NON PARAMETRIC MODEL REFERENCE ADAPTIVE CONTROL

BY

Hien Minh Thi TRAN, B.E. (Hons.)

Submitted in fulfillment of the requirements
for the degree of the
Master of Engineering Science

University of Tasmania
Hobart
1989

I declare that this thesis does not contain anything that has been accepted for the award of a degree or diploma in any other university, and also that, to the best of my knowledge and belief, the thesis does not contain a copy or a paraphrase of material written and/or published by anyone else except where due reference to it has been made in the text of the thesis.

Hien Minh Thi TRAN, B.E.(Hons.)

This thesis is dedicated to the memory of my beloved mother

**The road is not always to the swift,
but to those who keep on running.**

Chapter 1 INTRODUCTION. 1

1.1 CONTRIBUTIONS. 1

1.2 ADAPTIVE CONTROL SYSTEMS. 1

1.2.1 The self-tuning adaptive control (STAC). 2

1.2.2 The model reference adaptive control(MRAC). 3

1.3 A NON-PARAMETRIC MODEL-REFERENCE ADAPTIVE CONTROL . . . 5
(NP-MRAC)

1.4 THESIS OUTLINE. 6

CHAPTER 2 THE LEAST MEAN SQUARE ALGORITHM 8

2.1 DERIVATION OF THE LMS ALGORITHM. 8

2.1.1 The performance function. 9

2.1.2 Gradient search and the optimal weight vector 12

2.1.3 Gradient search by the method of steepest descent. 13

2.2 THE LMS ALGORITHM FOR PLANT IDENTIFICATION 16

2.2.1 The method. 17

2.2.2 Convergence of the weight vector. 20

2.2.3 The weight vector of the FIR adaptive filter in plant 21
identification

2.3 THE FILTERED-X LMS ALGORITHM FOR AN OPEN LOOP MODEL REFERENCE ADAPTIVE CONTROL. 22

2.3.1 The filtered-X LMS algorithm. 23

2.3.2 CONVERGENCE OF THE WEIGHT VECTOR. 26

CHAPTER 3 NP-PI AND NP-MRAC SIMULATIONS . . . . . 35 ON THE NEC-APC3 COMPUTER

3.1 TEST INPUT SIGNALS. 35

3.2 A NON-PARAMETRIC PLANT IDENTIFICATION (NP-PI). 37

3.2.1 Plant models. 38

3.2.2 Simulations of the NP-PI. 41

3.3 OPEN LOOP NON-PARAMETRIC MODEL-REFERENCE ADAPTIVE CONTROL (NP-MRAC). 48
CHAPTER 4 CLOSED LOOP NON PARAMETRIC - MODEL REFERENCE ADAPTIVE CONTROL 55
4.1 THE CLOSED LOOP NP-MRAC. 55
4.2 TRACKING ABILITY OF THE CLOSED LOOP NP-MRAC TO THE PLANT WITH BOTH PARAMETER AND TIME DELAY VARIATIONS. 57
A - The ability to track plant with parameter variation. 57
B-The ability to track plant with both parameter and time delay variations 61
4.3 SET POINT TRACKING PROPERTY. 65
4.4 THE NP-MRAC TO CONTROL A PLANT WHICH INCLUDES INTEGRATORS. 68
Chapter 5 IMPLEMENTATION OF A CLOSED-LOOP NP-MRAC ON THE TMS320C20. 73
5.1 INTRODUCTION TO TMS320C20 SIGNAL PROCESSING CHIP. 74
5.1.1 Flexibility. 75
5.1.2 Internal hardware. 75
5.1.3 Auxiliary Registers with their own arithmetic unit 76
5.1.4 Instruction set support for Floating-point operations. . . . 77
5.1.5 System Control 78
5.2 A MICROPROCESSOR BLOCK DIAGRAM FOR THE NP-MRAC 78
5.3 CONTROLLER PROGRAM. 80
5.4 TESTING THE SYSTEM WITH A PLANT IMPLEMENTED ON THE ANALOG COMPUTER. 85
5.5 SOFTWARE DESCRIPTION. 92
5.5.1 Initialization routine. 92
5.5.2 Floating Point Conversion Routine (FXFL routine). 93
5.5.3 Floating point addition and multiplication routines. 99
5.5.4 Plant Identification Routine (PIDTFY routine). 99

5.5.5 LMS controller Routine (FIR controller). 105
5.5.6 Fixed-point conversion Routine [FLFX routine]. 113
5.5.7 Reference Model Routine (MODEL routine) 116
Chapter 6 CONCLUSIONS AND FUTURE WORK. 117
6.1 CONCLUSIONS. 117
6.2 A NP-MRAC USING RECURSIVE LEAST SQUARE ALGORITHM (RLS)
. 120
6.3 FUTURE WORK. 123
APPENDIX. 124

SUMMARY

This thesis introduces a new method of adaptive control: Non-Parametric Model-Reference Adaptive-Control (NP-MRAC). The method relies on estimating the shape of the Finite Impulse Response (FIR) of the plant and the FIR of the controller rather than determining the plant and the controller parameters. An FIR adaptive filter was used to identify the FIR of the plant and calculate the FIR of the controller. The NP-MRAC was studied both by digital simulation on the NEC-APC-3 computer and by real-time hybrid-simulation on the TMS320C20 signal processor.

In this thesis, the NP-MRAC employed the LMS algorithm as an adaptive algorithm. This algorithm is simple, yet performs satisfactorily. The NP-MRAC, using the LMS algorithm, has excellent adaption capabilities in the presence of both parameter variations and time delay variations. It also allows for closed-loop pole placement and because there are no restrictions on closed-loop zeros, good set-point tracking can be achieved by a suitable choice of the reference-model.

The scheme's major limitations are relatively slow rate of convergence and sensitivity to variations in the eigenvalue spread, defined as the ratio of the maximum to minimum eigenvalue of the correlation matrix of the FIR adaptive filter input signal. However the NP-MRAC should have great potential in applications where time-delay and parameter variations are relatively gradual, as it is generally known that FIR filters are more robust than their recursive Infinite Impulse Response IIR counterpart.

ACKNOWLEDGEMENTS

I wish to acknowledge the assistance of the following people:

My parents and my fiance's parents who gave me lots of encouragement and support throughout my Master's Degree;

Mr. G. The for his technical guidance;

The staff of the Electrical Engineering department for their assistance;

Dr. M. Lucas for his consideration and patience while writing this thesis;

The staff of the Medicine department for their cooperation;

Mr. G. C. Hudson, who allowed the use of his computer and spent time to facilitate the preparation of this thesis;

Finally my fiance for his help in typing, drawing and editing this thesis.

Chapter 1

INTRODUCTION

1.1 CONTRIBUTIONS

The contributions of this thesis are:

* The development of the theory of a Non-Parametric Model Reference Adaptive Control (NP-MRAC)

* Simulation of the NP-MRAC on a plant with time-varying parameters as well as time-varying time-delay.

* Implementation of the closed-loop NP-MRAC on a TMS320C20 signal processing chip to control in real-time a plant with time-varying parameters and time delay .

* Part of this thesis formed the basis of a paper on "Model-Reference Adaptive Control using an FIR controller" which was represented at the IFAC Work shop on Robust Adaptive Control on 22-24 August 1988 at Newcastle, Australia.

1.2 ADAPTIVE CONTROL SYSTEMS

An adaptive control system is a system whose parameter is adjustable in such a way that it attempts to avoid degradation of the dynamic performance of a control system when environmental variations occur.This is usually the case in many practical situations, e.g.

The dynamic behavior of an aircraft depends on its altitude and speed .

The dynamic behavior of a dc motor varies with the moment of inertia and the friction of the load. This situation occurs in a variety of applications such as rolling mills, machine tools ,etc.

In adaptive control, it is desired to control the plant input such that the plant output signal follows a desired output response. In general, the plant parameters are unknown or time-varying. Therefore to design such a system we first have to identify the plant then to generate a controller . The adaptive control system ,then, consists of two functions, they are the plant parameter estimations and the controller parameter estimations.

Many different approaches to adaptive control have been proposed in the literature. However they can be grouped into two classes. They are the Self Tuning Adaptive Control (STAC) ,see Astrom et all (1977) and the Model Reference Adaptive Control (MRAC), see e.g., Laudau(1979).

1.2.1 The self-tuning adaptive control (STAC)

The concept of the STAC is illustrated in figure 1.2.1-1 in which a parameter estimation technique is used to identify the unknown parameters of the plant. These estimated parameters are then used to design an optimum controller. This approach of using the estimated parameters as if they were the true parameters for the purpose of controller design is called the "certainty equivalence principle".

The STAC closed loop system is nonlinear and time varying. It can be thought of as having two loops. The inner loop consists of the plant and a series linear time-varying controller. The parameters of this controller are adjusted by the outer loop, which is composed of the recursive parameter estimator and the controller synthesis.

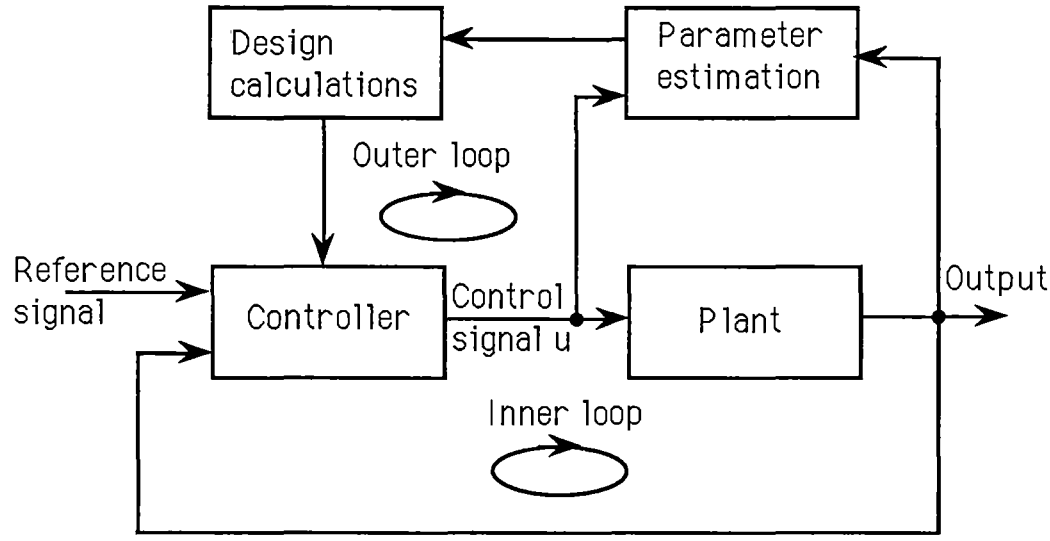


Figure 1.2.1-1 Block diagram of a self-tuning adaptive control (STAC).

The STAC is very flexible with respect to the system identification package and the controller synthesis package. Many different identification schemes may be used e.g. gradient projection, least squares, extended least squares, generalized least squares, maximum likelihood, instrumental variables, extended Kalman filtering. Various controller synthesis methods have also been reported in the literature e.g. phase and gain margins, pole placement, minimum variance, detuned minimum variance, linear quadratic Gaussian.

1.2.2 The model reference adaptive control(MRAC)

The MRAC was originally developed for servo problems and is shown in figure 1.2.2-1.

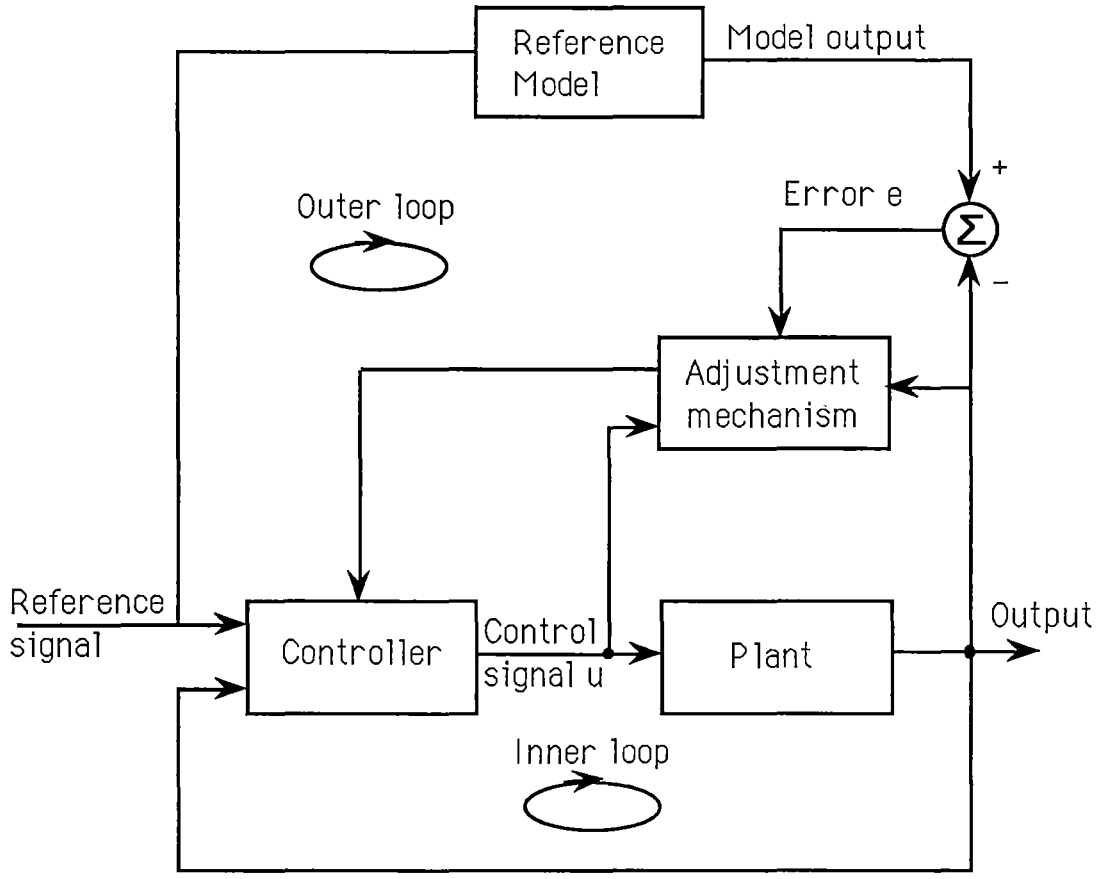


Figure 1.2.2-1 Block diagram of a Model-Reference Adaptive Control (MRAC).

The performance specifications are given in terms of a reference model, which determines how the plant output ideally should be in response to the reference command signal. Notice that the reference model is part of the control system. Again a system employing MRAC can be thought of as consisting of two loops. The inner loop is an ordinary control composed of the plant and a series linear time varying controller. The parameters of this controller are adjusted by the outer loop in such a way that the error between the model output and the plant output becomes small. The key problem is to determine the adjustment mechanism so that a stable system is obtained which brings the error to zero.

1.3 A NON-PARAMETRIC MODEL-REFERENCE ADAPTIVE CONTROL (NP-MRAC)

Both the STA and MRAC have flexibility in dealing with a variety of plants with unknown parameters and ability to keep the desired closed-loop dynamic behavior in an adaptive mode. But because they are mainly based on plant parameter estimation, therefore several problems limit the application of plant with time-varying parameters and time-varying time-delay. These problems are listed below:

(1) Parameter-adaptive controllers yield in many cases large variations of the process input signal during the adaptation phase (Iserman and Lachmann,1985).

(2) Unknown time-varying time-delay is difficult to identify the plant parameters under the assumption of a known constant time delay (Pupeikis 1985, Kaminskas 1979).

Many on-line procedures capable of tracking time-varying time-delay plant have been introduced. However they over-parameterize the discrete model (Bokor and Keviczky 1985, Kurz and Goedecke 1981). An on-line identification method proposed by Gawthrop and Nihtila (1985) is limited to systems with small time-delay and known dynamics.

(3) In estimating the plant parameters, unstable zeros are often introduced (M'Saad, Ortega and Landau, 1985). These zeros can not be canceled, hence good tracking is difficult to be achieved.

This thesis presents a new method of a Non-Parametric Model-Reference Adaptive Control (NP-MRAC), that is conceptually very simple and does not suffer from the above drawbacks. The method relies on determining the shape of an impulse response of the plant

rather than the plant parameters.

The NP-MRAC is based on the well known Least-Mean-Square (LMS) algorithm . The LMS algorithm (Widrow and Stearns, 1985) was normally applied to the adaptive equalization of communication channels and now apply the same principle to effect a model reference adaptive control. Since its inception by the Bell System group (Gersho, 1969; Lucky,1967; Sondhi, 1967), the LMS algorithm has always been implemented in a hybrid mode, i.e. the signal on analog form passes through a series of taps of a delay line and the taps are adjusted digitally.

However, the advent of high speed microprocessors such as the 8086/80186/80286 and their numeric co-processor made it possible to implement the series of the 30-tap delay line in real time to effect a model-reference adaptive control (Tran,1986).

The research work was carried on from the above principle and applied to a model-reference adaptive control. The Non-Parametric Model Reference Adaptive Control (NP-MRAC) was introduced . This work shows by simulation studies and real-time hybrid simulations that the LMS algorithm can be used effectively in the NP-MRAC. The NP-MRAC was implemented using a high speed signal processor chip, the Texas Instrument TMS320C20.

1.4 THESIS OUTLINE

This thesis contains 5 other chapters in addition to present one. Below is a brief outline of them.

CHAPTER 2 discuses an FIR adaptive filter , the LMS algorithm, the convergence of the weight vector and the method of a Non-Parametric Plant Identification (NP-PI) and an open loop Non-

Parametric Model-Reference Adaptive Control (NP-MRAC).

CHAPTER 3 presents the conditions of a reference-input signal. Simulations on the NP-PI and the open loop NP-MRAC are included.

CHAPTER 4 represents the closed-loop Non-Parametric Model-Reference Adaptive Control with two methods of model-reference selection. Simulations on a plant with time-delay and parameter variation are observed. So does the ability of set-point tracking of the system . The control of a type-1 plant is also discussed.

CHAPTER 5 shows an implementation of the closed loop NP-MRAC on the TMS320C20 signal processing. The chapter contains a TMS320C20 introduction, hardware introduction, software controller program summary, laboratory results and software descriptions.

CHAPTER 6 summarizes the contributions , discusses briefly the Recursive Least Square (RLS) algorithm technique applied to the NP-MRAC and gives possible extensions to the work as motivation for further work.

APPENDIX A shows the discrete impulse response of the second order system.

APPENDIX B contains the listing of all the Fortran 77 programs written for this thesis.

APPENDIX C details in the modification of the floating point routine.

APPENDIX D contains the listing of all the TMS320C20 programs written for the hybrid simulation.

APPENDIX E contains the paper on "Model-Reference Adaptive Control using an FIR controller".

CHAPTER 2

THE LEAST MEAN SQUARE ALGORITHM

The theory of a Non-Parametric Model Reference Adaptive Control (NP-MRAC) is built around a Finite Impulse Response (FIR) adaptive filter. The FIR of such a filter is defined by a set of tap weights. In this work, for updating the tap weights of the FIR adaptive filter, the Least Mean Square (LMS) algorithm is used.

This chapter is divided into 3 major parts :

Section 2.1 introduces the FIR adaptive filter and the Least-Mean-Square (LMS) algorithm. The convergence of the mean weight-vector of the FIR adaptive filter , which is adjusted using the LMS algorithm and the gradient search by the method of steepest descent, is also discussed.

Section 2.2 shows how the FIR adaptive filter can be applied to effect plant identification . Discussion on the convergence of the mean of the weight-vector of the FIR adaptive filter in this identification process is included.

Section 2.3 modifies the LMS algorithm so that it can be used as an adaptive controller in a model-reference adaptive control ; this principle is known as the filtered-X LMS algorithm. The convergence of the mean of the weight vector of the FIR adaptive filter by means of the filtered-X LMS algorithm is also studied.

2.1 DERIVATION OF THE LMS ALGORITHM

Our purpose here is to introduce the LMS algorithm and to describe the performance characteristics of the LMS algorithm applied to adjust the weight vector of the FIR adaptive filter .

2.1.1 The performance function:

The FIR adaptive filter is fundamental to our design . It consists of two basic parts

(1) an FIR filter with L adjustable tap weights whose values at time k are denoted by $w_{0\,k}, w_{1\,k}, \ldots, w_{L-1\,k}$ and

(2) a mechanism for adjusting these tap weights in an adaptive manner.

The FIR filter consists of a set of delay-line elements (each of which is represented by the unit-delay operator z^{-1}) and a corresponding set of adjustable coefficients, which are interconnected in the manner shown in figure 2.1.1-1. The input signals to the variable weights are the signals at the delay-line taps. At time k, these signals are defined as $x_k, x_{k-1}, \ldots, x_{k-L+1}$,where L is a number of the taps. These signals are sequential samples taken at $k, k-1, \ldots k-L+1$ going back in time through the sequence of data samples and they are group together to form an input signal vector

$$X_k = [x_k, x_{k-1}, \ldots, x_{k-L+1}]^T \qquad (2.1.1-1)$$

the subscript k is used as a time index.

Similarly we define the tap-weight vector as

$$W_k = [w_{0\,k}, w_{1\,k}, \ldots, w_{L-1\,k}]^T \qquad (2.1.1-2)$$

Each sample of the input signal vector is multiplied by a corresponding set of adjustable tap weights $w_{0\,k}, w_{1\,k}, \ldots, w_{L-1\,k}$, to produce an output signal denoted by y_k.

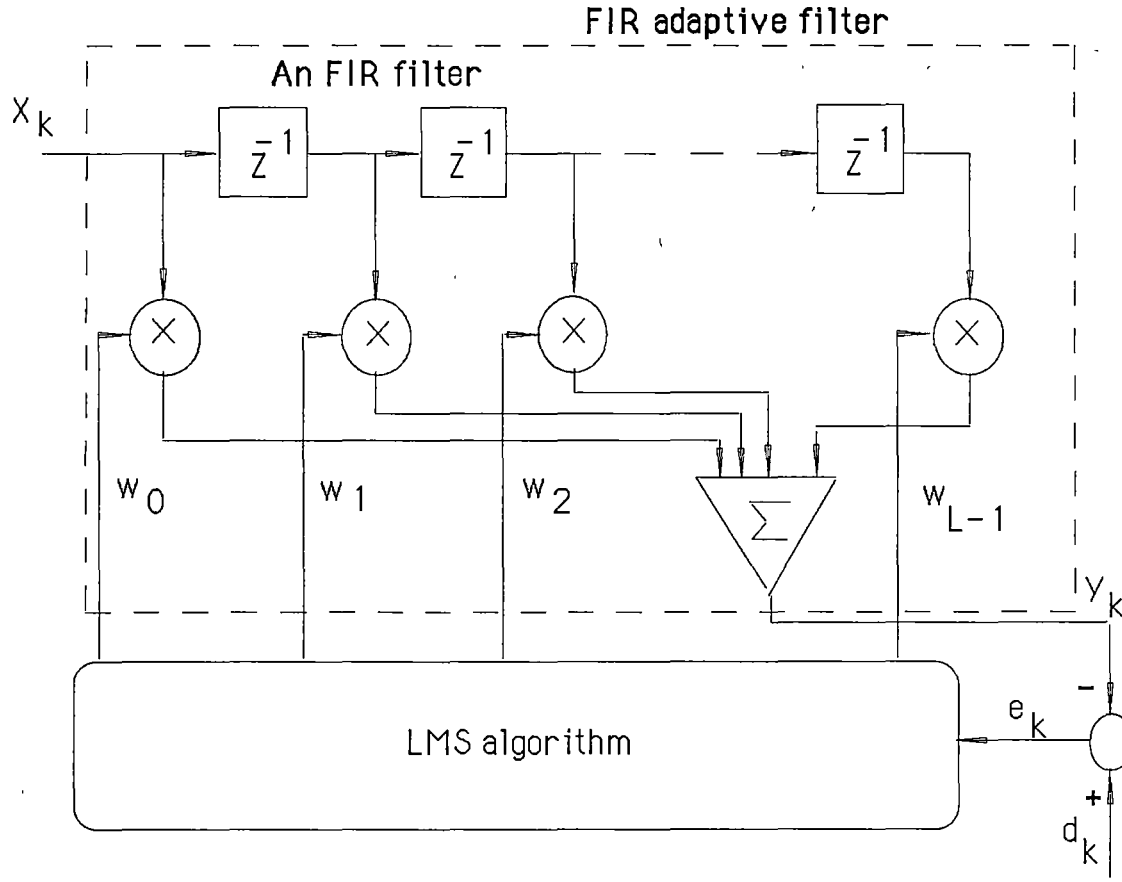


Figure 2.1.1-1 The FIR adaptive filter using a tapped-delay line.

During the filtering process, the output of the FIR adaptive filter , y_k ,is compared with the desired response output d_k to produce an estimation error e_k . Hence we have

$$e_k = d_k - y_k$$

then

$$e_k = d_k - X_k^T W_k$$

or

$$e_k = d_k - W_k^T X_k \qquad (2.1.1-3)$$

Since we do not vary W_k in the next discussion, therefore for convenience we now drop the subscript k from the weight vector,

W . If the tap-input vector , X_k ,and the desired response output, d_k, are stationary, then the mean-square error , ξ , which is defined as $\xi = E[e_k^2]$, at time k is given by

$$\begin{aligned}\xi &= E[e_k^2] \\ &= E[d_k^2] - 2E[d_k X_k^T]W + W^T E[X_k X_k^T]W \\ \xi &= E[d_k^2] - 2P^T W + W^T RW \end{aligned} \qquad (2.1.1-4)$$

where P is the cross-correlation vector between the tap input vector , X_k ,and the desired response , d_k, which is known as

$$\begin{aligned} P &= E[d_k X_k] \\ P &= E[d_k x_k, d_k x_{k-1}, \ldots, d_k x_{k-L+1}]^T \end{aligned} \qquad (2.1.1-5)$$

and R is the auto-correlation matrix of the tap-input vector X_k, which is defined as

$$\begin{aligned} R &= E[X_k X_k^T] \\ R &= E\begin{bmatrix} x_k^2 & x_k x_{k-1} & \cdots & x_k x_{k-L+1} \\ x_{k-1} x_k & x_{k-1}^2 & \cdots & x_{k-1} x_{k-L+1} \\ \vdots & \vdots & \cdots & \vdots \\ x_{k-L+1} x_k & x_{k-L+1} x_{k-1} & \cdots & x_{k-L+1}^2 \end{bmatrix} \end{aligned} \qquad (2.1.1-6)$$

From equation (2.1.1-4) ,we visualize the dependence of the mean-squared error , ξ , on the elements of the tap-weight vector W as a bowl-shaped surface with an unique minimum. We refer to this surface as the error-performance surface of the FIR adaptive filter. The adaptive process has the task of continually seeking the bottom (minimum point) of this surface. At the minimum point of

the error-peformance surface, the tap-weight vector is designated as the optimum value , W^*.

2.1.2 Gradient search and the optimal weight vector :

The optimum weight vector can be sought out by gradient techniques, which is discussed in this section.

The gradient of the mean-square-error performance surface, designated by ∇ , can be obtained by differentiating (2.1.1-4) to obtain the column vector.

$$\nabla = \left[\frac{\partial \xi}{\partial w_0}, \frac{\partial \xi}{\partial w_1}, \cdots \frac{\partial \xi}{\partial w_{L-1}} \right]^T \qquad (2.1.2-1)$$

The input signal and desired response are assumed to be stationary ergodic which are independent of W_k, hence both R and P are not function of W . Therefore , ∇ can be calculated as

$$\nabla = 2RW - 2P \qquad (2.1.2-2)$$

To obtain the minimum mean-square error ,the gradient must be at zero ($\nabla = 0$) and the weight vector W is set at its optimal value W^* . Hence,

$$2RW^* - 2P = 0$$

Assuming that R is non-singular , the optimal weight vector W^* (also known as the Weiner weight vector) is then found as

$$W^* = R^{-1}P \qquad (2.1.2-3)$$

The requirement that an FIR adaptive filter has to satisfy is to find a solution for its tap-weight vector that satisfies the equation (2.1.2-3). One way of doing this would be to solve this equation by some analytical means. Although, this procedure is quite straightforward, nevertheless, it presents serious computational

difficulties, especially when the filter contains a large number of tap weights. An alternative procedure is to use the method of steepest descent .

2.1.3 Gradient search by the method of steepest descent

To find the minimum value of the mean-squared error, ξ_{min} , by the method of steepest descent , the tap-weight vector is adjusted in the direction of the negative of the gradient ($-\nabla_k$) at each step. The updated value of the tap-weight vector ,at the k+1th sampling ,is computed by using the simple recursive relation

$$W_{k+1} = W_k + \mu(-\nabla_k) \qquad (2.1.3-1)$$

where μ is a positive real constant that regulates the step size .

By substituting equation (2.1.2-2) for the gradient term in equation (2.1.3-1), we compute the updated value of the tap-weight vector W_{k+1} by using the simple recursive relation

$$W_{k+1} = W_k + 2\mu(-RW_k + P) \qquad (2.1.3-1b)$$

Equation (2.1.3-1b) then describes the mathematical formulation of the steepest-descent algorithm. To determine the condition for the stability of the steepest-descent algorithm, we follow Widrow and Stearns(1985).

Substituting equation (2.1.2-3) into equation (2.1.3-1b) we have

$$W_{k+1} = W_k + 2\mu R(W^* - W_k)$$

$$W_{k+1} = (I - 2\mu R)W_k + 2\mu RW^* \qquad (2.1.3-2)$$

Equation(2.1.3-2) is solved by transforming to the principal coordinate system. We begin the analysis by defining a weight-error vector at time k as $V_k = W_k - W^*$, equation (2.1.3-2) then

becomes

$$V_{k+1} + W^* = (I - 2\mu R)V_k + W^* - 2\mu RW^* + 2\mu RW^*$$
$$V_{k+1} = (I - 2\mu R)V_k \qquad (2.1.3-3)$$

We define V_k^* as

$$V_k^* = Q^{-1}V_k \qquad (2.1.3-4)$$

where Q is the eigenvector matrix of R.

Substituting equation (2.1.3-4) into equation (2.1.3-3), we now have

$$QV_{k+1}^* = (I - 2\mu R)QV_k^* \qquad (2.1.3-5)$$

Multiplying both sides of equation (2.1.3-5) by Q^{-1}, we have

$$Q^{-1}QV_{k+1}^* = Q^{-1}(I - 2\mu R)QV_k^*$$
$$V_{k+1}^* = Q^{-1}(I - 2\mu R)QV_k^*$$
$$= (Q^{-1}IQ - 2\mu Q^{-1}RQ)V_k^*$$

but $Q^{-1}RQ = \Lambda$ where Λ is the eigenvalue matrix of R (The property of eigenvalues and eigenvectors) and Λ is a diagonal matrix which is defined as

$$\Lambda = \begin{bmatrix} \lambda_0 & 0 & \dots & 0 \\ 0 & \lambda_1 & \dots & 0 \\ 0 & 0 & \dots & 0 \\ 0 & 0 & \dots & \lambda_{L-1} \end{bmatrix}$$

Then , we have

$$V_{k+1}^* = (I - 2\mu\Lambda)V_k^*$$

Since there is no cross-coupling in the principal-coordinate

system , we have

$$V_k^* = (I - 2\mu\Lambda)^k V_0^* \qquad (2.1.3-7)$$

Because the product of two diagonal matrices is just the matrix of products of corresponding elements , therefore

$$(I - 2\mu\Lambda)^k = \begin{bmatrix} (1-2\mu\lambda_0)^k & 0 & \cdots & 0 \\ 0 & (1-2\mu\lambda_1)^k & \cdots & 0 \\ 0 & 0 & \cdots & 0 \\ 0 & 0 & \cdots & 0 \\ 0 & 0 & \cdots & (1-2\mu\lambda_{L-1})^k \end{bmatrix}$$

W_k and W^* are column vectors of L elements ,therefore V_k^* and V_0^* are also column vectors of L elements .

We define the elements of these vectors as $v_{i\,k}^*$ and $v_{i\,0}^*$, respectively , *where* $i = 0, 1, \ldots, L-1.$

Hence,

$$v_{i\,k}^* = (1-2\mu\lambda_i)^k v_{i\,0}^* \qquad (2.1.3-8)$$

Since W_k is the sum of V_k and W^*, therefore to guarantee the stability of the steepest-descent algorithm, we must have

$$\begin{bmatrix} \lim\limits_{k\to\infty}(1-2\mu\lambda_0)^k & 0 & \cdots & 0 \\ 0 & \lim\limits_{k\to\infty}(1-2\mu\lambda_1)^k & \cdots & 0 \\ 0 & 0 & \cdots & 0 \\ 0 & 0 & \cdots & 0 \\ 0 & 0 & \cdots & \lim\limits_{k\to\infty}(1-2\mu\lambda_{L-1})^k \end{bmatrix} = 0$$

$$(2.1.3-9)$$

In this form , we see that the convergence condition is satisfied by choosing μ so that:

$$-1 < 1 - 2\,\mu\lambda_i < 1 \qquad \text{for all i =0,1,...,L-1.}$$

The eigenvalues of the correlation matrix R are all real and positive because R is real , symmetric and in general positive definite (Widrow,1985). It therefore follows that the necessary and sufficient condition for the convergence or stability of the steepest-descent algorithm is that the step size parameter μ satisfy the following condition:

$$0 < \mu < \frac{1}{\lambda_{\max}} \qquad (2.1.3-10)$$

where $\lambda_{\max}$ is the largest eingenvalue of R.

If this condition is satisfied, it follows from equation (2.1.3-7) that

$$\lim_{k \to \infty} V_k^* = 0 \qquad (2.1.3-11)$$

Now if we substitute (equation 2.1.3-4) in which

$$\begin{aligned} V_k^* &= Q^{-1} V_k \\ &= Q^{-1}(W_k - W^*) \end{aligned}$$

into equation (2.1.3-11) , we find that

$$\lim_{k \to \infty} W_k = W^* \qquad (2.1.3-12)$$

Equation (2.1.3-12) shows that if the condition in (2.1.3-10) is satisfied then the steepest-descent method will guarantee the weight vector to converge to its minimum.

2.2 THE LMS ALGORITHM FOR PLANT IDENTIFICATION

If it were possible to make exact measurements of the gradient vector at each iteration, and if the step-size parameter μ is suit-

able chosen, then the tap-weight vector computed by using the method of steepest descent would indeed converge to the optimum solution. In reality, however, exact measurements of the gradient vector are not possible, and the gradient vector must be estimated from the available data. On other words, the tap-weight vector is updated in accordance with an algorithm that adapts to the incoming data. One such algorithm ,that is used in this work , is the Least Mean Square (LMS) algorithm (Widrow and Hoff,1960). A significant feature of the LMS algorithm is its simplicity; it does not require measurements of the correlation functions, nor does it require matrix inversion.

This section describes an application of the FIR adaptive filter to plant identification using the LMS algorithm.

2.2.1 The method

Figure 2.2-1 shows how the schematic configuration of the FIR adaptive filter is used in the plant identification. Both the unknown plant and the FIR adaptive filter are driven by the same input , x_k. Our aim in this application is to predict an unknown FIR of the plant through the minimization of the mean square error, $\xi = E[e_k^2]$, where the estimation error , e_k , is produced by the difference between the plant output and output of the FIR adaptive filter. At the k^{th} iteration , the estimation error e_k is non-zero, implying that the filter output deviates from the reference-model output. In an attempt to account for this deviation, the estimation error e_k is used as the input to an adaptive control algorithm, whereby it controls the corrections applied to the indi-

vidual tap weights in the FIR filter. As a result, the tap weights of the filter have a new set of values for use on the next iteration. Thus , at the $k+1^{th}$ iteration, a new filter output is produced, and with it a new value for the estimation error. The operation described is then repeated. This process is continued for a sufficiently large number of iterations (starting at iteration k=0), until the deviation of the model from the unknown dynamic plant, measured by the estimation error e_k, becomes sufficiently small. This method of determining the FIR of the plant is referred to as the Non-Parametric Plant Identification (NP-PI).

In figure 2.2-1(see next page) , we define the weight vector of the FIR adaptive filter as W_k . Then ,we have :

$$e_k = d_k - X_k^T W_k \qquad (2.2.1-1)$$

where

$$X_k = [x_k, x_{k-1}, x_{k-2}, \dots , x_{k-L+1}]^T \qquad (2.2.1-2)$$

and

$$W_k = [w_{0\,k}, w_{1\,k}, \dots , w_{L-1\,k}]^T \qquad (2.2.1-3)$$

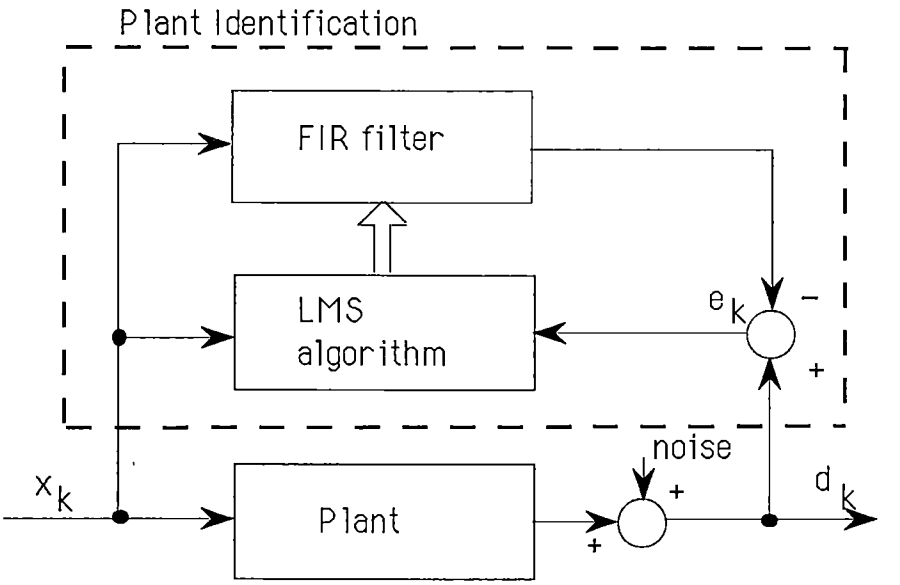


Figure 2.2-1: Block diagram of the NP-PI.

To develop the LMS algorithm , at each iteration in the adaptive process , we take a gradient estimate , $\hat{\nabla}_k$, of the form:

$$\hat{\nabla}_k = \begin{bmatrix} \frac{\partial e_k^2}{\partial w_0} \\ \vdots \\ \vdots \\ \frac{\partial e_k^2}{\partial w_{L-1}} \end{bmatrix} = 2e_k \begin{bmatrix} \frac{\partial e_k}{\partial w_0} \\ \vdots \\ \vdots \\ \frac{\partial e_k}{\partial w_{L-1}} \end{bmatrix}$$

$$\hat{\nabla}_k = -2e_k X_k \qquad (2.2.1-4)$$

and the weights are adjusted in the direction of the negative gradient at each step ,by definition , we have

$$W_{k+1} = W_k - \mu \hat{\nabla}_k$$

$$W_{k+1} = W_k + 2\mu e_k X_k \qquad (2.2.1-5)$$

where μ is a constant that regulates the step size .

Equation (2.2.1-5) is called the LMS algorithm, which is used to

adjust the weight vector of the FIR adaptive filter throughout this work.

2.2.2 Convergence of the weight vector

In this section , we will study the convergence of the LMS algorithm as applied to plant identification.

To examine LMS convergence ,we first note that the gradient estimate in equation(2.2.1-4) can readily be shown to be unbiased when the weight vector is held constant. The expected value of equation (2.2.1-4) with W_k held equal to W is

$$\begin{aligned} E[\hat{\nabla}_k] &= -2E[e_k X_k] \\ &= -2E[d_k X_k - X_k X_k^T W] \\ &= -2(P - RW) \\ &= 2(RW - P) \end{aligned}$$

and from equation (2.1.2-2) we have

$$E[\hat{\nabla}_k] = \nabla \qquad (2.2.2-1)$$

Since the expected value $\hat{\nabla}_k$ is equal to the true gradient ∇, $\hat{\nabla}_k$must be an unbiased estimate. However, with the weight vector changing at each iteration , we need to examine the weight vector convergence , as follows .

From equation (2.2.1-5) we can see that the weight vector W_k is a function only of the past input vectors $x_{k-L+1}, x_{k-L+2}, \cdots, x_{k-1}, x_k$. For stationary input processes we have W_k is independent of X_k , because successive input vectors are independent over time . Therefore taking the expected value of both sides of equation(2.2.1-5) yields

$$E[W_{k+1}] = E[W_k] + 2\mu E[e_k X_k] \qquad (2.2.2-2)$$
$$= E[W_k] + 2\mu\{E[d_k X_k] - E[X_k X_k^T W_k]\}$$
$$= E[W_k] + 2\mu\{P - RE[W_k]\}$$
$$E[W_{k+1}] = (I - 2\mu R)E[W_k] + 2\mu P \qquad (2.2.2-3)$$
$$E[W_{k+1}] = (I - 2\mu R)E[W_k] + 2\mu R W^* \qquad (2.2.2-4)$$
$$(as\ from\ equation\ (2.1.2-3)\ W^* = R^{-1}P).$$

Equation (2.2.2-4) has just the expected form of equation (2.1.3-2) , which was solved by changing to the principal-axis coordinate system. Using expected values , the solution is

$$E[V_k^*] = (I - 2\mu\Lambda)^k V_0^* \qquad (2.2.2-5)$$

where V_k^* is the weight vector, W_k , in the principal-axis system, Λ is the diagonal eigenvalue matrix of R, and V_0^* is the initial weight vector in the principal-axis system.

Therefore , as k increases without bound , the expected weight vector in equation (2.2.2-5) reaches the optimum solution(i.e., zero in the principal-axis system) only if the right hand side of the equation converges to zero. We have seen in section (2.1.3) that such convergence is guaranteed only if

$$0 < \mu < \frac{1}{\lambda_{max}} \qquad (2.2.2-6)$$

where λ_{max} is the largest eigenvalue of R.

Hence the LMS algorithm based on the steepest-descent gradient searching method will control the taps of the FIR filter to its optimum value.

2.2.3 The weight vector of the FIR adaptive filter in plant identification

If the input signal, x_k ,is assumed to be a stationary ergodic white sequence with variance , δ^2 ,then from equation (2.1.1-5) and (2.1.1-6) we have

$$P = \delta^2 G \qquad (2.2.3-1)$$

$$R = \delta^2 I \qquad (2.2.3-2)$$

where

$$G = [\, g_0,\ g_1,\ g_2,\ \cdots,\ g_{L-1}]^T \qquad (2.2.2-3)$$

G is called the unit impulse response of the plant and $g_0, g_1, g_2, \cdots, g_{L-1}$ are constant parameters of G.

The optimum weight vector , W^* , in equation (2.1.2-3) then becomes

$$W^* = R^{-1}P$$

$$= (\delta^2 I)^{-1}(\delta^2 G)$$

$$W^* = G \qquad (2.2.3-4)$$

which shows that the weight setting of the FIR adaptive filter matches with the plant unit impulse response.

So, if the input signal is stationary ergodic white sequence and condition (2.2.2-6) is satisfied, then the LMS steepest-descent method will guarantee that the shape of the weight vector of the FIR adaptive filter converges to a true plant FIR.

2.3 THE FILTERED-X LMS ALGORITHM FOR AN OPEN LOOP MODEL REFERENCE ADAPTIVE CONTROL

The LMS algorithm of the form as in equation (2.2.1-5) has to be modified to apply into model-reference adaptive control. The required modification is known as the filtered-X LMS algorithm, (Widrow and Stearns 1985).

2.3.1 The filtered-X LMS algorithm

As we have seen in section (2.2.1) for plant identification that to control tap-weights of the FIR adaptive filter, the estimation error e_k is used as the input to an adaptive control algorithm. The LMS algorithm adjusts the tap-weight vector in the form of equation (2.2.1-5) which is rewritten here for convenient,

$$W_{k+1} = W_k + 2\mu e_k X_k \qquad (2.3.1-1)$$

where μ is a constant and e_k is the difference between the plant output (d_k) and the FIR adaptive filter output ($W^T X_k$). i.e.,

$$e_k = d_k - W^T X_k \qquad (2.3.1-2)$$

This principle was illustrated in figure 2.2-1.

However , to apply the LMS algorithm and the FIR adaptive filter to model-reference adaptive control , the requirement is to develop a controller in the form of the FIR adaptive filter to control the plant so that its output signal, c_k, follows the model-reference output signal , d_k. This principle is illustrated in figure 2.3.1-1.

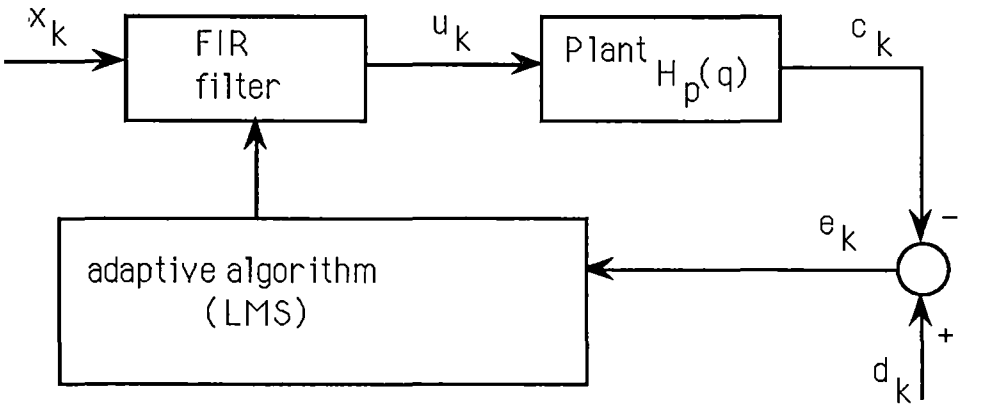


Figure2.3.1-1 The FIR filter applied to model reference adaptive control.

Figure 2.3.1-1 shows that when the LMS algorithm is applied to

a model-reference adaptive control, the signal e_k is not the error at the FIR filter output, but at the plant output ($e_k = d_k - c_k$) .

Now, if we define the pulse transfer function of the plant as a rational function in the shift operator q (for more detail, see chapter 3), $H_p(q)$, then the plant output , c_k , can be written as

$$c_k = H_p(q)u_k \qquad (2.3.1-3)$$

where u_k is the plant input signal .

Then we have

$$\begin{aligned} e_k &= d_k - c_k \\ &= d_k - H_p(q)u_k \end{aligned}$$

But u_k is the output of the FIR filter, whose weight vector is W_k therefore

$$u_k = W_k^T X_k$$

Hence

$$\begin{aligned} e_k &= d_k - H_p(q)(W_k^T X_k) \\ e_k &= d_k - W_k^T(H_p(q)X_k) \qquad (2.3.1-4) \end{aligned}$$

As seen from equation (2.3.1-4), in model-reference adaptive control the error e_k is the difference between the desired reference-model response d_k and plant output signal in the form of $W_k^T(H_p(q)X_k)$, whereas in plant identification the error e_k is the difference between the plant response d_k and and the FIR adaptive filter output $W^T X_k$.

Therefore to be able to apply to model-reference adaptive control, the LMS algorithm , in the form of equation (2.3.1-1), has to be modified as

$$W_{k+1} = W_k + 2\mu e_k(H_p(q)X_k) \qquad (2.3.1-5)$$

where $H_p(q)X_k$ is the plant output vector subject to the plant input vector X_k.

Alternatively, this modification can also be derived by taking the gradient estimate as in equation (2.2.1-4) which is

$$\hat{\nabla}_k = [\frac{\partial e_k^2}{\partial w_0}, \frac{\partial e_k^2}{\partial w_1}, \cdots \frac{\partial e_k^2}{\partial w_{L-1}}]^T$$

with e_k as given in equation (2.3.1-4).

Then we have

$$\hat{\nabla}_k = \begin{bmatrix} \frac{\partial e_k^2}{\partial w_0} \\ \vdots \\ \frac{\partial e_k^2}{\partial w_{L-1}} \end{bmatrix} = 2e_k \begin{bmatrix} \frac{\partial e_k}{\partial w_0} \\ \vdots \\ \frac{\partial e_k}{\partial w_{L-1}} \end{bmatrix}$$

$$\hat{\nabla}_k = -2e_k[H_p(q)X_k] \qquad (2.3.1-6)$$

where $X_k = [x_k, x_{k-1}, \cdots, x_{k-L+1}]^T$

$$W_k = [w_{0\,k}, w_{1\,k}, \cdots, w_{L-1\,k}]^T$$

and using the method of steepest descent, we obtain

$$W_{k+1} = W_k - \mu \hat{\nabla}_k$$
$$W_{k+1} = W_k + 2\mu e_k[H_p(q)X_k]$$

This equation is exactly as same as equation (2.3.1-5) and is known as the filtered-X LMS algorithm. It is used to adjust the weight vector of the controller in the model reference adaptive control and the method is named as an open-loop Non-Parametric Model Reference Adaptive Control (NP-MRAC).

Figure 2.3.1-2 illustrates mathematically how the NP-MRAC is realized.

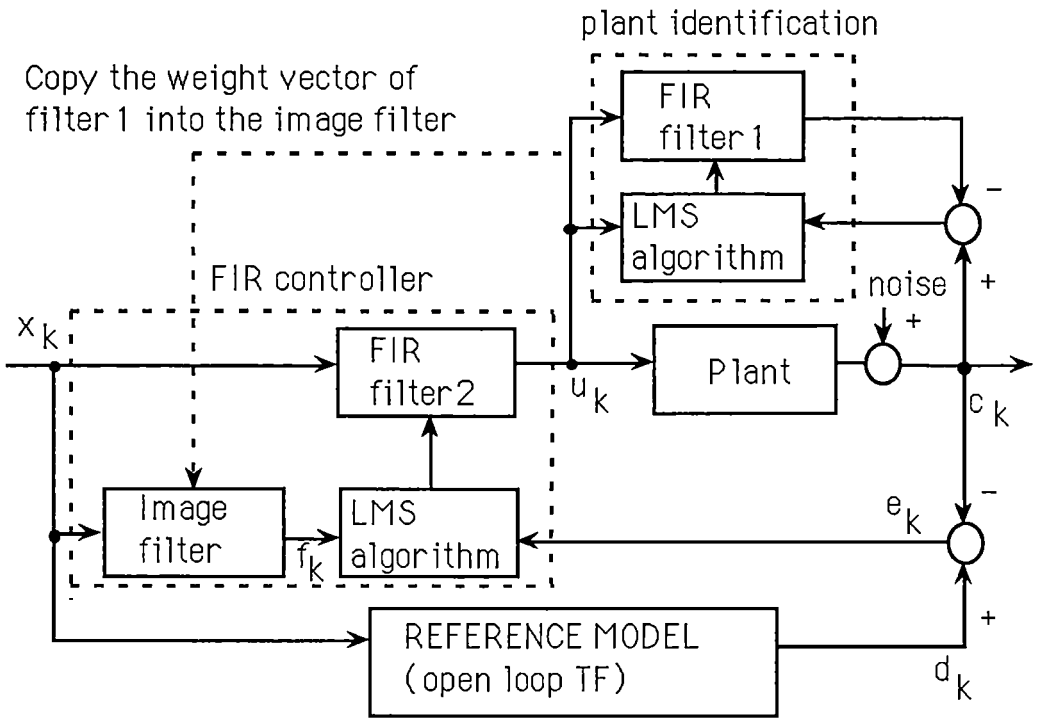


Figure 2.3.1-2: Block diagram of the open loop NP-MRAC.

As seen in this figure , the open loop NP-MRAC uses two adaptive FIR filters whose weight vectors are adjusted by the LMS algorithm : one (FIR filter1) to identify the FIR of the plant and the other (FIR filter2) to control the plant .

The weight vector of the FIR filter 1 is transferred into the image filter. The input signal, x_k, is now modified by the image filter to produce the signal f_k .This signal,f_k, and the error signal,e_k, are used by the LMS algorithm to adjust the weight vector of the FIR controller.

2.3.2 CONVERGENCE OF THE WEIGHT VECTOR

In this section we will discuss on the convergence of the tap-

setting of the controller FIR filter which is based on the filterred-X LMS algorithm applying to Non_Parametric Model-Reference Adaptive Control (NP-MRAC).

Let

$$G = [g_0, g_1, \cdots, g_{L-1}]^T \qquad (2.3.2-1)$$

be the plant unit impulse response vector.

Let

$$M = [m_0, m_1, \cdots, m_{L-1}]^T \qquad (2.3.2-2)$$

be the reference-model unit impulse response vector.

Let

$$W = [w_0, w_1, \cdots, w_{L-1}]^T \qquad (2.3.2-3)$$

be the controller weight vector.

Without loss of generality, we can assume that the length of vectors G , M and W is L , which is a maximum number of non-zero samples of the impulse response of the plant, model and controller respectively(as seen in equations 2.3.2-1, 2.3.2-2 and 2.3.2-3).

We define an input signal vector as

$$X_k = [x_k, x_{k-1}, x_{k-2}, \cdots, x_{k-L+1}]^T \qquad (2.3.2-4)$$

Then the FIR controller output signal is

$$u_k = X_k^T W$$

or

$$u_k = W^T X_k \qquad (2.3.2-5)$$

and we define an output signal vector as

$$U_k = [u_k, u_{k-1}, \cdots, u_{k-L+1}]^T \qquad (2.3.2-6)$$

The FIR model output signal is

$$d_k = X_k^T M$$

or

$$d_k = M^T X_k \qquad (2.3.2-7)$$

The plant output signal is

$$y_k = G^T U_k$$

$$y_k = G^T \begin{bmatrix} X_k^T W \\ X_{k-1}^T W \\ \vdots \\ X_{k-L+1}^T W \end{bmatrix} \qquad (2.3.2-8)$$

The error signal e_k is defined as

$$e_k = d_k - y_k \qquad (2.3.2-9)$$

Hence

$$e_k = d_k - G^T \begin{bmatrix} X_k^T W \\ X_{k-1}^T W \\ \vdots \\ X_{k-L+1}^T W \end{bmatrix} \qquad (2.3.2-10)$$

or

$$e_k = M^T X_k - G^T \begin{bmatrix} X_k^T W \\ X_{k-1}^T W \\ \vdots \\ X_{k-L+1}^T W \end{bmatrix} \qquad (2.3.2-10b)$$

Now let

$$\xi = E[e_k^2]$$

then we have

$$\xi = E[d_k^2] - 2E\left[d_k G^T \begin{bmatrix} X_k^T W \\ X_{k-1}^T W \\ \vdots \\ X_{k-L+1}^T W \end{bmatrix} \right] + W^T E\left[\begin{bmatrix} G^T X_k \\ G^T X_{k-1} \\ \vdots \\ G^T X_{k-L+1} \end{bmatrix} \begin{bmatrix} G^T X_k \\ G^T X_{k-1} \\ \vdots \\ G^T X_{k-L+1} \end{bmatrix}^T \right] W \qquad (2.3.2-11)$$

$$\xi = E[d_k^2] - 2E\left[d_k \begin{bmatrix} G^T X_k \\ G^T X_{k-1} \\ \vdots \\ G^T X_{k-L+1} \end{bmatrix}^T \right] W + W^T E\left[\begin{bmatrix} G^T X_k \\ G^T X_{k-1} \\ \vdots \\ G^T X_{k-L+1} \end{bmatrix} \begin{bmatrix} G^T X_k \\ G^T X_{k-1} \\ \vdots \\ G^T X_{k-L+1} \end{bmatrix}^T \right] W \qquad (2.3.2-12)$$

$$\xi = E[d_k^2] - 2P^T W + W^T R W \qquad (2.3.2-13)$$

where

$$P = E\left[d_k \begin{bmatrix} G^T X_k \\ G^T X_{k-1} \\ \vdots \\ G^T X_{k-L+1} \end{bmatrix} \right] \qquad (2.3.2-14)$$

and

$$R = E\left[\begin{bmatrix} G^T X_k \\ G^T X_{k-1} \\ \vdots \\ G^T X_{k-L+1} \end{bmatrix}\begin{bmatrix} G^T X_k \\ G^T X_{k-1} \\ \vdots \\ G^T X_{k-L+1} \end{bmatrix}^T\right] \qquad (2.3.2-15)$$

Differentiating equation (2.3.2-13) , we obtain

$$\frac{d\xi}{dW} = 2RW - 2P \qquad (2.3.2-16)$$

Now solving for the optimum weight vector, we have

$$W^* = R^{-1}P \qquad (2.3.2-17)$$

The result of equation (2.3.2-17) indicates that optimum weight vector of the controller using the NP-MRAC method can be found by the gradient techniques.

Same principle as section (2.2.1) , at each iteration in the adaptive process, we take a gradient estimate of the form:

$$\hat{\nabla}_k = [\frac{\partial e^2}{\partial w_0}, \frac{\partial e^2}{\partial w_1}, \cdots, \frac{\partial e^2}{\partial w_{L-1}}]^T$$

then

$$\hat{\nabla}_K = -2e_k\begin{bmatrix} G^T X_k \\ G^T X_{k-1} \\ \vdots \\ G^T X_{k-L+1} \end{bmatrix} \qquad (2.3.2-18)$$

To show that the weight vector converges to its minimum, first we can show that the gradient is unbiased, as follows.

$$E\left[\hat{\nabla}_k\right] = -2E\left[e_k \begin{bmatrix} G^T X_k \\ G^T X_{k-1} \\ \vdots \\ G^T X_{k-L+1} \end{bmatrix} \right] \qquad (2.3.2-19)$$

and from equation (2.3.2-10), we obtain

$$E\left[\hat{\nabla}_k\right] = -2E\left[\left(d_k - \begin{bmatrix} G^T X_k \\ G^T X_{k-1} \\ \vdots \\ G^T X_{k-L+1} \end{bmatrix}^T W \right) \begin{bmatrix} G^T X_k \\ G^T X_{k-1} \\ \vdots \\ G^T X_{k-L+1} \end{bmatrix} \right] \qquad (2.3.2-20)$$

$$E\left[\hat{\nabla}_k\right] = -2E\left[\left(d_k \begin{bmatrix} G^T X_k \\ G^T X_{k-1} \\ \vdots \\ G^T X_{k-L+1} \end{bmatrix} - \begin{bmatrix} G^T X_k \\ G^T X_{k-1} \\ \vdots \\ G^T X_{k-L+1} \end{bmatrix} \begin{bmatrix} G^T X_k \\ G^T X_{k-1} \\ \vdots \\ G^T X_{k-L+1} \end{bmatrix}^T W \right) \right] \qquad (2.3.2-21)$$

Hence,

$$\begin{aligned} E\left[\hat{\nabla}_k\right] &= 2RW - 2P \\ &= \nabla \end{aligned} \qquad (2.3.2-22)$$

where P and R are defined in equation (2.3.2-14) and (2.3.2-15), respectively.

Equation (2.3.2-22) shows that the expected value $\hat{\nabla}_k$ is equal

to the true gradient ∇. Therefore $\hat{\nabla}_k$ must be unbiased estimate. Next we will examine the weight vector convergence.

Let us assume that we know exactly plant impulse response (G), then the filtered-X LMS algorithm (in the form of equation 2.3.1-5) can be rewritten as

$$W_{k+1} = W_k + 2\mu e_k \begin{bmatrix} G^T X_k \\ G^T X_{k-1} \\ \vdots \\ G^T X_{k-L+1} \end{bmatrix}$$

Taking the expecting weight vector of this equation , we have

$$E[W_{k+1}] = E[W_k] + 2\mu E\left[e_k \begin{bmatrix} G^T X_k \\ G^T X_{k-1} \\ \vdots \\ G^T X_{k-L+1} \end{bmatrix} \right]$$

$$(2.3.2-23)$$

e_k is as defined as in equation (2.3.2-9) and the controller weight vector is set at W_k . Thus , we have

$$E[W_{k+1}] = E[W_k] + 2\mu E\left[M^T X_k \begin{bmatrix} G^T X_k \\ G^T X_{k-1} \\ \vdots \\ G^T X_{k-L+1} \end{bmatrix} \right] - 2\mu E\left[G^T \begin{bmatrix} W_k^T X_k \\ W_k^T X_{k-1} \\ \vdots \\ W_k^T X_{k-L+1} \end{bmatrix} \begin{bmatrix} G^T X_k \\ G^T X_{k-1} \\ \vdots \\ G^T X_{k-L+1} \end{bmatrix} \right] \qquad (2.3.2-24)$$

or

$$E[W_{k+1}] = E[W_k] + 2\mu E\left[M^T X_k \begin{bmatrix} G^T X_k \\ G^T X_{k-1} \\ \vdots \\ G^T X_{k-L+1} \end{bmatrix} \right] - 2\mu E\left[\begin{bmatrix} G^T X_k \\ G^T X_{k-1} \\ \vdots \\ G^T X_{k-L+1} \end{bmatrix} \begin{bmatrix} G^T X_k \\ G^T X_{k-1} \\ \vdots \\ G^T X_{k-L+1} \end{bmatrix}^T \right] E[W_k] \qquad (2.3.2-25)$$

$$E[W_{k+1}] = (I - 2\mu R)E[W_k] + 2\mu P \qquad (2.3.2-26)$$

where P and R are as defined as in equation (2.3.2-14) and (2.3.2-15) , respectively. Substituting $P = RW^*$ (from equation 2.3.2-17) into equation (2.3.2-26) , we have

$$E[W_{k+1}] = (I - 2\mu R)E[W_k] + 2\mu RW^*$$

This equation has the same form as equation (2.1.3-2), therefore it leads to a conclusion that the weight vector of the control-

ler in the NP-MRAC will converge to its minimum if a condition

$$0 < \mu < \frac{1}{\lambda_{max}}$$

is satisfied.

In this section the convergence of the tap-setting was examined under the assumption that the plant FIR was exactly known. In the situation where the plant FIR is not known, then an error will be introduced. For this case, the proof for convergence of the filtered-X LMS algorithm is not available yet. However simulation studies seem to show that the Non-Parametric Model-Reference Adaptive Control (NP-MRAC) does converge to the correct value. This is evident in chapter 3 where simulation studies of both the NP-PI and the NP-MRAC are presented.

CHAPTER 3

NP-PI AND NP-MRAC SIMULATIONS ON THE NEC-APC3 COMPUTER

In chapter 2, we have seen the theory of applying the FIR adaptive filter to Non-Parametric Plant Identification (NP-PI) and Non-Parametric Model-Reference Adaptive Control (NP-MRAC) . In this chapter we will demonstrate its effectiveness by simulation .

The test input signal, a pseudorandom binary signal, used in the simulations is discussed in section 3.1. Simulation studies are observed for the NP-PI in section 3.2 and for the NP-MRAC in section 3.3. All simulations were done on the NEC-APC-3 computer. All programs were written in Fortran 77 and their listings are included in the appendix B.

3.1 TEST INPUT SIGNALS

It has been shown in chapter two that the shape of the weight vector of the FIR filter would match with the shape of the FIR of the plant, only if the input signal to the FIR filter is a stationary wide-band white signal. The Pseudorandom Binary Signal (PRBS), though bandlimitted has characteristics which are close to a wide-band white signal (Speedy, Brown and Goodwin1970). In this work, unless specified otherwise PRBS was chosen as a test input signal throughout the simulations and the experiments. The PRBS is a periodic binary signal in which the switching between one level and the other takes place in a random manner, but is discretised in time by allowing the switching to take place only at mul-

tiples of period, T . A typical waveform of the PRBS is shown in figure 3.1-1.

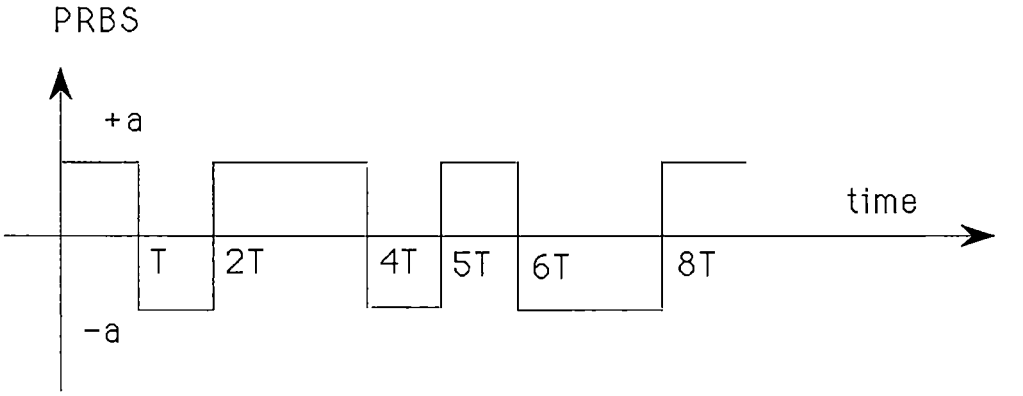


Figure 3.1-1 A typical section of a PRBS.

