

Portland State University

**PDXScholar**

---

Dissertations and Theses

Dissertations and Theses

---

1-14-1994

# Adaptive Notch Filter

Yuchen Huang

*Portland State University*

Follow this and additional works at: [https://pdxscholar.library.pdx.edu/open\\_access\\_etds](https://pdxscholar.library.pdx.edu/open_access_etds)



Part of the [Electrical and Computer Engineering Commons](#)

**Let us know how access to this document benefits you.**

---

## Recommended Citation

Huang, Yuchen, "Adaptive Notch Filter" (1994). *Dissertations and Theses*. Paper 4802.


<https://doi.org/10.15760/etd.6686>

This Thesis is brought to you for free and open access. It has been accepted for inclusion in Dissertations and Theses by an authorized administrator of PDXScholar. Please contact us if we can make this document more accessible: [pdxscholar@pdx.edu](mailto:pdxscholar@pdx.edu).

## THESIS APPROVAL

The abstract and thesis of Yuchen Huang for the Master of Science in Electrical Engineering were presented January 14, 1994, and accepted by the thesis committee and the department.

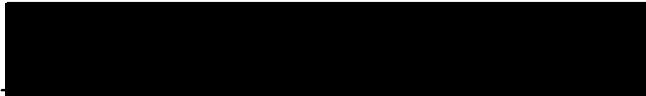
### COMMITTEE APPROVALS:

  
Y. C. Jenq, Chair

  
Andrew M. Fraser

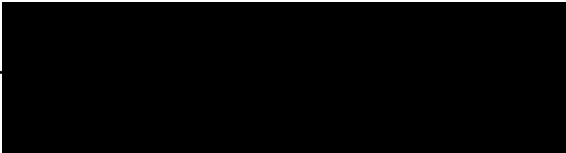
  
Leonard G. Swanson  
Representative of the Office of Graduate Studies

### DEPARTMENT APPROVAL:

  
Rolf Schaumann, Chair  
Department of Electrical Engineering

\*\*\*\*\*

ACCEPTED FOR PORTLAND STATE UNIVERSITY BY THE LIBRARY

by  on March 2, 1994

## ABSTRACT

An abstract of the thesis of Yuchen Huang for the Master of Science in Electrical Engineering presented January 14, 1994.

Title: Adaptive Notch Filter

The thesis presents a new adaptive notch filter (ANF) algorithm that is more accurate and efficient and has a faster convergent rate than previous ANF algorithms. In 1985, Nehorai designed an infinite impulse response (IIR) ANF algorithm that has many advantages over previous ANF algorithms. It requires a minimal number of parameters with constrained poles and zeros. It has higher stability and sharper notches than any ANF algorithm until now. Because of the special filter structure and the recursive prediction error (RPE) method, however, the algorithm is very sensitive to the initial estimate of the filter coefficient and its covariance. Furthermore, convergence to the true filter coefficient is not guaranteed since the error-performance surface of the filter has its global minimum lying on a fairly flat region.

We propose a new ANF algorithm that overcomes the convergence problem. By choosing a smaller notch bandwidth control parameter that makes the error-performance surface less flat, we can more easily detect a global minimum. We also propose a new convergence criterion to be used with the algorithm and a self-adjustment feature to reset the initial estimate of the filter coefficient and its covariance. This results in guaranteed convergence with more accurate results and more efficient computations than previous ANF algorithms.

ADAPTIVE NOTCH FILTER

by

YUCHEN HUANG

A thesis submitted in partial fulfillment of the  
requirements for the degree of

MASTER OF SCIENCE  
in  
ELECTRICAL ENGINEERING

Portland State University  
1994

## ACKNOWLEDGEMENT

I owe my advisor, Professor Yih-Chyun Jenq, a great deal. First, for introducing me to the possibilities of adaptive signal processing. Second, for encouraging and supporting me in my graduate studies. Third, for the guidance and advice he gave me and last, for being a great advisor.

Similarly, I would like to thank my committee members Dr. Andrew M. Fraser and Dr. Leonard G. Swanson for participating on my thesis committee and for taking time out of their busy schedules to read my thesis.

I would also like to acknowledge all the other people at Portland State University that made my time here so enjoyable.

Finally, I owe my deepest gratitude to my parents. Without their support and understanding, none of this would have been possible.

## TABLE OF CONTENTS

	PAGE
ACKNOWLEDGMENT .....	ii
LIST OF TABLES .....	v
LIST OF FIGURES .....	vi
CHAPTER	
I INTRODUCTION .....	1
I.1 Motivation .....	1
I.2 Thesis organization .....	3
II A MINIMAL PARAMETER IIR ADAPTIVE NOTCH FILTER WITH CONSTRAINED POLES AND ZEROS .....	5
II.1 Introduction .....	5
II.2 The Proposed model and its advantages .....	5
II.3 Algorithm derivation .....	9
II.4 Simulation results .....	13
III ERROR-PERFORMANCE SURFACE .....	15
III.1 Introduction .....	15
III.2 Error-performance surface experiment .....	15
III.2.1 Method to plot error-performance surface .....	15
III.2.2 Zero of the system function .....	17
III.3 Error-performance surface analysis .....	18
IV PERFORMANCE ANALYSIS OF THE ORIGINAL ANF .....	25

IV.1	Introduction .....	25
IV.2	Performance analysis of the original ANF .....	25
IV.2.1	Performance vs. different initial conditions .....	26
IV.2.2	Performance vs. different notch bandwidth control parameter .....	30
IV.2.3	Non-convergence performance analysis .....	30
IV.3	Convergence criterion evaluation .....	32
V	A FAST ALGORITHM FOR ADAPTIVE NOTCH FILTERING ..	33
V.1	Introduction .....	33
V.2	Design of a fast algorithm for adaptive notch filtering .....	34
V.2.1	Design of notch bandwidth control parameter .....	34
V.2.2	Design of initial conditions .....	40
V.2.3	Design of convergence criterion .....	41
	V.2.3.1 Relationship between noise-to-signal power and frequency error .....	41
	V.2.3.2 Fast Fourier transform of the input signal .....	46
	V.2.4 Design of criterion to reset initial conditions .....	47
V.3	A fast algorithm for adaptive notch filtering .....	49
V.4	Simulation results .....	53
VI	CONCLUSIONS AND FUTURE WORK .....	57
VI.1	Summary and conclusions .....	57
VI.2	Suggestions for future work .....	58
	REFERENCES .....	60

## LIST OF TABLES

TABLE		PAGE
I	Simulation Results .....	14
II	Convergence Criterion Reference Table .....	44
III	New ANF Results for $f_s/f_1 = 19.1$ .....	54
IV	Original ANF Results for $f_s/f_1 = 19.1$ .....	54
V	New ANF Results for $f_s/f_1 = 7.0$ .....	55
VI	Original ANF Results for $f_s/f_1 = 7.0$ .....	55



## LIST OF FIGURES

FIGURE	PAGE
1. Pole-zero Configuration for a Notch Filter of a Single Sine Wave in an Additive Broadband Process .....	6
2. Error-performance Surface for $\rho = 0.8$ , $SNR = 40$ , $f_s/f_1 = 19.1$ ....	19
3. Error-performance Surface for $\rho = 0.8$ , $SNR = 10$ , $f_s/f_1 = 19.1$ ....	20
4. Error-performance Surface for $\rho = 0.8$ , $SNR = 40$ , $f_s/f_1 = 13.8$ ....	20
5. Error-performance Surface for $\rho = 0.8$ , $SNR = 10$ , $f_s/f_1 = 13.8$ ....	21
6. Error-performance Surface for $\rho = 0.985$ , $SNR = 40$ , $f_s/f_1 = 19.1$ .....	22
7. Error-performance Surface for $\rho = 0.995$ , $SNR = 40$ , $f_s/f_1 = 19.1$ .....	23
8. Frequency Magnitude Response of the Filter with $\rho = 0.8$ .....	34
9. Frequency Magnitude Response of the Filter with $\rho = 0.985$ .....	35
10. Error-performance Surface for $\rho = 0.2$ , $SNR = 40$ , $f_s/f_1 = 19.1$ ....	37
11. Error-performance Surface for $\rho = 0.1$ , $SNR = 40$ , $f_s/f_1 = 19.1$ ....	37
12. Error-performance Surface for $\rho = 0.001$ , $SNR = 40$ , $f_s/f_1 = 19.1$ .....	38
13. Error-performance Surface for $\rho = 0.0001$ , $SNR = 40$ , $f_s/f_1 = 19.1$ .....	38
14. Relationship between Frequency Magnitude Response and Frequency Error .....	42
15. New ANF Algorithm Flowchart .....	50

16.	Transfer Function of the Convergent Filter for	
	One Sine Wave in Additive White Noise .....	56

## CHAPTER I

### INTRODUCTION

#### I.1 MOTIVATION

Detection and elimination of time-varying narrowband or sine wave signals embedded among signals and other broadband noises are the major tasks in adaptive signal processing. Applications are in such fields as communications, radar, sonar, seismology and biomedical engineering.

A practical example is the case in which a sinusoidal power grid pick up of 50 or 60 HZ corrupts a measurement signal. Another example is echo cancellation in long distance telephone communication. An echo is created on a telephone circuit because of impedance mismatches on a network. It can degrade the quality of communication to such a degree that conversation is unintelligible.

While there are numerous other examples, these two are sufficient to illustrate some of the main reasons for the need of adaptive notch filters. In the second example above, the impedance mismatch is usually unknown. That is, the sheer number of local telephone lines that must be accessed effectively prohibits the impedance for any one local line to be accurately matched to the long distance link. Even if the resources were available to match the impedance of each local line to the long distance link, there is still the problem that due to aging, inaccurate component values, moisture, etc., the impedance of each local line may be unknown and time-varying.

Indeed, many applications of adaptive notch filtering involve removing sine wave signals and noise due to physical processes that are unknown and time-varying. These

types of processes represent some of the most difficult problems in transmitting and receiving information. Adaptive notch filtering technique provides an approach for removing distortion in communications, as well as extracting information about unknown physical processes.

Previous adaptive notch filters were designed by Thompson [2], Kung and Rao [3], [4], and Friedlander and Smith [5]. These filters usually satisfied only part of the desired properties of the notch filter. For example, the infinite impulse response (IIR) notch filters of [2] and [4] did not constrain the zeros to be on the unit circle and required  $2n$  parameters, where  $n$  is the number of input sine waves. Since zeros were not constrained to be on the unit circle, their convergent angles were, in general, slightly different from the sine wave frequencies. Another method in [3] and [4] was based on estimating the second-order factors of the IIR transfer function. This method suffers from high nonlinearity in the minimization problem, which complicates the algorithm and deteriorates its performance.

In 1985, Nehorai designed an IIR adaptive notch filter [1] which has many advantages over the previous notch filters. The filter used a minimal number of parameters with constrained poles and zeros, which rendered sharper notches than previously possible. Its special filter structure exhibited high stability, which is different from previous adaptive notch filter (ANF) algorithms. Because of the special filter structure and the corresponding recursive prediction error (RPE) method, however, the algorithm is very sensitive to the initial conditions and can not guarantee convergence. Furthermore, there is not a convergence criterion. Nehorai used the number of recursions as a convergence criterion. In actual situations, however, the input signal is usually unknown and time-varying. Thus, the number of recursions can not be a sufficient convergence criterion.

Based on the RPE method of Nehorai's ANF, we design a new ANF algorithm which solves the convergence problem of the ANF [1]. We also provide a convergence

criterion for the algorithm. Our new algorithm will self-adjust its initial conditions, and has the advantages of faster convergence, more accurate results and computational efficiency than the previous ANF algorithms.

## I.2 THESIS ORGANIZATION

This thesis is divided into six chapters. In Chapter II, we introduce the adaptive IIR notch filter designed by Nehorai [1]. After a brief derivation of the algorithm, examples are given to test the filter. We find that the algorithm has a convergence problem.

The non-convergence behavior of Nehorai's ANF leads us to the analysis of the error-performance surface of the filter in Chapter III. Error-performance surfaces under different signal-to-noise ratios and sampling frequencies are studied. The effect of notch bandwidth control parameter on the performance surface is also explored. We find that the performance surface has a unique shape with its global minimum lying in a fairly flat region. This makes the RPE type of algorithm difficult to find the global minimum.

In Chapter IV, we re-examine the behavior of Nehorai's ANF algorithm given the type of error-performance surface discussed in Chapter III. We find that there are three reasons that make the convergence of the algorithm difficult. First, the special filter structure makes a unique type of performance surface. This type of performance surface is not quadratic. Finding a global minimum on such a surface can be very difficult. Second, as the number of recursions increases, the value of notch bandwidth control parameter  $\rho$  increases. This causes the number of local minima on the performance surface to increase. Convergence to a global minimum becomes even more difficult. Third, the algorithm is very sensitive to the initial conditions of filter coefficient estimate and the covariance of this estimate.

Chapter V presents a new ANF algorithm with simulation results. The new ANF is designed to overcome the drawbacks of the ANF in [1]. By starting the algorithm with

a smaller  $\rho$ , the sensitivity of the RPE type algorithm will be improved. This makes the filter coefficient quickly approach the convergent value. By using the noise-to-signal power, the algorithm can self-adjust to reset its initial estimate of filter coefficient and the covariance of this initial estimate to overcome the non-convergence behavior of the ANF in [1]. A convergence criterion is also provided. Simulation results show that the new ANF algorithm has a faster convergent rate with more accurate results than previous ANF algorithms. Chapter VI concludes the thesis.

## CHAPTER II

### A MINIMAL PARAMETER IIR ADAPTIVE NOTCH FILTER WITH CONSTRAINED POLES AND ZEROS

#### II.1 INTRODUCTION

The very difficult problem of eliminating time-varying narrowband or sine wave signals has led us to the study of adaptive notch filter techniques. Previous adaptive notch filters designed in [2], [3], [4] and [5] usually satisfied only part of the desired properties of the notch filter. They either did not have an accurate convergent angle or suffered from high nonlinearity in the minimization problem. From these previous adaptive notch filters, the IIR ANF designed by Nehorai had more advantages than the others. It requires a minimal number of parameters with constrained zeros and poles and uses a time-varying notch bandwidth control parameter to render sharper notches than before. It exhibits high stability and the filter structure is the simplest of the previous ANF structures. In the reminder of this chapter, we will first describe the ANF structure [1] and the advantages of this filter in comparison to other ANFs. Next, a brief derivation of the algorithm will be given. We will conclude by showing the non-convergence simulation results of the algorithm.

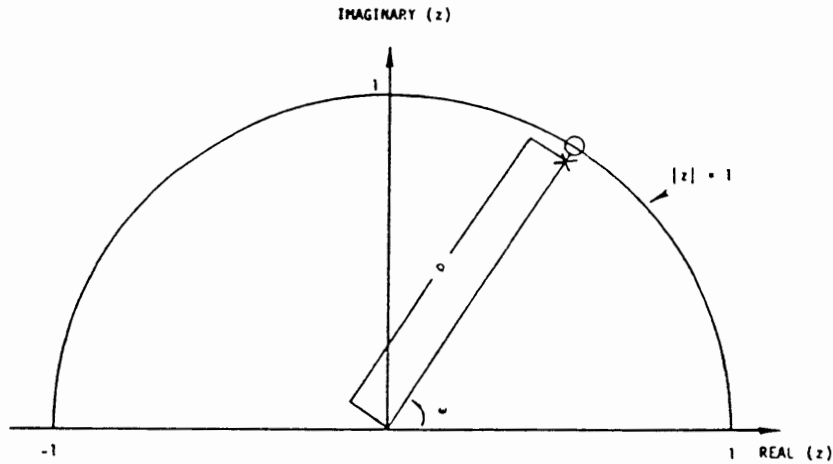
#### II.2 THE PROPOSED MODEL AND ITS ADVANTAGES

In 1985, Nehorai [1] designed an adaptive notch filter of infinite impulse response (IIR) for elimination of multiple unknown sine waves in a broadband signal. The desirable pole-zero configuration of such a filter is schematically illustrated in Fig. 1 for the

special case of a single sine wave in an additive broadband signal (only the positive imaginary part is shown for convenience). As illustrated in the figure, a desirable property of the notch filter is that the zeros of its transfer function will lie on the unit circle. A necessary condition for a polynomial to satisfy this property is that its coefficients have a mirror symmetric form, which can be written as

$$A(z^{-1}) = 1 + a_1 z^{-1} + \cdots + a_n z^{-n} + \cdots + a_1 z^{-2n+1} + z^{-2n} \quad (2.1)$$

Here  $z$  is a complex variable.



