985)

80. Wellstead P.E., Edmunds J.M., Prager D. and Zanker P. *Self-Tuning Pole/Zero Assignment Regulators* Int.J.Control Vol. 30 No. 1, 1 - 26 (1979)

81. Wellstead P.E., Prager D. and Zanker P. *Pole Assignment Self-Tuning Regulator* IEE Proceedings Vol. 126 Part D No. 8, 781 - 787 (1979)

82. Wellstead P.E. and Sanoff S.P. *Extended Self-Tuning Algorithm* Int.J.Control Vol. 34 No. 4, 433 - 455 (1981)

83. Wittenmark B. and Astrom K.J. *Practical Issues in the Implementation of Self-Tuning Control* Automatica Vol. 20 No. 5, 595 - 605 (1984)

84. Xianya X. and Evans R.J. *Discrete-Time Adaptive Control for Deterministic Time-Varying Systems* Automatica Vol. 20 No. 3, 309 - 319 (1984)

77. Tan C.I. and McInnis B.C. *Adaptive Digital Control Implemented Using Residue Number Systems* IEEE Transactions on Automatic Control Vol. AC-27 No. 2, 499 - 502 (1982)

78. Tsay Y.T. and Shieh L.S. *State-Space Approach for Self-Tuning Feedback Control with Pole Assignment* IEE Proceedings Vol. 128 Part D No. 3, 93 - 101 (1981)

79. Tuffs P.S. and Clarke D.W. *Self-Tuning Control of Offset: A Unified Approach* IFE Proceedings Vol. 132 Part D No. 3, 100 - 109 (1985)

80. Wellstead P.E., Edmunds J.M., Prager D. and Zanker P. *Self-Tuning Pole/Zero Assignment Regulators* Int.J.Control Vol. 30 No. 1, 1 - 26 (1979)

81. Wellstead P.E., Prager D. and Zanker P. *Pole Assignment Self-Tuning Regulator* IEE Proceedings Vol. 126 Part D No. 8, 781 - 787 (1979)

82. Wellstead P.E. and Sanoff S.P. *Extended Self-Tuning Algorithm* Int.J.Control Vol. 34 No. 3, 433 - 455 (1981)

83. Wittenmark B. and Astrom K.J. *Practical Issues in the Implementation of Self-Tuning Control* Automatica Vol. 20 No. 5, 595 - 605 (1984)

84. Xianya X. and Evans R.J. *Discrete-Time Adaptive Control for Deterministic Time-Varying Systems* Automatica Vol. 20 No. 3, 309 - 319 (1984)

Author Bergesen Michel Ludvic
Name of thesis The Implementation Of A Generalised Robust Adaptive Controller. 1988

PUBLISHER:
University of the Witwatersrand, Johannesburg
©2013

LEGAL NOTICES:

Copyright Notice: All materials on the University of the Witwatersrand, Johannesburg Library website are protected by South African copyright law and may not be distributed, transmitted, displayed, or otherwise published in any format, without the prior written permission of the copyright owner.

Disclaimer and Terms of Use: Provided that you maintain all copyright and other notices contained therein, you may download material (one machine readable copy and one print copy per page) for your personal and/or educational non-commercial use only.

The University of the Witwatersrand, Johannesburg, is not responsible for any errors or omissions and excludes any and all liability for any errors in or omissions from the information on the Library website.