The auto-correlation of the PRBS is shown in figure 3.1-2. The function consists of an infinite series of triangular spikes centered at $\tau = kLT$ for $k = -\infty, \ldots, -1, 0, 1, \ldots + \infty$,

where L is the number of elements in one period of the PRBS.

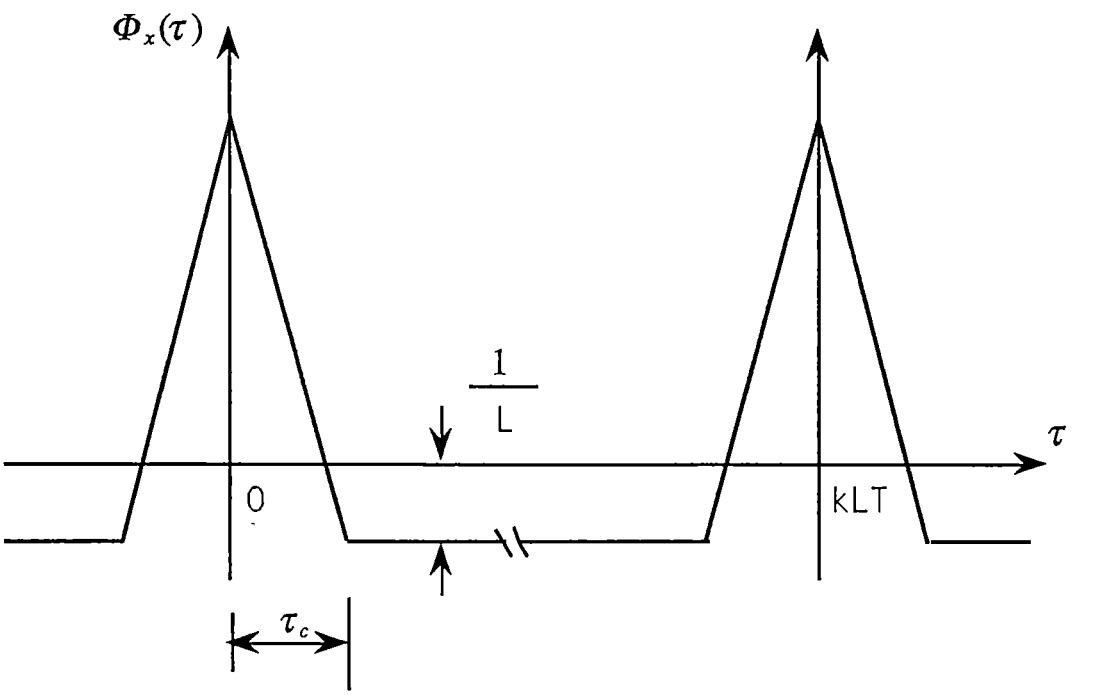


Figure 3.1-2 Auto-Correlation Function of an N-bit PRBS

The principle of generation of the PRBS has been presented by many authors . In this work a 10-bit PRBS generated by the software was used. The software consisted of the simulation of a10-bit shift register and an exclusive-or gate. The generation scheme is illustrated in figure 3.1-3. The signal is obtained from the 10^{th} stage output of the shift register.

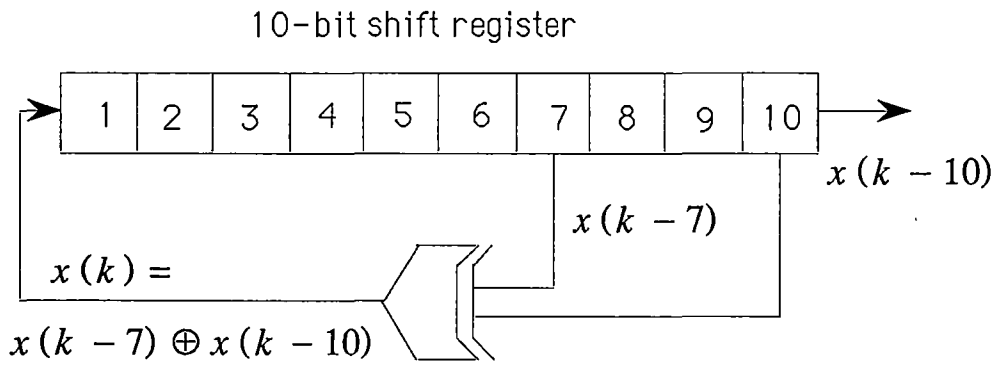


Figure 3.1-3 The principle of generation a 10-bit PRBS.

At the k^{th} iteration, the output of the exclusive or is

$$x(k) = x(k-7) \oplus x(k-10) \qquad (3.1-1)$$

where $\oplus$ denotes as the exclusive-or operation .

At the $(k+1)^{th}$ iteration, the contents of the shift registers are displaced to the right by one register, with the contents of the first register being replaced by $x(k)$, and with the contents of the tenth register $x(k-10)$ being output as an element of the PRBS. This result scheme produces a PRBS with L=1023 (= 2^{10}-1) elements in one period.

The PRBS subroutine was written in Fortran 77, and its listing is included in the appendix B.

3.2 A NON-PARAMETRIC PLANT IDENTIFICATION (NP-PI)

3.2.1 Plant models

Dynamic plants whether mechanical, electrical, thermal, hydraulic, economic, biological, etc., can be characterized by differential equations. For instance, a linear time-invariant plant can be defined by the following differential equation

$$a_0 y^{(n)} + a_1 y^{(n-1)} + \ldots + a_{n-1} \dot{y} + a_n y = b_0 x^{(m)} + b_1 x^{(m-1)} + \ldots + b_{m-1} \dot{x} + b_m x \qquad (n \geq m) \qquad (3.2.1-1)$$

where y is the output of the plant, x is the input of the plant, and a_i and b_j (where $0 \leq i \leq n$ and $0 \leq j \leq m$) are constants .

In control theory, a transfer functions is used to characterize the input-output relationship of the linear continuous-time time-invariant plant. It is defined to be the ratio of the Laplace transform of the output signal to the Laplace transform of the input signal, under the assumption that all initial conditions are zero. The transfer function of the plant represented by equation (3.2.1-1) is then obtained by taking the Laplace transforms of both sides of this equation , under the assumption that all initial conditions are zero. This is known as a continuous-time transfer function and is represented as

Transfer function $P(s)$=

$$\frac{y(s)}{x(s)} = \frac{b_0 s^m + b_1 s^{m-1} + \ldots + b_{m-1} s + b_m}{a_0 s^n + a_1 s^{n-1} + \ldots + a_{n-1} s + a_n} \qquad (3.2.1-2)$$

In this thesis, the FIR adaptive filter is realized in digital computer software, therefore for simulation studies the transfer function of the plant has to be discretized. Below is the descrip-

tion of the method to derive the discrete-time transfer function from the continuous-time transfer function.

The continuous-time transfer function of the plant is discritized by applying a Z-transform. The Z-transform has the same relationship to the linear time-invariant discrete-time plant as the Laplace transform has to the linear time-invariant continuous-time plant. The simple substitution

$$z = e^{sT} \qquad (3.2.1-3)$$

converts the Laplace transform to the Z-transform.

In computer control, a Digital to Analog (DA) converter is constructed so that it holds the analog signal constant until a new conversion is commanded. This structure can be modeled as a Zero Order Hold (ZOH) data-extrapolator. Its transfer function is

$$P_{ZOH}(s) = \frac{1 - e^{-sT}}{s} \qquad (3.2.1-4)$$

where T is the sampling interval.

Hence the resulting transfer function of the process including the ZOH data-extrapolator can be written as

$$P^*(s) = P_{ZOH}(s)P(s)$$

or

$$P^*(s) = \left(\frac{1 - e^{-sT}}{s}\right)\left(\frac{b_0 s^m + b_1 s^{m-1} + \ldots + b_{m-1}s + b_m}{a_0 s^n + a_1 s^{n-1} + \ldots + a_{n-1}s + a_n}\right) \qquad (3.2.1-5)$$

Using equation (3.2.1-3) , the discrete-time transfer function is then described as

$$P^*(z) = \frac{b_0^* + b_1^* z^{-1} + \ldots + b_m^* z^{-m}}{1 - a_1^* z^{-1} - a_2^* z^{-2} - \ldots - a_n^* z^{-n}} \quad (3.2.1-6)$$

where $P^*(z)$ represents a discrete-time transfer function of the plant which is represented as in equation (3.2.1-2).

a_i^* *and* b_j^* (*where* $0 \le i \le n$ *and* $0 \le j \le m$) are discrete-time constants of the plant .

Thus all poles in an s-plane of the plant are transferred into z-plane by equation (3.2.1-3).

Another way of the representation of the input-output relationship of the linear discrete-time time-invariant plant is by using shift operators. Shift-operators for discrete-time plants are equivalent to the use of differential operators for continuous-time plants. The back-ward shift-operator is defined as

$$q^{-1} f(k) = f(k-1) \quad (3.2.1-7)$$

where $\{f(k): k = -\infty, \ldots, -1, 0, 1, \ldots, +\infty\}$ is the infinite discrete sequence,

and k is a time index where the sampling period , T, is chosen as a time unit.

The discrete-time transfer function is then obtained using a state-space method. The state-space model including the ZOH device can be described using shift-operator, which is written as

$$P^*(q) = \frac{y_k}{x_k} = \frac{b_0^* + b_1^* q^{-1} + \ldots + b_m^* q^{-m}}{1 - a_1^* q^{-1} - a_2^* q^{-2} - \ldots - a_n^* q^{-n}} \quad (3.2.1-8)$$

where $P^*(q)$ represents a discrete-time transfer function of the plant which is represented as in equation (3.2.1-2).

y_K and x_K are the sampled output signal, y, and

input signal, x;

d_i^* and b_j^* (where $0 \leq i \leq n$ and $0 \leq j \leq m$) are equivalent to d_i^* and b_j^* in equation (3.2.1-6) .

In this chapter, plants to be identified were first selected as the continuous-time transfer-fuction. Then they were converted to the discrete-time transfer function, using the ZOH . The discrete-time transfer function was used for simulation.

3.2.2 Simulations of the NP-PI

Figure 3.2.2-1 shows the schematic configuration of an FIR adaptive filter used in the plant identification as discussed in chapter 2.

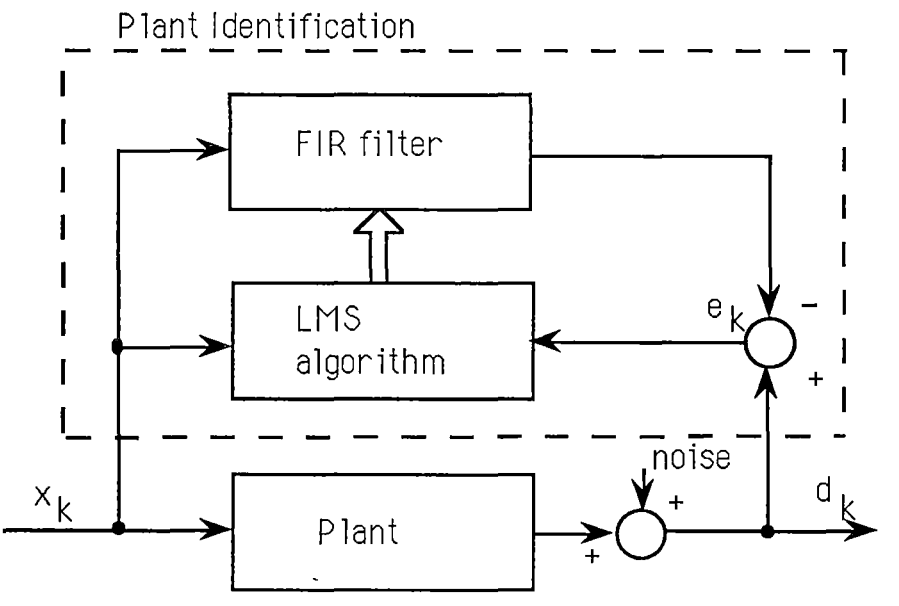


Figure 3.2.2-1: Block diagram of the NP-PI.

Simulations were done on two types of second order plants : one with complex poles and the other with two real poles. The number of taps of the FIR adaptive filter was adjustable. The simulation

results are attached here.

The first plant's continuous-time transfer-function was selected as

$$P(s) = \frac{w_n^2}{s^2 + 2\xi w_n s + w_n^2} \qquad (3.2.2-1)$$
$$where \quad \xi = 0.1 \quad and \quad w_n = 2$$

The sampling period was T_S=0.2 seconds. Using the ZOH equivalence , its discrete-time transfer-function was

$$P(q^{-1}) = \frac{0.151731q^{-1} - 0.000003q^{-2}}{1 - 1.771388q^{-1} + 0.923116q^{-2}} \qquad (3.2.2-2)$$

This plant was simulated on the NEC-APC-3 to test the ability of the FIR adaptive filter to track the shape of the impulse response of the plant.

Figure 3.2.2-2 shows the tap values of a 120-tap FIR filter and the impulse response of the plant. It was simulated using PRBS as a test input signal. (The mathematical model of the impulse response of the plant is discussed in the appendix B).

Figure 3.2.2-3 is similar to figure 3.2.2-2 . However, it was simulated using a square wave signal as the test input signal.

It is clear from these figures that the PRBS is a better signal to identify the plant, as properties of the PRBS are approximately same as the white noise (refer to section 3.1).

The following two pages are figures 3.2.2-2 and 3.2.2-3 , which show

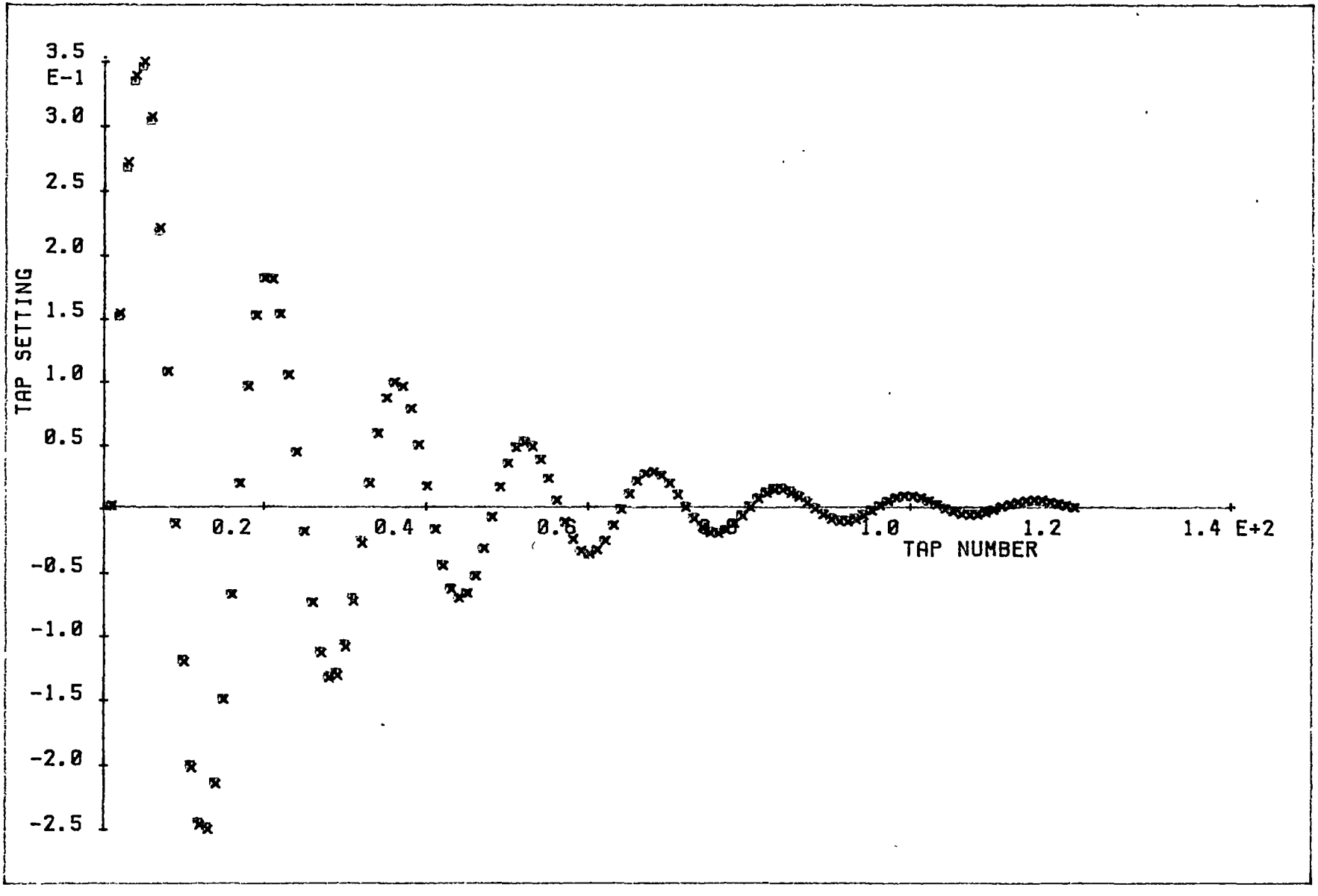
the shape of the weight vector of the 120-tap FIR adaptive filter , × , and

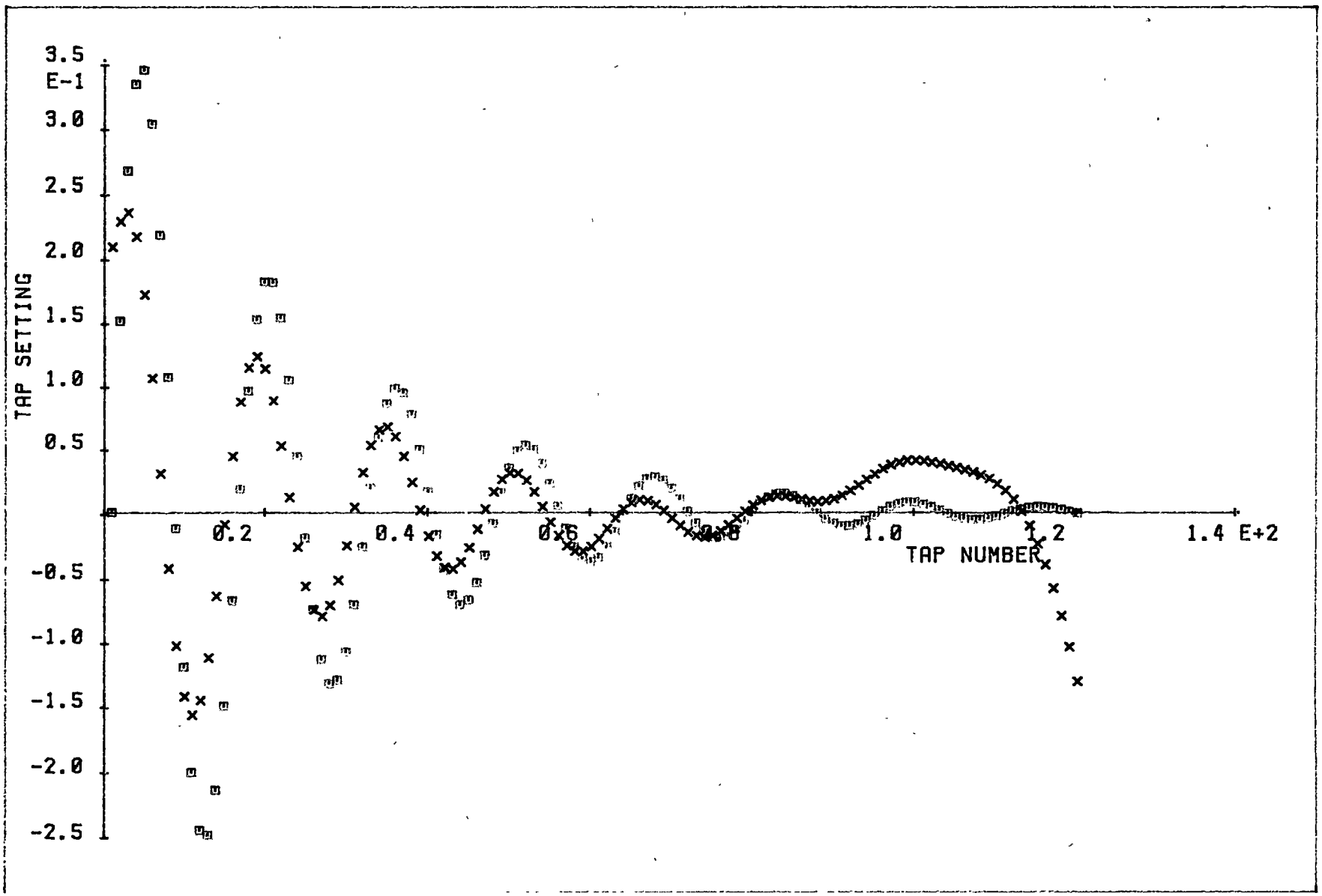
the finite impulse response of the plant , ⊡ .

The first graph ,figure 3.2.2-2 , was plotted for the NP-PI with the PRBS using as the test input signal.

The second graph ,figure 3.2.2-3 , was plotted for the NP-PI with square wave (period of 800 iterations) as the test input signal.

Both simulations were done with $\mu = 0.002$. The graphs are the response of the system after the end of the first PRBS period (1023 iterations).


3.5
E-1
3.0
2.5
2.0
1.5
1.0
0.5
-0.5
-1.0
-1.5
-2.0
-2.5
TAP SETTING
0.2
0.4
0.6
1.0
1.2
1.4 E+2
TAP NUMBER


3.5
E-1
3.0
2.5
2.0
1.5
1.0
0.5
-0.5
-1.0
-1.5
-2.0
-2.5
TAP SETTING
0.2
0.4
0.6
0.8
1.0
1.2
1.4 E+2
TAP NUMBER

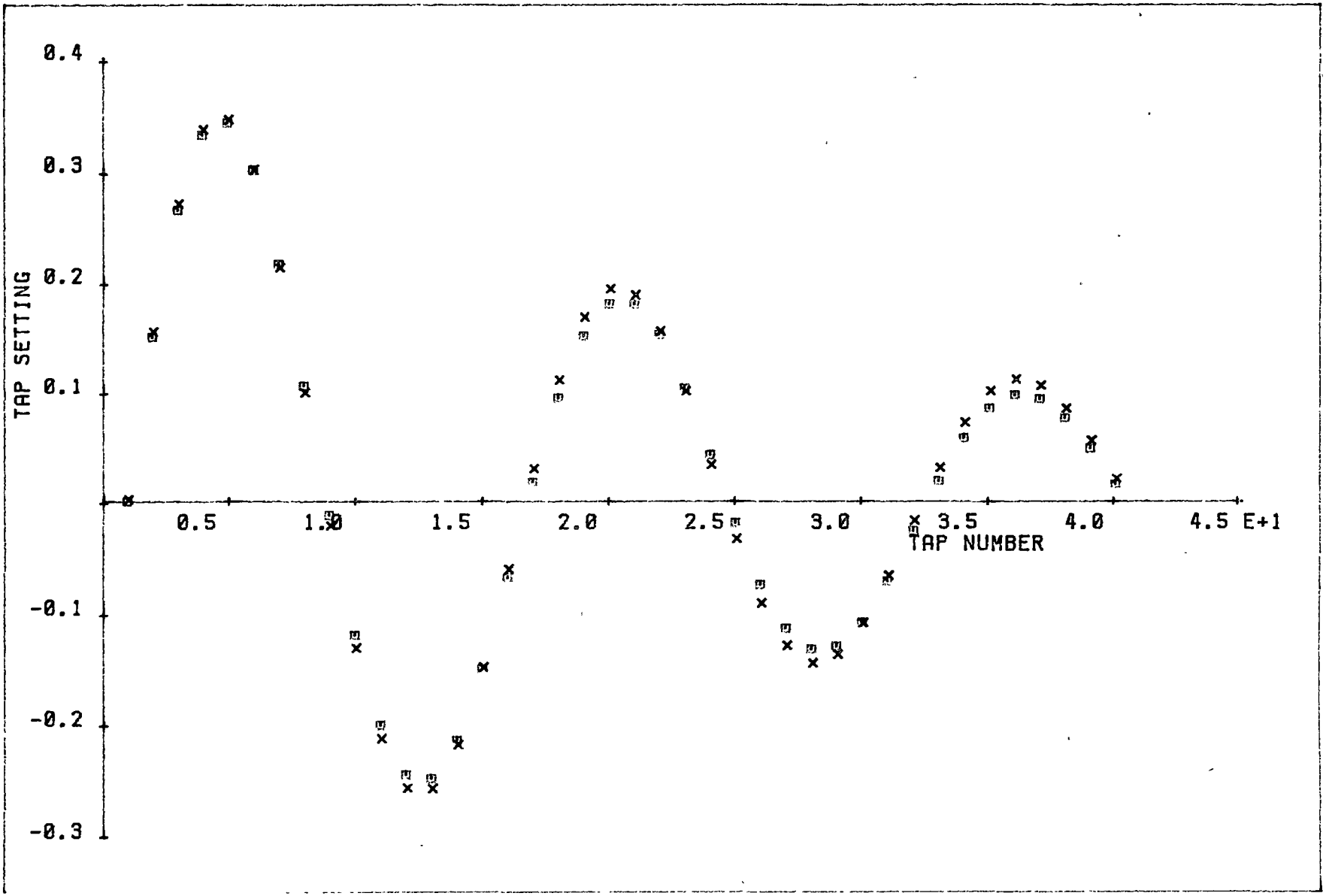
Figures 3.2.2-4a and 3.2.3-4b show the tap values after convergence of the 40-tap FIR filter and the 80-tap FIR filter respectively using the PRBS as the plant input. These simulations show that a better result can be obtained by using the FIR adaptive filter with more taps.

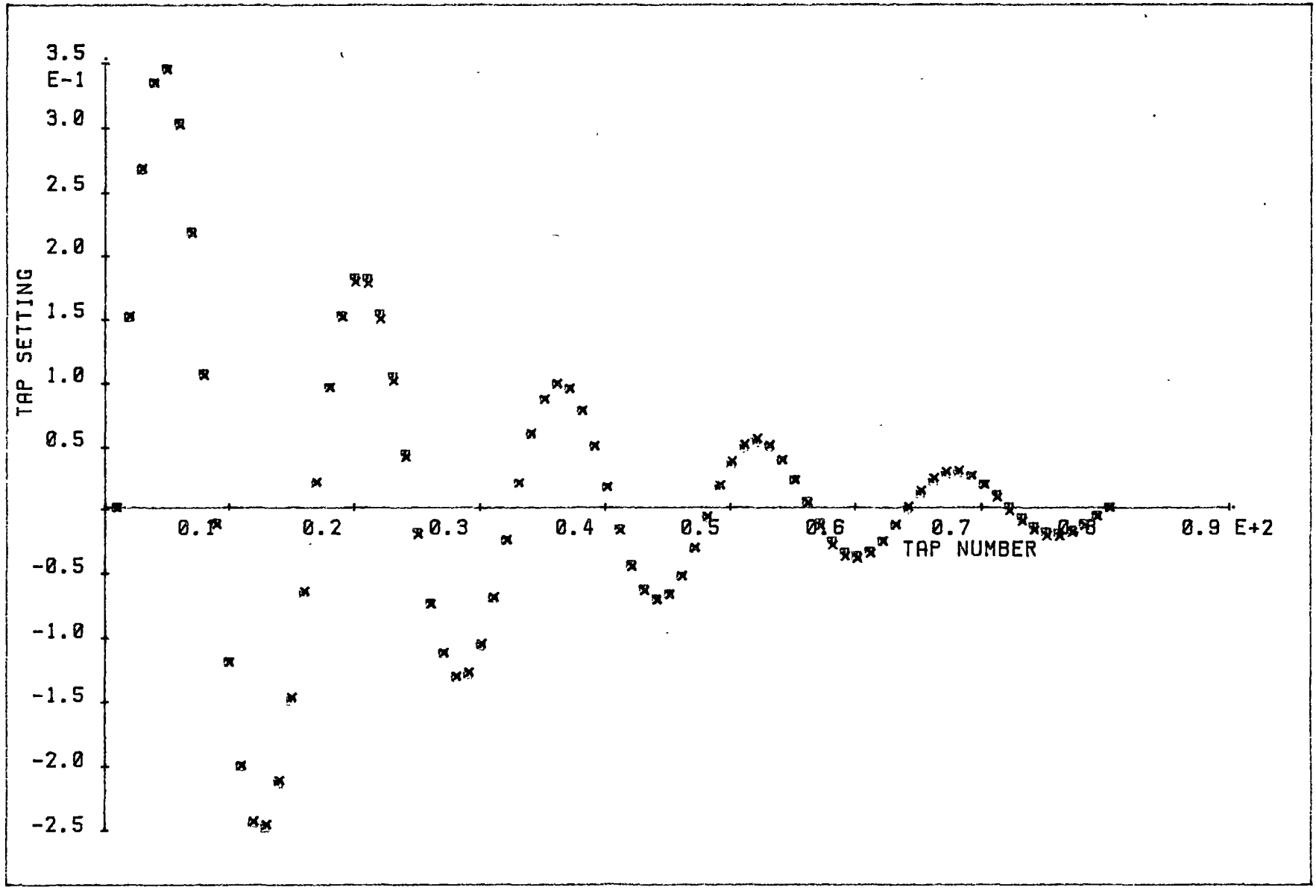
The following two pages are figure 3.2.2-4 , which shows **the shape of the weight vector of the FIR adaptive filter , X , and the finite impulse response of the plant , 回 . .**

The first graph, figure 3.2.2-4a, was plotted for the NP-PI with the 40-tap FIR adaptive filter.

The second graph,figure 3.2.2-4b, was plotted for the NP-PI with the 80-tap FIR adaptive filter.

Both simulations were done with μ = 0.002. The graphs are the response of the system after the end of the first PRBS period (1023 iterations).


0.4
0.3
0.2
0.1
-0.1
-0.2
-0.3
TAP SETTING
0.5
1.0
1.5
2.0
2.5
3.0
3.5
4.0
4.5 E+1
TAP NUMBER


3.5
E-1
3.0
2.5
2.0
1.5
1.0
0.5
-0.5
-1.0
-1.5
-2.0
-2.5
TAP SETTING
0.1
0.2
0.3
0.4
0.5
0.6
0.7
0.8
0.9 E+2
TAP NUMBER

The second plant is represented in continuous-time transfer function by

$$P(s) = \frac{1}{(0.13s + 1)(3.87s + 1)} \qquad (3.2.2-3)$$

Its discrete-time transfer function,With the sampling period Ts = 0.2 seconds, becomes

$$P(q^{-1}) = \frac{0.8248215q^{-1} + 0.0147311q^{-2}}{1 - 1.164344q^{-1} + 0.2038968q^{-2}} \qquad (3.2.2-4)$$

The simulations using the 80-tap FIR adaptive filter. Figures 3.2.2-5 and 3.2.2-6 show PRBS as the test input signal and square waveform as the test input signal,respectively.

The following two pages are figures 3.2.2-5 and 3.2.2-6, which show

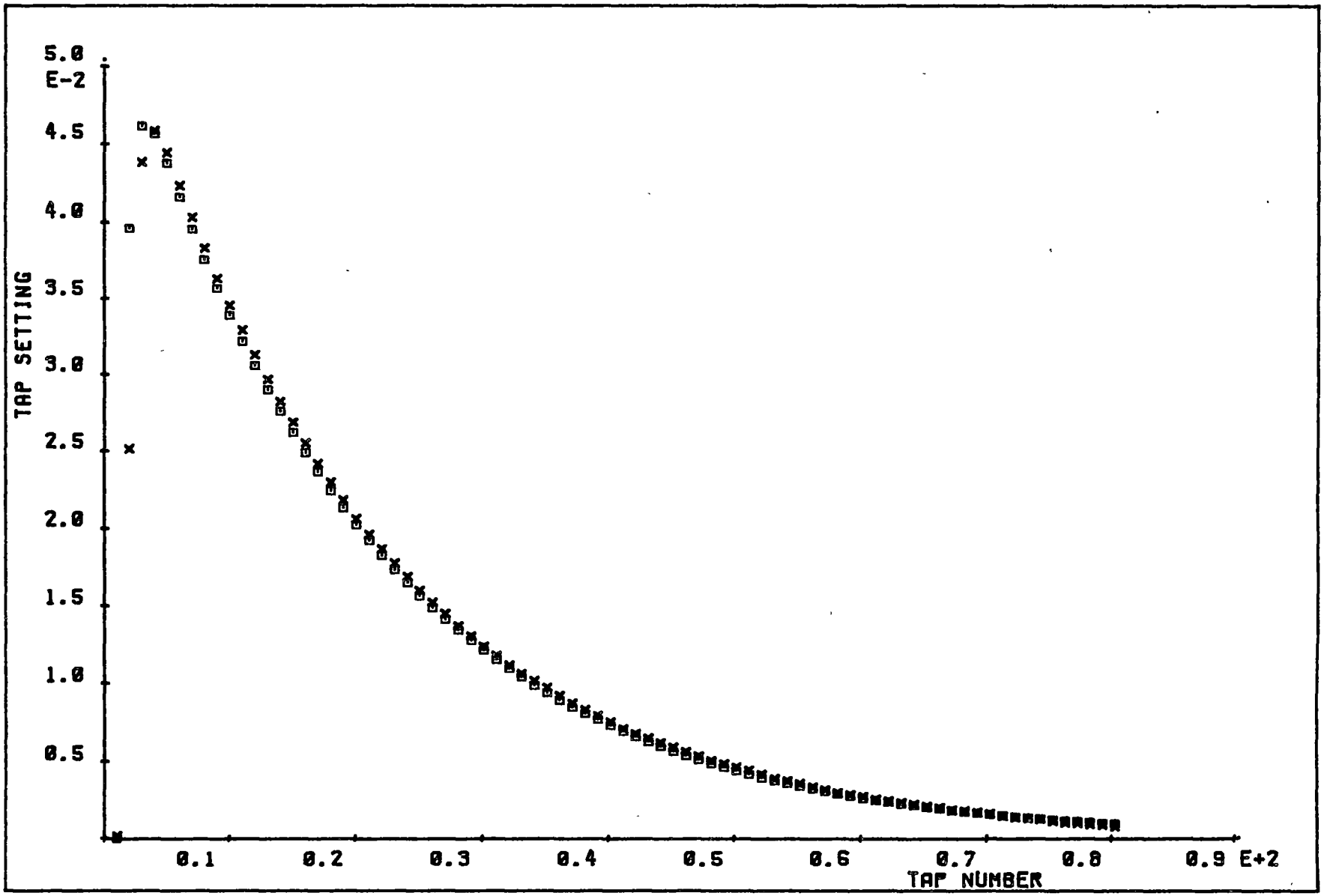
the shape of the weight vector of the 80-tap FIR adaptive filter , X , and

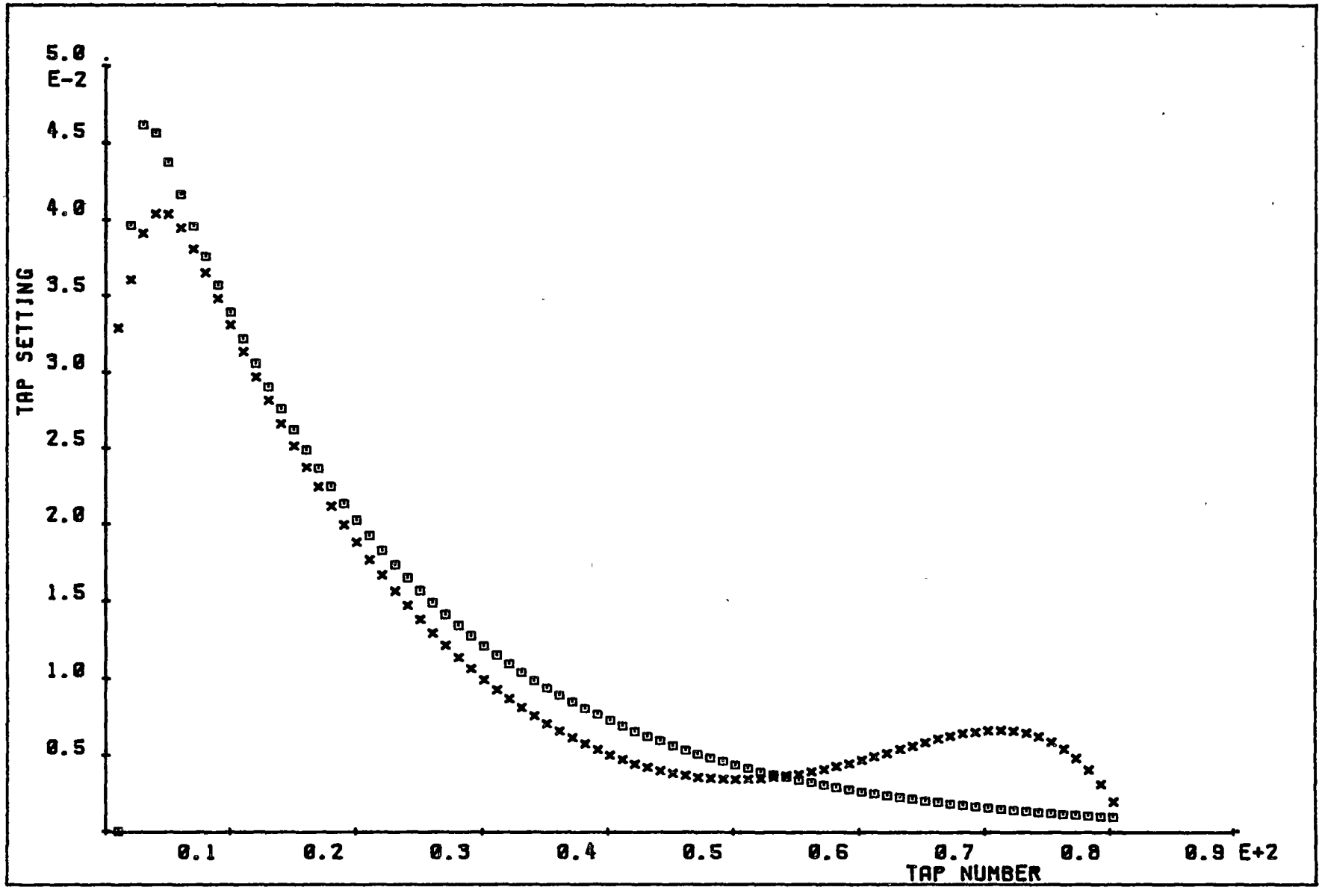
the finite impulse response of the plant , [2] .

The first graph,figure 3.2.2-5 ,was plotted for the NP-PI with PRBS as the test input signal. The graphs are the response of the system after the end of the first PRBS period (1023 iterations).

The second graph,figure 3.2.2-6 , was plotted for the NP-PI with square wave (period of 800 iterations) as the test input signal. The graphs are the response of the system after the end of the third PRBS period (2400 iterations).

Both simulations were done with $\mu = 0.002$.


5.0
E-2
4.5
4.0
3.5
3.0
2.5
2.0
1.5
1.0
0.5
TAP SETTING
0.1
0.2
0.3
0.4
0.5
0.6
0.7
0.8
0.9 E+2
TAP NUMBER


5.0
E-2
4.5
4.0
3.5
3.0
2.5
2.0
1.5
1.0
0.5
TAP SETTING
0.1
0.2
0.3
0.4
0.5
0.6
0.7
0.8
0.9 E+2
TAP NUMBER

3.3 OPEN LOOP NON-PARAMETRIC MODEL-REFERENCE ADAPTIVE CONTROL (NP-MRAC)

Figure 3.3-1 illustrates the position of the LMS algorithm in the open loop NP-MRAC which was discussed in chapter 2. The adaptive control scheme was simulated using a 40-tap FIR adaptive filter to identify the plant and to generate the controller.

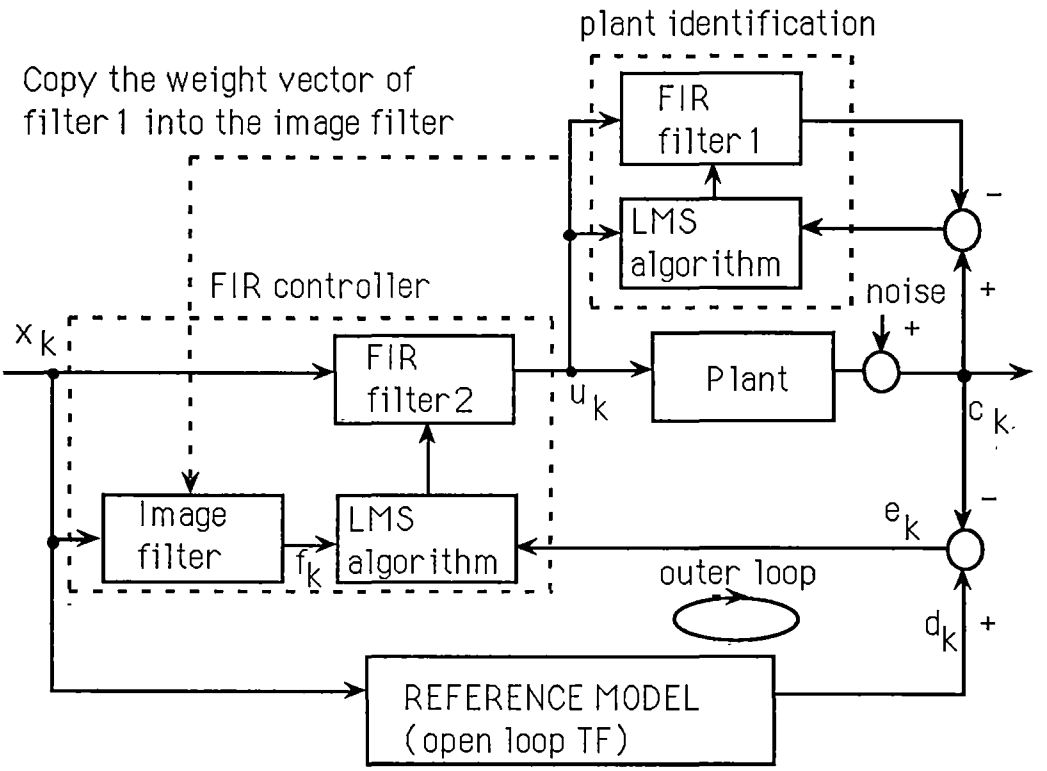


Figure 3.3-1: Block diagram of the open loop NP-MRAC.

Plant with parameter variation and time delay variation was simulated on the digital computer as follow.

(i) To test for parameter variation the transfer function of the plant was initially set as in equations (3.3-1). The change in plant parameter was simulated by changing the transfer function to equation (3.3-2). The two discrete transfer functions were selected so that the overshoot was 40% for the first one (equation 3.3-1) and 140% for the second one (equation 3.3-2).

$$p_1(q^{-1}) = \frac{0.22(q^{-1} + 2q^{-2} + q^{-3})q^{-2}}{1 - 0.75q^{-1} + 0.64q^{-2}} \qquad (3.3-1)$$

$$p_2(q^{-1}) = \frac{2.4q^{-d_p}(q^{-1} - 0.8q^{-2})}{1 - 0.1q^{-1} - 0.42q^{-2}} \qquad (3.3-2)$$

$$where \quad d_p = 3$$

Figure 3.3-2 demonstrates the ability of the open loop NP-MRAC to control the plant with parameter variation. The plant was switched from equation (3.3-1) to (3.3-2). The reference model pulse transfer function (equation 3.3-3) was selected in the discrete form such that its overshoot was 16%.

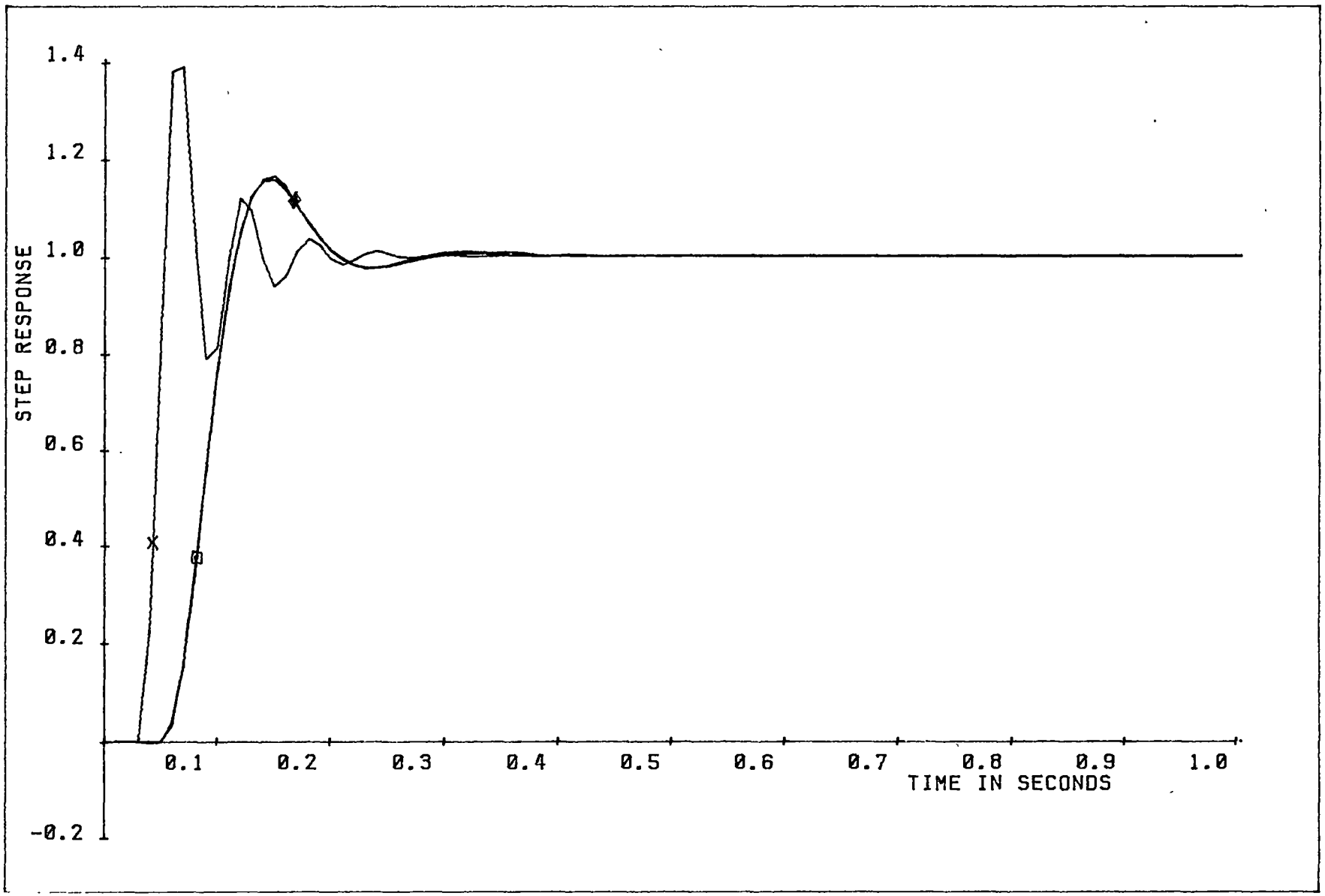
$$M(q^{-1}) = \frac{0.03q^{-5} + 2q^{-6} + q^{-7}}{1 - 1.53q^{-1} + 0.66q^{-2}} \qquad (3.3-3)$$

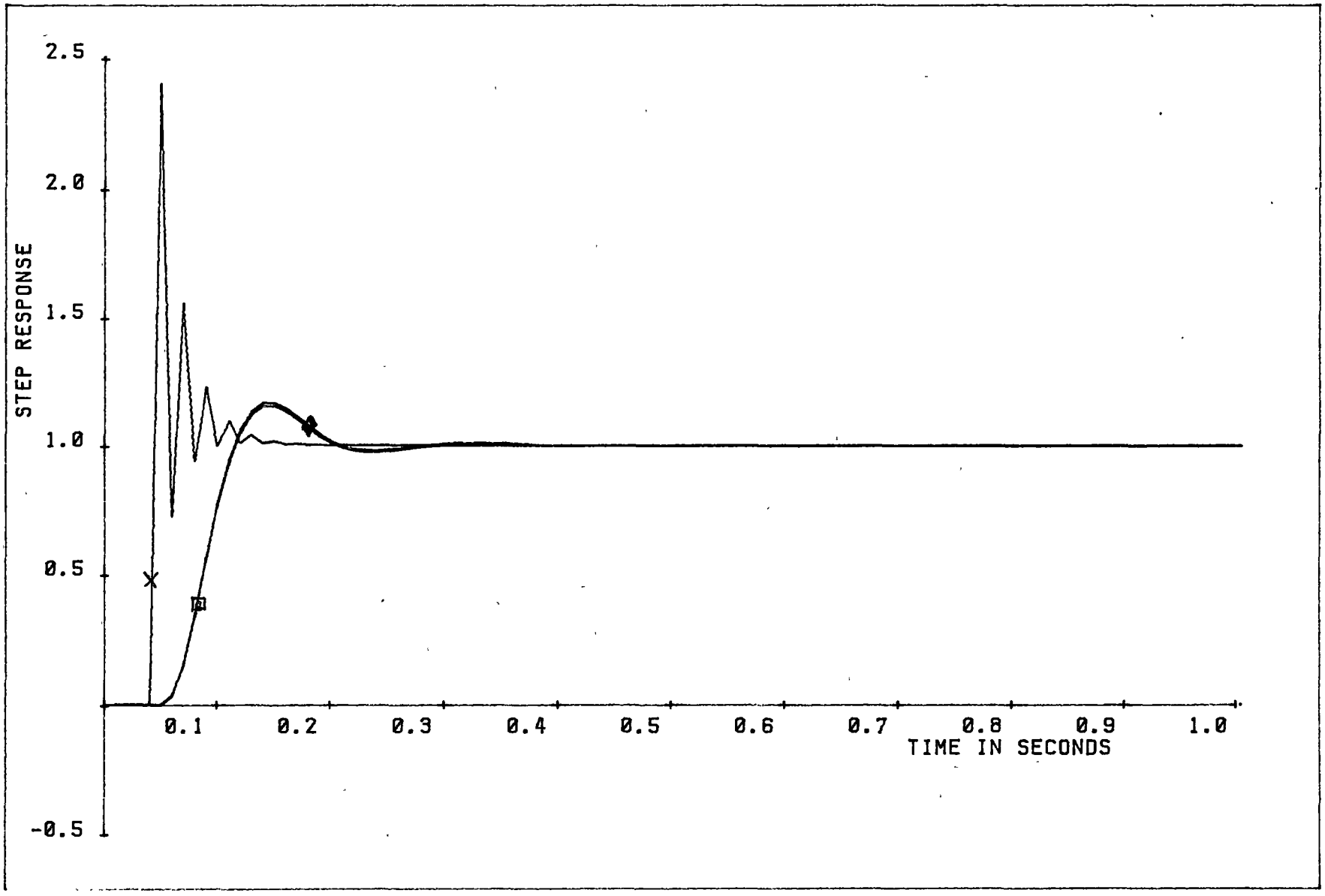
The following two pages is figure 3.3-2, which shows **the step response of the uncompensated plant, ×, compensated plant, ⊡, and reference model, ◆.**

The first graph was plotted for the plant with the transfer function of equation (3.3-1) which was plotted after 9 periods of the PRBS, and

the second graph was plotted for the plant with the transfer function of equation (3.3-2) which was plotted 6 periods of the PRBS after the change in parameters of the plant.

Both simulations were done with $\mu = \mu_p = 0.00004$.


1.4
1.2
1.0
0.8
0.6
0.4
0.2
-0.2
STEP RESPONSE
0.1
0.2
0.3
0.4
0.5
0.6
0.7
0.8
0.9
1.0
TIME IN SECONDS


2.5
2.0
1.5
1.0
0.5
-0.5
STEP RESPONSE
0.1
0.2
0.3
0.4
0.5
0.6
0.7
0.8
0.9
1.0
TIME IN SECONDS

(ii) For time delay variation, figure (3.3-3) demonstrates the ability of the open loop NP-MRAC to control the plant with time delay variation. The plant's transfer function was given by equation (3.3-2) with time delay (d_p) switched from 3 to 7, while the model pulse transfer function was given in the discrete form as in equation (3.3-4).

$$M(q^{-1}) = \frac{0.25q^{-11}}{1 - q^{-1} + 0.25q^{-2}} \qquad (3.3-4)$$

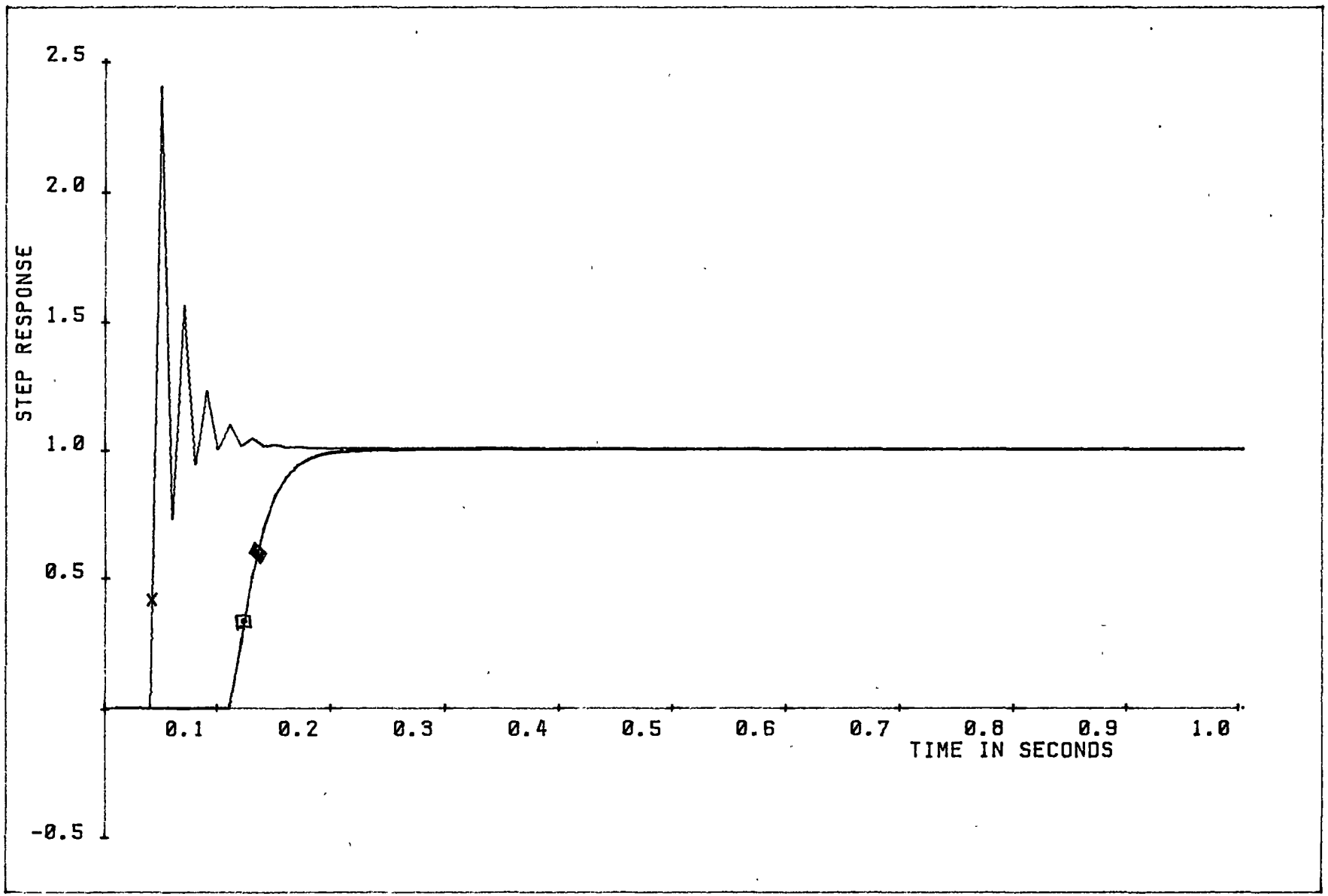
Tap values of the FIR adaptive controller are shown in figure (3.3-4) . Observation shows that as time delay of the plant was increased from 3 to 7 sampling time units, the tap values shifted 4 units to the left. This indicates that in the NP-MRAC, for large time delay variations the shape of the taps of the FIR filter will be easier to adapt and the NP-MRAC is more stable then the Self Tuning Controller (STC). Because in STC, to allow for large time delay variation, a large number of parameters must be estimated as the order of the numerator polynomial is increased by an upper limit of the time delay. Further, due to round off error in computation, estimation of zero parameters of the numerator polynomial becomes difficult hence imperfect pole-zero cancellation may take place which gives rise to instability.

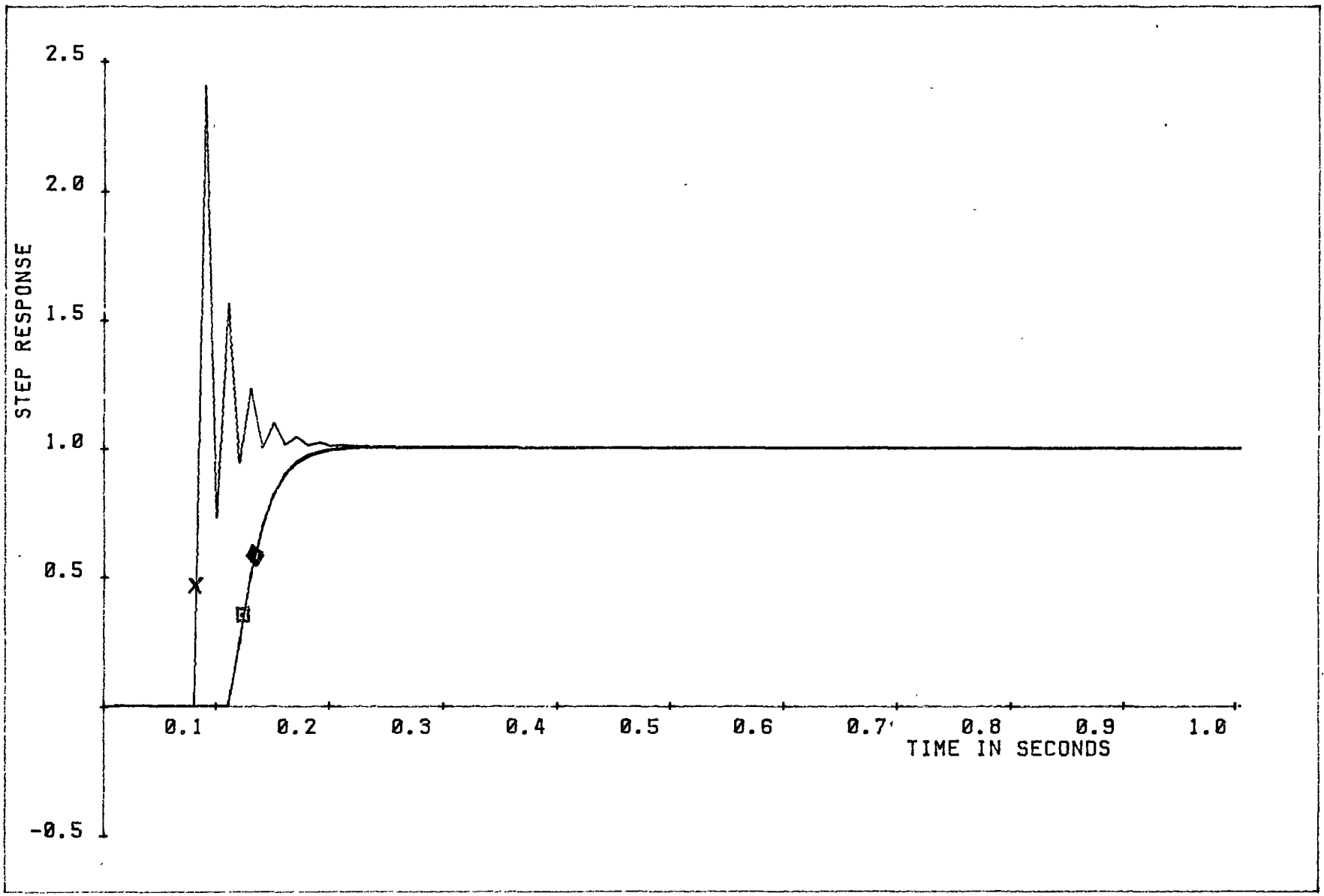
The following two pages is figure 3.3-3 ,which shows **the step response of the uncompensated plant , × , compensated plant, □ , and reference model, ◆ .**

The first graph is the plot of the response of the plant with 3 sampling time unit of the time-delay after 9 periods of the PRBS.

The second graph is the plot of the response of the plant 6 periods of PRBS after its time-delay changed from 3 ton 7 sampling time unit.

Both simulations were done with $\mu = 0.00004$.


