12-10-1996

Equalization of a Non-linear Phase of a Low-pass Filter

Rabih Khodor
Portland State University

THESIS APPROVAL

The abstract and thesis of Rabih Khodor for the Master of Science in Electrical and Computer Engineering were presented December 10, 1996, and accepted by the thesis committee and the department.

COMMITTEE APPROVALS: [redacted]

Yih–Chyun Jenq, Chair

[redacted]

Andy Fraser

[redacted]

Richard Crittenden
Representative of the Office of Graduate Studies

DEPARTMENT APPROVAL: [redacted]

Rolf Schaumann, Chair
Department of Electrical Engineering

* *

ACCEPTED FOR PORTLAND STATE UNIVERSITY BY THE LIBRARY

by [redacted] on 12 February 1997

ABSTRACT

An abstract of the thesis of Rabih Khodor for the Master of Science in Electrical and Computer Engineering presented December 10, 1996.

Title: Equalization of a Non–Linear Phase of a Low–Pass Filter

In practice, an IIR filter can distort the information content of the signal because of its inherent non–linear phase characteristics introduced through the design of the filter. If the the receiver of the signal is the human ear, e.g., when a speech or music signal is to be processed, phase distortion is quite tolerable. But, in other applications it can be rejected as phase characteristics is required to be fairly linear. Applications of this type include data transmission, where the signal is to be interpreted by digital hardware, and image processing, where the signal is used to reconstruct an image that is to be interepted by the human eye.

This thesis deals with the problem by introducing an all–pass equalizing filter, whose magnitude response is unity, in cascade with the given filter to produce the same filter magnitude but with a linear–phase instead. This accomplished by solving the problem of minimizing the approximation error for the group delay by using a suitable optimization algorithm, the free parameters (poles and zeros) are varied in a way to minimize the approximation error according to the assumed error criterion. Those set of parameters found after optimization determines the desired linear filter.

EQUALIZATION OF A NON–LINEAR PHASE OF A LOW–PASS FILTER

by
RABIH KHODOR

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

MASTER OF SCIENCE
in
ELECTRICAL AND COMPUTER ENGINEERING

Portland State University
1997

TABLE OF CONTENTS

PAGE

ACKNOWLEDGEMENTS

I would like to give special thanks to my advisor, Dr. Yih–Chyun Jenq, for his guidance and constructive criticism throughout my thesis preparations and research work. He also helped by checking my results and concluding my thesis.

I would also like to thank Dr. Andrew Fraser and Dr. Richard Crittenden for accepting to be on my committee .

I wish to thank Mrs. Shirley Clark for being such a helpful and supportive office secretary. Also, I like to offer my gratitude to the staff of the Electrical Engineering department for their patience and support.

Finally, I wish to dedicate this thesis to my family in Lebanon and especially my brother who was a torch of knowledge that enlightened my path, throughout my years of education, to become what I am now. May God bless his soul and may he rest in peace.

Portland, Oregon Rabih Khodor

December 1996

LIST OF FIGURES

FIGURE **PAGE**

CHAPTER I

INTRODUCTION

1.1 Digital Filter

A filter, As a circuit block, having an input and an output, that restricts the frequency range of the signals at the input. For example, we may have an input signal with a range of frequencies from 2 Hz to 36,000 Hz going into an amplifier, but only we want to use frequencies from 20 to 20,000 Hz, so we include a filter to remove unwanted frequencies.

Due to the nonlinear phase response of a filter some frequencies are delayed more than others. If two sine–waves, of 1kHz and 2 kHz, applied to the input of an ideal low–pass filter ($\omega_c > 2$ kHz) and because phase–shift varies linearly with frequency, the 2 kHz input suffers twice the phase shift of the 1 kHz input. But, both signals are delayed by the same time period as shown in fig 1.1 below. Therefore, if signals were in phase at the input of the filter they are still in phase at the output. Linear phase response is important in data transmission, to prevent pulse distortion. The phase response in Infinite Impulse Recursive (IIR) filters are in general quite nonlinear because of two reasons. First, the design methods used in Butterworth, Chebyshev, inverse Chebyshev, and Elliptic approximations are inherently nonlinear–phase approximations [18]. Second, the warping effect tends to increase the nonlinearity of the phase response. As a result of this, the group delay tends to vary with frequency and the application of these filters tends to introduce delay distortion. Constant group delay filters can sometimes be designed by using constant–delay approximations such as the Bessel approximation with design methods that maintain the linearity

in the phase response, for example, the invariant–impulse–response method. But, a constant delay and a given loss specifications are usually hard to achieve simultaneously.

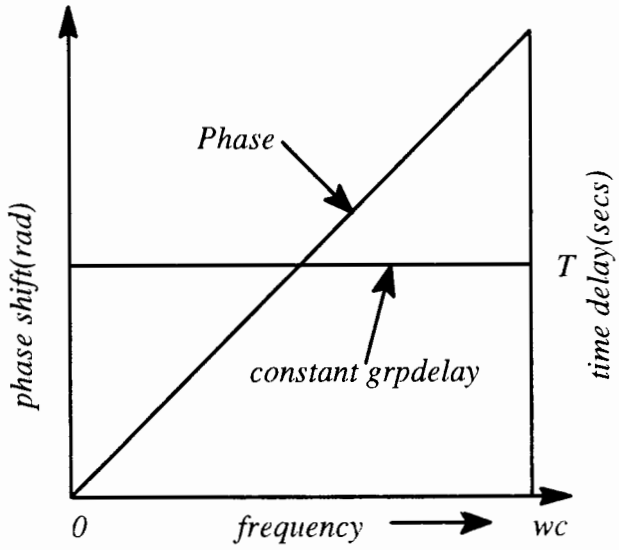

Figure 1–1 Phase shift and time delay .vs. frequency

1.2 Recursive (IIR) Instead of Non–Recursive Digital Filter

Recursive digital filters are commonly referred to as infinite impulse response (IIR) filters. The term *recursive* intrinsically means that the output of the digital filter, $y(n)T$, is computed using the present input, $x(n)T$, and previous inputs and outputs, namely, $x(n-1)T, x(n-2)T, \ldots, y(n-1)T, y(n-2)T, \ldots$, respectively. Whereas, Non–recursive digital filters [24] have a weighting sequence (impulse response), $g(n)T$, which is finite in length, and consequently this type of filter is commonly referred to as a finite impulse response (FIR) filter. The term *non–recursive* intrinsically means that the output of the filter, $y(n)T$, is computed using the present input, $x(n)T$, and the previous inputs,

$x(n-1)T, x(n-2)T, \ldots$,and furthermore the filter has no inherent feedback, which means that previous output values, $y(n-1)T, y(n-2)T, \ldots$, are not used in the computation of $y(n)T$. Recursive digital filters are generally more economical in execution time and storage requirements compared with their non–recursive counterparts. However, some types of recursive digital filter have non linear phase characteristics which may produce unacceptable waveform distortion. This type of filter is an attractive, economical and useful recursive digital filter in some simple applications.