**Figure 1.** Pole-zero configuration for a notch filter of a single sine wave in an additive broadband process.

From Fig. 1, we can see that the second requirement of the notch filter is that its poles be inside the unit circle at the same angles and as close as possible to the zeros. This can be achieved using filter denominators of the general form  $A(\rho z^{-1})$ , where  $\rho$  is a positive real number close to, but smaller than 1. To see this, note that if  $z^{2n} A(z^{-1})$  has a zero in  $z_i$ , then  $z^{2n} A(\rho z^{-1})$  will have a zero in  $\rho z_i$ .

By considering the two properties of the filter structure, Nehorai designed the IIR notch filter in the general set



$$H(z^{-1}) = \frac{A(z^{-1})}{A(\rho z^{-1})} = \frac{1 + a_1 z^{-1} + \cdots + a_n z^{-n} + \cdots + z^{-2n}}{1 + \rho a_1 z^{-1} + \cdots + \rho^n a_n z^{-n} + \cdots + \rho^{2n} z^{-2n}} \quad (2.2)$$

where the filter coefficients  $a_i$  ( $i = 1, \dots, n$ ) are the parameters to be estimated;  $n$  is the number of input sine waves;  $\rho$  is a parameter that controls the bandwidth of each notch.

Let  $y(t)$  be the observed time series at time  $t$ . The error output of the algorithm will be

$$\varepsilon(t) = \frac{A(q^{-1})}{A(\rho q^{-1})} y(t) \quad (2.3)$$

where  $q^{-1}$  is the unit delay operator, i.e.,  $q^{-1}y(t) = y(t-1)$ , etc.

The adaptive estimation algorithm will adjust the coefficients  $a_i$  so as to minimize the cost function

$$V_t(\theta(t-1)) = \frac{1}{2} \sum_{k=1}^t \beta(t, k) \varepsilon(k)^2 \quad (2.4)$$

where  $\beta(t, k)$  is a parameter introduced in time-varying systems that will be explained soon.

Observe that the mirror symmetric form of  $A(z^{-1})$  is not sufficient to guarantee that the zeros of  $H(z^{-1})$  will be on the unit circle. However, to minimize the squared error function of Eq. (2.4), the convergent filter must place the zeros on the unit circle.

The adaptive IIR notch filter model designed by Nehorai has several advantages over the previous designed IIR notch filters. Previous adaptive IIR notch filters were designed by Thompson [2], Kung and Rao [3], [4], and Friedlander and Smith [5]. These filters required  $2n$  parameters, where  $n$  is the number of input sine waves. For example, the IIR filters of [2] and [4] did not constrain the zeros to be on the unit circle and required  $2n$  parameters. Since zeros were not constrained to be on the unit circle, their convergent angles were, in general, slightly different from the sine wave frequencies.

Another method in [3] and [4] was based on estimating the second-order factors of the IIR transfer function. This method suffers from high nonlinearity in the minimization problem, which complicates the algorithm and deteriorates its performance.

The ANF in [1] uses a minimal number of parameters equal to  $n$ , the number of the input sine waves or narrowband signal components. As will be shown soon, the algorithm is of the RPE identification schemes. By the parsimony principle [6], among RPE algorithms using hierarchical models (i.e., one can be obtained from the other by applying some constraints), the most accurate asymptotically is the one with the simplest model.

The bandwidth of the complex notches created by each pole-zero pair is given by

$$B = \pi (1 - \rho)$$

In the algorithm, the previous fixed modulus  $\rho$  of the poles was replaced by a time-varying function  $\rho(t)$  for the following reasons. In actual situations, if no information is available for the input sine waves, and if the notches are too narrow, the notches may not fall over the sine wave frequencies and the algorithm will not "sense" the presence of the sine waves. Therefore, at the start of the data processing, wider notches (smaller values of  $\rho$ ) should be applied to the algorithm to increase the filter sensitivity to the presence of input sine waves. After convergence, narrower notches are recommended to obtain a smaller distortion in the wideband component by the filter.

The special model designed by Nehorai exhibits high stability, which is different from previous ANF algorithms. This saves computations needed to monitor the stability of RPE algorithms and enables the use of poles very close to the unit circle, resulting in significantly sharper notches than in the previous algorithms. The narrower bandwidth is needed to obtain a smaller distortion in the wideband component by the filter.

### II.3 ALGORITHM DERIVATION

Let  $y(t)$  be the observed time series at time  $t$ . The error output of the algorithm is expressed in Eq. (2.3). Assume the system is time-varying. To obtain an estimate that is representative of the current properties of the system, it is natural to consider the cost function expressed in Eq. (2.4). Rewriting Eq. (2.4), we have

$$V_t(\theta(t-1)) = \frac{1}{2} \sum_{k=1}^t \beta(t, k) \varepsilon(k)^2 \quad (2.5)$$

where  $\beta(t, k)$  is a parameter introduced in time-varying systems. The sequence of estimates can be computed recursively only if a certain structure for  $\beta(t, k)$  is introduced [7]. Assume that

$$\beta(t, k) = \lambda(t) \beta(t-1, k), \quad 1 \leq k \leq t-1 \quad (2.6)$$

This can also be written as

$$\beta(t, k) = \left[ \prod_{j=k+1}^t \lambda(j) \right] \alpha_k \quad (2.7)$$

where

$$\beta(k, k) = \alpha_k \quad (2.8)$$

Typically  $\lambda(k) \leq 1$ . In the algorithm,  $\alpha_k$  is set to 1.  $\lambda(k)$  is referred as the forgetting factor.

The filter parameter vector is defined by

$$\theta = [a_1 \cdots a_n]^T \quad (2.9)$$

where the superscript  $T$  denotes transpose. Let the gradient of  $\varepsilon(t)$  w.r.t. the model  $\theta$  be

$$\psi(t) = [\psi_1(t) \cdots \psi_n(t)]^T \quad (2.10)$$

where

$$\psi_i(t) = -\frac{\partial \varepsilon(t)}{\partial a_i} \quad (2.11)$$

Let  $\theta(t-1)$  be the estimate at time  $t-1$ . We wish to obtain a  $\theta(t)$  that approximately minimizes  $V_t(\theta(t-1))$ . By means of Taylor expansion of  $V_t(\theta)$  around  $\theta(t-1)$ , we have

$$\begin{aligned} V_t(\theta) = & V_t(\theta(t-1)) + V_t'(\theta(t-1))[\theta - \theta(t-1)] \\ & + \frac{1}{2}[\theta - \theta(t-1)]^T V_t''[\theta - \theta(t-1)] \\ & + o(|\theta - \theta(t-1)|^2) \end{aligned}$$

where the prime denotes differentiation with respect to  $\theta$ , and  $o(x)$  denotes a function such that  $o(x)|x| \rightarrow 0$  as  $|x| \rightarrow 0$ . Minimization of this expression with respect to  $\theta$  gives

$$\begin{aligned} \theta(t) = & \theta(t-1) - [V_t''(\theta(t-1))]^{-1} V_t'[\theta(t-1)]^T \\ & + o(|\theta(t) - \theta(t-1)|) \end{aligned} \quad (2.12)$$

From Eqs. (2.5), (2.6), (2.10) and (2.11), we have

$$\begin{aligned} [V_t'(\theta(t-1))]^T = & \lambda(t)[V_{t-1}'(\theta(t-1))]^T \\ & - \psi(t, \theta(t-1)) \varepsilon(t, \theta(t-1)) \end{aligned} \quad (2.13)$$

and, by differentiating Eq. (2.13) once more,

$$V_t''(\theta) = \lambda V_{t-1}''(\theta) + \psi(t, \theta) \psi^T(t, \theta) + \varepsilon''(t, \theta) \varepsilon(t, \theta) \quad (2.14)$$

In order to evaluate Eq. (2.12), a number of approximation have to be introduced:

First assume the next estimate  $\theta(t)$  is to be found in a small neighborhood of  $\theta(t-1)$ . This should be a reasonable approximation if  $t$  is large. That assumption leads to the following approximation:

Neglect  $o(|\theta(t) - \theta(t-1)|)$  in Eq. (2.12) and take

$$V_t''(\theta(t)) = V_t''(\theta(t-1)). \quad (2.15)$$

Then assume that  $\theta(t-1)$  is indeed the optimal estimate at time  $t-1$ , so that

$$V_{t-1}'(\theta(t-1)) = 0. \quad (2.16)$$

Finally set

$$\varepsilon''(t, \theta(t-1)) \varepsilon(t, \theta(t-1)) = 0. \quad (2.17)$$

This is because close to the true value  $\theta$ ,  $\varepsilon(t, \theta)$  will be almost white noise, so that we may approximately consider  $\varepsilon(t, \theta)$  to be of zero mean and independent of what happened up to time  $t-1$ . Inserting Eqs. (2.15) and (2.17) into Eq. (2.14), and denoting  $R(t) = V_t''(\theta(t))$ , we have

$$R(t) = \lambda(t)R(t-1) + \psi(t)\psi^T(t) \quad (2.18)$$

Inserting Eq. (2.16) into Eq. (2.13), we have

$$[V_t'(\theta(t-1))]^T = -\psi(t, \theta(t-1)) \varepsilon(t, \theta(t-1)) \quad (2.19)$$

Inserting Eq. (2.19) into Eq. (2.12) and neglecting  $o|\theta(t) - \theta(t-1)|$ , we have

$$\theta(t) = \theta(t-1) + R(t)^{-1} \psi(t) \varepsilon(t) \quad (2.20)$$

Letting  $P(t) = R(t)^{-1}$  in Eq. (2.18), we have

$$P(t) = \frac{1}{\lambda(t)} P(t-1) \quad (2.21a)$$

$$- \frac{1}{\lambda(t)} P(t-1) \psi(t) [\lambda(t) + \psi^T(t) P(t-1) \psi(t)]^{-1} \psi^T(t) P(t-1) \quad (2.21b)$$

$$\theta(t) = \theta(t-1) + P(t) \psi(t) \varepsilon(t)$$

From Eq. (2.3), it is possible to write the following differential equation:

$$\varepsilon(t) = y(t) + y(t-2n) - \rho^{2n} \varepsilon(t-2n) - \phi^T(t) \theta \quad (2.22)$$

where

$$\phi_i(t) = [\phi_1(t) \cdots \phi_n(t)]^T \quad (2.23a)$$

and

$$\phi_i(t) = -y(t-i) - y(t-2n+i) + \rho^i \varepsilon(t-i) \quad (2.23b)$$

$$+ \rho^{2n-i} \varepsilon(t-2n+i), \quad 1 \leq i \leq n-1,$$

$$\phi_i(t) = -y(t-n) + \rho^n \varepsilon(t-n), \quad i = n. \quad (2.23c)$$

From Eq. (2.3), we have

$$A(\rho q^{-1}) \varepsilon(t) = A(q^{-1})y(t). \quad (2.24)$$

Differentiating both sides of Eq. (2.24) w.r.t.  $a_i$ , one obtains

$$A(\rho q^{-1}) \frac{\partial \varepsilon(t)}{\partial a_i} + \rho^i \varepsilon(t-i) + \rho^{2n-i} \varepsilon(t-2n+i) \quad (2.25a)$$

$$= y(t-i) + y(t-2n+i) \quad 1 \leq i \leq n-1,$$

$$A(\rho q^{-1}) \frac{\partial \varepsilon(t)}{\partial a_n} + \rho^n \varepsilon(t-n) = y(t-n) \quad i = n. \quad (2.25b)$$

From these expressions and with Eq. (2.23), one obtains

$$\psi(t) = \frac{\phi(t)}{A(\rho q^{-1})} \quad (2.26a)$$

$y_F(t)$  and  $e_F(t)$  are correspondingly defined as

$$y_F(t) = \frac{y(t)}{A(\rho q^{-1}, t)} \quad (2.26b)$$

$$e_F(t) = \frac{e(t)}{A(\rho q^{-1}, t)} \quad (2.26c)$$

Based on the above derivation, the RPE type ANF algorithm will adjust the filter coefficients  $\theta$  to minimize the cost function of Eq. (2.5).

## II.4 SIMULATION RESULTS

In this section, we will test the RPE type ANF algorithm under different conditions. We wrote a C program to do the simulation on a SUN-SPARC workstation with double precision arithmetic. In all of the simulations, we assume that the algorithm converges if the percentage error of notch frequency is less than 0.1% within 2000 recursions.

The algorithm was tested under different signal-to-noise ratios (SNR) and different sampling frequencies. All the initial conditions are set according to the original design. We consider the input signal of the filter to be

$$y(t) = C_1 \sin 2\pi f_1 t + v(t) \quad (2.27)$$

where  $C_1$  is the sine wave amplitude and  $v(t)$  is a zero-mean unit-variance white Gaussian noise. The sampling ratio is defined by  $f_s/f_1$ , where  $f_s$  is the sampling frequency and  $f_1$  is the input sine wave frequency. Here  $f_1 = 0.1\text{HZ}$ . The sampling ratio varies from 2.0 to 18.0, with 0.1 increment.

Results from the experiment show that for the case when  $SNR = 40$  and the sampling ratio is one of the following values: 2.0, 2.1, 2.6, 2.7, 2.8, 6.8, 7.0, 7.2, 7.4, 7.6, 7.8, 8.0, 8.2, 8.4, 8.6, 8.8, 9.0, the algorithm will not converge. Also for the case when  $SNR = 10$  and the sampling ratio is one of the following values: 2.0, 2.1, 2.5, 2.7, 2.9, 3.1, 3.3, 5.4, 5.6, 5.8, 6.0, 6.2, 6.4, 6.6, 6.8, 7.0, 7.2, 7.4, 7.6, 7.8, 8.0, 8.2, 8.4, 8.6, 8.8, 9.0, 9.2, 9.4, 9.6, 9.8, 10.0, the algorithm will not converge. From the experiment, there seems to be no general rule what value of sampling ratio will make the algorithm converge. Table I shows the results. The non-convergence percentage value is calculated by the number of non-convergent experiments divided by the total number of experiments.

TABLE I  
SIMULATION RESULTS

SNR	Number of non-convergent experiments	Number of total experiments	Non-convergent percentage
40	26	160	16.25
10	31	160	23.13

The purpose of designing the ANF is to track time-varying and unknown sine wave signal. Both the sine wave signal amplitude and frequency are unknown. Consequently, we do not know the sampling ratio. If the sampling ratio happens to be one of the non-convergent sampling ratios mentioned earlier, the filter coefficient will not converge to the true value. From Table I, we see that the probability of non-convergence ranges from 16% to 23% for different SNRs.

Why will the filter coefficient not converge under some conditions? Recall that the RPE type algorithm is developed by minimizing the cost function of Eq. (2.4). The RPE algorithm is shown in [7] to converge to a local minimum of the cost function. This implies that if some other local minima exist on the error-performance surface, there is no guarantee that the algorithm will converge to the global minimum. Thus, it is very important to study the error-performance surface of the cost function in order to analyze the convergence performance of the algorithm. In the following chapter, the error-performance surface for the filter structure will be examined and analyzed.



## CHAPTER III

### ERROR-PERFORMANCE SURFACE

#### III.1 INTRODUCTION

As we discussed in Chapter II, the RPE type IIR ANF algorithm will adjust the filter coefficients  $\theta$  to minimize the cost function in Eq. (2.4). Rewriting the cost function, we have

$$V_t(\theta(t-1)) = \frac{1}{2} \sum_{k=1}^L \beta(t, k) \varepsilon(k)^2. \quad (3.1)$$

The RPE type algorithm converges to a local minimum of the cost function as  $t$  approaches infinity [7]. Since the algorithm is based on local minimum search, we can not expect convergence to a global minimum if the cost function has several local minima. The number of local minima will depend on the character of the model set used. In order to investigate the non-convergence of Nehorai's algorithm (we will call it the original algorithm later), it is important to study the error-performance surface for the model set used in [1]. In the reminder of this chapter, error-performance surfaces with different SNRs, different sampling frequencies and different notch bandwidth control parameters  $\rho$  will be analyzed. We will start with the description of the method to plot the error-performance surface, followed by examples and analysis.