2.5
2.0
1.5
1.0
0.5
-0.5
STEP RESPONSE
0.1
0.2
0.3
0.4
0.5
0.6
0.7
0.8
0.9
1.0
TIME IN SECONDS

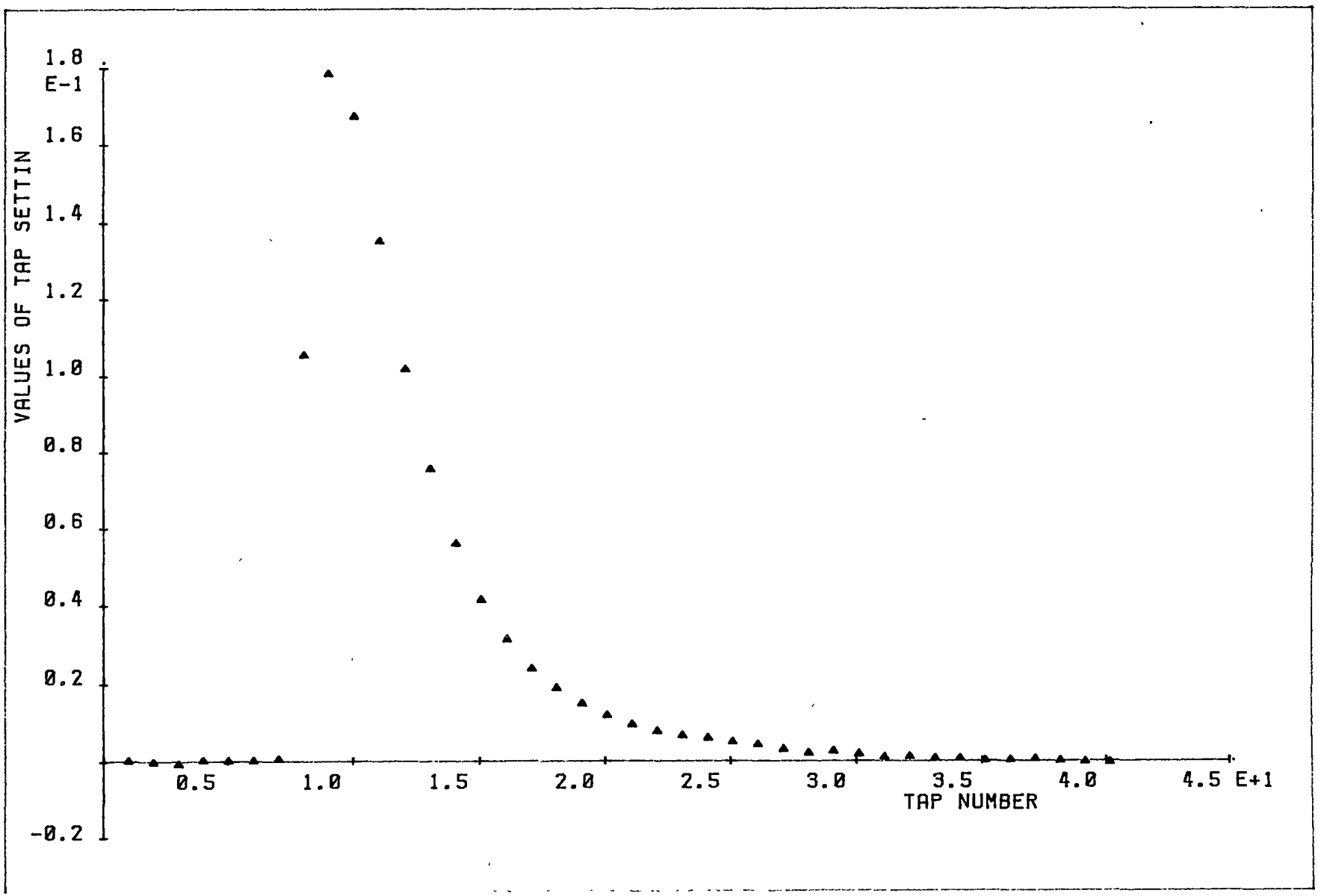

2.5
2.0
1.5
1.0
0.5
-0.5
STEP RESPONSE
0.1
0.2
0.3
0.4
0.5
0.6
0.7
0.8
0.9
1.0
TIME IN SECONDS

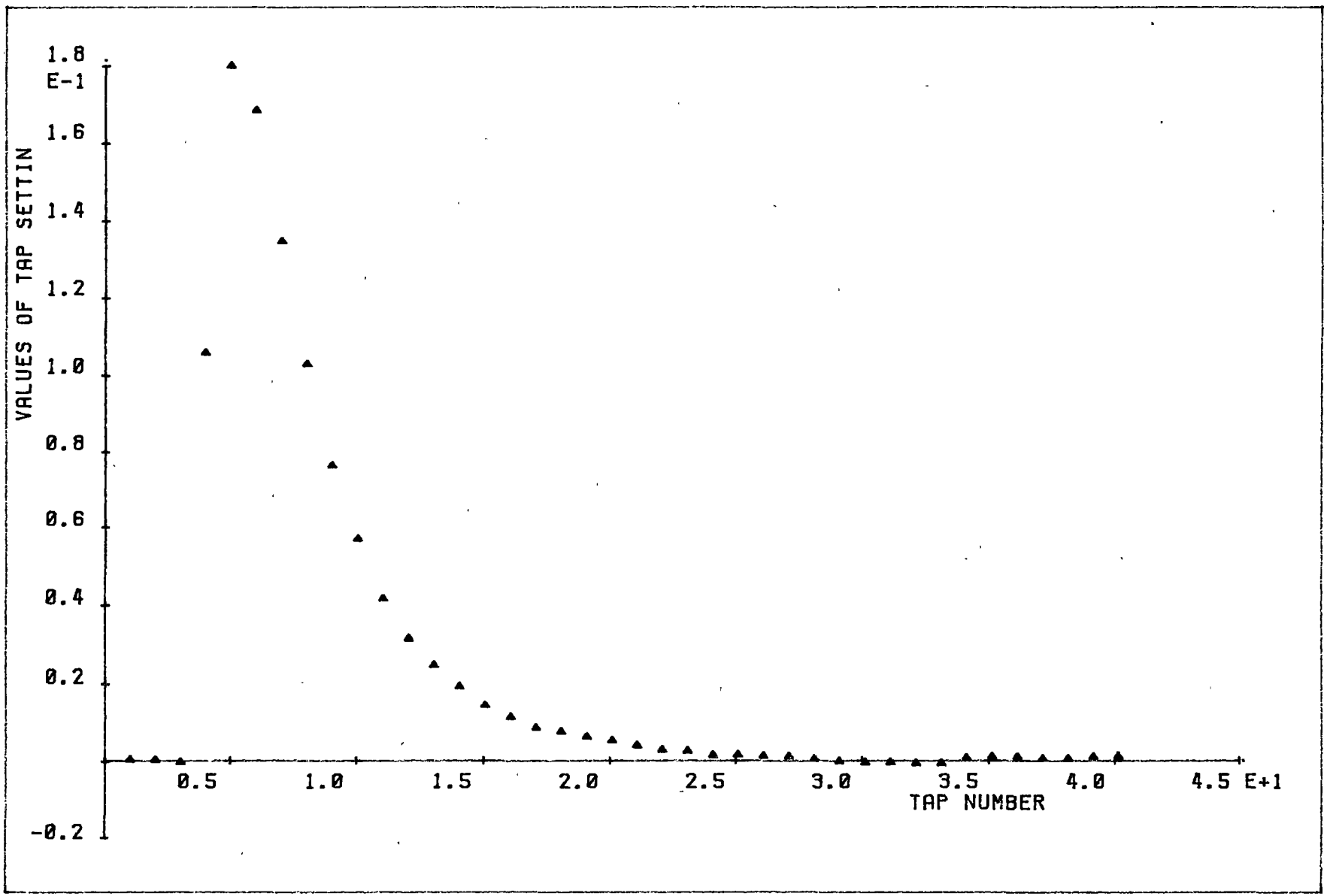
The following two pages is figure 3.3-4 , which shows **the tap setting of the FIR adaptive filter using as the controller to the plant with time delay variation.**

The first graph was plotted , when the time delay of the plant was 3 units.

The second graph was plotted , when the time delay of the plant was 7 units.

They are the shape of the controller's tap values for the compensated plant in figure 3.3-3.


1.8
E-1
1.6
1.4
1.2
1.0
0.8
0.6
0.4
0.2
-0.2
VALUES OF TAP SETTIN
0.5
1.0
1.5
2.0
2.5
3.0
3.5
4.0
4.5 E+1
TAP NUMBER


1.8
E-1
1.6
1.4
1.2
1.0
0.8
0.6
0.4
0.2
-0.2
VALUES OF TAP SETTIN
0.5
1.0
1.5
2.0
2.5
3.0
3.5
4.0
4.5 E+1
TAP NUMBER

Simulations indicate that the open loop NP-MRAC based on the FIR adaptive filter and the LMS algorithm has excellent adaptation capabilities in the presence of both parameter and time delay variations. However the main drawback of this scheme is its relative slowness to detect additive noise as described below. The plant is often subjected to disturbance and hence an additive noise is introduced at the plant output (known as a plant noise, as shown in figure 3.3-1. Therefore, even though the NP-MRAC is used to estimate the error, e_k , to actuate the adaptive process thereby closing the feedback loop (namely an outer loop, see figure 3.3-1), we still refer to it as the open loop NP-MRAC, because the the adaptive process is slow to detected the plant noise. To overcome the above drawback, in the next chapter the plant feedback is applied, the new system then known as a closed-loop NP-MRAC. An advantage of the closed-loop NP-MRAC is that the use of the feedback makes its response relatively insensitive to external disturbances.

CHAPTER 4

CLOSED LOOP NON PARAMETRIC - MODEL REFERENCE ADAPTIVE CONTROL

So far we have seen the method and simulation results of the open loop NP-MRAC. Although the theory of the closed loop NP-MRAC has not yet been developed , simulation studies (will be discussed later in this chapter) have shown that an FIR adaptive filter can be applied to a model-reference adaptive control in a closed loop form.

This chapter will describe the simulation studies of the NP-MRAC applied to a closed loop system. The method is discussed in section 4.1. The excellent adaptation capabilities of the closed loop NP-MRAC in the presence of both parameter and time delay variation is demonstrated in section 4.2. The closed loop NP-MRAC has good set point tracking property, because it can both allow for closed loop pole placement and place no restrictions on the closed loop zeros. The set point tracking property is demonstrated in section 4.3.

To handle a plant which includes integrators, small modification of the NP-MRAC is required. In section 4.4, the modification is discussed and also simulation study is included.

4.1 THE CLOSED LOOP NP-MRAC:

The NP-MRAC that has been discussed in chapter 2 and chapter 3 refers to as an open loop system. Because the system consists of only one loop , namely outer loop, that adjusts the controller

weight vector in such a way that the error, e_k, between the plant output and reference model output becomes small; the inner loop consisting of the plant and the linear feedback controller is left out. In this section, the closed loop NP-MRAC which includes both outer loop and inner loop is introduced.

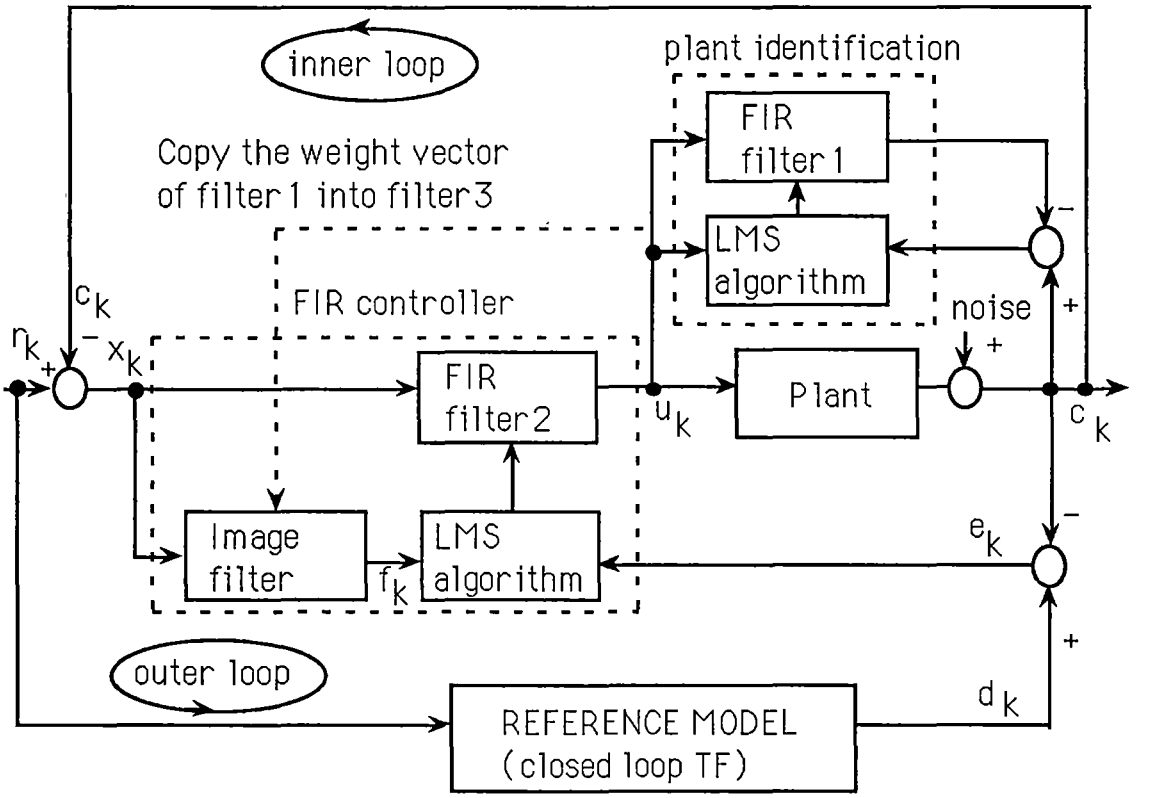


Figure 4.1-1: Block diagram of the closed loop NP-MRAC using a ref. model with closed loop transfer function

Figures 4.1-1 and 4.1-2 illustrate the closed loop NP-MRAC. As shown, the plant output signal is feedback to the system and is subtracted from the input signal; the difference ($x_k=r_k-c_k$) then passes through the adaptive controller.

The reference model can be chosen to have either the closed loop transfer function or the open loop transfer function characteristic. When the closed loop transfer function model is used ,

the input signal, r_k, is applied to the model. This is illustrated in figure 4.1-1. If on the other hand the open loop transfer function model is used, then the difference ($x_k=r_k-c_k$) is applied to the reference model , as seen in figure 4.1-2.

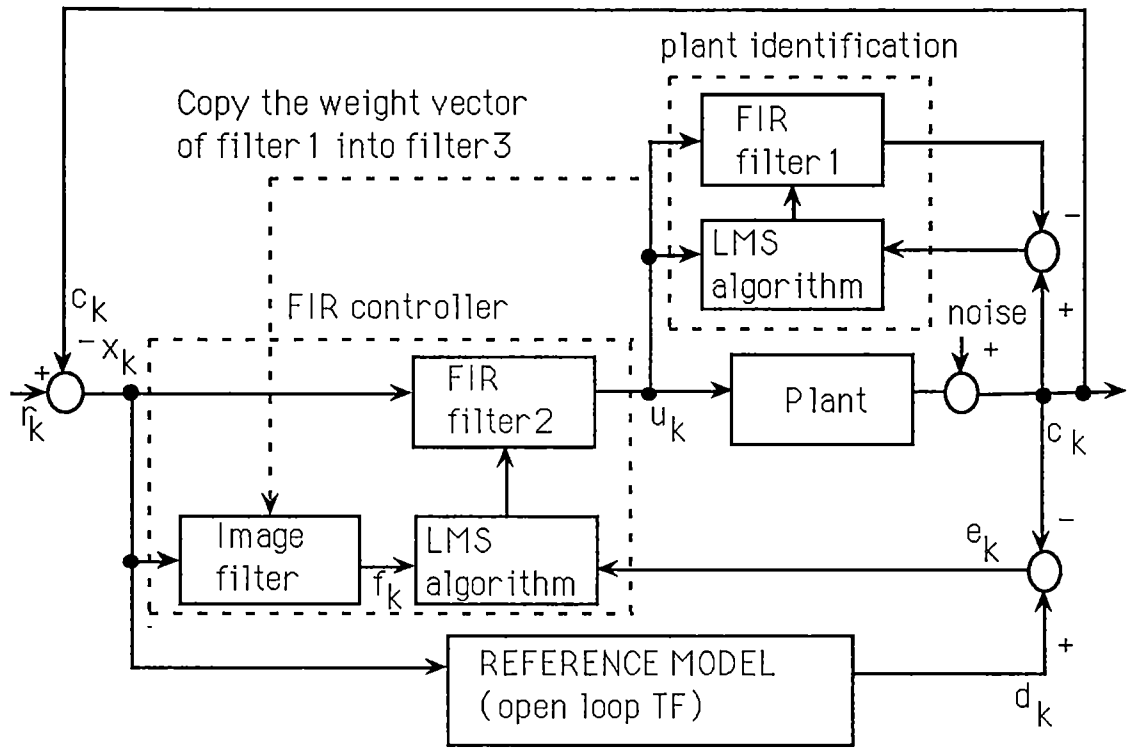


Figure 4.1-2: Block diagram of the closed loop NP-MRAC using a reference model with open loop transfer function

4.2 TRACKING ABILITY OF THE CLOSED LOOP NP-MRAC TO THE PLANT WITH BOTH PARAMETER AND TIME DELAY VARIATIONS:

The closed loop NP-MRAC was simulated using the FIR adaptive filter with 40 taps for both the plant identification and the controller synthesis .

A - The ability to track plant with parameter variation was observed on the closed loop NP-MRAC. The reference model was selected so that its closed loop pole is stable and is placed at q=0.8.

The closed loop transfer function of the reference model , as-shown in figure 4.1-1, was selected in the discrete-time transfer function

$$M(q^{-1}) = \frac{0.06q^{-5}}{1 - 0.8q^{-1}} \qquad (4.2-1)$$

The open loop transfer function reference model, as shown in figure 4.2-2, was selected

$$M(q^{-1}) = \frac{0.06q^{-5}}{1 - 0.8q^{-1} - 0.06q^{-5}} \qquad (4.2-2)$$

Two continuous-time transfer functions were selected as

$$P_1(s) = \frac{8.5}{(s+2)(s^2 + 4s + 4.25)} \qquad (4.2-3)$$

and

$$P_2(s) = \frac{w_n^2}{s^2 + 2\xi w_n s + w_n^2} \qquad (4.2-4)$$

$$where \quad \xi = 0.1 \quad and \quad w_n = 2$$

They were sampled at T_s=0.2s. Their discrete time transfer functions were

$$P_1(q^{-1}) = \frac{0.00842q^{-1} + 0.02500q^{-2} + 0.004619q^{-3}}{1 - 2.0043q^{-1} + 1.34345q^{-2} - 0.301194q^{-3}} \qquad (4.2-5)$$

and

$$P_2(q^{-1}) = \frac{0.151731q^{-1} - 0.000003q^{-2}}{1 - 1.771388q^{-1} + 0.923116q^{-2}} \qquad (4.2-6)$$

The output responses of the model and the plant to a square wave input are shown in figures 4.2-1 and 4.2-2 for model equation (4.2-1) and equation (4.2-2), respectively. The following salient points are noted :

(1) The first period is the output of the compensated plant whose transfer function was selected as in equation (4.2-5).

(2) The second period is the output of the plant when its transfer function was switched from equation (4.2-5) to (4.2-6) and the weight vector of the FIR controller (FIR filter-2) hadn't yet been adapted.

(3) The third period is the output of the plant, when the weight vector of the FIR controller readjusted itself to control the plant with the new transfer function (equation 4.2-6).

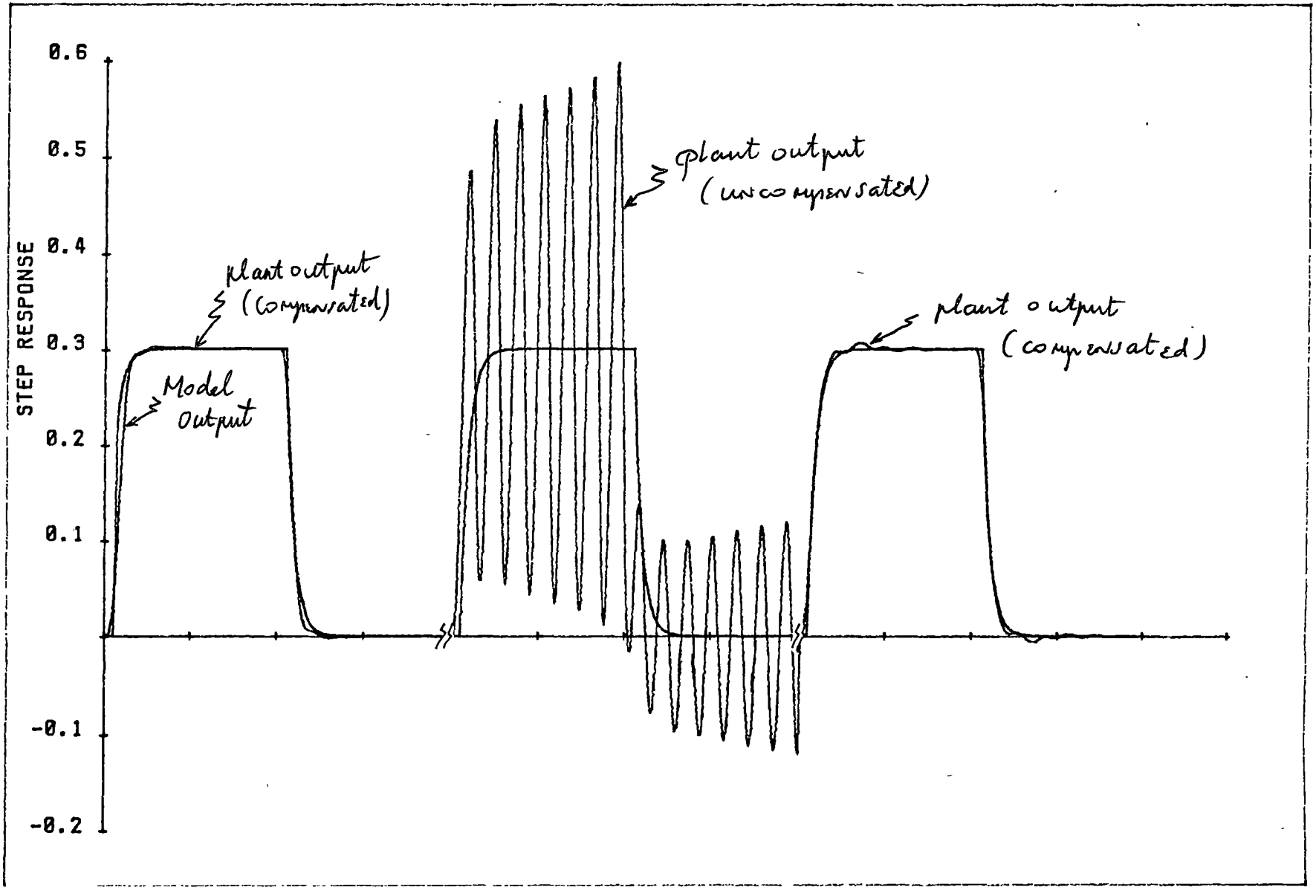
The following two pages are figures 4.2-1 and 4.2-2 ,

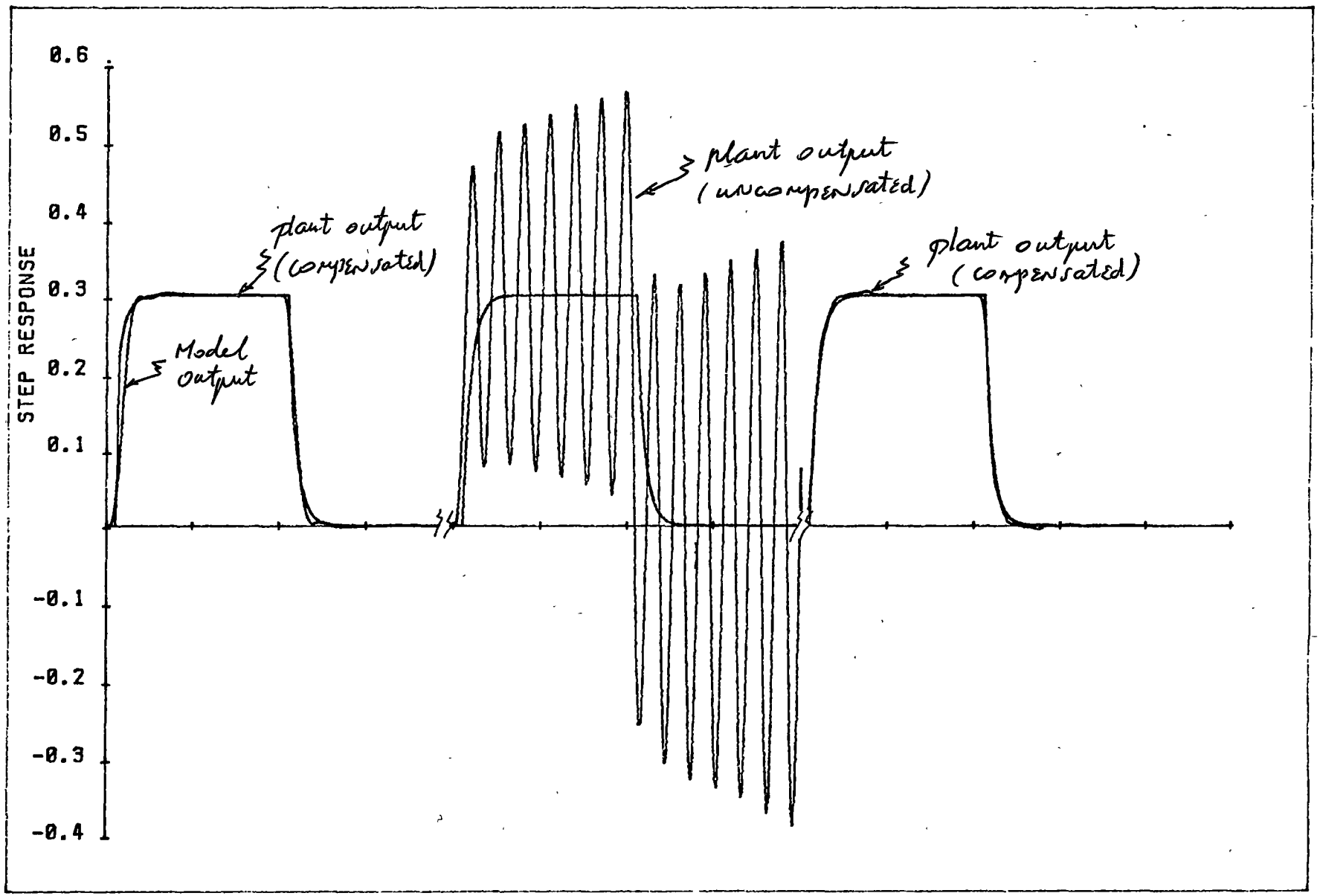
"The output responses of the model and the plant to a square wave input".

The first graph (figure 4.2-1) was plotted for the closed loop NP-MRAC with the closed loop reference model of equation 4.2-1.

The second graph (figure 4.2-2) was plotted for the closed loop NP-MRAC with the open loop reference model of equation 4.2-2.

Both simulations were done with $\mu = \mu_p = 0.002$.


STEP RESPONSE
0.6
0.5
0.4
0.3
0.2
0.1
-0.1
-0.2
plant output (uncompensated)
plant output (compensated)
Model output
plant output (compensated)


STEP RESPONSE
0.6
0.5
0.4
0.3
0.2
0.1
-0.1
-0.2
-0.3
-0.4
plant output (compensated)
Model output
plant output (uncompensated)
plant output (compensated)

B-The ability to track plant with both parameter and time delay variations was tested on the closed loop NP-MRAC, whose reference model was represented in a closed-loop transfer-function form (figure 4.1-1). The plant transfer function was switched from equation (4.2-7) to (4.2-8).

$$P_1(s) = \frac{2.5}{(s+1)(s^2+s+1.25)} \qquad (4.2-7)$$

$$P_2(s) = \frac{8.5\ e^{-0.6s}}{(s+2)(s^2+4s+4.25)} \qquad (4.2-8)$$

Their discrete-time transfer functions, sampled at a sampling rate of T_s=0.2s , were as in equation (4.2-9) and (4.2-10), respectively.

$$P_1(q^{-1}) = \frac{0.00301q^{-1} + 0.01088q^{-2} + 0.00247q^{-3}}{1 - 2.59233q^{-1} + 2.27083q^{-2} - 0.67032q^{-3}} \qquad (4.2-9)$$

$$P_2(q^{-1}) = \frac{0.00842q^{-4} + 0.02500q^{-5} + 0.004619q^{-6}}{1 - 2.0043q^{-1} + 1.34345q^{-2} - 0.301194q^{-3}} \qquad (4.2-10)$$

The closed-loop reference-model transfer function in the continuous time was

$$M(s) = \frac{G(s)}{1 + G(s)H(s)}$$

where

$$H(s) = 1$$

and

$$G(s) = \frac{e^{-0.8s}}{(3.87s+1)(0.13s+1)} \qquad (4.2-11)$$

Its discrete-time transfer function with sampling period T_s= 0.2s

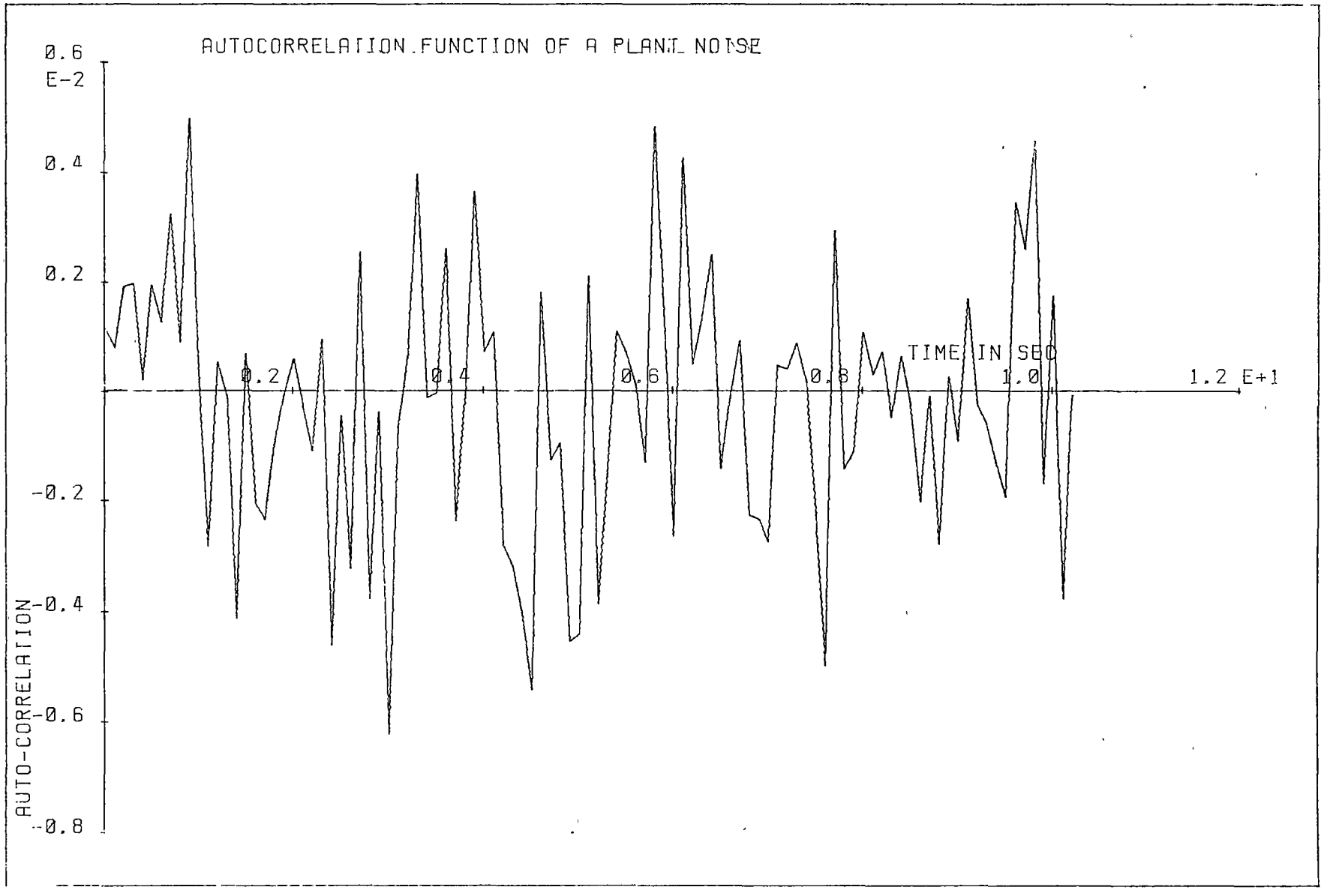
was

$$M(q^{-1}) = \frac{0.0249q^{-5} + 0.0147q^{-6}}{1 - 1.1618q^{-1} + 0.2015q^{-2} + 0.0249q^{-5} + 0.0147q^{-6}} \qquad (4.2-12)$$

A square wave signal plus PRBS (standard deviation of 1.34 units) was applied as the input signal. The plant and model responses are shown in figure 4.2-3. Figure 4.2-4 shows that there is no excessive control action present in the input to the plant at the instants when change in plant parameters take place . As in figures 4.2-1 and 4.2-2, these graphs show the response of the plant at three different periods. In the first period , the transfer function of the plant was as in equation (4.2-9) and the controller vector had converged. In the second period, the transfer function of the plant had changed to equation (4.2-10) but the controller weight vector function had not yet converged . In the third period the controller weight vector function had converged and the transfer function of the plant was as in equation (4.2-10).

The following page is figure 4.2-3.

" The auto correlation function of the plant noise ".


AUTOCORRELATION FUNCTION OF A PLANT NOISE
0.6
E-2
0.4
0.2
-0.2
-0.4
-0.6
-0.8
AUTO-CORRELATION
0.2
0.4
0.6
0.8
1.0
1.2 E+1
TIME IN SEC

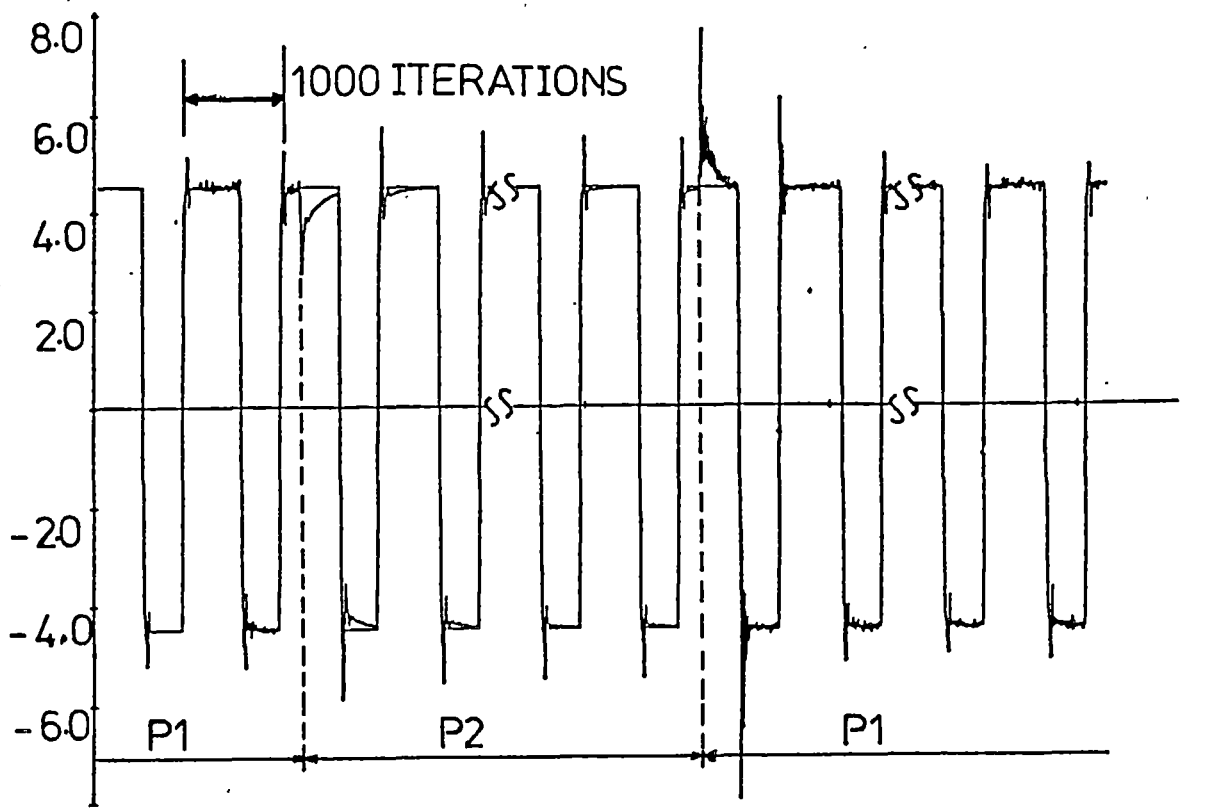
The following two pages are figures 4.2-4 and 4.2-5 .

"The ability of the closed loop NP-MRAC with closed loop reference model to adapt a plant with both parameter and time delay variation".

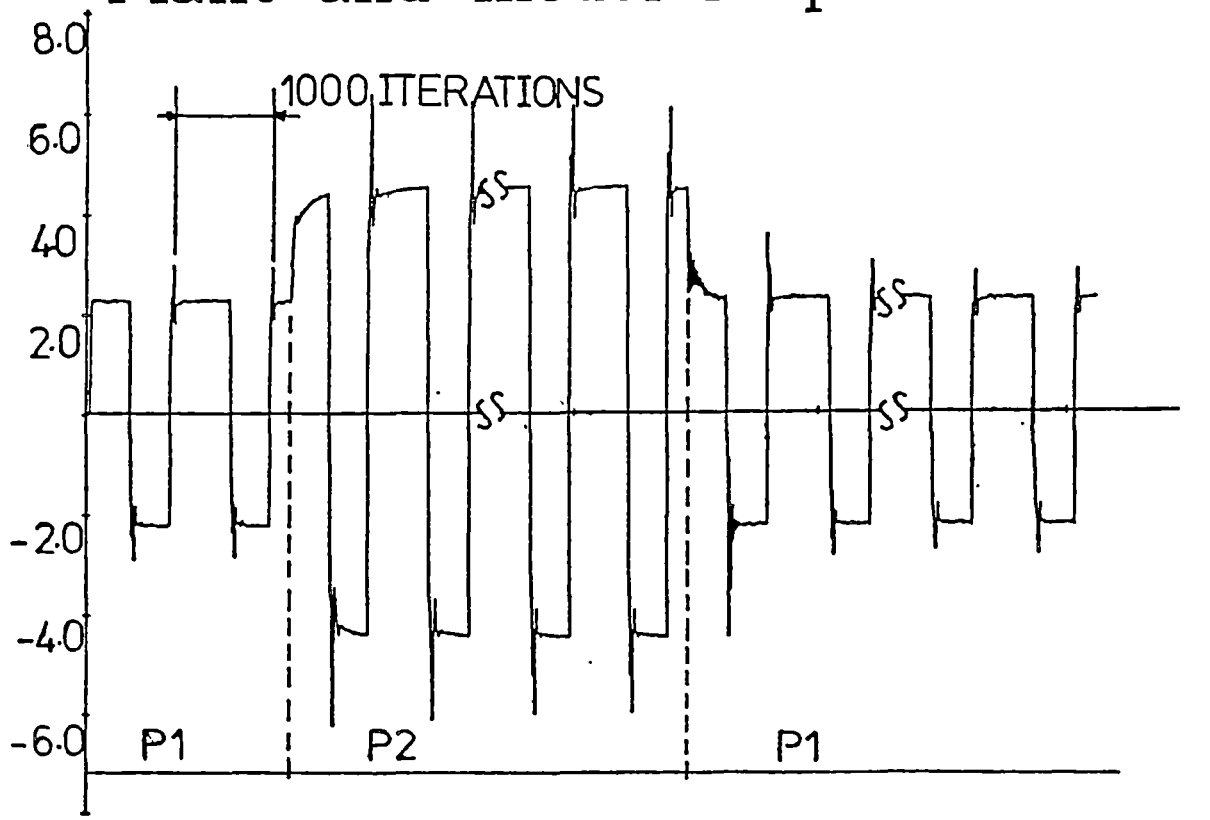
The first graph (figure 4.2-3) shows the plant and model response due to a square wave input signal.

The second graph (figure 4.2-4) shows the plant control input signal.

Simulation was done with $\mu = 0.000005$ and $\mu_p = 0.0003$.



Plant and model responses.



Plant control input signal.

4.3 SET POINT TRACKING PROPERTY

The closed loop NP-MRAC can also be designed to have good set point tracking property . By selecting a reference model whose gain is unity over all frequencies of interest present in the input signal, the LMS algorithm will adjust the FIR controller tap setting so that the tracking errors are minimized.

The plant transfer function was selected as

$$P(s) = \frac{\exp(-0.3s)(s-5)}{(s+1)(s+3)} \qquad (4.3-1)$$

It was sampled at a sampling interval of 0.1 seconds. Its discrete-time transfer function was

$$P(q^{-1}) = \frac{0.0601q^{-4} - 0.1012q^{-5}}{1 - 1.6457q^{-1} + 0.6703q^{-2}} \qquad (4.3-2)$$

Simulations were done for both open loop and closed loop methods of model selection. The closed loop poles were placed at s=-2 and s=-4 (i.e., q_1=0.81908 and q_2=0.67002) . In this way the closed loop reference-model transfer-function was as in equation (4.3-3) and the open loop reference model transfer function was as in equation (4.3-4).

$$M(q^{-1}) = \frac{0.02985q^{-5}}{1 - 1.4891q^{-1} + 0.5488q^{-2}} \qquad (4.3-3)$$

and

$$M(q^{-1}) = \frac{0.02985q^{-5}}{1 - 1.4891q^{-1} + 0.5488q^{-2} - 0.02985q^{-5}} \qquad (4.3-4)$$

The reference input signal was selected as

$$r(t) = \exp(-0.4t)\sin(2t) \qquad (4.3-5)$$

The set point tracking behavior is shown in figures (4.3-1) and

(4.3-2); corresponding to 2 types of model-selection represented as in equations (4.3.3) and (4.3.4) , respectively. For this demonstration we selected the same plant, the same test input signal and the same closed loop poles as used by Liu and Sinha (1987) on a self tuning controller with pole zero placement of the error transfer function , the output of their model is shown in figure 4.3.3. In this way, the behavior of the closed loop NP-MRAC may be compared with the behavior of Liu's and Sinha's method. As seen from figures 4.3-1,4.3-2 and 4.3-3, they all track the reference signal well. However, in Liu and Sinha 's case, the disadvantages are that the reference signal must be known in advance and be Laplace transformable. In addition, the time delay of the plant has to be precisely determined. But in the closed loop NP-MRAC, the prior knowledge of the time-delay of the plant is not a necessity, although some information about the plant characteristics would be helpful when choosing the reference model time delay and the number of taps for the FIR adaptive filter.

The following two pages are figures 4.3-1 and 4.3-2.

"The output responses of the compensated plant and the reference model due to the reference input signal of equation (4.3-5)",

The third page is figure 4.3-3 .

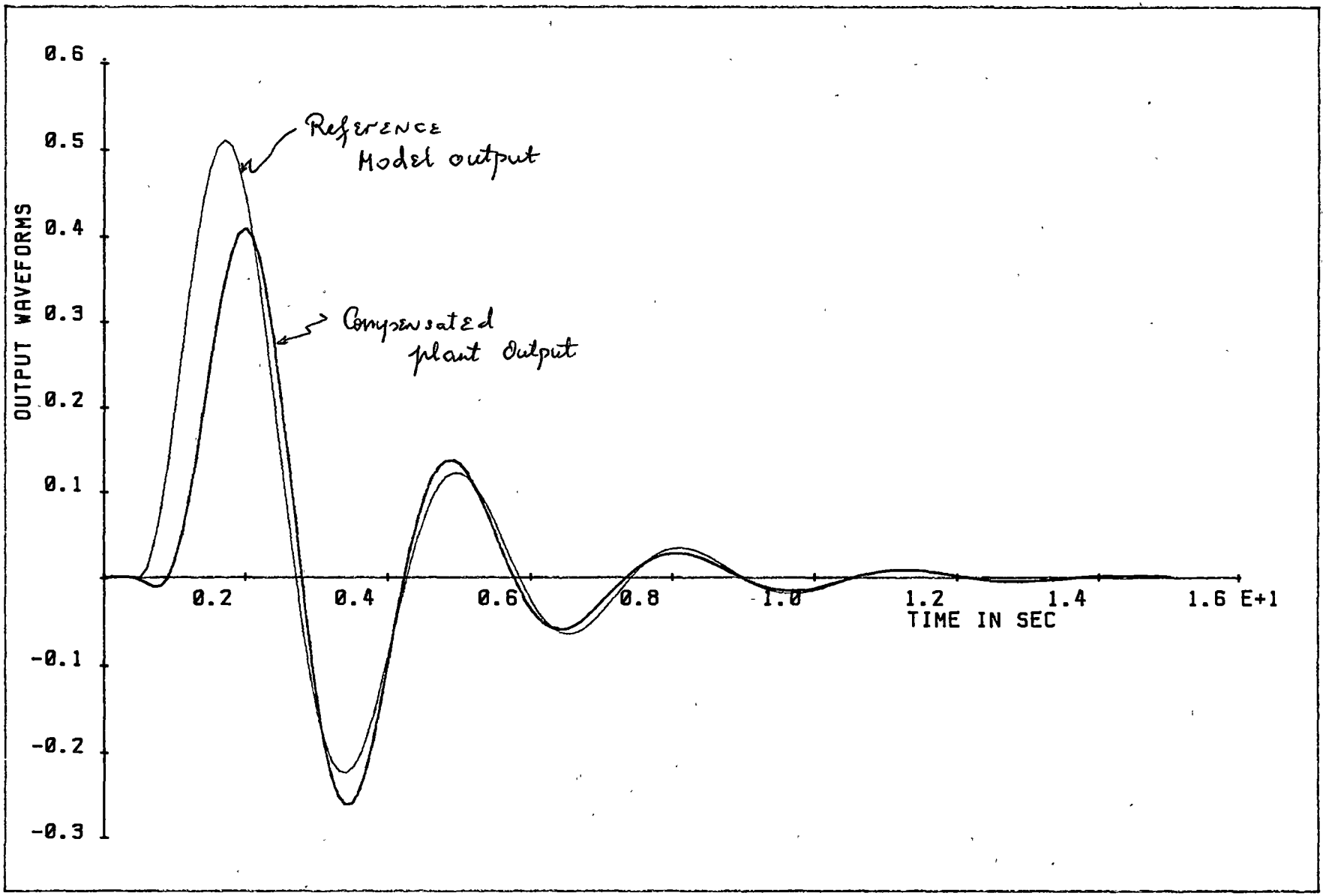
"The output response of the plant and the reference input signal".

The first graph (figure 4.3-1) was plotted for the closed loop NP-MRAC with the closed loop reference model of equation 4.3-3.

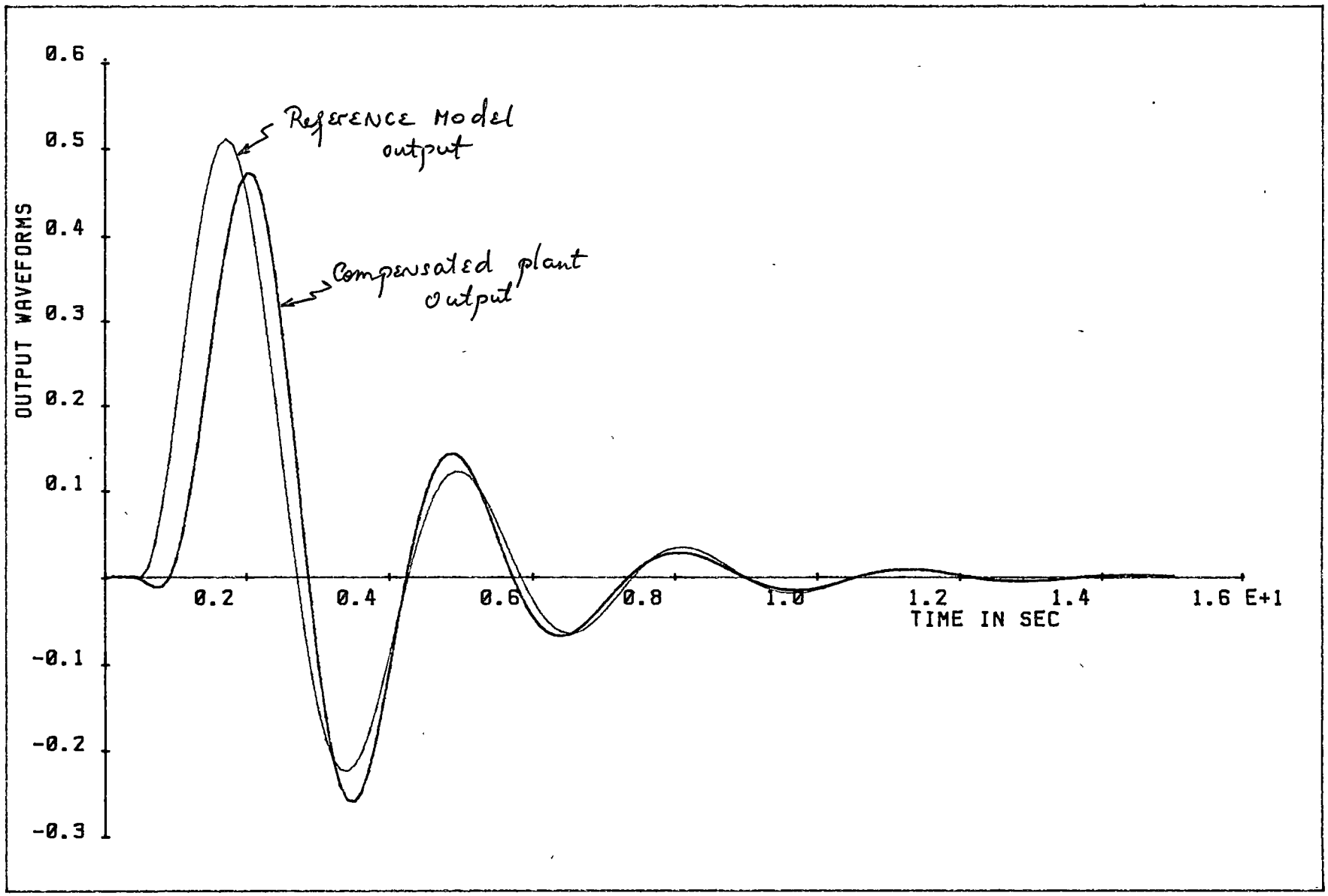
The second graph (figure 4.3-2) was plotted for the closed loop NP-MRAC with the open loop reference model of equation 4.3-4.

Both simulations were done with $\mu = \mu_P = 0.0001$.

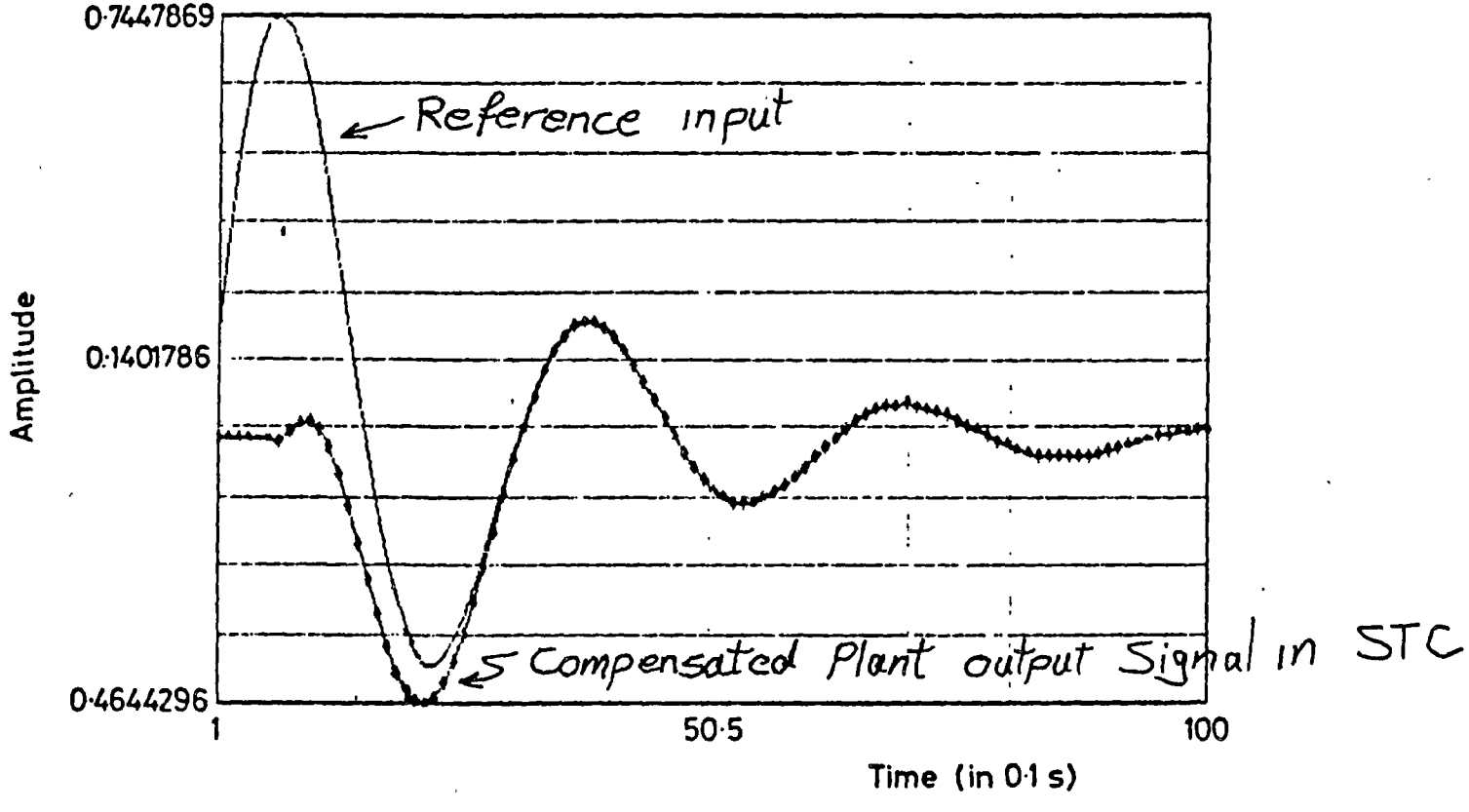
The third graph (figure 4.3-3) is a copy of the graph of the deterministic quadratic control with pole/zero placement in error transfer function drawn by Liu and Sinha (1987).



Model 2
14/2/89



Model 1
14/2/89


0·7447869
0·1401786
0·4644296
Amplitude
Reference input
Compensated Plant output Signal in STC
1
50·5
100
Time (in 0·1 s)

(*a*)

4.4 THE NP-MRAC TO CONTROL A PLANT WHICH INCLUDES INTEGRATORS

In chapter 3 we saw the method of the representation of the transfer function of a dynamic plant. There are many ways to interpret the polynomials of the transfer function, however we restrict our discussions to a plant of type-0 and a plant of type-n. The type-0 plant is a plant which does not have poles at the origin of the s-plane ($s \neq 0$) , while the type-n plant has n poles at the origin i.e., the plant has n integrators. So the continuous-time transfer function of any type-n plant can be written as

$$P_n(s) = \frac{1}{s^n}\left(P_0(s)\right) \qquad (4.4-1)$$

where $P_0(s)$ is the type-0 transfer function of the plant; both $P_0(s)$ and $P_n(s)$ have the form of equation (3.2.1-2). As described in section (3.2), all poles of P(s) are transferred into z-domain by equation $z=e^{sT}$,so that a pole s=0 will be transferred into z=1. Since the coefficients of the plant discrete-time transfer function polynomials , represented either in the z-notation or in the q-notation ,is the same. Hence, the type-n plant transfer function can be written in discrete-time as

$$P_n(q^{-1}) = \frac{1}{(1-q^{-1})^n}\left(P_0(q^{-1})\right) \qquad (4.4-2)$$

where $P_0(q^{-1})$ is the discrete-time transfer function of a type-0 plant. Both $P_n(q^{-1})$ and $P_0(q^{-1})$ have the form of equation (3.2.1-5).

Chapter 2 has shown that with wide band input signal, the shape of the FIR adaptive filter weight vector will match the shape of

the plant impulse response, if the tap length of the FIR adaptive filter is long enough to cover the impulse response of the plant. However , the length of the impulse response of the plant with integrators (type-n) goes to infinity. Therefore to obtain the same shape as the type-n plant impulse response, the FIR adaptive filter must have infinite number of taps. For type-n plant , the NP-MRAC technique must then be modified to handle this problem. Figure 4.4-1 and figure 4.4-2 illustrate a necessary modification. For this case, the output of the FIR controller passes through the block having a transfer function of $(1-q^{-1})^n$. The plant control input signal now becomes $(1-q^{-1})^n u_k$.

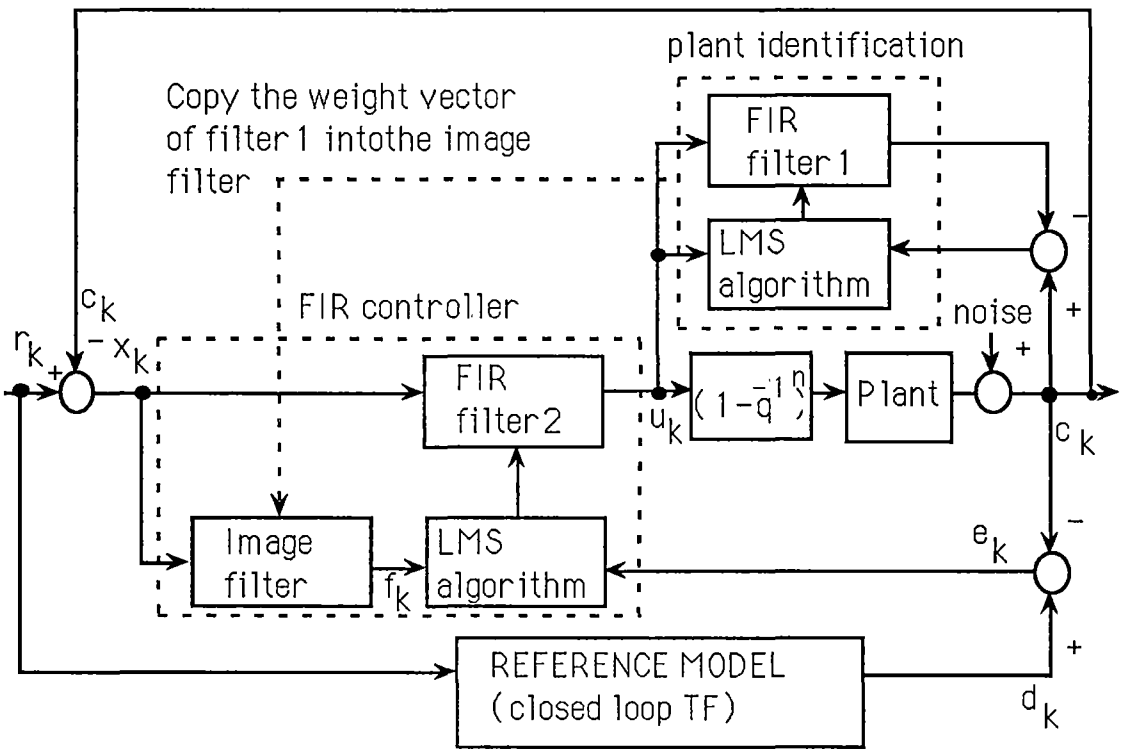


Figure 4.4-1 Block diagram of the closed loop NP-MRAC with the closed loop reference model used to control a type-n plant.

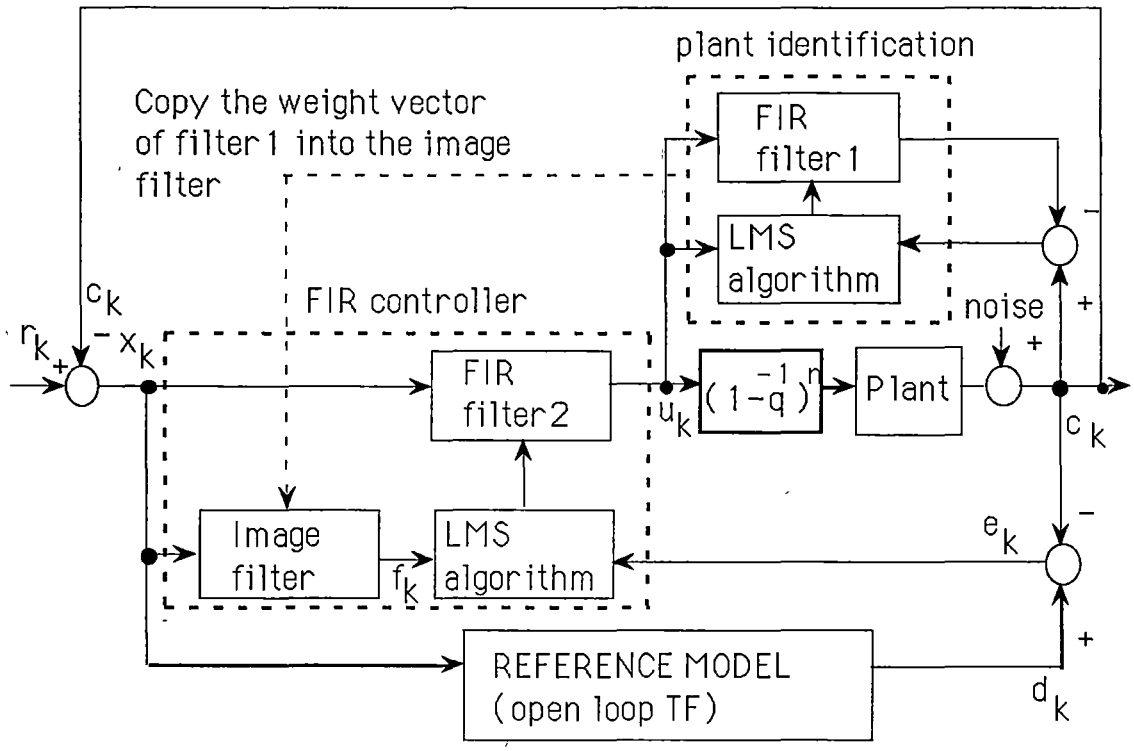


Figure 4.4-2 Block diagram of the closed loop NP-MRAC with the open loop reference model used to control a type-n plant.

The above closed loop NP-MRAC of figure 4.4-2 was simulated on a type-1 plant with transfer function

$$P(s) = \frac{8.5}{s(s+2)(s^2+4s+4.25)} \qquad (4.4-3)$$

The plant was sampled at a sampling rate (T_s) of 0.2 seconds and its discrete-time transfer function was

$$P(q^{-1}) = \frac{0.00078q^{-1} + 0.00298q^{-2} + 0.00387q^{-3} - 0.00002q^{-4}}{1 - 3.00426q^{-1} + 3.34776q^{-2} - 1.64469q^{-3} + 0.30119q^{-4}} \qquad (4.4-4)$$

The open loop transfer function reference model was selected such that its closed loop pole was at q=0.8 . Its discrete time

transfer function was

$$M(q^{-1}) = \frac{0.06q^{-5}}{1 - 0.8q^{-1} - 0.06q^{-5}} \qquad (4.4-5)$$

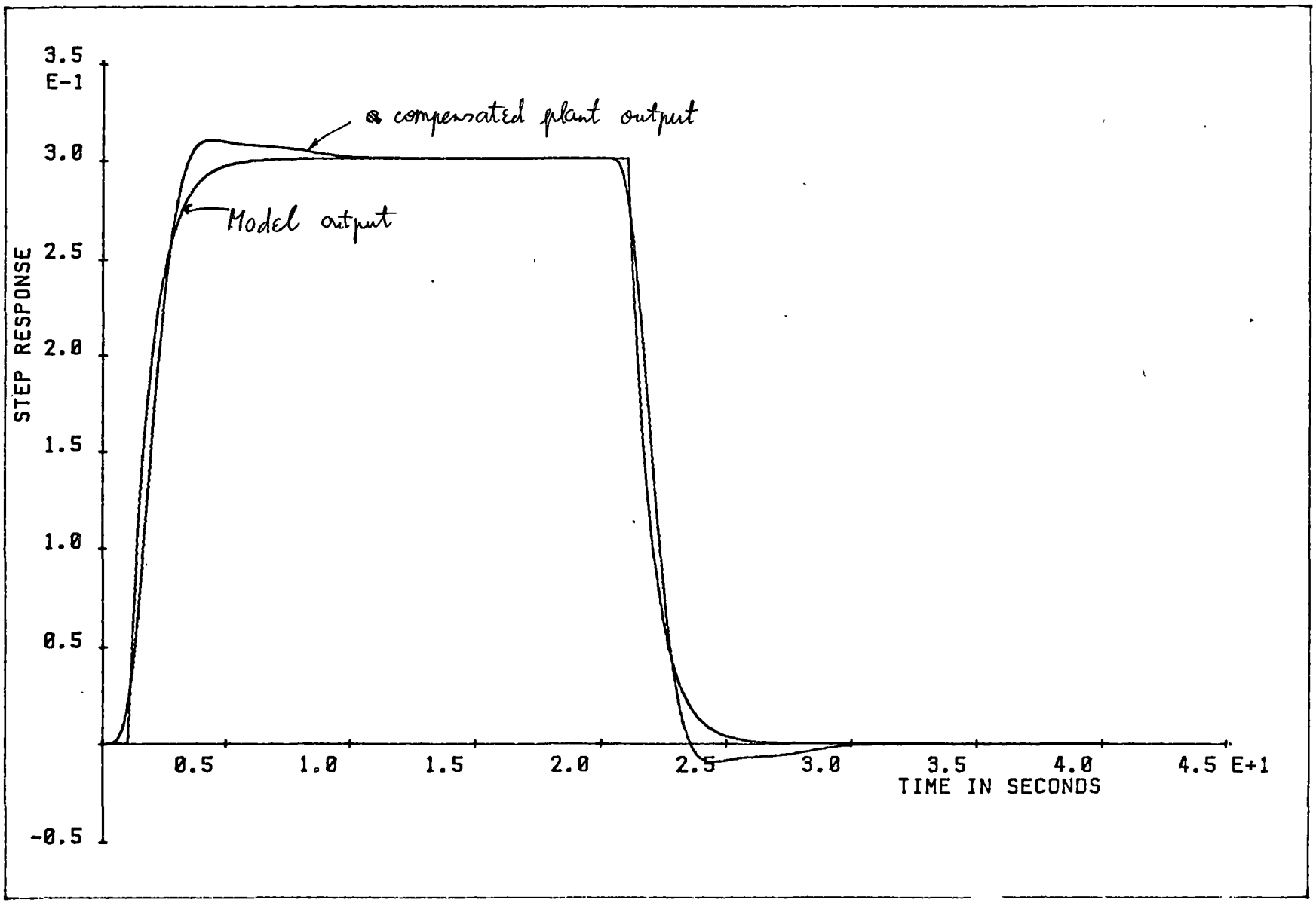
The compensated plant and the reference model output signals are displayed in figure 4.4-3.

This chapter has shown, by simulations, the adaptation ability of the closed loop NP-MRAC in many common industrial environments: plants with parameter and time delay variations. The next chapter will also demonstrate, by real-time hybrid simulations, the effectiveness of the closed loop NP-MRAC.

