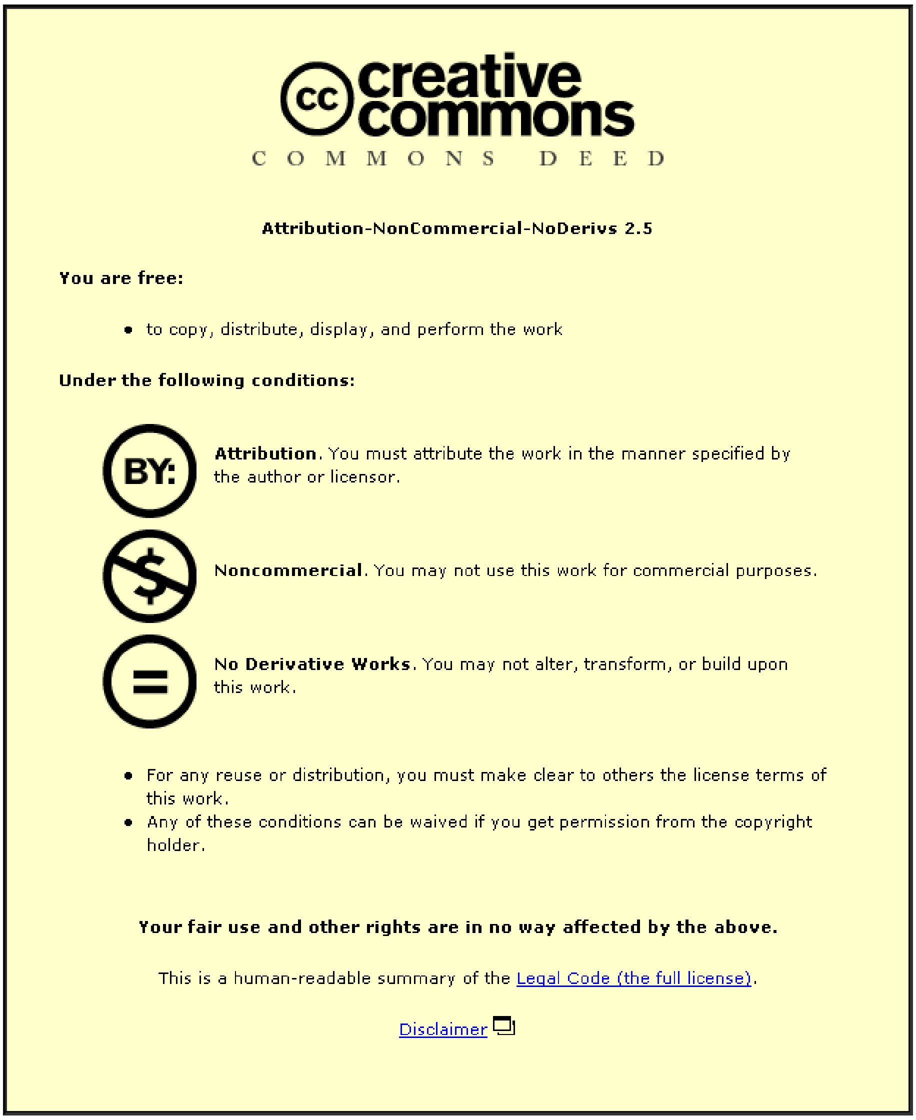


This book was bound by
Badminton Press
18 Half Croft, Syston, Leicester, LE7 8LD
Telephone: Leicester (0533) 602918.

ADAPTIVE ADJUSTMENT OF RECEIVER FOR DISTORTED DIGITAL SIGNALS

by

S F HAU, BSc, MSc

A Doctoral Thesis submitted in partial fulfilment of the
requirements for the award of
Doctor of Philosophy
of the Loughborough University of Technology

June 1986

Supervisor: Professor A P Clark
Department of Electronic and Electrical Engineering

CONTENTS

ABSTRACT

This report describes the investigation which has led to the development of a novel technique for the adaptive adjustment of the receiver in a digital data-transmission system, where the latter operates with additive noise and severe intersymbol interference in the received signal. This technique is suitable both for the adjustment of a conventional non-linear (decision-feedback) equaliser, and also for the adjustment of a linear feedforward transversal filter that is employed ahead of a near-maximum-likelihood detector. In the latter case, the technique provides, in addition, an estimate of the sampled impulse-response of the channel and filter, to give the information on the received signal needed by the detector. The adaptive system requires an estimate of the sampled impulse-response of the channel and it involves finding the roots (zeros) of the z-transform of the sampled impulse-response that lie outside the unit circle.

The aim of the investigation has been to develop a root-finding algorithm which would be used to find some or all of the roots, with moduli greater than unity, of a high-degree (>20) complex polynomial. The first part of the report describes the study of an on-line system, where the receiver operates on a continuous stream of received samples subject to both intersymbol interference and noise, and it presents the results of a large number of tests using computer simulation which show the root-finding capability and vulnerability of the system to white Gaussian noise. The results of the tests point to the potential advantages of an off-line system, which is much simpler to implement and requires less computation. The modified technique operates solely and directly on the estimate of sampled impulse-response of the channel, which must again be provided at the receiver. The report gives the results of computer-simulation tests measuring the speed and accuracy of two different arrangements of the system, for different channels, and it also studies the effects of double roots and roots just outside the unit circle.

The off-line system is suitable for a 16-point QAM system operating at 9600 bit/s over the British public switched telephone network, and with slight modification, it can also be used for a 64-point QAM system operating at 19200 bit/s.

ACKNOWLEDGEMENTS

The author wishes to thank his supervisor, Professor A P Clark, for his help and guidance.

The financial support of SERC is gratefully acknowledged.

LIST OF PRINCIPAL SYMBOLS

c	=	positive real constant in range 0 to 1
a_h	=	coefficient of z^{-h} in $Y(z)$ $Dm(z)$ (eqn. 4.12).
$A_i(z)$	=	z-transform of one-tap feedback transverse filter in Figure 2.3 (equation 2.20)
$B_i(z)$	=	z-transform of two-tap feedforward transverse filter in Figure 2.2 (equation 2.18).
$C_1(z)$	=	$A_k(z)B_k(z)$ (equation 2.22)
$D(z)$	=	z-transform of ideal adaptive linear transversal filter (equation 2.10)
$D_h(z)$	=	z-transform of adaptive filter when processing h roots of $Y(z)$ (equations 4.87 and 4.88)
$\tilde{D}(z)$	=	z^{-n+1} $A_i(z)B_i(z)$ (equation 3.23)
$e_{i,h}$	=	coefficient of z^{-h} in $Y(z)$ $A_i(z)$ (equation 4.106)
$f_{1,h}$	=	coefficient of z^{-h} in $F_1(z)$ (equation 4.123)
$f_{m,h}$	=	coefficient of z^{-h} in $F_m(z)$ (equation 4.130)
$F(z)$	=	z-transform of channel and ideal adaptive filter (equation 2.12)
$F_h(z)$	=	z-transform of channel and adaptive linear filter when processing h roots of $Y(z)$ (equations 4.123 and 4.130)
$g+1$	=	number of components of Y (equation 2.4)
j	=	$\sqrt{-1}$
k	=	number of steps in iterative process to locate β_1 (equation 4.79)
m	=	number of roots of $Y(z)$ outside the unit circle (equation 2.8)
$n+1$	=	number of taps of adaptive linear feedforward transversal filter (Figure 2.1)
r_i	=	sample value of demodulated baseband signal at time $t=iT$ (equation 2.5)

s_i = i^{th} transmitted data-symbol (equation 2.1)

$s_{0,i}$ = real part of s_i (equation 2.1)

$s_{1,i}$ = imaginary part of s_i (equation 2.1)

T = sampling interval (Figure 2.1)

u_h = (h+1)th component of sampled impulse-response of channel when the factor $(1+\beta_1 z)$ has been removed from $Y(z)$ (equation 3.22)

w_i = Gaussian noise component in r_i (equation 2.5)

y_h = (h+1)th component of Y (equation 2.3)

Y = sampled impulse-response of channel (equation 2.3)

$Y(z)$ = z-transform of Y (equation 2.4)

$Y_1(z)$ = factor of $Y(z)$ with all its roots inside the unit circle (equation 2.7)

$Y_2(z)$ = factor of $Y(z)$ with all its roots outside the unit circle (equation 2.8)

α_h = negative of root of $Y(z)$ that lies inside the unit circle (equation 2.7)

β_h = negative – reciprocal of a root of $Y(z)$ that lies outside the unit circle (equation 2.8)

θ_i = measure of error in λ_i

λ_i = an estimate of the negative-reciprocal of a root of $Y(z)$ that lie outside the unit circle

ϕ_i' = measure of error in output sequence from one-tap feedback transversal filter (equation 4.95) as in System 4.3.1

ϕ_i'' = measure of error in output sequence from one-tap feedback transversal filter (equation 4.135) as in System 4.4)

ψ_1 = parameter given by equation 4.96

ψ_2 = parameter given by equation 4.97

ψ_3 = parameter given by equation 4.98

ψ_4 = parameter given by equation 4.99

d_1 = threshold level for determining convergence of iterative process (equations 4.78 and 4.115)

d_2 = threshold level for use in the averaging period (equation 4.156)

CHAPTER 1

INTRODUCTION

The existing worldwide telephone network, in addition to providing a readily accessible voice communication channel between practically any two major towns or cities in the world, has long been utilized for digital data communication[1-5]. Today, with the era of Information Technology, an ever greater demand for higher transmission speed exists, to cater for the vast amount of digital data transmitted between computers, and between computers and their remote equipment.

The band-limited telephone circuit has a pass-band ranging from approximately 300 to 3000 Hz which is available for the transmission of information. Unfortunately, it introduces a number of different types of noise and distortions[6,7] which corrupt the received data and ultimately limit the speed at which the signal can be transmitted (in elements per second) for an acceptable error rate[6-11]. When a digital data-transmission system operates by sending a serial stream of m-level signal elements down the channel, four dominant types of distortions occur[6,7]: attenuation distortion, amplitude distortion, phase distortion and additive noise. At high transmission rates, when the duration between the transmitted signal-elements is short compared with the impulse-response of the channel, a received element will overlap with one or more of the following elements and therefore interfere with them. This effect, known as intersymbol interference, is the primary impediment to reliable high-speed data transmission, at high signal to noise ratios, using the telephone network[6,7].

When it is required to achieve both a high transmission rate and a good tolerance to additive white Gaussian noise but without undue equipment complexity, in the transmission of digital data over a band-limited linear baseband channel, a promising and well known technique[11-14] is to transmit the data-symbols (signal-elements) as close as is reasonably possible to the Nyquist rate for the channel and to pass the noisy data signal to the receiver input through an appropriate lowpass filter. The noisy data signal is assumed to be formed by adding stationary white Gaussian noise to the distorted data signal at the output of the channel, and the lowpass filter has a flat amplitude response, with unit gain (zero attenuation) and a linear phase characteristic over the pass band, a very high attentuation in the stopband and a sharp (ideally instantaneous) cut-off, with a cut-off frequency (in Hz) at half the data-symbol rate. The filtered signal is sampled once per data-symbol (effectively at the Nyquist rate) and the detection process in the receiver operates on these samples. The samples contain (at least for practical purposes) all the useful information in the received noisy data signal, and the noise components in these samples are statistically independent Gaussian random variables with zero mean and fixed variance[11-14]. Thus, the detection process operating on the given sample can, if required, be made to approach quite closely to the optimum detection process operating directly on the noisy data signal at the receiver input. The impulse-response of the linear baseband channel may be real or complex-valued, the precise nature of the channel in the latter case being considered in more detail in Chapter 2.

A maximum-likelihood detection process that has recently been widely studied for the application just described is the Viterbi-algorithm detector[7,15]. This selects as the detected message the possible sequence of transmitted data-symbols for which there is the minimum mean-square difference between the samples of the corresponding received data signal, for the given signal distortion but in the absence of noise, and the samples of the signal actually received. When the transmitted data-symbols are statistically independent and equally likely to have any of their possible values (a condition assumed in this report), the Viterbi-algorithm detector minimizes the probability of error in the detection of the received message[7,15]. Unfortunately, when the sampled impulse-response of the channel contains a large number of components (non-zero samples), which is often the case in practice, the Viterbi algorithm involves both an excessive amount of storage and an excessive number of operations per received data-symbol[7].

A very much simpler arrangement, that is, however, normally suboptimum, is the conventional nonlinear (decision-feedback) equalizer[3,16-21]. This is here taken to be the nonlinear equalizer that minimizes the mean-square error in the equalized signal subject to the constraint that the equalizer achieves the accurate equalization of the linear baseband channel[7,17,19-21]. In the perfect form of the arrangement assumed in this report, where the signal is transmitted over the channel at the Nyquist rate and the ideal lowpass filter (previously described), with a cut-off frequency (in Hz) equal to half the data-symbol rate, is used at the receiver input, the linear filter, that forms the first part of the optimum nonlinear equalizer, becomes the lowpass filter and sampler followed by a linear feedforward transversal filter. Since the lowpass filter does not change the frequency response or sampled impulse-response of the channel, the channel is now taken to include the ideal lowpass filter and sampler[22]. The transversal filter is an allpass network with ideally an infinite number of taps, that adjusts the sampled impulse-response of the channel and filter to be minimum phase, without however changing any amplitude distortion introduced by the channel and without changing the levels of the data signal and noise, which means that the absolute value of the frequency response of the channel and filter is the same as that of the channel itself[22-26]. The transversal filter can, for convenience, be considered to operate in two stages. First, it equalizes all phase distortion, a sampled impulse-response that is linear phase, and then it converts this linear phase response into the corresponding minimum phase response, having the same absolute value of the frequency response[7,26]. The particular virtue of the operation just described is that it concentrates the energy of the sampled impulse-response of the channel and filter towards the earliest samples, with especial emphasis on the first sample, without however changing the signal/noise ratio at the output of the filter.

When a Viterbi-algorithm detector is used at the receiver and the data signal is transmitted over the channel at below the Nyquist rate, a "noise-whitened matched filter" must be used ahead of the detector if this is to minimize the probability of error in the detection of the received message[15]. The same filter also forms the first part of the corresponding nonlinear equalizer, that minimizes the mean-square error in the equalized signal subject to the exact equalization of the channel, under the new conditions[19,21]. If now the data-symbol rate is increased to the Nyquist rate (a condition assumed in this report), the noise-whitened matched filter degenerates into the combination of the ideal lowpass filter, sampler and linear feedforward transversal filter just described for the nonlinear equalizer[22].

Computer-simulation tests over various linear baseband channels that introduce different levels of amplitude distortion, ranging from low to very severe, but with little or no phase distortion, have shown that the Viterbi-algorithm detector can be replaced, with no significant loss in tolerance to additive white Gaussian noise, by very much simpler detection processes that are a development of the Viterbi algorithm[27]. No adaptive linear filter is used here ahead of the detector but an important condition that must be satisfied here is that the magnitude of the first component of the sampled impulse-response of the channel is not too small. When the first component is very small, a substantially modified detection process must be used[28]. Computer-simulation tests on various QAM data-transmission systems operating at 9600 bit/s and 19200 bit/s over models of various telephone circuits[22-25] have shown that, provided the linear feedforward transversal filter (previously described as forming the first part of the nonlinear equalizer) is used ahead of the detector, a significant improvement in tolerance to additive white Gaussian noise (typically 1-6 dB) can be achieved by replacing the conventional nonlinear equalizer by any one of various detection processes[22-25] that are further developments of the simplified processes previously mentioned[27]. The detectors studied here[22-25,27] are suboptimum (but not seriously so) and are known as near-maximum-likelihood detectors. The detectors are practically feasible, being considerably less complex than the corresponding Viterbi-algorithm detector. Extensive tests by W. Ser have shown that, without the linear transversal filter ahead of the detector, the severe phase distortion introduced by the poorer telephone circuits prevents the satisfactory operation of the near-maximum-likelihood detectors (even when modified to handle a very small value of the first component of the sampled impulse-response of the channel[28]),unless the detectors are further modified into much more complex systems[29].

It has become clear that for the most reliable operation of any near-maximum-likelihood detector or nonlinear equalizer, in a 9600 or 19000 bit/s data-transmission system operating over a telephone circuit, the linear feedforward transversal filter ahead of the detector must be correctly adjusted for the given telephone circuit, to act as an all-pass network that makes the resultant response minimum phase[22-28]. Now, an important advantage of a near-maximum-likelihood detector over a conventional nonlinear equalizer is that, not only does the near-maximum-likelihood detector have a better tolerance to noise, but, unlike the equalizer, it does not rely critically upon the sampled impulse-response of the channel and linear filter being minimum phase in order to achieve its optimum performance[7,27,28]. Hence, with a near-maximum-likelihood detector, there is considerably more scope than there is with a nonlinear equalizer for simplifying the adaptive linear filter and its adjustment algorithm.

Simple iterative processes based on the gradient algorithm and on various developments of this can be used both to adjust the linear feedforward transversal filter and to estimate the sampled impulse-response of the channel and filter[3,7,18,22,30]. When correct convergence of an iterative process is reached, in the presence of additive white Gaussian noise at high signal-to-noise ratios, the resulting linear filter approximates reasonably well to the ideal transversal filter (described in the discussion on the nonlinear equalizer in Chapter 1). J.D. Harvey has however shown that, when operating over the poorer telephone circuits, a sufficient accurate adjustment of the filter is not always obtained, even with a more sophisticated algorithm[30]. Furthermore, the inaccurate adjustment of the linear filter can here lead to a significant degradation in tolerance to noise, whether a near-maximum-likelihood detector or nonlinear equalizer is used, and an unduly long training signal may now have to be employed at the start of transmission, for the initial adjustment of the adaptive filter[30]. Finally, although, in the case of an equalizer, the adjustment of the adaptive linear filter for the minimum mean-square error in the equalized signal is probably optimum for most applications, the resulting adjustment of the filter is not necessarily (or even usually) the most cost-effective arrangement when a near-maximum-likelihood detector is used[7,22].

This report describes a novel technique that has been developed for adjusting the linear feedforward transversal filter ahead of the detector and at the same time estimating the sampled impulse-response of the channel and filter, that avoids the problems or disadvantages associated with other systems. The success of the new technique follows firstly from the fact that it makes full use of the prior knowledge that the linear filter must be an allpass network such that the sampled impulse-response of the channel and filter is minimum phase, and secondly because it operates solely and directly on the estimate of the sampled impulse-response of the channel. This estimate can be obtained much more quickly and accurately than can the corresponding adjustment of the adaptive linear filter by conventional means[7,30-34], and the new technique exploits to the full both the time taken to obtain the estimate and its accuracy. The technique relies on the well known fact[7,15] that the linear feedforward transversal filter ahead of the detector operates by removing all the roots (zeros) of the z-transform of the sampled impulse-response of the channel that lie outside the unit circle in the z-plane, and replacing them by the complex conjugates of their reciprocals, while leaving the remaining roots unchanged. Traditionally, methods for finding the roots (zeros) of a polynomial are aimed at low-degree polynomials with real coefficients[35,47]. These are generally classified into two groups[47], direct methods and indirect methods. In the direct methods some or all of the roots of the polynomial are evaluated by a single application of the process whereas in the indirect methods a process of successive approximation is used in the determination of the roots. Speed and accuracy vary from method to method and also depend critically upon the polynomial itself, the nature of the roots, the location of the roots, their relative magnitudes and their relative distances from each other 35-36,47,48. Indeed, most of the techniques will fail when dealing with "ill-conditioned" polynomials, a phenomenon which is not yet fully understood. Very little information in the published literature is available on the solution of high-degree polynomials (of degree greater than twenty) with complex coefficients in which the roots are no longer necessarily real or in complex-conjugate pairs.

This study is concerned mainly with the development of a suitable scheme which can be used to evaluate some or all of the roots (zeros) of high-degree polynomials with complex coefficients (given by the z-transform of the sampled impulse-response of a baseband channel) where the roots to be determined have a modulus greater than 1.0.

The report describes first an on-line system where the receiver operates on a continuous stream of received samples subject to both intersymbol interference and additive white Gaussian noise and it presents the results of a feasibility test, using computer-simulation, on four channels whose z-transforms are low-degree polynomials with real-valued coefficients. Further developments of the on-line system are tested using two telephone circuits which represent a typical line and a worst line in the British public switched telephone network normally considered for the transmission of data at 9600 bit/s.
The results obtained from the tests of the on-line system point to the potential advantages of the off-line system, which is much simpler to implement and requires less computation. The latter system operates solely and directly on the sampled impulse-response of the channel, which must again be given at the receiver. The report gives results of computer-simulation tests measuring the speed and accuracy of different arrangements of the system for different channels, and it also studies the effects of double roots and roots lying just outside the

unit circle. Finally, the report presents the complete configuration and results of the "standard algorithm" and the "modified algorithm" which are suitable for a 16-point QAM system operating at 9600 bit/s over the British public switched telephone network and (with slight modifications) a 64-point QAM system operating 19200 bit/s.

CHAPTER 2

MODEL OF THE DATA-TRANSMISSION SYSTEM USING 16-POINT QAM SIGNALS

The data-transmission system considered here operates at 9600 bit/s over a telephone circuit, the transmitted signal being a serial stream of 16-level QAM signal-elements with a carrier frequency of 1800 Hz and a signal-element rate of 2400 bauds. The model of the data-transmission system is shown in Figure 2.1. The information to be transmitted is carried by the data-symbols $\{s_i\}$, where

$$s_i = s_{0,i} + j\, s_{1,i} \qquad (2.1)$$

and $j = \sqrt{-1}$, $s_{0,i} = \pm 1$ or ± 3 and $s_{1,i} = \pm 1$ or ± 3. It is assumed that $s_i = 0$ for $i \leqslant 0$, so that the impulse $s_i\, \delta(t-iT)$ is the i^{th} signal-element at the input to the transmitter filter. Furthermore the $\{s_i\}$ (for $i>0$) are statistically independent and equally likely to have any of their 16 different possible values.

The transmission path is a linear baseband channel that includes a telephone circuit together with a linear modulator at the transmitter and a linear demodulator at the receiver. The transmitter and receiver filters in Figure 2.1 include the baseband equivalents of the filters operating on the QAM signal at the transmitter and receiver, respectively. The QAM signal is the sum of two 4-level suppressed carrier AM signals in phase quadrature. One of these carries the data-symbols $\{s_{0,i}\}$ and the other the data-symbols $\{s_{1,i}\}$. Each AM signal is associated with the corresponding modulator and demodulator to give a baseband channel, considerable coupling between the two channels being introduced by the poorer telephone circuits. By allocating real values to the input and output baseband of one of the two channels and imaginary values to the corresponding baseband of the other, a complex-valued baseband signal is obtained at both the input and output of the resultant transmission path in Figure 2.1. The transmitter filter, transmission path and receiver filter together form a linear baseband channel whose impulse-response is the complex-valued waveform $y(t)$[10,51]. For practical purposes $y(t)$ may be considered to have a finite duration and to be time invariant over any one transmission, such that $y(t-iT)$ is a time-shifted version of $y(t-hT)$, for any $i \neq h$.

The waveform at the output of the receiver filter (Figure 2.1) is the complex-valued signal

$$r(t) = \sum_i s_i y(t-iT) + w(t) \tag{2.2}$$

where $w(t)$ is the Gaussian noise waveform originating from the additive white Gaussian noise at the output of the transmission path. The waveform $r(t)$ is sampled at time instants $\{iT\}$, to give the received samples $\{r_i\}$ which are fed to the adaptive linear transversal filter. The sampled impulse-response of the linear baseband channel (Figure 2.1) is given by the (g+1)-component row vector

$$Y = [y_0 \quad y_1 \quad . \quad . \quad . \quad y_g] \tag{2.3}$$

whose z-transform is

$$Y(z) = y_0 + y_1 z^{-1} + . \, . \, . + y_g z^{-g} \tag{2.4}$$

where $y_i = y(iT)$ and the $\{y_i\}$ are complex-valued. The delay in transmission is neglected here, and, for practical purposes, $y_i = 0$

for i < 0 and i > g. Thus the received sample, at time t = iT, is

$$r_i = \sum_{h=0}^{g} s_{i-h} y_h + w_i \qquad (2.5)$$

where $r_i = r(iT)$ and $w_i = w(iT)$. It is assumed that the real and imaginary parts of the noise components $\{w_i\}$ are statistically independent Gaussian random variables with zero mean and fixed variance σ^2. The conditions that must be satisfied by the receiver filter (Figure 2.1) for the above assumptions to hold are considered elsewhere[6,13,14]. Clearly, a Viterbi-algorithm detector operating directly on the received samples $\{r_i\}$ would (at least for practical purposes) minimize the probability of error in the detection of the received message[7,11-15].

The detector (Figure 2.1) uses the estimate, F_m, of the sampled impulse-response of the channel and adaptive linear filter, together with the signals (samples) at the output of the adaptive filter and a prior knowledge of the possible values of s_i (equation 2.1), to form the selected data-symbols $\{s'_i\}$. With a near-maximum-likelihood detector[22-25] there is normally a delay in detection (relative to the signal at the detector input) of 8 to 32 sampling intervals. With a conventional nonlinear equalizer, the detector in Figure 2.1 degenerates into a simple threshold-level detector (introducing no delay in detection) together with a feedback transversal filter and intersymbol-interference cancellation at the detector input[7,16-22]. The intersymbol interference involves only the data-symbols that have been detected. The feedback transversal filter uses the $\{s'_i\}$ together with F_m to generate the intersymbol-interference components at the detector input. When both the $\{s'_i\}$ and F_m are correct, all intersymbol interference components are removed from the signal at the detector input. The nonlinear equalizer itself comprises the combination of the adaptive linear filter and detector (just described). No assumptions are made in this report about the particular form of detector used, since the errors in the $\{s'_i\}$ do not directly affect the performance of the adaptive processes studied.

The channel estimator (Figure 2.1) uses the received samples $\{r_i\}$ and the detected data-symbols $\{s_i'\}$ to form an estimate of Y (equation 2.3). This estimate is then used by the channel estimator both to adjust the adaptive filter and to determine F_m. The $\{r_i\}$ employed in the estimator are delayed by n sampling intervals in the adaptive linear transversal filter and possibly by further sampling intervals in the estimator itself. The noise components $\{w_i\}$ (equation 2.5) only affect F_m and the adjustment of the adaptive filter insofar as they change the estimate of Y. Techniques for estimating Y are studied in detail elsewhere[30-34] and only the simple linear feedforward channel estimator is described in this report. Clearly, the time taken to adjust the linear filter ahead of the detector and the accuracy of this adjustment are limited by the corresponding parameters in the estimation of the sampled impulse-response of the channel. The aim of the new technique is to make the two parameters (time and accuracy of adjustment) for the adaptive filter as close as possible to the corresponding parameters for the channel estimate. It is therefore, for convenience, assumed here that the estimate of the sampled impulse-response of the channel (as given by the channel estimator in Figure 2.1) is correct. The new technique for adjusting the adaptive linear filter and estimating the sampled impulse-response of the channel and filter is evaluated on the basis of the time taken to adjust the adaptive linear filter and the accuracy both of the adjustment of the filter and of the estimate of the sampled impulse-response of the channel and filter, given the exact estimate of the channel.

Suppose now that

$$Y(z) = Y_1(z)\, Y_2(z) \qquad (2.6)$$

where

$$Y_1(z) = \eta(1 + \alpha_1 z^{-1})(1 + \alpha_2 z^{-1}) \;.\;.\;.\; (1 + \alpha_{g-m} z^{-1}) \qquad (2.7)$$

and

$$Y_2(z) = z^{-m} (1 + \beta_1 z)(1 + \beta_2 z) \ . \ . \ . \ (1 + \beta_m z) \qquad (2.8)$$

It is assumed here that no roots (zeros) of $Y(z)$ lie exactly on the unit circle in the z-plane (a condition normally satisfied in the given application). Also

$$|\alpha_i| < 1 \text{ and } |\beta_i| < 1 \qquad (2.9)$$

where α_i is the negative of a root of $Y(z)$, β_i is the negative of the reciprocal of a root of $Y(z)$, and η is the appropriate complex value needed to satisfy equations 2.6-2.8. $|\alpha_i|$ and $|\beta_i|$ are the absolute values of α_i and β_i, respectively. In applications (not considered here) where $Y(z)$ has one or more roots on the unit circle, these are taken to be roots of $Y_1(z)$, such that in each case $|\alpha_i| = 1$.

The adaptive linear transversal filter (Figure 2.1) has n+1 taps, where n is typically between 20 and 60. It will, for convenience, be assumed here that the filter is adjusted to its ideal form, as required in a nonlinear equalizer[7,22]. The z-transform of the filter is now approximately

$$D(z) = z^{-n} Y_2^{-1}(z) \, Y_3(z) \qquad (2.10)$$

where

$$Y_3(z) = (1 + \beta_1^* z^{-1})(1 + \beta_2^* z^{-1}) \ . \ . \ . \ (1 + \beta_m^* z^{-1}) \qquad (2.11)$$

and β_i^* is the complex conjugate of β_i. Thus, the z-transform of the channel and ideal linear filter is approximately

$$F(z) = Y(z)\ D(z)$$

$$= z^{-n}\ Y_1(z)\ Y_3(z) \tag{2.12}$$

Equations 2.10 and 2.12 only apply accurately to the given system when $n\to\infty$, but a very good approximation is usually obtained here without using an unduly large value of n. Clearly all the roots of F(z) lie inside the unit circle in the z-plane, which means the channel and linear filter together have a response that is minimum phase[7]. The roots of $Y_3(z)$ are the complex conjugates of the reciprocals of the roots of $Y_2(z)$, so that the linear filter replaces all roots of Y(z) that lie outside the unit circle (those of $Y_2(z)$) by the complex conjugates of their reciprocals, leaving the remaining roots (those of $Y_1(z)$) unchanged.

Consider a simple case where Y(z) has only one root $(-\frac{1}{\beta_1})$ outside the unit circle in the z-plane, so that $|\beta_1| < 1$. Now

$$Y(z) = z^{-1}(1 + \beta_1 z).\eta(1 + \alpha_1 z^{-1})(1 + \alpha_2 z^{-1})\ldots(1 + \alpha_{g-1} z^{-1})$$

$$= (\beta_1 + z^{-1}).\eta(1 + \alpha_1 z^{-1})(1 + \alpha_2 z^{-1})\ldots(1 + \alpha_{g-1} z^{-1}) \tag{2.13}$$

The z-transform of the (n+1)-tap linear transversal filter is

$$D(z) = z^{-n}\ (\beta_1 + z^{-1})^{-1}\ (1 + \beta_1^*\ z^{-1}) \tag{2.14}$$

where it can be seen that $(1 + \beta_1^*\ z^{-1})$ is obtained by reversing the coefficients of $(\beta_1 + z^{-1})$ and taking the complex conjugate of the reversed factor. Thus $(1 + \beta_1^* z^{-1})$ is the z-transform of the filter matched to $(\beta_1 + z^{-1})^7$. Now

$$D(z) = z^{-n} (\beta_1+z^{-1})^{-1} (1+\beta_1^* z^{-1})$$

$$= z^{-n+1} (1+\beta_1 z)^{-1} (1+\beta_1^* z^{-1})$$

$$= z^{-n+1} (1-\beta_1 z+\beta_1^2 z^2-\beta_1^3 z^3+ \ldots)(1+\beta_1^* z^{-1})$$

$$= z^{-n+1} (1-\beta_1 z+\beta_1^2 z^2-\beta_1^3 z^3 + \ldots$$

$$+\beta_1^* z^{-1}-\beta_1^* \beta_1+\beta_1^* \beta_1^2 z -\beta_1^* \beta_1^3 z^2 + \ldots)$$

$$= z^{-n+1} (\beta_1^* z^{-1}+(1-\beta_1^* \beta_1)-\beta_1(1-\beta_1^* \beta_1)z+\beta_1^2 (1-\beta_1^* \beta_1)z^2- \ldots)$$

$$= z^{-n+1} (\beta_1^* z^{-1}+(1-|\beta_1|^2)(1-\beta_1 z+\beta_1^2 z^2 \ldots)) \qquad (2.15)$$

From equation 2.15, it can be seen that the sum of the squares of the moduli (absolute values) of the tap gains of the linear transversal filter (whose z-transform D(z)) is

$$|D|^2 = |\beta_1^*|^2 + (1-|\beta_1|^2)^2 (1+|\beta_1|^2+|\beta_1{}^2|^2 + \ldots)$$

and as $n \to \infty$,

$$|D|^2 = |\beta_1|^2 + (1-|\beta_1|^2)^2 \frac{1}{(1-|\beta_1|^2)}$$

$$= |\beta_1|^2 + (1-|\beta_1|^2)$$

$$= 1 \qquad (2.16)$$

The result of equation 2.16 can be extended to the general case where Y(z) has m roots $(-\frac{1}{\beta_1}, -\frac{1}{\beta_2}, \ldots, -\frac{1}{\beta_m})$ outside the unit circle in the z-plane.

The linear filter, whether correctly adjusted or not, performs an orthogonal unitary transformation[51] on the received signal at all times, since it is constrained to be an all-pass network. When the filter is correctly adjusted for all the roots outside the unit circle in the z-plane, the resultant sampled impulse-response of the channel and filter is minimum phase. The effect of the transformation on the input signal (the sequence of the $\{r_i\}$) is such that the level of the data signal, together with all noise statistics, remain unchanged[51], and the tolerance to white Gaussian noise of a maximum-likelihood detector (such as a Viterbi-algorithm detector) is unaffected by the inclusion or omission of the filter[7]. (The linear filter is clearly the same as that forming the first part of the ideal nonlinear equalizer[7,22,30]) and it has all the various properties described for this filter in Chapter 1. Since the receiver is assumed to know Y(z), it can, in principle, evaluate D(z) and hence also F(z). However, this involves the determination of all the roots of Y(z) that lie outside the unit circle, and the conventional techniques for achieving the given result are generally too complex to be of any real practical value. An alternative technique has therefore been developed for evaluating D(z) and F(z) from Y(z), which involves a novel and basically simple method of determining the required roots of Y(z). The heart of this new method is based on the fact that the linear filter, when constrained to be an all-pass network, can be implemented very easily by two filter sections with very few and simple tap coefficients. This will now be described.

2.1 Alternative Implementation of the Linear Feedforward Transversal Filter

Suppose the z-transform of the sampled impulse-response of the channel has m roots (zeros) lying outside the unit circle in the z-plane. The linear filter that forms the first part of the ideal nonlinear equalizer [7], when correctly adjusted for the channel, replaces all the roots outside the unit circle by their complex conjugate-reciprocals while leaving those lying on or inside the unit circle unchanged. This process can be carried out in m stages, each one responsible for one of the m roots that lie outside the unit circle.

Let $D_1(z)$ be the z-transform of a (n+1)-tap linear transversal filter that removes one of the m roots of $Y(z)$ (equation 2.3) and replaces it by the reciprocal of its complex conjugate. From equation 2.14

$$D_1(z) = z^{-n} (\beta_1 + z^{-1})^{-1} (1 + \beta_1^* z^{-1})$$

$$= z^{-n+1} (1 + \beta_1 z)^{-1} (1 + \beta_1^* z^{-1}) \qquad (2.17)$$

where $|\beta_1| < 1$ and $-\frac{1}{\beta_1}$ is one of the m roots of $Y(z)$ outside the unit circle. Again equation 2.17 is exact when $n \to \infty$.

Consider now a two-tap feedforward transversal filter and a one-tap feedback filter as shown in Figures 2.2 and 2.3, respectively.

Clearly the z-transform of the two-tap feedforward transversal filter (Figure 2.2) is

$$B_i(z) = 1 + \lambda_i^* z^{-1} \tag{2.18}$$

where i = 0,1,... .

Consider the one-tap feedback filter as shown in Figure 2.3 with the z-transforms of the input and the output sequence equal to $A_i'(z)$ and $A_i''(z)$ respectively.

$$A_i''(z) = A_i'(z) - \lambda_i z^{-1} A_i''(z)$$

$$A_i''(z)(1 + \lambda_i z^{-1}) = A_i'(z)$$

Thus the z-transform of the one-tap feedback filter is

$$A_i(z) = \frac{A_i''(z)}{A_i'(z)} = \frac{1}{1 + \lambda_i z^{-1}}$$

$$= (1 + \lambda_i z^{-1})^{-1} \tag{2.19}$$

since $|\lambda_i z^{-1}| < 1$, which means that the filter is stable.

At the receiver the continuous stream of received samples $\{r_i\}$ is fed into the two-tap feedforward filter and a sequence $\{q_i\}$ is produced at the output. This output sequence $\{q_i\}$ is then delayed by n sampling intervals and then is reversed in order, such that it starts with the most recent sample, and is fed through the one-tap feedback filter. The sequence, passing through the filter in reverse order, is taken to be moving backwards in time, and a delay of one sampling interval T in the feedback filter (Figure 2.3) now becomes an advance of T, with z-transform z. Thus the effective z-transform of the feedback filter becomes

$$A_i(z) = (1 + \lambda_i z)^{-1} \tag{2.20}$$

and the z-transform of the filters connected in cascade which operate on the received samples $\{r_i\}$ in the manner described is

$$\begin{aligned} C_1(z) &= A_i(a)\ B_i(z) \\ &= (1 + \lambda_i z)^{-1}\ (1 + \lambda_i^* z^{-1}) \end{aligned} \tag{2.21}$$

Suppose now the value λ_i is adjusted by some iterative algorithm such that when $i = k$, $\lambda_k \simeq \beta_1$.

Now

$$\begin{aligned} C_1(z) &= (1 + \lambda_k z)^{-1}\ (1 + \lambda_k^* z^{-1}) \\ &\simeq (1 + \beta_1 z)^{-1}\ (1 + \beta_1^* z^{-1}) \end{aligned} \tag{2.22}$$

The effect of the combined filter on the channel (whose z-transform Y(z)) is, therefore, to remove the root $-\frac{1}{\beta_1}$ of Y(z) and replaces it by the complex conjugate of its reciprocal. It is thus an all-pass filter having the same basic properties as the ideal linear filter with z-transform $D_1(z)$ (equation 2.17). Thus, the process of replacing the m roots of Y(z) by their complex-conjugate reciprocals can be carried out in m stages where each $\beta_h(h = 1,2,\dots,m)$ is associated with a different filter, with z-transform $C_h(z)$.

2.2 Channels for Use with the Baseband Equivalent Model for QAM Signals

Four telephone circuits, chosen to represent a wide range of telephone lines in the British Public Switched Telephone Network (BPSTN), have been used for computer-simulation tests based on the QAM data-transmission system of Figure 2.1. The baseband equivalents of the four circuits have been derived for both the 16-point QAM signals operating at 9600 bit/s and the 64-point QAM signals operating at 19200 bit/s; these are designated as Channels 1-4 and Channels 5-8 respectively. The telephone circuits in Channels 1 and 2 introduce typical levels of attenuation distortion and group-delay distortion[6], and the telephone circuits in Channel 3 and Channel 4 are close to the typical worst circuits (in the BPSTN) normally considered for the transmission of data at 9600 bit/s and 1200 bit/s, respectively. The telephone circuit in Channel 3 is close to the standard network N6 and introduces severe group-delay distortion as well as considerable attentuation distortion, and the telephone circuit in Channel 4 is close to the standard network N3 and introduces extremely severe attenuation distortion[22-26]. The sampling rate here is quite close to the Nyquist rate for the linear baseband channel (Figure 2.1), the latter being determined essentially by the equipment filters. The attenuation and group-delay characteristics of the telephone circuits and equipment filters are shown in Figures 2.4 to 2.8.

For our purposes, the most important property of the linear baseband channel (Figure 2.1) is the location of the roots (zeros) of the z-transform of the sampled impulse-response of the channel, particularly those that lie outside the unit circle in the z-plane. These are numbered in order of decreasing absolute value, starting at No.1 with the root having the largest absolute value. The complex-valued sampled impulse-responses of the equivalent baseband channels, and the location of all the roots (those that lie outside the unit circle being numbered), are shown in Table 2.1 and Figures2.9 to 2.12 respectively. The resultant sampled impulse-responses of the baseband channels when they have been made minimum phase are given in Table 2.2. Furthermore, the magnitudes of the components of Y (equation 2.3), both before and after the minimization process are as shown in Figures 2.13-2.16. For the purpose of illustration,the magnitudes of the components of Y (equation 2.3) at various stages of the process whereby the roots outside the unit circle are removed and replaced by their complex conjugate-reciprocals for Channel 1 are as shown in Figure 2.17.

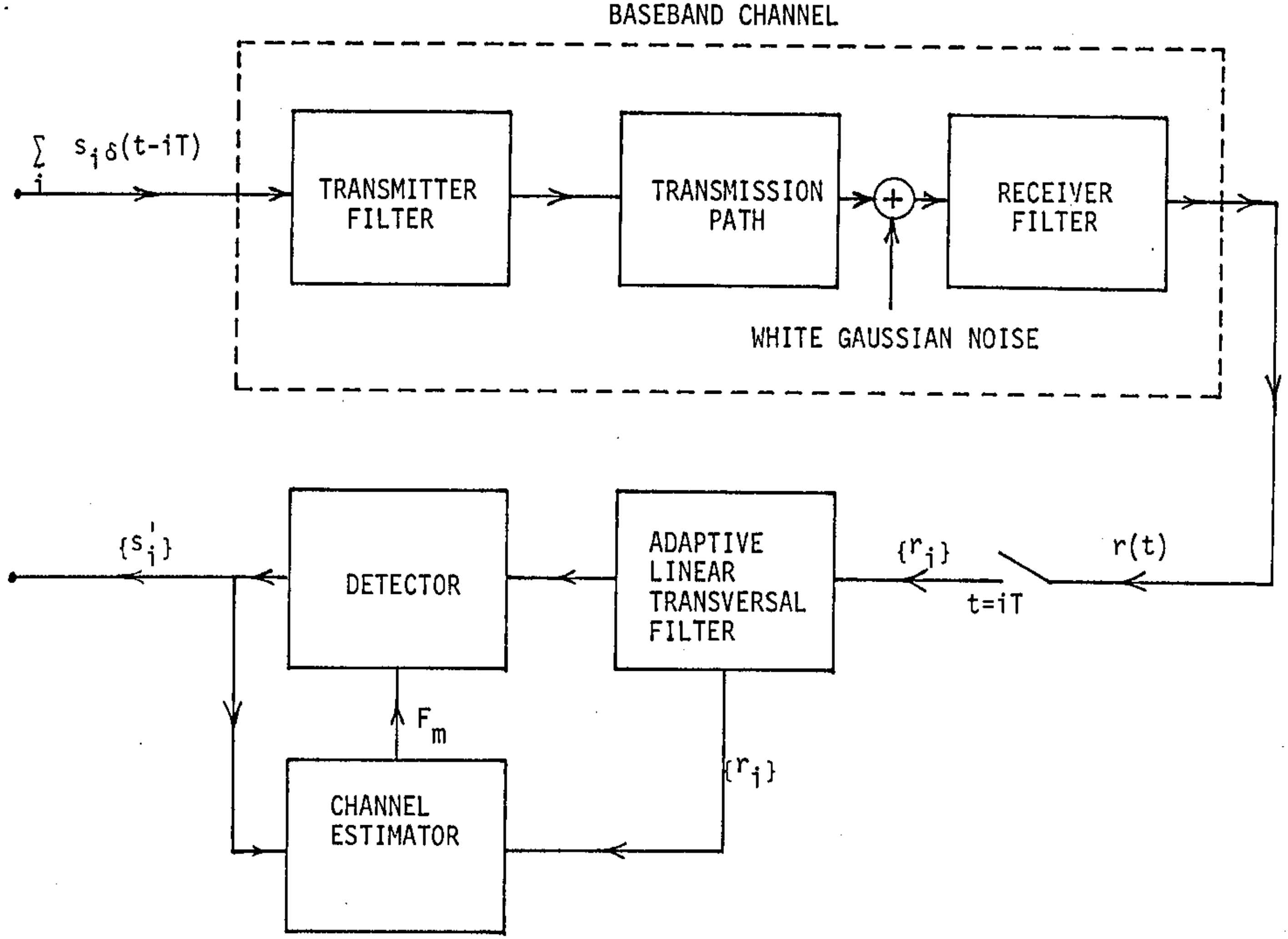


FIGURE 2.1: Model of Data-transmission System

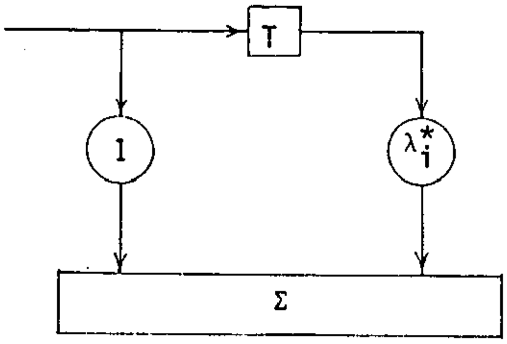


FIGURE 2.2: Two-tap feedforward filter

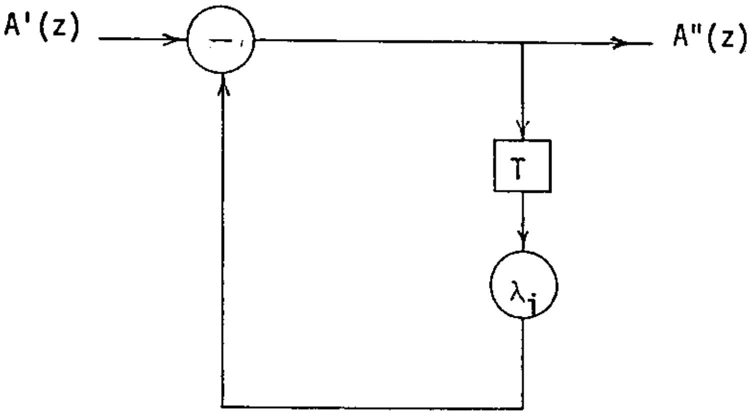


FIGURE 2.3: One-tap feedback filter

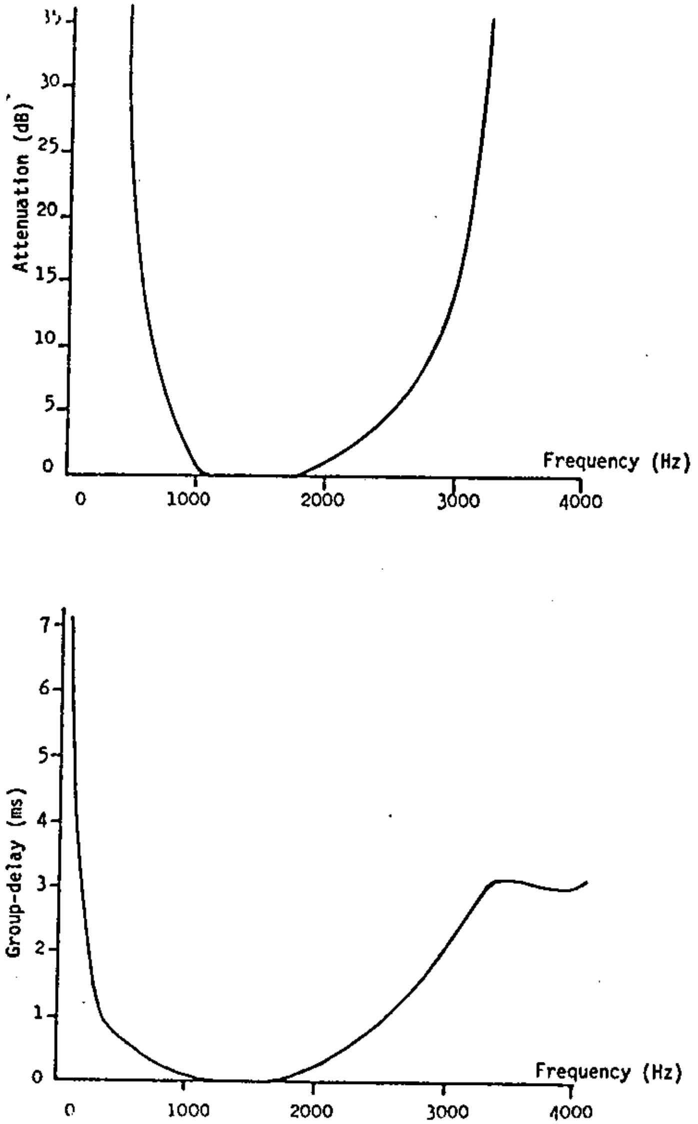


FIGURE 2.4: Attenuation and group-delay characteristics of telephone circuit 1 (Table 2.1)

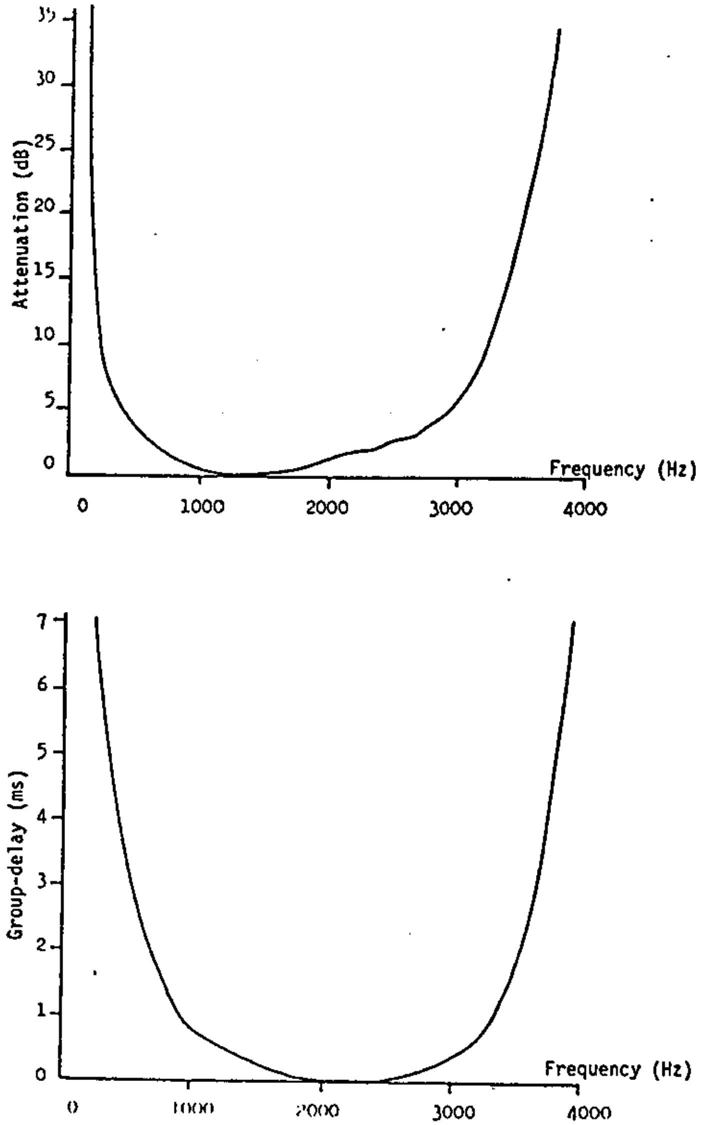


FIGURE 2.5: Attenuation and group-delay characteristics of telephone circuit 2 (Table 2.1)

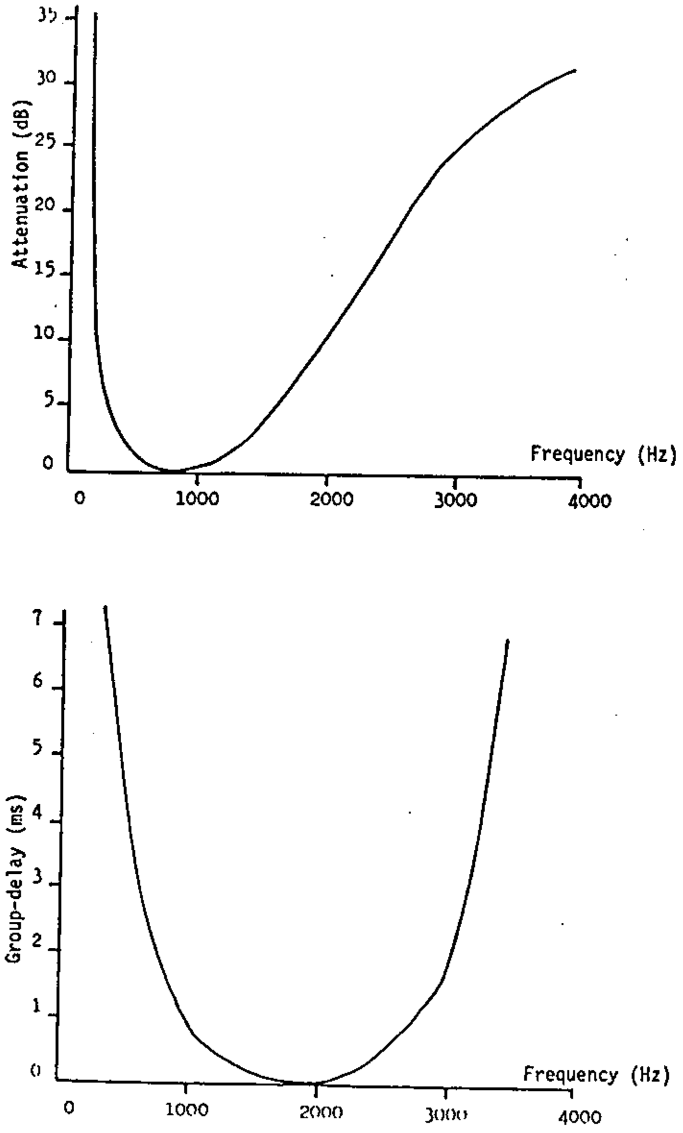


FIGURE 2.6: Attenuation and Group-delay characteristics of telephone circuit 3 (Table 2.1)

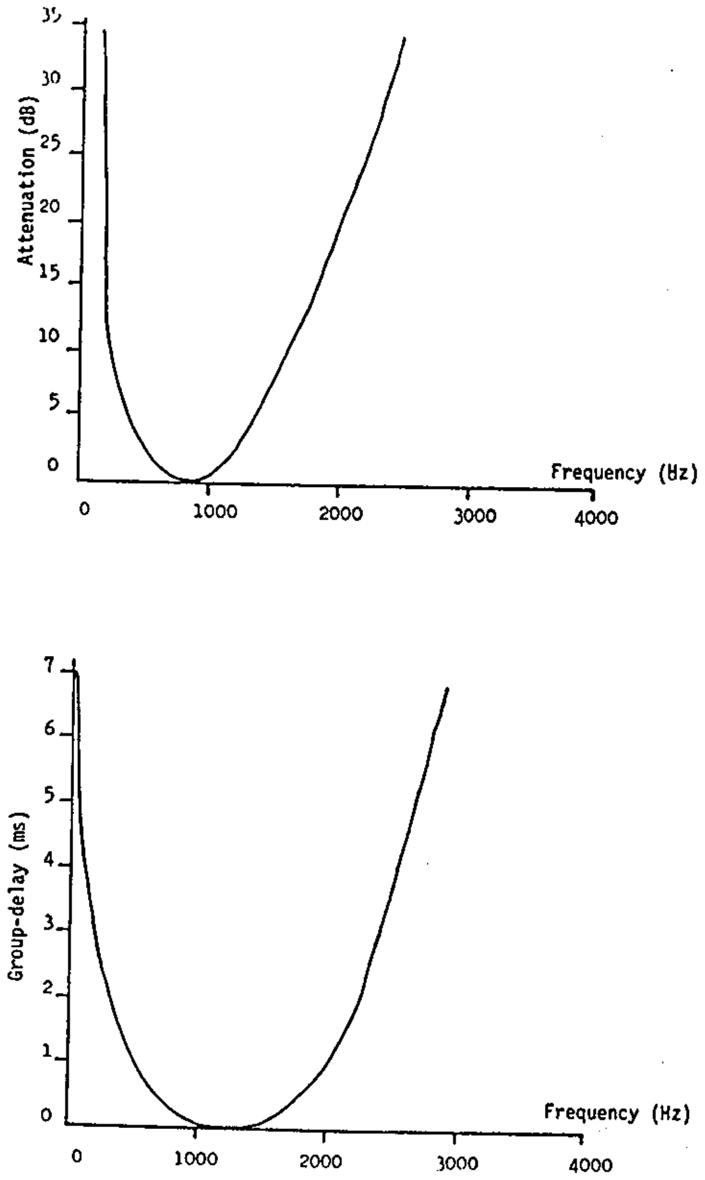
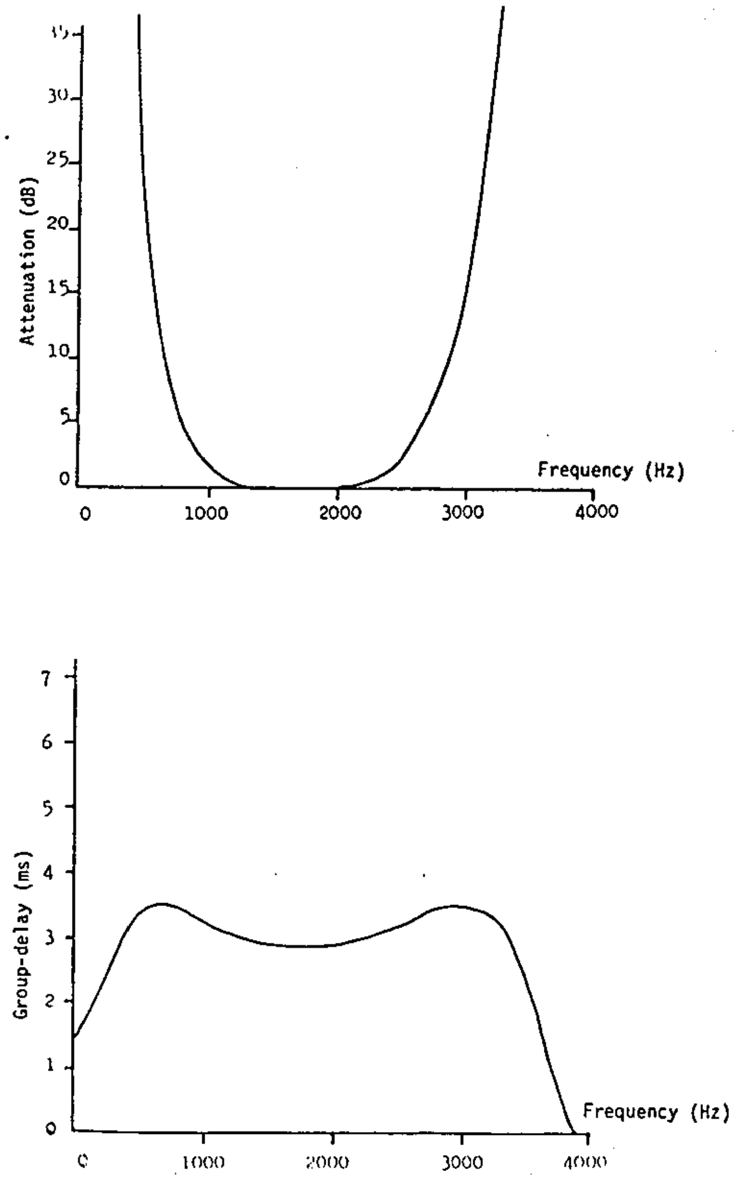


FIGURE 2.7: Attenuation and Group-delay characteristics of telephone circuit 4 (Table 2.1)



GROUP 2.8: Attenuation and group-delay characteristics of the combination of transmitter and receiver filters

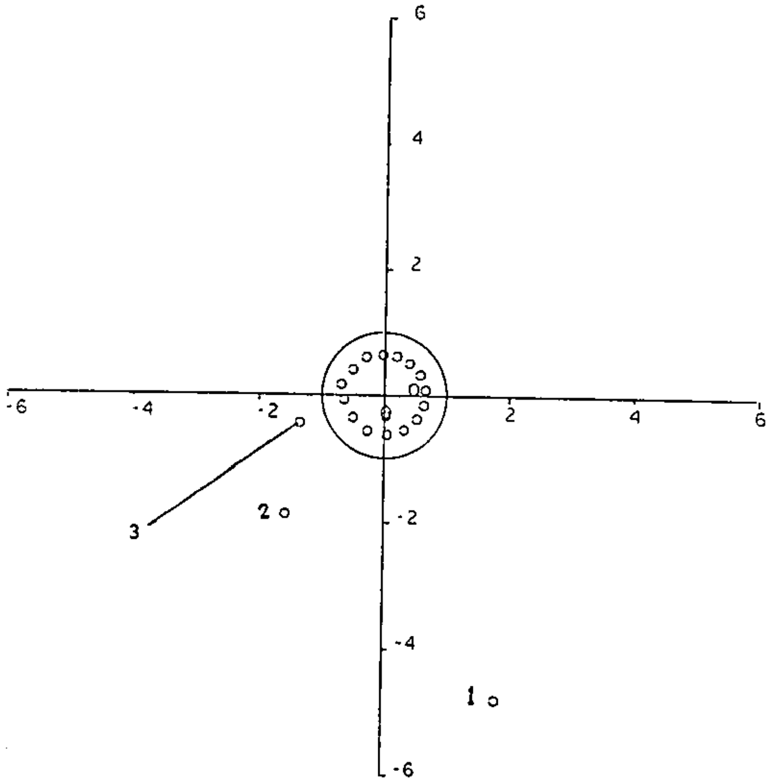


FIGURE 2.9: Roots (zeros) of Y(z) of Channel 1 (sampled at 2400 bauds)

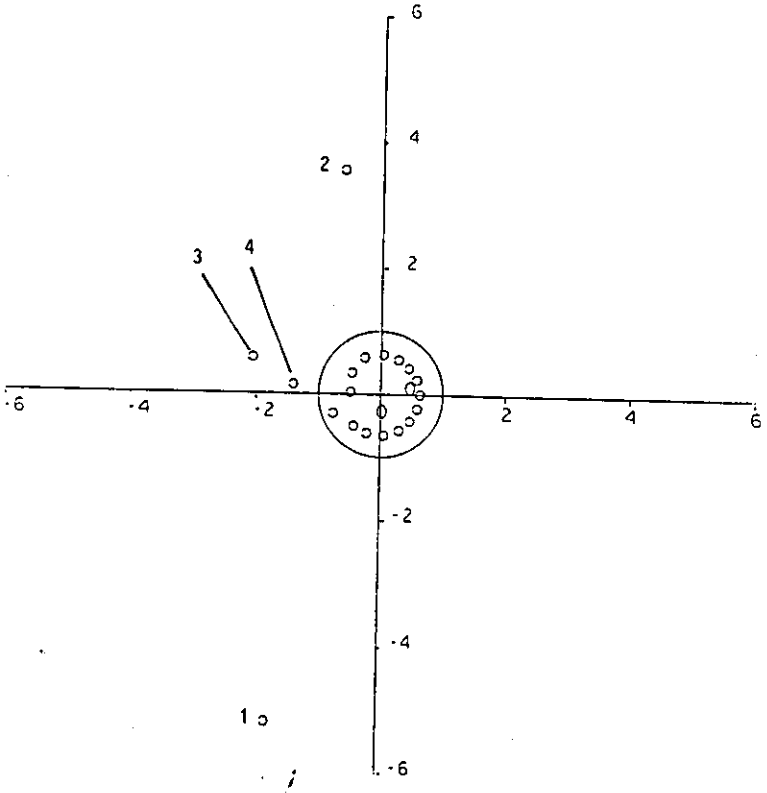


FIGURE 2.10: Roots (zeros) of Y(z) of Channel 2 (sampled at 2400 bauds)

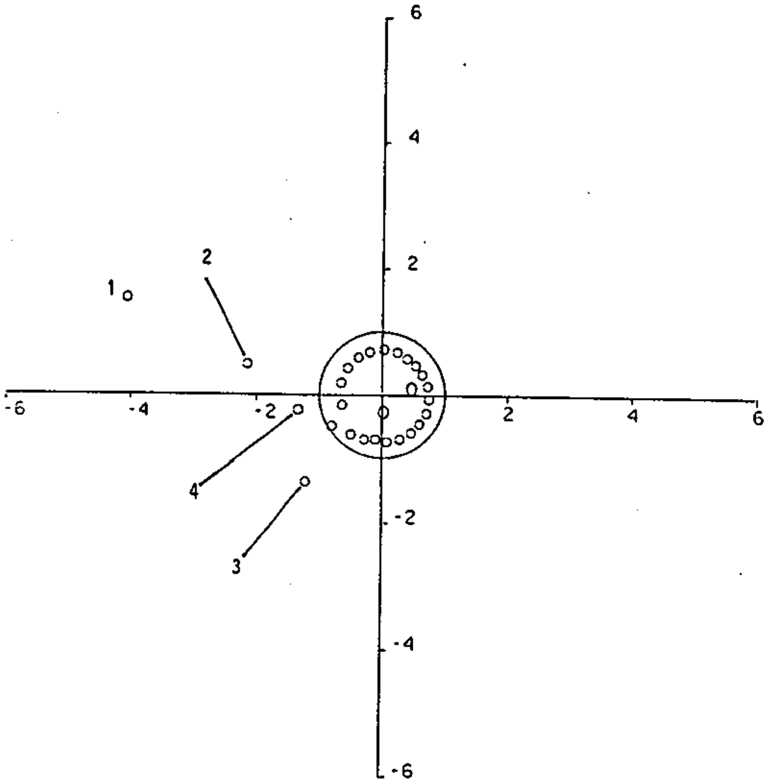


FIGURE 2.11: Roots (zeros) of Y(z) of Channel 3 (sampled at 2400 bauds)

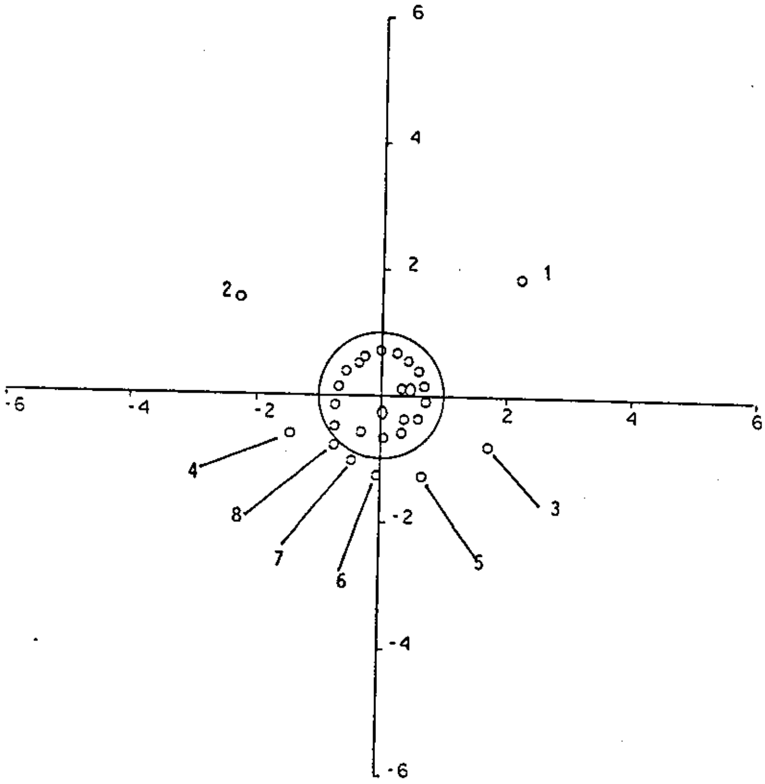


FIGURE 2.12: Roots (zeros) of Y(z) of Channel 4 (sampled at 2400 bauds)

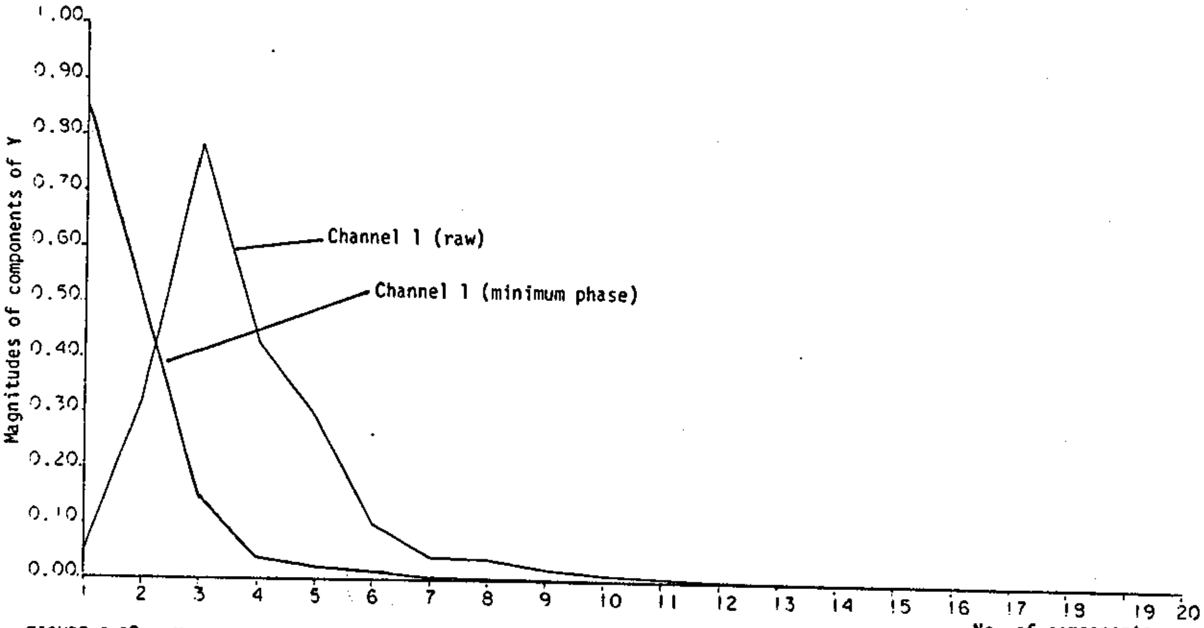


FIGURE 2.13: Magnitudes of the components of Y for Channel 1

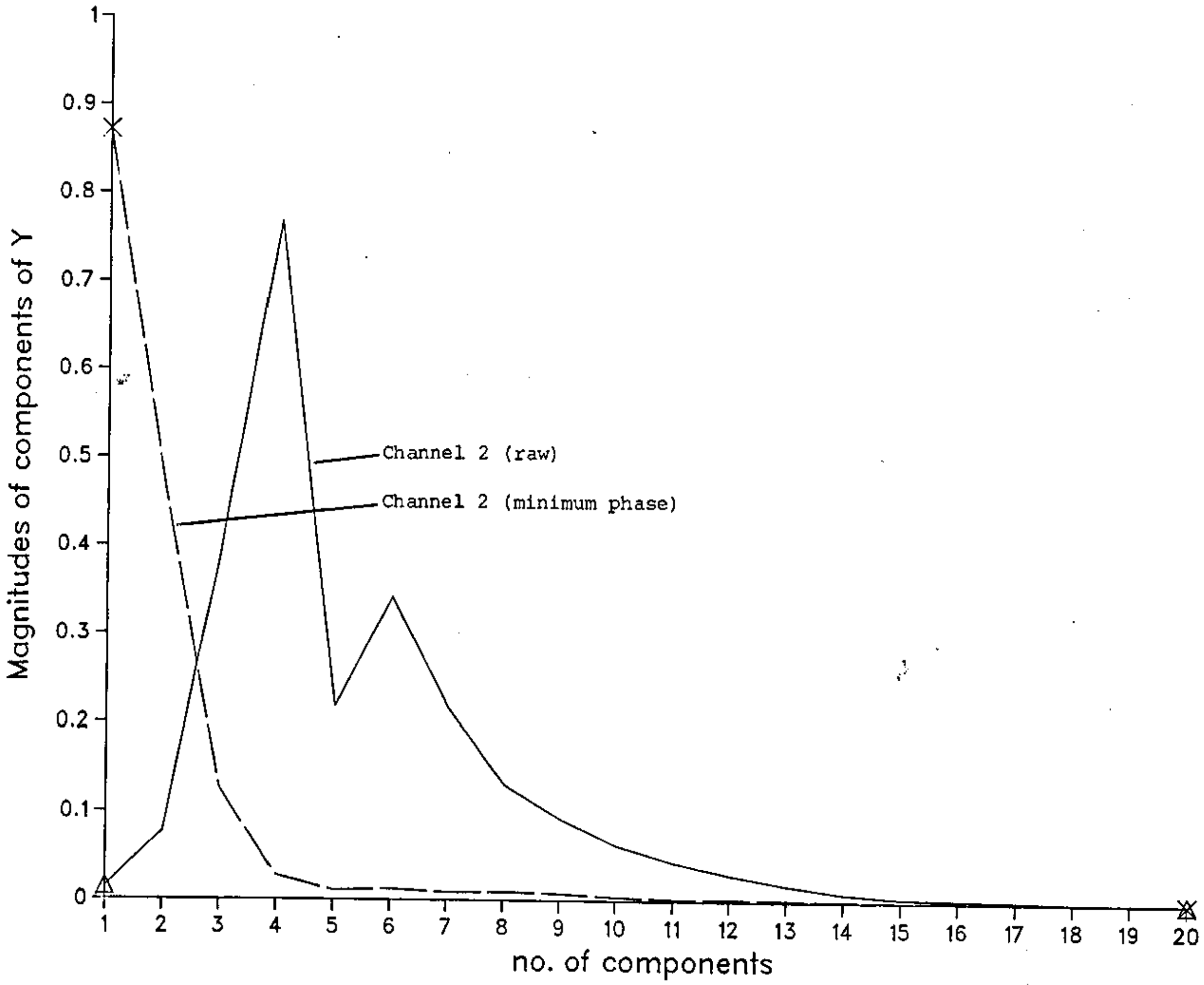


FIGURE 2.14: Magnitudes of the components of Y for Channel 2

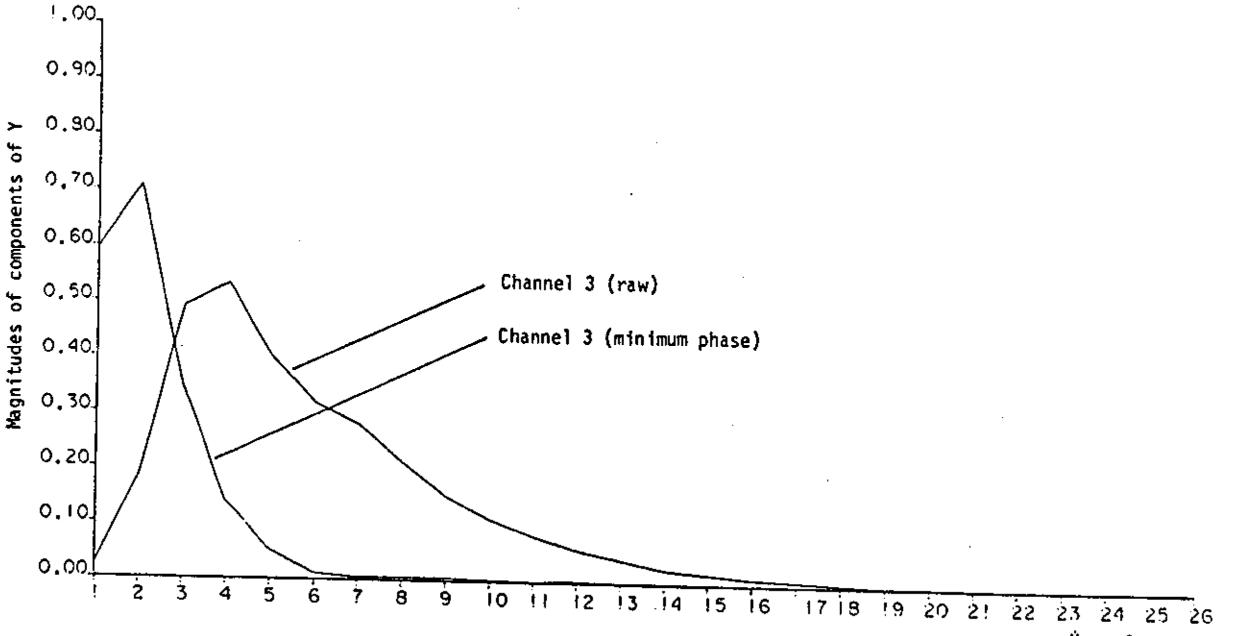


FIGURE 2.15: Magnitudes of components of Y for Channel 3

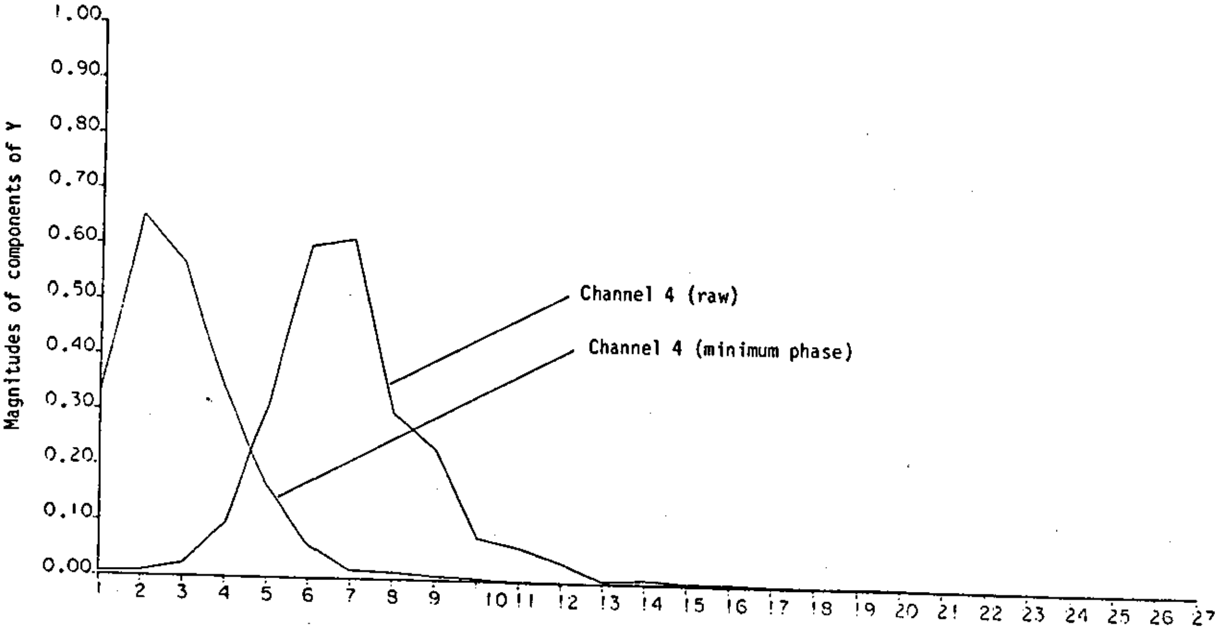


FIGURE 2.16: Magnitudes of components of Y for Channel 4

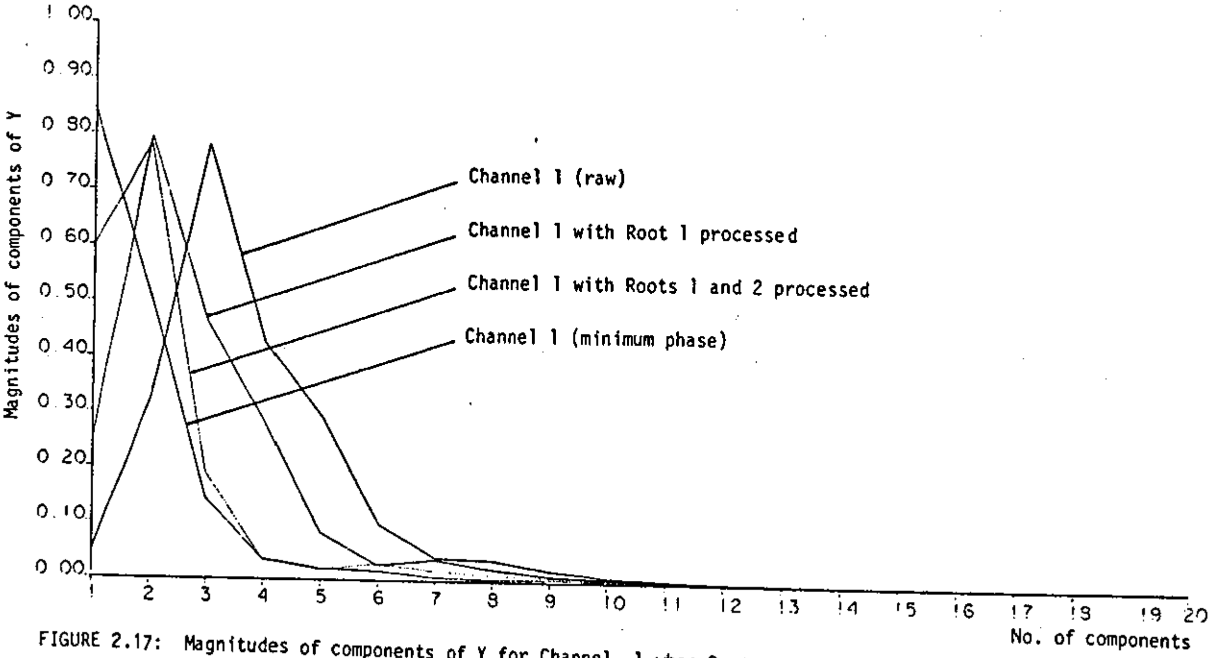


FIGURE 2.17: Magnitudes of components of Y for Channel 1 when Roots 1-3 are removed and replaced by their complex conjugate-reciprocals in turn

TABLE 2.1: The Sampled Impulse-Responses of Channels 1-4

CHANNEL 1		CHANNEL 2		CHANNEL 3		CHANNEL 4	
Real Part	Imag. Part	Real Part	Imag. Part	Real Part	Imag. Part	Real Part	Imag. Part
-0.0291	-0.0373	0.0145	-0.0006	0.0176	-0.0175	-0.0038	-0.0049
0.2290	-0.2244	0.0750	0.0176	0.1381	-0.1252	0.0077	-0.0044
0.7612	0.1817	0.3951	0.0033	0.4547	-0.1885	0.0094	0.0207
0.2988	0.3050	0.7491	-0.1718	0.5078	0.1622	-0.0884	0.0355
-0.0338	-0.2915	0.1951	0.0972	-0.1966	0.3505	-0.1138	-0.2869
-0.0789	0.0616	-0.2856	0.1894	-0.2223	-0.2276	0.5546	-0.2255
0.0291	0.0287	0.0575	-0.2096	0.2797	-0.0158	0.1903	0.5813
-0.0137	-0.0352	0.0655	0.1139	-0.1636	0.1352	-0.2861	-0.0892
0.0020	0.0204	-0.0825	-0.0424	0.0594	-0.1400	0.2332	-0.0384
0.0004	-0.0108	0.0623	0.0085	-0.0084	0.1111	-0.0652	0.0428
0.0028	0.0065	-0.0438	0.0034	-0.0105	-0.0817	0.0335	-0.0519
-0.0027	-0.0014	0.0294	-0.0049	0.0152	0.0572	-0.0323	0.0170
0.0000	-0.0013	-0.0181	0.0032	-0.0131	-0.0406	0.0044	-0.0023
0.0003	0.0006	0.0091	0.0003	0.0060	0.0255	0.0054	0.0076
-0.0002	0.0001	-0.0038	-0.0023	0.0003	-0.0190	0.0008	-0.0051
-0.0009	-0.0006	0.0019	0.0027	-0.0035	0.0116	-0.0056	0.0001
0.0005	-0.0000	-0.0018	-0.0014	0.0041	-0.0078	0.0018	0.0032
0.0003	0.0004	0.0006	0.0003	-0.0031	0.0038	-0.0009	-0.0015
0.0001	-0.0011	0.0005	0.0000	0.0018	-0.0005	-0.0022	-0.0026
0.0004	0.0001	-0.0008	-0.0001	-0.0018	-0.0005	0.0029	0.0019
				0.0007	0.0007	-0.0008	0.0009
				0.0004	0.0001	-0.0014	-0.0003
				-0.0004	0.0001	0.0019	-0.0002
				-0.0001	0.0010	-0.0003	0.0005
				0.0000	-0.0007	0.0007	0.0005
				0.0004	0.0008	-0.0007	-0.0001
						0.0002	-0.0008

TABLE 2.2: The Minimum Phase Sampled Impulse-Responses of Channels 1-4

CHANNEL 1		CHANNEL 2		CHANNEL 3		CHANNEL 4	
Real Part	Imag. Part	Real Part	Imag. Part	Real Part	Imag. Part	Real Part	Imag. Part
0.5349	0.6551	0.7320	-0.4731	0.5587	-0.2035	-0.3188	-0.0729
0.0452	0.5097	0.4632	-0.0910	0.4814	0.5210	0.0662	-0.6489
-0.1082	-0.0948	-0.1098	0.0624	-0.3165	0.1429	0.5328	0.1901
0.0350	-0.0093	0.0173	-0.0212	0.0526	-0.1286	-0.2726	0.2031
-0.0183	0.0081	0.0105	0.0012	0.0079	0.0523	0.0255	-0.1639
0.0008	-0.0166	-0.0117	0.0035	-0.0051	-0.0104	0.0261	0.0546
0.0035	0.0058	0.0076	-0.0043	0.0005	0.0050	-0.0065	-0.0144
-0.0023	-0.0033	-0.0085	0.0041	-0.0043	-0.0031	0.0025	0.0129
0.0034	0.0040	0.0071	-0.0022	0.0056	0.0013	-0.0038	-0.0069
-0.0034	-0.0005	-0.0041	0.0005	-0.0028	-0.0011	0.0046	0.0014
0.0006	-0.0019	0.0011	0.0017	0.0002	0.0008	-0.0001	0.0008
0.0007	0.0005	0.0001	-0.0021	0.0010	-0.0038	-0.0009	0.0011
-0.0001	-0.0000	0.0004	0.0017	0.0000	0.0016	0.0001	0.0000
-0.0008	-0.0007	-0.0007	-0.0003	0.0007	-0.0019	0.0003	0.0025
0.0007	0.0001	-0.0002	0.0000	-0.0004	-0.0004	-0.0017	-0.0020
0.0000	0.0005	0.0006	0.0000	0.0011	0.0007	0.0012	-0.0013
0.0003	-0.0004	-0.0003	-0.0000	-0.0011	-0.0004	0.0017	0.0015
0.0005	0.0002	-0.0002	0.0001	-0.0001	0.0000	-0.0019	-0.0000
-0.0000	0.0002	-0.0000	0.0000	-0.0000	0.0000	0.0006	-0.0010
-0.0000	-0.0000	-0.0000	0.0000	0.0002	0.0000	0.0003	-0.0000
				-0.0001	0.0007	0.0004	0.0002
				-0.0005	-0.0000	-0.0002	0.0003
				0.0000	0.0001	-0.0000	-0.0001
				0.0001	0.0004	0.0000	0.0000
				-0.0000	0.0002	0.0000	-0.0000
				0.0000	0.0000	-0.0000	0.0000
						0.0000	-0.0000

2.3 Linear Feedforward Estimator

The linear feedforward estimator is the simplest of all channel estimators which is implemented as a linear feedforward transversal filter. This structure was first proposed by F.R. Magee and J.G. Proakis[31] for use with a maximum-likelihood detector employing the Viterbi algorithm. In the case of QAM systems where the sampled impulse-response of the channel is complex-valued, the channel estimate can be modified for use with complex-valued signals as shown in Figure 2.18. Its operation will now be described.

Each square marked T is a store that holds the corresponding detected data-symbol s'_{i-h}, and each time the stores are triggered, on the reception of a sample r_i (equation 2.5) the store values are shifted one place to the right. Following the detection of s_i, the detected data-symbols s'_i , s'_{i-1}, s'_{i-2}, ..., s'_{i-g} are held in the estimator as shown. Each symbol s'_{i-h} is multiplied by the corresponding component $y'_{i-1,h}$ of the estimate Y'_{i-1} of the vector Y_{i-1} and the resulting products are added to give the estimate r'_i of the received sample r_i, where

$$r'_i = \sum_{h=0}^{g} s'_{i-h}\, y'_{i-1,h} \qquad (2.23)$$

The error in r'_i , which is $e_i = r_i - r'_i$, is multiplied by a small positive quantity b, and the resulting signal be_i is multiplied by the complex conjugate $(s'_{i-h})^*$ of each detected data-symbol s'_{i-h}, and the products are added to the corresponding components of Y'_{i-1} to give the new stored estimate Y'_i . Thus the $(h+1)^{th}$ component of Y'_i is

$$y'_{i,h} = y'_{i-1,h} + be_i\,(s'_{i-h})^* \qquad (2.24)$$

for h = 0,1, ..., g.

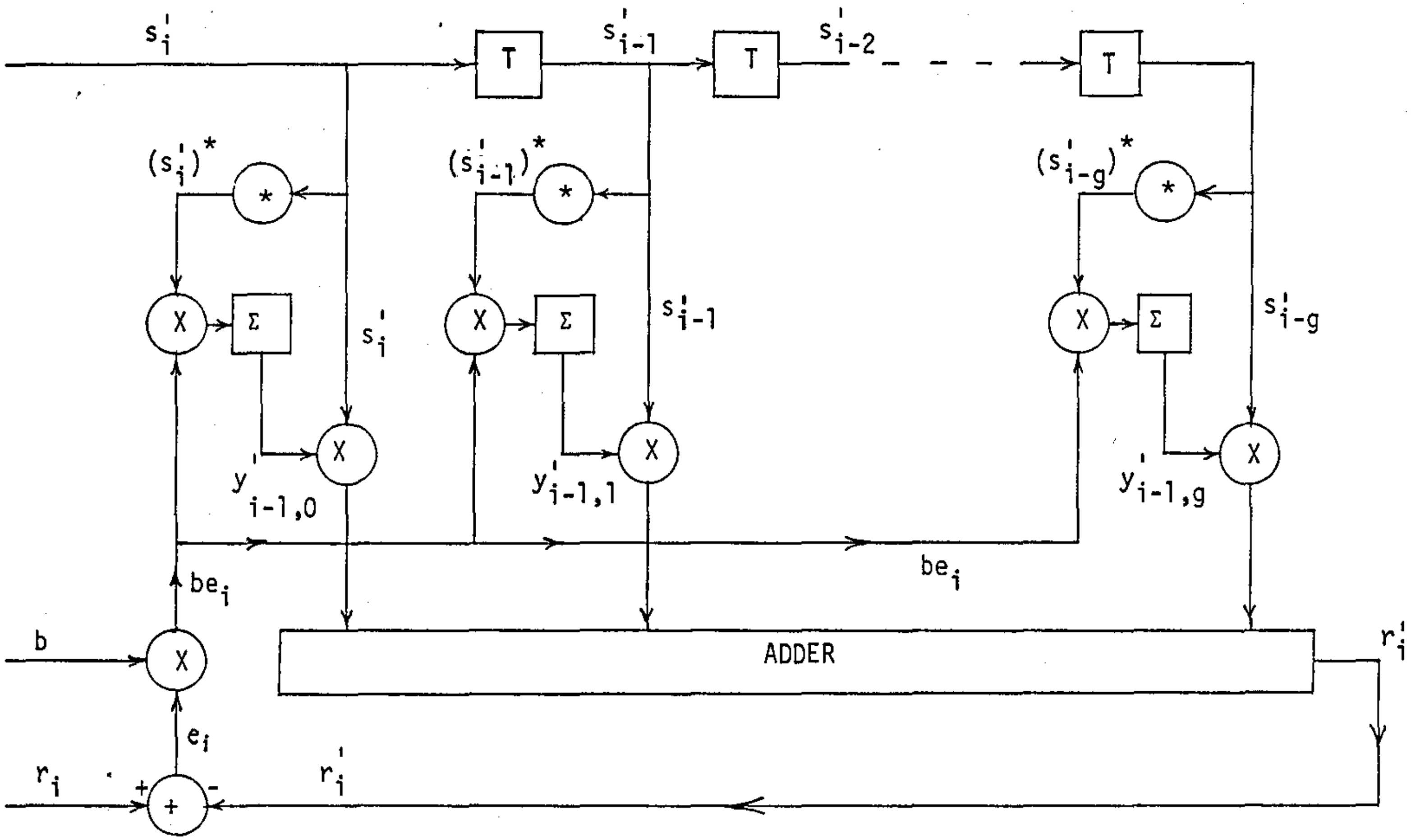


FIGURE 2.18: Linear feedforward channel estimator

The smaller the value of b, the smaller the effect of additive noise on Y_i' , but the slower the rate of response of Y_i' to changes in Y_i.

The quantity e_i can, if required, be constrained such that $|e_i| \leqslant c$ where c is a suitable positive constant. This constraint overrides the value of e_i given by $r_i - r_i'$, thus replacing an upper limit on the magnitude of e_i without however changing its sign.

2.4 Channel Equalization

In any serial data-transmission systems such as shown in Figure 2.1 where the channel introduces either one or both amplitude and phase distortion, the channel impulse-response is time-dispersive in that it has, for practical purposes, a duration of (g+1)T seconds where g is a positive integer and T seconds is the sampling interval. Thus, if the sampled impulse-response of the channel is a sequence of values $y_0, y_1, y_2, \ldots, y_g$ then the components of an individual received signal-element for the data-symbol s_i at the output of the sampler in Figure 2.1 are $s_iy_0, s_iy_1, s_iy_2, \ldots, s_iy_g$ which are received, in turn, at (g+1) consecutive sampling instants. Consequently, the received sample r_i at the output of the sampler contains not only s_i but also $s_{i-1}, s_{i-2}, s_{i-3}, \ldots, s_{i-g}$. This effect is caused by the overlapping of neighbouring signal-elements and is known as intersymbol interference. At high signal to noise ratios, such as in the voice channel, intersymbol interference becomes the primary impediment to the correct detection of the data-symbols $\{s_i\}$.

One way to overcome the effect of intersymbol interference in the received signal, thus enabling correct detection of the $\{s_i\}$ is by the use of equalizers[3,7,15-22,52-91]. The function of the equalizer is to correct the amplitude and phase distortions introduced by the channel and thereby restore the signal to being a copy of the original transmitted signal, hence acting as the inverse of the channel. The equalized signal, now free of intersymbol interference, can be detected very

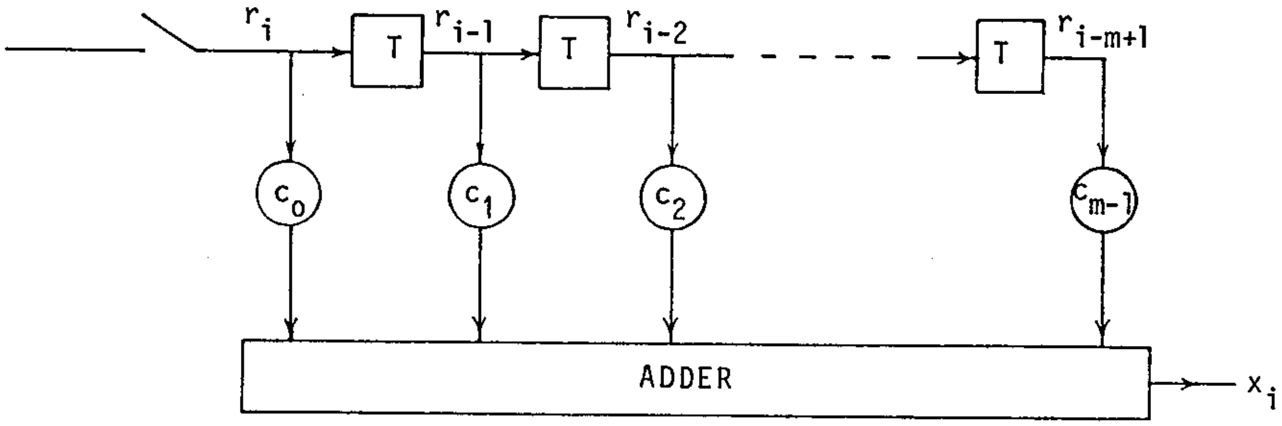


FIGURE 2.19a: Linear feedforward transversal equaliser

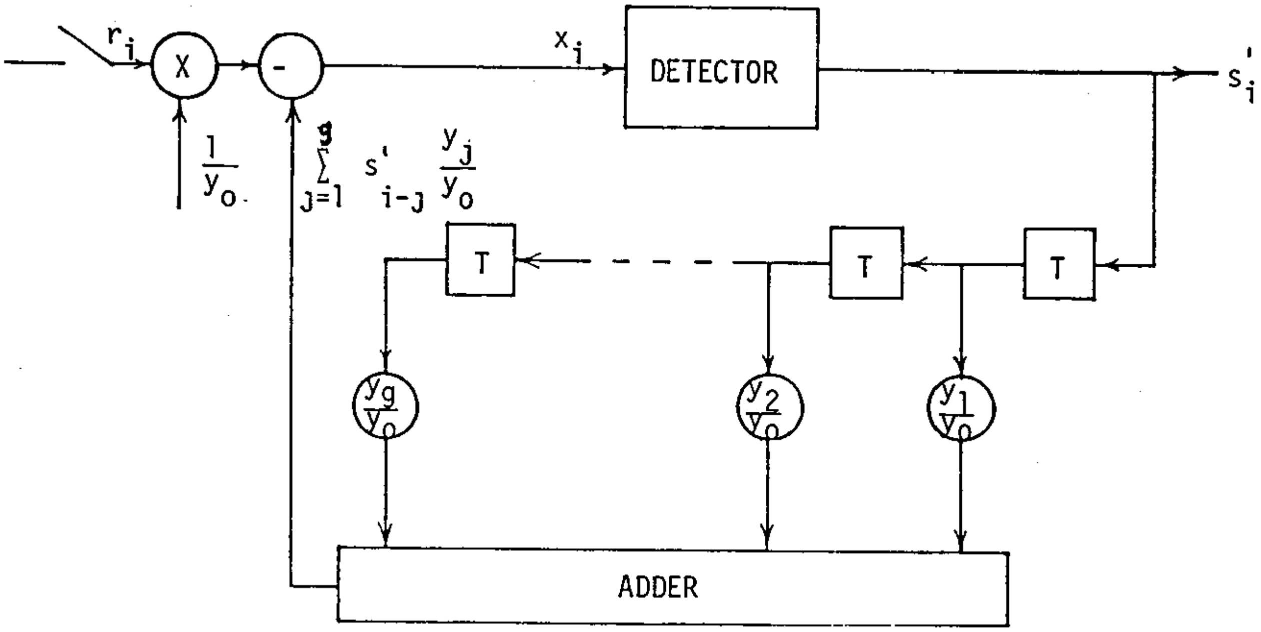


FIGURE 2.19b: Pure non-linear equaliser and detector

easily by comparing the corresponding sample-value with the appropriate threshold level (or levels). There are two main equalizer designs: the linear equalizer[51-73] and the non-linear equalizer[74-91], both of which can be adjusted for time-varying channels.

When a channel introduces pure phase distortion, it can be shown[7] that the linear feedforward transversal equalizer as shown in Figure 2.19a gives the best tolerance to additive white Gaussian noise. This equalizer, when correctly adjusted for pure phase distortion, acts as a matched filter that is matched to the channel and to each received signal-element, and in the presence of Gaussian noise, maximizes the signal to noise ratio at its output, and introduces no change in the level or other statistical properties of the noise samples, nor does it change the level of the data signal. However, in cases where the channel introduces both amplitude and phase disortions, the linear feedforward transversal equalizer no longer gives the best tolerance to Gaussian noise[7]. Indeed, when the z-transform of the sampled impulse-response of the channel Y(z), has all its roots inside the unit circle in the z-plane, at high signal to noise ratios when error extension effects can be neglected, the pure non-linear equalizer (Figure 2.19b) will always give a better tolerance to Gaussian noise than the linear equalizer. In general, when Y(z) has roots both inside and outside the unit circle, equalization of the channel is best carried out by the conventional non-linear equalizer[7]. The optimum design of the conventional non-linear equalizer will now be described.

2.4.1 Optimum Nonlinear Equalizer

When a channel is equalized by a linear equalizer (Figure 2.19a)the tolerance to additive white Gaussian noise of the system is determined by the noise variance at the equalizer output[7], subject to accurate equalization of the channel. The noise variance at the equalizer output is determined by the sum of the squares of the tap gains of the linear equalizer, and is given by

$$\eta^2 = \sum_{i=0}^{m-1} \sigma^2 c_i^2$$

where σ^2 is the variance of each input noise sample w_i and $c_o, c_1 \ldots, c_{m-1}$ are the coefficients of the m-tap linear feedforward transversal filter (Figure 2.19a).The pure nonlinear equalizer (Figure 2.19b on the other hand, does not alter the noise variance at its output when accurate equalization is achieved, and the tolerance to white Gaussian noise is a function of y_h, where y_h is the $(y+1)^{th}$ component of Y (equation 2.3), assuming that s_i is detected from the sample value x_{i+h}, at time $t = (i+h)T$, where h is an integer in the range 0 to g[7]. In general, when a channel contains both amplitude and phase distortions, an improvement in the tolerance to additive white Gaussian noise can be obtained by dividing the task of equalization between a linear equalizer (feedforward filter) and a pure nonlinear equalizer (feedback filter).

Consider the conventional nonlinear equalizer as shown in Figure 2.20) where the channel equalization is shared between the linear feedforward transversal filter and the nonlinear feedback transversal filter. There are various design techniques for combining the two filters and an infinite number of different combinations for achieving accurate equalization. [One such technique is the 'zero forcing' method[52,53,57,59] where the linear filter is used to set zero all the components preceding the largest in Y (equation 2.3).] Another method is to divide Y(z) into 'complementary factors'[7], so that

$$Y(z) = Y_1(z)\ Y_2(z) \qquad (2.25)$$

where all roots (zeros) of $Y_1(z)$ lie inside the unit circle in the z-plane, and all the roots (zeros) of $Y_2(z)$ lie outside. The linear filter now equalizes $Y_2(z)$ and the nonlinear filter equalizes $Y_1(z)$.

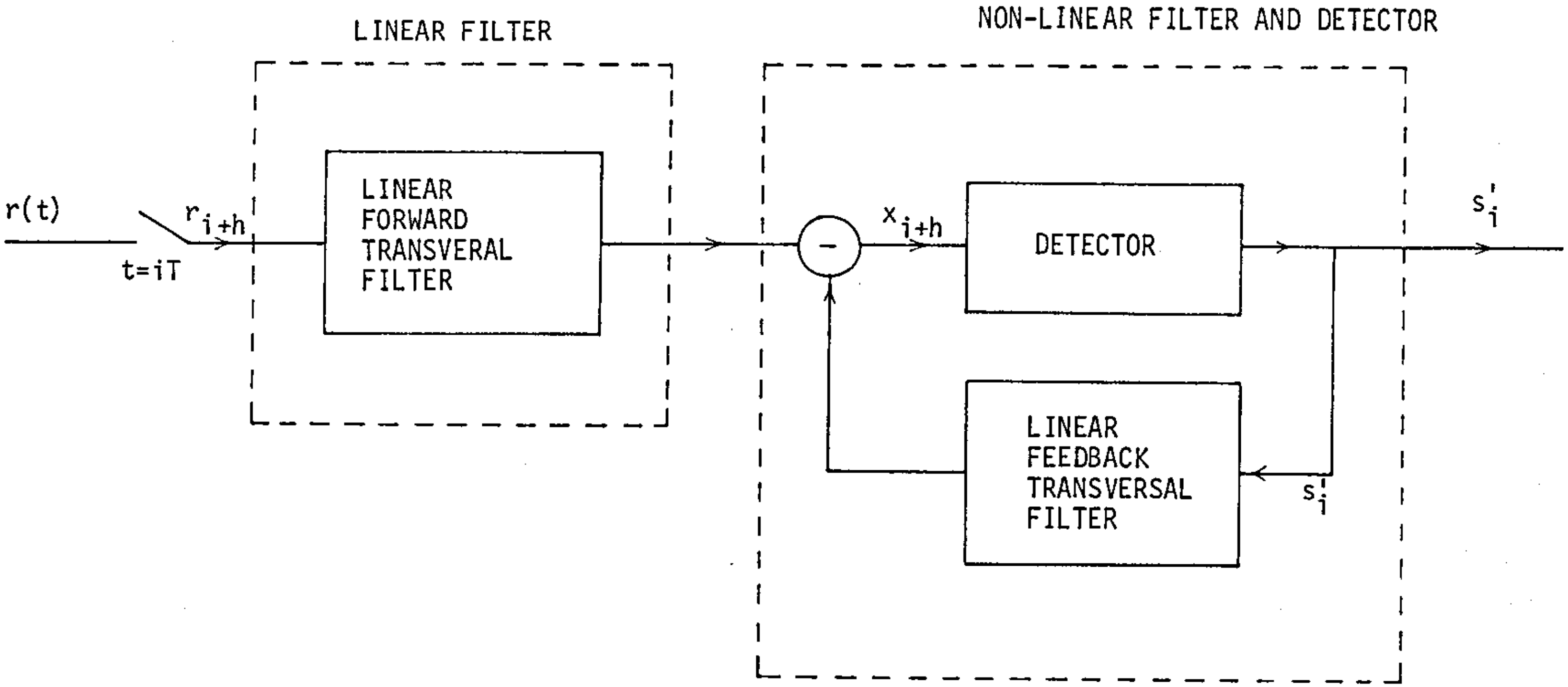

LINEAR FILTER
NON-LINEAR FILTER AND DETECTOR
r(t)
r_{i+h}
t=iT
LINEAR FORWARD TRANSVERAL FILTER
−
x_{i+h}
DETECTOR
s_i'
LINEAR FEEDBACK TRANSVERSAL FILTER
s_i'

FIGURE 2.20: Conventional non-linear equalizer

The optimum nonlinear equalizer is here defined as one which minimizes the mean-square error in the equalized signal subject to the accurate equalization of the channel. At high signal to noise ratios, this equalizer comes close to minimizing the error rate in the detected data-symbols subject to the accurate equalization of the channel[7].

Consider the (n+1)-tap linear feedforward transversal filter shown in Figure 2.20. Let the sampled impulse-response be

$$D = d_o \, d_1 \, \ldots \, d_n \tag{2.26}$$

and its z-transform be

$$D(z) = d_o + d_1 z^{-1} + d_2 z^{-2} + \ldots + d_n z^{-n} \tag{2.27}$$

Also let

$$D(z) = D'(z) \, H(z) \tag{2.28}$$

where

$$H(z) = 1 + h_1 z^{-1} + h_2 z^{-2} + \ldots + h_{n-m} z^{-n+m} \tag{2.29}$$

and D'(z) is the z-transform of the (m+1)-tap linear equalizer for the baseband channel with z-transform Y(z), such that

$$D'(z) \, Y(z) \simeq z^{-h}$$

or

$$D'(z) \simeq z^{-h} \, Y^{-1}(z) \tag{2.30}$$

where h is a non-negative integer in the range 0 to m+g.

The z-transform of the channel and linear filter is

$$Y(z)\ D(z) \approx Y(z)\ D'(z)\ H(z)$$

$$\approx z^{-h}\ H(z) \qquad (2.31)$$

so the sampled impulse-response of the channel and linear filter is given approximately by the (n-m+1)-component vector

$$H = [1 \quad h_1 \quad h_2 \ . \ . \ . \ h_{n-m}] \qquad (2.32)$$

neglecting the delay of hT seconds. Note that $\{h_i\}$ in equations 2.29 and 2.31 are not determined by the channel. The first component of H is fixed at unity, whereas its remaining n-m components $\{h_i\}$ may be selected to optimize the tolerance of the system to noise and they are removed by the nonlinear filter (Figure 2.20) with its (n-m) taps set equal to the $\{h_i\}$.

Thus, at time t = (i+h)T, the sample value at the output of the nonlinear filter is

$$x_{i+h} \approx s_i + u_{i+h} \qquad (2.33)$$

where u_{i+h} is the Gaussian noise component at the output. Since the nonlinear filter (Figure 2.20) does not change the statistical properties of the noise samples[7], u_{i+h} is also the noise sample at the nonlinear filter and its variance is given by

$$\eta^2 = \sigma^2 \, |D|^2 \tag{2.34}$$

where η^2 and σ^2 are the variances of the noise samples at the input and output, respectively, of the optimum nonlinear equalizer and $|D|^2$ is the length of the vector D (whose z-transform is D(z) (equation 2.27)). Clearly, to maximize the signal to noise ratio it is necessary to minimize η^2 which means that the tap gains D (equation 2.26) must be adjusted to minimize $|D|^2$.

For the 16-point QAM system where $s_i = s_{o,i} + js_{1,i}$ and $s_{o,i} = \pm 1$ or ± 3, $s_{1,i} = \pm 1$ or ± 3, it can be shown[7] that the average probability of error in the detection of $s_{o,i}$ or $s_{1,i}$, following the correct detection of the preceding n-m elements, is given by

$$\begin{aligned} p_e &= 1.5 \int_1^\infty \frac{1}{\sqrt{2\pi}\eta} \, \exp\left(\frac{-u^2}{2\eta^2}\right) du \\ &= 1.5 \, Q\left(\frac{1}{\eta}\right) \\ &= 1.5 \, Q\left(\frac{1}{\sigma|D|}\right) \end{aligned} \tag{2.35}$$

Again, from equation 2.35, to minimize the error rate in the detected element values $\{s_i'\}$, it is necessary to minimize $|D|$. The minimization is, of course, subject to the constraint imposed by equations 2.28 and 2.29 which imply the accurate equalization of the channel by the nonlinear filter.

Basically, the linear filter (whose z-transform is D(z)) used in the optimum nonlinear equalizer is an allpass network that does not change the amplitude distortion in the received signal but makes the sampled impulse-response of the channel and filter 'minimum phase', thus concentrating the energy of each received signal-element towards the start of that element[7]. Furthermore, the noise components at the output of the linear filter have the same statistical properties as those at its input, being statistically independent and with a fixed variance[7]. The linear filter, in essence, removes all roots (zeros) of Y(z) that lie outside the unit circle and replaces them by the complex conjugates of their reciprocals, while leaving all the remaining roots unchanged. This performs the same function as the 'whitened matched-filter' where the data signal is transmitted over the baseband channel (Figure 2.1) at the Nyquist rate[7,8,31].

Let the z-transform of the sampled impulse-response of the channel be

$$Y(z) = Y_1(z)\,Y_2(z)$$

$$= y_0 + y_1 z^{-1} + \dots + y_g z^{-g} \tag{2.36}$$

where

$$Y_1(z) = (1 + \alpha_1 z^{-1})(1 + \alpha_2 z^{-1}) \dots (1 + \alpha_{g-m} z^{-1})$$

$$= 1 + p_1 z^{-1} + p_2 z^{-2} + \dots + p_{g-m} z^{-g+m} \tag{2.37}$$

and

$$Y_2(z) = z^{-m}\ q_o\ (1 + \beta_1 z)(1 + \beta_2 z)\ \ldots\ (1 + \beta_m z)$$

$$= q_o + q_1 z^{-1} + \ldots + q_m z^{-m} \qquad (2.38)$$

It is assumed that no roots (zeros) of $Y(z)$ lie exactly on the unit circle, and $|\alpha_i| < 1$ and $|\beta_i| < 1$ so that $Y(z)$ has m roots (zeros) outside the unit circle.

The z-transform of the linear filter (Figure 2.20) that forms the first part of the optimum nonlinear equalizer is[7]

$$D(z) \simeq (q_m^*)^{-1}\ z^{-h}\ Y_2^{-1}(z)\ Y_3(z) \qquad (2.39)$$

where

$$Y_3(z) = q_m^* + q_{m-1}^*\ z^{-1} + \ldots + q_o^*\ z^{-m} \qquad (2.40)$$

and h is a positive integer to make $D(z)$ realisable. $Y_3(z)$ is obtained from $Y_2(z)$ by reversing the order of its coefficients and by taking the complex conjugatesof them. From the elementary theory of polynomials, the roots (zeros) of $Y_3(z)$ are the complex-conjugate reciprocals of the roots (zeros) of $Y_2(z)$. The z-transform of the sampled impulse-response of the channel and optimum nonlinear equalizer is

$$Y(z)\ D(z) \simeq Y_1(z)\ Y_2(z)\ (q_m^*)^{-1}\ z^{-h}\ Y_2^{-1}\ (z)\ Y_3(z)$$

$$= z^{-h}\ (q_m^*)^{-1}\ Y_1(z)\ Y_3(z) \qquad (2.41)$$

and all the roots (zeros) of Y(z) D(z) lie inside the unit circle. From equations 2.41, 2.37 and 2.40

$$Y(z)D(z) \simeq z^{-h}(q_m^*)^{-1}\ (1 + p_1 z^{-1} + p_2 z^{-2} + \ldots + p_{g-m} z^{-g+m})$$

$$\cdot(q_m^* + q_{m-1}^* z^{-1} + \ldots + q_o^*\ z^{-m})$$

$$= z^{-h}\ (1 + a_1 z^{-1} + a_2 z^{-2} + \ldots + a_g z^{-g}) \qquad (2.42)$$

and the first non-zero term of Y(z) D(z) has the value unity. Under these conditions the linear feedforward transversal filter, that forms the first part of the nonlinear equalizer, generally introduces a change in both the signal level and noise level, but always leaves the signal to noise ratio unchanged, and, apart from the level change, it performs a unitary transformation on the received signal, leaving the noise statistics unchanged[7,49].

When the nonlinear filter (Figure 2.20) is operated correctly with its g taps set to the coefficients, a_i, i=1, ..., g, given by Y(z) D(z) (equation 2.42), the sample value at the output of the detector, at t = (i+h)T, is

$$x_{i+h} = s_i + u_{i+h} \qquad (2.43)$$

where u_{i+h} is a Gaussian noise component with variance, η^2 equal to the input noise variance σ^2 multiplied by the sum of the squares of the moduli (absolute values) of the tap gains in the linear filter that forms the first part of the optimum nonlinear equalizer.

Thus,

$$\eta^2 = \sigma^2 \; |D|^2$$

$$= \sigma^2 \; |(q_m^*)^{-1} \; z^{-h} \; Y_2^{-1}(z) \; Y_3(z)|^2$$

$$= \sigma^2 \; |q_m|^{-2} \; |Y_2^{-1}(z) \; Y_3(z)|^2 \qquad (2.44)$$

Now, $Y_3(z)$ is formed by reversing the order of coefficients of $Y_2(z)$ and taking the complex conjugates of these. From equation 2.40,

$$Y_3(z) = q_m^* + q_{m-1}^* \; z^{-1} + \ldots + q_o^* \; z^{-m}$$

$$= z^{-m} \; (q_o^* + q_1^* \; z + \ldots + q_m^* \; z^m) \qquad (2.45)$$

From equation 2.38,

$$q_o + q_1 z^{-1} + \ldots + q_m z^{-m} = z^{-m} q_o (1 + \beta_1 z)(1 + \beta_2 z) \ldots (1 + \beta_m z)$$

Since $(A_1 + A_2)^* = A_1^* + A_2^*$ and $(A_1 A_2) = A_1^* \; A_2^*$ where $A_1 = (a_1 + jb_1)$, $A_2 = (a_2 + jb_2)$ where $j = \sqrt{-1}$ and a_1, a_2, b_1 and b_2 are real numbers. It follows that

$$q_o^* + q_1^* \; z^{-1} + \ldots + q_m^* \; z^{-m} = z^{-m} q_o^* \; (1 + \beta_1^* \; z)(1 + \beta_2^* \; z) \ldots (1 + \beta_m^* \; z)$$

and

$$q_o^* + q_1^* z + \ldots + q_m^* z^m = z^m q_o^* \ (1 + \beta_1^* z^{-1})(1 + \beta_2^* z^{-1})\ldots(1 + \beta_m^* z^{-1}) \tag{2.46}$$

Substituting equation 2.46 into equation 2.45,

$$Y_3(z) = z^{-m}(z^m q_o^* \ (1 + \beta_1^* z^{-1})(1 + \beta_2^* z^{-1}) \ \ldots \ (1 + \beta_m^* z^{-1}))$$

$$= q_o^* \ (1 + \beta_1^* z^{-1})(1 + \beta_2^* z^{-1}) \ \ldots \ (1 + \beta_m^* z^{-1}) \tag{2.47}$$

Now, from equations 2.37 and 2.47

$$Y_2^{-1} (z) \ Y_3(z) = (z^{-m} q_o \ (1 + \beta_1 z)(1 + \beta_2 z) \ \ldots \ (1 + \beta_m z))^{-1}$$

$$. \ q_o^* \ (1 + \beta_1^* z^{-1})(1 + \beta_2^* z^{-1}) \ \ldots \ (1 + \beta_m^* z^{-1})$$

$$= z^m \ (1 + \beta_1 z)^{-1}(1 + \beta_1^* z^{-1})(1 + \beta_2 z)^{-1}(1 + \beta_2^* z^{-1})\ldots$$

$$. \ (1 + \beta_m z)^{-1}(1 + \beta_m^* z^{-1}) \tag{2.48}$$

It has been shown in equations 2.15 and 2.16 that the sum of the squares of the moduli (absolute values) of the coefficients ofthe factor $(1 + \beta_1 z)^{-1}(1 + \beta_1^* z^{-1})$ is 1. Therefore it follows that the sum of the squares of the moduli of the coefficients of Y_2^{-1} (z) $Y_3(z)$ (equation 2.48) = 1, and equation 2.44 becomes

$$\eta^2 = \sigma^2 \ |q_m|^{-2} \ Y_2^{-1}(z) \ Y_3(z)|^2$$

$$\sigma^2/|q_m|^2 \tag{2.49}$$

From equation 2.35 the probability of error in the detected signal $\{s_i\}$ is approximately

$$P_e = 3 \ Q(\frac{1}{\eta})$$

$$= 3 \ Q(\frac{|q_m|}{\sigma}) \tag{2.50}$$

It can be shown[7] that the linear filter (whose z-transform is D(z) (equation 2.39)) that forms the first part of the optimum nonlinear equalizer when operating on a channel, with z-transform Y(z), introduces no distortion to the received signal, nor does it change the statistical properties of the noise samples. Furthermore, the signal to noise ratio of the received signal is greater when such a linear filter is used[7] and that the best signal to noise ratio is achieved when the linear filter is adjusted to account for all m of the roots (zeros) of Y(z) that lie outside the unit circle.

2.5 Maximum-Likelihood Detection

One of the weaknesses of channel equalization by the use of thé optimum nonlinear equalizer is that only a portion of the received signal-element is used in the detection of that element, the remaining part of that element being removed by cancellation and not involved in the detection process itself. The detection process that has the best tolerance to additive white Gaussian noise in the sense that it minimizes the probability of error in the detection of the complete message, must involve the whole of the received signal in a detection process for the complete message[7,11,13,14]. Clearly, a good detection process must involve at least the whole of a signal-element in a detection process. For our purposes the optimum detection process is taken to be a maximum-likelihood detector. When the transmitted data-symbols are statistically independent and equally likely to have any of their possible values and the Gaussian noise samples in the received signal are uncorrelated, the maximum-likelihood detector selects as the detected message the possible sequence of transmitted element (digit) values for which there is the minimum mean-square difference (cost) between the corresponding received data signal, for the given signal distortion but in the absence of noise, and the signal actually received. This detector minimizes the probability of error in the detection of the received message[7,15].

Let the i^{th} received signal value in the data-transmission system of Figure 2.1, at time t = iT, be

$$r_i = q_i + w_i \qquad (2.51)$$

where

$$q_i = \sum_{h=0}^{g} s_{i-h}\, y_h \qquad (2.52)$$

Assuming that transmission starts at time t=T and $s_i=0$ for $i\leqslant 0$. The components $\{r_i\}$, $\{q_i\}$, $\{w_i\}$, $\{s_i\}$ and $\{y_i\}$ are, in general, complex-valued quantities. The quantities $y_0, y_1, \ldots, y_g$ are the (g+1) components of the channel sampled impulse-response, the $\{s_i\}$ are the transmitted data-symbol values, and the $\{w_i\}$ are the noise components whose real and imaginary parts are statistically independent Gaussian random variables with zero mean and variance σ^2.

Let the total number of transmitted data-symbols be k, corresponding to a transmission period of kT seconds, the sample values $\{r_i\}$, $\{s_i\}$, $\{q_i\}$ and $\{w_i\}$ can be represented by the k-component vectors R_k, S_k, Q_k and W_k respectively, where

$$R_k = [r_1 \quad r_2 \quad \cdots \quad r_k] \tag{2.53}$$

$$S_k = [s_1 \quad s_2 \quad \cdots \quad s_k] \tag{2.54}$$

$$Q_k = [q_1 \quad q_2 \quad \cdots \quad q_k] \tag{2.55}$$

$$W_k = [w_1 \quad w_2 \quad \cdots \quad w_k] \tag{2.56}$$

Now, from equation 2.51

$$R_k = Q_k + W_k \tag{2.57}$$

Consider the 16-point QAM signals where there are 16 possible values of s_i given by equation 2.1. If each component s_i of S_k is equally likely to have any one of its 16 possible values and if the $k\{s_i\}$ are statistically independent, then the vector S_k is equally likely to have any one of its 16^k possible values, and so is the vector Q_k. Under these conditions the detection process that minimizes the probability of error in the detection of S_k from the received vector R_k, selects from the 16^k possible vectors $\{Q_k\}$ the vector at the minimum distance $|R_k-Q_k|$ from R_k and takes the corresponding vector S_k as the detected vector[7]. This detection process minimizes the probability of the incorrect detection of one or more of the $\{s_i\}$, from the received vector R_k, and at very high signal to noise ratios, the detection process comes close to minimizing the probability of error in the detection of any individual s_i[7].

In order to minimize the probability of error in detection, it is necessary to maximize the probability of correct detection P(c)[7], where

$$P(c) = \int_{-\infty}^{\infty} P(c/R_k)\ p(R_k)\ dR_k \qquad (2.58)$$

$P(c/R_k)$ is the conditional probability of a correct decision given the received vector R_k, and $p(R_k)$ is the probability density function of R_k. Since $p(R_k)$ is non-negative, it can be seen that P(c) is maximized by by maximizing $P(c/R_k)$ for every possible value of the vector R_k. For any given received vector R_k, $P(c/R_k)$ is maximized by selecting, as the detected vector, the possible vector S_k, for which the value of $P(S_k/R_k)$ is maximum, where $P(S_k/R_k)$ is the conditional probability of S_k given R_k and is thus the a posteriori probability of S_k. The optimum detection process is now reduced to the maximum a posteriori (MAP) detection process[92].

Since there is a one-to-one mapping between S_k and Q_k (equation 2.53)

$$P(S_k/R_k) = P(Q_k/R_k) \qquad (2.59)$$

Moreover, by Bayes' theorem[7],

$$P(Q_k/R_k) = \frac{P(Q_k)}{p(R_k)} p(R_k/Q_k) \qquad (2.60)$$

where $P(Q_k)$ is the a priori probability of Q_k, $p(R_k)$ is the probability density of R_k, and $p(R_k/Q_k)$ is the conditional probability density function of R_k given Q_k. For a given received vector R_k, $p(R_k)$ is independent of Q_k[92]. Therefore, from equations 2.59 and 2.60, maximizing $P(S_k/R_k)$ is equivalent to maximizing the product $P(Q_k)p(R_k/Q_k)$. When all the 16^k possible vectors $\{s_i\}$ are equally likely to be transmitted,

$$P(S_k) = P(Q_k) = 16^{-k} \qquad (2.61)$$

and the optimum detection process now reduces to the one that selects as the detected vector, the possible vector S_k for which the corresponding value of $p(R_k/Q_k)$ is maximized. $p(R_k/Q_k)$ is often known as the likelihood function of Q_k and the optimum detection process, under the assumed conditions, is thus the maximum-likelihood process.

Since $p(R_k/Q_k)$ is the conditional joint probability density function of the random variable with sample values $r_1, r_2, \ldots, r_k$ (equation 2.52) given the values $q_1, q_2, \ldots, q_k$ (equation 2.57) it can be written

$$p(R_k/Q_k) = p(r_1, r_2, .., r_k/q_1, q_2, \ldots, q_k) \qquad (2.62)$$

Let the complex values $\{r_i\}$ and $\{q_i\}$ be

$$r_i = r_{i,1} + j\, r_{i,2} \qquad (2.63)$$

$$q_i = q_{i,1} + j\, q_{i,2} \qquad (2.64)$$

for $i = 1, 2, \ldots, k$ and $j = \sqrt{-1}$.

For a given received vector S_k and hence for a given received vector Q_k, the received samples $\{r_i\}$ are statistically independent Gaussian random variables. Thus

$$P(R_k/Q_k) = p(r_{1,1}/q_{1,1})\; p(r_{1,2}/q_{1,2}) \cdots$$

$$\cdots\; p\,(r_{k,1}/q_{k,1})\; p(r_{k,2}/q_{k,2}) \qquad (2.65)$$

Furthermore, the real and imaginary parts of the noise components $\{w_i\}$ in equation 2.51 are statistically independent Gaussian random variables with zero mean and variance σ^2, therefore for a given value of $q_{i,h}$, $r_{i,h}$ is a Gaussian random variable with mean $q_{i,h}$ and variance σ^2. That is

$$p\,(r_{i,h}/q_{i,h}) = \frac{1}{\sqrt{2\pi\sigma^2}}\; \exp\,(\frac{-(r_{i,h} - q_{i,h})^2}{2\sigma^2}) \qquad (2.66)$$

for $h = 1, 2$ and $i = 1, 2, \ldots, k$.

Substituting equation 2.66 into 2.65

$$p\ (R_k/Q_k) = \prod_{i=1}^{k} \frac{1}{\sqrt{2\pi\sigma^2}}\ \exp\ (\frac{-\ (r_{i,1} - q_{i,1})^2}{2\ \sigma^2})$$

$$\cdot \prod_{i=1}^{k} \frac{1}{\sqrt{2\pi\sigma^2}}\ \exp\ (\frac{-\ (r_{i,2} - q_{i,2})^2}{2\ \sigma^2})$$

$$= \frac{1}{(\sqrt{2\pi\sigma^2})^k} \exp\ (\frac{-1}{2\sigma^2}\ \sum_{i=1}^{k}\ (r_{i,1}-q_{i,1})^2+(r_{i,2}-q_{i,2})^2)$$

$$= \frac{1}{(2\pi\sigma^2)^{k/2}}\ \exp\ (-\ \frac{|R_k - Q_k|^2}{2\ \sigma^2}) \qquad (2.67)$$

where $|R_k - Q_k|^2 = \sum_{i=1}^{k} |r_i - q_i|^2$

$$= \sum_{i=1}^{k}\ ((r_{i,1} - q_{i,1})^2 + (r_{i,2} - q_{i,2})^2) \qquad (2.68)$$

and $|R_k - Q_k|$ is the length of vector $(R_k - Q_k)$ and so is the unitary distance between the vectors R_k and Q_k. It can be seen from equation 2.67 that the value of the likelihood function $p(R_k/Q_k)$ is minimum when the unitary distance, $|R_k - Q_k|$, is minimum. Hence the maximum-likelihood detection process, under the assumed conditions, selects as the detected vector the possible vector S_k for which the corresponding signal vector Q_k is closest (interms of the unitary distance) to the received vector.

The detection process is implemented as follows. The receiver holds in store R_k, the k received samples $\{r_i\}$ and it generates in turn each of the 16^k (for the 16-point QAM signals) possible S_k. It then forms the vectors corresponding to Q_k using equation 2.52, and for each of these it measures the unitary distance, $|R_k - Q_k|$, from R_k to Q_k. Each time the measured unitary distance is smaller than the smallest unitary distance previously measured, the new unitary distance and the vector S_k are stored, replacing the corresponding quantities stored. Whenever the measured unitary distance is greater than or equal to the stored unitary distance, the new unitary distance together with the associated vector S_k are ignored. Thus at the end of the process the stored vector S_k is the required vector.

An error occurs in the detection of S_k when the received signal vector Q_k is not the one of the possible 16^k possible vectors $\{Q_k\}$that is closest to R_k. This necessarily involves one or more errors in the detected data signal values $\{s_i\}$. The exact evaluation of the probability of error in the detection of S_k is difficult, but an approximate upper bound to this average probability of error in the detection of the real or imaginary parts of $\{s_i\}$, at high signal to noise ratios may be obtained as $Q(\frac{d_{min}}{2})$[7], where $Q(\cdot)$ is the well-known Q-function, σ^2 is the variance of the real and imaginary parts of the noise components $\{w_i\}$, and d_{min} is the minimum unitary distance between any two possible received signal vectors $\{Q_k\}$.

2.5.1 The Viterbi-Algorithm Detection Process

The maximum-likelihood detection process requires the storage of 16^k possible data-symbol vectors $\{S_k\}$ and the same number of unitary distance measurements for a 16-point QAM signal and a sequence of transmitted data-symbols $s_1, s_2, \ldots, s_k$. Since k is usually very large, this method becomes too complex to be of any practical use. Fortunately, the maximum-likelihood detection process can be realised by the Viterbi-algorithm which not only saves storage but also a very large number of unnecessary calculations[7,93]. Its operation will now be described.