#### III.2 ERROR-PERFORMANCE SURFACE EXPERIMENT

##### III.2.1 Method to plot error-performance surface

For simplicity, we first exam the error-performance surface for the input signal

containing only one sine wave signal plus white noise. Thus, the error-performance surface plot is a two-dimensional plot. Similar analysis can be applied to multi-dimensional error-performance surfaces.

Recall that in the example of Chapter II, we tested the original algorithm using an input signal

$$y(t) = C_1 \sin 2\pi f_1 t + v(t) \quad (3.2)$$

where  $v(t)$  is a zero-mean unit-variance white Gaussian noise. This implies that the input signal is time-invariant. So, in this particular example, the system is time-invariant. In Chapter II, we mentioned that  $\lambda(t)$  is a parameter introduced in time-varying systems. For a time-invariant system,  $\lambda(t)$  should always be 1. Correspondingly,

$$\beta(t, k) = 1, \quad \text{for } 1 \leq k \leq t. \quad (3.3)$$

For our experiment, the input signal  $y(t)$  is determined by Eq. (3.2). From Eq. (2.2), the system function of the filter can be written as

$$H(z^{-1}) = \frac{1 + a_1 z^{-1} + z^{-2}}{1 + \rho a_1 z^{-1} + \rho^2 z^{-2}} \quad (3.4)$$

From Eq. (2.22), the output of the filter  $\varepsilon(t)$  will be

$$\varepsilon(t) = y(t) + a_1 y(t-1) + y(t-2) - \rho a_1 \varepsilon(t-1) - \rho^2 \varepsilon(t-2) \quad (3.5)$$

From Eqs. (3.3) and (2.4), we have

$$V_t(\theta) = \frac{1}{2} \sum_{k=1}^t \varepsilon(k)^2. \quad (3.6)$$

Equation (3.6) is an estimate of the system behavior during the time period  $1 \leq k \leq t$ . Since the system is time-invariant, the estimate of the system behavior during the time period  $N_1 \leq k \leq N_2$ , where  $N_1$  and  $N_2$  represent different times, is the same as that during the time period  $1 \leq k \leq t$ .

We wrote a C program to examine the error-performance surface. In the program, we use the following equation to calculate the cost function for a particular filter coefficient  $a_1$ .

$$V_t(\theta) = \frac{1}{2} \sum_{k=90}^{100} \varepsilon(k)^2 \quad (3.7)$$

In the program, we vary filter coefficient from -10 to 10, with 0.1 increment. For each value of filter coefficient, the program takes  $y(t)$  in Eq. (3.2) as the input starting from time  $t = 0$ , let  $y(t)$  filter through Eq. (3.5) to obtain the filter output  $\varepsilon(t)$  at each time, use Eq. (3.7) to calculate the cost function for the corresponding filter coefficient. Then change the filter coefficient to a different value, use the above steps to calculate the corresponding cost function again.

### III.2.2 Zero of the system function

Letting  $z = e^{j\omega}$  in Eq. (3.4), we have

$$\begin{aligned} H(e^{j\omega}) &= \frac{1 + a_1 e^{-j\omega} + e^{-2j\omega}}{1 + \rho a_1 e^{-j\omega} + \rho^2 e^{-2j\omega}} \\ &= \frac{(1 + a_1 \cos \omega + \cos 2\omega) - j(a_1 \sin \omega + \sin 2\omega)}{(1 + \rho a_1 \cos \omega + \rho^2 \cos 2\omega) - j(\rho a_1 \sin \omega + \rho^2 \sin 2\omega)} \end{aligned} \quad (3.8)$$

The magnitude response of  $H(e^{j\omega})$  is

$$|H(e^{j\omega})| = \frac{[(1 + a_1 \cos \omega + \cos 2\omega)^2 + (a_1 \sin \omega + \sin 2\omega)^2]^{\frac{1}{2}}}{[(1 + \rho a_1 \cos \omega + \rho^2 \cos 2\omega)^2 + (\rho a_1 \sin \omega + \rho^2 \sin 2\omega)^2]^{\frac{1}{2}}} \quad (3.9)$$

Letting the numerator in Eq. (3.9) to be zero, we have

$$a_1^2 + 4a_1 \cos \omega + 4\cos^2 \omega = 0. \quad (3.10)$$

Solving Eq. (3.10), we have

$$\cos \omega = -\frac{a_1}{2}. \quad (3.11a)$$

Thus, the zero of the system function is at

$$\omega = \arccos\left(-\frac{a_1}{2}\right). \quad (3.11b)$$

### III.3 ERROR-PERFORMANCE SURFACE ANALYSIS

Figures 2 - 7 show the ANF error-performance surface of the cost function under different signal-to-noise ratios and sampling ratios. The sampling ratio is defined by  $f_s/f_1$ , where  $f_s$  is the sampling frequency and  $f_1$  is the notch frequency. We randomly choose the value of the sampling ratio to be 19.1 and 13.8. The SNR condition was obtained by  $SNR(dB) = 10\log(C_1^2/2)$ , where  $C_1$  is the sine wave amplitude. The input signal used here consists of one sine wave signal plus white Gaussian noise.

As mentioned before, the input signal is

$$y(t) = C_1 \sin 2\pi f_1 t + v(t).$$

Sampling  $y(t)$ , we have

$$\begin{aligned} y[n] &= C_1 \sin(2\pi f_1 T_s n) + v[n] \\ &= C_1 \sin(2\pi f_1 \frac{1}{f_s} n) + v[n]. \end{aligned} \quad (3.12)$$

Here  $T_s$  is the sampling period and  $f_s$  is the sampling frequency. Defining  $n_s = f_s/f_1$ , where  $f_1$  is the input sine wave frequency, and letting

$$\omega = \frac{2\pi}{n_s}. \quad (3.13)$$

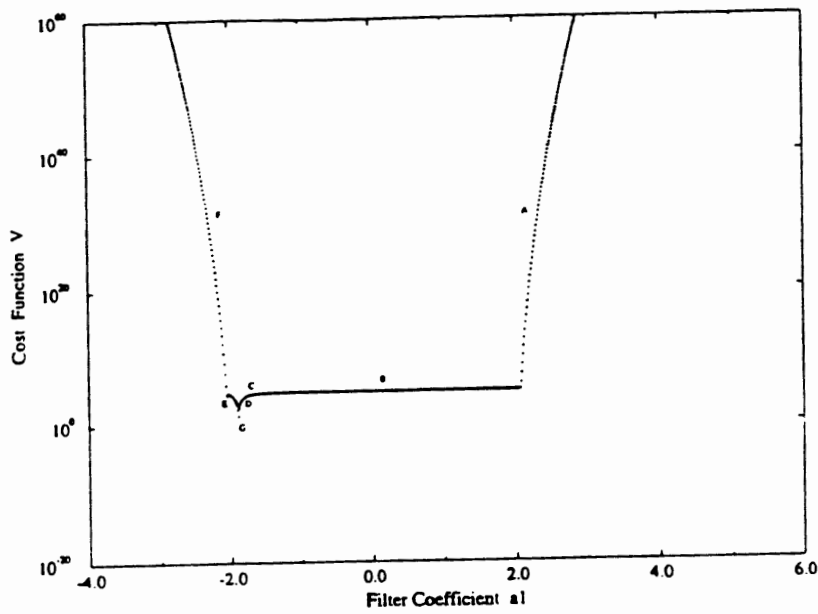
Equation (3.12) becomes

$$y[n] = C_1 \sin\left(\frac{2\pi}{n_s} n\right) + v[n] \quad (3.14a)$$

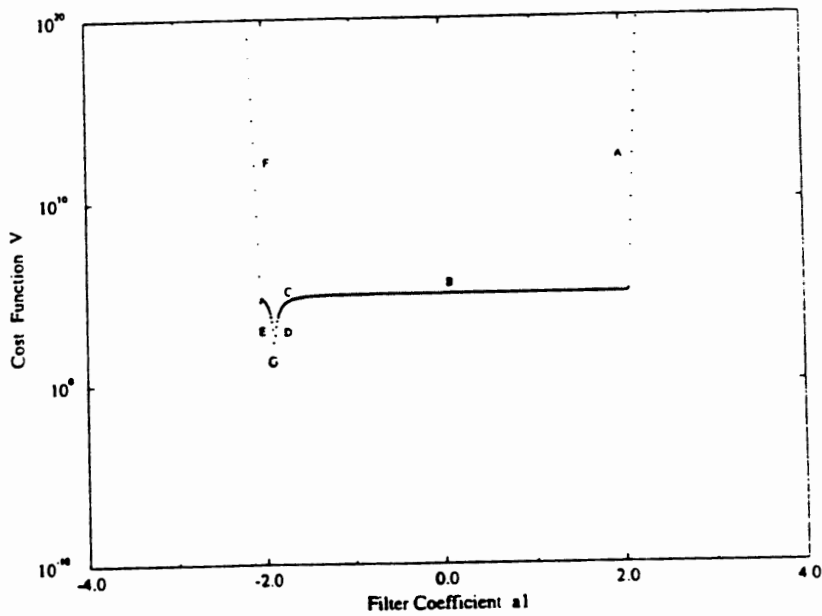
$$= C_1 \sin(\omega n) + v[n]. \quad (3.14b)$$

From Eq. (3.11b), we know the zero of the filter. Considering both Eqs. (3.11b) and (3.13), the true filter coefficient should be

$$a_1 = -2 \cos \omega = -2 \cos \frac{2\pi}{n_s} \quad (3.15)$$



(a)



(b)

Figure 2. Error-performance surface for  $\rho = 0.8$ ,  $SNR = 40$ ,  $f_s/f_1 = 19.1$ . (a) Error-performance surface. (b) Enlargement portion of error-performance surface near the true filter coefficient.

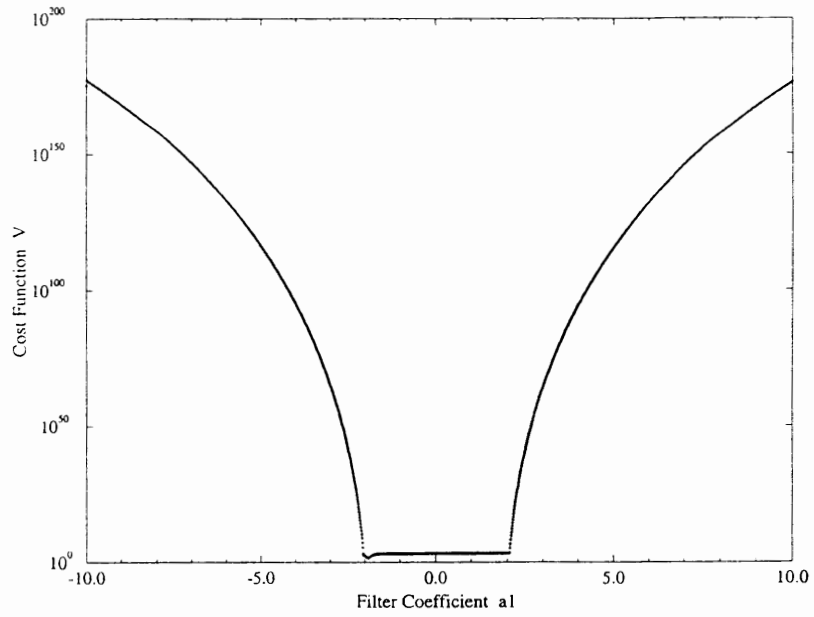


Figure 3. Error-performance surface for  $\rho = 0.8$ ,  $SNR = 10$ ,  $f_s/f_1 = 19.1$ .

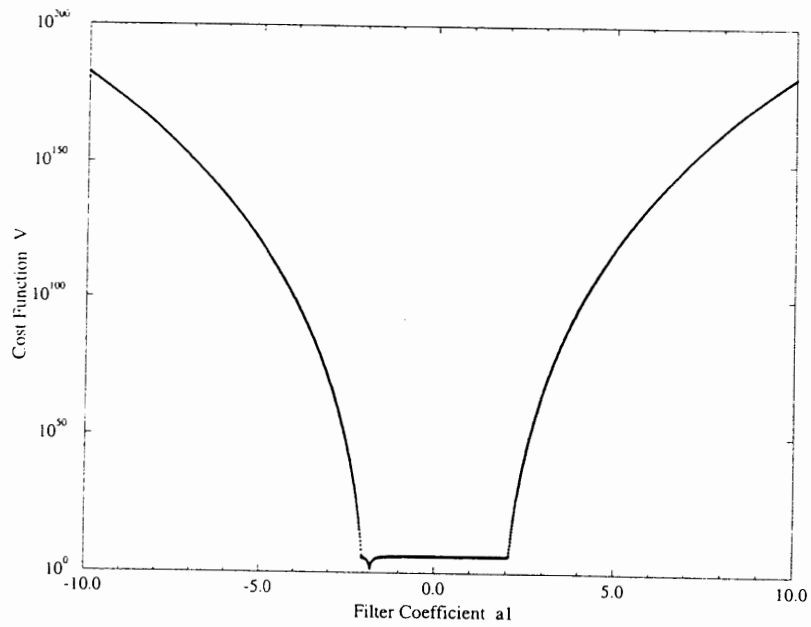


Figure 4. Error-performance surface for  $\rho = 0.8$ ,  $SNR = 40$ ,  $f_s/f_1 = 13.8$ .

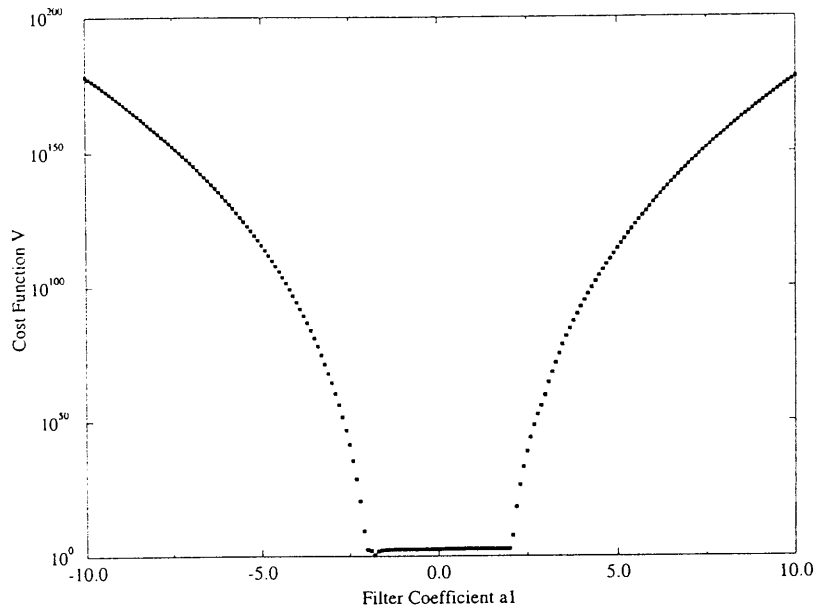


Figure 5. Error-performance surface for  $\rho = 0.8$ ,  $SNR = 10$ ,  $f_s/f_1 = 13.8$ .

Figures 2 - 5 show that the error-performance surface has several different types of shape corresponding to different values of the filter coefficient. On these performance surfaces, we use  $\rho = 0.8$ . In Fig. 2,  $n_s = 19.1$ . According to Eq. (3.15), the true filter coefficient should be  $-1.892756$ . As shown in Fig. 2, when the filter coefficient  $a_1$  is far away from its true value and is approaching its true value, the cost function  $V$  will generally decrease. This is shown in region A and F of Fig. 2. As the filter coefficient  $a_1$  is closer to its true value,  $V$  will pass a flat region, see region B of Fig. 2. In this flat region,  $V$  changes little. As  $a_1$  is closer to its true value,  $V$  will go through a transient region, see region C of Fig. 2. In this region,  $V$  will gradually decrease. As  $a_1$  is closer to the true value,  $V$  decreases quickly, see region D and E of Fig. 2. At the true value of  $a_1$ , the value of the cost function  $V$  is the minimum. This is shown in the convergent region G of Fig. 2. Thus, the cost function  $V$  will generally have a unique unconstrained global minimum. However, this minimum lies in a fairly flat region.