A non–recursive filter has some advantages

(1) they are always stable because:

(a) there is no feedback between output and input, and

(b) the impulse response is finite; and

(2) the non–recursive digital filter can have a linear phase characteristic, if the impulse response satisfies the symmetry conditions, thereby eliminating the possibility of phase distortion in the output waveform.

The other disadvantage of non–recursive filters is

(1) compared with a recursive counterpart, a non–recursive filter will generally use more memory and arithmetic for its implementation

1.3 The All–Pass Filter

Rather than think in terms of phase shift, many filter engineers concentrate on the group delays introduced by filters. In many ways , this is more logical since a filter with a delay that is independent of frequency will pass a waveform without distorting it. This is a very simple criterion to have for a filter since most filters [22] introduce delay distortion, that is, their delay is a function of frequency, various tricks are used to cancel, at least in part, this time–dependent delay.

The all–pass filter is the answer to the question 'When is a filter not a filter?'. In audio frequency work, it often happens that a filter is used that has precisely the required amplitude response but introduces a delay that is frequency dependent. To correct this we need a 'filter' that has a flat amplitude response together with a counteracting delay. It is not possible tocancel out a delay, since that would involve going backwards in time. The idea is to add to an existing delay, so that the overall delay is independent of frequency. This is the purpose of all–pass filter.

1.4 Delay Compensation

The design of constant–delay digital filters satisfying given loss specifications is almost accomplished in two steps. First a filter is designed satisfying the loss specifications ignoring the group delay. Then a delay equalizer is designed which can be used in cascade with the filter to compensate for variations in the group delay of the filter.

Let $H_F(z)$ and $H_E(z)$ be the transfer functions of the filter and equalizer, respectively. The group delays of the filter and equalizer are given by

$$\tau_F(\omega) = -\frac{d\theta_F(\omega)}{d\omega} \quad \text{and} \quad \tau_E(\omega) = -\frac{d\theta_E(\omega)}{d\omega}$$

respectively, where

$$\theta_F(\omega) = \arg H_F\left(e^{j\omega T}\right) \quad \text{and} \quad \theta_E(\omega) = \arg H_E\left(e^{j\omega T}\right)$$

The overall transfer function of the filter–equalizer combination is

$$H_{FE}(z) = H_F(z)H_E(z)$$

Hence

$$\left|H_{FE}\left(e^{j\omega T}\right)\right| = \left|H_F\left(e^{j\omega T}\right)\right| \left|H_E\left(e^{j\omega T}\right)\right|$$

and the overall phase response is

$$\theta_{FE}(\omega) = \theta_F(\omega) + \theta_E(\omega)$$

Then from the previous equation the overall group delay of the filter–equalizer combination can be written as

$$\tau_{FE}(\omega) = \tau_F(\omega) + \tau_E(\omega)$$

Therefore, a digital filter that satisfies prescribed loss specifications and has constant group delay with respect to some passband $\omega_{p1} \leq \omega \leq \omega_{p2}$ can be designed using the following steps:

1. Design a filter satisfying the loss specifications using bilinear transformation and prewarping method.
2. Design an equalizer with

$$\left|H_E\left(e^{j\omega T}\right)\right| = 1 \qquad \text{for } 0 \leq \omega \leq \frac{\omega_s}{2}$$

and

$$\tau_E(\omega) = \tau - \tau_F(\omega) \qquad \text{for } \omega_{p1} \leq \omega \leq \omega_{p2}$$

where τ is a constant.

From step 2, $H_E(z)$ must be an *allpass* transfer function of the form

$$H_E(z) = \prod_{j=1}^{M} \frac{1 + C_{1j}z + C_{0j}z^2}{C_{0j} + C_{1j}z + z^2}$$

The equalizer can be designed by finding a set of values for C_{0j}, C_{1j}, τ, and M such that; (a) $\tau_E(\omega) = \tau - \tau_F(\omega)$ is satisfied to within a prescribed error in order to achieve approximately constant group delay with respect to the passband, and (b) the poles of $H_E(z)$ are inside the unit circle of the z plane to ensure that the equalizer is stable. Equalizers can be designed by using optimization methods.

1.5 Design of Recursive Delay Equalizers

Consider a filter characterized by the transfer function

$$H_F(z) = H_0 \prod_{j=1}^{J} \frac{a_{0j} + a_{1j}z + a_{2j}z^2}{b_{0j} + b_{1j}z + b_{2j}z^2} \tag{1.1}$$

The group delay of the filter [22] is given by

$$\tau_F(\omega) = - \frac{d\theta_F(\omega)}{d(\omega)} \tag{1.2}$$

Where

$$\theta_F(\omega) = \arg H_F(e^{j\omega T}) \tag{1.3}$$

From Eqs. (1.1) and (1.2), we can show that

$$\tau_F(\omega) = - T \sum_{j=1}^{J} \frac{\tilde{N}_j(\omega)}{N_j(\omega)} + T \sum_{j=1}^{J} \frac{\tilde{D}_j(\omega)}{D_j(\omega)} \tag{1.4}$$

Where

$$\tilde{N}_j(\omega) = a_{2j}^2 - a_{0j}^2 + a_{1j}(a_{2j} - a_{0j})\cos(\omega T)$$

$$N_j(\omega) = (a_{2j} - a_{0j})^2 + a_{1j}^2 + 2a_{1j}(a_{2j} + a_{0j})\cos(\omega T) + 4a_{0j}a_{2j}\cos^2\omega T$$

$$\tilde{D}_j(\omega) = b_{2j}^2 - b_{0j}^2 + b_{1j}(b_{2j} - b_{0j})\cos(\omega T)$$

$$D_j(\omega) = (b_{2j} - b_{0j})^2 + b_{1j}^2 + 2b_{1j}(b_{2j} + b_{0j})\cos(\omega T) + 4b_{0j}b_{2j}\cos^2\omega T$$

The group delay of the filter can be equalized with respect to a frequency range $\omega_1 \le \omega \le \omega_I$ by connecting an all–pass delay equalizer in cascade with the filter.