At time t = (i-1)T, just before the receipt of the sample r_i at the detector input (Figure 2.21), the Viterbi-algorithm detector holds in store

r(t)
r_i
Viterbi-algorithm detector
$\{s_i'\}$

FIGURE 2.21: Viterbi-algorithm Detector

m_v (i-1)-component vectors $\{Z_{i-1}\}$, where

$$Z_{i-1} = [x_1 \; x_2 \; . \; . \; . \; x_{i-1}] \qquad (2.69)$$

with their corresponding m_v costs $\{C_{i-1}\}$, where x_h has one of the 16 possible values of the complex-valued data-symbol s_h. The number m_v is suitably chosen and will be discussed shortly. On receiving r_i, each of the m_v $\{Z_{i-1}\}$ is expanded into 16 vectors $\{Z_i\}$, where

$$Z_i = [\underbrace{x_1 \; x_2 \; . \; . \; . \; x_{i-1}}_{Z_{i-1}} \; x_i] \qquad (2.70)$$

The first (i-1) components in Z_i are as in the original vector Z_{i-1} and the last component x_i takes on the 16 different possible values of s_i in the 16 expanded vectors $\{Z_i\}$ as shown in Figure 2.22. The cost

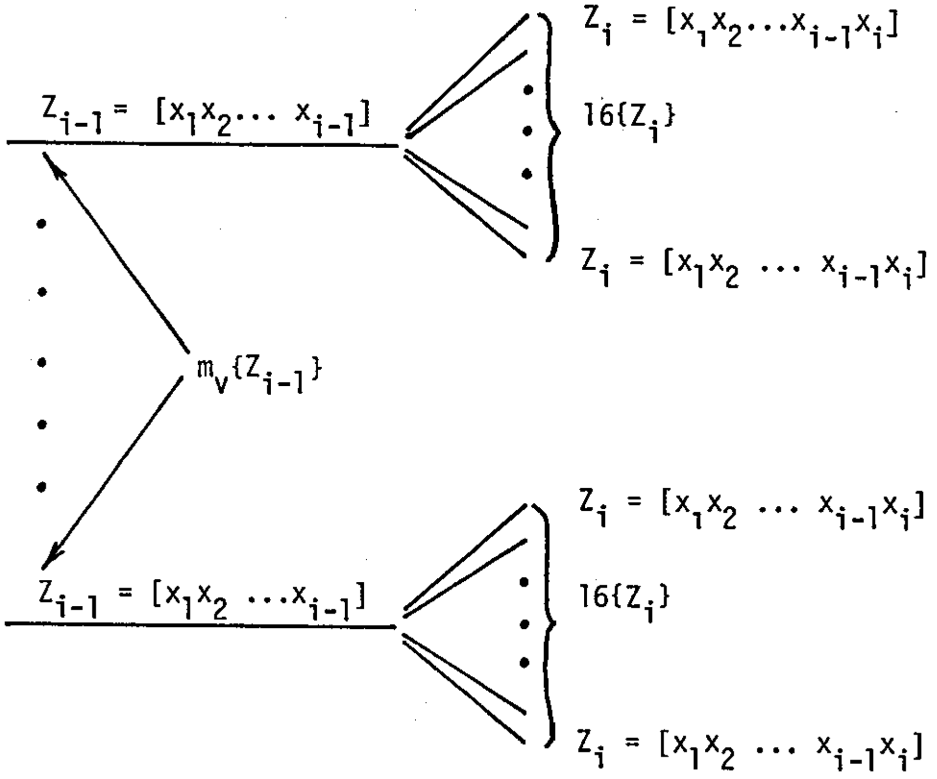

$Z_{i-1} = [x_1x_2 \dots x_{i-1}]$
$Z_i = [x_1x_2 \dots x_{i-1}x_i]$
$16\{Z_i\}$
$Z_i = [x_1x_2 \dots x_{i-1}x_i]$
$m_v\{Z_{i-1}\}$
$Z_{i-1} = [x_1x_2 \dots x_{i-1}]$
$Z_i = [x_1x_2 \dots x_{i-1}x_i]$
$16\{Z_i\}$
$Z_i = [x_1x_2 \dots x_{i-1}x_i]$

FIGURE 2.22: Each Vector $\{Z_i\}$ is Expanded into 16 Vectors $\{Z_i\}$

C_i associated with each expanded vector Z_i is defined as

$$C_i = c_1 + c_2 + \dots + c_i \tag{2.71}$$

where

$$c_1 = |r_1 - (x_1y_o + x_oy_1)|^2 \tag{2.72}$$

$$c_2 = |r_2 - (x_2y_o + x_1y_1 + x_oy_2)|^2 \tag{2.73}$$

$$c_3 = |r_3 -(x_3y_o + x_2y_1 + x_1y_2 + x_oy_3)|^2 \tag{2.74}$$

etc.

In general,

$$c_j = |r_j - \sum_{h=0}^{g} x_{j-h}\, y_h|^2 \tag{2.75}$$

where j = 1,2,3..., and $x_{j-h} = 0$ for $j-h \leqslant 0$. The complex-valued quantities y_0, y_1, ..., y_g are, of course, the (g+1) components of the channel sampled impulse-response Y (equation 2.3). Therefore,

$$C_i = C_{i-1} + c_i \tag{2.76}$$

where C_{i-1} has already been evaluated previously and c_i is evaluated by equation 2.75.

Recall from equations 2.52 and 2.68

$$q_i = \sum_{h=0}^{g} s_{i-h}\, y_h \tag{2.77}$$

$$|R_k-Q_k|^2 = \sum_{i=1}^{k} |r_i-q_i|^2 = \sum_{i=1}^{k} |r_i - \sum_{h=0}^{g} s_{i-h}\, y_h|^2 \tag{2.78}$$

it can be seen from equations 2.75 and 2.78 that the cost C_i is actually identical to the square of the unitary distance, $|R_k-Q_k|^2$, between the received vector R_i (equation 2.53) and the possible vector of Q_i (equation 2.55) corresponding to the possible vector of S_i (equation 2.54) that has the same components as those of the expanded vector Z_i (equation 2.70). Thus, in the original maximum-likelihood detection process, all the expanded vectors $\{Z_i\}$ are stored and used for the following signal processing until the end of the transmission when the vector associated with the minimum cost is taken as the detected data-symbol vector. In the Viterbi-algorithm detection process, however, only m_v of the 16 m_v expanded vectors $\{Z_i\}$ are selected and stored, while the rest, being redundant, are discarded from further consideration. The value m_v will now be derived.

From equation 2.75

$$\vdots$$

$$c_{i-3} = r_{i-3} - (x_{i-3}y_0 + x_{i-4}y_1 + \dots + x_{i-g-3}y_g) \tag{2.79}$$

$$c_{i-2} = r_{i-2} - (x_{i-2}y_0 + x_{i-3}y_1 + \dots + x_{i-g-2}y_g) \tag{2.80}$$

$$c_{i-1} = r_{i-1} - (x_{i-1}y_0 + x_{i-2}y_1 + \dots + x_{i-g-1}y_g) \tag{2.81}$$

$$c_{i} = r_{i} - (x_{i}y_0 + x_{i-1}y_1 + \dots + x_{i-g}\ y_g) \tag{2.82}$$

$$c_{i+1} = r_{i+1} - (x_{i+1}y_0 + x_{i}y_1 + \dots + x_{i-g+1}y_g) \tag{2.83}$$

$$c_{i+2} = r_{i+2} - (x_{i+2}y_0 + x_{i+1}y_1 + \dots + x_{i-g+2}y_g) \tag{2.84}$$

$$c_{i+3} = r_{i+3} - (x_{i+3}y_0 + x_{i+2}y_1 + \dots + x_{i-g+3}y_g) \tag{2.85}$$

$$\vdots$$

It can be seen from equations 2.83-85 that c_{i+1}, c_{i+2}, ... are independent of the values of x_1, x_2, ..., x_{i-g}, where $i \geqslant g$. Consider the case where there are k data-symbols transmitted, so that

$$Z_k = [x_1x_2 \cdots\ x_{i-g-1}\ x_{i-g}\ x_{i-g+1} \cdots x_i \cdots x_k] \tag{2.86}$$

since x_h has one of the 16 possible values of the complex-valued data-symbol s_h, therefore, there are a total of 16^k possible vectors $\{Z_k\}$.

Suppose that there is a set of n vectors $\{Z_i\}$ (equation 2.70) which have the same set of values for the last g components x_{i-g+1}, x_{i-g+2}, ..., x_i, where $i \geqslant g$, and, in general, the $n\{Z_i\}$ will have different costs, C_i, associated with them. Now, the vectors $\{Z_k\}$ (equation 2.86) originated from this set of $n\{Z_i\}$ must have the same values for c_{i+1}, c_{i+2}, ... c_k (equations 2.83-2.85) so long as these $\{Z_i\}$ have the same set of values for x_{i+1}, X_{i+2}, ..., x_k. Since the cost, C_k, of the vector Z_k is

$$C_k = c_1 + c_2 + \dots + c_i + c_{i+1} + c_{i+2} + \dots + c_k \qquad (2.87)$$

which can be written as

$$C_k = C_i + c_{i+1} + c_{i+2} + \dots + c_k \qquad (2.88)$$

It now follows from equation 2.88 that any of the vectors $\{Z_k\}$ (which have the same set of values for c_{i+1}, c_{i+2}, ..., c_k) that has a larger value of C_i must also have a larger value of C_k. However, only the vector Z_k associated with the minimum cost C_k will be taken as the detected data-symbol vector in the maximum-likelihood detection process.

It therefore follows that for a given set of vectors $\{Z_k\}$ that have the same set of last g components x_{i-g+1}, x_{i-g+2}, ..., x_i, all vectors $\{Z_k\}$originating from these $\{Z_i\}$, except the Z_i associated with the smallest cost C_i for this set, will never be selected as the detected data-symbol vector and can be discarded from further consideration. Consequently, for those vectors of the $\{Z_i\}$ that have the same set of last g components it is necessary to store just the vector that has the smallest cost C_i. For a 16-point QAM signal, there are altogether 16^g possible combinations for the g components x_{i-g+1}, x_{i-g+2}, ..., x_i of Z_i and the total number of vectors $\{Z_i\}$ required to be stored here is therefore

$$m_v = 16^g \tag{2.89}$$

where g+1 is the number of components in the channel sampled impulse-response. Thus, the Viterbi-algorithm detection process selects the 16 m_v expanded vectors $\{Z_i\}$, the m_v vectors associated with the smallest cost $\{C_i\}$, subject to the constraint that these m_v selected vectors have the same m_v possible combinations for the last g components of Z_i. The detection process continues in this way, so that each time a sample is received, 16 m_v cost measurements are carried out for the 16 m_v expanded vectors, and m_v vectors are then selected in the way described before. At the end of the transmission, the stored vector Z_k associated with the minimum cost C_k is then taken as the detected data-symbol vector which is obviously the same as that obtained in the maximum-likelihood detection process. The maximum-likelihood detection process can therefore be implemented by the Viterbi-algorithm detection just described without any loss in tolerance to noise.

CHAPTER 3

THE ON-LINE SYSTEM

This Chapter is concerned with the development of techniques for adjusting the linear filter ahead of the detector, which involves various methods for finding the roots (zeros) of Y(z) (eqn. 2.4) that lie outside the unit circle in the z-plane. The techniques considered here are on-line systems, where the receiver (Figure 2.1) operates on the continuous stream of received samples $\{r_i\}$ which are corrupted by additive white Gaussian noise. The received samples $\{r_i\}$, after some preliminary processing by the receiver, are operated on by the two-tap feedforward and two-tap feedback filters in a particular manner to give an output sequence. The latter sequence is manipulated to give an estimate of β_1 (negative reciprocal of a root (zero) of Y(z) outside the unit circle) which is a factor of the tap gains of the linear filter ahead of the detector. In theory, the tap gains of the filter can be calculated from the values of all of the roots (zeros) of Y(z) outside the unit circle. This Chapter is concerned only with the estimation of one single root of Y(z) outside the unit circle, and in cases where there are m rootsoutside the unit circle they are treated as m separate and distinct cases, represented by a set of m polynomials, {Y(z)}, with m,(m-1),...,1 root (zero) outside the unit circle.

A feasibility study is first carried out in order to gain an insight into the problem and to ascertain the difficulties associated with this mode of operation. For this purpose and for the simplicity and ease in checking and interpreting the results, the computer-simulation tests are initially based on a synchronous serial binary data-transmission system[6,7] which is equivalent to the QAM system previously described except that all signals are now real-valued instead of complex-valued, and, of course, s_i has two possible values, ±1.

Four channels are used for this investigation and all of them are specified by a three-component real-valued vector Y (the sampled impulse-response) whose z-transform Y(z) has one root outside the unit circle with magnitude range from approximately 1.43 to 10.

The latter part of this Chapter describes the results of further studies of the techniques developed from the feasibility test, and its modified versions, when applied to the 16-point QAM system operating at 9600 bit/s.

3.1 Feasibility Study Based on the Synchronous Serial Binary Data-Transmission System

Consider the arrangement of the synchronous serial binary data-transmission system as shown in Figure 3.1 which is used in this part of the investigation for the feasibility test. The baseband channel is as described in Chapter 2, which includes a transmitter filter, a transmission path and a receiver filter. The transmission path could be a baseband or passband channel such as a telephone circuit. In the latter case, the transmission path is assumed to include a linear modulator at the transmitter and a linear demodulator at the receiver. The information to be transmitted is carried by the data-symbols $\{s_i\}$, where

$$s_i = \pm 1 \tag{3.1}$$

It is assumed that $s_i = 0$ for $i \leqslant 0$, so that the impulse $s_i\delta(t-iT)$ is the i^{th} signal-element at the input to the transmitter filter. Furthermore, the $\{s_i\}$ (for $i>0$) are statistically independent and equally likely to have any of their two possible values. The impulse-response y(t) of the linear baseband channel is real-valued and has, for practical purposes, a finite duration of (g+1)T seconds, where g is a positive integer and T seconds is the sampling interval. It is assumed that y(t) is known and time-invariant over any one transmission.

Let the sampled impulse-response of the baseband channel be

$$Y = [y_0 \;\; y_1 \;\; . \;\; . \;\; . \;\; y_g] \tag{3.2}$$

and its z-transform be

$$Y(z) = y_0 + y_1 z^{-1} + . \; . \; . + y_g z^{-g} \tag{3.3}$$

where $y_i = y\{iT\}$ and the $\{y_i\}$ are real-valued. The delay in the transmission is, for convenience, neglected here so that $y_i = 0$ for $0 > i > g$. The received sample, at time $t = iT$, is

$$r_i = \sum_{h=0}^{g} s_{i-h} \, y_h + w_i \tag{3.4}$$

where $r_i = r(iT)$ is the sampled value of the received waveform $r(t)$ and $w_i = w(t)$ is the sampled value of a Guassian noise waveform $w(t)$. It is assumed that $\{w_i\}$ are statistically independent real-valued Gaussian variables with zero mean and fixed variance σ^2.

Let the delay introduced by the (n+1)-tap adaptive linear filter be n and let the delay caused by the detector, for convenience, be zero so that the total delay in detection is nT. Thus, s_i is detected at time $t = (i+n)T$.

At time $t = (i+n)T$, the buffer store in Figure 3.1 holds a sequence of (n+1) samples of $\{r_i\}$ of the received signal given by the vector

$$R_{i+n} = [r_i \quad r_{i+1} \quad . \; . \; . \quad r_{i+n}] \qquad (3.5)$$

where

$$r_{i+h} = s_{i+h}y_o + s_{i+h-1}y_1 + \ldots + s_{i+h-g}y_g + w_{i+h}$$

$$= \sum_{j=0}^{g} s_{i+h-j}y_j + w_{i+h} \qquad (3.6)$$

The receiver uses the detected data symbols s'_{i-1}, s'_{i-2}, ..., s'_{i-g} together with the channel estimate of y_o, y_1, ..., y_g (which are assumed to be provided correctly at the receiver) to form the terms corresponding to the intersymbol interference in R_{i+n} (equation 3.5). Therefore, assuming the correct detection of s_{i-1}, s_{i-2}, ..., s_{i-g} together with the correct estimation of Y (equation 3.2), the intersymbol-interference synthesizer forms, at time t = (i+n)T, the vector

$$Z_{i+n} = [z_i \quad z_{i+1} \; . \; . \; . \quad z_{i+n}] \qquad (3.7)$$

at its output, where

$$z_{i+h} = s_{i-1}\,y_{h+1} + s_{i-2}\,y_{h+2} + \ldots + s_{i-g+h}\,y_g$$

$$= \sum_{j=1}^{g-h} s_{i-j}\,y_{h+j} \qquad (3.8)$$

The vector Z_{i+n} is subtracted from R_{i+n} to give the vector

$$R'_{i+n} = R_{i+n} - Z_{i+n}$$

$$= [r'_i \quad r'_{i+1} \; . \; . \; . \quad r'_{i+n}] \qquad (3.9)$$

which is held in the buffer store as shown in Figure 3.1.

Now, from equation 3.6

$$r_i = s_i y_o + s_{i-1} y_1 + s_{i-2} y_2 + s_{i-3} y_3 + \ldots + s_{i-g} y_g + w_i \qquad (3.10)$$

$$r_{i+1} = s_{i+1} y_o + s_i y_1 + s_{i-1} y_2 + s_{i-2} y_3 + \ldots + s_{i-g+1} y_g + w_{i+1} \qquad (3.11)$$

$$r_{i+2} = s_{i+2} y_o + s_{i+1} y_1 + s_i y_2 + s_{i-1} y_3 + \ldots + s_{i-g+2} y_g + w_{i+2} \qquad (3.12)$$

$$r_{i+3} = s_{i+3} y_o + s_{i+2} y_1 + s_{i+1} y_2 + s_i y_3 + \ldots + s_{i-g+3} y_g + w_{i+3} \qquad (3.13)$$

.
.
.
etc

and from equation 3.8

$$z_i = s_{i-1} y_1 + s_{i-2} y_2 + \ldots + s_{i-g} y_g \qquad (3.14)$$

$$z_{i+1} = s_{i-1} y_2 + s_{i-2} y_3 + \ldots + s_{i-g+1} y_g \qquad (3.15)$$

$$z_{i+2} = s_{i-1} y_3 + s_{i-2} y_4 + \ldots + s_{i-g+2} y_g \qquad (3.16)$$

$$z_{i+3} = s_{i-1} y_4 + s_{i-2} y_5 + \ldots + s_{i-g+3} y_g \qquad (3.17)$$

.
.
.
etc

It follows that (equation 3.9)

$$r'_i = s_i y_o + w_i \tag{3.18}$$

$$r'_{i+1} = s_{i+1} y_o + s_i y_1 + w_{i+1} \tag{3.19}$$

$$r'_{i+2} = s_{i+2} y_o + s_{i+1} y_1 + s_i y_2 + w_{i+2} \tag{3.20}$$

$$r'_{i+3} = s_{i+3} y_o + s_{i+2} y_1 + s_{i+1} y_2 + s_i y_3 + w_{i+3} \tag{3.21}$$

Clearly, the components of the vector R'_{i+n} (equation 3.9) contains no intersymbol interference caused by the previous detected data symbols. Thus, at time t = (i+n)T, the adaptive linear filter has, at its input, a set of samples $\{r'_j\}$, j=i, i+1,...,i+n, which are obtained from the corresponding n+1 received samples $\{r_j\}$, through the removal of the intersymbol interference.

Note that in the arrangement (Figure 3.1), the removal of the intersymbol interference components is carried out before the adaptive linear filter whereas in the non-linear equalizer, this operation is carried out after the filter and just before the detector (Figure 2.20). This has the advantage of avoiding the problem of coupling between the adjustment of the adaptive linear filter and the cancellation of intersymbol interference, since, in this case, the decision-directed cancellation of intersymbol interference is carried out ahead of the adaptive linear filter and therefore the adjustment of the filter does not affect the adjustment of the 'intersymbol-interference synthesizer'.

Suppose that the channel whose z-transform Y(z) has one root (zero) outside the unit circle in the z-plane such that

$$Y(z) = y_o + y_1 z^{-1} + \ldots + y_g z^{-g}$$

$$= (1 + \beta_1 z)(u_o z^{-1} + u_1 z^{-2} + \ldots + u_{g-1} z^{-g}) \qquad (3.22)$$

where $|\beta_1| < 1$, so that $-\frac{1}{\beta_1}$ is the root (zero) of Y(z) that lies outside the unit circle and β_1 is its negative reciprocal.

Let $\tilde{D}(z)$ be the z-transform of the linear filter (Figure 3.1) ahead of the detector which is constrained to be an all-pass filter at all times. From equation 2.14

$$\tilde{D}(z) = z^{-n} (\lambda_i + z^{-1})^{-1} (1 + \lambda_i^* z^{-1})$$

$$= z^{-n+1} (1 + \lambda_i z)^{-1} (1 + \lambda_i^* z^{-1})$$

$$= z^{-n+1} (1 - \lambda_i z + \lambda_i^2 z^2 - \ldots)(1 + \lambda_i^* z^{-1}) \qquad (3.23)$$

Thus the z-transform of the channel and filter is

$$Y(z)\tilde{D}(z) = (1 + \beta_1 z)(u_o z^{-1} + u_1 z^{-2} + \ldots + u_{g-1} z^{-g})$$

$$. z^{-n+1} (1 + \lambda_i z)^{-1} (1 + \lambda_i^* z^{-1})$$

$$= z^{-n+1} (1 + \beta_1 z)(1 + \lambda_i z)^{-1}$$

$$. (u_o z^{-1} + (u_1 + \lambda_i^* u_o) z^{-2} + \ldots + (u_{g-1} + \lambda_i^* u_{g-2}) z^{-g} + \lambda_i^* u_{g-1} z^{-g-1})$$

$$= z^{-n} (1 + \beta_1 z)(1 + \lambda_i z)^{-1} (q_o + q_1 z^{-1} + \ldots + q_g z^{-g})$$

$$(3.24)$$

where $$q_o = u_o \tag{3.25}$$

$$q_1 = u_1 + \lambda_i^* u_o \tag{3.26}$$

$$\vdots$$

$$q_{g-1} = u_{g-1} + \lambda_i^* u_{g-2} \tag{3.27}$$

$$q_g = \lambda_i^* u_{g-1} \tag{3.28}$$

In general

$$q_h = u_h + \lambda_i^* u_{h-1} \tag{3.29}$$

where $0 \leqslant h \leqslant g$, and $u_h = 0$ for $0 > h \geqslant g$.

Suppose now that $\lambda_i = \beta_1$,

$$\begin{aligned}\tilde{D}(z) &= z^{-n+1} (1 + \beta_1 z)^{-1} (1 + \beta_1^* z^{-1}) \\ &= z^{-n} (z(1 + \beta_1 z)^{-1}(1 + \beta_1^* z^{-1})) \\ &= z^{-n} (\beta_1 + z^{-1})^{-1}(1 + \beta_1^* z^{-1}) \qquad (3.30)\end{aligned}$$

and $\tilde{D}(z)$ is now the same as the z-transform of the sampled impulse-response of the (n+1)-tap ideal linear filter, D(z) (equation 2.14), for Y(z) (equation 3.3) which has one root outside the unit circle. From equations 3.24 and 3.30

$$\begin{aligned}Y(z)\,\tilde{D}(z) &= z^{-n} (q_o + q_1 z^{-1} + \ldots + q_g z^{-g}) \\ &= z^{-n} (1 + \beta_1^* z^{-1})(u_o + u_1 z^{-1} + \ldots + u_{g-1} z^{-g+1}) \qquad (3.31)\end{aligned}$$

Thus, in the z-transform of the resultant sampled impulse-response of the channel and filter, the root $-\frac{1}{\beta_1}$ is removed and replaced by its complex conjugate-reciprocal, $-\beta_1^*$.

The adaptive linear filter (Figure 3.1) is, in fact, implemented by means of the simple combination of the two-tap feedforward transversal filter (Figure 2.2) and the one-tap feedback transversal filter (Figure 2.3) as shown in Figure 3.2, together with a buffer store.

The z-transformsof the feedforward and feedback filters are (equations 2.18 and 2.19)

$$B_i(z) = 1 + \lambda_i^* z^{-1} \tag{3.32}$$

and

$$A_i(z) = (1 + \lambda_i z^{-1})^{-1} \tag{3.33}$$

respectively, where λ_i is an estimate of the negative reciprocal of a root (zero) of $Y(z)$ which lie outside the unit circle in the z-plane. At time $t = (i+n)T$, the operation of the adaptive linear filter is effectively carried out by the two filters and the process is divided into two stages. Firstly, the sequence $\{r_j'\}$, $j=1,2,\ldots,i+n$, is fed through the two-tap feedforward transversal filter, starting with the sample r_i' and finishing with the sample r_{i+n}', to produce a $(n+2)$-component output sequence of samples $\{b_i\}$ which are fed to the buffer store. Thus, at time $t = (i+n)T$, the store holds the sequence

$$B_i' = [b_i \quad b_{i+1} \ \cdot \ \cdot \ \cdot \ b_{i+n+1}] \tag{3.34}$$

which is then fed through the one-tap feedback filter in reverse order starting with the sample b_{i+n+1} and finishing with b_i. The sequence B_i', passing through the feedback filter in the reverse order as described, is taken to be moving backwards in time, and a delay of one sampling interval T in the feedback filter now becomes an advance of T, with z-transform z. Thus, the effective z-transform of the one-tap feedback filter becomes

$$A_i(z) = (1 + \lambda_i z)^{-1} \tag{3.35}$$

The filters operatedtogether in the manner just described has, in effect, a resultant z-transform

$$A_i(z)\ B_i(z) = (1 + \lambda_i z)^{-1}\ (1 + \lambda_i^* z^{-1}) \tag{3.36}$$

When the filters are correctly adjusted by an iterative process such that when i = k,

$$\lambda_k = \beta_1 \tag{3.37}$$

The resultant z-transform of the sampled impulse-response of the filter is

$$\begin{aligned} C_1(z) &= A_k(z)\ B_k(z) \\ &= (1 + \beta_1 z)^{-1}\ (1 + \beta_1^* z^{-1}) \end{aligned} \tag{3.38}$$

and the arrangement as shown in Figure 3.2 now processes the root (zero) of Y(z) that lies outside the unit circle, so that in the resultant z-transform of the sampled impulse-response of the channel and filter, the root $-\frac{1}{\beta_1}$ is now removed and replaced by a root at $z = -\beta_1^*$. In general, where there are m roots outside the unit circle, m sequential (successive) operations of the two filters are required, each one being responsible for one of the m roots.

Let the z-transform of the sequence R'_{i+n} (equation 3.9) be

$$R'_{i+n}(z) = r'_i z^{-i} + r'_{i+1} z^{-i-1} + \ldots + r'_{i+n} z^{-i-n} \qquad (3.39)$$

and let the z-transform of the stored sequence B'_i (equation 3.34) at the output of the two-tap feedforward filter be

$$B'_i(z) = b_i z^{-i} + b_{i+1} z^{-i-1} + \ldots + b_{i+n+1} z^{-i-n-1} \qquad (3.40)$$

Since

$$\begin{aligned} B'_i(z) &= R'_i(z)\, B_i(z) \\ &= (r'_i z^{-i} + r'_{i+1} z^{-i-1} + \ldots + r'_{i+n} z^{-i-n})(1 + \lambda_i^* z^{-1}) \\ &= r'_i z^{-i} + (r'_{i+1} + \lambda_i^* r'_i) z^{-i-1} + \ldots + \lambda_i^* r'_{i+n} z^{-i-n-1} \end{aligned} \qquad (3.41)$$

it follows that

$$b_i = r_i^{'} \tag{3.42}$$

$$b_{i+1} = r_{i+1}^{'} + \lambda_i^* r_i^{'} \tag{3.43}$$

$$\vdots$$

$$b_{i+n+1} = \lambda_i^* r_{i+n}^{'} \tag{3.44}$$

When the sequence $B_i^{'}$ is fed through the one-tap feedback filter in the manner as described, the z-transform of the feedback filter becomes $A_i(z)$ as given in equation 3.35. The z-transform of the signal at the output of the feedback filter is

$$X_i(z) = B_i^{'}(z)A_i(z)$$

$$= (b_i z^{-i} + b_{i+1} z^{-i-1} + \ldots + b_{i+n+1} z^{-i-n-1})(1 + \lambda_i z)^{-1} \tag{3.45}$$

but $B_i^{'}(z) = R_i^{'}(z)\ B_i(z)$ (equation 3.41), so that

$$X_i(z) = R_i^{'}(z)\ B_i(z)\ A_i(z)$$

$$= (r_i^{'} z^{-i} + r_{i+1}^{'} z^{-i-1} + \ldots + r_{i+n}^{'} z^{-i-n})(1 + \lambda_i^* z^{-1})(1 + \lambda_i z)^{-1} \tag{3.46}$$

Substituting the values of $r_i^{'}$, $r_{i+1}^{'}$, ..., $r_{i+n}^{'}$ (equations 3.18-3.21) into equation 3.46,

$$
\begin{aligned}
X_i(z) = \{(s_i y_o) z^{-i} \\
&+ (s_{i+1} y_o + s_i y_1) z^{-i-1} \\
&\cdot \\
&\cdot \\
&\cdot \\
&+ (s_{i+n} y_o + s_{i+n-1} y_1 + \ldots + s_{i+n-g} y_g) z^{-i-n} \\
&+ w_i z^{-i} + w_{i+1} z^{-i-1} + \ldots + w_{i+n} z^{-i-n}\}(1 + \lambda_i^* z^{-1})(1 + \lambda_i z)^{-1} \\
= X_i'(z) + V_i(z) \qquad (3.47)
\end{aligned}
$$

where

$$
\begin{aligned}
X_i'(z) = \{(s_i y_o z^{-i} \\
&+ (s_{i+1} y_o + s_i y_1) z^{-i-1} \\
&\cdot \\
&\cdot \\
&\cdot \\
&+ (s_{i+n} y_o + s_{i+n-1} y_1 + \ldots + s_{i+n-g} y_g) z^{-i-n}\}(1 + \lambda_i^* z^{-1})(1 + \lambda_i z)^{-1} \\
= \ldots + x_i' z^{-i} + x_{i+1}' z^{-i-1} + \ldots + x_{i+n+1}' z^{-i-n-1} \qquad (3.48)
\end{aligned}
$$

and

$$V_i(z) = (w_i z^{-i} + w_{i+1} z^{-i-1} + \ldots + w_{i+n} z^{-i-n})(1 + \lambda_i^* z^{-1})(1 + \lambda_i z)^{-1}$$

$$= \ldots + v_i z^{-i} + v_{i+1} z^{-i-1} + \ldots + v_{i+n+1} z^{-i-n-1} \qquad (3.49)$$

Thus, $X_i'(z)$ and $V_i(z)$ represent the z-transforms of the signal component (caused by the $\{s_i\}$) and the noise component (caused by $\{w_i\}$) of the output signal from the one-tap feedback filter, respectively. Since $(1 + \lambda_i^* z^{-1})(1 + \lambda_i z)^{-1}$ performs an orthogonal transformation (or an unitary transformation when complex values are used), the statistical properties of the noise components $\{v_i\}$ at the output of the linear filter are identical to those of $\{w_i\}$ at its input(7).

Consider the z-transform $X_i'(z)$ (equation 3.48)

$$X_i'(z) = \{(s_i y_0) z^{-i}$$

$$+ (s_{i+1} y_0 + s_i y_1) z^{-i-1}$$

$$+ (s_{i+2} y_0 + s_{i+1} y_1 + s_i y_2) z^{-i-2}$$

$$\vdots$$

$$+ (s_{i+g} y_0 + s_{i+g-1} y_1 + \ldots + s_i y_g) z^{-i-g}$$

$$+ (s_{i+g+1} y_0 + s_{i+g} y_1 + \ldots + s_{i+1} y_g) z^{-i-g-1}$$

$$+ (s_{i+g+2}y_o + s_{i+g+1}y_1 + \dots + s_{i+2}y_g)z^{-i-g-2}$$

$$\vdots$$

$$+ (s_{i+n-g}y_o + s_{i+n-g-1}y_1 + \dots + s_{i+n-2g}y_g)z^{-i-n-g}$$

$$+ (s_{i+n-g+1}y_o + s_{i+n-g}y_1 + \dots + s_{i+n-2g+1}y_g)z^{-i-n+g-1}$$

$$\vdots$$

$$+ (s_{i+n}y_o + s_{i+n-1}y_1 + \dots + s_{i+n-g}y_g)z^{-i-n}\}(1 + \lambda_i^* z^{-1})(1 + \lambda_i z)^{-1}$$

$$= \{s_i z^{-i}(y_0 + y_1 z^{-1} + \dots + y_g z^{-g})$$

$$+ s_{i+1} z^{-i-1}(y_0 + y_1 z^{-1} + \dots + y_g z^{-g})$$

$$+ s_{i+2} z^{-i-2}(y_0 + y_1 z^{-1} + \dots + y_g z^{-g})$$

$$\vdots$$

$$+ s_{i+n-g} z^{-i-n+g}(y_0 + y_1 z^{-1} + \dots + y_g z^{-g})$$

$$+ s_{i+n-g+1} z^{-i-n+g-1}(y_0 + y_1 z^{-1} + \dots + y_{g-1} z^{-g+1})$$

$$+ s_{i+n-g+2} z^{-i-n+g-2}(y_0 + y_1 z^{-1} + \dots + y_{g-2} z^{-g+2})$$

$$\vdots$$

$$+ s_{i+n} z^{-i-n}(y_0)\} \, (1 + \lambda_i^* z^{-1})(1 + \lambda_i z)^{-1} \qquad (3.50)$$

It can be seen from equation 3.50 that $(y_o + y_1z^{-1} + \ldots + y_gz^{-g})$ is a common factor to all the terms s_iz^{-i}, $s_{i+1}z^{-i-1}$, ..., $s_{i+n-g}z^{-i-n+g}$. However for terms $s_{i+n-g+1}z^{-i-n+g-1}$, $s_{i+n-g+2}z^{-i-n+g-2}$, ..., $s_{i+n}z^{-i-n}$, this is no longer true and the multiplication is carried out with a modified version of $(y_o + y_1z^{-1} + \ldots + y_hz^{-h})$ where $0 \leqslant h < g$ and h changes accordingly.

Now,

$$(y_o + y_1z^{-1} +\ldots + y_gz^{-g})(1 + \lambda_i^*z^{-1})(1 + \lambda_iz)^{-1}$$

$$= (1 + \beta_1z)(u_oz^{-1} + u_1z^{-2} + \ldots + u_{g-1}z^{-g})(1 + \lambda_i^* z^{-1})(1 + \lambda_iz)^{-1}$$

$$= (1 + \beta_1z)z^{-1}(u_o + u_1z^{-1} + \ldots + u_{g-1}z^{-g+1})(1 + \lambda_i^*z^{-1})$$

$$.(1 - \lambda_iz + \lambda_i^2z^2-\ldots.)$$

$$= z^{-1} (1 + (\beta_1-\lambda_i)z - \lambda_i(\beta_1-\lambda_i)z^2 + \lambda_i{}^2(\beta_1-\lambda_i) z^3 - \ldots)$$

$$.(q_o + q_1z^{-1} +\ldots + q_gz^{-g}) \qquad (3.51)$$

Consider the z-transform

$$y_o + y_1z^{-1} + \ldots + y_hz^{-h}$$

$$= (1 +\beta_1^{(h)}z)(u_o^{(h)}z^{-1} + u_1^{(h)}z^{-2} + \ldots + u_{h-1}^{(h)} z^{-h}) \qquad (3.52)$$

Let the negative-reciprocals of the roots (zeros) of the z-transform

$$y_o + y_1 z^{-1} + \ldots + y_h z^{-h}$$

where $0 < h < g$, that lie outside the unit circle in the z-plane be $\{\beta_j^{(h)}\}$ $1 \leqslant j \leqslant h$. Now, for $0 < h < g$

$$(y_o + y_1 z^{-1} + \ldots + y_h z^{-h})(1 + \lambda_i^* z^{-1})(1 + \lambda_i z)^{-1}$$

$$= (1 + \beta_1^{(h)} z) z^{-1} (u_o^{(h)} + u_1^{(h)} z^{-1} + \ldots + u_{h-1}^{(h)} z^{-h+1})$$

$$.(1 + \lambda_i^* z^{-1})(1 - \lambda_i z + \lambda_i^2 z^2 - \ldots)$$

$$= z^{-1}(1 + (\beta_1^{(h)} - \lambda_i)z - \lambda_i(\beta_1^{(h)} - \lambda_i)z^2 + \lambda_i^2(\beta_1^{(h)} - \lambda_i)z^3 - \ldots)$$

$$.(q_o^{(h)} + q_1^{(h)} z^{-1} + \ldots + q_h^{(h)} z^{-h}) \qquad (3.53)$$

and, of course,

$$q_o^{(h)} = u_o^{(h)} \qquad (3.54)$$

$$q_1^{(h)} = u_1^{(h)} + \lambda_i^* u_o^{(h)} \qquad (3.55)$$

.
.
.
etc

Substituting 3.51 and 3.53 into 3.50:

$$
\begin{aligned}
X_i'(z) = \{ & (s_i z^{-i} + s_{i+1} z^{-i-1} + \ldots + s_{i+n-g} z^{-i-n+g}) \\
& .(y_o + y_1 z^{-1} + \ldots + y_g z^{-g}) \\
& + s_{i+n-g+1} z^{-i-n+g-1} (y_o + y_1 z^{-1} + \ldots + y_{g-1} z^{-g+1}) \\
& + s_{i+n-g+2} z^{-i-n+g-2} (y_o + y_1 z^{-1} + \ldots + y_{g-2} z^{-g+2}) \\
& \vdots \\
& + s_{i+n} z^{-i-n} \; (y_o) \} \; (1 + \lambda_i^* z^{-1})(1 + \lambda_i z)^{-1} \\
= & (s_i z^{-i} + s_{i+1} z^{-i-1} + \ldots + s_{i+n-g} z^{-i-n+g})(q_o + q_1 z^{-1} + \ldots + q_g z^{-g}) \\
& .z^{-1}(1 + (\beta_1 - \lambda_i) z - \lambda_i (\beta_1 - \lambda_i) z^2 + \lambda_i{}^2 (\beta_1 - \lambda_i) z^3 - \ldots) \\
& + s_{i+n-g+1} z^{-i-n+g-1} (q_o^{(g-1)} + q_1^{(g-1)} z^{-1} + \ldots + q_{g-1}^{(g-1)} z^{-g+1}) \\
& .z^{-1}(1 + (\beta_1^{(g-1)} - \lambda_i) z - \lambda_i (\beta_1^{(g-1)} - \lambda_i) z^2 + \lambda_i{}^2 (\beta_1^{(g-1)} - \lambda_i) z^3 - \ldots) \\
& + s_{i+n-g+2} z^{-i-n+g-2} (q_o^{(g-2)} + q_1^{(g-2)} z^{-1} + \ldots + q_{g-2}^{(g-2)} z^{-g+2}) \\
& .z^{-1}(1 + (\beta_1^{(g-2)} - \lambda_i) z - \lambda_i (\beta_1^{(g-2)} - \lambda_i) z^2 + \lambda_i{}^2 (\beta_1^{(g-2)} - \lambda_i) z^3 - \ldots) \\
& + \ldots
\end{aligned}
$$

$$=z^{-1}\{\vdots$$

$$+z^{-i+1}[((\beta_1-\lambda_i)q_0-\lambda_i(\beta_1-\lambda_i)q_1+\lambda_i^{\,2}(\beta_1-\lambda_i)q_2-\ldots)s_i$$

$$+(\qquad -\lambda_i(\beta_1-\lambda_i)q_0+\lambda_i^{\,2}(\beta_1-\lambda_i)q_1-\ldots)s_{i+1}$$

$$+(\qquad \lambda_i^{\,2}(\beta_1-\lambda_i)q_0-\ldots)s_{i+2}$$

$$\vdots$$

$$+((-\lambda_i)^{n-g+1}(\beta_1^{(g-1)}-\lambda_i)q_0^{(g-1)}+(-\lambda_i)^{n-g+2}(\beta_1^{(g-1)}-\lambda_i)q_1^{(g-1)}+\ldots)s_{i+n-g+1}$$

$$+((-\lambda_i)^{n-g+2}(\beta_1^{(g-2)}-\lambda_i)q_0^{(g-2)}+(-\lambda_i)^{n-g+3}(\beta_1^{(g-2)}-\lambda_i)q_1^{(g-2)}+\ldots)s_{i+n-g+2}$$

$$+\ldots]$$

$$+z^{-i}[(q_0+(\beta_1-\lambda_i)q_1-\lambda_i(\beta_1-\lambda_i)q_2+\lambda_i^{\,2}(\beta_1-\lambda_i)q_3-\ldots)s_i$$

$$+(\quad (\beta_1-\lambda_i)q_0-\lambda_i(\beta_1-\lambda_i)q_1+\lambda_i^{\,2}(\beta_1-\lambda_i)q_2-\ldots)s_{i+1}$$

$$+(\qquad -\lambda_i(\beta_1-\lambda_i)q_0+\lambda_i^{\,2}(\beta_1-\lambda_i)q_1-\ldots)s_{i+2}$$

$$\vdots$$

$$+((-\lambda_i)^{n-g}(\beta_1^{(g-1)}-\lambda_i)q_0^{(g-1)}+(-\lambda_i)^{n-g+1}(\beta_1^{(g-1)}-\lambda_i)q_1^{(g-1)}+\ldots)s_{i+n-g+1}$$

$$+((-\lambda_i)^{n-g+1}(\beta_1^{(g-2)}-\lambda_i)q_0^{(g-2)}+(-\lambda_i)^{n-g+2}(\beta_1^{(g-2)}-\lambda_i)q_1^{(g-2)}+\ldots)s_{i+n-g+2}$$

$$+\ldots]$$

$$+z^{-i-1}[(q_1+(\beta_1-\lambda_i)q_2-\lambda_i(\beta_1-\lambda_i)q_3+\lambda_i^2(\beta_1-\lambda_i)q_4-\ldots)s_i$$

$$+(q_o+(\beta_1-\lambda_i)q_1-\lambda_i(\beta_1-\lambda_i)q_2+\lambda_i^2(\beta_1-\lambda_i)q_3-\ldots)s_{i+1}$$

$$+(\quad(\beta_1-\lambda_i)q_o-\lambda_i(\beta_1-\lambda_i)q_1+\lambda_i^2(\beta_1-\lambda_i)q_2-\ldots)s_{i+2}$$

$$\vdots$$

$$+((-\lambda_i)^{n-g-1}(\beta_1^{(g-1)}-\lambda_i)q_o^{(g-1)}+(-\lambda_i)^{n-g}(\beta_1^{(g-1)}-\lambda_i)q_1^{(g-1)}+\ldots)s_{i+n-g+1}$$

$$+((-\lambda_i)^{n-g}(\beta_1^{(g-2)}-\lambda_i)q_o^{(g-2)}+(-\lambda_i)^{n-g+1}(\beta_1^{(g-2)}-\lambda_i)q_1^{(g-2)}+\ldots)s_{i+n-g+2}$$

$$+\ \ldots]$$

$$+$$

$$\vdots\}\qquad(3.56)$$

From equation 3.47,

$$X_i(z) = X_i^{'}(z) + V_i(z)$$

$$= \ldots + x_i z^{-i} + x_{i+1}z^{-i-1} + \ldots + x_{i+n+1}z^{-i-n-1}\qquad(3.57)$$

where

$$x_h = x_h^{'} + v_h\qquad(3.58)$$

for $-\infty\leqslant h\leqslant i+n+1$.

The samples $\{x_h\}$, for $-\infty<h\leqslant i+n+1$, form the sequence at the output of the one-tap feedback filter at time $t=(i+n)T$ when the sequence $R_{i+n}^{'}$ (equation 3.39) is fed through the feedforward and feedback filters (Figure 3.2) in the manner described. Since the operation performed on the sequence $R_{i+n}^{'}$ is carried out at the receiver after the detection

of s_{i-1} at time $t=(i+n)T$, therefore all the signals, $\{x_h\}$, are available at this time instance.

Equating coefficients of z^{-i} in equations 3.48, 3.56 and 3.57,

$$
\begin{aligned}
x_i =\{(\beta_1-\lambda_i)[&(q_o-\lambda_i q_1+\lambda_i^2 q_2 -\ldots)s_i \\
&+(\quad -\lambda_i q_o+\lambda_i^2 q_1 -\ldots)s_{i+1} \\
&+(\qquad\quad \lambda_i^2 q_o -\ldots)s_{i+2} \\
&\vdots] \\
+((-\lambda_i)^{n-g+1}&(\beta_1^{(g-1)}-\lambda_i)q_o^{(g-1)}+(-\lambda_i)^{n-g+2}(\beta_1^{(g-1)}-\lambda_i)q_1^{(g-1)}+\ldots)s_{i+n-g+1} \\
+((-\lambda_i)^{n-g+2}&(\beta_1^{(g-2)}-\lambda_i)q_o^{(g-2)}+(-\lambda_i)^{n-g+3}(\beta_1^{(g-2)}-\lambda_i)q_1^{(g-2)}+\ldots)s_{i+n-g+2} \\
+ \ldots\} &\; + v_i
\end{aligned}
\qquad (3.59)
$$

Similarly,

$$
\begin{aligned}
x_{i+1}=\{&q_o s_i \\
&+(\beta_1-\lambda_i)[(q_1-\lambda_i q_2+\lambda_i^2 q_3-\ldots)s_i \\
&\qquad +(q_o-\lambda_i q_1+\lambda_i^2 q_2-\ldots)s_{i+1} \\
&\qquad +(\quad -\lambda_i q_o+\lambda_i^2 q_1-\ldots)s_{i+2} \\
&\qquad \vdots]
\end{aligned}
$$

$$+((-\lambda_i)^{n-g}(\beta_1^{(g-1)}-\lambda_i)q_o^{(g-1)}+(-\lambda_i)^{n-g+1}(\beta_1^{(g-1)}-\lambda_i)q_1^{(g-1)}+\ldots)s_{i+n-g+1}$$

$$+((-\lambda_i)^{n-g+1}(\beta_1^{(g-2)}-\lambda_i)q_o^{(g-2)}+(-\lambda_i)^{n-g+2}(\beta_1^{(g+2)}-\lambda_i)q_1^{(g+2)}+\ldots)s_{i+n-g+2}$$

$$+\ \ldots\} + v_{i+1} \qquad (3.60)$$

Clearly, when the adaptive linear filter is correctly adjusted such that $\lambda_i=\beta_1$, and $|\lambda_i|<1$ so that $(-\lambda_i)^{n-g-1}$ and higher powers of λ_i can be ignored,

$$x_{i-1} \simeq v_{i-1} \qquad (3.61)$$

$$x_i \simeq v_i \qquad (3.62)$$

$$x_{i+1} \simeq s_i q_o + v_{i+1} \qquad (3.63)$$

$$x_{i+2} \simeq s_i q_1 + s_{i+1} q_o + v_{i+2} \qquad (3.64)$$

$$\vdots$$

and so on.

From equation 3.61

$$\frac{x_{i+1}}{q_o} = s_i + v_{i+1} \qquad (3.65)$$

Since v_{i+1} is a Gaussian random variable with zero mean, this means that the detected value of s_i can be taken to be

$$\boxed{s_i' = \frac{x_{i+1}}{q_o}} \qquad \underline{(3.66)}$$

The basic operation of the receiver, and in particular, the feedforward and feedback filters (Figure 3.2) is the same throughout this Chapter. Various techniques for estimating one of the root (zero) of Y(z) that lies outside the unit circle when the received signal is corrupted by additive white Gaussian noise have been developed and tested for four test Channels, Channels A, B, C and D. The components of Y (equation 3.2) and the roots (zeros) of the corresponding Y(z) are as given in Table 3.1. The Channels are chosen so that the roots (zeros) of {Y(z)} that lie outside the unit circle range from approximately-1.43 to-10 which is fairly typical of the majority of the roots for a mild to poor telephone circuit. The different systems will now be described.

TABLE 3.1: Sampled Impulse-Responses Y of Channels A, B, C and D

	Sampled Impulse-Response Y			Zeros of Y(z)		q_0
Channel A	0.09901	1.0	0.09901	-0.1	-10	0.9901
Channel B	0.2798	1.0	0.2798	-0.306	-3.268	0.9144
Channel C	0.4	1.0	0.4	-0.5	-2	0.8
Channel D	0.4698	1.0	0.4698	-0.7	-1.4286	0.6711

3.1.1 System 3.1

The arrangement of the receiver is as shown in Figure 3.1 where, at time t=(i+n)T, the receiver operates on received samples $\{r_i\}$ in the presence of both intersymbol interference and additive white Gaussian noise as described in Section 3.1 to produce a sequence $\{x_i\}$ (equation 3.57) at the output of the one-tap feedback filter. Although any number of the sequence $\{x_i\}$ can be produced, in practice, only n+2 of them are ever generated.

Let the tap gain of the feedback filter be λ_i (an estimate of negative reciprocal of the root (zero) of Y(z) outside the unit circle) and assuming that $|\lambda_i|$ is small so that all the terms containing λ_i^3 and higher orders of λ_i can be ignored, then from equations 3.56 and 3.57,

$$\begin{aligned} x_i \simeq (\beta_1-\lambda_i)\ [&(q_o-\lambda_i q_1 + \lambda_i{}^2 q_2)s_i \\ +&(\ \ -\lambda_i q_o + \lambda_i{}^2 q_1)s_{i+1} \\ +&(\qquad\qquad \lambda_i{}^2 q_o)s_{i+2}] + v_i \end{aligned} \tag{3.67}$$

Now,

$$\frac{x_i}{\varepsilon_i} = (\beta_1-\lambda_i) + \frac{v_i}{\varepsilon_i} \tag{3.68}$$

where

$$\begin{aligned} \varepsilon_i = &(q_o-\lambda_i q_1 + \lambda_i{}^2 q_2)s_i + (-\lambda_i q_o + \lambda_i{}^2 q_1)s_{i+1} \\ &+ \lambda_i^2\ q_o s_{i+2} \end{aligned} \tag{3.69}$$

which means that x_i/ε_i is an estimate of $(\beta_1-\lambda_i)$. So, if λ_o, is the original estimate of β_1, the new estimate is

$$\begin{aligned} \lambda_1 &= \lambda_o + x_i/\varepsilon_i \\ &= \beta_1 + v_i/\varepsilon_i \end{aligned} \tag{3.70}$$

This suggests the following algorithm for estimating β_1,

$$\lambda_{i+1} = \lambda_i + c\,\frac{x_i}{\varepsilon_i} \tag{3.71}$$

where c is a real positive constant in the range 0 to 1. The evaluation of ε_i (equation 3.69) requires the prior knowledge of the parameters q_0, q_1 and q_2 (equation 2.29) which are dependent on the roots (zeros) of Y(z) and generally not available. Furthermore, at time t=(i+n)T, the data-symbols s_{i+1} and s_{i+2} have not yet been detected and, therefore, ε_i cannot be evaluated in practice. It is, however, interesting to see if the algorithm derived from the much simplified version of the equation for x_i can be used to give an accurate estimate of the value β_1, assuming that the unknown quantities are now available at the receiver. The results of this test should provide an upper bound (or reference) on the performance of further developments of the algorithm based on similar techniques and indicate the complexities of algorithms required.

It is assumed that the initial estimate of β_1, at the beginning of the iterative root-finding process, is λ_0=0. At time t=(i+n)T, the receiver operates on the signals $\{r_i\}$ to produce R'_{i+n} (equation 3.9) which is then fed through the feedforward and feedback filters as described in Section 3.1 to produce the signal x_i. The latter is used in the algorithm (equation 3.70) to give a new estimate of β_1. Having determined the new estimate, λ_i, the receiver appropriately adjusts the tap gains of the feedforward and feedback filters for the next iterative process. The detected symbol s'_i is then used to form the terms corresponding to the intersymbol interference given by the components of the vector Z_{i+n+1} (equation 3.7) which are subsequently removed from the corresponding components of the new sequence R_{i+n+1} at the next sampling instant t=(i+n+1)T. Thus, at time t=(i+n+1)T, the receiver has, at the input of the feedforward filter, the (n+1)-component sequence

$$R'_{i+n+1} = [r'_{i+1} \quad r'_{i+2} \cdot \cdot \cdot \quad r'_{i+n+1}] \qquad (3.72)$$

The whole of the process is repeated as described using R'_{i+n+1} in place of R'_{i+n} and with the filters taps set to the newly updated value λ_{i+1}. Clearly, the estimation of β_1 is carried out once every sampling interval T, and then the feedforward and feedback filters are updated accordingly.

The performance of the root-finding algorithm is measured in terms of θ_i which is the difference between β_1 and its estimate λ_i at each sampling interval T, and in terms of E_i which is the error in the detected symbol s'_i where s'_i is calculated from equation 3.66. More precisely,

$$\theta_i = 10 \log_{10} (\overline{|\beta_1 - \lambda_i|^2}) \qquad (3.73)$$

and

$$E_i = 10 \log_{10} (\overline{|s_i - s'_i|^2}) \qquad (3.74)$$

where $\overline{|\beta_1 - \lambda_i|^2}$ and $\overline{|s_i - s'_i|^2}$ are the mean-square errors measured over a number of independent computer-simulation runs.

For the purposes of this investigation, four polynomials with real-valued coefficients and of degree two were used. They can be thought of as baseband channels for a synchronous serial binary data-transmission systèm whose sampled impulse-responses are given by the vector Y with z-transform Y(z). These are designated as Channels A, B, C and D whose sampled impulse-responses Y and the corresponding roots (zeros) of Y(z) are shown in Table 3.1. The Channels are chosen such that their Y(z) have one root outside the unit circle in the z-plane whose magnitudes range from approximately 1.43 to 10, thus providing a reasonably wide selection of their distance from the unit circle. Now, for these channels,

$$Y(z) = y_o + y_1 z^{-1} + y_2 z^{-2}$$

$$= (1 + \beta_1 z)(u_o z^{-1} + u_1 z^{-2}) \qquad (3.75)$$

From equation 3.29,

$$q_o = u_o = y_o/\beta_1 \qquad (3.76)$$

$$q_1 = u_1 + \lambda_i^* u_o = y_2 + \lambda_i^* \; (y_o/\beta_1) = y_2 + \lambda_i (y_o/\beta_1) \qquad (3.77)$$

$$q_2 = \lambda_i^* \; u_1 = \lambda_i^* \; y_2 = \lambda_i y_2 \qquad (3.78)$$

where it can be seen that the parameters q_1 and q_2 depend on the estimate λ_i ($\lambda_i^* = \lambda_i$ in this case). The sampled impulse-response Y, the roots of Y(z), and the corresponding parameter q_o when the root outside the unit circle is being processed, for the four channels are as given in Table 3.1.

Results of computer-simulation tests for the four channels are given in Figures 3.3-3.10 which show the performance of the algorithm (equation 3.71) for a range of values (0.1, 0.25, 0.5, 0.75, 1.0) for both the noiseless condition and for a signal to noise ratio of 30 dB. In all the cases, the graphs of θ_i and E_i versus i (number of sampling intervals) were obtained by averaging the errors over 50 runs.

It can be seen that, with the exception of Channel D, where $\beta_1 = 0.7$, the algorithm gives a very good estimate of β_1 providing, of course, that [one has an accurate knowledge of the parameters q_o, q_1 and q_2.] Very fast convergence rates are possible for all three channels A, B and C, and in the case where the signal to noise ratio is 30 dB, convergence is achieved within less than 10 iterations (or sampling intervals) for c=1. Furthermore, in the noiseless situation, extremely accurate results (better than -100 dB in θ_i and E_i) are obtained both for the estimate of β_1 and the detected data-symbol s_i'.

The reason for the breakdown of the algorithm in the case of Channel D which happened even in the absence of additive white Gaussian noise is due to its relative large value of β_1 (β_1=0.7), which means λ_i^3 is too large to be ignored, and furthermore, relative magnitudes of q_o, q_1 and q_2 are such that the approximation made in deriving the algorithm (equation 3.71) no longer holds.

3.1.2 System 3.2

The algorithm used in System 3.1 (equation 3.71) requires prior and accurate knowledge of the quantities q_o, q_1 and q_2 (equations 3.76-3.78) which, in practice, have to be estimated. Unfortunately, the estimation of these quantities is quite difficult and involved, and it is not certain whether they can be estimated accurately from the sequence $\{x_i\}$ (equation 3.57). From equation 3.59

$$\begin{aligned} x_i = (\beta_1-\lambda_i)\ [&(q_o - \lambda_i q_1 + \lambda_i^2 q_2 - \lambda_i^3 q_3 + \ldots)s_i \\ &+(\quad - \lambda_i q_o + \lambda_i^2 q_1 - \lambda_i^3 q_2 + \ldots)s_{i+1} \\ &+(\qquad \lambda_i^2 q_o - \lambda_i^3 q_1 + \ldots)s_{i+2} \\ &+ \ldots] \ + v_i \end{aligned} \qquad (3.79)$$

Assume that $|\lambda_i|<<1$, so that all terms containing λ_i or higher orders of λ_i can be ignored. Thus,

$$x_i \simeq (\beta_1-\lambda_i)\, q_0 s_i + v_i \tag{3.80}$$

and

$$(\beta_1-\lambda_i) \simeq \frac{x_i}{s_i q_0} - \frac{v_i}{s_i q_0}$$

so that

$$\beta_1 \simeq \lambda_i + \frac{x_i}{s_i q_0} - \frac{v_i}{s_i q_0} \tag{3.81}$$

This means that, if c is a positive constant in the range 0 to 1, then,

$$\lambda_{i+1} = \lambda_i + c\,\frac{x_i}{s_i q_0} \tag{3.82}$$

is a better estimate of β_1 than is λ_i. Equation 3.82 represents a simplified algorithm for estimating β_1 from that of System 3.1, where, in this case, the algorithm only requires the knowledge of q_0 instead of q_0, q_1 and q_2 as required previously.

The receiver operates exactly as in System 3.1 where the stored vector R'_{i+n} (equation 3.39) is fed through the two-tap feedforward filter and one-tap feedback filter arrangement (Figure 3.2) whose initial tap gain, λ_0, is set to zero. The manner in which the vectors R'_{i+n} and B'_i are manipulated is as described before and the new estimate of β_1 is obtained by the above algorithm (equation 3.82).

Channels A, B, C and D (Table 3.1) were used in this experiment, and the results of computer-simulation tests are given in Figures 3.11-3.18 which show the performance of the algorithm for a range of c (0.1, 0.25, 0.5, 0.75, 1.0), for both the noiseless condition and for a signal to noise ratio of 30 dB. The variation of θ_i (equation 3.73) and E_i (equation 3.74) with i are measured for 100 iterations (sampling intervals) and obtained from averaging the errors $|\beta_1-\lambda_i|^2$ and $|s_i- s_i'|^2$ over 50 runs. The results show that although the rate of convergence is slightly slower, particularly for channels C and D (whose values of β_1 are 0.5 and 0.7 respectively). The degradation in the speed of locating the root $-\frac{1}{\beta_1}$ is more severe when there is no additive noise in the received samples $\{r_i\}$ and it also increases with the smaller step size. With a suitably large step size such as 0.75, this can be confined to within 10 iterations and for signal to noise ratio of 30 dB, the difference in speed between this system and System 3.1 is almost negligible. However, in the case of Channel D, the algorithm actually gives quite acceptable results and that the performance is much better than the previous system (equation 3.70). This is very encouraging because it means that much simpler algorithms can be used for the estimation of β_1, especially where the absolute value of β_1 is small.

3.1.3 System 3.3

It has been shown that the root-finding algorithm used in System 3.2

$$\lambda_{i+1} = \lambda_i + c \frac{x_i}{s_i q_o}$$

where c is a positive constant in the range of 0 to 1 and x_i and q_o are given by equations 3.59 and 3.76 respectively, works very well with the test channels and gives comparable results to those of System 3.1. It is a much simplified form of System 3.1 and requires only the knowledge of the parameter q_o.

Let us now assume that only the sign of q_o is known, so that the equation above becomes

$$\lambda_{i+1} = \lambda_i + c \frac{x_i}{s_i \frac{q_o}{|q_o|}} \qquad (3.83)$$

where $q_o/|q_o|$ is ±1 depending on whether q_o is positive or negative, and c is, of course, a positive real constant in the range 0 to 1.

It would be extremely informative to see how the algorithm performs with such limited knowledge of the channel, namely, the sign of the parameter q_o (equation 3.76). Again, Channels A, B, C and D were used in this test and the results are given in Figures 3.19-3.26 which show the variation of θ_i and E_i (equations 3.73 and 3.74) with i for 100 iterations (sampling intervals) averaged over 50 runs. For the purpose of comparison, the same set of step sizes, c (0.1, 0.25, 0.5, 0.75, 1.0), were used and the algorithm was tested for the noiseless condition and for the signal to noise ratio of 30 dB. Results show the performance of this system is very close to that of System 3.2

where prior knowledge of the exact value of q_0 is required. Significant differences occur only in Channels C and D where the valuesof q_0 are 0.8 and 0.671 (Table 3.1) instead of 1 which is the value assumed for q_0 in here and in cases where c is 0.25 or less. The speed of convergence of the algorithm is only slightly slower than that of System 3.1 where prior knowledge of the quantities q_0, q_1 and q_2 (equations 3.75-3.77) is assumed. The degradation in speed increases for channels with larger values of β_1, and in the worst case where Channel D was used (β_1=0.7) with signal to noise ratio of 30 dB and c=0.1, the degradation is less than 10 iterations at convergence. The accuracy of the final estimate of β_1 is very close to that obtained before. Furthermore, a satisfactory operation is obtained with Channel D.

3.1.4 System 3.4

So far, it has been shown that the algorithm developed in System 3.3,

$$\lambda_{i+1} = \lambda_i + c \; \frac{x_i}{s_i \dfrac{q_0}{|q_0|}}$$

(equation 3.83), is capable of providing reasonably accurate results for the estimation of β_1 in the presence of additive white Gaussian noise. The algorithm performs satisfactorily for the four channels A, B, C and D (Table 3.1) provided, of course, that the receiver has prior knowledge of the sign of the parameter q_0 (equation 3.76). In this system, an algorithm is derived which involves the use of an estimate of the quantity q_0 derived from the sample x_{i+1} (equation 3.60) at the output of the one-tap feedback filter.

As before, the receiver operates on the stored vector R'_{i+n} (equation 3.39) which represents the sequence of n+1 received samples with the intersymbol interference terms removed. In the cancellation of the terms representing intersymbol interference, it is assumed that the sampled impulse-response Y (equation 3.1) of the channel is provided at the receiver and that all previous data-symbols are detected correctly. The initial tap gains of the feedforward and feedback filters (Figure 3.2) are set to λ_0=0 at the beginning of the iterative root-finding process and the stored sequence R'_{i+n} is fed through the two-tap feedforward filter in the order in which they are received, i.e. r'_i, r'_{i+1},. . . , r'_{i+n} (equations 3.18-3.21). The output sequence B'_i (equation 3.34) from the feedforward filter is, however, fed through the feedback filter in reverse order, starting with the component b_{i+n+1} and finishing with the component b_i, to give an output sequence X_i (equation 3.57). Recall from equations 3.59 and 3.60

$$\begin{aligned} x_i = {} & [(\beta_1-\lambda_i)q_0 - \lambda_i(\beta_1-\lambda_i)q_1 + \lambda_i^2 (\beta_1-\lambda_i)q_2 - \ldots]\, s_i \\ & +[\quad - \lambda_i(\beta_1-\lambda_i)q_0 + \lambda_i^2 (\beta_1-\lambda_i)q_1 - \ldots]\, s_{i+1} \\ & +[\quad + \lambda_i^2 (\beta_1-\lambda_i)q_0 - \ldots]\, s_{i+2} \\ & + \ldots + v_i \end{aligned} \tag{3.84}$$

$$\begin{aligned} x_{i+1} = {} & [q_0 - (\beta_1-\lambda_i)q_1 - \lambda_i(\beta_1-\lambda_i)q_2 + \lambda_i^2 (\beta_1-\lambda_i)q_3 - \ldots]\, s_i \\ & +[\quad (\beta_1-\lambda_i)q_0 - \lambda_i(\beta_1-\lambda_i)q_1 + \lambda_i^2 (\beta_1-\lambda_i)q_2 - \ldots]\, s_{i+1} \\ & +[\quad -\lambda_i(\beta_1-\lambda_i)q_0 + \lambda_i^2 (\beta_1-\lambda_i)q_1 - \ldots]\, s_{i+2} \\ & + \ldots + v_{i+1} \end{aligned} \tag{3.85}$$

Let $|\lambda_i|<<1$, so that all terms containing λ_i or higher orders of λ_i can be ignored. Thus, from equation 3.84,

$$x_i \simeq (\beta_1-\lambda_i)q_o s_i + v_i \tag{3.86}$$

and

$$(\beta_1-\lambda_i) \simeq \frac{x_i}{s_i q_o} - \frac{v_i}{s_i q_o}$$

so that

$$\beta_1 \simeq \lambda_i + \frac{x_i}{s_i q_o} - \frac{v_i}{s_i q_o} \tag{3.87}$$

Thus if c is a positive constant in the range 0 to 1, then

$$\lambda_{i+1} = \lambda_i + c\,\frac{x_i}{s_i q_o} \tag{3.88}$$

is a better estimate of β_1 than is λ_i. This is the same algorithm as that of System 3.2 which, can be seen, requires a prior knowledge of the quantity q_o. Now, from equation 3.85, assuming that $|(\beta_1-\lambda_i)|$ is small so that terms containing $(\beta_1-\lambda_i)$ can also be ignored,

$$x_{i+1} \simeq q_o s_i + v_{i+1}$$

$$q_o \simeq \frac{x_{i+1}}{s_i} - \frac{v_{i+1}}{s_i} \tag{3.89}$$

Since v_{i+1} is a Gaussian random variable with zero mean, this means that q_o can be estimated in an iterative process as follows

$$q_o'^{(i+1)} = q_o'^{(i)} + c\,(\frac{x_{i+1}}{s_i} - q_o'^{(i)}) \qquad (3.90)$$

where $q_o'^{(i)}$ is the estimate of the quantity q_o at the i^{th} iteration, and c is a positive real constant.

Substituting $q_o'^{(i)}$ from equation 3.90 for q_o in equation 3.88, the algorithm for estimating β_1 becomes

$$\lambda_{i+1} = \lambda_i + c\,\frac{x_i}{s_i q_o'^{(i)}} \qquad (3.91)$$

and

$$q_o'^{(i)} = q_o'^{(i-1)} + c\,(\frac{x_{i+1}}{s_i} - q_o'^{(i-1)}) \qquad (3.92)$$

It is assumed that the first estimate of q_o is zero, i.e. $q_o'^{(0)}=0$, and for simplicity, the step sizes in equation 3.91 and equation 3.92 are both set to equal c, a positive real constant in the range of 0 to 1.

Again, the four channels A, B, C and D were used for this investigation and the results of computer-simulation tests are given in Figures 3.27 -3.42 showing the variation of the mean-square error in the estimation of β_1, θ_i, and of the mean-square error in the detected data-symbol s_i', E_i, for each iteration (sampling interval). Three different levels of additive noise corresponding to signal to noise ratios of infinity, 30 dB and 20 dB were used and in addition to the step sizes used previously, $c = \frac{1}{i}$, where i is the number of iterations (i=1,2,...), was also tested for all the cases where the received signal is corrupted by noise. The results were obtained using 50 different sequences of $\{s_i\}$ and $\{w_i\}$ (equation 3.4) and the graphs, therefore, show the mean-square error of $(\beta_1-\lambda_i)$ and (s_i-s_i') in dB taken over 50 runs. Comparing this set of results with those of System 3.2 (Figures 3.11-3.18), where it uses the algorithm

$$\lambda_{i+1} = \lambda_i + c\,\frac{x_i}{s_i q_o}$$

for estimating β_1 and where it was assumed that the receiver has prior knowledge of the quantity q_o, there is an excellent agreement, both in the speed and the accuracy of the algorithms between the two systems. Extremely accurate estimates of β_1 are achieved in the noiseless condition, giving an error θ_i (equation 3.73) of better than -100 dB in Channels A, B and C and about -60 dB in Channel D. Closer examination of the results show that there is a slight degradation in the speed or rate of convergence when compared with System 3.2. For Channel A, where $\beta_1=0.1$, there is virtually no difference in rate of convergence at a signal to noise ratio of 30 dB, but appreciable difference exists for the other Channels,B, C and D, particularly in Channel D where $\beta_1=0.7$. However, this difference in speed is very small, about 10 iterations (sampling intervals) for Channel D at 30 dB signal to noise ratio, and in the other less severe cases, this difference is less than 10 iterations. The close similarity between the two sets of results suggests that the value q_o can be accurately estimated by equation 3.92.

One encouraging point is that there is no evidence of breakdown of the algorithm(even at a signal to noise ratio of 20 dB)and that the algorithm continues to work, giving satisfactory results. It was also found that, in the presence of additive white Gaussian noise, the algorithm (equations 3.91-3.92) using $c = \frac{1}{i}$, where i=1,2,..., gives more accurate estimates of β_1 for channels A and B than for any other values of c, and the rate of convergence is about that of c=0.1 to 0.25. However for Channels C and D, this method produces less accurate results at a signal to noise ratio of 30 dB than when c is a positive real constant in the range of 0.1 to 1 but at a higher signal to noise ratio of 20 dB, it becomes slightly better for Channel C and about the same as c=0.1 for Channel D. This means that $c = \frac{1}{i}$, where i=1,2,..., is suitable, perhaps, for cases where β_1 is small or when the signal to noise ratio is high.

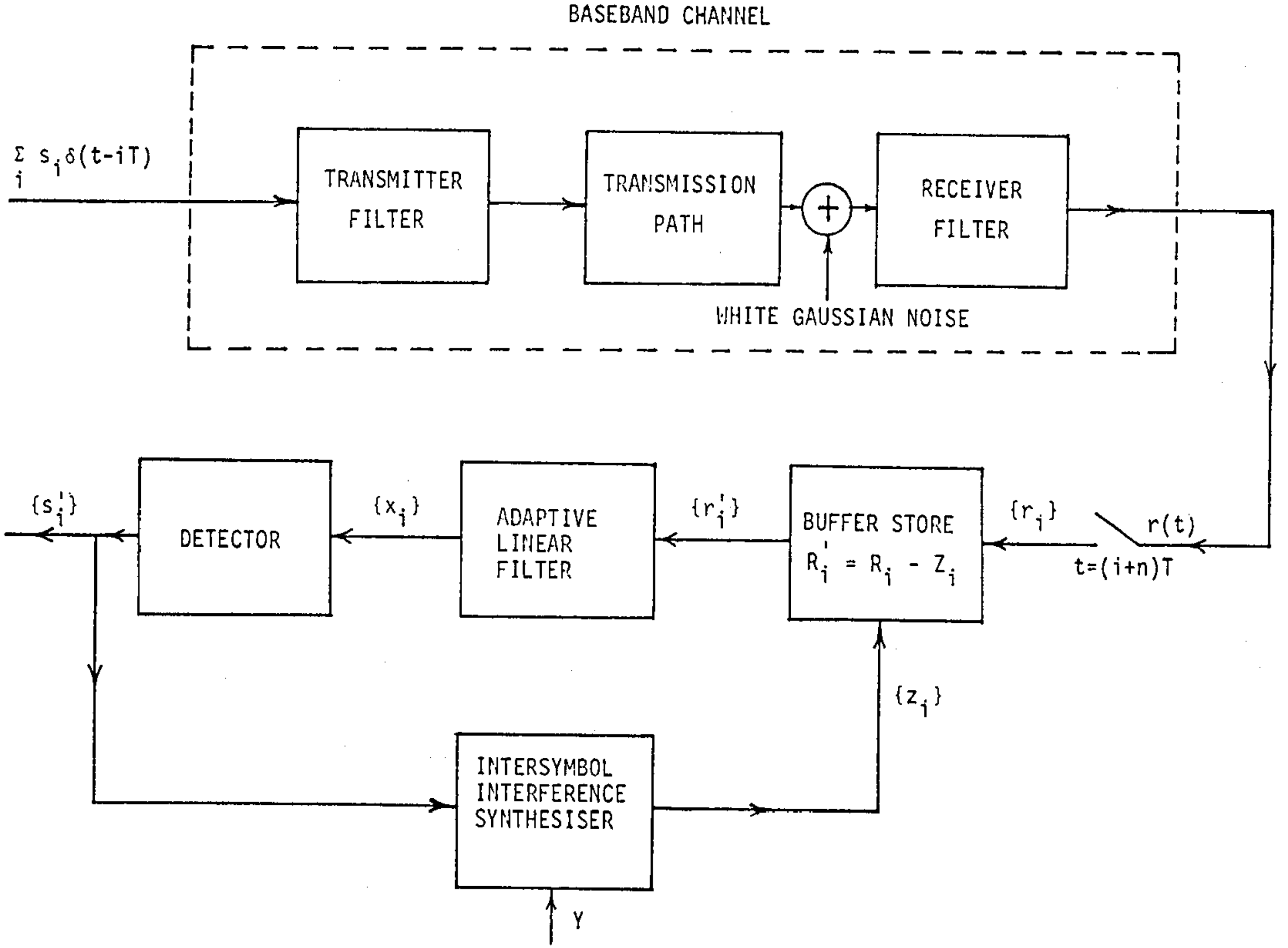


FIGURE 3.1: Model of the Synchronous Serial Binary Data-Transmission System

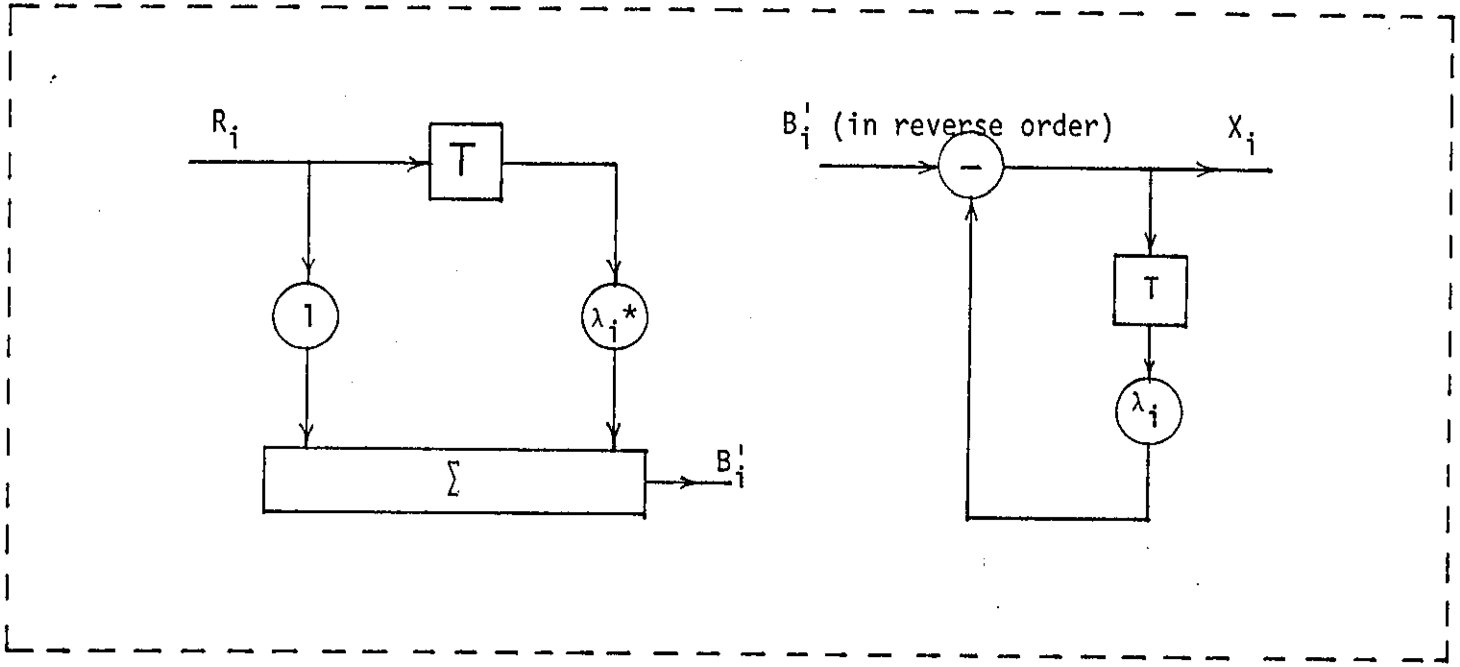


FIGURE 3.2: Adaptive linear filter implemented by the combination of the one-tap feedback and one-tap feedforward filters