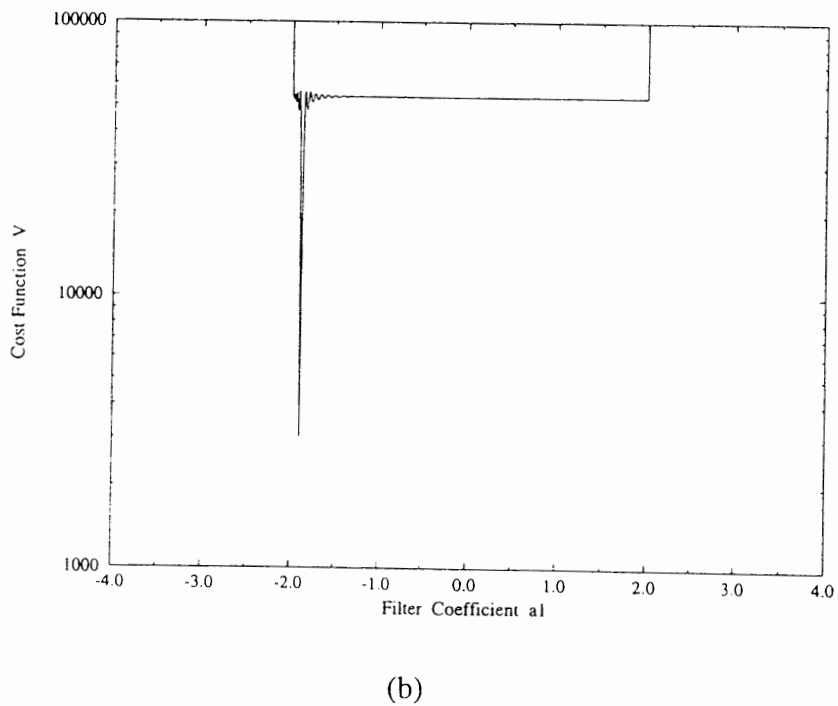
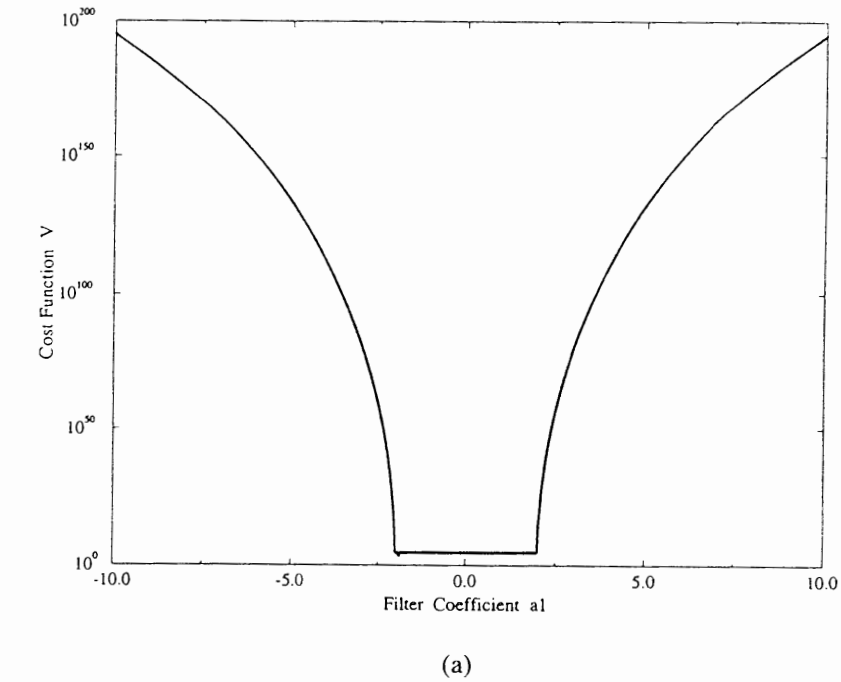


Figure 6. Error-performance surface for  $\rho = 0.985$ ,  $SNR = 40$ ,  $f_s/f_1 = 19.1$ . (a) Error-performance surface. (b) Enlargement portion of error-performance surface near the true filter coefficient.



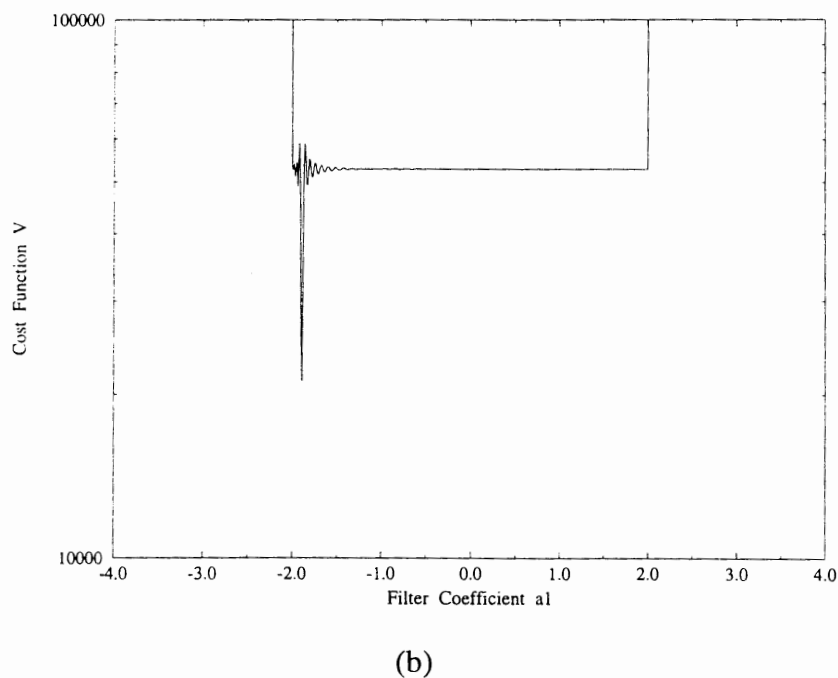
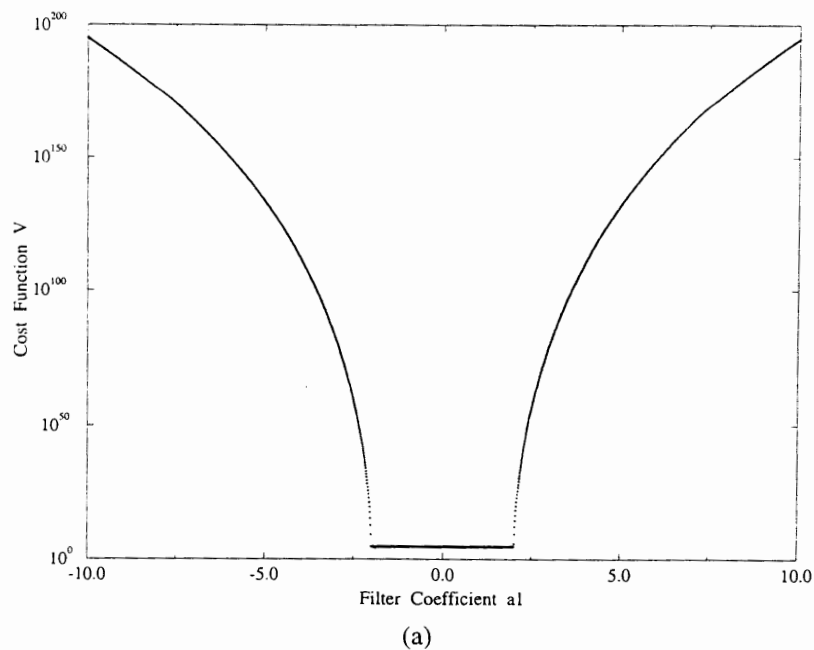


Figure 7. Error-performance surface for  $\rho = 0.995$ ,  $\overline{SNR} = 40$ ,  $f_s/f_1 = 19.1$ . (a) Error-performance surface. (b) Enlargement portion of error-performance surface near the true filter coefficient.

We also notice from Figs. 2 - 5 that when the absolute value of the filter coefficient is greater than 2, the value of cost function becomes very large. This is because the pole of the ANF filter is outside the unit circle. This means the filter is in an unstable region.

Also, for the case when  $\rho$  is very close to 1 ( $\rho = 0.985$ ), there exists several local minima on the error-performance surface. Fig. 6(a) shows the error-performance surface for  $\rho = 0.985$ ,  $SNR = 40$  and  $f_s/f_1 = 19.1$ . Fig. 6(b) is the enlargement portion of Fig. 6(a) near the true filter coefficient. Fig. 6(b) shows that there are several local minima near the global minimum. As the value of  $\rho$  is closer to 1, the number of local minima on the performance surface increases. Fig. 7(a) is the error-performance surface for  $\rho = 0.995$ ,  $SNR = 40$  and  $f_s/f_1 = 19.1$ . Fig. 7(b) is the enlargement portion of Fig. 7(a) near the true filter coefficient. Comparing Fig. 6(b) and Fig. 7(b), we see that the number of local minima on the performance surface for  $\rho = 0.995$  is more than that for  $\rho = 0.985$ . Also, there are not many local minima on the performance surface for  $\rho = 0.8$  under the same conditions. This is shown in Fig. 2(a) and Fig. 2(b). Therefore, the value of  $\rho$  also plays an important role on the error-performance surface.

Given this type of error-performance surface, it is very difficult for the RPE type algorithm to search the global minimum of the cost function since the global minimum lies in a very flat region. Also, it is possible that the estimated filter coefficient will be far away from its true value for a limited number of recursions. In the next chapter, we will re-examine the performance of the original algorithm given the type of error-performance surface described above.

## CHAPTER IV

### PERFORMANCE ANALYSIS OF THE ORIGINAL ANF

#### IV.1 INTRODUCTION

In this chapter, we will re-examine the performance of the original ANF algorithm given the special error-performance surface described in Chapter III. In Chapter III, we concluded that it is very difficult for the RPE type of algorithm to search the error-performance surface for the global minimum of the cost function. With this in mind, we will analyze the algorithm performance in two steps: first, with respect to different initial conditions; second, with respect to different values of the notch bandwidth control parameter. We then will summarize the reasons for non-convergence of the original ANF algorithm. The convergence criterion will also be evaluated. We will conclude by addressing the problems of the original algorithm and their causes.

#### IV.2 PERFORMANCE ANALYSIS OF THE ORIGINAL ANF

We wrote another C program to do the simulation of the original ANF on a SUN-SPARC workstation with double precision arithmetic. In all of the simulations, we assume that the algorithm converges if the percentage error of notch frequency is less than 0.1% within 2000 recursions. Assume the input data is  $y(t) = C_1 \sin(2\pi f_1 t) + v(t)$ , where  $v(t)$  is a zero-mean unit-variance white Gaussian noise.  $f_1$  is the input sine wave frequency and  $f_1 = 0.1\text{HZ}$ .

#### IV.2.1 Performance vs. different initial conditions

Experiment 1. We modify the original algorithm by setting  $P[0]$  ranging from  $1.0e-6$  to  $1.0e+5$ . Results show that for each SNR and sampling ratio, there are certain values of  $P[0]$  that can ensure convergence. For example, for  $SNR = 40$  and  $f_s/f_1 = 11.1$ , only when the initial covariance  $P[0]$  is within  $1.0e-4$  to  $1.0e-2$ , the algorithm converges. If  $P[0]$  is chosen to be any value outside of this range, the algorithm will not converge. This indicates that the value of  $P[0]$  will affect the performance of the algorithm.

Experiment 2. A close examination of the non-convergence case shows that the initial value of the filter coefficient  $a_1[0]$  and the initial covariance  $P[0]$  are crucial to the performance of the algorithm. The convergence analysis of RPE type of algorithm is shown in [7] to converge to a local minimum of the cost function  $V$  as  $t$  approaches infinity. The estimated parameter  $a_1$  will converge in distribution to the normal distribution with zero mean and asymptotic covariance matrix  $P[t]$ . Thus,  $P[t]$  is an important information to update  $a_1[t]$ .  $a_1[0]$  can be considered as a prior estimate of filter coefficient  $a_1$ , and the covariance of this estimate is  $P[0]$ .

In Chapter III, we discussed the error-performance surface of the cost function. This surface has a unique shape with its global minimum lying in a fairly flat region. In this experiment, we choose  $a_1[0]$  to be in different regions on the error-performance surface. For each  $a_1[0]$ , we examine the performance as the value of  $P[0]$  varies. The sampling ratio is randomly chosen to be 19.1. According to Eq. (3.15), the true filter coefficient is -1.892756. We encounter the following different types of performance.

Case 1.  $a_1[0] = 4$ , where  $a_1[0]$  is in region A of Fig. 2(a).

- (a)  $P[0] = 1.0e-6$  to  $1.0e-4$ ,  $a_1$  does not converge. For the first 8 to 15 recursions,  $a_1[t]$  approaches region B of Fig. 2(a), then stays in region B. The

non-convergence is because of the mismatch of the relationship between  $a_1[0]$  and  $P[0]$ . Since  $a_1[0]$  is far away from the true value,  $P[0]$  should be large in order to make  $a_1$  converge. As  $P[0]$  is small, too much confidence is placed on  $a_1[0]$  and  $a_1$  does not converge. Region B is a fairly flat region. Because of the special shape of the performance surface, the RPE algorithm considers  $a_1[t]$  has already reached a local minimum.

- (b)  $P[0] = 1.0e-3$  to  $1.0e-2$ ,  $a_1$  converges. The transient behavior of  $a_1[t]$  shows that it passes through regions B-F-E-D-G for  $P[0] = 1.0e-3$  and F-E-D-G for  $P[0] = 1.0e-2$ . It converges because the value of  $P[0]$  is chosen properly to match the prior estimate  $a_1[0]$  and when the number of recursions increases, each  $P[t]$  matches  $a_1[t]$ . Here the word "match" means that if the filter coefficient is far away from its true value, its covariance should be large in order to ensure convergence; on the other hand, if the filter coefficient is close to the true value, its covariance should be small in order to ensure convergence.
- (c)  $P[0] \geq 1.0e-1$ ,  $a_1$  does not converge. The transient behavior demonstrates a very interesting point.  $a_1[t]$  passes through regions E-D or F-E-D after only a few recursions ( $t = 8$  to  $9$ ). At that time,  $P[t]$  is already too small to update  $a_1[t]$  to the convergent region G.  $a_1$  stays in region D with very small variations. We also notice that  $a_1[1]$  jumps from  $a_1[0] = 4$  in region A to region E or F. The large change of  $a_1$  is because  $P[0]$  is a large covariance.

Case 2.  $a_1[0] = 1$ , where  $a_1[0]$  is in region B of Fig. 2(a).

- (a)  $P[0] = 1.0e-6$  to  $1.0e-4$ ,  $a_1$  does not converge. Transient behavior shows  $a_1[t]$  will always stay in region B. Since  $a_1[0]$  is in a fairly flat region, and  $P[0]$  is too small to update  $a_1$  to other regions, the RPE algorithm determines  $a_1$  is minimized. This is similar to case 1(a).
- (b)  $P[0] = 1.0e-3$  to  $1.0e-2$ ,  $a_1$  converges. Transient behavior shows that  $a_1[t]$  passes through regions B-F-E-D-G for  $P[0] = 1.0e-3$  and F-E-D-G for  $P[0] = 1.0e-2$ . This case is similar to case 1(b).
- (c)  $P[0] \geq 1.0e-1$ ,  $a_1$  does not converge. Transient behavior shows  $a_1$  passes through regions F-E-D after a few recursions ( $t = 8$  to  $9$ ) and  $P[t]$  becomes very small.  $a_1$  stays in region D with very small variations. This is similar to case 1(c).

Case 3.  $a_1[0] = -1.65$ , where  $a_1[0]$  is in region C of Fig. 2(a).