Let the transfer function of the equalizer be

$$H_E(z) = \prod_{j=1}^{M} \frac{1 + c_{1j}z + c_{0j}z^2}{c_{0j} + c_{1j}z + z^2} \tag{1.5}$$

The group delay of the equalizer can be obtained as

$\tau_e(c, \omega) = - d\theta_e(c, \omega)/d\omega$

Where $\theta_e(c, \omega) = \arg H_e\left(e^{j\omega T}\right)$

Hence $$\tau_e(c, \omega) = 2T \sum_{j=1}^{M} \frac{\tilde{C}_j(\omega)}{C_j(\omega)} \qquad (1.6)$$

Where $\tilde{C}_j(\omega) = 1 - c_{0j}^2 + c_{1j}(1 - c_{0j})\cos\omega T$

$C_j(\omega) = (1 - c_{0j})^2 + c_{1j}^2 + 2c_{1j}(1 + c_{0j})\cos\omega T + 4c_{0j}\cos^2\omega T$

$\mathbf{c} = [\, c_{01}, c_{11}, c_{02}, c_{12}, \ldots, c_{1M} \,]^T$

The equalizer is stable if and only if the transfer function coefficients satisfy the relations

$c_{0j}<1,\ c_{1j} - c_{0j}<1,\ c_{1j} + c_{0j} > -1$ For j=1,2,3,. . . , M .

The region of stability in the (c_0,c_1)–plane is illustrated in fig 1.2. This may be referred to as the feasible region of the parameter space.

The group delay of the filter /equalizer combination can be expressed as

$$\tau_{FE}(c,\omega) = \tau_F(\omega) + \tau_E(c,\omega)$$

Where $\tau_F(\omega)$ and $\tau_E(c,\omega)$ are given by Eqs. (1.4) and (1.6) , respectively.

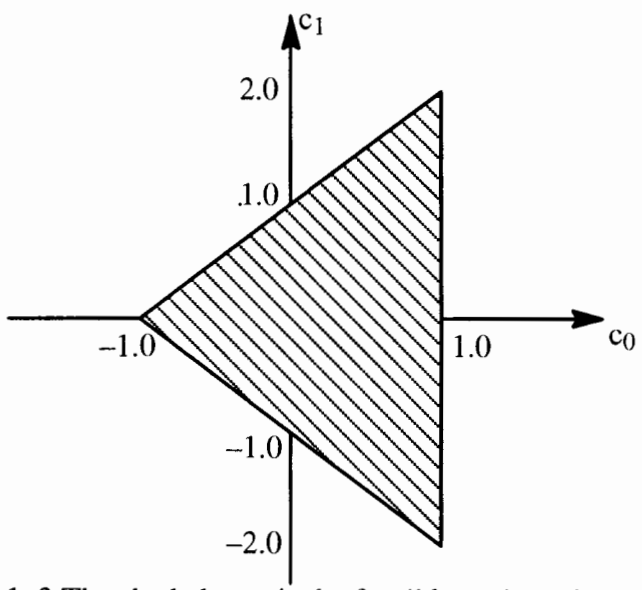

Figure 1–2 The shaded area is the feasible region of (c_0,c_1)–plane

The required equalizer can be designed by solving the optimization problem

$$\text{Minimize}_{\,x}\, E(X)$$

Where

$$E(\mathbf{x}) = \sum_{i=1}^{L} |e_i(\mathbf{x})|^{2q} \tag{1.7}$$

$$e_i(\mathbf{x}) = \frac{1}{T}\tau_{FE}(\mathbf{x}, \omega_i) - \tau_0 \tag{1.8}$$

$$\mathbf{X} = [c^T, \tau_0]^T \,,\ \tau_0 = \frac{\tau}{T} \tag{1.9}$$

And $\omega_1 \le \omega_i \le \omega_L$

The gradient of $|e_i(\mathbf{x})|$, which is required for the evaluation of $\nabla\Psi(\mathbf{x})$ can be obtained by using the derivatives of $e_i(\mathbf{x})$, and Maple V to solve for the following equations:

$$\frac{\partial e_i(\mathbf{x})}{\partial c_{0l}} = \frac{U_{0l} + U_{1l}\cos\omega_i T + U_{2l}\cos^2\omega_i T + U_{3l}\cos^3\omega_i T}{\left[c_l(\omega_i)\right]^2} \tag{1.10}$$

$$\frac{\partial e_i(\mathbf{x})}{\partial c_{1l}} = \frac{V_{0l} + V_{1l}\cos\omega_i T + V_{2l}\cos^2\omega_i T + V_{3l}\cos^3\omega_i T}{\left[c_l(\omega_i)\right]^2} \tag{1.11}$$

$$\frac{\partial e_i(\mathbf{x})}{\partial \tau_0} = -1 \tag{1.12}$$

For $l=1,2,3,...,M$ and $i=1,2,3,...,L$

where

$$U_{0l} = 4[(1 - c_{0l})^2 - c_{0l}c_{1l}^2] \,,\ U_{1l} = -2c_{1l}(1 + 6c_{0l} + c_{0l}^2 + c_{1l}^2) \tag{1.13}$$

$$U_{2l} = -8(1 + c_{0l}^2 + c_{1l}^2),\ U_{3l} = -8c_{1l} \tag{1.14}$$

$$V_{0l} = -4c_{1l}(1 - c_{0l})(1 + c_{0l}), \tag{1.15}$$

$$V_{1l} = -2(1 - c_{0l})(1 + 6c_{0l} + c_{0l}^2 + c_{1l}^2) \tag{1.16}$$

$$V_{2l} = 0,\ V_{3l} = 8(1 - c_{0l})c_{0l} \tag{1.17}$$

The chapter that follows will classifies and introduce some optimization techniques for Multi–dimensional methods (Indirect and Direct). For instance, Unconstrained, Lagrange Multipliers (Indirect), Steepest Descent, Fletcher–Reeves, and Davidon–Fletcher–Powell (Direct).