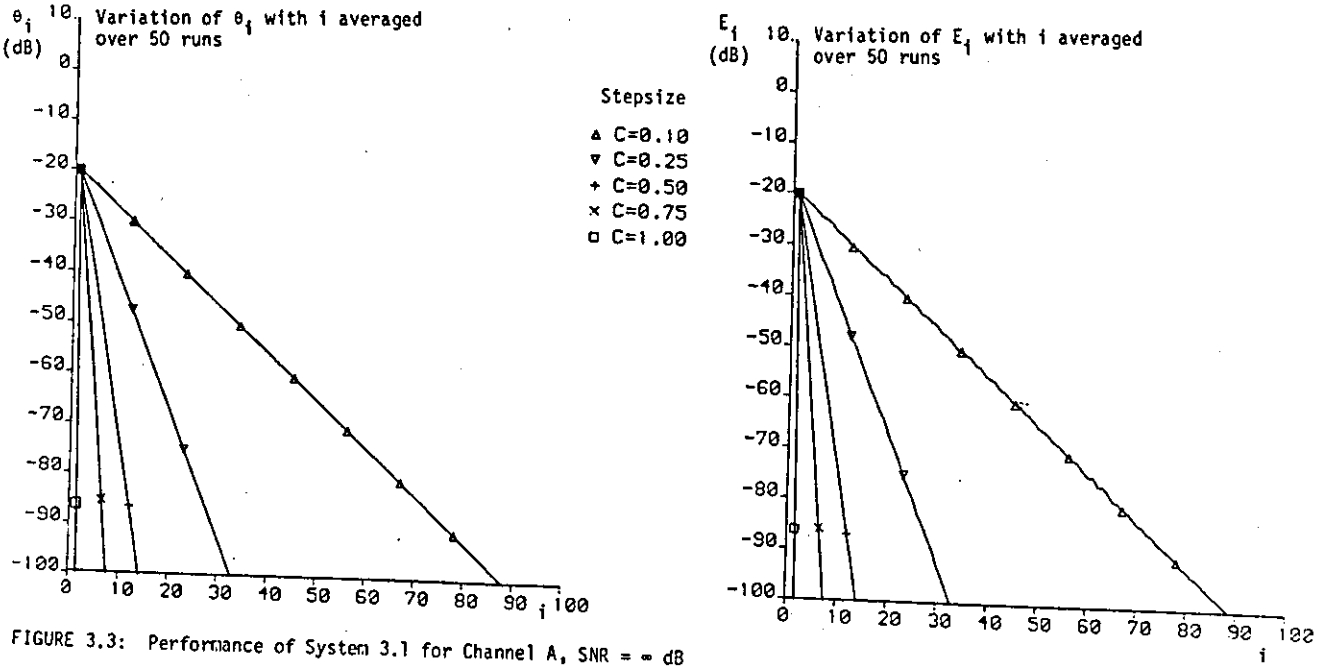


FIGURE 3.3: Performance of System 3.1 for Channel A, SNR = ∞ dB

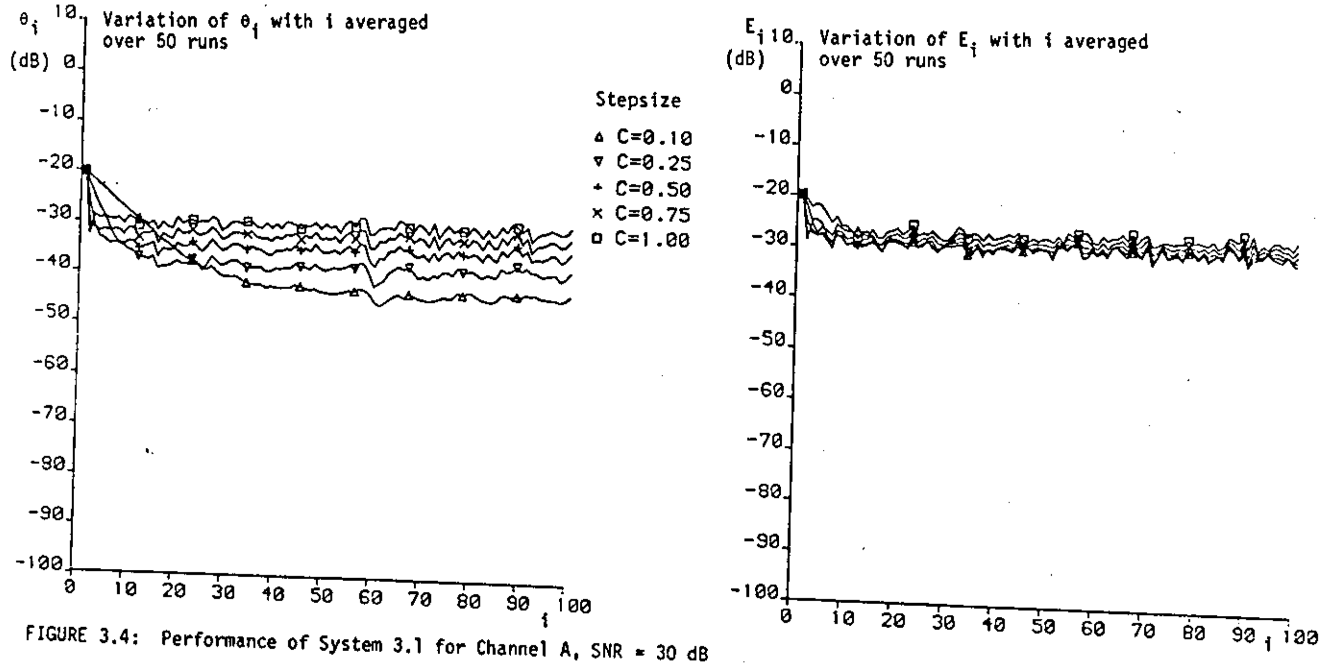


FIGURE 3.4: Performance of System 3.1 for Channel A, SNR = 30 dB

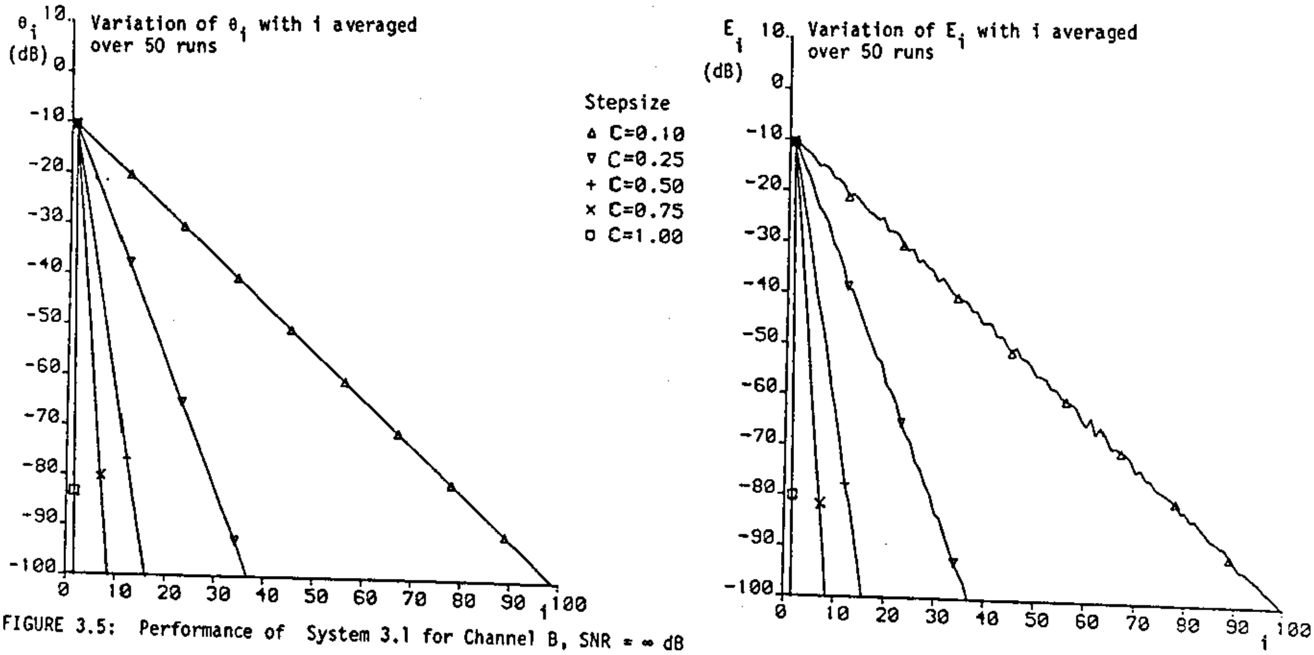


FIGURE 3.5: Performance of System 3.1 for Channel B, SNR = ∞ dB

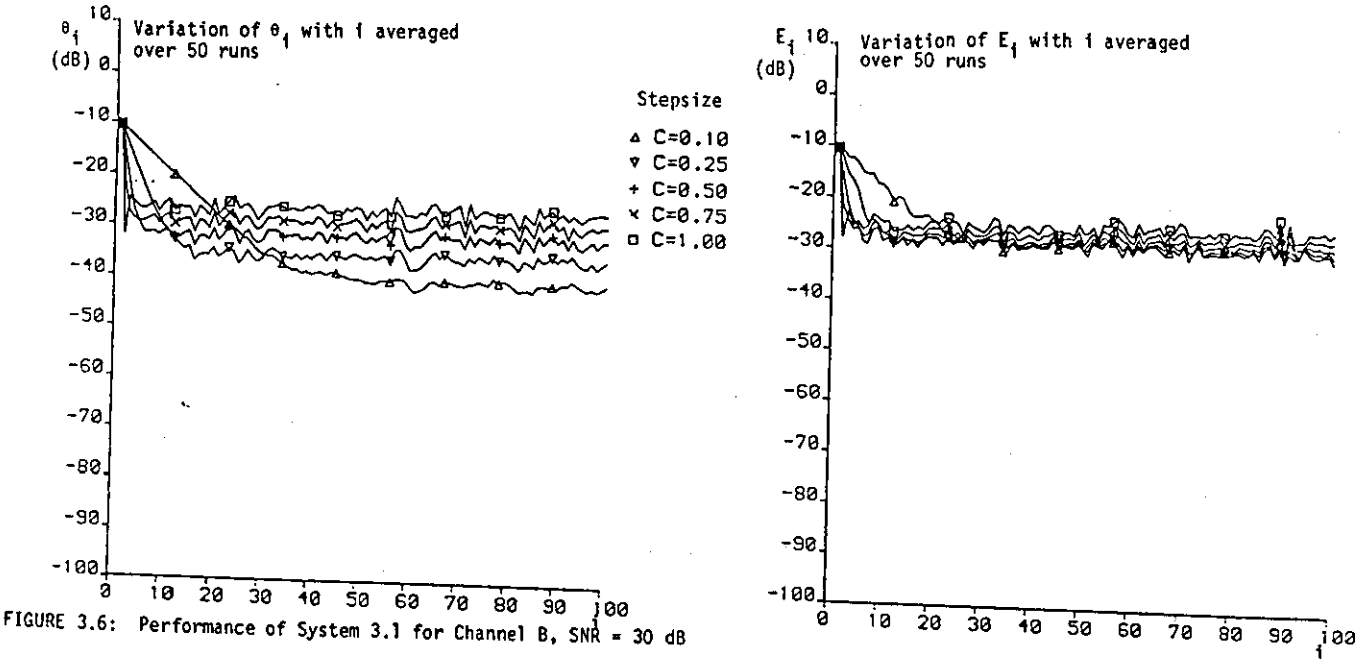


FIGURE 3.6: Performance of System 3.1 for Channel B, SNR = 30 dB

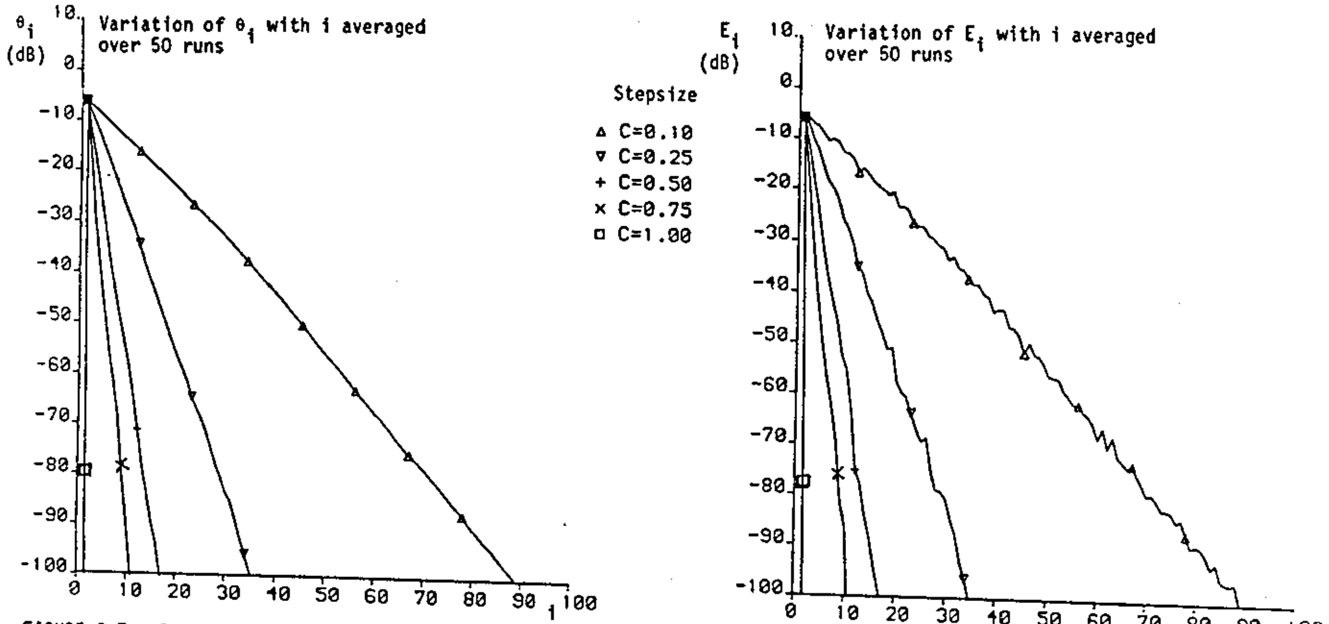


FIGURE 3.7: Performance of System 3.1 for Channel C, SNR = ∞ dB

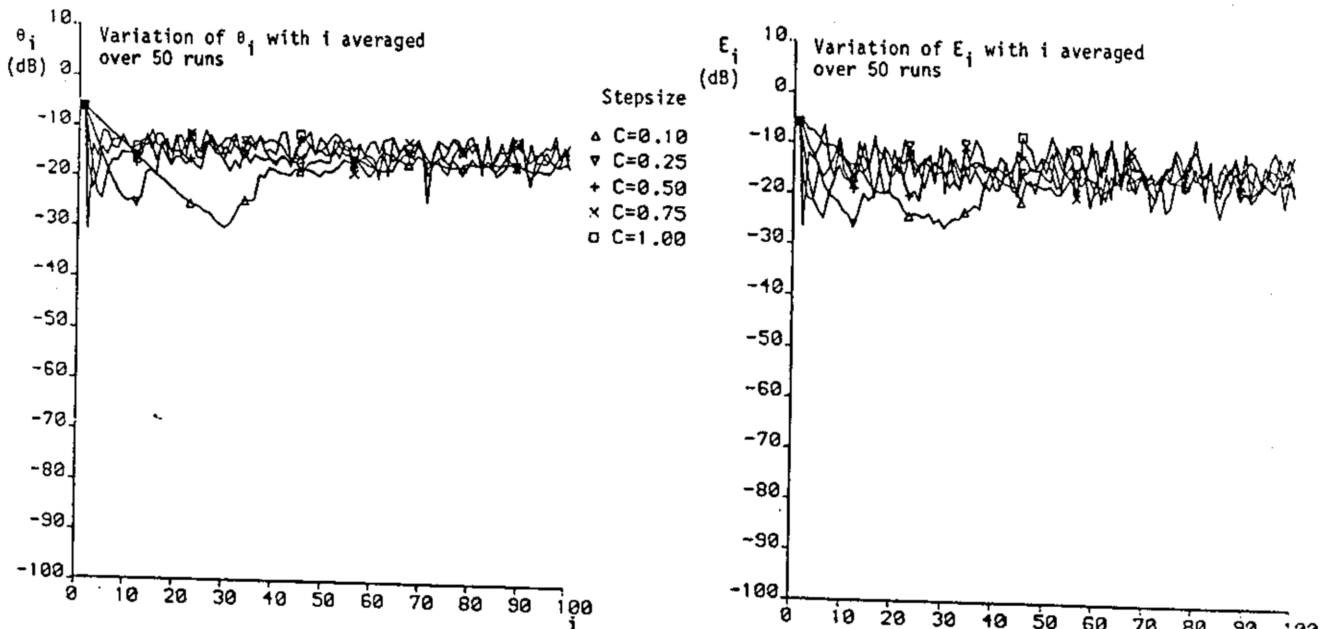


FIGURE 3.8: Performance of System 3.1 for Channel C, SNR = 30 dB

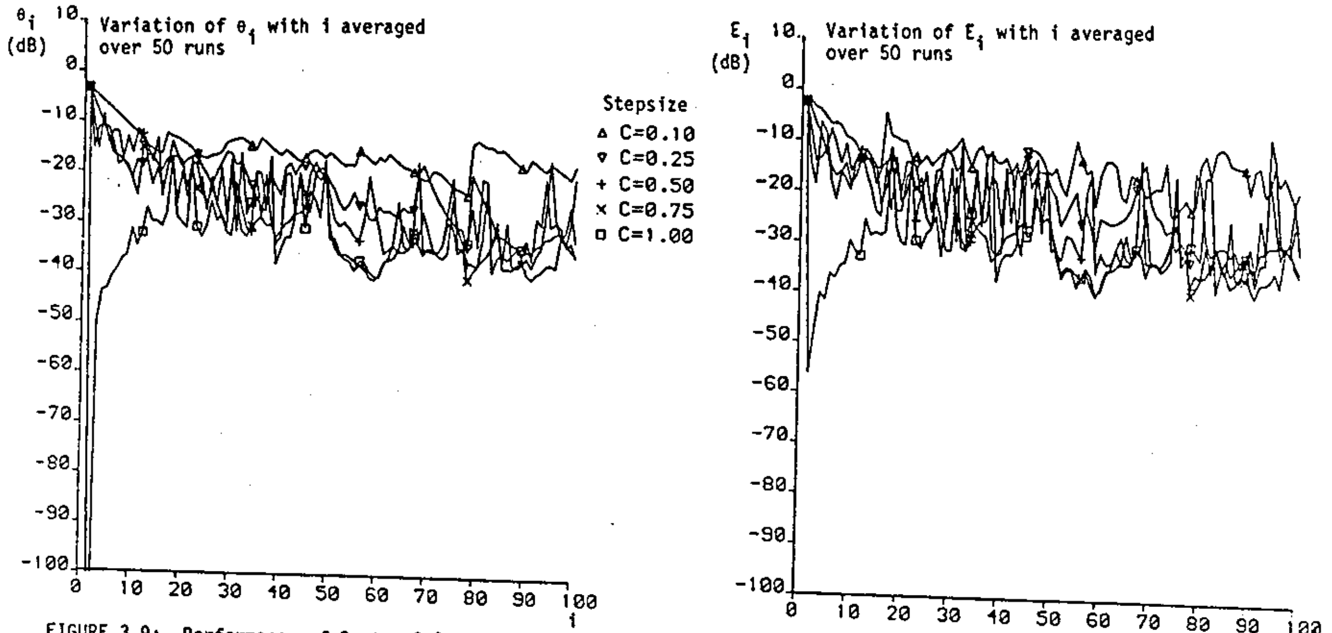


FIGURE 3.9: Performance of System 3.1 for Channel D, SNR = ∞ dB

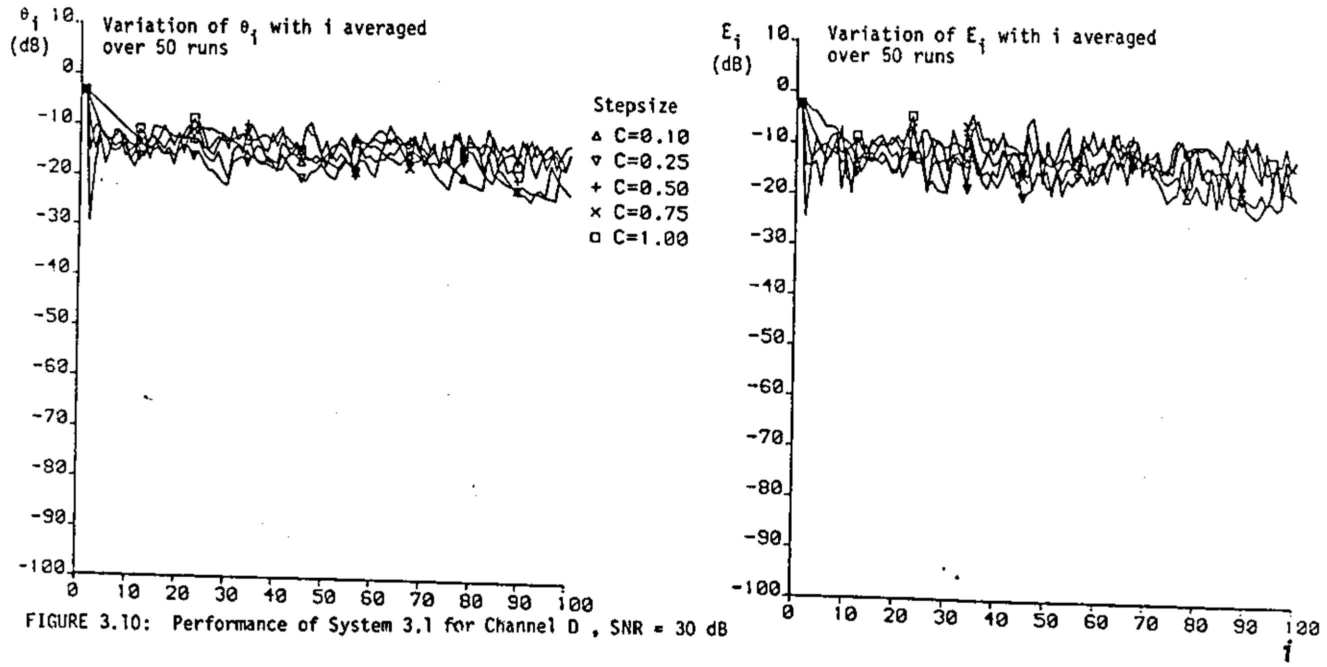


FIGURE 3.10: Performance of System 3.1 for Channel D , SNR = 30 dB

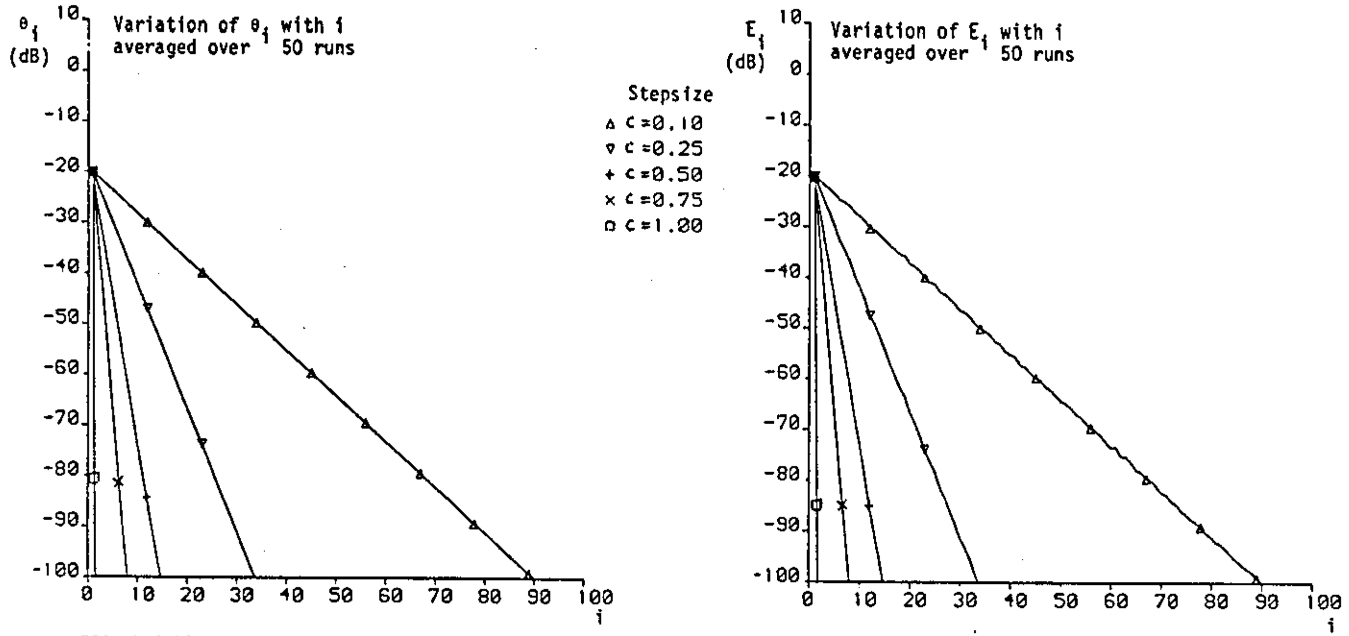


FIGURE 3.11: Performance of System 3.2 for Channel A, SNR = ∞ dB

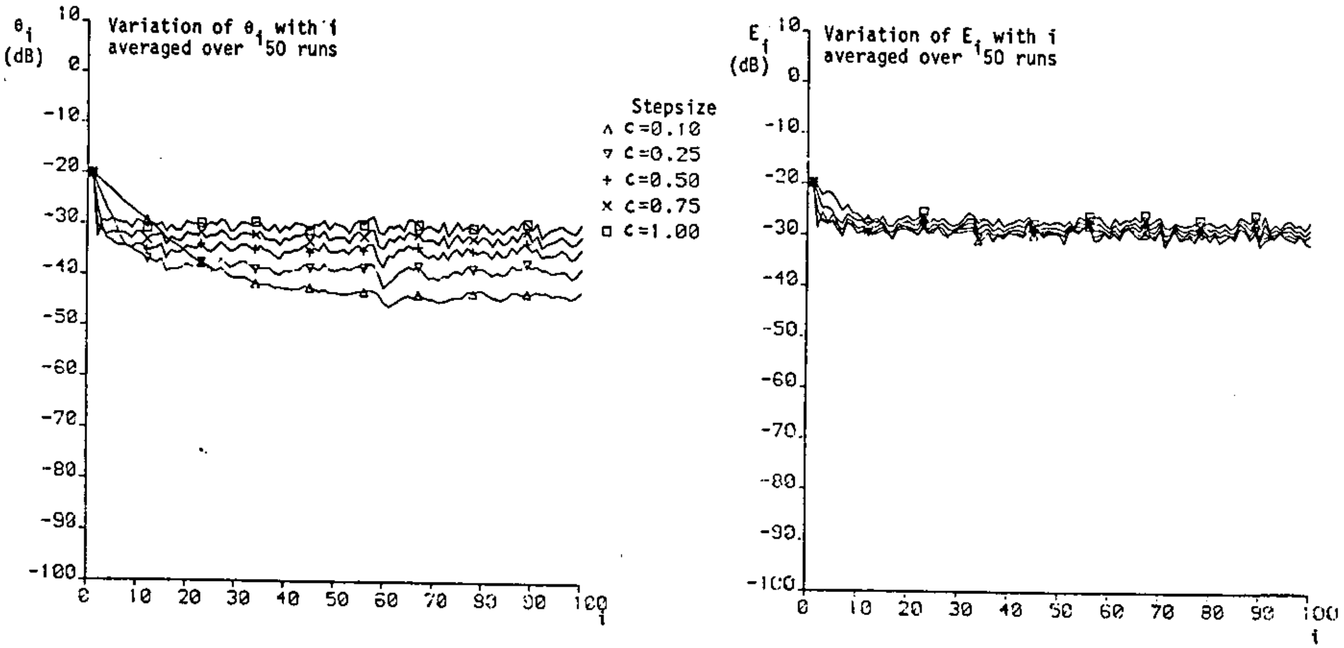


FIGURE 3.12: Performance of System 3.2 for Channel A, SNR = 30 dB

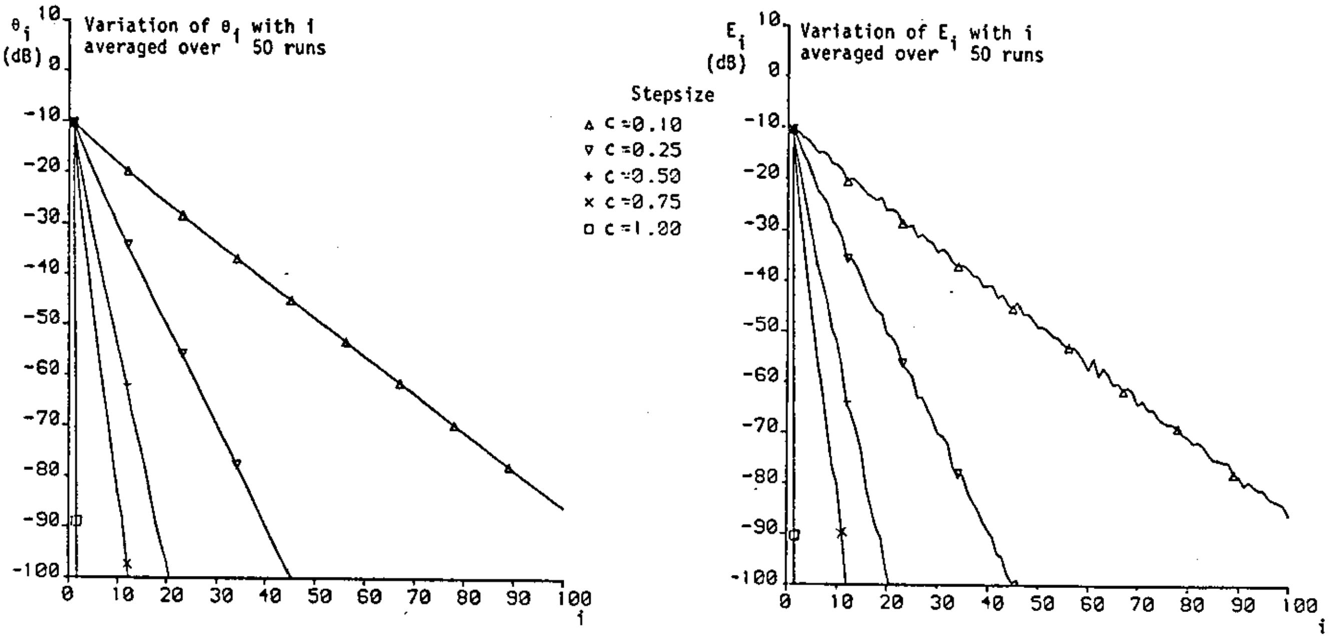


FIGURE 3.13: Performance of System 3.2 for Channel B, SNR= ∞dB

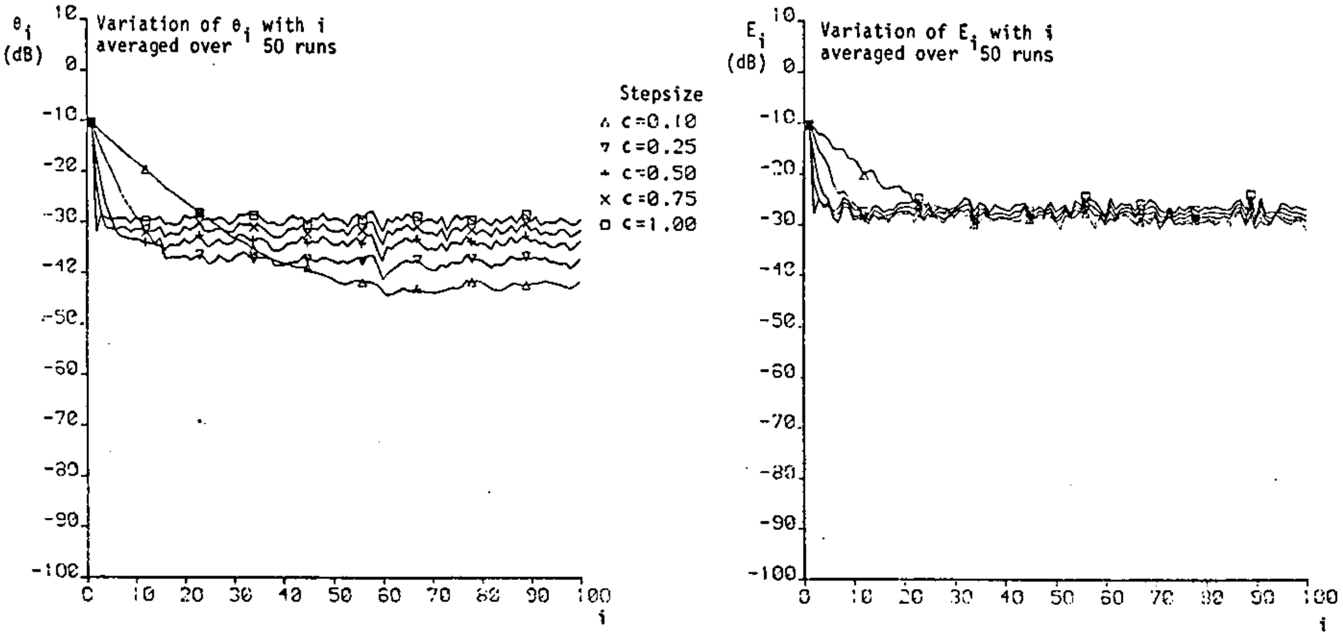


FIGURE 3.14: Performance of System 3.2 for Channel B, SNR = 30 dB

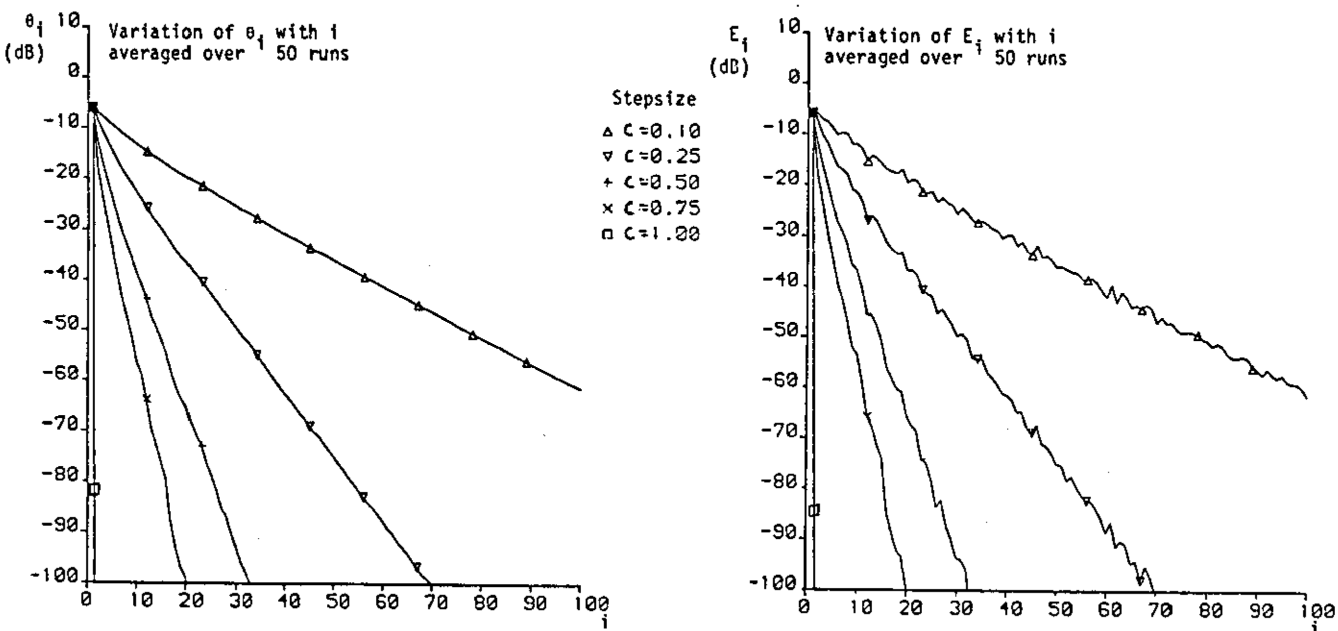


FIGURE 3.15: Performance of System 3.2 for Channel C, SNR = ∞ dB

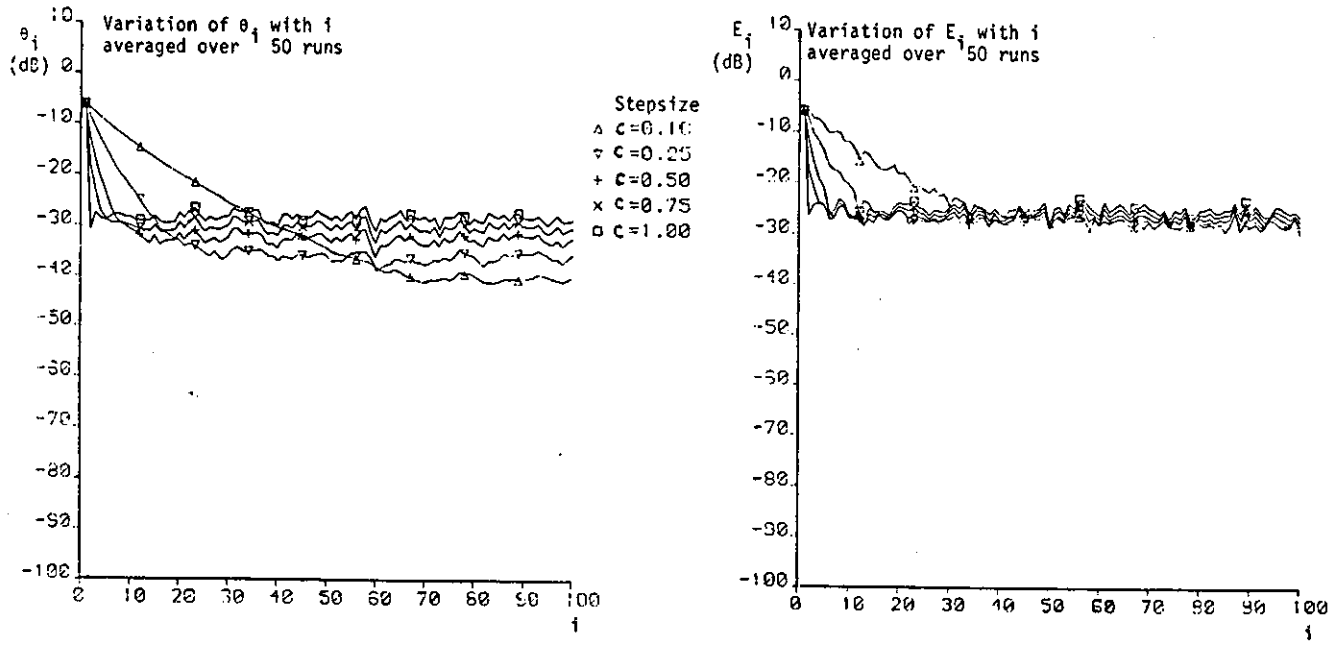


FIGURE 3.16: Performance of System 3.2 for Channel C, SNR = 30 dB

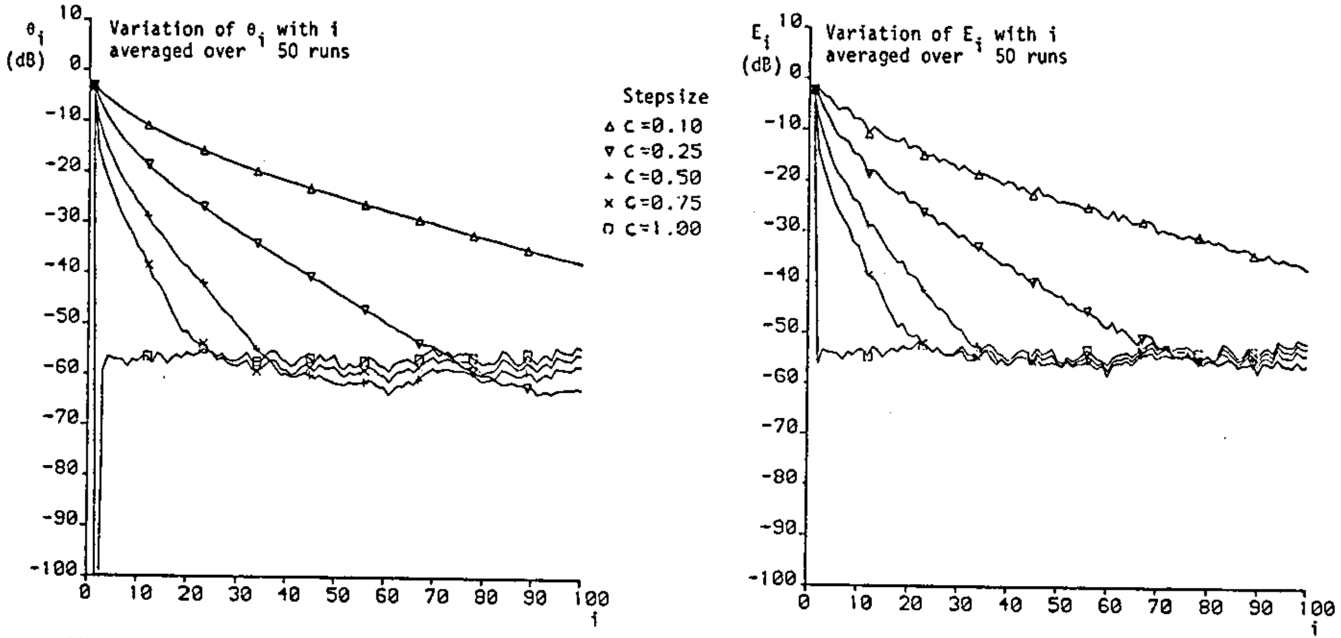


FIGURE 3.17: Performance of System 3.2 for Channel D, SNR = ∞ dB

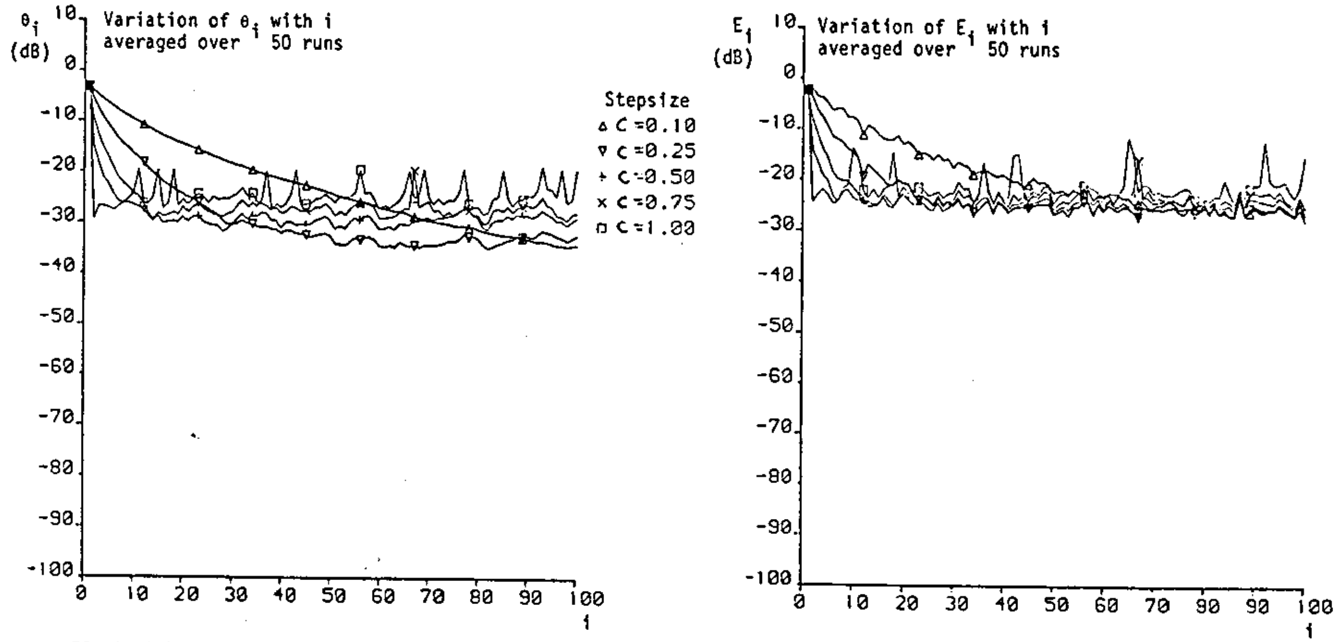


FIGURE 3.18: Performance of System 3.2 for Channel D, SNR = 30 dB

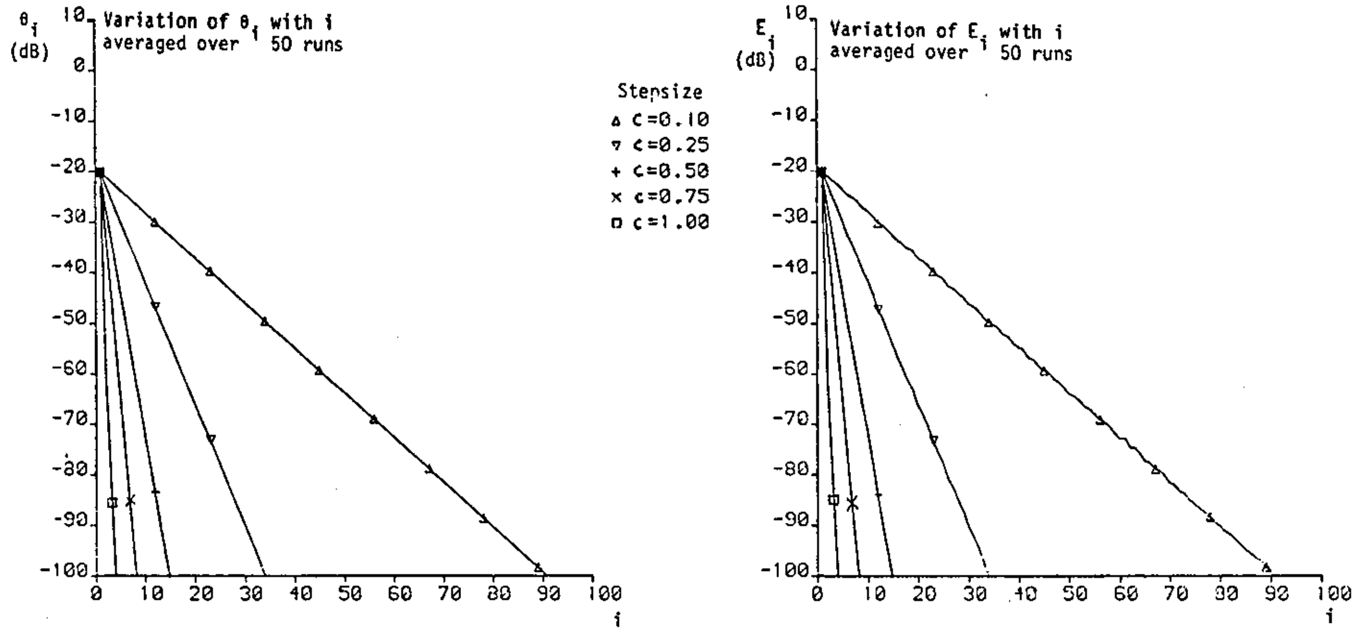


FIGURE 3.19: Performance of System 3.3 for Channel A, SNR = ∞ dB

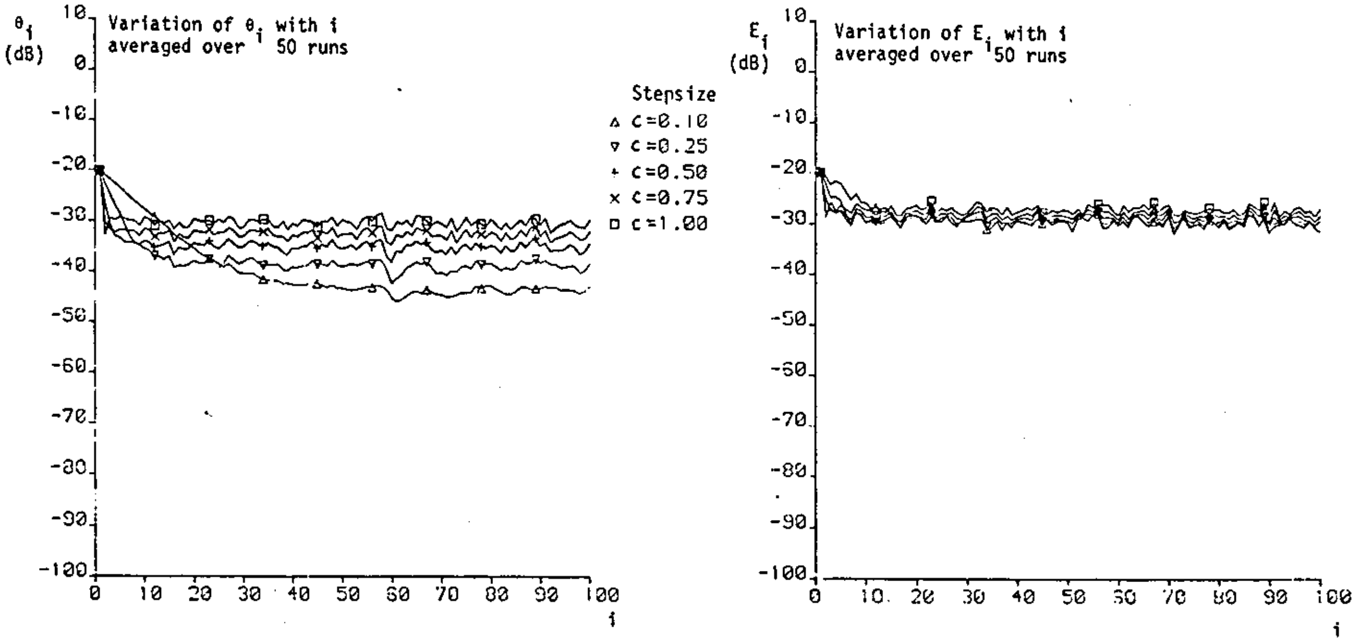


FIGURE 3.20: Performance of System 3.3 for Channel A, SNR = 30 dB

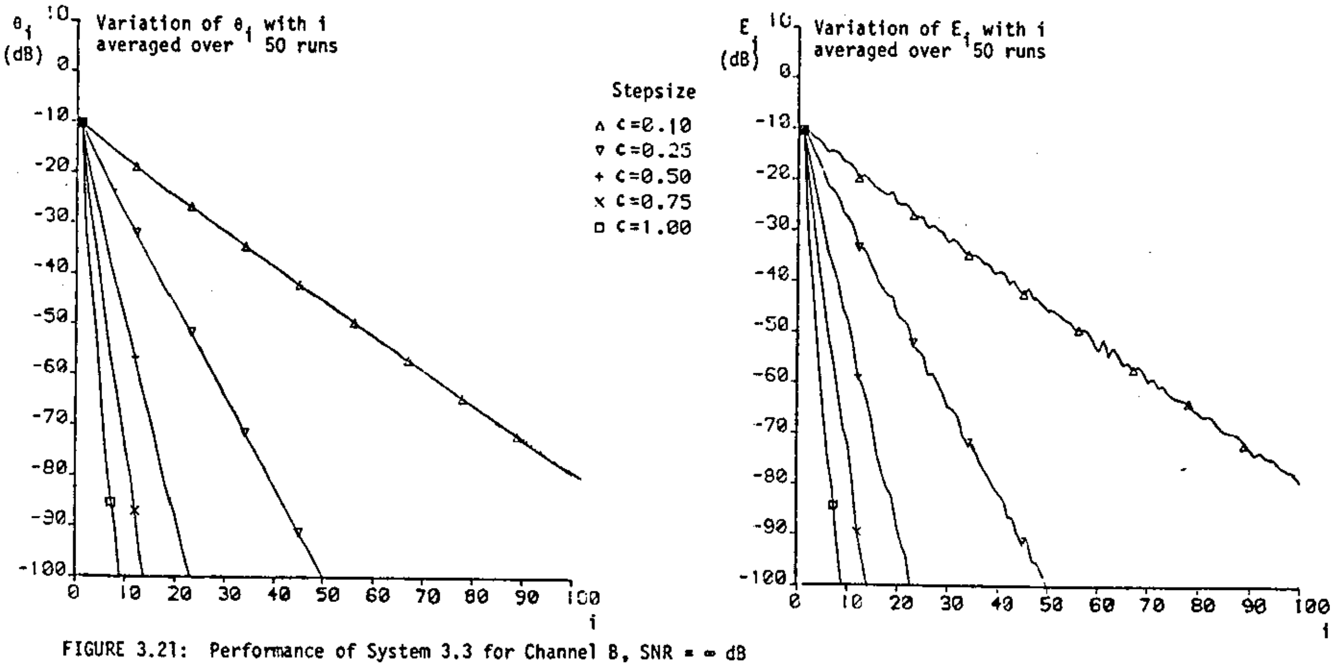


FIGURE 3.21: Performance of System 3.3 for Channel B, SNR = ∞ dB

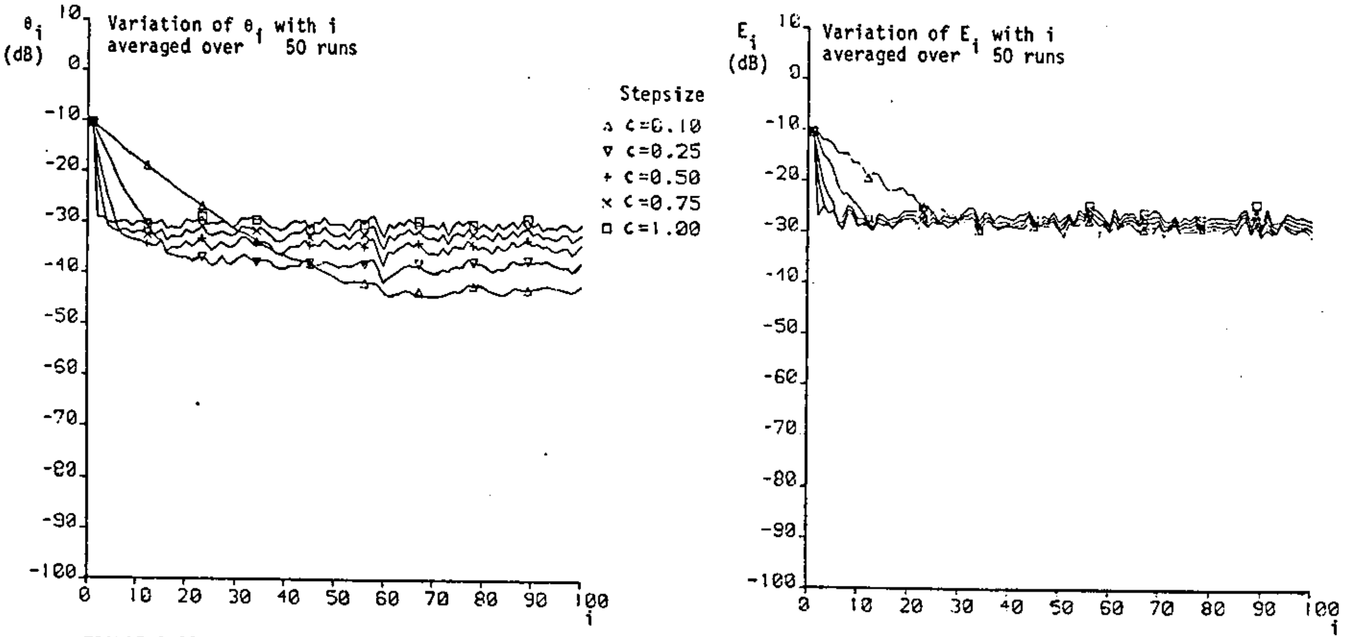


FIGURE 3.22: Performance of System 3.3 for Channel B, SNR = 30 dB

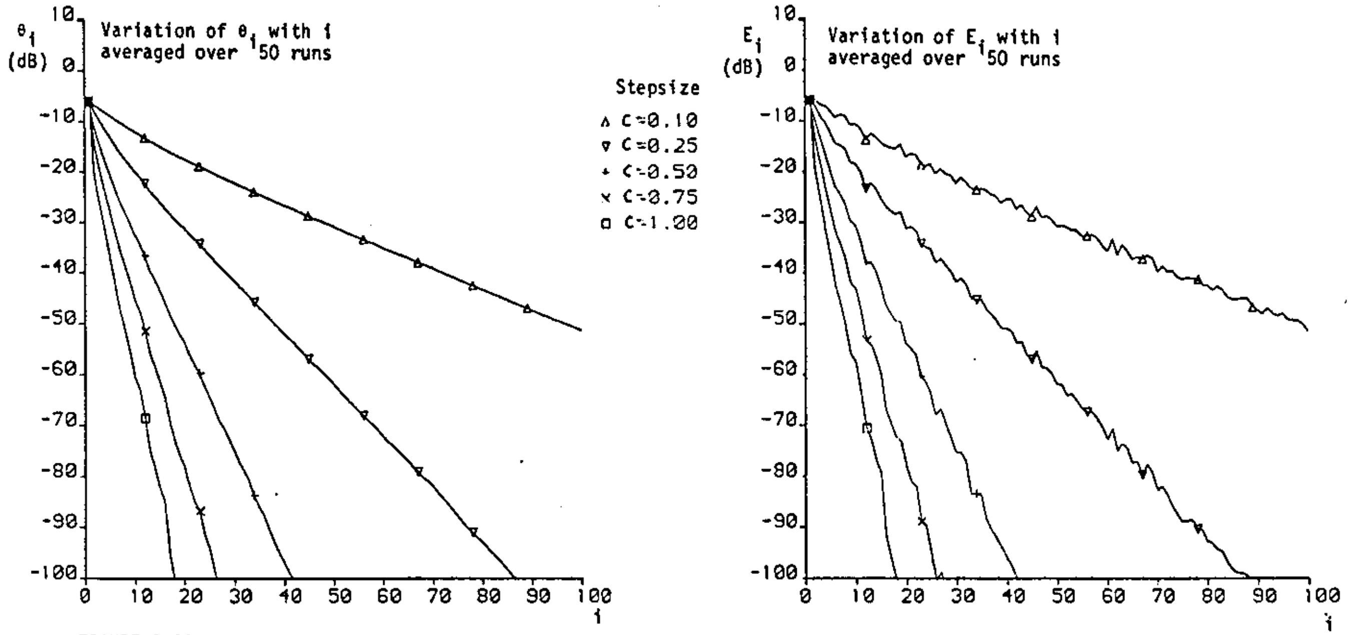


FIGURE 3.23: Performance ofSystem 3.3 for Channel C, SNR = ∞ dB

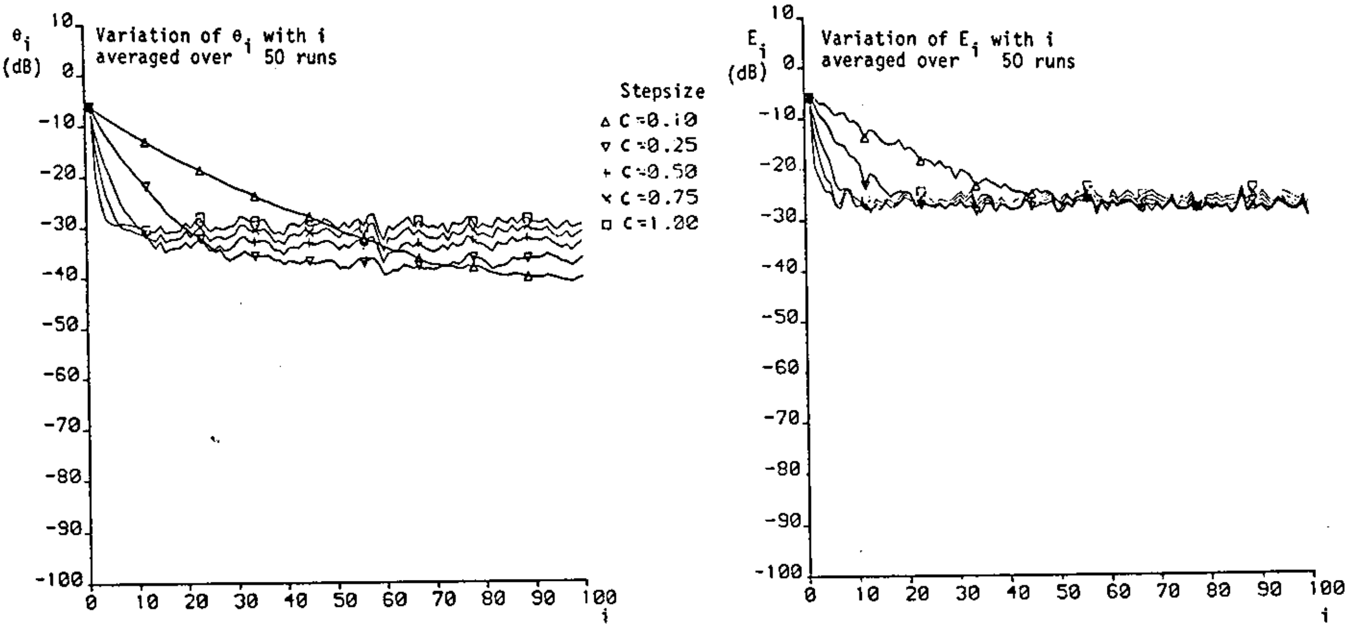


FIGURE 3.24: Performance of System 3.3 for Channel C, SNR = 30 dB

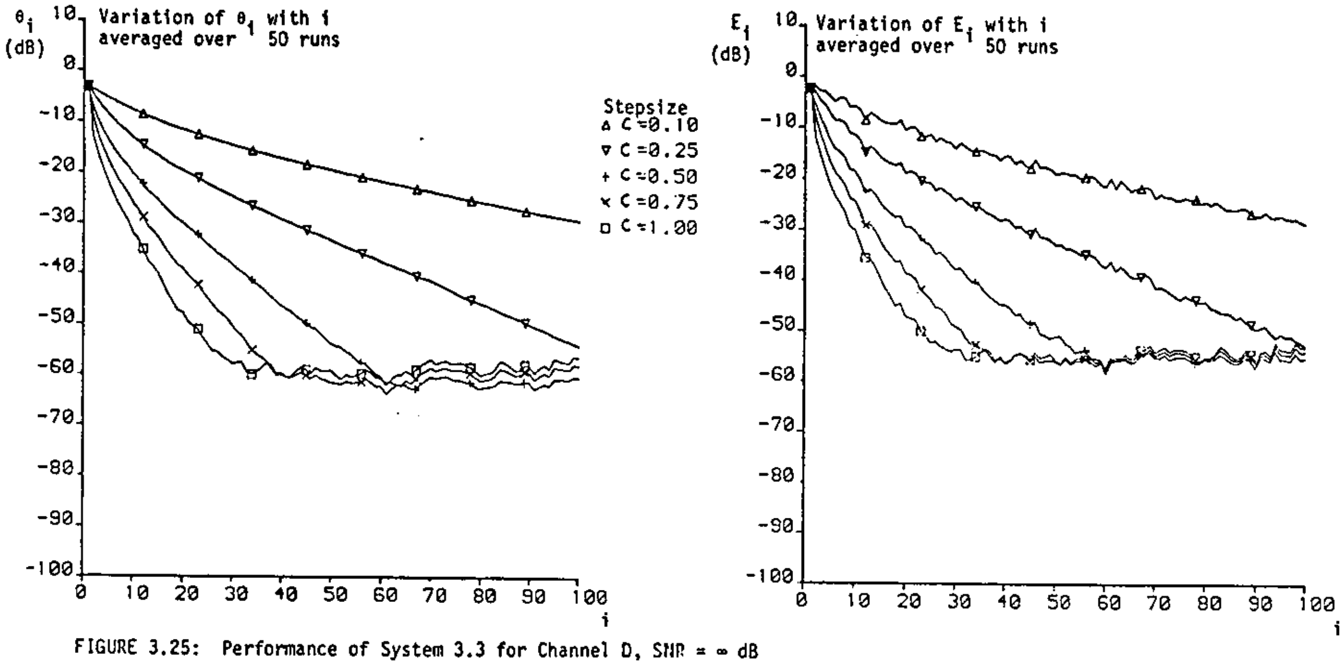


FIGURE 3.25: Performance of System 3.3 for Channel D, SNR = ∞ dB

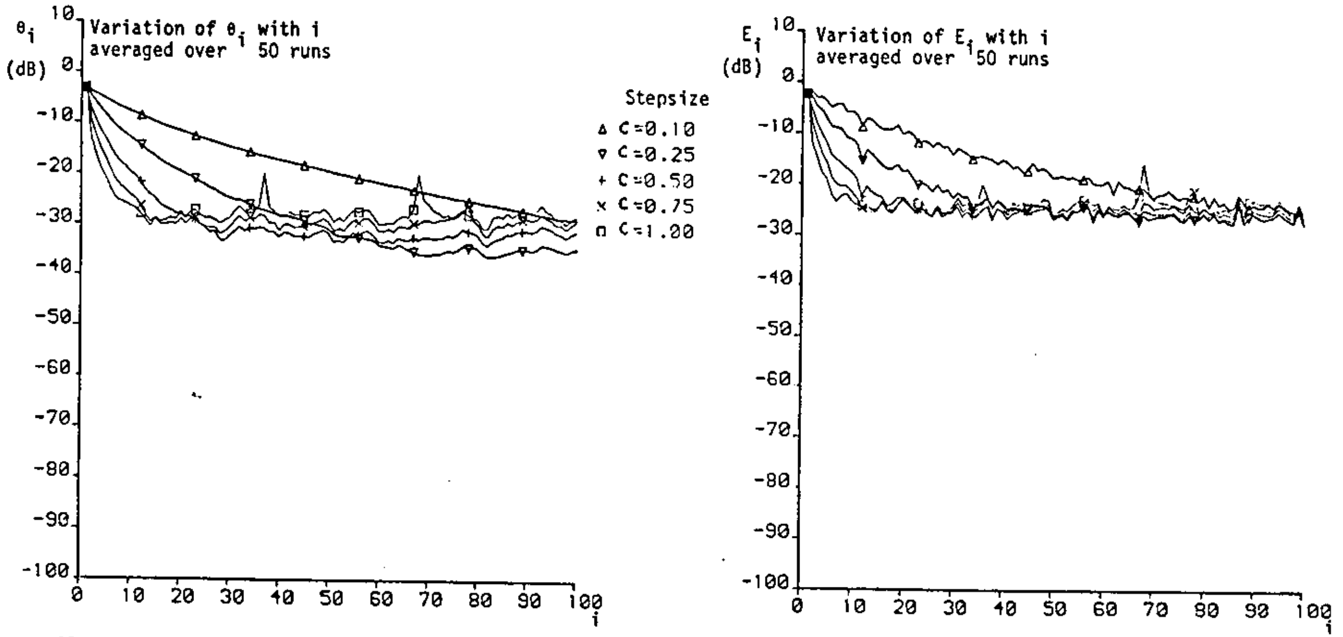


FIGURE 3.26: Performance of System 3.3 for Channel D, SND = 30 dB

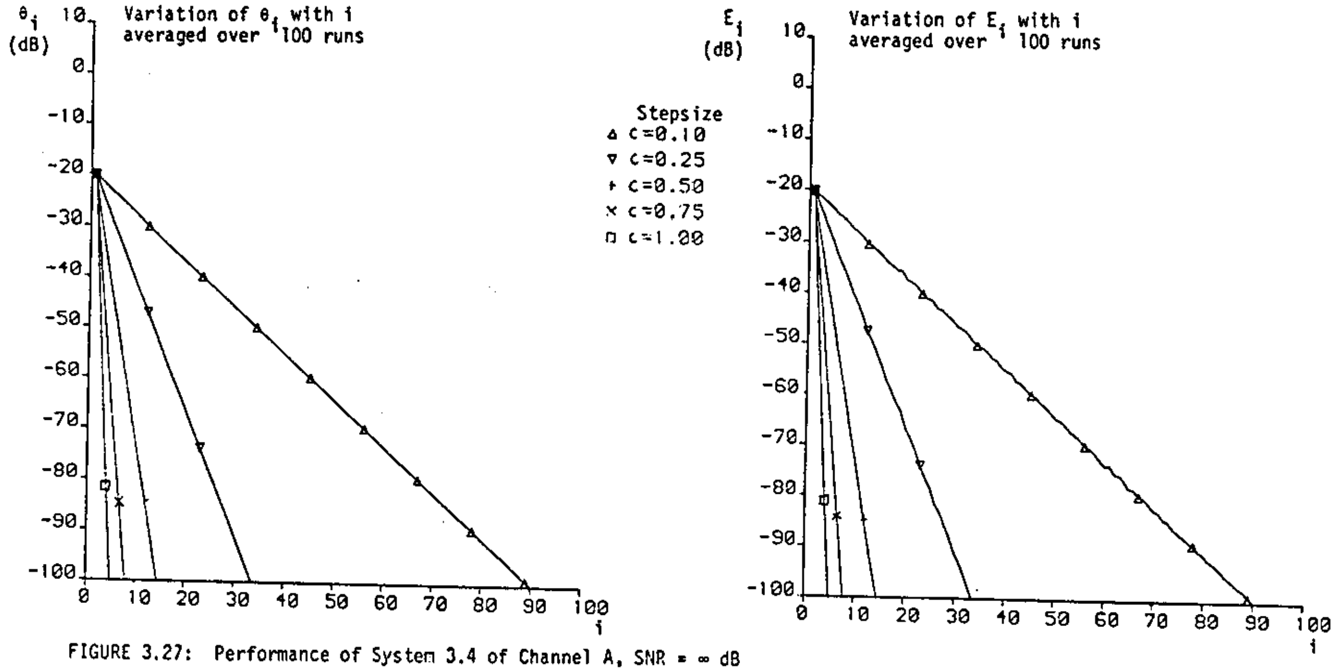


FIGURE 3.27: Performance of System 3.4 of Channel A, SNR = ∞ dB

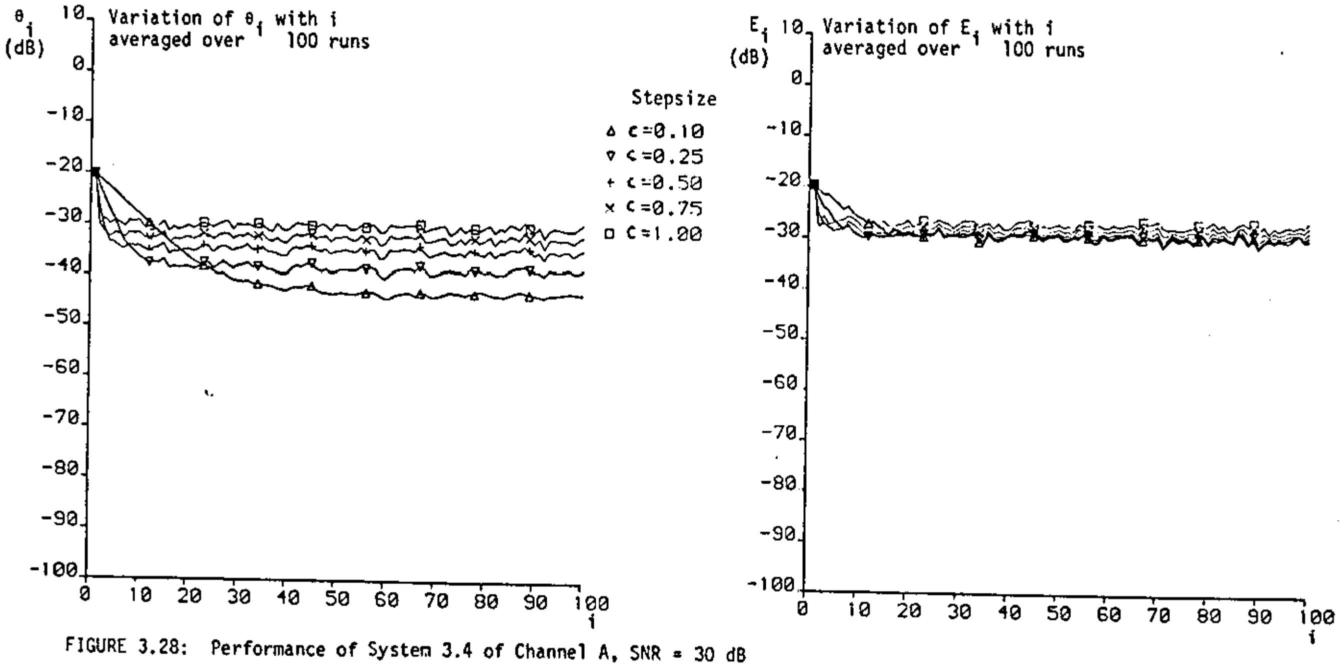


FIGURE 3.28: Performance of System 3.4 of Channel A, SNR = 30 dB

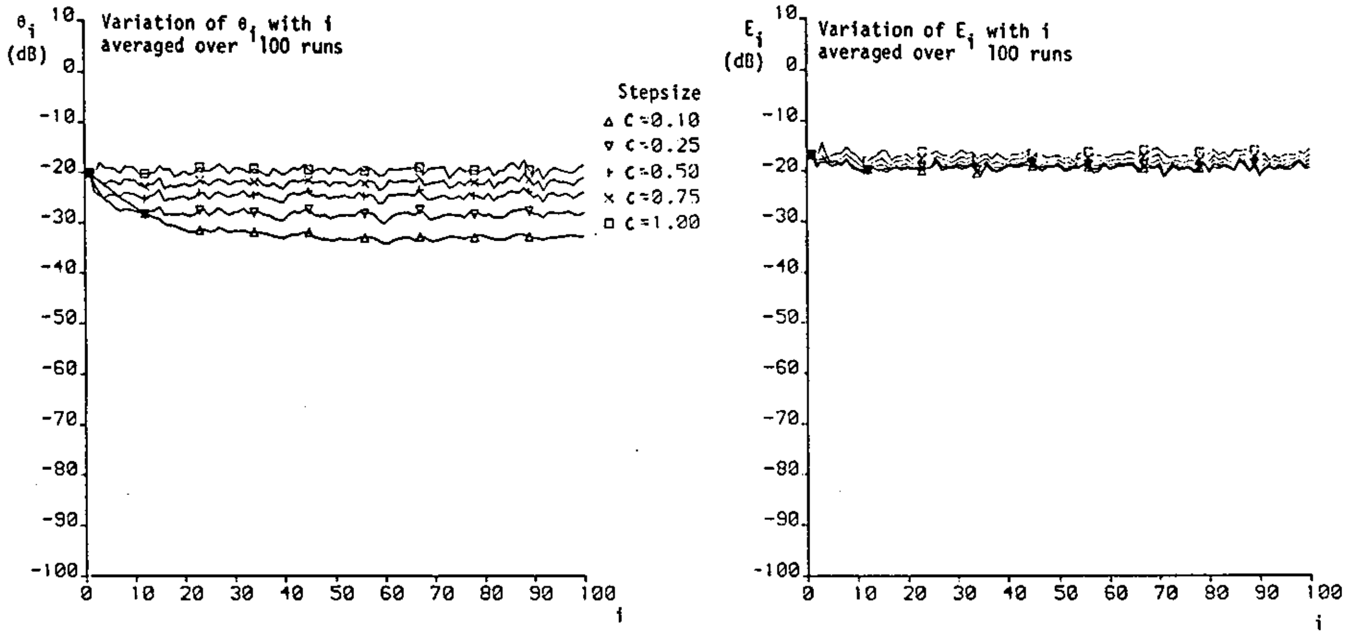


FIGURE 3.29: Performance of System 3.4 of Channel A, SNR = 20 dB

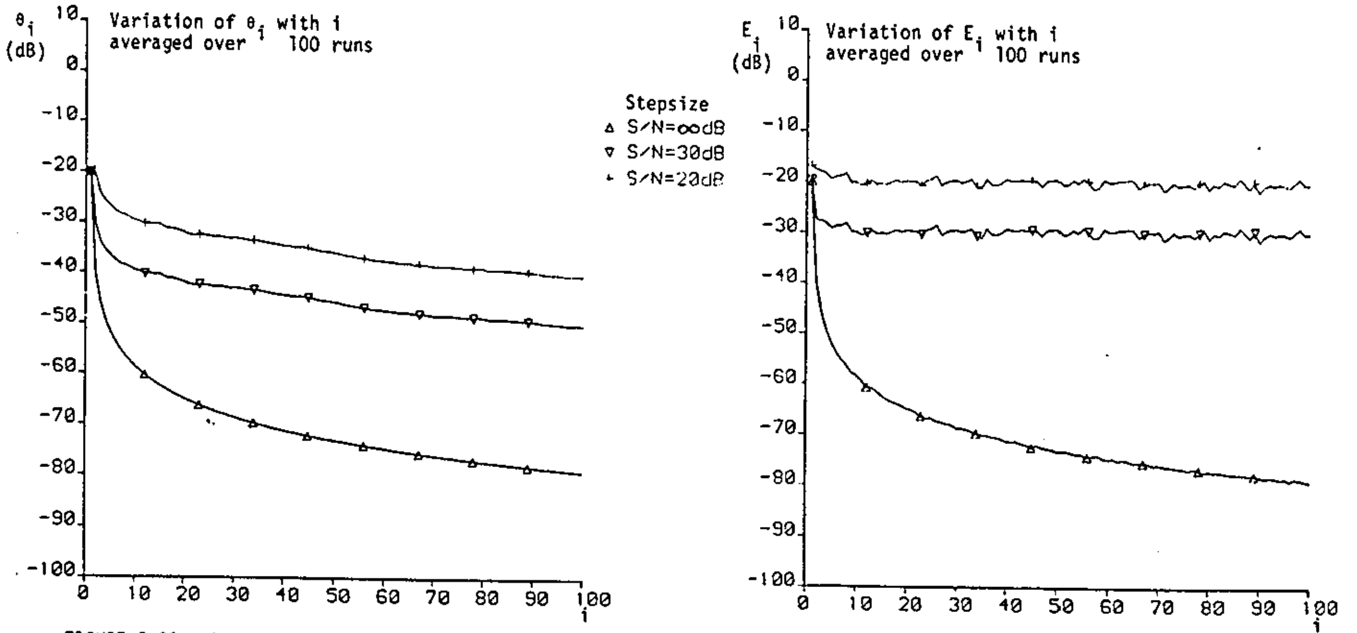


FIGURE 3.30: Performance ofSystem 3.4 for Channel A, c=1/i

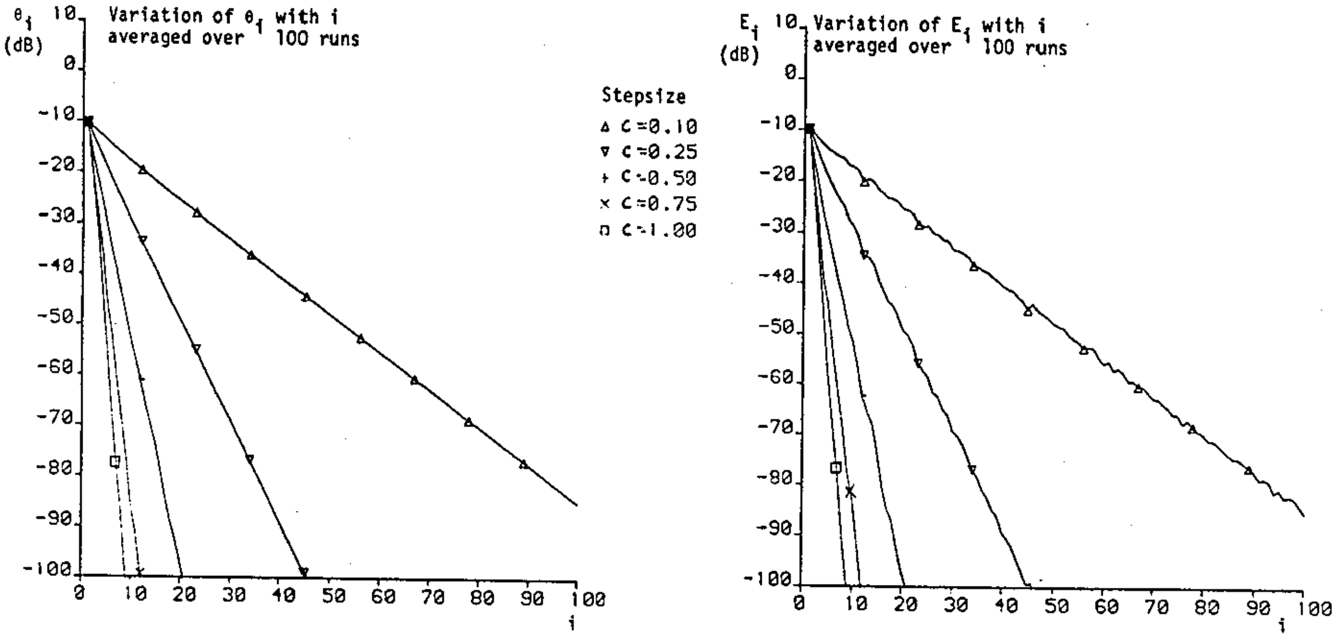


FIGURE 3.31: Performance of System 3.1 of Channel B, SNR = ∞ dB

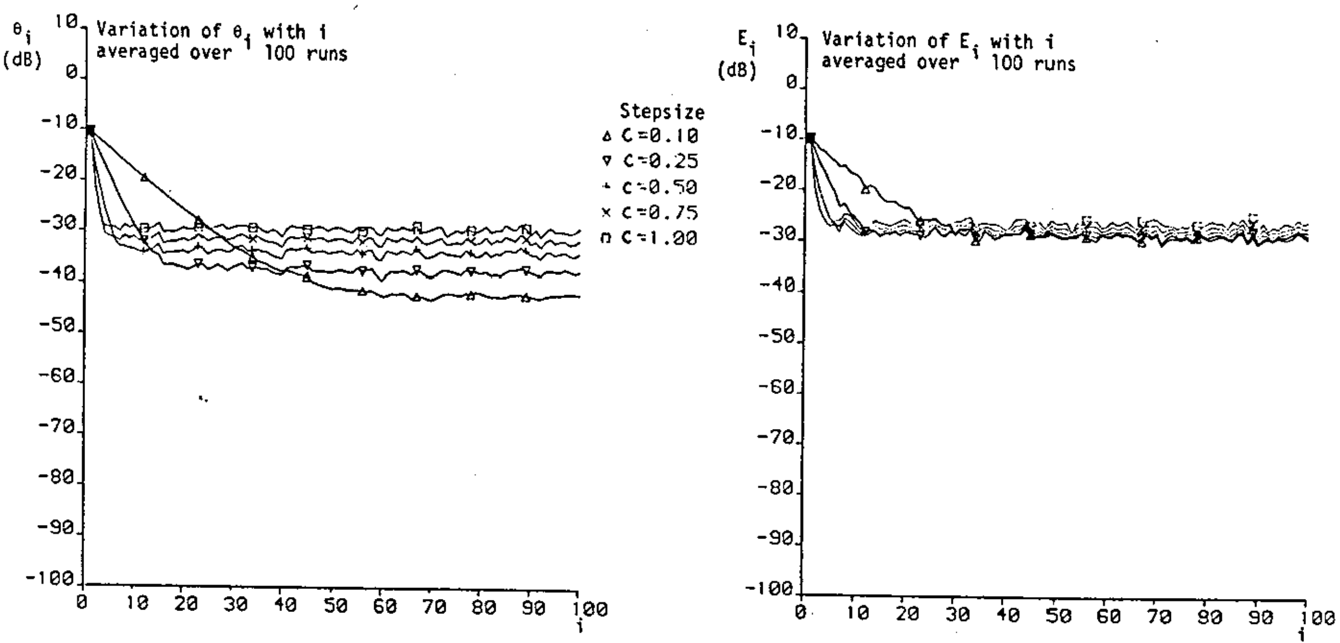


FIGURE 3.32: Performance of System 3.4 of Channel B, SNR = 30 dB

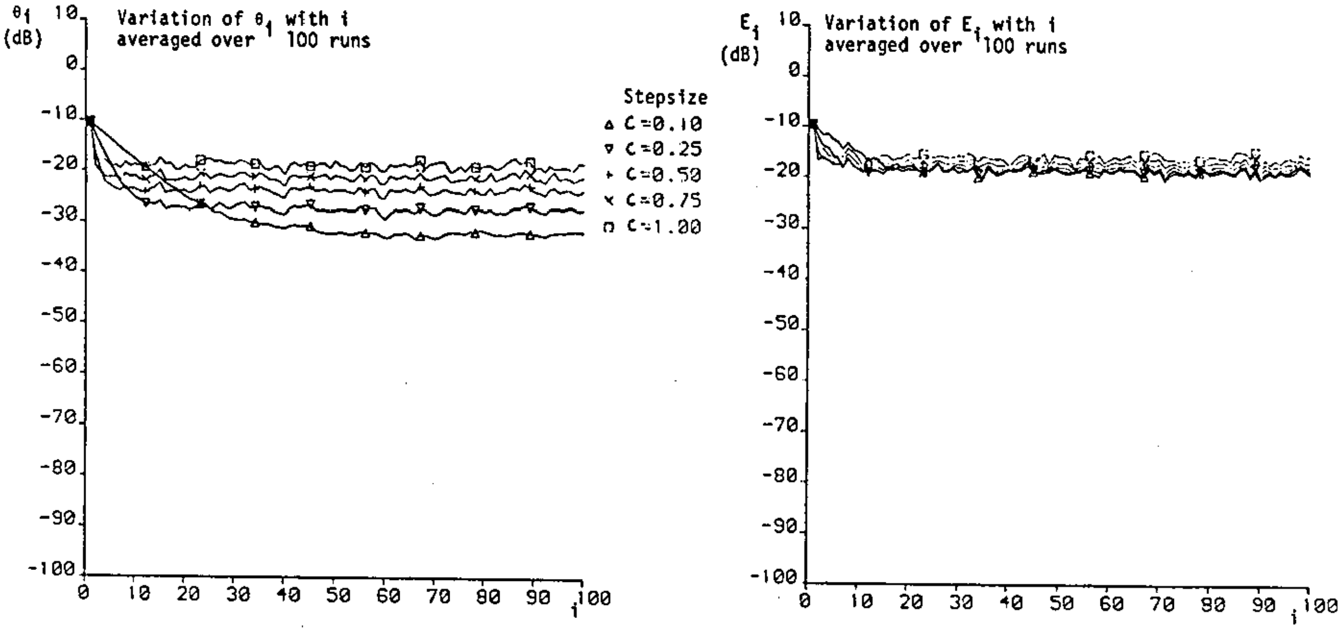


FIGURE 3.33: Performance of System 3.4 of Channel B, SNR = 20 dB

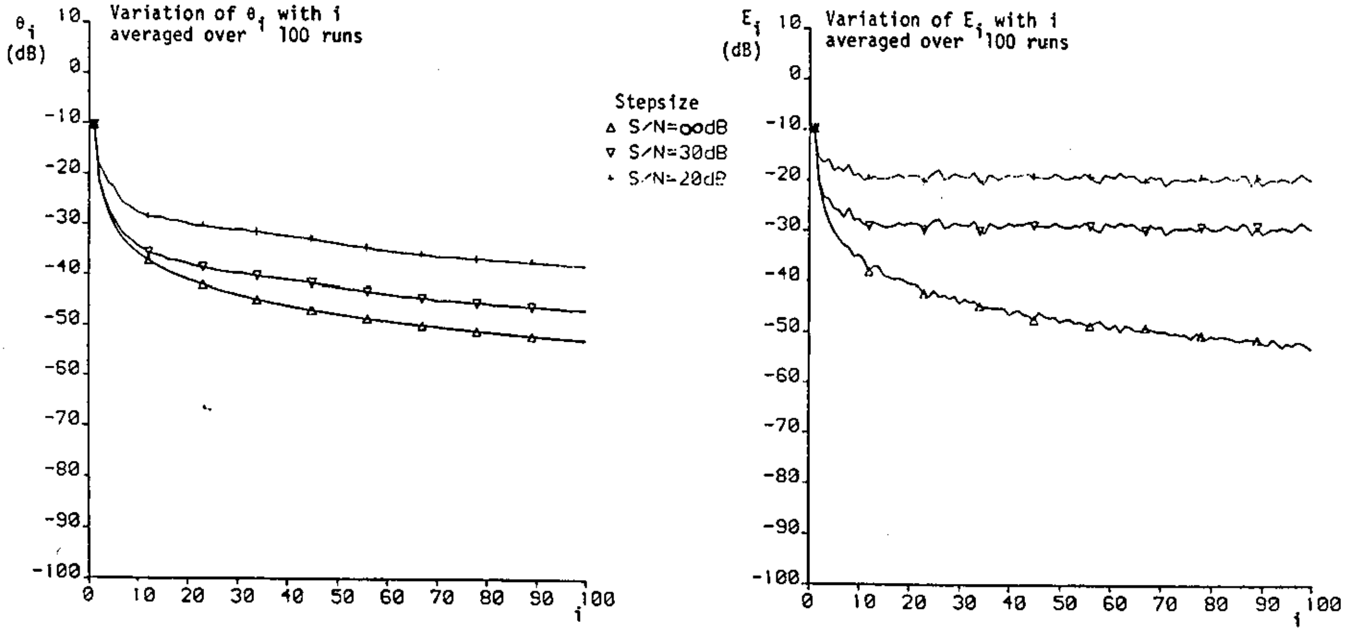


FIGURE 3.34: Performance of System 3.4 for Channel B, c=1/i

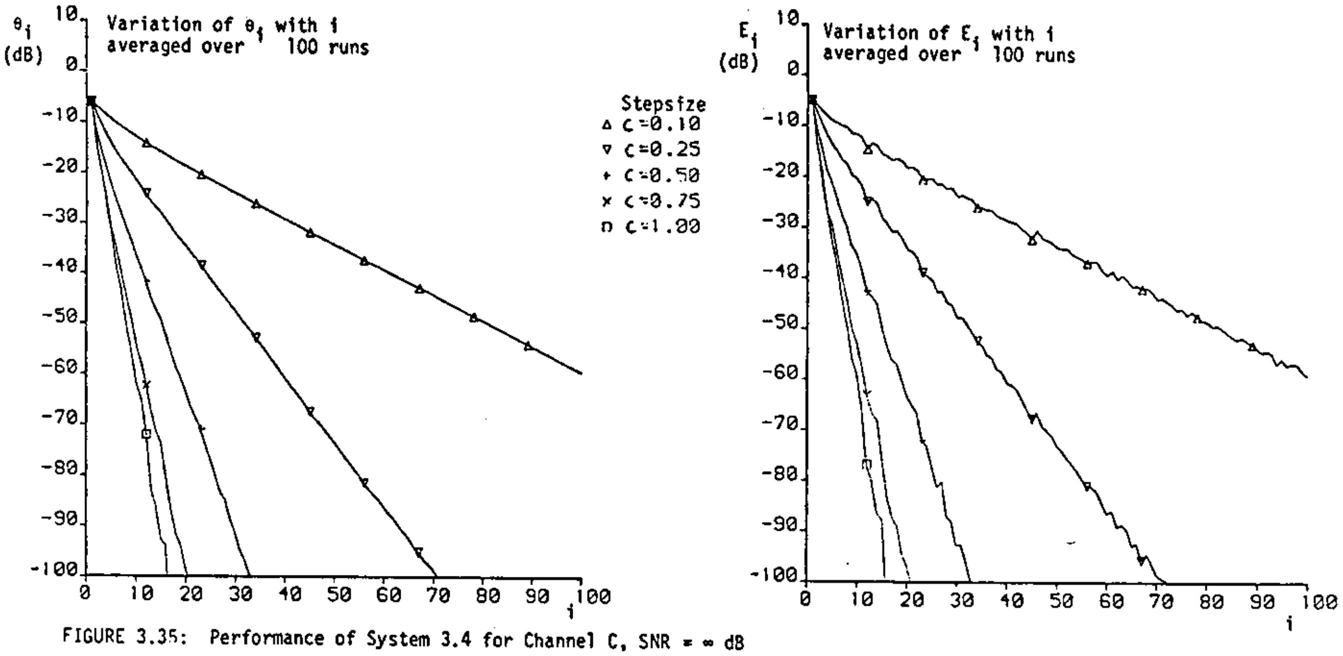


FIGURE 3.35: Performance of System 3.4 for Channel C, SNR = ∞ dB

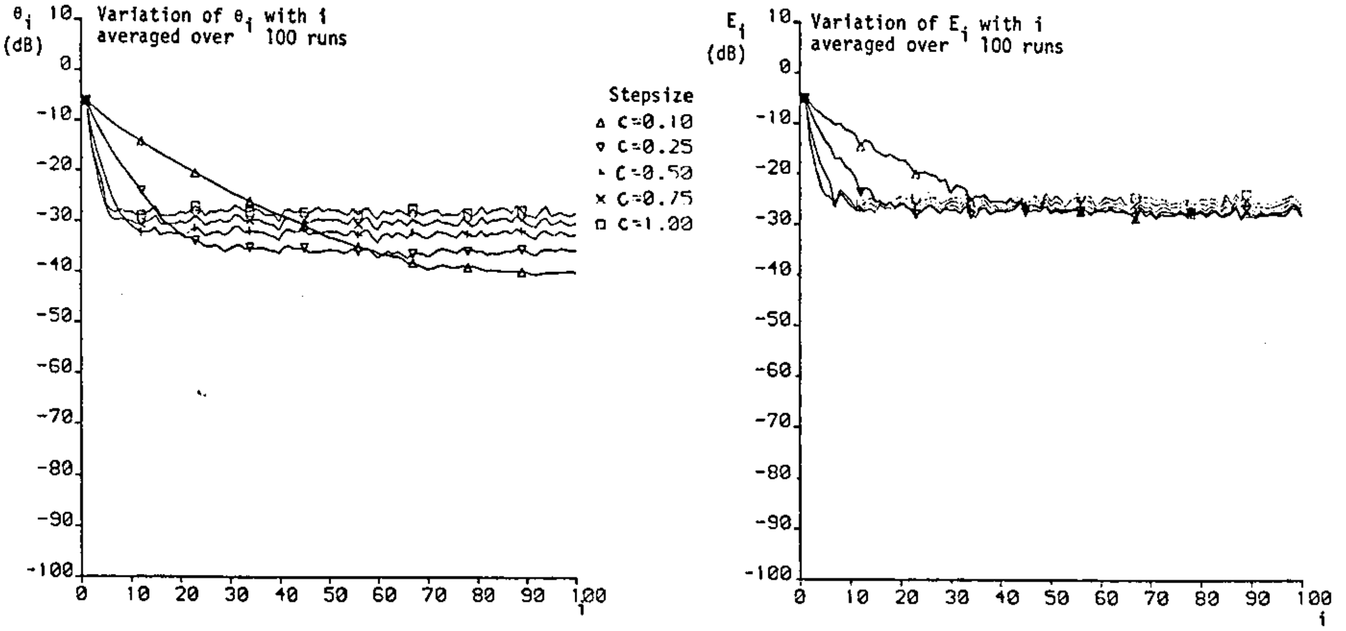


FIGURE 3.36: Performance of System 3.4 for Channel C, SNR = 30 dB

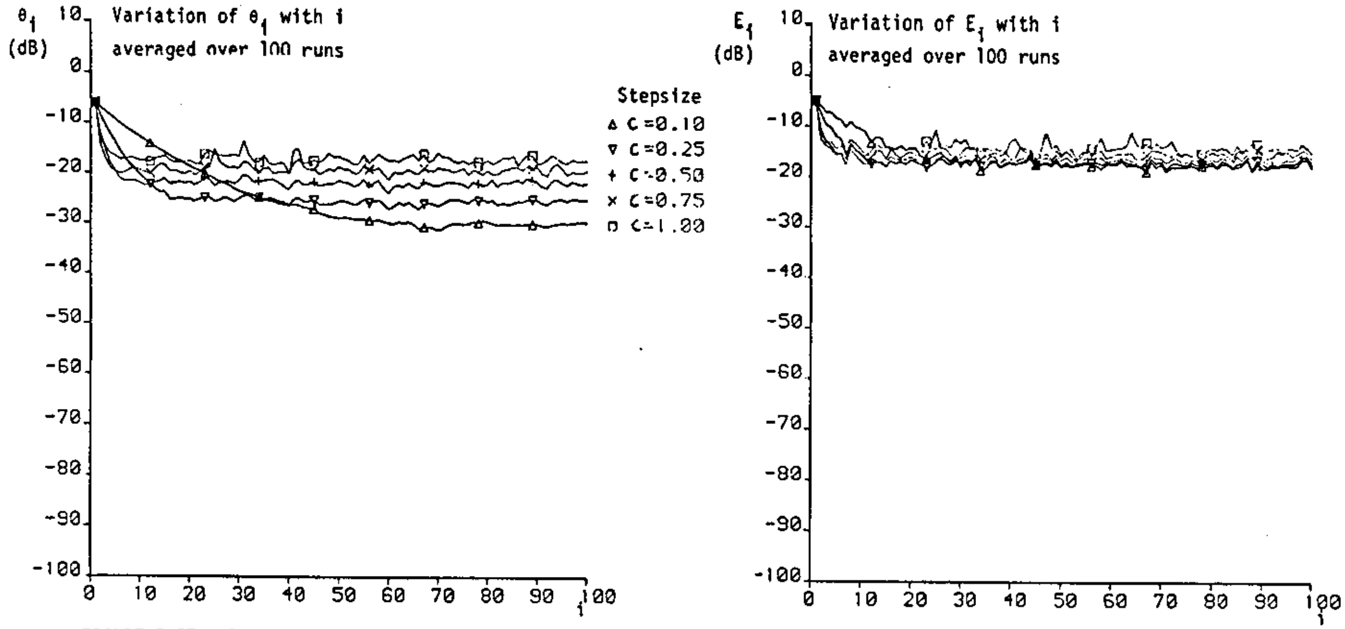


FIGURE 3.37: Performance of System 3.4 for Channel C, SNR = 20 dB

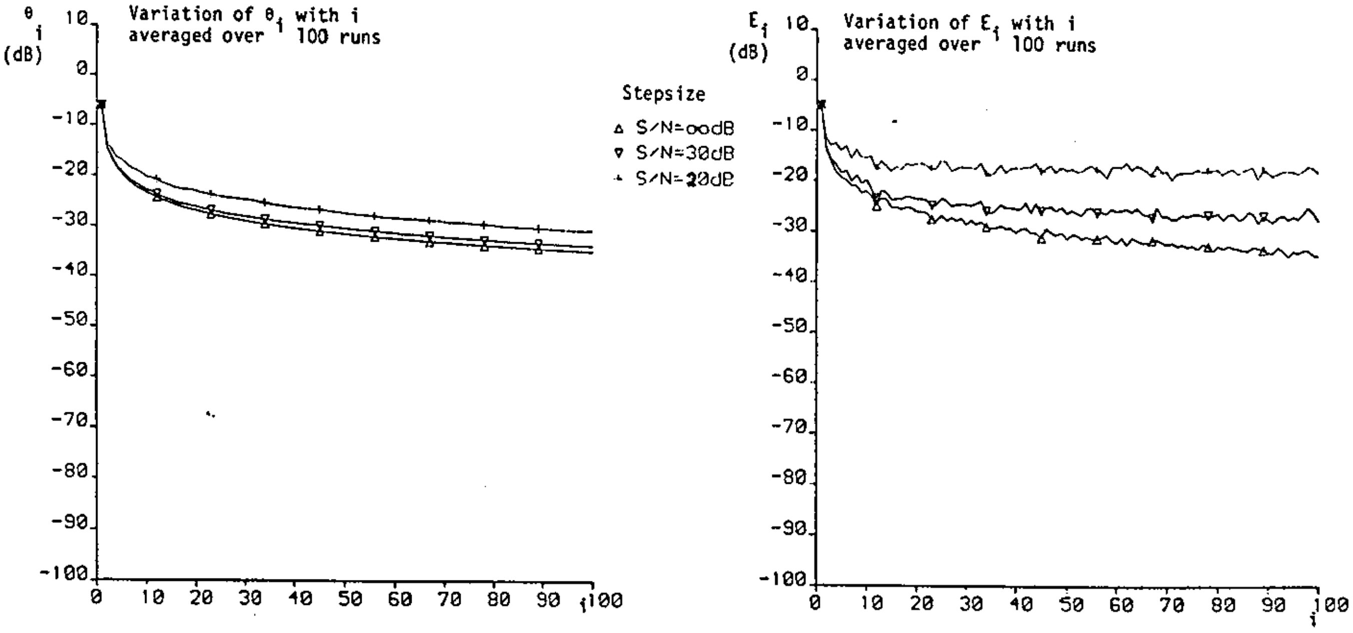


FIGURE 3.38: Performance of System 3.4 for Channel C, c=1/i

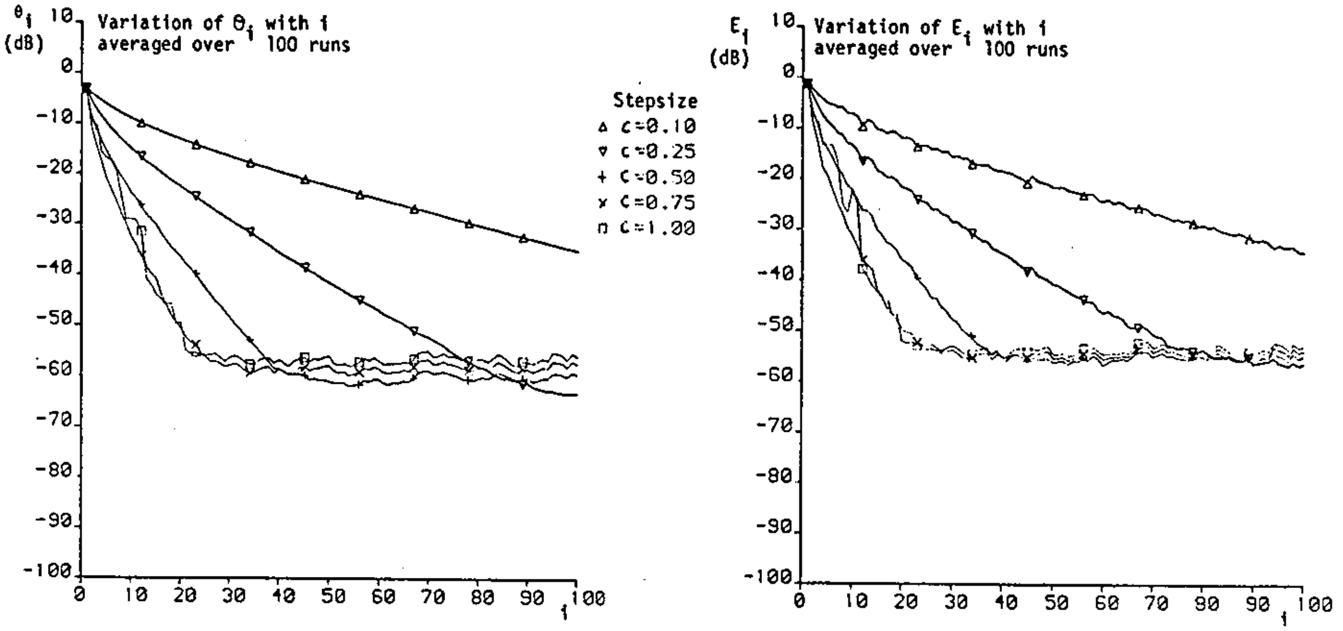


FIGURE 3.39 Performance of System 3.4 for Channel D, SNR = ∞ dB

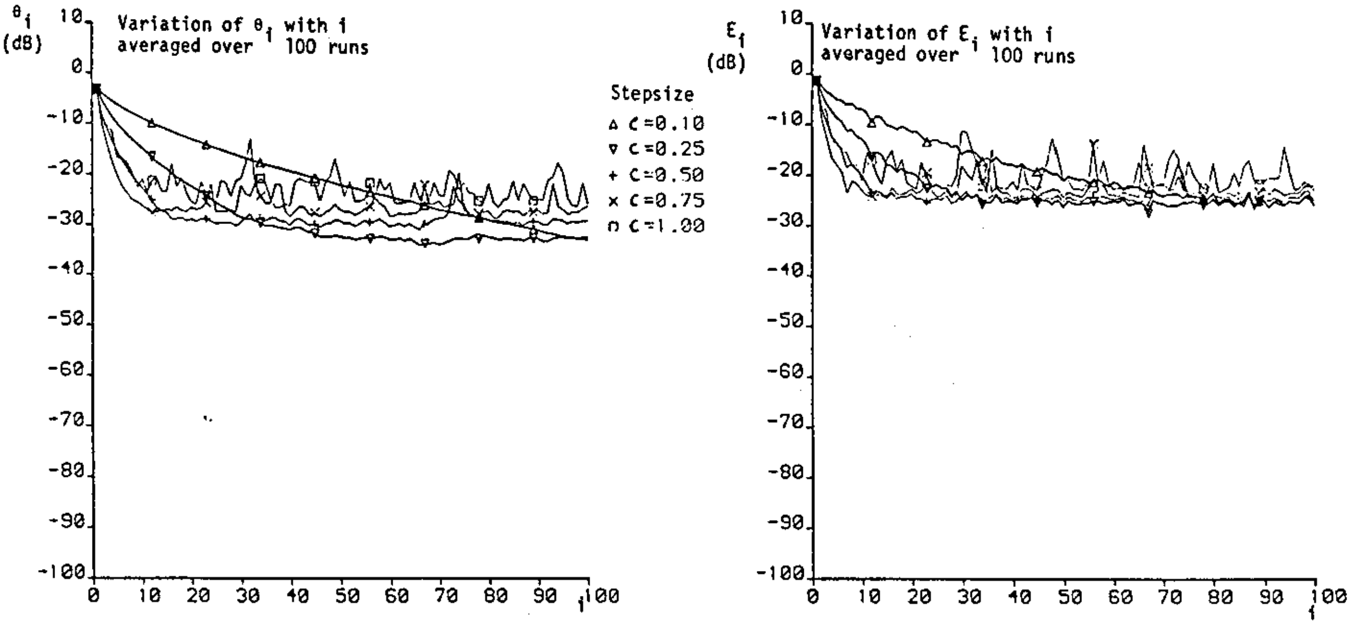


FIGURE 3.40: Performance of System 3.4 for Channel D, SNR = 30 dB

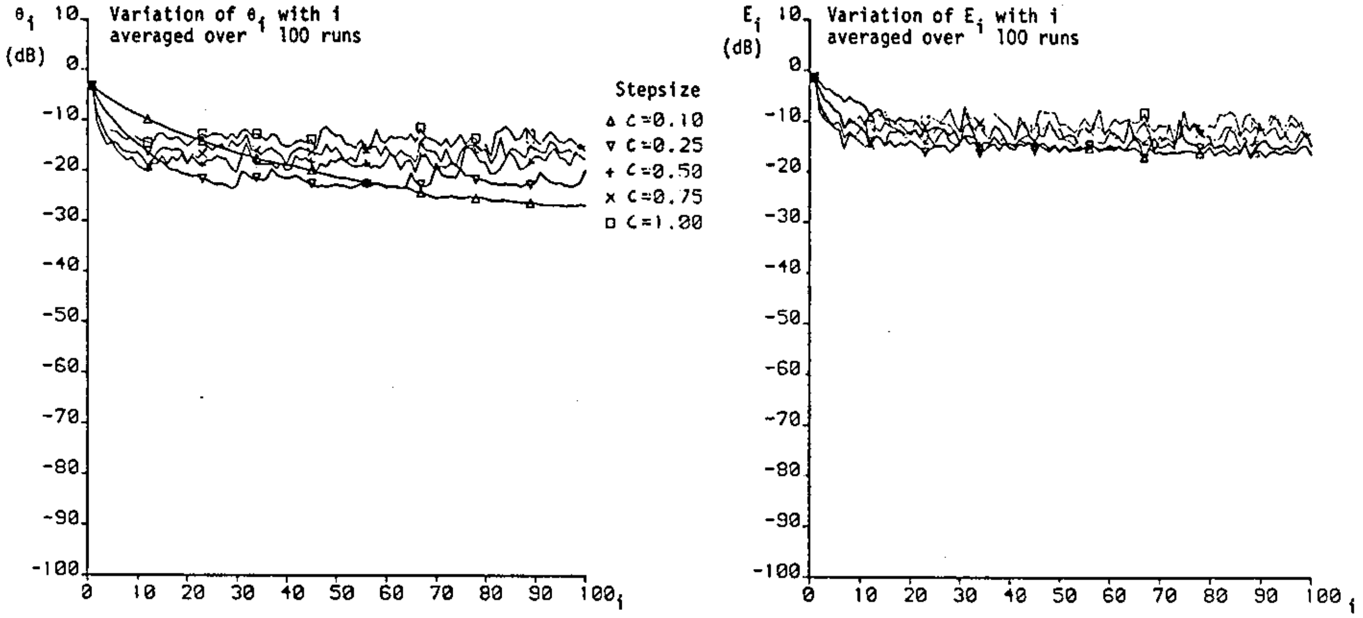


FIGURE 3.41: Performance of System 3.4 for Channel D, SNR = 20 dB

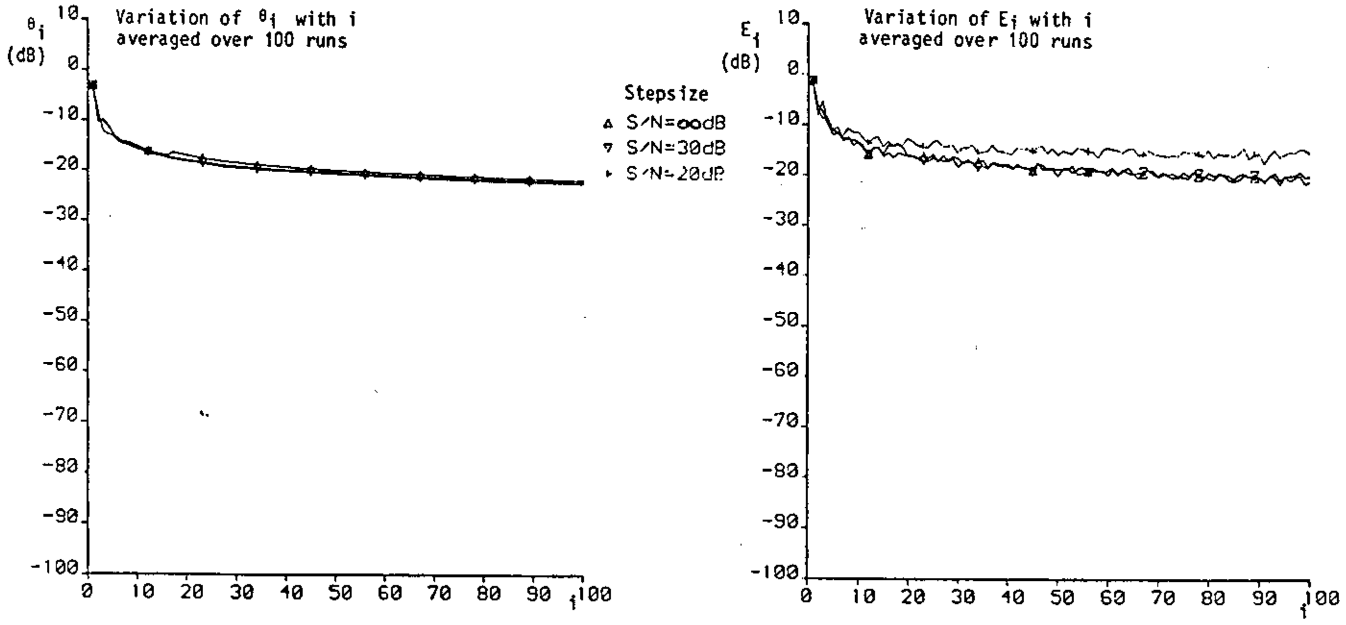


FIGURE 3.42: Performance of System 3.4 for Channel D, c=1/i

3.2 Studies of Various On-line Systems Based on the 16-point QAM System Operating at 9600 bit/s

The feasibility study of Section 3.1 based on the model of a synchronous serial binary data-transmission system[6,7] has revealed that a suitable algorithm can be found for estimating the roots (zeros) of Y(z) (equation 3.3) that lie outside the unit circle in the z-plane. Although the four test channels, Channels A, B, C and D (Table 3.1), do not represent practical telephone circuits in the British public switched telephone network at high transmission rates such as 9600 bit/s using 16-point QAM signal in that there are only three real-valued components in {Y} (equation 3.2), but the range of the magnitudes of the single root (zero) of {Y(z)} outside the unit circle is fairly typical of a majority of roots (zero) of Y(z) outside the unit circle for the mild to bad channels (Figures 2.9-2.12). Thus, the computer-simulation tests can be taken as quick indications on the suitability of the different algorithms developed for the finding of a root (zero) of Y(z) in simple cases. The results of the latest development, System 3.4, show that very fast convergent rates and very accurate estimates of the roots are possible in the absence of additive white Gaussian noise and, furthermore, the algorithm behaves surprisingly well in signal to noise ratios of both 30 and 20 dB.

This section is concerned with the investigation of System 3.4 and further developments of the on-line system for the 16-point QAM system operating at 9600 bit/s over the British public switched telephone network. The characteristics of four telephone circuits with varying degrees of amplitude and group-delay distortions are shown in Figures 2.4-2.7 and thesebaseband sampled impulse-responses are given in Table 2.1. For the purpose of this investigation, Channel 1 (Figure 2.4) and Channel 3 (Equation 2.6) are chosen which represent a typical line and a typical worst line for transmission of digital data at 9600 bit/s, respectively. The sampled impulse-responses of Channels 1 and 3 are given in Table 2.1 and the roots (zeros) of Y(z) for the two channels are shown in Figures 2.9 and 2.11 respectively, where the roots outside the unit circle are numbered in order of decreasing magnitude, starting at number 1 with the root having the greatest absolute value. It can be seen that Channel 1 has 3 roots outside the unit circle and Channel 3 has 4. Let the sampled impulse-response of the channel be

$$Y = [y_0 \quad y_1 \; . \; . \; . \; y_g] \tag{3.93}$$

and its z-transform be

$$Y(z) = y_0 + y_1 z^{-1} + \ldots + y_g z^{-g} \tag{3.94}$$

where $\{y_i\}$ are complex-valued. Suppose that there are m roots (zeros) of Y(z) lying outside the unit circle, so that

$$Y(z) = Y_1(z)\, Y_2(z) \tag{3.95}$$

where

$$Y_1(z) = \eta(1+\alpha_1 z^{-1})(1+\alpha_2 z^{-1})\ldots(1+\alpha_{g-m} z^{-1}) \tag{3.96}$$

and

$$Y_2(z) = z^{-m}(1+\beta_1 z)(1+\beta_2 z)\ldots(1+\beta_m z) \tag{3.97}$$

where $|\alpha_i|<1$ and $|\beta_i|<1$, and η is the appropriate complex value needed to satisfy equations 3.95-3.97. From equation 3.94,

$$\begin{aligned} Y(z) &= y_0 + y_1 z^{-1} + \ldots + y_g z^{-g} \\ &= (1+\beta_1 z)(u_0 z^{-1} + u_1 z^{-2} + \ldots + u_{g-1} z^{-g}) \end{aligned} \tag{3.98}$$

where $-\frac{1}{\beta_1}$ is one of the m roots outside the unit circle.

Suppose that the root, $-\frac{1}{\beta_1}$, can be found by a root-finding algorithm and that the receiver operates on the channel such that the root $-\frac{1}{\beta_1}$ is removed and replaced by $(-\beta_1)^*$ in z-transform of the sampled impulse-response of the channel. Thus, the z-transform of the resultant sampled impulse-response of the channel has now (m-1) roots outside the unit circle. As these m roots are processed by the receiver, one at a time, a set of (m-1) resultant sampled impulse--responses are produced whose z-transforms have (m-1),(m-2),...,1 roots outside the unit circle, and when all m roots have been processed, the channel is then minimum phase. In order to investigate the performance of the different algorithms developed in this section which are designed, essentially, to find only one root of Y(z) outside the unit circle, the said (m-1) resultant sampled impulse-responses need to be generated separately. To do this, the roots of Y(z) are, first of all, calculated using the NAG subroutine CO2ADF[50] and those that lie outside the unit circle are, depending on the outcome of the root-finding algorithm being tested, removed and replaced by their complex conjugate-reciprocals thus producing the set of (m-1) {Y(z)}, with (m-1), (m-2),...,1 rootslying outside the unit circle. The generation of the (m-1) {Y(z)} is dependent on the root-finding algorithm which is being tested.It will become apparent later because when there are more than one root lying outside the unit circle, the order in which the roots are found is not predictable. Therefore, in order to simulate the condition where the set of roots of Y(z) of the raw channel are processed one by one as it would by the algorithm without having the algorithm actually carrying out the task, the generation of the (m-1) {Y(z)} must depend on the order in which the roots are found. Thus, for each of the channels used in this part of the investigation, m{Y(z)}(including the raw Y(z)) are produced and tested using computer-simulation. These are summarized as in Table 3.2. Thus CH20R0 and CH19R0 denote the raw sampled impulse-responses of Channels 1 and 3 respectively.

TABLE 3.2: Names Given to Resultant Sampled Impulse-Responses of Channels 1 and 3 at Different Stages of the 'Minimum-Phase' Process

CH20R0	-	Channel 1 without any root processed
CH20R1	-	Channel 1 with root 1 processed
CH20R2	-	Channel 1 with roots 1 and 2 processed
CH19R0	-	Channel 3 without any root processed
CH19R1	-	Channel 3 with root 1 processed
CH19R2	-	Channel 3 with roots 1 and 4 processed
CH19R3	-	Channel 3 with roots 1, 4 and 3 processed

The QAM system is as described in Chapter 2 and the arrangement of the receiver is identical to that of the synchronous serial binary data-transmission system shown in Figure 3.1, except that, in this case, all the signals concerned are complex-valued. The detailed description of the operation of the receiver, in particular that of the feedforward and feedback filters and the method of intersymbol-interference cancellation, is given in Chapter 3.1. Briefly, at time t=(i+n)T, the receiver operates on the stream of received signals $\{r_i\}$ (equation 3.4) in the manner described and produces a sequence of $\{x_i\}$ (equation 3.58). The latter sequence can then be used in the estimation of β_1 (negative reciprocal of the root of Y(z)). The performance of the root-finding algorithm is measured in terms of θ_i (equation 3.73) and E_i (equation 3.74).

3.2.1 Performance of System 3.4 for the 16-point QAM System

Results of computer-simulation tests of Channels A, B, C and D (Table 3.1) given in Section 3.1.4 have shown that the root-finding algorithm of System 3.4, whereby the estimate of β_1 is obtained by

$$\lambda_{i+1} = \lambda_i + c\,\frac{x_i}{s_i q_o'(1)}$$

and

$$q_o'(1) = q_o'(i-1) + c\left(\frac{x_{i+1}}{s_i} - q_o'(i-1)\right)$$

(equations 3.91 and 3.92), is capable of giving accurate estimate of β_1 and a fast convergent rate at high signal to noise ratios. The investigation was based on the model of a synchronous serial binary data-transmission system (Figure 3.1). Prompted by the encouraging outcome of the tests, the same algorithm is now employed for the 16-point QAM system operating at 9600 bit/s. The operation of the receiver is as described in Section 3.1 and the derivation of the algorithm is repeated here. At time t=(i+n)T, where (n+1) is the number of taps in the adaptive linear filter (Figure 3.43) and it is assumed that there is no delay in the detector itself, the receiver, having operated in the manner as described, produces a sequence of $\{x_i\}$ (equation 3.58) at the output of the one-tap feedback filter (Figure 3.2).

Assuming that $|\lambda_i|<<1$, so that all terms containing λ_i or higher orders of λ_i can be ignored, then from equation 3.59

$$x_i \simeq (\beta_1-\lambda_i)q_o s_i + v_i$$

and

$$(\beta_1-\lambda_i) \simeq \frac{x_i}{s_i q_o} - \frac{v_i}{s_i q_o}$$

so that

$$\beta_1 \simeq \lambda_i + \frac{x_i}{s_i q_o} - \frac{v_i}{s_i q_o} \qquad (3.99)$$

Thus, if c_1 is a positive constant in the range 0 to 1, then

$$\lambda_{i+1} = \lambda_i + c_1 \frac{x_i}{s_i q_o} \qquad (3.100)$$

is a better estimate of β_1 than is λ_i. Now, from equation 3.60 and assuming that $|(\beta_1-\lambda_i)|$ is small so that terms containing $(\beta_1-\lambda_i)$ can also be ignored,

$$x_{i+1} \approx q_o s_i + v_{i+1}$$

$$q_o \approx \frac{x_{i+1}}{s_i} - \frac{v_{i+1}}{s_i} \qquad (3.101)$$

Since v_{i+1} is a Gaussian random variable with zero mean, this means that q_o can be estimated in an iterative process as follows,

$$q_o^{'(i+1)} = q_o^{'(i)} + c_2(\frac{x_{i+1}}{s_i} - q_o^{'(i)}) \qquad (3.102)$$

where $q_o^{'(i)}$ is the estimate of the quantity q_o at the i^{th} iteration, and c_2 is a positive real constant.

Substituting $q_o^{'(i)}$ for q_o in equation 3.100 the complete algorithm now becomes

$$\lambda_{i+1} = \lambda_i + c_1 \frac{x_i}{s_i q_o^{'(i)}} \qquad (3.103)$$

and

$$q_o^{'(i)} = q_o^{'(i-1)} + c_2(\frac{x_{i+1}}{s_i} - q_o^{'(i-1)}) \qquad (3.104)$$

where c_1 and c_2 are positive real constants in the range 0 to 1, and $q_o^{'(0)} = \lambda_o = 0$ for starting-up. Note that in the feasibility study based on the synchronous serial binary data-transmission, c_1 was set equal to c_2, for convenience, in the above algorithm.

Results of computer-simulation tests for Channel 1 and Channel 3 are given in Figures 3.44-3.135 which show the variation of θ_i (equation 3.73) and E_i (equation 3.74) with the number of iteration i. Since the estimate of β_1 is updated once every sampling interval by the iterative root-finding algorithm, therefore, an iteration, i, is equivalent to the time of one sampling interval. The mean-square errors, θ_i and E_i, were obtained by taking the average of the results of 100 independent tests where it was assumed that a 40-tap filter was used and that the delay introduced by the detector itself is zero. The effects of three different levels of additive white Gaussian noise on the performance of the algorithm were tested and these corresponded to signal to noise ratios of ∞, 40 dB and 30 dB. In each case, different combinations of c_1 (0.1, 0.25, 0.5, 0.75, 1.0) and c_2 (0.1, 0.25, 0.5, 0.75, 1.0) were tested and the results were obtained over 100 iterations so as to give time for the algorithm to converge.

A detailed study is carried out using Channel 1 (whose z-transform has 3 roots outside the unit circle) in order to investigate the effect of c_1 and c_2 on the speed and accuracy of the algorithm and to determine the best combination for the operation. The results of the study are presented in Figures 3.44-3.115 which show the variations of θ_i and E_i with i for CH2OR0, CH2OR1 and CH2OR2 (which denote Channel 1 with no root, 1 root and 2 roots processed, respectively) when c_1 is held constant at either 1.0, 0.75, 0.50 or 0.25 while c_2 is varied from 1.0 to 0.25 in steps of 0.25, and vice versa. It is clear from the results that in the absence of noise, the algorithm performs extremely well, giving fast and accurate estimates of the roots. Error measurements of better than -100 dB are achieved in both θ_i and E_i in all the cases tested. The number of iterations taken to achieve this varies from 25 to 95, depending on the values of c_1 and c_2. When c_1 and c_2 are set equal to 0.5 or larger, the number of iterations taken for the algorithm to give values of better than -100 dB in both θ_i and E_i is under 50. In some cases such as Figure 3.46, however, the algorithm sometimes locates some other roots outside the unit circle rather than the one (Root 1) used in θ_i and E_i and hence giving a result showing a false sense of poor performance. Furthermore, it was found that, for a chosen set of c_1 and c_2, the order in which the roots outside the unit circle are located is not fixed and is dependent to some extent on the sequence of received samples $\{r_i\}$ (equation 3.4). Whilst the fastest convergence is achieved when both c_1 and c_2 are large, and in general, $c_1 = c_2 = 1$ being the best, the parameter c_2 seems to be the more influential factor in that for a fixed c_2, the difference in the number of iterations taken by the algorithm to reach convergence when c_1 is varied from 1.0 to 0.25 is smaller than the difference in the number of iterations taken in the case where c_1 is held constant and c_2 is varied from 1.0 to 0.25. At θ_i = -100 dB, a typical value for the former case is 10 iterations (Figure 3.80) and 50 iterations for the latter (Figure 3.68). This is particularly so when more roots outside the unit have been removed and replaced by their complex conjugate-reciprocals. However, the fact remains that for a fast convergence of the algorithm c_2 has to be set fairly large.

In the presence of additive white Gaussian noise, the algorithm yields more accurate results when there are less roots remaining outside the unit circle. Again the parameter c_2 seems to be the more critical factor in the speed and accuracy of the root-finding process. For a more accurate estimate of β_1, the smaller value of c_2 is desirable but this has the penalty of slowing down the rate of convergence. However, at the signal to noise ratios of 40 dB and 30 dB, once a value of c_2 is chosen, the difference in speed is negligible for the different c_1 tested (1.0, 0.75, 0.50 and 0.25), as long as c_2 is not different from c_1 by a large amount. This is clearly illustrated in Figures 3.84-3.91. The figures also show that an improvement of about 10 dB in the estimate of β_1 can be obtained by using $c_2 = 0.25$ instead of 1.0 although there is a degradation of about 10 iterations in the speed of convergence due to the smaller step size. The results suggest that in the presence of additive white Gaussian noise, no great advantage is gained by setting

c_2 differently to c_1. However, the value of c_2 has to be chosen as a compromise between speed and accuracy of the root-finding process. In the case of CH20R0 (Channel 1 with none of its roots outside the unit circle processed), the algorithm performs badly compared with the other cases where one or two roots of Channel 1 outside the unit circle are assumed to have been processed perfectly, especially when c_2 and c_1 are $\geqslant 0.5$. Figures 3.60-3.63 show that, even at a high signal to noise ratio of 40 dB, the best value for θ_i is about -32 dB and this is obtained with c_2 set to 0.25 (Figure 3.63), and for values of $c_2 = 0.75$ or 1.0, the errors in the estimate become quite large (-25 dB to -20 dB, respectively). This is unfortunate because it is most desirable to get the first located root (or the earlier ones) as accurately as possible so that, when this is used to reduce the z-transform of the sampled impulse-response to give one which has now one less root outside the unit circle, the error in the estimate of the first root would not be carried through the process and cause the system to collapse.

Having established that once a value for c_2 is decided upon, there is no great advantage to be gained in choosing a different value for c_1 which is particularly so in the presence of additive white Gaussian noise, therefore, in the tests whose results are presented hereafter, c_1 is set equal to the value of c_2. Figures 3.116-3.135 give the results of the computer-simulation tests for Channel 3 (Figure 2.6) which represents a typical worst telephone circuit for the transmission rate of 9600 bit/s in the British public switched telephone networkand has 4 roots outside the unit circle (Figure 2.11). The roots of the z-transform of the raw channel (denoted as CH19R0 (Table 3.2)) are determined independently using the NAG subroutine C02ADF[50] which are subsequently used in the generation of CH19R1, CH19R2 and CH19R3 whose z-transforms correspond to that of Channel 3 with 3 roots, 2 roots and 1 root outside the unit circle, respectively (Table 3.2).

It can be seen that in the absence of noise the algorithm converges to the intended root (assumed in the calculation of the mean-square errors, θ_i and E_i) rapidly and accurately (Figures 3.116-3.122). In general, θ_i and E_i of better than -100 dB can be achieved within the 100 iterations used in the test. However, when there are more than one root lying outside the unit circle, the algorithm does not always land itself to the same root and because of this, the values of θ_i and E_i when averaged over 100 runs appear as though the algorithm has failed to locate any of the wanted roots (Figures 3.117 and 3.120). The results are misleading and the Figures 3.118 and 3.121, in which θ_i and E_i versus i are presented for 1 run (snapshot), show clearly that the algorithm does locate a root outside the unit circle and in this case the one assumed in the calculation of θ_i and E_i, (Roots 4 and 3 respectively). When there are more than one root outside the unit circle, both the root itself and the speed at which it is located by the algorithm depend on the particular sequence of received samples $\{r_i\}$. This is verified by the results presented in Figures 3.119 and 3.122 which show that, in a snapshot (the use of only one run in the calculation of θ_i and E_i) of θ_i and E_i versus i for $c_1 = c_2 = 0.5$, there is a difference of as many as 70 iterations between different sequences of received $\{r_i\}$ (Figure 3.119), an increase of more than double, and that different sequences may also cause a different root to be found (Figure 3.122). Figure 3.122 shows that only two out of four sequences of received $\{r_i\}$ result in the correct location of Root 3 whilst the other two cause the algorithm to latch onto Root 2 (as indicated by the two curves with large errors, θ_i and E_i).

The presence of additive white Gaussian noise in the received $\{r_i\}$, even at high signal to noise ratio of 40 dB, degrades the performance of the algorithm significantly and the degradation is more prominent when there are more roots lying outside the unit circle. Although snapshots of θ_i and E_i versus i for CH19R1 and CH19R2 (Figures 3.126, 3.128, 3.132 and 3.134) give values of better than -40 dB in places during the test period of 100 iterations, but this accuracy is not sustained and on the whole, when a large number of runs is used, the mean-square errors of θ_i and E_i is about -10 to -15 dB (Figures 3.125, 3.127, 3.131 and 3.133). These errors become less severe when there is only one root left lying outside the unit circle (Figures 3.129 and 3.135). In this case, θ_i is about -30 dB and -40 dB with $c_1 = c_2 = 0.35$ for signal to noise ratio of 30 dB and 40 dB respectively, and convergence is achieved in about 40 iterations.Although the results are not given here further studies have shown that the increase in the number of taps of the adaptive linear filter (Figure 3.43) from 40 taps to 70 and 100 taps do not improve the accuracy in the estimation of the roots appreciably which suggest that the poor performance of the algorithm is not caused by the insufficient length of the filter.

It is apparent that in the presence of additive white Gaussian noise the roots are located less accurately when there are a lot of them lying outside the unit circle. This is most undesirable because it means that initially when the z-transform of the sampled impulse-response of the channel has the largest number of roots outside the unit circle and accuracy is most needed in the estimation of these roots, they will be found least accurately.

The error in the early stage of the root-finding process will no doubt affect all the subsequent stages and, in a continuous operation of the system, may even prevent the location of the correct roots when a large error has accumulated.

3.2.2 System 3.5

A further development of the root-finding algorithm is to divide the operation into two separate parts, which operate in different manners, thus providing a further degree of freedom in optimizing the system. Its modified operation will now be described.

The receiver operates exactly as described in Section 3.1 to give the sequence of $\{x_i\}$ (equation 3.58) at the output of the one-tap feedback filter, and for $0 \leqslant i < p$, where p is a positive number, the algorithm is given by the equations below (same as System 3.4)

$$\lambda_{i+1} = \lambda_i + c \frac{x_i}{s_i q_o'(i)} \qquad (3.105)$$

and

$$q_o'(i) = q_o'(i-1) + c\left(\frac{x_{i+1}}{s_i} - q_o'(i-1)\right) \qquad (3.106)$$

where c is a real positive number in the range 0 to 1, and $\lambda_o = q_o'(0) = 0$. Now, at the p^{th} operation (p corresponds to a change-over point in the iterative root-finding process), the algorithm is modified so as to incorporate an extra "averaging" process. Thus, for $i \geqslant p$, in addition to the equations above (equations 3.105-3.106), an extra computation step is performed,

$$\lambda_{i+1}' = \left(1 - \frac{1}{K_i}\right) \lambda_i + \frac{1}{K_i} \lambda_{i+1} \qquad (3.107)$$

where K_i is a positive number which has the value

$$K_i = (i - p + 2) \qquad (3.108)$$

such that $K_i = 2,3,4,\ldots$ for $i = p, (p+1), (p+2)\ldots$ The value λ_{i+1}' is then taken to be the final estimate of β_1 at the $(i+1)^{th}$ iteration. The diagram in Figure 3.136 shows clearly the two different modes of the operation of the algorithm.

To illustrate the effect of this extra "averaging" step used by the algorithm starting from the p^{th} iteration (change-over point), for $i \geqslant p$

$$(K_p = 2) \qquad \lambda'_{p+1} = \frac{1}{2}\lambda_p + \frac{1}{2}\lambda_{p+1}$$

$$= \frac{1}{2}(\lambda_{p+1} + \lambda_p) \tag{3.109}$$

$$(K_{p+1} = 3) \qquad \lambda'_{p+2} = \frac{2}{3}\lambda_{p+1} + \frac{1}{3}\lambda_{p+2}$$

$$= \frac{1}{3}(\lambda_{p+2} + \lambda_{p+1} + \lambda_p) \tag{3.110}$$

$$(K_{p+2} = 4) \qquad \lambda'_{p+3} = \frac{3}{4}\lambda_{p+2} + \frac{1}{4}\lambda_{p+3}$$

$$= \frac{1}{4}(\lambda_{p+3} + \lambda_{p+2} + \lambda_{p+1} + \lambda_p) \tag{3.111}$$

$$(K_{p+3} = 5) \qquad \lambda'_{p+4} = \frac{4}{5}\lambda_{p+3} + \frac{1}{5}\lambda_{p+4}$$

$$= \frac{1}{5}(\lambda_{p+4} + \lambda_{p+3} + \lambda_{p+2} + \lambda_{p+1} + \lambda_p) \tag{3.112}$$

So in general,

$$\lambda'_{p+h} = \frac{1}{(h+1)} \sum_{j=0}^{h} \lambda_{p+j}$$

It can be seen that the algorithm performs an averaging process on the past estimates of β_1, starting at the p^{th} iteration, giving equal weightings to all the previous estimates $\{\lambda_i\}$ up to the $(p+h)^{th}$ if the initial value of K_i (equation 3.123) is set to 2 (i.e. $K_p = 2$).

The idea behind this two-part operation of the root-finding algorithm is to choose a suitable value of c for the first part of the iterative process, such that a fairly good estimate of β_1 can be obtained in a reasonably short time. Thus, when the algorithm is switched to the second part, a much smaller value of c can be used to reduce the fluctuations which occur when the signal is corrupted by noise. The object of this test is to investigate the merits of this system and, if possible, to try and find the best compromise between the speed and accuracy of the operation by using suitably chosen values of c and of the change-over points.

So far, it has been shown (Section 3.2) that the accuracy and the speed in locating β_1 by the root-finding algorithm is more inferior when there are more roots lying outside the unit circle in the presence of additive white Gaussian noise. Therefore, to test the performance of this system, the raw sampled impulse-responses of Channels 1 and 3 (CH20R0 and CH19R0) are initially chosen, whose z-transforms have 3 and 4 roots (zeros) outside the unit circle in the z-plane, respectively. The algorithm was tested for two step-sizes, c = 0.5 and c = 1, and change-over points of 10, 20, 30 and 40 iterations. The results of computer-simulation tests are presented in Figures 3.137-3.148 which show the variations of mean-square errors θ_i and E_i with i averaged over 100 runs for signal to noise ratios of ∞, 40 dB and 30 dB.

When there is no noise in the system, the algorithm can be seen to converge very quickly to the value of β_1 during the first part of the algorithm, and as the second part of the algorithm is switched in, the curves of θ_i and E_i versus i level off from the change-over point, as expected (Figures 3.137, 3.138, 3.143 and 3.144). However, when there is additive white Gaussian noise in the signal, the algorithm performs only marginally better than that of System 3.4. The latter is identical to this system without the use of the second part of the averaging operation. Furthermore, the improvement in performance depends on the change-over period (Figure 3.139) and at best it is only about 10 dB and it occurs after a long time (100 iterations) and at a signal to noise ratio of 40 dB. This can be seen from comparing Figure 3.48 with Figures 3.139 and 3.140. At signal to noise ratios of 30 dB, the difference in performance between this system and System 3.4 is negligible. However, it should be noted that the variations in θ_i and E_i versus i is much smaller during the second part of the algorithm, which shows a smoothing of the noise. More importantly, the results show that because the first part of the algorithm is not capable of locating β_1 reasonably accurately in the presence of noise and because the second part of the algorithm is designed to smooth out the noise only, the algorithm cannot eventually yield the necessary accuracy in the estimation of β_1. Further tests were carried out using different values of K_p (equation 3.123), i.e. the starting value of K_i at the change-over point, and no noticeable improvement was observed.