- (a)  $P[0] = 1.0e-6$  to  $1.0e-4$ ,  $a_1$  converges. Transient behavior shows that  $a_1[t]$  passes through regions C-D-E-G for  $P[0] = 1.0e-6$ ; C-D-G for  $1.0e-5 \leq P[0] \leq 1.0e-4$ . Because  $a_1[0]$  is close to the true value, covariance of the prior estimate should be small. Thus, when  $P[0]$  is small,  $a_1$  converges.
- (b)  $P[0] \geq 1.0e-3$ ,  $a_1$  does not converge. Performance is similar to case 1(c). When  $a_1[0]$  is close to the true value,  $P[0]$  should be small. If  $P[0]$  is large,  $a_1$  does not converge.

Case 4.  $a_1[0] = -1.88$ , where  $a_1[0]$  is in region D of Fig. 2(a).

Performance is similar to case 3(a) for  $1.0e-6 \leq P[0] \leq 1.0e-3$  and case 3(b) for  $P[0] \geq 1.0e-2$ .

Case 5.  $a_1[0] = -1.98$ , where  $a_1[0]$  is in region E of Fig. 2(a).

- (a)  $P[0] = 1.0e-6$  to  $1.0e-3$ ,  $a_1$  converges. Transient behavior shows that  $a_1[t]$  passes through regions E-G for  $P[0] = 1.0e-6$ ; E-D-G for  $1.0e-5 \leq P[0] \leq 1.0e-4$ ; F-E-D-G for  $P[0] = 1.0e-3$ . This case is similar to case 3(a).
- (b)  $P[0] \geq 1.0e-2$ ,  $a_1$  does not converge. Performance is similar to case 3(b).

Case 6.  $a_1[0] = -9$ , where  $a_1[0]$  is in region F of Fig. 2(a).

- (a)  $P[0] = 1.0e-6$ ,  $a_1$  does not converge. Transient behavior shows that  $a_1[t]$  passes through regions F-E. After a few recursions ( $t = 15$ ),  $a_1[t]$  is in region E, and at that time  $P[t]$  is too small to update  $a_1$  to region G.
- (b)  $P[0] \geq 1.0e-5$ ,  $a_1$  does not converge. Transient behavior is the same as case 1(c).

Experiment 3. In this experiment, we randomly choose the sampling ratio to be  $f_s/f_1 = 13.8$ , and do the same simulation as in Experiment 2. Results are similar to those of  $f_s/f_1 = 19.1$  except for the non-convergence cases when  $P[0]$  is large. For example, when  $P[0] = 100$ ,  $a_1[t]$  passes through regions F-E on the error-performance surface, and stays in region E without changing much, instead of staying in region D. We notice that  $a_1[t]$  passes into region E only after a few recursions ( $t = 10$ ), and  $P[t]$  is already very small at that time. While  $a_1$  is in region E, its value still oscillates slightly.

#### IV.2.2 Performance vs. different notch bandwidth control parameter

Experiment 4. As discussed in Chapter II,  $\rho$  is a parameter that controls the bandwidth of the notch frequency. In Chapter III, we have shown that  $\rho$  plays an effect on the error-performance surface. Increasing  $\rho$  causes the number of local minima on the performance surface to increase. This experiment is to show the effect of  $\rho$  on the performance of the algorithm. The input signal is the same as that of Experiment 1. We use  $f_s/f_1 = 19.1$ ,  $SNR = 40$ ,  $P[0] = 10$ . After about 300 recursions,  $\rho$  is around 0.985, and  $a_1$  is always around -1.84 while the true value of  $a_1$  should be -1.892756. Fig. 6(a) is the corresponding error-performance surface for  $\rho = 0.985$ . Fig. 6(b) is the enlargement portion of Fig. 6(a) near the true filter coefficient. Notice that  $a_1 = -1.84$  is in between several local minima and is in a fairly flat region on the performance surface. Thus, as the number of recursions increases, it will be even more difficult for  $a_1$  to converge to the global minimum. When we modify the algorithm by fixing  $\rho = 0.8$ , the performance is better than using a time-varying  $\rho$  because there are not several local minima on the error-performance surface for  $\rho = 0.8$  as shown in Fig. 2. But  $a_1$  does not converge either because  $P[0]$  and  $a_1[0]$  are the major crucial factors on the performance of the algorithm.

#### IV.2.3 Non-convergence performance analysis

From the above experiments, we summarize that there are three reasons that the algorithm is difficult to converge. First, the special filter structure makes a unique type of error-performance surface. This type of performance surface is not quadratic, but has a global minimum lying in a very flat region, which is very difficult for the algorithm to search the global minimum. Second, as the number of recursions increases, the notch bandwidth control parameter  $\rho$  makes the convergence even more difficult. This means that at the start of the algorithm,  $\rho = 0.8$ . The error-performance surface for  $\rho = 0.8$



does not consist of several local minima. There is only a global minimum on the performance surface. As the number of recursions increases,  $\rho$  becomes larger and closer to 1. When the value of  $\rho$  increases, the error-performance surface will have several local minima. As  $\rho$  becomes larger, the number of local minima on the performance surface will increase. The performance surface with several local minima will make the algorithm even more difficult to search the global minimum because the RPE algorithm can only guarantee to search a local minimum. Third, the algorithm is very sensitive to the initial estimate of filter coefficient  $a_1[0]$  and the covariance of this initial estimate  $P[0]$ . Whenever a mismatch of  $a_1[0]$  and  $P[0]$  occurs,  $a_1$  can not converge. In other words, if the initial estimate of filter coefficient is far away from the true value, the initial covariance of this estimate should be large in order to ensure convergence; if the initial covariance is small, the algorithm will not converge. Also, if the initial estimate of the filter coefficient is close to the true value, the initial covariance should be small in order to ensure convergence; if the initial covariance is large, the algorithm will not converge either.

Because of the above three reasons, there are two types of non-convergence performance for the original ANF. First,  $P[0]$  is too large to match the prior estimate  $a_1[0]$ ,  $a_1$  will stay in region D or E on the performance surface without changing much. When  $a_1$  is in region D or E, its value oscillates slightly. Second,  $P[0]$  is too small to match the prior estimate  $a_1[0]$ ,  $a_1$  will either stay in the flat region B or region E, D, C.

From Experiment 2 and 3, we can also see that for the non-convergence cases, even though the algorithm did not converge, if the initial covariance  $P[0]$  is chosen to be very large, for example, if  $P[0] = 100$ , the filter coefficient  $a_1$  will pass into region D or E on the error-performance surface only within a few recursions no matter  $a_1[0]$  is near or far away from the true filter coefficient. The reason  $a_1$  can pass into region D or E quickly is because  $P[0]$  is considered to be a large covariance of the initial filter

coefficient estimate. A large  $P[0]$  can always update  $a_1$  to the unstable region on the error-performance surface quickly. As discussed in Chapter II, the algorithm is based on the gradient search of the error-performance surface; when  $a_1$  is updated to the unstable region on the performance surface, the value of the cost function  $V$  will be very large. This can increase the sensitivity of the algorithm to search the global minimum and  $a_1$  can pass into region D or E on the performance surface quickly. After  $a_1$  passes into region D or E on the error-performance surface, it stays there with very small variations. The non-convergence is because of the reason that at the time  $a_1$  passes into region D or E,  $P[t]$  is already too small to update  $a_1$  to the true filter coefficient.

### IV.3 CONVERGENCE CRITERION EVALUATION

For the original ANF algorithm, there is no self-adjusted criterion to determine whether the algorithm converges or not; and if it converges, how close the estimated filter coefficient is to the true filter coefficient. Nehorai used the number of recursions  $N$  as a criterion to analyze convergence [1]. Since for the simulation in [1], the input sine wave was actually known, the true filter coefficient  $\theta$  was also known. Nehorai set the length of  $N$  to be 100, 500, 1000 respectively and compared the filter coefficient from the algorithm to the true filter coefficient. But in actual situations, the input data is usually unknown and time-varying. Thus, the number of recursions can not be a sufficient convergence criterion.

From the above experiments and analysis, we conclude that the original ANF algorithm has two drawbacks. First, it can not guarantee convergence. Second, it does not have a convergence criterion. In the next chapter, we will develop a new ANF algorithm in order to overcome the drawbacks of the original algorithm.

## CHAPTER V

### A FAST ALGORITHM FOR ADAPTIVE NOTCH FILTERING

#### V.1 INTRODUCTION

In this chapter we address the primary goal of this thesis: to demonstrate a fast algorithm for adaptive notch filtering. Through the topics in Chapter IV, we analyze the reasons that the original ANF algorithm is difficult to converge. There are three reasons that cause the original algorithm difficult to converge. First, the filter structure makes a unique type of error-performance surface with its global minimum lying in a fairly flat region, which makes the algorithm very difficult to search the global minimum. Second, as the number of recursions increases, the number of local minima on the performance surface increases because of the increasing value of notch bandwidth control parameter  $\rho$ . Third, the algorithm is very sensitive to the initial filter coefficient estimate and the covariance of this estimate. Results of this investigation demonstrate that we can improve the convergent rate of the original ANF algorithm by improving its error-performance surface and by properly controlling the relationship between its initial filter coefficient estimate and the covariance of this estimate.

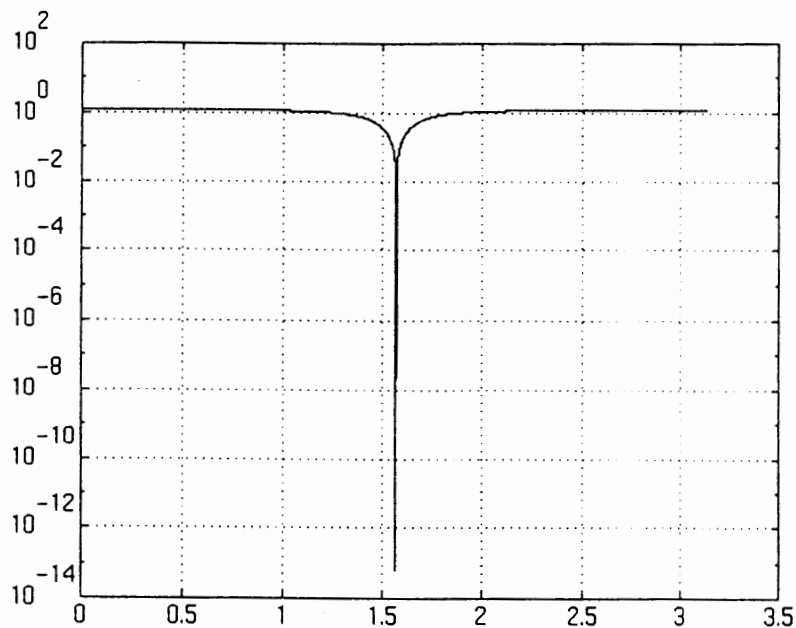
With this in mind, we will pursue a fast ANF algorithm based on the original ANF by the following new designs: we will improve the error-performance surface of the filter by designing a new notch bandwidth control parameter; we will increase the convergent rate by resetting the initial filter coefficient estimate and the covariance of this estimate; we will design a new criterion to provide information on the algorithm convergence and when to reset initial conditions.

In the reminder of this chapter, we will present the details of our efforts to design a fast ANF algorithm. Section V.2 describes the designs of the new ANF algorithm. In Section V.3 we present the new ANF algorithm description. Simulation results will be given in Section V.4.

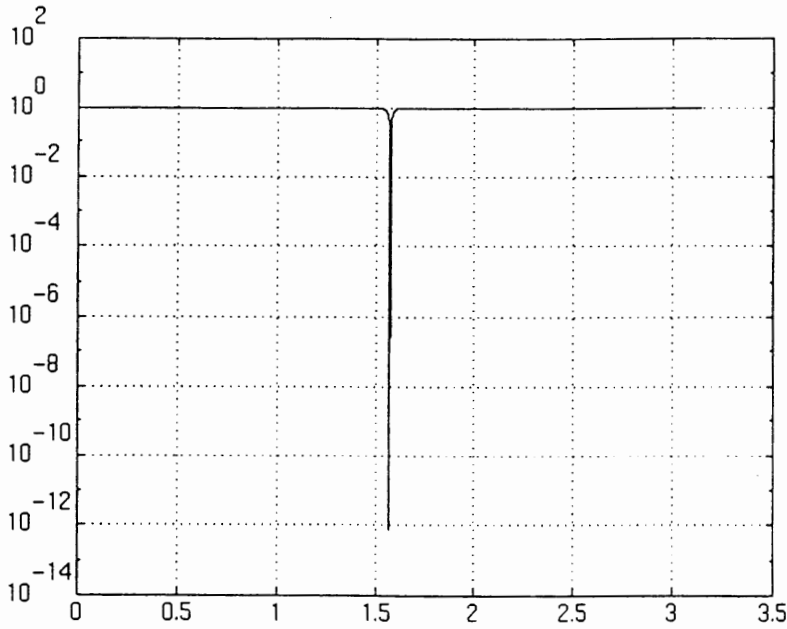
## V.2 DESIGN OF A FAST ALGORITHM FOR ADAPTIVE NOTCH FILTERING

### V.2.1 Design of notch bandwidth control parameter

In Chapter III, we described that the error-performance surface for the original ANF is very difficult for the RPE type of algorithm to search the global minimum because the global minimum lies in a very flat region. Fig. 2 and Fig. 6 are the error-performance surface for  $\rho = 0.8$  and  $\rho = 0.985$ . Fig. 8 and Fig. 9 are the corresponding frequency magnitude response when  $a_1 = 0$ .



**Figure 8.** Frequency magnitude response of the filter with  $\rho = 0.8$ .



**Figure 9.** Frequency magnitude response of the filter with  $\rho = 0.985$ .

As discussed in Chapter II,  $\rho$  is a notch bandwidth control parameter. The closer  $\rho$  is to 1, the narrower the bandwidth is. In practical situations, if no information is available on the input sine waves and if the notches are too narrow, the notches may not fall over the sine wave frequencies and the algorithm will not "sense" the presence of the sine waves. This may prevent the algorithm from converging to the desired transfer function.

The filter coefficient  $a_1$  only determines the location of notch frequency. As explained in Eq. (3.11b), for each  $a_1$ , the corresponding notch frequency  $\omega$  is such that  $\omega = \arccos(-\frac{a_1}{2})$ , provided  $a_1$  is in the stable region.

If the filter coefficient  $a_1$  is still far away from the true filter coefficient, and if the notch is too narrow, the input sine wave can not be cancelled out, sine wave signal will dominate the output of the filter, resulting in the flat region B on the error-performance surface in Fig. 2(a). The value on region B mainly reflects the input sine wave signal.

In order to increase the sensitivity of the ANF algorithm, wider notch (i.e. smaller values of  $\rho$ ) should be used at the start of the data processing. After convergence, it is recommended that a larger modulus of poles (narrower notch) be used in order to obtain a smaller distortion in the wideband component of the input signal by the filter.

Since applying  $\rho = 0.8$  in the ANF structure still results a fairly flat region on the error-performance surface, 0.8 is not small enough to increase the sensitivity of the algorithm. Our design is to use a value of  $\rho$  less than 0.8 in order to improve the shape of the error-performance surface, thus increasing the algorithm sensitivity to the best.

Figs. 10 - 13 show the error-performance surface for different values of  $\rho$  for the same signal-to-noise ratio  $SNR = 40$ . The error-performance surface plots in Figs. 10 - 13 and Fig. 2 show that for  $\rho \geq 0.2$ , performance surface consists a flat region; but for  $\rho \leq 0.1$ , the flat region on the performance surface gradually disappears. This is because we use a smaller  $\rho$ , widen the notch bandwidth, and thus improve the error-performance surface shape and the sensitivity of the algorithm. From the experiments,  $\rho = 0.001$  is a reasonable value that will improve the shape of the performance surface and increase the sensitivity of the algorithm. The following example confirms the above analysis.