CHAPTER II

CLASSIFICATION OF OPTIMIZATION ALGORITHMS

2.1 Optimization Techniques

Numerical optimization algorithms can be classified according to the number of design variables and further according to the nature of the design space. Figure 2–1. shows a small part of the different groups of un–constrained optimization techniques

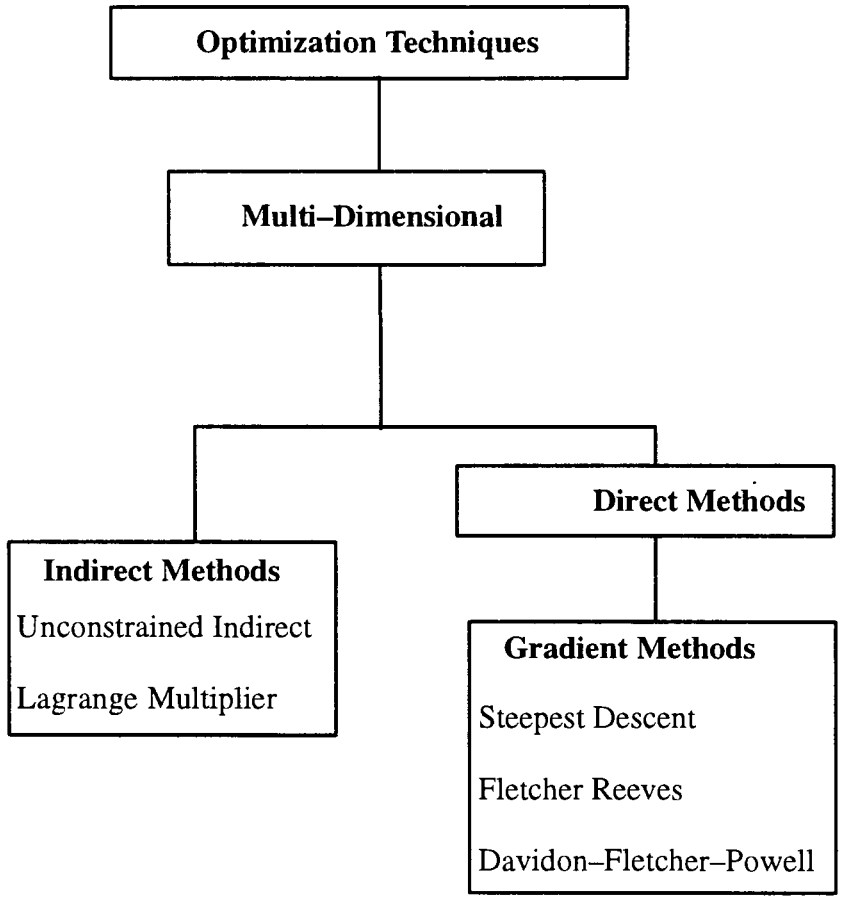

Figure 2–1 The group of Optimization techniques

2.2 MULTIDIMENSIONAL SEARCH METHODS

At first thought the designer may think that the difference between multi–dimensional search techniques and single–dimensional search techniques is only one of increased effort, and if the designer were willing to spend a bit more time in the calculation process he or she could extend single–variable methods to N–dimensional methods. Unfortunately, this is not true since the nature of multi–dimensional space is considerably different from one–dimensional space. For one thing, as the number of dimensions increases, the likelihood that the objective function will be unimodal decreases. In addition, the size of multi–dimensional space is overwhelming. For instance, if in one–dimensional space 19 evaluations are needed to achieve f=0.1, then 361 evaluations will be required to achieve the same accuracy in two dimensions, 6859 in three dimensions, 130,321 in four dimensions, and 2,476,099 in five dimensions. Since it is not uncommon to have five or more design variables in a general optimization problem, the seriousness of multidimensionality becomes painfully obvious.

Traditionally, optimization methods in multidimensional space are classified in terms of two broad categories called direct methods and indirect methods. *Direct methods* uses a comparison of functional evaluation; *indirect methods* employ the mathematical principles of maximization or minimization. Direct methods try to establish a way to 'zero in' on the optimum; indirect methods try to satisfy the conditions of the problem without examining non optimal points. In the following paragraphs we look only at *direct* methods now in use for multidimensional optimization.

2.2.1 Indirect Optimization

The treatment in [25] of multi–dimensional optimization would not be complete without a discussion of the calculus of stationary points. For a multi–dimensional function

to have a minimum, a maximum, or a saddle point, it is necessary that all first derivatives with respect to each of the n independent variables be zero. Thus for the function

$$M(x) = F(x_1, x_2, \ldots, x_n)$$

a stationary point will satisfy

$$\frac{\partial F}{\partial x_1} = 0, \quad \frac{\partial F}{\partial x_2} = 0, \quad \ldots, \quad \frac{\partial F}{\partial x_n} = 0$$

To determine whether a stationary point is a minimum, a maximum, or a saddle point, it is necessary to examine the second derivatives of the function. A better way to describe the nature of the second derivatives is by means of the Hessian matrix, which is of the form

$$\text{Hessian} = \begin{bmatrix} \frac{\partial^2 F}{\partial x_1^2} & \frac{\partial^2 F}{\partial x_1 \partial x_2} & \cdots & \cdots & \cdots & \frac{\partial^2 F}{\partial x_1 \partial x_n} \\ \frac{\partial^2 F}{\partial x_1 \partial x_2} & \frac{\partial^2 F}{\partial x_2^2} & \cdots & \cdots & \cdots & \frac{\partial^2 F}{\partial x_2 \partial x_n} \\ \vdots & \vdots & \vdots & \vdots & \vdots & \vdots \\ \frac{\partial^2 F}{\partial x_n \partial x_1} & \cdots & \cdots & \cdots & \cdots & \frac{\partial^2 F}{\partial x_n^2} \end{bmatrix}$$

A necessary and sufficient condition for a stationary point to be a local minimum is that its Hessian matrix be positive definite. This means that all its eigen–values will be positive. A necessary and sufficient condition for a stationary point to be a local maximum is that its Hessian matrix be negative definite. This means that all its eigen–values will be negative. One way to mechanize this information is shown in Figure 2–2. First the system of equations corresponding to the n first partial derivatives is found. This system must be solved for all–possible sets of design values that satisfy the equations. If these equations are linear, the problem is straightforward since only one solution set will exist. If the system is nonlinear, as is most often the case, there may be many solutions sets. Once the solution sets are isolated, the designer must discard all the solution sets that are not of the de-

sired extremum type. This requires a check of the eigen–values of the Hessian matrix of second partial derivatives evaluated at each of the solution design points. Once the solution sets have been reduced to a final group, the designer must check to see which of the group has the most desirable objective value. This one will be declared the optimum.