The following page is figure 4.4-3.

"The output responses of the model and the type-1 plant due to a square wave input".

Simulation was done with μ = 0.00004; it was plotted after 6 periods of the PRBS.


3.5
E-1
3.0
2.5
2.0
1.5
1.0
0.5
-0.5
STEP RESPONSE
compensated plant output
Model output
0.5
1.0
1.5
2.0
2.5
3.0
3.5
4.0
4.5 E+1
TIME IN SECONDS

Chapter 5

IMPLEMENTATION OF A CLOSED-LOOP NP-MRAC ON THE TMS320C20

In the past decade there has been an accelerated growth in the application of microprocessors in a variety of control systems. However they suffer from speed and data storage problems.

To overcome the data storage problem, in 1983, A.L. Dexter described Self-tuning control algorithm for single-chip microcomputer implementation. The control program was written in Intel 8022 assembly language and is 1100 bytes in length. The time taken by program execution limited the sampling interval to a minimum of approximately 100 msec. However, the drawback of this simplified self-tunner is its inability to deal with unknown or varying time delay.

To overcome the speed problem , in 1982, G. Fromme represented two procedures for optimizing control parameters for high sampling frequency in the order of 1kHz (or 1 millisecond sampling period). However, the controller parameters are adjusted off-line or in larger intervals.

Sheirah, Malik and Hope introduced an on-line computational self-tuning regulator , with the ability to deal with varying plant parameters. Unfortunately, limited processing power and limited availability of data storage limits its application. As on-line controller synthesis requires the solution of a large set of simultaneous equations to allow the varying time delay, hence a large number of parameters has to be estimated. Thus the computation time is large. Therefore the processing capacity is

limited to slow processes. The example given in their paper, a self tuning regulator of a third order system, was implemented on an Intel 8085 microprocessor using a 32-bit floating-point arithmetic. This system required around 4 kbytes of ROM and 0.5 kbytes of RAM. The computation time was about 230 msec per iteration. Even after modifying the algorithm to reduce the sampling period to about 35 msec, the controller converged rather slowly since the new algorithm performed plant identification routine in every 15 samples. The NP-MRAC overcomes both the above problems.

This chapter will discuss in detail the implementation of the NP-MRAC on a TMS320C20 signal processor. The TMS320C20 signal processing chip is introduced in section 5.1. A brief discussion of the NP-MRAC hardware is given in section 5.2. The controller program is presented in section 5.3. The NP-MRAC was implemented on the TMS320C20 and it was used to control a plant which was setup on an analog computer. The simulation results are displayed in section 5.4. Software details are included in section 5.5. This section is written to help a reader to follow the program, listed in the appendix D.

5.1 INTRODUCTION TO TMS320C20 SIGNAL PROCESSING CHIP

Texas instruments introduced the digital signal processor, TMS32010, in 1983. Since then many versions of the TMS320 were introduced. In this research work, the 2nd generation of the device, TMS320C20 microprocessor was used.

The Texas Instrument TMS320C20, due to its architecture ,has several beneficial features for implementing digital control

system elements speed and instruction set. Some of these are outlined in this section.

5.1.1 Flexibility

Two large on-chip RAM blocks [544 words] are configurable either as separate program and data blocks or as two contiguous data blocks. The program and data memory reside in two separate address blocks. This permits a full overlap of instruction fetch and execution .

The TMS320 architecture also allows transfers between program and data blocks, thereby increasing the flexibility of the device. This performance permits coefficients stored in program memory to be read into the RAM, eliminating the need for a separate coefficient ROM.

Most of the processor 's program instructions execute in a single machine cycle from either fast external program memory or on-chip program RAM. The flexibility of the TMS32020 also allows it to communicate to slower off-chip memories or peripherals by using the Ready signal ; instruction then becomes multicycle.

5.1.2 Internal hardware

Multipliers and shifters in the TMS320 family are hardware implemented functions .

Hardware shifters are located in the Arithmetic Logic Unit (ALU), connected to the output of the multiplier and the accumulator.

These features enable the TMS family to operate at a considerably greater speed compared to many other processors. For exam-

ple, the hardware multiplier performs signed or unsigned 16x16 bit multiplication in a single 200 nanosecond cycle, whereas for example the Intel 8086 performs the same operation in 6500 nanoseconds, i.e. 32.5 times slower.

5.1.3 Auxiliary Registers with their own arithmetic unit

The TMS32020 provides a register file containing five auxiliary registers (ARO-AR4). These auxiliary registers may be used for indirect addressing of data memory or for temporary data storage.

The auxiliary register file (ARO-AR4) is connected to the Auxiliary Register Arithmetic Unit (ARAU), shown in fig. (5.1.3-1). The ARAU may auto-index the current auxiliary register while the data memory location is being addressed. Indexing may be performed either by +1/-1 or by the contents of ARO . As a result, accessing tables of information does not require the Central Arithmetic Logic Unit (CALU) for address manipulation , thus freeing the CALU for other operations.

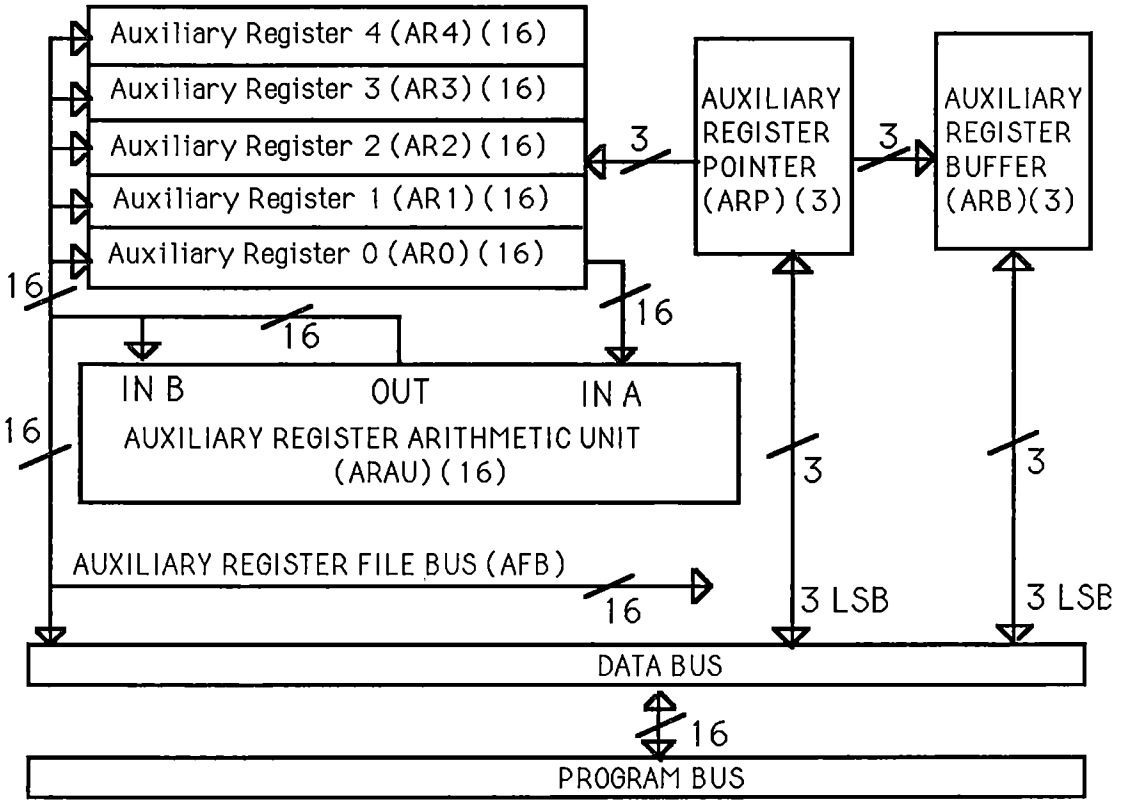


Figure 5.1.3-1: Auxiliary Register File.

5.1.4 Instruction set support for Floating-point operations

The TMS32020 instruction set contains special instructions for fast floating-point operations. A normalization (NORM) instruction is available to normalize fixed-point numbers. The Load Accumulator (LACT) with shift specified by the T register instruction is available to denormalizes a floating-point number.

These special instruction sets (and many others) make floating-point operations extremely fast at a speed comparable to some dedicated floating-point processors. Using the IEEE standard for the 23-bit mantissa and 8-bit exponent floating point format,

the TMS32020 performs a floating point multiplication in 7.8 microseconds, a floating-point addition in 15.4 microseconds and a floating point division in 22.8 microseconds. For comparison , the Intel 8086 performs floating point multiplication in 2 milliseconds which is 600 times slower. Even with a help of a hardware function units 8087 numeric data processor (NDP), floating point multiplication performs in 27 microseconds which is 8 times slower.

5.1.5 System Control

The TMS32020 provides special control operations such as an on-chip timer and a repeat counter.

A memory-mapped 16-bit timer was used for synchronous sampling clock in the closed loop NP-MRAC to sample the plant .

The TMS32020 design includes a repeat counter that allows a single instruction to be performed up to 256 times. The repeat feature is used with instruction such as block moves (BLKP, BLKD).

5.2 A MICROPROCESSOR BLOCK DIAGRAM FOR THE NP-MRAC

The microprocessor-controlled system is schematically described in figure (5.2-1). The output from the plant, c(t) , is a continuous signal. The signal c(t) passes through the input low-pass filter (LPF) and the sample and hold circuit. It is then converted into digital form by the Analog to Digital (A/D) converter. The converted signal, c_k , is now interpreted as a sequence of number and is represented in a 12-bit fixed point format. It is then processed by the Central Processing Unit (CPU) and a new sequence of

number , u_k , is generated. The digital output signal , u_k , is used to control an external plant . It is referred to as a plant control input signal. Signal u_k is first stored in Digital to Analog (D/A) output buffers, then converted to an analog signal by a D/A converter and filtered by a low pass filter before it is passed to the external plant.

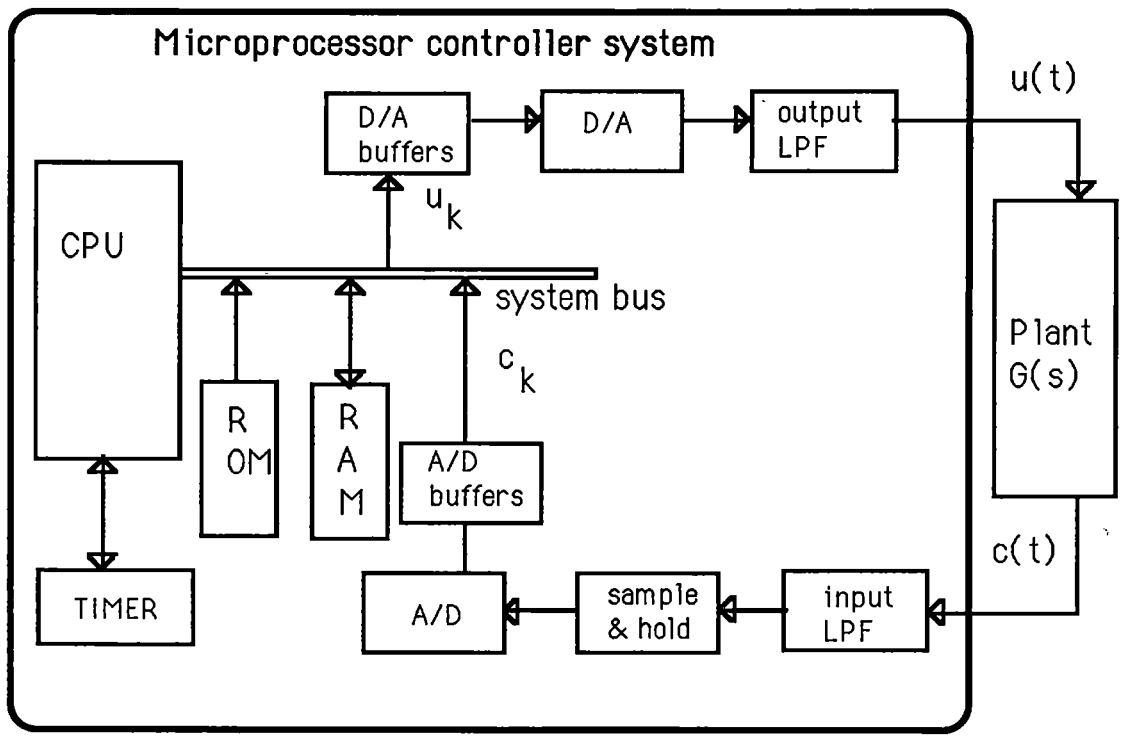


Figure 5.2-1: A microprocessor-based Control System.

The events are synchronized by the real-time clock at a sampling time Ts, which is controlled by the timer of the microprocessor. ROM contains the fixed part of the controller program and RAM contains the variable data.

In this work the components of the experiments were Analog Interface board (AIB), Software Development System (SWDS) and the IBM-AT computer. Figure 5.2-1 only represents the functional

block diagram of the closed loop NP-MRAC implemented on the microprocessor. The detail of hardware are covered in the TMS320 User's Guide Book.

5.3 CONTROLLER PROGRAM

The controller program , for the closed loop NP-MRAC was written in TMS320C20 assembly language and is 1.8K 16-bit words in length. The number of the taps of the controller FIR filter, as well as the number of the taps of the plant identification FIR filter, are programmable and the maximum number of taps is fixed at 40. The number of iterations and the sampling period are programmable . The minimum sampling period is approximately 10 milliseconds for the closed loop NP-MRAC with 40-tap FIR plant identification and 40-tap FIR controller .

The software is composed of 7 routines. Following is a brief description of them (detail explanation will be given later):

(1) "initialize routine" This routine disables/enables interrupts , loads system parameters into data memory and initializes the registers.

(2) "floating point conversion (FXFL) routine" This routine converts an output of the D/A into a 23-bit mantissa and an 8-bit exponent .

(3) "floating point addition (FADD) and multiplication (FMULT) routines" These routines are for floating point additions and multiplications. They were provided in the TMS320 software libraries. However they were modified to suit this application (For details of the modifications please refer to the appendix C).

(4) "plant identification (PIDTFY) routine" This routine calculates the shape of the finite impulse response of the plant .

(5) "LMS controller routine" This routine filters the input

signal of the controller and adjusts the weight vector of the controller . The filtering is done by calling a subroutine PREFTR. This routine will be explained in detail in section 5.5.5.

(6) "fixed-point conversion (FLFX) routine" This routine converts a controller output signal ,which is in a floating point format, into a fixed point format and sends it to the D/A converter.

(7) "reference model routine" This routine calculates the output signal, d_k, of the model .

Following is a brief description of the controller program (please read this part in conjunction with the block diagram of the closed loop NP-MRAC in figure 5.3-1, and the flow-chart in figure 5.3-2).

In the beginning of each k^{th} sampling period the digital plant output, c_k, is represented by a 12-bit fixed point format. To communicate with the NP-MRAC controller software, the floating point conversion routine (FXFL), is called which converts this signal into the standard floating-point format.

To determine the error signal, e_k , the plant output signal (c_k) is compared with the model reference output signal (d_k). Now the parameter, $2\mu e_k$, where μ is a constant , is calculated and stored in the RAM (memory location NO2UE) .

The Central Processing Unit (CPU) then calls the plant identification routine (PIDTFY) which predicts the shape of the-FIR of the plant by calculating new values for the weight vector, W_p. The weight vector , W_p , is used by the Image Filter in PREFTR subroutine (please see below).

The CPU now calls the subroutine (PREFTR). The PREFTR

subroutine is the Image Filter which filters the input signal $x_k = r_k - c_k$. The output of this filter at k^{th} iteration is

$$f_k = \sum w_{p\,i} x_{k-i} \qquad for\ 0 \le i \le L-1$$

where L is number of taps of the FIR controller.

$f_k$is then stored in the RAM (memory locations from >CA4 to >CA7). The LMS algorithm uses the output , f_k, of the Image Filter and parameter 2 μe_k to adjust the weight vector of the FIR controller . The CPU updates the weight vector of the FIR controller (FIR filter-2) by the LMS algorithm using the equation (5.3-1).

$$w_{i\,k+1} = w_{i\,k} + 2\,\mu e_k f_{k-i} \qquad for \quad 0 \le i \le L-1 \qquad (5.3-1)$$
where L is the number of taps.

Now the signal X_k passes through the FIR controller. The control signal is calculated using the equation (5.3-2).

$$u_k = \sum w_{i\,k} x_{k-i} \qquad for\ 0 \le i \le L-1 \qquad (5.3-2)$$

where $w_{i\,k}$ is the value that obtains from equation (5.3-1).

The u_k , at present, is represented in floating point format. It is then passed through the fixed point conversion routine (FLFX). This routine produces an output signal , u_k , in the 12-bit fixed point format which is then used to control the external plant.

u_k is sent to the plant through the 12-bit D-A converter.

The CPU then waits for the next sampling period to restart the process.

Figure (5.3-2) shows the block diagram of the control program described above.

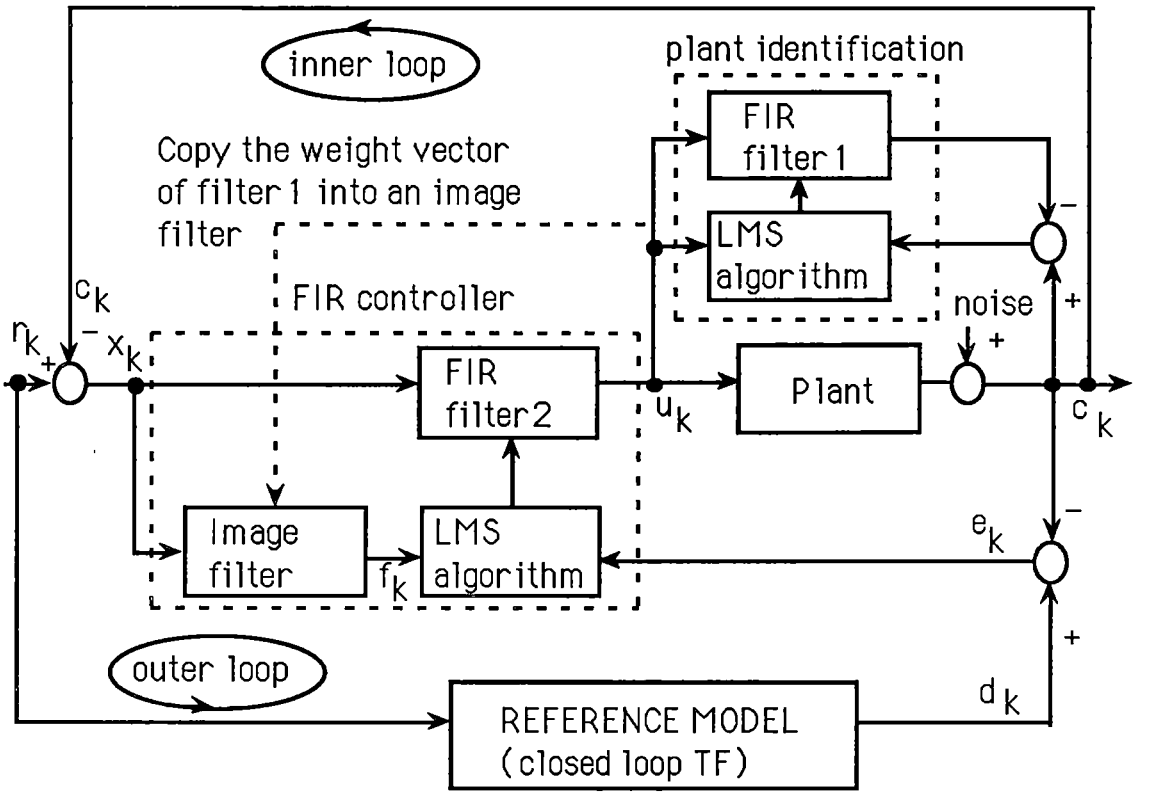


Figure 5.3-1: Block diagram of the closed loop NP-MRAC using a ref. model with closed loop transfer function

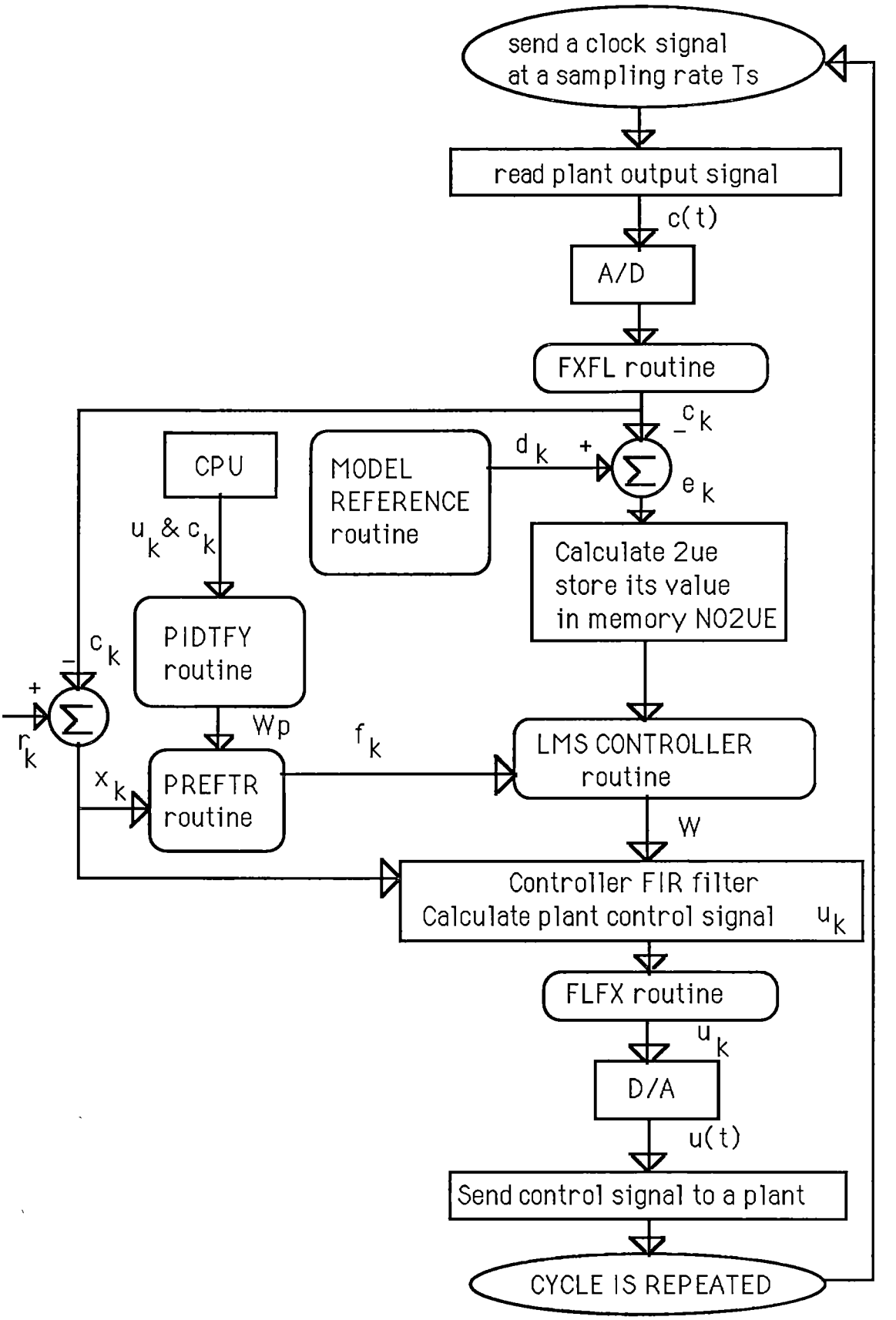


Figure 5.3-2: An overall performance block diagram.

5.4 TESTING THE SYSTEM WITH A PLANT IMPLEMENTED ON THE ANALOG COMPUTER

The laboratory experiment was conducted by setting up the equipment as shown in figure (5.4-1). They were an analog-computer, the TMS320 Analog Interface Board (AIB) and the TMS320 Software Development System (SWDS) plugged into an IBM-AT backplane computer. The digital control program was interrupt-driven, controlled by the on-chip timer register (TIM) inside the TMS320c20 signal processing chip, which was controlled by the IBM computer. The interrupt was programed to occur at the sampling rate .The program and data were stored in the SWDS static RAM.

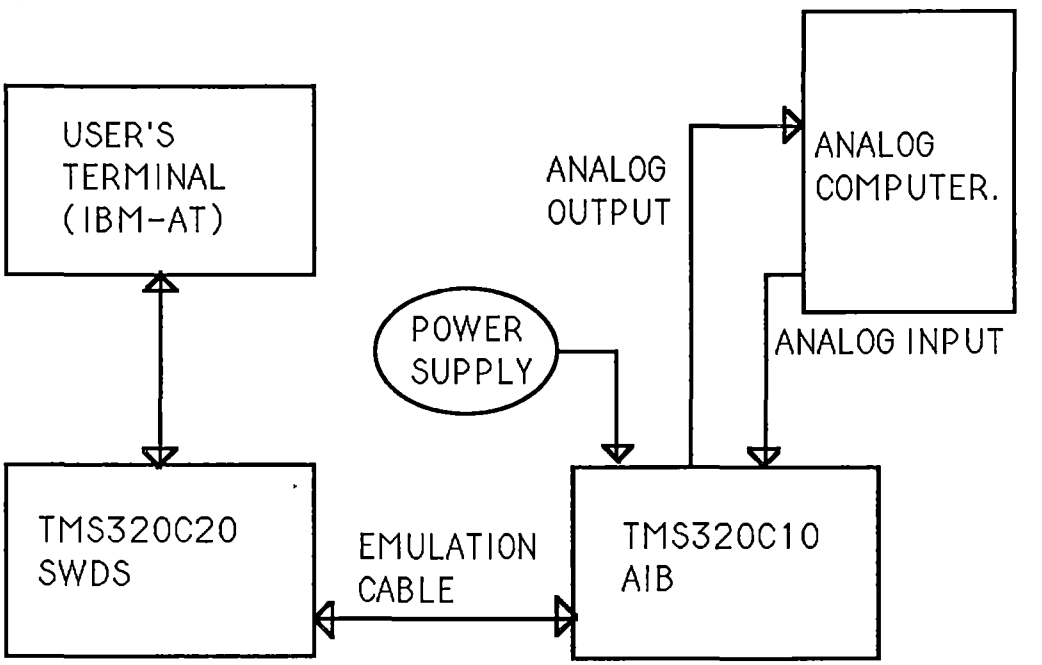


Figure 5.4-1: Laboratory set-up experiment block diagram.

In the experiment the sampling period was set at 20 msec. The plant transfer-function was switched from

$$P_1(s) = \frac{8.5}{(s+2)(s^2+4s+4.25)} \qquad (5.4-1)$$

to

$$P_2(s) = \frac{4.0}{s^2+0.8s+4.0} \qquad (5.4-2)$$

These plants were set up on the analog computer. The details of the set-up are shown in figures 5.4-2 and 5.4-3.

The reference-model transfer-function was selected such that its closed loop pole was kept at q=0.8. Its discrete transfer function is then as shown in equation (5.4-3).

$$M(q^{-1}) = \frac{0.06q^{-5}}{1-0.8q^{-1}} \qquad (5.4-3)$$

The output response of this reference-model to a square wave input is shown in figure (5.4-4).

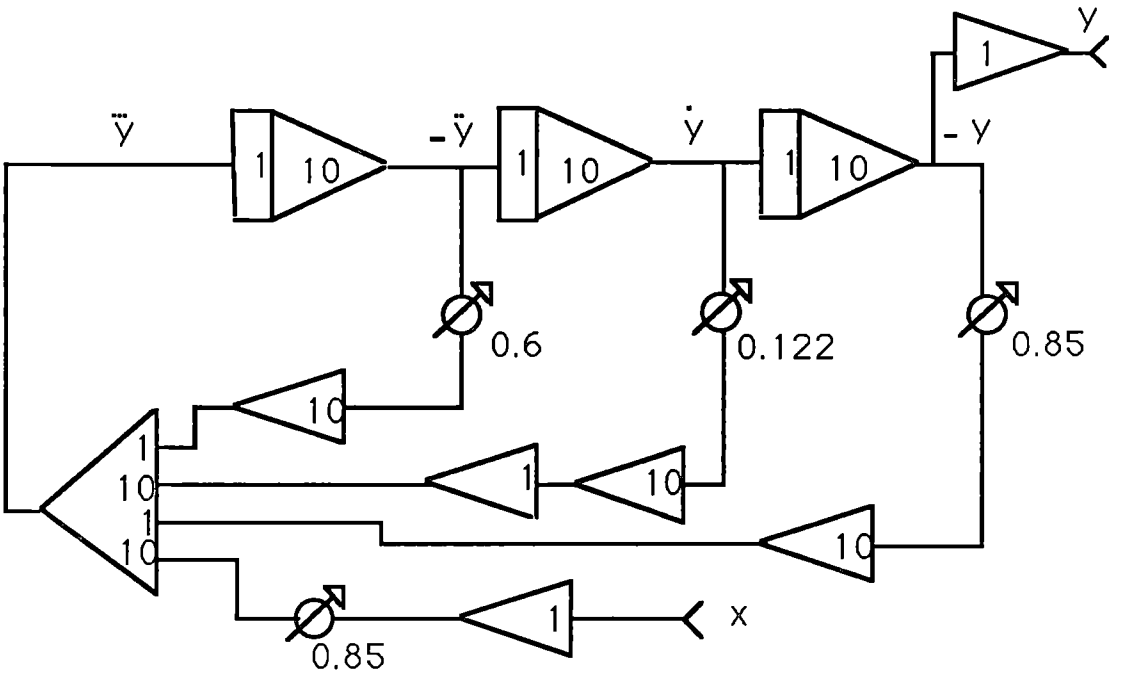


Figure 5.4-2: Block diagram for the first plant on the analog computer.

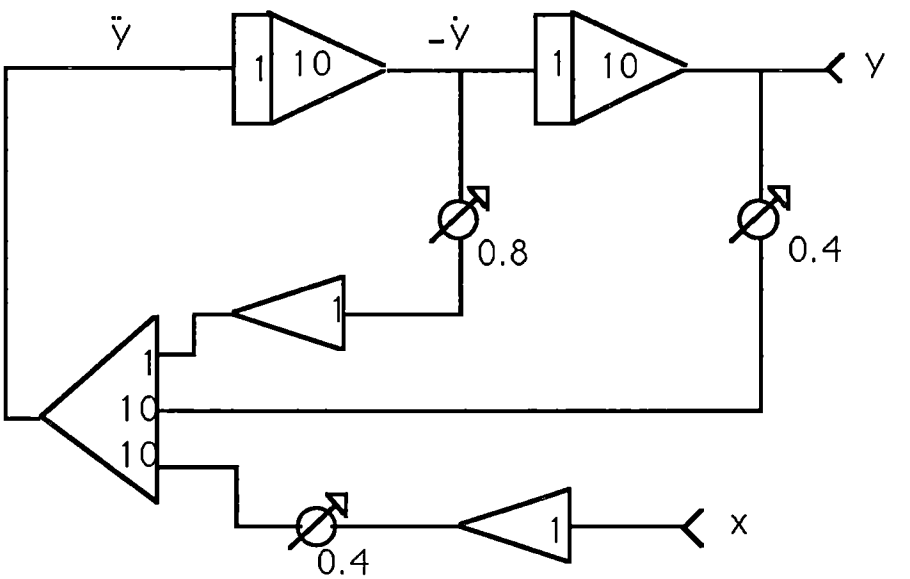
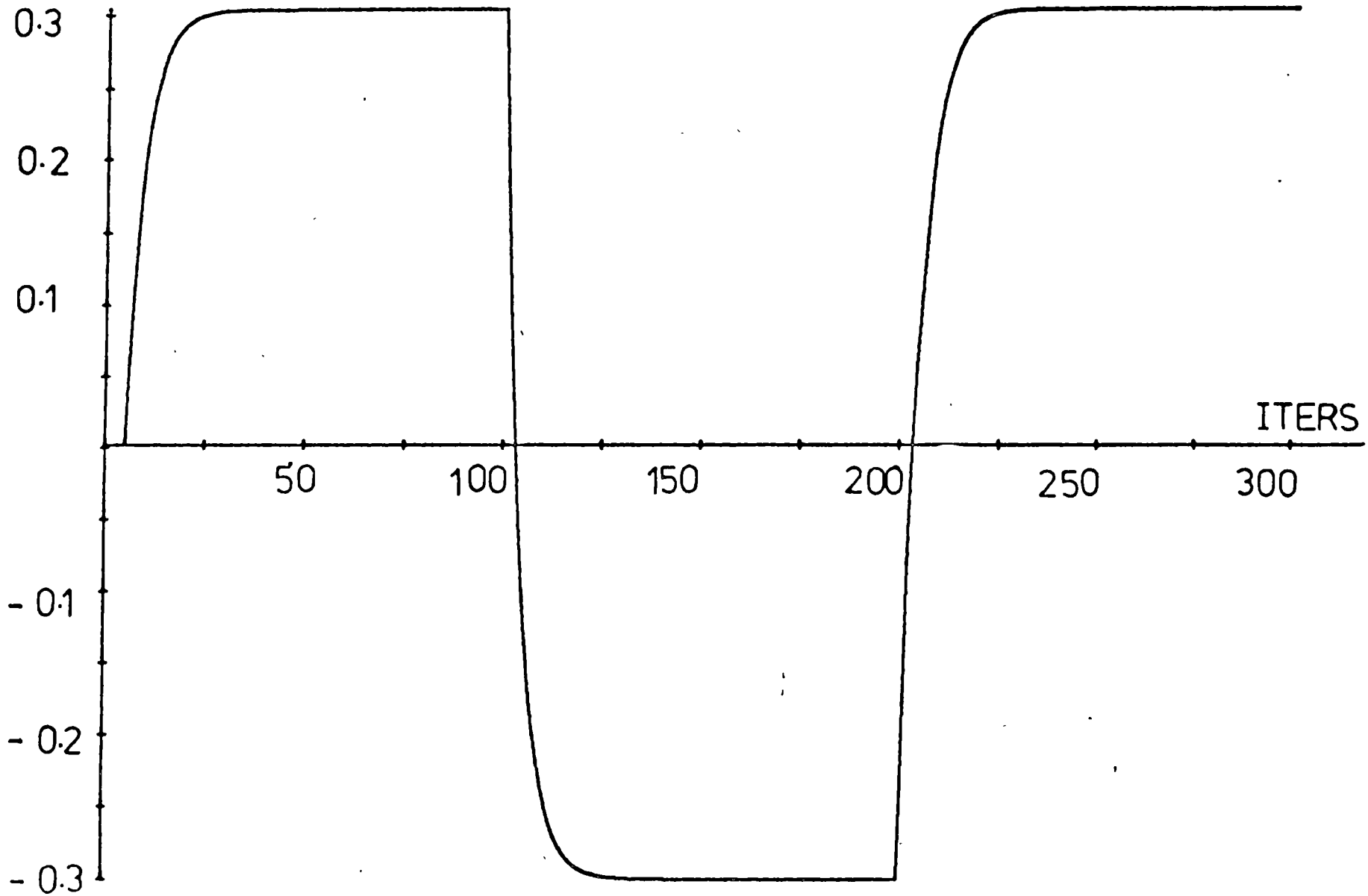


Figure 5.4-3: Block diagram for the second plant on the analog computer.

The following page is figure 5.4-4 ,

"The output response of the reference model to a square wave input".


0·3
0·2
0·1
- 0·1
- 0·2
- 0·3
ITERS
50
100
150
200
250
300

Simulation was done with $\mu = \mu_P = 0.004$ and using the PRBS as an input signal during the adaptive process. However, for ease of comparison the output response of the plant was recorded due to a square wave input signal. They are explained as below.

The output response of the plant with transfer function $P_1(s)$ (equation 5.4-1) to an square wave input when the taps of the FIR controller converged was recorded after 6 period of the PRBS, and it is shown in figure (5.4-5). The plant was then switched to the second form , $P_2(s)$. The square wave response of the new plant with the old tap-setting of the FIR controller was recorded as shown in figure (5.4-6) . It shows that due to plant parameter variation, the system oscillated drastically. This is due to the fact that the second order transfer function (equation 5.4-2) has a damping factor ξ of 0.2 and the percentage overshoot is 53% . The tap-setting of the FIR controller filter then readjusted itself to a new set of tap-controller in 6 period of the PRBS. Figure (5.4-7) shows the output response of the plant to an square wave input when the new value of the tap-setting of the controller had converged . This demonstrates the effectiveness of the closed-loop NP-MRAC in real time application.

Details of the experiment set-up are shown in figures 5.4-8, 5.4-9 and 5.4-10 .

The following page is figure 5.4-5, 5.4-6, 5.4-7.

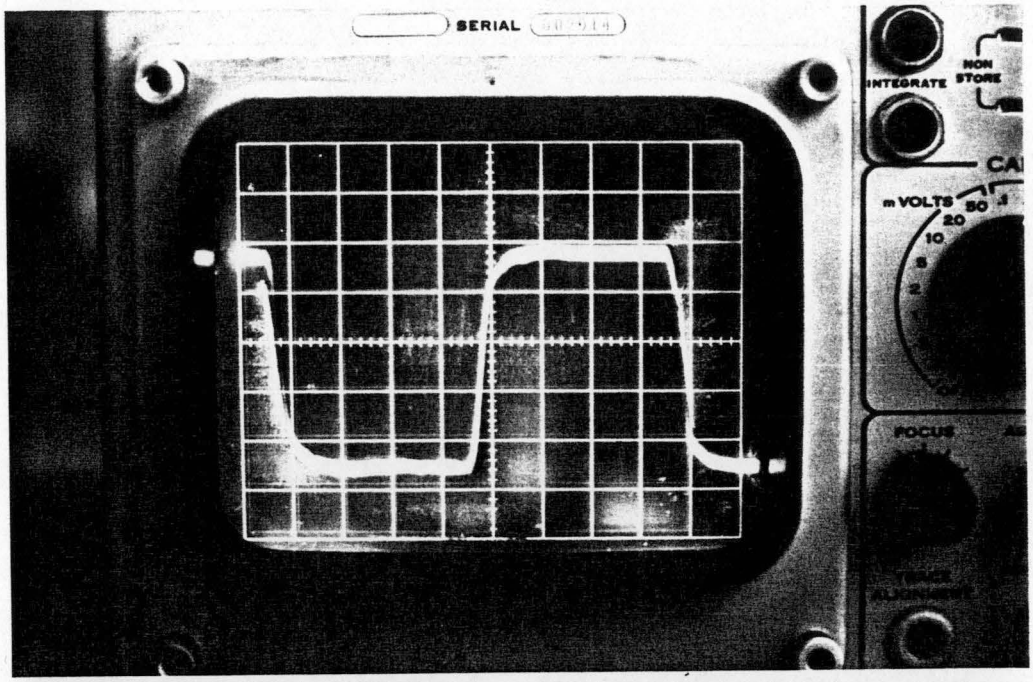
"The output response of the plant to an square wave input".

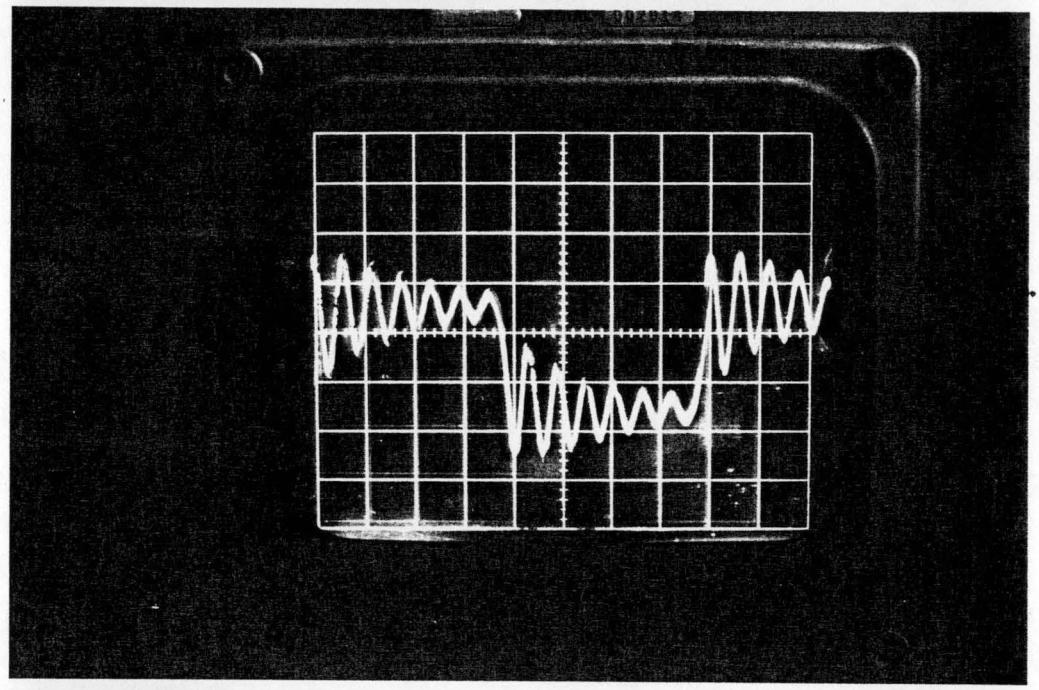
The first graph ,figure 5.4-5, has a vertical scale of 0.1V/cm and the horizontal scale of 0.5sec/cm. (each square is $1cm^2$)

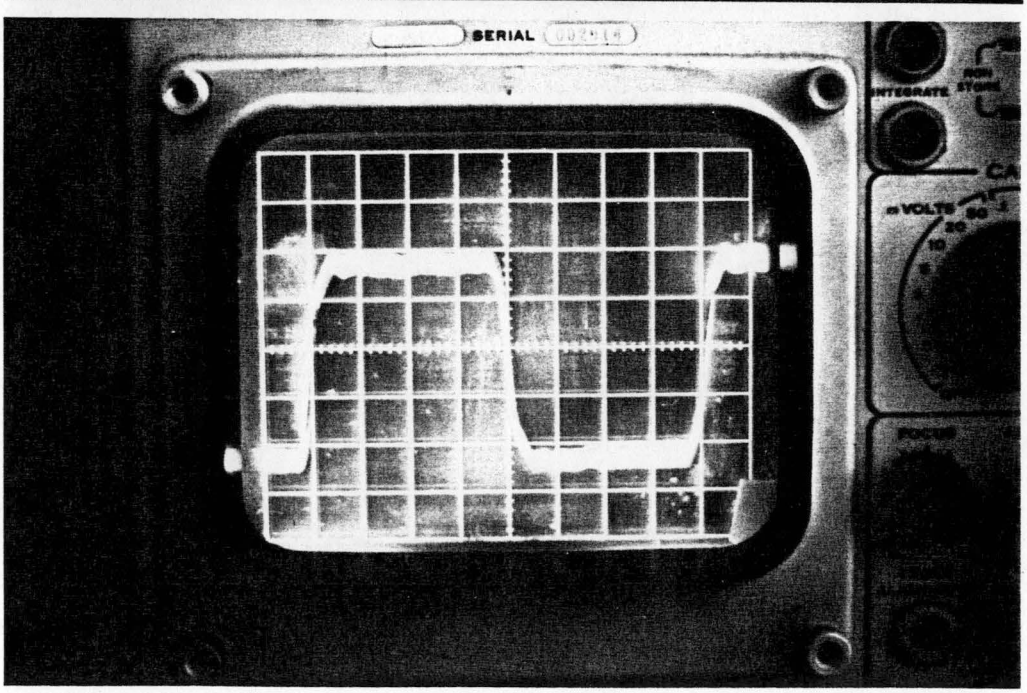
The second graph ,figure 5.4-6, has a vertical scale of 0.2V/cm and the horizontal scale of 0.5sec/cm. (each square is $1cm^2$)

The third graph ,figure 5.4-7, has a vertical scale of 0.1V/cm and the horizontal scale of 0.5sec/cm. (each square is $1cm^2$)

All simulations were done with $\mu = \mu_p = 0.004$ and using PRBS as the input signal during the adaptive process.


SERIAL
INTEGRATE
NON STORE
m VOLTS
FOCUS




SERIAL
INTEGRATE

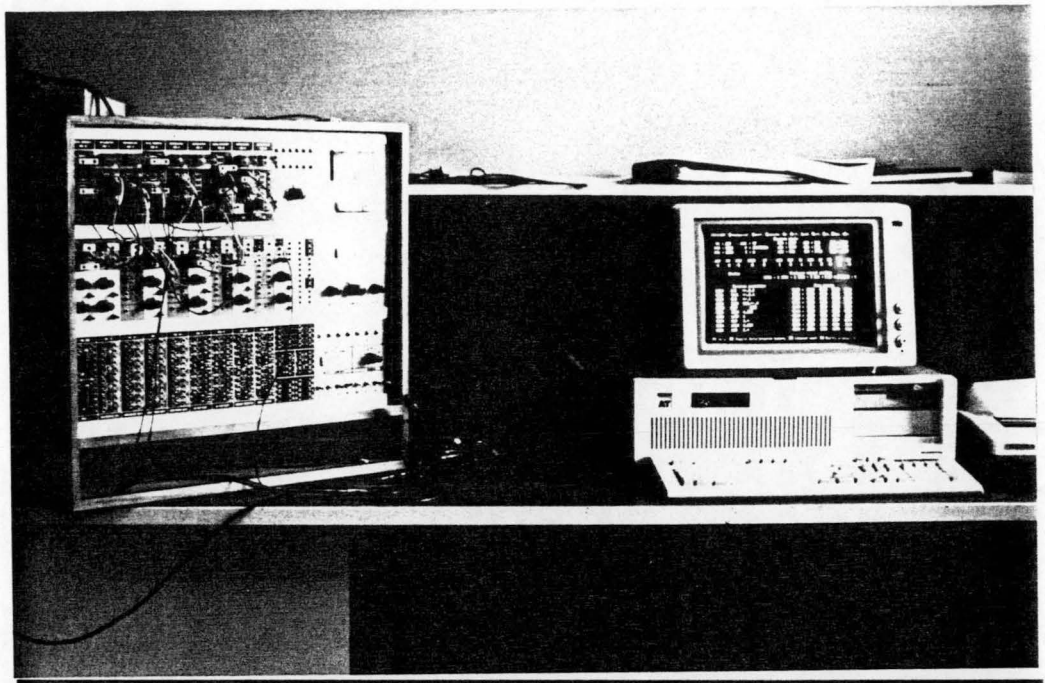
The following page is figure 5.4-8, 5.4-9, 5.4-10.

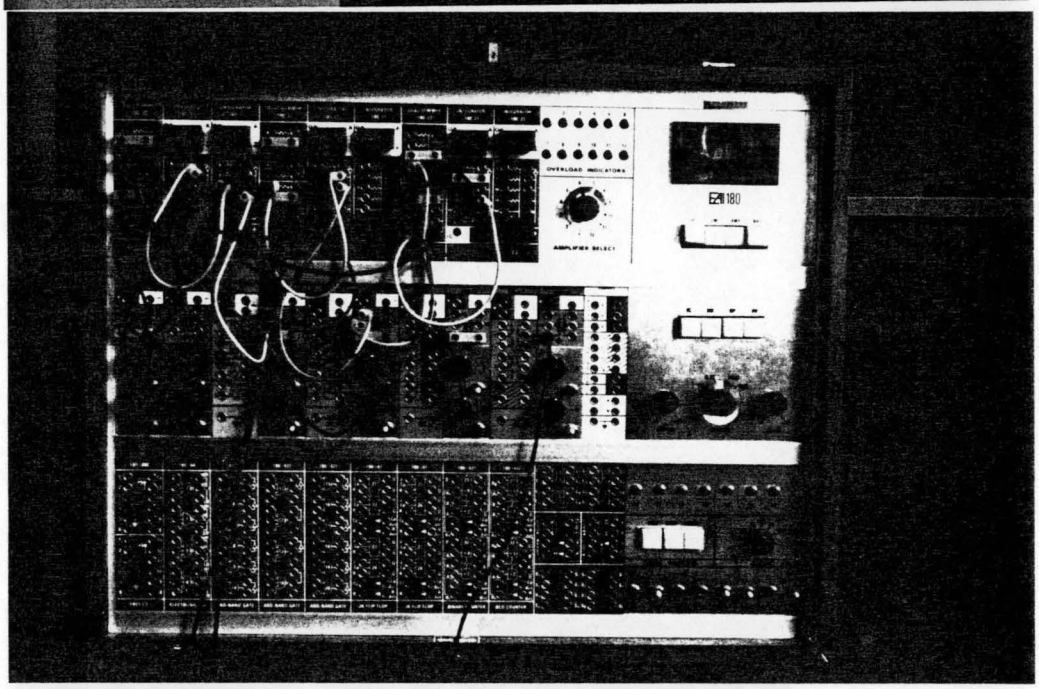
"The setup of the experiment".

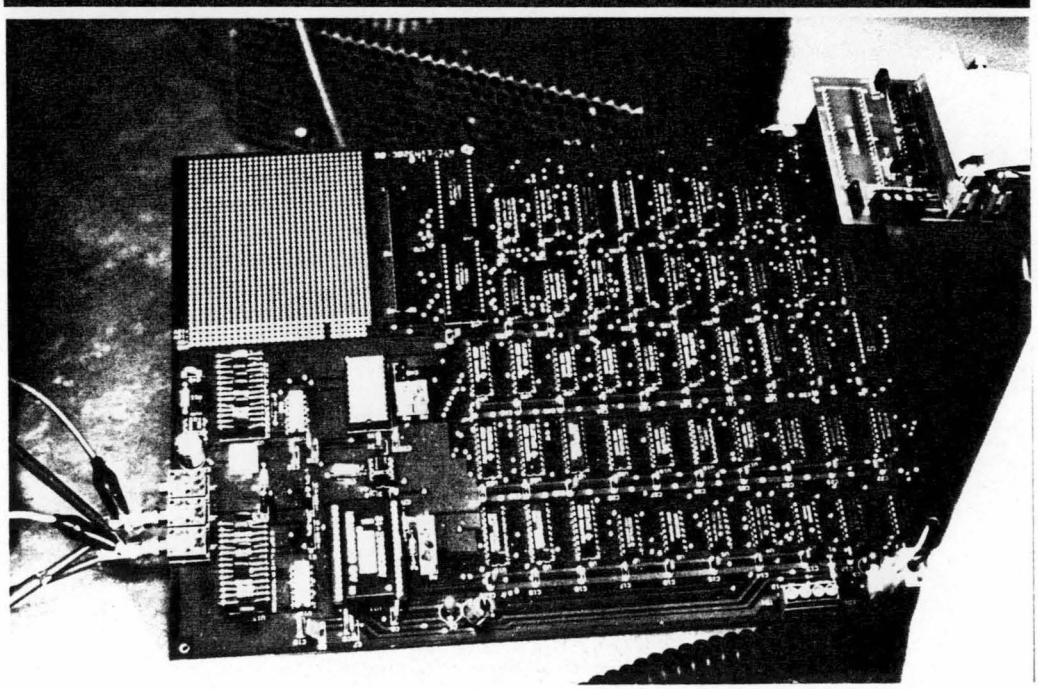
The first picture , figure 5.4-8 , shows an overall set-up experiment.

The second picture , figure 5.4-9, shows an analog computer.

The third picture , figure 5.4-10, shows an Analog Interface Board (AIB).




OVERLOAD INDICATORS
AMPLIFIER SELECT
180



5.5 SOFTWARE DESCRIPTION

This section contains the detail description of the programs written in TMS320C20, listed in the appendix D. Hence, it is written for a reader who wish to go into very fine information and it is advised to be read in conjunction with the program and the TMS320C20 User's Guide . The routines described are as follow: the initialization routine, the floating-point conversion routine, the floating-point addition and multiplication routines, the plant identification routine, the LMS controller routine, the fixed-point conversion routine, and the reference-model routine.

5.5.1 Initialization routine

The initialization routine is used

(i) to initialize an analog interface board (AIB).

(ii) to store system parameters into data memories . The parameters are speed adaptation of the FIR controller (μ) , speed adaptation of the FIR plant identification (μ_p), sampling period (T_s) and FIR tap delay length (L). These parameters are programmable and they can be selected by the user.

(iii) to transfer constants and coefficients which are stored in program memories into data memories.

(iv) to enable timer interrupt (TINT) to control a sampling period.

(v) to define initial conditions of the test input signal (PRBS) , plant control input signal (u_k), plant output signal (c_k) , FIR controller tap weight vector (W) and FIR plant identification tap weight vector (W_p). These parameters are changeable. Please refer to appendix D for the setting procedure.

5.5.2 Floating Point Conversion Routine (FXFL routine)

The plant output signal has a voltage range of (-10v..+10v). The analog input signal in this range can be digitally represented using the 12-bit A-D. The corresponding output of A/D is defined in the range of (-1..+1).

The FXFL routine is used to

(1) amplify the output of the A/D converter (to increase the accuracy of the calculation);

(2)convert the A/D output signal to the standard floating-point format which is defined by the TMS software package. This format will be explained later in this section.

The output of the A/D is represented in a binary form whose most significant-bit (MSB) is known as a 1-bit sign field (s) and other 11 bits are known as an 11-bit fraction field (f). This is represented as shown in figure (5.5.2-1).

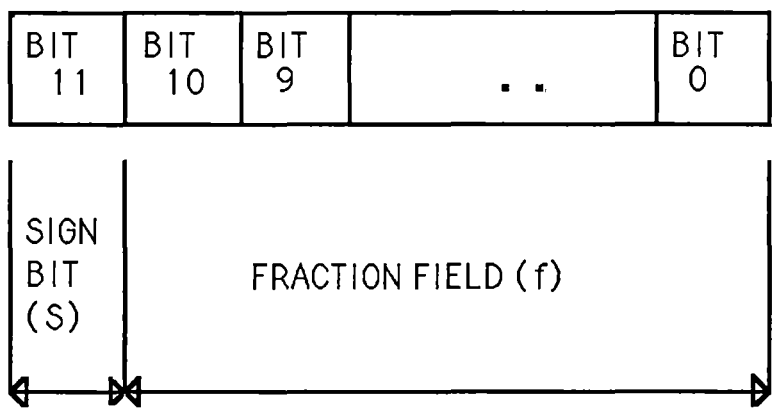


Figure 5.5.2-1: A 12-bit A/D binary number.

To scale up the output of the A/D to the range (-8 to +8), it is multiplied by 2^3. The value of 12-bit binary number X is then represented by the formula

$$X=(-1)^{s+1}*2^{3}*0.f$$

where s can be either one or zero

and f varies from >000 to >7FF[1].

The TMS floating-point routines require the input floating-point numbers to have a format of four 16-bit fields. This format is shown in figure (5.5.2-2) in which A and B are the input floating-point numbers and C is the resultant output of A and B. In this form, the value of a binary floating-point number X is represented as

$$(-1)^{s}*0.1f_{ms}f_{ls}*2^{e}$$

where s, f_{ms}, f_{ls} and e are as shown as in figure 5.5.2-2.

Numbers are required to be represented in a two's complement form.

The 12-bit digital signal , which is stored in the A/D buffers, is converted into this standard binary floating-point format by the floating-point conversion routine (FXFL routine). Its flow chart diagram is shown in figure (5.5.2-3).

1. > means hexadecimal number

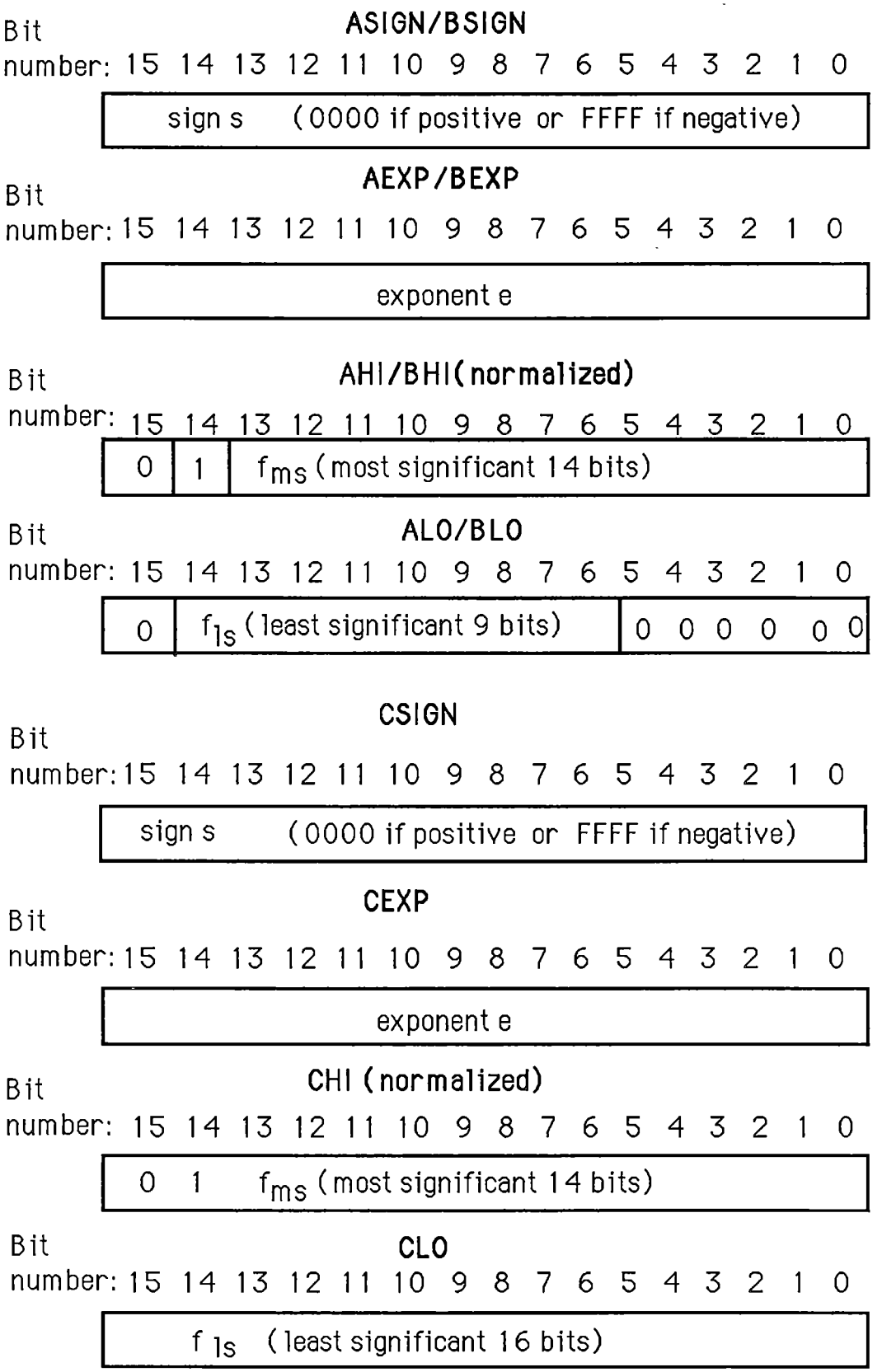


Figure 5.5.2-2 Floating-Point Format.

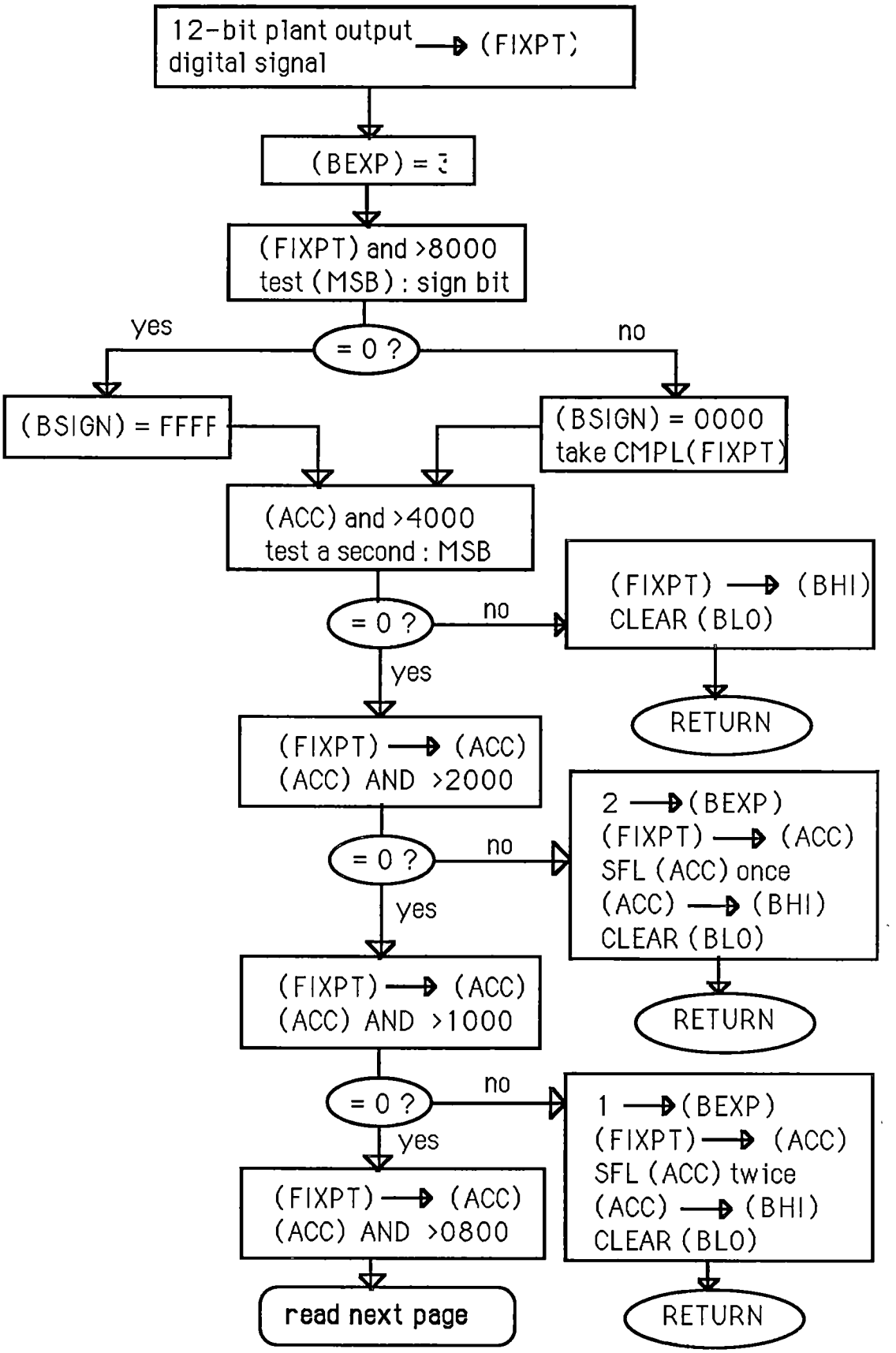


Figure 5.5.2-3: The floating-point conversion routine .