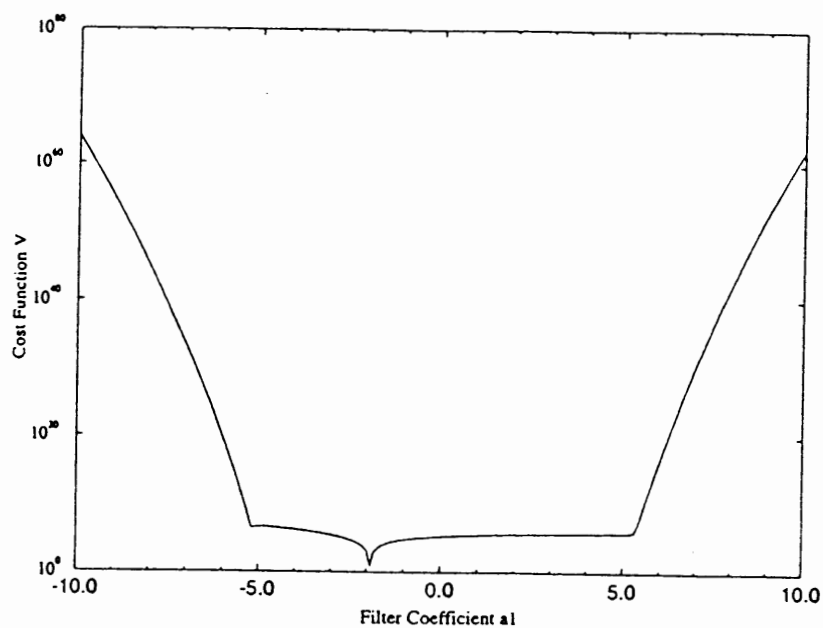


Figure 10. Error-performance surface for  $\rho = 0.2$ ,  $SNR = 40$ ,  $f_s/f_1 = 19.1$ .

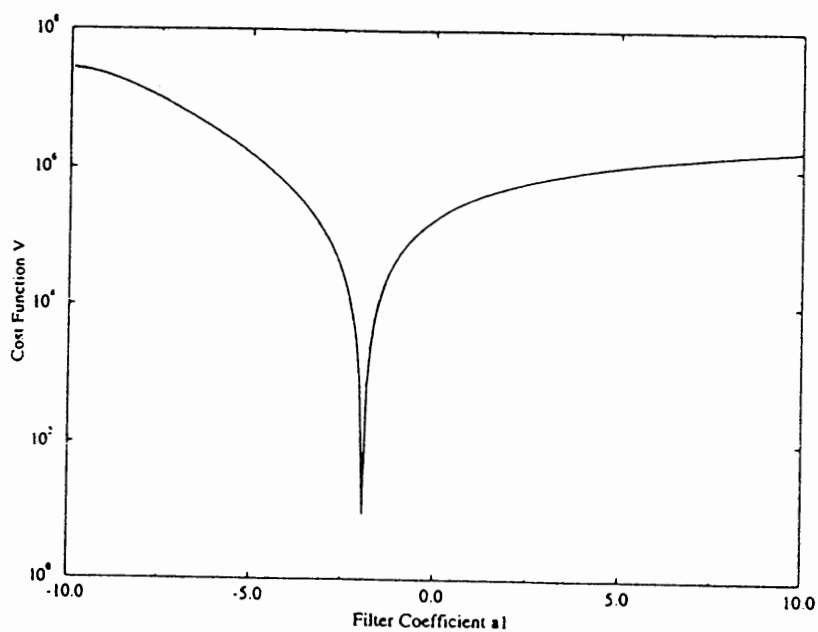


Figure 11. Error-performance surface for  $\rho = 0.1$ ,  $SNR = 40$ ,  $f_s/f_1 = 19.1$ .

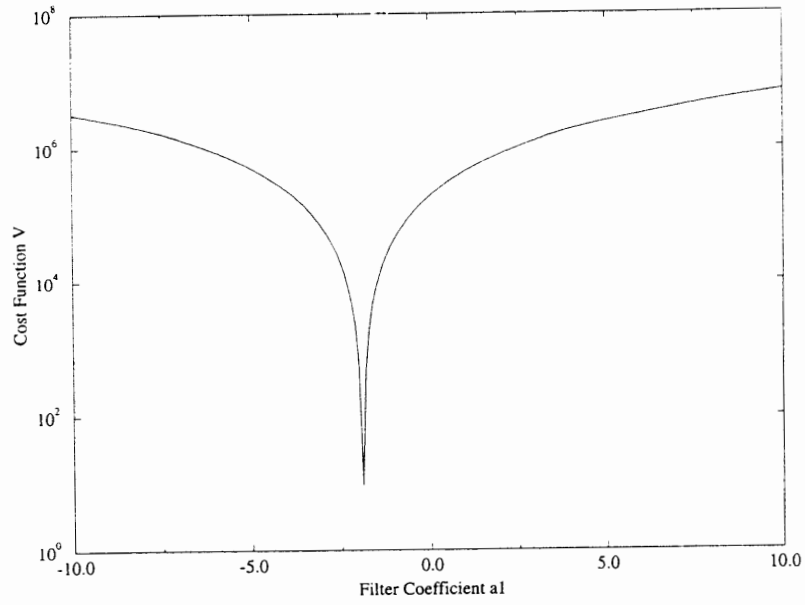


Figure 12. Error-performance surface for  $\rho = 0.001$ ,  $SNR = 40$ ,  $f_s/f_1 = 19.1$ .

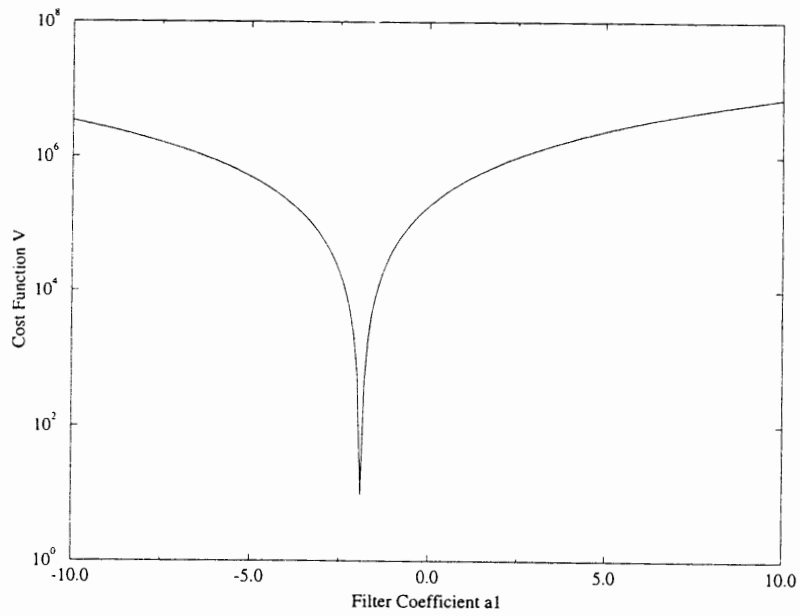


Figure 13. Error-performance surface for  $\rho = 0.0001$ ,  $SNR = 40$ ,  $f_s/f_1 = 19.1$ .



In this example, we modify Experiment 2 of Chapter IV by using  $\rho < 0.8$  instead of using a time-varying  $\rho$  starting at  $\rho = 0.8$ . As in Experiment 2 of Chapter IV, we test the original ANF by choosing the initial filter coefficient estimate  $a_1[0]$  at different regions on the performance surface. Results show that by using a large initial covariance  $P[0]$ , for example,  $P[0] = 100$ , it takes only 2 recursions for  $a_1$  to pass into region D or E on the performance surface with  $\rho = 0.001$  and it takes 4 recursions for  $a_1$  to pass into region D or E with  $\rho = 0.1$  while in Experiment 2 of Chapter IV, it took 8 to 10 recursions. So using  $\rho = 0.001$ , it takes fewer recursions for  $a_1$  to pass into region D or E than using  $\rho > 0.001$ . We also try to use  $\rho < 0.001$ , results show that it takes 2 recursions for  $a_1$  to pass into region D or E on the error-performance surface. Figures. 12 and 13 show that the error-performance surface shapes for  $\rho = 0.001$  and  $\rho = 0.0001$  are almost the same. Thus, using a  $\rho$  less than 0.001 will improve the sensitivity of the algorithm as much as using a  $\rho$  which equals to 0.001. That is the reason it takes the same number of recursions for  $a_1$  to pass into region D or E on the error-performance surface for  $\rho = 0.001$  and  $\rho < 0.001$ . Therefore,  $\rho \leq 0.001$  will be a reasonable value to increase the algorithm sensitivity. In the new algorithm, we use  $\rho = 0.001$ .

The above example confirms that by using a smaller value of  $\rho$ , the sensitivity of the algorithm increases and the convergent rate of the algorithm is improved. So our first design of the new algorithm is to use a smaller fixed-value of  $\rho$  at the start of the data processing until the filter coefficient is near the true value. Then we will use a time-varying  $\rho$  as proposed by the original ANF. By using a smaller  $\rho$  at the beginning of the data processing, the sensitivity of the algorithm will increase. After the filter coefficient is near the true value, a larger time-varying  $\rho$  will make the convergent notch narrower and minimize the distortion in the wideband component by the filter.

### V.2.2 Design of initial conditions

Using a smaller fixed-value of  $\rho$  will increase the sensitivity of ANF algorithm and improve the performance of the algorithm. As analyzed in Chapter IV, the algorithm is very sensitive to the initial conditions. If the initial filter coefficient estimate and the initial covariance of this estimate are not chosen properly, the algorithm will not converge even if the error-performance surface is improved. In the example of Section V.2.1, we notice that for the non-convergence cases, once the filter coefficient  $a_1$  passes into region D or E on the error-performance surface, it stays there. This is because at the time  $a_1$  passes into region D or E,  $P[t]$  is already too small to update  $a_1$  to the true value.

In order to have the original algorithm converge, the initial values of  $a_1[0]$  and  $P[0]$  must be chosen properly to match the relationship with each other. Specifically, if the prior estimate of filter coefficient is far from its true value, the initial covariance should be large in order to ensure convergence. On the other hand, if the prior estimate of filter coefficient is near the true value, the initial covariance should be small in order to ensure convergence. We also notice that if  $P[0]$  is too large to match  $a_1[0]$ ,  $a_1$  will pass into region D or E on the error-performance surface within a few recursions, where it will stay without change much. While  $a_1$  is in region D or E, its value still oscillates slightly. This transient behavior occurs after only a few recursions ( $k = 2$  for  $\rho = 0.001$ ). The reason  $a_1$  can pass into region D or E quickly is because  $P[0]$  is a large covariance, which will update  $a_1$  to the unstable region on the error-performance surface quickly. As discussed in Chapter II, the algorithm is based on the gradient search of the error-performance surface. After  $a_1$  is updated to the unstable region of the performance surface, the value of cost function  $V$  will be very large. This will increase the sensitivity of the algorithm to search the global minimum. The reason  $a_1$  does not change much once it passes into region D or E is because  $P[t]$  is already too small to update  $a_1$  to the

true value. The slow or non-convergence performance is because of the small  $P[t]$  after a few recursions. This type of performance is actually consistent with the convergence analysis of RPE type of algorithm. RPE type of algorithm will converge to a local minimum as  $t$  approaches infinity. The analysis of the asymptotic rate of convergence usually does not provide any hints about how large  $t$  has to be for the results to be applicable. It may be  $t = 100$  or  $t = 1.0e+6$ , which clearly makes a large difference in real-time processing. For an adaptive notch filter, we need a fast convergent rate. Otherwise, the filter is no longer adaptive.

We also discover that at the time  $a_1$  is in region D or E of the error-performance surface, it is actually not very far from the true value. It is the small  $P[t]$  that causes the slow convergence or non-convergence. In order to overcome the convergence problem, we reset the initial conditions of the new algorithm such that  $a_1[0] = a_1[t]$  and  $P[0] > P[t]$ .

Based on the above idea, the new ANF algorithm will reset the initial conditions of  $a_1[0]$  and  $P[0]$  whenever  $P[t]$  is too small to update  $a_1$  to the true value. But what kind of information can be used to decide when to reset  $a_1[0]$  and  $P[0]$ ? This question will be answered following the design of convergence criterion in the next section.

### V.2.3 Design of convergence criterion

#### V.2.3.1 Relationship between noise-to-signal power and frequency error

Assume Fig. 14 is the frequency magnitude response for the ANF with  $\rho = 0.8$ . In Fig. 14, assume  $\omega_1$  is the zero frequency on the frequency response,  $\omega_0$  is the input sine wave frequency.  $\Delta\omega$  is the frequency difference between  $\omega_1$  and  $\omega_0$ . In other words,  $\Delta\omega$  is the frequency error. Referring to Eq. (3.11a), we have

$$\cos \omega_1 = -\frac{a_1}{2}, \quad (5.1a)$$

$$\sin \omega_1 = \frac{\sqrt{4-a_1^2}}{2}. \quad (5.1b)$$

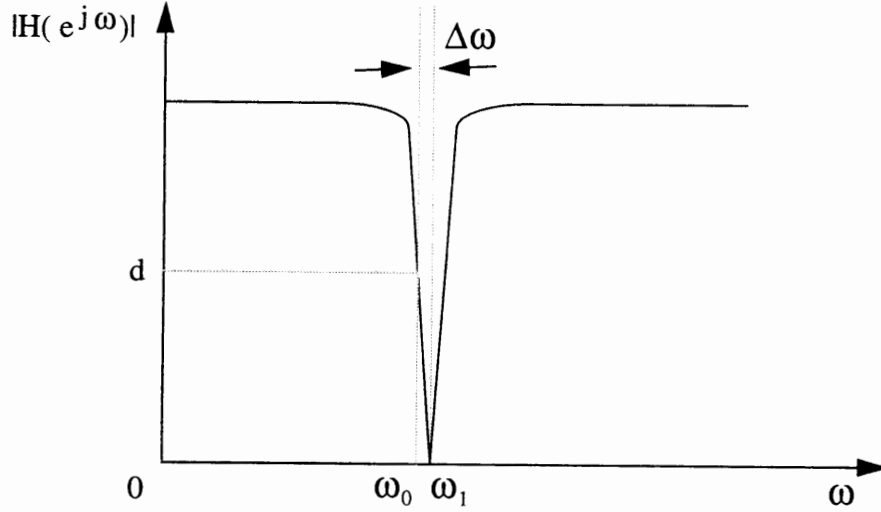


Figure 14. Relationship between frequency magnitude response and frequency error.

Denote  $d = |H(e^{j\omega_0})|$ . If the input sine wave amplitude is large, in other words, if the signal-to-noise ratio is large, the filter output is mainly the attenuated sine wave with an amplitude of  $fC_1$ , where  $C_1$  is the input sine wave amplitude.

Let  $y(t)$  be the filter input and  $\varepsilon(t)$  be the filter output, the noise-to-signal power can be expressed as

$$\frac{E \varepsilon(t)^2}{E y(t)^2} = d^2. \quad (5.2)$$

The above equation relates the filter noise-to-signal power to the frequency magnitude response at  $\omega = \omega_0$ . From Fig. 14, we can see that the smaller  $\Delta\omega$  is, the smaller the noise-to-signal power is.

From Eq. (3.9), we have

$$\begin{aligned}
 d &= |H(e^{j\omega_0})| = |H(e^{j(\omega_1 - \Delta\omega)})| \\
 &= \frac{[(1 + a_1 \cos \omega_0 + \cos 2\omega_0)^2 + (a_1 \sin \omega_0 + \sin 2\omega_0)^2]^{\frac{1}{2}}}{[(1 + \rho a_1 \cos \omega_0 + \rho^2 \cos 2\omega_0)^2 + (\rho a_1 \sin \omega_0 + \rho^2 \sin 2\omega_0)^2]^{\frac{1}{2}}}
 \end{aligned} \quad (5.3)$$

Letting  $\omega_0 = \omega_1 - \Delta\omega$  in the above equation and using

$$\cos(\omega_1 - \Delta\omega) = \cos \omega_1 \cos \Delta\omega + \sin \omega_1 \sin \Delta\omega \quad (5.4a)$$

$$\cos 2(\omega_1 - \Delta\omega) = 2\cos(\omega_1 - \Delta\omega)^2 - 1 \quad (5.4b)$$

$$\sin(\omega_1 - \Delta\omega) = \sin \omega_1 \cos \Delta\omega - \cos \omega_1 \sin \Delta\omega \quad (5.4c)$$

$$\sin 2(\omega_1 - \Delta\omega) = 2\sin(\omega_1 - \Delta\omega)\cos(\omega_1 - \Delta\omega) \quad (5.4d)$$

Substituting Eq. (5.1a) and (5.1b) into Eq. (5.4), and substituting Eq. (5.4) into Eq. (5.3), we can express  $|H(e^{j\omega_0})|$  in terms of  $\Delta\omega$ . We can also calculate  $|H(e^{j\omega_0})|$  for different values of  $\Delta\omega$ . Since the filter coefficient  $a_1$  only determines the location of zero of the transfer function, it does not effect the overall shape of the frequency response for a given  $\rho$ . For a certain value of  $\rho$ , the value of  $|H(e^{j\omega_0})|$  will only be effected by  $\Delta\omega$ , not by  $a_1$ .