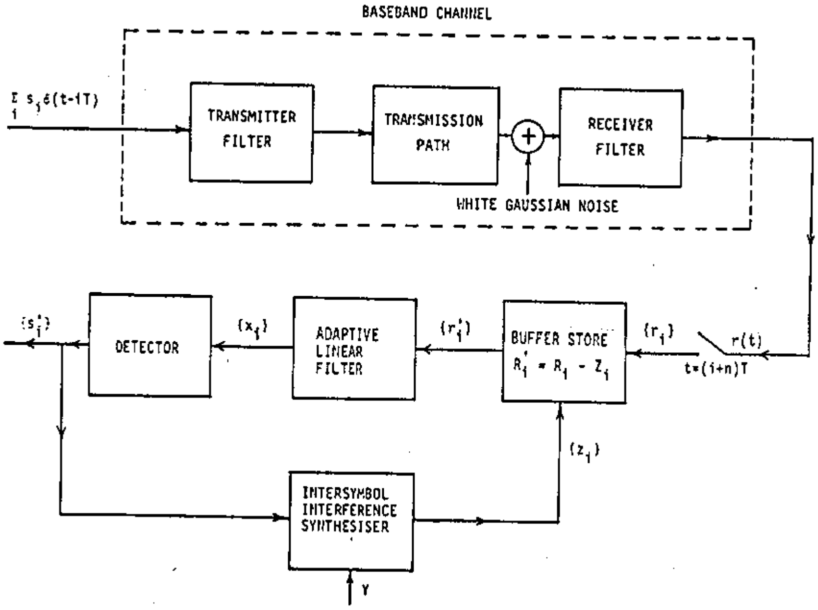


FIGURE 3.43 Model of the Synchronous Serial Binary Data-Transmission System

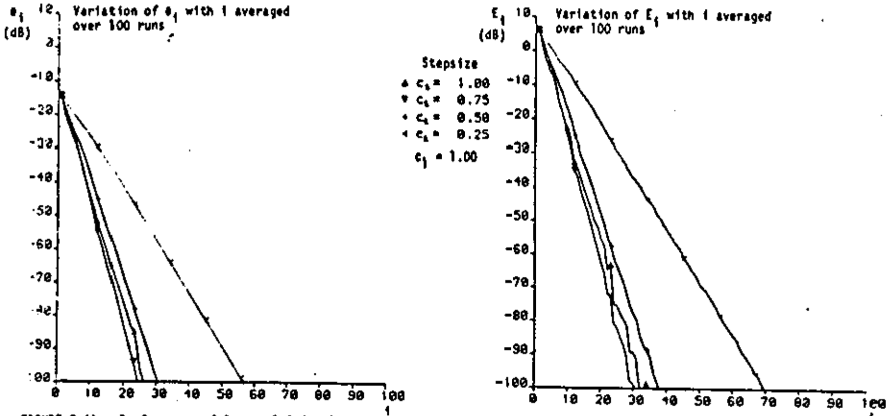


FIGURE 3.44: Performance of System 3.4 for CH2ORO, SNR = ∞ dB

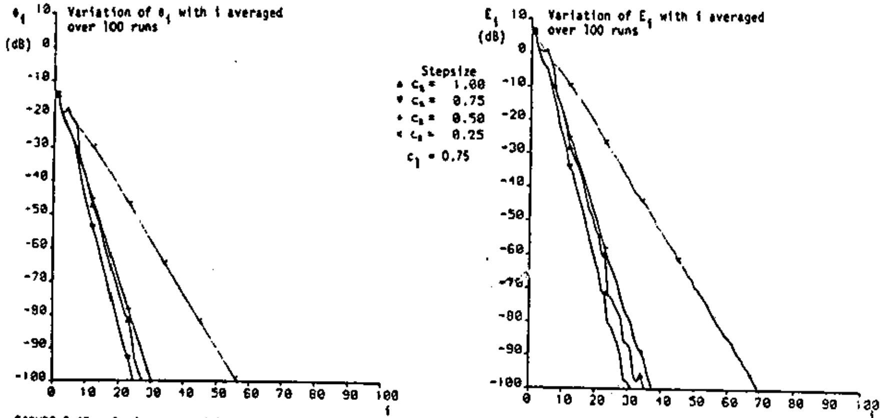


FIGURE 3.45: Performance of System 3.4 for CH2ORO, SNR = ∞ dB

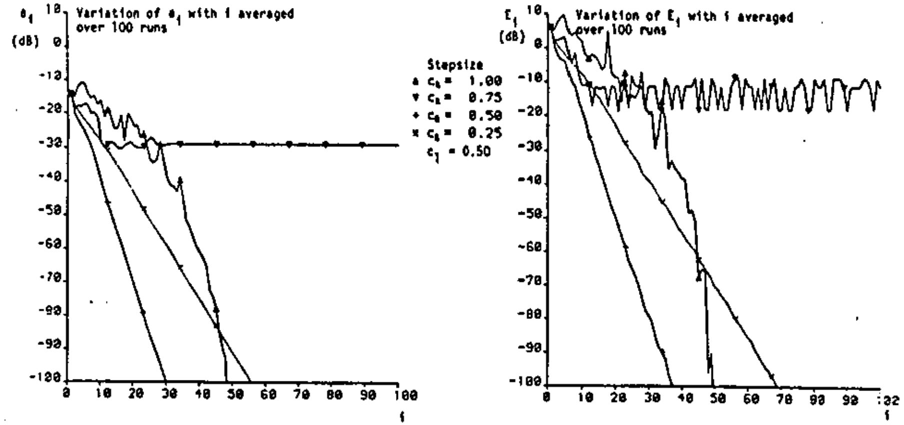


FIGURE 3.46: Performance of System 3.4 for CH2ORO, SNR = ∞ dB

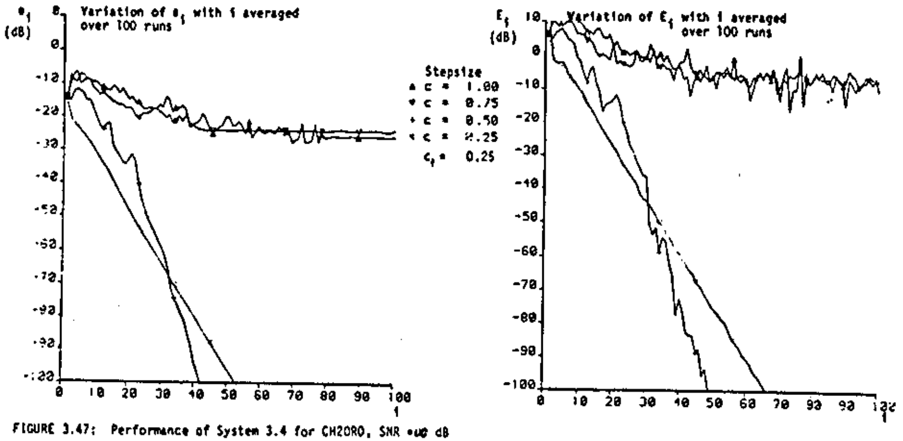


FIGURE 3.47: Performance of System 3.4 for CH2ORO, SNR = ∞ dB

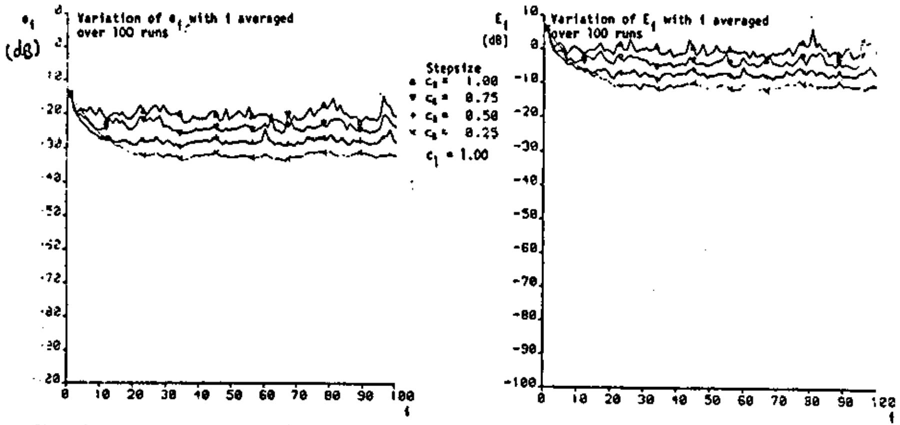


FIGURE 3.48: Performance of System 3.4 for CH2ORO, SNR = 40 dB

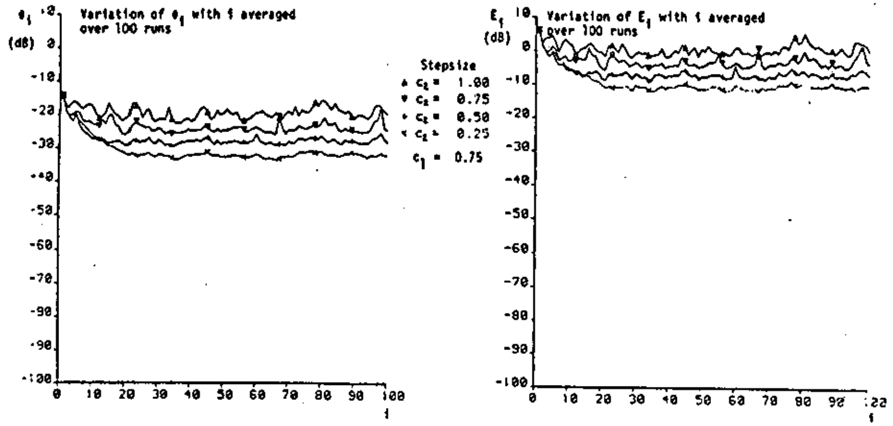


FIGURE 3.49: Performance of System 3.4 for CH2ORO, SNR = 40 dB

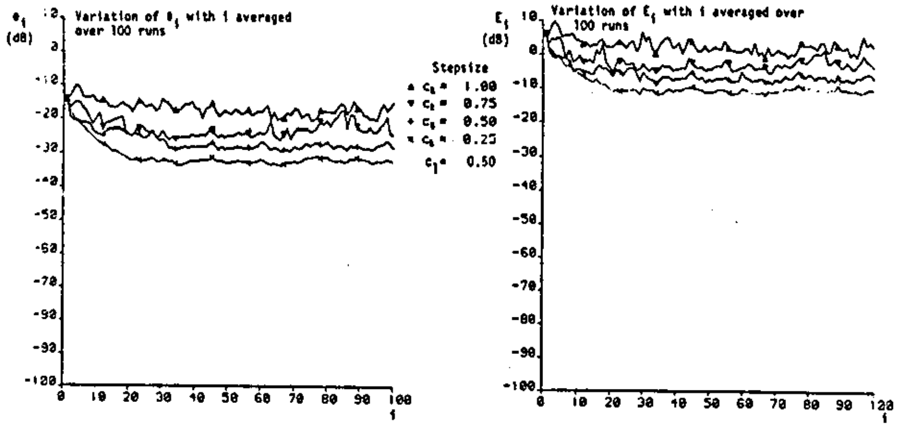


FIGURE 3.50: Performance of System 3.4 for CH2ORO, SNR = 40 dB

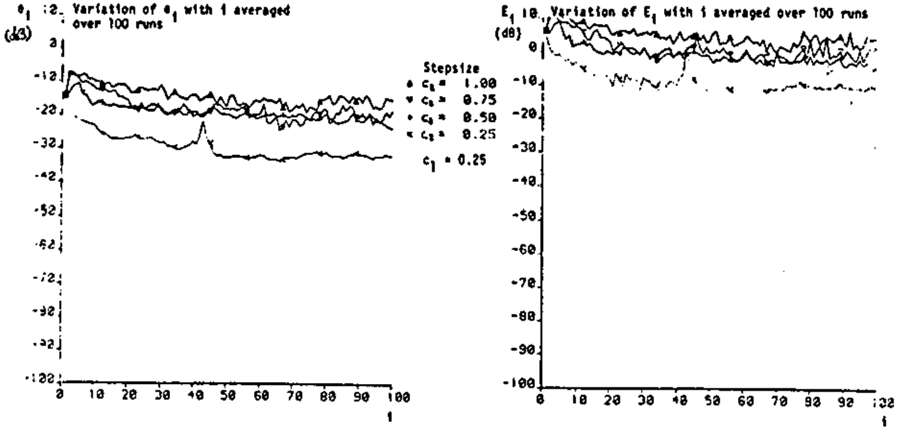


FIGURE 3.51: Performance of System 3.4 for CH2ORO, SNR = 40 dB

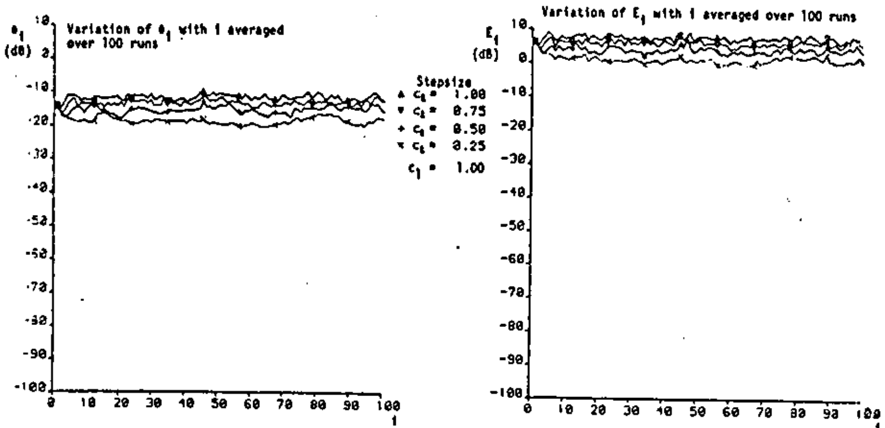


FIGURE 3.52: Performance of System 3.4 for CH2ORO, SNR = 30 dB

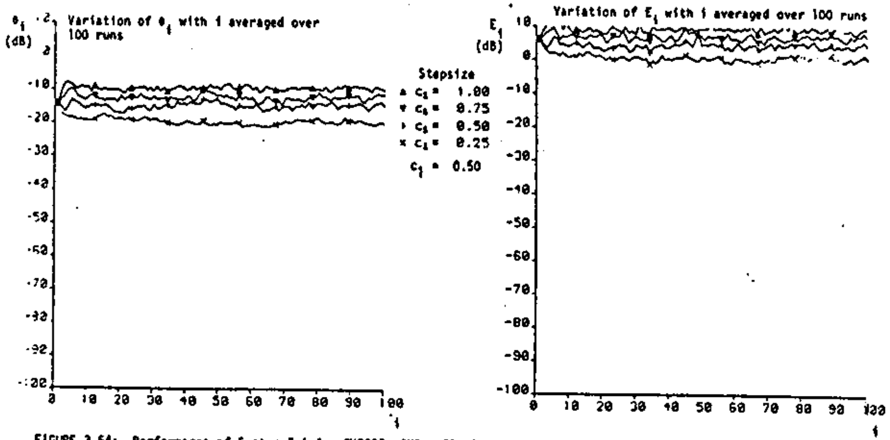


FIGURE 3.54: Performance of System 3.4 for CH2ORO, SNR = 30 dB

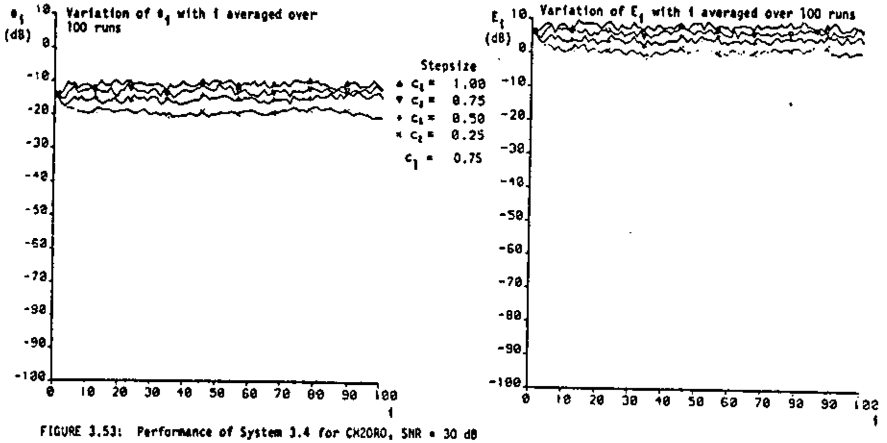


FIGURE 3.53: Performance of System 3.4 for CH2ORO, SNR = 30 dB

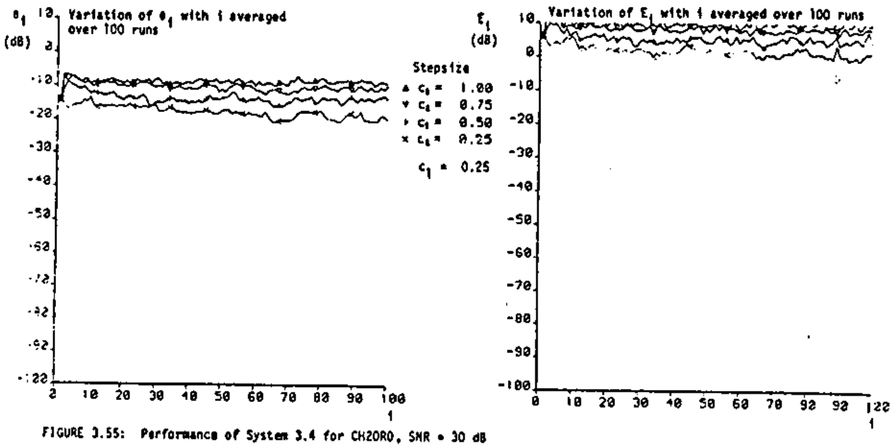


FIGURE 3.55: Performance of System 3.4 for CH2ORO, SNR = 30 dB

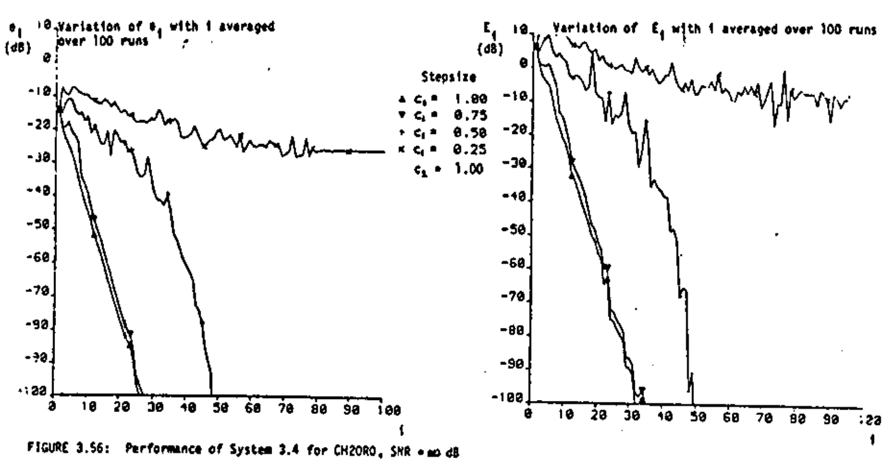


FIGURE 3.56: Performance of System 3.4 for CH2ORO, SNR = ∞ dB

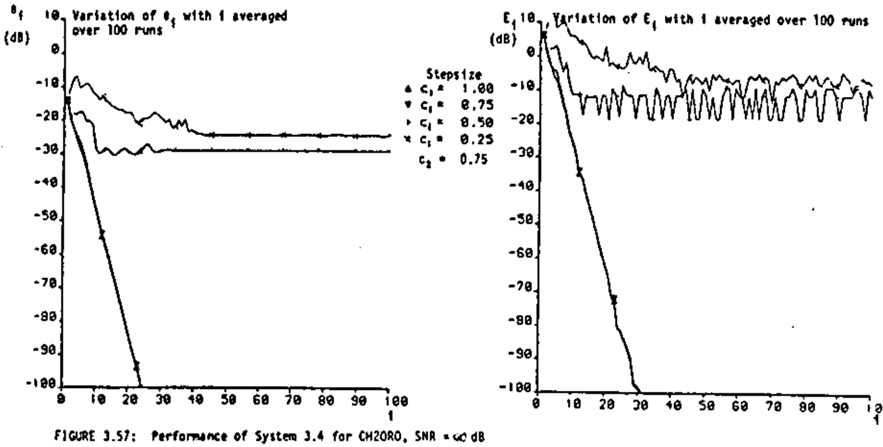


FIGURE 3.57: Performance of System 3.4 for CH2ORO, SNR = ∞ dB

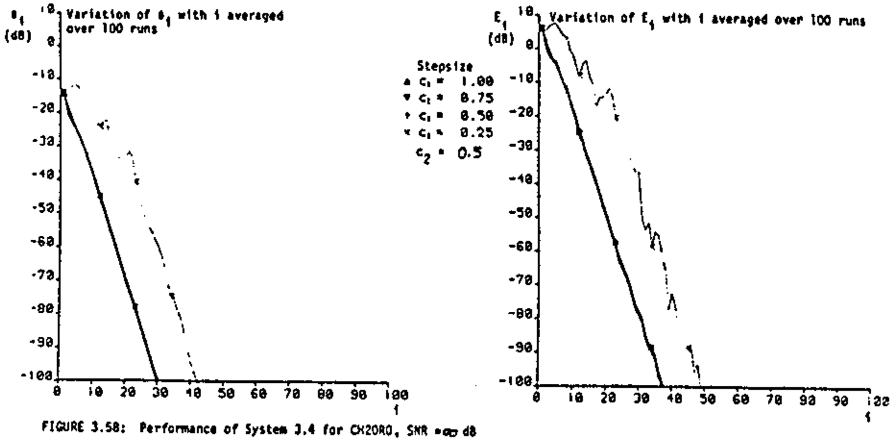


FIGURE 3.58: Performance of System 3.4 for CH2ORO, SNR = ∞ dB

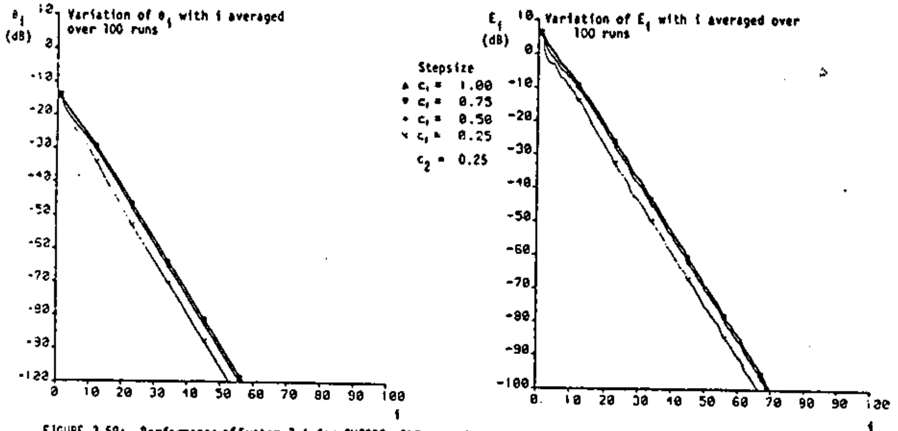


FIGURE 3.59: Performance of System 3.4 for CH2ORO, SNR = ∞ dB

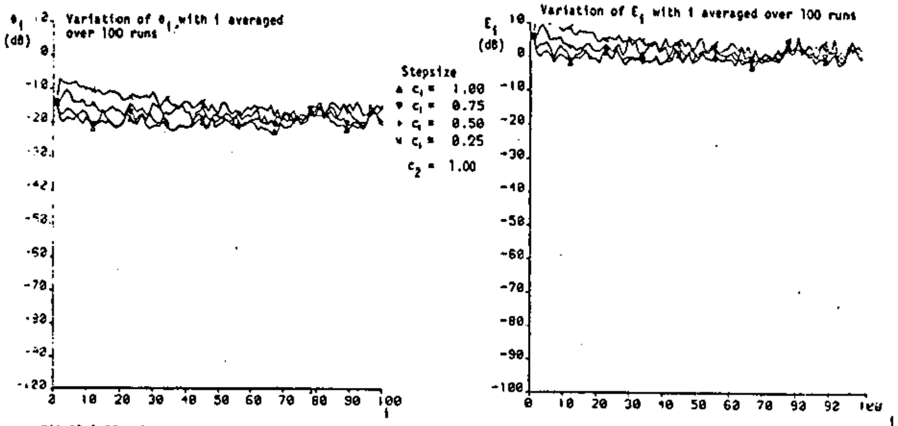


FIGURE 3.60: Performance of System 3.4 for CH2ORO, SNR = 40 dB

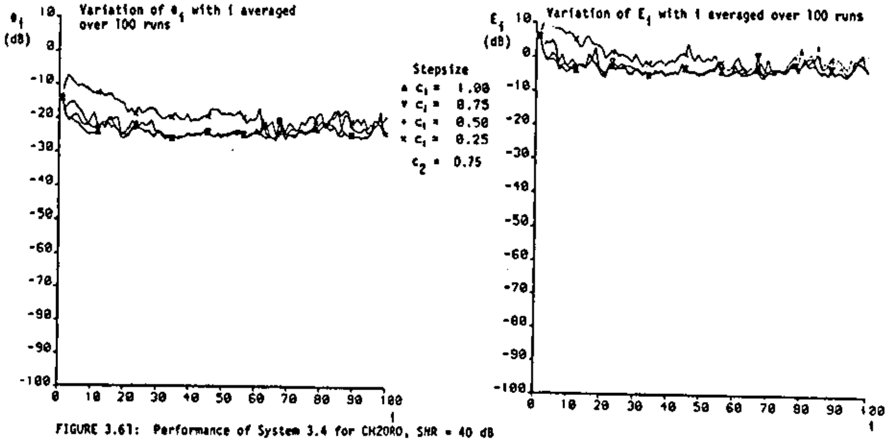


FIGURE 3.61: Performance of System 3.4 for CH2ORO, SNR = 40 dB

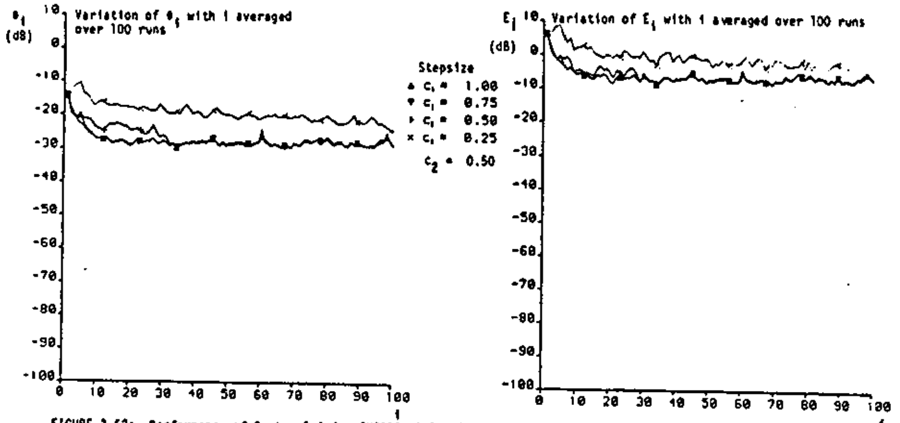


FIGURE 3.62: Performance of System 3.4 for CH2ORO, SNR = 40 dB

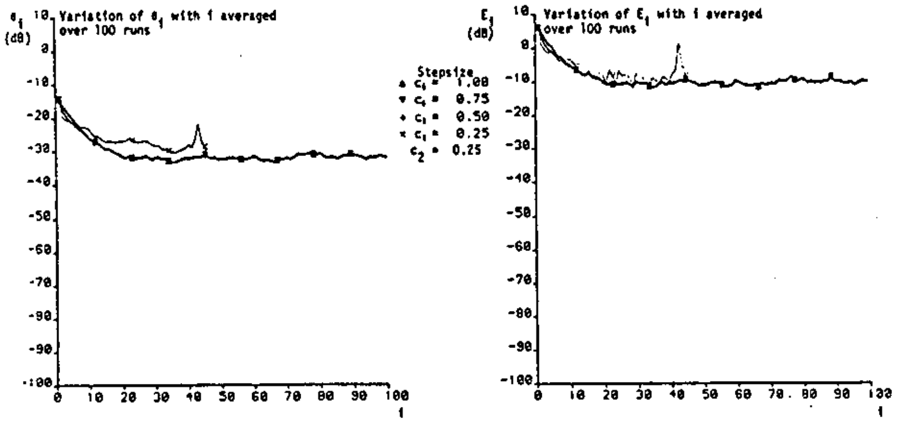


FIGURE 3.63: Performance of System 3.4 for CH2ORO, SNR = 40 dB

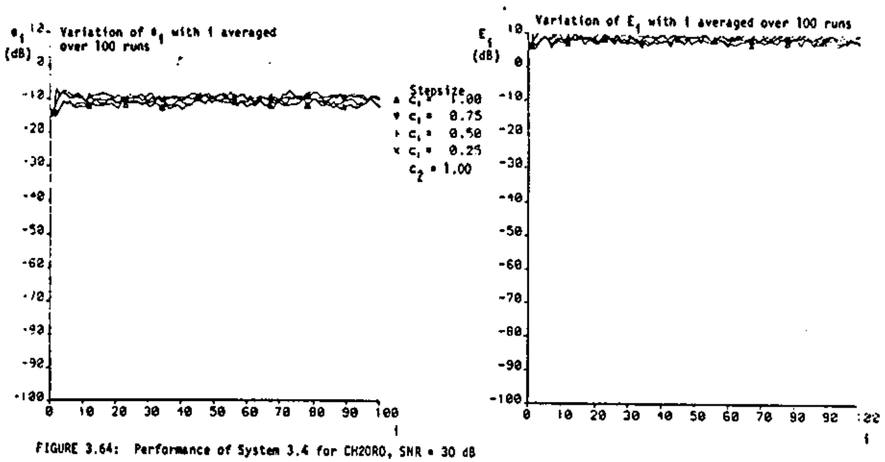


FIGURE 3.64: Performance of System 3.4 for CH2ORO, SNR = 30 dB

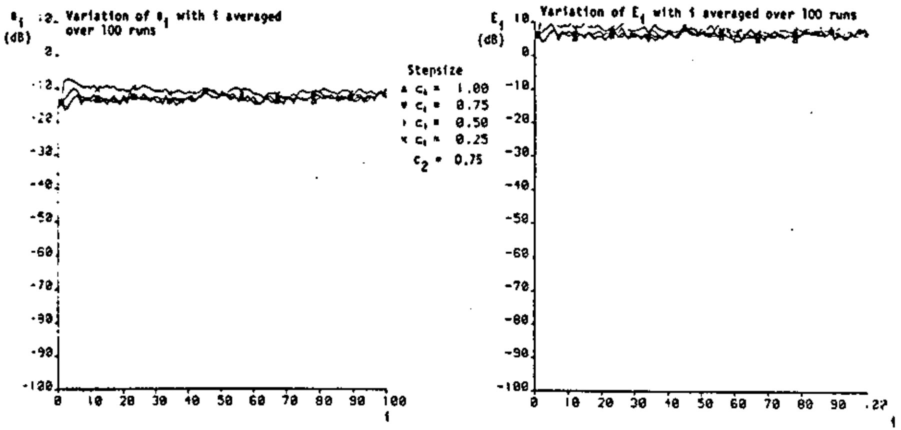


FIGURE 3.65: Performance of System 3.4 for CH2ORO, SNR = 30 dB

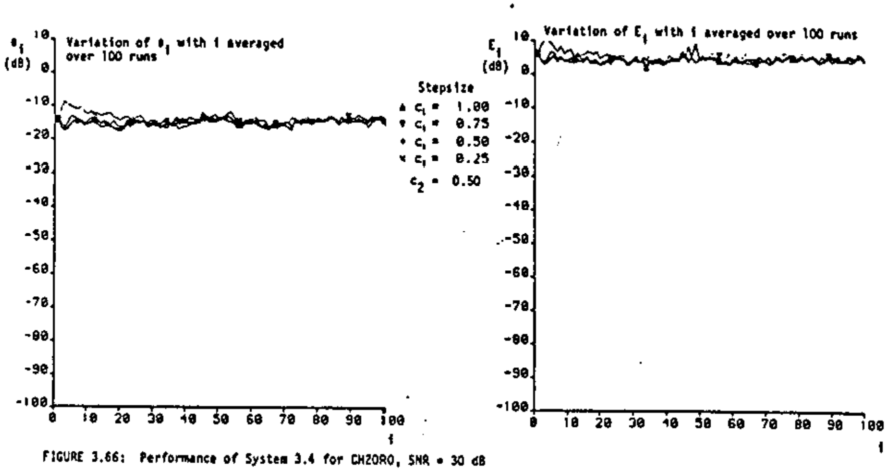


FIGURE 3.66: Performance of System 3.4 for CH2ORO, SNR = 30 dB

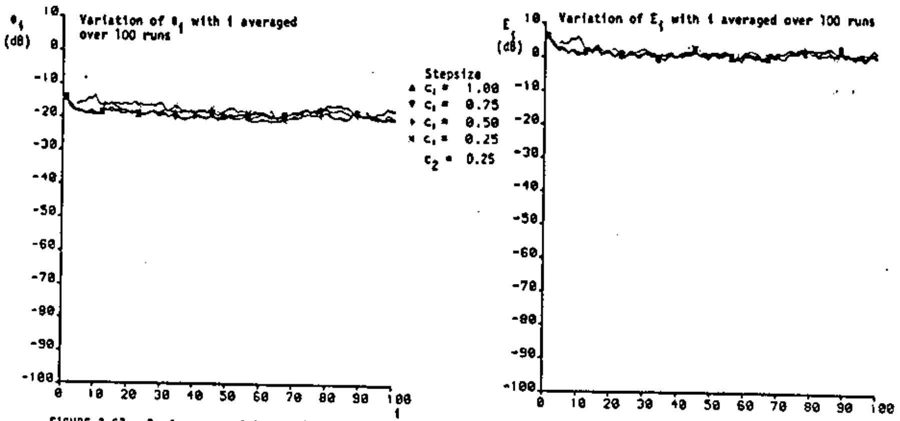


FIGURE 3.67: Performance of System 3.4 for CH2ORO, SNR = 30 dB

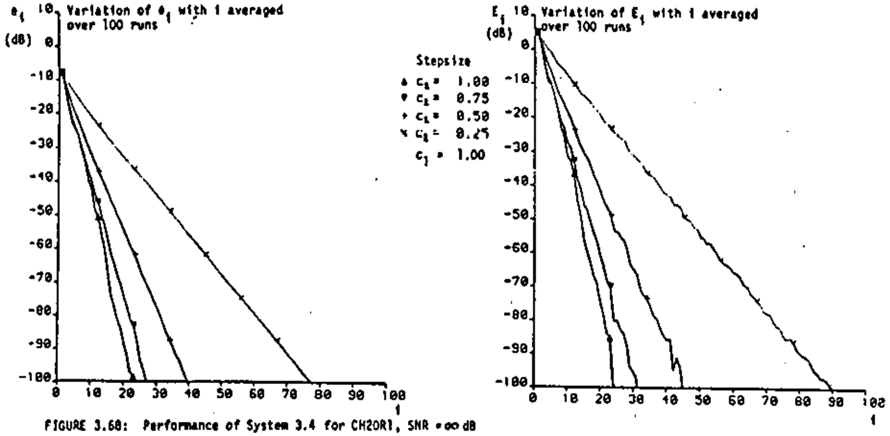


FIGURE 3.68: Performance of System 3.4 for CH2OR1, SNR = ∞ dB

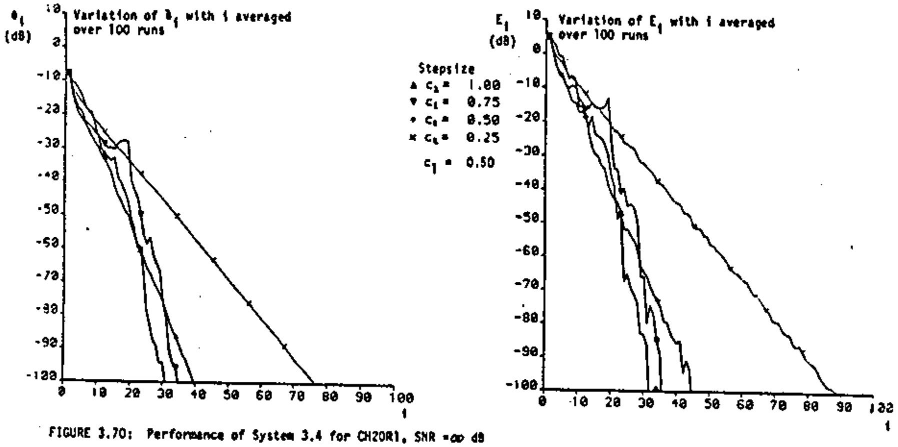


FIGURE 3.70: Performance of System 3.4 for CH2OR1, SNR = ∞ dB

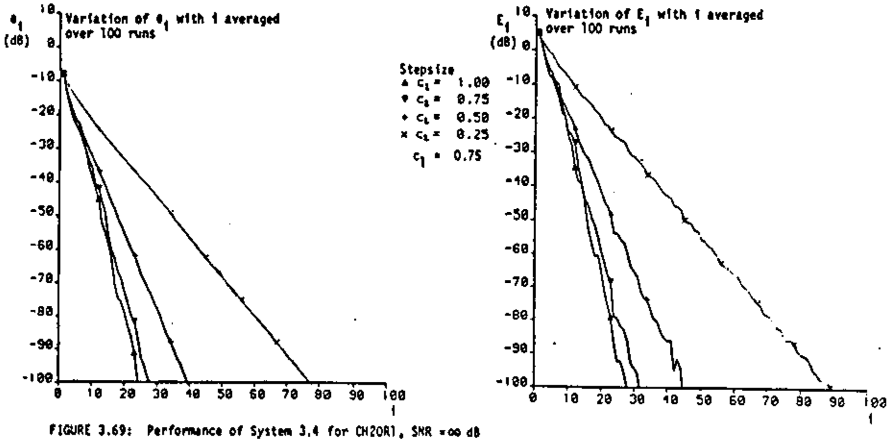


FIGURE 3.69: Performance of System 3.4 for CH2OR1, SNR = ∞ dB

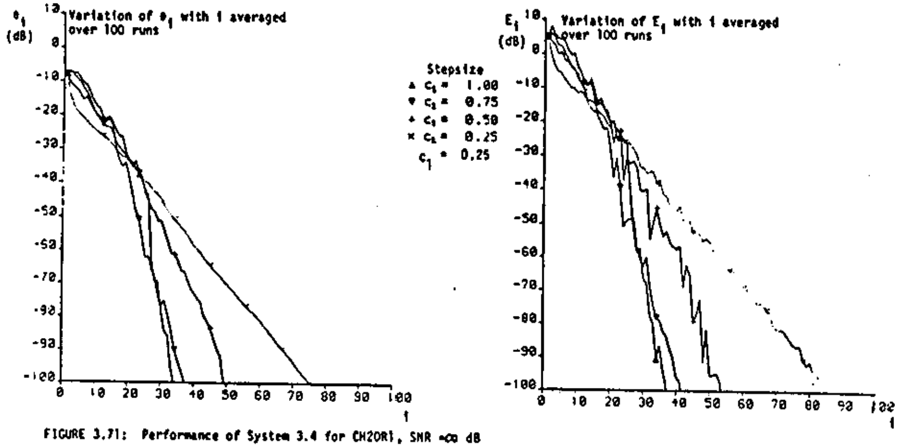


FIGURE 3.71: Performance of System 3.4 for CH2OR1, SNR = ∞ dB

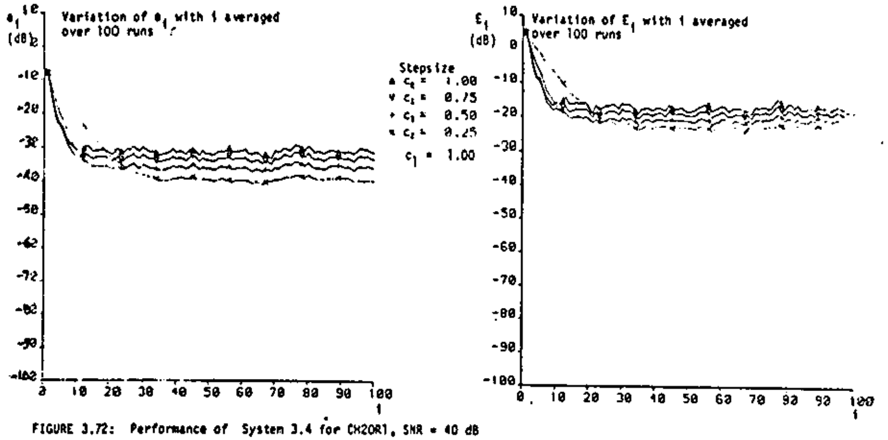


FIGURE 3.72: Performance of System 3.4 for CH2OR1, SNR = 40 dB

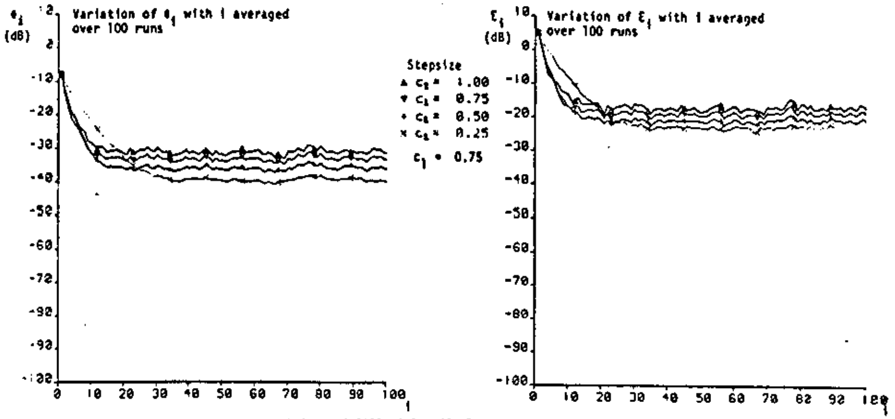


FIGURE 3.73: Performance of System 3.4 for CH2OR1, SNR = 40 dB

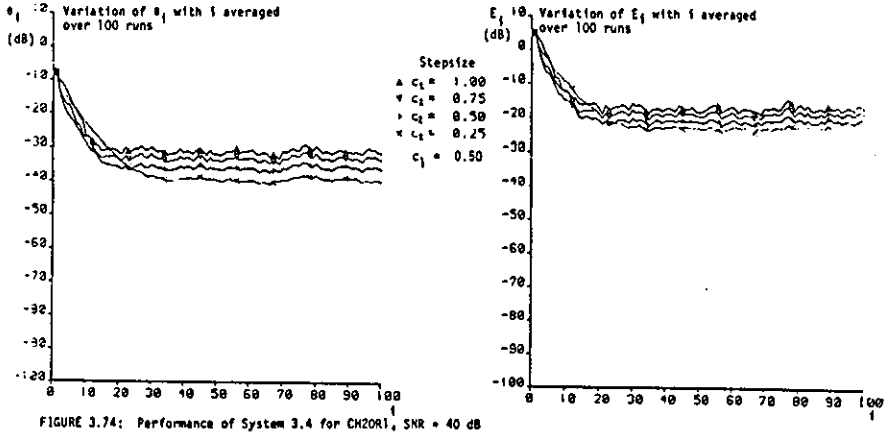


FIGURE 3.74: Performance of System 3.4 for CH2OR1, SNR = 40 dB

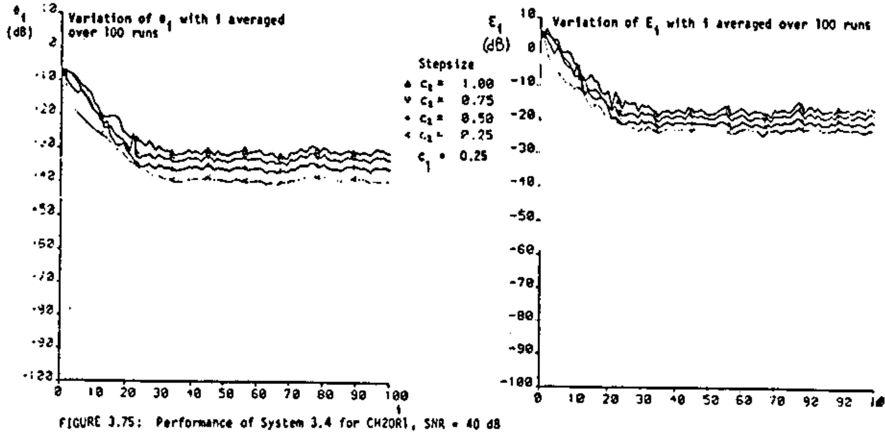


FIGURE 3.75: Performance of System 3.4 for CH2OR1, SNR = 40 dB

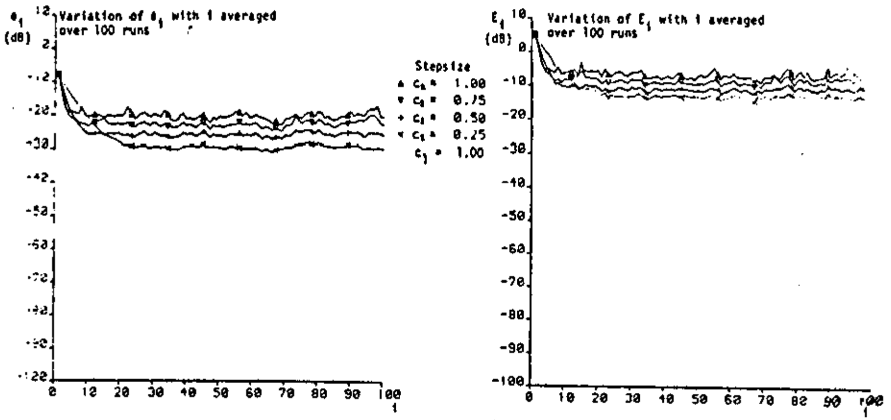


FIGURE 3.76: Performance of System 3.4 for CH2OR1, SNR = 30 dB

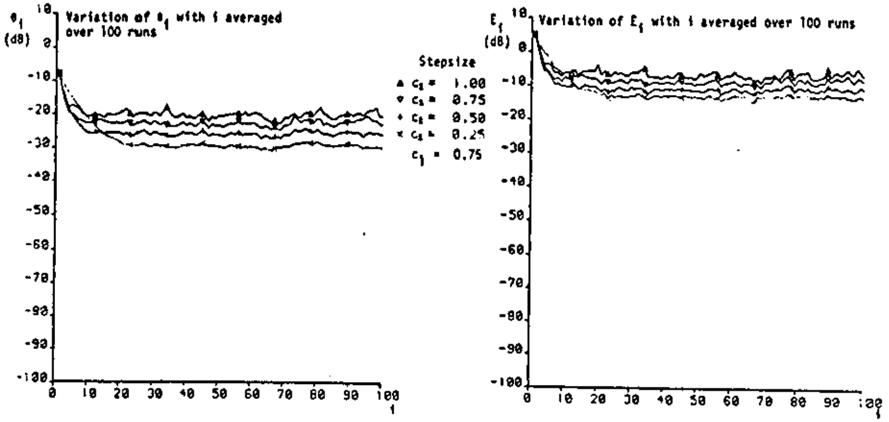


FIGURE 3.77: Performance of System 3.4 for CH2OR1, SNR = 30 dB

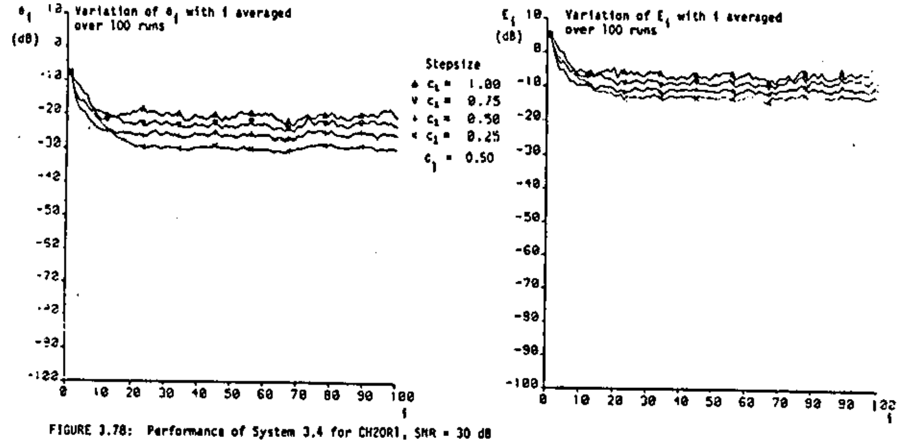


FIGURE 3.78: Performance of System 3.4 for CH2OR1, SNR = 30 dB

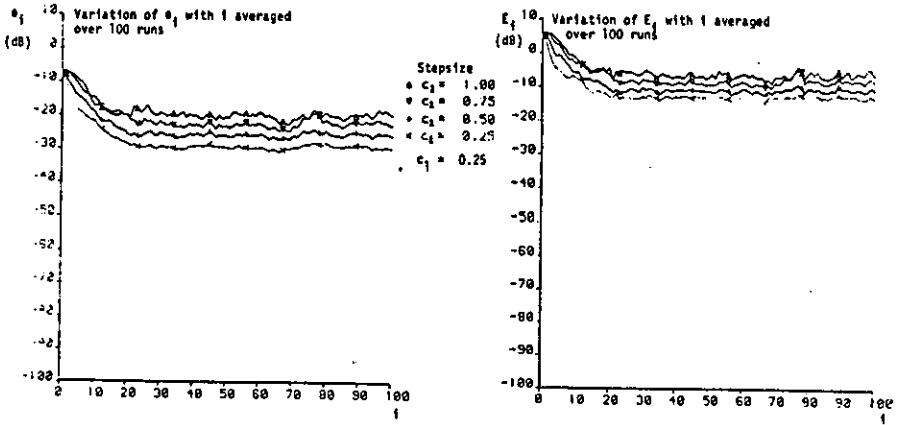


FIGURE 3.79: Performance of System 3.4 for CH2OR1, SNR = 30 dB

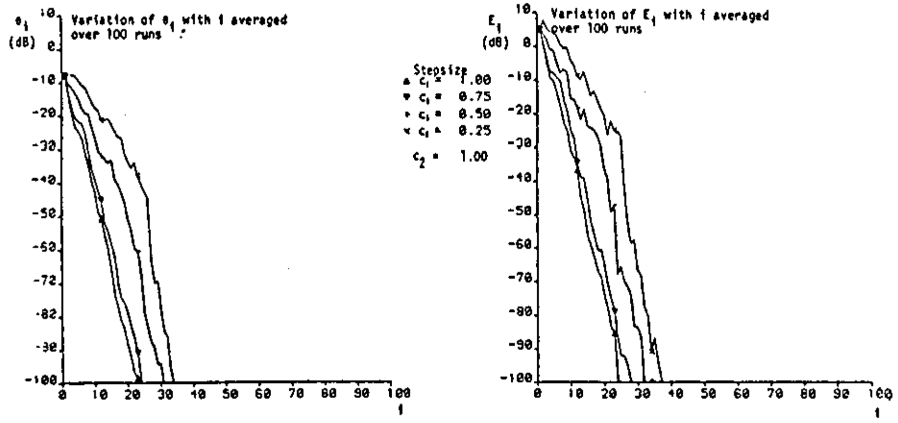


FIGURE 3.80: Performance of System 3.4 for CH2OR1, SNR =∞dB

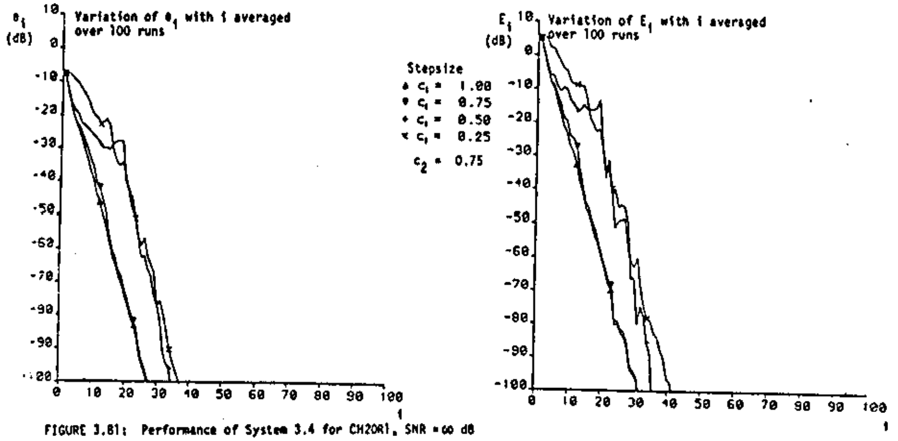


FIGURE 3.81: Performance of System 3.4 for CH2OR1, SNR =∞ dB

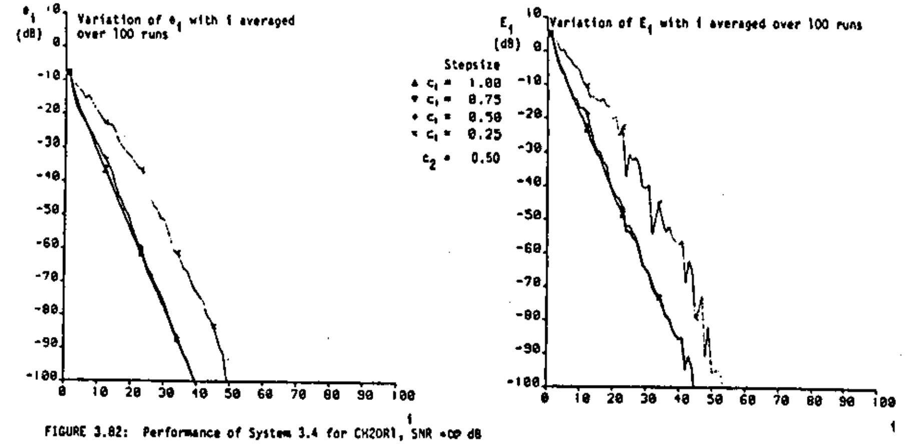


FIGURE 3.82: Performance of System 3.4 for CH2OR1, SNR =∞ dB

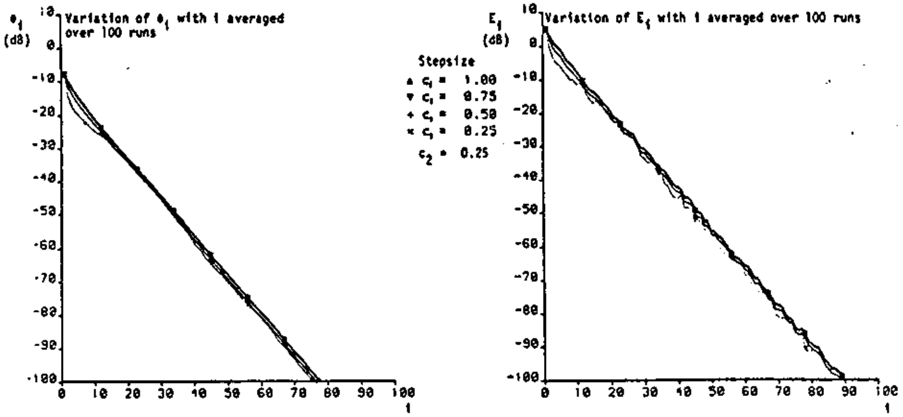


FIGURE 3.83: Performance of System 3.4 for CH2OR1, SNR =∞ dB

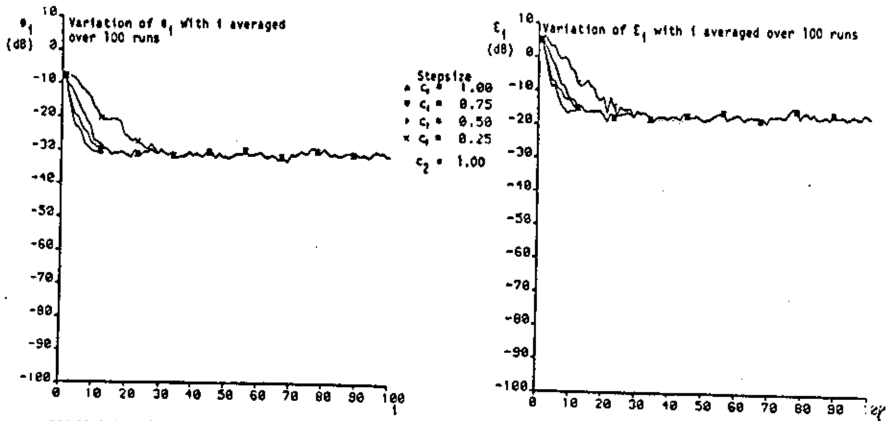


FIGURE 3.84: Performance of System 3.4 for CH2OR1, SNR = 40 dB

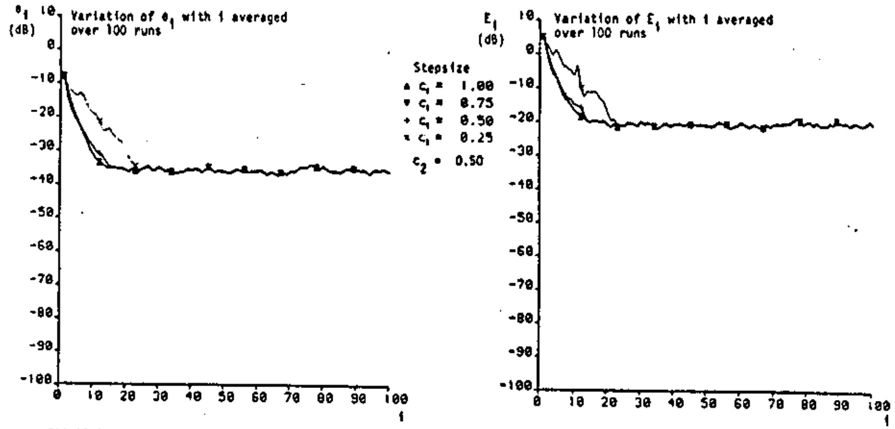


FIGURE 3.86: Performance of System 3.4 for CH2OR1, SNR = 40 dB

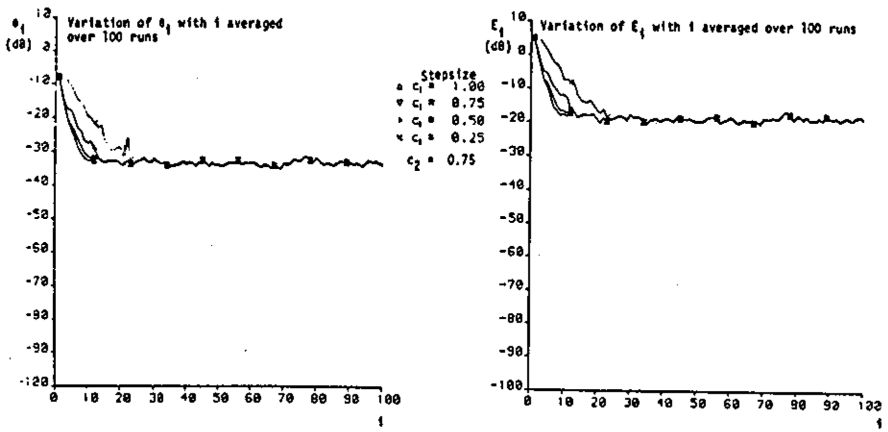


FIGURE 3.85: Performance of System 3.4 for CH2OR1, SNR = 30 dB

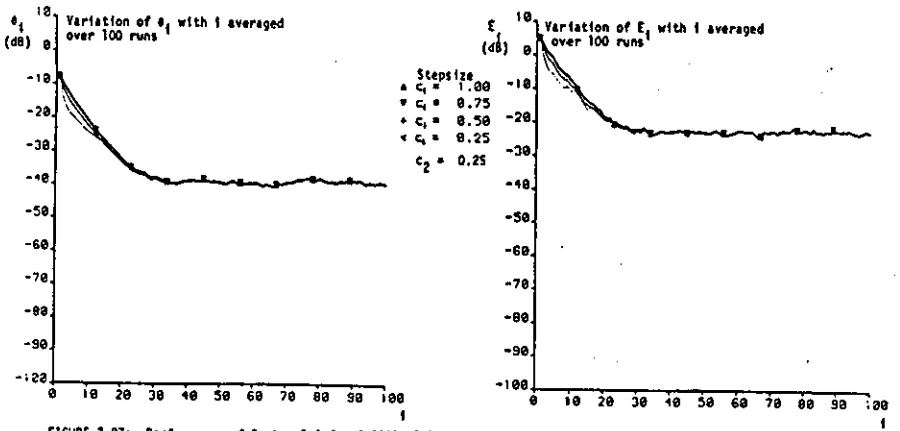


FIGURE 3.87: Performance of System 3.4 for CH2OR1, SNR = 30 dB

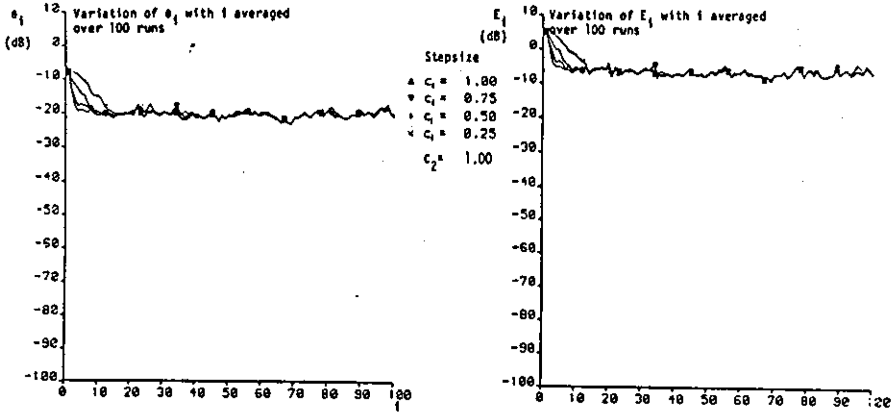


FIGURE 3.88: Performance of System 3.4 for CH2OR1, SNR = 30 dB

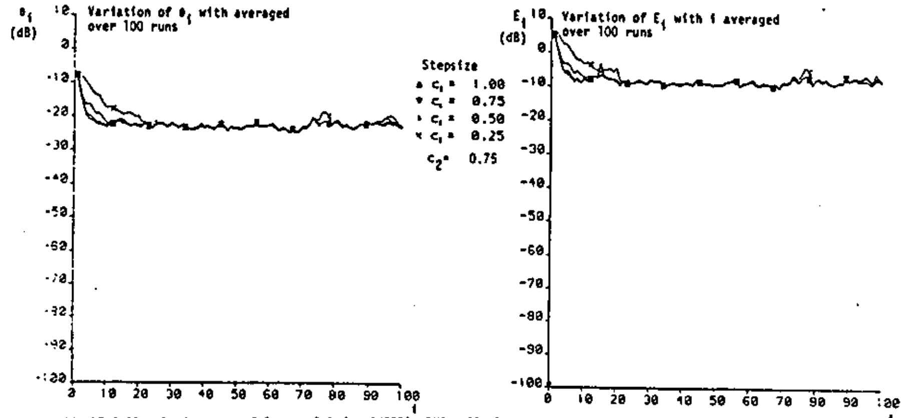


FIGURE 3.89: Performance of System 3.4 for CH2OR1, SNR = 30 dB

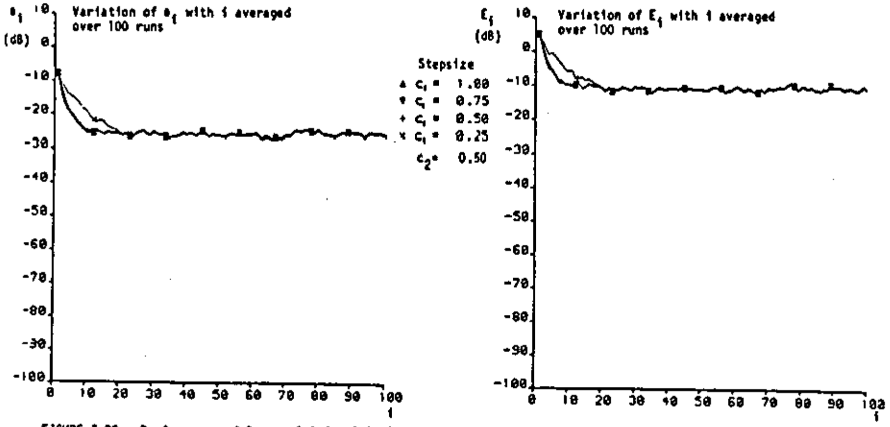


FIGURE 3.90: Performance of System 3.4 for CH2OR1, SNR = 30 dB

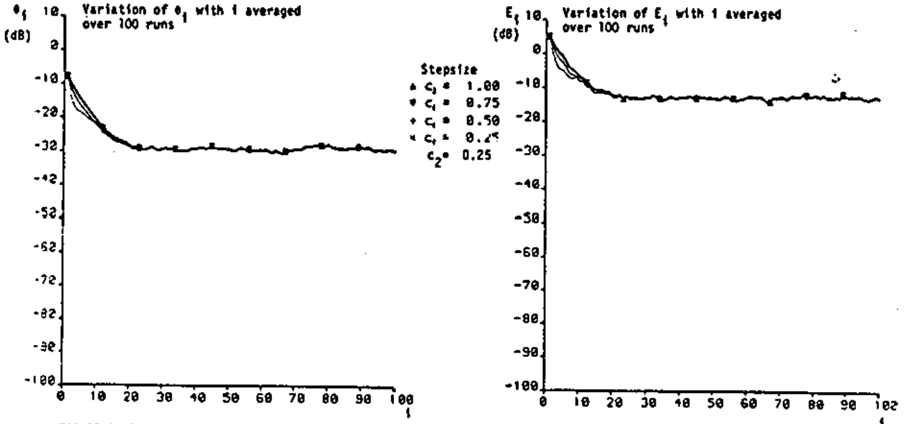


FIGURE 3.91: Performance of System 3.4 for CH2OR1, SNR = 30 dB

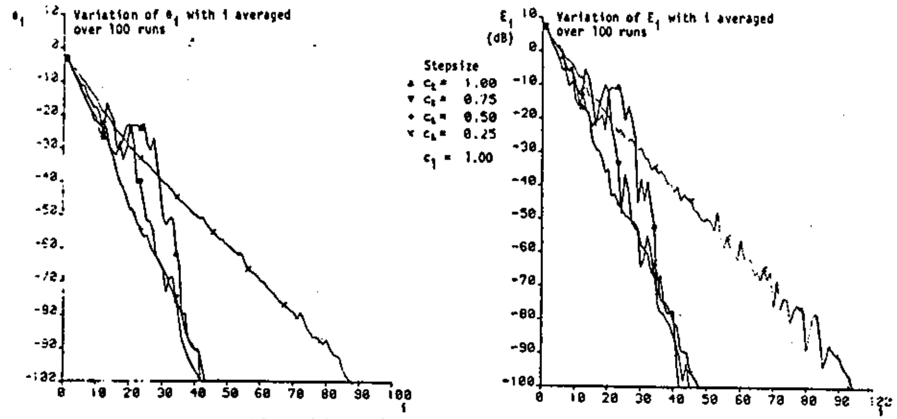


FIGURE 3.92: Performance of System 3.4 for CH2OR2, SNR =∞ dB

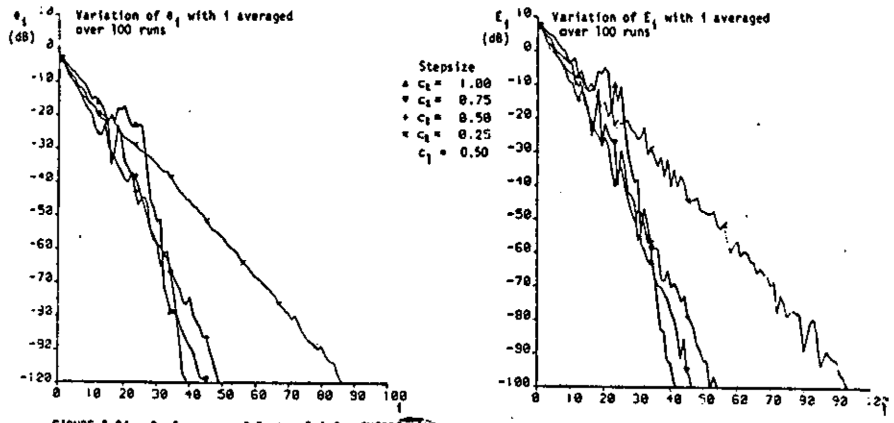


FIGURE 3.94: Performance of System 3.4 for CH2OR2, SNR =∞ dB

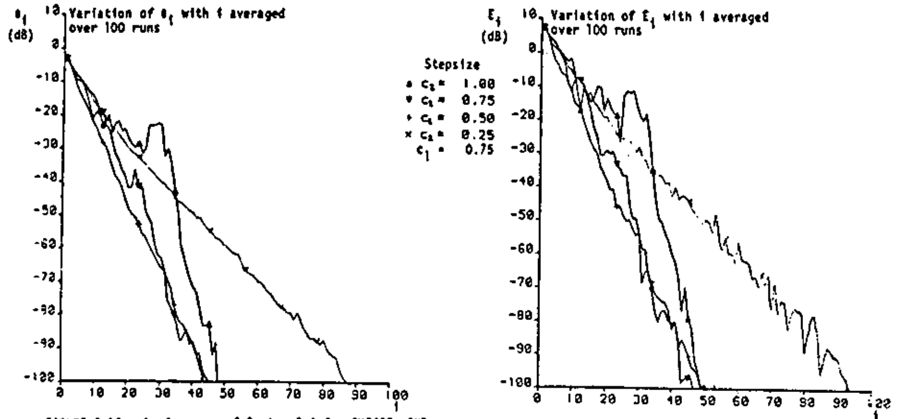


FIGURE 3.93: Performance of System 3.4 for CH2OR2, SNR =∞ dB

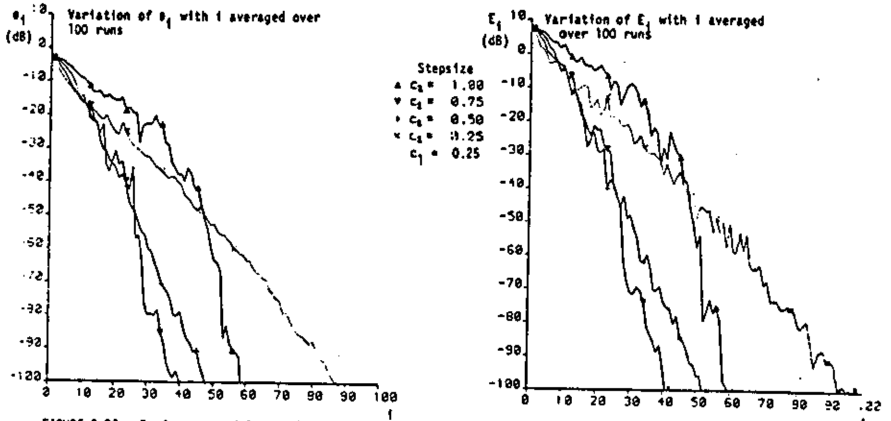


FIGURE 3.95: Performance of System 3.4 for CH2OR2, SNR =∞ dB

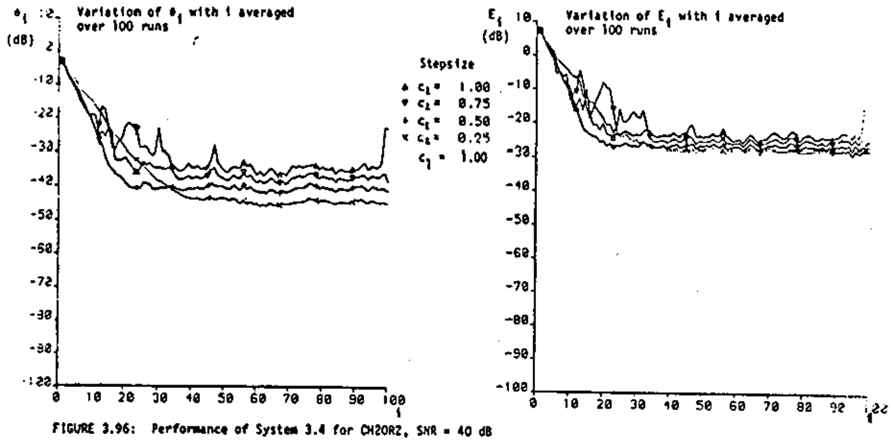


FIGURE 3.96: Performance of System 3.4 for CH2OR2, SNR = 40 dB

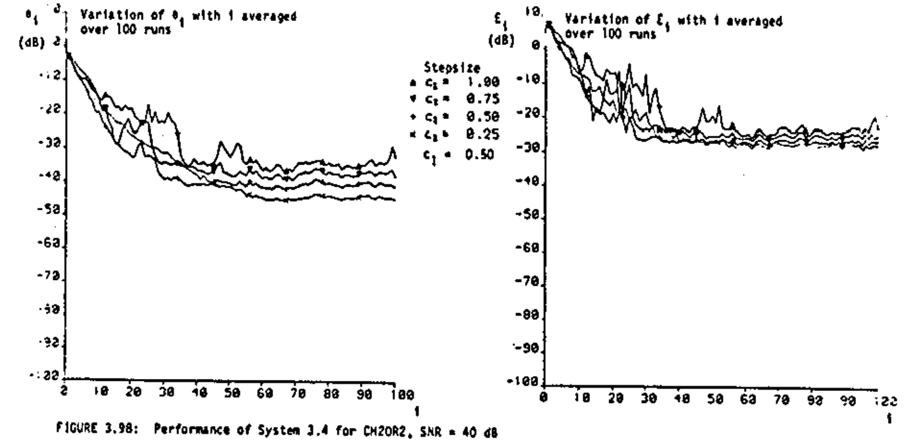


FIGURE 3.98: Performance of System 3.4 for CH2OR2, SNR = 40 dB

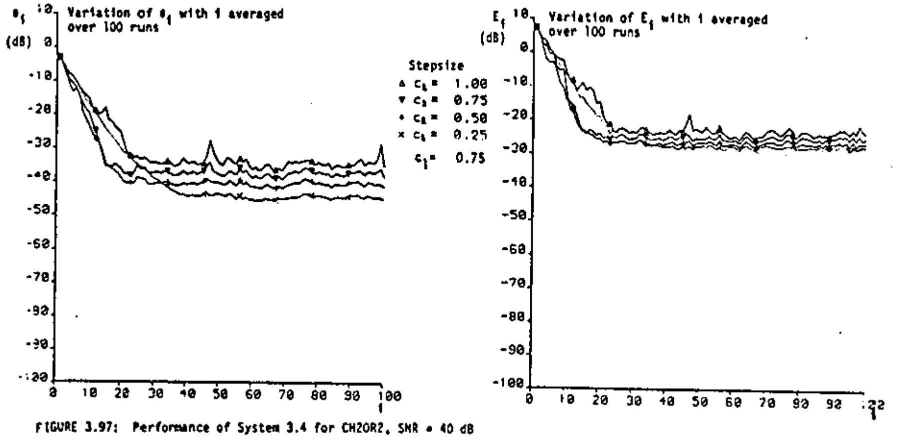


FIGURE 3.97: Performance of System 3.4 for CH2OR2, SNR = 40 dB

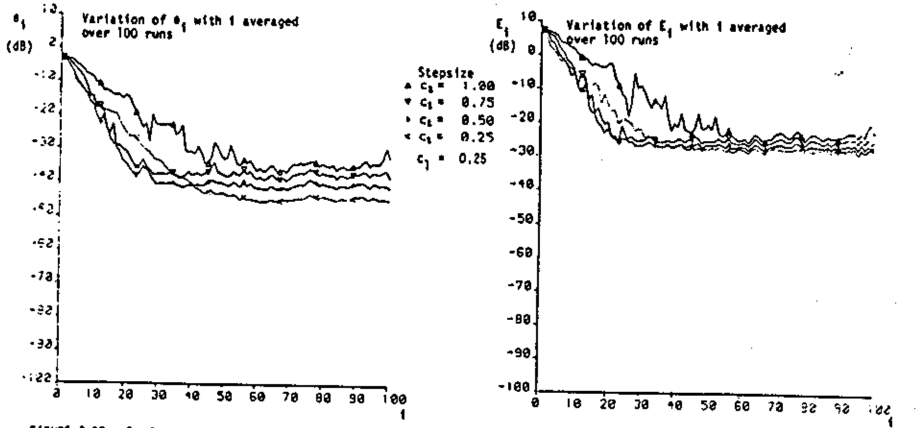


FIGURE 3.99: Performance of System 3.4 for CH2OR2, SNR = 40 dB

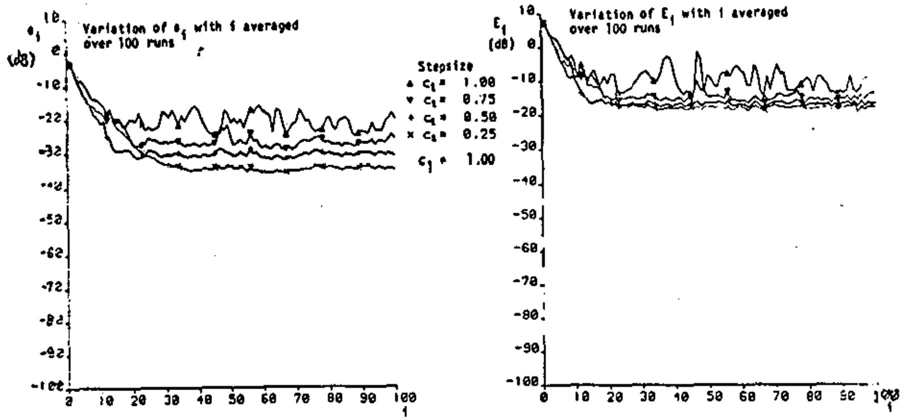


FIGURE 3.100: Performance of System 3.4 for CH2OR2, SNR = 30 dB

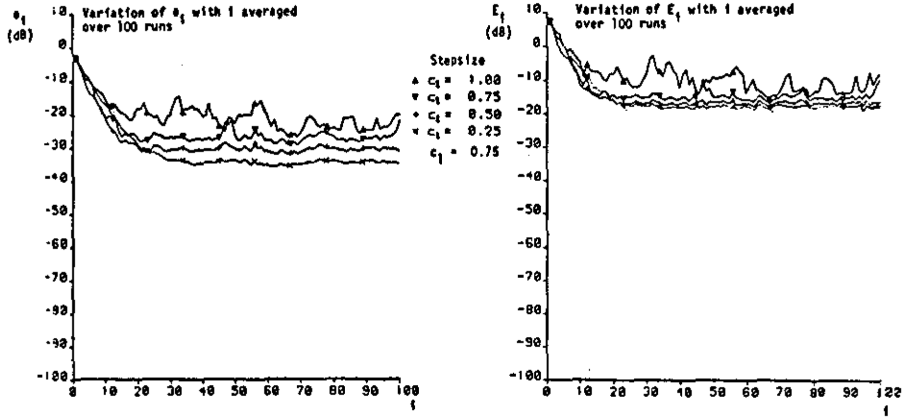


FIGURE 3.101: Performance of System 3.4 for CH2OR2, SNR = 30 dB

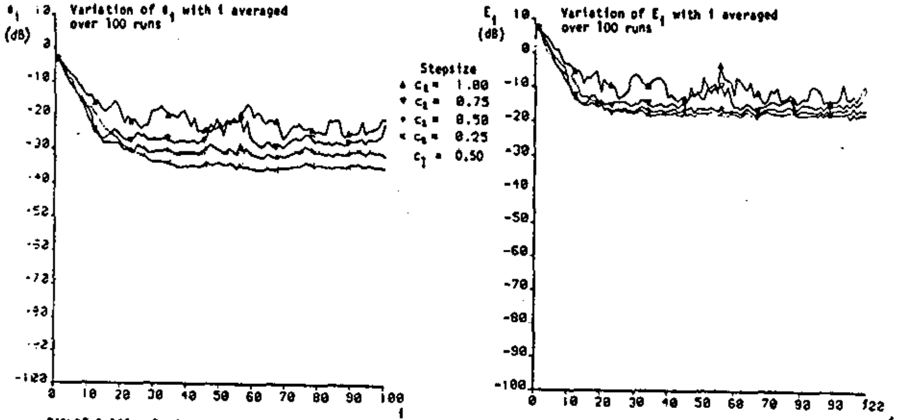


FIGURE 3.102: Performance of System 3.4 for CH2OR2, SNR = 30 dB

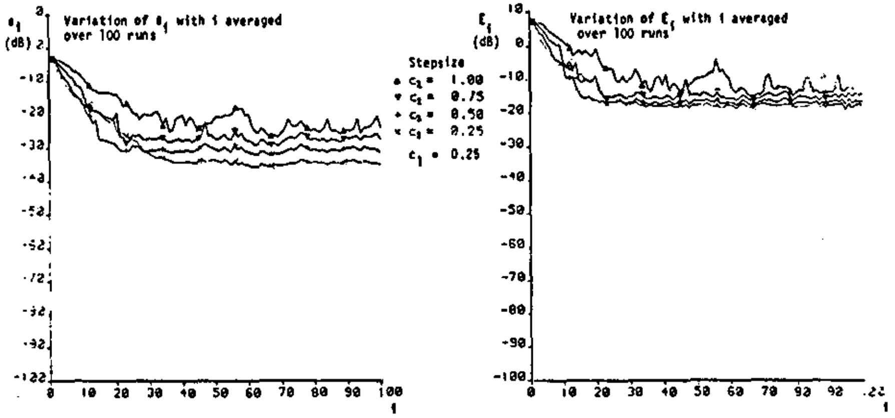


FIGURE 3.103: Performance of System 3.4 for CH2OR2, SNR = 30 dB

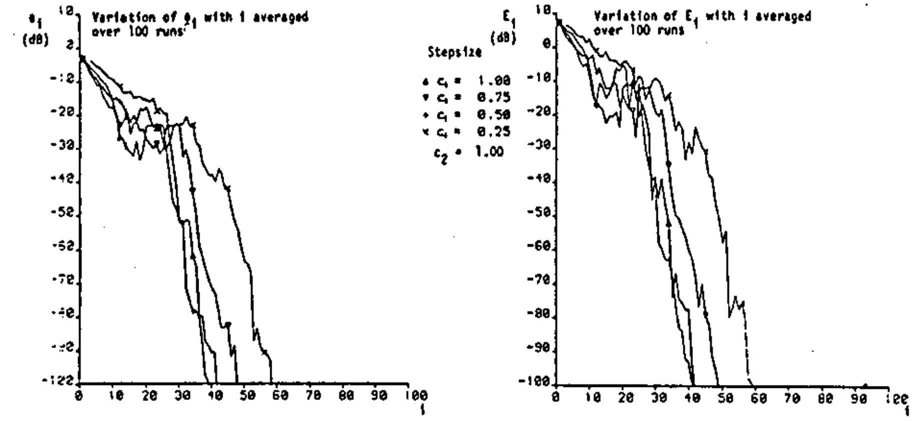


FIGURE 3.104: Performance of System 3.4 for CH2OR2, SNR = ∞ dB

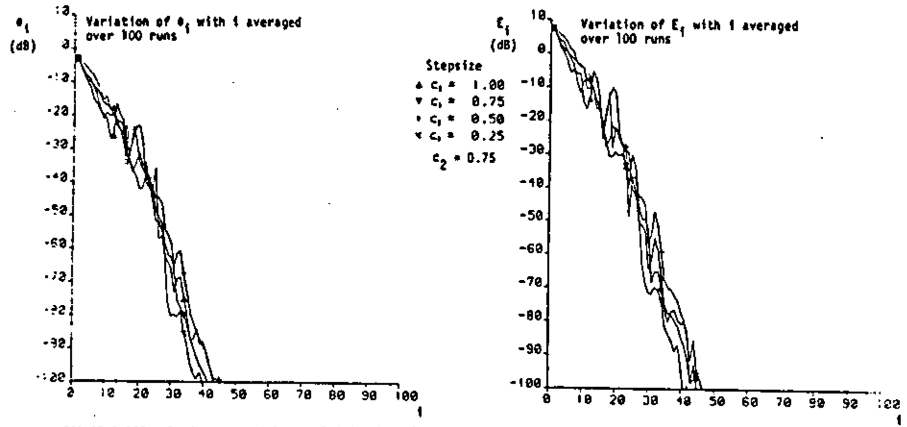


FIGURE 3.105: Performance of System 3.4 for CH2OR2, SNR = ∞ dB

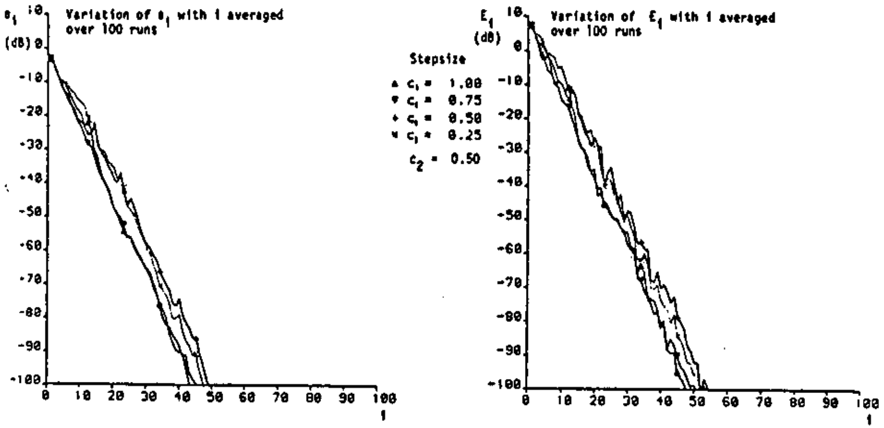


FIGURE 3.106: Performance of System 3.4 for CH2OR2, SNR = ∞ dB

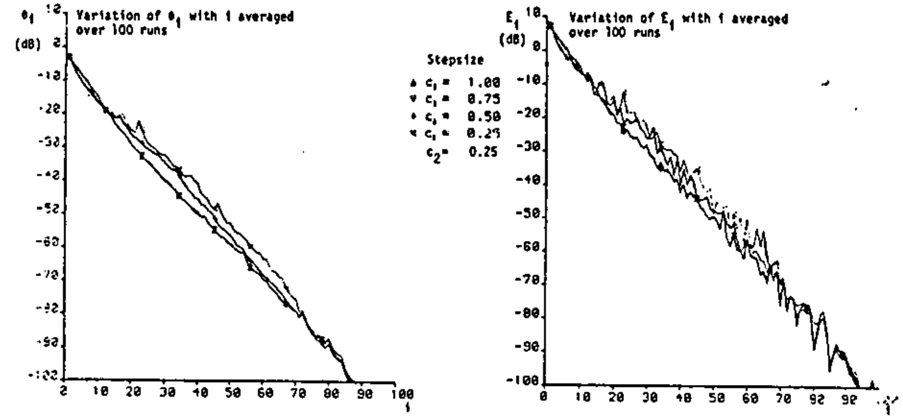


FIGURE 3.107: Performance of System 3.4 for CH2OR2, SNR = ∞ dB

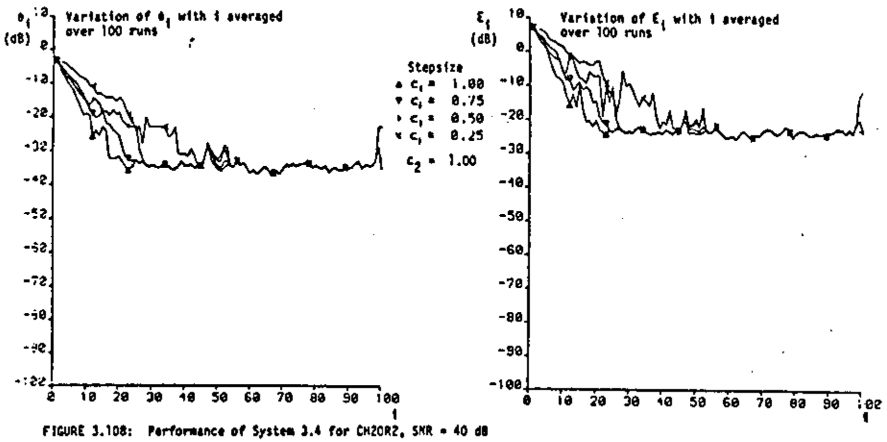


FIGURE 3.108: Performance of System 3.4 for CH2OR2, SNR = 40 dB

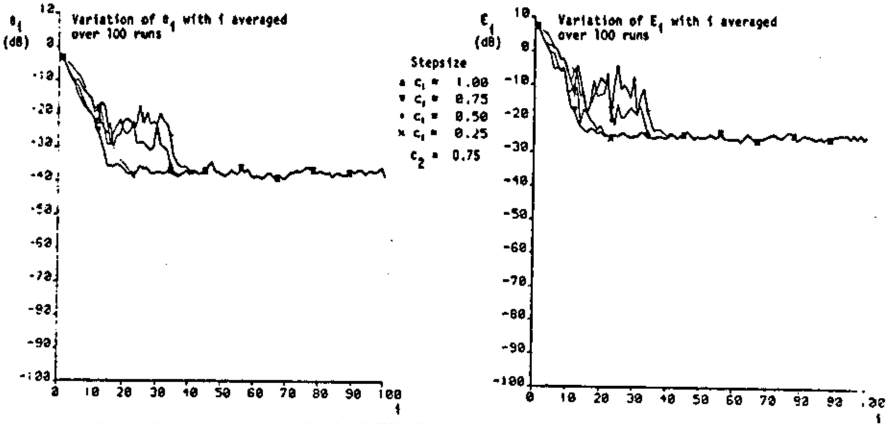


FIGURE 3.109: Performance of System 3.4 for CH2OR2, SNR = 40 dB

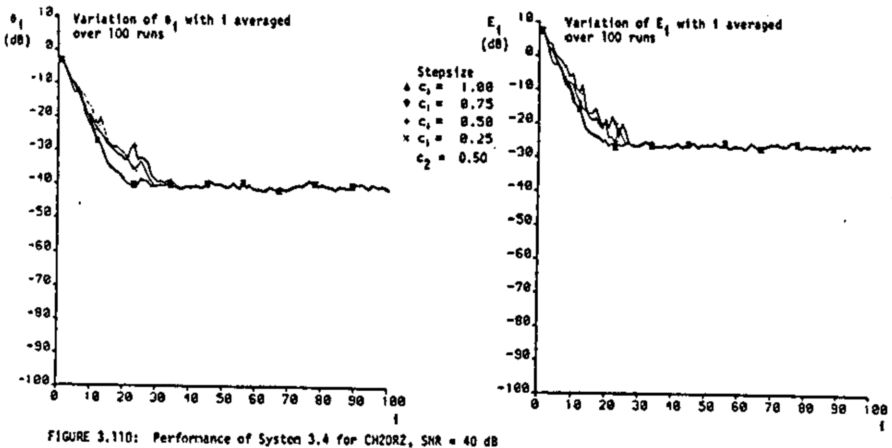


FIGURE 3.110: Performance of System 3.4 for CH2OR2, SNR = 40 dB

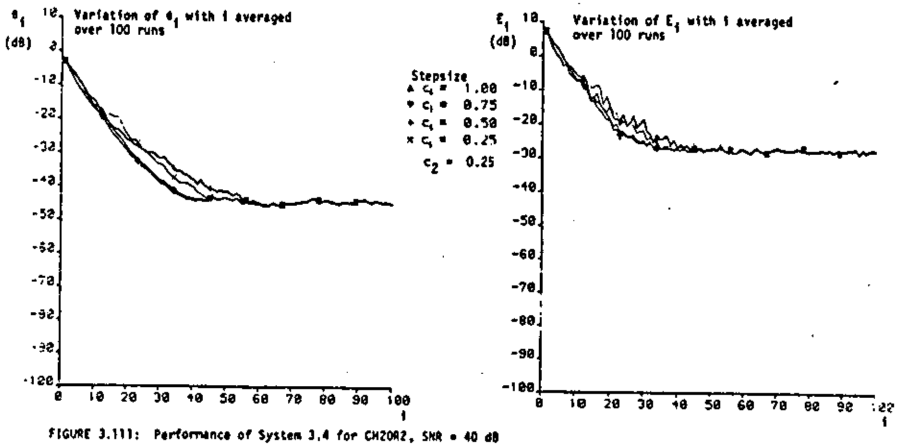


FIGURE 3.111: Performance of System 3.4 for CH2OR2, SNR = 40 dB

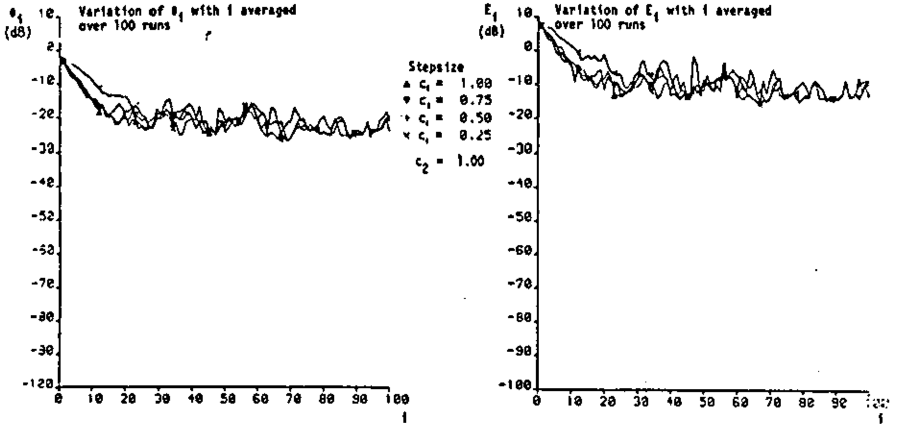


FIGURE 3.112: Performance of System 3.4 for CH2OR2, SNR = 30 dB

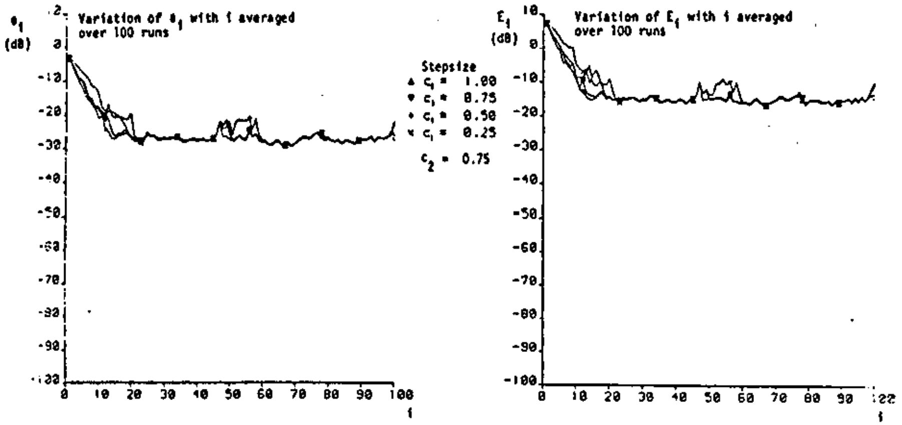


FIGURE 3.113: Performance of System 3.4 for CH2OR2, SNR = 30 dB

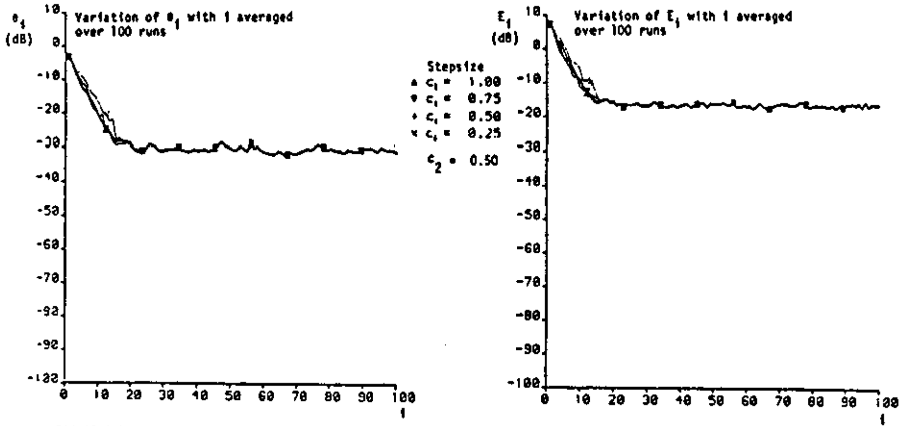


FIGURE 3.114: Performance of System 3.4 for CH2OR2, SNR = 30 dB

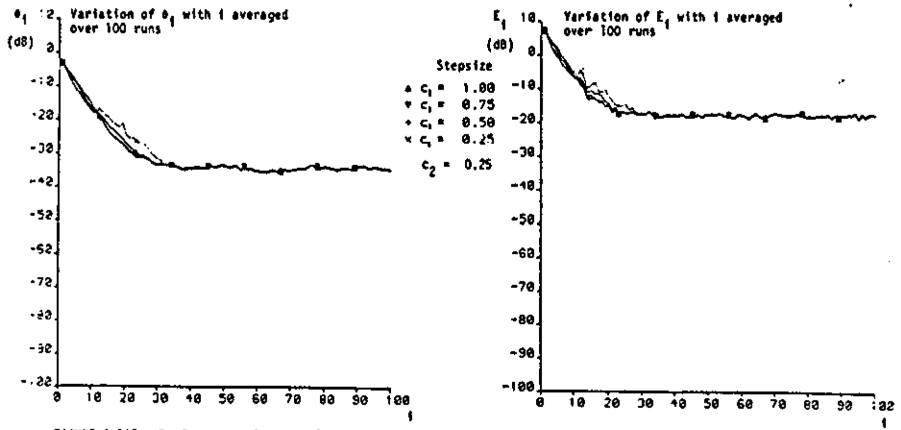


FIGURE 3.115: Performance of System 3.4 for CH2OR2, SNR = 30 dB

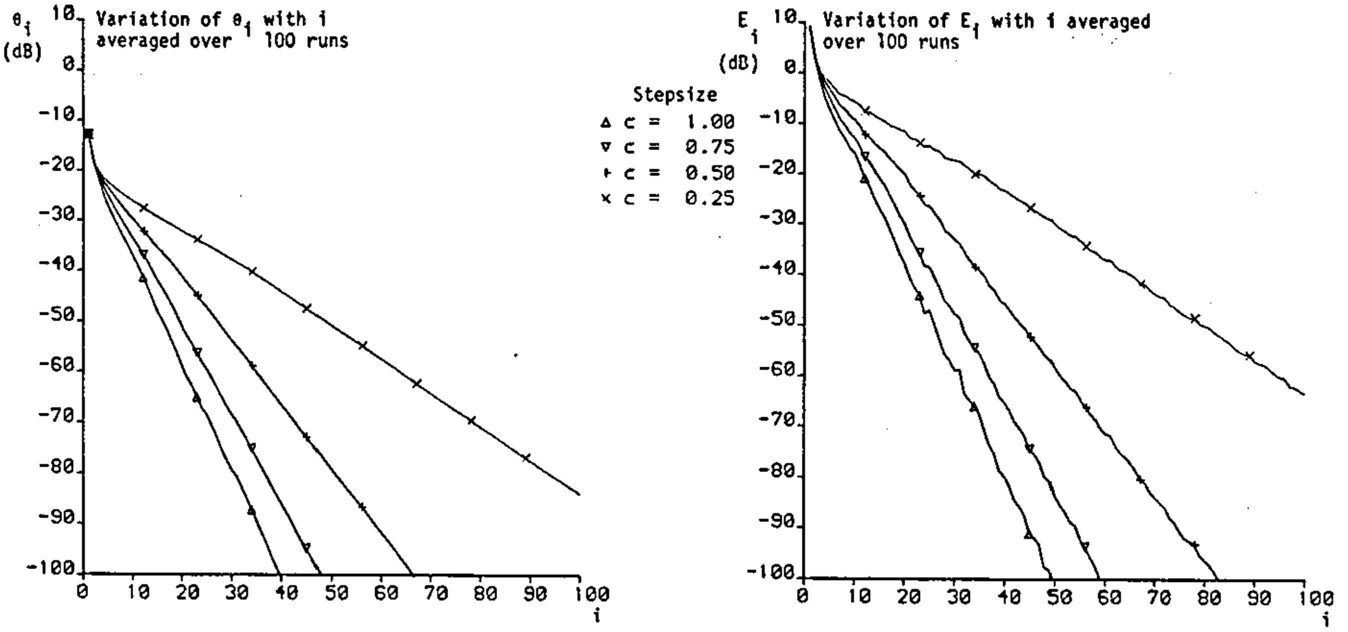


FIGURE 3.116: Performance of System 3.4 for CH19R0, SNR = ∞ dB

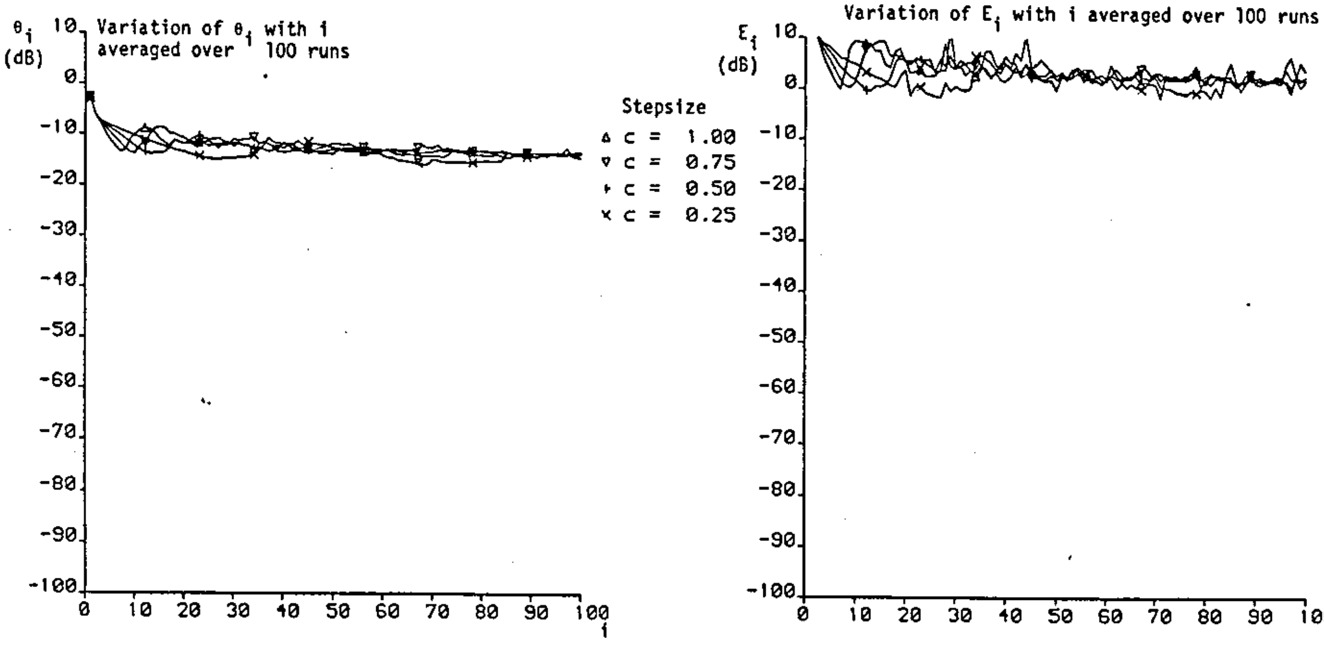
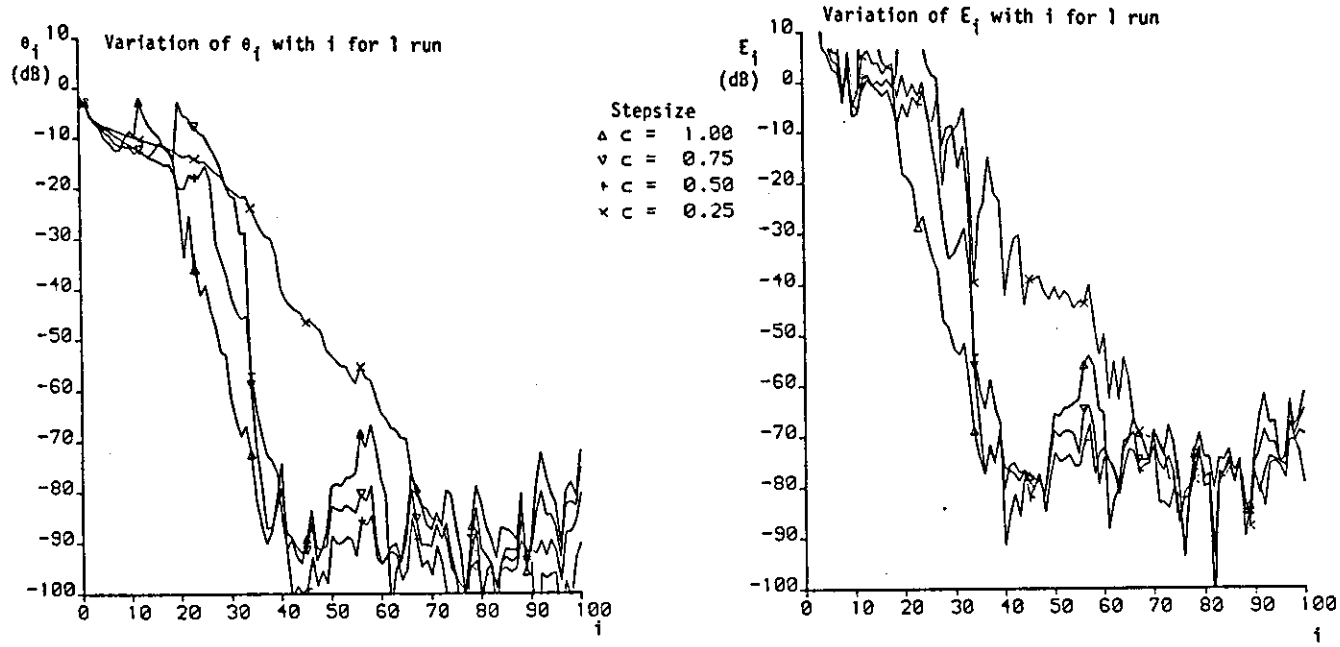
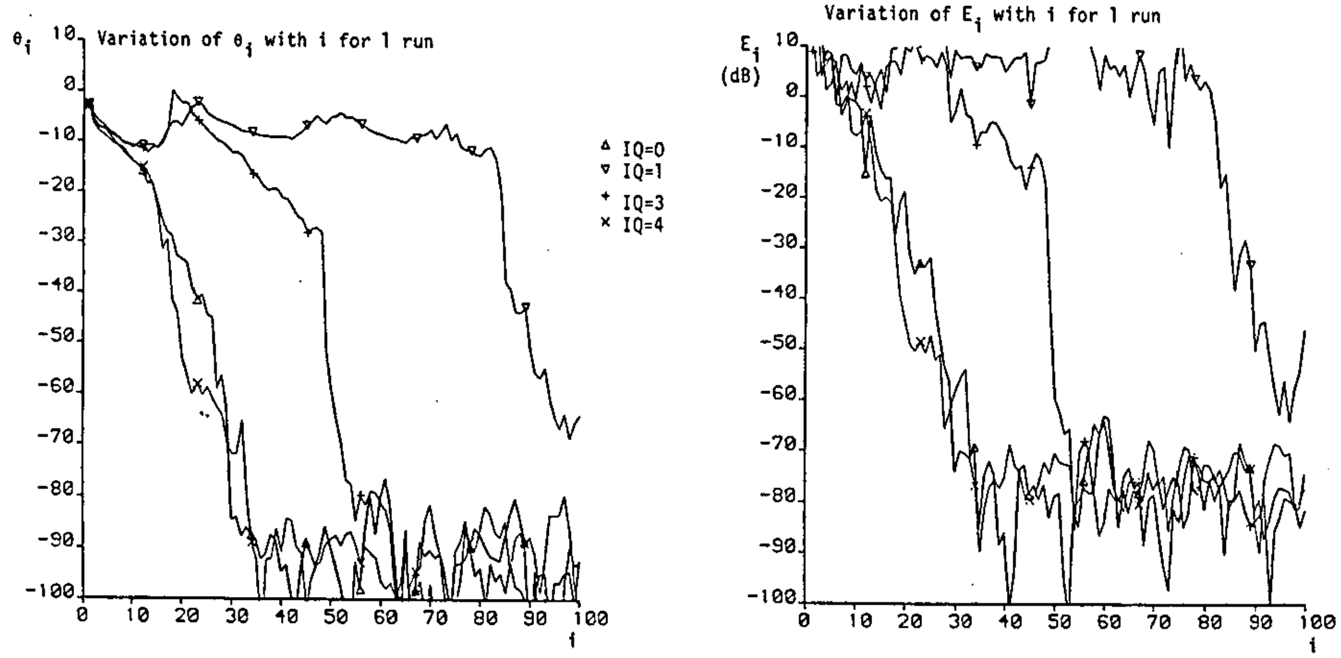


FIGURE 3.117: Performance of System 3.4 for CH19R1, SNR = ∞ dB


Variation of θ_i with i for 1 run

Variation of E_i with i for 1 run

FIGURE 3.118: Performance of System 3.4 for CH19R1, SNR = ∞dB


Variation of θ_i with i for 1 run

Variation of E_i with i for 1 run

FIGURE 3.119: Performance of System 3.4 for CH19R1, SNR = ∞dB, different $\{r_i\}$ c=0.5

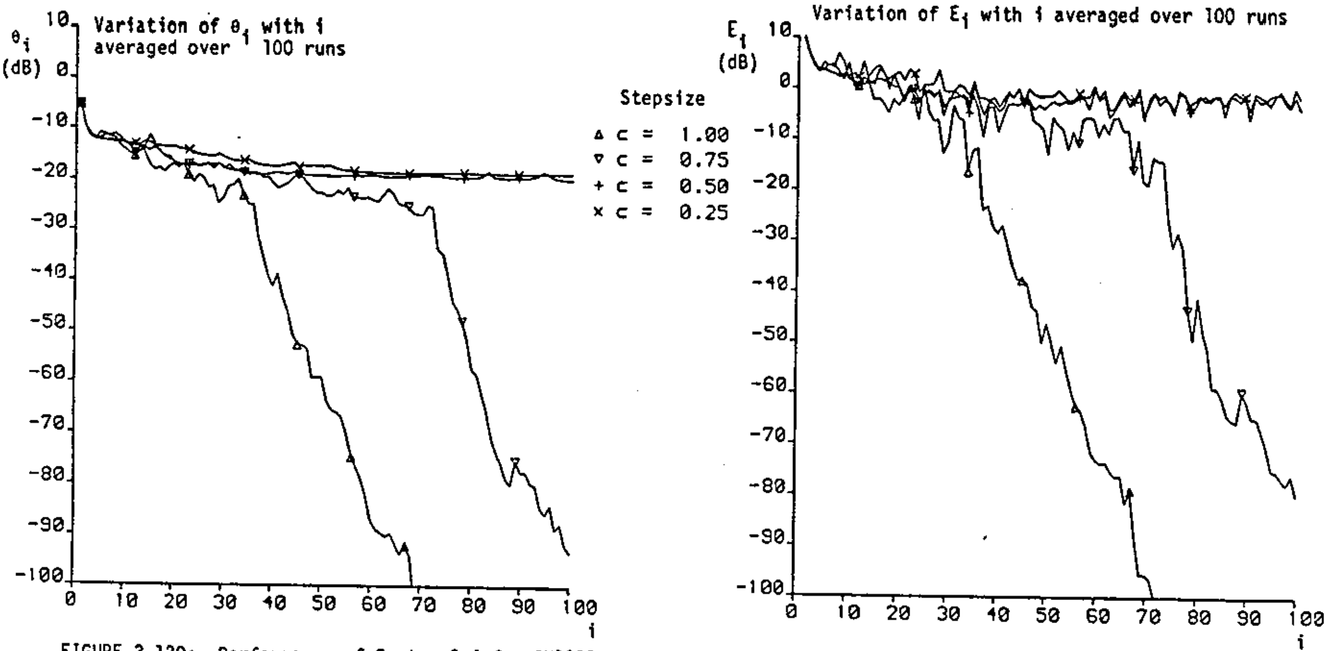


FIGURE 3.120: Performance of System 3.4 for CH19R2, SNR = ∞ dB

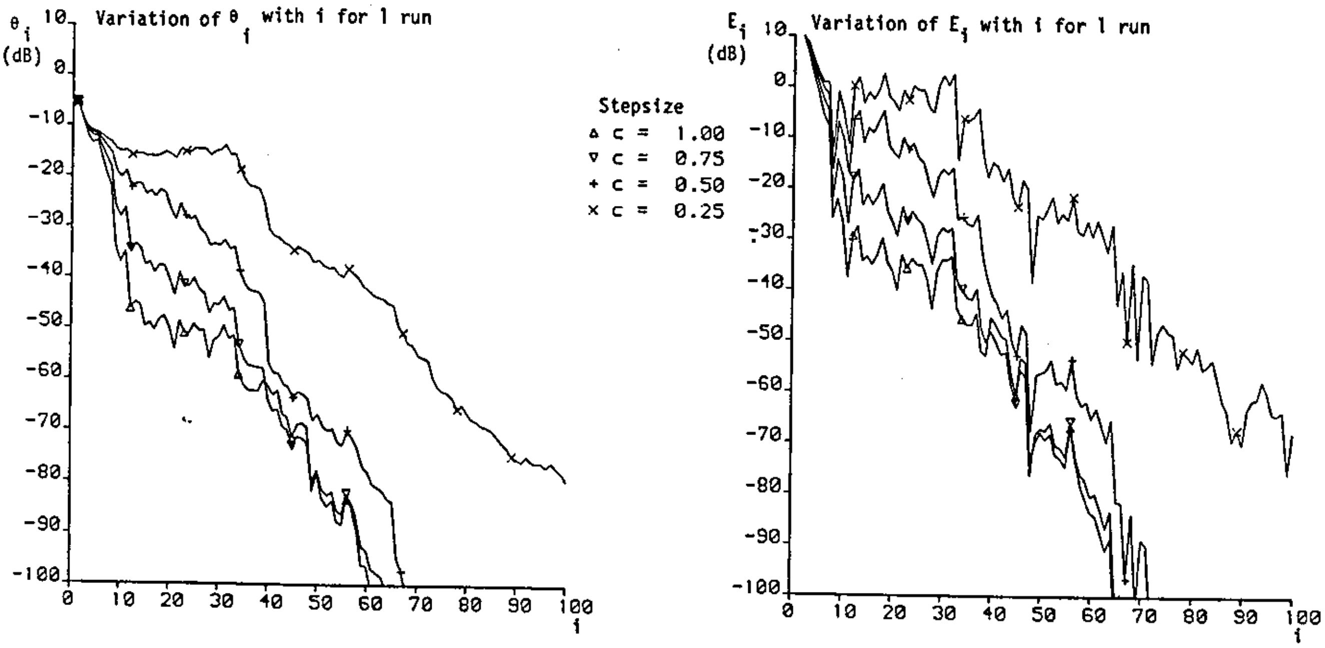


FIGURE 3.121: Performance of System 3.4 for CH19R2, SNR = ∞dB

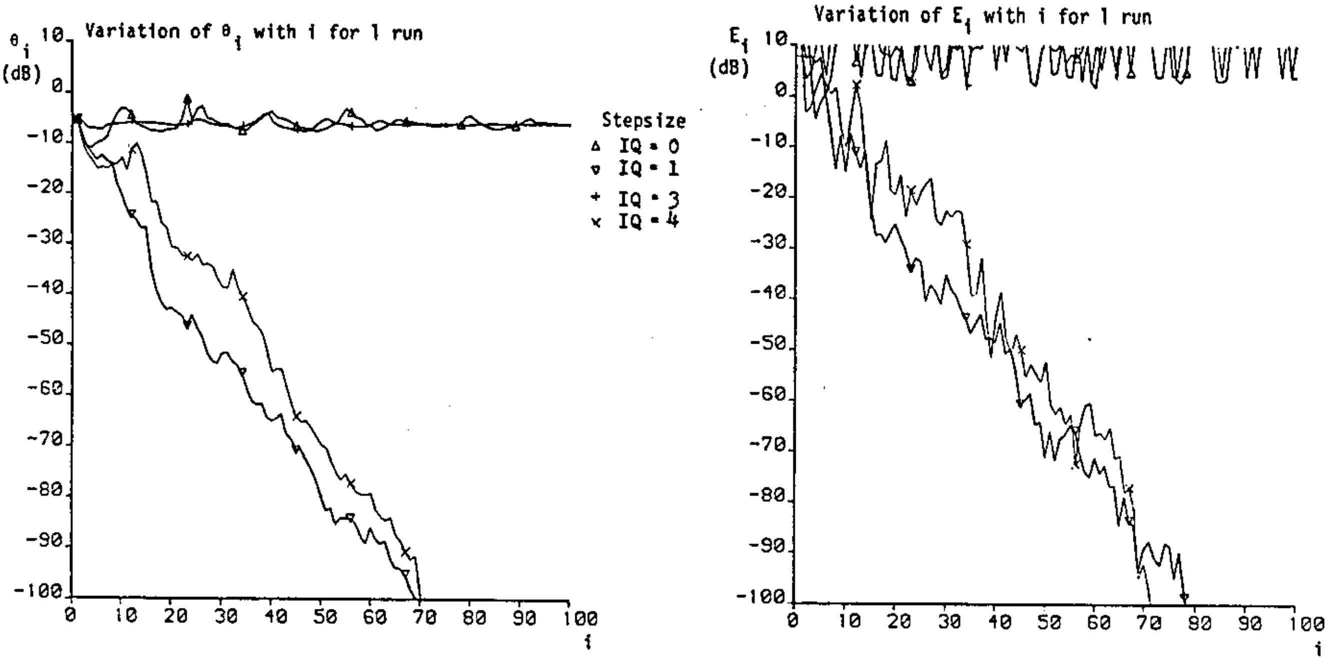


FIGURE 3.122: Performance of System 3.4 for CH19R2, SNR=∞dB, different $\{r_i\}$ c=0.5

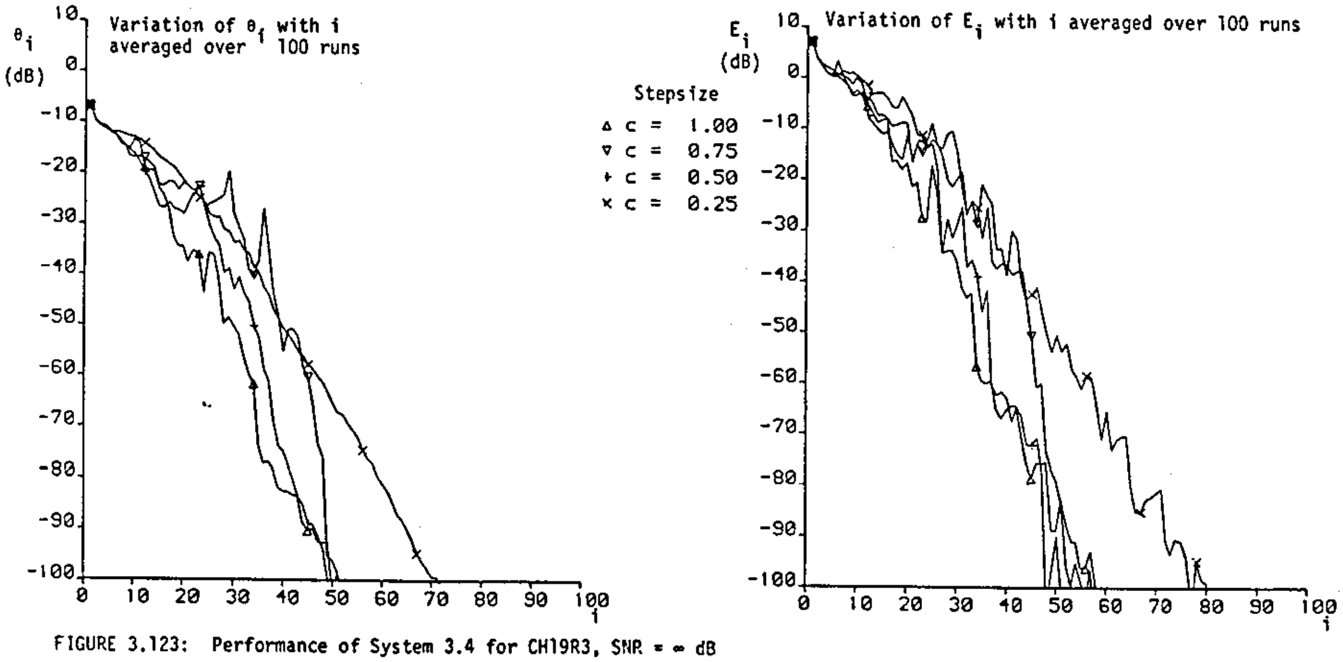


FIGURE 3.123: Performance of System 3.4 for CH19R3, SNR = ∞ dB

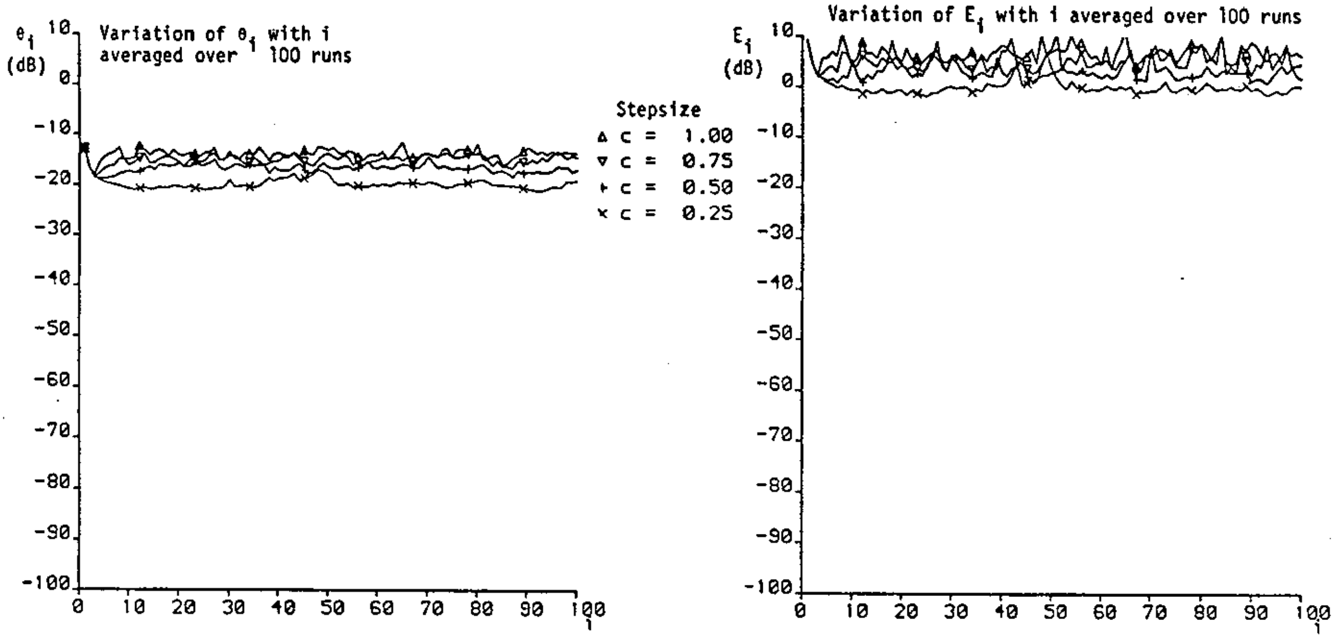


FIGURE 3.124: Performance of System 3.4 for CH19R0, SNR = 40 dB

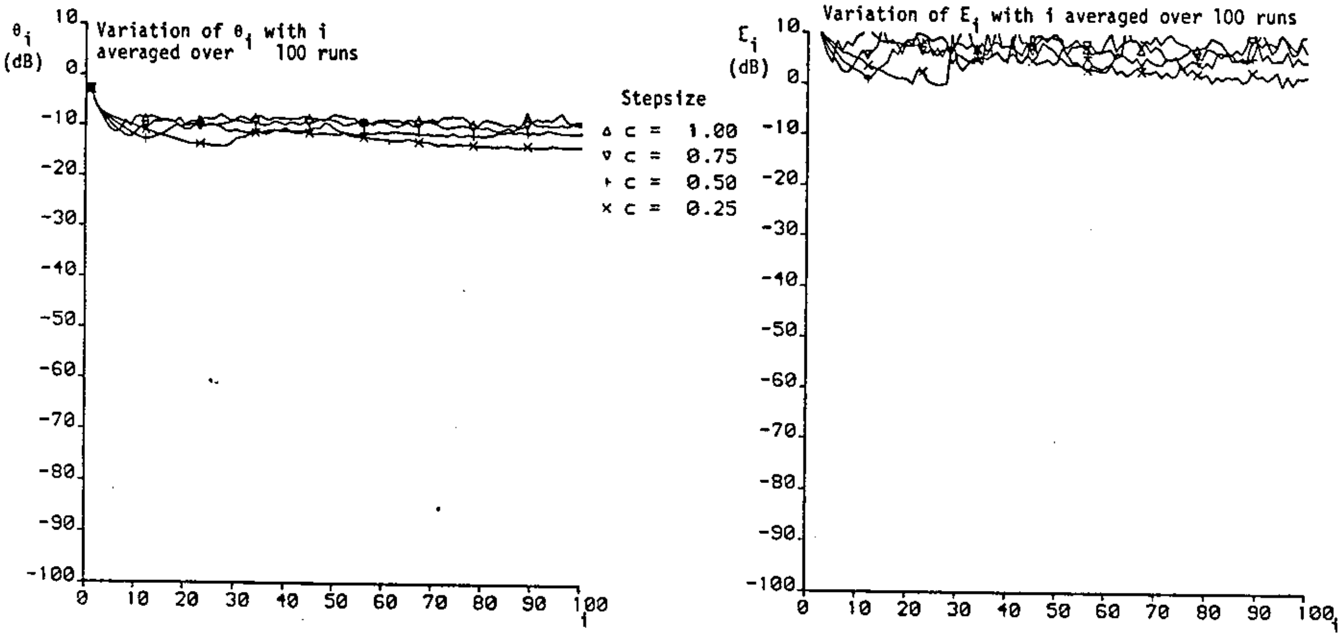


FIGURE 3.125: Performance of System 3.4 for CH19R1, SNR = 40 dB

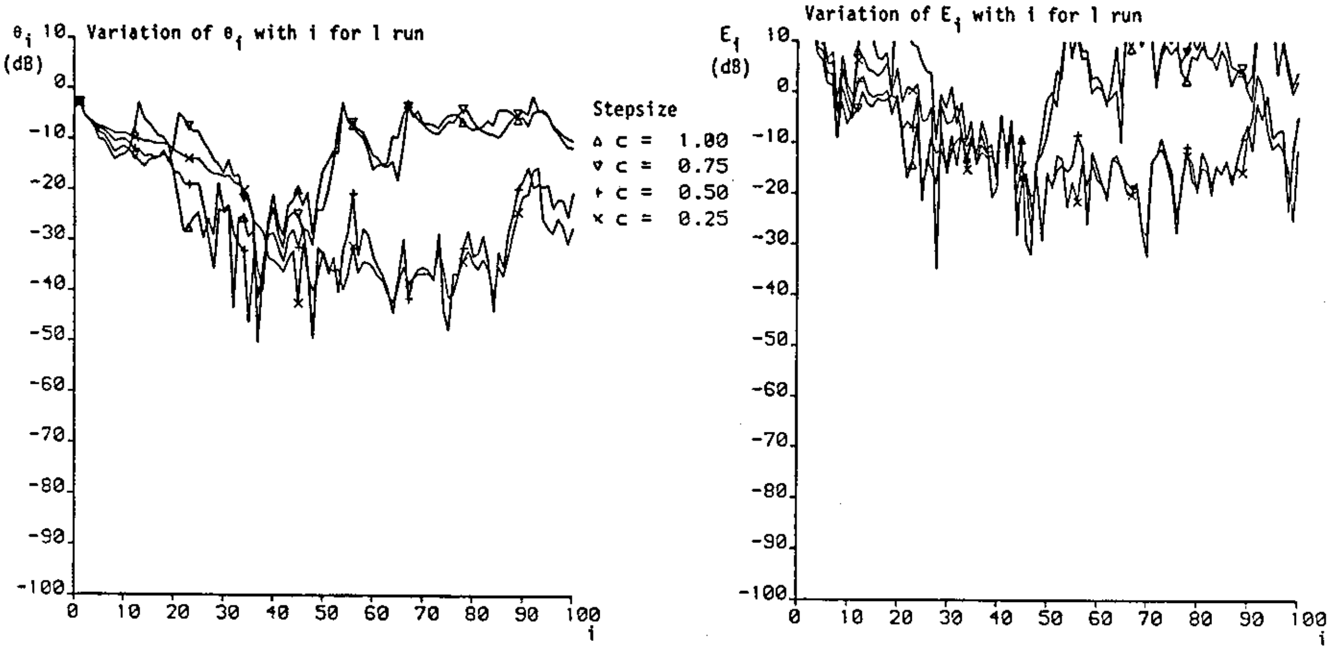


FIGURE 3.126: Performance of System 3.4 for CH19R1, SNR = 40 dB

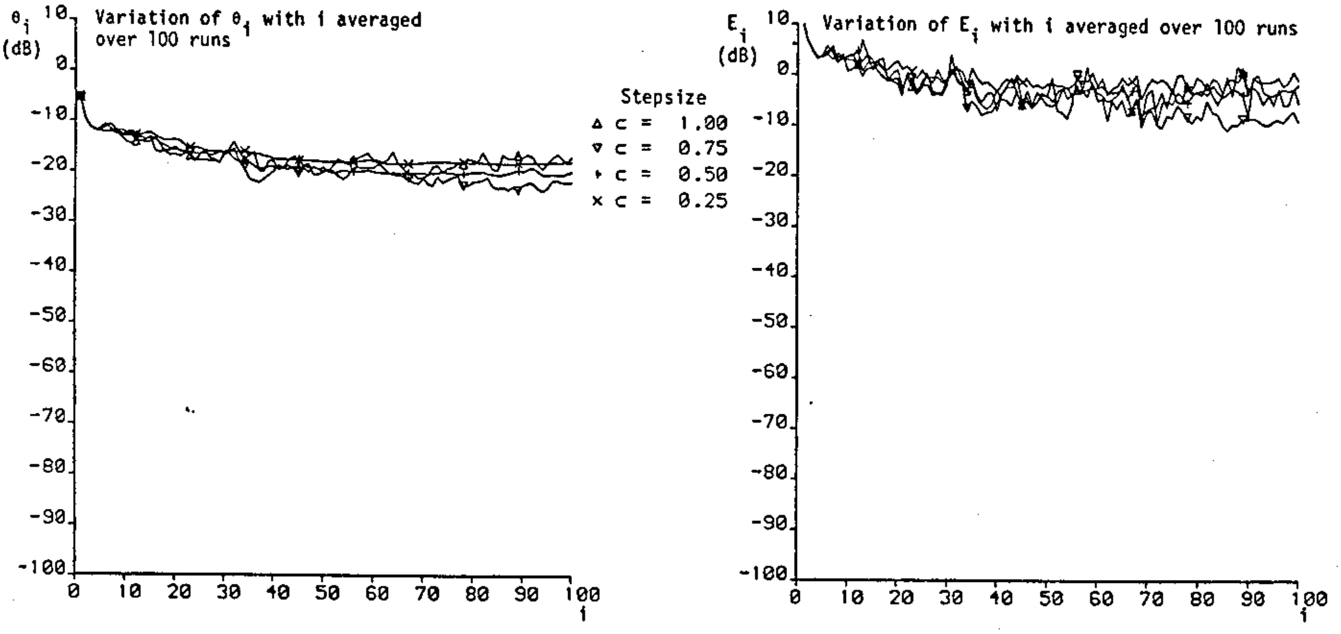


FIGURE 3.127: Performance of System 3.4 for CH19R2, SNR = 40 dB

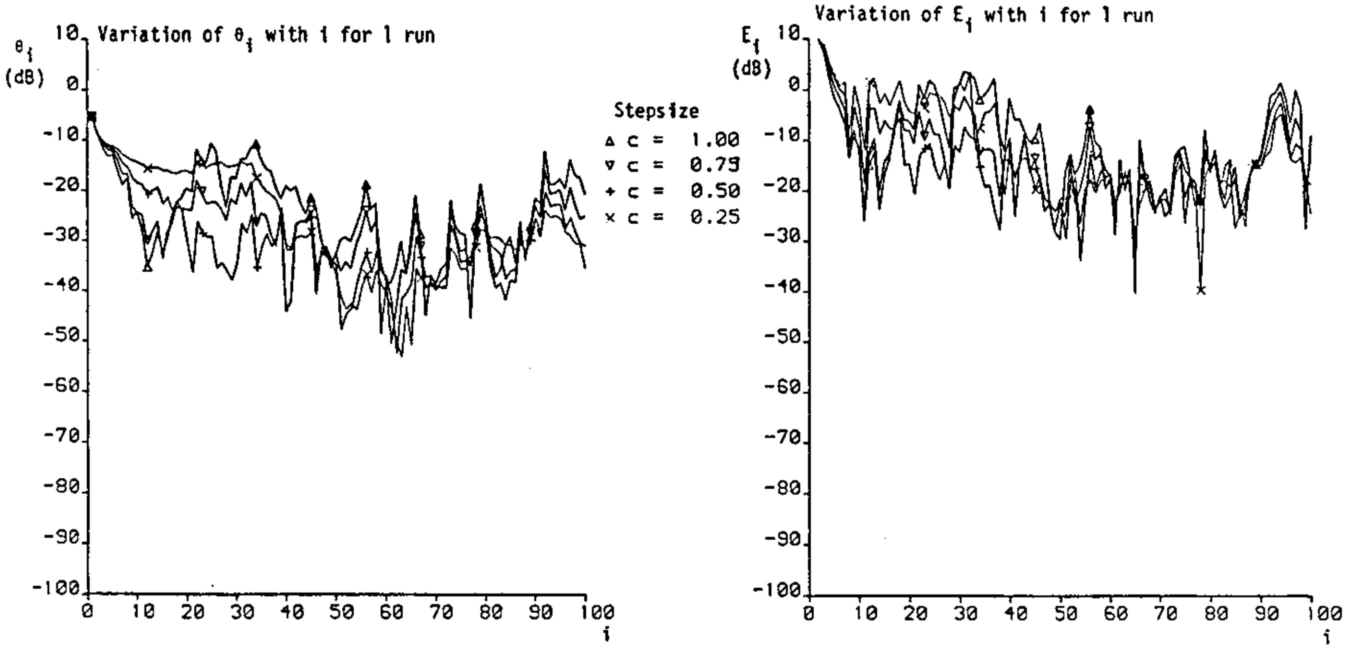


FIGURE 3.128: Performance of System 3.4 for CH19R2, SNR = 40 dB

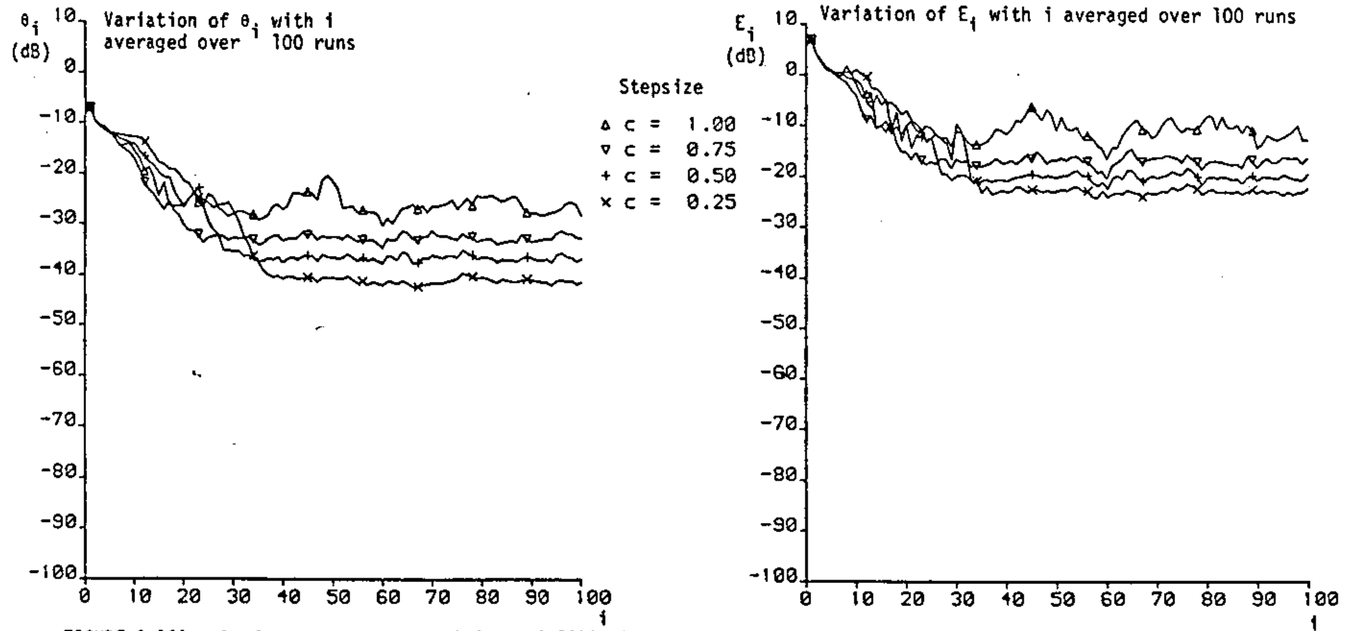


FIGURE 3.129: Performance of System 3.4 for CH19R3, SNR = 40 dB

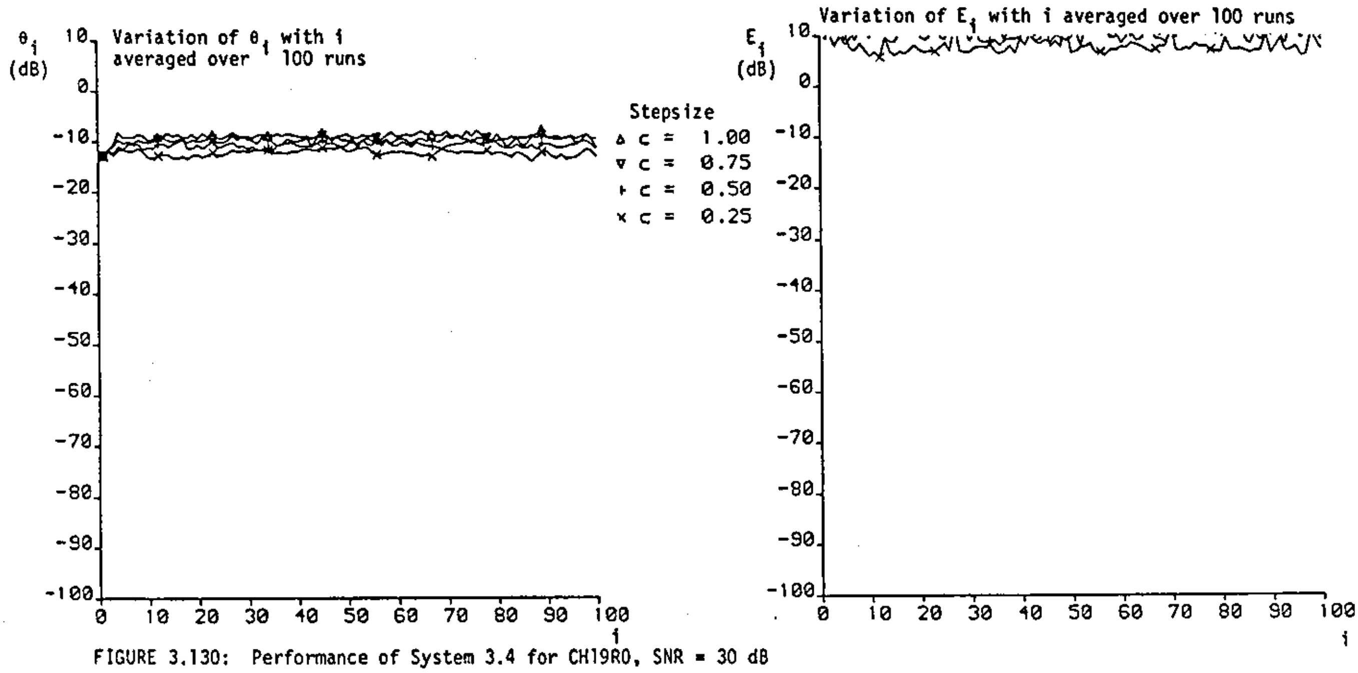


FIGURE 3.130: Performance of System 3.4 for CH19R0, SNR = 30 dB

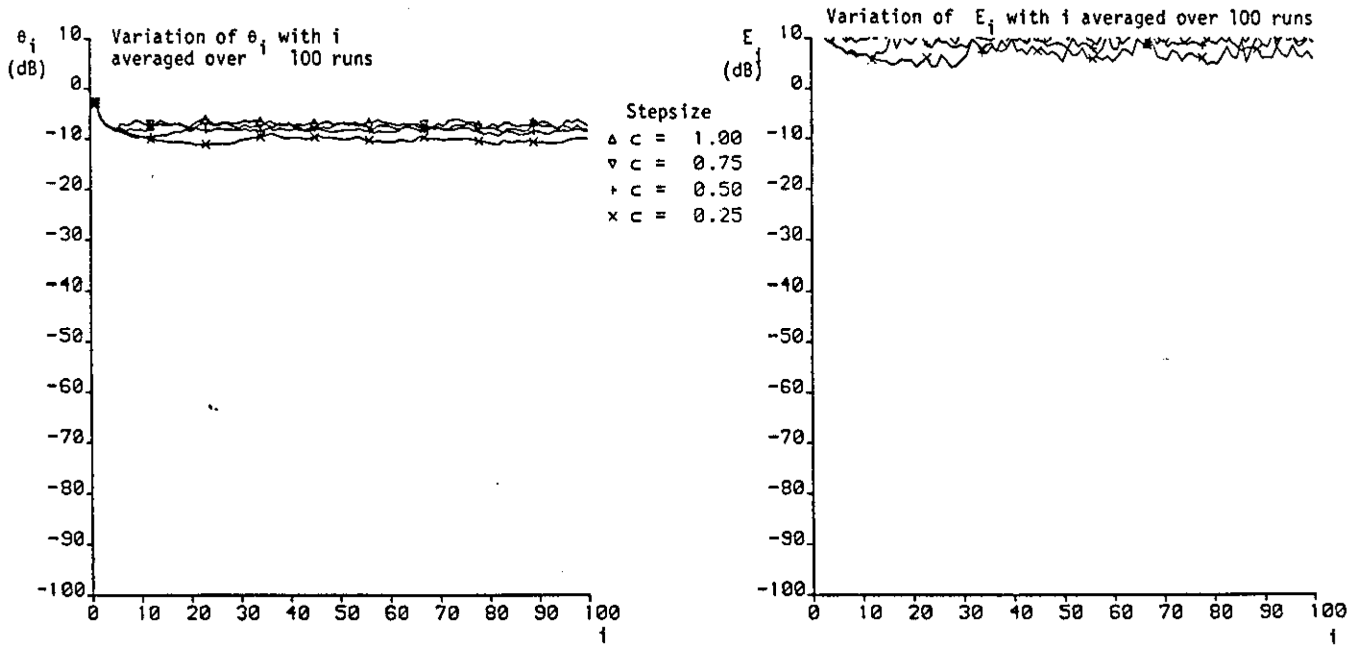


FIGURE 3.131: Performance of System 3.4 for CH19R1, SNR = 30 dB

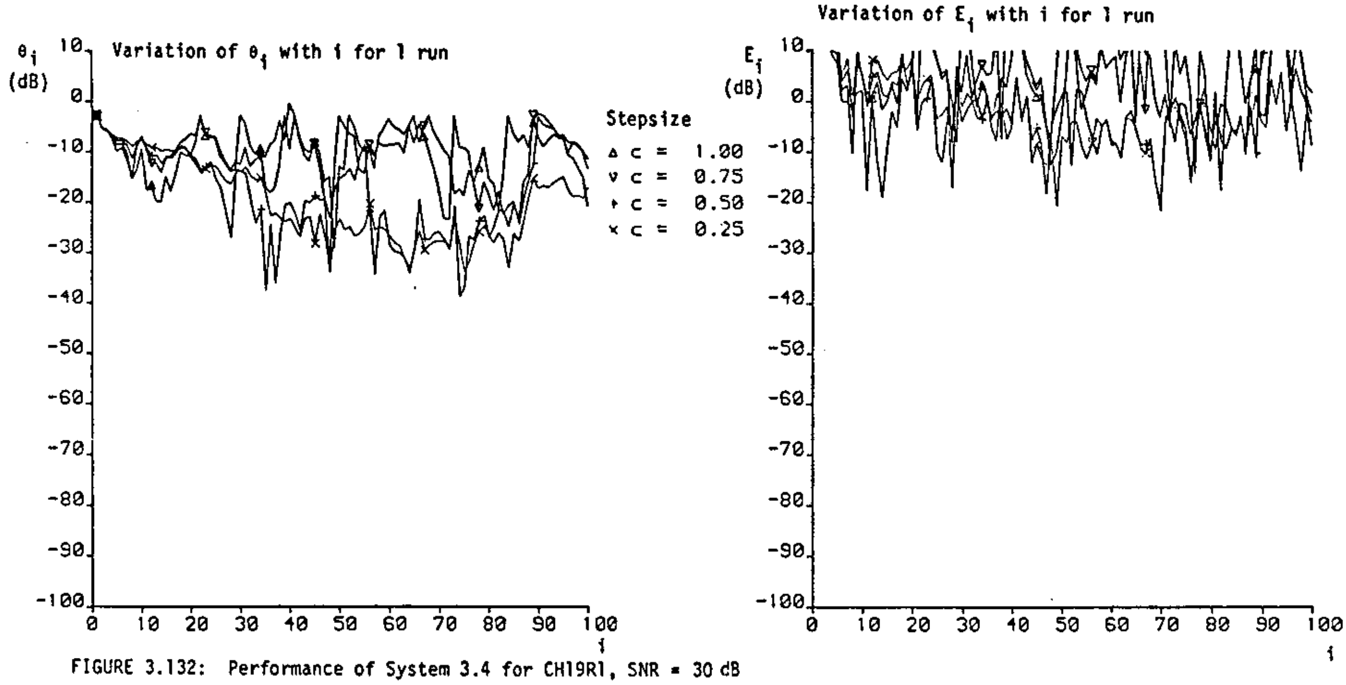


FIGURE 3.132: Performance of System 3.4 for CH19R1, SNR = 30 dB

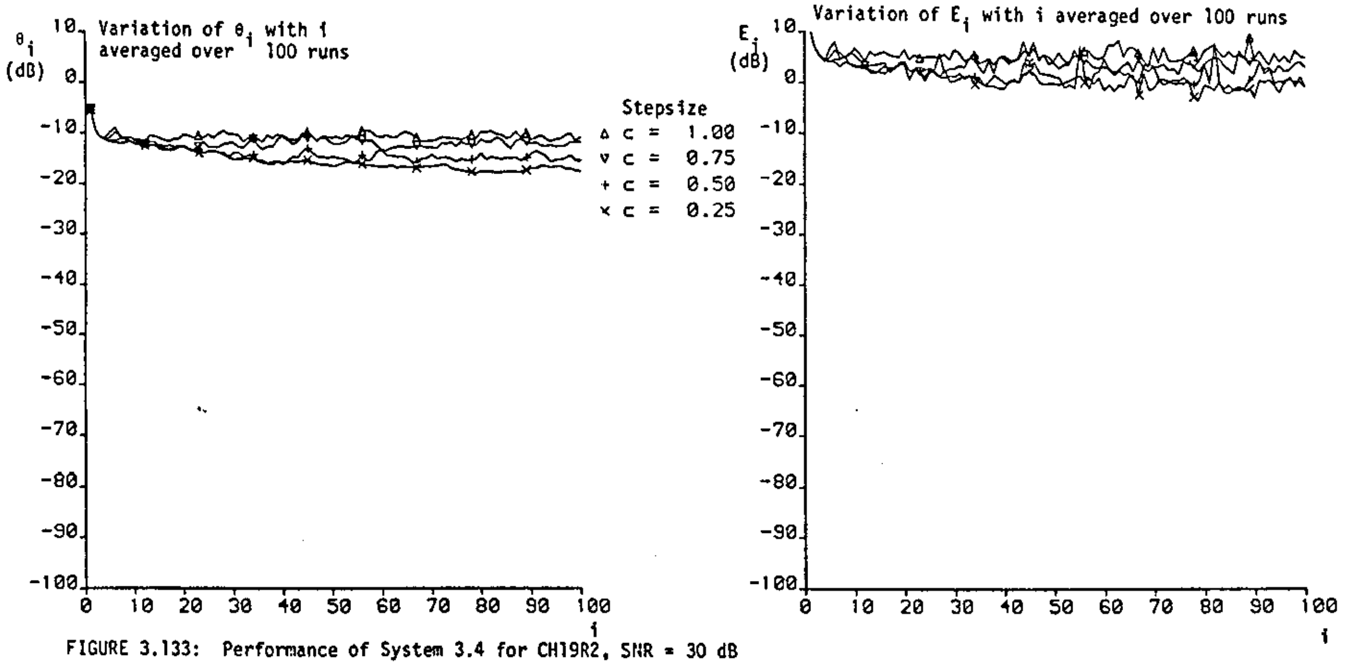


FIGURE 3.133: Performance of System 3.4 for CH19R2, SNR = 30 dB

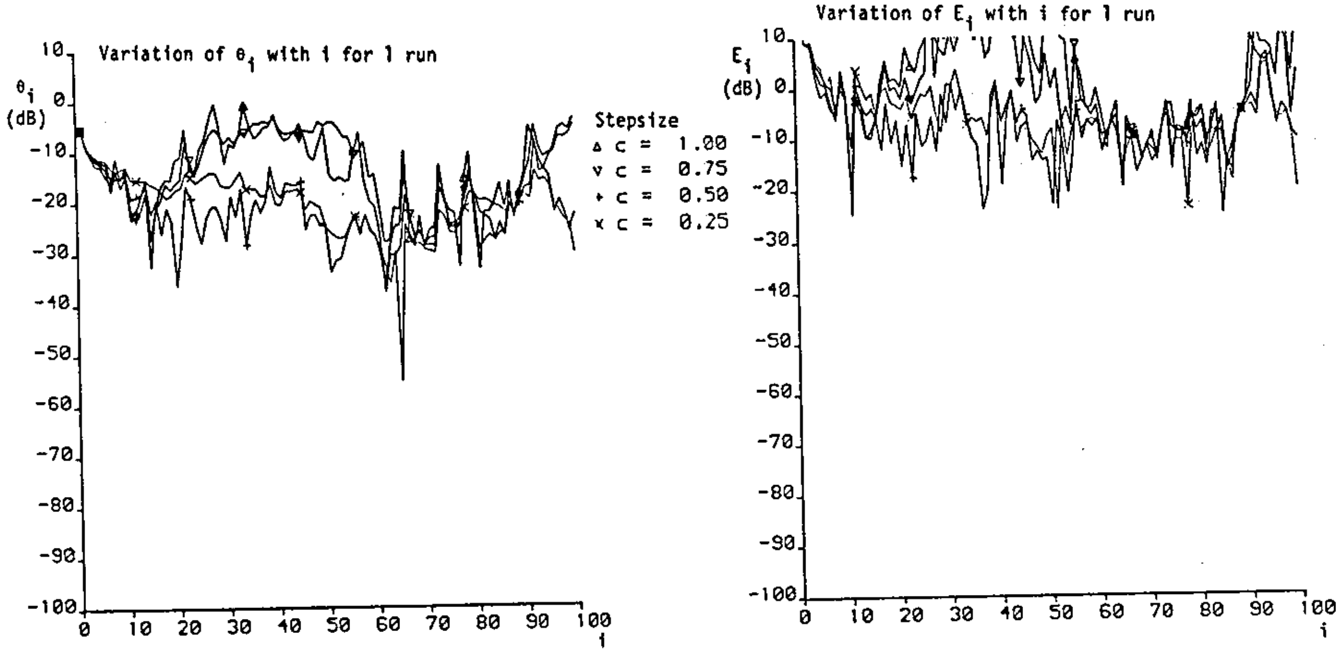


FIGURE 3.134: Performance of System 3.4 for CH19R2, SNR = 30 dB

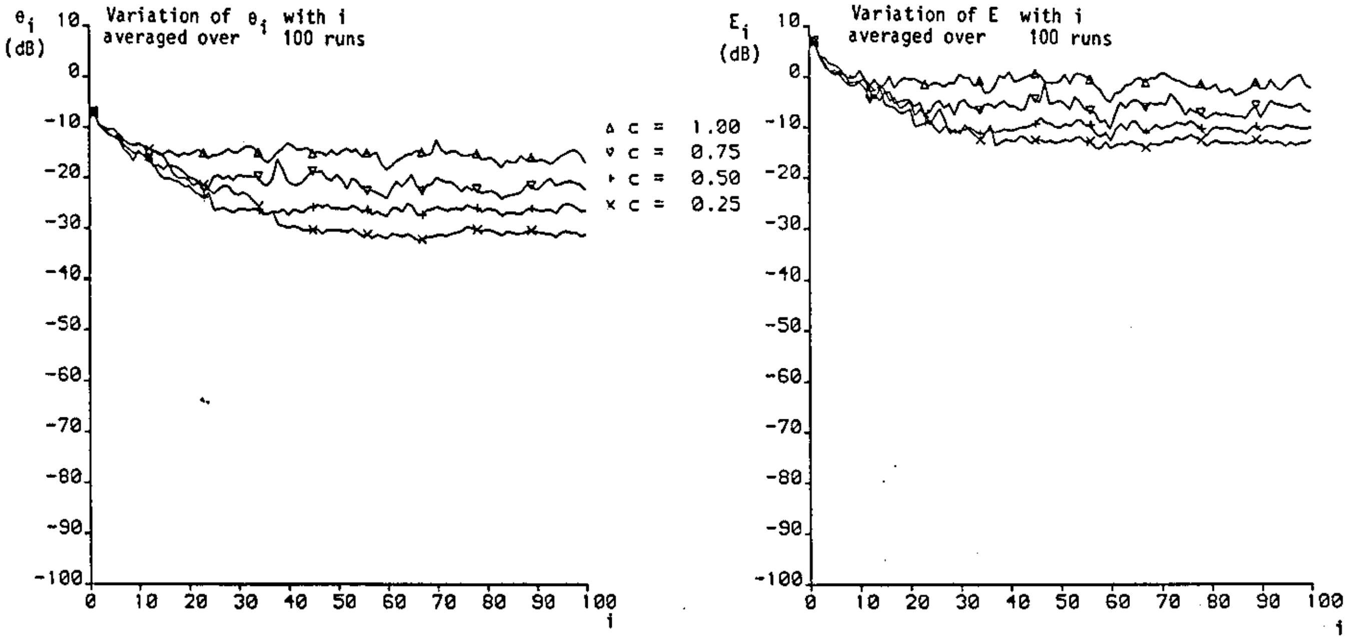


FIGURE 3.135: Performance of System 3.4 for CH19R3, SNR = 30 dB

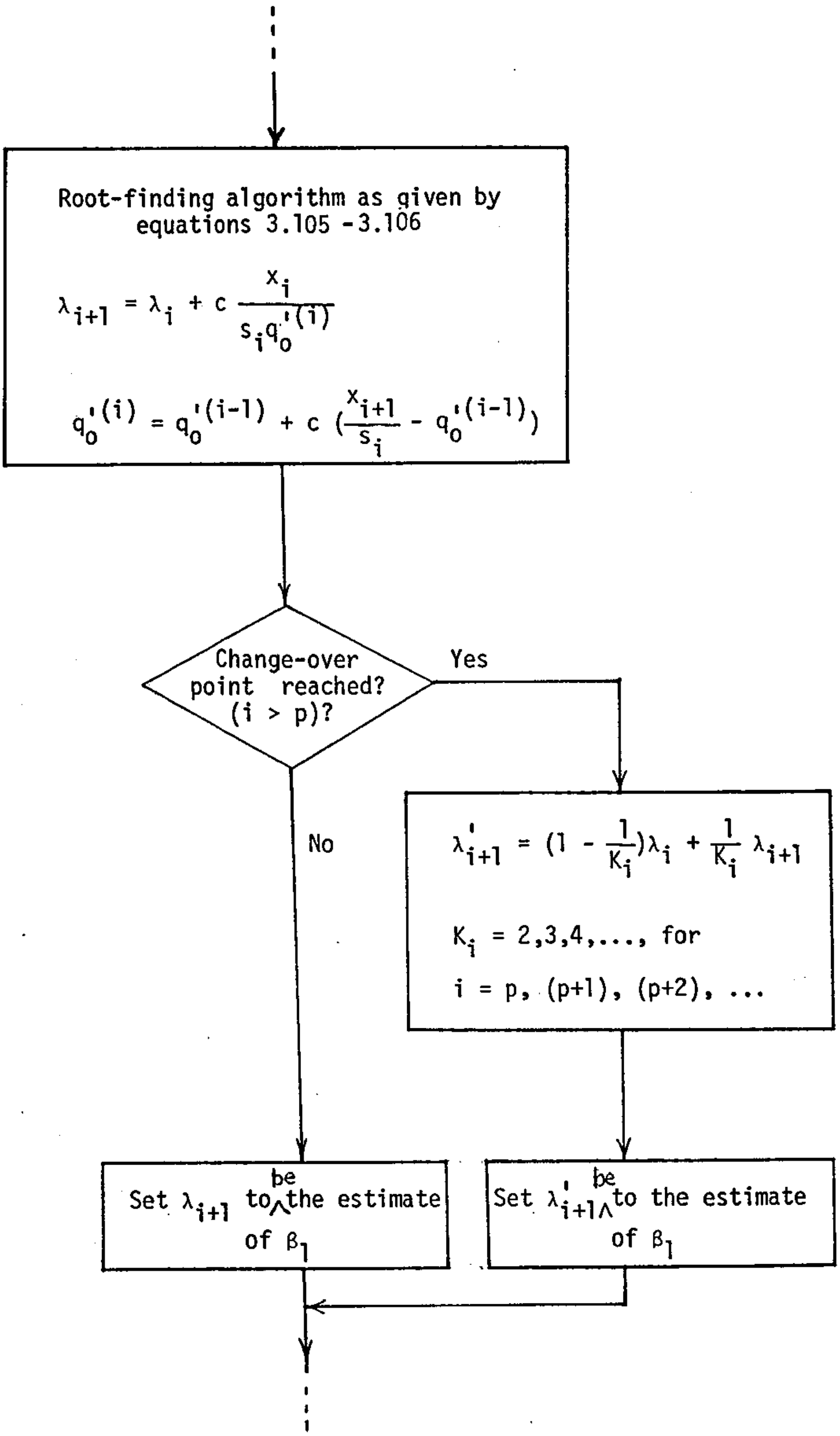

Root-finding algorithm as given by equations 3.105 -3.106
$\lambda_{i+1} = \lambda_i + c \frac{x_i}{s_i q_o'(i)}$
$q_o'(i) = q_o'(i-1) + c (\frac{x_{i+1}}{s_i} - q_o'(i-1))$
Change-over point reached? (i > p)?
Yes
No
$\lambda_{i+1}' = (1 - \frac{1}{K_i})\lambda_i + \frac{1}{K_i} \lambda_{i+1}$
$K_i = 2,3,4,\ldots,$ for
$i = p, (p+1), (p+2), \ldots$
Set λ_{i+1} to be the estimate of β_1
Set λ_{i+1}' to be the estimate of β_1

FIGURE 3.136: Flow Diagram Showing the Operation of the Two Different Modes of the Algorithm at the ith Iteration

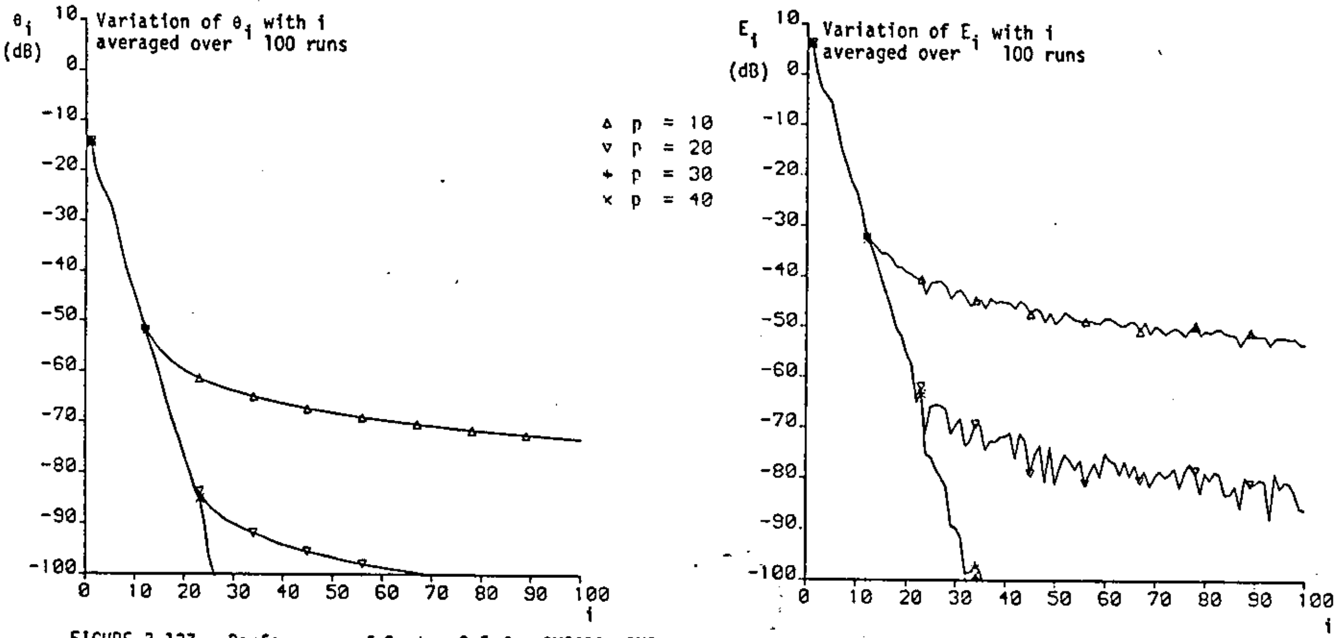


FIGURE 3.137: Performance of System 3.5 for CH2ORO, SNR = ∞ dB, c=1

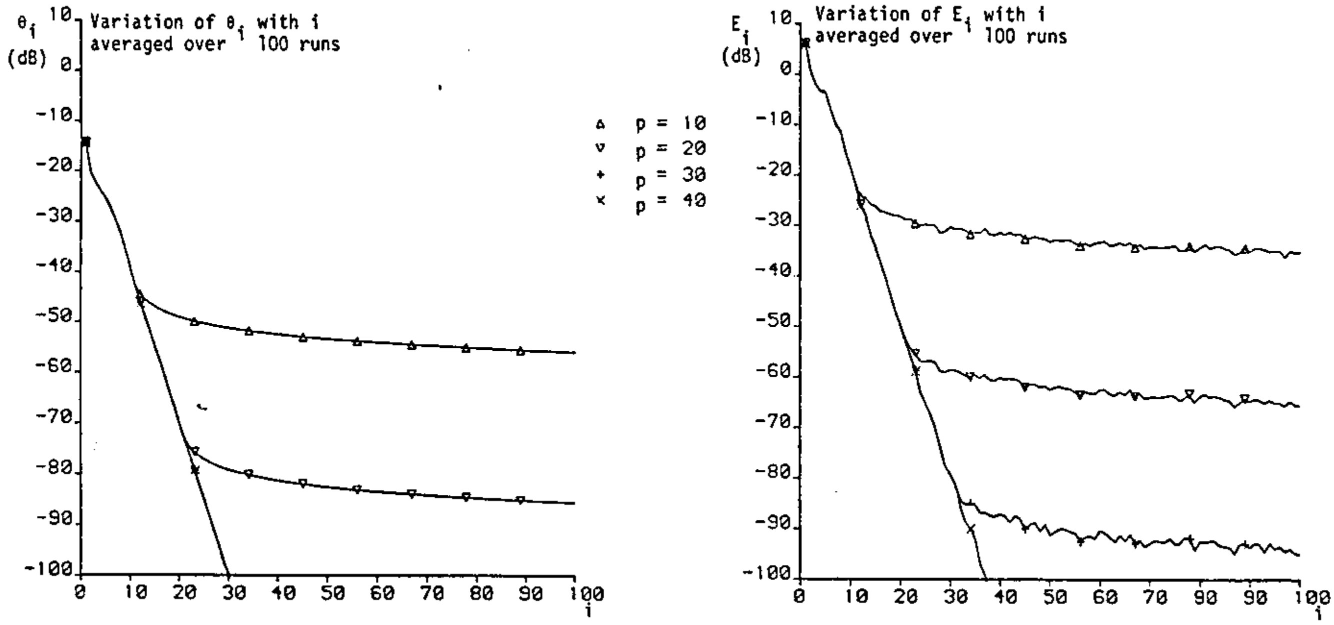


FIGURE 3.138: Performance of System 3.5 for CH2ORO, SNR = ∞ dB, c=0.5

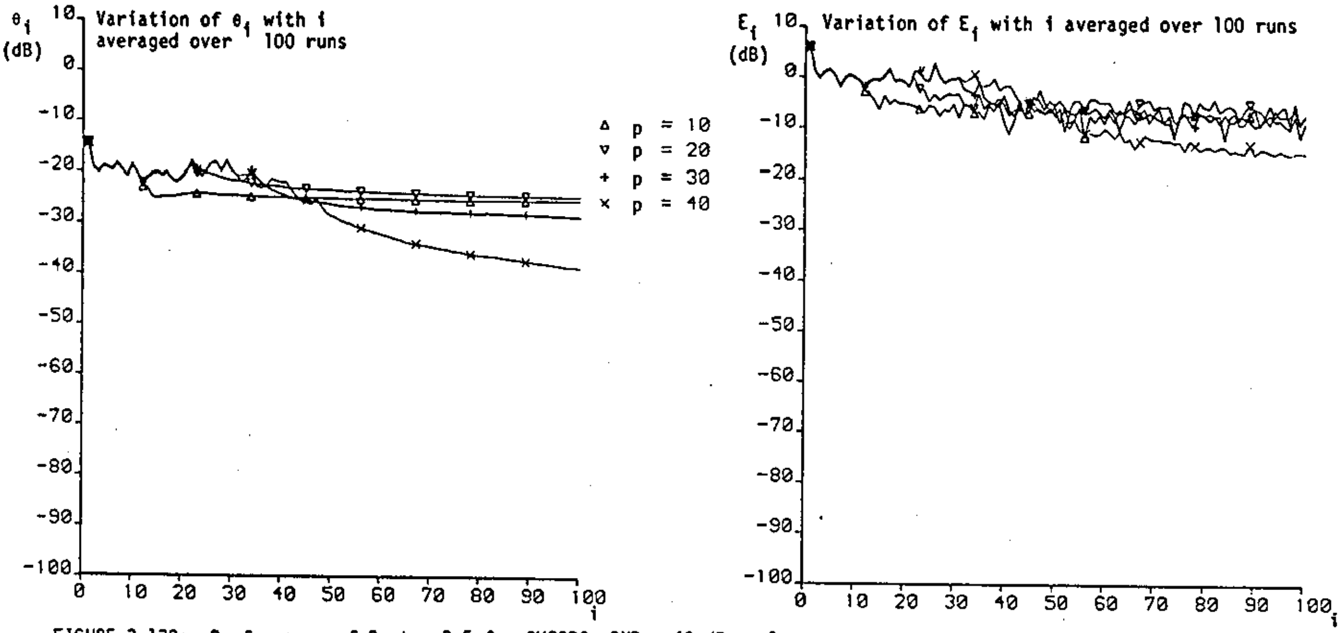


FIGURE 3.139: Performance of System 3.5 for CH2ORO, SNR = 40 dB, c=1

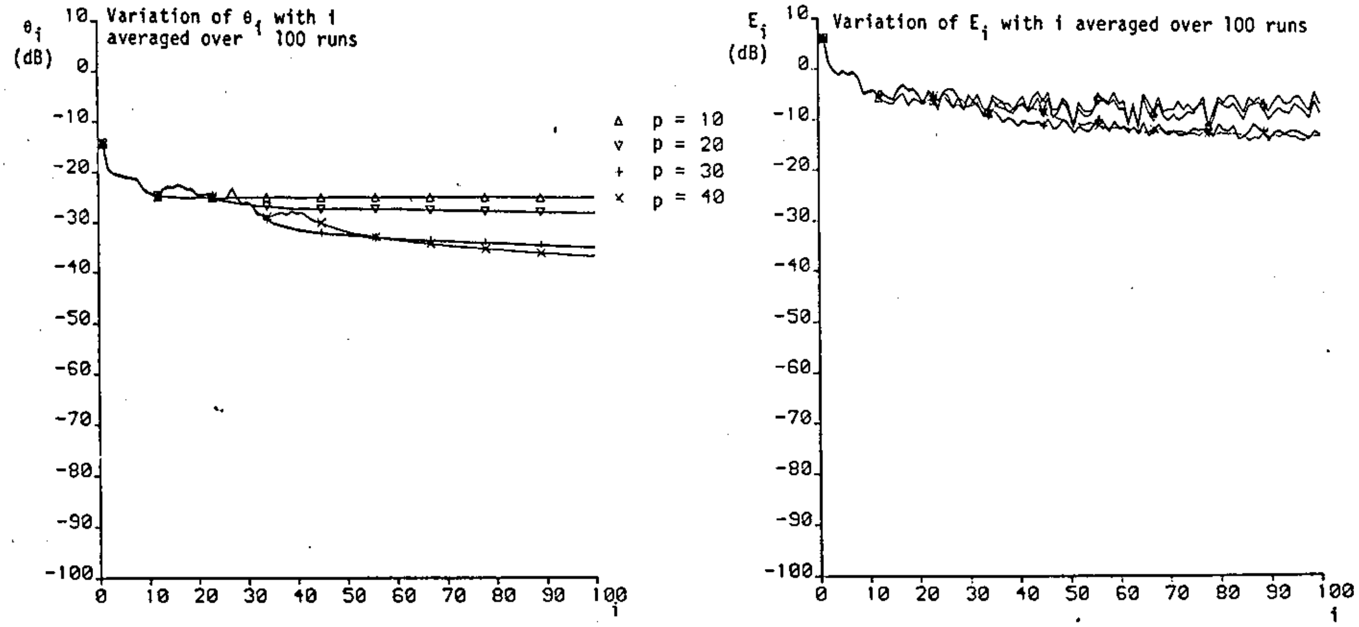


FIGURE 3.140: Performance of System 3.5 for CH2ORO, SNR = 40 dB, c=0.5

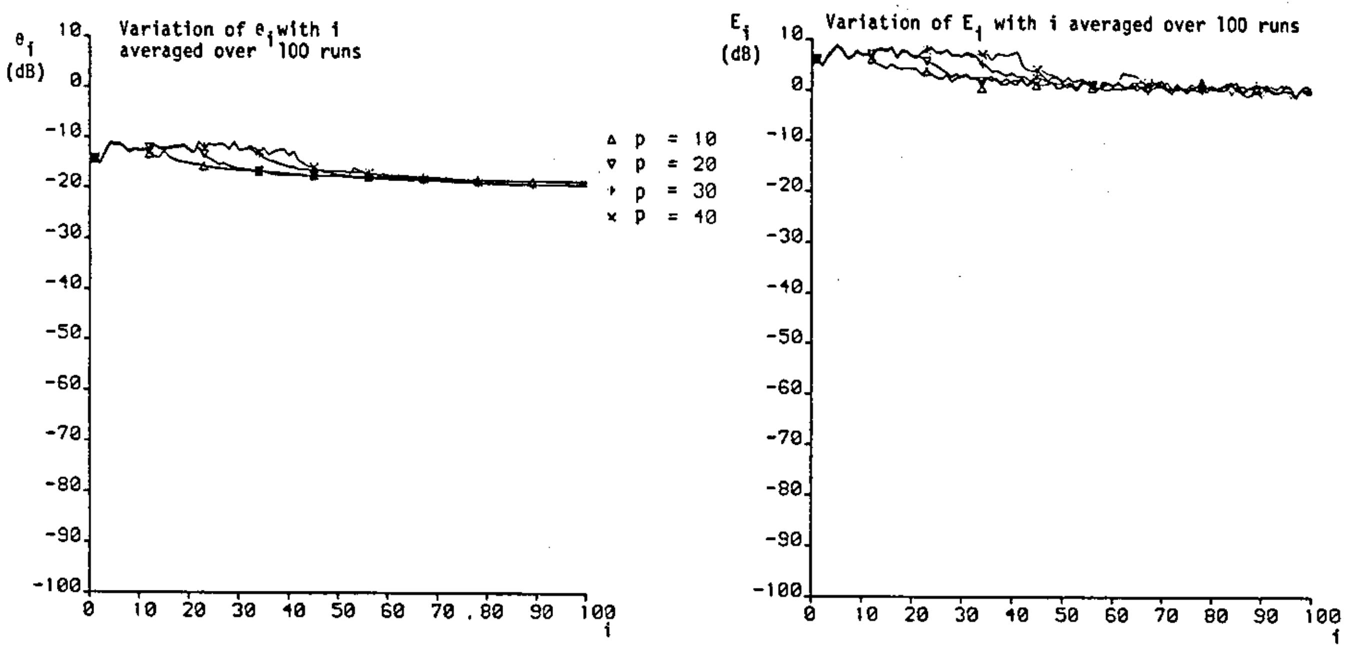


FIGURE 3.141: Performance of System 3.5 for CH2ORO, SNR = 30 dB, c=1

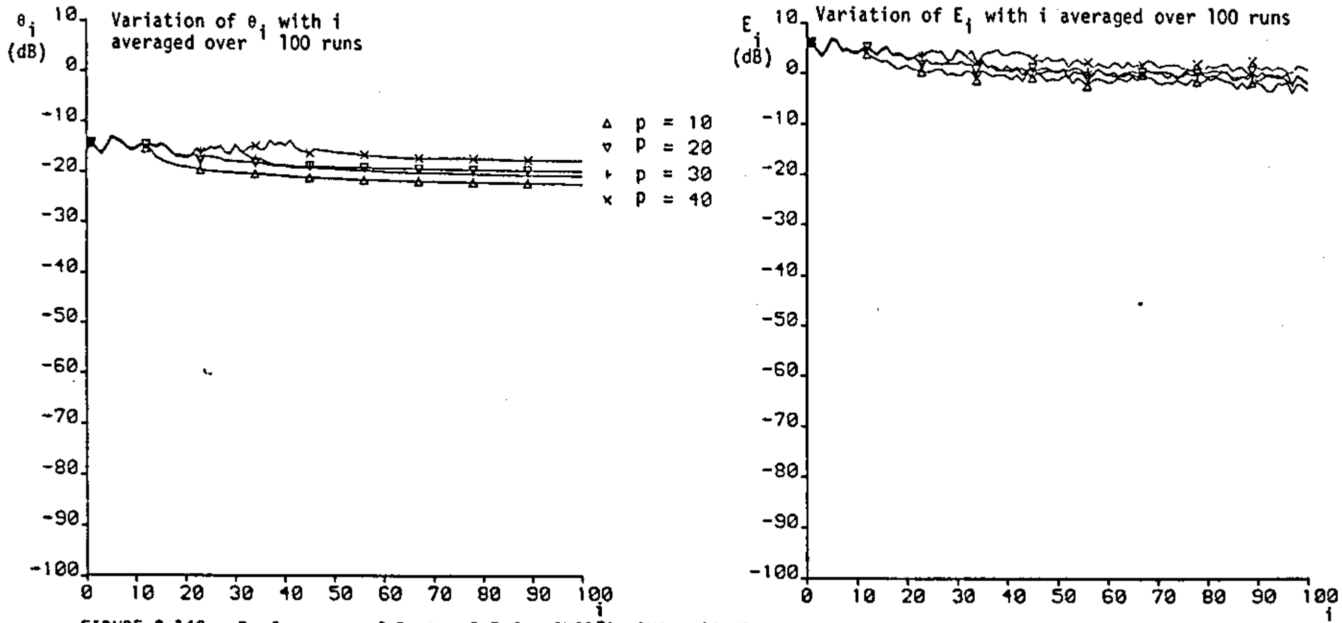


FIGURE 3.142: Performance of System 3.5 for CH2ORO, SNR = 30 dB, c=0.5

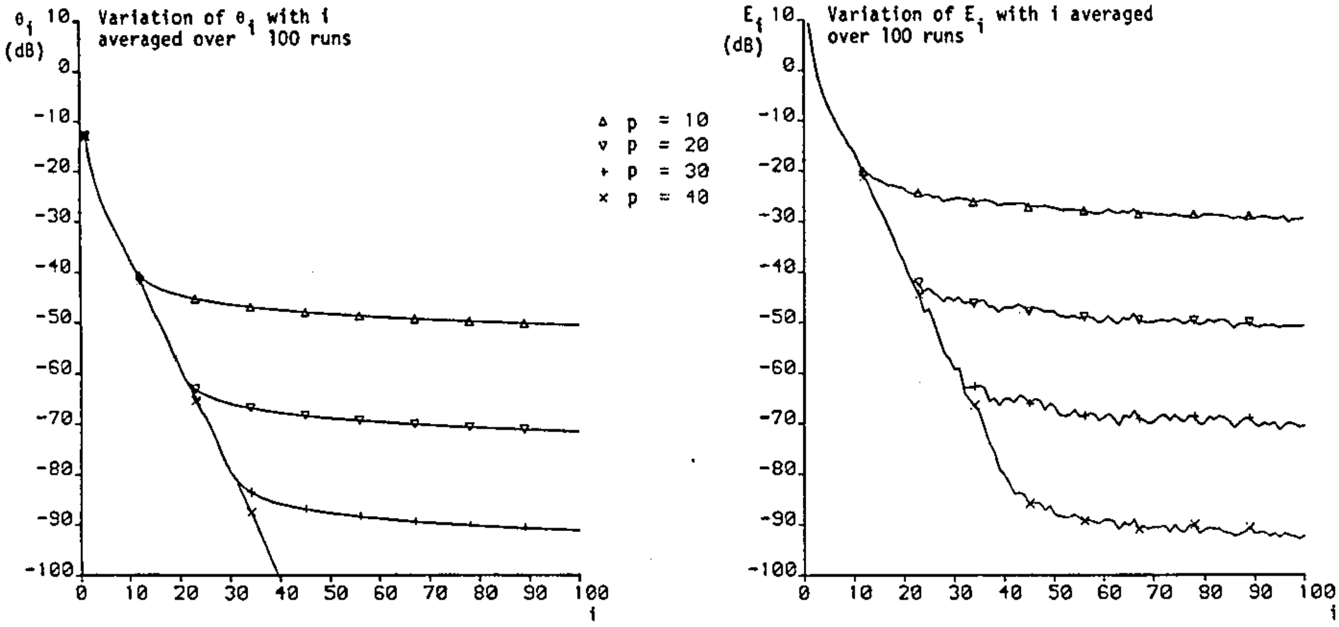


FIGURE 3.143: Performance of System 3.5 for CH19RO, SNR = ∞, c=1

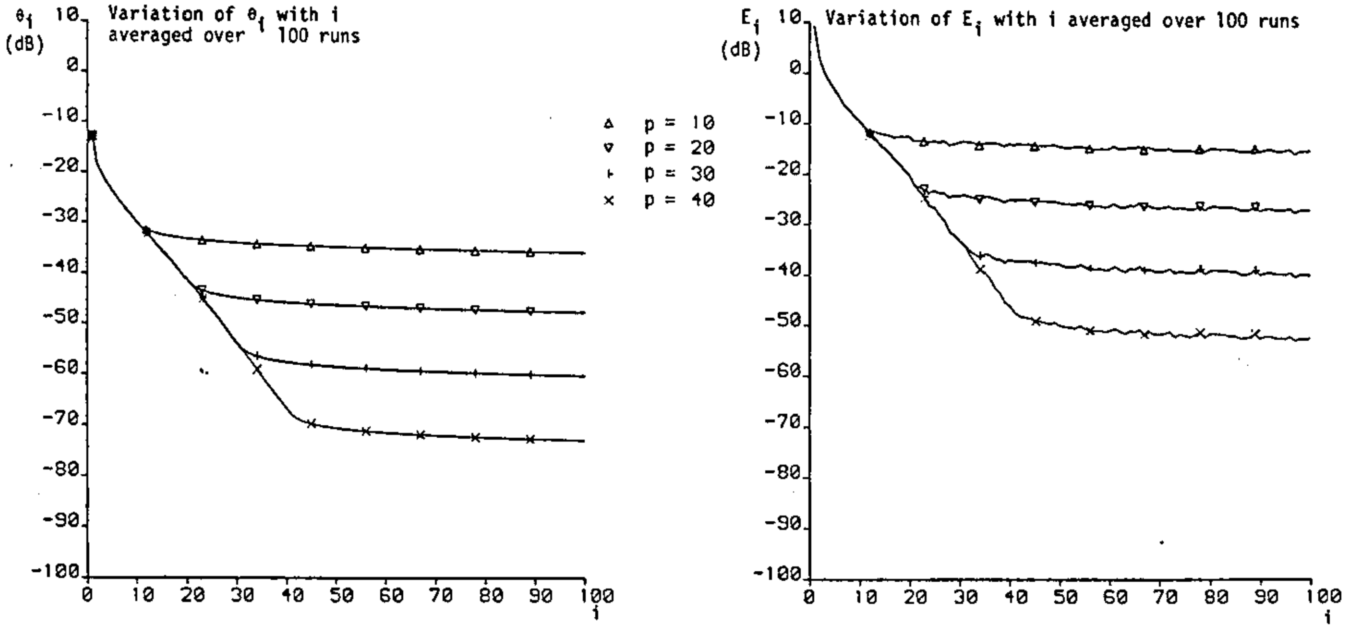


FIGURE 3.144: Performance of System 3.5 for CH19RO, SNR = ∞, c=0.5

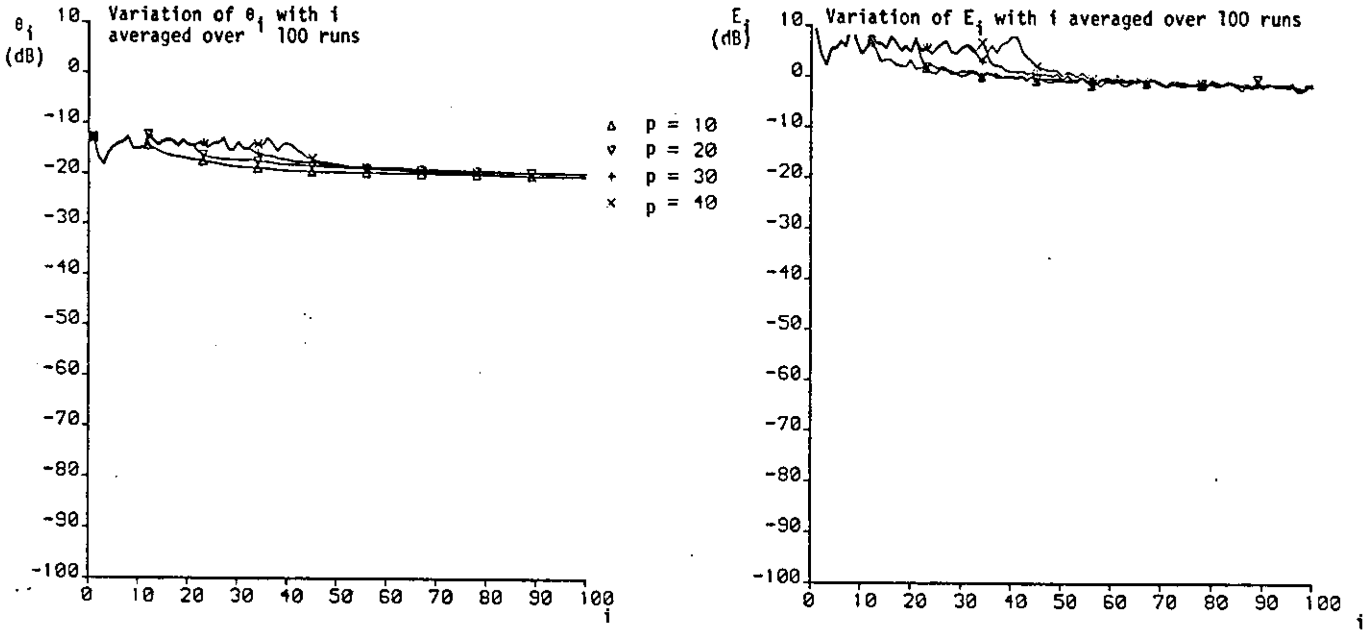


FIGURE 3.145: Performance of System 3.5 for CH19R0, SNR = 40 dB, c=1

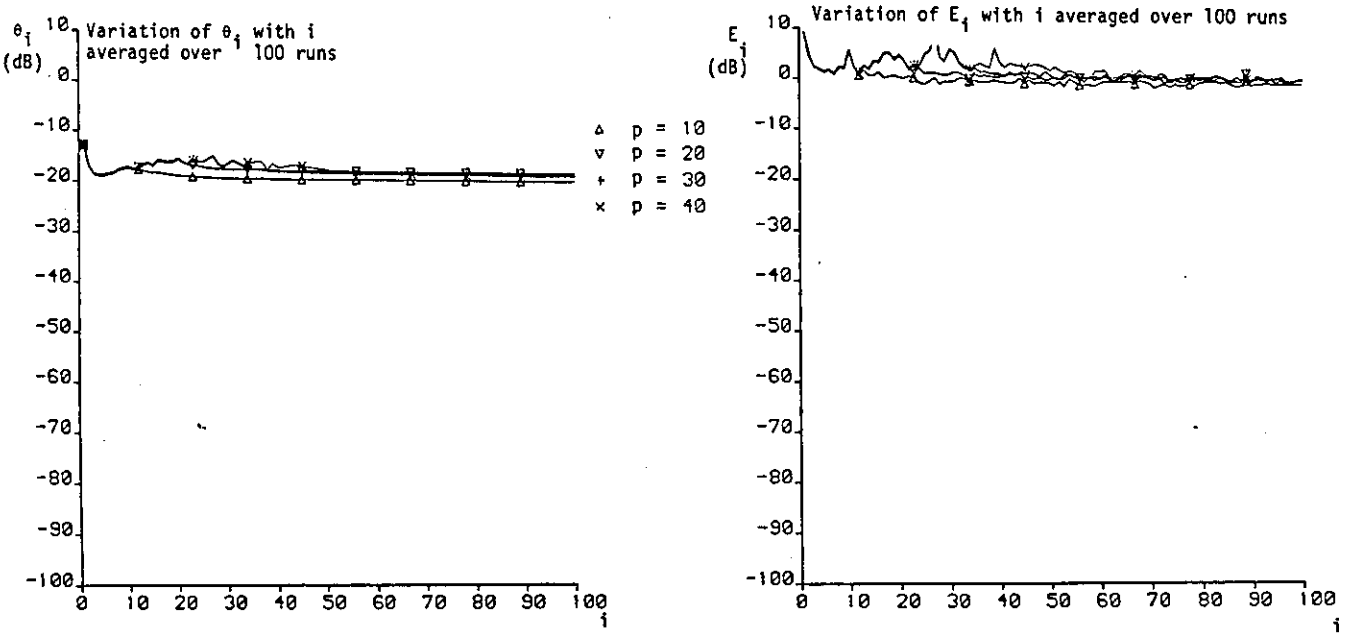


FIGURE 3.146: Performance of System 3.5, for CH19R0, SNR = 40 dB, c=0.5

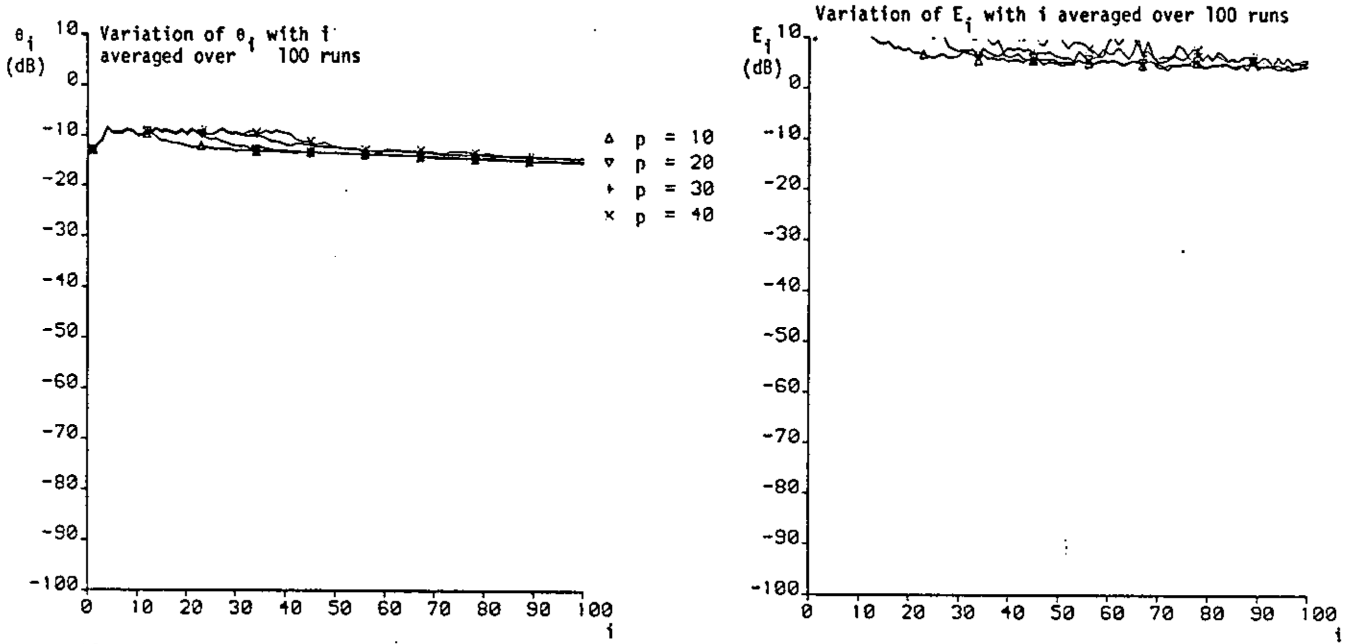


FIGURE 3.147: Performance of System 3.5 for CH19RO, SNR = 30 dB, c=1

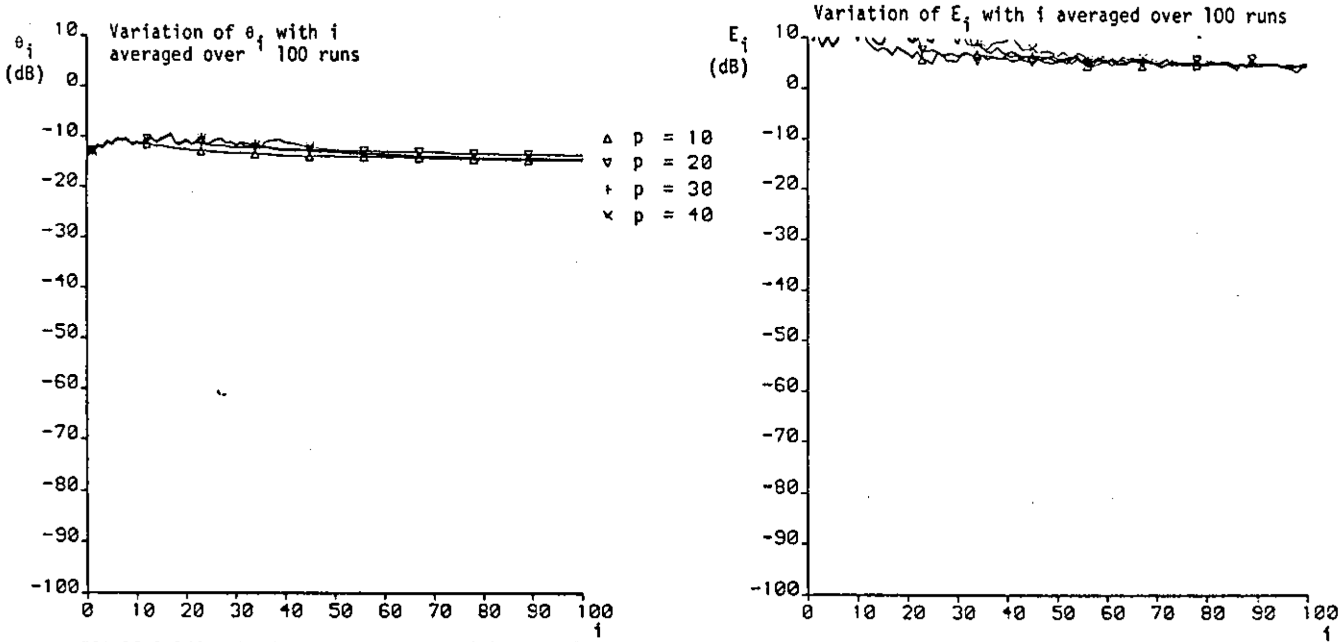


FIGURE 3.148: Performance of System 3.5 for CH19RO, SNR = 30 dB, c=0.5

3.3 Discussion

The purpose of this Chapter is to develop suitable on-line technique/s for the adjustment of the adaptive linear filter ahead of the detector which is essentially a problem of finding the roots (zeros) of the z-transform of the sampled impulse-response of a channel that lies outside the unit circle. In this mode of operation, the receiver operates on a continuous stream of received samples $\{r_i\}$ (equation 3.4) in the presence of both intersymbol interference and additive white Gaussian noise. It is assumed that the receiver has exact knowledge of the sampled impulse-response of the channel and the data-symbols $\{s_i\}$ are detected correctly. The receiver manipulates the received samples $\{r_i\}$ to give a stored sequence of $\{r_i'\}$ given by the (n+1)-component vector R_{i+n}' (equation 3.9) where (n+1) corresponds to the number of taps in the linear filter. The samples in R_{i+n}' are the most recent (n+1) samples of $\{r_i\}$ that arefree of intersymbol interference caused by the past data-symbols. This sequence of $\{r_i'\}$is operated on by the two-tap feedforward and the one-tap feedback filters in cascade as described in Section 3.1 so that a sequence $\{x_i\}$ (equation 3.58) appears at the output of the one-tap filter. The latter sequence forms the basis of all the root-finding algorithms described in this Chapter. The number n+1 is taken to be 40 throughout this Chapter unless otherwise stated.