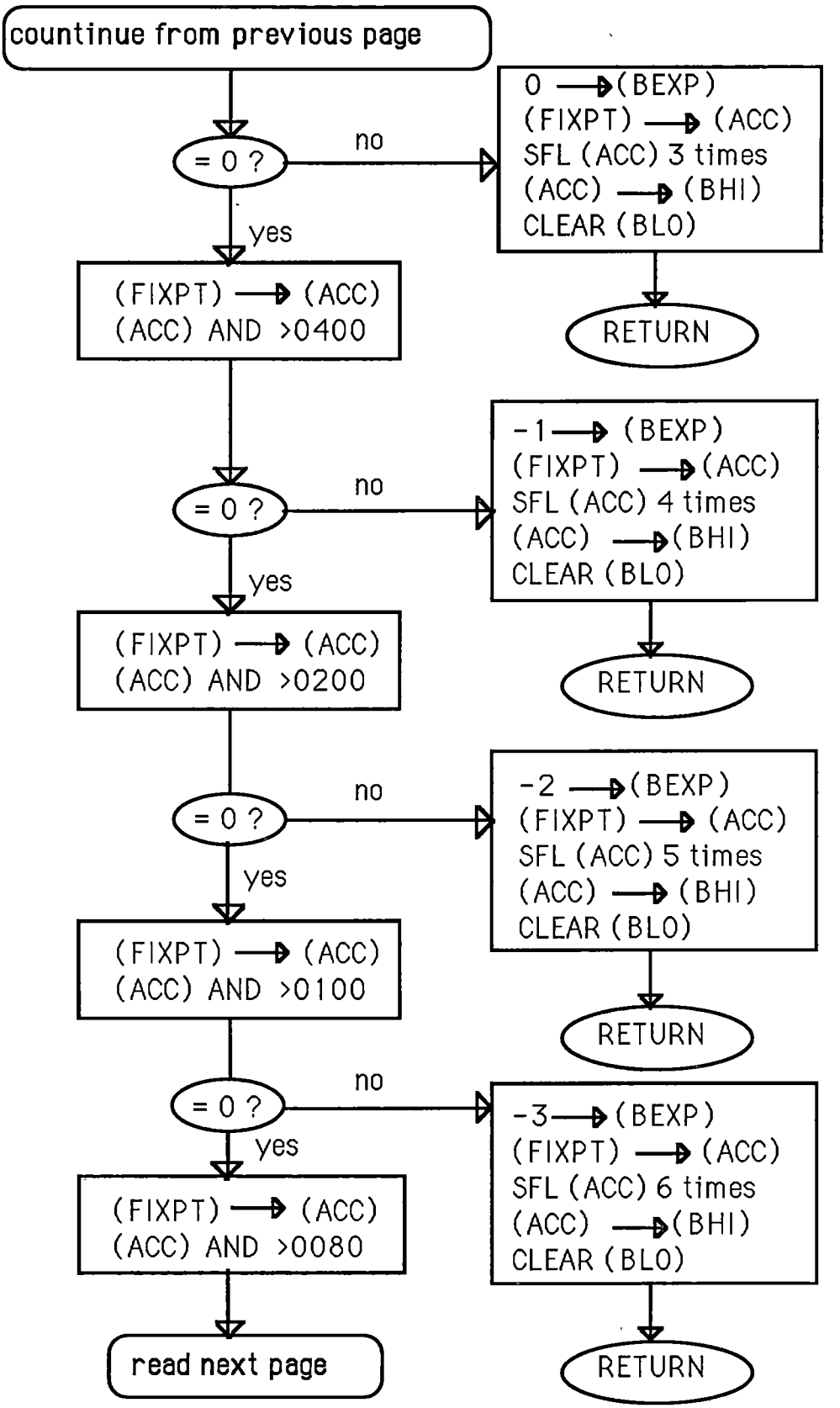


Figure 5.5.2-3(continue): The floating-point conversion routine (FXFL).

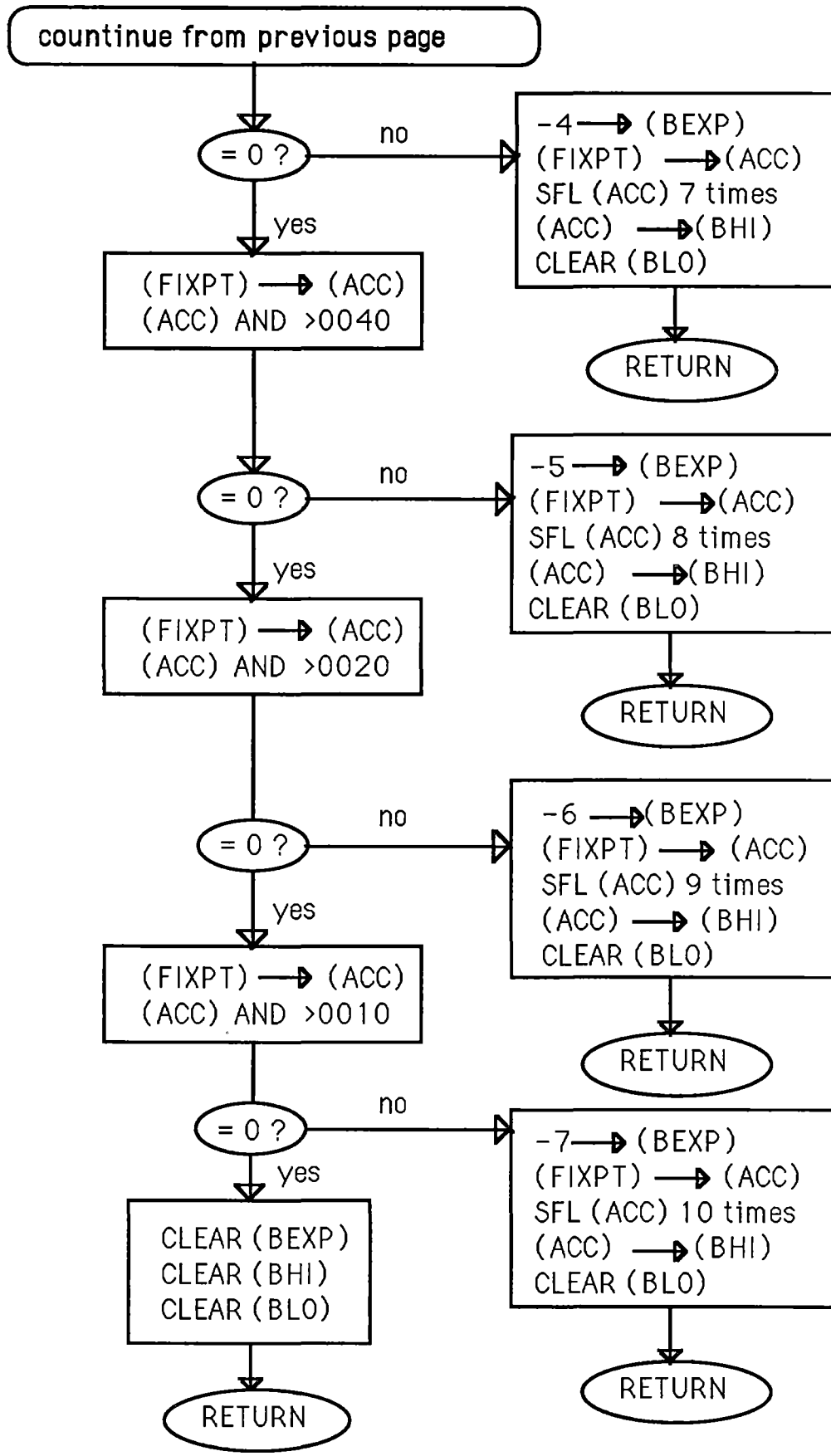


Figure 5.5.2-3(continue): The floating-point conversion routine (FXFL).

5.5.3 Floating point addition and multiplication routines

Floating point addition and multiplication routines were provided in the TMS libraries. The original software had fault in it. The TMS floating point addition and multiplication routines were corrected and modified to suit our applications. Details of modifications are shown in the appendix C. To minimize the execution time of the floating-point arithmetic, the two floating-point numbers A and B and the resultant output floating-point number C are represented in the format that is shown in figure (5.5.2-2).

Before executing the routine, the numbers A and B are stored at the fixed memory locations(>60 to >63 and >64 to >67, respectively). The result of the floating-point arithmetic is stored in C memory locations(>68 to >6B). This can then be transferred into any other desirable memory locations.

Flow chart diagrams are included in the three following sections (5.5.4 , 5.5.5 and 5.5.6). Due to word processor restrictions in flow-chart diagrams, $W_{p\,k+1}$ is represented as Wp(k+1) .

5.5.4 Plant Identification Routine (PIDTFY routine)

The plant identification routine (memory address >5D1 to >714) is used to identify a finite impulse response of the plant. This is a portion of the model-reference adaptive control and its block diagram is shown in figure (5.5.4-1).

The adaptive filter tap length is fixed at 40 . At each iteration, the tap weight vector, $W_{p\,k+1}$, is updated by the recursive equations (5.5.4-1) and (5.5.4-2) (as seen in chapter 2).

$$W_{p\,k+1} = W_{p\,k} + 2\,\mu e_k u_k \qquad (5.5.4-1)$$

$$where\ W_{p\,k} = [w_{p\,k},\ w_{p\,k-1},\ \cdots,\ w_{pk-L+1}]^T$$

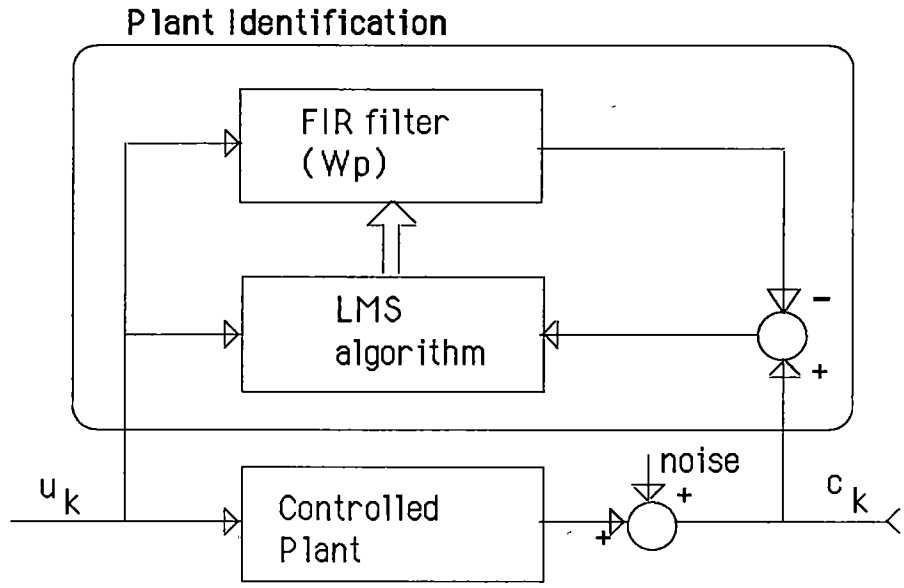


Figure 5.5.4-1: Plant identification routine block diagram.

e'_k is the error function at the k^{th} sampled iteration,

$$e'_k = c_k - U_k^T W_{p\,k} \qquad (5.5.4-2)$$

$$where\ U_k^T = [u_k,\ u_{k-1},\ \cdots,\ u_{k-L+1}]$$

Equations 5.5.4-1 and 5.5.4-2 show that, at each iteration we first calculate e'_k, then $2\mu_p e'_k$ then $2\mu_p e'_k u_k$ and finally $W_{p\,k+1}$.

Each floating-point number is represented by the 4-memory location format. Hence each element of the predicted FIR tap weight vector of the plant , $W_{p\,k}$, plant control input signal vector , U_k, and plant output signal vector , C_k , are all required 4-memory locations for each element of these vectors. (We define $C_k = [c_k,\ c_{k-1},\ \cdots,\ c_{k-L+1}]$). These vectors located as shown in figure (5.5.4-2).

40-tap FIR plant (Wp)

Address	Wp
>A00/3	wp_0
>A04/7	wp_1
>A98/B	wp_{38}
>A9C/F	wp_{39}

Plant control input signal (U_k)

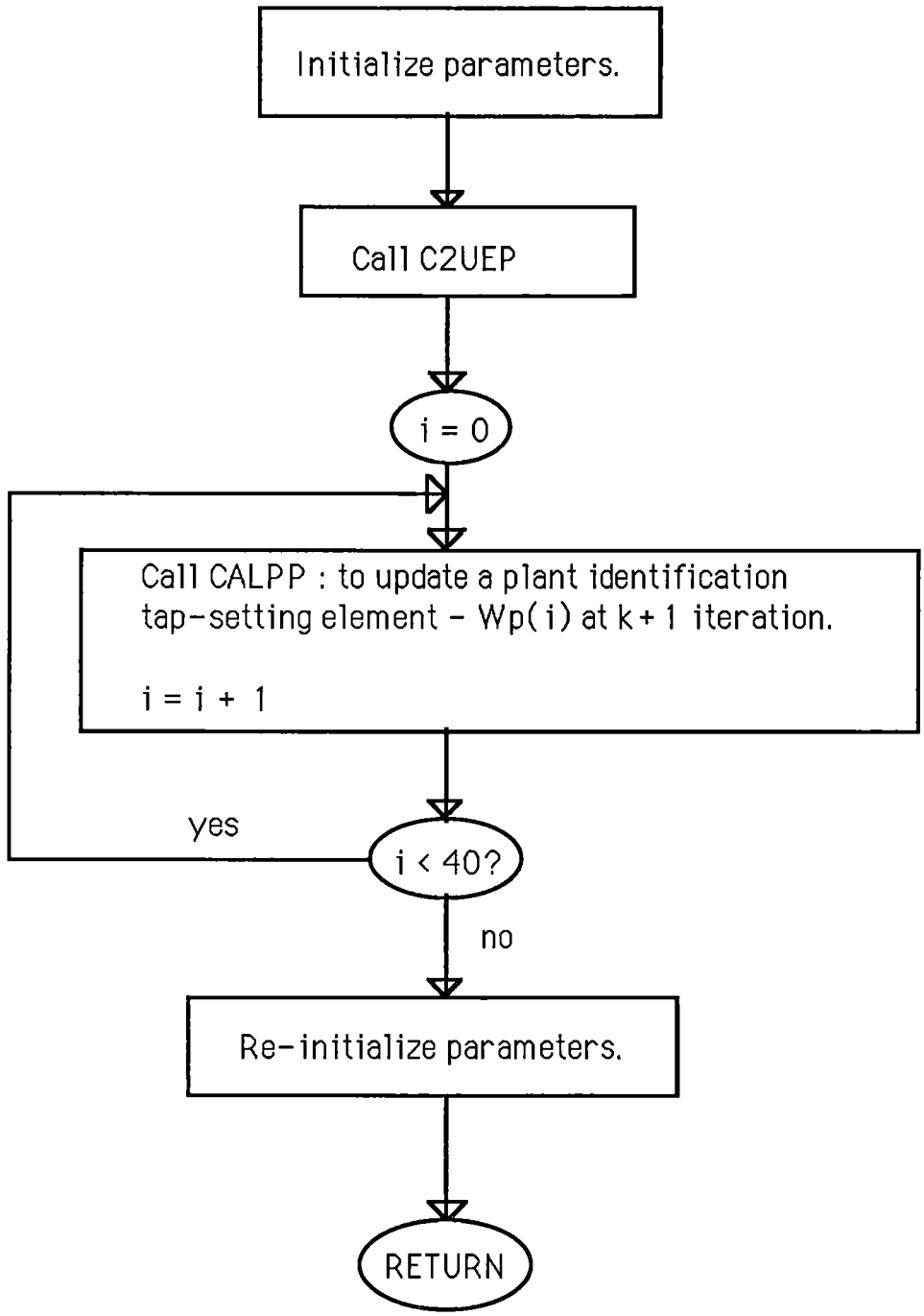
Address	U_k
>9F7/4	u_k
>9F3/0	u_{k-1}
>957/4	u_{k-38}
>95B/8	u_{k-39}

Plant output signal (C_k)

Address	C_k
>1EA7/4	c_k
>1EA3/0	c_{k-1}
>1E07/4	c_{k-38}
>1E0B/8	c_{k-39}

Figure 5.5.4-2 : The arrangement of plant tap-setting vector (Wp),plant control signal vector (Uk) and plant output signal (Ck).

The flow chart diagram of the plant identification routine, PIDTFY, the C2UE and the CALPP subroutines (see following pages) are shown in figures (5.5.4-3), (5.5.4-4a) and (5.5.4-4b) respectively. The C2UE subroutine calculates $2\mu_p e'_k$ and stores a result in the RAM ("P2UE" memory). The CALPP subroutine calculates $2\mu_p e'_k u_{k-i}$, then w_{pik+1} and stores an updated-tap element , w_{pik} , into its reserve RAM locations. (u_{k-i} and w_{pik+1} are the i^{th}-element of the control input vector, U_k, and the weight vector W_{pk+1}, respectively.)



Figuren5.5.4-3: The Plant Identification routine (PIDTFY) flow chart diagram.

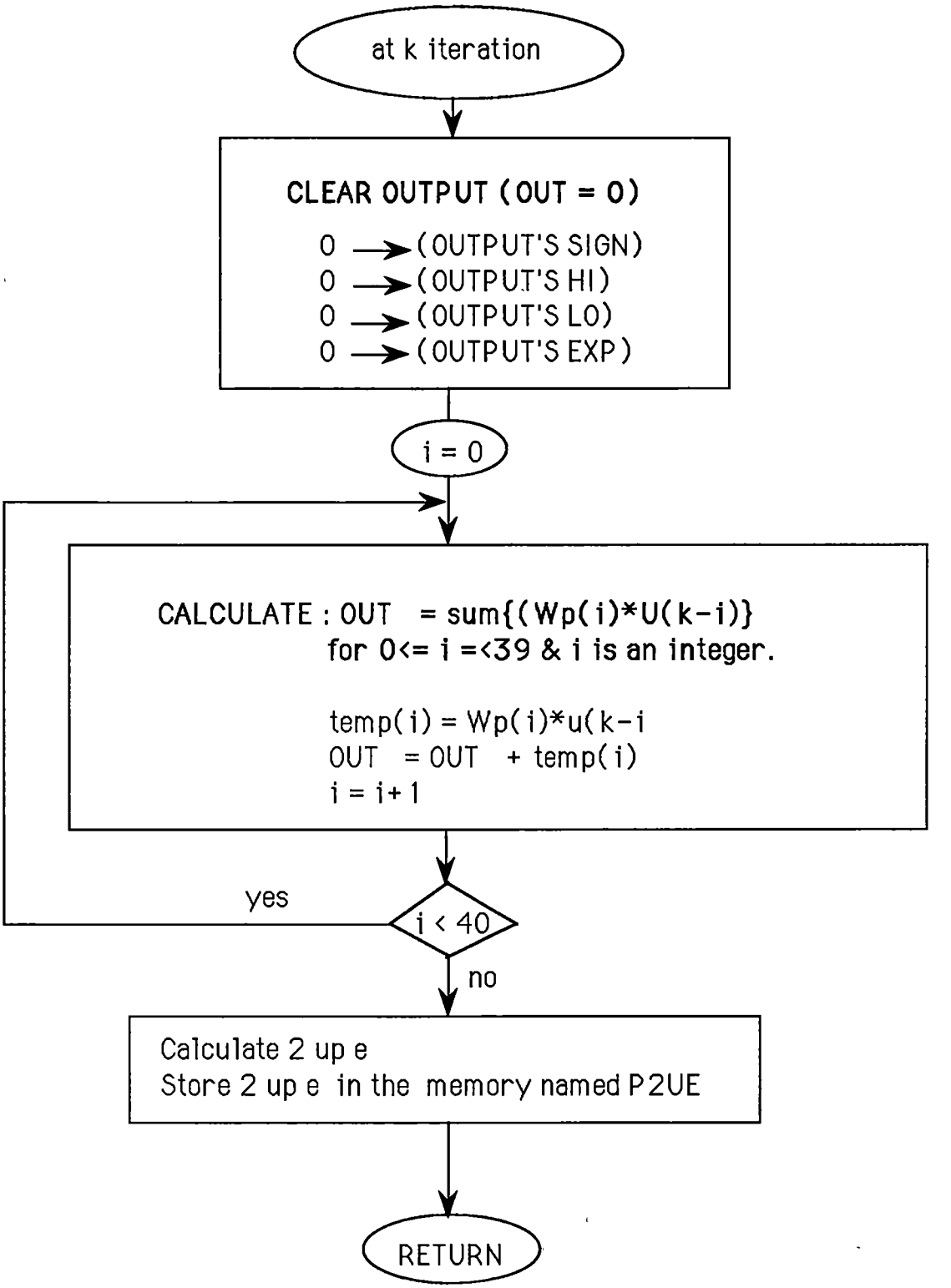


Figure 5.5.4-4a: The C2UEp subbroutine flow chart diagram.

Note in this diagram $2\mu_p e'_k$ is represented by 2 up e .

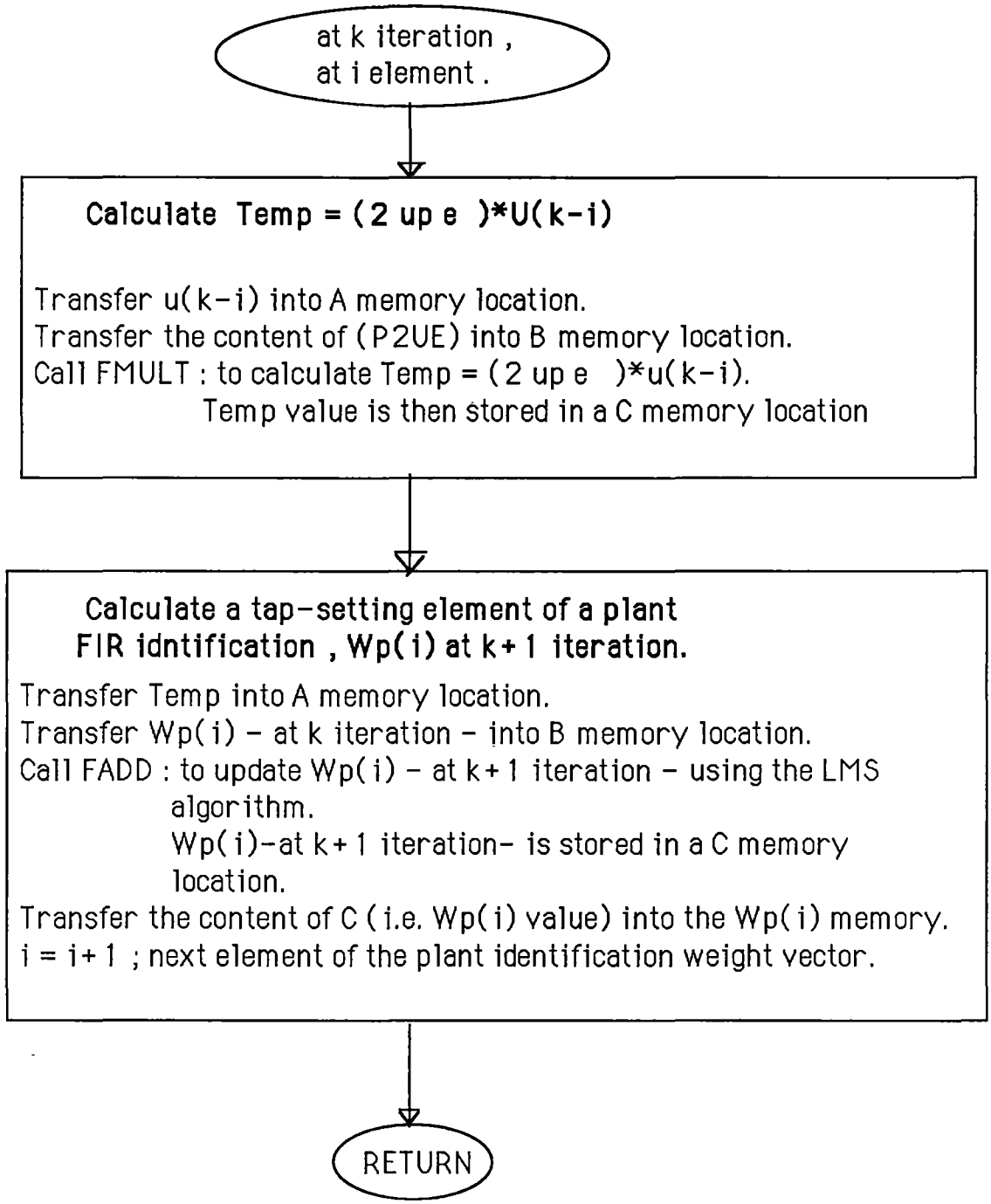


Figure 5.5.4-4b: The CALPP subroutine;
to update a plant tap-setting element.

Note in this diagram $2\mu_p e'_k$ is represented by 2 up e .

5.5.5 LMS controller Routine (FIR controller)

The LMS controller routine is the heart of the FIR controller software . Its block diagram is shown in figure (5.5.5-1). It is used to adjust the FIR controller tap-setting. Parameter μ and the controller tap-length , L, are programmable .

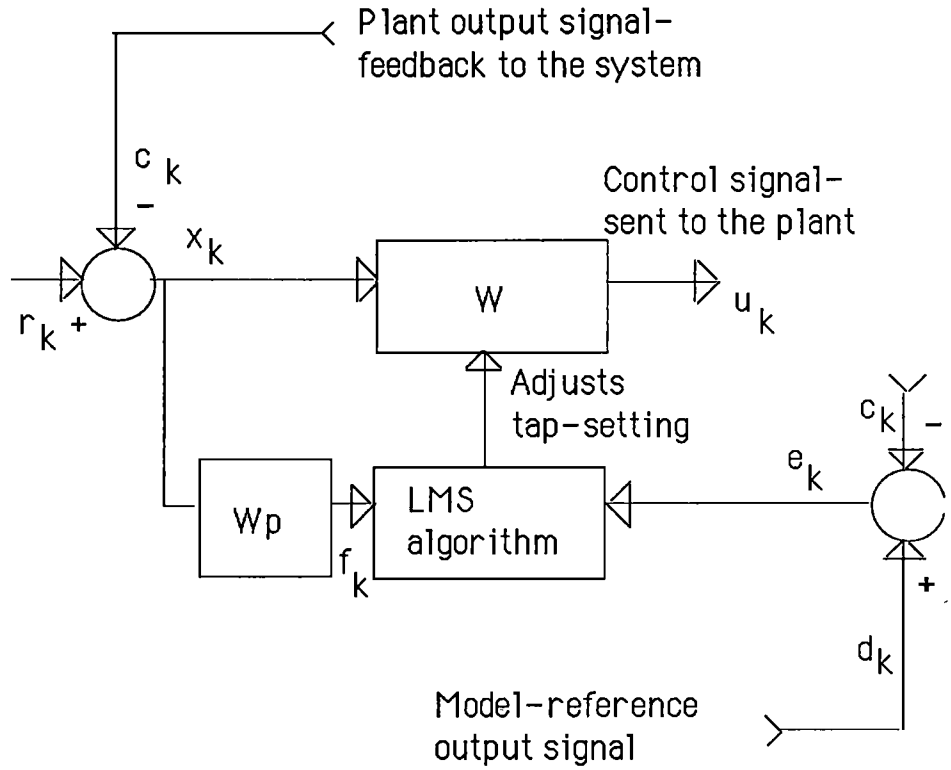


Figure 5.5.5-1: An FIR LMS controller block diagram.

The routine in every iteration performs the following calculations:

(i) It calculates

$$2\mu e_k \qquad where \quad e_k = d_k - c_k$$

d_k is a model reference output signal at the k^{th} iteration.

c_k is an plant output signal at the k^{th} iteration.

(ii) It determines the FIR plant tap-setting , W p, (calls the plant identification routine PIDTFY).

(iii) It calculates $f_k = W_p^T X_k$ where $x_k = r_k - c_k$; x_k is a first element of vector X_k , then using the LMS algorithm adjusts the FIR controller tap-setting.

(iv) It updates the controller tap-vector by calculating

$$a = (2\mu e_k)(W_p^T X_k) \quad then\ w_{i\,k+1} = w_{i\,k} + a \quad for\ 0 \le i \le L-1 \qquad (5.5.5-1)$$

LMS controller routine calculations are as following :

-Step 1 calculates $2\mu e_k$ where μ is a constant and e_k is an error signal between the model reference output signal and the plant output signal. The C2UE is used to perform this step . It is a sub-routine of the LMS controller routine . The C2UE sub-routine flow-chart diagram is shown in figure 5.5.5-3 .

In this diagram :

k is a time index; i is varied from 0 to L-1 and L is set at 40.

$2\mu e_k$ is represented by 2 up e .

e_k, c_k and d_k are represented by e, c and d , respectively.

-Step 2 calculates the weight vector of the FIR plant identification filter, W_p (using PIDTFY routine). This routine has been discussed in section 5.5-4.

-Step 3 calculates the output signal of the image filter (refer to figure 5.3-1). The tap setting W_p, calculated in step 2, is now used to calculate the output signal of the image filter , f_k ,where $f_k = W_p^T X_k$ and X_k is the input vector of the filter. The process is

shown in figure (5.5.5-4).

The flow chart diagram for this subroutine is shown in figure (5.5.5-5) .

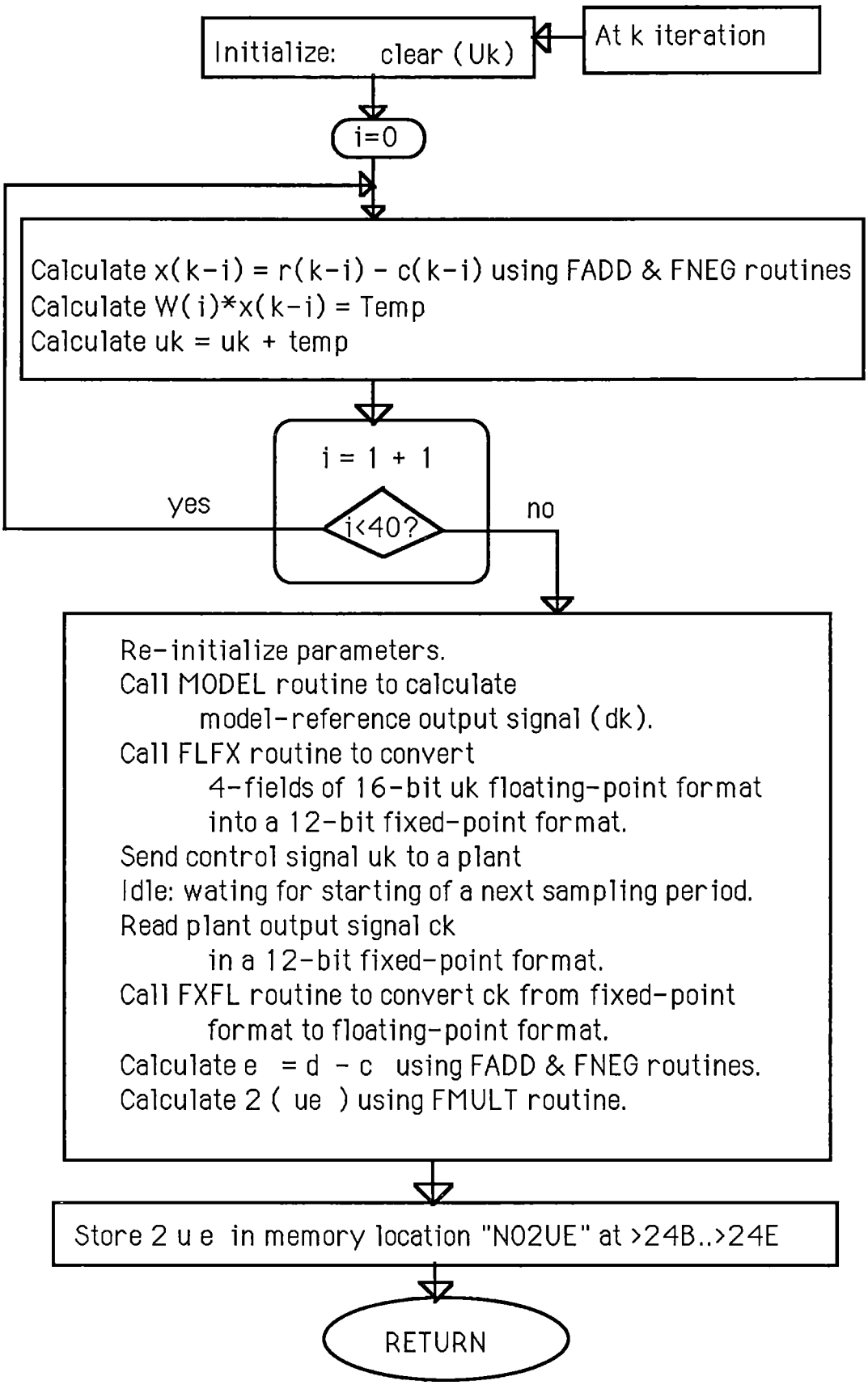


Figure 5.5.5-3:The C2UE routine flow chart diagram.

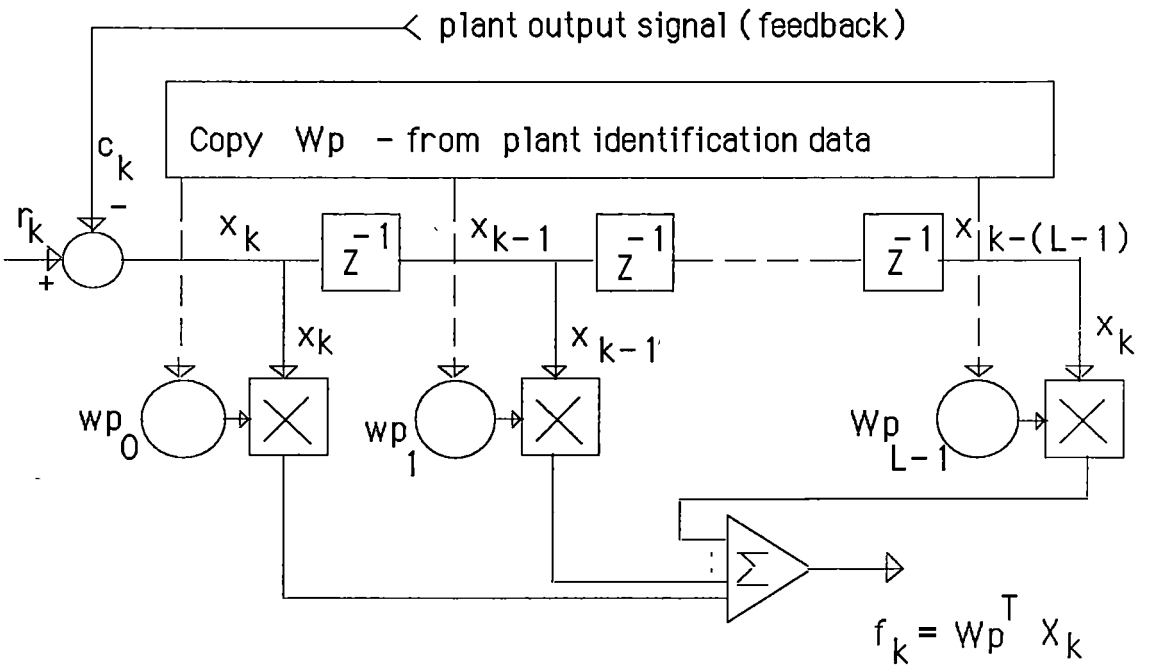


Figure 5.5.5-4 : A FIR tapped-delay line.

-Step 4 calculates W_{k+1}. The CALP subroutine calculates each element w_{ik} (where 0<i<39) of the 40-tap-control-weight-vector (W_k). It is calculated as in equation (5.5.5-2). (refer to chapter 2)

$$W_{k+1} = W_k + 2\mu e_k F_k \qquad (5.5.5-2)$$

$$where \quad W_k = [w_{0k}, w_{1k}, \cdots, w_{39\,k}]^T$$

$$and \quad F_k = [f_k, f_{k-1}, \cdots, f_{k-39}]^T$$

The $2\mu e_k$ is calculated from step 1 and is stored in the memory locations "NO2UE" .

Vectors W_k and F_k are located as shown in figure (5.5.5-6).

40-tap FIR controller (W_k)

Address	W_k
>B00/3	w_0
>B04/7	w_1
>B98/B	w_{38}
>B9C/F	w_{39}

Pre-filter output signal (F_k)

Address	F_k
>CA7/4	f_k
>CA3/0	f_{k-1}
>C07/4	f_{k-38}
>C0B/8	f_{k-39}

Figure 5.5.5-6: Relationship between W_k and F_k memories.

The flow-chart diagram of the CALP subroutine is as shown in figure (5.5.5-7).

The final flow chart for the LMS controller is in figure(5.5.5-8).

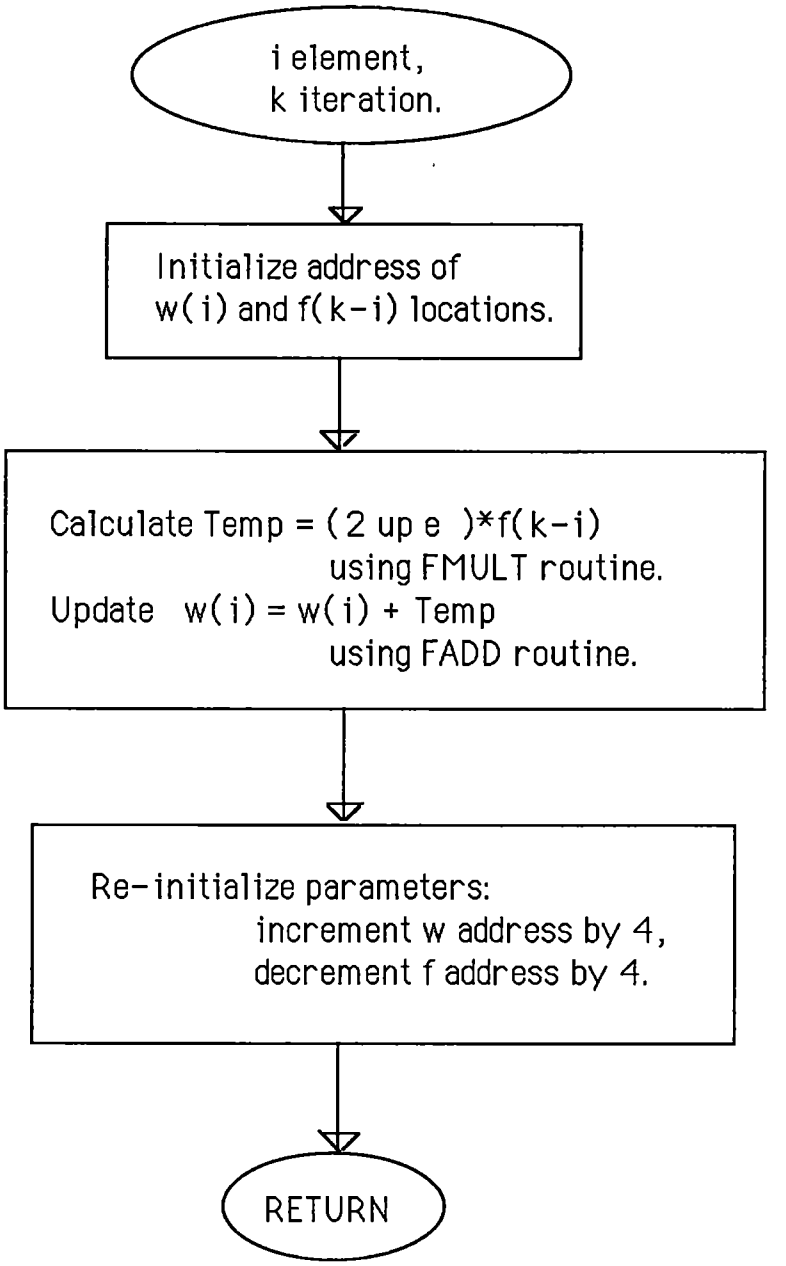


Figure 5.5.5-7: The CALP subroutine flow chart diagram.

Note in this diagram $2\mu e_k$ is represented by 2 up e .

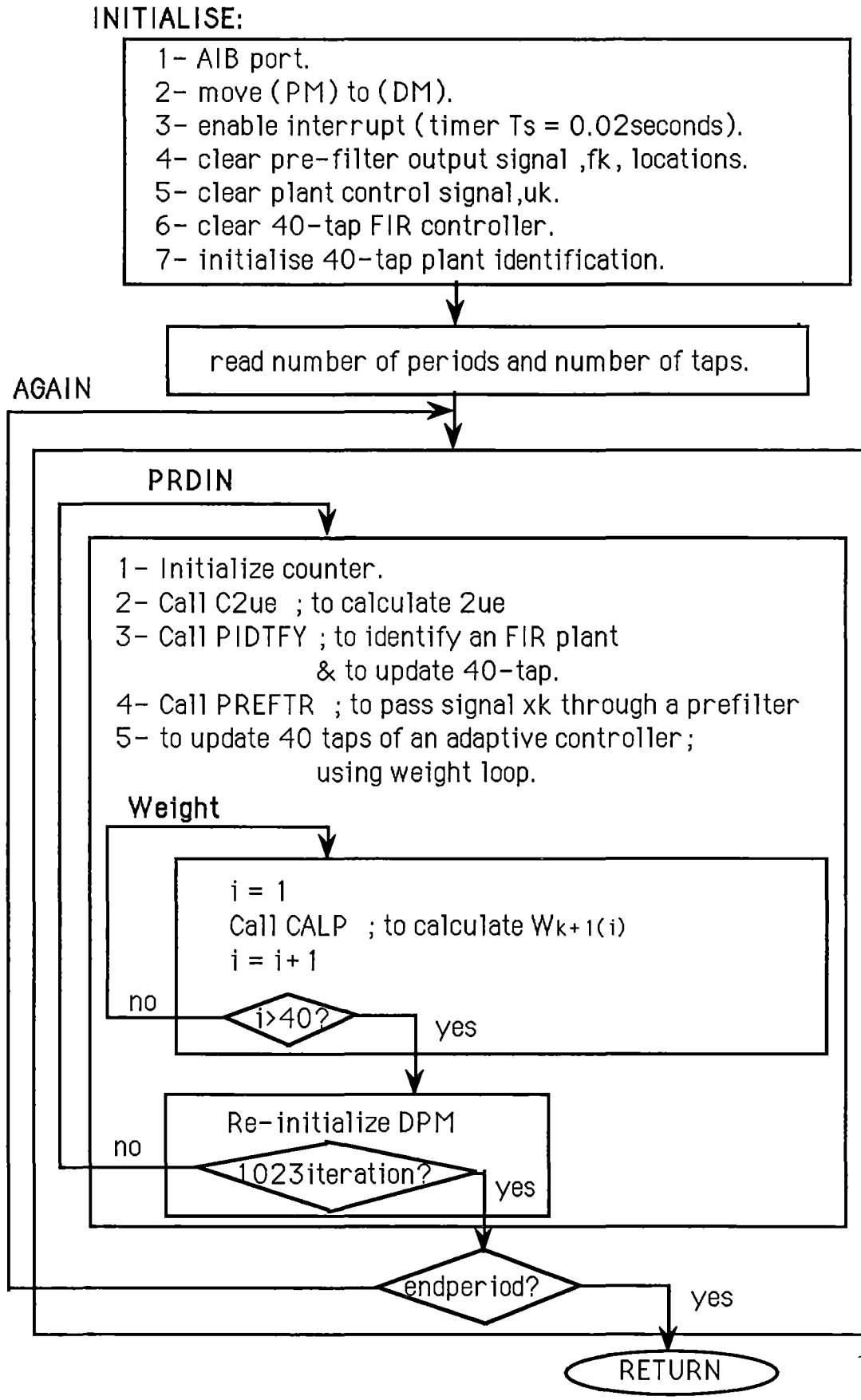


Figure 5.5.5-8: A LMS controller flow chart diagram.

5.5.6 Fixed-point conversion Routine [FLFX routine]

The external plant is in real-time therefore the digital control input signal must be converted to an analog form. The 12-bit D/A converter is used to convert a 12-bit digital signal to its analog signal.

However the LMS controller routine calculates the plant control input signal (u_k) in the four-memory 16-bit floating-point format. The fixed-point conversion routine was therefore written to communicate between the LMS controller routine and the D/A converter. (It is located from memory address >812 to >835).

In section 5.5-2 we have seen that by the means of the 12-bit A/D the external plant sends its analogue output signal to the digital controller. So here in order to make the same scalling factor, the floating point number from the range

$(-2^3 x 0.111\ 1111\ 1111\ ,\ +2^3 x 0.111\ 1111\ 1111\ )$

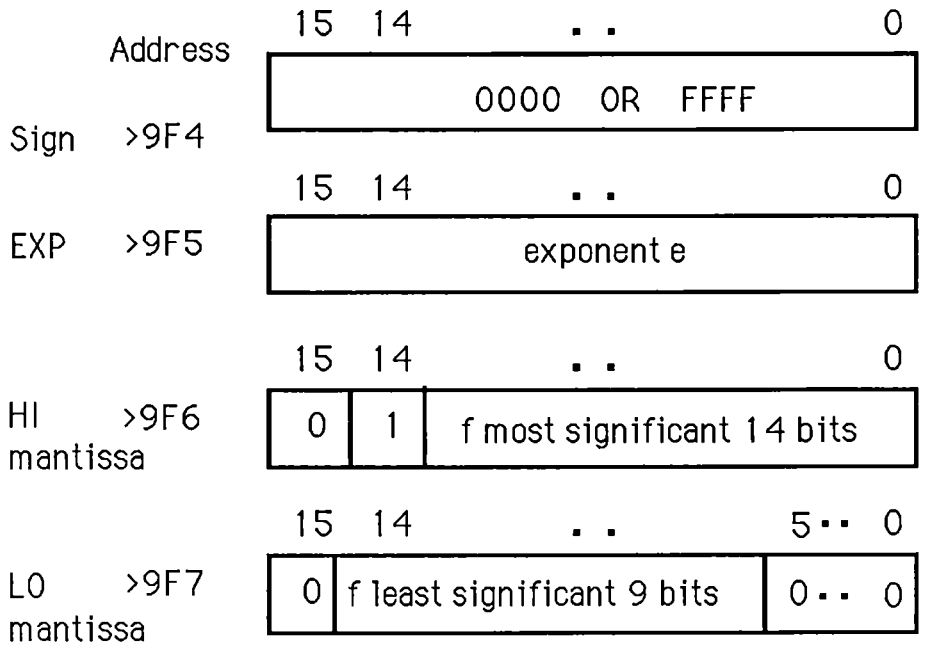
must be converted into the fixed-point format of

(0111 1111 1111 XXXX , 1000 0000 0000 XXXX)

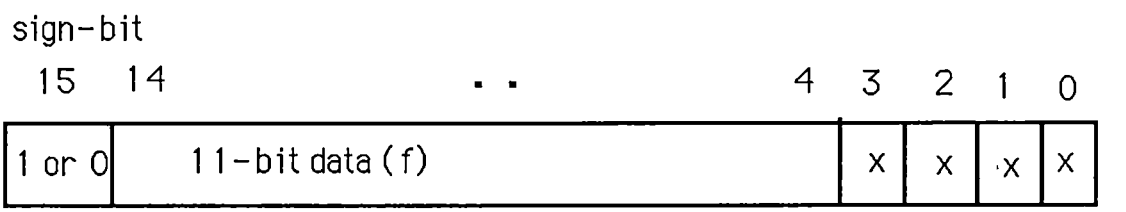
The fixed-point conversion routine [FLFX routine] converts the plant control input signal (u_k) which is represented in the floating point format, into the system standard 12-bit fixed-point format. The memory locations of a digital control input signal (u_k) are from >9F4 to >9F7. After conversion into the system 12-bit format, its signal is stored in the FIXPT memory location , at >7C address. It is then transmitted to the plant through the 12-bit D-A converter using the "OUT" instruction.

These memory locations are set up as in figure (5.5.6-1).

The flow-chart diagram of the FLFX routine is shown in figure (5.5.6-2).



The 4-field 16-bit control signal



The 12-bit fixed-point control signal

Figure 5.5.6-1: Control signal memory arrangements.

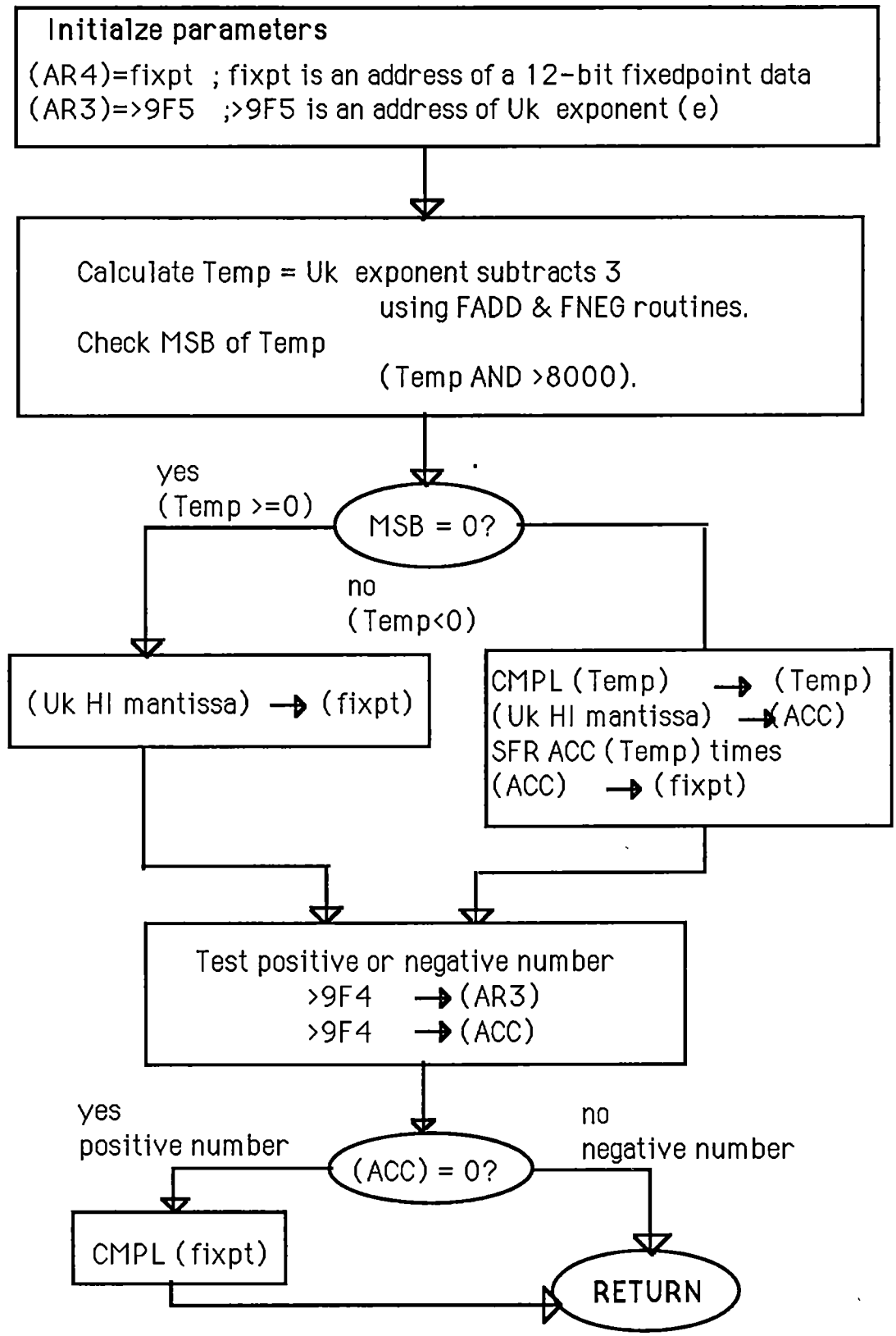


Figure 5.5.6-2: The FLFX routine flow chart diagram.

5.5.7 Reference Model Routine (MODEL routine)

A reference model can be implemented either in discrete or in continuous form. In our system, we used a discrete model. The reference model routine (MODEL routine), was written for this application.

The reference model routine reads in numerator and denominator coefficients of the model pulse transfer function and calculates its output signal subject to the reference input signal.

This subroutine is located from memory address >4D8 to >5D0 and performs the standard transfer function form of

$$M(q^{-1}) = \frac{(aq^{-1} + bq^{-2})q^{-d_m}}{(1 + cq^{-1} + dq^{-2})}$$

where a, b, c, d are constant and d_m are time-delays.

d_m was set at 4, however it can be easily modified to implement a suitable model time delay.

Chapter 6

CONCLUSIONS AND FUTURE WORK

6.1 CONCLUSIONS

In this thesis the Non-Parametric Model-Reference Adaptive Control (NP-MRAC) method is used to control a single-input single-output linear plant. By simulation on the NEC-APC-3 computer (chapters 3 and 4) and real-time hybrid simulation on the TMS320C20 signal processor (chapter 5) it has been shown that the Least Mean Square (LMS) algorithm can be used to effect a Non-Parametric Model Reference Adaptive Control (NP-MRAC).

The FIR adaptive filter is used to carry out the two functions in the NP-MRAC (refer to section 3.3 and 4.1). They are

(1) to determine the shape of the impulse response of the unknown or time-varying plant rather than the plant parameters,

(2) to adjust the shape of the finite impulse response of the controller rather than to determine the controller's parameters.

The first function is referred to as the Non-Parametric Plant Identification (NP-PI). In this function, the tap-weight vector of the FIR adaptive filter (FIR filter 1) is adjusted so that it matches with the plant impulse response. This is done by using an estimation error (the difference between the plant output and the FIR adaptive filter) as the input to the LMS identification algorithm . The FIR adaptive filter uses only feed-forward multipliers, therefore it is always stable as long as the step size parameter μ satisfy the condition

$$0 < \mu < \frac{1}{\lambda_{max}} \qquad (6.1-1)$$

where λ_{max} is the largest eigenvalue of the correlation matrix of the tap input vector , R.

In the NP-PI method, it is not required to determine the plant parameters. Therefore, when the plant has both parameter and time delay variations , the NP-MRAC does not face same problems as the STC and MRAC face when the plant has both parameter and time delay variations. (refer to chapter 1 and 3).

The second function causes the error signal, e_k, (between the model output, d_k, and the plant output, c_k) to minimize so that the compensated plant output response matches with the reference model output response. In this function, the tap weight vector of the FIR adaptive filter (FIR filter 2) is adjusted by using both the error estimation ($e_k = d_k - c_k$) and the tap-weight vector of the plant identification FIR filter (calculated from the first function) as the input to the LMS adaptive algorithm.

There are two types of NP-MRAC methods: an open loop NP-MRAC and a closed loop NP-MRAC . The theory of the open loop NP-MRAC is discussed in chapter 2 .Similar to the NP-PI method, the open loop NP-MRAC method uses only feed-forward multipliers, therefore it is always stable under the condition of equation (6.1-1). In the case of the closed loop NP-MRAC method the theory is not yet explored, however the simulations show that the system is stable. (refer to chapter 4 and 5.)

This thesis has shown the effectiveness of the NP-MRAC based on the LMS algorithm. Simulations were done for the NP-PI, the open loop NP-MRAC and the closed loop NP-MRAC, using the NEC-APC-3 computer. Simulation results have shown that the NP-MRAC can well handle the plant with both parameter variation and time

delay variation. Good set point tracking property can also be achieved since zero cancellation is not involved in this method. The LMS algorithm is simple, therefore it is easy to be implemented on a microprocessor to control a plant in real-time. The closed loop NP-MRAC was implemented on the TMS320C20 signal processor to control a plant in real time. The plant with parameter variation was built on an analog computer. The program was written in TMS320C20 assembly language, with 1.8K word length. The control program was able to handle a minimum sampling period of 10ms for the 40-tap FIR adaptive filter. This sampling period could be reduce even further if any later version of a TMS320 was implemented (eg., TMS320C25 or TMS320C30) . Thus the NP-MRAC based on the LMS algorithm is a superior scheme to control a fast plant.

As mentioned , the LMS algorithm was used as an adaptive algorithm for the NP-MRAC in this thesis. The LMS algorithm does not require measurements of the pertinent correlation function, nor does it require matrix inversion. It was selected as an adaptive algorithm because of its simplicity.

The principal parameters that affect the response of the LMS algorithm are: the step size of the parameter, μ, the number of taps of the FIR adaptive filter, L, and the eigenvalues of the correlation matrix of the tap input vector. The third factor is required for the effective operation of the algorithm. However when the eigenvalue spread is large , the LMS algorithm slows down since it requires a large number of iterations to converge. (Widrow and Stearn, 1985) . This is the main drawback of the LMS algorithm when applied to the NP-MRAC. This problem needs further investigation. We suggest that the least-square (LS) algorithm may be applied to the NP-MRAC.

6.2 A NP-MRAC USING RECURSIVE LEAST SQUARE ALGORITHM (RLS)

Briefly, the LMS algorithm is derived from averages with the result that one filter (optimum in a probabilistic sense) is obtained for all the operational environment (assuming the input signal is stationary in a wide-sense). On the other hand, the method of Recursive Least Squares (RLS) yields a different filter for each collection of input data (Haykin S., 1986).

In the RLS algorithm, the index of performance which consists of the sum of weighted error squares is minimized. (the error is defined in chapter 2). The derivation of the RLS algorithm relies on a basic result in linear algebra known as the matrix-inversion lemma. An important feature of the RLS algorithm is that it utilizes all the information contained in the input data, extending back to the instant of time when the algorithm is initiated. The resulting rate of convergence is therefore typically an order of magnitude faster than the simple LMS algorithm. This improvement in performance, however, is achieved at the expense of large increase in computational complexity.

Included in this section is the result of the simulation of the Non-Parametric Plant Identification NP-PI using LS algorithm based on U-D factorization by Bierman and Thornton (see Astrom and Wittenmark,1984).

The continuous-time transfer function plant ,to be controlled, was selected as

$$P(s) = \frac{w_n^2}{s^2 + 2\xi w_n s + w_n^2} \qquad (6.2-1)$$

$$where \quad \xi = 0.2 \quad and \quad w_n = 2$$

With a sampling period of 0.2 seconds, the discrete time transfer function was

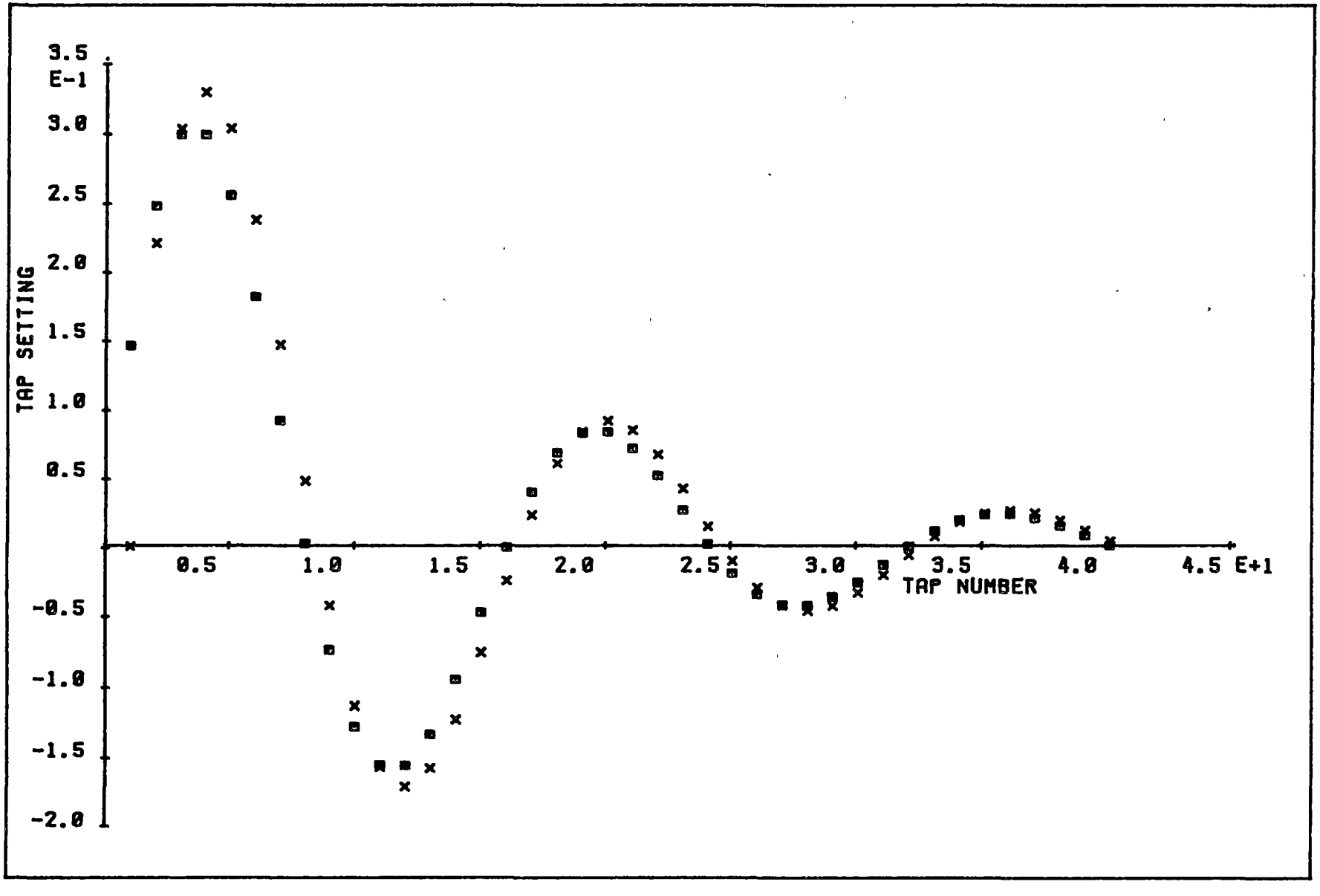
$$P(q^{-1}) = \frac{0.218849q^{-1} - 0.072953q^{-2}}{1 - 1.706248q^{-1} + 0.852144q^{-2}} \qquad (6.2-2)$$

The simulation were done on the NEC-APC-3 computer using the 40-tap FIR adaptive filter.

Figure (6.2-1) shows the shape of the tap-weight vector and the Finite Impulse Response (FIR) of the plant. In this thesis the theory of the NP-MRAC based on the LS algorithm is not discussed. However the convergence of the least square methods is explained in many Control or Adaptive Filter text books (e.g.,. Astrom and Wittenmark-1985, Haykin S. -1989). Simulation of the NP-PI method shows that the LS algorithm is ideal for adjusting the FIR adaptive filter applied in the NP-MRAC. The proof and demonstration of this are left to an interested reader.

The following page is figure (6.2-1).

" The shape of the tap-weight vector, [X] and the Finite impulse Response (FIR) of the plant [R] "


3.5
E-1
3.0
2.5
2.0
1.5
1.0
0.5
-0.5
-1.0
-1.5
-2.0
TAP SETTING
0.5
1.0
1.5
2.0
2.5
3.0
3.5
4.0
4.5 E+1
TAP NUMBER

6.3 FUTURE WORK

Although, the NP-MRAC based on the LMS algorithm has good adaptation capability, further studies are still required before the method can be implemented widely. Below is a list of few suggestions for future work on this topic.

(1) The proof for convergence of the closed loop NP-MRAC using the LMS algorithm should be developed.

(2) The NP-MRAC using the LMS algorithm has been successful when used to control the plant whose characteristic function was set up on an analog computer (chapter 5). To make the theory useful in industry the tests should include a real plant as a controlled element.

(3) The major drawback of the LMS algorithm is that its convergence rate is rather slow, while the Recursive-Least-Square (RLS) algorithm promises a fast rate of convergence. Despite the fact that RLS computation is rather complex, it is an ideal algorithm for the NP-MRAC where the plant sampling period is not a critical element. The NP-MRAC using the RLS algorithm provides a scope for further studies.

APPENDIX A

Impulse response of a second-order system

A second-order system which was used in this thesis has a continuous pulse transfer function of the form :

$$H(s) = \frac{w_n^2}{s^2 + 2\xi w_n s + w_n^2}$$

where $0 \leq \xi < 1$.

The impulse response of this system in a discrete form is given by (Katsuhiko Ogata, 1970, Mayhan,1983)

$$h_k = \frac{T w_n}{\sqrt{1-\xi^2}} \left\{ \sin\left(w_n \sqrt{1-\xi^2}\right) kT \right\} e^{-\xi w_n kT} u_k$$

where

$$u_k = \begin{cases} 1 & ; k \geq 0 \\ 0 & ; k < 0 \end{cases}$$

and k is a time index.

APPENDIX B

This appendix contains three programs , which was written in Fortran 77. They were used to test the Non-Parametric Plant Identification and the Non-Parametric Model Reference Adaptive Control by simulations on the NEC-APC-3 computer (refer to chapter 3 and chapter 4).