Combining Eqns. (5.2) and (5.3), we have

$$\frac{E_{\epsilon}(t)^2}{E_y(t)^2} = |H(e^{j(\omega_1 - \Delta\omega)})|^2 \quad (5.5)$$

Equation (5.5) expresses the noise-to-signal power in terms of the frequency error  $\Delta\omega$ . Each  $\Delta\omega$  corresponds to a certain value of the noise-to-signal power. The larger the  $\Delta\omega$ , the larger the noise-to-signal power.

Equation (5.5) is derived with the assumption that SNR is large. The higher the SNR, the smaller  $\Delta\omega$  that can be applied to Eq. (5.5); the lower the SNR, the less smaller  $\Delta\omega$  that can be applied. Denote the smallest  $\Delta\omega$  that can be applied to Eq. (5.5) by  $\Delta\omega_{\min}$ .

This value varies as the SNR varies. When  $\omega < \Delta\omega_{\min}$ , the difference between  $\frac{E_{\epsilon}(t)^2}{E_y(t)^2}$

and  $|H(e^{j(\omega_1 - \Delta\omega)})|^2$  will become significant. Equation (5.5) will no longer be valid since when  $\omega < \Delta\omega_{\min}$ , white noise dominates the filter output instead of the attenuated sine wave signal. The variance of the white noise is constant at  $\Delta\omega = 0$ , so is the noise-to-signal power  $\frac{E \varepsilon(t)^2}{E y(t)^2}$ . Denote  $\frac{E \varepsilon(t)^2}{E y(t)^2}$  at  $\Delta\omega = 0$  by *eoylimit*. The SNR will be approximately  $10\log \frac{1}{eoylimit}$ .

Table II shows  $\Delta\omega_{\min}$  and *eoylimit* for different SNRs. Signal-to-noise ratio  $SNR_0$  is calculated by  $SNR_0 = 10\log \frac{C_1^2}{2}$ . SNR-err is calculated by  $SNR-err = \frac{|SNR_0 - SNR|}{SNR_0}$ . The data in Table II is obtained in the following way.

TABLE II

CONVERGENCE CRITERION REFERENCE TABLE

$SNR_0$	$\Delta\omega_{\min}$	eoylimit	SNR	SNR-err
90	6.0e-5	1.103699e-9	89.571493	4.761189e-3
70	7.0e-4	1.103733e-7	69.571359	6.123443e-3
50	6.0e-3	1.104064e-5	49.570059	8.598820e-3
30	6.0e-2	1.106514e-3	29.560432	1.465220e-2
10	3.5e-1	1.047749e-1	9.797426	2.025740e-2
5	5.0e-1	2.869670e-1	5.421676	8.433520e-2

Suppose that the input signal of the filter is  $y(t) = C_1 \sin 2\pi f_1 t + v(t)$ , where  $f_1$  is the input sine wave frequency and  $v(t)$  is a zero-mean unit-variance white Gaussian noise. Here  $f_1 = 0.1\text{HZ}$ . We randomly choose the sampling frequency to be  $f_s = 0.7\text{HZ}$

to sample  $y(t)$ . According to Eq. (3.15), the true filter coefficient should be  $-2\cos(\frac{2\pi}{7})$ . When the filter coefficient is its true value,  $\Delta\omega = 0$ . Accordingly, when the filter coefficient is  $-2\cos(\frac{2\pi}{7} + \Delta\omega)$ , the frequency difference between the input sine wave frequency and the notch frequency of the filter is  $\Delta\omega$ .

We use  $y(t)$  as the filter input, sample it by  $f_s = 0.7\text{HZ}$ , filter it through the original ANF with  $\rho = 0.8$  and  $a_1 = -2\cos(\frac{2\pi}{7} + \Delta\omega)$ . We vary  $\Delta\omega$  from 0 to 0.5. Each  $\Delta\omega$  corresponds to a different  $a_1$ . For each  $a_1$ , we calculate the noise-to-signal power  $\frac{E \varepsilon(t)^2}{E y(t)^2}$ . Thus, each  $\Delta\omega$  is related to a certain value of the noise-to-signal power. Denote the noise-to-signal power by  $eo_y$ . The noise-to-signal power is calculated by

$$eo_y = \frac{E \varepsilon(t)^2}{E y(t)^2} = \frac{\sum_{k=1}^{N_w} \varepsilon[k]^2}{\sum_{k=1}^{N_w} y[k]^2} \quad (5.6)$$

where  $N_w$  is an integer and should be a large value in order to minimize the window effect. If  $N_w$  is large enough, the sampling frequency should not effect the value of  $eo_y$ . According to our experimental results,  $N_w = 200$  will be a reasonable value to use.

Each  $eo_y$  corresponds to a different value of  $\Delta\omega$ . We use the same  $\Delta\omega$  in Eq. (5.3) to calculate the corresponding  $|H(e^{j(\omega_1 - \Delta\omega)})|$ . Since  $|H(e^{j(\omega_1 - \Delta\omega)})|$  is only effected by  $\Delta\omega$  for a given  $\rho$ , we use  $a_1 = 1$  to calculate  $|H(e^{j(\omega_1 - \Delta\omega)})|$  with  $\rho = 0.8$ .

For different SNR, there is a range of  $\Delta\omega$  that can be applied to Eq. (5.5). In other words,  $eo_y$  should be equal to  $|H(e^{j(\omega_1 - \Delta\omega)})|^2$  for a certain range of  $\Delta\omega$ . The  $\Delta\omega_{\min}$  in Table II is obtained such that the difference between  $eo_y$  and  $|H(e^{j(\omega_1 - \Delta\omega)})|^2$  is less than  $1.0e-3$ .

Table II can be used as a convergence criterion reference table for obtaining convergence criterion. For a given SNR, if the maximum allowable frequency error  $\Delta\omega_{\max}$  of

the filter is larger than  $\Delta\omega_{\min}$  in Table II, use  $\Delta\omega = \Delta\omega_{\max}$  in Eq. (5.3) to calculate the frequency magnitude response  $|H(e^{j(\omega_1 - \Delta\omega)})|$ , use  $|H(e^{j(\omega_1 - \Delta\omega)})|^2$  as the convergence criterion *eoylemit* since Eq. (5.5) is still valid for  $\Delta\omega = \Delta\omega_{\max}$ . If  $\Delta\omega_{\max} < \Delta\omega_{\min}$ , Eq. (5.5) is not valid for  $\Delta\omega = \Delta\omega_{\max}$ , use the *eoylemit* in Table II as the convergence criterion.

### V.2.3.2 Fast Fourier transform of the input signal

Table II can be used as a convergence criterion reference provided the SNR is known. But in actual situations, the input sine wave is unknown. The amplitude of the sine wave is unknown, so is the signal-to-noise ratio.

Fast Fourier transform (FFT) provides a method to estimate the sine wave signal amplitude. Suppose a signal is  $y(t) = C_1 \sin \omega_0 t$ , where  $C_1$  is the amplitude of the sine wave.  $y[n]$  is the sampled sequence of  $y(t)$ ,

$$y[n] = C_1 \sin \omega_0 n = C_1 \cos(\omega_0 n - \frac{\pi}{2}). \quad (5.7)$$

By applying a rectangular window  $w[n]$  of length  $N_f$  to  $y[n]$ , we have the windowed sequence  $d[n]$ ,

$$d[n] = C_1 w[n] \cos(\omega_0 n - \frac{\pi}{2}). \quad (5.8)$$

To obtain the Fourier transform of  $d[n]$ , we expand Eq. (5.8) in terms of complex exponentials,

$$\begin{aligned} d[n] &= C_1 w[n] \cos(\omega_0 n - \frac{\pi}{2}) \\ &= \frac{C_1}{2} w[n] e^{j(-\frac{\pi}{2})} e^{j\omega_0 n} + \frac{C_1}{2} w[n] e^{-j(-\frac{\pi}{2})} e^{-j\omega_0 n}. \end{aligned} \quad (5.9)$$

Taking the Fourier transform of  $d[n]$  and utilizing the frequency-shifting property in Eq. (5.9), we have



$$D(e^{j\omega}) = \frac{C_1}{2} e^{j(-\frac{\pi}{2})} W(e^{j(\omega - \omega_0)}) + \frac{C_1}{2} e^{-j(-\frac{\pi}{2})} W(e^{j(\omega + \omega_0)}). \quad (5.10)$$

The maximum value  $M$  on the frequency magnitude response is  $M = \frac{C_1}{2} N_f$ . The sine wave signal amplitude is

$$C_1 = \frac{2M}{N_f} \quad (5.11)$$

If the signal contains both sine wave and white noise, Eq. (5.11) can still be used as a rough guess of the input sine wave magnitude. So our design of the new ANF will start by taking the FFT of the input signal, obtain an estimate of the SNR, then choose the closest  $SNR_0$  in Table II to obtain the convergence criterion  $eoylimit$ .

#### V.2.4. Design of criterion to reset initial conditions

In Section V.2.2, we analyze that in order to make the algorithm converge, we need to reset the initial condition of the filter coefficient estimate  $a_1[0]$  and the covariance of the initial estimate  $P[0]$ . If the algorithm starts with a small fixed-value of  $\rho$  ( $\rho = 0.001$ ) and a large  $P[0]$ , within only a few recursions the filter coefficient will pass into region D or E on the error-performance surface. But once it passes into region D or E, it stays there without going further toward the true filter coefficient. This is because at that time  $P[t]$  is already too small to update  $a_1$  fast enough to the true value.

From Section V.2.3, we know that each value of filter coefficient corresponds to a value of noise-to-signal power  $eo_y$ . Thus, at the time  $a_1$  stays in region D or E with very small variations,  $eo_y$  also changes slightly.

From Section V.2.3, we also know that by using FFT technique and Table II, one can always obtain a convergence criterion  $eoylimit$ . This is the value of the noise-to-signal power at the desired filter coefficient.

Let  $eoyvar$  reflect the variation of  $eoy$  and how fast  $eoy$  can move toward the convergent criterion  $eoylimit$ , in other words,  $eoyvar$  reflects how fast  $a_1$  can converge to the true value.

In our design,  $eoyvar$  is defined by

$$eoyvar = \frac{\sum_{i=k_1-N_1}^{k_1-1} eoydiff(i)}{N_1}, \quad k_1 = 6, 7, 8, \dots$$

where  $eoydiff(i)$  is calculated by

$$eoydiff(i-1) = Eeoy(t_i) - Eeoy(t_{i-1}), \quad i = 2, 3, 4, \dots$$

and  $Eeoy(t_i)$  is

$$Eeoy(t_i) = \frac{\sum_{k=(i-1)N+1}^{iN} eoy(k)}{N}, \quad i = 1, 2, 3, \dots$$

and  $N = 10, N_1 = 5$ .

Suppose the algorithm can be said to have a slow convergent rate if after  $Nlimit$  number of recursions, it can not converge. Here  $Nlimit$  is an integer. Define  $eoycom(k, Nlimit)$  as

$$eoycom(k, Nlimit) = eoy(t) + (Nlimit - k)eoyvar \quad (5.12)$$

where  $k$  denote the  $k$ th recursion. The  $eoyvar$  reflects the current variation of  $eoy$ ; in other words, it reflects the current convergent rate. The  $eoycom(k, Nlimit)$  gives an estimate that based on the current convergent rate, after  $Nlimit$  number of recursions, how small  $eoy$  can reach. For each recursion, compare  $eoycom(k, Nlimit)$  with  $eoylimit$ . If  $eoycom(k, Nlimit) > eoylimit$ , that means  $eoy$  can not reach the desired convergent cri-

terion even after  $Nlimit$  number of recursions. Since if after  $Nlimit$  number of recursions, the algorithm can not converge, it can be considered as having a slow convergent rate and the algorithm needs to reset its initial conditions of  $a_1[0]$  and  $P[0]$  since this is the time  $P[t]$  is already too small to update  $a_1$  to the true value.

The *eoycom* is an application dependent parameter. Its value depends on the definition of  $Nlimit$ . Different applications have different definition of slow convergent rate and  $Nlimit$ .

### V.3. A FAST ALGORITHM FOR ADAPTIVE NOTCH FILTERING

Based on the designs in the previous sections, we present the following new ANF algorithm. Fig. 15 shows the flow chart of the new algorithm.

The new algorithm starts from taking the FFT of the input signal, thus obtaining an estimate of the signal-to-noise ratio as explained in Section V.2.3.2. By using the SNR and the maximum allowable frequency error  $\Delta\omega_{max}$ , we can obtain the convergence criterion *eoylimit* as described in Section V.2.3.1. We will compare the noise-to-signal power *eoy* with the convergent criterion *eoylimit* in each recursion. If it is less than *eoylimit*, the algorithm converges.

We set the initial estimate of filter coefficient to be  $a_1[0] = 0$  and the initial covariance to be  $P[0] = 100$ . Set  $p = 0.001$ . All the other initial conditions are set according to the original algorithm. A large initial covariance of  $P[0] = 100$  makes the filter coefficient pass into region D or E on the error-performance surface only within a few recursions. A small fixed value of  $p$  improves the error-performance surface as explained in Section V.2.1.

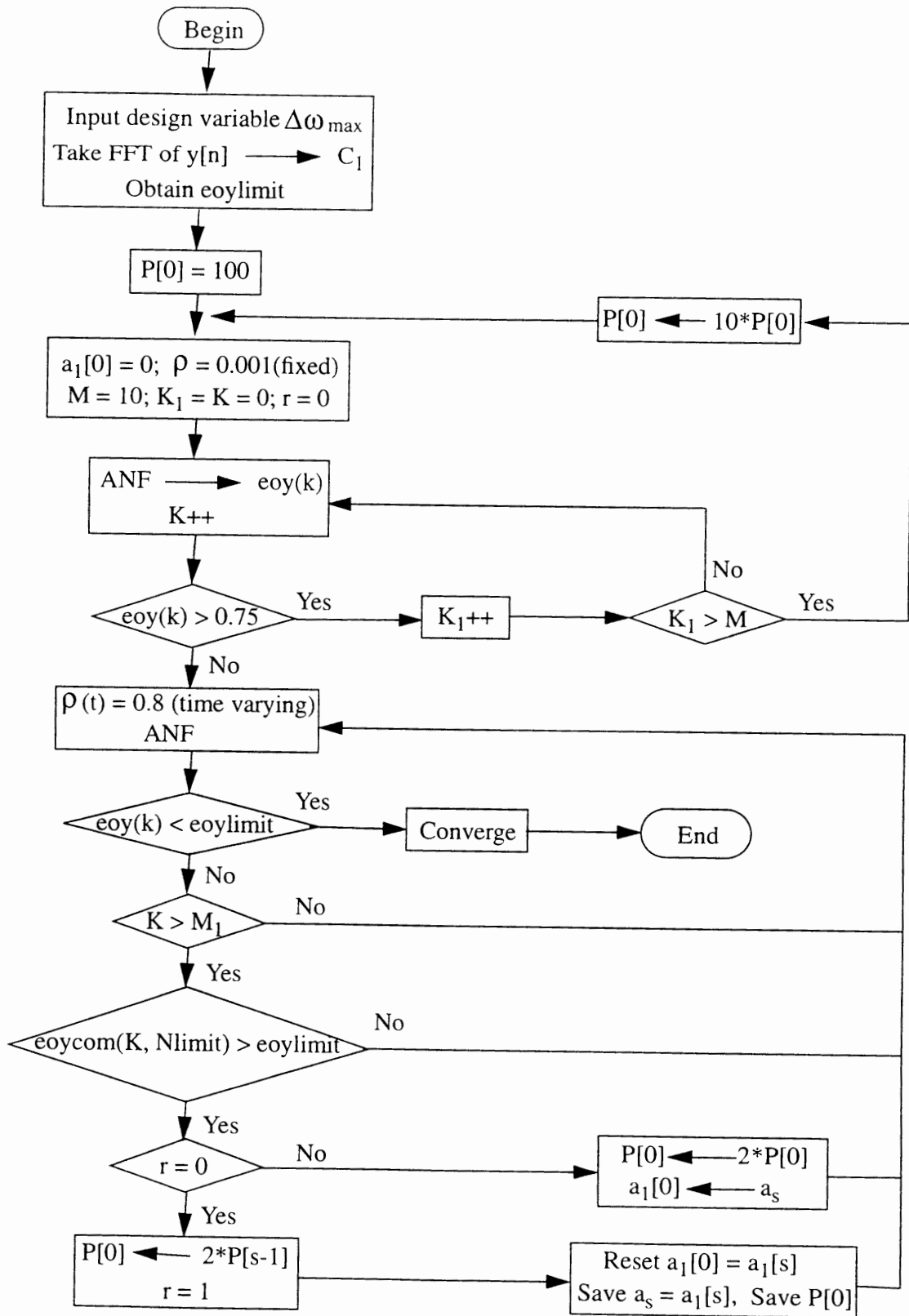


Figure 15. New ANF algorithm flowchart

As the number of recursions increases, we check if  $eoy < 0.75$  or not. If it is not less than 0.75 after  $M$  recursions,  $P[0]$  is increased by 10 and we start the algorithm again. This procedure will be repeated until a  $P[0]$  is found such that it can make  $eoy < 0.75$  within  $M$  recursions. As discussed in Section V.2.2, a large  $P[0]$  can always makes the filter coefficient pass into region D or E of the error-performance surface only within a few recursions. In the algorithm we use  $M = 10$ .