All efforts have been concentrated on finding one of the m roots (zeros) of Y(z) outside the unit circle in isolation and no attempt has been made to develop the root-finding process into a continuous operation when m>1. Furthermore, it is always assumed that there is at least one root lying outside the unit circle and no study has been carried out to establish suitable criteria for the termination of the algorithm when a root has been found and when there are no more roots lying outside the unit circle. Thus, the systems described in the Chapter cannot be used in practical situations without further development and modifications.

The first part of this section describes the feasibility tests based on the model of a synchronous serial binary data-transmission system where four test channels are used in the investigation. Systems 3.1 to 3.4 are the step-by-step development of a suitable root-finding algorithm which is capable of finding one root (zero) of Y(z) in very simple cases where Y(z) has three real-valued coefficients and one root outside the unit circle. The performance of the algorithm is measured in terms of the mean-square errors in the estimate of β_1 (negative reciprocal of a root outside the unitcircle) and in the detected data-symbols, given more precisely by θ_i and E_i in equations 3.73 and 3.74.

In System 3.1, a root-finding algorithm is derived using the signals $\{x_i\}$ from the output of the one-tap feedback filter. It assumes that the estimate of β_1 is small so that λ_i^3 and higher orders of λ_i can be ignored and it also assumes the accurate knowledge of the parameter q_0, q_1 and q_2 (equations 3.76-3.78). Furthermore, it assumes the correct detection of s_i, s_{i-1}, s_{i-2}, ... and the knowledge of s_{i+1} and s_{i+2} which are not available at time t=(i+n)T. Therefore, the system is not realisable but it gives a crude indication of the particular virtue of such an arrangement. The results show that in the absence of noise, the roots can be found accurately and quickly except for Channel D (Table 3.1). By using stepsize c = 1.0, the roots can be found to an accuracy of better than -100 dB with 2 iterations. When the received samples $\{r_i\}$ are corrupted by additive white Gaussian noise of signal to noise ratio = 30 dB, the algorithm achieves -30 dB in both θ_i and E_i for Channels A and B but degrades noticeably to -20 dB for Channel C. The algorithm seems to fail with Channel D even with no noise and it is most likely to have been caused by the approximations made in its derivation and the large β_1 which is associated with the channel (= 0.7).

System 3.2 uses a much simplified algorithm where only the knowledge of the quantity q_0 is required and, of course, correct detection is again assumed. The results show that the rate of convergence is slower in the absence of noise particularly when small stepsizes are used. For larger stepsizes ($c \geqslant 0.75$), the difference between this and System 3.1 is less than 10 iterations at θ_i = -100 dB. The difference in speed is almost negligible at signal to noise ratio of 30 dB. Note that this degradation is more severe for channels with larger β_1 which suggests that the roots nearer to the unit circle are more difficult to handle. There is a definite improvement in the accuracy of the estimate for Channel C at 30 dB signal to noise ratio and, furthermore, no collapse is evident in the case of Channel D. With suitable stepsize such as $c = 0.5$, a reasonable compromise can be made between speed and accuracy of the operation.

Less prior knowledge of the channel is assumed in System 3.3 in that only the sign of q_0 (a function of y_0 and β_1 (equation 3.76)) is required and the rest of the design is exactly the same as System 3.2. Results show that the performance is very close to that of System 3.2. Significant differences occur only in Channels C and D where β_1 are large, and only when small stepsizes ($c \leqslant 0.25$) are used. For Channel D with $c = 0.1$ and signal to noise ratio of 30 dB there is an increase of less than 10 iterations at convergence when compared with System 3.1 where it is assumed that parameters q_0, q_1 and q_2 are known. The accuracy of the estimates at convergence remain approximately the same.

In the previous systems (Systems 3.1-3.3), in addition to the assumption of correct channel estimation and correct detection, some knowledge of the parameters q_0, q_1 and q_2 (equations 3.76-3.78) is assumed. These depend on the sampled impulse-response of the channel, the root $-\frac{1}{\beta_1}$ and its estimate $-\frac{1}{\lambda_i}$, and are generally unavailable at the receiver. System 3.4 uses an estimate of the parameter q_0 in the algorithm for estimating β_1 (equations 3.91-3.92). The estimate of q_0 is derived from the signals $\{x_i\}$ at the output of the feedback filter, all other quantities used in the algorithm are known at the receiver. In addition, to the set of stepsize ($c = 0.1, 0.25, 0.5, 1.0$) used previously, $c = \frac{1}{i}$, where i = the number of iterations, is also employed. This has the property of giving a better estimate in noisy situations[33]. The system is tested for no noise and signal to noise ratios of 30 and 20 dB. The results show that in the absence of noise, very accurate estimates of the root can be achieved (better -100 dB for Channels A, B and C and about -60 dB for Channel D) in a reasonably short time depending on the value of c. There is no difference in performance between this system and System 3.3 for Channel A, where β_1 is very small (= 0.1), in noisy conditions. However, a significant degradation is observed in the speed of operation which is more severe for channels with larger β_1 and for Channel D where β_1 = 0.7, this is about 10 iterations at signal to noise ratio of 30 dB. But the degradation in the performance is not severe especially when a large c is used. The results are very encouraging because the system requires only the correct channel estimate and correct detection in $\{s_i\}$ and that it performs very well for all four test channels, Channels A-D.

Having found a suitable algorithm (System 3.4) for the simple cases in the feasibility test, work is quickly moved onto the application of the algorithm and its further developments for the 16-point QAM data-transmission system operating at 9600 bit/s. Here, the components of the channel sampled impulse-response Y are complex-valued and therefore the roots (zeros) of Y(z) are no longer necessarily real or occur in complex-conjugate pairs. Channels 1 and 3 (Table 2.1) are used for this part of the investigation which represent an average line and a typical worst line in the British public switched telephone network at 9600 bit/s. The algorithms are tested for their ability of founding one of the m roots of Y(z) that lie outside the unit circle and their performance is measured in terms of θ_i and E_i (equations 3.73 and 3.74). In order to do this, the 'raw' sampled impulse-responses of Channels 1 and 3 are pre-processed to give a set of resultant sampled impulse-responses which have 1, 2, ..., (m-1) of their m roots outside the unit circle removed and replaced by their complex conjugate-reciprocals. Thus, for each telephone circuit, there are m sampled impulse-responses (including the initial one) representing different stages of the process during which the channel is being converted into minimum phase.

Section 3.2.1 describes the application of System 3.4 to the QAM system where the algorithm is expanded to allow different values for the stepsizes (c_1 and c_2) for the estimation of β_1 and q_0 (equations 3.103 and 3.104). The estimation of q_0 is used in the estimation of β_1 (negative reciprocal of the root). A detailed study is carried out to find the effect and the optimum combination of c_1 and c_2 on the speed and accuracy of the root-finding process. The results of computer-simulation tests show that the algorithm performs very well in the absence of noise and depending on the values of c_1 and c_2, accuracy of -100 dB in θ_i and E_i can be achieved within 50 iterations. The effect of c_1 and c_2 are best illustrated in Figures 3.66 and 3.80 where the more influential nature of c_2 is clearly shown. In the former case, it can be seen that for $c_1 = 1$, the difference in the number of iterations taken for the algorithm to reach -100 dB in both θ_i and E_i is 50 when c_2 is changed from 0.25 to 1.0, and in the latter case, when $c_2 = 1$ and c_2 is changed from 0.25 to 1.0 this difference is only about 10. Generally, c_2 has to be set fairly large ($\geqslant$ 0.5) for fast convergence and once set to such a value, no great advantage can be gained for having c_1 differently from c_2. The fastest rate of convergence is, of course, when $c_1 = c_2 = 1$.

When the received samples $\{r_i\}$ are corrupted by additive white Gaussian noise, it is again demonstrated that c_2 is the more critical factor but tests have shown that there is no real advantage in setting c_1 differently from c_2. It is found that, in general, the more roots there are outside the unit circle, the less accurately will these roots be located. For example, at 40 dB signal to noise ratio, the best value of θ_i is about -32 dB (Figures 3.72-3.75), -40 dB (Figures 3.48-3.51) and -45 dB (Figures 3.96-3.99) where there are 3 roots, 2 roots, 1 root lying outside the unit circle for Channel 1, respectively. Furthermore, for the more severely distorted channel, Channel 3, the error in locating β_1 is very large and quite unacceptable (-20 dB) when starting from the raw channel, even at high signal to noise ratio of 40 dB (Figure 3.124). However, this improves to about -40 dB when there is only one root left outside the unit circle (Figure 3.12) whilst the others are assumed to have been removed and replaced by their complex conjugate-reciprocals. It is also found that the roots themselves and the accuracy at which they are located are dependent on the particular sequence of received samples $\{r_i\}$ and that for a fixed stepsize c and for a particular root in question, the difference in speed can be as large as 70 iterations (Figure 3.119).

The results suggest that the algorithm is not suitable for the QAM system at 9600 bit/s because of the problems of accuracy in the estimate of β_1 even at high signal to noise ratios and also its dependence on the sequence of received samples $\{r_i\}$.

System 3.5 is an attempt to improve the accuracy in the estimate of β_1 in the presence of additive white Gaussian noise. The algorithm is divided into two parts and the idea is that a rough estimate of β_1 can be found reasonably quickly by the first part of the algorithm so that the second part can be brought in to smooth the noise. The action of the second part of the algorithm is similar to that of an averaging process where even weightings are given to all previous estimates of β_1. Different change-over points at which to commence the second part of the algorithm and different values of K_p (parameter at change-over) are tested in cases where signal to noise ratios are ∞, 40 dB and 30 dB. Again $c_1 = c_2 = c$ (equations 3.105-3.106) and its value is tested from 0.25 to 1 in steps of 0.25. The results have shown that in the presence of noise the system is no better than that of System 3.4 and that the system is not capable of dealing with severely distorted channels such as Channel 3.

Minor modifications have been made to the algorithms but it is clear that the accuracy of the estimate is greatly affected by the presence of additive white Gaussian noise and that the errors caused in the earlier stages of the minimum phase process will, if allowed to filter through, accumulate and eventually cause the whole system to collapse in poor conditions.

CHAPTER 4

THE OFF-LINE SYSTEM

It is evident, from the results presented in Chapter 3, that the online system (Figure 3.43), in which the receiver operates directly on a continuous stream of received samples $\{r_i\}$ subject to both intersymbol interference and noise, do not provide the satisfactory solution for the adjustment of the adaptive filter in Figure 2.1. The major factor which inhibits the successful operation of the many techniques developed for the on-line system is that of the additive white Gaussian noise. Although techniques have been developed for cases where the received samples $\{r_i\}$ are free of noise and work extremely well giving errors of better than -100 dB in the estimation of the roots, but when the received samples $\{r_i\}$ are corrupted by additive white Gaussian noise, all attempts to produce a reliable and satisfactory root-finding algorithm for the 16-point QAM system for poor channels have been unsuccessful. The algorithms so far tested have failed to give the necessary accuracy in the estimate even at a relatively high signal to noise ratio of 30 dB and performance of the algorithms degrades as the number of roots (zeros) of the z-transform of the sampled impulse-response of the channel outside the unit circle increases. Furthermore, when there are more than one root outside the unit circle, the order in which the roots are located is dependent to a certain extent on the particular sequence of received samples $\{r_i\}$, and for a given root the difference in the speed of the algorithm can be very large, more than twice as much (from 30 iterations to 100 iterations at θ_i = -60 dB) as shown in Figure 3.119.

The alternative approach, now to be considered, is one where the receiver operates solely and directly on the estimate of the sampled impulse-response of the channel which is provided by the channel estimator in Figure 2.1. [This estimate can be obtained by a number of different methods[30-34] and is essentially free of additive noise.]

Such a system is potentially more attractive because it is an off-line system which operates on a set of noise-free samples (the components of Y in equation 2.3), and therefore the speed of the root-finding operation is only limited by the processingpower of the hardware and the accuracy is not directly affected by the noise in the received samples $\{r_i\}$. Furthermore, since it is independent of both the input samples $\{s_i\}$ and the received samples $\{r_i\}$ (Figure 4.1), the amount of computation is considerably less. The approach is possible because, in practice, the sampled impulse-response of the channel Y, which is the only information required by the root-finding process, is provided at the receiver for use in the detection process.

4.1 System 4.1

The operation of this system is similar to the on-line system (Chapter 3) in that it also involves the combined operation of the two-tap feedforward transversal filter (whose z-transform $B_i(z)$) and the one-tap feedback transversal filter (whose z-transform $A_i(z)$) as shown in Figure 4.1. From equations 2.18 and 2.19, the z-transforms of the two-tap feedforward filter and the one-tap feedback filter are

$$B_i(z) = 1 + \lambda_i^* z^{-1} \tag{4.1}$$

and

$$A_i(z) = (1 + \lambda_i z^{-1})^{-1} \tag{4.2}$$

respectively. In this case, instead of operating on the continuous sequence of received samples $\{r_i\}$, the receiver operates solely and directly on the sampled impulse-response of the channel, Y (equation 2.3). The sequence Y (whose z-transform Y(z)) ; assumed to be provided accurately at the receiver, is fed through the two-tap feedforward

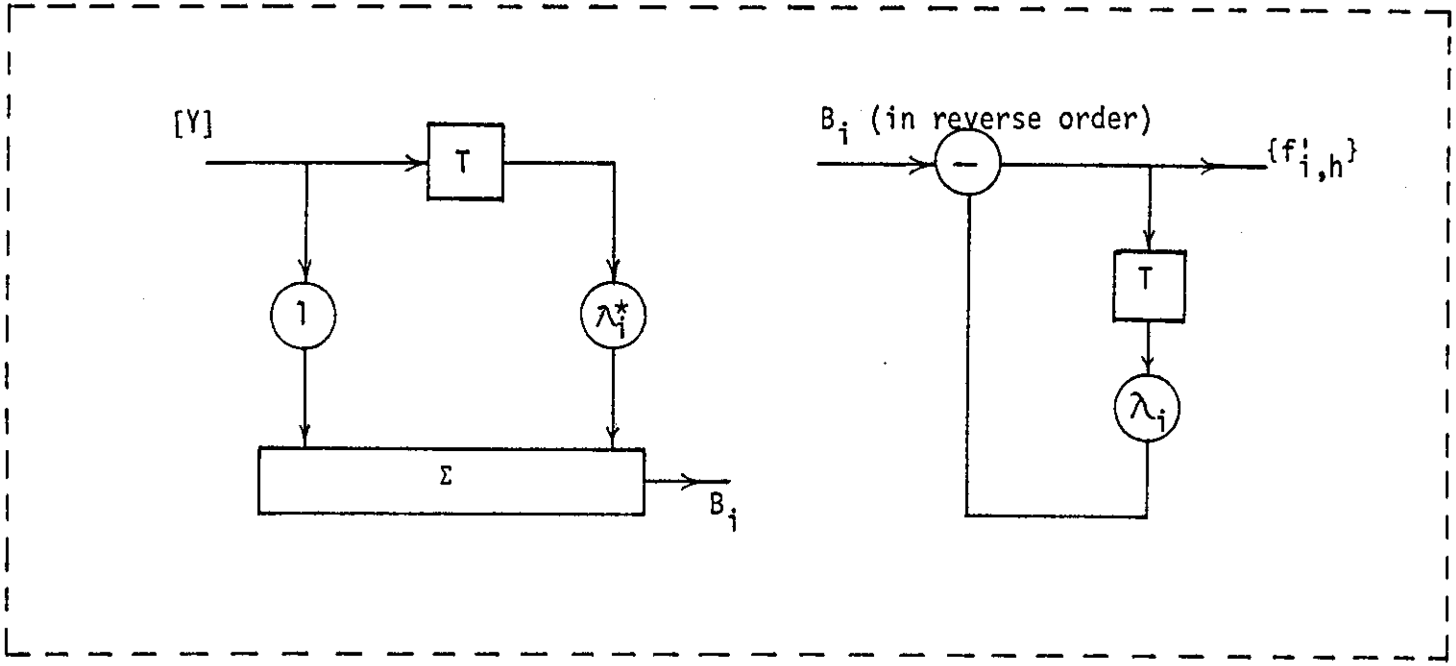


FIGURE 4.1: The combined operation of the two filters in System 4.1

filter, starting with the component y_o, to give the resultant sequence $\{b_{i,h}\}$, h=0,1,...,g+1 which is held in store.

Let the z-transform of the sampled impulse-response of the channel be

$$Y(z) = y_o + y_1 z^{-1} + \dots + y_g z^{-g} \qquad (4.3)$$

The resultant z-transform of the channel and the two-tap feedforward filter is

$$Y(z)\ B_i(z) = (y_o + y_1 z^{-1} + \dots + y_g z^{-g})(1 + \lambda_i^* z^{-1})$$

$$= y_o + (y_1 + \lambda_i^* y_o) z^{-1} + (y_2 + \lambda_i^* y_1) z^{-2} + \dots + \lambda_i^* y_g z^{-g-1}$$

$$(4.4)$$

So that, if

$$B_i' = [b_{i,o} \quad b_{i,1} \ \cdot \ \cdot \ \cdot \ b_{i,g+1}] \qquad (4.5)$$

is the output sequence, then

$$b_{i,o} = y_o \qquad (4.6)$$

$$b_{i,1} = y_1 + \lambda_i^* y_o \qquad (4.7)$$

$$\vdots$$

$$b_{i,g} = y_g + \lambda_i^* y_{g-1} \qquad (4.8)$$

$$b_{i,g+1} = \lambda_i^* y_g \tag{4.9}$$

In general,

$$b_{i,h} = y_h + \lambda_i^* y_{h-1} \tag{4.10}$$

where $0 \leqslant h \leqslant g+1$ and it is assumed that $y_h = 0$ for $0 > h > g$.

The output sequence B_i' is then reversed in order, so that it begins with the component $b_{i,g+1}$ and finishes with $b_{i,o}$ and is then fed through the one-tap feedback filter. The sequence B_i', passing through the filter in reverse order as described, is taken to be moving backwards in time, and a delay of one sampling interval T in the feedback filter now becomes an advance of T, with z-transform z. Thus, the effective z-transform of the one-tap feedback filter becomes

$$A_i(z) = (1 + \lambda_i z)^{-1} \tag{4.11}$$

Suppose now from equation 4.3,

$$\begin{aligned} Y(z) &= y_o + y_1 z^{-1} + \dots\dots + y_g z^{-g} \\ &= (1 + \beta_1 z)(u_o z^{-1} + u_1 z^{-2} + \dots + u_{g-1} z^{-g}) \end{aligned} \tag{4.12}$$

where $-\frac{1}{\beta_1}$ is the root that lies outside the unit circle in the z-plane. Now, the resultant z-transform of the channel and the two filters is

$$
\begin{aligned}
Y(z)\ A_i(z)\ B_i(z) &= (1+\beta_1 z)(u_o' z^{-1} + u_1 z^{-2} + \ldots + u_{g-1} z^{-g})(1+\lambda_i z)^{-1}(1+\lambda_i^* z^{-1}) \\
&= z^{-1}(1+\lambda_i z)^{-1}(1+\beta_1 z)(1+\lambda_i^* z^{-1})(u_o+u_1 z^{-1}+\ldots+u_{g-1} z^{-g+1})) \\
&= z^{-1}(1+\lambda_i z)^{-1}(1+\beta_1 z)(u_o+(u_1+\lambda_i^* u_o)z^{-1}+\ldots+(u_{g-1}+\lambda_i^* u_{g-2})z^{-g+1} \\
&\qquad + \lambda_i^* u_{g-1} z^{-g}) \\
&= z^{-1}(1-\lambda_i z+\lambda_i^2 z^2- \ldots)(1+\beta_1 z)(q_o+q_1 z^{-1}+\ldots+q_g z^{-g}) \\
&= z^{-1}(1+(\beta_1-\lambda_i)z-\lambda_i(\beta_1-\lambda_i)z^2+\ldots)(q_o+q_1 z^{-1}+\ldots+q_g z^{-g}) \\
&= z^{-1}(\ldots+(\qquad -\lambda_i(\beta_1-\lambda_i)q_o+\lambda_i^2(\beta_1-\lambda_i)q_1-\ldots)z^2 \\
&\qquad +(\quad (\beta_1-\lambda_i)q_o-\lambda_i(\beta_1-\lambda_i)q_1+\lambda_i^2(\beta_1-\lambda_i)q_2-\ldots)z^1 \\
&\qquad +(q_o+(\beta_1-\lambda_i)q_1-\lambda_i(\beta_1-\lambda_i)q_2+\lambda_i^2(\beta_1-\lambda_i)q_3-\ldots)z^0 \\
&\qquad +(q_1+(\beta_1-\lambda_i)q_2-\lambda_i(\beta_1-\lambda_i)q_3+\lambda_i^2(\beta_1-\lambda_i)q_4-\ldots)z^{-1} \\
&\qquad +(q_2+(\beta_1-\lambda_i)q_3-\lambda_i(\beta_1-\lambda_i)q_4+\lambda_i^2(\beta_1-\lambda_i)q_5-\ldots)z^{-2} \\
&\qquad + \ldots \\
&\qquad + q_g z^{-g}) \qquad (4.13)
\end{aligned}
$$

where

$$q_o = u_o' \qquad (4.14)$$

$$q_1 = u_1 + \lambda_i^* u_o \tag{4.15}$$

$$\vdots$$

$$q_{g-1} = u_{g-1} + \lambda_i^* u_{g-2} \tag{4.16}$$

$$q_g = \lambda_i^* u_{g-1} \tag{4.17}$$

In general,

$$q_h = u_h + \lambda_i^* u_{h-1} \tag{4.18}$$

where $0 \leqslant h \leqslant g$, and $u_h = 0$ for $0 > h \geqslant g$.

Let

$$Y(z)A_i(z)B_i(z) = \ldots + f'_{i,-1}z^1 + f'_{i,o} + f'_{i,1}z^{-1} + \ldots + f'_{i,g+1}z^{-g-1} \tag{4.19}$$

where $\{f'_{i,h}\}$, for $-\infty < h \leqslant g+1$, is the sequence at the output of the two filters (connected in cascade (Figure 4.1)) when the sequence Y (whose z-transform Y(z)) is fed through them. Now,

$$f'_{i,-1} = \quad -\lambda_i(\beta_1-\lambda_i)q_o + \ldots + (-\lambda_i)^{g+1}(\beta_1-\lambda_i)q_g \tag{4.20}$$

$$f'_{i,o} = (\beta_1-\lambda_i)q_o - \lambda_i(\beta_1-\lambda_i)q_1 + \ldots + (-\lambda_i)^g (\beta_1-\lambda_i)q_g \tag{4.21}$$

$$f'_{i,1} = q_o + (\beta_1-\lambda_i)q_1 - \lambda_i(\beta_1-\lambda_i)q_2 + \ldots + (-\lambda_i)^{g-1}(\beta_1-\lambda_i)q_g \tag{4.22}$$

$$f'_{i,2} = q_1+(\beta_1-\lambda_i)q_2-\lambda_i(\beta_1-\lambda_i)q_3 + \dots +(-\lambda_i)^{g-2}(\beta_1-\lambda_i)q_g \tag{4.23}$$

$$\vdots$$

$$f'_{i,g+1} = q_g \tag{4.24}$$

In practice only g+2 components of the sequence of $\{f'_{i,h}\}$ are generated by the receiver and the iterative process operates entirely on the g+2 components $f'_{i,o}$, $f'_{i,2} \dots f'_{i,g+1}$.

Clearly, when $\lambda_k = \beta_1$,

$$\begin{aligned} f'_{k,h} &= 0 \quad \text{for} \quad h \leqslant 0 \\ &= q_{h-1} \quad \text{for } 0 < h \leqslant g+1 \end{aligned} \tag{4.25}$$

and

$$\begin{aligned} Y(z)A_i(z)B_i(z) &= q_o z^{-1} + q_1 z^{-2} + \dots + q_g z^{-g-1} \\ &= u_o z^{-1} + (u_1+\lambda_k^* u_o)z^{-2} + \dots + (\lambda_k^* u_{g-1})z^{-g-1} \\ &= u_o z^{-1} + (u_1+\beta_1^* u_o)z^{-2} + (u_2+\beta_1^* u_1)z^{-3} + \dots + (\beta_1^* u_{g-1})z^{-g-1} \\ &= (u_o z^{-1} + u_1 z^{-2} + \dots + u_{g-1}z^{-g})(1+\beta_1^* z^{-1}) \end{aligned} \tag{4.26}$$

which means that the root (zero), $-\frac{1}{\beta_1}$, in Y(z) has been removed and replaced by the complex conjugate of its reciprocal, $-\beta_1^*$, by the action of the filters (Figure 4.1).

Consider the case where $\lambda_i \rightarrow \beta_1$, from equations 4.21-4.23

$$f'_{i,o} \simeq (\beta_1-\lambda_i)q_o \simeq 0 \tag{4.27}$$

$$f'_{i,1} \simeq q_o \tag{4.28}$$

$$f'_{i,2} \simeq q_1 \tag{4.29}$$

$$\vdots$$

$$f'_{i,g} \simeq q_{g-1} \tag{4.30}$$

$$f'_{i,g+1} \simeq q_g \tag{4.31}$$

In general,

$$\begin{aligned} f'_{i,h} &\simeq 0 \quad \text{for } h \leqslant 0 \\ &\simeq q_{h-1} \text{ for } 0 < h \leqslant g+1 \end{aligned} \tag{4.32}$$

Substitute equation 4.28 into 4.27

$$f'_{i,o} \simeq (\beta_1-\lambda_i)f'_{i,1}$$

$$(\beta_1-\lambda_i) \simeq \frac{f'_{i,o}}{f'_{i,1}} \tag{4.33}$$

which means that $\frac{f'_{i,o}}{f'_{i,1}}$ is an estimate of $(\beta_1-\lambda_i)$. So that if λ_i is the original estimate of β_1,

$$\lambda_{i+1} = \lambda_i + \frac{f'_{i,o}}{f'_{i,1}} \tag{4.34}$$

is a better estimate than λ_i. Thus it suggests the following algorithm,

$$\lambda_{i+1} = \lambda_i + c \frac{f'_{i,o}}{f'_{i,1}} \tag{4.35}$$

where c is a small positive constant.

At the beginning of the iterative process (i=0), the receiver holds in store the sequence Y (equation 2.3) and an initial estimate of the quantity β_1 in equation 4.12 which is $\lambda_o=0$. The sequence Y is fed through the two-tap feedforward filter, whose tap is now set to $\lambda_o=0$, starting with the first component y_o, to give an (g+2) output sequence B_i' (equation 4.5). The sequence B_i' is then reversed in order, so that its first component is $b_{i,g+1}$ and its last component is $b_{i,o}$, and fed through the one-tap feedback filter with its tap set to $\lambda_o^* =0$ to give an (g+2) output sequence $\{f'_{i,h}\}$. These are, in the order in which they are generated, $f'_{i,g+1}$, $f'_{i,g}$, ..., $f'_{i,o}$, and the samples $f'_{i,o}$ and $f'_{i,1}$ are used in the manner given by equation 4.35 to give a new estimate of β_1.

The new estimate λ_{i+1} is now used in place of λ_i and the whole process is repeated as described until a change-over point (i=p) is reached, where an optional 'averaging' process is available. The 'averaging' process is the same as that described in Section 3.2.2 where it was found that in System 3.5 when the estimate λ_i experienced oscillatory behaviour the introduction of this averaging process at the appropriate time enabled a reduction in the fluctuations and produced a more smooth final estimate of β_1. The algorithm for updating λ_i is best understood with the aid of the flow diagram in Figure 4.2 which shows clearly the two different modes of operation. At every iteration, a new and improved estimate is obtained using the following equation

$$\lambda_{i+1} = \lambda_i + c \frac{f'_{i,o}}{f'_{i,1}}$$

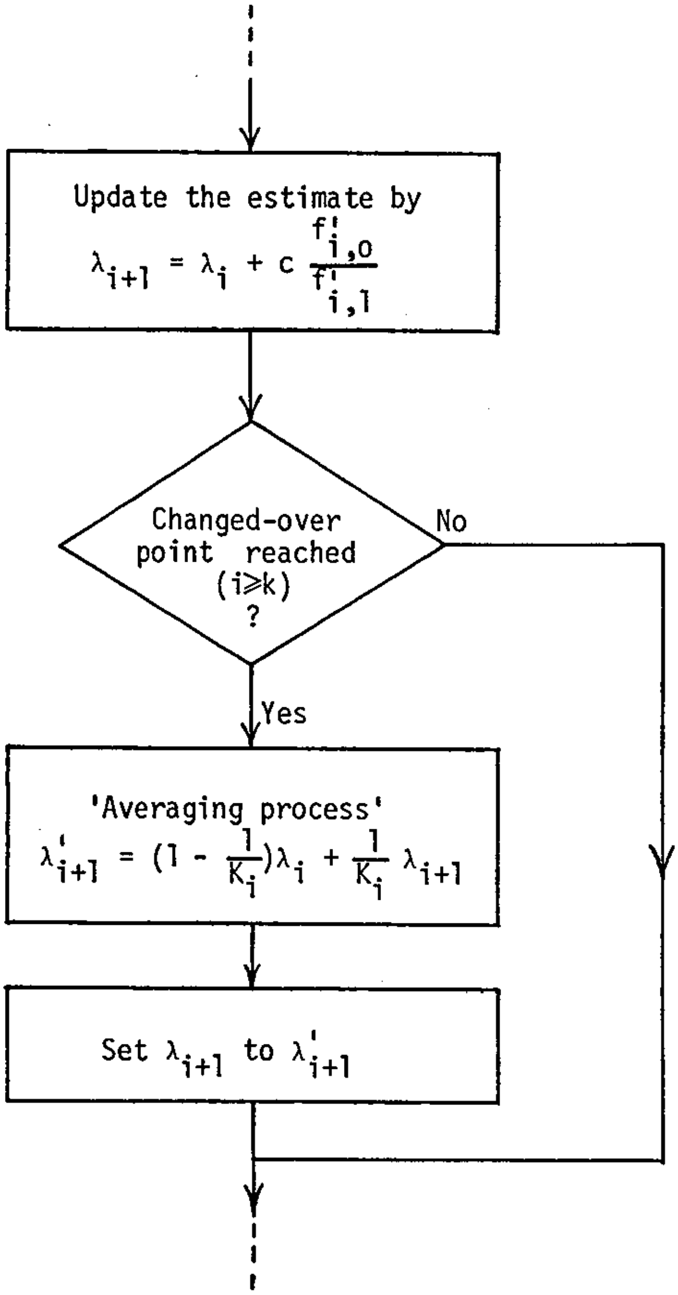

Update the estimate by
$\lambda_{i+1} = \lambda_i + c\,\frac{f'_{i,o}}{f'_{i,1}}$
Changed-over point reached ($i \geqslant k$) ?
No
Yes
'Averaging process'
$\lambda'_{i+1} = (1 - \frac{1}{K_i})\lambda_i + \frac{1}{K_i}\lambda_{i+1}$
Set λ_{i+1} to λ'_{i+1}

FIGURE 4.2: Flow-diagram which shows the algorithm for the estimation of β_1 with an optional 'averaging' process

where $\lambda_o = 0$ and c is a small positive constant. Starting from the p^{th} iteration, an extra step which corresponds to the 'averaging' process is carried out, so that for $i \geqslant p$, a computation step is performed as follows

$$\lambda'_{i+1} = (1 - \frac{1}{K_i})\lambda_i + \frac{1}{K_i}\lambda_{i+1} \tag{4.36}$$

where $K_i = 2,3,4,\ldots$ and $i = p, p+1, p+2 \ldots$. The final value λ'_{i+1} is then taken to be the estimate λ_{i+1} and is used for the next iteration.

Experiments are based on the 16-point QAM system transmitting at 9600 bit/s and initial investigations are carried out on Channels 1 and 3 (Table 2.1) which represent a moderate and a typical worst circuit in the British public switched telephone network for the transmission of data at 9600 bit/s, respectively. The receiver operates on Y (equation 2.3) using the algorithm described, and for the purpose of this investigation, the computer-simulation tests are carried out for 100 iterations for each root so as to ascertain the speed and accuracy of this root-finding technique and the difficulties associated with it. As in Chapter 3, the performance of the algorithm for the finding of only one of the m roots of Y(z) outside the unit circle is tested. To test the performance of the algorithm for a channel it is necessary to simulate the situation whereby the algorithm operates first on Y to find one of the m roots to some predetermined accuracy then removes and replaces it by its complex conjugate-reciprocal (which results in a new Y whose z-transform has (m-1) roots outside the unit circle), then continues, each time using the appropriate Y, until all the roots outside the unit circle have been found. To do this, the roots (zeros) of Y(z) are calculated independently by the NAG subroutine C02ADF[50] and later used as a reference. The root-finding algorithm then operates on Y, and each time a root is located by the algorithm its value is checked against the reference set. When a valid root (zero of Y(z) outside the unit circle) is located, i.e. $\lambda_k \simeq \beta_h$, where k is a positive integer and β_h is one of the m roots outside the unit circle, a new polynomial Y'(z) is formed using the exact value of β_h, previously determined by the root-finding NAG subroutine C02ADF[50], such that Y'(z) is equal to Y(z) with the root $-\frac{1}{\beta_h}$ removed and replaced by its complex conjugate-reciprocal $-\beta_h^*$, and

$$Y'(z) = Y(z)\ (1 + \beta_h z)^{-1}(1 + \beta_h^* z^{-1}) \tag{4.37}$$

Y'(z) is, of course, calculated separately and is independent of the root-finding algorithm. The new sequence Y' (whose z-transform Y'(z)) is then used by the receiver for the finding of further, if any, roots outside the unit circle. In the tests described here and in the next section (Section 4.2), the following notation (Table 4.1) is adopted in order to indicate which of the Y' is being operated on by the root-finding algorithm.

CH20R0	-	Channel 1 without any root processed
CH20R1	-	Channel 1 with root 1 processed
CH20R2	-	Channel 1 with roots 1 and 2 processed
CH18R0	-	Channel 2 without any root processed
CH18R1	-	Channel 2 with root 1 processed
CH18R2	-	Channel 2 with roots 1 and 2 processed
CH18R3	-	Channel 2 with roots 1, 2 and 3 processed
CH19R0	-	Channel 3 without any root processed
CH19R1	-	Channel 3 with root 1 processed
CH19R2	-	Channel 3 with roots 1 and 4 processed
CH19R3	-	Channel 3 with roots 1, 4 and 3 processed

TABLE 4.1: Notation adapted to indicate different stages of Y(z) during the process of phase minimization

The performance of the algorithm is measured by the error in λ_i, obtained when processing β_h and is given by

$$\theta_i = 10 \log_{10}(|\beta_h - \lambda_i|^2) \tag{4.38}$$

Results of computer-simulation tests on Channels 1 and 3 are presented in Figures 4.3 and 4.4 respectively, which give the variation of θ_i with i for 100 iterations (i=100). These show that, for the first time in this investigation, all the roots (zeros) of Y(z) that lie outside the unit circle are located for both Channel 1 and Channel 3. More importantly, with no 'averaging' process and a stepsize of 1, any individual roots that lie outside the unit circle in the z-plane for the two channels are located in less than 60 iterations with an error θ_i (equation 4.38) of better than -60 dB at convergence. Furthermore, the results of Channel 3 (Figure 4.4) show clearly that the roots are not located in any particular order (this being root 1, root 4, root 3 and root 2), which agree with the results obtained in the on-line system. When dealing with Channel 2, the results in Figures 4.5 and 4.6 show that smaller values of c such as 0.9, 0.75, 0.50 and 0.25 were necessary for locating the first root (zero) of Y(z) (which happened to be the largest of the four roots), but for the remaining three roots, c=1.0 can be used to give the correct results. The results, so far, are encouraging because it appears that the algorithm can be used for the 16-point QAM channels operating at 9600 bit/s, and because there is no 'averaging' process involved, the speed of the algorithm is at its fastest.

Consider now Channel 4 (Figure 2.7) which is the baseband equivalent of a severely distorted and a typical worst telephone circuit in the British public switched telephone network normally considered for the transmission rate of 1200 bit/s. Let us consider the algorithm as given in equation 4.35

$$\lambda_{i+1} = \lambda_i + c\,\frac{f'_{i,0}}{f'_{i,1}}$$

At the start of the iterative process, λ_0 is set to zero, from equations 4.21 and 4.22,

$$f'_{i,o} = \beta_1 q_o \tag{4.39}$$

$$f'_{i,1} = q_o + \beta_1 q_1 \tag{4.40}$$

Since $q_o = u_o$ and $q_1 = u_1 + \lambda_1^* u_o$ (equations 4.14 and 4.15) and $\lambda_o^* = \lambda_o = 0$, then equations 4.39 and 4.40 become

$$f'_{i,o} = \beta_1 u_o \tag{4.41}$$

$$f'_{i,1} = u_o + \beta_1 u_1 \tag{4.42}$$

But $\beta_1 u_o = y_o$ and $u_o + \beta_1 u_1 = y_1$ (from equation 4.12), therefore, at the first iteration,

$$\lambda_1 = \lambda_o + c\,\frac{f'_{i,o}}{f'_{i,1}}$$

$$= c\ (y_o / y_1) \tag{4.43}$$

It is clear, from equation 4.43, that with c=1 and a sequence Y (equation 2.3) whose components are such that $|y_o|>|y_1|$, $|\lambda_1|$ would be greater than 1. Thus, with a stepsize of 1, the algorithm is started with an estimate of the quantity β_1 whose magnitude is greater than 1 which means that the algorithm would most likely be trying to locate a root (zero) of Y(z) which is inside the unit circle in the z-plane. This will always happen when the modulus of the first component in the sampled impulse-response of the channel is smaller than the modulus of the second component and when $\lambda_Q = 0$, c = 1. Therefore, in order to start the estimate from the "correct" side of the unit circle, the value of c (equation 4.43) must be set to less than unity initially, and the exact value of c must depend, to a large extent, on the valué of $|y_0/y_1|$.

Experiments carried out on Channel 4 has revealed that the condition $|y_0|>|y_1|$ does occur, and it occurs at three separate occasions during the process in which the channel impulse-response is being reduced to minimum phase. On one occasion, the z-transform of the impulse-response of the channel is such that $|y_0/y_1| \approx 3.2$, which means that a stepsize of less than 0.3 has to be used at the beginning of the iterative process so that $|\lambda_1|<1$. Even when $|y_0|$ is not greater than $|y_1|$ as in the raw sampled impulse-response of Channel 4 before any processing and $|y_0/y_1| \approx 0.7$, a stepsize of less than 0.9 was found necessary for the algorithm to locate one of the required roots (Figure 4.7).

Since the smaller value of c necessarily implies a slower convergence rate, and the maximum value of the stepsize varies from case to case depending on $|y_0/y_1|$, it would be difficult to choose a fixed stepsize for all the possible channels that one is likely to meet without paying a heavy penalty in speed for many of the good channels. Oné possible solution is to use a suitable change-over point, p and an appropriate initial value for K_i in equation 4.36, while keeping the stepsize c (equation 4.35) unchanged at unity for the whole of the iterative process.

Different combinations of p (ranging from 1 to 40) and K_p (ranging from 0 to 30) were used in order to achieve this goal, and it was found that, for Channel 4, only six out of a total number of eight roots were located successfully using this method. The order in which the six roots were located is Roots 6, 5, 4, 1, 7 and 8 (Figure 2.12). The remaining two roots, Roots 2 and 3, could not be found by this method using the above combinations of p and K_p but if it is assumed that one of the two roots (either Root 2 or Root 3) were already located, then the remaining root could also be found. In general, there is no rule as regards to

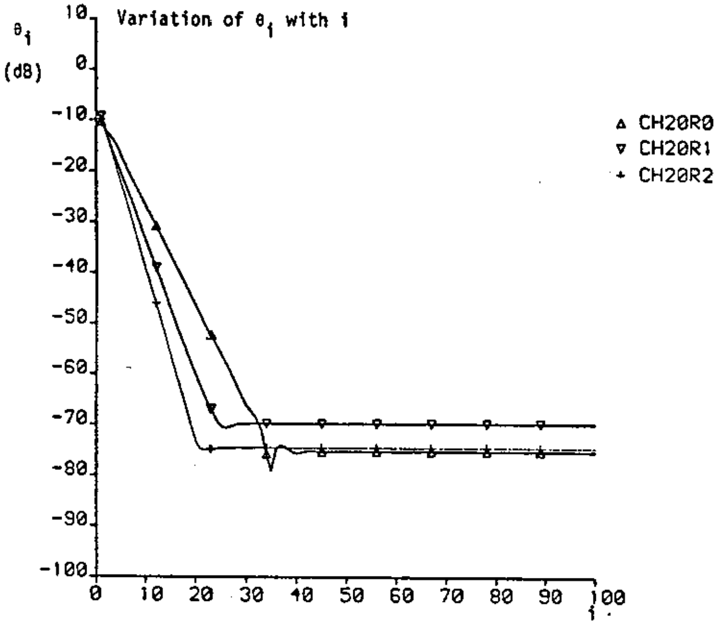


FIGURE 4.3: Performance of system 4.1 for Channel 1, c=1

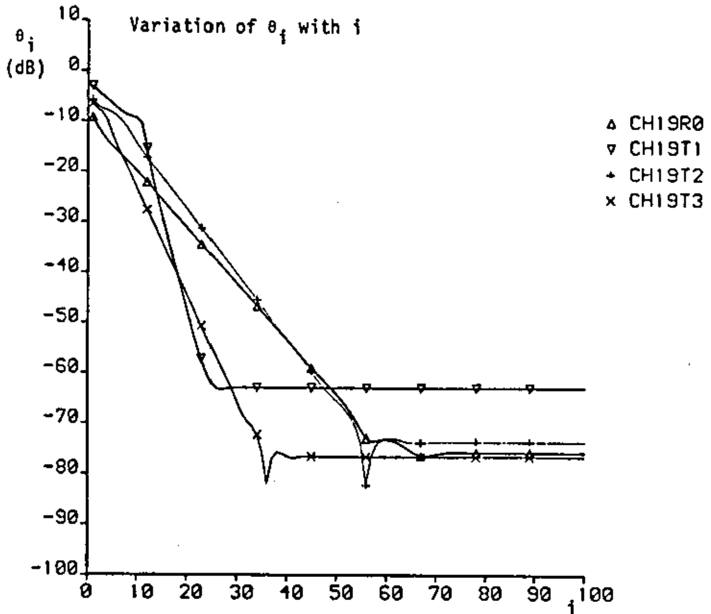


FIGURE 4.4: Performance of system 4.1 for Channel 3, c= 1

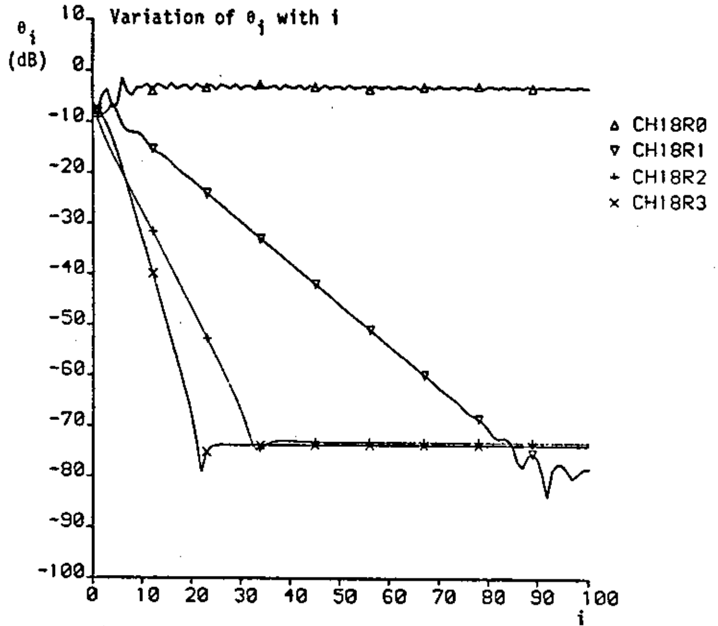


FIGURE 4.5: Performance of system 4.1 for Channel 2, c=1

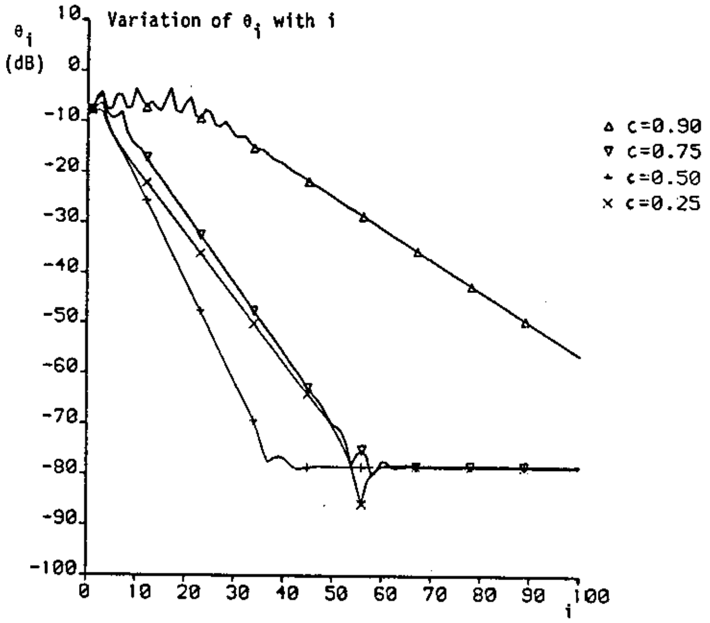


FIGURE 4.6: Performance of system 4.1 for Channel 2

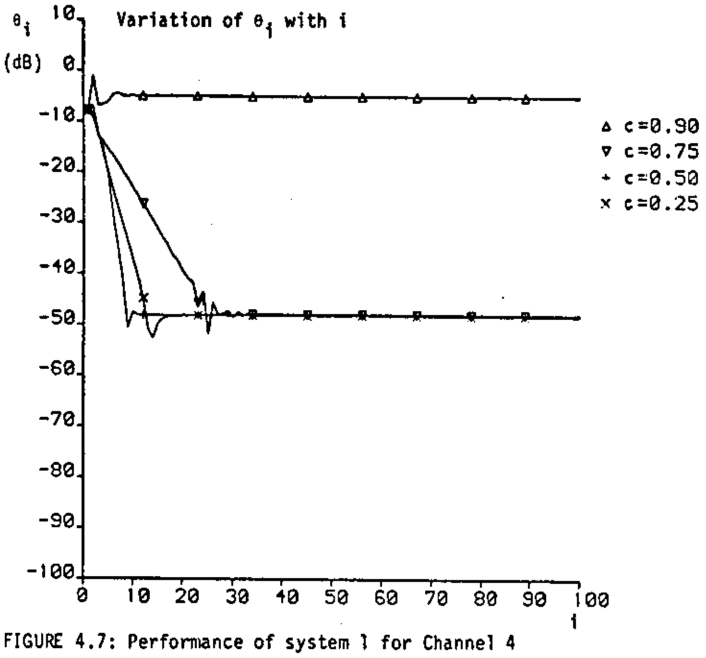


FIGURE 4.7: Performance of system 1 for Channel 4

the best combination of change-over period, p, and the initial value of K_i. In all the cases where $|y_0/y_1|>1$ the averaging process is commenced from the beginning of the iterative process and allowed to continue to run for the whole of the 100 iterations.

This system is very promising in that, with the exception of Channel 4, which is a typical worst circuit for the transmission of data at 1200 bit/s, the roots of the other three channels can be found to an accuracy of better than -60 dB in about 30 iterations using the suitable step-size ($c \leqslant 0.9$)

4.2 System 4.2

This is a modification of System 4.1 in that the order in which the two parts of the iterative process is reversed so that the 'averaging' process is performed first, starting from the first iteration for a period of ℓ iterations, then followed by the algorithm given in equation 4.35. The arrangement is designed specifically to overcome the situation experienced with Channel 2 (Table 2.1), where a smaller value of c (less than unity) is necessary in order to locate the first of the four roots (zeros) of Y(z) that lie outside the unit circle. The idea is that after some ℓ iterations, the estimate λ_ℓ is brought close enough to β_1 (negative reciprocal of the root outside the unit circle) by the 'averaging' process that the second part of the iterative process could be used with c=1 to give a fast and accurate final convergence. The diagram in Figure 4.8 shows how this arrangement compares with System 4.1 (Figure 4.2).

The operation of the receiver is the same as in System 4.1 where at the start of the iterative process, λ_0 is set to zero and the sequence Y (whose z-transform Y(z)) is fed through the two-tap feedforward filter (Figure 4.1) starting with the component y_0 to give the $g+2$ component sequence B_i' (equation 4.5). The sequence B_i' is then reversed in order and is fed through the one-tap feedback filter (Figure 4.1) starting with its last component to give an output sequence $\{f'_{i,k}\}$ (equation 4.32). An improved estimate of β_1 is now given by

$$\lambda_{i+1} = \lambda_i + c\,\frac{f'_{i,o}}{f'_{i,1}} \tag{4.44}$$

(equation 4.35) where c is a positive constant. An extra step, which corresponds to the averaging step, is carried out for ℓ iterations as shown in the flow diagram of Figure 4.8, so that for $0 < i < \ell$,

$$\lambda'_{i+1} = (1 - \frac{1}{K_i})\,\lambda_i + \frac{1}{K_i}\,\lambda_{i+1} \tag{4.45}$$

where $K_i = 2,3,4,\ldots,\ell+1$, $i=1,2,\ldots,\ell$ and $\lambda_0=0$. The value λ'_{i+1} is then taken to be the estimate λ_{i+1} and is used in the subsequent estimate, λ_{i+2}. When the change-over point, ℓ, is reached, the algorithm by-passes the 'averaging' process and the estimate of β_1 is given solely by equation 4.44.

Results of computer simulation tests for Channel 2 show that this arrangement works well and is therefore particularly suitable for cases where a smaller value of c is required at the beginning of the iterative process. This technique is employed successfully in a further modification of this system (System 4.6) to be described later.

However, as far as Channel 4 is concerned, the problem remains, firstly, that it is not possible to find all the roots and, secondly, when $|y_o/y_l|>1$ it is not clear how to determine the initial value of K_i used in the averaging period. An attempt was made to find a way of determining K_i automatically which involves a simple searching operation. At the beginning of the iterative root-finding process K_i is set to 2 and the process continues until $|\lambda_i|$ (magnitude of λ_i) is

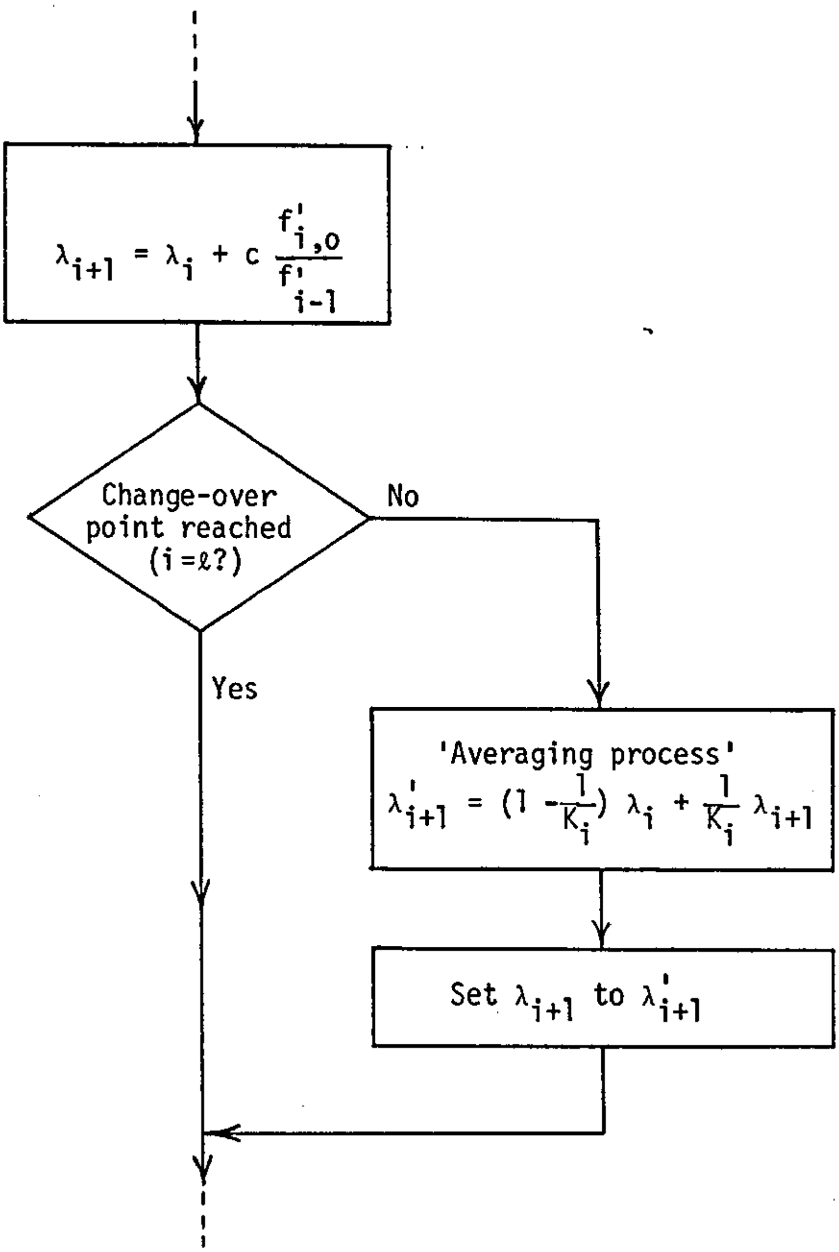

$\lambda_{i+1} = \lambda_i + c \frac{f'_{i,o}}{f'_{i-1}}$
Change-over point reached $(i=\ell?)$
No
Yes
'Averaging process'
$\lambda'_{i+1} = (1 - \frac{1}{K_i})\lambda_i + \frac{1}{K_i}\lambda_{i+1}$
Set λ_{i+1} to λ'_{i+1}

FIGURE 4.8: Diagram which shows the algorithm for estimating β_1 with an averaging process at the beginning of the iterative process

greater than unity, at that point, the iterative process is stopped and restarted with $\lambda_0=0$ and K_i set to 3. The value of K_i is incremented by a fixed amount (equals 1 in this case), every time the process is restarted. It was hoped that this would eventually lead to a successful location of a new root in situations where $|y_0/y_1| > 1$. However, the results (not given here) indicated that this method did not necessarily lead to the finding of a new root and it could involve a very large number of iterations.

4.3 Continuous Operation of the Receiver in the Off-Line System

When the z-transform of the sampled impulse-response of the channel, Y(z) has m roots (zeros) outside the unit circle in the z-plane, the linear filter that is ahead of the detector in the optimum equalizer is one which replaces all of the m roots by their complex conjugate-reciprocals. This can be realised by a single linear feedforward transversal filter or m separate linear feedforward transversal filters where each one is responsible for the processing of one or the m roots. During the process which begins with the raw channel impulse-response and eventually leads to a minimum-phase channel impulse-response, a set of m resultant sampled impulse-responses are produced by the receiver whose z-transforms have (m-1), (m-2),..., 0 roots (zeros) outside the unit circle. This assumes, of course, that the m roots are available by some means and are known to the receiver. Up to now, the study is concentrated on the development of suitable techniques for finding a root from any one of the m z-transforms of the channel sampled impulse-responses which have 1 to m roots outside the unit circle. In order to investigate the performance of various root-tracking algorithms, the set of m{Y(z)} are assumed to be exact. To do this, the set of m roots (zeros) of Y(z) are determined independently by the NAG subroutine CO2ADF[50] and their exact values are used in the forming of the m{Y(z)}. For example, when Y(z) (with m roots outside the unit circle) is operated on and that one of the m roots is located by the algorithm, whose value is checked against the set determined independently by the NAG subroutine CO2ADF[50], the resultant sampled impulse-response now has (m-1) roots outside the unit circle is calculated using the exact values of $\{\beta_h\}$ and is used by the receiver for the processing of further roots. No attempt, however, has been made to combine the m separate operations — the finding of the m roots and the subsequent reduction or forming of the m sampled impulse-responses — in one continuous operation.

This section is concerned with the study of the normal and continuous operation of a receiver in the off-line system where the receiver operates on the sampled impulse-response Y (provided by the channel estimator) to give the required roots of Y(z), and at the end of the root-finding process produces an equivalent all-pass linear filter for the set of located roots. Again, the receiver operates solely and directly on the sequence y_0, y_1, ..., y_g (equation 2.3) until it locates one of the roots of Y(z) that lies outside the unit circle, which is then removed and replaced by the complex conjugate of its reciprocal. The resultant sampledimpulse-response of the channel is then taken as the new input to the receiver, and the process is repeated until all the roots of Y(z) that lie outside the unit circle are found. At the end of the root-finding process, the tap gains of an (n+1)-tap linear feedforward transversal filter are determined, such that this filter removes all m roots of Y(z) and replaces them by the complex conjugate of their reciprocals.

An immediate problem which arises directly as a result of the continuous operation of the receiver is that of finding a suitable criterion to indicate the convergence of a root. This is the condition upon which the algorithm is assumed to have converged to a required root and, if so, any further processing of that root can be stopped. Studies of the possible methods have been made which involved the detailed analysis of the value λ_i (the estimate of β_1 at the i^{th} iteration) and the output signal $\{f'_{i,h}\}$ (equation 4.19) from the one-tap feedback filter at every iteration in order to find some general rules regarding the variations of these parameters at convergence. One such method is to store the five most recent values of $\{\lambda_i\}$ and compare the differences between the neighbouring pairs and when all these differences fall below some threshold value, the algorithm is taken to have converged. Variations of this idea were tested but they all entailed some extra computation and some comparisons at every iteration and then it was difficult to establish a suitable threshold value and a suitable starting place to commence the comparisons. Another similar problem is that of determining the case where all of the m roots of Y(z) outside the unit circle have been located so that the root-finding process can be terminated.

Fortunately, techniques have been found to solve these two problems which will be described in the following sections. The effect of the length of the adaptive linear filter and accuracy in the roots are compared.

4.3.1 System 4.3.1

Consider again the two-tap linear feedforward transversal filter and the one-tap feedback filter as shown in Figure 4.9 below which, together, form the heart of the systems studied so far. From equations 2.18 and 2.19, the z-transforms of the two-tap feedforward filter and the one-tap feedback filter are

$$B_i(z) = 1 + \lambda_i^* z^{-1} \tag{4.46}$$

and

$$A_i(z) = (1 + \lambda_i z^{-1})^{-1} \tag{4.47}$$

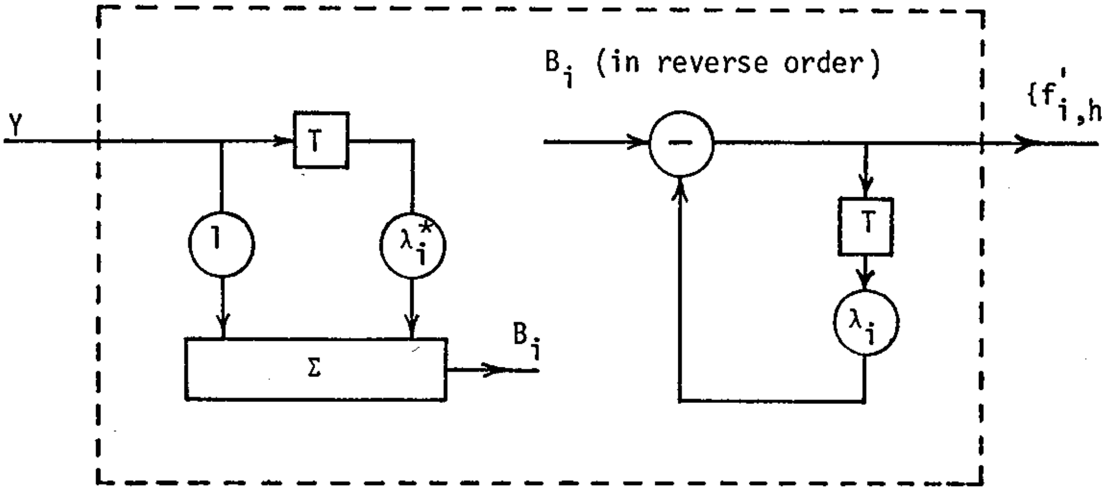


FIGURE 4.9: The Combined Operation of the Two Filters

respectively. At the start of the iterative process, the receiver holds in store the sampled impulse-response of the baseband channel (Figure 2.1), Y, which is assumed to be provided correctly and is given by a (g+1)-component row vector

$$Y = [y_0 \quad y_1 \quad . \; . \; . \quad y_g] \tag{4.48}$$

whose z-transform is

$$Y(z) = y_0 + y_1 z^{-1} + \ldots + y_g z^{-g} \tag{4.49}$$

In describing the operation of the system, the sequence Y is, for convenience, treated as occurring in real time, with its first component y_0 at time t = 0 and its last component y_g at t=gT. Suppose now that (in equation 4.49)

$$Y(z) = (1+\beta_1 z)(u_0 z^{-1} + u_1 z^{-2} + \ldots + u_{g-1} z^{-g}) \tag{4.50}$$

where $-1/\beta_1$ is a root (zero) of Y(z) lying outside the unit circle in the z-plane.

With the tap gains of the filters in Figure 4.9 set to λ_i (an estimate of β_1 (equation 4.50)), the sequence Y is fed through the two-tap feed-forward transversal filter starting with the first component y_0 and finishing with the last component y_g to give a (g+2) output sequence B_i' (equation 4.5) whose z-transform is

$$B_i'(z) = (1+\lambda_i^* z^{-1})(y_0 + y_1 z^{-1} + \ldots + y_g z^{-g})$$

$$= y_0 + (y_1 + \lambda_i^* y_0) z^{-1} + \ldots + \lambda_i^* \, y_g z^{-g-1} \tag{4.51}$$

If now

$$B_i' = [b_{i,o} \quad b_{i,1} \cdot \cdot \cdot \quad b_{i,g+1}] \tag{4.52}$$

then

$$b_{i,o} = y_o \tag{4.53}$$

$$b_{i,1} = y_1 + \lambda_i^* y_o \tag{4.54}$$

$$\vdots$$

$$b_{i,g} = y_g + \lambda_i^* y_{g-1} \tag{4.55}$$

$$b_{i,g+1} = \lambda_i^* y_g \tag{4.56}$$

In general,

$$b_{i,h} = y_h + \lambda_i^* \, y_{h-1} \tag{4.57}$$

where $0 \leqslant h \leqslant g+1$ and $y_h = 0$ for $0 > h > g$.

The output sequence B_i' is then reversed in order so that it begins with $b_{i,g+1}$ and ends with $b_{i,o}$ and is fed through the one-tap feedback filter. The sequence B_i', passing through the filter in the manner described is taken to be moving backwardsin time and thus changes the z-transform of the one-tap feedback filter to, effectively,

$$A_i(z) = (1 + \lambda_i z)^{-1} \tag{4.58}$$

[Therefore, the resultant sampled impulse-response of the channel and the two filters is (see equation 4.13)]

$$\underline{Y}(z)\ \underline{A_i(z)B_i}(z) = (1+\beta_1 z)(u_o z^{-1}+u_1 z^{-2}+\ldots+u_{g-1}z^{-g})(1+\lambda_i z)^{-1}(1+\lambda_i^* z^{-1})$$

$$= (1+\beta_1 z)(1+\lambda_i z)^{-1}(u_o z^{-1}+(u_1+\lambda_i^* u_o)z^{-2}+\ldots+(u_{g-1}+\lambda_i^* u_{g-2})z^{-g}+\lambda_i^* u_{g-1}z^{-g-1})$$

$$\begin{aligned}
= z^{-1}(\ldots+ (& \qquad\qquad\qquad - \lambda_i(\beta_1-\lambda_i)q_o+\lambda_i^2\ (\beta_1-\lambda_i)q_1\ldots)z^2 \\
+ (& \quad (\beta_1-\lambda_i)q_o \qquad - \lambda_i(\beta_1-\lambda_i)q_1+\lambda_i^2\ (\beta_1-\lambda_i)q_2-..)z^1 \\
+ (& q_o+(\beta_1-\lambda_i)q_1 \qquad - \lambda_i(\beta_1-\lambda_i)q_2+\lambda_i^2\ (\beta_1-\lambda_i)q_3-..)z^0 \\
+ (& q_1+(\beta_1-\lambda_i)q_2 \qquad - \lambda_i(\beta_1-\lambda_i)q_3+\lambda_i^2\ (\beta_1-\lambda_i)q_4-..)z^{-1} \\
+ (& q_2+(\beta_1-\lambda_i)q_3 \qquad - \lambda_i(\beta_1-\lambda_i)q_4+\lambda_i^2\ (\beta_1-\lambda_i)q_5-..)z^{-2} \\
+ & \ldots \\
+ & q_g z^{-g})
\end{aligned} \tag{4.59}$$

where
$$q_o = u_o \tag{4.60}$$

$$q_1 = u_1 + \lambda_i^* u_o \tag{4.61}$$

$$\vdots$$

$$q_{g-1} = u_{g-1} + \lambda_i^* u_{g-2} \tag{4.62}$$

$$q_g = \lambda_i^* u_{g-1} \tag{4.63}$$

In general,

$$q_h = u_h + \lambda_i^* u_{h-1} \tag{4.64}$$

where $0 \leqslant h \leqslant g$, and $u_h = 0$ for $0 > h \geqslant g$.

Now if,

$$Y(z)A_i(z)B_i(z) = \ldots + f'_{i,-1}z^{1} + f'_{i,o} + f'_{i,1}z^{-1} + \ldots + f'_{i,g+1}z^{-g-1} \tag{4.65}$$

where $\{f'_{i,h}\}$, for $-\infty < h \leqslant g+1$, is the sequence of the output of the two filters (Figure 4.9) when the sequence Y (equation 4.48) is fed through it in the manner described. Then,

$$f'_{i,o} = (\beta_1-\lambda_i)q_o - \lambda_i(\beta_1-\lambda_i)q_1 + \ldots + (-\lambda_i)^g(\beta_1-\lambda_i)q_g \tag{4.66}$$

$$f'_{i,1} = q_o + (\beta_1-\lambda_i)q_1 - \lambda_i(\beta_1-\lambda_i)q_2 + \ldots + (-\lambda_i)^{g-1}(\beta_1-\lambda_i)q_g \tag{4.67}$$

$$f'_{i,2} = q_1 + (\beta_1-\lambda_i)q_2 - \lambda_i(\beta_1-\lambda_i)q_3 + \ldots + (-\lambda_i)^{g-2}(\beta_1-\lambda_i)q_g \tag{4.68}$$

$$\vdots$$

$$f'_{i,g+1} = q_g \tag{4.69}$$

and in practice, only g+2 components of the sequence of $\{f'_{i,h}\}$ are generated by the receiver and the iterative root-finding process operates entirely on this g+2 component $f'_{i,o}\ f'_{i,1} \ldots f'_{i,g+1}$.

As $\lambda_i \to \beta_1$, from equations 4.66-4.69,

$$f'_{i,o} \simeq (\beta_1-\lambda_i)q_o \simeq 0 \tag{4.70}$$

$$f'_{i,1} \simeq q_o \tag{4.71}$$

$$\vdots$$

$$f'_{i,g+1} \simeq q_g \tag{4.72}$$

and from equation 4.66

$$f'_{i,o} = (\beta_1-\lambda_i)\ [q_o-\lambda_i q_1 + \dots + (-\lambda_i)^g\ q_g] \ \simeq 0$$

$$\simeq (\beta_1-\lambda_i)(f'_{i,1} - \lambda_i f'_{i,2} + \dots + (-\lambda_i)^g f'_{i,g+1}) \simeq 0 \tag{4.73}$$

so that

$$(\beta_1-\lambda_i) \simeq f'_{i,o}/\varepsilon_i \tag{4.74}$$

where $\qquad \varepsilon_i = f'_{i,1} - \lambda_i f'_{i,2} + \dots + (-\lambda_i)^g f'_{i,g+1} \qquad$ (4.75)

and $\qquad \beta_1 = \lambda_i + f'_{i,o}/\varepsilon_i \qquad$ (4.76)

This means that, if c is a small positive constant in the range 0 to 1, then

$$\lambda_{i+1} = \lambda_i + c\,\frac{f'_{i,o}}{\varepsilon_i} \tag{4.77}$$

is a better estimate of β_1 than λ_i.

It can be seen from equation 4.77 that $f'_{i,o}/\varepsilon_i$ represents an error term which is to be added to the old estimate at every iteration, and as $\lambda_i \rightarrow \beta_1$, this term becomes smaller and smaller. In the limit, when $\lambda_i = \beta_1$, $f'_{i,o}$ becomes zero and therefore the term $f'_{i,o}/\varepsilon_i$ can be used to indicate the convergence of a root.

Thus, the iterative process begins with the filters set to an initial value λ_0, and the sequence Y(equation 4.48) is fed through the two-tap feedforward filter and then the one-tap feedback filter in the manner as described to give the sequence $\{f'_{i,h}\}$ (equations 4.66-4.69), and the new value of λ_i is updated according to the algorithm given in equation 4.77. The process is repeated, using the new estimate for the tap gains, until $|f'_{i,o}/\varepsilon_i|^2$ is less than some predetermined threshold value, at which point, the iterative process is stopped and the value λ_i taken to be the estimate of β_1. Therefore, the stopping criterion for any given root is

$$|f'_{i,o}/\varepsilon_i|^2 < d_1 \qquad (4.78)$$

where d_1 is a small threshold value.

Let the value of i at convergence be k, so that

$$\lambda_k \simeq \beta_1 \qquad (4.79)$$

The z-transform of the sequence at the output of the feedback filter is approximately

$$f'_{k,1}\, z^{-1} + f'_{k,2}\, z^{-2} + \ldots + f'_{k,g+1}\, z^{-g-1} \qquad (4.80)$$

(equations 4.65-4.69) which is approximately equal to $Y(z)A_k(z)B_k(z)$. The resultant effect on the sequence Y of the two filters, giving the sequence $\{f'_{k,h}\}$ approximates to that of a single filter with z-transform

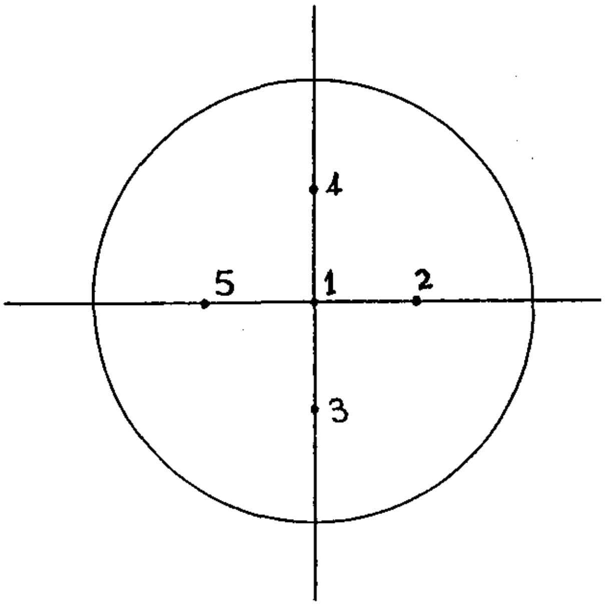
$$
\begin{aligned}
C_1(z) &= A_k(z)B_k(z) \\
&\simeq (1 + \beta_1 z)^{-1}(1 + \beta_1^* z^{-1}) \qquad (4.81)
\end{aligned}
$$

as in equation 2.22. This filter is an all-pass filter having the same basic properties as the ideal adaptive linear transversal filter with z-transform D(z). Let F_1' be the sequence of $\{f_{k,h}'\}$ advanced by one sampling interval whose z-transform is

$$
\begin{aligned}
F_1'(z) &= f_{k,1}' + f_{k,2}' z^{-1} + \ldots + f_{k,g+1}' z^{-g} \\
&\simeq z\, C_1(z)Y(z) \qquad (4.82)
\end{aligned}
$$

For practical purposes, the root (zero) $-1/\beta_1$ in $Y(z)$ is replaced by its complex conjugate-reciprocal, $-\beta_1^*$, and the sequence F_1' is an estimate of the sampled impulse-response of the channel and adaptive linear transversal filter, when the z-transform of the latter is $zC_1(z)$ (equation 4.81).

Since the derivation of the algorithm (equation 4.77) is based on the assumption that $\lambda_i \simeq \beta_1$, it means that the algorithm has a better chance of succeeding if the initial value of λ_i (λ_0) is set such that the distance between λ_0 and β_1 is small in the vector space. This suggests the following starting procedure.



1. $\lambda_o = 0.0 + j0.0$

2. $\lambda_o = 0.5 + j0.0$

3. $\lambda_o = 0.0 - j0.5$

4. $\lambda_o = 0.0 + j0.5$

5. $\lambda_o = -0.5 + j0.0$

FIGURE 4.10: Starting-points for λ_o

Figure 4.10 shows a set of 5 starting-points for λ_o which are numbered in the order in which they are used. That is, the iterative process starts at starting-point 1 and continues as described until it either meets the condition $|f'_{i,o}/\varepsilon_i|^2 < d_1$ in which case it is taken to have converged to a root, or when $|\lambda_i|>1$. In the latter case, the iterative process is stopped and restarted again with λ_o set to the next of the five possible starting-points (starting-point number 2), and the process is repeated. Whenever i reaches 40, the algorithm is assumed to have failed to locate any root and the action taken is identical to the case of $|\lambda_i|>1$ above. When all of the five starting-points have been tried and failed to produce a root with moduli greater than 1 then it is assumed that all the roots of Y(z) that lie outside the unit circle have been found (an assumption which worked successfully in all the cases tested).

At the end of the iterative process when all of the m required roots of Y(z) have been found, the receiver uses these roots to adjust the tap gains of the (n+1)-tap adaptive linear feedforward transversal filter (Figure 2.1) as follows. All the tap gains of the linear filter are set to zero execpt for the last tap whose gain is set to unity.

The initial z-transform of the filter is, therefore,

$$D_o(z) = z^{-n} \tag{4.83}$$

and the initial z-transform of the channel and filter is $z^{-n}Y(z)$. The sequence D_o with n+1 components and the z-transform $D_o(z)$ is fed through the two-tap feedforward transversal filter (Figure 2.2) with its tap set to $\lambda_k^* \simeq \beta_1^*$, starting with the first component of D_o, to give an output sequence with n+2 components and the z-transform $D_o(z)B_k(z)$. The latter sequence is fed in reversed order, and starting with the last component, through the one-tap feedback transversal filter (Figure 2.3), whose effective z-transform is now $A_k(z)=(1+\lambda_k z)^{-1}$, to give the output sequence with a z-transform that is approximately

$$D_o(z)\ C_1(z) = D_o(z)\ A_k(z)\ B_k(z) \tag{4.84}$$

where

$$\begin{aligned} A_k(z) &= (1 + \lambda_k z)^{-1} \\ &\simeq (1 + \beta_1 z)^{-1} \end{aligned} \tag{4.85}$$

$$\begin{aligned} B_k(z) &= (1 + \lambda_k^* z^{-1}) \\ &\simeq (1 + \beta_1 z^{-1}) \end{aligned} \tag{4.86}$$

When n+1 components of the output sequence have been obtained, the process is halted. These n+1 components, in the order in which they are received, are the coefficients of z^{-n-1}, z^{-n},..., z^{-1} in $D_o(z)C_1(z)$. The tap gain of the h^{th} tap from the front end of the adaptive filter is now set to the coefficient of z^{-h}, for h = 1,2,..., n+1, to give the required tap gains. The h^{th} tap of the filter is associated with a delay of h-1 sampling intervals, whose z-transform is z^{-h+1}. Thus the z-transform of the adaptive filter is approximately

$$D_1(z) = z\ D_o(z)\ C_1(c) \tag{4.87}$$

The whole process for adjusting the adaptive filter is repeated, but using $D_1(z)$ in place of $D_o(z)$ and the tap coefficients $\lambda_k \simeq \beta_2$ in place of $\lambda_k \simeq \beta_1$, so that when the process is repeated m times, the z-transform of the filter is, now approximately

$$D_m(z) \simeq D(z) = z^{-n}\ Y_2^{-1}(z)\ Y_3(z) \tag{4.88}$$

from equation 2.10,where $Y_2(z)$ and $Y_3(z)$ are given by equations 2.8 and 2.11 respectively. Thus the z-transform of the channel and adaptive filter is

$$\begin{aligned} & a_o + a_1 z^{-1} + \ldots + a_{n+g}\ z^{-n-g} \\ \simeq\ & Y(z)\ D_m(z) \\ \simeq\ & F(z) \end{aligned} \tag{4.89}$$

(equation 2.12), where ideally $a_h \simeq 0$ for $h = 0,1,\ldots,n-1$. The estimate of the sampled impulse-response of the channel and adaptive filter, that is employed by the detector, is the sequence F_m', with z-transform

$$\begin{aligned} F_m'(z) &= f'_{m,1} + f'_{m,2}z^{-1} + \dots + f'_{m,g+1}z^{-g} \\ &\simeq z^n\ F(z) \\ &= Y_1(z)\ Y_3(z) \end{aligned} \tag{4.90}$$

(equations 2.7 and 2.11). The delay of n sampling intervals introduced by the adaptive filter is, for convenience, ignored here but must obviously be taken into account when comparing Y(z) $D_m(z)$ and $F_m'(z)$.

The performance of the algorithm is measured in terms of two parameters, θ_i and ϕ_i'. θ_i is the error in the estimate of β_h at the i^{th} iteration which is given by

$$\theta_i = 10\log_{10}(|\beta_h - \lambda_i|^2) \tag{4.91}$$

ϕ_i' is the error in the g+1 output sequence at the one-tap feedback filter, $\{f'_{i,h}\}$, h = 1,2,...g+1, compared with their actual values. The sequence at the output of the one-tap feedback filter during the iterative adjustment of λ_i towards β_h is

$$[f'_{i,1}\quad f'_{i,2}\ \cdot\ \cdot\ \cdot\ f'_{i,g+1}] \tag{4.92}$$

The correct value of this sequence is, of course, that obtained when h of the roots of Y(z) have been replaced by the complex conjugates of their reciprocals and $\lambda_i = \beta_h$ (thus locating the h^{th} root). Let the correct sequence be

$$[v_1'\ v_2'\ \cdot\ \cdot\ \cdot\ v'_{g+1}] \tag{4.93}$$

hose z-transform is

$$v_1' + v_2' z^{-1} + \ldots + v_{g+1}' z^{-g-1}$$

$$= \underbrace{z^h Y(z) \prod_{i=1}^{h} (1 + \beta_i z)^{-1}(1 + \beta_i^* z^{-1})}_{} \qquad (4.94)$$

T s,

$$\phi_i' = 10 \log_{10} \Big(\sum_{j=1}^{g+1} |f_{i,j}' - v_j'|^2 \Big)) \qquad (4.95)$$

Tw other very important measurements of the system's performance are th errors ψ_1 and ψ_2 introduced into the sampled impulse-response of th channel and adaptive filter by the finite number of taps of this fi r. ψ_1 is here defined to be the discrepancy between the actual sa ed impulse-response of the channel and adaptive filter and that ass ed by the detector, over the g+1 components of this response that are eally the only nonzero components. Thus ψ_1 is a measure of the acc cy of the g+1 components of the estimate of the sampled impulse-res se of the channel and adaptive filter that are used by the detector. ore precisely,

$$\psi_1 = 10 \log_{10} \Big(\sum_{h=0}^{g} |a_{n+h} - f_{m,h+1}'|^2 \Big) \qquad (4.96)$$

CT - YMT

where$_h$} is the sampled impulse-response of the channel and filter (equa n 4.94) and $\{f_{m,h}'\}$ is the sampled impulse-response of the chann nd filter that is employed by the detector (equation 4.90). ψ_2 is ined as the total discrepancy between the actual and the assume ampled impulse-response of the channel and filter, taking into a unt now all the components assumed to be zero by the detector. e precisely,

$$\psi_2 = 10 \log_{10} \left(\sum_{h=0}^{n-1} |a_h|^2 + \sum_{h=0}^{g} |a_{n+h} - f'_{m,h+1}|^2 \right) \qquad (4.97)$$

For completeness, the discrepancy between the ideal sampled impulse-response of the channel and filter and that assumed by the detector are given by ψ_3 and ψ_4, where

P YMT

$$\psi_3 = 10 \log_{10} \left(\sum_{h=0}^{g} |f_{n+h} - f'_{m,h+1}|^2 \right) \qquad (4.98)$$

$$\psi_4 = 10 \log_{10} \left(\sum_{h=0}^{n-1} |f_h|^2 + \sum_{h=0}^{g} |f_{n+h} - f'_{m,h+1}|^2 \right) \qquad (4.99)$$

where $f_o + f_1 z^{-1} + \ldots + f_{n+g} z^{-n-g}$

$$= F(z) \qquad (4.100)$$

(equation 2.12), and F(z) is the ideal sampled impulse-response of the channel and filter which is the same as Y(z) $D_m(z)$ (equation 4.89) with $n \to \infty$.

CT

The results of computer-simulation tests for Channels 1-4 (Table 2.1) are shown in Figures 4.11-4.14 which give the variation of θ_i and ϕ'_i with i for a threshold value $d_1 = 10^{-10}$ (equation 4.78), and stepsize c = 1.0. The figures also show the order in which the roots are found and the number of iterations taken to locate each of the roots are given immediately after the root in parentheses. These are listed in the graphs starting from the top. The total number of iterations taken to locate all the roots, which includes the iterations required to search through the set of five starting-points before stopping, is also given in the graphs. Tables 4.2 to 4.9 present the values ψ_1, ψ_2, ψ_3 and ψ_4, as defined in equations 4.96-4.99, for each of the four channels when the number of taps in the linear filter (Figure 2.1) ahead of the detector is assumed to vary from 20 to 100 in steps of 10. Furthermore, a wide range of thresholds ($d_1 = 10^{-7}$, 10^{-10}, 10^{-15} and 10^{-20}) and stepsizes ($c = 0.5$, 0.7 and 1.0) are tested in order to give the most suitable design of the algorithm.

The results show that by using a set of five different values for λ_0 (Figure 4.2) for the restart procedure, the algorithm can now be used to find all the roots of Y(z) outside the unit circle, even for the most severely distorted Channel 4. For a threshold of 10^{-10} and $c = 1.0$, any individual root can be located in less than nineteen iterations and the error in the estimate, θ_i, is of the order of -100 dB and that once θ_i has reached about -25 to -30 dB, the convergence to the final value of the root can be achieved within three iterations. It was also found that in order to obtain more accurate estimates of roots by using a smaller threshold value, such as 10^{-20}, only a few extra iterations are required. For example, in the case of Channel 3, the number of iterations taken for the algorithm to find all the roots using $c = 1.0$, is 38 for $d_1 = 10^{-7}$ and 45 for $d_1 = 10^{-20}$. After examining the set of results for ψ_1 and ψ_2, it is clear that the accuracy in the estimates of $\{\beta_h\}$ is also reflected in ψ_1 and ψ_2, and judging from ψ_2 for the four channels, a design using a linear filter of 40 taps together with $d_1 = 10^{-10}$ and $c = 1.0$ gives a good compromise between equipment complexity and accuracy of the operation. For the milder channels, Channels 1-3 (Table 2.1), the results show that all the roots can be located within 40 iterations (in contrast with 118 iterations for Channel 4) and that a linear filter of 30 taps can be used.

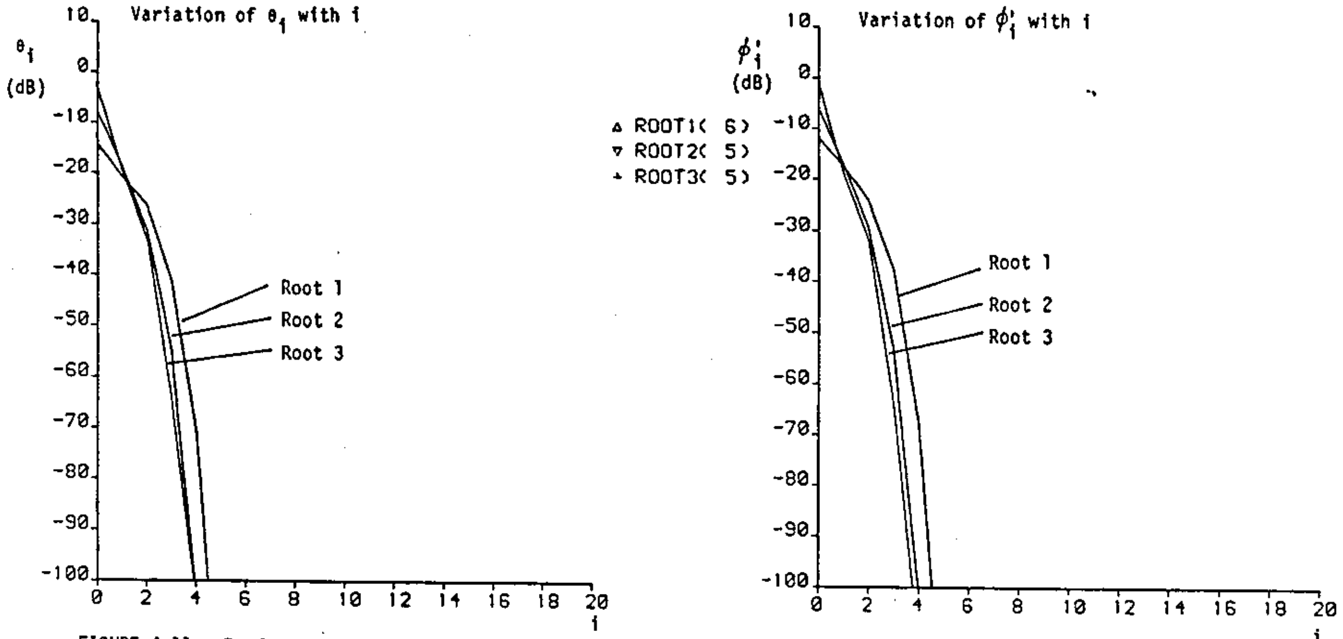


FIGURE 4.11: Performance of System 4.3.1 for Channel 1, c=1, $d_1=10^{-10}$, No. of iterations taken = 21

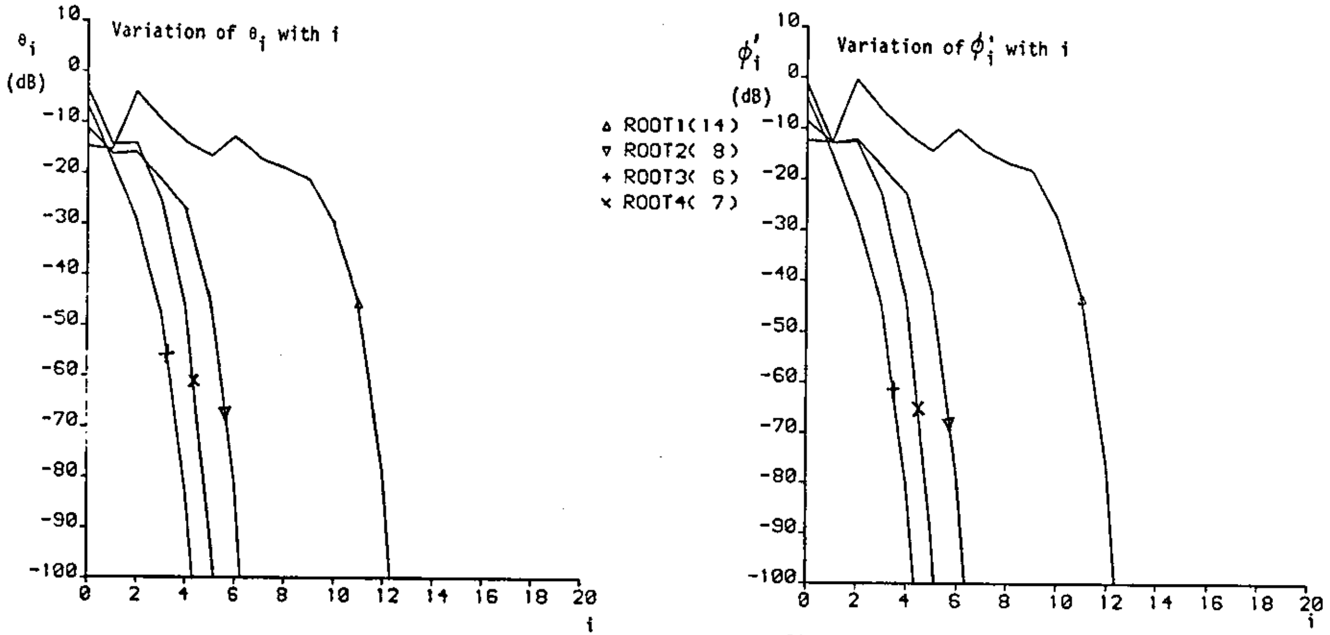


FIGURE 4.12: Performance of System 4.3.1 for Channel 2, c=1, $d_1=10^{-10}$, No. of iterations taken = 40

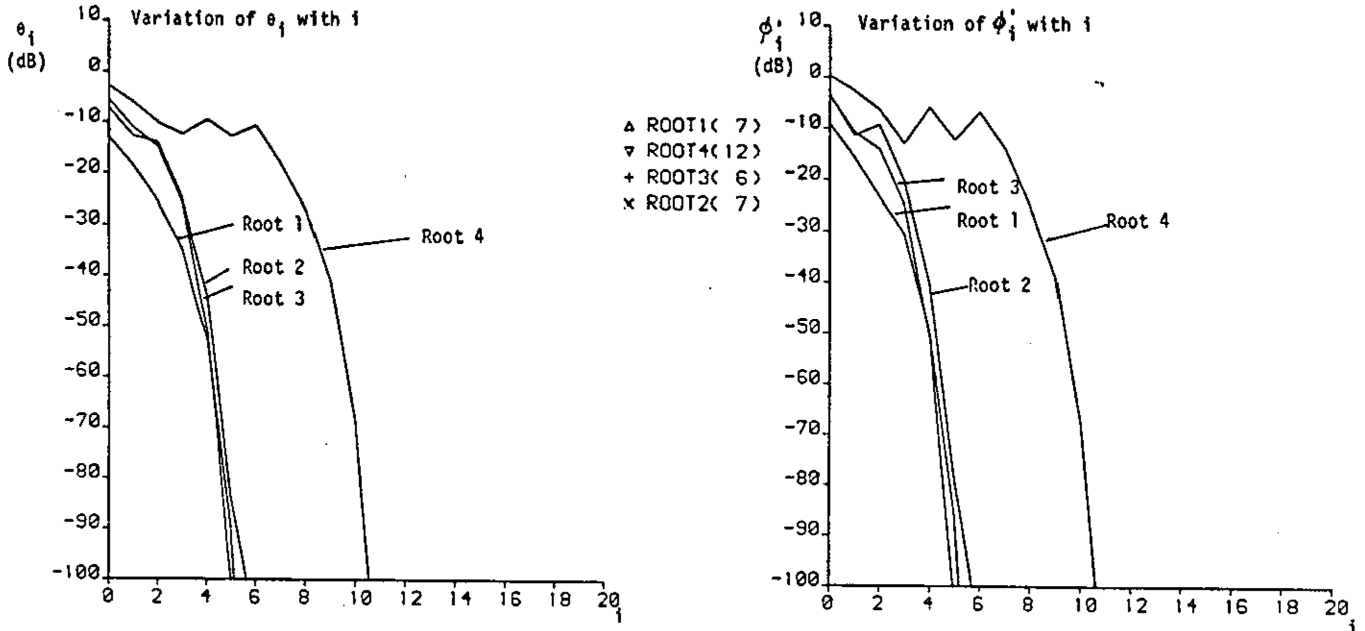


FIGURE 4.13: Performance of System 4.3.1 for Channel 3, c=1, $d_1=10^{-10}$, No. of iterations taken = 40

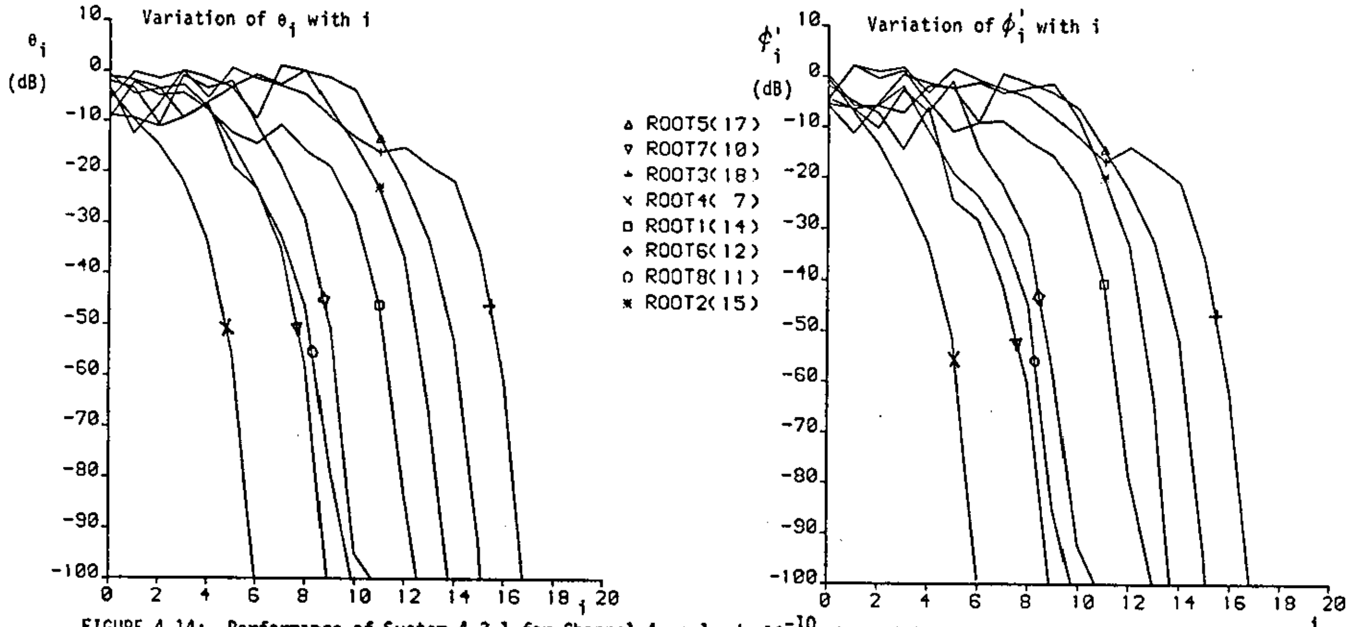


FIGURE 4.14: Performance of System 4.3.1 for Channel 4, c=1, $d_1=10^{-10}$, No. of iterations taken = 118

	c =	$d_1 = 10^{-7}$			$d_1 = 10^{-10}$			$d_1 = 10^{-15}$			$d_1 = 10^{-20}$		
	c =	0.5	0.7	1.0	0.5	0.7	1.0	0.5	0.7	1.0	0.5	0.7	1.0
	No of Iterations	47	29	20	62	38	21	86	53	24	113	68	25
No of taps in filter	20	-257.12 -57.24	-256.23 -57.25	-256.85 -57.33	-256.85 -57.31	-257.72 -57.31	-259.14 -57.31	-259.18 -57.31	-258.41 - 57.31	-258.64 - 57.31	-259.44 -57.31	-257.47 -57.31	-257.68 -57.31
	30	-257.12 -74.71	-256.23 -74.99	-256.85 -82.22	-256.85 -87.93	-257.72 -87.96	-259.14 -88.00	-259.18 -88.02	-258.41 -88.02	-258.64 -88.02	-259.44 -88.02	-257.47 -88.02	-257.68 -88.02
	40	-257.12 -74.92	-256.23 -75.21	-256.85 -83.52	-256.85 -104.84	-257.72 -106.32	-259.14 -109.95	-259.18 -118.72	-258.41 -118.72	-258.64 -118.73	-259.44 -118.73	-257.47 -118.73	-257.68 -118.73
	50	-257.12 -74.92	-256.23 -75.21	-256.85 -83.52	-256.85 -105.02	-257.72 -106.58	-259.14 -110.57	-259.18 -147.90	-258.41 -148.96	-258.64 -149.43	-259.44 -149.43	-257.47 -149.43	-257.68 -149.43
	60	-257.12 -74.92	-256.23 -75.21	-256.85 -83.52	-256.85 -105.02	-257.72 -106.58	-259.14 -110.57	-259.18 -153.18	-258.41 -158.84	-258.64 -180.13	-259.44 -180.13	-257.47 -180.13	-257.68 -180.14
	70	-257.12 -74.92	-256.23 -75.21	-256.85 -83.52	-256.85 -105.02	-257.72 -106.58	-259.14 -110.57	-259.18 -153.19	-258.41 -158.87	-258.64 -204.80	-259.44 -265.76	-257.47 -207.99	-257.68 -210.84
	80	-257.12 -74.92	-256.23 -75.21	-256.85 -83.52	-256.85 -105.02	-257.72 -106.58	-259.14 -110.57	-259.18 -153.19	-258.41 -158.87	-258.64 -206.04	-259.64 -207.38	-257.47 -211.15	-257.68 -241.20
	90	-257.12 -74.92	-256.23 -75.21	-256.85 -83.52	-258.85 -105.02	-257.72 -106.58	-259.14 -110.57	-259.18 -153.19	-258.41 -158.87	-258.64 -206.04	-259.44 -207.38	-257.47 -211.16	-257.68 -252.28
	100	-257.12 -74.92	-256.23 -75.21	-256.85 -83.52	-256.85 -105.02	-257.72 -106.58	-259.14 -110.57	-259.18 -153.19	-258.41 -158.87	-258.64 -206.04	-259.44 -207.38	-257.47 -211.16	-257.68 -252.32

TABLE 4.2: ψ_1, ψ_2 for Channel 1

	c =	$d_1 = 10^{-7}$			$d_1 = 10^{-10}$			$d_1 = 10^{-15}$			$d_1 = 10^{-20}$		
	c =	0.5	0.7	1.0	0.5	0.7	1.0	0.5	0.7	1.0	0.5	0.7	1.0
	No of Iterations	59	42	36	78	54	40	112	74	41	144	92	44
No of taps in filter	20	-263.67 -49.22	-262.51 -49.25	-263.69 -49.25	-261.50 -49.25	-266.46 -49.25	-264.17 -49.25	-263.84 -49.25	-262.80 -49.25	-264.60 -49.25	-263.85 -49.25	-263.84 -49.25	-265.25 -49.25
	30	-263.67 -73.68	-262.51 -75.13	-263.69 -78.78	-261.50 -79.41	-266.46 -79.41	-264.17 -79.42	-263.84 -79.42	-262.80 -79.42	-264.60 -79.42	-263.85 -79.42	-263.84 -79.42	-265.25 -79.42
	40	-263.67 -75.04	-262.51 -77.14	-263.69 -87.47	-261.50 -103.69	-266.46 -105.99	-264.17 -109.58	-263.84 -109.58	-262.80 -109.58	-264.60 -109.58	-263.85 -109.58	-263.84 -109.58	-265.25 -109.58
	50	-263.67 -75.04	-262.51 -77.15	-263.69 -87.49	-261.50 -104.98	-266.46 -108.50	-264.17 -139.68	-263.84 -139.58	-262.80 -139.70	-264.60 -139.70	-263.85 -139.73	-263.84 -139.73	-265.25 -139.73
	60	-263.67 -75.04	-262.51 -77.15	-263.69 -87.49	-261.50 -104.99	-266.46 -108.50	-264.17 -158.84	-263.84 -154.14	-262.80 -160.28	-264.60 -160.51	-263.85 -169.89	-263.84 -169.89	-265.25 -169.89
	70	-263.67 -75.04	-262.51 -77.15	-263.69 -87.49	-261.50 -104.99	-266.46 -108.50	-264.17 -159.20	-263.84 -154.26	-262.80 -160.79	-264.60 -161.04	-263.85 -198.06	-263.84 -198.47	-265.25 -200.05
	80	-263.67 -75.04	-262.51 -77.15	-263.69 -87.49	-261.50 -104.99	-266.46 -108.50	-264.17 -159.20	-263.84 -154.26	-262.80 -160.79	-264.60 -161.04	-263.85 -202.42	-263.84 -203.62	-265.25 -230.20
	90	-263.67 -74.04	-262.51 -77.15	-263.69 -87.49	-261.50 -104.99	-266.46 -108.50	-264.17 -159.20	-263.84 -154.26	-262.80 -160.79	-264.60 -161.04	-263.85 -202.42	-263.84 -203.63	-265.25 -258.58
	100	-263.67 -75.04	-262.51 -77.15	-263.69 -87.49	-261.50 -104.99	-266.46 -108.50	-264.17 -159.20	-263.84 -154.26	-262.80 -160.79	-264.60 -161.04	-263.85 -202.42	-263.84 -203.63	-265.25 -263.30

TABLE 4.3: ψ_1, ψ_2 for Channel 2

	c =	$d_1 = 10^{-7}$			$d_1 = 10^{-10}$			$d_1 = 10^{-15}$			$d_1 = 10^{-20}$		
	c =	0.5	0.7	1.0	0.5	0.7	1.0	0.5	0.7	1.0	0.5	0.7	1.0
	No of Iterations	78	56	38	99	67	40	131	85	42	164	105	45
No of taps in filter	20	-93.80 -44.06	-93.84 -44.09	-93.83 -44.08	-93.83 -44.08	-93.83 -44.08	-93.83 -44.08	-93.83 -44.08	-93.83 -44.08	-93.83 -44.08	-93.83 -44.08	-93.83 -44.08	-93.83 -44.08
	30	-257.99 -69.97	-258.86 -70.74	-257.42 -70.96	-258.80 -71.06	-257.92 -71.07	-258.34 -71.06	-257.53 -71.06	-257.39 -71.06	-257.34 -71.06	-256.83 -71.06	-259.12 -71.06	-259.25 -71.06
	40	-257.99 -76.61	-258.86 -81.78	-257.42 -86.75	-258.80 -97.49	-257.92 -97.89	-258.34 -97.94	-257.53 -98.02	-257.39 -98.02	-257.34 -98.02	-256.83 -98.02	-259.12 -98.02	-259.25 -98.02
	50	-257.99 -76.64	-258.86 -81.88	-257.42 -87.09	-258.80 -106.80	-257.92 -112.80	-258.34 -114.69	-257.53 -124.97	-257.39 -124.97	-257.34 -124.97	-256.83 -124.97	-259.12 -124.97	-259.25 -124.97
	60	-257.99 -76.64	-258.86 -81.88	-257.42 -87.09	-258.80 -106.87	-257.92 -113.07	-258.34 -115.11	-257.53 -150.20	-257.39 -150.68	-257.34 -151.75	-256.83 -151.93	-259.12 -151.93	-259.25 -151.93
	70	-257.99 -76.64	-258.86 -81.88	-257.42 -87.09	-258.80 -106.87	-257.92 -113.08	-258.34 -115.11	-257.53 -155.01	-257.39 -156.67	-257.34 -165.58	-256.83 -178.87	-259.12 -178.88	-259.25 -178.88
	80	-257.99 -76.64	-258.86 -81.88	-257.42 -87.09	-258.80 -106.87	-257.92 -113.08	-258.34 -115.11	-257.53 -155.03	-257.39 -156.69	-257.34 -165.79	-256.83 -203.82	-259.12 -204.12	-259.25 -205.83
	90	-257.99 -76.64	-258.86 -81.88	-257.42 -87.09	-258.80 -106.87	-257.92 -113.08	-258.34 -115.11	-257.53 -155.03	-257.39 -156.69	-257.34 -165.79	-256.83 -208.11	-259.12 -208.96	-259.25 -232.70
	100	-257.99 -76.64	-258.86 -81.88	-257.42 -87.09	-258.80 -106.87	-257.92 -113.08	-258.34 -115.11	-257.53 -155.03	-257.39 -156.69	-257.34 -165.79	-256.83 -208.13	-259.12 -208.98	-259.25 -249.24

TABLE 4.4: ψ_1, ψ_2 for Channel 3

	c =	$d_1 = 10^{-7}$			$d_1 = 10^{-10}$			$d_1 = 10^{-15}$			$d_1 = 10^{-20}$		
	c =	0.5	0.7	1.0	0.5	0.7	1.0	0.5	0.7	1.0	0.5	0.7	1.0
	No of Iterations	155	116	114	195	139	118	259	176	123	326	216	128
No of taps in filter	20	-67.19 -22.22	-67.18 -22.20	-67.18 -22.21	-67.18 -22.21	-67.18 -22.21	-67.18 -22.21	-67.18 -22.21	-67.18 -22.21	-67.18 -22.21	-67.18 -22.21	-57.18 -22.21	-57.18 -22.21
	30	-255.42 -40.43	-253.70 -40.59	-255.49 -40.58	-254.90 -40.60	-255.53 -40.61	-256.28 -40.61	-256.74 -40.61	-255.47 -40.61	-255.90 -40.61	-255.50 -40.61	-253.73 -40.61	-254.98 -40.61
	40	-255.42 -39.35	-253.70 -39.23	-255.49 -39.27	-254.90 -39.28	-255.53 -39.28	-256.28 -39.28	-256.74 -39.28	-255.47 -39.28	-255.90 -39.28	-255.50 -39.28	-253.73 -39.28	-254.98 -39.28
	50	-255.42 -51.23	-253.70 -51.13	-255.49 -51.25	-254.90 -51.28	-255.53 -51.27	-256.28 -51.28	-256.74 -51.28	-255.47 -51.28	-255.90 -51.28	-255.50 -51.28	-253.73 -51.28	-254.98 -51.28
	60	-255.42 -54.65	-253.70 -54.48	-255.49 -54.57	-254.90 -54.60	-255.53 -54.60	-256.28 -54.60	-256.74 -54.60	-255.47 -54.60	-255.90 -54.60	-255.50 -54.60	-253.73 -54.60	-254.98 -54.60
	70	-255.42 -63.37	-253.70 -63.13	-255.49 -63.45	-254.90 -63.49	-255.53 -63.49	-256.28 -63.49	-256.74 -63.49	-256.47 -63.49	-255.90 -63.49	-255.50 -63.49	-255.73 -63.49	-253.98 -63.49
	80	-255.42 -68.72	-253.70 -68.47	-255.49 -69.22	-254.90 -69.28	-255.53 -69.27	-256.28 -69.28	-257.74 -69.27	-255.47 -69.27	-255.90 -69.27	-255.50 -69.27	-253.73 -69.27	-254.98 -69.27
	90	-255.42 -74.13	-253.70 -73.82	-255.49 -76.69	-254.90 -76.76	-255.53 -76.75	-256.28 -76.76	-256.74 -76.76	-255.47 -76.76	-255.90 -76.76	-255.50 -76.76	-253.73 -76.76	-254.98 -76.76
	100	-255.42 -76.49	-253.70 -76.21	-255.49 -83.35	-254.90 -83.45	-255.53 -83.45	-256.28 -83.45	-256.74 -83.46	-255.74 -83.46	-255.90 -83.46	-255.50 -83.46	-253.73 -83.46	-254.98 -83.46

TABLE 4.5: ψ_1, ψ_2 for Channel 4

		$d_1 = 10^{-7}$			$d_1 = 10^{-10}$			$d_1 = 10^{-15}$			$d_1 = 10^{-20}$		
c = No of Iterations		0.5 47	0.7 29	1.0 20	0.5 62	0.7 38	1.0 21	0.5 86	0.7 53	1.0 24	0.5 113	0.7 68	1.0 25
No of taps per filter	20	-70.52 -57.04	-71.18 -57.18	-83.42 -57.32	-100.62 -57.31	-102.55 -57.31	-110.34 -57.31	-148.79 -57.31	-154.83 -57.31	-206.08 -57.31	-203.00 -57.31	-207.14 -57.31	-228.93 -57.31
	30	-70.62 -69.12	-71.18 -69.67	-83.42 -79.76	-100.62 -87.71	-102.55 -87.81	-110.34 -87.97	-148.79 -88.02	-154.83 -88.02	-206.08 -88.02	-203.00 -88.02	-207.14 -88.02	-228.93 -88.02
	40	-70.52 -69.18	-71.18 -69.73	-83.42 -80.46	-100.62 -99.28	-102.55 -101.03	-110.34 -107.13	-148.79 -118.72	-154.83 -118.72	-206.08 -118.73	-203.00 -118.73	-207.14 -118.73	-228.93 -118.73
	50	-70.52 -69.18	-71.18 -69.73	-83.42 -80.46	-100.62 -99.28	-102.55 -101.10	-110.34 -107.44	-148.79 -145.31	-154.83 -147.96	-206.08 -149.43	-203.00 -149.43	-207.14 -149.43	-228.93 -149.43
	60	-70.52 -69.18	-71.18 -69.73	-83.42 -80.46	-100.62 -99.28	-102.55 -101.10	-110.34 -107.44	-148.79 -147.44	-153.83 -153.38	-206.08 -180.11	-203.00 -180.11	-207.14 -180.13	-228.93 -180.14
	70	-70.52 -69.18	-71.18 -69.73	-83.42 -80.46	-100.62 -99.28	-102.55 -101.10	-110.34 -107.44	-148.79 -147.44	-154.83 -183.39	-206.08 -202.38	-203.00 -201.16	-207.14 -204.53	-228.93 -210.78
	80	-70.82 -69.18	-71.18 -69.73	-83.42 -80.46	-100.62 -99.28	-102.55 -101.10	-110.34 -107.44	-148.79 -147.44	-154.83 -153.39	-206.08 -203.05	-203.00 -201.65	-207.14 -205.69	-228.93 -228.68
	90	-70.52 -69.18	-71.18 -69.73	-83.42 -80.46	-100.62 -99.28	-102.55 -101.10	-110.34 -107.44	-148.79 -147.44	-153.83 -153.39	-206.08 -203.05	-203.00 -201.65	-207.14 -205.69	-228.93 -228.91
	100	-70.52 -69.18	-71.18 -69.73	-83.42 -80.46	-100.62 -99.28	-102.55 -101.10	-110.34 -107.44	-149.79 -147.44	-154.83 -153.39	-206.09 -203.05	-203.00 -201.65	-207.14 -205.69	-228.93 -228.91

TABLE 4.6: ψ_3, ψ_4 for Channel 1

		$d_1 = 10^{-7}$			$d_1 = 10^{-10}$			$d_1 = 10^{-15}$			$d_1 = 10^{-10}$		
c = No of Iterations		0.5 59	0.7 42	1.0 36	0.5 78	0.7 54	1.0 40	0.5 112	0.7 74	1.0 41	0.5 144	0.7 92	1.0 44
No of taps in filter	20	-72.23 -49.20	-75.30 -49.24	-87.46 -49.25	-102.17 -49.25	-106.65 -49.25	-158.95 -49.25	-151.04 -49.25	-158.94 -49.25	-160.96 -49.25	-199.27 -49.25	-201.48 -49.25	-228.94 -49.25
	30	-72.23 -69.88	-75.30 -72.20	-87.46 -78.23	-102.17 -79.38	-106.65 -79.41	-158.95 -79.42	-151.04 -79.42	-158.94 -79.42	-160.96 -79.42	-199.27 -79.42	-201.48 -79.42	-228.94 -79.42
	40	-72.23 -70.40	-75.30 -73.11	-87.46 -84.45	-102.17 -99.85	-106.65 -103.30	-158.95 -109.58	-151.04 -109.59	-158.94 -109.58	-160.96 -109.58	-199.27 -109.58	-201.48 -109.58	-228.94 -109.58
	50	-72.23 -70.40	-75.30 -73.11	-87.46 -84.47	-102.17 -100.34	-106.65 -104.47	-158.95 -139.63	-151.04 -139.28	-158.94 -139.65	-160.96 -139.67	-199.27 -139.73	-201.48 -139.73	-228.94 -139.73
	60	-72.23 -70.40	-75.30 -73.11	-87.46 -84.47	-102.17 -100.34	-106.65 -104.47	-154.95 -155.89	-151.04 -149.31	-158.94 -156.55	-160.96 -157.72	-195.27 -169.88	-201.48 -169.88	-228.94 -169.89
	70	-72.23 -70.40	-75.30 -73.11	-87.46 -84.47	-102.17 -100.34	-106.65 -104.47	-158.95 -156.06	-151.04 -149.35	-158.94 -156.76	-160.96 -157.99	-199.27 -195.61	-201.48 -196.71	-228.94 -200.04
	80	-72.23 -70.40	-75.30 -73.11	-87.46 -84.47	-102.17 -100.34	-106.65 -104.47	-158.95 -156.06	-151.04 -149.35	-158.94 -156.76	-160.95 -157.99	-199.27 -197.55	-201.48 -199.41	-228.94 -226.51
	90	-72.23 -70.40	-75.30 -73.11	-87.46 -84.47	-102.17 -100.34	-106.65 -104.47	-158.95 -156.06	-151.04 -149.35	-158.94 -156.76	-160.96 -157.99	-199.27 -197.55	-201.48 -199.41	-228.94 -228.93
	100	-72.23 -70.40	-75.30 -73.11	-87.46 -84.47	-102.17 -100.34	-106.65 -104.47	-158.95 -156.06	-151.04 -149.35	-158.94 -156.76	-166.96 -157.99	-199.27 -197.55	-201.48 -199.41	-228.94 -228.94

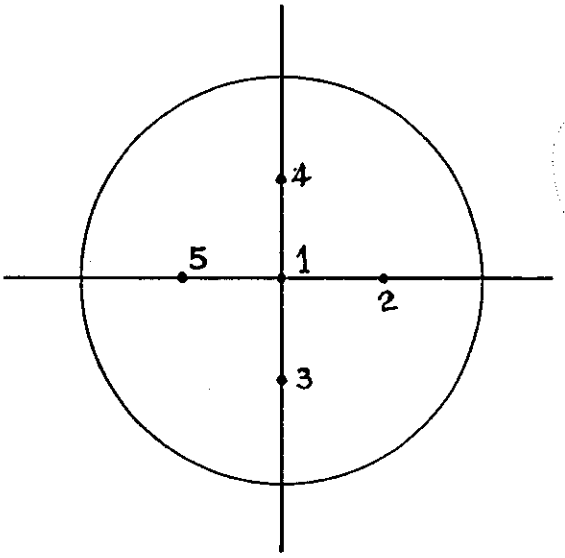
TABLE 4.7: ψ_3, ψ_4 for Channel 2

		$d_1 = 10^{-7}$			$d_1 = 10^{-10}$			$d_1 = 10^{-15}$			$d_1 = 10^{-20}$		
c = No of Iterations		0.5 70	0.7 56	1.0 38	0.5 99	0.7 67	1.0 40	0.5 131	0.7 85	1.0 42	0.5 164	0.7 105	1.0 45
No of taps in filter	20	-70.88 -44.05	-77.13 -44.09	-78.93 -44.08	-94.23 -44.08	-93.41 -44.08	-93.48 -44.08	-93.84 -44.08	-93.63 -44.08	-93.83 -44.08	-93.83 -44.08	-93.83 -44.08	-93.83 -44.08
	30	-70.68 -67.30	-77.44 -69.89	-78.77 -70.29	-100.66 -71.06	-108.87 -71.06	-108.83 -71.06	-148.81 -71.06	-148.23 -71.06	-158.96 -71.06	-200.86 -71.06	-199.02 -71.06	-214.16 -71.06
	40	-70.68 -69.70	-77.44 -76.08	-78.77 -78.13	-100.66 -95.78	-108.87 -97.55	-108.83 -97.60	-148.81 -98.02	-148.23 -98.02	-158.96 -98.02	-200.86 -98.02	-199.02 -98.02	-214.16 -98.02
	50	-70.68 -69.70	-77.44 -76.10	-78.77 -78.18	-100.66 -99.72	-108.87 -107.39	-108.83 -107.82	-148.81 -124.95	-148.23 -124.95	-158.96 -124.97	-200.86 -124.97	-199.02 -124.97	-214.16 -124.97
	60	-70.68 -69.70	-77.44 -76.10	-78.77 -78.18	-100.66 -99.73	-108.87 -107.47	-108.83 -107.91	-148.81 -146.44	-148.23 -146.27	-158.96 -150.99	-200.86 -151.93	-199.02 -151.93	-214.16 -151.93
	70	-70.68 -69.70	-77.44 -76.10	-78.77 -78.18	-100.66 -99.73	-108.87 -107.47	-108.83 -107.91	-148.81 -147.88	-148.23 -147.65	-158.96 -158.10	-200.86 -178.85	-199.02 -178.83	-214.16 -178.88
	80	-70.68 -69.70	-77.44 -76.10	-78.77 -78.18	-100.66 -99.73	-108.87 -107.47	-108.83 -107.91	-148.81 -147.88	-148.23 -147.65	-158.96 -158.14	-200.86 -199.09	-199.02 -197.85	-214.16 -205.24
	90	-70.68 -69.70	-77.44 -76.10	-78.77 -78.18	-100.66 -99.73	-108.87 -107.47	-108.83 -107.91	-148.81 -147.88	-148.23 -147.65	-158.96 -158.14	-200.86 -200.11	-199.02 -198.60	-214.16 -214.10
	100	-70.68 -69.70	-77.44 -76.10	-78.77 -78.18	-100.66 -99.73	-108.87 -107.47	-108.83 -107.91	-148.81 -147.88	-148.23 -147.65	-158.95 -158.14	-200.86 -200.12	-199.02 -198.60	-214.16 -214.15

TABLE 4.8: ψ_3, ψ_4 for Channel 3

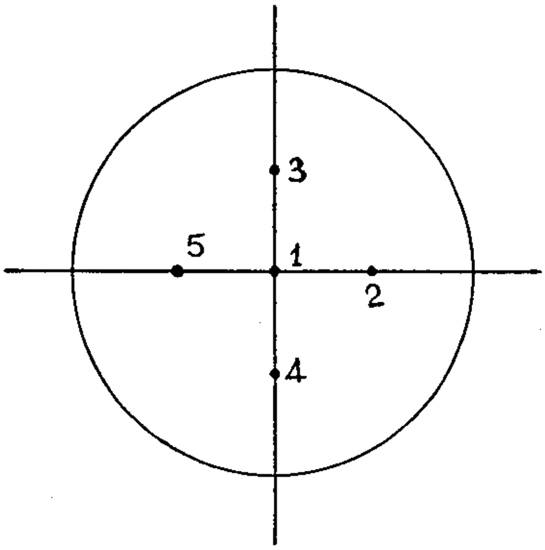
		$d_1 = 10^{-7}$			$d_1 = 10^{-10}$			$d_1 = 10^{-15}$			$d_1 = 10^{-20}$		
c = No of Iterations		0.5 155	0.7 116	1.0 114	0.5 195	0.7 139	1.0 118	0.5 259	0.7 176	1.0 123	0.5 326	0.7 216	1.0 128
No of taps in filter	20	-65.65 -22.22	-62.77 -22.20	-67.05 -22.21	-67.16 -22.21	-67.22 -22.21	-67.19 -22.21	-67.18 -22.21	-67.18 -22.21	-67.18 -22.21	-67.18 -22.21	-67.18 -22.21	-67.18 -22.21
	30	-72.68 -40.43	-63.81 -40.57	-85.78 -40.58	-102.79 -40.60	-94.69 -40.61	-119.61 -40.61	-150.95 -40.61	-149.37 -40.61	-197.60 -40.61	-206.30 -40.61	-198.13 -40.61	-205.10 -40.61
	40	-72.68 -39.35	-63.81 -39.22	-85.78 -39.27	-102.79 -39.28	-94.69 -39.28	-119.61 -39.28	-150.95 -39.28	-149.37 -39.28	-197.60 -39.28	-206.30 -39.28	-198.13 -39.28	-205.10 -39.28
	50	-72.68 -51.20	-63.81 -50.90	-85.78 -51.24	-102.79 -51.27	-94.69 -51.27	-119.61 -51.28	-150.95 -51.28	-149.37 -51.28	-197.60 -51.28	-206.30 -51.28	-198.13 -51.28	-205.10 -51.28
	60	-72.68 -54.59	-63.81 -54.00	-85.78 -54.57	-102.79 -54.60	-94.69 -54.60	-119.61 -54.60	-150.95 -54.60	-149.37 -54.60	-197.60 -54.60	-206.30 -54.60	-198.13 -54.60	-205.10 -54.60
	70	-72.68 -62.89	-63.81 -60.44	-85.78 -63.42	-102.79 -63.49	-94.69 -63.49	-119.61 -63.49	-150.95 -63.49	-149.37 -63.49	-197.60 -63.49	-206.30 -63.49	-198.13 -63.49	-205.10 -63.49
	80	-72.68 -67.25	-63.81 -62.53	-85.78 -69.13	-102.79 -69.27	-94.69 -69.26	-119.61 -69.28	-150.95 -69.27	-149.37 -69.27	-197.60 -69.27	-206.30 -69.27	-198.13 -69.27	-205.10 -69.27
	90	-72.68 -70.34	-63.81 -63.39	-85.78 -76.19	-102.79 -76.75	-94.69 -76.68	-119.61 -76.76	-150.95 -76.76	-149.37 -76.76	-197.60 -76.76	-206.30 -76.76	-198.13 -76.76	-205.10 -76.76
	100	-72.68 -71.17	-63.81 -63.56	-85.78 -81.39	-102.79 -83.40	-94.69 -83.13	-119.61 -83.46	-150.95 -83.46	-149.37 -83.46	-197.60 -83.46	-206.30 -83.46	-198.13 -83.46	-205.10 -83.46

TABLE 4.9: ψ_3, ψ_4 for Channel 4



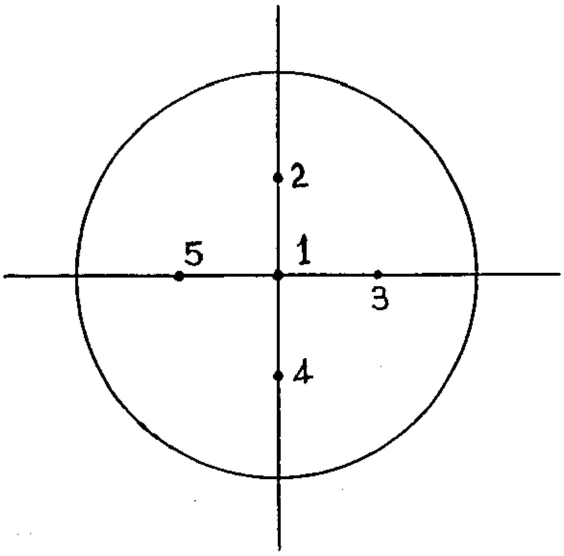
1. λ_o = 0.0 + j0.0
2. λ_o = 0.5 + j0.0
3. λ_o = 0.0 - j0.5
4. λ_o = 0.0 + j0.5
5. λ_o = -0.5 + j0.0

FIGURE 4.15: Arrangement A



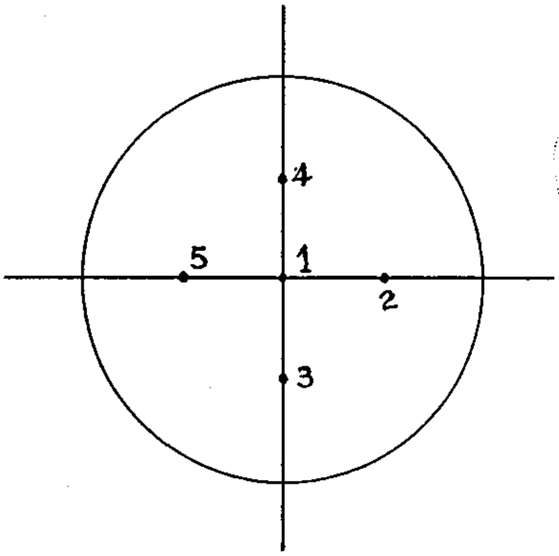
1. λ_o = 0.0 + j0.0
2. λ_o = 0.5 + j0.0
3. λ_o = 0.0 + j0.5
4. λ_o = 0.0 - j0.5
5. λ_o = -0.5 + j0.0

FIGURE 4.16: Arrangement B



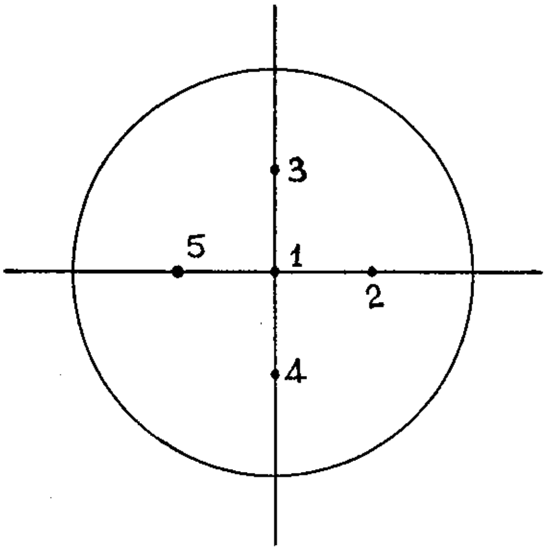
1. λ_o = 0.0 + j0.0
2. λ_o = 0.0 + j0.5
3. λ_o = 0.5 - j0.0
4. λ_o = 0.0 - j0.5
5. λ_o = -0.5 + j0.0

FIGURE 4.17: Arrangement C



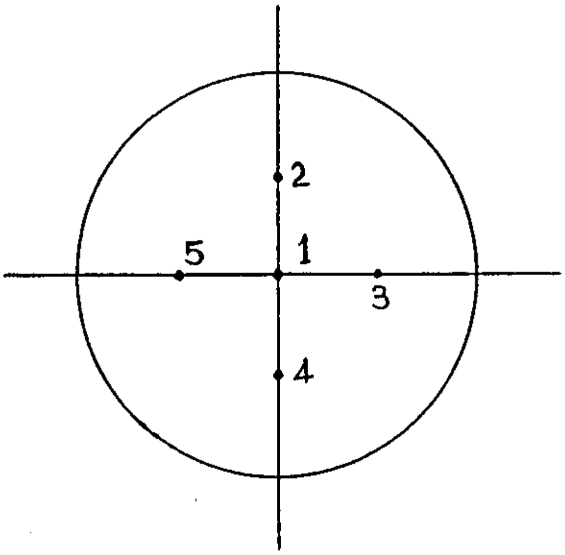
1. λ_o = 0.0 + j0.0
2. λ_o = 0.5 + j0.0
3. λ_o = 0.0 - j0.5
4. λ_o = 0.0 + j0.5
5. λ_o = -0.5 + j0.0

FIGURE 4.15: Arrangement A



1. λ_o = 0.0 + j0.0
2. λ_o = 0.5 + j0.0
3. λ_o = 0.0 + j0.5
4. λ_o = 0.0 - j0.5
5. λ_o = -0.5 + j0.0

FIGURE 4.16: Arrangement B



1. λ_o = 0.0 + j0.0
2. λ_o = 0.0 + j0.5
3. λ_o = 0.5 - j0.0
4. λ_o = 0.0 - j0.5
5. λ_o = -0.5 + j0.0

FIGURE 4.17: Arrangement C

	CHANNEL 1											
	$d_1 = 10^{-7}$			$d_1 = 10^{-10}$			$d_1 = 10^{-15}$			$d_1 = 10^{-20}$		
ARRANGEMENT	c=0.5	c=0.7	c=1.0	c=0.5	c=0.7	c=1.0	c=0.5	c=0.7	c=1.0	c=0.5	c=0.7	c=1.0
A	47	29	20	62	38	21	86	53	24	113	68	25
B	47	29	20	62	38	21	86	53	24	113	68	25
C	47	29	20	62	38	21	86	53	24	113	68	25

	CHANNEL 2											
	$d_1 = 10^{-7}$			$d_1 = 10^{-10}$			$d_1 = 10^{-15}$			$d_1 = 10^{-20}$		
ARRANGEMENT	c=0.5	c=0.7	c=1.0	c=0.5	c=0.7	c=1.0	c=0.5	c=0.7	c=1.0	c=0.5	c=0.7	c=1.0
A	59	42	36	78	54	40	112	74	41	144	92	44
B	59	42	36	78	54	40	112	74	41	144	92	44
C	59	42	36	78	54	40	112	74	41	144	92	44

	CHANNEL 3											
	$d_1 = 10^{-7}$			$d_1 = 10^{-10}$			$d_1 = 10^{-15}$			$d_1 = 10^{-20}$		
ARRANGEMENT	c=0.5	c=0.7	c=1.0	c=0.5	c=0.7	c=1.0	c=0.5	c=0.7	c=1.0	c=0.5	c=0.7	c=1.0
A	78	56	38	99	67	40	131	85	42	164	105	45
B	78	56	38	99	67	40	131	85	42	164	105	45
C	78	62	38	99	73	40	131	92	42	164	111	45

	CHANNEL 4											
	$d_1 = 10^{-7}$			$d_1 = 10^{-10}$			$d_1 = 10^{-15}$			$d_1 = 10^{-20}$		
ARRANGEMENT	c=0.5	c=0.7	c=1.0	c=0.5	c=0.7	c=1.0	c=0.5	c=0.7	c=1.0	c=0.5	c=0.7	c=1.0
A	155	116	114	195	139	118	259	176	123	326	216	128
B	155	125	111	195	145	114	259	183	119	326	223	125
C	172	134	111	212	159	114	277	197	119	346	235	125

TABLE 4.10: Number of Iterations Taken for the Different Start-up Arrangements

4.4 System 4.4

A major design feature which is common in all the different variations of the off-line systems described previously in this Chapter is that the basic operation, which the receiver employs for the root-finding process, requires both the two-tap feedforward transversal filter and the one-tap feedback filter working in cascade (Figure 4.1). This system (and those which will be described in the latter part of the report) differs from the previous systems in that only the one-tap feedback filter is used in the iterative root-finding process (Figure 4.18) and it is simpler to implement and requires fewer arithmetic operations per iteration.