This program was written for the Non-Parametric Plant Identification simulations (chapter 3).

```
      INTEGER I,J,H,L,M,Q,B,STEP,TERM,A,PERIOD,C,S,P,K,O,O1
      INTEGER X(10),Y0(1143),REF(1143)
      REAL TIME,U,WN,E,T,F0,F1,F2,F3,F4,F5
      REAL D(1143),W(120),IDEAL(120),UNIT(120),WAV(120)
      REAL   HZSQ(103),OUTPLN(103),DSQ(103),OUTSQ(1143)

      COMMON/STORE1/REF
      COMMON/STORE2/D
      COMMON/STORE3/L
      COMMON/STORE4/W
      COMMON/STORE5/WN,E,T
      COMMON/STORE6/U
      COMMON/STORE7/OUTSQ
      COMMON/STORE8/WAV

C     VARIABLES FOR PLOTTING
      LOGICAL TED,TED2
      CHARACTER*25       FNAME,HZNAME,VTNAME,FNAME2,HZ2NAME,VT2NAME
      CHARACTER*1        KEY,KEY2

      OPEN(100,FILE= 'WEIGHT',STATUS='NEW')

C     READ INFORMATIONS OF A REFERENCE MODEL
C     WRITE(*,5)
C  5  FORMAT(//,3X,'TYPE IN W,E,T')
C     READ(*,*)WN,E,T

      WN = 2.0
      E = 0.1
      T = 0.2

C     STEP = 10
C     READ INFORMATION OF CONDITIONS OF A LMS ALG.

      WRITE(*,15)
 15   FORMAT(/,3X,'HOW MANY PERIODS ?, NCHOICE?,STEP?')
      READ(*,*)PERIOD,NCHOICE,STEP
      WRITE(*,17)
 17   FORMAT(/,3X,'HOW MANY TAPS ?')
      READ(*,*)L
      WRITE(*,19)
 19   FORMAT(/,3X,'U? ')
      READ(*,*)U

C     GENERATE PRBS

C     STORE 10 INITIAL VALUES INPUT
      DO 20 I = 1, 10
      X(I) = STEP
 20   CONTINUE

      DO 60 J = 1, 1023
        Y0(J) = X(10)
C       SHIFT VALUES TO THE RIGHT
        DO 25 H =2,10
          X(12-H) = X(11-H)
 25     CONTINUE
```

```
C   TAKE EXCLUSIVE 10TH & 7TH TERMS
C   NOTE SINCE VALUES ARE ALREADY SHIFTED
C   THUS 7TH TERM NOW INDEED 8TH TERM
C   AND Y0(J) IS A 10TH TERM

    TERM = STEP*STEP
    IF (Y0(J)*X(8) .EQ. TERM) GOTO 30
     X(1) = STEP
     GOTO 40
30    X(1) = -STEP
40   CONTINUE
60  CONTINUE

    DO 62 S = 1 ,L
     W(S) = 0.0
     OUTSQ(S) = 0.0
     Y0(1023+S) = Y0(S)
     IDEAL(S) = WIMP(S-1)
     UNIT(S)  = 1.0*S
62  CONTINUE

    IF (NCHOICE .EQ. 1) THEN
      REF(1) = Y0(1)
      REF(2) = Y0(2)
    ENDIF
    IF (NCHOICE .EQ. 2) THEN
      REF(1) = STEP
      REF(2) = STEP
    ENDIF

C   CALCULATE DESIRE VALUES
C   FNUM1 = 0.218849
C   FNUM2 = -0.072953
C   DEN1  = -1.706248
C   DEN2  = 0 852144

    FNUM1 = 0.151731
    FNUM2 = -0.000003
    DEN1  = -1.771388
    DEN2  = 0.923116

    D(1) = 0.0
    D(2) = 0.0
    DO 65 M=3,1023+L
     IF (NCHOICE .EQ. 1) REF(M) = Y0(M)
     IF (NCHOICE .EQ. 2) THEN
       IF ((M .LT. 400+L) .OR. (M .GT. 800+L)) THEN
        REF(M) = STEP
       ELSE
        REF(M) = -STEP
       ENDIF
     ENDIF

     D(M) = FNUM1*REF(M-1)+FNUM2*REF(M-2)
    +         -DEN1*D(M-1)-DEN2*D(M-2)
65  CONTINUE

    DO 67 I3 = 1,PERIOD
     DO 66 I2 = 1,L
66      OUTSQ(I2) = OUTSQ(1023+I2)
    CALL WEIGHT

    DO 70 I = 1,103
      I1 = (L+1)+((I-1)*10)
      HZSQ(I) = (I1-I)*1.0
     OUTPLN(I) = OUTSQ(I1)
     DSQ(I) = D(I1)
70  CONTINUE

C   PRINT TAPS' OUTPUT VALUES
C   CALCULATE ALL IDEAL WEIGHTS VALUE
C   USING DISCRETE IMPULSE FUNCTION
```

```
C   AND PRINT RESULTS OF TAPS BY USING LMS ALG.

    DO 140 Q = 1,L
      WRITE(*,150)Q,W(Q),IDEAL(Q),WAV(Q)
      WRITE(100,150)Q,W(Q),IDEAL(Q),WAV(Q)
150    FORMAT(/,3X,'AT ',I3,' W , IDEAL,WAV ',3F12.8)
140 CONTINUE

C   PLOT TAPS' VALUES
    FNAME ='FIRPAR'
    CALL QOPEN(FNAME)
    CALL QFRAME(1)

    TED = .TRUE.
    CALL QRANGE(UNIT,W,L,TED)
    CALL QRANGE(UNIT,IDEAL,L,TED)
    HZNAME = 'TAP NUMBER'
    VTNAME = 'TAP SETTING'
    CALL QXYAXES(HZNAME,VTNAME)
    CALL QMARK(UNIT,W,L,1,5,1)
    CALL QMARK(UNIT,IDEAL,L,2,6,1)
    READ(*,230)KEY
230 FORMAT(A1)
    CALL QCLOSE

C   PLOT TAPS' VALUES
    FNAME ='FIRAVPAR'
    CALL QOPEN(FNAME)
    CALL QFRAME(1)
    TED = .TRUE.
    CALL QRANGE(UNIT,WAV,L,TED)
    CALL QRANGE(UNIT,IDEAL,L,TED)
    HZNAME = 'TAP NUMBER'
    VTNAME = 'AVERAGE TAP SETTING'
    CALL QXYAXES(HZNAME,VTNAME)
    CALL QMARK(UNIT,WAV,L,1,5,1)
    CALL QMARK(UNIT,IDEAL,L,2,6,1)
    READ(*,232)KEY
232 FORMAT(A1)
    CALL QCLOSE
C %%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%

C   PLOT TAPS' VALUES
    IF (NCHOICE .EQ. 2) THEN
    FNAME2 ='SQRESP'
    CALL QOPEN(FNAME2)
    CALL QFRAME(1)
    TED2 = .TRUE.
    CALL QRANGE(HZSQ,OUTPLN,103,TED2)
    CALL QRANGE(HZSQ,DSQ,103,TED2)
    HZ2NAME = 'NUMBER OF ITERATIONS'
    VT2NAME = 'SQUARE RESPONSE'
    CALL QXYAXES(HZ2NAME,VT2NAME)
    CALL QPLOT(HZSQ,OUTPLN,103,1,1)
    CALL QPLOT(HZSQ,DSQ,103,2,1)
    READ(*,231)KEY2
231 FORMAT(A1)
    CALL QCLOSE
    ENDIF

67  CONTINUE

C %%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%
    CLOSE(100)
    CLOSE(12)
    STOP
    END

C   CALCULATE WEIGHT FUNCTION USING L.M.S. THEOREM
    SUBROUTINE WEIGHT
    INTEGER L,C,K,B,H,REF(1143)
    REAL    U,W(120),OUTSQ(1143),D(1143),WAV(120)
```

```
      COMMON/STORE1/REF
      COMMON/STORE2/D
      COMMON/STORE3/L
      COMMON/STORE4/W
      COMMON/STORE6/U
      COMMON/STORE7/OUTSQ
      COMMON/STORE8/WAV

      DO 20 I = 1,L
20    WAV(I) = 0.0
      DO 40 K=L+1,1023+L
        OUTSQ(K) = TAP(K)
        ERROR = D(K)- OUTSQ(K)
        DO 30 B = 1,L
          W(B) = W(B) + 2*U*ERROR*REF(K-B+1)
          WAV(B) = WAV(B) + W(B)/1023
30    CONTINUE
40    CONTINUE

      RETURN
      END

C     CALCULTATE TAP OUTPUT VALUES
      REAL FUNCTION TAP(K)
      INTEGER A,L,REF(1143)
      REAL    OUT,W(120)
      COMMON/STORE1/REF
      COMMON/STORE3/L
      COMMON/STORE4/W

      OUT = 0.0
      DO 20 A = K,K-L+1,-1
        OUT = OUT + (REF(A)*W(K-A+1))
20    CONTINUE
      TAP = OUT
      RETURN
      END

C     CALCULATE ALL WEIGHTS VALUES
C     USING DISCRETE IMPULSE FUNCTION

      REAL FUNCTION WIMP(K)
      REAL F1,E,T,WN
      COMMON/STORE5/WN,E,T

      F1 = SQRT(1-E*E)
      WIMP = WN*T/F1*SIN(WN*F1*K*T)*EXP(-E*WN*K*T)
      RETURN
      END
```

This program was written for the open loop Non-Parametric Model Reference Adaptive Control simulations (chapter 3).

```
      INTEGER H,B,bb,TERM,A,PERIOD,C
      INTEGER HH,CC
      INTEGER X(1023),Y0(1066),AA,AAA,AB,AC
      REAL    Y(1066),WK0(40),NSEFTOR
      REAL    ST(100),ST0(100),INST,INST1,INST2
      REAL    HTIME(100),UNIT(40),U,TS,NOISE(1066),IDEN(1066)
      REAL    YC(1066),W(40),OUTPLT(1066),DF(1066),er(102)
      REAL    TERM0(1066),TERM1(1066),TERM2(1066)
      REAL    OUTP(102),DFP(102) , KP,TP,KM,TM
      REAL    NUMF0,NUMF1,NUMF2,MNUMF0,MNUMF1,MNUMF2
      REAL    denf1,denf2,MDENF1,MDENF2
      REAL    FCTOR0,MFCTOR0,WNP,EP,F1P,F2P,F3P,F4P
      REAL    WNM,EM,F1M,F2M,F3M,F4M
      real    NST(200),NSTM(200),NST0(200),NINST,NINST1,NINST2,GM,GP
      REAL    PLTNSE(102)
      REAL    HZ(1023),HZP(102),SUMTAPS,modgain
      REAL WP(40),WPAV(40)
```

```
      REAL hn(200),WAV(40)
      INTEGER  PRD1,prd2
      REAL   PLTOUT(78),PLTDF(78),TAPC(2,78),HZPIN(78)
      REAL FINST(100),FINST1(100),FINST2(100)
      INTEGER D1,D3,AC0,AD,AE,AF,AG

C     VARIABLES FOR PLOTTING
      LOGICAL TED,TED22,TED3,TED4,TED5
      CHARACTER*25 FNAME,HZNAME,VTNAME
      CHARACTER*25        FNAME3,HZ3,VT3,FNAM4,H4NAME,V4NAME,FNAME5,HZ5,VT5
      CHARACTER*25 F22NAME,H22NAME,V22NAME
      CHARACTER*1 KEY,KEY22,KEY3,KEY5
      CHARACTER*50 NSTNAME,MODNAME,NST0NAME
      CHARACTER*3  NAMEB0,NAMEB1,NAMEB2,NAMEA1,NAMEA2
      CHARACTER*60 TLE3,TLE32,TLE33,TLE4

      COMMON/STORE1/K,IDELAY
      COMMON/STORE2/TERM0,TERM1,TERM2,OUTPLT
      COMMON/STORE3/L
      COMMON/STORE4/W,YC
      COMMON/STORE6/MDELAY,INIT
      COMMON/STORE7/DF,NOISE,WK0,IDEN,Y,WP,UP,WPAV
      COMMON/STORE8/HZ,WAV
      COMMON/STORE9/U,NUMF0,NUMF1,NUMF2,MNUMF0,MNUM
     +          MNUM F2, DENF1,DENF2,MDENF1,MDENF2,NSEFTOR,TS

      OPEN(10,FILE='FBKOUT',STATUS='NEW')
       open(12,file='data',status='new')

C     INFORMATONS OF DESIRED MODEL
C     ASK FOR INFORMATIONS OF A PLANT MODELING

       WRITE(*,11)
11    FORMAT(/,3X,'GP?,GM?')
      READ(*,*)GP,GM
      WRITE(*,13)
13    FORMAT(/,3X,'INIT & NSEFTOR ?')

      READ(*,*)INIT,NSEFTOR
C     LMS ADAPTIVE FILTER INFORMATIONS
      WRITE(*,15)
15    FORMAT(/,3X,'PRD1,prd2, PERIODS ?')
      READ(*,*)PRD1,prd2,PERIOD
      L = 40
      WRITE(*,17)
17    FORMAT(/,3X,'TS? , U? , UP?')
      READ(*,*)TS,U,UP
      WRITE(*,19)
19    FORMAT(/,3X,'NSTEP?, NCHOICE?')
      READ(*,*)NSTEP,NCHOICE

C     GENERATE PRBS
C     STORE 10 INITIAL VALUES INPUT
      DO 20 I=1,10
20      X(I) = NSTEP
      DO 40 J=1,1023
        Y0(J) = X(10)
C     SHIFT VALUES TO THE RIGHT
      DO 32 H=2,10
32      X(12-H) = X(11-H)

      TERM = NSTEP**2
      IF (Y0(J)*X(8).EQ.TERM) GOTO 30
       X(1) = NSTEP
       GOTO 40
30     X(1) = -NSTEP
40    CONTINUE

      DO 62 I=1,L+3
62      Y0(1023+I) = Y0(I)
      DO 61 I = 1,L
        WP(I) = 0.0
```

```
      wk0(I)= 0.0
61  CONTINUE

    DO 63 I = 1,1066
     IF (NCHOICE .EQ. 1) YC(I) = Y0(I)
     IF (NCHOICE .EQ. 2) THEN
        IF ( (I .LT. 400+L+3) .OR. (I .GT. 800+L+3) ) THEN
        YC(I) = NSTEP
       ELSE
        YC(I) = -NSTEP
       ENDIF
     ENDIF
63  CONTINUE
      NUMF0 = 0.22361
      NUMF1 = 2*NUMF0
      NUMF2 = NUMF0
      DENF1 = -0.74776
      DENF2 = 0.64222

      MNUMF0 = 0.03356
      MNUMF1 = 2*MNUMF0
      MNUMF2 = MNUMF0
      MDENF1 = -1.5302
      MDENF2 = 0.66443

    DO 66 ML=1,L+3
      NOISE(ML) = RND(INIT)*NSEFTOR
      TERM0(ML) = 0.0
      TERM1(ML) = 0.0
      TERM2(ML) = 0.0
      DF(ML)    = NOISE(ML)
      OUTPLT(ML)= NOISE(ML)
      IDEN(ML)  = NOISE(ML)
66  CONTINUE
    WRITE(*,67)
67  FORMAT(/,3X,'IDELAY?,MDELAY ')
    READ(*,*)IDELAY,MDELAY
    CALL WEIGHT
    WRITE(*,69)
69  FORMAT(/,3X,' PERIOD 1')
    WRITE(10,97)GP,GM,NSEFTOR
97  FORMAT(/,3X,'GP ',F5.2,'  GM ',F5.2,'  NSEFTOR ',F5.2)
    WRITE(10,96)PERIOD,TS,U
96  FORMAT(/,3X,'PERIOD ',I3,'  TS ',F7.4,'  U ',F15.10)
    WRITE(10,95)NSTEP,NCHOICE
95  FORMAT(/,3X,'NSTEP,NCHOICE ',2I3)

C   REPEAT LOOP
    DO 175 A=2,PRD1
    DO  173   MA = 1,L+3
      TERM0(MA) = TERM0(1023+MA)
      TERM1(MA) = TERM1(1023+MA)
      TERM2(MA) = TERM2(1023+MA)
      YC(MA)    = YC(1023+MA)
      OUTPLT(MA)= OUTPLT(1023+MA)
      IDEN(MA)  = IDEN(1023+MA)
      NOISE(MA) = NOISE(1023+MA)
      DF(MA)    = DF(1023+MA)
173 CONTINUE
     CALL WEIGHT
    WRITE(*,169)A
169 FORMAT(/,3X,' PERIOD ',I3)
175 CONTINUE

    GOTO 200

C   REPEAT LOOP
C   MEMORISE PLANT PARAMETERS
 C  CHANGE PARAMETERS OF A PLANT

201    NUMF0 = 0.0
       NUMF1 = 2.4*GP
```

```
     NUMF2 = -1.92*GP
     DENF1 = -0.1
     DENF2 = -0.42

    DO 186 A=prd1+1,PRD2
    DO 174  MA = 1,L+3
     TERM0(MA) = TERM0(1023+MA)
     TERM1(MA) = TERM1(1023+MA)
     TERM2(MA) = TERM2(1023+MA)
     YC(MA)    = YC(1023+MA)
     OUTPLT(MA)= OUTPLT(1023+MA)
     IDEN(MA)  = IDEN(1023+MA)
     NOISE(MA) = NOISE(1023+MA)
     DF(MA)    = DF(1023+MA)
174 CONTINUE
     CALL WEIGHT
    WRITE(*,171)A
171 FORMAT(/,3X,' PERIOD ',I3)
186 CONTINUE

    GOTO 200
C   REPEAT LOOP
C   PLANT CHANGES BACK TO ITS ORIGINAL PARAMETERS

202   NUMF0 = 0.22361
     NUMF1 = 2*NUMF0
     NUMF2 = NUMF0
     DENF1 = -0.74776
     DENF2 = 0.64222

    DO 75 A=PRD2+1,PERIOD
    DO 73  MA = 1,L+3
     TERM0(MA) = TERM0(1023+MA)
     TERM1(MA) = TERM1(1023+MA)
     TERM2(MA) = TERM2(1023+MA)
     YC(MA)    = YC(1023+MA)
     OUTPLT(MA)= OUTPLT(1023+MA)
     IDEN(MA)  = IDEN(1023+MA)
     NOISE(MA) = NOISE(1023+MA)
     DF(MA)    = DF(1023+MA)
73  CONTINUE
     CALL WEIGHT
    WRITE(*,70)A
70  FORMAT(/,3X,' PERIOD ',I3)
75  CONTINUE
200 DO 132 III = 1,L
132        UNIT(III)=III

    SUMTAPS = 0.0
    DO 135 II3 = 1,L
135   SUMTAPS = SUMTAPS + WAV(II3)
    WRITE(10,94)SUMTAPS
94  FORMAT(/,3X,'SUMTAPS ',F15.10)

    DO 136 II4 = 1,L
136    WAV(II4) = WAV(II4)/SUMTAPS

C   %%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%
C   PLOT TAPS' VALUES

    FNAME ='TAPS'
    CALL QOPEN(FNAME)
    CALL QFRAME(1)
    TED = .TRUE.
    CALL QRANGE(UNIT,WAV,40,TED)
    HZNAME = 'TAP NUMBER'
    VTNAME = 'VALUES OF TAP SETTING'
    CALL QXYAXES(HZNAME,VTNAME)
    CALL QMARK(UNIT,WAV,40,1,2,1)
    READ(*,131)KEY
131 FORMAT(A1)
    CALL QCLOSE
```

```
C   PLOT PLANT FIR PARAMETERS
    FNAME ='PLNFIR'
    CALL QOPEN(FNAME)
    CALL QFRAME(1)
    TED = .TRUE.
    CALL QRANGE(UNIT,WPAV,40,TED)
    HZNAME = 'PLANT FIR PARAMETER NUMBER'
    VTNAME = 'VALUES OF PLANT FIR PARAMETERS'
    CALL QXYAXES(HZNAME,VTNAME)
    CALL QMARK(UNIT,WPAV,40,1,2,1)
    READ(*,133)KEY
133 FORMAT(A1)
    CALL QCLOSE

    DO 141 I=1,102
    I1 = 1+(I-1)*10
    HZP(I)  = HZ(I1)
    PLTNSE(I)= NOISE(I1)
     outp(I) = outplt(I1)
     dfp(I)  = df(I1)
C    er(I)   = dfp(I)-outp(I)
141 CONTINUE

C %%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%-
C   PLOT STEP RESPONSE FOR CLOSED LOOP FEEDBACK CONTROL SYSTEM

    F22NAME = 'OUT&MOD'
    CALL QOPEN(F22NAME)
    CALL QFRAME(1)
    TED22 = .TRUE.
    CALL QRANGE(HZP,OUTP,102,TED22)
    CALL QRANGE(HZP,DFP,102,TED22)
    H22NAME = 'TIME IN SEC'
    V22NAME = 'OUTPUT WAVEFORMS'
    CALL QXYAXES(H22NAME,V22NAME)
    CALL QPLOT(HZP,OUTP,102,1,1)
    CALL QPLOT(HZP,DFP,102,2,2)
    READ(*,147)KEY22
147 FORMAT(A1)
    CALL QCLOSE

    IF (A .LE. PRD2 ) GOTO 201
    IF (A .LE. PERIOD) GOTO 202
C   INITIALISE

   DO 251 I = 1,MDELAY
     NSTM(I)= 0.0
     NST(I) = 0.0
     ST(I)  = 0.0
     NST0(I)= 0.0
251 CONTINUE

C   NOFEEDBACK Square RESPONSE

    DO 145 I = 1,L
145 W(I) = WAV(I)
    NST(IDELAY+1) = NUMF0*W(1)
    NINST1 = W(1)
    NINST  = W(1)+W(2)
    NST(IDELAY+2) = NUMF0*NINST +NUMF1*NINST1-DENF1*NST(IDELAY+1)
    DO 253 I0=3,L
     NINST2 = NINST1
     NINST1 = NINST
     NINST  = NINST + W(I0)
     NST(IDELAY+I0) =                  NUMF0*NINST+NUMF1*NINST1+NUMF2*NINST2
    +        -DENF1*NST(IDELAY+I0-1) - DENF2*NST(IDELAY+I0-2)
253 CONTINUE
    NST(IDELAY+L+1) = NUMF0*NINST+NUMF1*NINST+NUMF2*NINST1
    +       -DENF1*NST(IDELAY+L)-DENF2*NST(IDELAY+L-1)
    DO 255 I1=L+2,100+idelay
     NST(I1) = (NUMF0+NUMF1+NUMF2)*NINST
    +   -DENF1*NST(I1-1)-DENF2*NST(I1-2)
```

```
255 CONTINUE

    do 257 i=1,l
     ninst2 = ninst1
     ninst1 = ninst
     ninst  = ninst-w(i)
     nst(100+idelay+i) = numf0*ninst+numf1*ninst1+numf2*ninst2
    +        -denf1*nst(100+idelay+i-1)-denf2*nst(100+idelay+i-2)
257 continue
    nst(100+idelay+l+1)=(numf0+numf1)*ninst+numf2*ninst1
    +        -denf1*nst(100+idelay+l)-denf2*nst(100+idelay+l-1)
    nst(100+idelay+l+2)=(numf0+numf1+numf2)*ninst
    +        -denf1*nst(100+idelay+l+1)-denf2*nst(100+idelay+l)
    do 259 i1 = l+100+idelay+3,200
     nst(i1) = (numf0+numf1+numf2)*ninst-denf1*nst(i1-1)
    +        -denf2*nst(i1-2)
259 continue

C   MODEL
    NSTM(MDELAY+1) = MNUMF0
    NSTM(MDELAY+2) = MNUMF0+MNUMF1-MDENF1*NSTM(MDELAY+1)
    DO 261 I2=3+MDELAY,100+mdelay
     NSTM(I2)=(MNUMF0+MNUMF1+MNUMF2)
    +       -MDENF1*NSTM(I2-1)-MDENF2*NSTM(I2-2)
261 CONTINUE
    nstm(mdelay+101) = mnumf1+mnumf2-mdenf1*nstm(mdelay+100)
    +        -mdenf2*nstm(mdelay+99)
    nstm(mdelay+102) = mnumf2-mdenf1*nstm(mdelay+101)
    +        -mdenf2*nstm(mdelay+100)
    do 263   i2 = mdelay+103,200
     nstm(i2) = -mdenf1*nstm(i2-1)-mdenf2*nstm(i2-2)
263 continue

    NST0(IDELAY+1) = NUMF0
    NST0(IDELAY+2) = NUMF0+NUMF1-DENF1*NST0(IDELAY+1)
    DO 101 I2=3+IDELAY,100+Idelay
     NST0(I2)=(NUMF0+NUMF1+NUMF2)
    +       -DENF1*NST0(I2-1)-DENF2*NST0(I2-2)
101 CONTINUE
    nst0(Idelay+101) = numf1+numf2-denf1*nst0(Idelay+100)
    +        -denf2*nst0(Idelay+99)
    nst0(Idelay+102) = numf2-denf1*nst0(Idelay+101)
    +        -denf2*nst0(Idelay+100)
    do 110   i2 = Idelay+103,200
     nst0(i2) = -denf1*nst0(i2-1)-denf2*nst0(i2-2)
110 continue

    do 10 i = 1,200
     hn(i) = i * ts
10  continue

C   STEP RESPONSE OF A SYSTEM WITH NO FEEDBACK
    FNAME3 = 'SQRESP'
    CALL QOPEN(FNAME3)
    CALL QFRAME(1)
    TED3 = .TRUE.
    CALL QRANGE(Hn,NST,200,TED3)
    CALL QRANGE(Hn,NSTM,200,TED3)
    HZ3 = 'TIME IN SECONDS'
    VT3 = 'STEP RESPONSE'
    CALL QXYAXES(HZ3,VT3)
    CALL QPLOT(Hn,NST,200,1,LSTYLE)
    CALL QPLOT(Hn,NSTM,200,1,LSTYLE)
    READ(*,160)KEY3
160 FORMAT(A1)
   CALL QCLOSE

C    ---------------------------------------------
C   STEP RESPONSE OF A SYSTEM WITH NO FEEDBACK
    FNAME5 = 'STRESP'
    CALL QOPEN(FNAME5)
    CALL QFRAME(1)
```

```
      TED5 = .TRUE.
      CALL QRANGE(Hn,NST,100,TED5)
      CALL QRANGE(Hn,NSTM,100,TED5)
      CALL QRANGE(HN,NST0,100,TED5)
      HZ5 = 'TIME IN SECONDS'
      VT5 = 'STEP RESPONSE'
      CALL QXYAXES(HZ5,VT5)
      CALL QPLOT(Hn,NST,100,1,1)
      CALL QPLOT(Hn,NSTM,100,1,1)
      CALL QPLOT(HN,NST0,100,1,1)
      READ(*,112)KEY5
112   FORMAT(A1)
      CALL QCLOSE

C     ------------------------------------------
      WRITE(*,161)SUMTAPS
161   FORMAT(/,3X,'SUMTAPS = ',F15.5)
      CLOSE(10)
      close(12)

      STOP
      END

C %%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%
C  CALCULATE WEIGHT FUNCTION USING L.M.S THEOREM

      SUBROUTINE WEIGHT
      INTEGER B,A,E,AA
      REAL   Y(1066),WK0(40),WK1(40),WK2(40),NOISE(1066),NSEFTOR
      REAL U,W(40),YC(1066),OUTPLT(1066),IDEN(1066)
      REAL DF(1066),NUMF0,NUMF1,NUMF2,MNUMF0,MNUMF1,MNUMF2
      REAL DENF1,DENF2,MDENF1,MDENF2,HZ(1023)
      REAL TERM0(1066),TERM1(1066),TERM2(1066)
      REAL   WP(40),WPAV(40),WAV(40)

      COMMON/STORE1/K,IDELAY
      COMMON/STORE2/TERM0,TERM1,TERM2,OUTPLT
      COMMON/STORE3/L
      COMMON/STORE4/W,YC
      COMMON/STORE6/MDELAY,INIT
      COMMON/STORE7/DF,NOISE,WK0,IDEN,Y,WP,UP,WPAV
      COMMON/STORE8/HZ,WAV
      COMMON/STORE9/U,NUMF0,NUMF1,NUMF2,MNUMF0,MNUMF1,
     +       MNUMF2 DENF1,DENF2,MDENF1,MDENF2,NSEFTOR,TS


C  CALCULATE WEIGHTS

      DO 21 I = 1,L
       WAV(I)  = 0.0
       WPAV(I) = 0.0
21    CONTINUE

      DO 40 K=L+4,1023+L+3
      TERM2(K) = TERM1(K-1)
      TERM1(K) = TERM0(K-1)
       DO 42 I=1,L
42       W(I)=WK0(I)
      TERM0(K) = PLT(K)
       NOISE(K) = RND(INIT)*NSEFTOR
      OUTPLT(K)= NUMF0*TERM0(K-IDELAY)+NUMF1*TERM1(K-IDELAY)
     +   +NUMF2*TERM2(K-IDELAY)-DENF1*OUTPLT(K-1)-DENF2*OUTPLT(K-2)
     +   +NOISE(K)

      OUT = 0.0
      DO 16 I = K,K-L+1,-1
       OUT = OUT + (TERM0(I)*WP(K-I+1))
16    CONTINUE
      ER = OUTPLT(K) - OUT

      DO 15 I = 1,L
       WP(I) = WP(I) + 2*UP*ER*TERM0(K-I+1)
```

```
      WPAV(I) = WPAV(I) + (WP(I)/1023)
15  CONTINUE

    IDEN(K) = 0.0
    DO 14 I = 1,L
14    IDEN(K) = IDEN(K) + WP(I)*YC(K-I+1)

22  DF(K) = MNUMF0*YC(K-MDELAY)+MNUMF1*YC(K-1-MDELAY)
   +   +MNUMF2*YC(K-2-MDELAY)-MDENF1*DF(K-1)
     +                    -MDENF2*DF(K- 2)

91    DO 30 B=1,L
       WK0(B)=WK0(B) + 2*U*(DF(K)-OUTPLT(K))*IDEN(K-B+1)
       WAV(B)=WAV(B) + WK0(B)/1023
30    CONTINUE

    HZ(K-43) = (K-43)*1.0
40  CONTINUE

    RETURN
    END

C  %%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%

C   CALCULATE TAP OUTPUT VALUES
    REAL FUNCTION PLT(K)
    INTEGER A,L
    REAL    OUT,W(40),YC(1066)

    COMMON/STORE3/L
    COMMON/STORE4/W,YC

    OUT = 0.0
    DO 20 A=K,K-L+1,-1
      OUT = OUT + (YC(A)*W(K-A+1))
20  CONTINUE
    PLT = OUT
    RETURN
    END

C  %%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%

C   GENERATE NOISE
    REAL FUNCTION RND(IS)
    IS = MOD(193*IS,37447)
    RND= ((IS/37447.0)-0.5)
    RETURN
    END
```

This program was written for the closed loop Non-Parametric Model Reference Adaptive Control simulations (chapter 4).

```
    INTEGER I,J,S,KC,K,H,L,M,bb,TERM,PERIOD,STEP,IA,CHOICE
    INTEGER HH,III,MM,CC,MDELAY,NPDELAY
    INTEGER X(1023),Y0(1066),AA,AAA,AB,AC,ML,MN,MA,MB,INIT
    REAL    Y(1066),WK0(40),FCTR
    REAL    ST(200),ST0(200),INST,INST1,INST2,NSEFTOR,WAV(40)
    REAL    HTIME(200),UNIT(40),U,TS,NOISE(1066),IDEN(1066)
    REAL    YC(1066),W(40),OUTPLT(1066),DF(1066)
    REAL TERM0(1066),TERM1(1066),TERM2(1066),TERM3(1066)
    REAL    NUM1,NUM2,NUM3,MNUM4,mnum5,MNUM6,WP(40),WPAV(40)
    REAL den1,den2,DEN3,MDEN1,MDEN2
    REAL    FCTOR0,MFCTOR0,WNP,EP,F1P,F2P,F3P,F4P,hsq(200)
    real    NST(500),NSTM(500),NST0(200),NINST,NINST1,NINST2
    INTEGER I0,I1,I2,HHH,HH1,II2,II1,II3,II4
    REAL    HZ(1023),modgain
```

```
      REAL HZSQ(107),OUTSQ(107),DFSQ(107),INCNT(642),HZIN(642)
      INTEGER  IRLS,JRLS,NRLS,PRD1,prd2,INPUT
      REAL HZP(103),TAP1(103),TAP11(103),TAP21(103),TAP31(103)
      REAL TAP41(103),YY(1066)

      INTEGER JC,JC2
      REAL   HST(500)

      INTEGER D1,D2,D3,AC0,AD,AE,AF,AG
C     VARIABLES FOR PLOTTING
      LOGICAL TED,TED12,tedST,TEDSQ,tedrf,TEDCNT,TEDP,TEDT
      CHARACTER*25          FNAME,HZNAME,VTNAME,FNAME12,HNAME,VNAME,FNAMEP
      CHARACTER*25              fnameST,hZST,fnamerf,hzrf,vtrf,HZPNAME,VTPNAME
      CHARACTER*25 FSQ,HZSQNAME,VTSQ,FCNTIN,HZCNT,VTCNT
      CHARACTER*25 vTST,HZT,VTT,FNAMET
      CHARACTER*1 KEY,KEY12,keyST,KEYSQ,keyrf,KEYCNT,KEYP,KEYT
      CHARACTER*20 NAME1,NAME11,NAME21,NAME31,NAM41
      CHARACTER*60 TITLE,TLE12,TITP

      COMMON/STORE1/K,NPDELAY
      COMMON/STORE2/TERM0,TERM1,TERM2,TERM3,OUTPLT
      COMMON/STORE3/L
      COMMON/STORE4/W,YC
      COMMON/STORE6/MDELAY,INIT,IA,PERIOD
      COMMON/STORE7/NOISE,WK0,WAV,IDEN,U,NSEFTOR,WPAV,WP,UP
      COMMON/STORE8/HZ,HZP,TAP1,TAP11,TAP21,TAP31,TAP41
      COMMON/STORE9/NUM1,NUM2,NUM3,DEN1,DEN2,DEN3,YY,GM,FCTR,DF,
     + MNUM4,MNUM5,MNUM6,MDEN1,MDEN2 ,TS

      OPEN(10,FILE='FBKOUT',STATUS='NEW')
C     ------------------------------------------------

C     INFORMATONS OF DESIRED MODEL

C     ASK FOR INFORMATIONS OF A PLANT MODELING

      WRITE(*,5)
5     FORMAT(/,3X,'INIT,CHOICE,nplant?')
      READ(*,*)INIT,CHOICE,nplant
      WRITE(*,7)
7     FORMAT(/,3X,' FCTR , U ,NSEFTOR?,TS?,UP,GM ')
      READ(*,*)FCTR,U,NSEFTOR,TS,UP,GM
C     LMS ADAPTIVE FILTER INFORMATIONS
      WRITE(*,9)
9     FORMAT(/,3X,'PRD1,prd2, PERIODS ?')
      READ(*,*)PRD1,prd2,PERIOD

      L = 40

      WRITE(*,11)
11    FORMAT(/,3X,' NPDELAY,MDELAY ?')
      READ(*,*)NPDELAY,MDELAY

      WRITE(*,13)
13    FORMAT(/,3X,' STEP ? ')
      READ(*,*)STEP
C     ------------------------------------------------
      CALL SLTPLT(CHOICE,nplant)
      write(*,14)num1,num2,num3,den1,den2,den3
14    format(/,3x,'num,den= ',6f10.7)

C     GENERATE PRBS
C     STORE 10 INITIAL VALUES INPUT
      DO 20 I=1,10
       X(I) = STEP
20    CONTINUE

      DO 60 J=1,1023
       Y0(J) = X(10)
C     SHIFT VALUES TO THE RIGHT
      DO 32 H=2,10
       X(12-H) = X(11-H)
```

```
32  CONTINUE
    TERM = STEP**2
    IF (Y0(J)*X(8).EQ.TERM) GOTO 30
     X(1) = STEP
     GOTO 40
30   X(1) = -STEP
40  CONTINUE
60  CONTINUE
    DO 62 S=1,L+3
     Y0(1023+S) = Y0(S)
62  CONTINUE
   DO 61 I = 1,L
     WK0(I) = 0.0
     WP(I)  = 0.0
61  CONTINUE

C   CALCULATE PRBS OUTPUT FROM A RC LOWPASS FILTER
C   BANDWIDTH = FC/2

    YY(1) = 0.0
    DO 163 KC=2,1023+L+3
     YY(KC) = 0.6*YY(KC-1)+0.2*(Y0(KC)*FCTR)+0.2*(Y0(KC-1)*FCTR)
163 CONTINUE

C     ---------------------------------------------

     MNUM4 = 0.0597*GM
     MNUM5 = 0.0
     MNUM6 = 0.0
     MDEN1 = -1.4891
     MDEN2 = 0.5488

10   write(10,18)mnum4,MNUM5,mden1,mden2
18   format(/,3x,'mnum,mden ',4f10.7)


    DO 66 ML = 1,L+3
     NOISE(ML) = RND(INIT)*NSEFTOR
     OUTPLT(ML)= NOISE(ML)
     IDEN(ML)  = NOISE(ML)
     TERM0(ML) = 0.0
     TERM1(ML) = 0.0
     TERM2(ML) = 0.0
     TERM3(ML) = 0.0
     YC(ML) = YY(ML) - OUTPLT(ML)
66  CONTINUE

    CALL WEIGHT
    WRITE(*,69)
69  FORMAT(/,3X,' PERIOD 1')
    WRITE(10,99)CHOICE,NPDELAY,MDELAY
99  FORMAT(/,3X, 'CHOICE ',I3,'NPDELAY,MDELAY ',2I3)
    WRITE(10,96)PERIOD,TS,U
96  FORMAT(/,3X,'PERIOD ',I3,'  TS ',F7.4,'  U ',F15.10)
    WRITE(10,95)STEP,FCTR,NSEFTOR
95  FORMAT(/,3X,' STEP ',I3,'FCTR,NSEFTOR ',2F10.8)

    DO 21 IJ = 1,107
      IJ1 = 1 + (iJ-1)*10
     INCNT(IJ) = TERM0(IJ1)
     HZIN(IJ)  = IJ * TS
21  CONTINUE

C   REPEAT LOOP
    DO 175 IA=2,PRD1
    DO  173   MA = 1,L+3
     TERM0(MA) = TERM0(1023+MA)
     TERM1(MA) = TERM1(1023+MA)
     TERM2(MA) = TERM2(1023+MA)
     TERM3(MA) = TERM3(1023+MA)
     YC(MA)    = YC(1023+MA)
     OUTPLT(MA)= OUTPLT(1023+MA)
```

```
      IDEN(MA) = IDEN(1023+MA)
      NOISE(MA) = NOISE(1023+MA)
      DF(MA)   = DF(1023+MA)
173  CONTINUE
     CALL WEIGHT
     WRITE(*,169)IA
169  FORMAT(/,3X,' PERIOD ',I3)
175  CONTINUE

     DO 22 IJ = 1,107
       IJ1 = 1 + (iJ-1)*10
       IJ2 = 107 + IJ
       INCNT(IJ2) = TERM0(IJ1)
       HZIN(IJ2)  = IJ2 * TS
22   CONTINUE

     GOTO 199
C    ------------------------------------------------
C    REPEAT LOOP
C    MEMORISE PLANT PARAMETERS
201    irem6 = idelay
       irem7 = mdelay
       IREM8 = CHOICE
       FREM9 = U
C    CHANGE PARAMETERS OF A PLANT
     WRITE(*,77)
77   FORMAT(/,3X,'NPDELAY?,MDELAY,choice ?,nplant? ')
     READ(*,*)NPDELAY,MDELAY,choice,nplant
     WRITE(*,78)
78   FORMAT(/,3X,'U ?')
     READ(*,*)U

     CALL SLTPLT(CHOICE,nplant)
     DO 186 IA=prd1+1,PRD2
      DO 174  MA = 1,L+3
       TERM0(MA) = TERM0(1023+MA)
       TERM1(MA) = TERM1(1023+MA)
       TERM2(MA) = TERM2(1023+MA)
       TERM3(MA) = TERM3(1023+MA)
       YC(MA)    = YC(1023+MA)
       OUTPLT(MA)= OUTPLT(1023+MA)
       IDEN(MA) = IDEN(1023+MA)
       NOISE(MA) = NOISE(1023+MA)
       DF(MA)   = DF(1023+MA)
174  CONTINUE
     CALL WEIGHT
     IF (IA .EQ. PRD1+1) THEN
     DO 23 IJ = 1,107
       IJ1 = 1 + (iJ-1)*10
       IJ2 = 214 + IJ
       INCNT(IJ2) = TERM0(IJ1)
       HZIN(IJ2)  = IJ2 * TS
23   CONTINUE
     ENDIF
     WRITE(*,171)IA
171  FORMAT(/,3X,' PERIOD ',I3)
186  CONTINUE
     DO 24 IJ = 1,107
       IJ1 = 1 + (iJ-1)*10
       IJ2 = 321 + IJ
       INCNT(IJ2) = TERM0(IJ1)
       HZIN(IJ2)  = IJ2 * TS
24   CONTINUE
     GOTO 200

C    REPEAT LOOP
C    PLANT CHANGES BACK TO ITS ORIGINAL PARAMETERS
202    idelay = irem6
       mdelay = irem7
       choice = irem8
       nplant = irem9
       U    = FREM9
```

```
      CALL SLTPLT(CHOICE,nplant)

     DO 75 IA=PRD2+1,PERIOD
     DO  73   MA = 1,L+3
      TERM0(MA) = TERM0(1023+MA)
      TERM1(MA) = TERM1(1023+MA)
      TERM2(MA) = TERM2(1023+MA)
      TERM3(MA) = TERM3(1023+MA)
      YC(MA)    = YC(1023+MA)
      OUTPLT(MA)= OUTPLT(1023+MA)
      IDEN(MA)  = IDEN(1023+MA)
      NOISE(MA) = NOISE(1023+MA)
      DF(MA)    = DF(1023+MA)
73   CONTINUE
      CALL WEIGHT

     IF ( IA .EQ. PRD2+1) THEN
     DO 25 IJ = 1,107
       IJ1 = 1 + (iJ-1)*10
       IJ2 = 428 + IJ
       INCNT(IJ2) = TERM0(IJ1)
       HZIN(IJ2)  = IJ2 * TS
25   CONTINUE
     ENDIF
71    WRITE(*,79)IA
79    FORMAT(/,3X,' PERIOD ',I3)
75    CONTINUE

      DO 26 IJ = 1,107
       IJ1 = 1 + (iJ-1)*10
       IJ2 = 535 + IJ
       INCNT(IJ2) = TERM0(IJ1)
       HZIN(IJ2)  = IJ2 * TS
26   CONTINUE

C    PLOT TAP
199   FNAME12 = 'TAP'
     CALL QOPEN(FNAME12)
     CALL QFRAME(1)
     TLE12 = ' INDIVIDUAL TAP PLOT'
     CALL QTEXT(TLE12,20,12000,31000)
     TED12 = .TRUE.
     CALL QRANGE(HZP,TAP1,103,TED12)
     CALL QRANGE(HZP,TAP11,103,TED12)
     CALL QRANGE(HZP,TAP21,103,TED12)
     CALL QRANGE(HZP,TAP31,103,TED12)
     HNAME = 'TIME IN SECONDS'
      VNAME = 'TAP1,6,11,21,31'
     CALL QXYAXES(HNAME,VNAME)
     CALL QPLOT(HZP,TAP1,103,1,1)
     CALL QPLOT(HZP,TAP11,103,1,1)
     CALL QPLOT(HZP,TAP21,103,1,1)
     CALL QPLOT(HZP,TAP31,103,1,1)
    READ(*,145)KEY12
145   FORMAT(A1)
     CALL QCLOSE
200   DO 132 III = 1,L
         UNIT(III)=III*TS
132   CONTINUE

     SUMTAPS = 0.0
     SUMP    = 0.0
     DO 135 II3 = 1,L
      SUMTAPS = SUMTAPS + WAV(II3)
      SUMP    = SUMP + WPAV(II3)
135   CONTINUE
     WRITE(10,94)SUMTAPS,SUMP
94    FORMAT(/,3X,'SUMTAPS,SUMP ',2F15 10)

C    PLOT TAPS' VALUES
     FNAME ='TAPS'
     CALL QOPEN(FNAME)
```

```
      CALL QFRAME(1)
      TITLE='VALUES OF TAPS SETTING'
      CALL QTEXT(TITLE,22,12000,31000)
      TED = .TRUE.
      CALL QRANGE(UNIT,WAV,40,TED)
      HZNAME = 'NO OF TAPS'
      VTNAME = 'TAPS SETTING'
      CALL QXYAXES(HZNAME,VTNAME)
      CALL QMARK(UNIT,WAV,40,1,2,1)
      READ(*,131)KEY
131   FORMAT(A1)
      CALL QCLOSE

C     PLOT TAPS' VALUES
      FNAMEP ='TAPSP'
      CALL QOPEN(FNAMEP)
      CALL QFRAME(1)
      TITP='VALUES OF PLANT TAPS '
      CALL QTEXT(TITP,22,12000,31000)
      TEDP = .TRUE.
      CALL QRANGE(UNIT,WPAV,40,TEDP)
      HZPNAME = 'NO OF TAPS'
      VTPNAME = 'TAPS SETTING'
      CALL QXYAXES(HZPNAME,VTPNAME)
      CALL QMARK(UNIT,WPAV,40,1,2,1)
      READ(*,170)KEYP
170   FORMAT(A1)
      CALL QCLOSE

C     %%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%
C     INITIALISE
      DO 263 KC=1,150
      IF (CHOICE .EQ. 1 ) THEN
        Y(KC) = exp(-0.4*(kc*ts))*sin(2*kc*ts)/GM
      ENDIF
      IF ((CHOICE .EQ. 2) .AND. (KC .LE.75)) THEN
       y(KC) = KC*FCTR/GM
      ENDIF
      IF ((CHOICE .EQ. 2) .AND. (KC .GT. 75)) THEN
        y(KC) = (2.0*Y(75))-KC*fctr/GM
      ENDIF
263   CONTINUE
      DO 264 I = 1,6
         DO 255 KC = 1,150
         Y(KC+I*150) = Y(KC)
255   CONTINUE
264   CONTINUE

      DO 256 I = 1,16
        Y(I+1050) = Y(I+134)
256   CONTINUE

C     FEEDBACK STEP RESPONSE
      DO 149 I =1 ,NPDELAY
       nst(I) = 0.0
       NST0(I)= 0.0
149   CONTINUE
      NST(NPDELAY+1) = NUM1*WAV(1)*Y(1)
      NINST1 = WAV(1)*Y(1)
      NINST  = WAV(1)*(Y(2)-NST(2)) + WAV(2)*(Y(1)-NST(1))
      NST(NPDELAY+2) = NUM1*NINST+NUM2*NINST1-DEN1*NST(NPDELAY+1)
     +           - DEN2*NST(NPDELAY)
      DO 151 I0=NPDELAY+3,500
       NINST2 = NINST1
       NINST1 = NINST
       NINST  = 0.0
       DO 164 I = 1,L
        IF ((I0 - I -NPDELAY + 1) .GT. 0) THEN
         NINST = NINST+WAV(I)*(Y(I0-I-NPDELAY+1)-NST(I0-I-NPDELAY+1))
        ELSE
         GOTO 164
        ENDIF
```

```
164   CONTINUE
      NST(I0) = NUM1*NINST+NUM2*NINST1+NUM3*NINST2
     +      - DEN1*NST(I0-1) - DEN2*NST(I0-2)
151  CONTINUE

C   MODEL
     DO 150 I = 1,MDELAY
       NSTM(I) = 0.0
150  CONTINUE
     NSTM(MDELAY+1) = MNUM4*Y(1)
     NSTM(MDELAY+2) = MNUM4*Y(2) + MNUM5*Y(1)-MDEN1*NSTM(MDELAY+1)
     DO 153 I2=MDELAY+3,500
       NSTM(I2) = MNUM4*Y(I2-MDELAY) + MNUM5*Y(I2-MDELAY-1)
     +          + MNUM6*Y(I2-MDELAY-2)
     +          - MDEN1*NSTM(I2-1)-MDEN2*NSTM(I2-2)
153  CONTINUE
     NST0(NPDELAY+1) = 0.0
     NST0(NPDELAY+2) = NUM2*y(1) - DEN1*NST0(NPDELAY-1)
     DO 155 I = NPDELAY+3 ,200
     NST0(I) = NUM2*y(I-2-NPDELAY+1) + NUM3*y(I-3-NPDELAY+1)
     +     - DEN1*NST0(I-1) - DEN2*NST0(I-2)
155  CONTINUE
     DO 154 I = 1,500
      HST(I) = I*TS
154  CONTINUE

C   PLOT OPEN LOOP STEP RESPONSE
C   FOR A CLOSED LOOP FEEDBACK CONTROL SYSTEM
     FNAMEST = 'STRESP'
     CALL QOPEN(FNAMEST)
     CALL QFRAME(1)
     TEDST = .TRUE.
     CALL QRANGE(HST,NST,500,TEDST)
     CALL QRANGE(HST,NSTM,500,TEDST)
     HZST = 'TIME IN SEC'
     VTST = 'OUTPUT WAVEFORMS'
     CALL QXYAXES(HZST,VTST)
     CALL QPLOT(HST,NST,500,1,1)
     CALL QPLOT(HST,NSTM,500,2,1)
     READ(*,147)KEYST
147  FORMAT(A1)
     CALL QCLOSE
      FNAMErf = 'out&rf'
     CALL QOPEN(FNAMErf)
     CALL QFRAME(1)
     TEDRF = .TRUE.
     CALL QRANGE(HST,NST,500,TEDRF)
     CALL QRANGE(HST,Y,500,TEDRF)
     HZRF = 'TIME IN SEC'
     VTRF = 'OUTPUT & REF-SIGNAL WAVEFORMS'
     CALL QXYAXES(HZRF,VTRF)
     CALL QPLOT(HST,NST,500,1,1)
     CALL QPLOT(HST,Y,500,2,1)
     READ(*,148)KEYRF
148  FORMAT(A1)
     CALL QCLOSE

     DO 286 IJ = 1,107
       IJ1 = 1 + (IJ-1)*10
       HZSQ(IJ)  = IJ1*TS
       OUTSQ(IJ)  = OUTPLT(IJ1)
       DFSQ(IJ)  = DF(IJ1)
286  CONTINUE

C   SQUARE RESPONSE OF A SYSTEM
     FSQ = 'SQRESP'
     CALL QOPEN(FSQ)
     CALL QFRAME(1)
     TEDSQ = .TRUE.
     CALL QRANGE(HZSQ,OUTSQ,107,TEDSQ)
     CALL QRANGE(HZSQ,DFSQ,107,TEDSQ)
     HZSQNAME = 'TIME IN SEC'
```

```
      VTSQ = 'SQUARE RESPONSE'
      CALL QXYAXES(HZSQNAME,VTSQ)
      CALL QPLOT(HZSQ,OUTSQ,107,1,1)
      CALL QPLOT(HZSQ,DFSQ,107,2,1)
      READ(*,290)KEYSQ
290   FORMAT(A1)
      CALL QCLOSE

      FCNTIN = 'CNTIN'
      CALL QOPEN(FCNTIN)
      CALL QFRAME(1)
      TEDCNT = .TRUE.
      CALL QRANGE(HZIN,INCNT,642,TEDCNT)
      HZCNT = 'TIME IN SEC'
      VTCNT = 'CONTROL INPUT SIGNAL'
      CALL QXYAXES(HZCNT,VTCNT)
      CALL QPLOT(HZIN,INCNT,642,1,1)
      READ(*,289)KEYCNT
289   FORMAT(A1)
      CALL QCLOSE

      WRITE(*,161)SUMTAPS
161   FORMAT(/,3X,'SUMTAPS = ',F15.5)
      IF (IA .LE. PRD2) GOTO 201
      IF (IA .LE. PERIOD) GOTO 202
      CLOSE(10)
      STOP
      END

C %%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%
C   CALCULATE WEIGHT FUNCTION USING L M S THEOREM

      SUBROUTINE WEIGHT
      INTEGER L,K,B,A,E,AA
      INTEGER B00,B11,B22,NPDELAY,MDELAY,INIT,IA,PERIOD
      REAL    YY(1066),WK0(40),NOISE(1066),NSEFTOR
      REAL U,W(40),YC(1066),OUTPLT(1066),IDEN(1066)
      REAL DF(1066),NUM1,NUM2,NUM3,MNUM4,MNUM5,MNUM6,WAV(40)
      REAL DEN1,DEN2,DEN3,HZ(1023)
      REAL MDEN1,MDEN2,WPAV(40),WP(40)
      REAL TERM0(1066),TERM1(1066),TERM2(1066),TERM3(1066)
      REAL    HZP(103),TAP1(103),TAP11(103),TAP21(103),TAP31(103)
      REAL TAP41(103),sumtaps

      COMMON/STORE1/K,NPDELAY
      COMMON/STORE2/TERM0,TERM1,TERM2,TERM3,OUTPLT
      COMMON/STORE3/L
      COMMON/STORE4/W,YC
      COMMON/STORE6/MDELAY,INIT,IA,PERIOD
      COMMON/STORE7/NOISE,WK0,WAV,IDEN,U,NSEFTOR,WPAV,WP,UP
      COMMON/STORE8/HZ,HZP,TAP1,TAP11,TAP21,TAP31,TAP41
      COMMON/STORE9/NUM1,NUM2,NUM3,DEN1,DEN2,DEN3,YY,GM,FCTR,DF,
     +  MNUM4,MNUM5,MNUM6,MDEN1,MDEN2,TS

C   CALCULATE WEIGHTS
        DO 20 I = 1,L
          WAV(I) = 0.0
          WPAV(I)= 0.0
20    CONTINUE

          DO 40 K=L+4,1023+L+3
      TERM3(K) = TERM2(K-1)
      TERM2(K) = TERM1(K-1)
      TERM1(K) = TERM0(K-1)
       NOISE(k) = RND(INIT)*NSEFTOR
      OUTPLT(K)= NUM1*TERM1(K-NPDELAY)+NUM2*TERM2(K-NPDELAY)
     +   +NUM3*TERM3(K-NPDELAY)
     +   -DEN1*OUTPLT(K-1)-DEN2*OUTPLT(K-2)-DEN3*OUTPLT(K-3)
     +   +NOISE(K)
        YC(K) = YY(K) - OUTPLT(K)
      DO 42 I = 1,L
42    W(I) = WK0(I)
```

```
      TERM0(K) = PLT(K)
      OUT = 0.0
      DO 12 I = K , K-L+1,-1
       OUT = OUT + (TERM0(I)*WP(K-I+1))
   12 CONTINUE
      ER = OUTPLT(K) - OUT
      DO 13 I = 1,L
       WP(I)  = WP(I) + 2*UP*ER*TERM0(K-I+1)
       WPAV(I) = WPAV(I) + (WP(I)/1023)
   13 CONTINUE
      IDEN(K) = 0.0
      DO 14 I = 1,L
       IDEN(K) = IDEN(K) + WP(I)*YC(K-I+1)
   14 CONTINUE
       DF(K) = MNUM4*YY(K-MDELAY) + MNUM5*YY(K-MDELAY-1)
      +       + MNUM6*YY(K-MDELAY-2)
      +       - MDEN1*DF(K-1) - MDEN2*DF(K-2)
       DO 30 B=1,L
        WK0(B)=WK0(B) + 2*U*(DF(K)-OUTPLT(K))*IDEN(K-B+1)
         WAV(B)=WAV(B) + ( WK0(B)/1023)
   30  CONTINUE
      IF (MOD(K-43,10) .EQ. 0) THEN
       I = ( (K-43)/10 ) + 1
       TAP1(I) = WK0(1)
       TAP11(I)= WK0(11)
       TAP21(I)= WK0(21)
       TAP31(I)= WK0(31)
       TAP41(I)= WK0(41)
       HZP(I)  = (K-43) * TS
      ENDIF
       HZ(K-43) = (K-43)*TS
   40 CONTINUE
      RETURN
      END

      SUBROUTINE SLTPLT(I,nplant)
      REAL NUM1,NUM2,NUM3,DEN1,DEN2,DEN3,GP,YY(1066),GM,FCTR
      REAL DF(1066),MNUM4,MNUM5,MNUM6,MDEN1,MDEN2,TS
      INTEGER MDELAY,INIT,IA,PERIOD

      COMMON/STORE6/MDELAY,INIT,IA,PERIOD
      COMMON/STORE9/NUM1,NUM2,NUM3,DEN1,DEN2,DEN3,YY,GM,FCTR,DF,
     +   MNUM4,MNUM5,MNUM6,MDEN1,MDEN2,TS
      if (nplant .eq. 1) then
       num1 = 0.0
       num2 = 0.0601
       num3 = -0.1012
       den1 = -1.6457
       den2 = 0.6703
       den3 = 0.0
      else
       NUM1 = 0.0
       NUM2 = 2.4
       NUM3 = -0.8*2.4
       DEN1 = -0.1
       DEN2 = -0.42
       DEN3 = 0.0
      endif
      DO 17 II = 1,MDELAY
       DF(II) = 0.0
   17 CONTINUE
      DF(MDELAY+1) = MNUM4*YY(1)
      DF(MDELAY+2) = MNUM4*YY(2) + MNUM5*YY(1) - MDEN1*DF(MDELAY+1)
      DO 16 II = MDELAY+3 , L+3
       DF(II) = MNUM4*YY(II-MDELAY) + MNUM5*YY(II-MDELAY-1)
      +       + MNUM6*YY(II-MDELAY-2)
      +       - MDEN1*DF(II-1) - MDEN2*DF(II-2)
   16 CONTINUE
      RETURN
      END
```

```
C   CALCULATE TAP OUTPUT VALUES
    REAL FUNCTION PLT(K)
    INTEGER A,L
    REAL   OUT,W(40),YC(1066)

    COMMON/STORE3/L
    COMMON/STORE4/W,YC

    OUT = 0.0
    DO 20 A=K,K-L+1,-1
20     OUT = OUT + (YC(A)*W(K-A+1))
    PLT = OUT
    RETURN
    END

C %%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%
C   GENERATE NOISE
    REAL FUNCTION RND(IS)
    IS = MOD(193*IS,37447)
    RND= ((IS/37447.0)-0.5)
    RETURN
    END
```

APPENDIX C

The floating point addition routine which is shown in the "Floating-Point Arithmetic with the TMS32020" booklet cannot add two numbers, whose exponents's difference by more than >F [1], when it is written in the binary floating-point as discussed in chapter 5.

For example, to add a= 0.00000059 with b=0.10222959 , if using the floating point addition routine which is written in the above booklet , the result c of the addition routine will be 0.14078356. In this example , it occurs that the 4-bit values of the floating point a is ASIGN = >0000 , AEXP = >FFEC , AHI= >4EF3 , ALO = >5200 and the 4-bit values of the floating point b is BSIGN = >0000 , BEXP = >FFFE , BHI = >3457 , BLO = >6FC0. The TMS32020 addition routine gives the result c of CSIGN = >0000 , CEXP = >FFFE , CHI = >4814 and CLO = >C880. To avoid this faulty, the floating point addition routine was rewritten and the listing is attached in the appendix D.

The floating point multiplication routine which is provided from the "Floating-Point Arithmetic with the TMS32020" booklet is also need to modify; since it has no detection on overflow calculation. The floating point multiplication was modified and its listing is attached in the appendix D

1. > means hexadecimal ; >F=15 in decimal

APPENDIX D

This appendix contains the TMS320C20 program for testing the closed loop Model Reference Adaptive Control hybrid simulation.