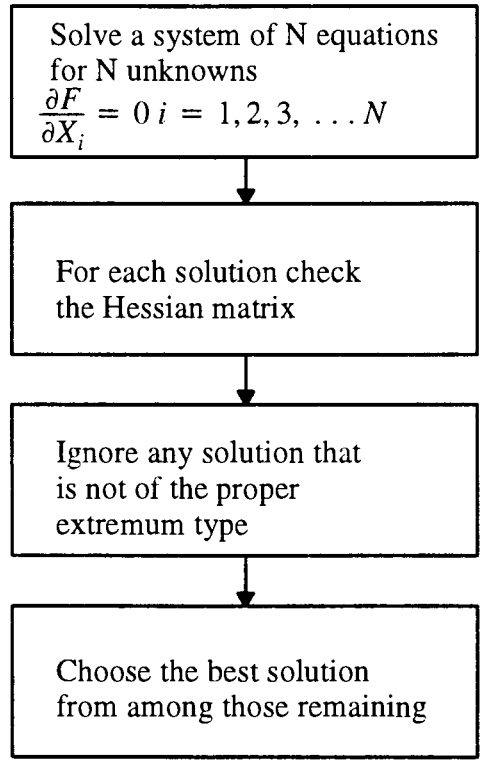

Figure 2–2 The indirect method of optimization

Although the previous technique does seem mathematically straightforward, it is, in fact, not extremely practical since the optimum in many design situations will occur at a boundary rather than at a stationary point. The technique does point out the need for methods to extract eigen–values and for methods to solve systems of nonlinear algebraic equations.

One interesting extension of the technique of stationary points is the method of La-

grange Multipliers. This technique has the advantage of allowing equality constraints of the form

$$Q_1(x_1, x_2, \ldots, x_n) = 0$$

$$\vdots$$

$$Q_j(x_1, x_2, \ldots, x_n) = 0$$

to be satisfied in the optimization process. To facilitate the solution of this problem, a new objective function must be formed that is linear combination of the old objective function and each of the constraint equations multiplied by a unique constant. This new objective function will be

$$M(x_i, \lambda_i) = F(x_i) + \lambda_1 Q_1 + \lambda_2 Q_2 + \ldots + \lambda_j Q_j$$

The λ_j values are called *Lagrange multipliers* and are said to be treated as additional unknowns to be determined in the solution process. Thus the system used to locate stationary points consists of $j + n$ equations and $j + n$ unknowns. If each of the constraints is satisfied, the additional λ_j terms each contribute nothing to the new objective function. In this case the optimization of M is equivalent to the optimization of F. It should be noted that in the equations to be solved for the stationary point, the partial derivatives of the new objective function with respect to the unknown Lagrange multipliers revert to the constraint equations.

2.2.2 Direct Optimizations

A large number of direct multi–dimensional optimization algorithms depend in some way on gradient information. The basis for this fact can be seen in a simple illustration. Suppose that a mountain climber was blindfolded and told to climb to the top of a single peak mountain. Even without the benefit of being able to see the peak, the climber

could reach the top simply by remembering always to walk uphill. Since any rising path will eventually lead to the top, the path where the slope is steepest is the best, provided that the climber does not encounter a vertical cliff that he or she cannot scale. (The mathematical equivalent of a cliff would be a ridge caused by a constraint in the surface in question.) For now it will be assumed that the optimization problem is unconstrained. The optimization equivalent of the steepest path idea is known as the *method of steepest ascent* or the *method of steepest descent.* The gradient vector is perpendicular to a contour and can be used to locate a new design point. To understand the logic behind the gradient methods, it is better to look into the nature of the gradient. Consider a system of independent unit vectors $\mathbf{e}_1, \mathbf{e}_2, \mathbf{e}_3, \ldots, \mathbf{e}_n$ that are parallel with the design variable axes $x_1, x_2, x_3, \ldots, x_n$. The gradient vector for a general objective function $F(x_1, x_2, x_3, \ldots, x_n)$ will be of the form

$$\textbf{gradient} = \nabla.\mathrm{F} = \frac{\partial \mathrm{F}}{\partial \mathrm{x}_1}\,\mathbf{e}_1 + \frac{\partial \mathrm{F}}{\partial \mathrm{x}_2}\,\mathbf{e}_2 + \ldots + \frac{\partial \mathrm{F}}{\partial \mathrm{x}_n}\,\mathbf{e}_n$$

where the partial derivatives are evaluated at the point being considered. This vector points in the upward or ascent direction and its negative points in the descent direction. The unit gradient vector is often written as

$$\frac{\nabla.\mathbf{F}}{|\nabla.\mathbf{F}|} = \mathbf{g}_1\mathbf{e}_1 + \mathbf{g}_2\mathbf{e}_2 + \ldots + \mathbf{g}_n\mathbf{e}_n$$

where

$$\mathrm{g_i} = \frac{\dfrac{\partial F}{\partial x_i}}{\displaystyle\sum_{j=1}^{n}\left[\left(\frac{\partial F}{\partial x_j}\right)^2\right]^{1/2}}$$

In some cases the nature of the objective function is better known to allow differentiation to calculate the gradient vector components. If the partial derivatives can not be found in this way, they may be approximated by central finite difference formula to get

$$\frac{\partial F}{\partial x_i} \simeq \frac{F(x_1, x_2, \ldots, x_i + \Delta x_i, \ldots, x_n) - F(x_1, x_2, \ldots, x_i - \Delta x_i, \ldots, x_n)}{2\Delta x_i}$$

where Δx_i is a small difference along the x_i direction. Once the gradient direction is known, it can be used in a variety of ways to implement a search strategy. In the chapter III and IV we will introduce and develop the different optimization methods that can be used to minimize a general function of n–variables starting from the steepest descent to Fletcher and Reeves, and up to the variable metric method of Davidon, Fletcher, and Powell.