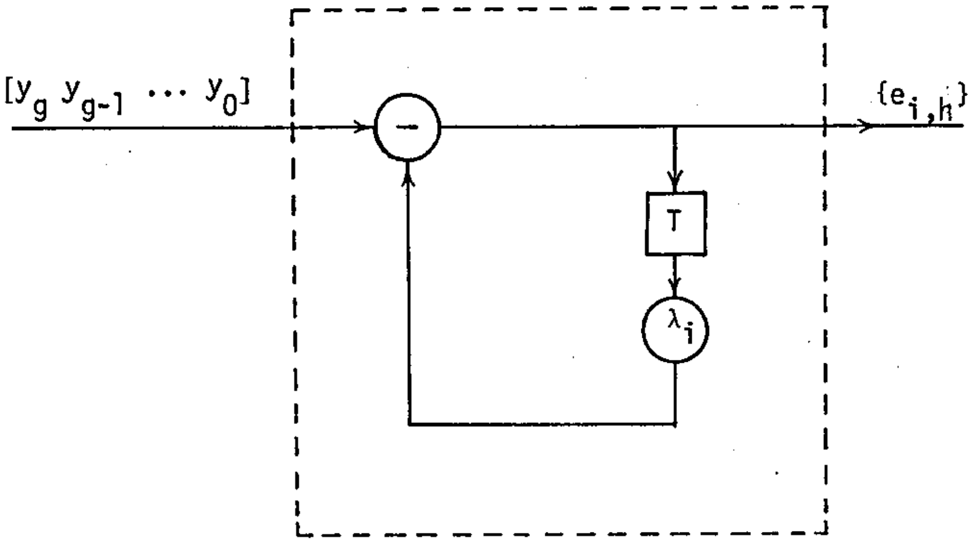


FIGURE 4.18: The One-tap Feedback Filter used in System 4.4

The basic principle behind the adjustment of the adaptive linear filter is as follows. The receiver first forms a filter with the z-transform

$$A_i(z) = (1 + \lambda_i z)^{-1} \tag{4.101}$$

for i = 0, 1, ..., k, in turn, using an iterative process to adjust λ_i so that, as i increases, $\lambda_i \rightarrow \beta_1$ (equation 2.8). β_1 is the negative of the reciprocal of the first of the roots of Y(z) to be processed by the system, and, of course $|\beta_1|<1$. Since the filter with z-transform $A_i(z)$ does not operate on the received signal in real time (as will be explained later), its z-transform is not limited to zero and negative powers of z. At the end of the iterative process, when i=k, the z-transform of the filter is

$$A_k(z) \simeq (1 + \beta_1 z)^{-1} \qquad (4.102)$$

The receiver now forms a filter with the z-transform

$$\begin{aligned} C_1(z) &= (1 + \lambda_k z)^{-1}(1 + \lambda_k^* z^{-1}) \\ &\simeq (1 + \beta_1 z)^{-1}(1 + \beta_1^* z^{-1}) \qquad (4.103) \end{aligned}$$

The whole of this process is carried out for each β_h (h=1,2,...,m), to give a total of m filters, with z-transforms $\{C_h(z)\}$, which are combined together (connected in cascade) and a delay of n-m sampling intervals is added. The m filters and associated delay are in fact implemented as a single filter whose z-transform now approximates to D(z) (equation 2.10). The operation to determine D(z) is performed on the estimate of Y (equation 2.3), which is held in store and is here assumed to be correct. In describing the action of the system, the sequence Y is, for convenience, treated as occurring in real time, with its first component y_0 at time t=0 and its last component y_g at time t=gT.

Suppose now that the z-transform of the sampled impulse-response of the channel is

$$Y(z) = y_0 + y_1 z^{-1} + \ldots + y_g z^{-g}$$

$$Y(z) = (1 + \beta_1 z)(u_0 z^{-1} + u_1 z^{-2} + \ldots u_{g-1} z^{-g}) \tag{4.104}$$

where $-\frac{1}{\beta_1}$ is a root (zero) of Y(z) lying outside the unit circle in the z-plane. Then

$$\begin{aligned} Y(z)A_i(z) &= (1+\beta_1 z)(1+\lambda_1 z)^{-1}(u_0 z^{-1} + u_1 z^{-2}+\ldots+ u_{g-1} z^{-g}) \\ &= (1+\beta_1 z)(1-\lambda_i z+\lambda_i^2 z^2-\lambda_i^3 z^3 + \ldots)(u_0 z^{-1}+u_1 z^{-2}+\ldots+u_{g-1} z^{-g}) \\ &= (1+(\beta_1-\lambda_i)z-\lambda_i(\beta_1-\lambda_i)z^2+\lambda_i^2(\beta_1-\lambda_i)z^3-\ldots)(u_0 z^{-1}+u_1 z^{-2} \\ &\qquad +\ldots+ u_{g-1} z^{-g}) \\ &= \{\ldots \\ &\quad +[\qquad\qquad\qquad\qquad +\ldots+u_{g-1}(-\lambda_i)^{g+1}(\beta_1-\lambda_i)]\, z^2 \\ &\quad +[\qquad\qquad -u_0\lambda_i(\beta_1-\lambda_i)+\ldots+u_{g-1}(-\lambda_i)^{g}(\beta_1-\lambda_i)]\, z^1 \\ &\quad +[\ +u_0(\beta_1-\lambda_i)-u_1\lambda_i(\beta_1-\lambda_i)+\ldots+u_{g-1}(-\lambda_i)^{g-1}(\beta_1-\lambda_i)]\, z^0 \\ &\quad +[u_0+u_1(\beta_1-\lambda_i)-u_2\lambda_i(\beta_1-\lambda_i)+\ldots+u_{g-1}(-\lambda_i)^{g-2}(\beta_1-\lambda_i)]\, z^{-1} \\ &\quad +[u_1+u_2(\beta_1-\lambda_i)-u_3\lambda_i(\beta_1-\lambda_i)+\ldots+u_{g-1}(-\lambda_i)^{g-3}(\beta_1-\lambda_i)]\, z^{-2} \\ &\quad +\ldots\} \end{aligned} \tag{4.105}$$

But if

$$Y(z)A_i(z) = \ldots + e_{i,-1}z + e_{i,0} + e_{i,1}z^{-1} + \ldots + e_{i,g}z^{-g} \qquad (4.106)$$

the $\{e_{i,h}\}$, for $-\infty < h \leqslant g$, form the sequence at the output of the filter with z-transform $A_i(z)$, when the sequence Y is fed into it. Since, as previously mentioned, the operation just described is not carried out in real time but on the stored sequence Y (equation 2.3), there is nothing to prevent the components $e_{i,-1}$, $e_{i,-2}, \ldots$ being non-zero. However, as it happens, these components are never used or even generated in the iterative process, which operates entirely on the sequence of g+1 components $e_{i,0}$, $e_{i,1}$, ..., $e_{i,g}$. Now,

$$e_{i,o} = u_o(\beta_1-\lambda_i)-u_1\lambda_i(\beta_1-\lambda_i)+u_2\lambda_i^2(\beta_1-\lambda_i)- \ldots + u_{g-1}(-\lambda_i)^{g-1}(\beta_1-\lambda_i)$$

$$= (\beta_1-\lambda_i)(u_o-u_1\lambda_i + u_2\lambda_i^2- \ldots + u_{g-1}(-\lambda_i)^{g-1}) \qquad (4.107)$$

$$e_{i,1} = u_o+u_1(\beta_1-\lambda_i)-u_2\lambda_i(\beta_1-\lambda_i)+ \ldots + u_{g-1}(-\lambda_i)^{g-2}(\beta_1-\lambda_i)$$

$$= u_o+(\beta_1-\lambda_i)(u_1-u_2\lambda_i + u_3\lambda_i^2- \ldots + u_{g-1}(-\lambda_i)^{g-2}) \qquad (4.108)$$

$$e_{i,2} = u_1+u_2(\beta_1-\lambda_i)-u_3\lambda_i(\beta_1-\lambda_i)+ \ldots + u_{g-1}(-\lambda_i)^{g-3}(\beta_1-\lambda_i)$$

$$= u_1+(\beta_1-\lambda_i)(u_2-u_3\lambda_i + u_4\lambda_i^2- \ldots + u_{g-1}(-\lambda_i)^{g-3}) \qquad (4.109)$$

and so on. Thus, when $\lambda_i \rightarrow \beta_1$,

$$e_{i,h} \simeq u_{h-1} \tag{4.110}$$

for h = 1,2,...,g, and, from equation 4.107

$$e_{i,0} = (\beta_1-\lambda_i)(u_o-u_1\lambda_i + u_2\lambda_i^2 - \dots + u_{g-1}(-\lambda_i)^{g-1})$$

$$\simeq (\beta_1-\lambda_i)(e_{i,1}-e_{i,2}\lambda_i + e_{i,3}\lambda_i^2 - \dots + e_{i,g}(-\lambda_i)^{g-1})$$

$$= (\beta_1-\lambda_i)\varepsilon_i \simeq 0 \tag{4.111}$$

where

$$\varepsilon_i = e_{i,1} - e_{i,2}\lambda_i + e_{i,3}\lambda_i^2 - \dots + e_{i,g}(-\lambda_i)^{g-1} \tag{4.112}$$

so that

$$\beta_1 \simeq \lambda_i + e_{i,o}/\varepsilon_i \tag{4.113}$$

This means that, if c is a positive real constant in the range 0 to 1, then

$$\lambda_{i+1} = \lambda_i + c\ e_{i,o}/\varepsilon_i \tag{4.114}$$

is a better estimate of β_1 than is λ_i. Furthermore, from equation 4.107, as the value λ_i approaches that of β_1, the quantity $e_{i,o}/\varepsilon_i$ becomes smaller and in the limit, when $\lambda_i = \beta_1$, this term becomes zero, which suggests the following stopping criterion for convergence

$$|e_{i,o}/\varepsilon_i|^2 < d_1 \qquad (4.115)$$

where d_1 is a small threshold value.

The restart procedure is the same as that used in System 4.3.1 where it employs a set of five possible starting points for λ_o as shown in Figure 4.10. The starting-points (0, 0.5, -j0.5, j0.5, -0.5) are shown in the correct order in which they are selected in the iterative root-finding process.

The receiver holds in store the sequence Y (equation 2.3) and an estimate λ_i of the quantity β_1 in equation 2.8. The first estimate of β_1, at the start of the process, is $\lambda_o = 0$. Having determined λ_i, the receiver appropriately adjusts the one-tap feedback transversal filter shown in Figure 4.18. This filter is stable so long as $|\lambda_i|<1$[7], which is always arranged to be so, and it has the z-transform

$$\frac{1}{1+\lambda_i z^{-1}} = (1+\lambda_i z^{-1})^{-1}$$

$$= 1 - \lambda_i z^{-1} + \lambda_i^2 z^{-2} - \lambda_i^3 z^{-3} + \ldots \qquad (4.116)$$

The stored sequence Y is now reversed in order, so that it starts with the component y_g, and it is fed through the feedback transversal filter. The sequence, passing through the filter in reverse order, is taken to be moving backwards in time, starting with the component y_g, at time t=gT, and ending with the component y_o, at time t=0. The delay of one sampling interval T in the feedback filter (Figure 2.3) now becomes an advance of T, with z-transform z. Thus the effective z-transform of the feedback filter becomes $A_i(z)$ in equation 4.101, and the output signal from the filter is the sequence of the $\{e_{i,h}\}$ in equation 4.106. Only the g+1 components $e_{i,o}$, $e_{i,1}$, ..., $e_{i,g}$ of this sequence are in fact generated.

An improved estimate of β_1 is now given by λ_{i+1} in equation 4.114 which is now used in place of λ_i in the feedback filter. The effective z-transform of this filter, when operating in the sequence Y in reverse order, is now

$$A_{i+1}(z) = (1 + \lambda_{i+1}z)^{-1} \tag{4.117}$$

and the coefficient of z^{-h} in $Y(z)A_{i+1}(z)$ is $e_{i+1,h}$. The constant c in equation 4.114 is set to unity to give the fastest possible convergence rate, but it might have to be reduced for the most unfavourable channels.

The iterative process continues in the manner described, until the term $e_{i,o}/\varepsilon_i$ in equation 4.114 satisfies

$$|e_{i,o}/\varepsilon_i|^2 < d_1$$

where d_1 is an appropriate small positive real constant, or else until either i=40 or $|\lambda_i|>1$, and in each case the process is terminated. The action taken when i=40 or $|\lambda_i|>1$ will be considered later. When $|e_{i,o}/\varepsilon_i|^2 < d_1$ is satisfied, the iterative process is taken to have converged. Let the value of i at convergence be k, so that

$$\lambda_k \simeq \beta_1 \tag{4.118}$$

The sequence obtained at the output of the feedback filter now has the z-transform

$$Y(z)A_k(z) \simeq e_{k,o} + e_{k,1}z^{-1} + \dots + e_{k,g}z^{-g}$$

$$\simeq u_o z^{-1} + u_1 z^{-2} + \dots + u_{g-1} z^{-g} \qquad (4.119)$$

where the $\{u_h\}$ are given by equation 4.104 and $e_{k,o} \simeq 0$.

The receiver next appropriately adjusts the two-tap feedforward transversal filter shown in Figure 2.2, which has the z-transform

$$B_k(z) = 1 + \lambda_k^* z^{-1} \qquad (4.120)$$

The sequence of the $\{e_{k,h}\}$, for $h = 0,1,\dots,g$, is now fed through this filter in the correct order, starting with the component $e_{k,o}$, which is taken to occur at time t=0. This gives the output sequence of g+2 components with z-transform

$$f_{1,-1} + f_{1,0}z^{-1} + \dots + f_{1,g}z^{-g-1} \qquad (4.121)$$

which is approximately equal to $Y(z)A_k(z)B_k(z)$, and where $f_{1,-1} \simeq 0$. The resultant effect on the sequence Y of the two filters (Figures 2.2 and 2.3), giving the sequence of the $\{f_{1,h}\}$, approximates to that of a single filter with z-transform

$$\begin{aligned} C_1(z) &= A_k(z)B_k(z) \\ &\simeq (1 + \beta_1 z)^{-1}\ (1 + \beta_1^{*} z^{-1}) \end{aligned} \qquad (4.122)$$

as in equation 4.103. This filter is an allpass network having the same basic properties as the ideal adaptive linear transversal filter with z-transform D(z). Finally, the output sequence of the $\{f_{1,h}\}$ is advanced by one place (sampling interval) and the first component, $f_{1,-1}$, discarded, to give the sequence F_1, with z-transform

$$\begin{aligned} F_1(z) &= f_{1,0} + f_{1,1}z^{-1} + \dots + f_{1,g}z^{-g} \\ &\simeq zC_1(z)Y(z) \end{aligned} \qquad (4.123)$$

For practical purposes, the linear factor $(1 + \beta_1 z)$ of Y(z), in equation 2.8 is replaced in $F_1(z)$ by the linear factor $(1 + \beta_1^{*} z^{-1})$. Thus the root $-\frac{1}{\beta_1}$ of Y(z) is replaced by the root $-\beta_1^{*}$, which is the complex conjugate of its reciprocal and lies inside the unit circle. $F_1(z)$ contains in addition an advance of one sampling interval. The sequence F_1, (with z-transform $F_1(z)$) is an estimate of the sampled impulse-response of the channel and adaptive linear transversal filter (Figure 2.1), when the z-transform of the latter is z $C_1(z)$ (equation 4.122).

It can be seen from equations 4.105 and 4.106

$$e_{i,h} = (-\lambda_i)^{-h}\ e_{i,o} \qquad (4.124)$$

for h = -1, -2, ... This means that whenever $e_{i,o} = 0$ then $e_{i,h} = 0$

for h = -1, -2, ... But it is evident from equations 4.101 and 4.106 that, under these conditions and so long as $\lambda_i \neq 0$, it must follow that $\lambda_i = \beta_1$, where β_1 is the negative of the reciprocal of a root of Y(z) (equation 2.8) and is not confined to being any particular one of the $\{\beta_h\}$. Thus, when $\lambda_i \neq 0$, a necessary and sufficient condition for $\lambda_i = \beta_1$ is that $e_{i,o} = 0$. This means that, if equation 4.110 holds and if $\varepsilon_i \approx 0$, which would prevent the computation of $1/\varepsilon_i$ for use in equation 4.114, there is in any case no need to compute the reciprocal since now $e_{i,o} \approx 0$ (equation 4.111) and the iterative process has in fact converged. All that is necessary here is to ensure that sufficient accuracy is available in the computation process so that, when $\varepsilon_i \to 0$, a sufficiently small value of ε_i can be computed to achieve the required accuracy of convergence. If equation 4.110 does not hold, which means the iterative process is not near convergence, then $e_{i,o}$ is not near zero and may even be quite large, but it is possible (at least in principle) for $\varepsilon_i \to 0$. This condition can however be handled by checking $e_{i,o}$, whenever $\varepsilon_i \to 0$. If it is found that $e_{i,o}$ is not also near zero, λ_i can be adjusted to increase the magnitude of ε_i to an adequate level in order to permit the iterative process to proceed. The effect has, however, not been observed in any of the tests so far carried out and it will therefore not be considered further here.

At every step in the iterative process $|\lambda_i|$ is checked. Whenever $|\lambda_i| > 1$ or when i = 40 (the roots were located in well under 40 iterations in all the tests), theiterative process is taken to have diverged and so is terminated. The process is then restarted with λ_0 set to the next of its five possible values, which, in the order of selection are 0, 0.5, -0.5j, 0.5j and 0.5. Neither the possible values of λ_0 nor the order of selection are necessarily optimum for any particular channel, but they give satisfactory performance with the given channels, and tests in Section 4.3.2 have suggested that the order of selection does not critically affect the performance of the iterative process over any channel. The maximum value of 40 set for i is a compromise between the need for an adequate number of iterations to determine a root and the need to minimise the time wasted when the iterative process has diverged.

The value of 40 is not necessarily optimum for any given application but tests have shown that it works satisfactorily in all cases.

When all five possible values of λ_0 have been selected and again $|\lambda_i|>1$ for some i, or else i reaches 40, it is assumed that there are no roots of Y(z) outside the unit circle (that is, m=0 in equation 2.8), and no change is made either to the adaptive linear transversal filter (whose initial adjustment has yet to be described) or to the estimate of the sampled impulse-response of the channel and filter, these being left as previously determined to give the final adjustment of the adaptive system.

To adjust the tap gains of the (n+1)-tap adaptive linear feedforward transversal filter (Figure 2.1), whose ideal z-transform is D(z) (equation 2.10), all tap gains of the filter are initially set to zero except for the last tap which is set to unity. Thus the initial z-transform of the filter is

$$D_0(z) = z^{-n} \tag{4.125}$$

and the initial z-transform of the channel and filter is $z^{-n}Y(z)$. When convergence has been obtained in the iterative process previously described, such that $\lambda_k \approx \beta_1$, the sequence D_0 (with n+1 components and the z-transform $D_0(z)$) is fed through the two-tap feedforward transversal filter (Figure 2.2) with z-transform $B_k(z)$ (equation 4.120), starting with the first component of D_0, to give an output sequence with n+2 components and the z-transform $D_0(z)B_k(z)$. The latter sequence is fed in reverse order, and starting with the last component, through the one-tap feedback transversal filter (Figure 2.3), whose effective z-transform is now $A_k(z)$ (equations 4.101 and 4.102) to give the output sequence with a z-transform that is approximately

$$D_0(z)C_1(z) = D_0(z)A_k(z)B_k(z) \tag{4.126}$$

When n+1 components of the output sequence have been obtained, the process is halted. These n+1 components, in the order in which they are received, are the coefficients of z^{-n-1}, z^{-n},..., z^{-1} in $D_0(z)C_1(z)$. The tap gain of the h^{th} tap of the adaptive filter is now set to the coefficient of z^{-h}, for h = 1,2,..., n+1, to give the required tap gains. The h^{th} tap of the filter is associated with a delay of h-1 sampling intervals, whose z-transform is z^{-h+1}. Thus the z-transform of the adaptive filter is approximately

$$D_1(z) = z\, D_0(z)C_1(z) \tag{4.127}$$

The whole of the above procedure just described, including the adjustment of λ_i and all following operations, is now repeated, but using $F_1(z)$ (equation 4.123) in place of Y(z), and $D_1(z)$ in place of $D_0(z)$. At the end of the iterative process here, $\lambda_k \simeq \beta_2$, and the values of λ_k, $F_1(z)$ and $D_1(z)$ determine $F_2(z)$ and $D_2(z)$, which are then used, in place of $F_1(z)$ and $D_1(z)$, for processing β_3. The system continues in this way until, on the h^{th} repetition of the whole procedure, no root $F_h(z)$ outside the unit circle is found, starting from any of the five possible values of λ_0 (Figure 4.10). In the latter case, $|\lambda_i|>1$ for some value of i or else i reaches 40, from each starting point. It is now assumed that all m roots of Y(z) outside the unit circle have been located, such that h=m, and this being so, in the z-transform of the channel and adaptive filter, all roots of Y(z) that lie outside the unit circle are replaced by the complex conjugates of their reciprocals. The z-transform of the adaptive filter is now approximately

$$D_m(z) \simeq D(z) \tag{4.128}$$

(equation 2.10), so that the z-transform of the channel and adaptive filter is

$$\begin{aligned} & a_o + a_1 z^{-1} + \ldots + a_{n+g} z^{-n-g} \\ & \simeq Y(z)\, D_m(z) \\ & \simeq F(z) \end{aligned} \qquad (4.129)$$

(equation 2.12), where $a_h \simeq 0$ for $h = 0,1,\ldots,n-1$. The estimate of the sampled impulse-response of the channel and adaptive filter, that is employed by the detector, is the sequence F_m, with z-transform

$$\begin{aligned} & F_m(z) = f_{m,o} + f_{m,1} z^{-1} + \ldots + f_{m,g} z^{-g} \\ & \simeq z^n F(z) \\ & = Y_1(z)\, Y_3(z) \end{aligned} \qquad (4.130)$$

(equations 2.7 and 2.11). The delay of n sampling intervals introduced by the adaptive filter is, for convenience, ignored here but must obviously be taken into account when comparing $Y(z)D_m(z)$ and $F_m(z)$.

The performance of the algorithm is measured by two parameters, θ_i and ϕ_i''. The parameter θ_i is here a measure of the error in λ_i, obtained when processing β_h (the negative of the reciprocal of the h^{th} root of $Y(z)$ outside the unit circle), and

$$\theta_i = 10 \log_{10} (|\beta_h - \lambda_i|^2) \tag{4.131}$$

The parameter ϕ_i'' is a measure of the error in the sequence

$$[e_{i,o} \quad e_{i,1} \; . \; . \; . \; e_{i,g}] \tag{4.132}$$

at the output of the one-tap feedback transversal filter, during the iterative adjustment of λ_i towards β_h. The correct value of this sequence, obtained when h-1 of the roots of Y(z) have been replaced by the complex conjugates of their reciprocals and $\lambda_i = \beta_h$ (thus locating the h^{th} root), is

$$[v_o \quad v_1 \; . \; . \; . \; v_g] \tag{4.133}$$

where $v_o + v_1 z^{-1} + \ldots + v_g z^{-g}$

$$= z^{h-1} \, Y(z)(1+\beta_h z)^{-1} \prod_{i=1}^{h-1} (1+\beta_i z)^{-1}(1+\beta_i^* z^{-1}) \tag{4.134}$$

and $v_o = 0$. When h = 1, $v_i = u_{i-1}$ (equation 4.104) for i = 1,2,...,g. Thus

$$\phi_i'' = 10 \log_{10} (\sum_{h=0}^{g} |e_{i,h} - v_h|^2) \tag{4.135}$$

Again, the two other parameters ψ_1 and ψ_2 (equations 4.96 and 4.97), which give the discrepancy between the actual sampled impulse-response of the channel and adaptive filter and that assumed by the detector, are

$$\psi_1 = 10 \log_{10} |\sum_{h=0}^{g}| a_{n+h} - f_{m,h}|^2 \tag{4.136}$$

$$\psi_2 = 10 \log_{10} (\sum_{h=0}^{n-1}|a_h|^2 + \sum_{h=0}^{g}| a_{n+h} - f_{m,h}|^2) \tag{4.137}$$

Thus, ψ_1 is a measure of the accuracy of the g+1 components of the estimate of the sampled impulse-response of the channel and adaptive filter that are used by the detector, and ψ_2 is a measure of the total discrepancy between the actual and assumed sampled impulse-response of the channel and filter, taking into account now all the components assumed to be zero by the detector. Also, for completeness, the parameters ψ_3 and ψ_4 are included where the discrepancy is compared between the ideal and the assumed sampled impulse-responses of the channel and adaptive filter. More precisely,

$$\psi_3 = 10 \log_{10} (\sum_{h=0}^{g} |f_{n+h} - f_{m,h}|^2) \tag{4.138}$$

and

$$\psi_4 = 10 \log_{10} (\sum_{h=0}^{n-1} |f_h|^2 + \sum_{h=0}^{g}|f_{n+h} - f_{m,h}|^2) \tag{4.139}$$

where $f_o + f_1z^{-1} + \ldots + f_{n+g}z^{-n-g}$

$$= F(z) \tag{4.140}$$

Extensive tests were carried out using the four channels, Channels 1-4 to determine the speed and the accuracy of the root-finding algorithm for different values of c (equation 4.114) and d_1 (equation 4.115). The variation of θ_i and ϕ_i'' with i for c = 1.0, $d_1 = 10^{-10}$ for the four channels are shown in Figures 4.19-4.22. The symbol associated with each curve in Figures 4.19-4.22 indicates the corresponding root of Y(z) in Figures 2.9-2.13 that is being operated on (tracked) by the iterative process and is clearly named after each symbol. The number which appears in parentheses after the name of the root (e.g. ROOT1(7)) is the number of iterations taken for the algorithm to locate that root. The order in which the roots are processed and the total number of iterations taken to find all the roots, which includes the number taken to search through the five possible starting-points after the last one has been found, are also given in the graphs. The same information is also presented in Tables 4.11 and 4.12 respectively for clarity. A further set of results is given in Tables 4.13-4.20, which shows the variation of ψ_1, ψ_2, ψ_3 and ψ_4 with different number of taps in the adaptive linear filter, ranging from 20 to 100 in steps of 10, for c = 0.5, 0.7 and 1.0, and $d_1 = 10^{-7}$, 10^{-10}, 10^{-15} and 10^{-20}. In each case, the total number of iterations taken to find all the roots is also included.

The results show that this technique operates very well for the four channels, and the accuracy in the location of the roots, given by the two parameters θ_i and ϕ_i'', seems to be limited only by the threshold value d_1 and the machine accuracy. [For example, at a threshold of 10^{-10}, both θ_i and ϕ_i'' were about -100 dB, which were subsequently reduced to approximately -200 dB for a threshold of 10^{-20}.] Furthermore, none of the roots of Y(z) outside the unit circle have been missed in any of the tests, so that the adaptive system could be used with a nonlinear equalizer, giving the optimum adjustment of the latter.

It can be seen from Table 4.11 that although there is a tendency for the roots with the larger absolute value to be processed first, the roots are not necessarily processed in the order of their absolute values. Indeed, the order in which they are located can be different for different values of c. However, the speed at which the algorithm converges to a root is very fast, and even for the most severely distorted channels, such as Channel 4, any one of its 8 roots can be found in less than 19 iterations using a stepsize of 1.0 and threshold of 10^{-10}, giving an estimate of $\{\beta_h\}$ accurate to 5 significant figures. Table 4.12 shows that the speed of the algorithm is greatly affected by the value c (equation 4.114) and that the fastest convergence is achieved when c=1 as expected. The differences in the speed differ for the different channels and for different d_1 (equation 4.115).

From the results given in Tables 4.13-4.20, it can be seen that the discrepancy between the actual sampled impulse-response of the channel and filter and that assumed by the detector, given by the parameters ψ_1 and ψ_2, and the discrepancy between the ideal and assumed sampled impulse-responses, given by ψ_3 and ψ_4, decreases as the number of taps in the adaptive linear filter increases from 20 to 100. Furthermore, ψ_1 is usually small compared with ψ_2 (Tables 4.13-4.16), so that no useful advantage would normally be gained by using the convolution of the estimate of the sampled impulse-response of the channel and the actual tap gains of the adaptive filter, to determine the g+1 components of the estimate of the sampled impulse-response of the channel and filter. The results also suggest, from the values of ψ_2, that with typical and poor channels (such as Channels 1-3), 30 taps should be sufficient for the adaptive linear filter, whereas, with very poor channels (such as Channel 4), 50 taps should be sufficient. A most significant result here is the accuracy to which the adaptive filter can be adjusted when operating over Channel 4. The error in the channel estimates is likely to be quite small[22] and should not unduly reduce the accuracy. Tests over Channel 4, using the conventional gradient algorithm and some improved variants of this developed by J.D. Harvy, have shown that these techniques do not lead to satisfactory adjustment of the adaptive filter and furthermore require an unduly long training period[22].

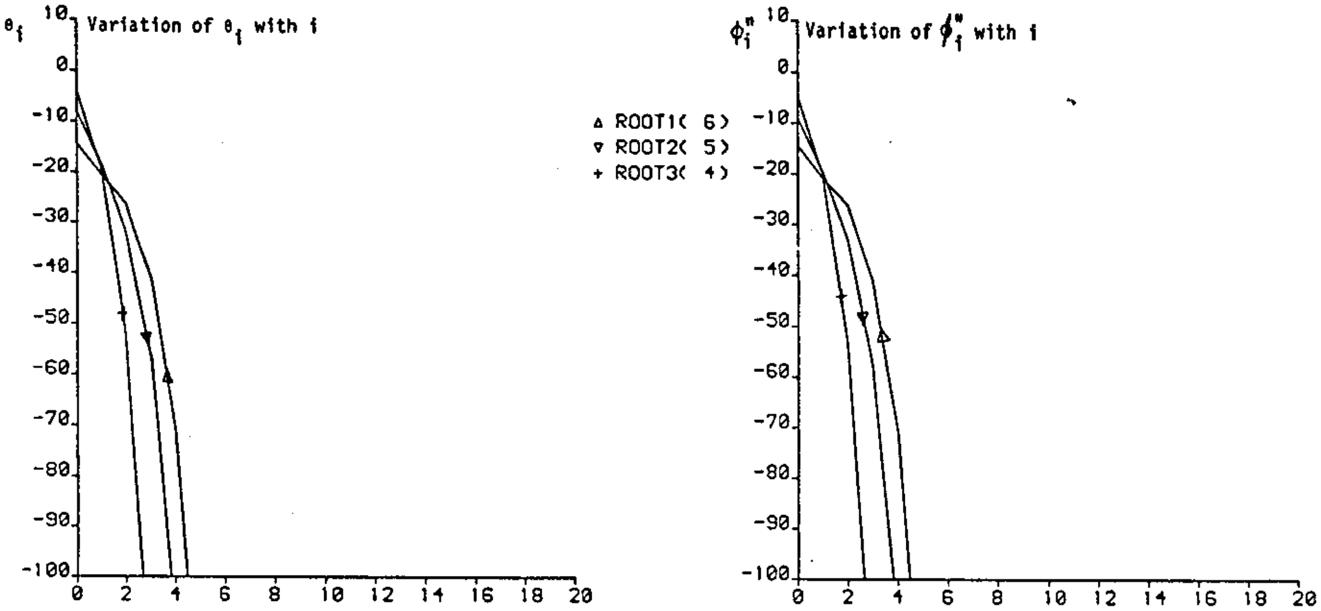


FIGURE 4.19: Performance of Standard algorithm with Channel 1 , c=1, $d_1=10^{-10}$, no. of iterations = 20 (SYSTEM 4.4)

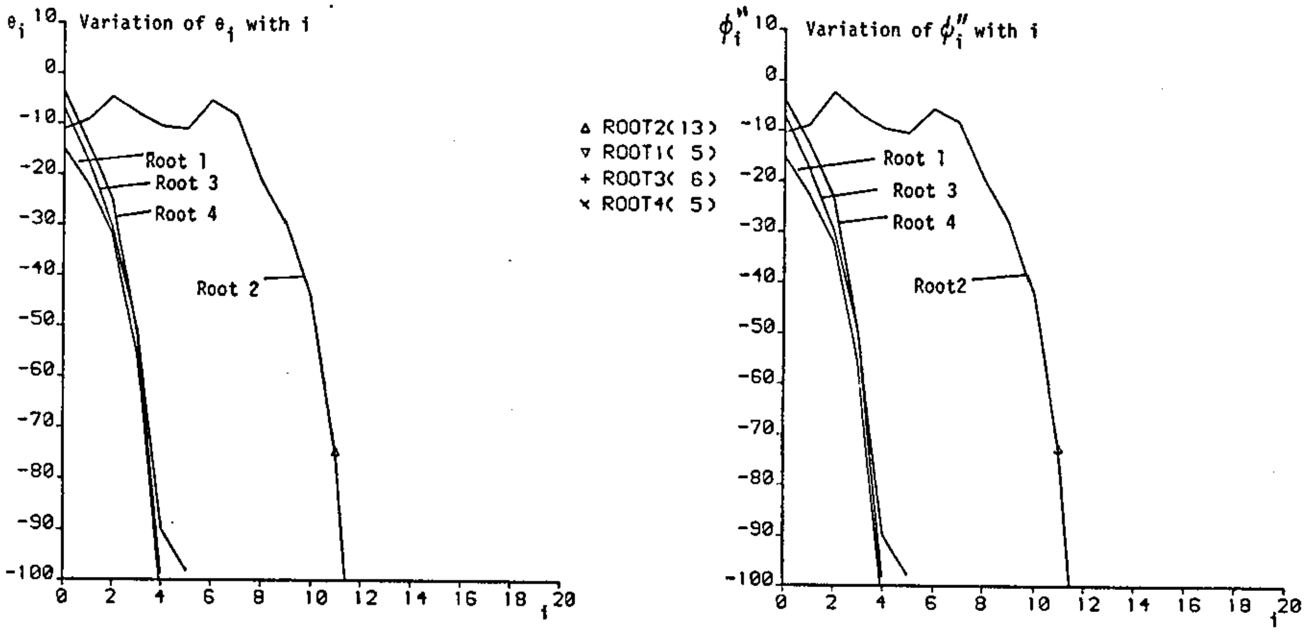


FIGURE 4.20: Performance of Standard algorithm with Channel 2, c=1, $d_1=10^{-10}$, no. of iterations = 34(SYSTEM 4.4)

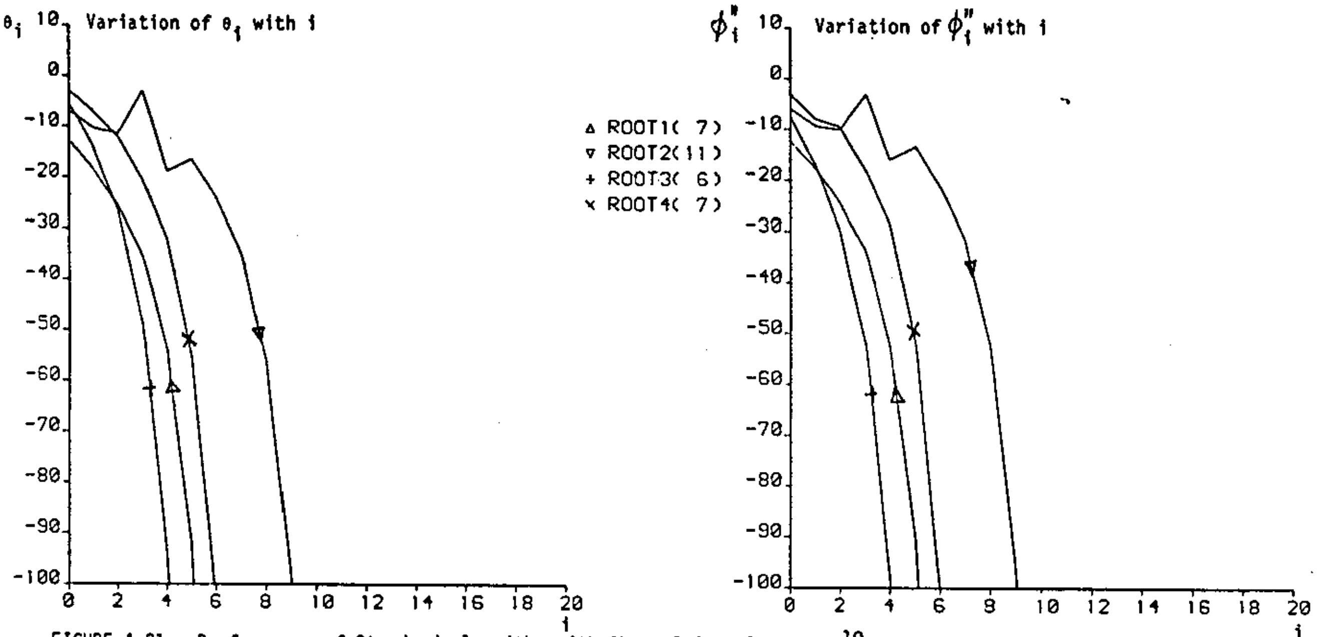


FIGURE 4.21: Performance of Standard algorithm with Channel 3, c=1, $d_1=10^{-10}$, no. of iterations = 39 (SYSTEM 4.4)

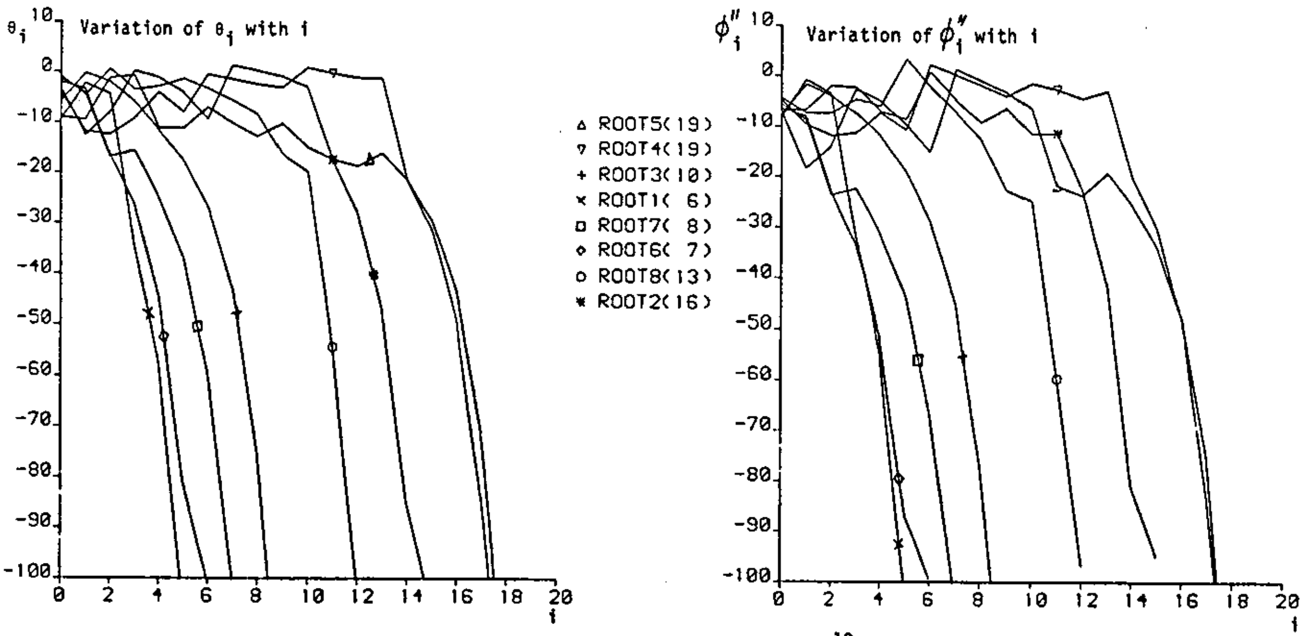


FIGURE 4.22: Performance of Standard algorithm with Channel 4, c=1, $d_1=10^{-10}$, no. of iterations = 112 (SYSTEM 4.4)

c = 1.0		$d_1 = 10^{-10}$	
Channel 1	Channel 2	Channel 3	Channel 4
1	1	1	5
2	2	2	4
3	3	3	3
	4	4	1
			7
			6
			8
			2

TABLE 4.11: Order in which the roots of Y(z) are processed

d_1 =	10^{-7}			10^{-10}			10^{-15}			10^{-20}		
c =	0.5	0.7	1.0	0.5	0.7	1.0	0.5	0.7	1.0	0.5	0.7	1.0
Channel 1	45	29	19	60	38	20	85	52	23	110	66	23
Channel 2	57	41	32	77	52	34	110	71	37	143	90	39
Channel 3	71	53	36	91	64	39	124	84	40	156	103	44
Channel 4	139	116	107	180	140	112	247	177	117	312	214	122

TABLE 4.12: Number of Iterations Taken for the Four Channels

	c = No of Iterations	$d_1=10^{-7}$			$d_1=10^{-10}$			$d_1=10^{-15}$			$d_1=10^{-20}$		
		0.5 45	0.7 29	1.0 19	0.5 60	0.7 38	1.0 20	0.5 85	0.7 52	1.0 23	0.5 110	0.7 66	1.0 23
No of taps in filter	20	-258.75 -57.22	-258.87 -57.28	-259.59 -57.33	-258.90 -57.31	-258.80 -57.31	-259.42 -57.31	-258.07 -57.31	-258.54 -57.31	-259.98 -57.31	-261.30 -57.31	-259.05 -57.31	-259.98 -57.31
	30	-258.75 -73.32	-258.87 -77.48	-259.59 -82.52	-258.90 -87.90	-258.80 -87.99	-259.42 -88.01	-258.07 -88.02	-258.54 -88.02	-259.98 -88.02	-261.30 -88.02	-259.05 -88.02	-259.98 -88.02
	40	-258.75 -73.47	-258.87 -77.88	-259.59 -83.95	-258.90 -103.45	-258.80 -108.79	-259.42 -114.06	-258.07 -118.72	-258.54 -118.72	-259.98 -118.73	-261.30 -118.73	-259.05 -118.73	-259.98 -118.73
	50	-258.75 -73.47	-258.87 -77.88	-259.59 -83.95	-258.90 -103.56	-258.80 -108.25	-258.88 -115.88	-258.07 -118.54	-258.54 -118.26	-259.98 -118.43	-261.30 -118.43	-259.05 -118.43	-259.98 -118.43
	60	-258.75 -73.47	-258.87 -77.88	-259.59 -83.95	-258.90 -103.58	-258.80 -109.25	-259.42 -115.88	-258.07 -148.65	-258.54 -148.52	-259.98 -149.14	-261.30 -149.12	-259.05 -149.13	-259.98 -149.14
	70	-258.75 -73.47	-258.87 -77.88	-259.59 -83.95	-258.90 -103.58	-258.90 -109.25	-259.42 -115.88	-258.07 -155.87	-258.54 -154.53	-259.98 -180.15	-261.30 -180.10	-259.05 -180.74	-259.98 -180.15
	80	-258.75 -73.47	-258.87 -77.88	-259.59 -83.95	-258.90 -103.58	-258.80 -109.25	-259.42 -115.88	-258.07 -155.87	-258.54 -154.53	-259.98 -210.41	-261.30 -302.63	-259.05 -204.96	-259.98 -210.41
	90	-258.75 -73.47	-258.87 -77.88	-259.59 -83.95	-258.90 -103.58	-258.80 -109.25	-259.42 -115.88	-258.07 -155.87	-258.54 -154.53	-259.98 -218.43	-261.30 -204.63	-259.05 -205.96	-259.98 -218.43
	100	-258.75 -73.47	-258.87 -77.88	-259.59 -83.95	-258.90 -103.58	-258.80 -109.25	-259.42 -115.88	-258.07 -155.87	-258.54 -154.53	-259.98 -218.43	-261.30 -204.63	-259.05 -205.96	-259.98 -218.43

TABLE 4.13: ϕ_1, ϕ_2 for Channel 1

	c= No of Iterations	$d_1=10^{-7}$			$d_1=10^{-10}$			$d_1=10^{-15}$			$d_1=10^{-20}$		
		0.5 57	0.7 41	1.0 32	0.5 77	0.7 52	1.0 34	0.5 110	0.7 71	1.0 37	0.5 143	0.7 90	1.0 39
No of taps in filter	20	-262.71 -49.22	-262.38 -49.26	-263.75 -49.24	-262.43 -49.25	-261.28 -49.25	-261.45 -49.25	-262.05 -49.25	-263.67 -49.25	-263.78 -49.25	-264.62 -49.25	-263.12 -49.25	-263.33 -49.25
	30	-262.71 -72.56	-262.38 -71.64	-263.75 -79.15	-262.43 -79.40	-261.28 -79.40	-261.45 -79.41	-262.05 -79.42	-q63.67 -79.42	-263.78 -79.42	-264.62 -79.42	-263.12 -79.42	-263.33 -79.42
	40	-262.71 -73.58	-262.38 -72.41	-263.75 -91.54	-262.43 -102.68	-261.28 -102.69	-261.45 -104.24	-262.05 -109.58	-263.67 -109.58	-263.78 -109.58	-264.62 -109.58	-263.12 -109.58	-263.33 -109.58
	50	-262.71 -73.58	-262.38 -72.41	-263.75 -91.61	-262.43 -103.67	-261.28 -103.68	-261.45 -105.74	-262.05 -139.51	-263.67 -139.60	-263.78 -139.73	-264.62 -139.73	-263.12 -139.73	-263.33 -139.73
	60	-262.71 -73.58	-262.38 -72.41	-263.78 -91.61	-262.43 -103.68	-261.28 -103.68	-261.45 -105.75	-262.05 -152.45	-263.67 -154.85	-263.78 -169.67	-264.62 -169.89	-263.12 -169.89	-263.33 -169.89
	70	-262.71 -73.58	-262.38 -72.41	-263.75 -91.61	-262.43 -103.68	-261.28 -103.68	-261.45 -105.75	-262.05 -152.53	-263.67 -154.99	-263.78 -182.61	-264.62 -198.90	-263.12 -199.05	-263.33 -199.77
	80	-262.71 -73.58	-262.38 -72.41	-263.75 -91.61	-262.43 -103.68	-261.28 -103.68	-261.45 -105.75	-262.05 -152.53	-263.67 -154.99	-263.78 -182.68	-264.62 -205.23	-263.12 -205.91	-263.33 -211.80
	90	-262.71 -73.58	-262.38 -72.41	-263.75 -91.61	-262.43 -103.68	-261.28 -103.68	-261.45 -105.75	-262.05 -152.53	-263.67 -154.99	-263.78 -182.68	-264.62 -205.24	-263.12 -205.92	-263.33 -211.87
	100	-262.71 -73.58	-262.38 -72.41	-263.75 -91.61	-262.43 -103.68	-261.28 -103.68	-261.45 -105.75	-262.05 -152.53	-263.67 -154.99	-263.78 -182.68	-264.62 -205.24	-263.12 -205.92	-263.33 -211.87

TABLE 4.14: ϕ_1, ϕ_2 for Channel 2

	c = No. of Iterations	$d_1=10^{-7}$			$d_1=10^{-10}$			$d_1=10^{-15}$			$d_1=10^{-20}$		
		0.5 71	0.7 53	1.0 36	0.5 91	0.7 64	1.0 39	0.5 124	0.7 84	1.0 40	0.5 156	0.7 103	1.0 44
No. of taps in filter	20	-93.84 -44.09	-93.83 -44.06	-93.83 -44.08	-93.83 -44.08	-93.83 -44.08	-93.83 -44.08	-93.83 -44.08	-93.83 -44.08	-93.83 -44.08	-93.83 -44.08	-93.83 -44.08	-93.83 -44.08
	30	-257.49 -70.24	-256.92 -69.92	-259.30 -71.06	-257.91 -71.06	-258.50 -71.06	-256.44 -71.06	-257.83 -71.06	-257.71 -71.06	-258.13 -71.06	-257.41 -71.06	-257.75 -71.06	-257.58 -71.06
	40	-257.49 -77.66	-256.92 -76.29	-259.30 -97.20	-257.91 -97.59	-258.50 -97.56	-256.44 -97.73	-257.83 -98.02	-257.71 -98.02	-258.13 -98.02	-257.41 -98.02	-257.75 -98.02	-257.58 -98.02
	50	-257.49 -77.70	-256.92 -76.32	-259.30 -104.78	-257.91 -107.71	-258.50 -107.45	-256.44 -109.49	-257.83 -124.97	-257.71 -124.97	-258.13 -124.97	-257.41 -124.97	-257.75 -124.97	-257.58 -124.97
	60	-257.49 -77.70	-256.92 -76.32	-259.30 -104.83	-257.91 -107.70	-258.80 -107.52	-256.44 -109.62	-257.83 -150.61	-257.71 -151.27	-258.13 -151.93	-257.41 -151.93	-257.75 -151.93	-257.58 -151.93
	70	-257.49 -77.70	-256.92 -76.32	-259.30 -104.83	-257.91 -107.80	-258.30 -107.52	-256.44 -109.62	-257.83 -156.43	-257.71 -159.76	-258.13 -178.66	-257.41 -178.87	-257.75 -178.88	-257.58 -178.88
	80	-257.49 -77.70	-256.92 -76.32	-259.30 -104.83	-257.91 -107.80	-258.50 -107.52	-256.44 -109.62	-257.83 -156.45	-257.71 -159.81	-258.13 -191.65	-257.41 -202.17	-257.75 -204.52	-257.58 -205.83
	90	-257.49 -77.70	-256.92 -76.32	-259.30 -104.83	-257.91 -107.80	-258.50 -107.52	-256.44 -109.62	-257.83 -156.45	-257.71 -159.81	-258.13 -191.82	-257.41 -204.61	-257.75 -210.34	-257.58 -232.77
	100	-257.49 -77.70	-256.92 -76.32	-259.30 -104.83	-257.91 -107.80	-258.50 -107.52	-256.44 -109.62	-257.83 -156.45	-257.71 -159.81	-258.13 -191.82	-257.41 -204.62	-257.75 -210.36	-257.58 -255.22

TABLE 4.15: ϕ_1, ϕ_2 for Channel 3

	c = No of Iterations	$d_1=10^{-7}$			$d_1=10^{-10}$			$d_1=10^{-15}$			$d_1=10^{-20}$		
		0.5 139	0.7 116	1.0 107	0.5 180	0.7 140	1.0 112	0.5 247	0.7 177	1.0 117	0.5 312	0.7 214	1.0 122
No of taps in filter	20	-67.19 -22.21	-67.18 -22.20	-67.18 -22.21	-67.18 -22.21	-67.18 -22.21	-67.18 -22.21	-67.18 -22.21	-67.18 -22.21	-67.18 -22.21	-67.18 -22.21	-67.18 -22.21	-67.18 -22.21
	30	-258.54 -40.50	-257.14 -40.69	-256.62 -40.59	-258.29 -40.60	-261.13 -40.61	-258.60 -40.61	-259.51 -40.61	-258.16 -40.61	-257.58 -40.61	-257.23 -40.61	-257.38 -40.61	-257.37 -40.61
	40	-258.54 -39.34	-257.14 -39.33	-256.62 -39.27	-258.29 -39.28	-261.13 -39.28	-258.60 -39.28	-259.51 -39.28	-258.16 -39.28	-257.58 -39.28	-257.23 -39.28	-257.38 -39.28	-257.37 -39.28
	50	-258.54 -51.27	-257.14 -51.46	-256.62 -51.24	-258.29 -51.28	-261.13 -51.28	-258.60 -51.27	-259.51 -51.28	-258.16 -51.28	-257.58 -51.28	-257.23 -51.28	-257.38 -51.28	-257.37 -51.28
	60	-258.54 -54.67	-257.14 -54.72	-256.62 -54.57	-258.29 -54.60	-261.13 -54.60	-258.60 -54.60	-259.51 -54.60	-258.16 -54.60	-257.58 -54.60	-257.23 -54.60	-257.38 -54.60	-257.37 -54.60
	70	-257.54 -63.44	-257.14 -63.62	-256.62 -63.43	-258.29 -63.49	-261.13 -63.50	-258.60 -63.49	-259.51 -63.49	-258.16 -63.49	-257.58 -63.49	-257.23 -63.49	-257.38 -63.49	-257.37 -63.49
	80	-258.54 -68.89	-257.14 -69.11	-256.62 -69.17	-258.29 -69.28	-261.13 -69.28	-258.60 -69.27	-259.51 -69.27	-258.16 -69.27	-257.58 -69.27	-257.23 -69.27	-257.38 -69.27	-257.37 -69.27
	90	-258.54 -74.68	-257.14 -75.18	-256.62 -76.43	-258.29 -76.76	-261.13 -76.77	-258.60 -76.75	-259.51 -76.76	-258.16 -76.76	-257.58 -76.76	-257.23 -76.76	-257.38 -76.76	-257.37 -76.76
	100	-258.54 -77.45	-257.14 -78.29	-256.62 - 82.28	-258.29 -83.45	-261.13 -83.46	-258.60 -83.46	-259.51 -83.46	-258.16 -83.46	-257.58 -83.46	-257.23 -83.46	-257.38 -83.46	-257.37 -83.46

TABLE 4.16: ϕ_1, ϕ_2 for Channel 4

		$d_1=10^{-7}$			$d_1=10^{-10}$			$d_1=10^{-15}$			$d_1=10^{-20}$		
	c =	0.5	0.7	1.0	0.5	0.7	1.0	0.5	0.7	1.0	0.5	0.7	1.0
	No. of Iterations	45	29	19	60	38	20	85	52	23	110	66	23
No. of taps in filter	20	-68.86 -56.93	-74.48 -57.19	-83.78 -57.32	-98.96 -57.31	-105.86 -57.31	-114.84 -57.31	-152.07 -57.31	-149.66 -57.31	-217.18 -57.31	-200.61 -57.31	-201.50 -57.31	-217.18 -57.31
	30	-68.86 -67.53	-74.48 -72.72	-83.78 -80.09	-98.96 -87.57	-105.86 -87.92	-114.84 -88.00	-152.07 -88.02	-149.66 -88.02	-217.18 -88.02	-200.61 -88.02	-201.50 -88.02	-217.18 -88.02
	40	-68.86 -67.57	-74.48 -72.85	-83.78 -80.85	-98.96 -97.64	-105.86 -104.07	-114.84 -111.43	-152.07 -118.72	-149.66 -118.72	-217.18 -118.73	-200.61 -118.73	-201.50 -118.73	-217.18 -118.73
	50	-68.86 -67.57	-74.48 -72.85	-83.78 -80.85	-98.96 -97.67	-105.86 -104.22	-114.84 -112.32	-152.07 -146.95	-149.66 -145.89	-217.18 -149.43	-200.61 -149.43	-201.50 -149.43	-217.18 -149.43
	60	-68.86 -67.57	-74.48 -72.85	-83.78 -80.85	-98.96 -97.67	-105.86 -104.22	-114.84 -112.32	-152.07 -150.55	-149.66 -148.43	-217.18 -180.14	-200.61 -180.08	-201.50 -180.09	-217.18 -180.14
	70	-68.86 -67.57	-74.48 -72.85	-83.78 -80.45	-98.96 -97.67	-105.86 -104.22	-114.84 -112.32	-152.07 -150.56	-149.66 -148.43	-217.18 -209.36	-200.61 -198.88	-201.50 -199.81	-217.18 -209.36
	80	-68.86 -67.57	-74.48 -72.85	-83.78 -80.85	-98.96 -97.67	-105.86 -104.22	-114.84 -112.32	-152.07 -150.56	-149.66 -148.43	-217.18 -214.74	-200.61 -199.16	-201.50 -200.17	-217.18 -214.74
	90	-68.86 -67.57	-74.48 -72.85	-83.78 -80.85	-98.96 -97.67	-105.86 -104.22	-114.84 -112.32	-152.07 -150.56	.149.66 -148.43	-217.18 -214.75	-200.61 -199.16	-201.50 -200.17	-217.18 -214.75
	100	-68.86 -67.57	-74.48 -72.85	-83.78 -80.85	-98.96 -97.67	-105.86 -104.22	-114.84 -112.32	-152.07 -150.56	-149.66 -148.43	-217.18 -214.75	-200.61 -199.16	-201.50 -200.17	-217.18 -214.75

TABLE 4.17: ψ_3, ψ_4 for Channel 1

		$d_1=10^{-7}$			$d_1=10^{-10}$			$d_1=10^{-15}$			$d_1=10^{-20}$		
	c =	0.5	0.7	1.0	0.5	0.7	1.0	0.5	0.7	1.0	0.5	0.7	1.0
	No. of Iterations	57	41	32	77	52	34	110	71	37	143	90	39
No of taps in filter	20	-70.02 -49.19	-69.59 -49.22	-91.08 -49.24	-100.12 -49.25	-100.91 -49.25	-102.11 -49.25	-148.86 -49.25	-152.38 -49.25	-181.90 -49.25	-202.06 -49.25	-204.10 -49.25	-208.94 -49.25
	30	-70.02 -68.10	-69.59 -67.48	-91.08 -78.88	-100.12 -79.37	-100.91 -79.37	-102.11 -79.39	-146.86 -79.42	-152.38 -79.42	-181.90 -79.42	-202.06 -79.42	-204.10 -79.42	-208.94 -79.42
	40	-70.02 -68.44	-69.59 -67.77	-91.08 -88.29	-100.12 -98.20	-100.91 -98.70	-102.11 -100.03	-148.86 -109.58	-152.38 -109.58	-181.90 -109.58	-202.06 -109.58	-204.10 -109.58	-208.94 -109.58
	50	-70.02 -68.44	-69.59 -67.77	-91.08 -88.33	-100.12 -98.53	-100.91 -99.07	-102.11 -100.54	-148.86 -139.03	-159.38 -139.38	-181.90 -139.73	-202.06 -139.73	-204.10 -139.73	-208.94 -139.73
	60	-70.02 -68.44	-69.59 -67.77	-91.08 -88.33	-100.12 -98.53	-100.91 -99.07	-102.11 -100.54	-148.86 -147.28	-152.38 -150.43	-181.90 -169.41	-202.06 -169.89	-204.10 -169.89	-208.94 -169.89
	70	-70.02 -68.44	-69.59 -67.77	-91.08 -88.33	-100.12 -98.53	-100.91 -99.07	-102.11 -100.54	-148.86 -147.30	-152.38 -150.48	-181.90 -179.23	-202.06 -197.19	-204.10 -197.87	-208.94 -199.27
	80	-70.02 -68.44	-69.59 -67.77	-91.08 -88.33	-100.12 -98.53	-100.91 -99.07	-102.11 -100.54	-148.86 -147.30	-152.38 -150.48	-181.90 -179.27	-202.06 -200.35	-204.10 -201.90	-208.94 -207.13
	90	-70.02 -68.44	-69.59 -67.77	-91.08 -88.33	-102.12 -98.53	-100.91 -99.07	-102.11 -100.54	-148.86 -147.30	-152.38 -150.48	-181.90 -179.27	-202.06 -200.35	-204.10 -201.91	-208.94 -207.15
	100	-70.02 -68.44	-69.59 -67.77	-91.08 -88.33	-100.12 -98.53	-100.91 -99.07	-102.11 -100.54	-148.86 -147.30	-152.38 -150.48	-181.90 -179.27	-202.06 -200.35	-204.10 -201.91	-208.94 -207.15

TABLE 4.18: ψ_3, ψ_4 for Channel 2

		$d_1=10^{-7}$			$d_1=10^{-10}$			$d_1=10^{-15}$			$d_1=20^{-20}$		
	c	0.5	0.7	1.0	0.5	0.7	1.0	0.5	0.7	1.0	0.5	0.7	1.0
	No. of Iterations	71	53	36	91	64	39	124	84	40	156	103	44
No. of taps in filter	20	-70.66 -44.08	-67.43 -44.04	-90.48 -44.08	-93.90 -44.08	-93.29 -44.08	-92.72 -44.08	-93.83 -44.08	-93.83 -44.08	-93.83 -44.08	-93.83 -44.08	-93.83 -44.08	-93.83 -44.08
	30	-70.53 -67.37	-67.36 -65.45	-94.45 -71.04	-100.65 -71.06	-98.56 -71.06	-102.67 -71.06	-150.11 -71.06	-150.85 -71.06	-183.12 -71.06	-197.97 -71.06	-202.01 -71.06	-213.99 -71.06
	40	-70.53 -69.76	-67.36 -66.84	-94.45 -92.60	-100.65 -95.84	-98.56 -95.02	-102.67 -96.52	-150.11 -98.02	-150.85 -98.02	-183.12 -98.02	-197.97 -98.02	-202.01 -98.02	-213.99 -98.02
	50	-70.53 -69.77	-67.36 -66.84	-94.45 -94.07	-100.65 -99.87	-98.56 -98.03	-102.67 -101.85	-156.11 -124.96	-150.85 -124.96	-183.12 -124.97	-197.97 -124.97	-202.01 -124.97	-213.99 -124.97
	60	-70.53 -69.77	-67.36 -66.84	-94.45 -94.07	-100.65 -99.88	-98.56 -98.04	-102.67 -101.87	-150.11 -148.35	-150.85 -148.04	-183.12 -151.92	-197.97 -151.93	-202.01 -151.93	-213.99 -151.93
	70	-70.53 -69.77	-67.36 -66.84	-94.45 -94.07	-100.65 -99.88	-98.56 -98.04	-102.67 -101.87	-150.11 -149.20	-150.85 -150.32	-183.12 -177.33	-197.97 -178.81	-202.01 -178.86	-213.99 -178.88
	80	-70.53 -69.77	-67.36 -66.84	-94.45 -94.07	-100.65 -99.88	-98.56 -98.04	-102.67 -101.87	-150.11 -149.21	-150.85 -150.33	-183.12 -182.55	-197.97 -196.57	-202.01 -200.07	-213.99 -205.22
	90	-70.53 -69.77	-67.35 -66.84	-94.45 -94.07	-100.65 -99.88	-98.56 -98.04	-102.67 -101.87	-150.11 -149.21	-150.85 -150.33	-183.12 -182.57	-197.97 -197.12	-202.01 -201.41	-213.99 -213.94
	100	-70.53 -69.77	-67.36 -66.84	-94.45 -94.07	-100.65 -99.88	-98.56 -98.04	-102.67 -101.87	-150.11 -149.21	-150.85 -150.33	-183.12 -182.57	-197.97 -197.12	-202.01 -201.42	-213.99 -213.95

TABLE 4.19: ψ_3,ψ_4 for Channel 3

		$d_1=10^{-7}$			$d_1=10^{-10}$			$d_1=10^{-15}$			$d_1=10^{-20}$		
	c =	0.5	0.7	1.0	0.5	0.7	1.0	0.5	0.7	1.0	0.5	0.7	1.0
	No. of Iterations	139	116	107	180	140	112	247	177	117	312	214	122
No. of taps in filter	20	-66.63 -22.21	-65.08 -22.20	-67.18 -22.21	-67.20 -22.21	-67.15 -22.21	-67.16 -22.21	-67.18 -22.21	-67.18 -22.21	-67.18 -22.21	-67.18 -22.21	-67.18 -22.21	-67.18 -22.21
	30	-72.78 -40.50	-70.58 -40.69	-80.26 -40.59	-101.55 -40.60	-101.80 -40.61	-103.12 -40.61	-149.34 -40.61	-153.03 -40.61	-152.26 -40.61	-194.83 -40.61	-194.40 -40.61	-204.40 -40.61
	40	-72.78 -39.34	-70.58 -39.32	-80.26 -39.27	-101.55 -39.28	-101.80 -39.28	-103.12 -39.28	-149.34 -39.28	-152.03 -39.28	-152.26 -39.28	-194.83 -39.28	-194.40 -39.28	-204.40 -39.28
	50	-72.78 -51.24	-70.58 -51.41	-80.26 -51.23	-101.55 -51.28	-101.80 -51.28	-103.12 -51.27	-149.34 -51.28	-153.03 -51.28	-152.26 -51.28	-194.83 -51.28	-194.40 -51.28	-204.40 -51.28
	60	-72.78 -54.60	-70.58 -54.60	-80.26 -54.56	-101.55 -54.60	-101.80 -54.60	-103.12 -54.60	-149.34 -54.60	-153.03 -54.60	-152.26 -54.60	-194.83 -54.60	-194.40 -54.60	-204.40 -54.60
	70	-72.78 -62.96	-70.58 -62.83	-80.26 -63.34	-101.55 -63.49	-101.80 -63.50	-103.12 -63.49	-149.34 -63.49	-153.03 -63.49	-152.26 -63.49	-194.83 -63.49	-194.40 -63.49	-204.40 -63.49
	80	-72.78 -67.40	-70.58 -66.77	-80.26 -68.85	-101.55 -69.27	-101.70 -69.28	-103.12 -69.27	-149.34 -69.27	-153.03 -69.27	-152.26 -69.27	-194.83 -69.27	-194.40 -69.27	-204.40 -69.27
	90	-72.78 -70.62	-70.58 -69.29	-80.26 -74.92	-101.55 -76.75	-101.80 -76.75	-103.12 -76.74	-149.34 -76.76	-153.03 -76.76	-152.26 -76.76	-194.83 -76.76	-194.40 -76.76	-204.40 -76.76
	100	-72.78 -71.51	-70.58 -69.90	-80.26 -78.14	-101.55 -83.39	-101.80 -83.40	-103.12 -83.41	-149.34 -83.46	-152.03 -83.46	-152.26 -83.46	-194.83 -83.46	-194.40 -83.46	-204.40 -83.46

TABLE 4.20: ψ_3 , ψ_4 for Channel 4

4.4.1 Adjustment Time of Adaptive Filter

So far, extensive tests carried out using computer-simulations have shown that the off-line system, System 4.4, is capable of dealing with the 16-point QAM system operating at 9600 bit/s. Test results of Channels 1-4 have shown that it is possible to achieve very fast speed of operation, and the accuracy of both the root-finding process and the adjustment of the filter taps seem to be only limited by the machine accuracy and the parameters employed. The system is far superior to the on-line systems (Chapter 3) in that it is simpler to implement, requires less computation time and most important of all, it avoids the problems caused by additive Gaussian noise in the received samples $\{r_i\}$ (equation 2.5), which is the main cause of failure in the on-line system. Many sets of results using a wide range of parameters are given, from which the optimum set can be chosen to suit any particular requirement on speed and accuracy.

The system, when operating with $c = 1.0$ (equation 4.114), $d_1 = 10^{-10}$ (equation 4.115) and with 50 taps in the adaptive linear filter, gives a good compromise between the speed and accuracy of the operation, and is here proposed for both the adjustment of the adaptive linear filter and the estimation of the resultant sampled impulse-response of the channel and filter for the 16-point QAM system operating at 9600 bit/s.

The adjustment time of the adaptive linear filter is the sum of the time taken for the algorithm to find all the roots outside the unit circle and the time required to form the filter. Each step of the iterative process involves 2g complex multiplications, (g+1 for the generation of $e_{i,o}$ $e_{i,1} \ldots e_{i,g}$ (equation 4.106) and g-1 for the evaluation of ε_i (equation 4.112)),and 1 complex division, $1/\varepsilon_i$. An additional 2n+g+2 complex multiplications are required for each root outside the unit circle, with only n+g+1 complex multiplications for the first root. This follows because, for each root outside the unit circle, g+1 complex multiplications are required to find the resultant sampled impulse-response of channel and filter (equation 4.121) and 2n+1 complex multiplications are required for the setting up of the linear filter (n for the two-tap feedforward filter and n+1 for the one-tap feedback filter). In the case of the first root, the setting up of the linear filter does not require the n complex multiplications for the two-tap feedforward filter since these correspond to passing an (n+1)-component sequence 000...001 through the filter.

Consider now an extreme case where there are 30 components in the sampled impulse-response of the channel, 8 roots of Y(z) outside the unit circle, 50 taps in the adaptive filter and a total of 120 iterations involved in locating the required roots of Y(z). The complex operation involved in adjusting the adaptive linear filter and estimating the sampled impulse-response of the channel and filter now entails some 8000 complex multiplications and the evaluation of 120 complex reciprocals. Bearing in mind that this operation is carried out immediately after the generation of an acceptable estimate of the sampled impulse-response of the channel, and uses no other input signals but the estimate, it should be possible to complete the adjustment of the adaptive filter, together with the estimation of the sampled impulse-response of the channel and filter, within some 10 ms after the receipt of the estimate of the sampled impulse-response of the channel. The operation of the latter estimate requires about 50 ms to ensure reasonable accuracy, giving a total period of some 60 ms for the complete adjustment of the channel estimator and adaptive filter in Figure 2.1, during the synchronization process at the start of transmission. This appears to be at least an order of magnitude better than is obtainable with more conventional (and not unduly complex) techniques, for the extremely unfavourable conditions assumed here.

4.5 System 4.5

This is a further modification of System 4.4 described previously, which takes advantage of the fact that the rate of convergence towards a root (zero) of $Y(z)$ outside the unit circle is very fast when $\lambda_i \approx \beta_1$, where λ_i is an estimate of the value β_1 which is the negative reciprocal of the root. When convergence is achieved, a condition indicated by $|e_{i,o}/\varepsilon_i|^2 < d_1$ (equation 4.115), it is possible to obtain the next estimate of β_1 (in effect a more accurate estimate of β_1) with only 1/3 of the number of iterations normally required.

When the iterative process is operating normally as described in Section 4.4, the estimate λ_{i+1} is updated according to

$$\lambda_{i+1} = \lambda_i + c\,\frac{e_{i,o}}{\varepsilon_i}$$

(equation 4.114) where c is a real positive constant, and

$$\varepsilon_i = e_{i,1} - e_{i,2}\lambda_i + e_{i,3}\lambda_i^3 - \dots + e_{i,g}(-\lambda_i)^{g-1}$$

$\{e_{i,h}\}$, $h = g, g-1, \dots, 0$ is the output sequence at the one-tap feedback filter when the sequence Y (equation 2.3) is fed through it in the reverse order such that it begins with y_g and finishes with y_o.

Now, at the k^th iteration, the tap gain in the feedback filter is set to λ_k, which produces the sequence

$$[\dots e_{k,o} \quad e_{k,1} \quad e_{k,2} \cdots e_{k,g}] \tag{4.141}$$

at its output when Y is fed through it in reverse order. Assuming now

$$|e_{k,o}/\varepsilon_k|^2 < d_1$$

where d_1 is small real positive constant. This indicates that the process has converged and so that the value λ_k is taken to be the value of β_1 and further tracking of β_1 is stopped. The receiver then takes the sequence $\{e_{k,h}\}$, $(h = 0,1,\dots,g)$, and feeds it through the two-tap linear feedforward filter (Figure 2.2) in the correct order, starting with the component $e_{k,o}$. The output sequence at the feedforward filter gives the resultant sampled impulse-response of the channel (with z-transform Y(z)) and filter (with z-transform $(1 + \beta_1 z)^{-1}(1 + \beta_1^* z^{-1})$).

So far, the estimate λ_k has been used in the calculation of ε_k for use in $|e_{k,o}/\varepsilon_k|^2 < d_1$, and the generation of the resultant sampled impulse-response of the channel and filter, which require g-1 and g complex multiplications, respectively.

Supposing now, when the condition

$$|e_{k,o}/\varepsilon_k|^2 < d_1 \qquad (4.142)$$

is satisfied and instead of terminating the iterative process, the new estimate for β_1 is obtained using

$$\lambda_{k+1} = \lambda_k + c\,\frac{e_{k,o}}{\varepsilon_k} \qquad (4.143)$$

and the sequence Y is fed through the one-tap feedback filter, with its tap set to λ_{k+1}, starting with the component y_g, to give the output sequence

$$[\dots e_{k+1,o} \quad e_{k+1,1} \cdots e_{k+1,g}] \tag{4.144}$$

When this sequence is fed through the two-tap feedforward filter, with its tap set to λ^*_{k+1}, in the manner described before, the resultant sampled impulse-response of channel and filter obtained at the output of the feedforward filter is much more accurate than that obtained previously because λ_{k+1} is a better estimate than λ_k and even more so because the rate of convergence to a root is greater as $\lambda_i \to \beta_1$.

The total number of complex operations required in this case is (g-1)+g+g = 3g-1 multiplications and 1 division. Since there is no need to test λ_{k+1} for convergence, this modification will give a more accurate estimate of β_1 and the resultant sampled impulse-response of the channel and filter at the expense of g extra complex multiplications and 1 complex division.

Results of computer-simulation tests for the four Channels 1-4 (Table 2.1) are given in Tables 4.21-4.28, which show that the values ψ_1, ψ_2, ψ_3, ψ_4 (equations 4.196-4.199) are better than those obtained previously, and the improvement is similar to cases where the threshold d_1 is made smaller by many orders of magnitude. Thus, it is possible to start with a higher threshold value ($d_1 = 10^{-7}$ say) and obtain an estimate of β_1 corresponding to a much smaller threshold (such as $d_1 = 10^{-10}$) but requires fewer number of iterations than it would by System 4.4 with the smaller d_1.

		$d_1 = 10^{-7}$			$d_1 = 10^{-10}$			$d_1 = 10^{-15}$			$d_1 = 10^{-20}$		
c =		0.5	0.7	1.0	0.5	0.7	1.0	0.5	0.7	1.0	0.5	0.7	1.0
No of Iterations		45	29	19	60	38	20	85	52	23	110	66	23
No of taps in filter	20	-260.07 -57.29	-258.82 -57.31	-258.08 -57.31	-260.09 -57.31	-258.26 -57.31	-258.92 -57.31	-259.05 -57.31	-257.50 -57.31	-258.81 -57.31	-258.74 -57.31	-257.22 -57.31	-258.81 -57.31
	30	-260.07 -78.93	-258.82 -85.17	-258.08 -88.02	-260.09 -87.99	-258.26 -88.02	-258.92 -88.02	-259.05 -88.02	-257.50 -88.02	-258.81 -88.02	-258.74 -88.02	-257.22 -88.02	-258.81 -88.02
	40	-260.07 -79.49	-258.82 -88.33	-258.08 -118.71	-260.09 -109.10	-258.26 -116.18	-258.92 -118.73	-259.05 -118.72	-257.50 -118.73	-258.81 -118.73	-258.74 -118.73	-257.22 -118.73	-258.81 -118.73
	50	-260.08 -79.50	-258.82 -88.34	-258.08 -142.79	-260.09 -109.60	-258.26 -119.71	-258.92 -49.43	-259.05 -149.19	-257.50 -149.31	-258.81 -149.43	-258.74 -149.43	-257.22 -149.43	-258.81 -149.43
	60	-260.07 79.50	-258.82 -88.34	-258.08 -143.85	-260.09 -109.60	-258.26 -119.71	-258.92 -180.14	-259.05 -161.83	-257.50 -164.86	-258.81 -180.14	-258.74 -180.13	-257.22 -180.14	-258.81 -180.14
	70	-260.07 -79.50	-258.82 -88.34	-258.08 -143.85	-260.09 -109.60	-258.25 -119.71	-258.92 -210.15	-259.05 -161.89	-257.50 -164.99	-258.81 -210.84	-258.74 -207.74	-257.22 -209.78	-258.81 -210.84
	80	-260.07 -79.50	-258.82 -88.34	-258.08 -143.85	-260.09 -109.60	-258.26 -119.71	-258.92 -218.41	-259.05 -161.89	-257.50 -164.99	-258.81 -241.46	-258.74 -210.65	-257.22 -214.61	-258.81 -241.46
	90	-260.07 -79.50	-258.62 -88.34	-258.08 -143.85	-260.09 -109.60	-258.26 -119.71	-258.92 -218.43	-259.05 -161.89	-257.50 -164.99	-258.81 -258.29	-258.74 -210.65	-257.22 -216.43	-258.81 -258.29
	100	-260.07 -79.50	-258.82 -88.34	-258.08 -143.85	-260.09 -109.60	-258.26 -119.71	-258.92 -218.43	-259.05 -161.89	-257.50 -164.99	-258.81 -258.47	-258.74 -210.65	-257.22 -216.43	-258.81 -258.47

TABLE 4.21: ψ_1, ψ_2 for Channel 1

		$d_1 = 10^{-7}$			$d_1 = 10^{-10}$			$d_1 = 10^{-15}$			$d_1 = 10^{-20}$		
c =		0.5	0.7	1.0	0.5	0.7	1.0	0.5	0.7	1.0	0.5	0.7	1.0
No of Iterations		57	41	32	77	52	34	110	71	37	143	90	39
No of taps in filter	20	-263.59 -49.24	-261.80 -49.26	-263.22 -49.25	-264.34 -49.25	-264.59 -49.25	-264.80 -49.25	-265.23 -49.25	-264.85 -49.25	-265.36 -49.25	-265.03 -49.25	-262.62 -49.25	-265.57 -49.25
	30	-263.59 -76.49	-261.80 -77.82	-263.22 -79.42	-264.34 -79.41	-264.59 -79.42	-264.80 -79.42	-265.23 -79.42	-264.85 -79.42	-265.36 -79.42	-265.03 -79.42	-262.62 -79.42	-265.57 -79.42
	40	-263.59 -79.59	-261.80 -82.87	-263.22 -109.58	-264.34 -106.62	-264.59 -108.27	-264.80 -109.58	-265.23 -109.58	-264.85 -109.58	-265.36 -109.58	-265.03 -109.58	-262.62 -109.58	-265.57 -109.58
	50	-263.59 -79.60	-261.80 -82.88	-263.22 -139.54	-264.34 -109.69	-264.59 -114.13	-264.80 -139.73	-265.23 -139.68	-264.85 -139.72	-265.36 -139.73	-265.03 -139.73	-262.62 -139.73	-265.57 -139.73
	60	-263.59 -79.60	-261.80 -82.88	-263.22 -153.01	-264.34 -109.70	-264.59 -114.14	-264.80 -169.89	-265.23 -158.24	-264.85 -164.11	-265.36 -169.89	-265.03 -169.89	-262.62 -169.89	-265.57 -169.89
	70	-263.59 -79.60	-261.80 -82.88	-263.22 -153.10	-264.34 -109.70	-264.59 -114.14	-264.80 -199.01	-265.23 -158.55	-264.85 -165.44	-265.36 -200.05	-265.03 -199.73	-262.62 -199.95	-265.57 -200.05
	80	-263.59 -79.60	-261.80 -82.88	-263.22 -153.10	-264.34 -109.70	-264.59 -114.14	-264.80 -205.73	-265.23 -158.55	-264.85 -165.45	-265.36 -230.20	-265.03 -211.20	-262.62 -216.20	-265.57 -230.20
	90	-263.59 -79.60	-261.80 -82.88	-263.22 -153.10	-264.34 -109.70	-264.59 -114.14	-264.80 -205.75	-265.23 -158.55	-264.85 -165.45	-265.36 -258.72	-265.03 -211.26	-262.62 -216.38	-265.57 -258.72
	100	-263.59 -79.60	-261.80 -82.88	-263.22 -153.10	-264.34 -109.70	-264.59 -114.14	-264.80 -205.75	-265.23 -158.55	-264.85 -165.45	-265.36 -263.73	-265.03 -211.26	-262.62 -216.38	-265.57 -263.74

TABLE 4.22: ψ_1, ψ_2 for Channel 2

		$d_1 = 10^{-7}$			$d_1 = 10^{-10}$			$d_1 = 10^{-15}$			$d_1 = 10^{-20}$		
c =		0.5	0.7	1.0	0.5	0.7	1.0	0.5	0.7	1.0	0.5	0.7	1.0
No of Iterations		71	53	36	91	64	39	124	84	40	157	103	44
No of taps in filter	20	-93.84 -44.09	-93.83 -44.07	-93.83 -44.08	-93.83 -44.08	-93.83 -44.08	-93.83 -44.08	-93.83 -44.08	-93.83 -44.08	-93.83 -44.08	-93.83 -44.08	-93.83 -44.08	-93.83 -44.08
	30	-259.30 -70.85	-259.51 -70.94	-257.21 -71.06	-258.69 -71.07	-258.03 -71.06	-258.15 -71.06	-258.85 -71.06	-259.10 -71.06	-258.74 -71.06	-257.42 -71.06	-258.97 -71.06	-258.69 -71.06
	40	-259.30 -83.56	-259.51 -86.38	-257.21 -98.02	-258.69 -97.91	-258.03 -97.98	-258.15 -98.02	-258.85 -98.02	-259.10 -98.02	-258.74 -98.02	-257.42 -98.02	-258.97 -98.02	-258.69 -98.02
	50	-259.30 -83.72	-259.51 -86.69	-257.21 -124.97	-258.69 -113.50	-258.03 -117.19	-258.15 -124.97	-258.85 -124.97	-259.10 -124.97	-258.74 -124.97	-257.42 -124.97	-258.97 -124.97	-258.69 -124.97
	60	-259.30 -83.72	-259.51 -86.69	-257.21 -151.93	-258.69 -113.82	-258.03 -117.98	-258.15 -151.93	-258.85 -151.56	-259.10 -151.86	-258.74 -151.93	-257.42 -151.93	-258.97 -151.93	-258.69 -151.93
	70	-259.30 -83.72	-259.51 -86.69	-257.21 -178.66	-258.69 -113.82	-258.03 -117.98	-258.15 -178.87	-258.85 -162.38	-259.10 -169.71	-258.74 -178.88	-257.42 -178.88	-258.97 -178.88	-258.69 -178.88
	80	-259.30 -83.72	-259.51 -86.69	-257.21 -191.65	-258.69 -113.82	-258.03 -117.98	-258.15 -202.33	-258.85 -162.47	-259.10 -170.27	-258.74 -205.83	-257.42 -204.65	-258.97 -205.70	-258.65 -205.83
	90	-259.30 -83.72	-259.51 -86.69	-257.21 -191.82	-258.69 -113.82	-258.03 -117.98	-258.15 -204.90	-258.85 -162.47	-259.10 -170.27	-258.74 -232.77	-257.42 -210.83	-258.97 -220.53	-258.69 -232.77
	100	-259.30 -83.72	-259.51 -86.69	-257.21 -191.82	-258.69 -113.82	-258.03 -117.98	-258.15 -204.91	-258.85 -162.47	-259.10 -170.27	-258.74 -255.76	-257.42 -210.86	-258.97 -220.80	-258.69 -255.71

TABLE 4.23: ψ_1, ψ_2 for Channel 3

		$d_1 = 10^{-7}$			$d_1 = 10^{-10}$			$d_1 = 10^{-15}$			$d_1 = 10^{-20}$		
c =		0.5	0.7	1.0	0.5	0.7	1.0	0.5	0.7	1.0	0.5	0.7	1.0
No of Iterations		139	116	107	180	140	112	247	177	117	312	214	122
No of taps in filter	20	-67.18 -22.21	-67.18 -22.20	-67.18 -22.21	-67.18 -22.21	-67.18 -22.21	-67.18 -22.21	-67.18 -22.21	-67.18 -22.21	-67.18 -22.21	-67.18 -22.21	-67.18 -22.21	-67.18 -22.21
	30	-258.74 -40.56	-255.20 -40.63	-257.03 -40.61	-256.59 -40.61	-256.11 -40.61	-254.76 -40.61	-256.87 -40.61	-255.13 -40.61	-255.16 -40.61	-255.61 -40.61	-254.23 -40.61	-254.76 -40.61
	40	-258.74 -39.31	-255.20 -39.29	-257.03 -39.28	-256.59 -39.28	-256.11 -39.28	-254.76 -39.28	-256.87 -39.28	-255.13 -39.28	-255.16 -39.28	-255.61 -39.28	-254.23 -39.28	-254.79 -39.28
	50	-258.74 -51.28	-255.20 -51.33	-257.03 -51.28	-256.59 -51.28	-256.11 -51.28	-254.76 -51.28	-256.87 -51.28	-255.13 -51.28	-255.15 -51.28	-255.61 -51.28	-254.23 -51.28	-254.79 -51.28
	60	-258.74 -54.64	-255.20 -54.64	-257.03 -54.60	-256.59 -54.60	-256.11 -54.60	-254.76 -54.60	-256.87 -54.60	-255.13 -54.60	-255.16 -54.60	-255.61 -54.60	-254.23 -54.60	-254.79 -54.60
	70	-258.74 -63.51	-255.20 -63.55	-257.03 -63.49	-256.59 -63.39	-256.11 -63.49	-254.76 -63.49	-256.87 -63.49	-255.13 -63.49	-255.16 -63.49	-255.61 -63.49	-254.23 -63.49	-254.79 -63.49
	80	-258.74 -69.21	-255.20 -69.31	-257.03 -69.27	-256.59 -69.28	-256.11 -69.28	-254.76 -69.27	-256.87 -69.27	-255.13 -69.27	-255.16 -69.27	-255.61 -69.27	-254.23 -69.27	-254.79 -69.27
	90	-258.74 -76.17	-255.20 -76.65	-257.03 -76.76	-256.59 -76.76	-256.11 -76.76	-254.76 -76.76	-256.87 -76.76	-255.13 -76.76	-255.16 -76.76	-255.61 -76.76	-254.23 -76.76	-254.79 -76.76
	100	-258.74 -81.05	-255.20 -82.69	-257.03 -83.46	-256.59 -83.46	-256.11 -83.47	-254.76 -83.46	-256.87 -83.46	-255.13 -83.46	-255.16 -83.46	-255.61 -83.46	-254.23 -83.46	-254.79 -83.46

TABLE 4.24: ψ_1, ψ_2 for Channel 4

		$d_1 = 10^{-7}$			$d_1 = 10^{-10}$			$d_1 = 10^{-15}$			$d_1 = 10^{-20}$		
	c =	0.5	0.7	1.0	0.5	0.7	1.0	0.5	0.7	1.0	0.5	0.7	1.0
No of	Iterations	45	29	19	60	38	20	85	52	23	110	66	23
No of taps in filter	20	-74.88 -57.22	-84.94 -57.30	-143.71 -57.31	-104.98 -57.31	-116.31 -57.31	-217.18 -57.31	-158.09 -57.31	-160.11 -57.31	-228.96 -57.31	-206.65 -57.31	-211.89 -57.31	-228.96 -57.31
	30	-74.88 -73.44	-84.94 -82.04	-143.71 -88.02	-104.98 -87.90	-116.31 -88.01	-217.18 -88.02	-158.09 -88.02	-160.11 -88.02	-228.96 -88.02	-206.65 -88.02	-211.89 -88.02	-228.96 -88.02
	40	-74.88 -73.59	-84.94 -83.30	-143.71 -118.70	-104.98 -103.56	-116.31 -113.24	-217.18 -118.73	-158.09 -118.72	-160.11 -118.72	-228.96 -118.73	-206.65 -118.73	-211.89 -118.73	-228.96 -118.73
	50	-74.88 -73.59	-84.94 -83.30	-143.71 -140.22	-104.98 -103.69	-116.31 -114.68	-217.18 -149.43	-158.09 -148.66	-160.11 -148.97	-228.96 -149.43	-206.65 -149.43	-211.89 -149.43	-228.96 -149.43
	60	-74.88 -73.59	-84.94 -83.30	-143.71 -140.77	-104.98 -103.69	-116.31 -114.68	-217.18 -180.14	-158.09 -156.56	-160.11 -158.86	-228.96 -180.14	-206.65 -180.12	-211.89 -180.13	-228.96 -180.14
	70	-74.88 -73.59	-84.94 -83.30	-143.71 -140.77	-104.98 -103.69	-116.31 -114.68	-217.18 -209.36	-158.09 -156.58	-160.11 -158.89	-228.96 -210.78	-206.65 -204.15	-211.89 -207.70	-228.96 -210.78
	80	-74.88 -73.59	-84.94 -83.30	-143.71 -140.77	-104.98 -103.69	-116.31 -114.68	-217.18 -214.74	-158.09 -156.58	-160.11 -158.89	-228.96 -228.73	-206.65 -205.19	-211.89 -210.58	-228.96 -228.73
	90	-74.88 -73.59	-84.94 -83.30	-143.71 -140.77	-104.98 -103.69	-116.31 -114.68	-217.18 -214.75	-158.09 -156.58	-160.11 -158.89	-228.96 -228.96	-206.65 -205.20	-211.89 -210.58	-228.96 -228.96
	100	-74.88 -73.59	-84.94 -83.30	-143.71 -140.77	-104.98 -103.69	-116.31 -114.68	-217.18 -214.75	-158.09 -156.58	-160.11 -158.89	-228.96 -228.96	-206.65 -205.20	-211.89 -210.58	-228.96 -228.96

TABLE 4.25: ψ_3, ψ_4 for Channel 1

		$d_1 = 10^{-7}$			$d_1 = 10^{-10}$			$d_1 = 10^{-15}$			$d_1 = 10^{-20}$		
	c =	0.5	0.7	1.0	0.5	0.7	1.0	0.5	0.7	1.0	0.5	0.7	1.0
No of	Iterations	57	41	32	77	52	34	110	71	37	143	90	39
No of taps in filter	20	-76.04 -49.23	-80.05 -49.26	-153.11 -49.25	-106.14 -49.25	-111.37 -49.25	-205.26 -49.25	-154.88 -49.25	-162.84 -49.25	-228.89 -49.25	-208.16 -49.25	-214.66 -49.25	-228.92 -49.25
	30	-76.04 -73.25	-80.05 -75.78	-153.11 -79.42	-106.14 -79.41	-111.37 -79.42	-205.26 -79.42	-154.88 -79.42	-162.84 -79.42	-228.89 -79.42	-208.16 -79.42	-214.66 -79.42	-228.92 -79.42
	40	-76.04 -74.45	-80.05 -78.23	-153.11 -109.58	-106.14 -103.36	-111.37 -106.54	-205.26 -109.58	-154.88 -109.58	-162.84 -109.58	-228.89 -109.58	-208.16 -109.58	-214.66 -109.58	-228.92 -109.58
	50	-76.04 -74.45	-80.05 -78.23	-153.11 -139.35	-106.14 -104.55	-111.37 -109.52	-205.26 -139.73	-154.88 -139.55	-162.84 -139.70	-228.89 -139.73	-208.16 -139.73	-214.66 -139.73	-228.92 -139.73
	60	-76.04 -74.45	-80.05 -78.23	-153.11 -150.05	-106.14 -104.55	-111.37 -109.53	-205.26 -169.89	-154.88 -153.23	-162.84 -160.42	-228.89 -169.89	-208.16 -169.89	-214.66 -169.89	-228.92 -169.89
	70	-76.04 -74.45	-80.05 -78.23	-153.11 -150.10	-106.14 -104.55	-111.37 -109.53	-205.26 -198.09	-154.88 -153.32	-162.84 -160.94	-228.89 -200.04	-208.16 -199.15	-214.66 -199.80	-228.92 -200.04
	80	-76.04 -74.45	-80.05 -78.23	-153.11 -150.10	-106.14 -104.55	-111.37 -109.53	-205.26 -202.48	-154.88 -153.33	-162.84 -160.94	-228.89 -226.48	-208.16 -206.41	-214.66 -212.35	-228.92 -226.50
	90	-76.04 -74.45	-80.05 -78.23	-153.11 -150.10	-106.14 -104.55	-111.37 -109.53	-205.26 -202.49	-154.88 -153.33	-162.84 -160.94	-228.89 -228.88	-208.16 -206.43	-214.66 -212.42	-228.92 -228.92
	100	-76.04 -74.45	-80.05 -78.23	-153.11 -150.10	-106.14 -104.55	-111.37 -109.53	-205.26 -202.49	-154.RR -153.33	-162.84 -160.94	-228.89 -228.89	-208.16 -206.43	-214.66 -212.42	-228.92 -228.92

TABLE 4.26: ψ_3, ψ_4 for Channel 2

		$d_1 = 10^{-7}$			$d_1 = 10^{-10}$			$d_1 = 10^{-15}$			$d_1 = 10^{-20}$		
	c =	0.5	0.7	1.0	0.5	0.7	1.0	0.5	0.7	1.0	0.5	0.7	1.0
No of	Iterations	71	53	36	91	64	39	124	84	40	157	103	44
No of taps in filter	20	-76.18 -44.09	-77.82 -44.07	-93.83 -44.08	-94.10 -44.08	-93.97 -44.08	-93.83 -44.08	-93.83 -44.08	-93.83 -44.08	-93.83 -44.08	-93.83 -44.08	-93.83 -44.08	-93.83 -44.08
	30	-76.56 -69.82	-77.67 -70.11	-183.12 -71.06	-106.67 -71.06	-109.01 -71.06	-194.44 -71.06	-156.13 -71.06	-161.32 -71.06	-214.01 -71.06	-203.93 -71.06	-218.46 -71.06	-214.02 -71.06
	40	-76.56 -75.77	-77.67 -77.12	-183.12 -98.02	-106.67 -97.37	-109.01 -97.65	-194.44 -98.02	-156.13 -98.02	-161.32 -98.02	-214.01 -98.02	-203.93 -98.02	-218.46 -98.02	-214.02 -98.02
	50	-76.56 -75.79	-77.67 -77.16	-183.12 -124.97	-106.67 -105.85	-109.01 -108.40	-194.44 -124.97	-156.13 -124.97	-161.32 -124.97	-214.01 -124.97	-203.93 -124.97	-218.46 -124.97	-214.02 -124.97
	60	-76.56 -75.79	-77.67 -77.16	-183.12 -151.92	-106.67 -105.91	-109.01 -108.49	-194.44 -151.93	-156.13 -150.26	-161.32 -151.40	-214.01 -151.93	-203.93 -151.93	-218.46 -151.93	-214.02 -151.93
	70	-76.56 -75.79	-77.67 -77.16	-183.12 -177.33	-106.67 -105.91	-109.01 -108.49	-194.44 -178.75	-156.13 -155.21	-161.32 -160.73	-213.01 -178.88	-203.93 -178.86	-218.46 -178.88	-214.02 -178.88
	80	-76.56 -75.79	-77.67 -77.16	-183.12 -182.55	-106.67 -105.91	-109.01 -108.49	-194.44 -193.79	-156.13 -155.23	-161.32 -160.80	-214.01 -205.22	-203.93 -201.27	-218.46 -205.47	-214.02 -205.22
	90	-76.56 -75.79	-77.67 -77.16	-183.12 -182.57	-106.67 -105.91	-109.01 -108.49	-194.44 -194.07	-156.13 -155.23	-161.32 -160.80	-214.01 -213.96	-203.93 -203.13	-218.46 -216.37	-214.02 -213.96
	100	-76.56 -75.79	-77.67 -77.16	-183.12 -182.57	-106.67 -105.91	-109.01 -108.49	-194.44 -194.07	-156.13 -155.23	-161.32 -160.80	-214.01 -214.01	-203.93 -203.12	-218.46 -216.47	-214.02 -214.02

TABLE 4.27: ψ_3, ψ_4 for Channel 3

		$d_1 = 10^{-7}$			$d_1 = 10^{-10}$			$d_1 = 10^{-15}$			$d_1 = 10^{-20}$		
	c =	0.5	0.7	1.0	0.5	0.7	1.0	0.5	0.7	1.0	0.5	0.7	1.0
No of	Iterations	139	116	107	180	140	112	247	177	117	312	214	122
No of taps in filter	20	-67.19 -22.21	-66.80 -22.20	-67.18 -22.21	-67.19 -22.21	-67.17 -22.21	-67.18 -22.21	-67.18 -22.21	-67.18 -22.21	-67.18 -22.21	-67.18 -22.21	-67.18 -22.21	-67.18 -22.21
	30	-78.92 -40.55	-80.92 -40.63	-149.18 -40.61	-107.59 -40.61	-112.25 -40.61	-207.15 -40.61	-155.35 -40.61	-163.52 -40.61	-207.90 -40.61	-199.37 -40.61	-209.46 -40.61	-207.90 -40.61
	40	-78.92 -39.31	-80.92 -39.29	-149.18 -39.28	-107.59 -39.28	-112.25 -39.28	-207.15 -39.28	-155.35 -39.28	-163.52 -39.28	-207.90 -39.28	-199.37 -39.28	-209.46 -39.28	-207.90 -39.28
	50	-78.92 -51.28	-80.92 -51.33	-149.18 -51.28	-107.59 -51.28	-112.25 -51.28	-207.15 -51.28	-155.35 -51.28	-163.52 -51.28	-207.90 -51.28	-199.37 -51.28	-209.46 -51.28	-207.90 -51.28
	60	-78.92 -54.63	-80.92 -54.63	-149.18 -54.60	-107.59 -54.60	-112.25 -54.60	-207.15 -54.60	-155.35 -54.60	-163.52 -54.60	-207.90 -54.60	-199.37 -54.60	-209.46 -54.60	-207.90 -54.60
	70	-78.92 -63.39	-80.92 -63.48	-149.18 -63.49	-107.59 -63.49	-112.25 -63.49	-207.15 -63.49	-155.35 -63.49	-163.52 -63.49	-207.90 -63.49	-199.37 -63.49	-209.46 -63.49	-207.90 -63.49
	80	-78.92 -68.77	-80.92 -69.02	-149.18 -69.27	-107.59 -69.27	-112.25 -69.28	-207.15 -69.27	-155.35 -69.27	-162.52 -69.27	-207.90 -69.27	-199.37 -69.27	-209.56 -69.27	-207.90 -69.27
	90	-79.82 -74.32	-80.92 -75.27	-149.18 -76.76	-107.59 -76.76	-112.25 -76.76	-207.15 -76.76	-155.35 -76.76	-163.52 -76.76	-207.90 -76.76	-199.37 -76.76	-209.46 -76.76	-207.90 -76.76
	100	-78.92 -76.85	-80.92 -78.71	-149.18 -83.46	-107.59 -83.45	-112.25 -83.46	-207.15 -83.46	-155.35 -83.46	-163.52 -83.46	-207.90 -83.46	-199.37 -83.46	-209.46 -83.46	-207.90 -83.46

TABLE 4.28: ψ_3, ψ_4 for Channel 4

4.6 System 4.6

The results so far have indicated that as the value of the estimate, λ_i, approaches to one of the required $\{\beta_h\}$ (the negative reciprocals of the roots (zeros) of Y(z) outside the unit circle), the rate of convergence and the improvement in the accuracy of the estimate becomes greater. This is also evident from equations 4.111 and 4.114. In other words, the closer the estimate to the exact value $\{\beta_h\}$, the faster the rate of convergence and when λ_i differs considerably from all $\{\beta_h\}$, the rate of convergence of the iterative process tends to be slow.

The algorithm is now modified to incorporate an 'averaging' process at the beginning of the iterative process for a few iterations with the aim of improving the initial rate of convergence. This idea has been used successfully in System 4.2 of the off-line system (Section 4.2) for finding the first of the four roots of Channel 2. A detailed description of the 'averaging' process can be found in Section 4.2.

As before, the receiver holds in store the sequence Y (equation 2.3) and a set of five possible starting-points (Figure 4.10) at the beginning of the iterative process. The tap gain of the feedback filter is set to the initial estimate of β_1, which is $\lambda_o = 0$. The sequence Y (equation 2.3) is then fed through the feedback filter, in reverse order, starting with the component y_g to give an output sequence $\dots e_{i,o}\ e_{i,1}\ \dots\ e_{i,g}$ (equation 4.106). From the sequence of $\{e_{i,h}\}$, h = 0,1,...,g, a new estimate of β_1 is given by (equation 4.114)

$$\lambda_{i+1} = \lambda_i + c\,\frac{e_{i,o}}{\varepsilon_i} \tag{4.145}$$

where
$$\varepsilon_i = e_{i,1} - e_{i,2}\lambda_i + e_{i,3}\lambda_i^2 - \dots + e_{i,g}(-\lambda_i)^{g-1} \tag{4.146}$$

and c is a real positive constant. However, for the first ℓ iterations, an extra computation step is performed as follows which gives a new estimate of β_1 (denoted by λ'_{i+1}) and is then taken to be the actual estimate λ_{i+1}.

$$\lambda'_{i+1} = (1 - \frac{1}{K_i})\lambda_i + \frac{1}{K_i}\lambda_{i+1} \tag{4.147}$$

where $i = 0,1,2,\ldots,\ell$ and $K_i = 2,3,\ldots,\ell+1$. To illustrate how this works in practice, at the start of the iterative process, $i = 0$, $\lambda_o = 0$ (or equal to any one of its five possible values) and from equations 4.145 and 4.147

$$\lambda_1 = \lambda_o + c\,\frac{e_{o,o}}{\varepsilon_o} \tag{4.148}$$

and

$$\lambda'_1 = \frac{1}{2}\lambda_o + \frac{1}{2}\lambda_1 \tag{4.149}$$

λ'_1 is then taken to be actual estimate λ_1. Again, using equations 4.145 and 4.147

$$\lambda_2 = \lambda_1 + c\,\frac{e_{1,o}}{\varepsilon_1} \tag{4.150}$$

and

$$\lambda'_2 = \frac{2}{3}\lambda_1 + \frac{1}{3}\lambda_2 \tag{4.151}$$

Substituting equation 4.149 (λ_1' is taken to be the estimate λ_1) into equation 4.151

$$\lambda_2' = \frac{1}{3}\lambda_o + \frac{1}{3}\lambda_1 + \frac{1}{3}\lambda_2 \qquad (4.152)$$

Similarly, from equations 4.145 and 4.147

$$\lambda_3 = \lambda_2 + c\,\frac{e_{2,o}}{\varepsilon_2} \qquad (4.153)$$

and

$$\lambda_3' = \frac{3}{4}\lambda_2 + \frac{1}{4}\lambda_3 \qquad (4.154)$$

Substituting equation 4.152 (λ_2' is taken to be the estimate λ_2) into equation 4.154

$$\begin{aligned}\lambda_3' &= \frac{3}{4}\left(\frac{1}{3}\lambda_o + \frac{1}{3}\lambda_1 + \frac{1}{3}\lambda_2\right) + \frac{1}{4}\lambda_3\\ &= \frac{1}{4}\lambda_o + \frac{1}{4}\lambda_1 + \frac{1}{4}\lambda_2 + \frac{1}{4}\lambda_3 \qquad (4.155)\end{aligned}$$

To summarise the above results

$$\lambda_1' = \frac{1}{2}(\lambda_o + \lambda_1)$$

$$\lambda_2' = \frac{1}{3}(\lambda_o + \lambda_1 + \lambda_2)$$

$$\lambda_3' = \frac{1}{4}(\lambda_0 + \lambda_1 + \lambda_2 + \lambda_3)$$

The iterative process is, therefore, exactly the same as System 4.4 using the algorithm given in equation 4.145 to give the new estimate of β_1, but for the first ℓ iteration, an extra step is carried to produce λ_1', λ_2', ..., λ_ℓ' which are taken to be the actual estimates λ_1, λ_2, ..., λ_ℓ . At the end of the 'averaging' period, when $i = \ell$, the extra computation step given in equation 4.147 is switched out and is no longer required until a restart or when i=1. During the 'averaging' period, the value $|e_{i,o}/\varepsilon_i|^2$ is checked for each iteration, and if

$$|e_{i,o}/\varepsilon_i|^2 < d_2 \qquad (4.156)$$

where d_2 is a small positive real constant, the algorithm is immediately replaced by that given in equation 4.145. The condition given in equation 4.156 signifies that the estimate λ_i is sufficiently close to one of the $\{\beta_h\}$ for the averaging process to be terminated. This condition depends on the number of iterations, ℓ, within the 'averaging' period, and the threshold value d_2. The diagram in Figure 4.23 shows clearly the operation of this new algorithm. All the other details such as the restart procedure, stopping criterion for convergence, evaluation of the resultant sampled impulse-response of the channel after each root has been found, and the adjustment of the adaptive filter are exactly the same as System 4.4 (Section 4.4).

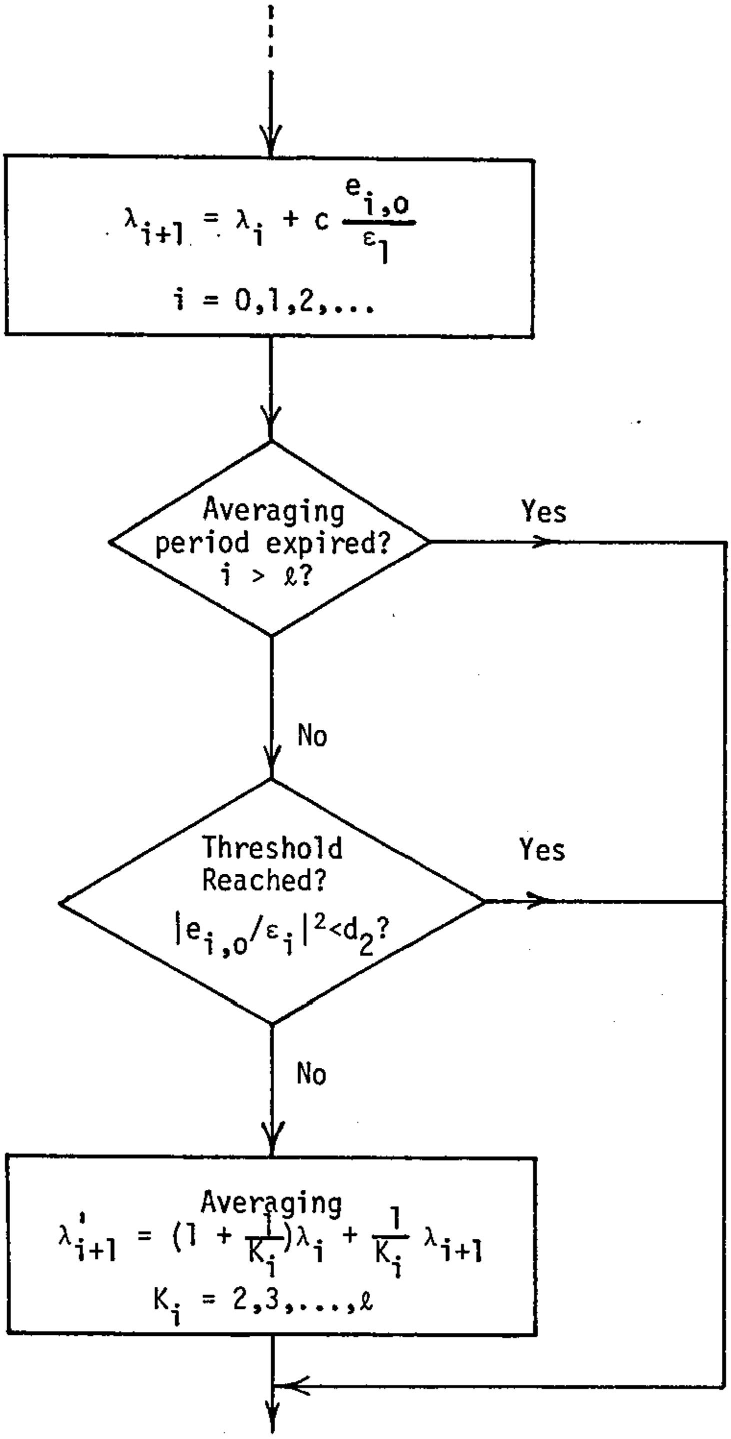

$\lambda_{i+1} = \lambda_i + c\,\frac{e_{i,o}}{\varepsilon_1}$
$i = 0,1,2,\ldots$
Averaging period expired? $i > \ell$?
Yes
No
Threshold Reached? $|e_{i,o}/\varepsilon_i|^2 < d_2$?
Yes
No
Averaging
$\lambda'_{i+1} = (1 + \frac{1}{K_i})\lambda_i + \frac{1}{K_i}\lambda_{i+1}$
$K_i = 2,3,\ldots,\ell$

FIGURE 4.23: Flow Diagram Showing the Operation of the Algorithm in System 4.6

Extensive tests were carried out by computer-simulation on the four Channels 1, 2, 3 and 4 with $c = 1.0$, $d_1 = 10^{-7}$, 10^{-10}, 10^{-15} and 10^{-20}. The number of taps used in the adaptive linear filter ranged from 20 to 100 in steps of 10. A number of averaging periods ($\ell = 3,4,5,6$) were used together with different threshold values ($d_2 = 0.05$, 0.075 and 0.1). The complete set of results is shown in Tables 4.29-4.44 which give the values ψ_1 and ψ_2 (equations 4.96-4.97) and the total number of iterations taken for the algorithm to find all the roots of Y(z) outside the unit circle for different combinations of d_1, d_2, ℓ and the number of filter taps.

The algorithm operates successfully, locating all the required roots, in the four cases tested. Again, very fast convergence is obtained, even for the severely distorted Channel 4. Furthermore, with c = 1.0, $d_1 = 10^{-10}$ and ℓ = 3,4,5 and 6, any of the roots of Y(z) outside the unit circle can be found in less than 20 iterations. Since the graphs of θ_i (equation 4.131) and ϕ_i'' (equation 4.135) versus i are similar to those of System 4.4 and convey very little extra information, and, therefore, not shown in this case. By far the most illuminating set of results is shown in Tables 4.45-4.46 which gives, for each of the values of the thresholds d_1 the total number of iterations taken by the algorithm to find all the required roots, and the corresponding parameters ψ_3 and ψ_4, for all the different combinations of ℓ and d_2 (as mentioned before) used in the 'averaging' period, when a 40-tap linear filter is assumed to be used. The results show that for a fixed 'averaging' period, ℓ, the fastest speed at which the algorithm finds all the required roots is obtained when d_2 = 0.1. Except in the case of Channel 4, where there are 8 roots outside the unit circle and 3 of them lying very close to it, an 'averaging' period of 3 gives the fastest speed for Channels 1-3. For Channel 4, ℓ = 4 is the best. The results suggest the best arrangement of this algorithm is probably that with $d_1 = 10^{-10}$, c = 1.0, d_2 = 0.1, ℓ = 4 and 40 taps for the linear filter.

Tables 4.49-4.52 give the number of iterations taken for this system (with d_2 = 0.1, c = 1.0, $d_1 = 10^{-7}$, 10^{-10}, 10^{-15} and 10^{-20}, and ℓ = 3, 4,5 and 6) and System 4.4 (with c = 1.0, $d_1 = 10^{-7}$, 10^{-10}, 10^{-15} and 10^{-20}) to find all the roots (zeros) of Y(z) that lie outside the unit circle for Channels 1-4. It shows that for Channels 1-3, System 4.4 takes less iterations to find the required roots whereas for Channel 4, this system is faster. It can be seen that in the case of Channel 3, a saving of 19 iterations is achieved. Detailed examination of the way in which the roots are tracked by the two algorithms has revealed that for moderate or mild channels (such as Channels 1 and 2) the roots are usually found by the first of the five possible starting-points but for the poorer channels (especially Channel 4), the roots are seldom found this way. Indeed, some of the roots of Channel 4 are only located by the fifth of the five possible starting-points in System 4.4. Such a costly process of going through a lot of starting-points when dealing with very poor channels before the root is found is avoided in this system.