This program was written for the closed loop Non-Parametric Model Reference Adaptive Control hybrid simulations (chapter 5).

```
NO$IDT     32020 FAMILY MACRO ASSEMBLER  PC 1.1 86 036    15:12:22  06-20-88
                                        PAGE 0001

0001 0000
0002 0000
0003 0000
0004 0000
0005 0000
0006 0060         AORG >60
0007 0060
0008 0060
0009          *DPM
0010 0060
0011 0060 0060  ASIGN   DATA  >60
0012 0061 0061  AEXP    DATA   >61
0013 0062 0062  AHI     DATA   >62
0014 0063 0063  ALO     DATA   >63
0015 0064 0064  BSIGN   DATA   >64
0016 0065 0065  BEXP    DATA   >65
0017 0066 0066  BHI     DATA   >66
0018 0067 0067  BLO     DATA   >67
0019 0068 0068  CSIGN   DATA   >68
0020 0069 0069  CEXP    DATA   >69
0021 006A 006A  CHI     DATA   >6A
0022 006B 006B  CLO     DATA   >6B
0023 006C 006C  D DATA    >6C
0024 006D 006D  ONE     DATA   >6D
0025 006E 006E  TEMP    DATA   >6E
0026 006F 006F  THREE   DATA   >6F
0027 0070 0070  SIXT    DATA   >70
0028 0071 0071  RESID   DATA   >71
0029 0072 0072  TTEEN   DATA   >72
0030 0073 0073  THI     DATA   >73
0031 0074 0074  NEGONE  DATA   >74
```

```
0032 0075 0075 TLO    DATA    >75
0033 0076
0034                                  *u = 0.004
0035 0076 0000 U      DATA  >0000
0036 0077 FFF9 U2     DATA    >FFF9
0037 0078 4189 U3     DATA    >4189
0038 0079 1B80 U4     DATA    >1B80
0039 007A
0040 007A
0041 007A 0000 NEGD     DATA >0
0042 007B 61A8 PRDTM   DATA >61A8    Ts=0.02sec.
0043 007C 0000 FIXPT    DATA 0
0044 007D FFFF  NEGS     DATA >FFFF
0045
0046 0200              AORG >200
0047 0200 FF80    B   START
   0201 0250
0048 0202
0049 0202 0DA8 STORE1  DATA >0DA8    starting adress of y(40)
0050 0203 02A8 STORE2  DATA >02A8    0D9C-029C = >B00
                                      starting address of taps
0051 0204 0000 STORE3  DATA 0
0052 0205 0B00  STORE4   DATA >0B00     starting address of taps
0053 0206 0DA8  STORE5  DATA >0DA8    memorized store1
0054 0207 02A8  STORE6  DATA >02A8    memorized store2
0055 0208 0028 NOTAPS  DATA 40
0056 0209 0027 CNTAPS  DATA   39

NO$IDT    32020 FAMILY MACRO ASSEMBLER  PC 1.1 86.036    15:12:22  06-20-88
                                  PAGE 0002

0057 020A 03FF  COUNT   DATA 1023
0058 020B 0000  CNST2  DATA >0000          number 2
0059 020C 0002  CNST22  DATA >0002
0060 020D 4000  CNST23  DATA >4000
0061 020E 0000  CNST24   DATA >0000
0062 020F
0063 020F 0000 CNST1     DATA >0000          0.01
0064 0210 FFFA  CNST12   DATA >FFFA
0065 0211 51EB  CNST13   DATA >51EB
0066 0212 4280  CNST14   DATA >4280
0067 0213
0068 0213 0000  REMAR2   DATA 0
0069 0214 0000  STORE7    DATA 0
0070 0215 0B00  STRE44   DATA >0B00
0071 0216 0A00  WPADDR  DATA >A00
0072 0217
0073
0074 0217
0075 0217 0005  PRD    DATA 5
0076 0218 0000  REMAR0  DATA 0
0077 0219
0078                                  *GM = 0.3
0079 0219 0000  MNUM5   DATA >0000
0080 021A FFFC  MNUM52  DATA >FFFC
0081 021B 7AE1  MNUM53  DATA >7AE1
0082 021C 23C0  MNUM54  DATA >23C0
0083 021D
0084 021D
0085 021D 0000  MNUM6   DATTA >0000
0086 021E 0000  MNUM62  DATA  >0000
0087 021F 0000  MNUM63  DATA  >0000
0088 0220 0000  MNUM64  DATA >0000
0089 0221
0090 0221 0000  MDEN1   DATA >0000
0091 0222 0000  MDEN12  DATA >0000
0092 0223 6666  MDEN13  DATA >6666
0093 0224 3300  MDEN14  DATA >3300
0094 0225
0095 0225 0000  MDEN2   DATA >0000
0096 0226 0000  MDEN22  DATA >0000
0097 0227 0000  MDEN23  DATA >0000
```

```
0098 0228 0000  MDEN24  DATA >0000
0099 0229
0100 0229 0004  CNST4   DATA 4
0101 022A 0000  MEM     DATA   >0
0102 022B 0000  MEM2    DATA >0
0103 022C 0000  MEM3    DATA >0
0104 022D 0000  MEM4    DATA >0
0105 022E
0106                                          *up = 0.004
0107 022E 0000  UP DATA >0000
0108 022F FFF9  UP2     DATA >FFF9
0109 0230 4189  UP3     DATA >4189
0110 0231 1B80  UP4     DATA >1B80
0111 0232
0112 0232 0000  OUT1    DATA >0000
0113 0233 0000  OUT2    DATA >0000

NO$IDT    32020 FAMILY MACRO ASSEMBLER  PC 1.1 86.036    15:12:22  06-20-88
                                        PAGE 0003

0114 0234 0000  OUT3    DATA >0000
0115 0235 0000  OUT4    DATA >0000
0116 0236
0117 0236 09F4  CNTLST  DATA >9F4
0118 0237 0000  UKB     DATA 0
0119 0238 0A00  PTAPS   DATA >A00
0120 0239
0121 0239 0000  DK21    DATA >0
0122 023A 0000  DK22    DATA >0
0123 023B 0000  DK23    DATA >0
0124 023C 0000  DK24    DATA >0
0125 023D
0126 023D 0000  DK11    DATA >0
0127 023E 0000  DK12    DATA >0
0128 023F 0000  DK13    DATA >0
0129 0240 0000  DK14    DATA >0
0130 0241
0131 0241 0000  DK1     DATA >0
0132 0242 0000  DK2     DATA >0
0133 0243 0000  DK3     DATA >0
0134 0244 0000  DK4     DATA >0
0135 0245
0136 0245 0CA4  LST     DATA >CA4
0137 0246 1EA4  CLMEM   DATA >1EA4
0138 0247 0000  FKB     DATA 0
0139 0248
0140 0248 0000  NO2UE    DATA 0
0141 0249 0000  NO2UE2   DATA 0
0142 024A 0000  NO2UE3   DATA 0
0143 024B 0000  NO2UE4   DATA 0
0144 024C
0145 024C 0000  P2UE    DATA 0
0146 024D 0000  P2UE2   DATA 0
0147 024E 0000  P2UE3   DATA 0
0148 024F 0000  P2UE4   DATA 0
0149 0250
0150                                          *initialise AIB
0151 0250 CE04   START   CNFD
0152 0251 CAFF    LACK >FF
0153 0252 6000    SACL 0
0154 0253 E300    OUT  0,3
0155 0254 E100    OUT    0,1
0156 0255
0157 0255 CAFE    LACK >FE
0158 0256 6000    SACL 0
0159 0257 E000    OUT  0,0
0160 0258
0161                                          *move data from pm to dm
0162 0258
0163 0258 5588    LARP AR0
0164 0259 D000    LRLK AR0,>76
   025A 0076
```

```
0165 025B CB07    RPTK >7
0166 025C FCA0    BLKP >76,*+
   025D 0076
0167 025E D000    LRLK AR0,>202
   025F 0202
```



```
0168 0260 CB4D    RPTK 77
0169 0261 FCA0    BLKP >202,*+
   0262 0202
0170 0263
0171                          *set up for interrupt service routine
0172 0263
0173 0263 207B    LAC  PRDTM
0174 0264 6003    SACL >03
0175 0265 6002    SACL >02
0176 0266 CA08    LACK 8
0177 0267 4D04    OR   >04
0178 0268 6004    SACL >04
0179 0269 CE00    EINT
0180 026A
0181                          *clear prbs(-2),prbs(-1)
0182 026A D000    LRLK AR0,>CF8
   026B 0CF8
0183 026C CA00    ZAC
0184 026D CB07    RPTK 7
0185 026E 60A0    SACL *+,0
0186 026F
0187                          *clear f(k-41)..f(k)
0188 026F D000    LRLK AR0,>C00
   0270 0C00
0189 0271 CA00    ZAC
0190 0272 CBA7    RPTK 167
0191 0273 60A0    SACL *+,0
0192 0274
0193                          *clear u(k-41)..u(k)
0194 0274 D000    LRLK AR0,>950
   0275 0950
0195 0276 CA00    ZAC
0196 0277 CBA7    RPTK 167
0197 0278 60A0    SACL *+,0
0198 0279
0199                          *clear c(K-41)..c(K)
0200 0279 D000    LRLK AR0,>1E00
   027A 1E00
0201 027B CA00    ZAC
0202 027C CBA7    RPTK 167
0203 027D 60A0    SACL *+,0
0204 027E
0205                          *clear 40 controller taps
0206 027E D000    LRLK AR0,>B00
   027F 0B00
0207 0280 CA00    ZAC
0208 0281 CB9F    RPTK 159
0209 0282 60A0    SACL *+,0
0210 0283
0211                          *set 0.01 for the first plant
0212 0283 558B    LARP 3
0213 0284 D300    LRLK AR3,CNST1
   0285 020F
0214 0286 D400    LRLK AR4,>A00
   0287 0A00
0215 0288 20AC    LAC  *+,0,AR4
0216 0289 60AB    SACL *+,0,AR3
```



```
0217 028A 20AC    LAC  *+,0,AR4
0218 028B 60AB    SACL *+,0,AR3
```

```
0219 028C 20AC    LAC  *+,0,AR4
0220 028D 60AB    SACL *+,0,AR3
0221 028E 20AC    LAC  *+,0,AR4
0222 028F 60A8    SACL *+,0,AR0
0223 0290
0224 0290
0225                                          *set 0.01 for 39 plant's taps
0226 0290 D000    LRLK AR0,>A04
   0291 0A04
0227 0292 CB9B    RPTK 155
0228 0293 FDA0    BLKD >A00,*+
   0294 0A00
0229 0295
0230 0295 558B    LARP 3
0231 0296 D300    LRLK AR3,PRD
   0297 0217
0232 0298 3080    LAR  AR0,*
0233 0299
0234 0299 D300  AGAIN  LRLK AR3,REMAR0
   029A 0218
0235 029B 7080    SAR  AR0,*
0236 029C
0237 029C 558C  PRDIN  LARP 4
0238 029D D400    LRLK AR4,CNTAPS
   029E 0209
0239 029F 3280    LAR  AR2,*                    * = AR4
0240 02A0
0241 02A0
0242 02A0 FE80    CALL C2UE
   02A1 02E6
0243 02A2 FE80    CALL   PIDTFY
   02A3 05D1
0244 02A4 FE80    CALL PREFTR
   02A5 0441
0245 02A6
0246 02A6 FE80  WEIGHT  CALL CALP
   02A7 03E8
0247                                          * count for number of taps
0248 02A8 558A    LARP 2
0249 02A9 FB90    BANZ WEIGHT,*-
   02AA 02A6
0250 02AB
0251                                          * re-initialise store3 as zero
0252 02AB D200    LRLK AR2,STORE3
   02AC 0204
0253 02AD CA00    ZAC
0254 02AE 60A0    SACL *+,0
0255 02AF
0256                                        * re-initialise store4 as >b00
0257                                        * BLKD >211,*+
0258 02AF
0259 02AF D400    LRLK AR4,>B00
   02B0 0B00
0260 02B1 74A0    SAR  AR4,*+                  *=AR2;(AR2)=STORE4
0261 02B2

NO$IDT    32020 FAMILY MACRO ASSEMBLER  PC 1.1 86.036   15:12:22  06-20-88
                                  PAGE 0006

0262                                        * increment k by 1
0263 02B2 2080    LAC  *,0
0264 02B3 D002    ADLK 4
   02B4 0004
0265 02B5 60AC    SACL *+,0,AR4
0266 02B6 D400    LRLK AR4,STORE1
   02B7 0202
0267 02B8 6080    SACL *,0
0268 02B9
0269                                        * count for a period
0270 02B9 D400    LRLK AR4,COUNT
   02BA 020A
0271 02BB 3189    LAR  AR1,*,AR1
```

```
0272 02BC 559C    MAR  *-,AR4
0273 02BD 7189    SAR  AR1,*,AR1
0274 02BE
0275 02BE
0276 02BE FB8B    BANZ PRDIN,*,AR3
   02BF 029C
0277 02C0
0278 02C0 D300    LRLK AR3,STORE1
   02C1 0202
0279 02C2 CB15    RPTK 21
0280 02C3 FCA0    BLKP >202,*+
   02C4 0202
0281 02C5 D300    LRLK AR3,REMAR0
   02C6 0218
0282 02C7 3088    LAR  AR0,*,AR0
0283 02C8 FB9B    BANZ AGAIN,*-,AR3
   02C9 0299
0284 02CA
0285 02CA 558C    LARP 4
0286 02CB C127    LARK AR1,39
0287 02CC D200    LRLK AR2,>BA0
   02CD 0BA0
0288 02CE D400    LRLK AR4,>B00
   02CF 0B00
0289 02D0
0290 02D0 55A0 NEXT   MAR  *+
0291 02D1 2080    LAC  *
0292 02D2 D004    ANDK >8000                    test a msb
   02D3 8000
0293 02D4 F680    BZ   UNCH
   02D5 02DF
0294 02D6 20A0    LAC  *+                       if negative take 2nd compl
0295 02D7 CE27    CMPL
0296 02D8 6060    SACL >60
0297 02D9 20AA    LAC  *+,0,AR2
0298 02DA 4B60    RPT  >60
0299 02DB CE19    SFR
0300 02DC 60AC    SACL *+,0,AR4
0301 02DD FFA9    B    FTAPS,*+,AR1
   02DE 02E3
0302 02DF
0303                        *for exp = 0 only
0304 02DF 55A0 UNCH   MAR  *+
0305 02E0 20AA    LAC  *+,0,AR2

NO$IDT    32020 FAMILY MACRO ASSEMBLER  PC 1.1 86.036    15:12:22  06-20-88
                                  PAGE 0007

0306 02E1 60AC    SACL *+,0,AR4
0307 02E2 55A9    MAR  *+,AR1
0308 02E3 FB9C FTAPS  BANZ   NEXT,*-,AR4
   02E4 02D0
0309 02E5
0310 02E5 5500    NOP
0311 02E6



0312                        *CAL PROCEDURE
0313 02E6
0314 02E6
0315                  *we need to use ar2 in this procedure while (ar2)
                           must not b
0316                  *altered in a main body . So we keep its value in
                           memory loc
0317                  *remar2

0318 02E6 D400    C2UE   LRLK AR4,REMAR2
   02E7 0213
0319 02E8 558C    LARP 4
0320 02E9 7280    SAR  AR2,*
0321 02EA
```

```
0322 02EA D400    LRLK   AR4,CNTAPS
   02EB 0209
0323 02EC 3280    LAR  AR2,*
0324 02ED D400    LRLK AR4,>950
   02EE 0950
0325 02EF CBA3    RPTK 163
0326 02F0 FDA0    BLKD >954,*+
   02F1 0954
0327 02F2
0328                                        * clear >9F4
0329 02F2 D400    LRLK AR4,>9F4
   02F3 09F4
0330 02F4 CA00    ZAC
0331 02F5 CB03    RPTK 3
0332 02F6 60A0    SACL *+
0333 02F7
0334                                        * update x(k)..x(k-40)
0335 02F7 D400    LRLK AR4,>1E00
   02F8 1E00
0336 02F9 CBA3    RPTK 163
0337 02FA FDA0    BLKD >1E04,*+
   02FB 1E04
0338 02FC
0339                                        * initial y(a-1)
0340                                        * read contents in extended memory
                                              address
0341                                        * pointed by ar3 and store in asign
0342 02FC D400 OUTL   LRLK AR4,STORE1
   02FD 0202
0343 02FE 338B    LAR  AR3,*,AR3
0344 02FF C460    LARK AR4,ASIGN
0345 0300 20AC    LAC  *+,0,AR4
0346 0301 60AB    SACL *+,0,AR3
0347 0302 20AC    LAC  *+,0,AR4
0348 0303 60AB    SACL *+,0,AR3
0349 0304 20AC    LAC  *+,0,AR4
0350 0305 60AB    SACL *+,0,AR3
0351 0306 20AC    LAC  *+,0,AR4
0352 0307 60A0    SACL *+,0
0353 0308

NO$IDT    32020 FAMILY MACRO ASSEMBLER  PC 1.1 86.036    15:12:22  06-20-88
                                    PAGE 0008

0354 0308
0355 0308 D400    LRLK AR4,CLMEM
   0309 0246
0356 030A 338B    LAR  AR3,*,AR3
0357 030B C464    LARK AR4,BSIGN
0358 030C 20AC    LAC  *+,0,AR4
0359 030D 60AB    SACL *+,0,AR3
0360 030E 20AC    LAC  *+,0,AR4
0361 030F 60AB    SACL *+,0,AR3
0362 0310 20AC    LAC  *+,0,AR4
0363 0311 60AB    SACL *+,0,AR3
0364 0312 20AC    LAC  *+,0,AR4
0365 0313 60AB    SACL *+,0,AR3
0366 0314
0367 0314 FE80    CALL FNEG
   0315 0715
0368 0316 FE80    CALL FADD
   0317 0719
0369 0318
0370                                        *transfer (c) to (a)
0371 0318 558B    LARP 3
0372 0319 C460    LARK AR4,ASIGN
0373 031A C368    LARK AR3,CSIGN
0374 031B 20AC    LAC  *+,0,AR4
0375 031C 60AB    SACL *+,0,AR3
0376 031D 20AC    LAC  *+,0,AR4
0377 031E 60AB    SACL *+,0,AR3
0378 031F 20AC    LAC  *+,0,AR4
```

```
0379 0320 60AB    SACL *+,0,AR3
0380 0321 20AC    LAC  *+,0,AR4
0381 0322 CE19    SFR
0382 0323 D004    ANDK >7FC0
   0324 7FC0
0383 0325 60A0    SACL *+,0
0384 0326
0385                                          * W(K-A+1)
0386 0326 D400    LRLK AR4,STRE44
   0327 0215
0387 0328 338B    LAR  AR3,*,AR3
0388 0329 C464    LARK AR4,BSIGN
0389 032A 20AC    LAC  *+,0,AR4
0390 032B 60AB    SACL *+,0,AR3
0391 032C 20AC    LAC  *+,0,AR4
0392 032D 60AB    SACL *+,0,AR3
0393 032E 20AC    LAC  *+,0,AR4
0394 032F 60AB    SACL *+,0,AR3
0395 0330 20AC    LAC  *+,0,AR4
0396 0331 60AB    SACL *+,0,AR3
0397 0332
0398                                          * Y(A-1)*W(K-A+1)
0399 0332 FE80    CALL FMULT
   0333 07DB
0400 0334
0401 0334 558B    LARP 3
0402 0335 C460    LARK AR4,ASIGN
0403 0336 C368    LARK AR3,CSIGN
0404 0337 20AC    LAC  *+,0,AR4

NO$IDT    32020 FAMILY MACRO ASSEMBLER  PC 1.1 86 036    15:12:22  06-20-88
                                PAGE 0009

0405 0338 60AB    SACL *+,0,AR3
0406 0339 20AC    LAC  *+,0,AR4
0407 033A 60AB    SACL *+,0,AR3
0408 033B 20AC    LAC  *+,0,AR4
0409 033C 60AB    SACL *+,0,AR3
0410 033D 20AC    LAC  *+,0,AR4
0411 033E CE19    SFR
0412 033F D004    ANDK >7FC0
   0340 7FC0
0413 0341 60AB    SACL *+,0,AR3
0414 0342
0415                                          * OUT1+(Y(A-1)*W(K-A+1))
0416 0342 D300    LRLK AR3,>9F4
   0343 09F4
0417 0344 20AC    LAC  *+,0,AR4
0418 0345 60AB    SACL *+,0,AR3
0419 0346 20AC    LAC  *+,0,AR4
0420 0347 60AB    SACL *+,0,AR3
0421 0348 20AC    LAC  *+,0,AR4
0422 0349 60AB    SACL *+,0,AR3
0423 034A 20AC    LAC  *+,0,AR4
0424 034B 60AB    SACL *+,0,AR3
0425 034C FE80    CALL FADD
   034D 0719
0426 034E 558B    LARP 3
0427 034F
0428                                          * store results in >9F4
0429 034F D400    LRLK AR4,>9F4
   0350 09F4
0430 0351 C368    LARK AR3,CSIGN
0431 0352 20AC    LAC  *+,0,AR4
0432 0353 60AB    SACL *+,0,AR3
0433 0354 20AC    LAC  *+,0,AR4
0434 0355 60AB    SACL *+,0,AR3
0435 0356 20AC    LAC  *+,0,AR4
0436 0357 60AB    SACL *+,0,AR3
0437 0358 20AC    LAC  *+,0,AR4
0438 0359 CE19    SFR
0439 035A D004    ANDK >7FC0
```

```
   035B 7FC0
0440 035C 60A0    SACL *+,0
0441 035D
0442                                        * reduced a by 1
0443 035D D400    LRLK AR4,STORE1
   035E 0202
0444 035F 2080    LAC  *,0
0445 0360 D003    SBLK 4
   0361 0004
0446 0362 6080    SACL *,0
0447 0363
0448                                        * reduced outk by 1
0449 0363 D400    LRLK AR4,CLMEM
   0364 0246
0450 0365 2080    LAC  *,0
0451 0366 D003    SBLK 4
   0367 0004
0452 0368 6080    SACL *,0

NO$IDT    32020 FAMILY MACRO ASSEMBLER  PC 1.1 86.036   15:12:22  06-20-88
                                   PAGE 0010

0453 0369
0454                                        * increment weight by 1
0455 0369 D400    LRLK AR4,STRE44
   036A 0215
0456 036B 2080    LAC  *,0
0457 036C D002    ADLK 4
   036D 0004
0458 036E 6080    SACL *
0459 036F
0460                                        * count for a loop
0461 036F 558A    LARP 2
0462 0370 FB9C    BANZ OUTL,*-,AR4
   0371 02FC
0463 0372
0464 0372
0465                                        * re-initialise
0466 0372 D300    LRLK AR3,>B00
   0373 0B00
0467 0374 7380    SAR  AR3,*
0468 0375
0469 0375
0470                                        * re-initialise STORE1
0471 0375 D400    LRLK AR4,STORE1
   0376 0202
0472 0377 FD80    BLKD >206,*
   0378 0206
0473 0379
0474 0379
0475                                        * re-initialise clmem
0476 0379 D400    LRLK AR4,CLMEM
   037A 0246
0477 037B D300    LRLK AR3,>1EA4
   037C 1EA4
0478 037D 7380    SAR  AR3,*
0479 037E
0480                                        * D(k)
0481 037E FE80    CALL MODEL
   037F 04D8
0482 0380
0483 0380 558B    LARP 3
0484 0381 D300    LRLK AR3,DK1
   0382 0241
0485 0383 C460    LARK AR4,ASIGN
0486 0384 20AC    LAC  *+,0,AR4
0487 0385 60AB    SACL *+,0,AR3
0488 0386 20AC    LAC  *+,0,AR4
0489 0387 60AB    SACL *+,0,AR3
0490 0388 20AC    LAC  *+,0,AR4
0491 0389 60AB    SACL *+,0,AR3
0492 038A 20AC    LAC  *+,0,AR4
```

```
0493 038B 60AB    SACL *+,0,AR3
0494 038C
0495                                        * out1(k)
0496 038C D300    LRLK AR3,>9F4
   038D 09F4
0497 038E FE80    CALL FLFX
   038F 0812
```



```
0498 0390 CE1F    IDLE
0499 0391 FE80    CALL FXFL                 *read plant output & store in B
   0392 0836
0500 0393
0501                                        * send B to >1ea4
0502 0393 558B    LARP 3
0503 0394 D400    LRLK AR4,>1EA4
   0395 1EA4
0504 0396 D300    LRLK AR3,BSIGN
   0397 0064
0505 0398 20AC     LAC  *+,0,AR4
0506 0399 60AB     SACL *+,0,AR3
0507 039A 20AC     LAC  *+,0,AR4
0508 039B 60AB     SACL *+,0,AR3
0509 039C 20AC     LAC  *+,0,AR4
0510 039D 60AB     SACL *+,0,AR3
0511 039E 20AC     LAC  *+,0,AR4
0512 039F 60AB     SACL *+,0,AR3
0513 03A0
0514                                        * ER(k)
0515 03A0 FE80    CALL   FNEG
   03A1 0715
0516 03A2 FE80    CALL FADD
   03A3 0719
0517 03A4
0518 03A4 558B    LARP 3
0519 03A5 C460    LARK AR4,ASIGN
0520 03A6 C368    LARK AR3,CSIGN
0521 03A7 20AC     LAC  *+,0,AR4
0522 03A8 60AB     SACL *+,0,AR3
0523 03A9 20AC     LAC  *+,0,AR4
0524 03AA 60AB     SACL *+,0,AR3
0525 03AB 20AC     LAC  *+,0,AR4
0526 03AC 60AB     SACL *+,0,AR3
0527 03AD 20AC     LAC  *+,0,AR4
0528 03AE CE19    SFR
0529 03AF D004    ANDK >7FC0
   03B0 7FC0
0530 03B1 60AB     SACL *+,0,AR3
0531 03B2
0532 03B2 C376    LARK AR3,U
0533 03B3 20AC     LAC  *+,0,AR4
0534 03B4 60AB     SACL *+,0,AR3
0535 03B5 20AC     LAC  *+,0,AR4
0536 03B6 60AB     SACL *+,0,AR3
0537 03B7 20AC     LAC  *+,0,AR4
0538 03B8 60AB     SACL *+,0,AR3
0539 03B9 20AC     LAC  *+,0,AR4
0540 03BA 60AB     SACL *+,0,AR3
0541 03BB
0542 03BB FE80    CALL FMULT
   03BC 07DB
0543 03BD 558B    LARP 3
0544 03BE C460    LARK AR4,ASIGN
0545 03BF C368    LARK AR3,CSIGN
0546 03C0 20AC     LAC  *+,0,AR4
0547 03C1 60AB     SACL *+,0,AR3
```

```
0548 03C2 20AC    LAC  *+,0,AR4
0549 03C3 60AB    SACL *+,0,AR3
0550 03C4 20AC    LAC  *+,0,AR4
0551 03C5 60AB    SACL *+,0,AR3
0552 03C6 20AC    LAC  *+,0,AR4
0553 03C7 CE19    SFR
0554 03C8 D004    ANDK >7FC0
   03C9 7FC0
0555 03CA 60AB    SACL *+,0,AR3
0556 03CB
0557                                        * move constant 2 to Bs'
0558 03CB D300    LRLK AR3,CNST2
   03CC 020B
0559 03CD C464    LARK AR4,BSIGN
0560 03CE 20AC    LAC  *+,0,AR4
0561 03CF 60AB    SACL *+,0,AR3
0562 03D0 20AC    LAC  *+,0,AR4
0563 03D1 60AB    SACL *+,0,AR3
0564 03D2 20AC    LAC  *+,0,AR4
0565 03D3 60AB    SACL *+,0,AR3
0566 03D4 20AC    LAC  *+,0,AR4
0567 03D5 60AB    SACL *+,0,AR3
0568 03D6
0569 03D6 FE80    CALL FMULT
   03D7 07DB
0570 03D8
0571 03D8 558B    LARP 3
0572 03D9 D400    LRLK AR4,NO2UE
   03DA 0248
0573 03DB C368    LARK AR3,CSIGN
0574 03DC 20AC    LAC  *+,0,AR4
0575 03DD 60AB    SACL *+,0,AR3
0576 03DE 20AC    LAC  *+,0,AR4
0577 03DF 60AB    SACL *+,0,AR3
0578 03E0 20AC    LAC  *+,0,AR4
0579 03E1 60AB    SACL *+,0,AR3
0580 03E2 20AC    LAC  *+,0,AR4
0581 03E3 CE19    SFR
0582 03E4 D004    ANDK >7FC0
   03E5 7FC0
0583 03E6 60A0    SACL *+,0
0584 03E7
0585 03E7 CE26    RET
0586 03E8
0587 03E8
0588          * F(K-B)
0589          * B = 1 INITIALLY
0590 03E8 558C CALP  LARP 4
0591 03E9 D400    LRLK AR4,LST
   03EA 0245
0592 03EB 208B    LAC  *,0,AR3
0593 03EC D300    LRLK AR3,STORE3
   03ED 0204
0594 03EE 1080    SUB  *,0
0595 03EF D300    LRLK AR3,FKB
   03F0 0247
0596 03F1 608C    SACL *,0,AR4

NO$IDT    32020 FAMILY MACRO ASSEMBLER  PC 1.1 86.036    15:12:22  06-20-88
                                        PAGE 0013

0597 03F2
0598 03F2 D400    LRLK AR4,FKB
   03F3 0247
0599 03F4 338B    LAR  AR3,*,AR3
0600 03F5 C460    LARK AR4,ASIGN
0601 03F6 20AC    LAC  *+,0,AR4
0602 03F7 60AB    SACL *+,0,AR3
0603 03F8 20AC    LAC  *+,0,AR4
0604 03F9 60AB    SACL *+,0,AR3
0605 03FA 20AC    LAC  *+,0,AR4
0606 03FB 60AB    SACL *+,0,AR3
```

```
0607 03FC 20AC    LAC  *+,0,AR4
0608 03FD 60AB    SACL *+,0,AR3
0609 03FE
0610                                        * no2ue
0611 03FE D300    LRLK AR3,NO2UE
   03FF 0248
0612 0400 20AC    LAC  *+,0,AR4
0613 0401 60AB    SACL *+,0,AR3
0614 0402 20AC    LAC  *+,0,AR4
0615 0403 60AB    SACL *+,0,AR3
0616 0404 20AC    LAC  *+,0,AR4
0617 0405 60AB    SACL *+,0,AR3
0618 0406 20AC    LAC  *+,0,AR4
0619 0407 60A0    SACL *+,0
0620 0408
0621 0408 FE80    CALL FMULT
   0409 07DB
0622 040A
0623 040A 558B    LARP 3
0624 040B C460    LARK AR4,ASIGN
0625 040C C368    LARK AR3,CSIGN
0626 040D 20AC    LAC  *+,0,AR4
0627 040E 60AB    SACL *+,0,AR3
0628 040F 20AC    LAC  *+,0,AR4
0629 0410 60AB    SACL *+,0,AR3
0630 0411 20AC    LAC  *+,0,AR4
0631 0412 60AB    SACL *+,0,AR3
0632 0413 20AC    LAC  *+,0,AR4
0633 0414 CE19    SFR
0634 0415 D004    ANDK >7FC0
   0416 7FC0
0635 0417 60A0    SACL *+,0
0636 0418
0637                                        * weight initially at >0b00
0638 0418 D400    LRLK  AR4,STORE4
   0419 0205
0639 041A 338B    LAR  AR3,*,AR3
0640 041B C464    LARK AR4,BSIGN
0641 041C 20AC    LAC  *+,0,AR4
0642 041D 60AB    SACL *+,0,AR3
0643 041E 20AC    LAC  *+,0,AR4
0644 041F 60AB    SACL *+,0,AR3
0645 0420 20AC    LAC  *+,0,AR4
0646 0421 60AB    SACL *+,0,AR3
0647 0422 20AC    LAC  *+,0,AR4
0648 0423 60AB    SACL *+,0,AR3

NO$IDT    32020 FAMILY MACRO ASSEMBLER  PC 1.1 86.036    15:12:22  06-20-88
                                  PAGE 0014

0649 0424
0650 0424 FE80    CALL FADD
   0425 0719
0651 0426 558B    LARP 3
0652                                        * update values of weights
0653 0427 D300    LRLK AR3,STORE4
   0428 0205
0654 0429 3480    LAR  AR4,*
0655 042A C368    LARK AR3,CSIGN
0656 042B 20AC    LAC  *+,0,AR4
0657 042C 60AB    SACL *+,0,AR3
0658 042D 20AC    LAC  *+,0,AR4
0659 042E 60AB    SACL *+,0,AR3
0660 042F 20AC    LAC  *+,0,AR4
0661 0430 60AB    SACL *+,0,AR3
0662 0431 20AC    LAC  *+,0,AR4
0663 0432 CE19    SFR
0664 0433 D004    ANDK >7FC0
   0434 7FC0
0665 0435 60A0    SACL *+,0
0666 0436
0667                                        * increment b
```

```
0668 0436 D400    LRLK AR4,STORE3
   0437 0204
0669 0438 2080    LAC  *,0
0670 0439 D002    ADLK 4
   043A 0004
0671 043B 60A0    SACL *+
0672 043C
0673                                          * and weight position
0674 043C 2080    LAC  *,0
0675 043D D002    ADLK 4
   043E 0004
0676 043F 6080    SACL *
0677 0440 CE26    RET
0678 0441
0679 0441
0680 0441
0681 0441 D400  PREFTR  LRLK AR4,REMAR2
   0442 0213
0682 0443 558C    LARP 4
0683 0444 7280    SAR  AR2,*
0684 0445
0685 0445 D400    LRLK   AR4,CNTAPS
   0446 0209
0686 0447 3280    LAR  AR2,*
0687 0448 D400    LRLK AR4,>C00
   0449 0C00
0688 044A CBA3    RPTK 163
0689 044B FDA0    BLKD >C04,*+
   044C 0C04
0690 044D
0691                                          * clear >CA4
0692 044D D400    LRLK AR4,>CA4
   044E 0CA4
0693 044F CA00    ZAC
0694 0450 CB03    RPTK 3

NO$IDT    32020 FAMILY MACRO ASSEMBLER  PC 1.1 86.036    15:12:22  06-20-88
                                  PAGE 0015

0695 0451 60A0    SACL *+
0696 0452
0697                                          * initial y(a-1)
0698                                          * read contents in extended memory
                                                    address
0699                                          * pointed by ar3 and store in asign
0700 0452 D400  OUTFTR  LRLK AR4,STORE1
   0453 0202
0701 0454 338B    LAR  AR3,*,AR3
0702 0455 C460    LARK AR4,ASIGN
0703 0456 20AC    LAC  *+,0,AR4
0704 0457 60AB    SACL *+,0,AR3
0705 0458 20AC    LAC  *+,0,AR4
0706 0459 60AB    SACL *+,0,AR3
0707 045A 20AC    LAC  *+,0,AR4
0708 045B 60AB    SACL *+,0,AR3
0709 045C 20AC    LAC  *+,0,AR4
0710 045D 60A0    SACL *+,0
0711 045E
0712 045E
0713 045E D400    LRLK AR4,CLMEM
   045F 0246
0714 0460 338B    LAR  AR3,*,AR3
0715 0461 C464    LARK AR4,BSIGN
0716 0462 20AC    LAC  *+,0,AR4
0717 0463 60AB    SACL *+,0,AR3
0718 0464 20AC    LAC  *+,0,AR4
0719 0465 60AB    SACL *+,0,AR3
0720 0466 20AC    LAC  *+,0,AR4
0721 0467 60AB    SACL *+,0,AR3
0722 0468 20AC    LAC  *+,0,AR4
0723 0469 60AB    SACL *+,0,AR3
0724 046A
```

```
0725 046A FE80    CALL FNEG
   046B 0715
0726 046C FE80    CALL FADD
   046D 0719
0727 046E
0728                                          *transfer (c) to (a)
0729 046E 558B    LARP 3
0730 046F C460    LARK AR4,ASIGN
0731 0470 C368    LARK AR3,CSIGN
0732 0471 20AC     LAC  *+,0,AR4
0733 0472 60AB     SACL *+,0,AR3
0734 0473 20AC     LAC  *+,0,AR4
0735 0474 60AB     SACL *+,0,AR3
0736 0475 20AC     LAC  *+,0,AR4
0737 0476 60AB     SACL *+,0,AR3
0738 0477 20AC     LAC  *+,0,AR4
0739 0478 CE19    SFR
0740 0479 D004    ANDK >7FC0
   047A 7FC0
0741 047B 60A0     SACL *+,0
0742 047C
0743                                          *  W(K-A+1)
0744 047C D400    LRLK AR4,WPADDR
   047D 0216
0745 047E 338B    LAR  AR3,*,AR3

NO$IDT    32020 FAMILY MACRO ASSEMBLER  PC 1.1 86.036    15:12:22  06-20-88
                                  PAGE 0016

0746 047F C464    LARK AR4,BSIGN
0747 0480 20AC     LAC  *+,0,AR4
0748 0481 60AB     SACL *+,0,AR3
0749 0482 20AC     LAC  *+,0,AR4
0750 0483 60AB     SACL *+,0,AR3
0751 0484 20AC     LAC  *+,0,AR4
0752 0485 60AB     SACL *+,0,AR3
0753 0486 20AC     LAC  *+,0,AR4
0754 0487 60AB     SACL *+,0,AR3
0755 0488
0756                                          *  Y(A-1)*W(K-A+1)
0757 0488 FE80    CALL FMULT
   0489 07DB
0758 048A
0759 048A 558B    LARP 3
0760 048B C460    LARK AR4,ASIGN
0761 048C C368    LARK AR3,CSIGN
0762 048D 20AC     LAC  *+,0,AR4
0763 048E 60AB     SACL *+,0,AR3
0764 048F 20AC     LAC  *+,0,AR4
0765 0490 60AB     SACL *+,0,AR3
0766 0491 20AC     LAC  *+,0,AR4
0767 0492 60AB     SACL *+,0,AR3
0768 0493 20AC     LAC  *+,0,AR4
0769 0494 CE19    SFR
0770 0495 D004    ANDK >7FC0
   0496 7FC0
0771 0497 60AB     SACL *+,0,AR3
0772 0498
0773                                          *  (CA4)+(Y(A-1)*W(K-A+1))
0774 0498 D300    LRLK AR3,>CA4
   0499 0CA4
0775 049A 20AC     LAC  *+,0,AR4
0776 049B 60AB     SACL *+,0,AR3
0777 049C 20AC     LAC  *+,0,AR4
0778 049D 60AB     SACL *+,0,AR3
0779 049E 20AC     LAC  *+,0,AR4
0780 049F 60AB     SACL *+,0,AR3
0781 04A0 20AC     LAC  *+,0,AR4
0782 04A1 60AB     SACL *+,0,AR3
0783 04A2 FE80    CALL FADD
   04A3 0719
0784 04A4 558B    LARP 3
```

```
0785 04A5
0786                                        * store results in >CA4
0787 04A5 D400   LRLK AR4,>CA4
   04A6 0CA4
0788 04A7 C368   LARK AR3,CSIGN
0789 04A8 20AC    LAC  *+,0,AR4
0790 04A9 60AB    SACL *+,0,AR3
0791 04AA 20AC    LAC  *+,0,AR4
0792 04AB 60AB    SACL *+,0,AR3
0793 04AC 20AC    LAC  *+,0,AR4
0794 04AD 60AB    SACL *+,0,AR3
0795 04AE 20AC    LAC  *+,0,AR4
0796 04AF CE19   SFR
0797 04B0 D004   ANDK >7FC0

NO$IDT    32020 FAMILY MACRO ASSEMBLER  PC 1.1 86.036    15:12:22  06-20-88
                                  PAGE 0017

   04B1 7FC0
0798 04B2 60A0    SACL *+,0
0799 04B3
0800                                        * reduced a by 1
0801 04B3 D400   LRLK AR4,STORE1
   04B4 0202
0802 04B5 2080    LAC  *,0
0803 04B6 D003   SBLK 4
   04B7 0004
0804 04B8 6080    SACL *,0
0805 04B9
0806                                        * reduced outk by 1
0807 04B9 D400   LRLK AR4,CLMEM
   04BA 0246
0808 04BB 2080    LAC  *,0
0809 04BC D003   SBLK 4
   04BD 0004
0810 04BE 6080    SACL *,0
0811 04BF
0812                                        * increment weight by 1
0813 04BF D400   LRLK AR4,WPADDR
   04C0 0216
0814 04C1 2080    LAC  *,0
0815 04C2 D002   ADLK 4
   04C3 0004
0816 04C4 6080    SACL *
0817 04C5
0818                                        * count for a loop
0819 04C5 558A    LARP 2
0820 04C6 FB9C    BANZ OUTFTR,*-,AR4
   04C7 0452
0821 04C8
0822                                        * re-initialise
0823 04C8 D300   LRLK AR3,>A00
   04C9 0A00
0824 04CA 7380    SAR  AR3,*
0825 04CB D400   LRLK AR4,STORE1
   04CC 0202
0826 04CD FD80   BLKD >206,*
   04CE 0206
0827 04CF
0828                                        * re-initialise CLMEM
0829 04CF D400   LRLK AR4,CLMEM
   04D0 0246
0830 04D1 D300   LRLK AR3,>1EA4
   04D2 1EA4
0831 04D3 7380    SAR  AR3,*
0832 04D4
0833                                        * return ar2's memory
0834 04D4 D400   LRLK AR4,REMAR2
   04D5 0213
0835 04D6 3280    LAR  AR2,*
0836 04D7
0837 04D7 CE26   RET
```

```
0838 04D8
0839 04D8
0840 04D8 C460  MODEL   LARK AR4,ASIGN

NO$IDT    32020 FAMILY MACRO ASSEMBLER  PC 1.1 86.036    15:12:22  06-20-88
                                    PAGE 0018

0841 04D9 D300    LRLK AR3,MNUM6
   04DA 021D
0842 04DB 558B    LARP 3
0843 04DC 20AC    LAC  *+,0,AR4  (20E) LOAD INTO ACC
0844 04DD 60AB    SACL *+,0,AR3  (ACC) LOAD INTO ASIGN
0845 04DE 20AC    LAC  *+,0,AR4
0846 04DF 60AB    SACL *+,0,AR3
0847 04E0 20AC    LAC  *+,0,AR4
0848 04E1 60AB    SACL *+,0,AR3
0849 04E2 20AC    LAC  *+,0,AR4
0850 04E3 60A8    SACL *+,0,AR0
0851 04E4
0852                                        *  Y(K-2)
0853 04E4 C464    LARK AR4,BSIGN
0854 04E5 D000    LRLK AR0,STORE5
   04E6 0206
0855 04E7 338B    LAR  AR3,*,AR3              *AR0
0856 04E8 CB17    RPTK 23
0857 04E9 5590    MAR  *-
0858 04EA
0859 04EA 20AC    LAC  *+,0,AR4
0860 04EB 60AB    SACL *+,0,AR3
0861 04EC 20AC    LAC  *+,0,AR4
0862 04ED 60AB    SACL *+,0,AR3
0863 04EE 20AC    LAC  *+,0,AR4
0864 04EF 60AB    SACL *+,0,AR3
0865 04F0 20AC    LAC  *+,0,AR4
0866 04F1 60A8    SACL *+,0,AR0
0867 04F2
0868                                        *  update Yk memories
0869 04F2 7380    SAR  AR3,*
                                            *AR0;(AR0)=STOREY ADDRESS
0870                                        *  (AR3)=(PRE-STOREY)+4
0871                                        *  Y(K-2)*NUM2
0872 04F3 FE80    CALL FMULT
   04F4 07DB
0873 04F5
0874                                    *  TRANSFER CONTENTS IN CS' TO MEMS'
0875 04F5 558B    LARP 3
0876 04F6 D400    LRLK AR4,MEM
   04F7 022A
0877 04F8 C368    LARK AR3,CSIGN
0878 04F9 20AC    LAC  *+,0,AR4
0879 04FA 60AB    SACL *+,0,AR3
0880 04FB 20AC    LAC  *+,0,AR4
0881 04FC 60AB    SACL *+,0,AR3
0882 04FD 20AC    LAC  *+,0,AR4
0883 04FE 60AB    SACL *+,0,AR3
0884 04FF 20AC    LAC  *+,0,AR4
0885 0500 CE19    SFR
0886 0501 D004    ANDK >7FC0
   0502 7FC0
0887 0503 60AB    SACL *+,0,AR3
0888 0504
0889                                            *  NUM1
0890 0504 C460    LARK AR4,ASIGN
0891 0505 D300    LRLK AR3,MNUM5
   0506 0219
NO$IDT    32020 FAMILY MACRO ASSEMBLER  PC 1.1 86.036    15:12:22  06-20-88
                                    PAGE 0019

0892 0507 20AC    LAC  *+,0,AR4  (20A)=(ACC)
0893 0508 60AB    SACL *+,0,AR3  (ACC)=(ASIGN)
0894 0509 20AC    LAC  *+,0,AR4
0895 050A 60AB    SACL *+,0,AR3
```

```
0896 050B 20AC    LAC  *+,0,AR4
0897 050C 60AB    SACL *+,0,AR3
0898 050D 20AC    LAC  *+,0,AR4
0899 050E 60A8    SACL *+,0,AR0
0900 050F
0901 050F
0902                                        * Y(k-1)
0903                                        * LARK AR4,BSIGN BSIGN = ASIGN +4
0904 050F D000    LRLK AR0,STORE5
   0510 0206
0905 0511 338B    LAR  AR3,*,AR3 *AR0
0906 0512 20AC    LAC  *+,0,AR4
0907 0513 60AB    SACL *+,0,AR3
0908 0514 20AC    LAC  *+,0,AR4
0909 0515 60AB    SACL *+,0,AR3
0910 0516 20AC    LAC  *+,0,AR4
0911 0517 60AB    SACL *+,0,AR3
0912 0518 20AC    LAC  *+,0,AR4
0913 0519 60AB    SACL *+,0,AR3
0914 051A
0915 051A
0916                                        * update Yk memories
0917 051A CB0F    RPTK 15
0918 051B 55A0    MAR  *+
0919 051C 5588    LARP 0
0920 051D 7380    SAR  AR3,*                *AR0;(AR0)=STOREY' ADDRESS
0921                                        * (AR3)=(PRE-STOREY)+4
0922 051E
0923 051E
0924 051E FE80    CALL FMULT
   051F 07DB
0925 0520
0926                                        * TRANSFER CONTENTS IN CS' TO AS'
0927                                        * Y(k-1)*num1
0928 0520 C460    LARK AR4,ASIGN
0929 0521 C368    LARK AR3,CSIGN
0930 0522 558B    LARP 3
0931 0523 20AC    LAC  *+,0,AR4
0932 0524 60AB    SACL *+,0,AR3
0933 0525 20AC    LAC  *+,0,AR4
0934 0526 60AB    SACL *+,0,AR3
0935 0527 20AC    LAC  *+,0,AR4
0936 0528 60AB    SACL *+,0,AR3
0937 0529 20AC    LAC  *+,0,AR4
0938 052A CE19    SFR
0939 052B D004    ANDK >7FC0
   052C 7FC0
0940 052D 60AB    SACL *+,0,AR3
0941 052E
0942 052E
0943                                        * Y(k-1)*num1+Y(k-2)*num2
0944                                        * LARK AR4,BSIGN BSIGN = ASIGN+4
0945 052E D300    LRLK AR3,MEM

NO$IDT    32020 FAMILY MACRO ASSEMBLER  PC 1.1 86.036    15:12:22  06-20-88
                                  PAGE 0020

   052F 022A
0946 0530 20AC    LAC  *+,0,AR4
0947 0531 60AB    SACL *+,0,AR3
0948 0532 20AC    LAC  *+,0,AR4
0949 0533 60AB    SACL *+,0,AR3
0950 0534 20AC    LAC  *+,0,AR4
0951 0535 60AB    SACL *+,0,AR3
0952 0536 20AC    LAC  *+,0,AR4
0953 0537 60A0    SACL *+,0
0954 0538
0955 0538 FE80    CALL FADD
   0539 0719
0956 053A 558B    LARP 3
0957                                        * TRANSFER CONTENTS IN CS' TO MEMS'
0958 053B D400    LRLK AR4,MEM
```

```
   053C 022A
0959 053D C368    LARK AR3,CSIGN
0960 053E 20AC    LAC  *+,0,AR4
0961 053F 60AB    SACL *+,0,AR3
0962 0540 20AC    LAC  *+,0,AR4
0963 0541 60AB    SACL *+,0,AR3
0964 0542 20AC    LAC  *+,0,AR4
0965 0543 60AB    SACL *+,0,AR3
0966 0544 20AC    LAC  *+,0,AR4
0967 0545 CE19    SFR
0968 0546 D004    ANDK >7FC0
   0547 7FC0
0969 0548 60AB    SACL *+,0,AR3
0970 0549
0971 0549
0972 0549
0973 0549
0974                                          * DEN1
0975 0549
0976 0549 C460    LARK AR4,ASIGN
0977 054A D300    LRLK AR3,MDEN1
   054B 0221
0978 054C 20AC    LAC  *+,0,AR4  (212)=(ACC)
0979 054D 60AB    SACL *+,0,AR3  (ACC)=(ASIGN)
0980 054E 20AC    LAC  *+,0,AR4
0981 054F 60AB    SACL *+,0,AR3
0982 0550 20AC    LAC  *+,0,AR4
0983 0551 60AB    SACL *+,0,AR3
0984 0552 20AC    LAC  *+,0,AR4
0985 0553 60AB    SACL *+,0,AR3
0986 0554
0987 0554
0988                                          * D(k-1)
0989 0554 D300    LRLK AR3,DK11
   0555 023D
0990 0556 20AC    LAC  *+,0,AR4
0991 0557 60AB    SACL *+,0,AR3
0992 0558 20AC    LAC  *+,0,AR4
0993 0559 60AB    SACL *+,0,AR3
0994 055A 20AC    LAC  *+,0,AR4
0995 055B 60AB    SACL *+,0,AR3
0996 055C 20AC    LAC  *+,0,AR4

NO$IDT    32020 FAMILY MACRO ASSEMBLER  PC 1.1 86.036   15:12:22  06-20-88
                                PAGE 0021

0997 055D 60A0    SACL *+,0
0998 055E
0999 055E
1000 055E FE80    CALL FMULT
   055F 07DB
1001 0560
1002                                          * TRANSFER CONTENTS IN CS' TO AS'
1003                                          * F(k-1)*den1
1004 0560 C460    LARK AR4,ASIGN
1005 0561 C368    LARK AR3,CSIGN
1006 0562 558B    LARP 3
1007 0563 20AC    LAC  *+,0,AR4
1008 0564 60AB    SACL *+,0,AR3
1009 0565 20AC    LAC  *+,0,AR4
1010 0566 60AB    SACL *+,0,AR3
1011 0567 20AC    LAC  *+,0,AR4
1012 0568 60AB    SACL *+,0,AR3
1013 0569 20AC    LAC  *+,0,AR4
1014 056A CE19    SFR
1015 056B D004    ANDK >7FC0
   056C 7FC0
1016 056D 60AB    SACL *+,0,AR3
1017 056E
1018 056E
1019                                          * Y(k)*num0+Y(k-1)*num1+Y(k-2)*num2
```

```
1020                                        * -F(k-1)*den1
1021 056E C464    LARK AR4,BSIGN
1022 056F D300    LRLK AR3,MEM
   0570 022A
1023 0571 20AC     LAC  *+,0,AR4
1024 0572 60AB     SACL *+,0,AR3
1025 0573 20AC     LAC  *+,0,AR4
1026 0574 60AB     SACL *+,0,AR3
1027 0575 20AC     LAC  *+,0,AR4
1028 0576 60AB     SACL *+,0,AR3
1029 0577 20AC     LAC  *+,0,AR4
1030 0578 60A0     SACL *+,0
1031 0579
1032 0579 FE80    CALL FADD
   057A 0719
1033 057B 558B     LARP 3
1034                                        * TRANSFER CONTENTS IN CS' TO MEMS'
1035 057C D400    LRLK AR4,MEM
   057D 022A
1036 057E C368    LARK AR3,CSIGN
1037 057F 20AC     LAC  *+,0,AR4
1038 0580 60AB     SACL *+,0,AR3
1039 0581 20AC     LAC  *+,0,AR4
1040 0582 60AB     SACL *+,0,AR3
1041 0583 20AC     LAC  *+,0,AR4
1042 0584 60AB     SACL *+,0,AR3
1043 0585 20AC     LAC  *+,0,AR4
1044 0586 CE19     SFR
1045 0587 D004    ANDK >7FC0
   0588 7FC0
1046 0589 60AB     SACL *+,0,AR3
1047 058A

NO$IDT    32020 FAMILY MACRO ASSEMBLER  PC 1.1 86.036    15:12:22  06-20-88
                                    PAGE 0022

1048 058A
1049                                        * MDEN2
1050 058A
1051 058A C460    LARK AR4,ASIGN
1052 058B D300    LRLK AR3,MDEN2
   058C 0225
1053 058D 20AC     LAC  *+,0,AR4
1054 058E 60AB     SACL *+,0,AR3
1055 058F 20AC     LAC  *+,0,AR4
1056 0590 60AB     SACL *+,0,AR3
1057 0591 20AC     LAC  *+,0,AR4
1058 0592 60AB     SACL *+,0,AR3
1059 0593 20AC     LAC  *+,0,AR4
1060 0594 60AB     SACL *+,0,AR3
1061 0595
1062 0595
1063                                        * D(k-2)
1064 0595 D300    LRLK AR3,DK21
   0596 0239
1065 0597 20AC     LAC  *+,0,AR4
1066 0598 60AB     SACL *+,0,AR3
1067 0599 20AC     LAC  *+,0,AR4
1068 059A 60AB     SACL *+,0,AR3
1069 059B 20AC     LAC  *+,0,AR4
1070 059C 60AB     SACL *+,0,AR3
1071 059D 20AC     LAC  *+,0,AR4
1072 059E 60A0     SACL *+,0
1073 059F
1074 059F
1075 059F FE80    CALL FMULT
   05A0 07DB
1076 05A1
1077                                        * TRANSFER CONTENTS IN CS' TO AS'
1078                                        * D(k-2)*den2
1079 05A1 C460    LARK AR4,ASIGN
1080 05A2 C368    LARK AR3,CSIGN
```

```
1081 05A3 558B    LARP 3
1082 05A4 20AC    LAC  *+,0,AR4
1083 05A5 60AB    SACL *+,0,AR3
1084 05A6 20AC    LAC  *+,0,AR4
1085 05A7 60AB    SACL *+,0,AR3
1086 05A8 20AC    LAC  *+,0,AR4
1087 05A9 60AB    SACL *+,0,AR3
1088 05AA 20AC    LAC  *+,0,AR4
1089 05AB CE19    SFR
1090 05AC D004    ANDK >7FC0
   05AD 7FC0
1091 05AE 60AB    SACL *+,0,AR3
1092 05AF
1093                                        * D(K) =
1094                                        * Y(k)*num0+Y(k-1)*num1+Y(k-2)*num2
1095                                        *  +D(k-1)*den1+D(k-2)*den2
1096 05AF C464    LARK AR4,BSIGN
1097 05B0 D300    LRLK AR3,MEM
   05B1 022A
1098 05B2 20AC    LAC  *+,0,AR4
1099 05B3 60AB    SACL *+,0,AR3

NO$IDT    32020 FAMILY MACRO ASSEMBLER  PC 1.1 86.036    15:12:22  06-20-88
                                  PAGE 0023

1100 05B4 20AC    LAC  *+,0,AR4
1101 05B5 60AB    SACL *+,0,AR3
1102 05B6 20AC    LAC  *+,0,AR4
1103 05B7 60AB    SACL *+,0,AR3
1104 05B8 20AC    LAC  *+,0,AR4
1105 05B9 60A0    SACL *+,0
1106 05BA
1107 05BA FE80    CALL FADD
   05BB 0719
1108 05BC
1109 05BC 558B    LARP 3
1110                                        * TRANSFER CONTENTS IN CS' TO DKS'
1111 05BD D400    LRLK AR4,DK1
   05BE 0241
1112 05BF C368    LARK AR3,CSIGN
1113 05C0 20AC    LAC  *+,0,AR4
1114 05C1 60AB    SACL *+,0,AR3
1115 05C2 20AC    LAC  *+,0,AR4
1116 05C3 60AB    SACL *+,0,AR3
1117 05C4 20AC    LAC  *+,0,AR4
1118 05C5 60AB    SACL *+,0,AR3
1119 05C6 20AC    LAC  *+,0,AR4
1120 05C7 CE19    SFR
1121 05C8 D004    ANDK >7FC0
   05C9 7FC0
1122 05CA 60A0    SACL *+,0
1123 05CB
1124 05CB D400    LRLK AR4,DK21
   05CC 0239
1125 05CD CB07    RPTK 7
1126 05CE FDA0    BLKD DK11,*+
   05CF 023D
1127 05D0
1128 05D0 CE26    RET
1129 05D1
1130 05D1 D400 PIDTFY  LRLK AR4,CNTAPS
   05D2 0209
1131 05D3 3280    LAR  AR2,*                *AR4  (ar2) = (cntaps) = 39
1132 05D4
1133 05D4 FE80    CALL C2UEP
   05D5 05E9
1134 05D6
1135 05D6 FE80 WGHGT2  CALL CALPP
   05D7 06BA
1136 05D8
1137 05D8
1138                                        * COUNT FOR NUMBER OF TAPS
```

```
1139 05D8 558A    LARP 2
1140 05D9 FB90    BANZ WGHGT2,*-
   05DA 05D6
1141 05DB
1142                                        * RE-INITIALISE STORE3 AS ZERO
1143 05DB D200    LRLK AR2,STORE3
   05DC 0204
1144 05DD CA00    ZAC
1145 05DE 6080    SACL *,0      *AR2 (AR2)=204; 205
1146 05DF

NO$IDT    32020 FAMILY MACRO ASSEMBLER  PC 1.1 86.036   15:12:22  06-20-88
                                  PAGE 0024

1147                                        * RE-INITIALISE PTAPS
1148 05DF D200    LRLK AR2,PTAPS
   05E0 0238
1149 05E1 D300    LRLK AR3,>A00
   05E2 0A00
1150 05E3 7380    SAR  AR3,*
1151 05E4
1152 05E4
1153 05E4 558C    LARP 4
1154                                        * return ar2's memory
1155 05E5 D400    LRLK AR4,REMAR2
   05E6 0213
1156 05E7 3280    LAR  AR2,*
1157 05E8
1158 05E8 CE26    RET
1159 05E9
1160                                        *CAL PROCEDURE
1161 05E9
1162 05E9
1163                                  *clear contents of output memory locations
1164                                  *by stored "directly" a smallest positive value in  their dma
1165                                  *NOTAPS = 40
1166 05E9
1167                                  *we need to use ar2 in this procedure while (ar2)  must not be
1168                                  *altered in a main body . So we keep its value in  memory loc
1169                                  *remar2
1170 05E9 D400  C2UEP   LRLK AR4,REMAR2
   05EA 0213
1171 05EB 558C    LARP 4
1172 05EC 7280    SAR  AR2,*                  *AR4 (ar2) is put into
1173                                           * this memory location
1174 05ED
1175 05ED D400    LRLK   AR4,CNTAPS
   05EE 0209
1176 05EF 3280    LAR  AR2,*                  *AR4 (ar2) = (cntaps) = 39
1177 05F0
1178                                           *clear taps
1179 05F0
1180 05F0 D400    LRLK AR4,OUT1
   05F1 0232
1181 05F2 CA00    ZAC
1182 05F3 CB03    RPTK 3
1183 05F4 60A0    SACL *+
1184 05F5
1185                                        * INITIAL Y(A-1)
1186                                        * READ CONTENTS IN EXTENDED MEMORY ADRRESS
1187                                        * POINTED BY AR3 AND STORE IN ASIGN
1188 05F5 D400  OUTL2   LRLK AR4,CNTLST           [UK .. UK-39]
   05F6 0236
1189 05F7 208B    LAC  *,0,AR3
1190 05F8 D300    LRLK AR3,STORE3
   05F9 0204
1191 05FA 1080    SUB  *,0
1192 05FB D300    LRLK AR3,UKB
   05FC 0237
1193 05FD 608C    SACL *,0,AR4
1194 05FE D400    LRLK AR4,UKB
```

```
   05FF 0237
1195 0600 338B    LAR  AR3,*,AR3
1196 0601
1197 0601 C460    LARK AR4,ASIGN
1198 0602
1199 0602 20AC    LAC  *+,0,AR4
1200 0603 60AB    SACL *+,0,AR3
1201 0604 20AC    LAC  *+,0,AR4
1202 0605 60AB    SACL *+,0,AR3
1203 0606 20AC    LAC  *+,0,AR4
1204 0607 60AB    SACL *+,0,AR3
1205 0608 20AC    LAC  *+,0,AR4
1206 0609 60A0    SACL *+,0
1207 060A
1208                                        * W(K-A+1)
1209 060A D400    LRLK AR4,PTAPS
   060B 0238
1210 060C 338B    LAR  AR3,*,AR3
1211 060D C464    LARK AR4,BSIGN
1212 060E
1213 060E 20AC    LAC  *+,0,AR4
1214 060F 60AB    SACL *+,0,AR3
1215 0610 20AC    LAC  *+,0,AR4
1216 0611 60AB    SACL *+,0,AR3
1217 0612 20AC    LAC  *+,0,AR4
1218 0613 60AB    SACL *+,0,AR3
1219 0614 20AC    LAC  *+,0,AR4
1220 0615 60AB    SACL *+,0,AR3
1221 0616
1222 0616
1223                                        * Y(A-1)*W(K-A+1)
1224 0616 FE80    CALL FMULT
   0617 07DB
1225 0618 558B    LARP 3
1226 0619
1227                                        * TRANSFER CONTENTS IN CS' TO AS'
1228 0619 C460    LARK AR4,ASIGN
1229 061A C368    LARK AR3,CSIGN
1230 061B 20AC    LAC  *+,0,AR4
1231 061C 60AB    SACL *+,0,AR3
1232 061D 20AC    LAC  *+,0,AR4
1233 061E 60AB    SACL *+,0,AR3
1234 061F 20AC    LAC  *+,0,AR4
1235 0620 60AB    SACL *+,0,AR3
1236 0621 20AC    LAC  *+,0,AR4
1237 0622 CE19    SFR
1238 0623 D004    ANDK >7FC0
   0624 7FC0
1239 0625 60AB    SACL *+,0,AR3
1240 0626
1241                                        * OUT1+(Y(A-1)*W(K-A+1))
1242 0626
1243                                        * LARK AR4,BSIGN
1244 0626 D300    LRLK AR3,OUT1
   0627 0232
1245 0628
1246                                        *block transfer
```



```
1247 0628 20AC    LAC  *+,0,AR4
1248 0629 60AB    SACL *+,0,AR3
1249 062A 20AC    LAC  *+,0,AR4
1250 062B 60AB    SACL *+,0,AR3
1251 062C 20AC    LAC  *+,0,AR4
1252 062D 60AB    SACL *+,0,AR3
1253 062E 20AC    LAC  *+,0,AR4
1254 062F 60AB    SACL *+,0,AR3
```

```
1255 0630
1256 0630 FE80    CALL FADD
   0631 0719
1257 0632 558B    LARP 3
1258 0633
1259                                        * STORE RESULTS IN OUT1
1260 0633 D400    LRLK AR4,OUT1
   0634 0232
1261 0635 C368    LARK AR3,CSIGN
1262 0636
1263                                        *block transfer
1264 0636 20AC     LAC  *+,0,AR4
1265 0637 60AB     SACL *+,0,AR3
1266 0638 20AC     LAC  *+,0,AR4
1267 0639 60AB     SACL *+,0,AR3
1268 063A 20AC     LAC  *+,0,AR4
1269 063B 60AB     SACL *+,0,AR3
1270 063C 20AC     LAC  *+,0,AR4
1271 063D CE19     SFR
1272 063E D004     ANDK >7FC0
   063F 7FC0
1273 0640 60A0     SACL *+,0
1274 0641
1275                                        * increment b
1276 0641 D400    LRLK AR4,STORE3
   0642 0204
1277 0643 2080    LAC  *,0
1278 0644 D002    ADLK 4
   0645 0004
1279 0646 60A0     SACL *+
1280 0647
1281                                        * INCREMENT WEIGHT BY 1
1282 0647 D400    LRLK AR4,PTAPS
   0648 0238
1283 0649 2080    LAC  *,0      *AR4;(ACC)=(214)
1284 064A D002    ADLK 4        UPDATE
   064B 0004
1285 064C 6080    SACL *        *AR4;(ACC)=(214)
1286 064D
1287                                        * COUNT FOR A LOOP
1288 064D 558A    LARP 2
1289 064E FB9C    BANZ OUTL2,*-,AR4
   064F 05F5
1290                                        * END OF CALCULATION OUTL2
1291 0650
1292                                        * RE-INITIALISE PTAPS
1293 0650 D300    LRLK AR3,>A00  (AR3)=A00
   0651 0A00
1294 0652 7380    SAR  AR3,*                 *=AR4,(AR4)=PTAPS;(PTAPS)=>A00

NO$IDT    32020 FAMILY MACRO ASSEMBLER  PC 1.1 86.036   15:12:22  06-20-88
                                   PAGE 0027

1295 0653
1296                                        * re-initialise store3
1297 0653 D400    LRLK AR4,STORE3
   0654 0204
1298 0655 CA00     ZAC
1299 0656 6080     SACL *,0
1300 0657
1301                                        * RETURN AR2'S MEMORY
1302 0657 D400    LRLK AR4,REMAR2
   0658 0213
1303 0659 3280    LAR  AR2,*                 *AR4
1304 065A
1305                                        * DK-1
1306 065A 558B     LARP 3
1307 065B D300    LRLK AR3,>1EA4
   065C 1EA4
1308 065D C460    LARK AR4,ASIGN
1309 065E 20AC     LAC  *+,0,AR4
1310 065F 60AB     SACL *+,0,AR3
```

```
1311 0660 20AC    LAC  *+,0,AR4
1312 0661 60AB    SACL *+,0,AR3
1313 0662 20AC    LAC  *+,0,AR4
1314 0663 60AB    SACL *+,0,AR3
1315 0664 20AC    LAC  *+,0,AR4
1316 0665 60AB    SACL *+,0,AR3
1317 0666
1318                                        * OUT1(K-1)
1319 0666
1320 0666 C464    LARK AR4,BSIGN
1321 0667 D300    LRLK AR3,OUT1
   0668 0232
1322 0669
1323                                        *block transfer
1324 0669 20AC    LAC  *+,0,AR4
1325 066A 60AB    SACL *+,0,AR3
1326 066B 20AC    LAC  *+,0,AR4
1327 066C 60AB    SACL *+,0,AR3
1328 066D 20AC    LAC  *+,0,AR4
1329 066E 60AB    SACL *+,0,AR3
1330 066F 20AC    LAC  *+,0,AR4
1331 0670 60AB    SACL *+,0,AR3
1332 0671
1333                                        * ER(K-1)
1334 0671 FE80    CALL  FNEG
   0672 0715
1335 0673 FE80    CALL FADD
   0674 0719
1336 0675 558B    LARP 3
1337                                        *transfer contents in c's to a's
1338 0676 C460    LARK AR4,ASIGN
1339 0677 C368    LARK AR3,CSIGN
1340 0678 20AC    LAC  *+,0,AR4
1341 0679 60AB    SACL *+,0,AR3
1342 067A 20AC    LAC  *+,0,AR4
1343 067B 60AB    SACL *+,0,AR3
1344 067C 20AC    LAC  *+,0,AR4
1345 067D 60AB    SACL *+,0,AR3

NO$IDT    32020 FAMILY MACRO ASSEMBLER  PC 1.1 86.036   15:12:22  06-20-88
                                  PAGE 0028

1346 067E 20AC    LAC  *+,0,AR4
1347 067F CE19    SFR
1348 0680 D004    ANDK >7FC0
   0681 7FC0
1349 0682 60AB    SACL *+,0,AR3
1350 0683
1351                                        * LARK AR4,BSIGN
1352 0683 D300    LRLK AR3,UP
   0684 022E
1353 0685
1354                                        * block transfer
1355 0685 20AC    LAC  *+,0,AR4
1356 0686 60AB    SACL *+,0,AR3
1357 0687 20AC    LAC  *+,0,AR4
1358 0688 60AB    SACL *+,0,AR3
1359 0689 20AC    LAC  *+,0,AR4
1360 068A 60AB    SACL *+,0,AR3
1361 068B 20AC    LAC  *+,0,AR4
1362 068C 60AB    SACL *+,0,AR3
1363 068D
1364 068D FE80    CALL FMULT
   068E 07DB
1365 068F 558B    LARP 3
1366                                        *transfer contents in cs' to as'
1367 0690 C460    LARK AR4,ASIGN
1368 0691 C368    LARK AR3,CSIGN
1369 0692 20AC    LAC  *+,0,AR4
1370 0693 60AB    SACL *+,0,AR3
1371 0694 20AC    LAC  *+,0,AR4
1372 0695 60AB    SACL *+,0,AR3
```

```
1373 0696 20AC    LAC  *+,0,AR4
1374 0697 60AB    SACL *+,0,AR3
1375 0698 20AC    LAC  *+,0,AR4
1376 0699 CE19    SFR
1377 069A D004    ANDK >7FC0
   069B 7FC0
1378 069C 60AB    SACL *+,0,AR3
1379 069D
1380                                        * MOVE CONTANT 2 TO BS'
1381 069D
1382 069D D300    LRLK AR3,CNST2
   069E 020B
1383 069F C464    LARK AR4,BSIGN
1384 06A0
1385 06A0 20AC    LAC  *+,0,AR4
1386 06A1 60AB    SACL *+,0,AR3
1387 06A2 20AC    LAC  *+,0,AR4
1388 06A3 60AB    SACL *+,0,AR3
1389 06A4 20AC    LAC  *+,0,AR4
1390 06A5 60AB    SACL *+,0,AR3
1391 06A6 20AC    LAC  *+,0,AR4
1392 06A7 60AB    SACL *+,0,AR3
1393 06A8
1394 06A8 FE80    CALL FMULT
   06A9 07DB
1395 06AA 558B    LARP 3
1396                                        *transfer contents in cs' to no2ues'

NO$IDT    32020 FAMILY MACRO ASSEMBLER  PC 1.1 86.036    15:12:22  06-20-88
                                PAGE 0029

1397 06AB D400    LRLK AR4,P2UE
   06AC 024C
1398 06AD C368    LARK AR3,CSIGN
1399 06AE 20AC    LAC  *+,0,AR4
1400 06AF 60AB    SACL *+,0,AR3
1401 06B0 20AC    LAC  *+,0,AR4
1402 06B1 60AB    SACL *+,0,AR3
1403 06B2 20AC    LAC  *+,0,AR4
1404 06B3 60AB    SACL *+,0,AR3
1405 06B4 20AC    LAC  *+,0,AR4
1406 06B5 CE19    SFR
1407 06B6 D004    ANDK >7FC0
   06B7 7FC0
1408 06B8 60A0    SACL *+,0
1409 06B9 CE26    RET
1410 06BA
1411 06BA 558C  CALPP   LARP 4
1412                                        * Y(K-B)
1413                                        * B = 1 INITIALLY
1414 06BB
1415 06BB D400    LRLK AR4,CNTLST           [UK .. UK-39]
   06BC 0236
1416 06BD 208B    LAC  *,0,AR3
1417 06BE D300    LRLK AR3,STORE3
   06BF 0204
1418 06C0 1080    SUB  *,0
1419 06C1 D300    LRLK AR3,UKB
   06C2 0237
1420 06C3 608C    SACL *,0,AR4
1421 06C4 D400    LRLK AR4,UKB
   06C5 0237
1422 06C6 338B    LAR  AR3,*,AR3
1423 06C7
1424 06C7 C460    LARK AR4,ASIGN
1425 06C8 20AC    LAC  *+,0,AR4
1426 06C9 60AB    SACL *+,0,AR3
1427 06CA 20AC    LAC  *+,0,AR4
1428 06CB 60AB    SACL *+,0,AR3
1429 06CC 20AC    LAC  *+,0,AR4
1430 06CD 60AB    SACL *+,0,AR3
1431 06CE 20AC    LAC  *+,0,AR4
```

```
1432 06CF 60AB    SACL *+,0,AR3
1433 06D0
1434 06D0 D300    LRLK AR3,P2UE
   06D1 024C
1435 06D2 20AC    LAC  *+,0,AR4
1436 06D3 60AB    SACL *+,0,AR3
1437 06D4 20AC    LAC  *+,0,AR4
1438 06D5 60AB    SACL *+,0,AR3
1439 06D6 20AC    LAC  *+,0,AR4
1440 06D7 60AB    SACL *+,0,AR3
1441 06D8 20AC    LAC  *+,0,AR4
1442 06D9 60AB    SACL *+,0,AR3
1443 06DA
1444 06DA FE80    CALL FMULT
   06DB 07DB
1445 06DC 558B    LARP 3

NO$IDT    32020 FAMILY MACRO ASSEMBLER  PC 1.1 86.036   15:12:22  06-20-88
                                  PAGE 0030

1446                                                    *transfer contents in cs' to as'
1447 06DD C460    LARK AR4,ASIGN
1448 06DE C368    LARK AR3,CSIGN
1449 06DF 20AC    LAC  *+,0,AR4
1450 06E0 60AB    SACL *+,0,AR3
1451 06E1 20AC    LAC  *+,0,AR4
1452 06E2 60AB    SACL *+,0,AR3
1453 06E3 20AC    LAC  *+,0,AR4
1454 06E4 60AB    SACL *+,0,AR3
1455 06E5 20AC    LAC  *+,0,AR4
1456 06E6 CE19    SFR
1457 06E7 D004    ANDK >7FC0
   06E8 7FC0
1458 06E9 60A0    SACL *+,0
1459 06EA
1460                                                    * WEIGHT INITIALLY AT >0A00
1461 06EA
1462 06EA D400    LRLK   AR4,PTAPS
   06EB 0238
1463 06EC 338B    LAR  AR3,*,AR3     *AR4
1464 06ED C464    LARK AR4,BSIGN
1465 06EE
1466 06EE 20AC    LAC  *+,0,AR4
1467 06EF 60AB    SACL *+,0,AR3
1468 06F0 20AC    LAC  *+,0,AR4
1469 06F1 60AB    SACL *+,0,AR3
1470 06F2 20AC    LAC  *+,0,AR4
1471 06F3 60AB    SACL *+,0,AR3
1472 06F4 20AC    LAC  *+,0,AR4
1473 06F5 60AB    SACL *+,0,AR3
1474 06F6
1475 06F6 FE80    CALL FADD
   06F7 0719
1476 06F8 558B    LARP 3
1477                                                    * UPDATE VALUES OF WEIGHTS
1478                                                    * MOVE CONTENTS OF CSIGN TO TAPS  LOCATION
1479 06F9 D300    LRLK AR3,PTAPS
   06FA 0238
1480 06FB 3480    LAR  AR4,*        AR3?
1481 06FC C368    LARK AR3,CSIGN
1482 06FD
1483                                                    *block transfer
1484 06FD 20AC    LAC  *+,0,AR4
1485 06FE 60AB    SACL *+,0,AR3
1486 06FF 20AC    LAC  *+,0,AR4
1487 0700 60AB    SACL *+,0,AR3
1488 0701 20AC    LAC  *+,0,AR4
1489 0702 60AB    SACL *+,0,AR3
1490 0703 20AC    LAC  *+,0,AR4
1491 0704 CE19    SFR
1492 0705 D004    ANDK >7FC0
   0706 7FC0
```

```
1493 0707 60A0    SACL *+,0
1494 0708
1495                                                  * INCREMENT B
1496 0708 D400    LRLK AR4,STORE3
   0709 0204

NO$IDT    32020 FAMILY MACRO ASSEMBLER  PC 1.1 86.036    15:12:22  06-20-88
                                    PAGE 0031

1497 070A 2080    LAC  *,0                            *AR4
1498 070B D002    ADLK 4
   070C 0004
1499 070D 6080    SACL *
1500 070E
1501                                                  * AND WEIGHT POSITION
1502 070E D400    LRLK AR4,PTAPS
   070F 0238
1503 0710 2080    LAC  *,0                            *AR4 (ar4) = store4 adress
1504 0711 D002    ADLK 4
   0712 0004
1505 0713 6080    SACL *                              *AR4
1506 0714 CE26    RET
1507 0715
1508 0715
1509 0715
1510 0715
1511                              * DEFINE FLOATING CALCULATION PROCEDURES
1512 0715
1513 0715
1514 0715 2064  FNEG   LAC  BSIGN
1515 0716 CE27    CMPL
1516 0717 6064    SACL BSIGN
1517 0718 CE26    RET
1518 0719
1519          ************************************************
1520       *
1521       *  THIS IS A FLOATING-POINT ADDITION ROUTINE  WHICH
1522       *  IMPLEMENTS THE IEEE PROPOSED FLOATING-POINT   FORMAT
1523       *  ON THE TMS32020.
1524       *
1525          ************************************************
1526       *
1527       *  INITIAL FORMAT (ALL 16 BIT WORDS)
1528          *    ------------------
1529       *  |   ALL 0 OR 1  |      ASIGN (0 OR -1)
1530          *    ------------------
1531       *
1532          *    ------------------
1533       *  |0|.  15 BITS   |      AHI (NORMALIZED)
1534          *    ------------------
1535       *
1536          *    ------------------
1537       *  |0|  9 BITS |--0-|      ALO
1538          *    ------------------
1539       *
1540          *    ------------------
1541       *  |          |      AEXP (-127 TO 128)
1542          *    ------------------
1543       *
1544       *  TO CORRESPOND WITH IEEE FORMAT,
1545       *   INPUT 0.1F * 2 ** (E + 1)
1546       *  INSTEAD OF 1.F * 2 **E, AND SUBTRACT 127 FROM  E.
1547       *
1548       *  THE FINAL FORMAT IS THE SAME AS THE INITIAL  FORMAT
1549       *  EXCEPT THAT FOR CLO WE HAVE:
1550       *

NO$IDT    32020 FAMILY MACRO ASSEMBLER  PC 1.1 86.036    15:12:22  06-20-88
                                    PAGE 0032

1551          *    ------------------
1552       *  |   16 BITS    |      CLO
```

```
1553          *  ------------------
1554          *
1555          *  ALL 16 BITS OF CLO ARE VALID. ANYTHING PAST  THESE HAS
1556          *  BEEN TRUNCATED.
1557          *
1558          *********************************************************
1559          *
1560          *  WORST CASE (EXCLUDING INITIALIZATION AND  I/O):
1561          *  15.4 MICROSECONDS.
1562          *  THIS TIMING INCLUDES THE NORMALIZATION.
1563          *  WORDS OF PROGRAM MEMORY: 170
1564          *
1565          *********************************************************
1566          *
1567          *
1568          *
1569          *  INITIALIZATION
1570          *
1571          *
1572          *
1573          *
1574 0719
1575 0719 CE07  FADD   SSXM                    SET SIGN EXTENSION.
1576 071A 5588    LARP 0
1577 071B D000    LRLK AR0,0                   CLEAR EXPONENT REGISTER.
   071C 0000
1578 071D CA01    LACK 1
1579 071E 606D    SACL ONE     ONE = 1
1580 071F CA10    LACK 16
1581 0720 6070    SACL SIXT
1582 0721 CA03    LACK 3
1583 0722 606F    SACL THREE
1584 0723 CA0D    LACK 13
1585 0724 6072    SACL TTEEN
1586          *
1587          *
1588          *  BEGIN FLOATING POINT ADD
1589          *
1590          *
1591 0725 2061  UPADD  LAC  AEXP              FIND LARGEST NUMBER.
1592 0726 1065    SUB  BEXP
1593 0727 606C    SACL D
1594 0728 F680    BZ   AEQB                  IF EXPONENTS ARE THE SAME, JUMP  TO AEQB.
   0729 0750
1595 072A F380    BLZ  ALTB                  IF A IS LESS THAN B, JUMP TO  ALTB.
   072B 075A
1596
1597 072C CE23  AGTB   NEG
1598 072D 0070    ADD  SIXT                  D = (16-D)
1599 072E 606C    SACL D
1600 072F
1601 072F F480    BGEZ CONT1
   0730 073E
1602 0731
1603                                           *  transfer a to c when b << a

NO$IDT    32020 FAMILY MACRO ASSEMBLER  PC 1.1 86.036    15:12:22  06-20-88
                                    PAGE 0033

1604 0731 C468    LARK AR4,CSIGN
1605 0732 C360    LARK AR3,ASIGN
1606 0733 558B    LARP 3
1607 0734 20AC    LAC  *+,0,AR4
1608 0735 60AB    SACL *+,0,AR3
1609 0736 20AC    LAC  *+,0,AR4
1610 0737 60AB    SACL *+,0,AR3
1611 0738 20AC    LAC  *+,0,AR4
1612 0739 60AB    SACL *+,0,AR3
1613 073A 20AC    LAC  *+,0,AR4
1614 073B CE18    SFL
1615 073C 60AB    SACL *+,0,AR3
1616 073D CE26    RET
```

```
1617 073E
1618 073E 3C6C CONT1  LT   D
1619 073F 2060    LAC  ASIGN          A IS LARGER THAN B.
1620 0740 6068    SACL CSIGN          THEREFORE, CSIGN = ASIGN.
1621 0741 2061    LAC  AEXP           ALIGN THE B MANTISSA.
1622 0742 6069    SACL CEXP
1623 0743 4266    LACT BHI            BHI IS SHIFTED RIGHT "D" TIMES.
1624 0744 6866    SACH BHI
1625 0745 6071    SACL RESID          RESIDUAL BITS MUST BE MAINTAINED.
1626 0746 4267    LACT BLO            BLO IS SHIFTED RIGHT "D" TIMES.
1627 0747 CE18    SFL                 MSB (THE 0) IS SHIFTED AWAY.
1628 0748 6867    SACH BLO
1629 0749 2067    LAC  BLO
1630 074A 4D71    OR   RESID          GET BITS THAT WERE SHIFTED  FROM BHI.
1631 074B 6067    SACL BLO
1632 074C 2163    LAC  ALO,1          GET RID OF EXTRA BIT.
1633 074D 6063    SACL ALO
1634 074E FF80    B    CHKSGN         DO BOTH NUMBERS HAVE THE SAME SIGN?
   074F 077B
1635
1636 0750 2060 AEQB   LAC  ASIGN     IF SIGNS ARE THE SAME,  CSIGN = ASIGN
1637 0751 6068    SACL CSIGN
1638 0752 2163    LAC  ALO,1          ALIGN MANTISSAS.
1639 0753 6063    SACL ALO
1640 0754 2167    LAC  BLO,1
1641 0755 6067    SACL BLO
1642 0756 2061    LAC  AEXP           SET C EXPONENT = A EXPONENT.
1643 0757 6069    SACL CEXP
1644 0758 FF80    B    CHKSGN         DO BOTH NUMBERS HAVE THE  SAME SIGN?
   0759 077B
1645
1646 075A 0070 ALTB   ADD  SIXT
1647 075B 606C    SACL D              D = (16-D)
1648 075C
1649 075C F480    BGEZ CONT2
   075D 076B
1650 075E
1651                                  * transfer b to c when b >> a
1652 075E C468    LARK AR4,CSIGN
1653 075F C364    LARK AR3,BSIGN
1654 0760 558B    LARP 3
1655 0761 20AC    LAC  *+,0,AR4
1656 0762 60AB    SACL *+,0,AR3
1657 0763 20AC    LAC  *+,0,AR4

NO$IDT    32020 FAMILY MACRO ASSEMBLER  PC 1.1 86.036   15:12:22  06-20-88
                                PAGE 0034

1658 0764 60AB    SACL *+,0,AR3
1659 0765 20AC    LAC  *+,0,AR4
1660 0766 60AB    SACL *+,0,AR3
1661 0767 20AC    LAC  *+,0,AR4
1662 0768 CE18    SFL
1663 0769 60AB    SACL *+,0,AR3
1664 076A CE26    RET
1665 076B
1666 076B
1667 076B 3C6C CONT2  LT   D
1668 076C 2064    LAC  BSIGN          B IS THE BIGGEST NUMBER.
1669 076D 6068    SACL CSIGN          THEREFORE, LET THE SIGN OF  C = BSIGN.
1670 076E 2065    LAC  BEXP           SET C EXPONENT = B EXPONENT.
1671 076F 6069    SACL CEXP
1672 0770 4262    LACT AHI            AHI GETS SHIFTED "D" TIMES.
1673 0771 6862    SACH AHI
1674 0772 6071    SACL RESID          MAINTAIN EXTRA BITS.
1675 0773 4263    LACT ALO            ALO GETS SHIFTED "D" TIMES.
1676 0774 CE18    SFL                 MSB (THE 0) IS SHIFTED AWAY.
1677 0775 6863    SACH ALO
1678 0776 2063    LAC  ALO
1679 0777 4D71    OR   RESID          GET RESIDUAL BITS.
1680 0778 6063    SACL ALO
1681 0779 2167    LAC  BLO,1          GET RID OF EXTRA BIT.
```

```
1682 077A 6067    SACL BLO
1683
1684 077B 2060 CHKSGN LAC  ASIGN          CHECK THE SIGNS.
1685 077C 1064    SUB  BSIGN
1686 077D F680    BZ   ADNOW              IF THEY ARE THE SAME, JUST ADD.
   077E 07AC
1687 077F F380    BLZ  AISNEG
   0780 078F
1688 0781 4062 BISNEG ZALH AHI            DO (|A| - |B|),
1689 0782 4963    ADDS ALO                SINCE B < 0 AND A > 0.
1690 0783 4567    SUBS BLO
1691 0784 4466    SUBH BHI
1692 0785 F680    BZ   CZERO
   0786 079D
1693 0787 F380    BLZ  CNEG
   0788 07A4
1694 0789 686A    SACH CHI
1695 078A 606B    SACL CLO
1696 078B CA00    ZAC
1697 078C 6068    SACL CSIGN
1698 078D FF80    B    NORMAL             GO AND NORMALIZE RESULT.
   078E 07B6
1699 078F 4066 AISNEG ZALH BHI            DO (|B| - |A|),
1700 0790 4967    ADDS BLO                SINCE A < 0 AND B > 0.
1701 0791 4563    SUBS ALO
1702 0792 4462    SUBH AHI
1703 0793 F680    BZ   CZERO
   0794 079D
1704 0795 F380    BLZ  CNEG
   0796 07A4
1705 0797 686A    SACH CHI
1706 0798 606B    SACL CLO
1707 0799 CA00    ZAC

NO$IDT    32020 FAMILY MACRO ASSEMBLER  PC 1.1 86.036    15:12:22  06-20-88
                                  PAGE 0035

1708 079A 6068    SACL CSIGN
1709 079B FF80    B    NORMAL             GO AND NORMALIZE RESULTS.
   079C 07B6
1710
1711 079D CA00 CZERO  ZAC                 HERE, ONLY IF RESULT = 0.
1712 079E 6069    SACL CEXP
1713 079F 6068    SACL CSIGN
1714 07A0 606A    SACL CHI
1715 07A1 606B    SACL CLO
1716 07A2 FF80    B    AROUND             OUTPUT A ZERO.
   07A3 07DA
1717
1718 07A4 CE1B CNEG   ABS                 HERE, IF RESULT IS NEGATIVE.
1719 07A5 686A    SACH CHI
1720 07A6 606B    SACL CLO
1721 07A7 D001    LALK >FFFF
   07A8 FFFF
1722 07A9 6068    SACL CSIGN
1723 07AA FF80    B    NORMAL             GO NORMALIZE RESULT.
   07AB 07B6
1724
1725 07AC 4062 ADNOW  ZALH AHI            IF SIGNS ARE THE SAME, JUST ADD.
1726 07AD 4963    ADDS ALO
1727 07AE 4967    ADDS BLO
1728 07AF 4866    ADDH BHI
1729 07B0 686A    SACH CHI
1730 07B1 606B    SACL CLO
1731 07B2 F080    BV   OVFLOW            DID AN OVERFLOW OCCUR?
   07B3 07C8
1732 07B4 F680    BZ   CZERO             IS RESULT = 0 ?
   07B5 079D
1733
1734                                      * NORMALIZE
1735
1736 07B6 206A NORMAL LAC  CHI            DOES CHI HAVE THE MSB?
```

```
1737 07B7 F680    BZ  LO1
   07B8 07BF
1738 07B9 406A    ZALH CHI                  IF YES, NORMALIZE RESULT.
1739 07BA 496B    ADDS CLO
1740 07BB 4B72    RPT  TTEEN                WILL PERFORM 14 "NORMS"
1741 07BC CEA2    NORM
1742 07BD FF80    B   OUTPUT                GO OUTPUT RESULTS.
   07BE 07D4
1743 07BF 406B LO1    ZALH CLO              HERE IF CLO HAS MSB.
1744 07C0 D000    LRLK AR0,16               OFFSET EXPONENT BY 16.
   07C1 0010
1745 07C2 F380    BLZ  NOFLOW               DID BIT SEARCH CAUSE OVERFLOW?
   07C3 07D1
1746 07C4 4B72    RPT  TTEEN                IF NOT, NORMALIZE RESULT.
1747 07C5 CEA2    NORM
1748 07C6 FF80    B   OUTPUT                GO OUTPUT RESULT.
   07C7 07D4
1749
1750
1751                                        * FINISHED WITH NORMALIZATION
1752
1753                                        * HERE ONLY IF OVERFLOW OCCURRED DURING ADDITION
NO$IDT    32020 FAMILY MACRO ASSEMBLER  PC 1.1 86.036   15:12:22  06-20-88
                                  PAGE 0036

1754
1755
1756 07C8 CE06 OVFLOW  RSXM               RESET SIGN EXTENSION TO SHIFT RIGHT.
1757 07C9 CE19    SFR                     SHIFT RIGHT.
1758 07CA 686A    SACH CHI                STORE NORMALIZED MANTISSA.
1759 07CB 606B    SACL CLO
1760 07CC 2069    LAC  CEXP               DECREMENT EXPONENT.
1761 07CD 006D    ADD  ONE
1762 07CE 6069    SACL CEXP
1763 07CF FF80    B   AROUND              GO OUTPUT RESULTS.
   07D0 07DA
1764                              *
1765                              * OVERLOW OCCURRED DURING BIT SEARCH
1766                              *
1767 07D1 5590 NOFLOW  MAR *-             DECREMENT EXPONENT.
1768 07D2 CE06    RSXM                    RSXM FOR LOGICAL RIGHT SHIFT.
1769 07D3 CE19    SFR                     PERFORM RIGHT SHIFT.
1770
1771
1772                              * TAKE CARE OF EXPONENT & NORMALIZED MANTISSA,
1773                              * THEN OUTPUT RESULTS.
1774
1775

1776 07D4 706E OUTPUT  SAR  AR0,TEMP      HERE AFTER NORMALIZATION.
1777 07D5 686A    SACH CHI                SAVE NORMALIZED MANTISSA.
1778 07D6 606B    SACL CLO
1779 07D7 2069    LAC  CEXP               ADJUST EXPONENT.
1780 07D8 106E    SUB  TEMP
1781 07D9 6069    SACL CEXP
1782        *
1783 07DA CE26 AROUND  RET
1784 07DB
1785 07DB
1786          ************************************************
1787       *
1788       *  THIS IS A FLOATING-POINT MULTIPLICATION  ROUTINE WHICH
1789       *  IMPLEMENTS THE IEEE PROPOSED FLOATING-POINT  FORMAT
1790       *  ON THE TMS32020.
1791       *
1792        ************************************************
1793       *
1794       *  INITIAL FORMAT (ALL 16-BIT WORDS)
1795          *   ------------------
1796       *  |  ALL 0 OR 1 |     ASIGN (0 OR -1)
1797          *   ------------------
1798       *
```

```
1799          *   ------------------
1800        *   |0|.  15 BITS  |      AHI (NORMALIZED)
1801          *   ------------------
1802        *
1803          *   ------------------
1804        *   |0|  9 BITS |--0-|      ALO
1805          *   ------------------
1806        *
1807          *   ------------------
1808        *   |          |      AEXP (-127 TO 128)
1809          *   ------------------

NO$IDT    32020 FAMILY MACRO ASSEMBLER  PC 1.1 86.036    15:12:22  06-20-88
                                        PAGE 0037
            ************************************************
1810        *
1811        *  TO CORRESPOND WITH IEEE FORMAT,
1812        *    INPUT 0.1F * 2 ** (E + 1)
1813        *   INSTEAD OF 1.F * 2 **E, AND SUBTRACT 127 FROM  E.
1814        *
1815        *  THE FINAL FORMAT IS THE SAME AS THE INITIAL  FORMAT
1816        *  EXCEPT THAT FOR CLO WE HAVE:
1817        *
1818          *   ------------------
1819        *   |  16 BITS   |      CLO
1820          *   ------------------
1821        *
1822        *  ALL 16 BITS OF CLO ARE VALID. ANYTHING PAST THESE HAS
1823        *  BEEN TRUNCATED.
1824        *
1825            ************************************************
1826        *
1827        *  WORST CASE (EXCLUDING INITIALIZATION AND  I/O):
1828        *  7.8 MICROSECONDS.
1829        *  THIS TIMING INCLUDES THE NORMALIZATION.
1830        *  WORDS OF PROGRAM MEMORY: 60
1831        *
1832            ************************************************
1833        *
1834        *
1835        *  INITIALIZATION
1836        *
1837        *
1838        *
1839 07DB CE07  FMULT   SSXM                    SET SIGN EXTENSION.
1840 07DC D000     LRLK AR0,0                   CLEAR EXPONENT REGISTER.
   07DD 0000
1841 07DE 5588     LARP 0
1842 07DF D001     LALK >FFFF
   07E0 FFFF
1843 07E1 6074     SACL NEGONE                  NEGONE = -1
1844        *
1845        *
1846        *  BEGIN FLOATING-POINT MULTIPLICATION.
1847        *
1848
1849 07E2 2061  UPMUL   LAC  AEXP               ADD EXPONENTS.
1850 07E3 0065     ADD  BEXP
1851 07E4 6069     SACL CEXP
1852
1853 07E5 3C63     LT   ALO                     FIRST PRODUCT (ALO * BHI)
1854 07E6 3866     MPY  BHI
1855 07E7 CE14     PAC
1856 07E8 6873     SACH THI
1857 07E9 6075     SACL TLO
1858
1859 07EA 3C62     LT   AHI                     SECOND PRODUCT (AHI * BLO)
1860 07EB 3867     MPY  BLO
1861
1862 07EC CE15     APAC                         HAS EFFECT OF (AHI * BLO + ALO * BHI) * 2 ** -15.
1863 07ED CE15     APAC
1864
```

```
1865 07EE 4873    ADDH THI
1866 07EF 4975    ADDS TLO
1867 07F0 6873    SACH THI
1868
1869 07F1 3866    MPY  BHI              (AHI * BHI)
1870 07F2 CE14    PAC
1871 07F3 4973    ADDS THI
1872
1873 07F4 696A    SACH CHI,1            GET RID OF EXTRA SIGN BITS.
1874 07F5 616B    SACL CLO,1
1875
1876 07F6 F580    BNZ  OK               IS RESULT ZERO?
   07F7 07FC
1877 07F8 CA00    ZAC
1878 07F9 6069    SACL CEXP
1879 07FA FF80    B    SETSIN
   07FB 0805
1880
1881 07FC 406A OK ZALH CHI              NORMALIZE AND WRAP UP.
1882 07FD 496B    ADDS CLO
1883 07FE CEA2    NORM
1884 07FF 686A    SACH CHI
1885 0800 606B    SACL CLO
1886 0801 706E    SAR  AR0,TEMP
1887 0802 2069    LAC  CEXP
1888 0803 106E    SUB  TEMP
1889 0804 6069    SACL CEXP
1890
1891 0805 4160 SETSIN ZALS ASIGN        WHAT IS SIGN OF RESULT?
1892 0806 4C64    XOR  BSIGN
1893 0807 F580    BNZ  NEG
   0808 080D
1894 0809 CA00    ZAC
1895 080A 6068    SACL CSIGN
1896 080B FF80    B    OUTMUL
   080C 080F
1897 080D 2074 NEG    LAC  NEGONE
1898 080E 6068    SACL CSIGN
1899 080F
1900 080F F080 OUTMUL BV   COV
   0810 0811
1901 0811 CE26 COV    RET
1902 0812
1903 0812
1904 0812
1905 0812 D400 FLFX   LRLK AR4,FIXPT
   0813 007C
1906 0814 55A0    MAR  *+
1907 0815 2080    LAC  *
1908 0816 D003    SBLK 3
   0817 0003
1909 0818 D004    ANDK >8000
   0819 8000
1910 081A F680    BZ   UNCH1
   081B 0827
1911 081C 20A0    LAC  *+
1912 081D D003    SBLK 3
```



```
   081E 0003
1913 081F CE27    CMPL
1914 0820 606E    SACL TEMP
1915 0821 20AC    LAC  *+,0,AR4
1916 0822 4B6E    RPT  TEMP
1917 0823 CE19    SFR
1918 0824 6080    SACL *,0
```

```
1919 0825 FF8C    B   SIGN,*,AR4
   0826 082A
1920 0827
1921 0827 55A0 UNCH1   MAR  *+
1922 0828 20AC    LAC  *+,0,AR4
1923 0829 6080    SACL *
1924 082A
1925 082A D300 SIGN    LRLK AR3,>9F4
   082B 09F4
1926 082C 558B    LARP 3
1927 082D 208C    LAC  *,0,AR4
1928 082E F580    BNZ  ESUB
   082F 0835
1929 0830 D400    LRLK AR4,FIXPT
   0831 007C
1930 0832 2080    LAC  *
1931 0833 CE27    CMPL
1932 0834 6080    SACL *
1933 0835
1934 0835 CE26 ESUB    RET
1935 0836
1936 0836
1937 0836
1938 0836 CA03 FXFL'   LACK 3
1939 0837 6065    SACL BEXP
1940 0838 207C    LAC  FIXPT
1941 0839 D004    ANDK >8000
   083A 8000
1942 083B F680    BZ   NEG1                MSB=0 IS NEGATIVE
   083C 0844
1943 083D
1944 083D CA00    ZAC                      MSB=1 IS POSITIVE
1945 083E 6064    SACL BSIGN
1946 083F 207C    LAC  FIXPT
1947 0840 CE27    CMPL                     ITS MAGNITUDE IS A COMP.
1948 0841 607C    SACL   FIXPT
1949 0842 FF80    B    MAG
   0843 0846
1950 0844
1951 0844 207D NEG1    LAC  NEGS
1952 0845 6064    SACL BSIGN
1953 0846
1954 0846 207C MAG     LAC  FIXPT
1955 0847 D004    ANDK >4000
   0848 4000
1956 0849 F680    BZ   NEXT1
   084A 084F
1957 084B 207C    LAC  FIXPT               (INFX)=01XX ...X ; (INEXP)=3
1958 084C 6066    SACL BHI
1959 084D FF80    B    FINCVT
```



```
   084E 08E9
1960 084F
1961 084F 207C NEXT1   LAC  FIXPT
1962 0850 D004    ANDK >2000
   0851 2000
1963 0852 F680    BZ   NEXT2
   0853 085D
1964 0854 2065    LAC  BEXP                (FIXPT)=001X ...X
1965 0855 D003    SBLK 1                   (AEXP)=2
   0856 0001
1966 0857 6065    SACL BEXP
1967 0858 207C    LAC  FIXPT
1968 0859 CE18    SFL
1969 085A 6066    SACL BHI
1970 085B FF80    B    FINCVT
   085C 08E9
1971 085D
1972 085D 207C NEXT2   LAC  FIXPT
```

```
1973 085E D004    ANDK >1000
   085F 1000
1974 0860 F680    BZ  NEXT3
   0861 086C
1975 0862 2065    LAC  BEXP              (FIXPT)=0001 ...X
1976 0863 D003    SBLK 2                 (AEXP)=1
   0864 0002
1977 0865 6065    SACL BEXP
1978 0866 207C    LAC  FIXPT
1979 0867 CB01    RPTK   1
1980 0868 CE18    SFL
1981 0869 6066    SACL BHI
1982 086A FF80    B   FINCVT
   086B 08E9
1983 086C
1984 086C 207C NEXT3  LAC  FIXPT
1985 086D D004    ANDK >0800
   086E 0800
1986 086F F680    BZ  NEXT4
   0870 087B
1987 0871 2065    LAC  BEXP              (FIXPT)=0000 1XXX
1988 0872 D003    SBLK 3                 (AEXP)=0
   0873 0003
1989 0874 6065    SACL BEXP
1990 0875 207C    LAC  FIXPT
1991 0876 CB02    RPTK   2
1992 0877 CE18    SFL
1993 0878 6066    SACL BHI
1994 0879 FF80    B   FINCVT
   087A 08E9
1995 087B
1996 087B
1997 087B 207C NEXT4  LAC  FIXPT
1998 087C D004    ANDK >0400
   087D 0400
1999 087E F680    BZ  NEXT5
   087F 088A
2000 0880 2065    LAC  BEXP              (FIXPT)=0000 01XX
2001 0881 D003    SBLK 4                 (AEXP)=-1
```



```
   0882 0004
2002 0883 6065    SACL BEXP
2003 0884 207C    LAC  FIXPT
2004 0885 CB03    RPTK   3
2005 0886 CE18    SFL
2006 0887 6066    SACL BHI
2007 0888 FF80    B   FINCVT
   0889 08E9
2008 088A
2009 088A
2010 088A 207C NEXT5  LAC  FIXPT
2011 088B D004    ANDK >0200
   088C 0200
2012 088D F680    BZ  NEXT6
   088E 0899
2013 088F 2065    LAC  BEXP              (FIXPT)=0000 001X
2014 0890 D003    SBLK 5                 (AEXP)=-2
   0891 0005
2015 0892 6065    SACL BEXP
2016 0893 207C    LAC  FIXPT
2017 0894 CB04    RPTK   4
2018 0895 CE18    SFL
2019 0896 6066    SACL BHI
2020 0897 FF80    B   FINCVT
   0898 08E9
2021 0899
2022 0899
2023 0899 207C NEXT6  LAC  FIXPT
2024 089A D004    ANDK >0100
```

```
   089B 0100
2025 089C F680   BZ  NEXT7
   089D 08A8
2026 089E 2065   LAC  BEXP            (FIXPT)=0000 0001 X..X
2027 089F D003   SBLK 6               (AEXP)=-3
   08A0 0006
2028 08A1 6065   SACL BEXP
2029 08A2 207C   LAC  FIXPT
2030 08A3 CB05   RPTK   5
2031 08A4 CE18   SFL
2032 08A5 6066   SACL BHI
2033 08A6 FF80   B  FINCVT
   08A7 08E9
2034 08A8
2035 08A8
2036 08A8 207C NEXT7  LAC  FIXPT
2037 08A9 D004   ANDK >0080
   08AA 0080
2038 08AB F680   BZ  NEXT8
   08AC 08B7
2039 08AD 2065   LAC  BEXP            (FIXPT)=0000 0000 1XXX
2040 08AE D003   SBLK 7               (AEXP)=-4
   08AF 0007
2041 08B0 6065   SACL BEXP
2042 08B1 207C   LAC  FIXPT
2043 08B2 CB06   RPTK   6
2044 08B3 CE18   SFL
2045 08B4 6066   SACL BHI

NO$IDT   32020 FAMILY MACRO ASSEMBLER  PC 1.1 86.036   15:12:22  06-20-88
                                  PAGE 0042

2046 08B5 FF80   B  FINCVT
   08B6 08E9
2047 08B7
2048 08B7
2049 08B7 207C NEXT8  LAC  FIXPT
2050 08B8 D004   ANDK >0040
   08B9 0040
2051 08BA F680   BZ  NEXT9
   08BB 08C6
2052 08BC 2065   LAC  BEXP               (FIXPT)=0000 0000 01XX
2053 08BD D003   SBLK 8                  (AEXP)=-5
   08BE 0008
2054 08BF 6065   SACL BEXP
2055 08C0 207C   LAC  FIXPT
2056 08C1 CB07   RPTK   7
2057 08C2 CE18   SFL
2058 08C3 6066   SACL BHI
2059 08C4 FF80   B  FINCVT
   08C5 08E9
2060 08C6
2061 08C6
2062 08C6 207C NEXT9  LAC  FIXPT
2063 08C7 D004   ANDK >0020
   08C8 0020
2064 08C9 F680   BZ  NEXT10
   08CA 08D5
2065 08CB 2065   LAC  BEXP               (FIXPT)=0000 0000 001X
2066 08CC D003   SBLK 9                  (AEXP)=-6
   08CD 0009
2067 08CE 6065   SACL BEXP
2068 08CF 207C   LAC  FIXPT
2069 08D0 CB08   RPTK   8
2070 08D1 CE18   SFL
2071 08D2 6066   SACL BHI
2072 08D3 FF80   B  FINCVT
   08D4 08E9
2073 08D5
2074 08D5
2075 08D5 207C NEXT10 LAC  FIXPT
2076 08D6 D004   ANDK >0010
```

```
   08D7 0010
2077 08D8 F680    BZ  NEXT11
   08D9 08E4
2078 08DA 2065    LAC  BEXP                  (FIXPT)=0000 0000 0001
2079 08DB D003    SBLK >A                    (AEXP)=-7
   08DC 000A
2080 08DD 6065    SACL BEXP
2081 08DE 207C    LAC  FIXPT
2082 08DF CB09    RPTK   9
2083 08E0 CE18    SFL
2084 08E1 6066    SACL BHI
2085 08E2 FF80    B   FINCVT
   08E3 08E9
2086 08E4
2087 08E4 CA00  NEXT11  ZAC
2088 08E5 6065    SACL BEXP
2089 08E6 6066    SACL BHI

NO$IDT    32020 FAMILY MACRO ASSEMBLER  PC 1.1 86.036    15:12:22  06-20-88
                                  PAGE 0043

2090 08E7 6067    SACL BLO
2091 08E8
2092 08E8
2093 08E8 CE26    RET
2094 08E9
2095 08E9
2096 08E9 CA00  FINCVT  ZAC
2097 08EA 6067    SACL BLO
2098 08EB CE26    RET
2099 08EC
2100 08EC
2101 08EC
2102 08EC 558C  ISR    LARP 4
2103 08ED D400    LRLK AR4,FIXPT
   08EE 007C
2104 08EF E280    OUT  *,2                   *AR4 (AR4)=FIXPT
2105 08F0 FA80  ALOOP   BIOZ ALOOP1
   08F1 08F4
2106 08F2 FF80    B   ALOOP
   08F3 08F0
2107 08F4
2108 08F4 8280  ALOOP1  IN   *,2
2109 08F5 CE00    EINT
2110 08F6 CE26    RET
2111 08F7
2112 08F7
2113          END
NO ERRORS, NO WARNINGS

NO$IDT    32020 FAMILY MACRO ASSEMBLER  PC 1.1 86.036    15:12:52  06-20-88
                                  PAGE 0044

0001 0000
NO ERRORS, NO WARNINGS
```