CHAPTER III

STEEPEST ASCENT BY STEPS

Some search methods move a fixed step up the gradient and recalculate the function. If an improvement has been found, a new gradient is calculated and the procedure is repeated, often with an increased step size. If no improvement or a negative improvement is found, the step size from the previous best point is decreased and the procedure is repeated. The process continues until no improvement can be found by decreasing the step size. Some search methods use the data about the gradient to perform a one–dimensional search along the direction of the steepest ascent or descent using the equation

$$x^{i+1} = x^i + \lambda S_i$$

where λ is the new one–dimensional step–size parameter along the unit gradient s_i, and where x^{i+1} is the new point of search in the direction of the gradient and x^i is the old point.

The unit gradient vector for the direction of steepest descent is:

$$s_i = \frac{-\frac{\partial F}{\partial x_i}}{\sum_{j=1}^{n}\left[\left(\frac{\partial F}{\partial x_j}\right)^2\right]^{1/2}} \quad i = 1, 2, 3, \ldots, n$$

Once the one–dimensional optimum along the gradient has been achieved, a new gradient is found and the process is repeated until no further improvement can be found. The primary advantage of this method is that the parameter λ may be used as the independent variable for a Fibonacci search, Quadratic or Cubic Interpolation search, and thus the method tends to be efficient. One of the principal advantages of the steepest gradient methods is their ability to avoid saddle points on the objective surface. It should be noted that

the gradient techniques will find only local optimum when applied to multimodal (many peaks) surfaces. For this reason, if the nature of the surface is not well known, several starting points should be considered to see if every start leads to the same optimum. Another difficulty that can restrict the efficiency of the gradient methods occurs when the technique encounters a ridge. Since a ridge represents a discontinuity in the slope of a contour line, it tends to give false information on the proper direction to move. Thus the search technique may slow down and zigzag back and forth across the ridge, making progress toward the optimum quite slow. In some cases the severity of this performance on a ridge is so slow that the algorithm must be abandoned. In reality, a large number of objective surfaces associated with problems from engineering design have one or more ridges. These ridges often point toward the optimum.

Thus the complexity associated with ridges can sometimes be turned into an advantage. Whenever a ridge is encountered, the best direction to move is along the ridge rather than in the direction of the local gradient.

3.1 STEEPEST DESCENT METHOD

The use of the negative of the gradient vector as a direction for minimization was first used by Cauchy in 1847. In this method, we start from an initial guess point x^1 and iteratively move towards the optimum point according to the equation

$$x^{i+1} = x^i + \lambda_i^* S_i = x^i - \lambda_i^* \nabla f_i$$

where λ_i^* is the optimal step length along the search direction $\mathbf{S}_i = -\nabla f_i$ The flow chart of this method is shown in Fig 3–1. The method of steepest descent may appear to be the best unconstrained minimization technique since each one–dimensional search starts in the "best" direction. However, because of the fact that the steepest descent direction is a local property, the method is not really effective in most of the problems.

In two–dimensional problems, the application of the steepest descent method leads to a path made up of parallel and perpendicular segments as shown in Fig. 3–2. It can be seen that the path is a zig–zag in much the same way as the one–dimensional method

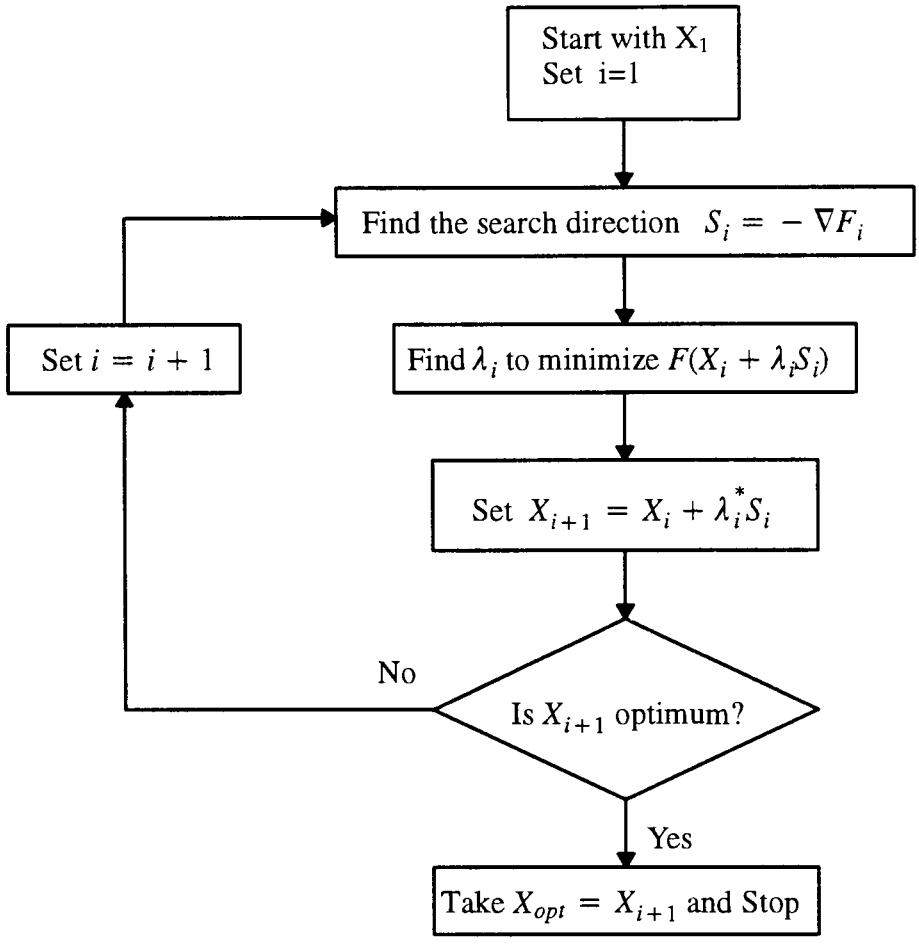

Figure 3–1 Flow chart for the steepest descent method

In higher dimensions, the path may not be made up of parallel and perpendicular segments and hence the method may have different characteristics than the one–dimensional method. For functions with great eccentricity, the methods converge into a steady n–dimensional zig–zag and the process will be unbearably slow. On the other hand, if the

contours of the objective function are not very much distorted, the method may converge faster as shown in Fig.3–2. The following criteria for terminating the iterative process can be used:

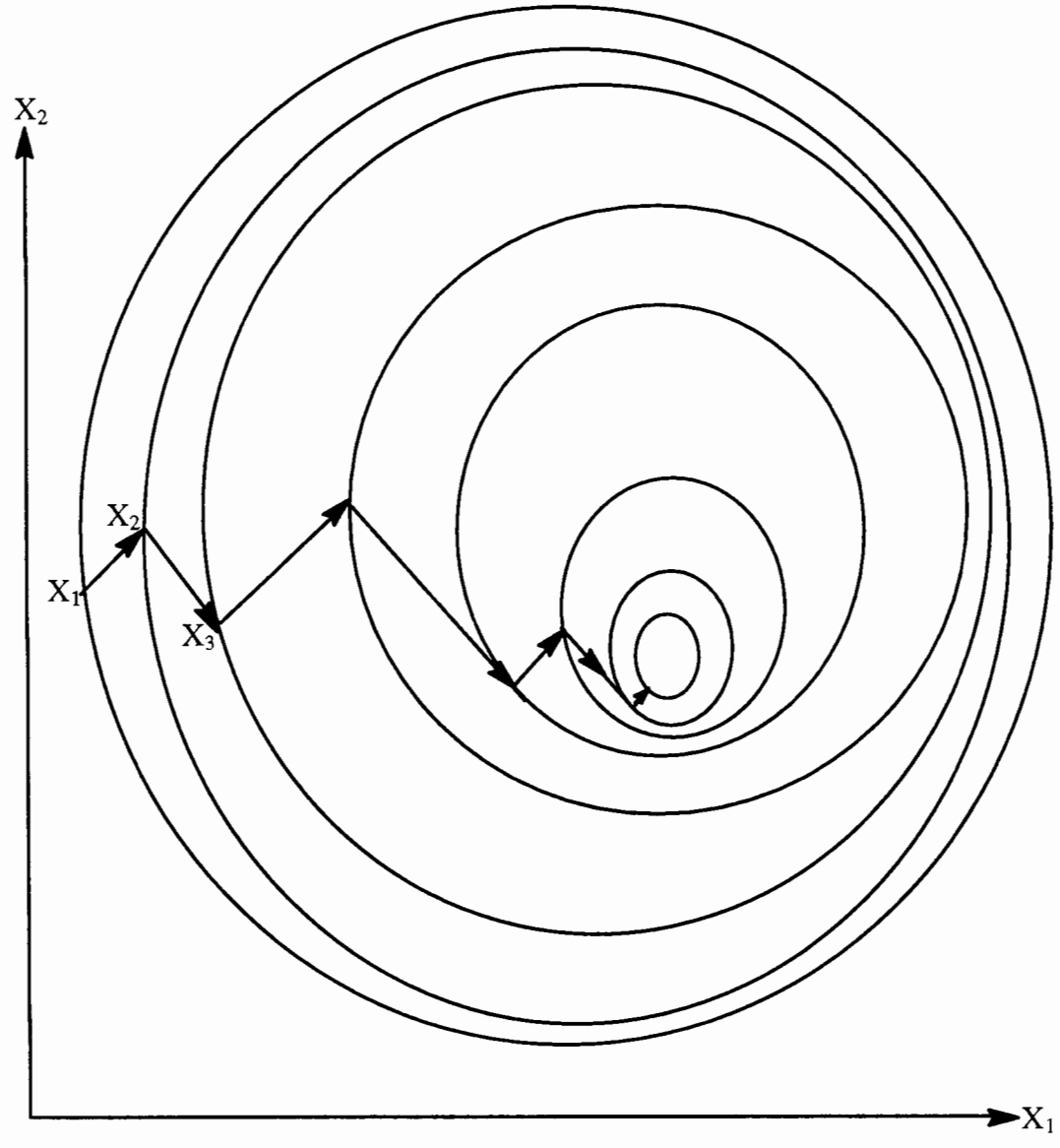

Figure 3–2 Convergence of steepest descent method showing the parallel and perpendicular segments

(i) $$\left|\frac{f(\mathrm{X}_{i+1}) - f(\mathrm{X}_i)}{f(\mathrm{X}_i)}\right| \leq \epsilon_1 \tag{3.1}$$

(ii) $$\left|\frac{\partial f}{\partial x_i}\right| \leq \epsilon_2, \; i = 1, 2, \ldots, n \tag{3.2}$$

(iii) $$|\mathrm{X}_{i+1} - \mathrm{X}_i| \leq \epsilon_3 \tag{3.3}$$

3.2 MODIFICATION OF THE METHOD

A lot of changes have been suggested over the years to accelerate the convergence of the steepest descent method. One of these changes is based on the concept of using the search direction

$$\mathrm{S}_i = \mathrm{X}_i - \mathrm{X}_{i-2}, \; i \geq 2 \tag{3.4}$$

from time to time instead of using the direction $-\nabla f(\mathrm{X}_i)$ always. This modification was suggested by Forsythe and Motzkin [26]. The benefit expected from this change can be seen from Fig. 2. Notice that the search directions defined by Eq. (3.4) lie in the general direction of the minimum and therefore one can expect to achieve a faster convergence by moving along these directions occasionally.

Another change was suggested by Shah [17], which can be considered as an extension of the previous idea. In this method, the search directions are taken alternately as the steepest descent direction and the direction given by an equation similar to Eq. (3.4). This method is called a gradient based PARTAN (parallel tangents) method. The algorithm of this method can be stated as follows.

(i) Start with an initial point $\mathbf{X}_1$.

(ii) Search for the minimum along the direction $\mathbf{S}_1 = -\nabla f(\mathbf{X}_1)$, and set the new point as: $\mathbf{X}_2 = \mathbf{X}_1 + \lambda_1^* \mathbf{S}_1$.