		$d_1 = 10^{-7}$			$d_1 = 10^{-10}$			$d_1 = 10^{-15}$			$d_1 = 10^{-20}$		
d_2 =		0.050	0.075	0.100	0.050	0.075	0.100	0.050	0.075	0.100	0.050	0.075	0.100
No of Iterations		25	25	25	27	27	26	30	30	29	30	30	29
No of taps in filter	20	-258.26 -57.31	-258.26 -57.31	-257.11 -57.31	-259.18 -57.31	-259.18 -57.31	-257.43 -57.31	-260.47 -57.31	-260.47 -57.31	-258.32 -57.31	-260.47 -57.31	-260.47 -57.31	-258.32 -57.31
	30	-258.26 -88.02	-258.26 -88.02	-257.11 -88.02	-259.18 -88.02	-259.18 -88.02	-257.43 -88.02	-260.47 -88.02	-260.47 -88.02	-258.32 -88.02	-260.47 -88.02	-260.47 -88.02	-258.32 -88.02
	40	-258.26 -118.71	-258.26 -118.71	-257.11 -118.71	-259.18 -118.73	-259.18 -118.73	-257.43 -118.73	-260.47 -118.73	-260.47 -118.73	-258.32 -118.73	-260.47 -118.73	-260.47 -118.73	-258.32 -118.73
	50	-258.26 -141.50	-258.26 -141.50	-257.11 -142.79	-259.18 -149.43	-259.18 -149.43	-257.43 -149.43	-260.47 -149.43	-260.47 -149.43	-258.32 -149.43	-260.47 -149.43	-260.47 -149.43	-258.32 -149.43
	60	-258.26 -142.27	-258.26 -142.27	-257.11 -143.85	-259.18 -180.14	-259.18 -180.14	-257.43 -180.14	-260.47 -180.14	-260.47 -180.14	-258.32 -180.14	-260.47 -180.14	-260.47 -180.14	-258.32 -180.14
	70	-258.26 -142.27	-258.26 -142.27	-257.11 -143.85	-259.18 -210.70	-259.18 -210.70	-257.43 -210.02	-260.47 -210.84	-260.47 -210.84	-258.32 -210.84	-260.47 -210.84	-260.47 -210.84	-258.32 -210.84
	80	-258.26 -142.27	-258.26 -142.27	-257.11 -143.85	-259.18 -225.52	-259.18 -217.52	-257.43 -241.45	-260.47 -241.49	-260.47 -241.45	-258.32 -241.45	-260.47 -241.49	-260.47 -241.49	-258.32 -241.45
	90	-258.28 -142.27	-258.26 -142.27	-257.11 -143.85	-259.18 -225.63	-259.19 -225.63	-257.43 -217.67	-260.47 -259.78	-260.47 -259.78	-258.32 -257.81	-260.47 -259.78	-260.47 -259.78	-258.32 -257.81
	100	-258.26 -142.27	-258.26 -142.27	-257.11 -143.85	-259.18 -225.63	-259.18 -225.63	-257.43 -217.67	-260.47 -260.04	-260.47 -260.04	-258.32 -257.96	-260.47 -260.04	-260.47 -260.04	-258.32 -257.96

TABLE 4.29: ψ_1, ψ_2 for Channel 1, ε = 3

		$d_1 = 10^{-7}$			$d_1 = 10^{-10}$			$d_1 = 10^{-15}$			$d_1 = 10^{-20}$		
d_2 =		0.050	0.075	0.100	0.050	0.075	0.100	0.050	0.075	0.100	0.050	0.075	0.100
No of Iterations		36	37	37	37	37	37	41	41	41	41	42	42
No of taps in filter	20	-144.49 -49.25	-144.49 -49.25	-144.49 -49.25	-144.49 -49.25	-144.49 -49.26	-144.49 -49.25	-144.49 -49.25	-144.49 -49.25	-144.49 -49.25	-144.49 -49.25	-144.49 -49.25	-144.49 -49.25
	30	-144.49 -79.42	-144.49 -79.42	-144.49 -79.42	-144.49 -79.42	-144.49 -79.42	-144.49 -79.42	-144.49 -79.42	-144.49 -79.42	-144.49 -79.42	-144.49 -79.42	-144.49 -79.42	-144.49 -79.42
	40	-144.49 -109.57	-144.49 -109.57	-144.49 -109.57	-144.49 -109.57	-144.49 -109.57	-144.49 -109.57	-144.49 -109.57	-144.49 -109.57	-144.49 -109.57	-144.49 -109.57	-144.49 -109.57	-144.49 -109.57
	50	-144.49 -137.64	-144.49 -138.48	-144.49 -138.48	-144.49 -138.48	-144.49 -138.48	-144.49 -138.48	-144.49 -138.48	-144.49 -138.48	-144.49 -138.48	-144.49 -138.48	-144.49 -138.48	-144.49 -138.48
	60	-144.49 -141.80	-144.49 -144.48	-144.49 -144.48	-144.49 -144.48	-144.49 -144.48	-144.49 -144.48	-144.49 -144.48	-144.49 -144.48	-144.49 -144.48	-144.49 -144.48	-144.49 -144.48	-144.49 -144.48
	70	-144.49 -141.80	-144.49 -144.49	-144.49 -144.49	-144.49 -144.49	-144.49 -144.49	-144.49 -144.49	-144.49 -144.49	-144.49 -144.49	-144.49 -144.49	-144.49 -144.49	-144.49 -144.49	-144.49 -144.49
	80	-144.49 -141.80	-144.49 -144.49	-144.49 -144.49	-144.49 -144.49	-144.49 -144.49	-144.49 -144.49	-144.49 -144.49	-144.49 -144.49	-144.49 -144.49	-144.49 -144.49	-144.49 -144.49	-144.49 -144.49
	90	-144.49 -141.80	-144.49 -144.49	-144.49 -144.49	-144.49 -144.49	-144.49 -144.49	-144.49 -144.49	-144.49 -144.49	-144.49 -144.49	-144.49 -144.49	-144.49 -144.49	-144.49 -144.49	-144.49 -144.49
	100	-144.49 -141.80	-144.49 -144.49	-144.49 -144.49	-144.49 -144.49	-144.49 -144.49	-144.49 -144.49	-144.49 -144.49	-144.49 -144.49	-144.49 -144.49	-144.49 -144.49	-144.49 -144.49	-144.49 -144.49

TABLE 4.30: ψ_1, ψ_2 for Channel 2, ε = 3

		$d_1 = 10^{-7}$			$d_1 = 10^{-10}$			$d_1 = 10^{-15}$			$d_1 = 10^{-20}$		
d_2 =		0.050	0.075	0.100	0.050	0.075	0.100	0.050	0.075	0.100	0.050	0.075	0.100
No of Iterations		49	48	48	52	51	51	54	53	53	56	56	56
No of taps in filter	20	-93.83 -44.08	-93.83 -44.08	-93.83 -44.08	-93.83 -44.08	-93.83 -44.08	-93.83 -44.08	-93.83 -44.08	-93.83 -44.08	-93.83 -44.08	-93.83 -44.08	-93.83 -44.08	-93.83 -44.08
	30	-258.30 -71.06	-256.23 -71.06	-256.23 -71.06	-258.35 -71.06	-258.32 -71.06	-258.32 -71.06	-259.04 -71.06	-257.39 -71.06	-257.39 -71.06	-257.38 -71.06	-259.37 -71.06	-259.37 -71.06
	40	-258.30 -98.02	-256.23 -98.02	-256.23 -98.02	-258.36 -98.02	-258.32 -98.02	-258.32 -98.02	-259.04 -98.02	-257.39 -98.02	-257.39 -98.02	-257.38 -98.02	-259.37 -98.02	-259.37 -98.02
	50	-258.30 -124.97	-256.23 -124.97	-256.23 -124.97	-258.36 -124.97	-258.32 -124.97	-258.32 -124.97	-259.04 -124.97	-257.39 -124.97	-257.39 -124.97	-257.38 -124.97	-259.37 -124.97	-259.37 -124.97
	60	-258.30 -150.54	-256.23 -150.35	-256.23 -150.35	-258.36 -151.93	-258.32 -151.93	-258.32 -151.93	-259.04 -151.93	-257.39 -151.93	-257.39 -151.93	-257.38 -151.93	-259.37 -151.93	-259.37 -151.93
	70	-258.30 -156.15	-256.23 -155.50	-256.23 -155.50	-258.35 -178.88	-258.32 -178.87	-258.32 -178.87	-259.04 -178.88	-257.39 -178.88	-257.39 -178.88	-257.38 -178.88	-259.37 -178.88	-259.37 -178.88
	80	-258.30 -156.18	-256.23 -155.52	-256.23 -155.52	-258.36 -205.78	-258.32 -202.33	-258.32 -202.33	-259.04 -205.83	-257.39 -205.83	-257.39 -205.83	-257.38 -205.83	-259.37 -205.83	-259.37 -205.83
	90	-258.30 -156.18	-256.23 -155.52	-256.23 -155.52	-258.36 -224.26	-258.32 -204.90	-258.32 -204.90	-259.04 -232.77	-257.39 -232.77	-257.39 -232.77	-257.38 -232.77	-259.37 -232.77	-259.37 -232.77
	100	-258.30 -156.18	-256.23 -155.52	-256.23 -155.52	-258.36 -224.92	-258.32 -204.90	-258.32 -204.90	-259.04 -255.85	-257.39 -254.94	-257.39 -254.94	-257.38 -254.94	-259.37 -255.96	-259.37 -255.96

TABLE 4.31: ψ_1, ψ_2 for Channel 3, ε = 3

		$d_1 = 10^{-7}$			$d_1 = 10^{-10}$			$d_1 = 10^{-15}$			$d_1 = 10^{-20}$		
d_2 =		0.050	0.075	0.100	0.050	0.075	0.100	0.050	0.075	0.100	0.050	0.075	0.100
No of Iterations		118	118	117	121	121	121	127	127	126	130	130	130
No of taps in filter	20	-67.18 -22.21	-67.18 -22.21	-67.18 -22.21	-67.18 -22.21	-67.18 -22.21	-67.18 -22.21	-67.18 -22.21	-67.18 -22.21	-67.18 -22.21	-67.18 -22.21	-67.18 -22.21	-67.18 -22.21
	30	-257.33 -40.61	-257.33 -40.61	-257.71 -40.61	-256.13 -40.61	-256.13 -40.61	-254.40 -40.61	-255.82 -40.61	-255.82 -40.61	-254.98 -40.61	-254.74 -40.61	-254.74 -40.61	-254.43 -40.61
	40	-257.33 -39.28	-257.33 -39.28	-257.71 -39.28	-256.13 -39.28	-256.13 -39.28	-254.40 -39.28	-255.82 -39.28	-255.82 -39.28	-254.98 -39.28	-254.74 -39.28	-254.74 -39.28	-254.43 -39.28
	50	-257.33 -51.28	-257.33 -51.28	-257.71 -51.28	-256.13 -51.28	-256.13 -51.28	-254.40 -51.28	-255.82 -51.28	-255.82 -51.28	-254.98 -51.28	-254.74 -51.28	-254.74 -51.28	-254.43 -51.28
	60	-257.33 -54.60	-257.33 -54.60	-257.71 -54.60	-256.13 -54.60	-256.13 -54.60	-254.40 -54.60	-255.82 -54.60	-255.82 -54.60	-254.98 -54.60	-254.74 -54.60	-254.74 -54.60	-254.43 -54.60
	70	-257.33 -63.49	-257.33 -63.49	-257.71 -63.49	-256.13 -63.39	-256.13 -63.49	-254.40 -63.49	-255.82 -63.49	-255.82 -63.49	-254.98 -63.49	-254.74 -63.49	-254.74 -63.49	-254.43 -63.49
	80	-257.33 -69.27	-257.33 -69.27	-257.71 -69.27	-256.13 -69.27	-256.13 -69.27	-254.40 -69.27	-255.82 -69.27	-255.82 -69.27	-254.98 -69.27	-254.74 -69.27	-254.74 -69.27	-254.43 -69.27
	90	-257.33 -76.76	-257.33 -76.76	-257.71 -76.76	-256.13 -76.76	-256.13 -76.76	-254.40 -76.76	-255.82 -76.76	-255.82 -76.76	-254.98 -76.76	-254.74 -76.76	-254.74 -76.76	-254.43 -76.76
	100	-257.33 -83.46	-257.33 -83.46	-257.71 -83.46	-256.13 -83.46	-256.13 -83.46	-254.40 -83.46	-255.82 -83.46	-255.82 -83.46	-254.98 -83.46	-254.74 -83.46	-254.74 -83.46	-254.43 -83.46

TABLE 4.32: ψ_1, ψ_2 for Channel 4, ε = 3

		$d_1 = 10^{-7}$			$d_1 = 10^{-10}$			$d_1 = 10^{-15}$			$d_1 = 10^{-20}$		
d_2 = No of Iterations		0.050 25	0.075 25	0.100 25	0.050 27	0.075 27	0.100 26	0.050 30	0.075 30	0.100 29	0.050 30	0.075 30	0.100 29
No of taps in filter	20	-258.26 -57.31	-258.26 -57.31	-257.11 -57.31	-259.18 -57.31	-259.18 -57.31	-257.43 -57.31	-260.47 -57.31	-260.47 -57.31	-258.32 -57.31	-260.47 -57.31	-260.47 -57.31	-258.32 -57.31
	30	-258.26 -88.02	-258.26 -88.02	-257.11 -88.02	-259.18 -88.02	-259.18 -88.02	-257.43 -88.02	-260.47 -88.02	-260.47 -88.02	-258.32 -88.02	-260.47 -88.02	-260.47 -88.02	-258.32 -88.02
	40	-258.26 -118.71	-258.26 -118.71	-257.11 -118.71	-259.18 -118.73	-259.18 -118.73	-257.43 -118.73	-260.47 -118.73	-260.47 -118.73	-258.32 -118.73	-260.47 -118.73	-260.47 -118.73	-258.32 -118.73
	50	-258.26 -141.50	-258.26 -141.50	-257.11 -142.79	-259.18 -149.43	-259.18 -149.43	-257.43 -149.43	-260.47 -149.43	-260.47 -149.43	-258.32 -149.43	-260.47 -149.43	-260.47 -149.43	-258.32 -149.43
	60	-258.26 -142.27	-258.26 -142.27	-257.11 -143.85	-259.18 -180.14	-259.18 -180.14	-257.43 -180.14	-260.47 -180.14	-260.47 -180.14	-258.32 -180.14	-260.47 -180.14	-260.47 -180.14	-258.32 -180.14
	70	-258.26 -142.27	-258.26 -142.27	-257.11 -143.85	-259.18 -210.70	-259.18 -210.70	-257.43 -210.02	-260.47 -210.84	-260.47 -210.84	-258.32 -210.84	-260.47 -210.84	-260.47 -210.84	-258.32 -210.84
	80	-258.26 -142.27	-258.26 -142.27	-257.11 -143.85	-259.18 -225.52	-259.19 -225.52	-257.43 -217.65	-260.47 -241.49	-260.47 -241.49	-258.32 -241.45	-260.47 -241.49	-260.47 -241.49	-258.32 -241.45
	90	-258.26 -142.27	-258.26 -142.27	-257.11 -143.85	-259.18 -225.63	-259.18 -225.63	-257.43 -217.67	-260.47 -249.76	-260.47 -259.78	-258.32 -257.81	-260.47 -259.78	-260.47 -259.78	-258.32 -257.81
	100	-258.26 -142.27	-258.26 -142.27	-257.11 -143.85	-259.18 -225.63	-259.18 -225.63	-257.43 -217.67	-250.47 -260.04	-260.47 -260.04	-258.32 -257.96	-260.47 -260.04	-260.47 -260.04	-258.32 -257.96

TABLE 4.33: ψ_1, ψ_2 for Channel 1, ε = 4

		$d_1 = 10^{-7}$			$d_1 = 10^{-10}$			$d_1 = 10^{-15}$			$d_1 = 10^{-20}$		
d_2 = No of Iterations		0.050 36	0.075 37	0.100 37	0.050 37	0.075 37	0.100 37	0.050 41	0.075 41	0.100 41	0.050 41	0.075 42	0.100 42
No of taps in filter	20	-144.49 -49.25	-144.49 -49.25	-144.49 -49.25	-144.49 -49.25	-144.49 -49.25	-144.49 -49.25	-144.49 -49.25	-144.49 -49.25	-144.49 -49.25	-144.49 -49.25	-144.49 -49.25	-144.49 -49.25
	30	-144.49 -79.42	-144.49 -79.42	-144.49 -79.42	-144.49 -79.42	-144.49 -79.42	-144.49 -79.42	-144.49 -79.42	-144.49 -79.42	-144.49 -79.42	-144.49 -79.42	-144.49 -79.42	-144.49 -79.42
	40	-144.49 -109.57	-144.49 -109.57	-144.49 -109.57	-144.49 -109.57	-144.49 -109.57	-144.49 -109.57	-144.49 -109.57	-144.49 -109.57	-144.49 -109.57	-144.49 -109.57	-144.49 -109.57	-144.49 -109.57
	50	-144.49 -137.64	-144.49 -138.48	-144.49 -138.48	-144.49 -138.48	-144.49 -138.48	-144.49 -138.48	-144.49 -138.48	-144.49 -138.48	-144.49 -138.48	-144.49 -138.48	-144.49 -138.48	-144.49 -138.45
	60	-144.49 -141.80	-144.49 -144.48	-144.49 -144.48	-144.49 -144.48	-144.49 -144.48	-144.49 -144.48	-144.49 -144.48	-144.49 -144.48	-144.49 -144.48	-144.49 -144.48	-144.49 -144.48	-144.49 -144.48
	70	-144.49 -141.80	-144.49 -144.49	-144.49 -144.49	-144.49 -144.49	-144.49 -144.49	-144.49 -144.49	-144.49 -144.49	-144.49 -144.49	-144.49 -144.49	-144.49 -144.49	-144.49 -144.49	-144.49 -144.49
	80	-144.49 -141.80	-144.49 -144.49	-144.49 -144.49	-144.49 -144.49	-144.49 -144.49	-144.49 -144.49	-144.49 -144.49	-144.49 -144.49	-144.49 -144.49	-144.49 -144.49	-144.49 -144.49	-144.49 -144.49
	90	-144.49 -141.80	-144.49 -144.49	-144.49 -144.49	-144.49 -144.49	-144.49 -144.49	-144.49 -144.49	-144.49 -144.49	-144.49 -144.49	-144.49 -144.49	-144.49 -144.49	-144.49 -144.49	-144.49 -144.49
	100	-144.49 -141.80	-144.49 -144.49	-144.49 -144.49	-144.49 -144.49	-144.49 -144.49	-144.49 -144.49	-144.49 -144.49	-144.49 -144.49	-144.49 -144.49	-144.49 -144.49	-144.49 -144.49	-144.49 -144.49

TABLE 4.34: ψ_1, ψ_2 for Channel 2, ε = 4

		$d_1 = 10^{-7}$			$d_1 = 10^{-10}$			$d_1 = 10^{-15}$			$d_1 = 10^{-20}$		
d_2 = No of Iterations		0.050 51	0.075 50	0.100 50	0.050 54	0.075 53	0.100 53	0.050 57	0.075 55	0.100 55	0.050 58	0.075 57	0.100 57
No of taps in filter	20	-93.83 -44.08	-93.83 -44.08	-93.83 -44.08	-93.83 -44.08	-93.83 -44.08	-93.83 -44.08	-93.83 -44.08	-93.83 -44.08	-93.83 -44.08	-93.83 -44.08	-93.83 -44.08	-93.83 -44.08
	30	-257.78 -71.06	-258.38 -71.06	-258.38 -71.06	-257.93 -71.06	-258.33 -71.06	-258.33 -71.06	-259.06 -71.06	-258.73 -71.06	-258.73 -71.06	-257.43 -71.06	-257.87 -71.06	-257.87 -71.06
	40	-257.78 -98.02	-258.38 -98.02	-258.38 -98.02	-257.93 -98.02	-258.33 -98.02	-258.33 -98.02	-259.06 -98.02	-258.73 -98.02	-258.73 -98.02	-257.43 -98.02	-257.87 -98.02	-257.87 -98.02
	50	-257.78 -124.96	-258.38 -124.96	-258.38 -124.96	-257.93 -124.97	-258.33 -124.97	-258.33 -124.97	-259.06 -124.97	-258.73 -124.97	-258.73 -124.97	-257.43 -124.97	-257.87 -124.97	-257.87 -124.97
	60	-257.78 -148.00	-258.48 -148.21	-258.38 -148.21	-257.93 -151.93	-258.33 -151.93	-258.33 -151.93	-259.06 -151.93	-258.73 -151.93	-258.73 -151.93	-257.43 -151.93	-257.87 -151.93	-257.87 -151.93
	70	-257.78 -150.24	-258.38 -150.61	-258.38 -150.61	-257.93 -178.88	-258.33 -178.88	-258.33 -178.88	-259.06 -178.88	-258.73 -178.88	-258.73 -178.88	-257.43 -178.88	-257.87 -178.88	-257.87 -178.88
	80	-257.78 -150.25	-258.38 -150.61	-258.38 -150.61	-257.93 -205.78	-258.33 -205.83	-258.33 -205.83	-259.06 -205.83	-258.73 -205.83	-258.73 -205.83	-257.43 -205.83	-257.87 -205.83	-257.87 -205.83
	90	-257.78 -150.25	-258.38 -150.61	-258.38 -150.61	-257.93 -224.25	-258.33 -232.68	-258.33 -232.58	-259.06 -232.77	-258.73 -232.77	-258.73 -232.77	-257.43 -232.77	-257.87 -232.77	-257.87 -232.77
	100	-257.78 -150.25	-258.38 -150.61	-258.38 -150.61	-257.93 -224.91	-258.33 -248.41	-258.33 -248.41	-259.06 -255.86	-258.73 -255.71	-258.73 -255.71	-257.43 -255.10	-257.87 -255.32	-257.87 -255.32

TABLE 4.35: ψ_1, ψ_2 for Channel 3, ε = 4

		$d_1 = 10^{-7}$			$d_1 = 10^{-10}$			$d_1 = 10^{-15}$			$d_1 = 10^{-20}$		
d_2 = No of Iterations		0.050 94	0.075 92	0.100 90	0.050 98	0.075 97	0.100 96	0.050 105	0.075 102	0.100 100	0.050 107	0.075 105	0.100 104
No of taps in filter	20	-67.18 -22.21	-67.18 -22.21	-67.18 -22.21	-67.18 -22.21	-67.18 -22.21	-67.18 -22.21	-67.18 -22.21	-67.18 -22.21	-67.18 -22.21	-67.18 -22.21	-67.18 -22.21	-67.18 -22.21
	30	-256.69 -40.61	-255.52 -40.61	-255.14 -40.61	-257.17 -40.61	-256.93 -40.61	-257.37 -40.61	-255.36 -40.61	-256.18 -40.61	-256.13 -40.61	-255.53 -40.61	-254.91 -40.61	-256.09 -40.61
	40	-256.69 -39.28	-255.52 -39.28	-255.14 -39.28	-257.17 -39.28	-256.93 -39.28	-257.37 -39.28	-255.36 -39.28	-256.18 -39.28	-256.13 -39.28	-255.53 -39.28	-254.91 -39.28	-256.09 -39.28
	50	-256.69 -51.28	-255.52 -51.28	-255.14 -51.28	-257.17 -51.28	-256.93 -51.28	-257.37 -51.28	-255.36 -51.28	-256.18 -51.28	-256.13 -51.28	-255.53 -51.28	-254.91 -51.28	-256.09 -51.28
	60	-256.69 -54.60	-255.52 -54.60	-255.14 -54.60	-257.17 -54.60	-256.93 -54.60	-257.37 -54.60	-255.36 -54.60	-256.18 -54.60	-256.13 -54.60	-255.53 -54.60	-254.91 -54.60	-256.09 -54.60
	70	-256.69 -63.49	-255.52 -63.49	-255.14 -63.49	-257.17 -63.49	-256.93 -63.49	-257.37 -63.49	-255.36 -63.49	-256.18 -63.49	-256.13 -63.49	-255.53 -63.49	-254.91 -63.49	-256.09 -63.49
	80	-256.69 -69.27	-255.52 -69.27	-255.14 -69.27	-257.17 -69.27	-256.93 -69.27	-257.37 -69.27	-255.36 -69.27	-256.18 -69.27	-256.13 -69.27	-255.53 -69.27	-254.91 -69.27	-256.09 -69.27
	90	-256.69 -76.76	-255.52 -76.76	-255.14 -76.76	-257.17 -76.76	-256.93 -76.76	-257.37 -76.76	-255.36 -76.76	-256.18 -76.76	-256.13 -76.76	-255.53 -76.76	-254.91 -76.76	-256.09 -76.76
	100	-256.69 -83.46	-255.52 -83.46	-255.14 -83.46	-257.17 -83.46	-256.93 -83.46	-257.37 -83.46	-255.36 -83.46	-256.18 -83.46	-256.13 -83.46	-255.53 -83.46	-254.91 -83.46	-256.09 -83.46

TABLE 4.36: ψ_1, ψ_2 for Channel 4, ε = 4

		$d_1 = 10^{-7}$			$d_1 = 10^{-10}$			$d_1 = 10^{-15}$			$d_1 = 10^{-20}$		
d_2 = No of Iterations		0.050 25	0.075 25	0.100 25	0.050 27	0.075 27	0.100 26	0.050 30	0.075 30	0.100 29	0.050 30	0.075 30	0.100 29
No of taps in filter	20	-258.25 -57.31	-258.26 -57.31	-257.11 -57.31	-259.18 -57.31	-259.18 -57.31	-257.43 -57.31	-260.47 -57.31	-260.47 -57.31	-258.32 -57.31	-260.47 -57.31	-260.47 -57.31	-258.32 -57.31
	30	-258.26 -88.02	-258.26 -88.02	-257.11 -88.02	-259.18 -88.02	-259.18 -88.02	-257.43 -88.02	-260.47 -88.02	-260.47 -88.02	-258.32 -88.02	-260.47 -88.02	-260.47 -88.02	-258.32 -88.02
	40	-258.26 -118.71	-258.26 -118.71	-257.11 -118.71	-259.18 -118.73	-259.18 -118.73	-257.43 -118.73	-260.47 -118.73	-260.47 -118.73	-258.32 -118.73	-260.47 -118.73	-260.47 -118.73	-258.32 -118.73
	50	-258.26 -141.50	-256.26 -141.50	-257.11 -142.79	-259.18 -149.43	-259.18 -149.43	-257.43 -149.43	-260.47 -149.43	-260.47 -149.43	-258.32 -149.43	-260.47 -149.43	-260.47 -149.43	-258.32 -149.43
	60	-258.26 -142.27	-258.26 -142.27	-257.11 -143.85	-259.18 -180.14	-259.18 -180.14	-257.43 -180.14	-260.47 -180.14	-260.47 -180.14	-258.32 -180.14	-260.47 -180.14	-260.47 -180.14	-258.32 -180.14
	70	-258.26 -142.27	-258.26 -142.27	-257.11 -143.85	-259.18 -210.70	-259.18 -210.70	-257.43 -210.02	-260.47 -210.84	-260.47 -210.84	-258.32 -210.84	-260.47 -210.84	-260.47 -210.84	-258.32 -210.84
	80	-258.26 -142.27	-258.26 -142.27	-257.11 -143.85	-259.18 -225.52	-259.18 -225.52	-257.43 -217.65	-260.47 -241.49	-260.47 -241.49	-258.32 -241.45	-260.47 -241.49	-260.47 -241.49	-258.32 -241.45
	90	-258.25 -142.27	-258.26 -142.27	-257.11 -143.85	-259.18 -225.63	-259.18 -225.63	-257.43 -217.67	-260.47 -259.78	-260.47 -259.78	-258.32 -257.81	-260.47 -259.78	-260.47 -259.78	-258.32 -257.81
	100	-258.26 -142.27	-258.26 -142.27	-257.11 -143.85	-259.18 -225.63	-259.18 -225.63	-257.43 -217.67	-260.47 -260.04	-260.47 -260.04	-258.32 -257.96	-260.47 -260.04	-260.47 -260.04	-258.32 -257.96

TABLE 4.37: ψ_1, ψ_2 for Channel 1, ε = 5

		$d_1 = 10^{-7}$			$d_1 = 10^{-10}$			$d_1 = 10^{-15}$			$d_1 = 10^{-20}$		
d_2 = No of Iterations		0.050	0.075	0.100	0.050	0.075	0.100	0.050	0.075	0.100	0.050	0.075	0.100
No of taps in filter	20	-262.59 -49.25	-264.37 -49.25	-264.37 -49.25	-263.48 -49.25	-263.15 -49.25	-263.15 -49.25	-264.27 -49.25	-264.71 -49.25	-264.71 -49.25	-263.95 -49.25	-264.87 -49.25	-264.87 -49.25
	30	-262.59 -79.42	-264.37 -79.42	-264.37 -79.42	-263.48 -79.42	-263.15 -79.42	-263.15 -79.42	-264.27 -79.42	-264.71 -79.42	-264.71 -79.42	-263.95 -79.42	-264.87 -79.42	-264.87 -79.42
	40	-262.59 -109.57	-264.37 -109.58	-264.37 -109.58	-263.48 -109.58	-263.15 -109.59	-263.15 -109.58	-264.27 -109.58	-264.71 -109.58	-264.71 -109.58	-263.95 -109.58	-264.87 -109.58	-264.87 -109.58
	50	-262.59 -138.64	-264.37 -139.73	-264.37 -139.74	-263.48 -139.73	-263.15 -139.73	-263.15 -139.73	-264.27 -139.73	-264.71 -139.73	-264.71 -139.73	-263.95 -139.73	-264.87 -139.73	-264.87 -139.73
	60	-262.59 -145.15	-264.37 -166.02	-264.37 -166.02	-263.48 -169.89	-263.15 -169.89	-263.15 -169.89	-264.27 -169.89	-264.71 -169.89	-264.71 -169.89	-263.95 -169.89	-264.87 -169.89	-264.87 -169.89
	70	-262.59 -145.16	-264.37 -168.31	-264.37 -168.31	-263.48 -200.05	-263.15 -200.04	-263.15 -200.04	-264.27 -200.05	-264.71 -200.05	-264.71 -200.05	-263.95 -200.05	-264.87 -200.05	-264.87 -200.05
	80	-262.59 -145.16	-264.37 -168.31	-264.37 -168.31	-263.48 -230.19	-263.15 -227.70	-263.15 -227.70	-264.27 -230.20	-264.71 -230.20	-264.71 -230.20	-263.95 -230.20	-264.87 -230.20	-264.87 -230.20
	90	-262.59 -145.16	-264.37 -168.31	-264.37 -168.31	-263.48 -253.90	-263.15 -231.28	-263.15 -231.28	-264.27 -258.15	-264.71 -258.38	-264.71 -258.38	-263.95 -258.21	-264.87 -258.51	-264.87 -258.51
	100	-262.59 -145.16	-264.37 -168.31	-264.37 -168.31	-263.48 -255.01	-263.15 -231.29	-263.15 -231.29	-264.27 -262.13	-264.71 -262.74	-264.71 -262.74	-263.95 -262.29	-264.87 -263.11	-264.87 -263.11

TABLE 4.38: ψ_1, ψ_2 for Channel 2, ε = 5

		$d_1 = 10^{-7}$			$d_1 = 10^{-10}$			$d_1 = 10^{-15}$			$d_1 = 10^{-20}$		
d_2 = No of Iterations		0.050 60	0.075 55	0.100 54	0.050 61	0.075 57	0.100 56	0.050 64	0.075 59	0.100 58	0.050 65	0.075 61	0.100 60
No of taps in filter	20	-93.83 -44.08	-93.83 -44.08	-93.83 -44.08	-93.83 -44.08	-93.83 -44.08	-93.83 -44.08	-93.83 -44.08	-93.83 -44.08	-93.83 -44.08	-93.83 -44.08	-93.83 -44.08	-93.83 -44.08
	30	-258.47 -71.06	-259.45 -71.06	-259.45 -71.06	-257.95 -71.06	-257.70 -71.06	-257.70 -71.06	-257.45 -71.06	-257.48 -71.06	-257.48 -71.06	-257.71 -71.06	-257.45 -71.06	-257.45 -71.06
	40	-258.47 -98.02	-259.45 -98.02	-259.45 -98.02	-257.95 -98.02	-257.70 -98.02	-257.70 -98.02	-257.45 -98.02	-257.48 -98.02	-257.48 -98.02	-257.71 -98.02	-257.45 -98.02	-257.45 -98.02
	50	-258.47 -124.97	-259.45 -124.97	-259.45 -124.97	-257.95 -124.97	-257.70 -124.97	-257.70 -124.97	-257.45 -124.97	-257.48 -124.97	-257.48 -124.97	-257.71 -124.97	-257.45 -124.97	-257.45 -124.97
	60	-258.47 -151.93	-259.45 -151.56	-259.45 -151.45	-257.95 -151.93	-257.70 -151.93	-257.70 -151.93	-257.45 -151.93	-257.48 -151.93	-257.48 -151.93	-257.71 -151.93	-257.45 -151.93	-257.45 -151.93
	70	-258.47 -178.76	-259.45 -162.33	-259.45 -162.33	-257.95 -178.88	-257.70 -178.88	-257.70 -178.88	-257.45 -178.88	-257.48 -178.88	-257.48 -178.88	-257.71 -178.88	-257.45 -178.88	-257.45 -178.88
	80	-258.47 -194.16	-259.45 -162.43	-259.45 -162.43	-257.95 -205.81	-257.70 -205.83	-257.70 -205.83	-257.45 -205.83	-257.48 -205.83	-257.48 -205.83	-257.71 -205.83	-257.45 -205.83	-257.45 -205.83
	90	-258.47 -194.47	-259.45 -162.43	-259.45 -162.43	-257.95 -227.26	-257.70 -232.63	-257.70 -232.63	-257.45 -232.77	-257.48 -232.77	-257.48 -232.77	-257.71 -232.77	-257.45 -232.77	-257.45 -232.77
	100	-258.47 -194.47	-259.45 -162.43	-259.45 -162.43	-257.95 -228.69	-257.70 -247.00	-257.70 -247.00	-257.45 -254.99	-257.48 -255.08	-257.48 -255.08	-257.71 -255.12	-257.45 -255.04	-257.45 -255.04

TABLE 4.39: ψ_1, ψ_2 for Channel 3, ε = 5

		$d_1 = 10^{-7}$			$d_1 = 10^{-10}$			$d_1 = 10^{-15}$			$d_1 = 10^{-20}$		
d_2 = No of Iterations		0.050 103	0.075 102	0.100 98	0.050 107	0.075 107	0.100 104	0.050 114	0.075 112	0.100 108	0.050 116	0.075 115	0.100 112
No of taps in filter	20	-67.18 -22.21	-67.18 -22.21	-67.18 -22.21	-67.18 -22.21	-67.18 -22.21	-67.18 -22.21	-67.18 -22.21	-67.18 -22.21	-67.18 -22.21	-67.18 -22.21	-67.18 -22.21	-67.18 -22.21
	30	-254.67 -40.61	-256.76 -40.61	-254.24 -40.61	-255.96 -40.61	-255.20 -40.61	-255.77 -40.61	-256.08 -40.61	-256.09 -40.61	-255.64 -40.61	-255.46 -40.61	-254.80 -40.61	-254.95 -40.61
	40	-254.67 -39.28	-256.76 -39.28	-254.24 -39.28	-255.96 -39.28	-255.20 -39.28	-255.77 -39.28	-256.08 -39.28	-256.09 -39.28	-255.64 -39.28	-255.46 -39.28	-254.80 -39.28	-254.95 -39.28
	50	-254.67 -51.28	-256.76 -51.28	-254.24 -51.28	-255.96 -51.28	-255.20 -51.28	-255.77 -51.28	-256.08 -51.28	-256.09 -51.28	-255.64 -51.28	-255.46 -51.28	-254.80 -51.28	-254.95 -51.28
	60	-254.67 -54.60	-256.76 -54.60	-254.24 -54.60	-255.96 -54.60	-255.20 -54.60	-255.77 -54.60	-255.08 -54.60	-256.09 -54.60	-255.64 -54.60	-255.46 -54.60	-254.80 -54.60	-254.95 -54.60
	70	-254.67 -63.49	-256.76 -63.49	-254.24 -63.49	-255.96 -63.49	-255.20 -63.49	-255.77 -63.49	-256.08 -63.49	-256.09 -63.49	-255.64 -63.49	-255.46 -63.49	-254.80 -63.49	-254.95 -63.49
	80	-254.67 -69.27	-256.76 -69.27	-254.24 -69.27	-255.96 -69.27	-255.20 -69.27	-255.77 -69.27	-256.08 -69.27	-256.09 -69.27	-255.64 -69.27	-255.46 -69.27	-254.80 -69.27	-254.95 -69.27
	90	-254.67 -76.76	-256.76 -76.76	-254.24 -76.76	-255.96 -76.76	-255.20 -76.76	-255.77 -76.76	-256.08 -76.76	-256.09 -76.76	-255.63 -76.76	-255.46 -76.76	-254.80 -76.76	-254.95 -76.76
	100	-254.67 -83.46	-256.76 -83.46	-252.24 -83.46	-255.96 -83.46	-255.20 -83.46	-255.77 -83.46	-256.08 -83.46	-256.09 -83.46	-255.64 -83.46	-255.46 -83.46	-254.80 -83.46	-254.95 -83.46

TABLE 4.40: ψ_1, ψ_2 for Channel 4, ε = 5

		$d_1 = 10^{-7}$			$d_1 = 10^{-10}$			$d_1 = 10^{-15}$			$d_1 = 10^{-20}$		
d_2 = No of Iterations		0.050 25	0.075 25	0.100 25	0.050 27	0.075 27	0.100 26	0.050 30	0.075 30	0.100 29	0.050 30	0.075 30	0.100 29
No of taps in filter	20	-258.26 -57.31	-258.26 -57.31	-257.11 -57.31	-259.18 -57.31	-259.18 -57.31	-257.43 -57.31	-260.47 -57.31	-260.47 -57.31	-258.32 -57.31	-260.47 -57.31	-260.47 -57.31	-258.32 -57.31
	30	-258.26 -88.02	-258.26 -88.02	-257.11 -88.02	-259.18 -88.02	-259.18 -88.02	-257.43 -88.02	-260.47 -88.02	-260.47 -88.02	-258.32 -88.02	-260.47 -88.02	-260.47 -88.02	-258.32 -88.02
	40	-258.26 -118.71	-258.26 -118.71	-257.11 -118.71	-259.18 -118.73	-259.18 -118.73	-257.43 -118.73	-260.47 -118.73	-260.47 -118.73	-258.32 -118.73	-260.47 -118.73	-260.47 -118.73	-258.32 -118.73
	50	-258.26 -141.50	-258.26 -141.50	-257.11 -142.79	-259.18 -149.43	-259.18 -149.43	-257.43 -149.43	-260.47 -149.43	-260.47 -149.43	-258.32 -149.43	-260.47 -149.43	-260.47 -149.43	-258.32 -149.43
	60	-258.26 -142.27	-258.26 -142.27	-257.11 -143.85	-259.18 -180.14	-259.18 -180.14	-257.43 -180.14	-260.47 -180.14	-260.47 -180.14	-258.32 -180.14	-260.47 -180.14	-260.47 -180.14	-258.32 -180.14
	70	-258.26 -142.27	-258.26 -142.27	-257.11 -143.85	-259.18 -210.70	-259.18 -210.70	-257.43 -210.02	-260.47 -210.84	-260.47 -210.84	-258.32 -210.84	-260.47 -210.84	-260.47 -210.84	-258.32 -210.84
	80	-258.26 -142.27	-258.26 -142.27	-257.11 -143.85	-259.18 -225.52	-259.18 -225.52	-257.43 -217.65	-260.47 -241.49	-260.47 -241.49	-258.32 -241.45	-260.47 -241.49	-260.47 -241.49	-258.32 -241.45
	90	-258.26 -142.27	-258.26 -142.27	-257.11 -143.85	-259.18 -225.63	-259.18 -225.63	-257.43 -217.67	-260.47 -259.78	-260.47 -259.78	-258.32 -257.81	-260.47 -259.78	-260.47 -259.78	-258.32 -257.81
	100	-258.26 -142.27	-258.26 -142.27	-257.11 -143.85	-259.18 -225.63	-259.18 -225.63	-257.43 -217.67	-260.47 -260.04	-260.47 -260.04	-258.32 -257.96	-260.47 -260.04	-260.47 -260.04	-258.32 -257.96

TABLE 4.41: ψ_1, ψ_2 for Channel 1, ℓ = 6

		$d_1 = 10^{-7}$			$d_1 = 10^{-10}$			$d_1 = 10^{-15}$			$d_- = 10^{-20}$		
d_2 = No of Iterations		0.050 37	0.075 38	0.100 38	0.050 40	0.075 40	0.100 40	0.050 42	0.075 42	0.100 42	0.050 44	0.075 44	0.100 44
No of taps in filter	20	-262.59 -49.25	-264.37 -49.25	-264.37 -49.25	-263.48 -49.25	-263.15 -49.25	-263.15 -49.25	-264.27 -49.25	-274.71 -49.25	-264.71 -49.25	-263.95 -49.25	-264.87 -49.25	-264.87 -49.25
	30	-262.59 -79.42	-264.37 -79.42	-264.37 -79.42	-263.48 -79.42	-263.15 -79.42	-263.15 -79.42	-264.27 -79.42	-264.71 -79.42	-264.71 -79.42	-263.95 -79.42	-264.87 -79.42	-264.87 -79.42
	40	-262.59 -109.57	-264.37 -109.58	-264.37 -109.58	-263.48 -109.58	-263.15 -109.58	-263.15 -109.58	-264.27 -109.58	-264.71 -109.58	-264.71 -109.58	-263.95 -109.58	-264.87 -109.58	-264.87 -109.58
	50	-262.59 -138.64	-264.37 -139.73	-264.37 -139.73	-263.48 -139.73	-263.15 -139.73	-263.15 -139.73	-264.27 -139.73	-264.71 -139.73	-264.71 -139.73	-263.95 -139.73	-264.87 -139.73	-264.87 -139.73
	60	-262.59 -145.15	-264.37 -166.02	-264.37 -166.02	-263.48 -169.89	-263.15 -169.89	-263.15 -169.89	-264.27 -169.89	-264.71 -169.89	-264.71 -169.89	-263.95 -169.89	-264.87 -169.89	-264.87 -169.89
	70	-262.59 -145.16	-264.37 -168.31	-264.37 -168.31	-263.48 -200.05	-263.15 -200.04	-263.15 -200.04	-264.27 -200.05	-264.71 -200.05	-264.71 -200.05	-263.95 -200.05	-264.87 -200.05	-264.87 -200.05
	80	-262.59 -145.16	-264.37 -168.31	-264.37 -168.37	-263.48 -230.19	-263.15 -227.70	-263.15 -227.70	-264.27 -230.20	-247.71 -230.20	-264.71 -230.20	-263.95 -230.20	-264.87 -230.20	-264.87 -230.20
	90	-262.58 -145.16	-264.37 -168.31	-264.37 -168.31	-263.48 -253.90	-263.15 -231.28	-263.15 -231.28	-264.27 -258.15	-264.71 -258.38	-264.71 -258.38	-263.95 -258.21	-264.87 -258.51	-264.87 -258.51
	100	-262.59 -145.16	-264.37 -168.31	-264.37 -168.31	-263.48 -255.01	-263.15 -231.29	-263.15 -231.29	-264.27 -262.13	-264.71 -262.74	-264.71 -262.74	-263.95 -262.29	-264.87 -263.11	-264.87 -263.11

TABLE 4.42: ψ_1, ψ_2 for Channel 2, ℓ = 6

		$d_1 = 10^{-7}$			$d_1 = 10^{-10}$			$d_1 = 10^{-15}$			$d_1 = 10^{-20}$		
d_2 = No of Iterations		0.050 60	0.075 56	0.100 55	0.050 63	0.075 59	0.100 58	0.050 65	0.075 60	0.100 59	0.050 67	0.075 63	0.100 62
No of taps in filter	20	-93.83 -44.08	-93.83 -44.08	-93.83 -44.08	-93.83 -44.08	-93.83 -44.08	-93.83 -44.08	-93.83 -44.08	-93.83 -44.08	-93.83 -44.08	-93.83 -44.08	-93.83 -44.08	-93.83 -44.08
	30	-258.65 -71.06	-257.97 -71.06	-257.97 -71.06	-259.03 -71.06	-257.70 -71.06	-257.70 -71.06	-257.35 -71.06	-257.01 -71.06	-257.01 -71.06	-258.61 -71.06	-257.45 -71.06	-257.45 -71.06
	40	-258.69 -98.02	-257.97 -98.02	-257.97 -98.02	-259.03 -98.02	-257.70 -98.02	-257.70 -98.02	-257.35 -98.02	-257.01 -98.02	-257.01 -98.02	-258.61 -98.02	-257.45 -98.02	-257.45 -98.02
	50	-258.65 -124.97	-257.97 -124.97	-257.97 -124.97	-259.03 -124.97	-257.70 -124.97	-257.70 -124.97	-257.35 -124.97	-257.01 -124.97	-257.01 -124.97	-258.61 -124.97	-257.45 -124.97	-257.45 -124.97
	60	-258.69 -149.39	-257.97 -151.55	-257.97 -151.55	-256.03 -151.93	-257.70 -151.93	-257.70 -151.93	-257.35 -151.93	-257.01 -151.93	-257.01 -151.93	-258.61 -151.93	-257.45 -151.93	-257.45 -151.93
	70	-258.69 -152.93	-257.97 -162.32	-257.97 -162.32	-259.03 -178.88	-257.70 -178.88	-257.70 -178.88	-257.35 -178.88	-257.01 -178.88	-257.01 -178.88	-258.61 -178.88	-257.45 -178.88	-257.45 -178.88
	80	-258.69 -152.94	-257.97 -162.42	-257.97 -162.42	-259.03 -205.82	-257.70 -205.83	-257.70 -205.83	-257.35 -205.83	-257.01 -205.83	-257.01 -205.83	-258.61 -205.83	-257.45 -205.83	-257.45 -205.83
	90	-258.69 -152.94	-257.97 -162.42	-257.97 -162.42	-259.03 -229.25	-257.70 -232.67	-257.70 -232.67	-257.35 -232.77	-257.01 -232.77	-257.01 -232.77	-258.61 -232.77	-257.45 -232.77	-257.45 -232.77
	100	-258.69 -152.94	-257.97 -162.42	-257.97 -162.42	-259.03 -231.78	-257.70 -248.28	-257.70 -248.28	-257.35 -255.06	-257.01 -254.76	-257.01 -254.76	-258.61 -255.71	-257.45 -255.04	-257.45 -255.04

TABLE 4.43: ψ_1, ψ_2 for Channel 3, ℓ = 6

		$d_1 = 10^{-7}$			$d_1 = 10^{-10}$			$d_1 = 10^{-15}$			$d_1 = 10^{-20}$		
No of	d_2 = Iterations	0.050 110	0.075 107	0.100 103	0.050 114	0.075 112	0.100 109	0.050 121	0.075 117	0.100 113	0.050 123	0.075 120	0.100 117
No of taps in filter	20	-67.18 -22.21	-67.18 -22.21	-67.18 -22.21	-67.18 -22.21	-67.18 -22.21	-67.18 -22.21	-67.18 -22.21	-67.18 -22.21	-67.18 -22.21	-67.18 -22.21	-67.18 -22.21	-67.18 -22.21
	30	-254.67 -40.61	-256.76 -40.61	-254.24 -40.61	-255.96 -40.61	-255.20 -40.61	-255.77 -40.61	-256.08 -40.61	-256.09 -40.61	-255.64 -40.61	-255.46 -40.61	-254.80 -40.61	-254.95 -40.61
	40	-254.67 -39.28	-256.76 -39.28	-254.24 -39.28	-255.96 -39.28	-255.20 -39.28	-255.77 -39.28	-256.08 -39.28	-256.09 -39.28	-255.64 -39.28	-255.46 -39.28	-254.80 -39.28	-254.95 -39.28
	50	-254.67 -51.28	-256.76 -51.28	-254.24 -51.28	-255.96 -51.28	-255.20 -51.28	-255.77 -51.28	-256.08 -51.28	-256.09 -51.28	-255.64 -51.28	-255.46 -51.28	-254.80 -51.28	-254.95 -51.28
	60	-254.67 -54.60	-256.76 -54.60	-254.24 -54.60	-255.96 -54.60	-255.20 -54.60	-255.77 -54.60	-256.08 -54.60	-256.09 -54.60	-255.64 -54.60	-255.46 -54.60	-254.80 -54.60	-254.95 -54.60
	70	-254.67 -63.49	-256.76 -63.49	-254.24 -63.49	-255.96 -63.49	-255.20 -63.49	-255.77 -63.49	-256.08 -63.49	-256.09 -63.49	-255.64 -63.49	-255.46 -63.49	-254.80 -63.49	-254.95 -63.49
	80	-254.67 -69.27	-256.76 -69.27	-254.24 -69.27	-255.96 -69.27	-255.20 -69.27	-255.77 -69.27	-256.08 -69.27	-256.09 -69.27	-255.64 -69.27	-255.46 -69.27	-254.80 -69.27	-254.95 -69.27
	90	-254.67 -76.76	-256.76 -76.76	-254.24 -76.76	-255.96 -76.76	-255.20 -76.76	-255.77 -76.76	-256.08 -76.76	-256.09 -76.76	-255.64 -76.76	-255.46 -76.76	-254.80 -76.76	-254.95 -76.76
	100	-254.67 -83.46	-256.76 -83.46	-254.24 -83.46	-255.96 -83.46	-255.20 -83.46	-255.77 -83.46	-256.08 -83.46	-256.09 -83.46	-255.64 -83.46	-255.46 -83.46	-254.80 -83.46	-254.95 -83.46

TABLE 4.44: ψ_1, ψ_2 for Channel 4, ℓ = 6

		$d_1 = 10^{-7}$			$d_1 = 10^{-10}$			$d_1 = 10^{-15}$			$d_1 = 10^{-20}$		
	$d_2 =$	0.05	0.075	0.1	0.05	0.075	0.1	0.05	0.075	0.1	0.05	0.075	0.1
ε =	3	(25) -142.26 -118.69	(25) -142.26 -118.69	(25) -143.71 -118.70	(27) -221.44 -118.73	(27) -221.44 -118.73	(26) -216.38 -118.73	(30) -228.96 -118.73	(30) -228.96 -118.73	(29) -228.95 -118.73	(30) -228.96 -118.73	(30) -228.96 -118.73	(29) -228.95 -118.73
	4	(25) -142.26 -118.69	(25) -142.26 -118.69	(25) -143.71 -118.70	(27) -221.44 -118.73	(27) -221.44 -118.73	(26) -216.38 -118.73	(30) -228.96 -118.73	(30) -228.96 -118.73	(29) -228.95 -118.73	(30) -228.96 -118.73	(30) -228.96 -118.73	(29) -228.95 -118.73
	5	(25) -142.26 -118.69	(25) -142.26 -118.69	(25) -143.71 -118.70	(27) -221.44 -118.73	(27) -221.44 -118.73	(26) -216.38 -118.73	(30) -228.96 -118.73	(30) -228.96 -118.73	(29) -228.95 -118.73	(30) -228.96 -118.73	(30) -228.96 -118.73	(29) -228.96 -118.73
	6	(25) -142.26 -118.69	(25) -142.26 -118.69	(25) -143.71 -118.70	(27) -221.44 -118.73	(27) -221.44 -118.73	(26) -216.38 -118.73	(30) -228.96 -118.73	(30) -228.96 -118.73	(29) -228.95 -118.73	(30) -228.96 -118.73	(30) -228.96 -118.73	(29) -228.95 -118.73

TABLE 4.45: ψ_3, ψ_4 and the Number of Iterations Taken for Channel 1 (System 4.6)

		$d_1 = 10^{-7}$			$d_1 = 10^{-10}$			$d_1 = 10^{-15}$			$d_1 = 10^{-20}$		
	$d_2 =$	0.05	0.075	0.1	0.05	0.075	0.1	0.05	0.075	0.1	0.05	0.075	0.1
ε =	3	(36) -144.96 -109.57	(37) -201.64 -109.58	(37) -206.88 -109.58	(37) -203.74 -109.58	(37) -201.64 -109.58	(37) -206.88 -109.58	(41) -228.82 -109.58	(41) -228.83 -109.58	(41) -228.82 -109.58	(41) -228.82 -109.58	(42) -228.82 -109.58	(42) -228.84 -109.58
	4	(36) -144.96 -109.57	(37) -201.64 -109.58	(37) -206.88 -109.58	(37) -203.74 -109.58	(37) -201.64 -109.58	(37) -206.88 -109.58	(41) -228.82 -109.58	(41) -228.83 -109.58	(41) -228.82 -109.58	(41) -228.82 -109.58	(42) -228.82 -109.58	(42) -228.84 -109.58
	5	(37) -145.07 -109.57	(38) -168.13 -109.58	(38) -168.13 -109.58	(40) -228.91 -109.58	(40) -225.60 -109.58	(40) -225.60 -109.58	(42) -228.92 -109.58	(42) -228.89 -109.58	(42) -228.89 -109.58	(44) -228.91 -109.58	(44) -228.92 -109.58	(44) -228.92 -109.58
	6	(37) -146.07 -109.57	(38) -168.13 -109.58	(38) -168.13 -109.58	(40) -228.91 -109.58	(40) -225.60 -109.58	(40) -225.60 -109.58	(42) -228.92 -109.58	(42) -228.89 -109.58	(42) -228.89 -109.58	(44) -228.91 -109.58	(44) -228.92 -109.58	(44) -228.92 -109.58

TABLE 4.46: ψ_3, ψ_4 and the Number of Iterations Taken for Channel 2 (System 4.6)

		$d_1 = 10^{-7}$			$d_1 = 10^{-10}$			$d_1 = 10^{-15}$			$d_1 = 10^{-20}$		
	$d_2 =$	0.05	0.075	0.1	0.05	0.075	0.1	0.05	0.075	0.1	0.05	0.075	0.1
ε =	3	(49) -151.58 -98.02	(48) -152.85 -98.02	(48) -152.85 -98.02	(52) -215.00 -98.02	(51) -194.44 -98.02	(51) -194.44 -98.02	(54) -213.99 -98.02	(53) -213.98 -98.02	(53) -213.98 -98.02	(56) -214.01 -98.02	(56) -214.00 -98.02	(56) -214.00 -98.02
	4	(51) -142.44 -98.02	(50) -144.46 -98.02	(50) -144.46 -98.02	(54) -215.02 -98.02	(53) -214.06 -98.02	(53) -214.06 -98.02	(57) -213.99 -98.02	(55) -213.99 -98.02	(55) -213.99 -98.02	(58) -214.01 -98.02	(57) -214.01 -98.02	(57) -214.01 -98.02
	5	(65) -191.76 -98.02	(55) -155.77 -98.02	(54) -155.77 -98.02	(61) -210.89 -98.02	(57) -214.03 -98.02	(56) -214.03 -98.02	(64) -213.99 -98.02	(59) -213.98 -98.02	(58) -213.98 -98.02	(65) -214.01 -98.02	(61) -213.99 -98.02	(60) -213.99 -98.02
	6	(60) -144.17 -98.02	(56) -156.35 -98.02	(55) -156.35 -98.02	(63) -212.77 -98.02	(59) -214.06 -98.02	(58) -214.06 -98.02	(65) -214.00 -98.02	(60) -213.99 -98.02	(59) -213.99 -98.02	(67) -213.99 -98.02	(63) -213.99 -98.02	(62) -213.99 -98.02

TABLE 4.47: ψ_3, ψ_4 and the Number of Iterations Taken for Channel 3 (System 4.6)

		$d_1 = 10^{-7}$			$d_1 = 10^{-10}$			$d_1 = 10^{-15}$			$d_1 = 10^{-20}$		
	$d_2 =$	0.05	0.075	0.1	0.05	0.075	0.1	0.05	0.075	0.1	0.05	0.075	0.1
ε =	3	(118) -151.57 -39.28	(118) -151.57 -39.28	(117) -153.32 -39.28	(121) -199.21 -39.28	(121) -199.21 -39.28	(121) -207.53 -39.28	(127) -207.88 -39.28	(127) -207.88 -39.28	(126) -207.88 -39.28	(130) -207.83 -39.28	(130) -207.83 -39.28	(130) -207.84 -39.28
	4	(94) -152.66 -39.28	(92) -154.76 -39.28	(90) -153.28 -39.28	(98) -203.56 -39.28	(97) -197.27 -39.28	(96) -204.61 -39.28	(105) -207.87 -39.28	(102) -207.85 -39.28	(100) -207.86 -39.28	(107) -207.86 -39.28	(105) -207.89 -39.28	(104) -207.87 -39.28
	5	(103) -145.65 -39.28	(102) -146.13 -39.28	(98) -147.98 -39.28	(107) -207.91 -39.28	(107) -195.66 -39.28	(104) -201.14 -39.28	(114) -207.91 -39.28	(112) -207.88 -39.28	(108) -207.88 -39.28	(116) -207.88 -39.28	(115) -207.87 -39.28	(112) -207.87 -39.28
	6	(110) -146.65 -39.28	(107) -146.13 -39.28	(103) -147.98 -39.28	(114) -207.91 -3928	(112) -195.66 -39.28	(109) -201.14 -39.28	(121) -207.91 -39.28	(117) -207.88 -39.28	(113) -207.88 -39.28	(123) -207.88 -39.28	(120) -207.87 -39.28	(117) -207.87 -39.28

TABLE 4.48: ψ_3, ψ_4 and the Number of Iterations Taken for Channel 4 (System 4.6)

	ℓ	$d_1 = 10^{-7}$ $d_2 = 0.1$	$d_1 = 10^{-10}$ $d_2 = 0.1$	$d_1 = 10^{-15}$ $d_2 = 0.1$	$d_1 = 10^{-20}$ $d_2 = 0.1$
System 4.6	3	25	26	29	29
	4	25	26	29	29
	5	25	26	29	29
	6	25	26	29	29
System 4.4		19	20	23	23

TABLE 4.49: Number of Iterations Taken for Channel 1 (c=1)

	ℓ	$d_1 = 10^{-7}$ $d_2 = 0.1$	$d_1 = 10^{-10}$ $d_2 = 0.1$	$d_1 = 10^{-15}$ $d_2 = 0.1$	$d_1 = 10^{-20}$ $d_2 = 0.1$
System 4.6	3	37	37	41	42
	4	37	37	41	42
	5	38	40	42	44
	6	38	40	42	44
System 4.4		32	34	37	39

TABLE 4.50: Number of Iterations Taken for Channel 2 (c=1)

	ℓ	$d_1 = 10^{-7}$ $d_2 = 0.1$	$d_1 = 10^{-10}$ $d_2 = 0.1$	$d_1 = 10^{-15}$ $d_2 = 0.1$	$d_1 = 10^{-20}$ $d_2 = 0.1$
System 4.6	3	48	51	53	56
	4	50	53	55	57
	5	54	56	58	60
	6	55	58	59	62
System 4.4		36	39	40	44

TABLE 4.51: Number of Iterations Taken for Channel 3 (c=1)

	ℓ	$d_1 = 10^{-7}$ $d_2 = 0.1$	$d_1 = 10^{-10}$ $d_2 = 0.1$	$d_1 = 10^{-15}$ $d_2 = 0.1$	$d_1 = 10^{-20}$ $d_2 = 0.1$
System 4.6	3	117	121	126	130
	4	90	96	100	104
	5	98	104	108	121
	6	103	109	113	117
System 4.4		107	112	117	122

TABLE 4.52: Number of Iterations Taken for Channel 4 (c=1)

4.7 System 4.7

Consider the algorithm

$$\lambda_{i+1} = \lambda_i + c\,\frac{e_{i,o}}{\varepsilon_i} \tag{4.157}$$

where $$\varepsilon_i = e_{i,1} - e_{i,2}\lambda_i + e_{i,3}\lambda_i^2 + \ldots + e_{i,g}(-\lambda_i)^{g+1} \tag{4.158}$$

and c is a real positive constant.

The operation of the receiver which finally leads to the use of the above algorithm to give the new estimate λ_{i+1} is the same as System 4.4 where at every step of the iterative process, the whole of the sequence Y (equation 4.104) is fed through the one-tap feedback filter (Figure 4. 18), starting with y_g and finishing with y_0, to give the sequence of $\{e_{i,h}\}$ (equation 4.106). This sequence, which is operated on to give the new estimate λ_{i+1}, is clearly a function of the sequence Y and the tap gain λ_i.

From equations 2.4 and 4.102

$$\begin{aligned} Y(z)A_i(z) &= (y_0 + y_1z^{-1} + \ldots + y_gz^{-g})(1 + \lambda_iz)^{-1} \\ &= (y_0 + y_1z^{-1} + \ldots + y_gz^{-g})(1 - \lambda_iz + \lambda_i^2 z^2 - \lambda_i^3z^3 + \ldots) \end{aligned} \tag{4.159}$$

If

$$Y(z)A_i(z) = \ldots + e_{i,-1}z^1 + e_{i,o}z^0 + e_{i,1}z^{-1} + \ldots + e_{i,g}z^{-g} \quad (4.160)$$

(equation 4.106) then

$$e_{i,o} = y_o - \lambda_i y_1 + \lambda_i^2 y_2 - \ldots + (-\lambda_i)^g y_g \quad (4.161)$$

$$e_{i,1} = y_1 - \lambda_i y_2 + \lambda_i^2 y_3 - \ldots + (-\lambda_i)^{g-1} y_g \quad (4.162)$$

$$e_{i,2} = y_2 - \lambda_i y_3 + \lambda_i^2 y_4 - \ldots + (-\lambda_i)^{g-2} y_g \quad (4.163)$$

$$e_{i,3} = y_3 - \lambda_i y_4 + \lambda_i^2 y_5 - \ldots + (-\lambda_i)^{g-3} y_g \quad (4.164)$$

$\vdots$

etc

Substituting the values of $e_{i,1}$, $e_{i,2}$, $e_{i,3}$, ... into equation 4.158

$$\begin{aligned}\varepsilon_i &= y_1 - \lambda_i y_2 + \lambda_i^2 y_3 - \lambda_i^3 y_4 + \ldots + (-\lambda_i)^{g-1} y_g \\ &\quad - \lambda_i (y_2 - \lambda_i y_3 + \lambda_i^2 y_4 - \lambda_i^3 y_5 + \ldots + (-\lambda_i)^{g-2} y_g) \\ &\quad + \lambda_i^2 (y_3 - \lambda_i y_4 + \lambda_i^2 y_5 - \lambda_i^3 y_6 + \ldots + (-\lambda_i)^{g-3} y_g) \\ &\quad - \lambda_i^3 (y_4 - \lambda_i y_5 + \lambda_i^2 y_6 - \lambda_i^3 y_7 + \ldots + (-\lambda_i)^{g-4} y_g) \\ &\quad + \ldots \\ &= y_1 - 2\lambda_i y_2 + 3\lambda_i^2 y_3 - 4\lambda_i^3 y_4 + \ldots + g(-\lambda_i)^{g-1} y_g \quad (4.165)\end{aligned}$$

The equation above shows clearly that ε_i is a function of a weighted sequence of Y, and that the weightings increase in powers of λ_i. Therefore if λ_i is small, terms containing higher powers of λ_i can be ignored. Furthermore, if one examines the z-transforms of the sampled impulse-responses of the Channels (Figures 2.13-2.16), it can be seen that most ofthe energy is concentrated in the leading portion of the sequence. Since the same sort of weightings are used in the evaluation of $e_{i,o}$ (equation 4.161), and therefore in the division $e_{i,o}/\varepsilon_i$, it may be possible to use a truncated sequence of Y and hence giving a saving in the number of calculations required.

The aim of this test is to determine the amount of truncation that can be used in this system without causing problems in the root-finding process, and the total number of iterations required for each case. Channels 1-4 are used in this experiment and the length of the sequence Y is varied from 4 to 16 components, starting with y_0, and the results are shown in Tables 4.53-4.56.

The results are obtained by the System 4.4 as described in Section 4.4 where a threshold of 10^{-10} was used. [Except in the case of Channel 4 where the sequence Y is truncated to eight components giving the sequence $y_0, y_1, \ldots, y_7$ and for c = 1.0, correct operation is obtained in all the other cases. This is extremely encouraging because it demonstrates that a truncated sequence Y can be used in the root-finding process. Indeed, the number of components in the sequence Y can be reduced to 12 without significantly increasing the total number of iterations taken to find all the required roots, and for the two milder channels, Channels 1-2, this number can be reduced to as little as eight components without increasing the total number of iterations required at all.]

Since the number of components in Y is reduced just for the convenience of the root-finding process, a slightly different procedure is required to obtain the estimate of the channel (with g+1 components) and filter. Having found the set of roots by this system, the one-tap feedback filter is set to the first of the $\{\beta_h\}$ (negative reciprocals of the roots), β_1, and the whole of the sequence Y is fed through it in the usual reverse order to give a (g+1) output sequence. This sequence is then fed through the two-tap feedforward filter whose tap is now set to β_1^*. Only (g+1) components (= F_1) from the output of the feedforward filter is stored. The process is repeated for each of the $\{\beta_h\}$ found, and each time a new F_h is used. At the end of the process, assuming that there are m roots located by the algorithm F_m is the required estimate of the sampled impulse-response of the channel and filter. The method for determining the coefficients of the adaptive filter remains unchanged.

As described in Section 4.4.1, the adjustment time of the filter is the sum of the time taken to find all the roots outside the unit circle and the time taken to evaluate the tap coefficient of the filter. For each iteration of the root finding process, 2g complex multiplications and 1 complex division are required. Furthermore there are 2n+g+2 complex multiplications required for each root located except for the first which requires g+1. Therefore, significant saving in the number of complex arithmetic operations can be achieved by truncating Y.

Channel 1 (3 Roots) g = 19					
Stepsize c	No. of Components Used				
	4	8	12	16	20
0.5	62	62	62	62	60
0.7	38	38	38	38	38
1.0	22	21	21	21	20

TABLE 4.53: Number of Iterations Taken to Find all {β} for Channel 1

Channel 2 (4 Roots) g = 19					
Stepsize c	No. of Components Used				
	4	8	12	16	20
0.5	79	78	78	78	77
0.7	91	54	54	54	52
1.0	68	40	40	40	34

TABLE 4.54: Number of Iterations Taken to Find all {β} for Channel 2

Channel 3 (4 Roots) g = 25						
Stepsize c	No. of Components Used					
	4	8	12	16	20	26
0.5	127	102	100	99	99	91
0.7	91	66	66	67	67	64
1.0	68	40	42	40	40	39

TABLE 4.55: Number of Iterations Taken to Find all {β} for Channel 3

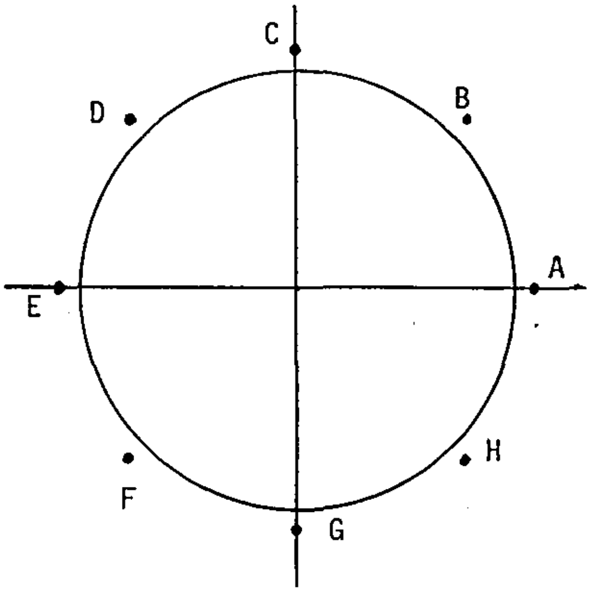
Channel 4 (8 Roots) g = 26						
Stepsize c	No. of Components Used					
	4	8	12	16	20	27
0.5	513	233	195	195	195	180
0.7	436	207	140	140	139	140
1.0	448	X	124	118	118	112

TABLE 4.56: Number of Iterations Taken to Find all {β} for Channel 4

4.8 Performance of System 4.4 for Roots Lying Just Outside the Unit Circle

This section is concerned with the study of the performance of System 4.4 when there is a root lying very close to, and just outside the unit circle in the z-plane. The study is necessary since it gives an idea of how vulnerable the algorithm is, and it also prepares the ground work for the potential application of the algorithm for the 64-point QAM system operating at 19200 bit/s. In such a system, the z-transform of the sampled impulse-response of the channel, Y(z), often has a number of roots (zero) lying just outside the unit circle.

Channels 1-4 were used for this experiment and, in here, in addition to the original roots outside the unit circle, an extra root,$-\frac{1}{\beta}$, is artificially introduced at one of the eight positions as shown in Figure 4.24.



A. $-1/\beta = 1.0830 + j0.0000$

B. $-1/\beta = 0.7658 + j0.7658$

C. $-1/\beta = 0.0000 + j1.0830$

D. $-1/\beta = -0.7658 + j0.7658$

E. $-1/\beta = -1.0830 + j0.0000$

F. $-1/\beta = -0.7658 - j0.7658$

G. $-1/\beta = 0.0000 - j1.0830$

H. $-1/\beta = 0.7658 - j0.7658$

and $|\frac{1}{\beta}| = 1.0830$

FIGURE 4.24: Positions for the Extra Root

The positions of the roots are chosen to spread evenly outside the unit circle and to have the same magnitude of 1.083 which is the same as that of Root 8 of Channel 4 (Figure 2.12). It is thought that the performance of the algorithm on the eight cases, given the proximity of the roots to the unit circle, is a good measure of its capability of handling roots just outside the unit circle. The operation of the root-finding algorithm and all the other details are given in Section 4.4. A threshold $d_1 = 10^{-10}$ was used.

The results of computer-simulation tests are summarised in Table 4.57 where the successful operation is indicated by a tick together with the total number of iterations taken to find all the roots outside the unit circle including the one introduced artificially, and the failure to locate all the roots is indicated by a cross. It can be seen that the algorithm works successfully in all of the eight cases with Channels 1 and 2 and the total number of iterations taken has doubled in the worst case which occurs with Channel 1 when the extra root is positioned at E (Figure 4.24). The introduction of the extra root proved to be critical to the successful operation of the algorithm for Channels 3 and 4 where it was found that both the number of roots and the order in which they were located were dependent on the actual position of the root. For example, in cases where the root is introduced at positions B, C, D and E of Channel 3 the algorithm failed only to find the extra root, whereas in Channel 4, the algorithm could only locate 1 root (Root 3) when the extra root is introduced at position A but failed only on the extra root when the latter is introduced at positions B, C and E. It has become clear that when there are many roots lying outside the unit circle and when some of them are lying very close to it, it is not possible to predict which root or roots are most likely to be found.

TABLE 4.57: Results of Computer-Simulation Tests

Position of Root	CHANNEL 1	CHANNEL 2	CHANNEL 3	CHANNEL 4
A	✓ 31	✓ 38	✓ 51	x
B	✓ 29	✓ 34	x	x
C	✓ 37	✓ 38	x	x
D	✓ 29	✓ 49	x	x
E	✓ 40	✓ 46	x	x
F	✓ 39	✓ 54	✓ 62	x
G	✓ 35	✓ 44	✓ 52	✓ 131
H	✓ 33	✓ 41	✓ 52	x

Detailed analysis of the value of λ_i at each iteration i has revealed that because of the proximity of the extra root to the unit circle, in cases such as C and D, the components in the resultant 'raw' sampled impulse-response is such that $|y_0| < |y_1|$. Furthermore, during the process in which the roots are removed and replaced by their complex conjugate-reciprocals, the above condition can also occur. The problems caused by $|y_0| < |y_1|$, together with the large stepsize (c=1) in the algorithm mean that during the iterative root-finding process, the algorithm actually overshoots across the unit circle and subsequently results in a restart of the process. One solution is to use a smaller stepsize in the algorithm. Different stepsizes were used in Channel 3 and Channel 4 and it was found that the smaller stepsize worked very well with Channel 3, enabling all the roots to be found. Although this modification did not result in all the roots to be found for Channel 4, significant improvement was obtained. The results of computer-simulation tests for Channel 4 are given in Table 4.58 where the number of roots and the total number of iterations taken, given in parentheses, are tabulated for each of the eight cases. Note that the smaller stepsize (0.5) does not always result in more roots but in cases such as C and D the improvement is quite considerable.

TABLE 4.58: Results of Computer-Simulation Test for Channel 4

Position of Root	STEPSIZE			
	1.0	0.75	0.5	0.25
A	1 (52)	9 (421)	9 (402)	8 (418)
B	8 (359)	9 (405)	8 (357)	8 (351)
C	8 (357)	0 (17)	8 (362)	9 (415)
D	1 (51)	1 (56)	8 (357)	8 (368)
E	8 (388)	8 (365)	8 (375)	8 (403)
F	6 (274)	9 (437)	9 (352)	9 (551)
G	9 (421)	9 (435)	9 (426)	9 (490)
H	3 (141)	9 (411)	9 (418)	9 (419)

Since the smaller the stepsize the slower the rate of convergence of the algorithm, therefore, the algorithm, even when a suitable stepsize can be found, has the disadvantage of operating at a slow speed for practically most of the time. An alternative approach is to allow the value of $|\lambda_i|$ (absolute value of λ_i) to exceed the value of unity once, and only on the second occurrence is the process restarted. The modification seems to operate very well for Channel 3 where it is possible for the algorithm to find all of the roots in all of the eight cases. However, in Channel 4, this modification does not yield any improvement on the number of located roots and the number of roots located by the algorithm remained unchanged even when $|\lambda_i|$ is allowed to exceed 1 twice. But as this number is increased to three times, more roots are resulted and, in particular, 8 roots are now located (as compared with 1 before) in case D and in cases E and H, all of the 9 roots are now found.

4.9 Performance of the System 4.4 for Repeated Roots

This section deals with another potential problem for the root-finding algorithm (System 4.4) which is that of repeated roots. Here, the results of the study should provide information on the effectiveness of the algorithm in situations where the z-transform of the channel, Y(z), has a pair of repeated roots, or more likely, two roots which lie very close to each other outside the unit circle in the z-plane.

Computer simulations were carried out on Channels 1-4 where it was assumed that each of the roots of Y(z) outside the unit circle is, in turn, repeated for each channel. For example, in Channel 1 (with 3 roots outside the unit circle (Figure 2.9)), an extra root is deliberately introduced such that the resultant z-transform of the sampled impulse-response has a pair of Root 1, Root 2 and Root 3 in three separate cases. The other roots of course remain unchanged. Details of the operation and the values of the different parameters used in the root-finding algorithm are as described in Section 4.4.

Results of this experiment are summarised in Table 4.59 which shows the performance of the algorithm for a threshold d_1 (equation 4.105) of 10^{-10} and c = 1 (equation 4.114) for the four different channels with the appropriate repeated roots as indicated. The successful operation of the algorithm in locating all the required roots is indicated by a tick together with the total number of iterations taken given in parentheses, whilst the failure to do so is indicated by a cross.

It can be seen that the algorithm works very well with Channels 1-3 and the introduction of the root puts an extra of about 20 iterations on the overhead (Tables 4.13-4.16). In the case of Channel 4, where there are 8 roots lying outside the unit circle, the introduction of a repeated root (making the total 9) caused the algorithm to fail to locate all of them in five cases out of eight(Table 4.59). This indicates that the stepsize, c=1, is too big. Table 4.60 shows the performance of System 4.4 for Channel 4 with threshold $d = 10^{-10}$ and different values of c (equation 4.114). Again, the table indicates the success of finding all the roots by a tick and the failure to do so by a cross which is followed by the actual number of roots located with the number of iterations taken given in parentheses. It can be seen that a smaller stepsize is all that is required for the algorithm of System 4.4 to locate all the roots, including the extra one, outside the unit circle. Although very small c must be avoided in practice because of the slow speed, $c = 0.75$ seems to operate fairly adequately in all cases and is about 30-50% slower than the case with $c=1$.

Drawing on the experience gained in Section 4.7 where the algorithm is modified to allow the value of $|\lambda_i|$ to exceed unity (once) in the iterative root-finding process when a root very close to the unit circle is deliberately introduced, the same technique is used here with the condition $|\lambda_i|>1$ allowed to happen for a number of times. It was found that even allowing $|\lambda_i|$ to exceed unity 6 times, it was not possible to find all of the required roots of Channel 4 for cases with repeated Root 1 and Root 4. However, for repeated Roots 6, 7 and 8, all of the roots are found in the 190, 224 and 184 iterations, respectively, when $|\lambda_i|>1$ is allowed for 5 times.

Repeated root	Channel 1 (3+1) roots	Channel 2 (4+1) roots	Channel 3 (4+1) roots	Channel 4 (8+1) roots
Root 1	✓ (37)	✓ (50)	✓ (56)	x
Root 2	✓ (44)	✓ (48)	✓ (56)	✓ (128)
Root 3	✓ (45)	✓ (45)	✓ (67)	✓ (105)
Root 4		✓ (52)	✓ (63)	x
Root 5				✓ (111)
Root 6				x
Root 7				x
Root 8				x

TABLE 4.59: Performance of System 4.4 with Repeated Roots

Repeated root	c=1.0	c=0.75	c=0.50	c=0.25
Root 1	x 2 (36)	✓ 9 (151)	✓ 9 (233)	✓ 9 (401)
Root 2	✓ 9 (128)	✓ 9 (170)	✓ 9 (225)	✓ 9 (402)
Root 3	✓ 9 (105)	✓ 9 (150)	✓ 9 (191)	✓ 9 (369)
Root 4	x 1 (37)	✓ 9 (118)	✓ 9 (240)	✓ 9 (449)
Root 5	✓ 9 (111)	✓ 9 (161)	✓ 9 (232)	✓ 9 (398)
Root 6	x 3 (50)	✓ 9 (163)	✓ 9 (230)	✓ 9 (480)
Root 7	x 6 (105)	✓ 9 (188)	✓ 9 (280)	✓ 9 (589)
Root 8	x 6 (102)	✓ 9 (196)	✓ 9 (254)	✓ 9 (551)

TABLE 4.60: Performance of System 4.4 for Channel 4

4.10 Discussion

The results in this Chapter have shown that the off-line systems are much more superior than the on-line systems studied in Chapter 3, and proved to be the answer for the adjustment of the linear filter ahead of the detector and, at the same time, giving the resultant sampled impulse-response of the channel and filter which is required by the detector. The fact that the off-line systems operate solely and directly on the sampled impulse-response of the channel, Y, avoids the problems caused by the additive white Gaussian noise in the received signal as experienced in Chapter 3, and also results in a simpler system configuration and a much faster operation.

Two different configurations have been studied, the systems which require both the two-tap feedforward and the one-tap feedback filters working in cascade (Figure 4.1), and those which require only the feedback filter (Figure 4.18) in the iterative root-finding process. System 4.1 and System 4.2 belong to the former type and are designed to deal with the problem of finding one of the remaining roots of Y(z) (equation 4.13) outside the unit circle assuming that all of the previously located roots, if any, have been removed and replaced by their complex conjugate-reciprocals. This assumption entails the computation of m-1 (where m is the number of roots lying outside the unit circle) 'correct' resultant sampled impulse-responses of the channel, where each one is assumed to have 1,2,...,m-1 roots processed perfectly. The computation is carried out separately and in parallel with the outcome of the root-finding algorithms using the root-finding NAG subroutine CO2ADF[50] to give the resultant Y just before any further root-finding operation. For example, starting with the raw Channel 1, when Root 3 (Figure 2.9) has been located by the algorithm, the resultant sampled impulse-response of Channel 1 with Root 3 removed and replaced by its complex conjugate-reciprocal is calculated separately and used in further root-finding operations. Since the finding of the m roots by the algorithm is not a continuous process and, furthermore, neither the termination at convergence to a required root nor at the end of the process where all the roots have been located is automatic, the systems cannot be used in any practical situations without further modifications. However, the results give the necessary information on the capabilities of the root-finding algorithms and set the definite direction for further developments. The algorithm in System 4.1 is divided into two parts, the first part is designed to give a fast convergent rate (equation 4.35) whilst the second part, which resembles an averaging process, is meant to reduce the fluctuations in the final estimate. System 4.2 reverses the order of the two-part operation of System 4.1 and performs the averaging process first. Although the algorithms give very promising results for most of the cases, they, nevertheless, fail to locate some of the roots in Channel 4, which is a typical worst telephone circuit for the transmission of digital data at 1200 bit/s. Furthermore, because the algorithms converge very quickly to a root and do not encounter large fluctuations in the estimates once the algorithms start to converge, no useful purpose is served by the averaging process and a faster convergence time can be achieved, for a given error in the estimate, by leaving out the averaging process.

System 4.3.1 is a continuous root-finding algorithm which operates on Y to give all its roots outside the unit circle in the z-plane, and at the end of the root-finding process, produces an estimate of the sampled impulse-response of the channel and filter. In addition to the various criteria it uses to enable a continuous and automated process, it employs five possible starting-points in its starting and restart procedures. The extra starting-points and the inclusion of a restart procedure in the process resulted in an excellent performance of the system on all four channels and although the order in which the five starting-points ärechosen is not optimum, there is very little between the various configurations studied (Section 4.3.2).

System 4.4 (Section 4.4) is a further development of the off-line system where only the one-tap feedback filter (Figure 4.18) is used in the iterative root-finding process instead of both the feedforward and the feedback filters (Figure 4.1). This configuration is simpler than System 4.3.1 and requires (g+2) less multiplications and (g+2) less additions per iteration. In terms of the number of arithmetic operations required, this represents about 50% reduction in the root-finding process. The results are extremely good and the time required to find all the roots, to form the resultant sampled impulse-response of the channel and filter and to calculate the coefficients of a 50-tap adaptive linear filter, assuming the use of a 16-bit signal processor with a 250 ns multiplying time (such as the NEC 7720[97]), is less than 10 ms for the worst case considered. Much better accuracy in the estimates of the roots can be achieved at a slight increase in the number of multiplications, or conversely, for a given accuracy, more savings in the number of arithmetic operations can be made using System 4.5.

Section 4.6 is a detailed study of System 4.6 as a possible alternative of System 4.4 where an averaging process is incorporated at the beginning of the iterative process. The idea is for the estimate to converge to the vicinity of one of the required roots during the first part of the algorithm, so that a much faster convergent rate can be achieved in the second part of the process where the normal algorithm (equation 4.145) is used. A small threshold value d_2 is associated with the averaging period whereby when $|e_{i,o}/\varepsilon_i|^2 < d_2$ (equation 4.156) the averaging process is switched over to the normal algorithm regardless of whether or not the averaging period has expired and continues with the normal algorithm thereafter. The results of this system have shown that, in most of the cases, it takes slightly longer for this system to find all the required roots than System 4.4, however, for the very worst case, Channel 4, it is appreciably faster. The extra processing of the system amounts to two multiplications and a few housekeeping operations and as these occur during the averaging period (usually between 3 to 6 iterations), therefore, the adjustment (set-up) time for this system would be roughly the same as that of System 4.4.

The set-up time for the adaptive filter is the total time taken to find all the roots outside the unit circle to form the estimate of the channel and filter, and the time taken to calculate the filter coefficients, which is largely determined by the former process. The amount of arithmetic operations per iteration of the root-finding algorithm depends mainly on the number of components in the sampled impulse-response of the channel. Clearly, if the number of components in Y can be reduced without significantly altering the number and position of the roots outside the unit circle then a large amount of processing time can be saved. System 4.7 is a modification of System 4.4 in which the number of components in Y is reduced by truncating the trailing end of Y, starting from y_g (equation 4.104). For a threshold value of 10^{-10} (i.e. $|e_{i,o}/\varepsilon_i|^2 < 10^{-10}$ (equation 4.115)) the results show that at least up to half of the number of components in the original Y can be truncated without significantly affecting any of the roots outside the unit circle and the number of iterations taken by the algorithm is only increased by 2 or 3.

Sections 4.8 and 4.9 looked at the performance of System 4.4 in cases where Y(z), in addition to the total number of roots outside the unit circle, has one extra root lying very close just outside the unit circle, and cases where Y(z) has one repeated root outside the unit circle. These are hypothetical situations which are unlikely to occur in practice, but the performance of the algorithm (System 4.4) would give an indication of the vulnerability of the system in some extremely unfavourable conditions. Again, the exact sampled impulse-responses for such cases have to be worked out separately by using the root-finding NAG subroutine CO2ADF[50]. The results have shown that the algorithm (System 4.4) is robust enough to cope with most of the cases without any modifications. However, in cases such as Channel 3 and Channel 4, a smaller stepsize (c = 0.75 say) is required. The modification to the algorithm which allows $|\lambda_i| > 1$ for b number of times before the process is restarted with a new starting-point proved to be very effective most of the time although no great advantages can be gained for using b > 3. Both of these techniques necessarily mean an increase in the number of iterations taken but, in general, the use of a smaller c works better than the other method and if one is prepared to settle with a slow convergent speed, it seems that a small c can be found for all the cases.

The results of various tests obtained for System 4.4 can be extended to that of System 4.6 because of similarity of the two algorithms. Indeed, System 4.6 is identical to that of System 4.4 except for the first few (3-6) iterations and therefore, all that has been said for System 4.4 would be true for System 4.6.

In view of the excellent performance of System 4.4 for the wide range of telephone circuits and its robustness in meeting potentially difficult situations and with areas for further improvement, the system as described in this Chapter is proposed to be the basic system design for the adjustment of the adaptive linear filter ahead of the detector for the transmission rate of 9600 bit/s hereafter known as the Standard algorithm. The operational details are exactly as described in Section 4.4 with $c = 1$ (equation 4.114), $d_1 = 10^{-10}$ (equation 4.114) and $(n+1) = 40$ (equation 4.125). System 4.6 (Section 4.6) is proposed to be an alternative version, designated as the Modified algorithm with $c = 1$ (equation 4.164), $d_1 = 10^{-10}$ (equation 4.115), $d_2 = 10^{-1}$ (equation 4.112) and ℓ (the averaging period) = 4 and number of taps in linear filter = 40.

CHAPTER 5

PERFORMANCE OF THE STANDARD AND MODIFIED ALGORITHMS FOR THE 64-POINT QAM SYSTEM OPERATING AT 19200 BIT/S.

As a result of the extensive computer-simulation tests based on on-line configuration, two very promising techniques have emerged for adjusting the tap-gains of the linear filter ahead of the detector and at the same time giving an estimate of the sampled impulse-response of the channel and filter. The techniques rely on the fact that the linear filter operates by removing all the roots (zeros) of Y(z) of the channel that lie outside the unit circle and replacing them by their complex conjugate-reciprocals while leaving the remaining roots unchanged. It is essentially a root-finding problem and the techniques, which require only the estimate of the sampled impulse-response ofthe channel give very fast speed of convergence and extremely accurate results in the root-finding process. Further tests have shown that the root-finding algorithms are versatile enough to operate satisfactorily in potentially difficult situations which might occur under very poor conditions. In view of encouraging outcome of the tests, the two algorithms as described in Sections 4.4 and 4.6 are proposed to be the Standard algorithm and the Modified algorithm (Section 4.10). The aim of this Chapter is to look into the possibility of applying the algorithms to a 64-point QAM system operating at 19200 bit/s in the British public switched telephone network.

The model of this QAM data-transmission system is the same as that of the 16-point QAM system (Figure 2.1) except that the transmitted signal is now a stream of 64-level QAM signal-elements with a carrier frequency of 1800 Hz and a signal rate of 3200 bauds. The information to be transmitted is carried by the data-symbols $\{s_i\}$, where

$$s_i = s_{0,i} + js_{1,i} \tag{5.1}$$

where $s_{0,i}$ and $s_{1,i}$ may take on any integer value in the range -7 to 7 and $j = \sqrt{-1}$. The statistical properties of the data-symbols and noise samples together with all the assumptions are the same as in the 16-point QAM system. A detailed description of the QAM system is given in Chapter 2.

The same set of telephone circuits (circuits 1-4) are used in the investigation and their attenuation and group delay characteristics are as shown in Figures 2.4-2.7. The equipment filters are different from those of the 16-point QAM system and their combined characteristics can be found in Figure 5.1. Tables 5.1 and 5.2 give both the raw sampled impulse-responses of the four telephone circuits and their corresponding sampled impulse-responses when made minimum phase, all sampled at 3200 bauds. The magnitudes of the components of the raw and minimum-phase sampled impulse-responses of the channels, Y (equation 2.3), are also shown graphically in Figures 5.2-5.9. The channels are designated as Channels 5-8 to distinguish them from the 16-point QAM system where the sampling rate is 2400 bauds. The roots (zeros) of Y(z) for the four channels are shown in Figures 5.10-5.13, where those lying outside the unit circle are numbered in the order of absolute value, starting at No. 1 with the root having the largest magnitude. It can be seen that the number of roots outside the unit circle has now increased to a number between 8 and 13 where many of them are lying just outside the unit circle. This condition might conceivably cause some problems to the algorithms of System 4.4 and System 4.6 which have been found, in some cases, to miss some of the required roots in hypothetical situations where an extra root is deliberately introduced amongst a large number of roots already existing outside the unit circle (Section 4.2.8).

The Standard algorithm and the Modified algorithm as proposed in Section 4.10 are tested whose results are given in the latter parts of this Chapter. It is found that the algorithms, with slight modifications, operate very well in all cases including Channel 8 for the 64-point QAM system operating at 19200 bit/s.

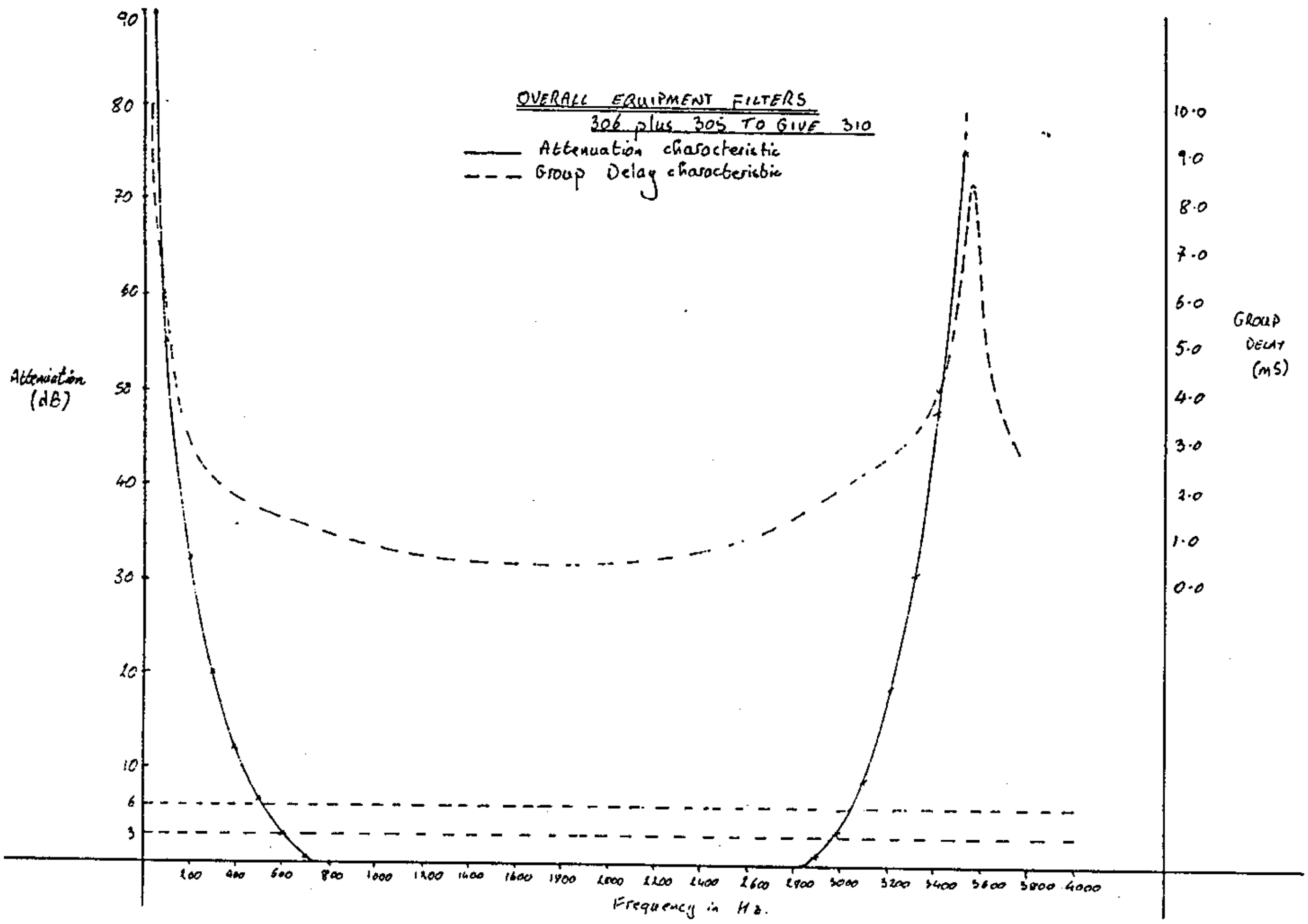


FIGURE 5.1: The combined characteristics of the overall equipment filters

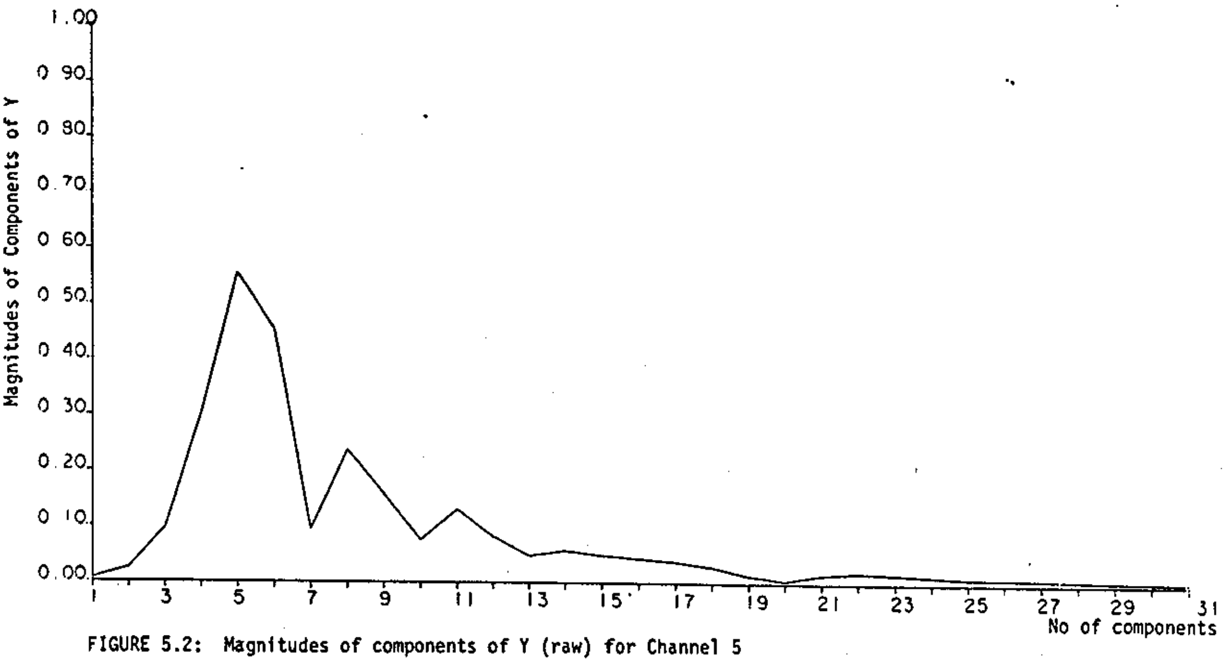


FIGURE 5.2: Magnitudes of components of Y (raw) for Channel 5

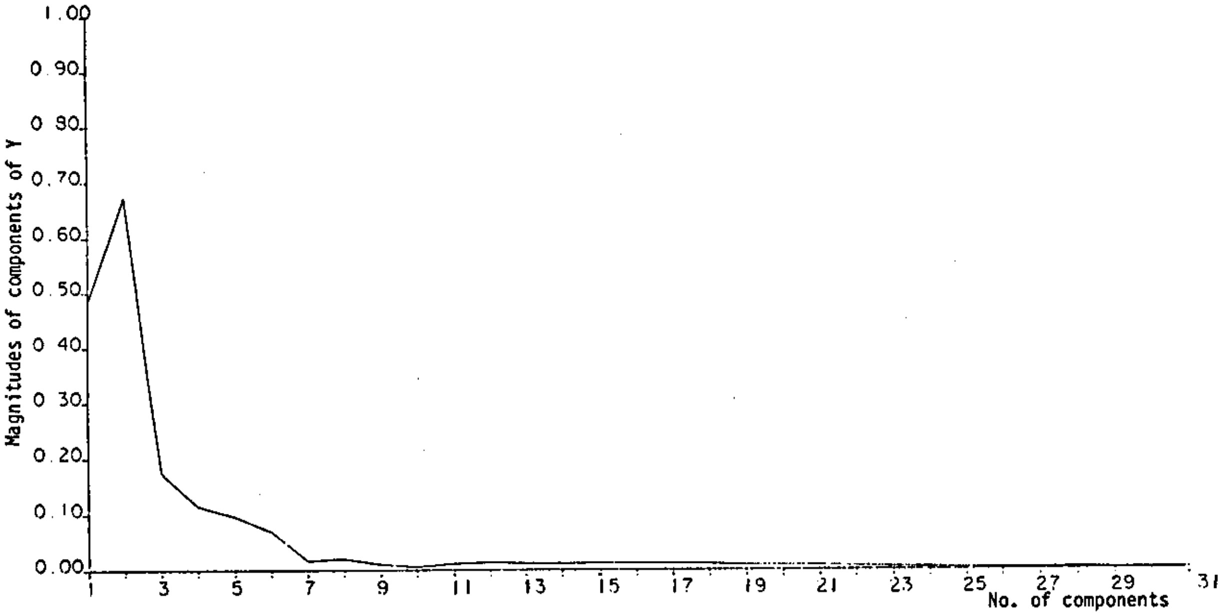


FIGURE 5.3: Magnitudes of components of Y (min. phase) for Channel 5

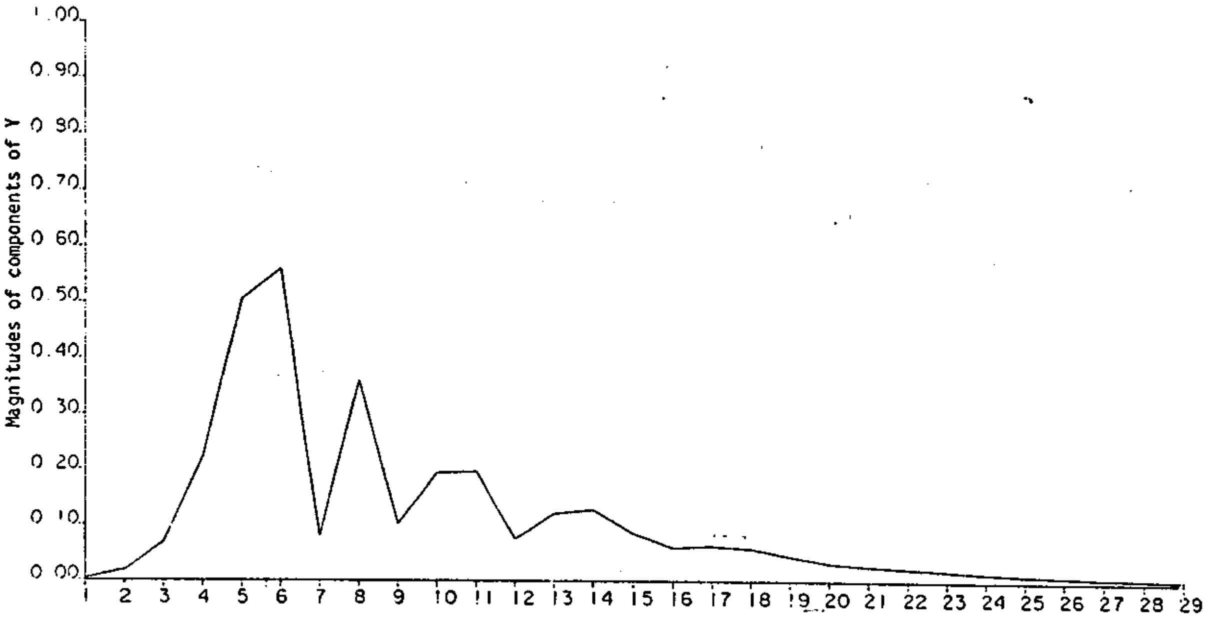


FIGURE 5.4: Magnitudes of components of Y (raw) for Channel 6

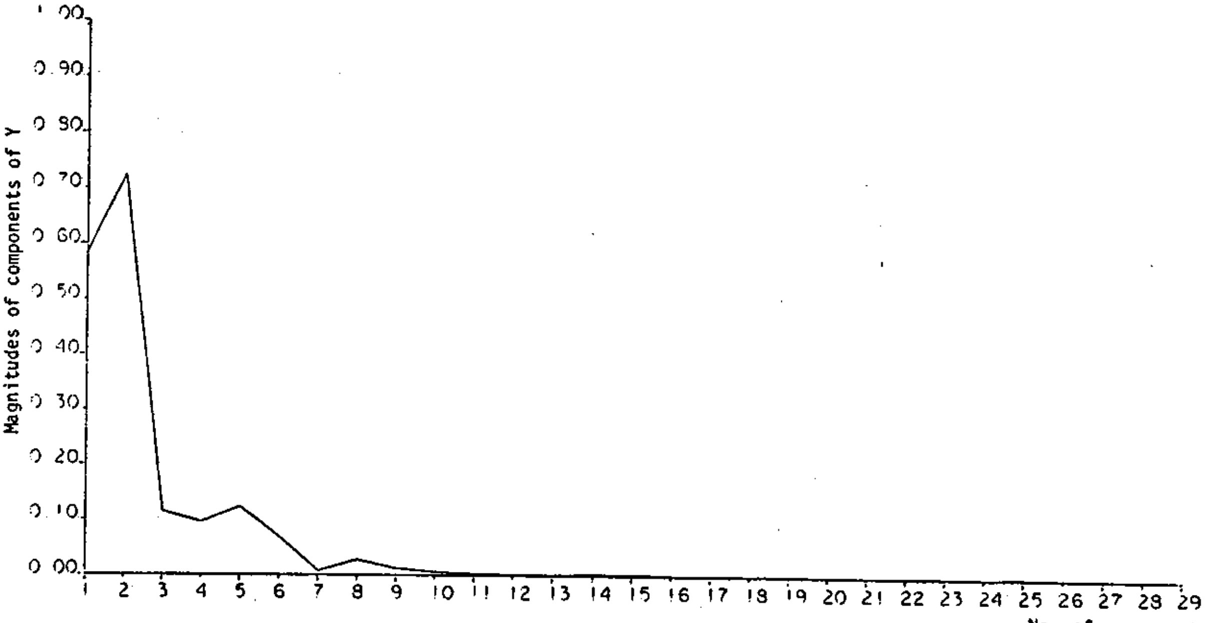


FIGURE 5.5: Magnitudes of components of Y (min. phase) for Channel 6

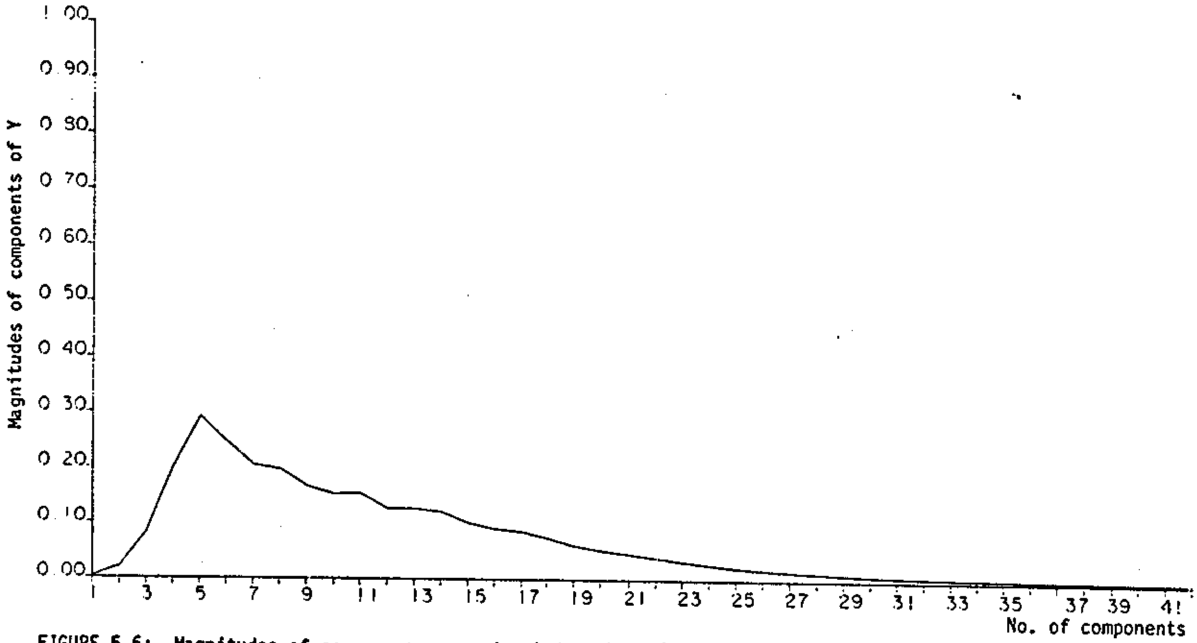


FIGURE 5.6: Magnitudes of components of Y (raw) for Channel 7

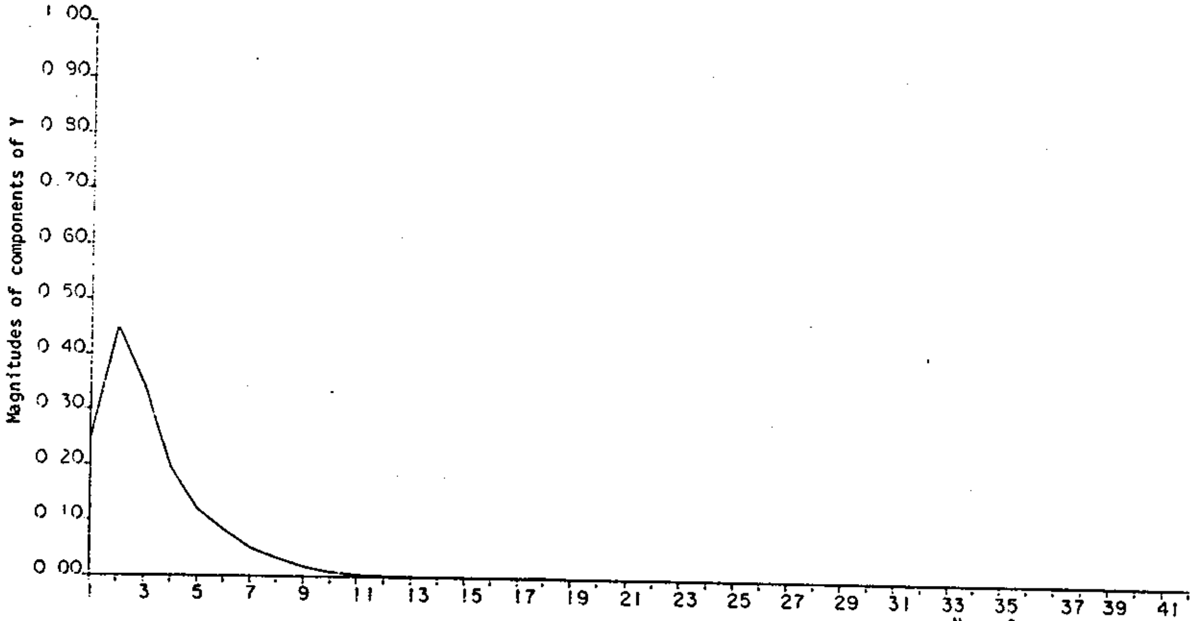


FIGURE 5.7: Magnitudes of components of Y (min. phase) for Channel 7

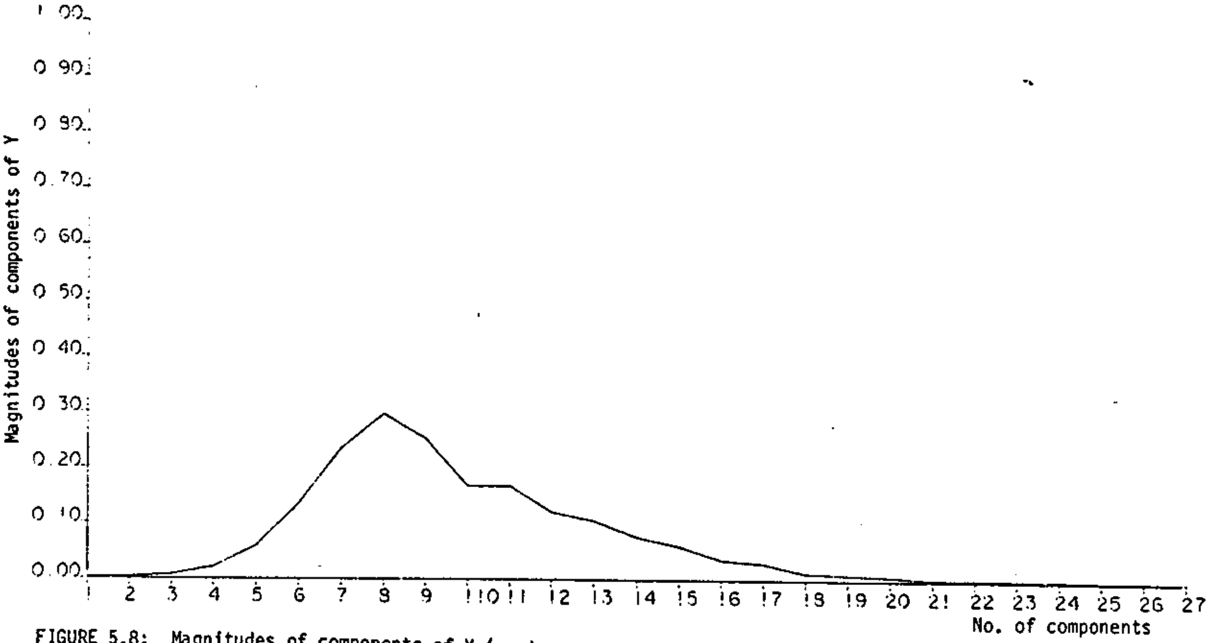


FIGURE 5.8: Magnitudes of components of Y (raw) for Channel 8

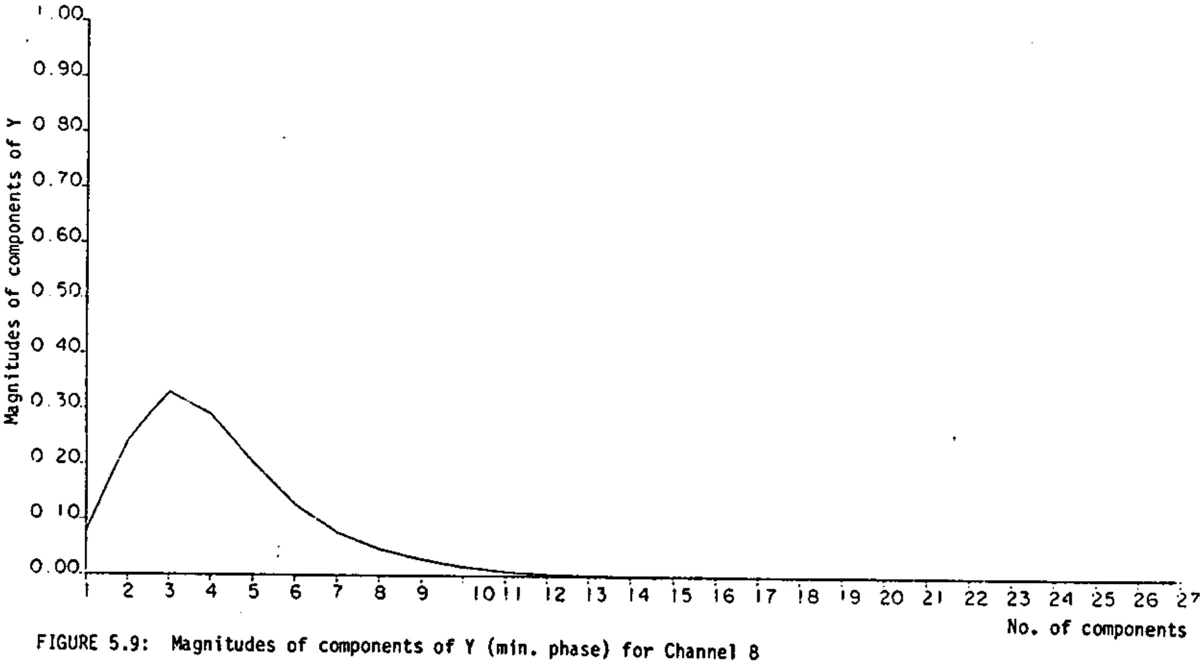


FIGURE 5.9: Magnitudes of components of Y (min. phase) for Channel 8

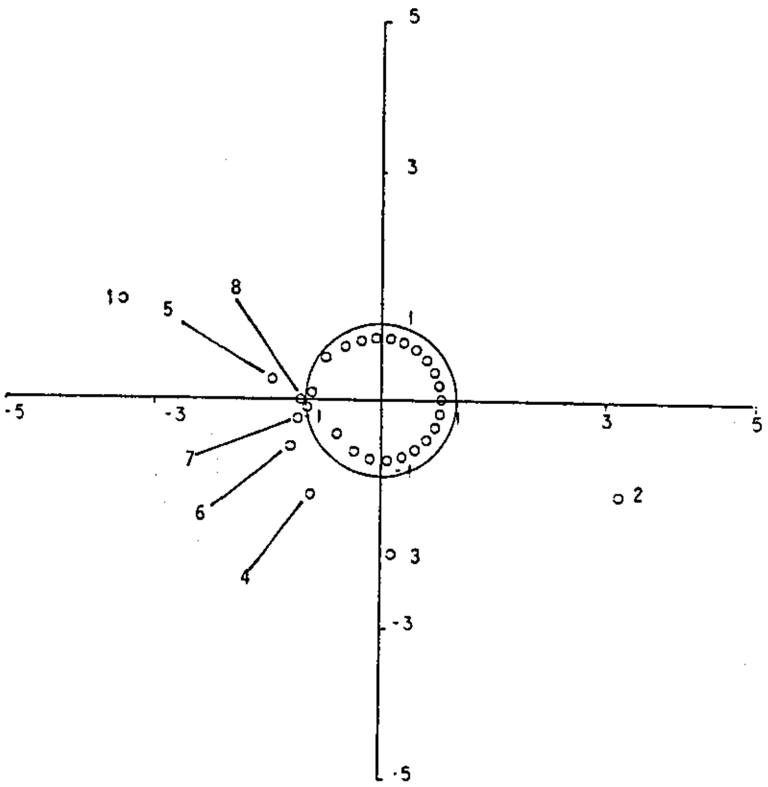


FIGURE 5.10: Roots (zeros) of Y(z) of Channel 5 (sampled at 3200 bauds)

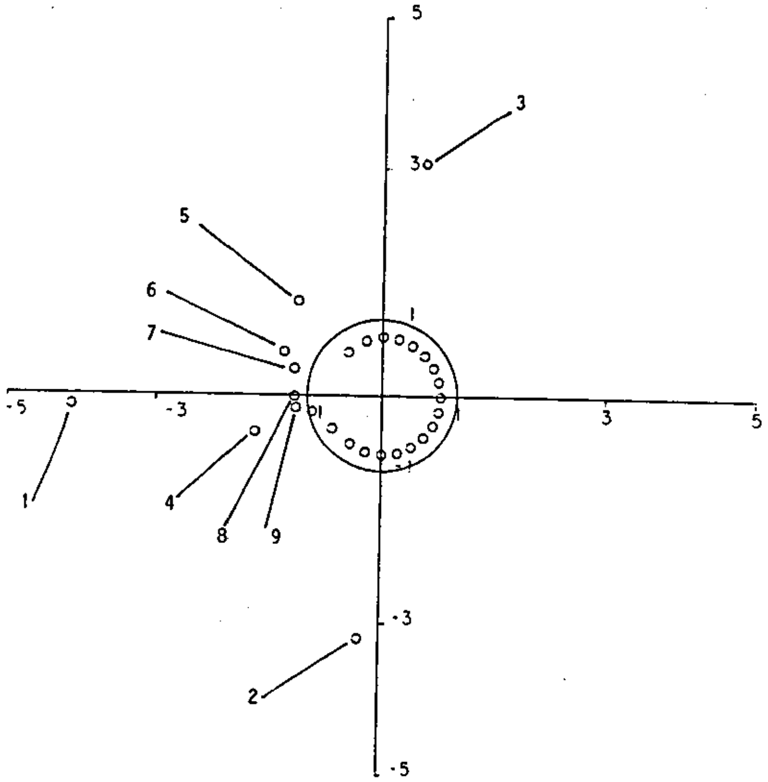


FIGURE 5.11: Roots (Zeros) of Y(z) of Channel 6 (sampled at 3200 bauds)

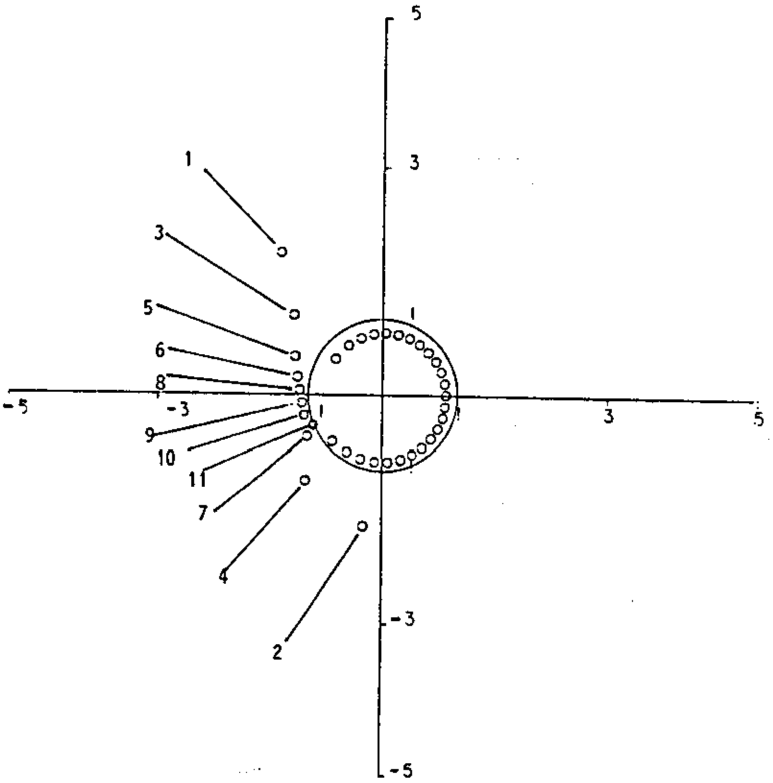


FIGURE 5.12: Roots (Zeros) of Y(z) of Channel 7 (sampled at 3200 bauds)

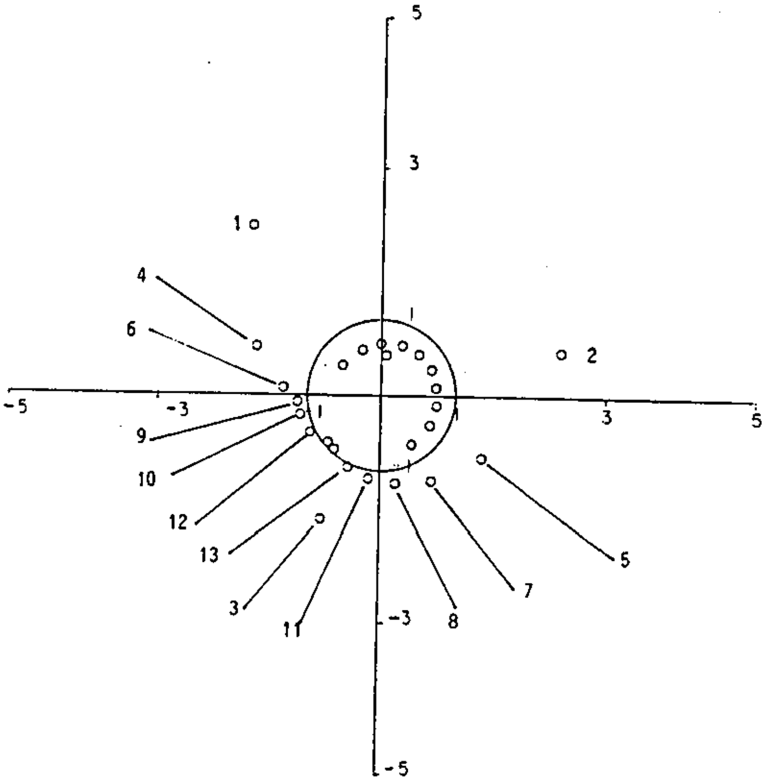


FIGURE 5.13 : Roots (Zeros) of Y(z) of Channel 8 (sampled at 3200 bauds)

TABLE 5.1: The Sampled Impulse-Responses of Channels 5-8

CHANNEL 5		CHANNEL 6		CHANNEL 7		CHANNEL 8	
REAL PART	IMAG. PART	REAL PART	IMAG. PART	REAL PART	IMAG. PART	REAL PART	IMAG. PART
-0.004807	0.001975	0.001966	0.000544	0.001155	-0.001275	0.000139	-0.000657
-0.023279	-0.005598	0.016707	0.003912	0.011734	-0.016035	0.002122	-0.001400
-0.003958	-0.095155	0.067284	0.008166	0.057043	-0.059210	0.005858	0.003295
0.249745	-0.169609	0.221772	0.000227	0.170236	-0.103845	-0.005860	0.020025
0.541549	0.119610	0.502139	-0.056595	0.289941	-0.041411	-0.058710	0.003587
0.297847	0.342398	0.548069	-0.110307	0.203587	0.140088	-0.055260	-0.122651
0.011628	-0.093299	0.046425	0.064073	-0.106333	0.175431	0.165603	-0.166552
0.069883	-0.227266	-0.294318	0.210038	-0.188929	-0.059493	0.264580	0.134973
-0.116629	0.107089	0.006498	-0.101885	0.098031	-0.135501	-0.084000	0.238554
-0.051507	0.055005	0.145968	-0.128412	0.116619	0.098686	-0.122758	-0.114973
0.120645	-0.051534	-0.070121	0.184022	-0.141349	0.062653	0.163238	-0.040380
-0.080646	-0.017534	-0.051347	-0.055243	0.004742	-0.126885	-0.027620	0.118527
-0.001307	0.048080	0.084641	-0.087113	0.115057	0.052773	-0.049954	-0.093587
0.054696	-0.017118	-0.017438	0.127455	-0.106427	0.058291	0.076453	0.006640
-0.042190	-0.025187	-0.048800	-0.070772	0.013733	-0.100340	-0.053434	0.029049
0.007644	0.042773	0.059209	-0.008137	0.067016	0.061483	0.008210	-0.035339
0.016672	-0.034243	-0.029553	0.055785	-0.085331	0.007381	0.007390	0.029327
-o.023740	0.016745	-0.007553	-0.058468	0.049681	-0.055936	-0.008383	-0.010218
0.013562	-0.003006	0.028324	0.035002	0.000236	0.061144	0.010418	0.002475
0.001807	-0.004991	-0.031788	-0.007764	-0.035259	-0.039069	-0.007445	-0.003145
-0.014160	0.007369	0.025419	-0.010526	0.045572	0.006828	0.002625	0.001647
0.017820	-0.008654	-0.014780	0.018476	-0.036460	0.016117	-0.003203	0.001599
-0.014543	0.010301	0.004397	-0.018811	0.019015	-0.027514	0.003120	-0.000777
0.007007	-0.012037	0.002551	0.014924	-0.001935	0.027598	-0.003116	-0.001415
0.001402	0.010854	-0.006499	-0.010226	-0.009838	-0.020545	0.001378	-0.000502
-0.007189	-0.007094	0.007904	0.005818	0.015003	0.011764	-0.000249	0.001776
0.009463	0.000670	-0.007652	-0.002228	-0.015829	-0.003476	0.001308	-0.000529
-0.006936	0.004000	0.006315	-0.000400	0.013445	-0.002168		
0.002667	-0.006328	-0.004811	0.001611	-0.009728	0.005539		
0.001578	0.005983			0.006697	-0.006761		
-0.004066	-0.003360			-0.004013	0.007442		
				0.001671	-0.007064		
				0.000127	0.006670		
				-0.001612	-0.005538		
				0.002745	0.004606		
				-0.003295	-0.003236		
				0.003444	0.002208		
				-0.003439	-0.001632		
				0.003402	0.000765		
				-0.003335	-0.000310		
				0.003124	-0.000448		
				-0.002731	0.000974		

TABLE 5.2: The Minimum-phase Sampled Impulse-responses of Channels 5-8

CHANNEL 5		CHANNEL 6		CHANNEL 7		CHANNEL 8	
REAL PART	IMAG. PART	REAL PART	IMAG. PART	REAL PART	IMAG. PART	REAL PART	IMAG. PART
-0.176657	0.450177	0.285649	-0.507757	0.245356	-0.032166	0.044694	-0.063505
-0.399730	0.542591	0.456534	-0.561966	0.342642	0.289367	0.239493	0.034044
-0.122502	-0.125097	0.012011	0.113587	-0.173279	0.294660	0.034113	0.327801
0.078662	-0.084614	-0.070731	0.064164	-0.131818	-0.144464	-0.288963	0.016585
-0.036314	0.089678	0.074859	-0.097823	0.119692	-0.006269	0.049581	-0.197055
0.008401	-0.069168	-0.030075	0.061777	-0.056124	0.060425	0.092456	0.088525
0.013867	-0.009683	-0.006970	0.001947	-0.006390	-0.049473	-0.077014	0.010535
-0.001927	0.020163	0.016624	-0.022139	0.026545	0.019287	0.027475	-0.039722
-0.007863	-0.006377	-0.005322	0.011138	-0.017021	0.004397	0.007395	0.029157
0.005561	-0.000853	0.004933	-0.004554	0.005391	-0.006058	-0.015325	-0.005654
-0.001619	0.011047	-0.003877	0.000286	-0.000058	0.004033	0.006906	-0.003861
-0.004959	-0.012396	0.002008	0.001193	-0.001746	-0.000675	0.000220	0.004039
0.007910	0.008348	0.001272	0.000674	0.001391	-0.000434	-0.001896	-0.000971
-0.010122	-0.003998	-0.003264	-0.000717	-0.001014	0.000034	0.000519	-0.001212
0.011565	-0.003163	0.002494	0.001114	0.000797	-0.001353	-0.000322	0.000558
-0.007692	0.008869	-0.001090	-0.000537	0.000718	0.000505	-0.000888	-0.000520
0.002620	-0.011546	0.000506	-0.000336	-0.000546	-0.000697	-0.000105	0.000323
0.003273	0.010234	0.000577	0.000712	0.001687	-0.000817	-0.000436	-0.000275
-0.007178	-0.005669	-0.001467	-0.000323	-0.000707	0.000733	0.001069	-0.000818
0.008546	-0.000367	0.001110	-0.000183	0.000674	-0.000876	0.000751	0.000584
-0.006399	0.005453	-0.000319	0.000245	-0.000133	0.000688	-0.000163	0.000613
0.002282	-0.006319	-0.000792	0.000471	-0.000300	-0.000296	0.000043	0.000214
0.002500	0.004701	0.000990	-0.000962	0.000028	0.000014	-0.000067	0.000032
-0.004552	-0.001078	-0.000093	-0.000125	-0.000586	-0.000010	-0.000100	0.000013
0.003247	-0.000924	-0.000603	0.001356	0.000733	-0.000096	-0.000022	-0.000025
-0.000182	0.002550	-0.000244	0.001014	-0.000640	0.000515	0.000008	0.000015
-0.001210	-0.002311	-0.000044	0.000305	0.000255	-0.000280	0.000012	0.000000
0.001877	-0.001446	0.000005	0.000106	0.000095	0.000304		
0.000747	0.000543	0.000001	0.000018	-0.000768	0.000082		
-0.000061	0.000180			0.000619	-0.000283		
-0.000056	0.000007			-0.000306	0.000386		
				-0.000531	0.000121		
				0.000636	-0.000180		
				-0.000120	0.000086		
				-0.000565	0.000086		
				0.000650	-0.000202		
				0.000439	0.000067		
				-0.000338	-0.000018		
				-0.000749	-0.000286		
				-0.000398	-0.000276		
				-0.000157	-0.000109		
				-0.000019	-0.000007		

5.1 Application of the Standard Algorithm to the 19200 bit/s 64-point QAM System

The Standard algorithm is as proposed in Section 4.10 and described in detail in Section 4.4 with parameters c = 1, $d_1 = 10^{-10}$ and (n+1) = 40. It is applied directly to Channels 5-8 without any modifications, and the results are given in Table 5.3 where it shows, for each channel, the roots located by the algorithm, the order in which they are found and the total number of iterations taken. The results show that the algorithm performs excellently for the four channels despite the fact that Y(z) (equation 2.3) has a large number of roots (zeros) outside the unit circle and, in particular, many of them lie very close to it (Figures 5.10-5.13). It can be seen that with the exception of Channel 8, which is a typical worst circuit for the transmission of data at 1200 bit/s in the British public switched telephone network, the other three channels present no real problems to the algorithm. In Channel 8, the algorithm located all but 2 (Roots 4 and 7) out of a total of 13 roots. It is clear, from the results of computer-simulation tests, that the Standard algorithm (Section 4.4) is suitable, at least in most of the cases, for use in the 64-point QAM system operating at 19200 bit/s.

To further improve the performance of the Standard algorithm, so that it can cope with cases such as Channel 8, where Y(z) has a large number of roots lying outside the unit circle, the technique whereby $|\lambda_i|$ (the absolute value of λ_i) is allowed to exceed unity a number of times in the iterative process is again used here. This technique has been used previously in the studies of repeated roots and roots that lie very close to, and outside, the unit circle, and has shown great improvement for the Standard algorithm in its ability to find roots.

Thus, the Standard algorithm operates exactly as before, where at the start of the iterative process,the receiver holds in store the estimate of the sampled impulse-response of the channel, Y(equation 2.3), and the set of five possible starting-points for λ_0 (Figure 4.10). The sequence Y, starting with its last component y_g, is fed through the one-tap feedback

TABLE 5.3: Performance of the Standard Algorithm with 5 Starting-Points

Channel 5 (8 roots) 117 iterations	Channel 6 (9 roots) 102 iterations	Channel 7 (11 roots) 137 iterations	Channel 8 (13 roots) 184 iterations
Root 3	Root 1	Root 1	Root 3
Root 1	Root 6	Root 3	Root 10
Root 6	Root 4	Root 5	Root 6
Root 8	Root 7	Root 9	Root 1
Root 5	Root 2	Root 4	Root 11
Root 2	Root 3	Root 10	Root 8
Root 7	Root 8	Root 7	Root 2
Root 4	Root 9	Root 11	Root 9
	Root 5	Root 6	Root 5
		Root 8	Root 13
		Root 2	Root 12
			~~Root 4~~
			~~Root 7~~

filter (Figure 4.9) with its tap set to the first of the five possible starting-points ($\lambda_0 = 0$) to give a sequence $e_{i,g}\ e_{i,g-1}\ \dots\ e_{i,o}$. An improved estimate of β_1 (the negative reciprocal of a root (zero) of Y(z) lying outside the unit circle) is given by

$$\lambda_{i+1} = \lambda_i + c\,\frac{e_{i,o}}{\varepsilon_i} \qquad (5.1)$$

where

$$\varepsilon_i = e_{i,1} - e_{i,2}\lambda_i + e_{i,3}\lambda_i^2 - \dots + e_{i,g}(-\lambda_i)^{g-1} \qquad (5.2)$$

and c = 1.0.