APPENDIX E

This appendix contains the paper on "Model- Reference Adaptive Control using an FIR controller" which was represented at the IFAC Work shop on Robust Adaptive Control on 22-24 August 1988 at Newcastle, Australia.

MODEL REFERENCE ADAPTIVE CONTROL USING AN F.I.R. CONTROLLER

H.M.T. Tran and G. Thé

Department of Electrical Engineering, University of Tasmania,
GPO Box 252C, Hobart, Australia

Abstract. This paper discusses the theory and implementation of an adaptive controller which can track plant with time-varying parameters as well as time-varying transport delay. It also allows for closed-loop pole placement and, because there are no restrictions on closed-loop zeros, good set-point tracking can be achieved. The scheme is based on the well-known least-mean-square algorithm to generate a finite impulse response (F.I.R.) controller and can therefore be implemented readily using one of the new generation signal processor chips.

Keywords. Adaptive control; time-varying systems; least-mean-square algorithm; digital signal processor.

INTRODUCTION

Self-tuning controllers based on pole-zero placement design have been developed for many years (Astrom and Wittenmark, 1980; Wellstead, Prager and Zanker, 1979; Wellstead and Sanoff, 1981). The motivation for the pole-placement self-tuning principle stems from the observation that a control engineer can easily relate pole locations to closed loop transient performance. However self-tuning controllers suffer from a number of weaknesses, notably:

(i) zero-placement is difficult to achieve due to the fact that cancellation of plant zeros outside the unit circle is not possible. Therefore the controller generally has poor servo-tracking properties without the use of an adaptive feed-forward compensator.

(ii) large changes in plant parameters frequently result in excessive changes in the control input and when control limits are imposed, as is usually the case in a practical situation, the self-tuning algorithm may loose control.

(iii) identification of a plant with variable deadtime requires over-parameterization of the model with the inherent sensitivity problems.

A new approach for the design of controllers and self-tuning regulators to allow closed-loop pole placement with better set-point tracking has recently been reported (Liu and Sinha, 1987; Puthenpura and Macgregor, 1987). However it is restricted to applications where the reference signal variations are Laplace-transformable and known in advance.

In this paper we look at the well-known least-mean-square (LMS) algorithm (Widrow and Stearns, 1985) as normally applied to the adaptive equalisation of communication channels and apply the same principle to effect a model reference adaptive control. Since its inception bythe Bell System group (Gersho, 1969; Lucky, 1967; Sondhi, 1967)., the LMS algorithm has always been implemented in a hybrid mode, i.e. the signal in analogue form passes through a series of delay lines and the taps are adjusted digitally. However, the advent of high-speed microprocessors such as the 8086/80186/80286 and their numeric co-processors made it possible to implement a 30-tap F.I.R. adaptive equaliser digitally in real time (Tran, 1986). Our investigations are motivated by the introduction of the latest high-speed signal processing chip, the Texas Instrument TMS320C25, which has an ultra high-speed multiplier and barrel shifter. Whereas self-tuning algorithm requires repeated divisions as well as multiplications, the LMS algorithn requires only repeated multiplications and therefore the TMS320 family of signal processors lend themselves ideally for this application. The TMS320C25 can carry out a 64-tap F.I.R. filtering at a sampling period of 8 microsecond (125 nanosec. per tap).

LMS ADAPTIVE EQUALISATION/IDENTIFICATION

Figure 1 shows the schematic configuration of an LMS adaptive equaliser, where it is desired to make the equaliser output, c_k, follow the desired output, d_k. By defining

$$X_k = [x_{k-0}\ x_{k-1}\ \cdots\ x_{k-L+1}]^T \qquad (1)$$

$$W_k = [w_0\ w_1\ \cdots\cdots\ w_{L-1}]^T \qquad (2)$$

and using the subscript k to denote the k^{th} sampled value, we define the error as:

$$e_k = d_k - X_k^T W_k \qquad (3)$$

and, because the mean-square-error cost function ($E(e_k^2)$) is a quadratic function of the weights, W_k we can use the following recursive equation

$$W_{k+1} = W_k + 2\mu e_k X_k \qquad (4)$$

to seek for the minimum point. Widrow and Stearns (1985) showed that this method is stable and convergent if and only if

$$0 < \mu < 1/\lambda_{max} \qquad (5)$$

where λ_{max} is the largest eigen value of the auto-correlation matrix of the input signal

$$R = E\{x_k x_k^T\} \qquad (6)$$

The LMS adaptive filter can also be used to identify the F.I.R. parameters of an unknown plant. This method is shown in figure 2 and it is shown in the appendix that if the input is an un-correlated sequence, then the filter taps will converge towards the true plant F.I.R. parameters.

OPEN-LOOP MODEL REFERENCE ADAPTIVE CONTROL

Figure 3 illustrates how the LMS algorithm can be used in a model-reference adaptive control in an open loop fashion (Widrow and Stearns, (1985). Note that the LMS algorithm is used twice per iteration. First, the LMS algorithm is used to determine the F.I.R. plant model approximation, $\hat{P}(z)$, of the plant, P(z). Next, this $\hat{P}(z)$ is used to pre-filter the input x_k before it is used by the LMS algorithm to adjust the F.I.R. controller parameters. The reference model, M(z), must have at least as long a time delay as the maximum time delay anticipated in the plant, P(z). Given that the F.I.R. filter has a sufficient number of taps, when the adaptive process converges, the output of the controller-plant combination will match that of the reference model in a least-mean-square sense.

Such an adaptive control scheme was simulated using a 40-tap F.I.R. filter to identify the plant and to realise the controller. A pseudo random binary sequence with a standard deviation of 1.34 units was used as the dither signal, while the plant noise standard deviation was kept at 0.29 units. To test the ability of the system to track plant variation the plant pulse transfer function was switched from

$$P_1(z) = \frac{0.22361(1+2z^{-1}+z^{-2})z^{-D_p}}{1-0.74776z^{-1}+0.64222z^{-2}} \qquad (7)$$

to

$$P_2(z) = \frac{2.4(z^{-1}-0.8z^{-2})z^{-D_p}}{1-0.1z^{-1}-0.42z^{-2}} \qquad (8)$$

with $D_p = 3$, while the reference model pulse transfer function was kept at

$$M(z) = \frac{0.03356(1+2z^{-1}+z^{-2})z^{-D_m}}{1-1.5302z^{-1}+0.66443z^{-2}} \qquad (9)$$

with $D_m = 5$. Figure 4 shows the unit step response of the compensated plant and the model.

Next the system was tested in the presence of time delay variation. The plant pulse transfer function was given by equation (8) but with the deadtime switched from $D_p = 3$ to $D_p = 7$ while the model pulse transfer function was given by:

$$M(z) = \frac{0.25z^{-11}}{1-z^{-1}+0.25z^{-2}} \qquad (10)$$

Figure 5 shows the unit step responses of the compensated plant and the model.

CLOSED LOOP MODEL REFERENCE ADAPTIVE CONTROL

Figure 6 shows a scheme for the closed loop model reference adaptive control. The desired closed loop behaviour of the plant must be represented by the model, which in the simulation was set to

$$M(z) = \frac{.0249z^{-5}+.0147z^{-6}}{1-1.1618z^{-1}+.2015z^{-2}+.0249z^{-5}+0.147z^{-6}} \qquad (11)$$

while the plant transfer function was switched from

$$P_1(z) = \frac{.00301z^{-1}+.01088z^{-2}+.00247z^{-3}}{1-2.59233z^{-1}+2.27083z^{-2}-.67032z^{-3}} \qquad (12)$$

to

$$P_2(z) = \frac{.00842z^{-4}+.02500z^{-5}+.004619z^{-6}}{1-2.0043z^{-1}+1.34345z^{-2}-.301194z^{-3}} \qquad (13)$$

A square wave signal plus p.r.b.s. dither was applied to the reference input and the plant and model responses are shown in figure 7. Figure 8 shows that there is no excessive control action present in the input to the plant at the instants changes in plant parameters take place.

ADAPTIVE CONTROL WITH SIGNAL TRACKING PROPERTIES

Our purpose here is to show how the LMS adaptive controller of figure 6 can be used to give the system good setpoint tracking characteristics. By selecting a reference model whose gain is unity over all frequencies of interest present in the input signal, the LMS algorithm will adjust the F.I.R. controller parameters so that tracking errors are minimised. The selection of the transfer function must be carried out judicially, because increasing the closed loop system bandwidth will be accompanied by excessive control effort if the plant bandwidth is not adequate.

For this demonstration we used the same test signal as used by Liu and Sinha (1987):

$$e_k = \exp(-0.04k)\sin(0.2k) \qquad (14)$$

The plant pulse transfer function is given by:

$$P(z) = \frac{.0601z^{-4}-.1012z^{-5}}{1-1.6457z^{-1}+.6703z^{-2}} \qquad (15)$$

The desired closed loop behaviour of the plant is represented by the model

$$M(z) = \frac{.02985z^{-5}}{1-1.4891z^{-1}+.5488z^{-2}} \qquad (16)$$

so that, while obtaining good tracking of the reference signal, the closed-loop poles are placed at $z_1 = 0.81908$ and at $z_2 = 0.67002$. The setpoint tracking behaviour of the model-reference adaptive control system using an LMS adaptive scheme is shown in figure 9. Simulation results show that this scheme handles reference signal following problems well even in the presence of variations in plant parameters and variations in time delay. This may be compared with the behaviour of a self-tuning controller with pole-zero placement of the error transfer function (Liu and Sinha, 1987), shown in figure 10. However in the latter case the reference signal must be known in advance and be Laplace transformable, and in addition the time delay of the plant needs to be precisely determined.

LMS ADAPTIVE CONTROL USING THE TMS320C20

The LMS closed-loop model adaptive control can be easily implemented using a TMS320 digital signal processor. In this section a TMS320C20 Software Development System (SWDS) plugged into a IBM-AT backplane is used for plant simulation and to implement a model reference adaptive control using a F.I.R. controller. The software is composed of five modules: initialisation routine, floating point conversion routine, plant identification routine, LMS controller routine and fixed point conversion routine.

The initialisation routine disables/enables interrupts, loads data memory with system parameters and initialises the registers. The floating point conversion routine converts an analogue sample into a 23-bit mantissa and an 8 bit exponent (Crowell, 1985). The plant identification routine samples the plant input/output and update the F.I.R. plant model. The LMS controller routine prefilters the input signal and adjusts the F.I.R. controller parameters. The fixed point conversion routine converts the controller output into fixed point format and sends it to the D/A converter. All routines are written in TMS320C20 assembly language. The plant pulse transfer function is given by:

$$P(z) = \frac{.022361(1+2z^{-1}+z^{-2})}{1-.77478z^{-1}+.66443z^{-2}} \qquad (17)$$

The desired closed loop behaviour of the plant is represented by the model

$$M(z) = \frac{.03356(1+2z^{-1}+z^{-2})}{1-1.5302z^{-1}+.66443z^{-2}} \qquad (18)$$

Figure 11 shows the unit step responses of the compensated plant and the model. The complete model reference adaptive control program for this particular example uses a 40-tap F.I.R. controller and takes 7 msec per iteration. This can be reduced to 3.0 msec if a TMS320C25 chip were used. Further work is currently proceeding to test the algorithm in a full hybrid mode.

CONCLUSIONS

The theory and implementation of a special type of model reference adaptive controller in the light of the latest developments in signal processing technology has been presented. The method is computationally simple. This paper shows by simulation studies that the LMS algorithm can be effectively used in a model reference adaptive F.I.R. controller to control a plant in a closed loop configuration. It has excellent adaptation capabilities in the presence of both parameters and dead time variations. It is not necessary to know a priory the time delay of the plant, although some knowledge of the plant characteristics would be helpful when choosing the model dead time and the number of taps on the F.I.R. filter. By a suitable choice of the reference model, it can be designed to have good servo-tracking properties. The main drawback of the scheme is the long time required before the taps converge to the best values. However the adaptive F.I.R. controller should have great potential in application where the dead time and parameter variations are relatively gradual, as it is generally known that F.I.R. filters are more robust than their recursive I.I.R. counterpart.

ACKNOWLEDGEMENT

The authors wish to thank Texas Instruments for providing the TMS320 Software Development System. The financial and technical support given by the University of Tasmania is also gratefully acknowledged.

REFERENCES

Astrom, K.J., and B. Wittenmark (1980). Self-tuning controller based on pole-zero placement. Proc. I.E.E. (D), 1980, Vol. 127. No. 3, pp. 120-130.

Crowell, C.D. (1985). Floating Point Arithmatic with the TMS320C20. Texas Instrument Digital Signal Processing Application Report.

Gersho, A (1969). Adaptive equalisation of highly dispersive channels for data transmission. Bell System Tech. J. 1969, Vol. 48. pp. 55-70.

Liu, L., and Sihha, N.K. (1987). Quadratic-criterion self-tuning control with error transfer-function pole/zero placement. Int. J. of Control, 1987, Vol. 45, No. 2, pp. 485-502.

Lucky, R.W. (1967). Technique for adaptive equalisation of digital communication systems. Bell System Tech. J. 1967, Vol. 46. pp. 285-286.

Puthenpura, S.C. and MacGregor,J.F. (1987). Pole-zero placement controllers and self-tuning regulators with better set-point tracking. Proc. I.E.E. Vol. 134, Pt. D, No. 1, January 1987, pp. 26-30.

Sondhi, M.M. (1967). An Adaptive Echo Canceller. Bell System Tech. J. 1967, Vol. 46, pp. 497-511.

Texas Instrument Inc. TMS320C25 Software Development System.

Tran, H.M.T. (1986). Hardware/software implementation of an adaptive F.I.R. controller. Honours project thesis, 1986, Department of Electrical Engineering, University of Tasmania.

Wellstead, P.E., D. Prager and P. Zanker. (1979). Pole assignment self-tuning regulator. Proc. I.E.E. (D), 1979, Vol. 126, No.8, pp.781-787.

Wellstead, P.E., and Sanoff, S.P. (1981). Extended self-tuning algorithm. Int. J. of Control 1981, Vol. 34, No. 3, pp. 433-355.

Widrow B., and Stearns, S.D. (1985). Adaptive Signal Processing. Prentice Hall Inc.

APPENDIX

From equation (3) the mean square error is given by:

$$J = E\{e_k^2\} = E\{d_k^2\} - 2P_k^T W_k + W_k^T R W_k \qquad (19)$$

where the cross correlation vector between input and the desired response is defined as:

$$P_k = E\{d_k x_k,\ d_k x_{k-1},\ \ldots\ d_k x_{k-L+1}\}^T \qquad (20)$$

and the input auto correlation matrix is given by:

$$R_k = E\{X_k X_k^T\} \qquad (21)$$

To obtain the minimum of the mean square error we set the gradient of J to zero:

$$\text{grad } J = 2R_k W_k - 2P_k = 0 \qquad (22)$$

If the input x_k is assumed to be a stationary ergodic white sequence with variance σ^2, then the subscript k can be dropped and we have:

$$R = \sigma^2 I \qquad (23)$$

$$P = \sigma^2 G \qquad (24)$$

where the plant sampled unit impulse response is given by:

$$G = [g_0,\ g_1,\ g_2,\ \ldots\ g_{L-1}]^T \qquad (25)$$

Equation (22) then becomes:

$$W = G \qquad (26)$$

which shows that the F.I.R. controller matches with the plant unit impulse response.

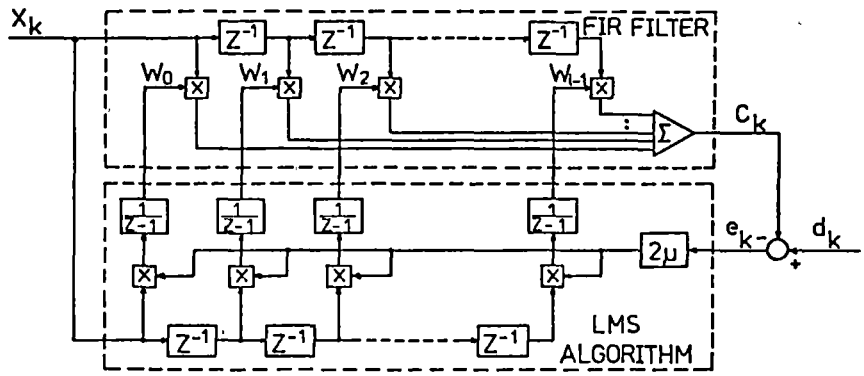


Figure 1. L.M.S. adaptive equaliser.

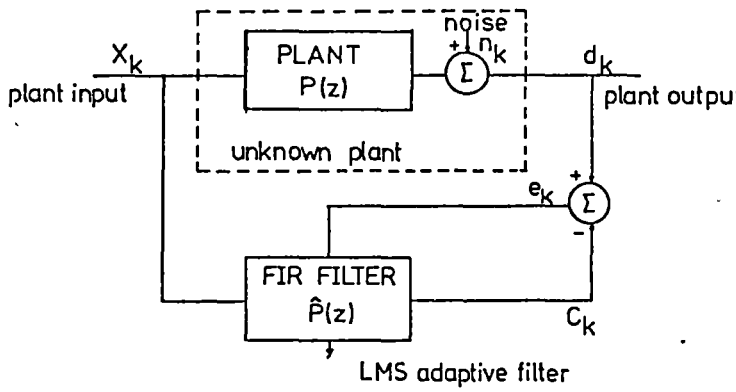


Figure 2. Plant identification.

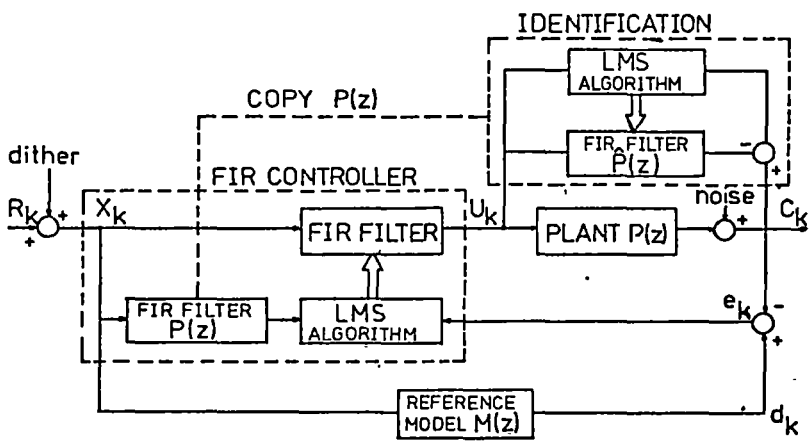


Figure 3. Open loop model reference adaptive control.

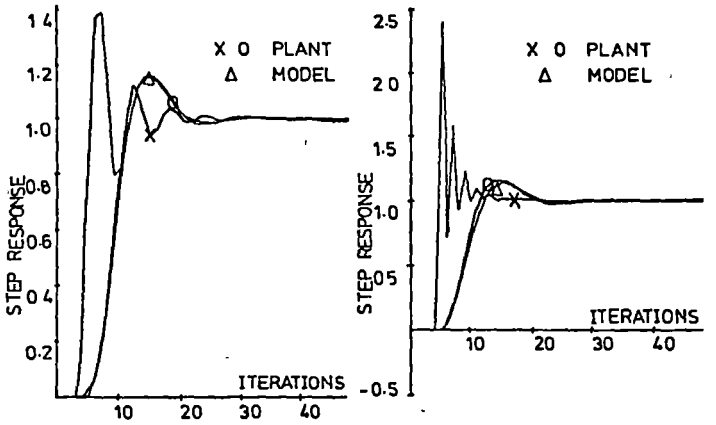


Figure 4. Unit step responses of compensated plant with parameter varintions.

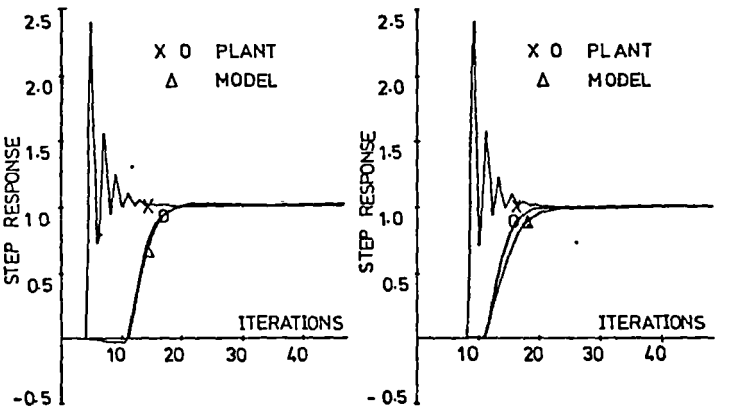


Figure 5. Unit step responses of compensated plant with delay variation.

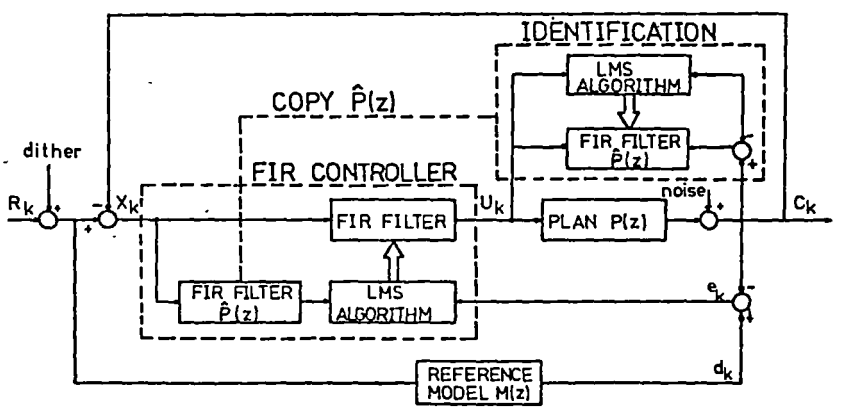


Figure 6. Closed loop model reference adaptive control.

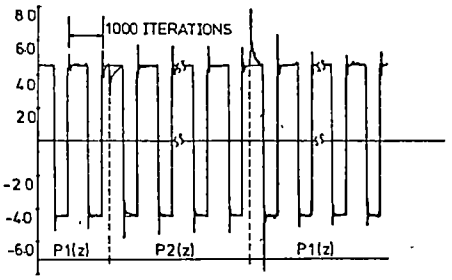


Figure 7. Model and plant responses due to a square wave reference input.

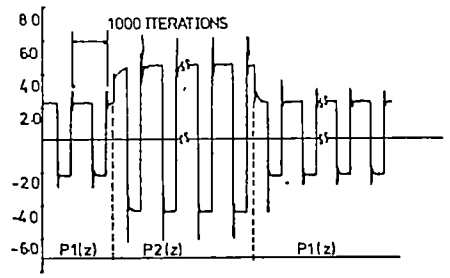


Figure 8. Plant control input due to a square wave reference input.

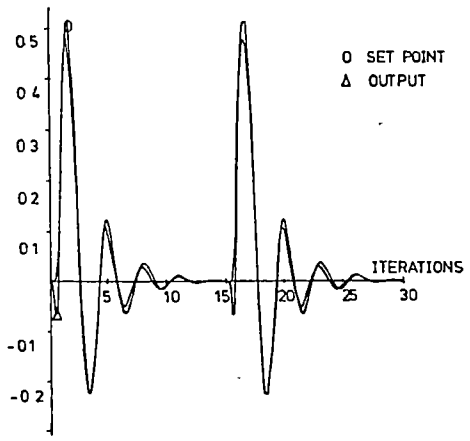


Figure 9. Set point tracking using an F.I.R. model reference adaptive controller.

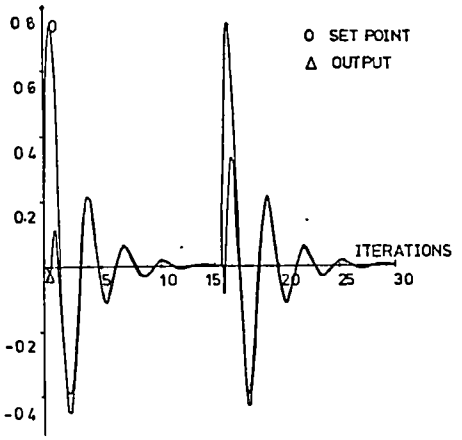


Figure 10. Set point tracking using a self-tuning controller with pole/zero movement in error transfer function. (Liu and Sinha, 1987).

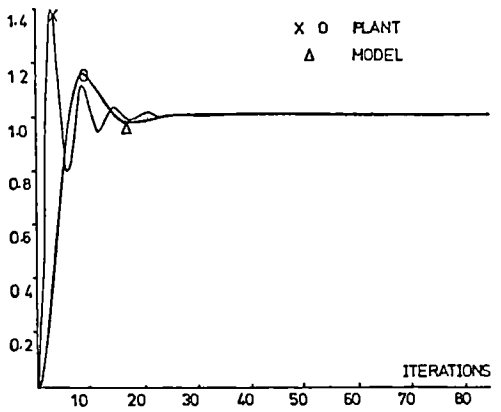


Figure 11. Plant and model unit step responses.

REFERENCES

Astrom K.J. Computer controlled systems: Theory and design. Prentice-hall (1984).

Astrom,K.J. , U.Borisson,L.Ljung and B. Wittenmark (1977). Theory and applications of self-tuning regulators.Automatica,No.13,pp. 457-476.

Bokor,J., and Keviczky,L.,(1985) Recursive structre, parameter and delay time estimation using ESS representations. Proc. 7th IFAC Symp. on Identification and System Parameter Estimation, York,U.K.

Dexter A. L. (Sept., 1983). Self-tuning control algorithm for single-chip microcomputer implementation. IEE Proceedings, Vol. 130, Pt. D,No. 5.

. Fromme G. (1982). Self optimizing controller employing microprocessor. ETG-Fachberichte Microelectronics in Power Elec. and Elec. Drives Conf. Rec., pp.117-126, 1982.

Gawthrop, P. J., and Nihtila, M. T., 1985, Syst. Control Lett., .

Gersho A.,(1969) Adaptive equalisation of highly dispersive channels for data transmission. Bell System Tech.J. ,Vol.48. pp.55-70.

Haykin S. Adaptive filter theory Prentice-hall (1986).

Isermann,R. and Lachmann K.H. (1985) Parameter-adaptive Control with Configuration Aids and Supervision Functions. Automatica, No.6, pp.625-638.

Kaminskas, V., 1979, Parameter estimation in systems with time delay and closed loop systems. IFAC Symp. on Identification and System Parameter Estimation, Darmstadt, Vol. 1, pp. 669-677.

Kurz,H., and Goedecke,W.,1981,Automatica,No.17,pp245.

Landau,I.D.(1979). Adaptive control- The Model Reference Approach. Marcel Dekker,New York.

Liu L. and Sinha N.K. (1987). Quadratic-criterion self-tuning control with error transfer-function pole/zero placement. Int. J. Control, 1987, vol. 45 no. 2, 485-502.

Lucky,R.W.(1967). Technique for adaptive equalisation of digital communication systems. Bell System Tech, J., Vol.46. pp.285-286.

Mayhan R.J. Discrete and continuous-time linear systems. Addison-Wesley Series in Electrical Engineering (1984).

M'Sadd M., Ortega R. and Landau I.D. (1985) Adaptive Controllers for Discrete-Time Systems with Arbitrary Zeros:An Overview. Automatica,No.4,pp.413-423.

Ogata k. Modern control engineering. Prentice-hall (1970).

Pupeikis, R., (1985) Recursive estimation of the parameters of linear systems with time delay. Proc.7th IFAC Symp. on Identification and System Parameter Estimation, York,U.K.

Sheirah M.A., Malik O.P. and Hope G.S. (Feb., 1982). Self-tuning microprocessoruniversal controller. IEEE Transaction on Industrial Electronics, Vol. IE-29, No. 1.

Sondhi M.M.,(1967) An Adaptive Echo Canceller. Bell System Tech.J. , Vol.46,pp.497-511.

Speedy C.B.,Brown R.F., and Goodwin G.C. (1970), Control Theory:Identification and optimal control. Oliver and Boyd. Edinburgh

TMS320C20 User's Guide Preliminary Digital Signal Processor Products, Texas Instruments.

Tran H.M.T. (1986) Hardware/Software implementation of an adaptive F.I.R. controller. Honours project thesis, Department of Electrical Engineering , University of Tasmania.

Widrow B., and Stearns S.D., (1985) Adaptive Signal Processing. Prentice Hall Inc.

Widrow, B. and M.E.Hoff,JR. (1960). Adaptive switching circuits , IRE Wescon Conv. Rec., Part 4,pp.96-104.