As soon as we detect  $eoy(k) < 0.75$ , we increase the value of  $\rho$  to be 0.8 and use the time-varying  $\rho$  scheme as proposed in the original algorithm. Since  $eoy(k)$  less than 0.75 means the filter coefficient is near its true value (in region D or E on the performance surface), a larger value of  $\rho$  at this time can render faster convergent rate and narrower notch that can minimize the distortion to the wideband component of the input signal by the filter. Near the true filter coefficient, the slope of the error-performance surface is steeper for  $\rho = 0.8$  than for  $\rho = 0.001$ . An error-performance surface with a steeper slope will have a faster convergent rate if the filter coefficient is near the true value. The reason for using 0.75 as a comparison criterion instead of using a value less than 0.75 is because at the time  $eoy < 0.75$ , the filter coefficient  $a_1$  is already near the true value; if  $\rho$  is not increased, the error-performance surface is still the one with  $\rho = 0.001$ . Notice from Fig. 12 that near the true filter coefficient, the slope of the performance surface with  $\rho = 0.001$  is not as steep as that with  $\rho = 0.8$ . Thus, the convergent rate will be slower if  $\rho$  is not increased at this time. Results from experiments show that 0.75 is a good value to choose. In our design, we use  $eoy < 0.75$  as a condition to change the value of  $\rho$ .

For each recursion,  $eoy(k)$  is compared with the convergent criterion  $eoylimit$ . If  $eoy(k) < eoylimit$ , the algorithm converges; otherwise, we check whether it is the time to reset the initial conditions of  $a_1[0]$  and  $P[0]$  or not.

As explained in Section V.2.4, we check the resetting condition by comparing  $eoycom(k, Nlimit)$  with  $eoylimit$ . If it is larger than  $eoylimit$ , that means according to

the current convergent rate,  $eoy$  can not reach the convergent criterion  $eoylimit$  even after  $Nlimit$  number of recursions because  $P[t]$  is already too small to update  $a_1$  to the convergent value; this is the time to reset the initial conditions.

Suppose at the  $s$ th recursion the initial conditions need to be reset, where  $s$  is an integer. The initial conditions are reset such that  $a_1[0] = a_1[s]$  and  $P[0] = 2P[s-1]$ .  $P[s-1]$  is the value of  $P[t]$  at  $(s-1)$ th recursion. Since for each recursion, the condition  $eoycom(k, Nlimit) < eoylimit$  will be checked to decide whether the convergent rate is too slow or not, and  $s-1$  is the most recent recursion at which the convergent rate is still fast enough. So at the  $(s-1)$ th recursion,  $P[s-1]$  is still a proper covariance to update  $a_1$ . As at the  $s$ th recursion, the convergent rate is detected to be too slow because of the small  $P[s]$ , we will exponentially increase  $P[s]$  to be the doubled value of  $P[s-1]$ . The increased value is given to the new  $P[0]$ . At the time  $a_1[0]$  and  $P[0]$  are reset,  $a_1[s]$  and  $P[s-1]$  are saved as well. The reason for saving them is because we are not sure whether  $2P[s-1]$  is a proper increasement value for  $P[s]$  or not. This will be known after we rerun the algorithm and compare  $eoycom(k, Nlimit)$  with  $eoylimit$ . If it is larger than  $eoylimit$ , then that means using the current increasement value of  $P[0]$ ,  $eoy$  can not converge to the desired value  $eoylimit$  even after  $Nlimit$  recursions.  $2P[s-1]$  is still too small to update  $a_1$  to its convergent value. Thus, reset  $P[0]$  again to be the doubled value of the current  $P[0]$ . Now,  $P[0] = 4P[s-1]$ . We will use  $a_1[0] = a_1[s]$  and  $P[0]$  to run the ANF algorithm again. The above steps will be repeated until a proper  $P[0]$  is found to make  $a_1$  converge within  $Nlimit$  recursions.

The advantage of the new ANF algorithm is its fast convergent rate. By redesigning the error-performance surface and resetting the initial conditions, the new algorithm will have better sensitivity to the presence of the input sine wave signal, faster convergent rate and more accurate results than the original ANF algorithm. Simulation results in the following section will show the superior performance of the new ANF algorithm over

the original ANF algorithm.

#### V.4 SIMULATION RESULTS

We tested the new ANF algorithm under different SNRs and sampling frequencies. Simulation results show that the new ANF algorithm has a faster convergent rate than the original ANF algorithm. The following are two representative examples of the simulation results. In each example, we will compare the convergent rate with the original ANF algorithm. In the examples, we use  $N_f = 128$ ,  $N_w = 200$ ,  $N = 10$ ,  $N_1 = 5$ ,  $M = 10$ ,  $\Delta w_{\max} = 1.0e-6$ ,  $Nlimit = 2000$ .

Example 1. This example is to show that for the convergence cases of the original ANF, under the same conditions, the new algorithm has a faster converge rate than the original algorithm.

Assume the input signal is  $y(t) = C_1 \sin 2\pi f_1 t + v(t)$ , where  $f_1 = 0.1\text{HZ}$  and  $v(t)$  is a zero-mean unit-variance white Gaussian noise. The sampling frequency ratio is  $f_s/f_1 = 19.1$ . We test the algorithm under different SNRs. Results are shown in Table III. Table IV shows the results of the original ANF under the same conditions.

We set the convergence condition to be that the frequency error is less than  $1.0e-6$ . Table III and IV show that our new algorithm has a faster convergent rate for different SNRs than the original algorithm. Especially for low signal-to-noise ratio condition, the convergent rate is dramatically improved compared to that of the original algorithm. For  $SNR = 10$ , it only takes 609 recursions for the new algorithm to converge while even after 2000 recursions, the original algorithm still has an frequency error of 0.130931.

TABLE III

NEW ANF RESULTS FOR  $f_s/f_1 = 19.1$ 

SNR	$f_s/f_1$	f-error	Number of recursions
50	19.1	$< 1.0e-6$	315
30	19.1	$< 1.0e-6$	390
10	19.1	$< 1.0e-6$	609

TABLE IV

ORIGINAL ANF RESULTS FOR  $f_s/f_1 = 19.1$ 

SNR	$f_s/f_1$	f-error	Number of recursions
50	19.1	$< 1.0e-6$	1880
30	19.1	$< 1.0e-6$	1926
10	19.1	$> 1.0e-1$	$> 2000$



TABLE V

NEW ANF RESULTS FOR  $f_s/f_1 = 7.0$ 

SNR	$f_s/f_1$	f-error	Number of recursions
50	7.0	$< 1.0e-6$	217
30	7.0	$< 1.0e-6$	226
10	7.0	$< 1.0e-6$	226

TABLE VI

ORIGINAL ANF RESULTS FOR  $f_s/f_1 = 7.0$ 

SNR	$f_s/f_1$	f-error	Number of recursions
50	7.0	$< 1.0e-6$	130,763
30	7.0	$> 2.0e-1$	$> 1e+4$
10	7.0	$> 3.5e-1$	$> 1e+6$

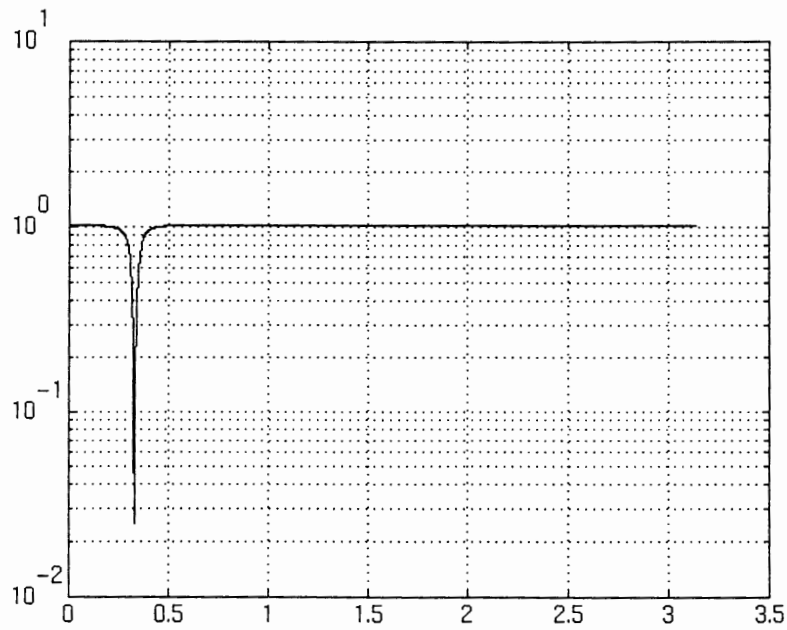
Example 2. This example is to show that for the slow or non-convergence cases of the original ANF, under the same conditions, the new algorithm not only converges but also has a fast convergent rate.

The input signal is the same as that of the previous example, here the sampling ratio is  $f_s/f_1 = 7.0$ . Table V and VI show the results under different SNRs and the comparison with the original ANF under the same conditions.

For high SNR, for example,  $SNR = 50$ , it takes 130,763 recursions for the original ANF to converge while it only take 217 recursions for the new ANF to converge. The

convergent rate of the new ANF is about 600 times faster than that of the original ANF. For lower SNR, for example,  $SNR = 10$ , it takes 226 recursions for the new algorithm to converge while for the original algorithm, even after 1,000,000 recursions, there still has an frequency error of 0.367996. From the experiments, the new algorithm can converge under a wide range of sampling frequency and signal-to-noise ratio.

Results from simulation show that the fast convergent rate is the biggest advantage of the new algorithm. The new ANF algorithm has all the advantages of the original algorithm, in addition, it overcomes the disadvantages of the original algorithm. It has a very narrow and accurate notch. A representation plot is shown in Fig. 16 for  $SNR = 30$  and  $f_s/f_1 = 19.1$ .



**Figure 16.** Transfer function of the convergent filter for one sine wave in additive white noise.

## CHAPTER VI

### CONCLUSIONS AND FUTURE WORK

#### VI.1 SUMMARY AND CONCLUSIONS

In the preceding chapters we have presented a new ANF algorithm. The algorithm is optimal in the sense that it can meet the desired properties of an adaptive notch filter. It has a fast convergent rate and accurate results. Four new designs that achieve these goals have been demonstrated: first, increasing the sensitivity of the algorithm by applying a wider notch (smaller  $\rho$ ) at the start of the data processing; second, improving the convergent rate by properly resetting the initial filter coefficient estimate and the covariance of this estimate; third and fourth, achieving accurate results and fast convergent rate by developing a new convergence criterion and a criterion to reset the initial conditions. By applying the above new designs to the original algorithm, the convergent rate is greatly improved. From the results of this thesis, we present the following conclusions:

First, the error-performance surface is an important key for the analysis and understanding of the performance of the ANF algorithm. As the ANF algorithm is of RPE type, it is based on local minimum search. Error-performance surface can provide direct information on the convergence behavior of the algorithm.

Second, the algorithm is very sensitive to the initial conditions. If the initial estimate of the filter coefficient is far from the true value, the initial covariance needs to be large in order to ensure convergence; if the initial estimate of filter coefficient is close to the true value, the covariance should be small in order to ensure convergence.

Finally, the filter output-to-input power (noise-to-signal power) reflects the fre-

quency error. From the variation rate of the noise-to-signal power, we can predict the convergent rate, and thus decide whether it is necessary to reset the initial conditions of the algorithm or not, as well as when the algorithm converges.

All of these aspects are important and necessary to design a RPE type of ANF algorithm with a fast convergent rate and accurate results. However, in this thesis we only concentrated on the study and design of the ANF algorithm applying one input sine wave. In the next section we suggest areas for future study on ANF applying multi input sine waves.

## VI.2 SUGGESTIONS FOR FUTURE WORK

In this thesis we have demonstrated, in principal, a fast ANF algorithm for eliminating one input sine wave plus white noise. In our research we have investigated all aspects that are important and necessary to design a fast ANF algorithm. But we have not applied the algorithm with multi input sine waves plus white noise yet. As a result of our efforts we see the following topics as those of greatest importance for ANF algorithm applying multi input sine waves.

First, the error-performance surface is still an important key for understanding the algorithm behavior; second, a cascaded ANF structure can be investigated. Since by taking the FFT of the input signals, we can obtain the frequency estimate and amplitude estimate of each sine wave signal. We can apply a one dimensional ANF to the input signal with the initial filter coefficient estimate close to the true filter coefficient of the input sine wave with the highest SNR and use a small initial covariance. After cancelling one sine wave, apply another one dimensional ANF to eliminate the input sine wave with the second highest SNR. This cascaded ANF process continues until all the input sine waves are eliminated. The cascaded structure provides an effective way to use multi-processors for real-time processing. With VLSI, this approach appears to be very promising. How-

ever, if the input sine wave frequencies are very close to each other, how good this structure can perform needs future work and investigation.

## REFERENCES

- [1] Arye Nehorai, "A minimal parameter adaptive notch filter with constrained poles and zeros." *IEEE Trans. Acoust., Speech, Signal Processing*, Vol. ASSP-33, No. 4, Aug. 1985.
- [2] P. A. Thompson, "A constrained recursive adaptive filter for enhancement of narrow-band signals in white noise." *Proc. 12th Asilomar Conf. Circuits, Syst. Comput.*, Pacific Grove, CA, Nov. 1978, pp. 214 - 218.
- [3] S. Y. Kung and D. V. Bhaskar Rao, "An unbiased adaptive method for retrieval of sinusoidal signals in colored noise." *Proc. 1982 IEEE Int. Conf. Acoust., Speech, Signal Processing*, Paris, May 1982, pp. 663 - 666.
- [4] D. V. Bhaskar Rao and S. Y. Kung, "Adaptive notch filtering for the retrieval of sinusoids in noise." *IEEE Trans. Acoust., Speech, Signal Processing*, Vol. ASSP-32, pp. 791-802, Aug. 1984.
- [5] B. Friedlander and J. O. Smith, "Analysis and performance evaluation of an adaptive notch filter." *IEEE Trans. Inform. Theory*, Vol. IT-30, PP. 283-295, Mar, 1984.
- [6] P. Stoica and T. Soderstrom, "On the parsimony principle." *Int. J. Control*, Vol. 36, no. 3, pp. 408 - 418, 1982.
- [7] L. Ljung and T. Soderstron, *Theory and Practice of Recursive Identification*. Cambridge, MA, MIT Press, 1983.
- [8] Alan V. Oppenheim, Ronald W. Schaffer, *Discrete-time Signal Processing*. Englewood Cliffs, NJ, Prentice-Hall, Inc, 1989.