The iterative process is repeated until

$$|e_{i,o}/\varepsilon_i|^2 < 10^{-10} \qquad (5.3)$$

which is taken to indicate that the algorithm has converged to a required root and the value λ_k, where k is a positive real number, is taken to be that of β_1 (equation 2.8). At every step of the iterative process $|\lambda_i|$ is checked. Whenever $|\lambda_i|>1$ or when i = 40, the iterative process is taken to have diverged and so, in the latter case, is terminated and restarted with λ_0 set to the next of its five possible values (Figure 4.10). The action taken for $|\lambda_i|>1$ has been modified so that the iterative process continues until either $|\lambda_i|>1$ or i = 40 occurs again. In the latter case, the same restart procedure is used. However, the repeated occurrence of $|\lambda_i|>1$ indicates the divergence of the algorithm i.e. the result of tracking a root inside of the unit circle, rather than as a result of crossing into the unit circle by too large an increment in equation 5.1 while tracking a root outside the unit circle (which sometimes occurs when the algorithm converges to roots lying very close to the unit circle). Therefore, when $|\lambda_i|>1$ occurs b times, where b is a positive real number, the iterative process is terminated and then restarted with λ_0 set to the next of its five possible starting-points. This, of course, assumes that the five possible starting-points have not been exhausted.

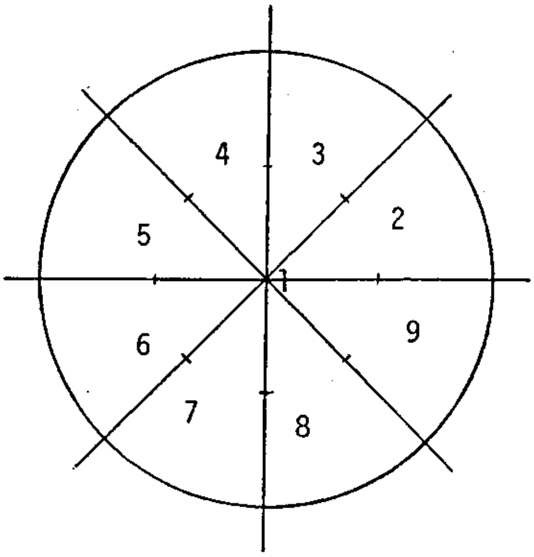
Thus, the Standard algorithm has been modified such that the value of $|\lambda_i|$ is allowed to exceed unity b time and for each occasion, the process is restarted again. On the (b+1)th occurrence of $|\lambda_i|>1$, the process is assumed to have found all the roots and so is terminated. Otherwise all the operations are as described before (Section 4.4). Tests were carried out on Channel 8 only, and different values of b were used. The results are given in Table 5.4, which shows that the modification has not been successful as far as Channel 8 is concerned. Note that although 11 out of 13 roots are found in each of the cases, the roots and the order in

TABLE 5.4: Results of Computer-Simulation Tests for Channel 8

Channel 8 (13 Roots)			
b=1	b=2	b=3	b=4
243 iterations	243 iterations	243 iterations	266 iterations
Root 3	Root 3	Root 3	Root 3
Root 10	Root 10	Root 10	Root 10
Root 6	Root 9	Root 9	Root 9
Root 1	Root 4	Root 4	Root 4
Root 11	Root 1	Root 1	Root 1
Root 8	Root 12	Root 12	Root 12
Root 2	Root 7	Root 7	Root 7
Root 9	Root 2	Root 13	Root 13
Root 5	Root 13	Root 8	Root 8
Root 13	Root 11	Root 4	Root 5
Root 12	Root 5	Root 2	Root 2
~~Root 4~~	~~Root 8~~	~~Root 5~~	~~Root 6~~
~~Root 7~~	~~Root 6~~	~~Root 6~~	~~Root 11~~

which they are found are different. Furthermore, the total number of iterations taken has now increased by more than 30%. The failure of the algorithm to find all the roots seems to have been caused by the great influence of the two roots lying just outside the unit circle in the third quadrant of the z-plane (Figure 5.13), whose effect is further enhanced when Roots 9-13 are replaced by their complex conjugate-reciprocals, which are also in the third quadrant.

In Section 4.4 the introduction of a set of possible starting-points for the algorithm proved to be the key to the success of locating all the roots outside the unit circle. It is apparent that the algorithm has a better chance of locating one of the roots, $\{\frac{-1}{\beta_h}\}$, if the initial value, λ_o, is set near to the value of β_h. The idea is now expanded such that the set of different values of λ_o is increased to nine, instead of the five which has been used so far. The values of λ_o are chosen to have an absolute value of 0.5 and evenly spread on the circle of radius 0.5, centredat the origin of the z-plane, as shown in Figure 5.14.



9 starting-points for λ_o

1. λ_o =	0.0	+ j0.0
2. λ_o =	0.5	+ j0.0
3. λ_o =	0.0	- j0.5
4. λ_o =	0.0	+ j0.5
5. λ_o =	-0.5	+ j0.0
6. λ_o =	0.354	- j0.354
7. λ_o =	0.354	+ j0.354
8. λ_o =	-0.354	+ j0.354
9. λ_o =	-0.354	- j0.354

FIGURE 5.14: The Set of 9 Starting-Points for λ_o

The order in which the starting-points are chosen, is as shown in the diagram, starting with No. 1. It can be seen that the first five starting-points are the same as before, and therefore, the new set will not alter either the order in which the roots are found or the number of iterations required for each root, when it is applied to the previous cases of the 16-point QAM system transmitted at 9600 bit/s. This is important, because in the implementation of the system, it is desirable to have one general algorithm where the extra starting-points can be switched in when dealing with the higher data transmission-rates or operating over very poor channels. There will, however, be a slight increase in the total number of iterations because the iterative root-finding process has to search through nine starting-points, instead of five, before stopping.

The Standard algorithm operates exactly as described before (Section 4.4) in every detail except for the design of the new set of starting-points. Note also that when $|\lambda_i|>1$ occurs, the algorithm is terminated immediately and restarted with λ_0 set to the next of the nine possible starting-points (Figure 5.14). Results of computer-simulation tests have shown that the modification enables the Standard algorithm to find all the roots of Y(z) outside the unit circle in the z-plane, for all the four channels, 5, 6, 7 and 8, with only a small increase in the total number of iterations required. The number of iterations required for the channels are given in Table 5.5, and the complete set of results for ψ_1 and ψ_2 (equations 4.96-4.97) for different values of c (equation 4.114) and the number of filter taps are given in Tables 5.6-5.9. Again, the accuracy of the root-finding process is limited only by the machine accuracy and the threshold (d_1) used in equation 4.115. For example, the value of $10 \log_{10}|\beta-\lambda_i|^2$ is about -100 dB when the threshold = 10^{-10} is used in equation 4.115 and it is reduced to about -200 dB when the threshold is changed to 10^{-20}. From the results of ψ_1 and ψ_2, it can be seen that the adaptive linear filter will probably require fifty taps instead of forty as previously suggested for the 16-point QAM system transmitted at 9600 bit/s.

5.2 Application of the Modified Algorithm to the 19200 bit/s 64-Point QAM System

This section is concerned with the possible application of the Modified algorithm as described in Section 4.6 to Channels 5-8, the baseband channels for the 64-point QAM system operating at 19200 bit/s. Their sampled impulse-response are given in Tables 5.1-5.2 and the roots (zeros) of Y(z) (equation 2.4) are shown in Figures 5.10-5.13. The Modified algorithm is exactly as described in Section 4.6 except for the set of starting points for which there are nine as shown in Figure 5.14. This set of starting-points has been proved to be very effective in the Standard algorithm when dealing with Channels 5-8.

The algorithm is divided into two parts, where the first part is designed to guide the estimate λ_i closer to one of the required $\{\beta_h\}$ (negative reciprocals of the roots (zeros) of Y(z) outside the unit circle) so that the algorithm has a better chance of locating a root. At the beginning of the iterative process, the receiver holds in store the channel estimate Y (equation 2.3) and the set of 9 possible starting-points (Figure 5.14). A one-tap feedback filter (Figure 4.18) is formed with its tap set to the first of the 9 possible values of λ_o (=0) and the sequence Y is reversed in order and fed through the feedback filter, starting with the component y_g and finishing with y_o, to give an output sequence of $\{e_{i,h}\}$ (equation 4.106). A new estimate is obtained by

$$\lambda_{i+1} = \lambda_i + c\,\frac{e_{i,o}}{\varepsilon_i} \qquad (5.4)$$

where $\varepsilon_i = e_{i,1} - e_{i,2}\lambda_i + e_{i,3}\lambda_i^2 - \ldots + e_{i,g}(-\lambda_i)^{g-1}$ (5.5)

and c = 1.

For the first ℓ iterations, where ℓ is usually between 3 and 6, one extra computation step is performed, to give

$$\lambda_{i+1}' = (1 - \frac{1}{K_i})\,\lambda_i + \frac{1}{K_i}\,\lambda_{i+1} \qquad (5.6)$$

where $K_i = 2,3,\ldots,\ell+1$ and of course $i = 1,2,\ldots,\ell$.

The value λ_{i+1}' is then taken to be the final value of λ_{i+1} and is then used to set the filter tap for the next iteration. During the first ℓ iterations, the value $|e_{i,o}/\varepsilon_i|^2$ is checked, and whenever

$$|e_{i,o}/\varepsilon_i|^2 < d_2 \qquad (5.7)$$

where d_2 is a small positive constant, the algorithm is switched to the second part of the operation which involves only equation 5.4 in the root-finding process (equivalent to the Standard algorithm). Whenever

$$|e_{i,o}/\varepsilon_i|^2 < d_1 \qquad (5.8)$$

where d_1 is a small positive constant, the root-finding process is taken to have converged to one of the required roots. As before, the value $|\lambda_i|$ is checked at every iteration and whenever $|\lambda_i|>1$ or $i = 40$, the process is taken to have diverged and so is terminated. The iteration process is then restarted with the λ_0 set to the next of the nine possible starting-points, and when all the nine possible λ_0 have been exhausted the process is assumed to have found all the required roots.

Results of computer-simulation tests for the four channels are given in Tables 5.10-5.21, which show the variation of ψ_1 and ψ_2 (equations 4.96-4.97) for different numbers of taps (= 20, 30, 40, 50, 60, 70, 80, 90 and 100) in the linear filter. These are obtained using ℓ = 4, 5 and 6 together with different combinations of threshold values d_1 (= 10^{-7}, 10^{-10}, 10^{-15} and 10^{-20}) and d_2 (= 0.05, 0.075 and 0.1). The results in the tables also give the number of roots located by the algorithm and the total number of iterations taken in each of the cases. It can be seen that for bad channels such as 7 and 8, longer averaging periods (K = 6) are necessary to find all the required roots. The best case is $d_1 = 10^{-7}$, $c = 1.0$, $d_2 = 0.1$ and $\ell = 6$ (although $\ell = 5$ might also be suitable), and this configuration enables all the roots (zeros) of Y(z) outside the unit circle for the four channels to be found. Judging from the results of ψ_1 and ψ_2 in Tables 5.10-5.21, a linear filter of 50 taps seems sufficient for this application.

TABLE 5.5: Performance of the Standard Algorithm with 9 Starting-Points

Channel 5 (8 Roots) 134 iterations	Channel 6 (9 Roots) 113 iterations	Channel 7 (11 Roots) 148 iterations	Channel 8 (13 Roots) 255 iterations
Root 3	Root 1	Root 1	Root 3
Root 1	Root 6	Root 3	Root 10
Root 6	Root 4	Root 5	Root 6
Root 8	Root 7	Root 9	Root 1
Root 5	Root 2	Root 4	Root 11
Root 2	Root 3	Root 10	Root 8
Root 7	Root 8	Root 7	Root 2
Root 4	Root 9	Root 11	Root 9
	Root 5	Root 6	Root 5
		Root 8	Root 13
		Root 2	Root 12
			Root 4
			Root 7

		$d_1 = 10^{-7}$			$d_1 = 10^{-10}$			$d_1 = 10^{-15}$			$d_1 = 10^{-20}$		
c =		0.5	0.7	1.0	0.5	0.7	1.0	0.5	0.7	1.0	0.5	0.7	1.0
No of Iterations		229	161	129	265	182	134	332	218	140	398	257	143
No of taps in filter	20	-48.90 -31.83	-49.90 -31.91	-48.85 -31.76	-48.88 -31.80	-48.88 -31.80	-48.88 -31.80	-48.88 -31.80	-48.88 -31.80	-48.88 -31.80	-48.88 -31.80	-48.88 -31.80	-48.88 -31.80
	30	-68.41 -26.96	-68.40 -26.95	-68.42 -26.97	-68.41 -26.96	-68.41 -26.96	-68.41 -26.96	-68.41 -26.96	-68.41 -26.96	-68.41 -26.96	-68.41 -26.96	-68.41 -26.96	-68.41 -26.96
	40	-257.01 -37.03	-255.80 -36.94	-255.90 -37.87	-255.30 -37.07	-256.37 -37.07	-256.88 -37.08	-257.24 -37.08	-256.74 -37.08	-286.28 -37.08	-255.02 -37.08	-256.79 -37.08	-255.88 -37.08
	50	-257.01 -44.00	-255.80 -44.03	-255.90 -43.88	-255.30 -43.93	-256.37 -43.93	-256.88 -43.93	-257.24 -43.93	-256.74 -43.93	-256.28 -43.93	-255.02 -43.93	-256.79 -43.93	-255.88 -43.93
	60	-257.01 -47.96	-255.80 -47.93	-255.90 -47.88	-255.30 -47.91	-256.37 -47.91	-256.88 -47.91	-257.24 -47.91	-256.74 -47.91	-256.28 -47.91	-255.02 -47.91	-256.79 -47.91	-255.88 -47.91
	70	-257.01 -55.20	-255.80 -55.13	-255.90 -55.10	-255.30 -55.18	-256.37 -55.18	-256.88 -55.18	-257.24 -55.18	-256.74 -55.18	-256.28 -55.18	-255.02 -55.18	-256.79 -55.18	-255.88 -55.18
	80	-257.01 -60.89	-255.80 -60.82	-255.90 -60.76	-255.30 -60.86	-256.37 -60.87	-256.88 -60.86	-257.24 -60.86	-256.74 -60.86	-256.28 -60.86	-255.02 -60.86	-256.79 -60.86	-255.88 -60.86
	90	-257.81 -66.42	-255.80 -66.17	-255.90 -66.44	-255.30 -66.58	-256.37 -66.59	-258.88 -66.58	-257.24 -66.58	-256.74 -66.58	-256.28 -66.58	-255.02 -66.58	-256.79 -66.58	-255.88 -66.58
	100	-257.81 -71.91	-255.80 -71.19	-255.90 -72.51	-255.30 -72.81	-256.37 -72.82	-256.88 -72.82	-257.24 -72.81	-256.74 -72.81	-256.28 -72.81	-255.02 -72.81	-256.79 -72.81	-255.88 -72.81

TABLE 5.6: ψ_1, ψ_2 for Channel 5

		$d_1 = 10^{-7}$			$d_1 = 10^{-10}$			$d_1 = 10^{-15}$			$d_1 = 10^{-20}$		
c =		0.5	0.7	1.0	0.5	0.7	1.0	0.5	0.7	1.0	0.5	0.7	1.0
No of Iterations		210	143	107	248	170	113	324	211	117	398	255	122
No of taps in filter	20	-38.38 -23.09	-38.38 -23.11	-38.38 -23.10	-38.38 -23.10	-38.38 -23.10	-38.38 -23.10	-38.38 -23.10	-38.38 -23.10	-38.38 -23.10	-38.38 -23.10	-38.38 -23.10	-38.38 -23.10
	30	-255.58 -23.36	-255.75 -23.36	-255.72 -23.36	-254.37 -23.36	-255.37 -23.36	-254.55 -23.36	-254.97 -23.36	-254.21 -23.36	-255.95 -23.36	-254.66 -23.36	-254.89 -23.36	-255.01 -23.36
	40	-255.59 -37.50	-255.75 -37.55	-255.72 -37.51	-254.37 -37.51	-255.37 -37.51	-254.55 -37.51	-254.97 -37.51	-254.21 -37.51	-255.95 -37.51	-254.66 -37.51	-254.89 -37.51	-255.01 -37.51
	50	-255.59 -49.97	-255.75 -50.04	-255.72 -49.96	-254.37 -49.98	-255.37 -49.98	-254.55 -49.98	-254.97 -49.98	-254.21 -49.98	-255.95 -49.98	-254.66 -49.98	-254.89 -49.98	-255.01 -49.98
	60	-255.59 -68.80	-255.75 -69.49	-255.72 -68.93	-254.37 -69.86	-255.37 -69.97	-254.55 -69.95	-254.97 -69.96	-252.21 -69.96	-255.95 -69.96	-254.66 -69.96	-254.89 -69.96	-255.01 -69.96
	70	-255.59 -74.79	-255.75 -78.28	-255.72 -75.17	-254.37 -87.51	-255.37 -87.55	-254.55 -87.60	-254.97 -87.58	-254.21 -87.58	-255.95 -87.58	-254.66 -87.58	-254.89 -87.58	-255.01 -87.58
	80	-255.59 -74.92	-255.75 -78.56	-255.72 -75.32	-254.37 -90.87	-255.37 -90.98	-254.55 -91.84	-254.97 -91.03	-254.21 -91.03	-255.95 -91.03	-254.66 -91.03	-254.89 -91.03	-255.01 -91.03
	90	-255.69 -75.02	-255.75 -78.78	-255.72 -75.43	-254.37 -99.89	-255.37 -100.85	-254.55 -101.45	-254.97 -101.47	-254.21 -101.47	-255.95 -101.47	-254.66 -101.47	-254.89 -101.47	-255.01 -101.47
	100	-255.59 -75.03	-255.75 -78.81	-255.72 -75.44	-254.37 -104.74	-255.37 -108.70	-254.55 -115.81	-254.97 -116.22	-254.21 -116.22	-255.95 -116.22	-254.66 -116.22	-254.89 -116.22	-255.01 -116.22

TABLE 5.7: ψ_1, ψ_2 for Channel 6

		$d_1 = 10^{-7}$			$d_1 = 10^{-10}$			$d_1 = 10^{-15}$			$d_1 = 10^{-20}$		
c =		0.5	0.7	1.0	0.5	0.7	1.0	0.5	0.7	1.0	0.5	0.7	1.0
No of Iterations		260	183	140	317	212	148	412	255	154	500	304	160
No of taps in filter	20	-25.29 -20.20	-31.87 -24.49	-29.28 -20.19	-29.29 -20.20	-31.86 -24.39	-29.29 -20.20	-29.29 -20.20	-31.86 -24.39	-29.29 -20.20	-29.29 -20.20	-31.86 -24.39	-29.29 -20.20
	30	-51.57 -28.72	-47.02 -30.48	-51.45 -28.72	-51.56 -28.72	-47.02 -30.49	-51.56 -28.72	-51.56 -28.72	-47.02 -30.49	-51.56 -28.72	-51.56 -28.72	-47.02 -30.49	-51.56 -28.72
	40	-257.97 -26.09	-257.23 -29.75	-257.32 -26.08	-258.75 -26.08	-257.32 -29.70	-257.45 -26.08	-258.44 -26.08	-256.89 -29.75	-257.44 -26.08	-257.90 -26.08	-257.94 -29.75	-257.36 -26.08
	50	-259.97 -37.83	-257.23 -41.55	-257.32 -37.87	-258.75 -37.87	-257.32 -41.52	-257.45 -37.88	-258.44 -37.88	-256.89 -41.52	-257.44 -37.88	-257.99 -36.88	-257.94 -41.52	-257.36 -37.88
	60	-257.97 -68.42	-257.23 -67.57	-257.32 -69.06	-258.75 -69.02	-257.32 -67.08	-257.45 -69.03	-258.44 -69.04	-256.89 -67.07	-257.44 -69.04	-257.90 -69.04	-257.94 -67.07	-257.36 -69.04
	70	-257.97 -64.54	-257.23 -60.18	-237.32 -64.48	-258.75 -64.27	-257.32 -60.35	-257.45 -64.26	-258.44 -64.26	-256.89 -60.35	-257.44 -64.26	-257.90 -64.26	-257.94 -60.35	-257.36 -64.26
	80	-257.97 -58.93	-257.23 -60.38	-257.32 -58.84	-258.75 -58.86	-257.32 -60.46	-257.45 -58.86	-258.44 -58.86	-256.89 -60.46	-257.44 -58.86	-257.90 -58.86	-257.94 -60.46	-257.36 -58.86
	90	-257.97 -67.48	-257.23 -70.69	-257.32 -67.64	-248.75 -67.67	-257.32 -70.76	-257.45 -67.67	-258.44 -67.67	-256.89 -70.76	-257.44 -67.67	-257.90 -67.67	-257.94 -70.76	-257.36 -67.67
	100	-257.97 -79.84	-257.23 -85.53	-257.32 -85.34	-258.76 -85.36	-237.32 -88.64	-257.45 -85.39	-258.44 -85.39	-256.89 -88.66	-257.44 -85.39	-257.90 -85.39	-257.94 -88.66	-257.36 -85.39

TABLE 5.8: ψ_1, ψ_2 for Channel 7

		$d_1 = 10^{-7}$			$d_1 = 10^{-10}$			$d_1 = 10^{-15}$			$d_1 = 10^{-20}$		
c =		0.5	0.7	1.0	0.5	0.7	1.0	0.5	0.7	1.0	0.5	0.7	1.0
No of Iterations		385	283	246	456	322	255	562	385	265	668	444	269
No of taps in filter	20	-54.92 -24.35	-54.91 -24.33	-54.93 -24.34	-54.92 -24.34	-54.92 -24.34	-54.92 -24.34	-54.92 -24.34	-54.92 -24.34	-54.92 -24.34	-54.92 -24.34	-54.92 -24.34	-54.92 -24.34
	30	-263.16 -25.94	-263.20 -25.96	-262.93 -25.97	-262.50 -25.96	-265.12 -25.97	-263.75 -25.97	-264.67 -25.97	-262.01 -25.97	-263.32 -25.97	-264.27 -25.97	-263.40 -25.97	-263.33 -25.97
	40	-263.16 -44.54	-263.20 -44.47	-262.83 -44.51	-262.50 -44.49	-265.12 -44.49	-263.75 -44.49	-264.67 -44.49	-262.01 -44.49	-263.32 -44.49	-264.27 -44.49	-263.40 -44.49	-263.33 -44.49
	50	-263.16 -45.01	-263.20 -45.21	-262.83 -45.07	-262.50 -45.06	-265.12 -45.06	-263.75 -45.06	-264.67 -45.06	-262.01 -45.06	-263.32 -45.06	-264.27 -45.06	-263.40 -45.06	-263.33 -45.06
	60	-263.16 -56.82	-263.20 -57.48	-262.83 -57.35	-262.50 -57.36	-265.12 -57.38	-263.75 -57.38	-264.67 -57.67	-262.01 -57.37	-263.32 -57.37	-264.27 -57.37	-263.40 -57.37	-263.33 -57.37
	70	-263.16 -54.17	-263.20 -54.50	-262.63 -54.26	-262.50 -54.30	-265.12 -54.31	-263.75 -54.30	-264.67 -54.30	-262.01 -54.30	-263.32 -54.30	-264.27 -54.30	-263.40 -54.30	-263.33 -54.30
	80	-263.16 -58.71	-263.20 -59.25	-262.83 -58.93	-262.50 -58.92	-265.12 -58.93	-263.75 -58.92	-264.67 -58.92	-262.01 -58.92	-263.32 -58.92	-264.27 -58.92	-263.40 -58.92	-263.33 -58.92
	90	-263.16 -59.48	-263.20 -59.93	-262.83 -59.61	-262.50 -59.60	-265.12 -59.61	-263.75 -59.60	-264.67 -59.61	-262.01 -59.61	-263.32 -59.61	-264.27 -59.61	-263.40 -59.61	-263.33 -59.61
	100	-263.16 -62.35	-263.20 -62.84	-262.83 -62.49	-262.50 -62.47	-265.12 -62.48	-263.75 -62.47	-264.67 -62.47	-262.01 -62.47	-263.32 -62.47	-264.27 -62.47	-263.40 -62.47	-263.33 -62.47

TABLE 5.9: ψ_1, ψ_2 for Channel 8

		$d_1 = 10^{-7}$			$d_1 = 10^{-10}$			$d_1 = 10^{-15}$			$d_1 = 10^{-20}$		
d .		0.050	0.075	0.100	0.050	0.075	0.100	0.050	0.075	0.100	0.050	0.075	0.100
No of Iterations		149	147	140	154	149	143	160	155	149	164	161	155
No of Located Roots		8	8	8	8	8	8	8	8	8	8	8	8
No of taps in filter	20	-48.88 -31.80	-48.88 -13.80	-48.88 -31.80	-48.88 -31.80	-48.88 -31.80	-48.88 -31.80	-48.88 -31.80	-48.88 -31.80	-48.88 -31.80	-48.88 -31.80	-48.88 -31.80	-48.88 -31.80
	30	-68.41 -26.96	-68.41 -26.96	-68.41 -26.96	-68.41 -26.96	-68.41 -26.96	-68.41 -26.96	-68.41 -26.96	-68.41 -26.96	-68.41 -26.96	-68.41 -26.96	-68.41 -26.96	-68.41 -26.96
	40	-255.45 -37.08	-254.15 -37.08	-256.05 -37.08	-256.08 -37.08	-254.84 -37.08	-255.26 -37.08	-256.89 -37.08	-255.34 -37.08	-255.69 -37.08	-255.96 -37.08	-255.36 -37.08	-254.65 -37.08
	50	-255.45 -43.93	-254.15 -43.93	-256.05 -43.93	-256.08 -43.93	-254.84 -43.93	-255.26 -43.93	-256.89 -43.93	-255.34 -43.93	-255.09 -43.93	-255.86 -43.93	-255.36 -43.93	-254.65 -43.93
	60	-255.45 -47.91	-254.15 -47.91	-256.05 -47.91	-256.08 -47.91	-254.84 -47.91	-255.26 -47.91	-256.89 -47.91	-255.34 -47.91	-255.09 -47.91	-255.86 -47.91	-255.36 -47.91	-254.65 -47.91
	70	-255.46 -55.18	-254.15 -55.18	-256.05 -55.18	-256.08 -55.18	-254.84 -55.18	-255.26 -55.18	-256.89 -55.18	-255.34 -55.18	-255.09 -55.18	-255.86 -55.18	-255.36 -55.18	-254.65 -55.18
	80	-255.46 -60.86	-254.15 -60.86	-256.05 -60.86	-256.08 -60.86	-254.84 -60.86	-255.26 -60.86	-256.89 -60.86	-255.34 -60.86	-255.09 -60.86	-255.86 -60.86	-255.36 -60.86	-254.65 -60.86
	90	-255.46 -66.58	-254.15 -66.58	-256.05 -66.58	-256.08 -66.58	-254.84 -66.58	-255.26 -66.58	-256.89 -66.58	-255.34 -66.58	-255.09 -66.58	-255.86 -66.58	-255.36 -66.58	-254.65 -66.58
	100	-255.46 -72.81	-254.15 -72.81	-256.05 -72.81	-256.08 -72.81	-254.84 -72.81	-255.26 -72.81	-256.89 -72.81	-255.34 -72.81	-255.09 -72.81	-255.86 -72.81	-255.36 -72.81	-254.65 -72.81

TABLE 5.10: ψ_1, ψ_2 for Channel 5 (8 roots), ℓ = 4

		$d_1 = 10^{-7}$			$d_1 = 10^{-10}$			$d_1 = 10^{-15}$			$d_1 = 10^{-20}$		
d_2 =		0.050	0.075	0.100	0.050	0.075	0.100	0.050	0.075	0.100	0.050	0.075	0.100
No of Iterations		137	133	127	143	140	134	148	146	139	152	150	144
No of Located Roots		9	9	9	9	9	9	9	9	9	9	9	9
No of taps in filter	20	-38.38 -23.10	-38.38 -23.10	-38.38 -23.10	-38.38 -23.10	-38.38 -23.10	-38.38 -23.10	-38.38 -23.10	-38.38 -23.10	-38.38 -23.10	-38.38 -23.10	-38.38 -23.10	-38.38 -23.10
	30	-254.35 -23.36	-255.76 -23.36	-254.41 -23.36	-253.58 -23.36	-255.75 -23.36	-253.65 -23.36	-255.14 -23.36	-254.09 -23.36	-255.32 -23.36	-254.45 -23.36	-254.81 -23.36	-253.61 -23.36
	40	-254.35 -37.51	-255.76 -37.51	-254.91 -37.51	-253.58 -37.51	-255.75 -37.51	-253.65 -37.51	-255.14 -37.51	-254.09 -37.51	-255.32 -37.51	-254.45 -37.51	-254.81 -37.51	-253.61 -37.51
	50	-254.35 -49.98	-255.76 -49.98	-254.91 -49.98	-253.58 -49.98	-255.75 -49.98	-253.65 -49.98	-255.14 -49.98	-254.09 -49.98	-255.32 -49.98	-254.45 -49.98	-254.81 -49.98	-253.61 -49.98
	60	-254.35 -69.96	-255.76 -69.96	-254.91 -69.96	-253.58 -69.96	-255.75 -69.96	-253.65 -69.96	-255.14 -69.96	-254.09 -69.96	-255.32 -69.96	-254.45 -69.96	-254.81 -69.96	-253.61 -69.96
	70	-254.35 -87.58	-255.76 -87.58	-254.91 -87.58	-253.58 -87.58	-255.75 -87.58	-253.65 -87.58	-255.14 -87.58	-254.09 -87.58	-255.32 -87.58	-254.45 -87.58	-254.81 -87.58	-253.61 -87.58
	80	-254.35 -91.03	-255.76 -91.03	-254.91 -91.03	-253.58 -91.03	-255.75 -91.03	-253.65 -91.03	-255.14 -91.03	-254.09 -91.03	-255.32 -91.03	-254.45 -91.03	-254.81 -91.03	-253.61 -91.03
	90	-254.35 -101.47	-255.76 -101.47	-254.51 -101.47	-253.58 -101.47	-255.75 -101.47	-253.65 -101.47	-255.14 -101.47	-254.09 -101.47	-255.32 -101.47	-254.45 -101.47	-254.81 -101.47	-253.61 -101.47
	100	-254.35 -116.21	-255.76 -116.20	-254.91 -116.21	-253.58 -116.22	-255.75 -116.22	-253.65 -116.22	-255.14 -116.22	-254.09 -116.22	-255.32 -116.22	-254.45 -116.22	-254.81 -116.22	-253.61 -116.22

TABLE 5.11: ψ_1, ψ_2 for Channel 6 (9 roots), ℓ = 4

		$d_1 = 10^{-7}$			$d_1 = 10^{-10}$			$d_1 = 10^{-15}$			$d_1 = 10^{-20}$		
d_2 =		0.050	0.075	0.100	0.050	0.075	0.100	0.050	0.075	0.100	0.050	0.075	0.100
No of Iterations		185	159	163	192	169	172	197	173	177	202	179	182
No of Located Roots		10	10	10	10	10	10	10	10	10	10	10	10
No of taps in filter	20	-36.52 -31.31	-31.86 -24.39	-31.86 -24.39	-36.52 -31.31	-31.86 -24.39	-31.86 -24.39	-36.52 -31.31	-31.86 -24.39	-31.86 -24.39	-36.52 -31.31	-31.86 -24.39	-31.86 -24.39
	30	-48.01 -37.47	-47.02 -30.49	-47.02 -30.49	-48.01 -37.47	-47.02 -30.49	-47.02 -30.49	-48.01 -37.47	-47.02 -30.49	-47.02 -30.49	-48.01 -37.47	-47.02 -30.49	-47.02 -30.49
	40	-257.91 -27.72	-258.37 -29.75	-256.80 -29.75	-257.65 -27.72	-258.06 -29.75	-257.33 -29.75	-256.91 -27.72	-258.47 -29.75	-257.78 -29.75	-257.30 -27.72	-258.18 -29.75	-256.50 -29.75
	50	-257.91 -40.69	-258.37 -41.52	-256.80 -41.52	-257.65 -40.69	-258.06 -41.52	-257.33 -41.52	-256.91 -40.69	-258.47 -41.52	-257.78 -41.52	-257.30 -40.69	-258.18 -41.52	-256.50 -41.52
	60	-257.91 -69.47	-258.37 -67.07	-256.80 -67.07	-257.65 -69.47	-258.06 -67.07	-257.33 -67.07	-256.91 -69.47	-258.47 -67.07	-257.78 -67.07	-257.30 -69.47	-258.18 -67.07	-256.50 -67.07
	70	-257.91 -61.69	-258.37 -60.35	-256.80 -60.35	-257.65 -61.68	-258.06 -60.35	-257.33 -60.35	-256.91 -61.68	-258.47 -60.35	-257.78 -60.35	-257.30 -61.68	-258.18 -60.35	-256.50 -60.35
	80	-257.91 -59.61	-258.37 -60.46	-256.80 -60.46	-257.65 -59.61	-258.06 -60.46	-257.33 -60.46	-256.91 -59.61	-258.47 -60.46	-257.78 -60.46	-257.30 -59.61	-258.18 -60.46	-256.50 -60.46
	90	-257.91 -69.74	-258.37 -70.76	-256.80 -70.76	-257.65 -69.74	-258.06 -70.76	-257.33 -70.76	-256.91 -69.74	-258.47 -70.76	-257.78 -70.76	-257.30 -69.74	-258.18 -70.76	-256.50 -70.76
	100	-257.91 -88.01	-258.37 -88.66	-256.80 -88.66	-257.65 -88.02	-258.06 -88.66	-257.33 -88.66	-256.91 -88.02	-258.47 -88.66	-257.78 -88.66	-257.30 -88.02	-258.18 -88.66	-256.50 -88.66

TABLE 5.12: ψ_1, ψ_2 for Channel 7 (11 roots), ℓ = 4

		$d_1 = 10^{-7}$			$d_1 = 10^{-10}$			$d_1 = 10^{-15}$			$d_1 = 10^{-20}$		
d_2 =		0.050	0.075	0.100	0.050	0.075	0.100	0.050	0.075	0.100	0.050	0.075	0.100
No of Iterations		293	211	245	299	220	254	309	229	263	316	232	267
No of Located Roots		13	11	12	13	11	12	13	11	12	13	11	12
No of taps in filter	20	-54.92 -24.34	-52.27 -24.95	-51.28 -28.04	-54.92 -24.34	-52.27 -24.95	-51.28 -28.04	-54.92 -24.34	-52.27 -24.95	-51.28 -28.04	-54.92 -24.34	-52.27 -24.95	-51.28 -28.04
	30	-262.54 -25.97	-262.09 -40.97	-261.31 -38.12	-261.46 -25.97	-260.78 -40.97	-263.12 -38.12	-263.44 -25.97	-262.80 -40.97	-260.76 -38.12	-261.70 -25.97	-259.86 -40.97	-262.70 -38.12
	40	-262.54 -44.49	-262.09 -49.66	-261.31 -51.52	-261.46 -44.49	-260.78 -49.66	-263.12 -51.52	-263.44 -44.49	-262.80 -49.66	-260.76 -51.52	-261.70 -44.49	-259.86 -49.66	-262.70 -51.52
	50	-262.54 -45.06	-262.69 -53.13	-261.31 -49.24	-261.46 -45.06	-260.78 -53.13	-263.12 -49.24	-263.44 -45.06	-262.80 -53.13	-260.76 -49.24	-261.70 -45.06	-259.86 -53.13	-262.70 -49.24
	60	-262.54 -57.37	-262.09 -54.71	-261.31 -56.55	-261.46 -57.37	-260.78 -54.71	-263.12 -56.55	-263.44 -57.37	-262.80 -54.71	-260.76 -56.55	-261.70 -57.37	-259.86 -54.71	-262.70 -56.55
	70	-262.54 -54.30	-262.09 -54.71	-261.31 -54.43	-261.46 -54.30	-260.78 -54.17	-263.12 -54.43	-263.44 -54.30	-262.80 -54.17	-260.76 -54.43	-261.70 -54.30	-259.86 -54.17	-262.70 -54.43
	80	-262.54 -58.92	-262.09 -56.81	-261.31 -57.28	-261.46 -58.92	-260.78 -56.81	-263.12 -57.29	-263.44 -58.92	-262.80 -56.81	-260.76 -57.28	-261.70 -58.92	-259.86 -56.81	-262.70 -57.28
	90	-262.54 -59.61	-262.09 -58.90	-261.31 -59.09	-261.46 -59.61	-260.78 -58.90	-263.12 -59.09	-263.44 -59.61	-262.80 -58.90	-260.76 -59.09	-261.70 -59.61	-259.86 -58.90	-262.70 -59.09
	100	-262.54 -62.47	-262.09 -62.09	-261.31 -62.11	-261.46 -62.47	-260.78 -62.08	-263.12 -62.11	-263.44 -62.47	-262.80 -62.08	-260.76 -62.11	-261.70 -62.47	-259.86 -62.08	-262.70 -62.11

TABLE 5.13: ψ_1, ψ_2 for Channel 8 (13 roots), ℓ = 4

		$d_1 = 10^{-7}$			$d_1 = 10^{-10}$			$d_1 = 10^{-15}$			$d_1 = 10^{-20}$		
d_2 =		0.050	0.075	0.100	0.050	0.075	0.100	0.050	0.075	0.100	0.050	0.075	0.100
No of Iterations		166	148	140	170	152	144	174	157	150	179	163	155
No of Located Roots		8	8	8	8	8	8	8	8	8	8	8	8
No of taps in filter	20	-48.88 -31.80	-48.88 -31.80	-48.88 -31.80	-48.88 -31.80	-48.88 -31.80	-48.88 -31.80	-48.88 -31.80	-48.88 -31.80	-48.88 -31.80	-48.88 -31.80	-48.88 -31.80	-48.88 -31.80
	30	-68.41 -26.96	-68.41 -26.96	-68.41 -26.96	-68.41 -26.96	-68.41 -26.96	-68.41 -26.96	-68.41 -26.96	-68.41 -26.96	-68.41 -26.96	-68.41 -26.96	-68.41 -26.96	-68.41 -26.96
	40	-256.46 -37.08	-254.44 -37.08	-254.71 -37.08	-255.40 -37.08	-255.71 -37.08	-257.10 -37.08	-254.03 -37.08	-256.69 -37.08	-255.41 -37.08	-257.01 -37.08	-254.33 -37.08	-255.22 -37.08
	50	-256.46 -43.93	-254.44 -43.93	-254.71 -43.93	-255.40 -43.93	-255.71 -43.93	-257.10 -43.93	-254.03 -43.93	-256.69 -43.93	-255.51 -43.93	-257.01 -43.93	-254.33 -43.93	-255.22 -43.93
	60	-256.46 -47.91	-254.44 -47.91	-254.71 -47.91	-255.40 -47.91	-255.71 -47.91	-257.10 -47.91	-254.03 -47.91	-256.69 -47.91	-255.51 -47.91	-257.01 -47.91	-254.33 -47.91	-255.22 -47.91
	70	-256.46 -55.18	-254.44 -55.18	-254.71 -55.18	-255.40 -55.18	-255.71 -55.18	-257.10 -55.18	-254.03 -55.18	-256.69 -55.18	-255.51 -55.18	-257.01 -55.18	-254.33 -55.18	-255.22 -55.18
	80	-256.46 -60.86	-254.44 -60.86	-254.71 -60.86	-255.40 -60.86	-255.71 -60.86	-257.10 -60.86	-254.03 -60.86	-256.69 -60.86	-255.51 -60.86	-257.01 -60.86	-254.33 -60.86	-255.22 -60.86
	90	-256.46 -66.58	-254.44 -66.58	-254.71 -66.58	-255.40 -66.58	-255.71 -66.58	-257.10 -66.58	-254.03 -66.58	-256.69 -66.58	-255.51 -66.58	-257.01 -66.58	-254.33 -66.58	-255.22 -66.58
	100	-256.46 -72.81	-254.44 -72.81	-254.71 -72.81	-255.40 -72.81	-255.71 -72.81	-257.10 -72.81	-254.03 -72.81	-256.69 -72.81	-255.51 -72.81	-257.01 -72.81	-254.33 -72.81	-255.22 -72.81

TABLE 5.14: ψ_1, ψ_2 for Channel 5 (8 roots), ℓ = 5

		$d_1 = 10^{-7}$			$d_1 = 10^{-10}$			$d_1 = 10^{-15}$			$d_1 = 10^{-20}$		
d_2 =		0.050	0.075	0.100	0.050	0.075	0.100	0.050	0.075	0.100	0.050	0.075	0.100
No of Iterations		136	136	129	141	142	136	147	149	141	151	152	146
No of Located Roots		9	9	9	9	9	9	9	9	9	9	9	9
No of taps in filter	20	-38.38 -23.10	-38.38 -23.10	-38.38 -23.10	-38.38 -23.10	-38.38 -23.10	-38.38 -23.10	-38.38 -23.10	-38.38 -23.10	-38.38 -23.10	-38.38 -23.10	-38.38 -23.10	-38.38 -23.10
	30	-255.51 -23.36	-254.60 -23.36	-254.91 -23.36	-255.95 -23.36	-255.92 -23.36	-253.65 -23.36	-255.99 -23.36	-253.91 -23.36	-255.32 -23.36	-255.39 -23.36	-255.23 -23.36	-253.61 -23.36
	40	-255.51 -37.51	-254.60 -37.51	-254.91 -37.51	-255.95 -37.51	-255.92 -37.51	-253.65 -37.51	-255.99 -37.51	-253.91 -37.51	-255.32 -37.51	-255.39 -37.51	-255.23 -37.51	-253.61 -37.51
	50	-255.51 -49.98	-254.60 -49.98	-254.91 -49.98	-255.95 -49.98	-255.92 -49.98	-253.65 -49.98	-255.99 -49.98	-253.91 -49.98	-255.32 -49.98	-255.39 -49.98	-255.23 -49.98	-253.61 -49.98
	60	-255.51 -69.96	-254.60 -69.96	-254.91 -69.96	-255.95 -69.96	-255.92 -69.96	-253.65 -69.96	-255.99 -69.96	-253.91 -69.96	-255.32 -69.96	-255.39 -69.96	-255.23 -69.96	-253.61 -69.96
	70	-255.51 -87.58	-254.60 -87.58	-254.91 -87.58	-255.95 -87.58	-255.92 -87.58	-253.65 -87.58	-255.99 -87.58	-253.91 -87.58	-255.32 -87.58	-255.39 -87.58	-255.23 -87.58	-253.61 -87.58
	80	-255.51 -91.03	-254.60 -91.03	-254.91 -91.03	-255.95 -91.03	-255.92 -91.03	-253.65 -91.03	-255.99 -91.03	-253.91 -91.03	-255.32 -91.03	-255.39 -91.03	-255.23 -91.03	-253.61 -91.03
	90	-255.51 -101.47	-254.60 -101.47	-254.91 -101.47	-255.95 -101.47	-255.92 -101.47	-253.65 -101.47	-255.99 -101.47	-253.91 -101.47	-255.32 -101.47	-255.39 -101.47	-255.23 -101.47	-253.61 -101.47
	100	-255.51 -116.21	-254.60 -116.20	-254.91 -116.21	-255.95 -116.22	-255.92 -116.22	-253.65 -116.22	-255.99 -116.22	-253.91 -116.22	-255.32 -116.22	-255.39 -116.22	-255.23 -116.22	-253.61 -116.22

TABLE 5.15: ψ_1, ψ_2 for Channel 6 (9 roots), ℓ = 5

		$d_1 = 10^{-7}$			$d_1 = 10^{-10}$			$d_1 = 10^{-15}$			$d_1 = 10^{-20}$		
d_2 =		0.050	0.075	0.100	0.050	0.075	0.100	0.050	0.075	0.100	0.050	0.075	0.100
No of Iterations		171	167	175	180	177	184	184	181	188	191	187	194
No of Located Roots		10	10	10	10	10	10	10	10	10	10	10	10
No of taps in filter	20	-36.52 -31.31	-36.52 -31.31	-36.52 -31.31	-36.52 -31.31	-36.52 -31.31	-36.52 -31.31	-36.52 -31.31	-36.52 -31.31	-36.52 -31.31	-36.52 -31.31	-36.52 -31.31	-36.52 -31.31
	30	-48.01 -37.47	-48.01 -37.47	-48.01 -37.47	-48.01 -37.47	-48.01 -37.47	-48.01 -37.47	-48.01 -37.47	-48.01 -37.47	-48.01 -37.47	-48.01 -37.47	-48.01 -37.47	-48.01 -37.47
	40	-257.57 -27.72	-257.14 -27.72	-257.75 -27.72	-256.69 -27.72	-257.63 -27.72	-256.93 -27.72	-256.65 -27.72	-257.81 -27.72	-257.58 -27.72	-257.23 -27.72	-257.52 -27.72	-257.31 -27.72
	50	-257.57 -40.69	-257.14 -40.69	-257.75 -40.69	-256.69 -40.69	-257.73 -40.69	-256.93 -40.69	-256.65 -40.69	-257.81 -40.69	-257.58 -40.69	-257.23 -40.69	-257.52 -40.69	-257.31 -40.69
	60	-257.57 -69.47	-257.14 -69.47	-257.75 -69.47	-256.69 -69.47	-257.73 -69.47	-256.93 -69.47	-256.65 -69.47	-257.81 -69.47	-257.58 -69.47	-257.23 -69.47	-257.52 -69.47	-257.31 -69.47
	70	-257.57 -61.68	-257.14 -61.69	-257.75 -61.69	-256.69 -61.68	-257.73 -61.68	-256.93 -61.68	-256.65 -61.68	-257.81 -61.68	-257.58 -61.68	-257.23 -61.68	-257.52 -61.68	-257.31 -61.68
	80	-257.57 -59.61	-257.14 -59.61	-257.75 -59.61	-256.69 -59.61	-257.73 -59.61	-256.93 -59.61	-256.65 -59.61	-257.81 -59.61	-257.58 -59.61	-257.23 -59.61	-257.52 -59.61	-257.31 -59.61
	90	-257.57 -69.74	-257.14 -69.74	-257.75 -69.74	-256.69 -69.74	-257.73 -69.74	-256.93 -69.74	-256.65 -69.74	-257.81 -69.74	-257.58 -69.74	-257.23 -69.74	-257.52 -69.74	-257.31 -69.74
	100	-257.57 -88.02	-257.14 -88.01	-257.75 -88.01	-256.69 -88.02	-257.73 -88.02	-256.93 -88.02	-256.65 -88.02	-257.81 -88.02	-257.58 -88.02	-257.23 -88.02	-257.52 -88.02	-257.31 -88.02

TABLE 5.16: ψ_1, ψ_2 for Channel 7 (11 roots), ℓ = 5

		$d_1 = 10^{-7}$			$d_1 = 10^{-10}$			$d_1 = 10^{-15}$			$d_1 = 10^{-20}$		
d_2 =		0.050	0.075	0.100	0.050	0.075	0.100	0.050	0.075	0.100	0.050	0.075	0.100
No of Iterations		287	278	261	295	288	271	300	294	276	308	302	284
No of Located Roots		12	13	13	12	13	13	12	13	13	12	13	13
No of taps in filter	20	-59.05 -25.75	-54.92 -24.34	-54.92 -24.34	-59.05 -25.75	-54.92 -24.34	-54.92 -24.34	-59.05 -25.75	-54.92 -24.34	-54.92 -24.34	-59.05 -25.75	-54.92 -24.34	-54.92 -24.34
	30	-260.47 -26.72	-261.64 -25.97	-261.01 -25.97	-261.35 -26.72	-260.81 -25.97	-259.30 -25.97	-262.55 -26.72	-262.07 -25.97	-259.86 -25.97	-264.01 -26.72	-259.97 -25.97	-260.94 -25.97
	40	-260.47 -44.02	-261.64 -44.49	-261.01 -44.49	-261.35 -44.02	-260.81 -44.49	-259.30 -44.49	-262.55 -44.02	-262.07 -44.49	-259.86 -44.49	-264.01 -44.02	-259.97 -44.49	-260.94 -44.49
	50	-260.47 -48.93	-261.64 -45.06	-261.01 -45.06	-261.35 -48.93	-260.81 -45.06	-259.30 -45.06	-262.55 -48.93	-262.07 -45.06	-259.86 -45.06	-264.01 -48.93	-259.97 -45.06	-260.94 -45.06
	60	-260.47 -58.32	-261.64 -57.57	-261.01 -57.37	-261.35 -58.32	-260.81 -57.37	-259.30 -57.37	-262.55 -58.32	-26.207 -57.37	-259.86 -57.37	-264.01 -58.32	-259.97 -57.37	-260.94 -57.37
	70	-260.47 -54.17	-261.64 -54.30	-261.01 -54.30	-261.35 -54.17	-260.81 -54.30	-259.30 -54.30	-262.55 -54.17	-262.07 -54.30	-259.86 -54.30	-264.01 -54.17	-259.97 -54.30	-260.94 -54.30
	80	-260.47 -58.40	-261.64 -58.92	-261.01 -58.92	-261.35 -58.40	-260.81 -58.92	-259.30 -58.92	-262.55 -58.40	-262.07 -58.92	-259.86 -58.92	-264.01 -58.40	-259.97 -58.92	-260.94 -58.92
	90	-260.47 -59.39	-261.64 -59.61	-261.01 -59.61	-261.35 -59.39	-260.81 -59.61	-259.30 -59.61	-262.55 -59.39	-262.07 -59.61	-259.86 -59.61	-264.01 -59.39	-259.97 -59.61	-260.94 -59.61
	100	-260.47 -62.39	-261.64 -62.47	-261.01 -62.47	-261.35 -62.39	-260.81 -62.47	-259.30 -62.47	-262.55 -62.39	-262.07 -62.47	-259.86 -62.47	-264.01 -62.39	-259.97 -62.47	-260.94 -62.47

TABLE 5.17: ψ_1, ψ_2 for Channel 8 (13 roots), ℓ=5

		$d_1 = 10^{-7}$			$d_1 = 10^{-10}$			$d_1 = 10^{-15}$			$d_1 = 10^{-20}$		
d_2 =		0.050	0.075	0.100	0.050	0.075	0.100	0.050	0.075	0.100	0.050	0.075	0.100
No of Iterations		171	151	141	175	156	145	179	160	151	185	166	156
No of Located Roots		8	8	8	8	8	8	8	8	8	8	8	8
No of taps in filter	20	-48.88 -31.80	-48.88 -31.80	-48.88 -31.80	-48.88 -31.80	-48.88 -31.80	-48.88 -31.80	-48.88 -31.80	-48.88 -31.80	-48.88 -31.80	-48.88 -31.80	-48.88 -31.80	-48.88 -31.80
	30	-68.41 -26.96	-68.41 -26.96	-68.41 -26.96	-68.41 -26.96	-68.41 -26.96	-68.41 -26.96	-68.41 -26.96	-68.41 -26.96	-68.41 -26.96	-68.41 -26.96	-68.41 -26.96	-68.41 -26.96
	40	-256.67 -37.08	-254.28 -37.08	-254.71 -37.08	-254.84 -37.08	-256.22 -37.08	-257.10 -37.08	-255.14 -37.08	-254.73 -37.08	-255.51 -37.08	-255.83 -37.08	-254.41 -37.08	-255.22 -37.08
	50	-256.67 -43.93	-254.28 -43.93	-254.71 -43.93	-254.84 -43.93	-256.22 -43.93	-257.10 -43.93	-255.14 -43.93	-254.73 -43.93	-255.51 -43.93	-255.83 -43.93	-254.41 -43.93	-255.22 -43.93
	60	-256.67 -47.91	-254.28 -47.91	-254.71 -47.91	-254.84 -47.91	-256.22 -47.91	-257.10 -47.91	-255.14 -47.91	-254.73 -47.91	-255.51 -47.91	-255.83 -47.91	-254.41 -47.91	-255.22 -47.91
	70	-256.67 -55.18	-254.28 -55.18	-254.71 -55.18	-254.84 -55.18	-256.22 -55.18	-257.10 -55.18	-255.14 -55.18	-254.73 -55.18	-255.51 -55.18	-255.83 -55.18	-254.41 -55.18	-255.22 -55.18
	80	-256.67 -60.86	-254.28 -60.86	-254.71 -60.86	-254.84 -60.86	-256.22 -60.86	-257.10 -60.86	-255.14 -60.86	-254.73 -60.86	-255.51 -60.86	-255.83 -60.86	-254.41 -60.86	-255.22 -60.86
	90	-256.67 -66.58	-254.28 -66.58	-254.71 -66.58	-254.84 -66.58	-256.22 -66.58	-257.10 -66.58	-255.14 -66.58	-254.73 -66.58	-255.51 -66.58	-255.83 -66.58	-254.41 -66.58	-255.22 -66.58
	100	-256.67 -72.81	-254.28 -72.81	-254.71 -72.81	-254.84 -72.81	-256.22 -72.81	-257.10 -72.81	-255.14 -72.81	-254.73 -72.81	-255.51 -72.81	-255.83 -72.81	-254.41 -72.81	-255.22 -72.81

TABLE 5.18: ψ_1, ψ_2 for Channel 5 (8 roots), $\ell = 6$

		$d_1 = 10^{-7}$			$d_1 = 10^{-10}$			$d_1 = 10^{-15}$			$d_1 = 10^{-20}$		
d_2 =		0.050	0.075	0.100	0.050	0.075	0.100	0.050	0.075	0.100	0.050	0.075	0.100
No of Iterations		139	137	129	144	143	136	150	150	141	154	153	146
No of Located Roots		9	9	9	9	9	9	9	9	9	9	9	9
No of taps in filter	20	-38.38 -23.10	-38.38 -23.10	-38.38 -23.10	-38.38 -23.10	-38.38 -23.10	-38.38 -23.10	-38.38 -23.10	-38.38 -23.10	-38.38 -23.10	-38.38 -23.10	-38.38 -23.10	-38.38 -23.10
	30	-254.94 -23.36	-254.60 -23.36	-254.91 -23.36	-256.60 -23.36	-255.92 -23.36	-253.65 -23.36	-254.55 -23.36	-253.91 -23.36	-255.32 -23.36	-254.04 -23.36	-255.23 -23.36	-253.61 -23.36
	40	-254.94 -37.51	-254.60 -37.51	-254.91 -37.51	-256.60 -37.51	-255.92 -37.51	-253.65 -37.51	-254.55 -37.51	-253.91 -37.51	-255.32 -37.51	-254.04 -37.51	-255.23 -37.51	-253.61 -37.51
	50	-254.94 -49.98	-254.60 -49.98	-254.91 -49.98	-256.60 -49.98	-255.92 -49.98	-253.65 -49.98	-254.55 -49.98	-253.91 -49.98	-255.32 -49.98	-254.04 -49.98	-255.23 -49.98	-253.61 -49.98
	60	-254.94 -69.96	-254.60 -69.96	-254.91 -69.96	-256.60 -69.96	-255.92 -69.96	-253.65 -69.96	-254.55 -69.96	-253.91 -69.96	-255.32 -69.96	-254.04 -69.96	-255.23 -69.96	-253.61 -69.96
	70	-254.94 -87.58	-254.60 -87.58	-254.91 -87.58	-256.60 -87.58	-255.92 -87.58	-253.65 -87.58	-254.55 -87.58	-253.91 -87.58	-255.32 -87.58	-254.04 -87.58	-255.23 -87.58	-253.61 -87.58
	80	-254.94 -91.03	-254.60 -91.03	-254.91 -91.03	-256.60 -91.03	-255.92 -91.03	-253.65 -91.03	-254.55 -91.03	-253.91 -91.03	-255.32 -91.03	-254.04 -91.03	-255.23 -91.03	-253.61 -91.03
	90	-254.94 -101.47	-254.60 -101.47	-254.91 -101.47	-256.60 -101.47	-255.92 -101.47	-253.65 -101.47	-254.55 -101.47	-253.91 -101.47	-255.32 -101.47	-254.04 -101.47	-255.23 -101.47	-253.61 -101.47
	100	-254.94 -116.21	-254.60 -116.20	-254.91 -116.21	-256.60 -116.22	-255.92 -116.22	-253.65 -116.22	-254.55 -116.22	-253.91 -116.22	-255.32 -116.22	-254.04 -116.22	-255.23 -116.22	-253.61 -116.22

TABLE 5.19: ψ_1, ψ_2 for Channel 6 (9 roots), $\ell = 6$

		$d_1 = 10^{-7}$			$d_1 = 10^{-10}$			$d_1 = 10^{-15}$			$d_1 = 10^{-20}$		
d_2 =		0.050	0.075	0.100	0.050	0.075	0.100	0.050	0.075	0.100	0.050	0.075	0.100
No of Iterations		216	220	206	223	230	214	229	236	220	236	241	226
No of Located Roots		10	11	11	10	11	11	10	11	11	10	11	11
No of taps in filter	20	-36.52 -31.31	-29.29 -20.20	-29.29 -20.20	-36.52 -31.31	-29.29 -20.20	-29.29 -20.20	-36.52 -31.31	-29.29 -20.20	-29.29 -20.20	-36.52 -31.31	-29.29 -20.20	-29.29 -20.20
	30	-48.01 -37.47	-51.56 -28.72	-51.56 -28.72	-48.01 -37.47	-51.56 -28.72	-51.56 -28.72	-48.01 -37.47	-51.56 -28.72	-51.56 -28.72	-48.01 -37.47	-51.56 -28.72	-51.56 -28.72
	40	-258.18 -27.72	-256.98 -26.08	-257.99 -26.08	-257.20 -27.72	-258.42 -26.08	-257.90 -26.08	-257.13 -27.72	-258.51 -26.08	-259.77 -26.08	-256.61 -27.72	-257.79 -26.08	-258.72 -26.08
	50	-258.18 -40.69	-256.98 -37.88	-257.99 -37.88	-257.20 -40.69	-258.42 -37.88	-257.90 -37.88	-257.13 -40.69	-258.51 -37.88	-259.77 -37.88	-256.61 -40.69	-257.79 -37.88	-258.72 -37.88
	60	-258.18 -69.47	-256.98 -69.03	-257.99 -69.03	-257.20 -69.47	-258.42 -69.04	-257.90 -69.04	-257.13 -69.47	-258.51 -69.04	-259.77 -69.04	-256.61 -69.47	-257.79 -69.04	-258.72 -49.04
	70	-258.18 -61.68	-256.98 -64.26	-257.99 -64.26	-257.20 -61.68	-258.42 -64.26	-257.90 -64.26	-257.13 -61.68	-258.51 -64.26	-259.77 -64.26	-256.61 -61.68	-257.79 -64.26	-258.72 -64.26
	80	-258.18 -59.61	-256.98 -58.85	-257.99 -58.86	-257.20 -59.61	-258.42 -58.86	-257.90 -58.86	-257.13 -59.61	-258.51 -58.86	-259.77 -58.86	-256.61 -59.61	-257.79 -58.86	-258.72 -58.86
	90	-258.18 -69.74	-256.98 -67.67	-257.99 -67.67	-257.20 -69.74	-258.42 -67.67	-257.90 -67.67	-257.13 -69.74	-258.51 -67.67	-259.77 -67.67	-256.61 -69.74	-257.79 -67.67	-258.72 -67.67
	100	-258.18 -88.02	-256.98 -85.29	-257.99 -85.39	-257.20 -88.02	-258.42 -85.39	-257.90 -85.39	-257.13 -88.02	-258.51 -85.39	-259.77 -85.39	-256.61 -88.02	-257.79 -85.39	-258.72 -85.39

TABLE 5.20: ψ_1, ψ_2 for Channel 7 (11 roots), $\ell = 6$

		$d_1 = 10^{-7}$			$d_1 = 10^{-10}$			$d_1 = 10^{-15}$			$d_1 = 10^{-20}$		
d_2 =		0.050	0.075	0.100	0.050	0.075	0.100	0.050	0.075	0.100	0.050	0.075	0.100
No of Iterations		325	348	265	335	357	275	343	365	283	350	372	288
No of Located Roots		13	13	13	13	13	13	13	13	13	13	13	13
No of taps in filter	20	-54.92 -24.34	-54.92 -24.34	-54.92 -24.34	-54.92 -24.34	-54.92 -24.34	-54.92 -24.34	-54.92 -24.34	-54.92 -24.34	-54.92 -24.34	-54.92 -24.34	-54.92 -24.34	-54.92 -24.34
	30	-260.23 -25.97	-260.72 -25.97	-260.69 -25.97	-260.90 -25.97	-262.25 -25.97	-260.01 -25.97	-260.77 -25.97	-262.81 -25.97	-259.16 -25.97	-260.71 -25.97	-260.94 -25.97	-259.68 -25.97
	40	-260.23 -44.49	-260.72 -44.49	-260.69 -44.49	-260.90 -44.49	-262.25 -44.49	-260.01 -44.49	-260.77 -44.49	-262.81 -44.49	-259.16 -44.49	-260.71 -44.49	-260.94 -44.49	-259.68 -44.49
	50	-260.23 -45.06	-260.72 -45.06	-260.69 -45.06	-260.90 -45.06	-262.25 -45.06	-260.01 -45.06	-260.77 -45.06	-262.81 -45.06	-259.16 -45.06	-260.71 -45.06	-260.94 -45.06	-259.68 -45.06
	60	-260.23 -57.37	-260.72 -57.37	-260.69 -57.37	-260.90 -57.37	-262.25 -57.37	-260.01 -57.37	-260.77 -57.37	-262.81 -57.37	-259.16 -57.37	-260.71 -57.37	-260.94 -57.37	-259.68 -57.37
	70	-260.23 -54.30	-260.72 -54.30	-260.69 -54.30	-260.90 -54.30	-262.25 -54.30	-260.01 -54.30	-260.77 -54.30	-262.81 -54.30	-259.16 -54.30	-260.71 -54.30	-260.94 -54.30	-259.68 -54.30
	80	-260.23 -58.92	-260.72 -58.92	-260.69 -58.92	-260.90 -58.92	-262.25 -58.92	-260.01 -58.92	-260.77 -58.92	-262.81 -58.92	-259.16 -58.92	-260.71 -58.92	-260.94 -58.92	-259.68 -58.92
	90	-260.23 -59.61	-260.72 -59.61	-260.69 -59.61	-260.90 -59.61	-262.25 -59.61	-260.01 -59.61	-260.77 -59.61	-262.81 -59.61	-259.16 -59.61	-260.71 -59.61	-260.94 -59.61	-259.68 -59.61
	100	-260.23 -62.47	-260.72 -62.47	-260.69 -62.47	-260.90 -62.47	-262.25 -62.47	-260.01 -62.47	-260.77 -62.47	-262.81 -62.47	-259.16 -62.47	-260.71 -62.47	-260.94 -62.47	-259.68 -62.47

TABLE 5.21: ψ_1, ψ_2 for Channel 8 (13 roots), $t = 6$

5.3 Adjustment Time of the Adaptive Filter

Table 5.22 shows the number of iterations taken by the Standard algorithm (Section 4.4) and the Modified algorithm (Section 4.6) (for $d_1 = 10^{-7}$ and 10^{-10}, $c = 1.0$, $\ell = 6$, $d_2 = 0.1$) to find all the required roots of Channels 5, 6, 7 and 8.

		Channel 5 8 roots	Channel 6 9 roots	Channel 7 11 roots	Channel 8 13 roots
Standard Algorithm	$d_1 = 10^{-10}$	134	113	148	255
Modified Algorithm	$d_1 = 10^{-7}$	141	129	206	265
	$d_1 = 10^{-10}$	145	136	214	275

TABLE 5.22: Number of Iterations Required for the Two Algorithms (with 9 starting-points) to find all the required roots of Channels 5-8

As described in Section 4.4.1, each step of the iterative process requires 2g complex multiplications and one complex division. In addition, each root outside the unit circle requires 2n+g+2 complex multiplications in the adjustment of the linear filter except for the first root which only requires n+g+1 complex multiplications.

Consider the worst case where there are 40 components in the sampled impulse-response, 13 roots of Y(z) outside the unit circle, 50 taps in the adaptive filter and a total of 280 iterations involved in locating the required roots of Y(z). The complete operation involved in adjusting the adaptive linear filter and estimating the sampled impulse-response of the channel and filter now entails 24195 complex multiplications and 280 complex divisions. This amount of computation can be achieved in some 25 ms using a digital signal processor which does 1 16x16 bit multiplication in 250 ns[7].

5.4 The Effect of a Noisy Channel Estimate on the System

Many methods are available for estimating the sampled impulse-response of a channel, Y,[7,31-33]. These vary from the simple but rather slow linear feedforward channel estimator[7,31] to the highly complex but fast Kalman filter channel estimator[93-94]. So far, no assumption has been made as to the particular form of channel estimator employed, and the sampled impulse-response of the channel has been assumed to be provided correctly at the receiver. In practice, of course, this information cannot be provided exactly, and therefore, it is important that this error in the estimate of the channel will not be amplified by the system so that it subsequently appears at the output.

Let Y(z) be the z-transform of the correct channel estimate, Y (with g+1 components), and $\overset{\approx}{Y}(z)$ be the z-transform of the noisy channel estimate $\overset{\approx}{Y}$ (this is what is provided at the receiver by the channel estimator). Since the noisy sequence $\overset{\approx}{Y}$ can be thought of as the sum of the correct sequence Y and an error sequence E, therefore

$$\overset{\approx}{Y} = Y + E \tag{5.9}$$

and

$$\overset{\approx}{Y}(z) = Y(z) + E(z) \tag{5.10}$$

As far as the receiver is concerned, $\overset{\approx}{Y}$ is the correct channel estimate and it is also the only one which is provided.

Suppose now that the linear filter (Figure 2.1) which corresponds to the noisy channel, whose estimate is $\overset{\approx}{Y}$, has a z-transform $\overset{\approx}{D}(z)$. When this filter (which is assumed to have n+1 taps) operates on $\overset{\approx}{Y}(z)$, its effect is to remove all roots (zeros) of $\overset{\approx}{Y}(z)$ that lie outside the unit circle in the z-plane and replace them by the complex conjugates of their reciprocals, so that the resultant z-transform $\tilde{\tilde{Y}}(z)\tilde{\tilde{D}}(z)$ has all roots inside the unit circle. Now from equation 5.10

$$\tilde{\tilde{Y}}(z)\ \tilde{\tilde{D}}(z) = (Y(z) + E(z))\ \tilde{\tilde{D}}(z)$$

$$= Y(z)\ \tilde{\tilde{D}}(z) + E(z)\ \tilde{\tilde{D}}(z) \qquad (5.11)$$

But the actual channel has a z-transform Y(z), and the receiver now, thinking that the channel estimate is $\tilde{\tilde{Y}}$ and subsequently performing an unitary transformation[49] on it, has in fact operated on Y(z). The error between $\tilde{\tilde{Y}}(z)\tilde{\tilde{D}}(z)$ and $Y(z)\tilde{\tilde{D}}(z)$ is

$$\xi(z) = \tilde{\tilde{Y}}(z)\tilde{\tilde{D}}(z) - Y(z)\tilde{\tilde{D}}(z)$$

$$= (\tilde{\tilde{Y}}(z) - Y(z))\tilde{\tilde{D}}(z)$$

$$= E(z)\tilde{\tilde{D}}(z) \qquad (5.12)$$

Since the linear filter is adjusted to perform an unitary transformation[49] at all times, its effect on the input signal (the noisy sequence $\tilde{\tilde{Y}}$) is such that the level of the signal, $|\tilde{\tilde{Y}}|^2$, together with all the noise statistics, remain unchanged[49] at its output. Thus, the sum of the squares of the moduli (absolute values) of ξ and that of E are the same,

$$|\xi|^2 = |E|^2 \qquad (5.13)$$

Hence, from equations 5.9, 5.12 and 5.13, the error $|E|^2$ which appears in the channel estimate at the input of the linear filter remains unchanged at the output. More precisely, if the square-error in the channel estimator is

$$|E|^2 = |\tilde{\tilde{Y}} - Y|^2 \qquad (5.14)$$

and the square-error in the signal at the output of the channel and filter is

$$|\xi|^2 = |v_o + v_1 + \ldots + v_{n+g}|^2 \qquad (5.15)$$

where $$v_o + v_1 z^{-1} + \ldots + v_{n+g} z^{-n-g} = \tilde{\tilde{Y}}(z)\tilde{\tilde{D}}(z) - Y(z)\tilde{\tilde{D}}(z) \qquad (5.16)$$

(equation 5.12), then the two quantities (5.14-5.15) are exactly the same.

Tests were carried out on the four channels, Channels 5, 6, 7 and 8, using computer-simulations in order to verify this result. The noisy channel estimate was generated by adding a random Gaussian sequence of variance σ^2 and zero mean to the real and imaginary parts of each component in the correct channel estimate. The Gaussian sequence, which gives the error sequence E (equation 5.9), was generated by the NAG subroutine (G05DDF $(0.0,\sigma)$)[50]. The error $|\tilde{\tilde{Y}}-Y|^2$ was calculated and stored. A root-finding subroutine in the NAG library (C02ADF)[50] was also used to find the roots (zeros) of $\tilde{\tilde{Y}}(z)$. Having found all the roots of $\tilde{\tilde{Y}}(z)$, the z-transform $\tilde{\tilde{D}}(z)$, and, subsequently $Y(z)\tilde{\tilde{D}}(z)$, $\tilde{\tilde{Y}}(z)\tilde{\tilde{D}}(z)$ were calculated. Random Gaussian sequences of variance 0.01, 0.001, 0.0001 and 0 were added to the correct channel estimate to generate the noisy data $\tilde{\tilde{Y}}$ (equation 5.9), these correspond to the values of approximately -20 dB, -30 dB, -40 dB and $-\infty$ dB, respectively, in the error $|\tilde{\tilde{Y}}-Y|^2$. The length of the linear filter used in this experiment ranged from 20 taps to 100 taps in steps of 20.

Results showed that $|\tilde{\tilde{Y}}|^2$ and the sum of the squares of the moduli (absolute values) of the sequence whose z-transform is $\tilde{\tilde{Y}}(z)\tilde{\tilde{D}}(z)$ were identical, which confirmed that the operation performed on the channel by the linear filter (whose z-transform $\overset{\simeq}{D}(z)$ is indeed an unitary transformation. It was found that as the number of taps in the linear filter increased, the error between $|Y-\overset{\simeq}{Y}|^2$ (equation 5.14) and $|\xi|^2$ (equation 5.15), became small, and when 100 taps were used, the error was practically zero and that the two values were the same.

CHAPTER 6

COMMENTS ON THE RESEARCH PROJECT

6.1 Discussions

The whole idea of the project centres on the fact that the linear filter ahead of the detector must be an all-pass network, such that the sampled impulse-response of the channel and filter is minimum phase, and the tap gains of the filter are determined by the combined operation of a one-tap feedback filter and a two-tap feedforward filter. The investigation has been concerned mainly with the development of a suitable root-finding algorithm that will yield all or most of the roots (zeros) of Y(z) that lie outside the unit circle (in the z-plane) which are absolutely essential in the determination of the tap gains of the adaptive linear filter ahead of the detector.

An on-line system is studied first. The receiver here operates on the continuous stream of received samples $\{r_i\}$ which is subject to both intersymbol interference and additive white Gaussian noise. It has been found that the system is very susceptible to noise, even at high signal to noise ratio, and all the test results point to the unsuitability of this system for use in practice. However, the experience gained from this part of the investigation was very valuable in the study of the off-line system. The latter requires only the sampled impulse-response of the channel, which must be provided at the receiver and it requires no other information. The receiver operates directly on this estimate in an off-line fashion thereby avoiding the problems caused by the additive white Gaussian noise in the received samples $\{r_i\}$ and requiring much less computation than the on-line system. This proved very successful and, for the first time in this investigation, it was possible to find all the required roots of Y(z) even in the case of the most severely distorted channel. A reduction of about 50% in the number of arithmetic operations has been achieved by changing the configuration so that only the one-tap feedback filter is usedin the estimation of the roots.

Extensive tests have been carried out to find an optimum arrangement for the 9600 bit/s QAM data-transmission system. Two algorithms, the Standard algorithm and the Modification algorithm (Section 4.10), were developed and the details are as described in Sections 4.4 and 4.6. It is possible that the adjustment of a 40-tap filter can be completed in less than 10 ms for a channel which represents the case of a typical worst circuit in the British public switched telephone network for the transmission rate of 1200 bit/s using some of the high-speed signal processors currently available. The adjustment time refers both to the time taken to set up the adaptive linear filter and that required to give the resultant sampled impulse-responseof the filter and channel (which is required by the detector).

The next stage of the research work was concerned with the transmission of data at 19200 bit/s using a 64-point QAM signal constellation. Schwartz[10] has indicated that the Shannon capacity expression can be extended to include the non-ideal frequency characteristics in the telephone circuits. If a bandwidth of approximatley 3 KHz and a signal to noise ratio of 30 dB are assumed in the calculation a figure of about 23,000 bit/s is obtained. Since 19200 bit/s is over 80% of the Shannon limit, it is inevitable that the channels are much more difficult to handle, and that conventional techniques are unlikely to operate satisfactorily. Indeed, as an indication of the worsening condition, the number of roots (zeros) of Y(z) that lie outside the unit circle in the z-plane has now increased from a number between 3 to 8 in the 9600 bit/s system to a number between 8 and 13. Furthermore, there are now more roots lying close to the unit circle. Different techniques aimed at improving the Standard and the Modified algorithms have been tested and it was found that the problem could best be solved by a slight modification in the starting-up procedure. Using 9 starting-points instead of 5 used in the 9600 bit/s system enabled all of the roots (zeros) of Y(z) for the four telephone channels to be found very accurately and reasonably quickly. The time required for adjusting a 50-tap adaptive linear filter (a number seems adequate for this situation) is around 25 ms.

The adjustment time can be reduced appreciably by truncating the sequence Y (sampled impulse response of the channel) so that a shorter sequence is used by the iterative root-finding process.

The Standard algorithm as described in Section 4.4 where the receiver operates solely and directly on the sampled impulse-response of the channel, Y, to give the required roots (zeros) of Y(z) is in fact, equivalent to the Newton-Raphson method for locating the zeros of a polynomial. Consider the polynomial Q(x) where

$$Q(x) = A_n x^n + A_{n-1} X^{n-1} + \ldots + A_o = 0$$

Let the zeros Q(x) be x_0, x_1, ..., x_n such that $Q(x_o) = Q(x_1) = \ldots = Q(x_n) = 0$. The Newton-Raphson method[35-37] enables the roots (zeros) of Q(x) to be computed as follows:

$$\alpha_{i+1} = \alpha_i - \frac{Q\ (\alpha_i)}{Q'(\alpha_i)} \tag{6.1}$$

where α_i is an estimate of one of the roots at the i^{th} iteration. $Q(\alpha_i)$ is the value of the polynomial Q(x) at $x = \alpha_i$ and $Q'(\alpha_i)$ is the value of the first derivative of Q(x) at $x = \alpha_i$ and as $i \to \infty$, $\alpha_i \to \{x_i\}$.

Suppose now that the z-transform of the sampled impulse-response of the channel is

$$Y(z) = y_0 + y_1 z^{-1} + \ldots y_g z^{-g}$$

$$= (1 + \beta_1 z)\ (u_0 z^{-1} + u_1 z^{-2} + \ldots + u_{g-1} z^{-g}) \tag{6.2}$$

where $-\frac{1}{\beta_1}$ is a root (zero) of Y(z) lying outside the unit circle in the z-plane. Now let

$$x = -z^{-1} \tag{6.3}$$

and

$$\begin{aligned} Q(x) &= Y(-z^{-1}) \\ &= y_0 - y_1x + y_2x^2 - y_3x^3 + \ldots + y_g(-x)^g \end{aligned} \tag{6.4}$$

where it can be seen that Q(x) has a root β_1 which lies inside the unit circle in the x-plane. The first derivative of Q(x) is

$$Q'(x) = -y_1 + 2y_2x - 3y_3x^2 + \ldots + gy_g(-x)^{g-1} \tag{6.5}$$

Substituting λ_i for x in equations 6.4 and 6.5, where λ_i is an estimate of β_1,

$$Q(\lambda_i) = y_0 - \lambda_iy_1 + \lambda_i^2y_2 - \lambda_i^3y_3 + \ldots+(-\lambda_i)^gy_g \tag{6.6}$$

$$Q'(\lambda_i)=-y_1 + 2\lambda_iy_2 - 3\lambda_i^2y_3 + \ldots - g(-\lambda_i)^{g-1}y_g \tag{6.7}$$

According to Newton-Raphson method (equation 6.1),

$$\begin{aligned} \lambda_{i+1} &= \lambda_i - \frac{Q\ (\lambda_i)}{Q'(\lambda_i)} \\ &= \lambda_i + \frac{y_0-\lambda_iy_1 + \lambda_i^2y_2 + \ldots +(-\lambda_i)^gy_g}{y_1-2\lambda_iy_2 + 3\lambda_i^2y_3+\ldots +g(-\lambda_i)^{g-1}y_g} \end{aligned} \tag{6.8}$$

It can be seen that the evaluation of λ_{i+1} (equation 6.8) is identical to that of the Standard algorithm when c=1 (equations 4.157, 4.158, 4.161 and 4.165).

It is assumed in this report that the iterative process operates on all roots of Y(z) that lie outside the unit circle. However, it is clear from equations 4.103 and 4.116 that, if there are one or more roots of Y(z) only just outside the unit circle, an excessive number of taps are required in the adaptive transversal filter in order to achieve the required accuracy of adjustment of the adaptive system. In the case of an adaptive nonlinear equalizer it is obvious that as many as possible of the roots of Y(z) that lie outside the unit circle should be processed by the system, since otherwise there will be a needless reduction in tolerance to additive noise[7]. However, in the case of a near-maximum-likelihood detector it is in fact only necessary to process those roots of Y(z) that lie further away from the unit circle[27-28]. It is most unlikely that any useful advantage would be gained here by processing roots whose absolute values are less than 1.1, and with a well-designed detector, roots with absolute values as high as 1.3 (and perhaps more) could be left unprocessed by the system without significantly degrading the performance of the detector. This would not only reduce the number of roots to be processed, and hence the total number of iterations involved, but it would also reduce the number of taps required in the adaptive filter. However, the more roots of Y(z) that are processed by the adaptive system, the simpler becomes the detector that achieves near-maximum-likelihood detection, so that clearly a compromise must be reached between the complexity of the adaptive system and the complexity of the detector. Since no particular assumptions are made in this report about the detector process used, the adaptive system is taken to be that appropriate to the least effective detector that would be suitable here, namely a conventional nonlinear equalizer.

6.2 Conclusion

The research project has been concerned with the adaptive adjustment of the receiver in a digital data-transmission system operating with additive noise and severe intersymbol-interference in the received signal. Many different adaptive techniques have been studied, including both "on-line" and "off-line" systems. The former have been rejected on account of their low tolerance to the additive noise in the signal, this noise beinq effectively eliminated in the off-line system. Extensive computer simulation tests on the latter have shown this to be potentially very cost effective. As a result of the project, a novel technique has been developed for the adjustment of the linear feedforward transversal filter that is employed ahead of a near-maximum-likelihood detector and at the same time for the estimation of the sampled impulse-response of the channel and filter, to give the information on the received signal needed by the detector. The latter two operations are equivalent to the adjustment of the feedforward and feedback transversal filters, respectively, in the corresponding nonlinear (decision feedback) equalizer. The equalizer is, of course, a degenerate form of a near-maximum-likelihood detector, being obtained when the latter is reduced to its simplest possible form. The adaptive system operates directly on the estimate of the sampled impulse-response of the channel, that must be provided at the receiver and it requires no other input signals. It employs a root-finding algorithm that determines, in sequence, the roots (zeros) of the z-transform of the sampled impulse-response of the channel, whose absolute values (moduli) exceed some constant value in the ring 1 to 1.5. It then uses a knowledge of these roots to determine the tap gains of the linear feedforward transversal filter and to form an estimate of the sampled impulse-response of the channel and filter. The technique exploits the fact that, quite contrary to the case of a nonlinear equalizer, a near-maximum-likelihood detector can tolerate a considerable departure from the ideal, in the adjustment of the adaptive filter, with no significant degradation in performance, provided only that the adaptive filter is constrained to be an allpass network which tends to make the resulting response of the channel and filter minimum phase. The technique thereby enables an appreciably more cost-effective design of the receiver to be obtained. It is ideally suited to applications of digital data-transmission at speeds of 9600 bit/s or higher (19200 bit/s) over voiceband channels such as telephone circuits.

6.3 Suggestions for Further Investigation

The report is concerned mainly with the development of a novel technique for the adjustment of the adaptive linear filter in the receiver and at the same time provides the resultant sampled impulse-response of the channel and filter for the detector. It is aimed for the transmission of data at either 9600 bit/s using 16-point QAM or 19200 bit/s using 64-point QAM for the British public switched telephone network. It is assumed, in the report, that the telephone circuits are time-invariant or varying very slowly with time so that the sampled impulse-response of the baseband channel can be taken to be constant, and that it is known at the receiver.

Despite efforts to create potentially damaging situations such as the use of a truncated sequence of the sampled impulse-response and the use of deliberately introduced roots to most of those of Y(z) in the z-plane, the algorithms perform remarkably well. In view of this, it would not be unreasonable to apply the same technique to time-varying channels such as HF radio links where it is well known that the channels can change rapidly with time. Indeed, the author has been involved in the development of such a modem for some time.

REFERENCES

1. JUNKER, S.L. and NOLLER, W.E.
"Digital Private Branch Exchange",
IEEE Comm. Magazine May 1983, pp 11-17.

2. DORROS, I.
"Telephone Nets go Digital",
IEEE Spectrum April 1983 pp 49-54.

3. WESTCOTT, R.J.
"An experimental adaptively equalized modem for data transmission over the switched telephone network",
Radio and Electron. Eng., 42, pp 499-507, November, 1972.

4. EDWARDSON, K. and HYMAN, H.
"Modems for data transmission on voice grade lines",
Elect. Comm., Vol. 48, pp 110-120, 1973.

5. NYMAN, H., SALCEDO, F., SHARPE, J.T.L., TARRY, C.W.
"Data Modem Evolution",
Elect. Comm. Vol. 57, No. 3, pp 187-194, 1982.

6. CLARK, A.P.
"Principles of digital data-transmission",
Pentech Press, 1976.

7. CLARK, A.P.
"Advanced data-transmission systems",
Pentech Press 1977.

8. LUCKY, R.W., SALZ, J. and WELDON, Jr. E.J.
"Principles of data communications",
McGraw-Hill, 1968.

9. CARLSON, A.B.
"Communication systems",
McGraw-Hill, 1975.

10. SCHWARTZ, M.
"Information transmission, modulation and noise",
McGraw-Hill, 1980.

11. SHANNON, C.E. and WEAVER, W.
"The mathematical theory of communication",
University of Illinois Press, 1949.

P 12. WOODWARD, P.M.
"Probability and information theory, with applications to radar",
Pergamon Press, 1953.

13. WOZENCRAFT, J.M. and JACOB, I.M.
"Principles of communication engineering",
John Wiley and Sons, 1965.

14. LATHI, B.P.
"Random signals and communication theory",
Intertext, 1968.

15. FORNEY, G.P.
"Maximum-likelihood sequence estimation for digital sequences in the presence of intersymbol interference",
IEEE Trans., 1972, IT-18, pp 368-378.

16. MONSEN, P.
"Feedback equalization for fading dispersive channels",
IEEE Trans., 1971, IT-17, pp 56-64.

17. CLARK, A.P.
"Design technique for non-linear equalizers",
Proc. IEE, 1973, 120, pp 329-333.

18. SALZ, J.
"Optimum mean-square decision-feedback equalization",
Bell Sys. Tech. J., 52, pp 1341-1373, Oct. 1973.

19. MESSERSCHMITT, D.G.
"A geometric theory of intersymbol interference. Part I: zero forcing and decision-feedback equalization", Bell. Syst. Tech. J., 52, pp 1483-1519, Nov. 1973.

20. CLARK, A.P. and TINT,U.S.
"Linear and non-linear transverse equalizers for baseband channels",
Radio Electron. Eng., 1975, 45, pp 271-283.

21. BELFIORE, C.A. and PARK, J.H.
"Decision feedback equalization",
Proc. IEEE, 1979, 67, pp 1143-1156.

22. CLARK, A.P. and FAIRFIELD, M.J.
"Detection processes for a 9600 bit /sec modem",
Radio Electron. Eng., 1981, 51, pp 455-465.

23. CLARK, A.P., IP, S.F.A. and SOON, C.W.
"Pseudobinary detection processes for a 9600 bit/sec modem",
Proc. IEE, 1982, 129, Pt. F, pp 305-314..

24. CLARK, A.P. and CLAYDEN, M.
"The pseudobinary Viterbi detector",
Proc. IEE, 1984, 131, Pt. F, pp 208-218.

25. CLARK, A.P., NAJDI, H.Y. and FAIRFIELD, M.J.
"Data transmission at 19.2 Kbit/sec over telephone circuits",
Radio Electron. Eng., 1983, 53, pp 157-166.

26. CLARK, A.P.
"Signal distortion in a sampled digital signal",
IERE Conference Proceedings No. 37, September 1977, pp 115-124.

27. CLARK, A.P., HARVEY, J.D. and DRISCOLL, J.P.
"Near-maximum-likelihood detection processes for distorted digital signals",
Radio Electron. Eng., 1978, 48, pp 301-309.

28. CLARK, A.P., KWONG, C.P. and HARVEY, J.D.
"Detection processes for severely distorted digital signals",
Proc. IEEE, 1979, 3, Pt. 6, pp 27-57.

29. SER, W.
"Detection of digital signals transmitted over a known time invariant channels,
PhD Thesis, Loughborough University of Technology, 1982.

30. HARVEY, J.D.
"Synchronization of a synchronous modem",
SERC Report No. GR/A/1200.7, SERC, Swindon, England, December 1980.

31. MAGEE, F.R. and PROAKIS, J.G.
"Adaptive maximum-likelihood sequence estimation for digital signalling in the presence of intersymbol interference",
IEEE Trans., 1973, IT-19, pp 120-124.

32. CLARK, A.P., KWONG, C.P. and McVERRY, F.
"Estimation of the sampled impulse-response of a channel",
Signal Processing, 1980, 2, pp 39-53.

33. CLARK, A.P. and McVERRY, F.
"Channel estimation for an HF radio link",
IEE Proc., 198L, Pt. F, pp 33-42.

34. HARUN, R.
"Techniques of channel estimation for HF radio links",
PhD Thesis, Loughborough University of Technology, 1984.

35. RALSTON, A.
"A first course in numerical analysis",
McGraw-Hill, 1965.

36. HILDEBRAND, F.B.
"Introduction to numerical analysis",
McGraw-Hill, 1965.

37. HOUSEHOLDER, A.S.
"Principle of numerical analysis",
McGraw-Hill, 1953.

38. LYON, W.V.
"Note on a method of evaluating the complex roots of a quartic equation",
Journal of Math. and Physics, Vol. 3, 1924, pp 188-190.

39. SHARP, H.S.
"A comparison of methods for evaluating the complex roots of quartic equations",
Journal of Maths. and Physics, Vol. 20, 1941, pp 243-258.

40. LIN, S.N.
"A method for finding roots of algebraic equations",
Journal of Math. and Physics, Vol. 22, 1943, pp 60-77.

41. LIN, S.N.
"A method of successive approximations of evaluating the real and complex roots of cubic and higher-order equations",
Journal of Math. and Physics, Vol. 20, 1941, pp 231-242.

42. AITKEN, A.C.
"On Bernoulli's numerical solutions of algebraic equations",
Proc. Roy. Soc. Edin., Vol. 46, 1925-26, Section A, pp 289-305.

43. AITKEN, A.C.
"Studies in practical mathematics. VI: On the factorization of polynomials by iterative methods",
Proc. Roy. Soc. Edin., Vol. 63-64, 1949-57, pp 174-191.

44. AITKEN, A.C.
"Stufies in practical mathematics. III: On the theory of methods of factorizing polynomials by iterative division",
Proc. Roy. Soc. Edin., Vol. 63-64, 1949-57, pp 326-335.

45. AITKEN, A.C.
"Further numerical studies in algebraic equations and matrices",
Proc. Roy. Soc. Edin., Vol. 51, 1931, pp 80-90.

46. ABERTH, O.
"Iteration methods for finding all zeros of a polynomial simultaneously",
Math. of Computation, Vol. 27, No. 122, April 1973, pp 339-344.

47. OLIVER, F.W.J.
"The evalaution of zeros of high-degree polynomials",
Phil. Trans. Roy. Soc. Series A, Vol. 244, pp 385-416.

48. PETERS, G. and WILKINSON, J.H.
"Practical problems arising in the solution of polynomial equations",
J. Inst. Maths. Applics., Vol. 8, 1971, pp 16-35.

49. AYRES, F.
"Theory and problems of matrices",
Schaum, McGraw-Hill, 1974.

50. NAG Fortran library routine document,
NAG Fortran Library Manual Mark 10 Vol. 1, Numerical Algorithms Group 1983.

51. SANDEGREN, G.
"QAM and SSB modulation for data transmission over telephone lines including equalizers",
Technical Report No. 104, Department of Telecommunication Theory, Royal Institute of Technology, Stockholm, Sweden, November 1975.

52. LUCKY, R.W.
"Automatic equalization for digital communication",
Bell Sys. Tech. J., 44, pp 547-588, April 1965.

53. LUCKY, R.W.
"Techniques for adaptive equalization equalizers",
Bell Syst. Tech. J., 45, pp 255-286, February 1966.

54. HIRSH, D. and WOLF, W.J.
"A simple adaptive equalizer for efficient data transmission",
IEEE Trans. on Communication Technology, COM-18, pp 5-12, February 1970.

55. NEWHALL, E.E., QUERESHI, S.U.H. and SIMONE, C.F.
"A technique for finding approximate inverse systems and its application to equalization",
IEEE Trans. on Communication Technology, COM-19, pp 1116-1127, December 1971.

56. GUIPA, A.
"Optimum tapped delay line for digital signals",
IEEE Trans. on Communciation, COM-21, pp 277-283, April 1973.

57. LUCKY, R.W., SALZ, J. and WELDON, E.J.
"Principle of data communication",
pp 128-165, McGraw-Hill, 1968.

58. LUCKY, R.W. and RUDIN, H.R.
"An automatic equalizer for general purpose communication channels",
Bell Syst. Tech. J., 46, pp 2179-2208, November 1968.

59. Di TORO, M.J.
"Communication in time-frequency spread media using adaptive equalization",
Proc. IEEE, 56, pp 1653-1769, October 1968.

60. GERSHO, A.
"Adaptive equalization of highly dispersive channels for data transmission",
Bell Syst. Tech. J., 48, pp 55-70, January 1969.

61. RUDIN, H.R.
"A continuously adaptive equalizer for general purpose communication channels",
Bell Syst. Tech. J., 48, pp 1865-1884, July-August 1969.

62. NIESSEN, C.W. and WILLIM, D.K.
"Adaptive equalizer for pulse transmission",
IEEE Trans. on Communication Technology, COM-18, pp 377-395.

63. MARK, J.W. and HAYKIN, S.S.
"Adaptive equalization for digital communication",
Proc. IEE, 188, pp 1711-1720, December 1971.

64. CHANG, R.W.
"A new equalizer structure for fast start-up digital communication",
Bell Syst. Tech. J., 50, pp 1969-2014, July-August 1971.

65. CHANG, R.W. and HO, E.Y.
"On fast start-up data communication systems using pseudo-random training sequences",
Bell Syst. Tech. J., 51, pp 2013-2027, November 1972.

66. MUELLER, K.H.
"A new, fast-converging mean-square algorithm for adaptive equalizers with partial response signalling",
Bell Syst. Tech. J., 54 pp 143-153, January 1975.

67. MARK, J.W. and BUDIHARDJO, P.S.
"Joint optimization of receiver filter and equalizer",
IEEE Trans. on Communications, COM-21, pp 264-267, March 1973.

68. QUERSHI, S.U.H.
"Adjustment of the position of the reference tap of an adaptive equalizer",
IEEE Trans. on Communications, COM-21, pp 1046-1052, September 1973.

69. BUTLER, P. and CONTONI, A.
"Non-iterative automatic equalization",
IEEE Trans. on Communications, COM-23, pp 621-630, June 1975.

70. SCHONFELD, T.J. and SCHWARTZ, M.
"A rapidly converging first-order training algorithm for an adaptive equalizer",
IEEE Trans. on IT, IT-17, pp 431-439, July 1971.

71. SCHONFELD, T.J. and SCHWARTZ, M.
"Rapidly converging second-order tracking algorithm for adaptive equalization",
IEEE Trans. on IT, IT-17, pp 572-579, September 1971.

72. KOSOVYCH, O.S. and PICKHOLTZ, R.L.
"Automatic equalization using a successive over-relaxation iterative technique",
IEEE Trans. on IT,IT-21, pp 51-58, January 1975.

73. GITLIN, R.D. and MAGEE, F.R.
"Self-orthogonalization adaptive equalization algorithms",
IEEE Trans. on Comms. COM-25, pp 666-672, January 1977.

74. SALOMONSSON, G.
"An equalizer with feedback filter",
Ericsson Technics, 28, pp 58-101, 1972.

75. KOBAYASI, H. and TANG, D.T.
"A decision-feedback receiver for channel with strong intersymbol interference",
IBM J. Res. Develop. Sept. 1973, pp 413-419.

P 76. BOYD, R.T. and MONDS, F.C.
"Adaptive equalizer for multipath channels",
Electronics Letters, 6, pp 556-558, 20 August 1970.

77. MONSEN, P.
"Adaptive equalization of the slow fading channel",
IEEE Trans. on Comms. COM-22, No. 8, pp 1064-1075, August 1974.

78. TOMLINSON, M.
"Data transmission over telephone lines",
PhD Thesis, Loughborough University of Technology, 1970.

79. TOMLINSON, M.
"New automatic equalizer employing modulo arithmetic",
Electronics Letters, 7, pp 138-139, 25 March 1971.

80. TAYLOR, D.P.
"Nonlinear feedback equalizer employing a soft limiter",
Electronics Letters, 7, pp 265-267, 20 May 1971.

81. GEORGE, D.A., BOWEN, R.R. and STOREY, J.R.
"An adaptive decision feedback equalizer",
IEEE Trans. on Communication Technology, COM-19, pp 281-293, June 1971.

82. UGERBOECK, G.
"Nonlinear equalization of binary signals in Gaussian noise",
IEEE Trans. on Communication Technology, COM-19, pp 1128-1137, December 1971.

83. HARASHIMA, H. and MIYAKAWA, H.
"Matched-transmission technique for channels with intersymbol interference",
IEEE Trans. on Communications, COM-20, pp 774-780, August 1972.

84. MARK, J.W. and BUDIHARDJO, P.S.
"Performance of jointly optimized prefilter-equalizer receivers",
IEEE Trans. on Communications, COM-21, pp 941-945, August 1973.

85. TAYLOR, D.P.
"The estimate feedback equalizer: a suboptimum nonlinear receiver".
IEEE Trans. on Communication, COM-21, pp 979-990, September 1973.

86. MACLEOD, C.J.
"Study of recursive equalizers for data transmission with a comparison of the performance of six systems",
Proc. IEE, Vol. 122, pp 1097-1104, October 1975.

87. FITCH, S.M. and KURZ, L.
"Recursive equalization in data transmission - a design procedure and performance evaluation",
IEEE Trans. on Comms. pp 546-550, May 1975.

88. SALZ, J.
"On mean-square decision feedback equalization and timing phase",
IEEE Trans. on Comms. COM-25, No. 12, pp 1471-1476, December 1977.

89. SPEIDEL, J.
"An automatic decision feedback equalizer with variable optimum step size",
Institut für Netzwerk-und Systemtheorie University of Stuttgart,
Stuttgart, W. Germany.

90. CLARK, A.P. and HUSSEIN, B.A.
"Non-linear equalizers having an improved tolerance to noise",
Radio and Electron. Eng. Vol. 52, No. 3, pp 145-153, March 1982.

91. CLARK, A.P., LEE, L.H. and MARSHALL, R.S.
"Developments of the conventional nonlinear equalizer",
IEE Proc. Vol. 129, Pt. F, No. 2, pp 85-94, April 1982.

92. WHALEN, A.D.
"Detection of signals in noise",
Academic Press, 1971.

93. FORNEY, G.D.
"The Viterbi algorithm",
Proc. IEEE, 61, pp 268-278, March 1973.

94. AZIZ, A. and MOHAMMAD, Q.G.
"On the zeros of a certain class of polynomials and related analytic functions",

95. BOZIC
"Kalman filtering",
Edward Arnold, 1981.

96. TRETTER, S.A.
"Introduction to discrete-time signal processing",
Wiley, 1976.

97. NEC Publication
"NEC μPD7720",
NEC Electronics (Europe) GmbH.

APPENDIX A1

```
C
C     COMPUTER SIMULATION OF THE STANDARD AND MODIFIED ALGORITHMS.
C     IT CONTAINS MORE TESTS AND FUNCTIONS THAN IS REQUIRED OF THE
C     ALGORITHMS FOR THE AUTHOR'S BENEFITS. THE READER MIGHT WISH
C     TO CHANGE OR IGNORE SOME.
C     ITS HAS 9 STARTING POINTS FOR ALPHA WHERE -1/ALPHA IS THE
C     ESTIMATE OF THE ROOT OUTSIDE THE UNIT CIRCLE.
      DIMENSION ALPSTI(200),ALPSTJ(200),ALPERR(200),DEL(200)
      DIMENSION ESQSIR(200)
      DOUBLE PRECISION HIJ(2,40),YIJ(2,40),ALPRI(100),ALPRJ(100)
      DOUBLE PRECISION USTI(40),USTJ(40)
      DOUBLE PRECISION ROOTI,ROOTJ,ALPHAI,ALPHAJ
      DOUBLE PRECISION ALPI,ALPJ,AK1,AK2,DELTAI,DELTAJ
      DOUBLE PRECISION YROOTI,YROOTJ
      DOUBLE PRECISION AII(10),AJJ(10)
      DATA AII/0.0D0,0.5D0,0.0D0,0.0D0,-.5D0,
     1 0.3535534D0, 0.3535534D0,-0.3535534D0,-0.3535534D0,0.0D0/
      DATA AJJ/0.0D0,0.0D0,-.5D0,0.5D0,0.0D0,
     1-0.3535534D0, 0.3535534D0, 0.3535534D0,-0.3535534D0,0.0D0/
C
      READ (5,10) IC,NIT,AM,THD,NP,NBRKK,THD2
 10   FORMAT(I2,1X,I3,5X,F7.4,1X,E12.3,4X,I3,1X,I3,1X,E12.3)
      WRITE (1,15) IC,NIT,AM,THD,NP,NBRKK,THD2
 15   FORMAT('IC='I3,3X,'NIT='I4,3X,'AM='F7.4,3X,'THD1='E12.4,3X,
     1'THD2='E12.4,3X,'NP='I4,3X,'NBRK='I4/)
C
      READ (6,30) NH,ROOTI,ROOTJ
 30   FORMAT(I2,1X,F16.12,1X,F16.12)
      READ (6,35) (HIJ(1,I),HIJ(2,I),I=1,NH),QQI,QQJ
 35   FORMAT(2F16.12)
C     REAL & IMAGINARY PARTS OF S.I.R.OF CHANNEL ARE READ INTO
C     THE FIRST AND SECOND ROWS OF HIJ RESPECTIVELY.
C     (QQI,QQJ)IS THE FIRST TERM OF Y WHEN ROOT(RROTI,RROTJ) HAS BEEN
C     REMOVED AND REPLACED PERFECTLY.
      WRITE (1,40) NH,QQI,QQJ,ROOTI,ROOTJ
 40   FORMAT(/'NO.OF CHANNEL S.I.R. = 'I3/
     1'QQ   = '2F16.12/'ROOT = '2F16.12/)
      IRR   =0
      NITTOT=0
      WRITE(1,36)
 36   FORMAT('ENTER ALTHD,IB')
C     ALTHD = LIMIT BELOW WHICH ROOTS ARE NOT REQUIRED(E.G. =1)
C     IB    = NO. OF TIMES ROOT IS ALLOWED TO EXCEED ALTHD(E.G. =1)
      READ (1,*)ALTHD,IB
C
C
C     READ IN CHANNEL S.I.R. FOR THE CASE OF PERFECT TRACKING OF ROOT.
C     THIS DATA IS USED FOR WORKING OUT OF MSE IN CHANNEL S.I.R.
 42   READ (6,30) NH,YROOTI,YROOTJ
      READ (6,35) (YIJ(1,I),YIJ(2,I),I=1,NH),YQQI,YQQJ
      DO 45 I=1,200,1
       ALPSTI(I)=0.0
       ALPSTJ(I)=0.0
       ALPERR(I)=0.0
       ESQSIR(I)=0.0
       DEL   (I)=0.0
 45    CONTINUE
```

```
      DO 47 I=1,40,1
      USTI(I)=0.0D0
      USTJ(I)=0.0D0
 47   CONTINUE
      ALPHAI=0.0D0
      ALPHAJ=0.0D0
      ALPI  =0.0D0
      ALPJ  =0.0D0
      IY    =0
      ICHK2 =0
      IFLG1 =1
      IFLG2 =0
      IREF  =0
      NH1   =NH-1
C
C
C     TEST THE COMPONENTS OF CHANNEL S.I.R.
      CALL TESTCH (HIJ,NH,IY,IZ)
C     IF Y0 IS BIGGER THAN Y1 THEN IT WOULD GIVE ALPHA > 1
C     FOR THE (0.0+J0.0) STARTING POINT.
C     THE VALUE NBRK IS READ FROM INPUT FILE AND IT SHOULD BE SET TO 1.
      IREF=IREF+1
      IF (IREF .GT. 9) GOTO 555
      CALL NEWST (NBRKK,AM,IY,NBRK,AC,K,AK1,AK2,ALPHAI,ALPHAJ,
     1              ALPI,ALPJ,AII,AJJ,IREF)
C     ALPHAI & ALPHAJ READ IN.
CCCCC WRITE (1,95) IREF,ALPHAI,ALPHAJ
 95   FORMAT(/'***** STARTING POINT NO.  ('I1')'2F10.1)
C
C
C
      DO 500 I=1,NIT,1
      CALL EQR (1,NH,HIJ,ALPHAI,ALPHAJ,USTI,USTJ)
      ALPSTI(I)=ALPHAI
      ALPSTJ(I)=ALPHAJ
      ALPERR(I)=(ROOTI-ALPHAI)**2+(ROOTJ-ALPHAJ)**2
C     WORK OUT THE ERROR IN CHANNEL S.I.R.
      ESQSIR(I)=0.0
      DO 300 J=1,NH1,1
      J1=J+1
      ESQSIR(I)=ESQSIR(I)+(USTI(J1)-YIJ(1,J))**2+(USTJ(J1)-YIJ(2,J))**2
 300  CONTINUE
      ESQSIR(I)=ESQSIR(I)+USTI(1)**2+USTJ(1)**2
C
      CALL ALG (ALPHAI,ALPHAJ,DELTAI,DELTAJ,USTI,USTJ,NH)
      DEL(I)=DELTAI**2+DELTAJ**2
      IF(DEL(I).LT.THD .AND. ALPM.LT.ALTHD) GOTO 559
      IF(ALPM.GE.ALTHD) GOTO 320
      AC=AM
      ALPHAI=ALPHAI+AC*DELTAI
      ALPHAJ=ALPHAJ+AC*DELTAJ
CCCCC INCLUDE THE FOLLOWING LINES FOR THE MODIFIED ALGORITHM
CCCCC START WITH AN AVERAGING ROUTINE FOR THE FIRST NBRKK ITERATIONS
CCCCC IF (I-(IFLG1).GE.NBRKK .OR. DEL(I).LE.THD2) GOTO 315
CCCCC K=K+1
CCCCC AK1=1.0D0/FLOAT(K)
CCCCC AK2=1.0D0-AK1
CCCCC ALPHAI=AK2*ALPI+AK1*ALPHAI
CCCCC ALPHAJ=AK2*ALPJ+AK1*ALPHAJ
CCCCC ALPI=ALPHAI
CCCCC ALPJ=ALPHAJ
 315  ALPM=SQRT(ALPHAI**2+ALPHAJ**2)
      IF (ALPM.LT.ALTHD.AND.(I-IFLG1).LT.NP) GOTO 500
C     ALPHA IS GREATER THAN ALTHD
 320  ICHK2=ICHK2+1
      IF(ICHK2.LT.IB.AND.(I-IFLG1).LT.NP) GOTO 500
CCCCC WRITE (1,325) ALPHAI,ALPHAJ,I
 325  FORMAT('ALPHAI = 'F10.5,2X,'ALPHAJ = 'F10.5,2X,' AT NI = 'I4/)
C     MODULES OF ALPHA >ALTHD. MOVE TO THE NEXT STARTING POINT.
      IREF =IREF+1
      ICHK2 =0
      IFLG1 =I
      IF (IREF .GT. 9) GOTO 555
      CALL NEWST (NBRKK,AM,IY,NBRK,AC,K,AK1,AK2,ALPHAI,ALPHAJ,
     1              ALPI,ALPJ,AII,AJJ,IREF)
      GOTO 500
C
C
 500  CONTINUE
C
C
C
C
      GOTO 559
 555  WRITE (1,557)
 557  FORMAT (/'ALL 9 STARTING POINTS HAVE BEEN EXHAUSTED'/)
 559  NIT1 =I
      NITTOT=NITTOT+NIT1
      WRITE (1,568) NIT1
 568  FORMAT(/'--------- LOOP FINISHED ---------'/
     1'---------AFTER'I4' ITER.--------'/)
      WRITE (1,15) IC,NIT,AC,THD,NP,NBRK
CCCCC WRITE (1,40) NH,QQI,QQJ,ROOTI,ROOTJ
CCCCC CALL PRT2 (HIJ,NH)
CCCCC WRITE (1,570)
 570  FORMAT(/1X,'ITER.',6X,'REAL',8X,'IMAG.',9X,'MOD.',7X,'RMSE OF',
     17X,'SIR ERR',7X,'U(M-1)'/2X,'NO.',6X,'ALPHA',8X,'ALPHA',
     18X,'ALPHA',8X,'ALPHA',10X,'DB'/)
CCCCC WRITE (1,571) NIT1
      WRITE (7,571) NIT1,NIT1
 571  FORMAT('ROOT(',I2,')'/I3)
      DO 620 J=1,NIT1,1
      J1=J-1
      ALPMOD=SQRT(ALPSTI(J)**2+ALPSTJ(J)**2)
      ALPERR(J)=10*ALOG10(ALPERR(J))
      ESQSIR(J)=10*ALOG10(ESQSIR(J))
CCCCC WRITE (1,617) J1,ALPSTI(J),ALPSTJ(J),ALPMOD,ALPERR(J),ESQSIR(J),
CCCCC1DEL(J)
      WRITE (7,617) J1,ALPSTI(J),ALPSTJ(J),ALPMOD,ALPERR(J),ESQSIR(J),
     1DEL(J)
 620  CONTINUE
 617  FORMAT(I4,4X,3(F9.5,4X),2(F9.2,4X),E12.4)
      IF (IFLG2 .EQ. 9  .OR.  IREF .GT. 9) GOTO 800
C
```

```
      CALL EQR1(NH,ALPHAI,ALPHAJ,USTI,USTJ)
C     SHIFT ALL ELEMENS IN USTI & USTJ 1 PLACE TO THE LEFT
      DO 580 J=1,NH,1
      J1=J+1
      USTI(J)=USTI(J1)
      USTJ(J)=USTJ(J1)
 580  CONTINUE
      USTI(NH+1)=0.0D0
      USTJ(NH+1)=0.0D0
CCCCC WRITE (1,590)
 590  FORMAT(/'THIS IS THE Y-SEQUENCE ACTUALLY USED FOR THE TRACKING OF
     1THE NEXT ROOT'/)
CCCCC CALL PRT1 (USTI,USTJ,NH)
C
      IFLG2=0
      IRR=IRR+1
      ALPRI(IRR)=ALPHAI
      ALPRJ(IRR)=ALPHAJ
      DO 655 J=1,40,1
      HIJ(1,J)=0.0D0
      HIJ(2,J)=0.0D0
      YIJ(1,J)=0.0D0
      YIJ(2,J)=0.0D0
 655  CONTINUE
C     SET THE REDUCED CHANNEL S.I.R. TO THE NEW Y-SEQUENCE.
      DO 660 J=1,NH,1
      HIJ(1,J)=USTI(J)
      HIJ(2,J)=USTJ(J)
 660  CONTINUE
      ROOTI=YROOTI
      ROOTJ=YROOTJ
      QQI  =YQQI
      QQJ  =YQQJ
CCCCC WRITE (1,700)
 700  FORMAT(//)
      GOTO 42
 800  WRITE (1,805) NITTOT
 805  FORMAT(/'TOTAL NUMBER OF ITERATIONS = 'I6/)
      WRITE (7,806) NITTOT
 806  FORMAT(I6)
      REWIND 6
C     MODIFY SUBROUTINE TRNFIL SO THAT IT SKIPS THE FIRST LINE OF DATA
C     FIL ON FORTRAN CHANNEL 6.
      READ(6,808) NH
 808  FORMAT(I2)
C     SET IC=2 SO THAT THE SIGNAL AT OUTPUT OF EQUALISER IS USED
C     TO COMPARE WITH THAT FROM LINEAR FILTER+CHANNEL.
      CALL TRNFIL (2,HIJ,NH,ALPRI,ALPRJ,IRR,20,100,10)
      CALL EXIT
      END
C
C
C
C
```

```
      SUBROUTINE EQR(IEQR,NH,HIJ,ALPHAI,ALPHAJ,USTI,USTJ)
C     THIS SUBROUTINE IS USED TO PASS A SEQUENCE THROUGH THE
C     CONBINATION OF THE 2-TAP F/F AND 1-TAP F/B FILTERS.
C     THE CONFIGURATION IS SET UP FOR THE ROOT
C     -1/(ALPHA) WHICH LIES OUTSIDE THE UNIT CIRCLE.
C
C      IEQR   =1   ONLY F/B FILTER
C             =2   BOTH FILTERS
C      NH     =    NO. OF COMPONENTS IN S.I.R. OF CHANNEL.
C      HIJ    =    2-D ARRAY CONTAINING THE S.I.R. OF CHANNEL.
C      ALPHAI =    REAL PART OF ALPHA.
C      ALPHAJ =    IMAG. PART OF ALPHA.
C      USTI   =    1-D ARRAY HOLDING REAL PARTS OF OUTPUT AT EQRII.
C      USTJ   =    1-D ARRAY HOLDING IMAG. PARTS OF OUTPUT AT EQRII.
C
      DOUBLE PRECISION ALPHAI,ALPHAJ,HIJ(2,40),USTI(40),USTJ(40)
      DOUBLE PRECISION CCALPI,CCALPJ,QI(40),QJ(40)
      DOUBLE PRECISION PI,PJ,PPI,PPJ
      PI =0.0D0
      PJ =0.0D0
      PPI=0.0D0
      PPJ=0.0D0
      DO 5 I=1,40,1
      QI(I)=0.0D0
      QJ(J)=0.0D0
 5    CONTINUE
      IF (IEQR .EQ. 2) GOTO 10
      CCALPI=0.0D0
      CCALPJ=0.0D0
      GOTO 20
 10   CCALPI =ALPHAI
      CCALPJ=-ALPHAJ
C     WORK OUR Q.
 20   QI(1)=HIJ(1,1)
      QJ(1)=HIJ(2,1)
      DO 30 J=1,NH,1
      J1=J+1
      QI(J1)=HIJ(1,J1)+(HIJ(1,J)*CCALPI-HIJ(2,J)*CCALPJ)
      QJ(J1)=HIJ(2,J1)+(HIJ(1,J)*CCALPJ+HIJ(2,J)*CCALPI)
 30   CONTINUE
C     WORK OUT P
      PI=0.0
      PJ=0.0
      N1=NH+1
      N2=NH+2
      DO 40 J=1,N1,1
      J1=N2-J
      PPI=QI(J1)-(ALPHAI*PI-ALPHAJ*PJ)
      PPJ=QJ(J1)-(ALPHAI*PJ+ALPHAJ*PI)
      PI=PPI
      PJ=PPJ
      USTI(J1)=PPI
      USTJ(J1)=PPJ
 40   CONTINUE
      RETURN
      END
```

```
C
      SUBROUTINE EQR1(NH,ALPHAI,ALPHAJ,AI,AJ)
C     THIS SUBROUTINE IS USED TO PASS A SEQUENCE THROUGH THE
C     2-TAP F/F FILTER .THE EFFECT WOULD BE TO INSERT
C     A ROOT AT -(ALPHA)*. NOTE THAT -1/(ALPHA) IS THE ROOT OUTSIDE
C     THE UNIT CIRCLE.
C
C     NH      =  NO. OF COMPONENTS IN S.I.R. OF CHANNEL.
C     ALPHAI  =  REAL  PART OF ALPHA.
C     ALPHAJ  =  IMAG. PART OF ALPHA.
C     AI      =  1-D ARRAY HOLDING REAL  PARTS OF OUTPUT FROM FILTER.
C     AJ      =  1-D ARRAY HOLDING IMAG. PARTS OF OUTPUT FROM FILTER.
C
      DOUBLE PRECISION ALPHAI,ALPHAJ,CCALPI,CCALPJ
      DOUBLE PRECISION AI(40),AJ(40),BI(40),BJ(40)
      CCALPI =ALPHAI
      CCALPJ=-ALPHAJ
      BI(1)=AI(1)
      BJ(1)=AJ(1)
      DO 10 J=1,NH,1
      J1=J+1
      BI(J1)=AI(J1)+(AI(J)*CCALPI-AJ(J)*CCALPJ)
      BJ(J1)=AJ(J1)+(AI(J)*CCALPJ+AJ(J)*CCALPI)
 10   CONTINUE
      N=NH+1
      DO 20 I=1,N,1
      AI(I)=BI(I)
      AJ(I)=BJ(I)
 20   CONTINUE
      RETURN
      END
C
C
C
C
      SUBROUTINE TRNFIL (IC,DIJ,NH,ALPRI,ALPRJ,IRR,IL1,IL2,ILN)
C     THIS SUBROUTINE IS USED TO WORK OUT THE EQUIVALENT LINEAR FILTER.
C     IT ALSO WORKS OUT THE PARAMETERS PSI1,PSI2 OR PSI3,PSI4 DEPENDING
C     ON THE INPUT CONDITIONS.
C     HIJ  =  ARRAY HOLDS THE CHANNEL S.I.R..
C     YIJ  =  ARRAY HOLDS THE CHANNEL S.I.R. WHEN ALL ROOTS
C             HAD BEEN TRACKED.
C     NTAP =  NO. OF TAPS REQUIRED IN LINEAR FILTER (MAX=110).
C     IC   =  1  USE SIR OF CHANNEL+FILTER WHEN ALL ROOTS ARE
C             PROCESSED PERFECTLY.
C     IC   =  2  USE SIR OF CHANNEL+FILTER FROM OUTPUT OF EQUALISER.
C     IL1  =  INITIAL NO. OF FILTER TAPS
C     IL2  =  FINAL NO. OF FILTER TAPS
C     ILN  =  NO. OF INCREMENTS FROM IL1 TO IL2
C
      DOUBLE PRECISION HIJ(2,40),YIJ(2,40),YYIJ(2,151),CONV(2,200)
      DOUBLE PRECISION QI(151),QJ(151),USTI(151),USTJ(151),CCALPI,CCALPJ
      DOUBLE PRECISION ALPRI(IRR),ALPRJ(IRR),DIJ(2,40)
      DOUBLE PRECISION ALPHAI,ALPHAJ,PI,PJ,PPI,PPJ,ERR,ERR1
```

```
C
      READ (6,30) (HIJ(1,J),HIJ(2,J),J=1,NH)
 30   FORMAT (2F16.12)
CCCCC WRITE (1,40)
 40   FORMAT (/'THE IS THE S.I.R. OF THE ORIGINAL CHANNEL')
CCCCC CALL PRT2 (HIJ,30)
      IF (IC .EQ. 1) GOTO 10
      WRITE (1,2)
 2  ? FORMAT(/'COMPARING THE RESULTANT SIR OF CHANNEL+FILTER WITH WHAT
     1THE RECEIVER THINKS IT IS'/'CHANNEL SIR (RECEIVER VERSION)'/)
      DO 6 I1=1,2,1
      DO 4 I2=1,40,1
      YIJ(I1,I2)=DIJ(I1,I2)
 4    CONTINUE
 6    CONTINUE
CCCCC CALL PRT2 (YIJ,30)
      GOTO 55
 10   READ (6,30) (YIJ(1,J),YIJ(2,J),J=1,NH)
      WRITE (1,50)
 50   FORMAT(/'COMPARING THE RESULTANT SIR OF CHANNRL+FILTER WITH WHAT
     1IT SHOULD IDEALLY BE.'/'CHANNEL SIR WHEN ALL ROOTS ARE
     1 TRACKED PERFECTLY'/)
CCCCC CALL PRT2 (YIJ,30)
 55   WRITE (1,60) IRR
 60   FORMAT('NO. OF ROOTS TRACKED'I6/)
      CALL PRT1 (ALPRI,ALPRJ,IRR)
      DO 300 NTAP=IL1,IL2,ILN
      WRITE (1,70) NTAP
 70   FORMAT(' NO. OF TAPS REQUIRED IN THE LINEAR FILTER IS 'I6)
      NTAP1=NTAP-1
      DO 80 I=1,151,1
      YYIJ(1,I)=0.0D0
      YYIJ(2,I)=0.0D0
 80   CONTINUE
      IP=40+NTAP
      YYIJ(1,IP)=1.0D0
      NH1=NH-1
      DO 200 I=1,IRR,1
      ALPHAI=ALPRI(I)
      ALPHAJ=ALPRJ(I)
C     INPUT OF F/F AND F/B FILTERS CONBINATION
      CCALPI= ALPHAI
      CCALPJ=-ALPHAJ
C     WORK OUR Q.
      QI(1)=YYIJ(1,1)
      QJ(1)=YYIJ(2,1)
      DO 90 J=1,150,1
      J1=J+1
      QI(J1)=YYIJ(1,J1)+(YYIJ(1,J)*CCALPI-YYIJ(2,J)*CCALPJ)
      QJ(J1)=YYIJ(2,J1)+(YYIJ(1,J)*CCALPJ+YYIJ(2,J)*CCALPI)
 90   CONTINUE
C     OUTPUT OF F/F FILTER
      PI=0.0
      PJ=0.0
      DO 100 J=1,151,1
      J1=152-J
      PPI=QI(J1)-(ALPHAI*PI-ALPHAJ*PJ)
      PPJ=QJ(J1)-(ALPHAI*PJ+ALPHAJ*PI)
      PI=PPI
      PJ=PPJ
      USTI(J1)=PPI
      USTJ(J1)=PPJ
 100  CONTINUE
C     OUTPUT OF F/F AND F/B FILTERS COMBINATION
      DO 130 J=1,NTAP,1
      J1=40+J
      J2=(IP+1)-NTAP+J
      YYIJ(1,J1)=USTI(J2)
      YYIJ(2,J1)=USTJ(J2)
 130  CONTINUE
CCCCC WRITE (1,140)
 140  FORMAT(/'THIS IS THE S.I.R. OF LINEAR FILTER (YYIJ)')
CCCCC CALL PRT2(YYIJ,151)
      DO 150 J=1,151,1
      QI(J)=0.0D0
      QJ(J)=0.0D0
 150  CONTINUE
 200  CONTINUE
CCCCC WRITE (1,140)
CCCCC CALL PRT2(YYIJ,151)
C
C
C     CONVOLUTION BETWEEN YYIJ AND HIJ TO WORK OUT THE RESULTANT
C     S.I.R. OF CHANNEL+LINEAR FILTER.
C     LINNING UP HIJ AND YYIJ FOR CONVOLUTION.
      DO 210 J=1,NTAP,1
      J1=40+J
      YYIJ(1,J)=YYIJ(1,J1)
      YYIJ(2,J)=YYIJ(2,J1)
      YYIJ(1,J1)=0.0D0
      YYIJ(2,J1)=0.0D0
 210  CONTINUE
CCCCC WRITE (1,140)
CCCCC CALL PRT2(YYIJ,NTAP)
      DO 220 J =1,200,1
      CONV(1,J)=0.0D0
      CONV(2,J)=0.0D0
 220  CONTINUE
      DO 240 J =1,NH,1
      DO 230 J1=1,NTAP,1
      K=J1+J-1
      CONV(1,K)=CONV(1,K)+HIJ(1,J)*YYIJ(1,J1)-HIJ(2,J)*YYIJ(2,J1)
      CONV(2,K)=CONV(2,K)+HIJ(1,J)*YYIJ(2,J1)+HIJ(2,J)*YYIJ(1,J1)
 230  CONTINUE
 240  CONTINUE
      NNPP=NH+NTAP-1
CCCCC WRITE (1,250)
 250  FORMAT(/'THE IS THE RESULTANT S.I.R OF CHANNEL+FILTAR')
CCCCC CALL PRT2(CONV,NNPP)
```

```
C     WORKING OUT THE ERROR (DISTANCE) BETWEEN THE RESULTANT S.I.R.
C     OF CHANNEL+FILTER
C     OBTAINED IN HERE AND THAT OBTAINED WHEN THE ROOTS HAD BEEN
C     TRACKED PERFECTLY.
      ERR1=0.0D0
      DO 260 J=1,NTAP1,1
      ERR1=ERR1+CONV(1,J)**2+CONV(2,J)**2
 260  CONTINUE
      ERR=0.0D0
      DO 270 J=1,NH,1
      J1=NTAP1+J
      ERR=ERR+(CONV(1,J1)-YIJ(1,J))**2+(CONV(2,J1)-YIJ(2,J))**2
 270  CONTINUE
      ERR1=ERR1+ERR
      XERR1=10*DLOG10(ERR1)
      XERR =10*DLOG10(ERR)
      WRITE (1,280) XERR,ERR,XERR1,ERR1
 280  FORMAT('******* ERROR  ERR  (NH)  'F12.6'DB ('D25.15')******'/
     1'******* ERROR  ERR1(NTAP) 'F12.6'DB ('D25.15')******')
      WRITE (7,290) XERR,XERR1
 290  FORMAT (F10.2)
 300  CONTINUE
      RETURN
      END
C
C
C
C
      SUBROUTINE ALG (ALPHAI,ALPHAJ,DELTAI,DELTAJ,USTI,USTJ,NH)
C     THIS SUBROUTINE WORKS OUT SIGNAL 'DELTA' FOR THE UPDATING OF
C     THE NEXT ESTIMATE OF ALPHA.
C     MORE COMPLICATED EXPRESSION IS USED.
C     ALPHAI & ALPHAJ = -VE OF THE RECIPROCAL OF THE ROOT.
C     DELTAI & DELTAJ = SIGNAL FOR THE UPDATE OF THE NEXT ALPHA.
C     USTI   & USTJ   = CONTAINS SIGNALS U(M-1),U(M),U(M+1),U(M+2),
C                       U(M-3),U(M-4).......ETC.
C     NH = NO. OF COMPONENTS IN CHANNEL S.I.R.
      DOUBLE PRECISION ALPHAI,ALPHAJ,AI,AJ,DELTAI,DELTAJ
      DOUBLE PRECISION UTI,UTJ,UU,U1I,U1J,PI,PJ
      DOUBLE PRECISION USTI(40),USTJ(40)
      NH1=NH-1
      AI =-ALPHAI
      AJ =-ALPHAJ
      PI =USTI(NH+1)
      PJ =USTJ(NH+1)
      DO 10 J=1,NH1,1
      J1 =NH-J+1
C     J1 STARTS FROM NH TO 2 INSIDE THIS LOOP
      UTI=PI*AI-PJ*AJ
      UTJ=PI*AJ+PJ*AI
      PI =UTI+USTI(J1)
      PJ =UTJ+USTJ(J1)
 10   CONTINUE
      UTI=PI
      UTJ=PJ
```

```
C     UTI& UTJ EQUALS TO THE EXPRESSION
C     U(M)-U(M+1)*ALPHA+U(M+2)*ALPHA**2-U(M+3)*ALPHA**3.......ETC.
C     WORK OUT THE RESULT OF U(M-1)/(UTI+JUTJ).
      UU=UTI**2+UTJ**2
      U1I=USTI(1)
      U1J=USTJ(1)
      DELTAI=(U1I*UTI+U1J*UTJ)/UU
      DELTAJ=(U1J*UTI-U1I*UTJ)/UU
      RETURN
      END
C
C
C
C
      SUBROUTINE TESTCH (HIJ,N,IY,IZ)
C     THIS SUBROUTINE PERFORMANCE TESTS ON THE RELATIVE MAGNITUDES
C     OF THE COMPONENTS IN THE CHANNEL S.I.R.
C     HIJ = CONTIAINS THE REAL AND IMAGINARY PARTS OF THE CHANNEL
C           S.I.R.
C     N   = NO. OF COMPONENTS IN THE CHANNEL S.I.R.
C     IY  = 1 IF YO > Y1
C         = 0 IF YO < Y1
C     IZ  = 1 IF YO IS THE LARGEST COMPONENT.
C         = 0 IF YO IS NOT THE LARGEST COMPONENT .
C
      DOUBLE PRECISION HIJ(2,N)
C     TEST TO SEE IF YO IS THE BIGGEST COMPONENT IS TH S.I.R RESPONSE.
      NH1=N-1
      IZ  =1
      A1=HIJ(1,1)**2+HIJ(2,1)**2
      DO 10 I=1,NH1,1
      I1=I+1
      A2=HIJ(1,I1)**2+HIJ(2,I1)**2
      IF (A1 .LE. A2) IZ=0
 10   CONTINUE
C     IZ REMAINS 1 IF YO IS THE BIGGEST COMPONENT.
      IF (IZ .EQ. 1) WRITE (1,12)
 12   FORMAT ('YO IS THE BIGGEST COMPONENT IN THE CHANNEL S.I.R.')
      IY=0
C     TEST TO SEE IF YO < Y1.
      YOM=HIJ(1,1)**2+HIJ(2,1)**2
      Y1M=HIJ(1,2)**2+HIJ(2,2)**2
      IF(YOM .LT. Y1M) GOTO 20
C     IY IS SET TO 1 IF YO > Y1.
      IY=1
      WRITE (1,15)
 15   FORMAT('YO   >  Y1   IS DETECTED')
 20   RETURN
      END
C
C
C
C
```

```
      SUBROUTINE NEWST (NBRKK,AM,IY,NBRK,AC,K,AK1,AK2,ALPHAI,ALPHAJ,
     1                  ALPI,ALPJ,AII,AJJ,IREF)
C     THIS SUBROUTINE PRESETS CONDITOINS FOR A NEW STARTING POINT.
C     IT ASSIGNS ALPHA TO THE NEXT STARTING VALUE STORED IN AII&AJJ.
      DOUBLE PRECISION AK1,AK2
      DOUBLE PRECISION ALPHAI,ALPHAJ,ALPI,ALPJ
      DOUBLE PRECISION AII(10),AJJ(10)
      ALPHAI=AII(IREF)
      ALPHAJ=AJJ(IREF)
      ALPI  =ALPHAI
      ALPJ  =ALPHAJ
      IF (IY .EQ. 0) GOTO 10
      NBRK=1
      AC   =1.0
      K    =1
      AK1  =0.0D0
      AK2  =0.0D0
      GOTO 20
   10 NBRK=NBRKK
      AC   =AM
      K    =0
   20 RETURN
      END
C
C
C
C
      SUBROUTINE PRT1 (AR,AC,N)
C     THIS SUBROUTINE PRINTS N COMPLEX NUMBERS STORED IN ARRAYS
C     AR & AC AND THEIR MODULI.
C     AR = ARRAY CONTAINS THE REAL PARTS OF THE COMPLEX NUMBERS.
C     AC = ARRAY CONTAINS THE IMAG PARTS OF THE COMPLEX NUMBERS.
C     N  = NO. OF COMPLEX NUMBERS REQUIRED PRINTING.
      DOUBLE PRECISION AR(N),AC(N),AA
      WRITE (1,10)
   10 FORMAT(8X,'REAL PART',7X,'IMAG.PART',9X,'MODULES')
      DO 250 I=1,N,1
      AA=SQRT(AR(I)**2+AC(I)**2)
      WRITE (1,260) I,AR(I),AC(I),AA
  250 CONTINUE
  260 FORMAT(I3,1X,F16.11,F16.11,F16.11)
      RETURN
      END
C
C
C
C
      SUBROUTINE PRT2 (HIJ,N)
C     THIS SUBROUTINE PRINTS N COMPLEX NUMBERS STORED IN ARRAY HIJ
C     AND THEIR MODULI.
C     HIJ(1,N) = ARRAY CONTAINS THE REAL PARTS OF THE COMPLEX NUMBERS.
C     HIJ(2,N) = ARRAY CONTAINS THE IMAG PARTS OF THE COMPLEX NUMBERS.
C     N        = NO. OF COMPLEX NUMBERS REQUIRED PRINTING.
      DOUBLE PRECISION HIJ(2,N),AA
      WRITE (1,10)
10    FORMAT(8X,'REAL PART',7X,'IMAG.PART',9X,'MODULES')
      DO 250 I=1,N,1
      AA=SQRT(HIJ(1,I)**2+HIJ(2,I)**2)
      WRITE (1,260) I,HIJ(1,I),HIJ(2,I),AA
250   CONTINUE
260   FORMAT(I3,1X,F16.8,F16.8,F16.8)
      RETURN
      END
```