(iii) Search along the direction $\mathbf{S}_2 = -\nabla f(\mathbf{X}_2)$, and obtain the new point $\mathbf{X}_3$.

(iv) Find the next search direction as $\mathbf{S}_3 = (\mathbf{X}_3 - \mathbf{X}_1)$, and obtain the new point $\mathbf{X}_4$.

(v) Take the new search direction as

$$\mathbf{S}_i = \begin{cases} -\nabla f(\mathbf{X}_i) & \text{for } i = 4, 6, 8, \ldots, 2k \\ (\mathbf{X}_i - \mathbf{X}_{i-2}) & \text{for } i = 5, 7, 9, \ldots, 2k - 1, \end{cases}$$

and find the new point as $\mathbf{X}_{i+1} = \mathbf{X}_i + \lambda_i^* \mathbf{S}_i$ where λ_i^* is the optimal step length in the direction $\mathbf{S}_i$.

It was shown that this method is a type of conjugate direction method by Pierre [14]. But, Sorenson [27] has shown that this gradient based PARTAN method is less efficient compared to the conjugate gradient method while minimizing a quadratic function.

3.3 FLETCHER–REEVES METHOD

The convergence of the steepest descent method can be greatly improved by changing it into a conjugate gradient method. It has been shown that any minimization method that makes use of the conjugate directions is quadratically convergent. This characteristic of quadratic convergence is very useful because it ensures that the method will minimize a quadratic function in n steps or less. Since any general function can be approximated by a quadratic near the optimum point, any quadratically convergent method is expected to find the optimum point in a finite number of iterations.

3.3.1. Development of the Conjugate Gradient Method

The technique used in the development of the conjugate gradient method is similar to the Gram–Schmidt orthogonalization procedure. This technique sets up each new search direction as a linear combination of all the previous search directions, and the newly found gradient.

If the search directions in the minimization process, $\mathbf{S}_1$, $\mathbf{S}_2$, $\mathbf{S}_3$, ..., $\mathbf{S}_i$, are mutually conjugate with respect to matrix $\mathbf{A}$ of the quadratic function $f(\mathbf{X}) = \frac{1}{2}\mathbf{X}^T\mathbf{A}\mathbf{X} + \mathbf{B}^T\mathbf{X} + C$.

Then $\mathbf{S}_k^T \nabla f_{i+1} = 0 \quad \text{for} \, k = 1, 2, ..., i$

The gradient of the function f, calculated at the point $\mathbf{X}_{i+1}$, is given by

$$\nabla f_{i+1} = \mathbf{A}\,\mathbf{X}_{i+1} + \mathbf{B} \tag{3.5}$$

Since $\mathbf{X}_{i+1}$ is reached after i minimization steps, it can be written as

$$\mathbf{X}_{i+1} = \mathbf{X}_1 + \lambda_1^* \mathbf{S}_1 + \lambda_2^* \mathbf{S}_2 + \ldots + \lambda_k^* \mathbf{S}_k + \lambda_{k+1}^* \mathbf{S}_{k+1} + \ldots + \lambda_i^* \mathbf{S}_i$$

$$= \mathbf{X}_{i+1} + \sum_{j=k+1}^{i} \lambda_j^* \mathbf{S}_j \tag{3.6}$$

where λ_j^* is the minimizing step length in the direction $\mathbf{S}_j$. In view of Eq. (3.5), Eq. (3.6) becomes

$$\nabla f_{i+1} = \mathbf{A}\left[\mathbf{X}_{k+1} + \sum_{j=k+1}^{i} \lambda_j^* \mathbf{S}_j\right] + \mathbf{B}$$

$$= \nabla f_{k+1} + \sum_{j=k+1}^{i} \lambda_j^* \mathbf{A}\,\mathbf{S}_j \tag{3.7}$$

Multiplying both sides of Eq. (3.7) by $\mathbf{S}_k^T$, we get

$$\mathbf{S}_k^T \nabla f_{i+1} = \mathbf{S}_k^T \nabla f_{k+1} + \sum_{j=k+1}^{i} \lambda_j^* \mathbf{S}_k^T \mathbf{A}\,\mathbf{S}_j \tag{3.8}$$

The first term on the right–hand side of Eq.(3.8) is zero since λ_k^* is the minimizing step length along the direction $\mathbf{S}_k$, and the second term is zero since the search directions $\mathbf{S}_1, \mathbf{S}_2, \mathbf{S}_3, ..., \mathbf{S}_i$ are given to be $\mathbf{A}$–conjugate. Therefore, we get the desired result

$$\mathbf{S}_k^T \nabla f_{i+1} = 0, \quad \text{for} \quad k = 1, 2, ..., i \tag{3.9}$$

3.3.2. Developing The New Algorithm

We are going to change the steepest descent method applied to a quadratic function

$f(\mathbf{X}) = \frac{1}{2}\mathbf{X}^T\mathbf{A}\mathbf{X} + \mathbf{B}^T\mathbf{X} + \mathbf{C}$ in the development of a new algorithm by forcing the condition that the successive directions be mutually conjugate. Let $\mathbf{X}_1$ be the starting point for the minimization and let the first search direction be the steepest descent direction. Then

$$\mathbf{S}_1 = -\nabla f_1 = -\mathbf{A}\mathbf{X}_1 - \mathbf{B} \tag{3.10}$$

and $$\mathbf{X}_2 = \mathbf{X}_1 + \lambda_1^* \mathbf{S}_1 \tag{3.11}$$

where λ_1^* is the minimizing step length in the direction $\mathbf{S}_1$ so that

$$\mathbf{S}_1^T \nabla f|_{\mathbf{X}2} = 0 \tag{3.12}$$

Equation (3.12) can be expanded as

$$\mathbf{S}_1^T\left\{\mathbf{A}\left(\mathbf{X}_1 + \lambda_1^* \mathbf{S}_1\right) + \mathbf{B}\right\} = 0$$

or

$$\mathbf{S}_1^T\mathbf{A}\mathbf{X}_1 + \lambda_1^* \mathbf{S}_1^T\mathbf{A}\mathbf{g}_1 + \mathbf{S}_1^T\mathbf{B} = 0$$

from which the value of λ_1^* can be obtained as

$$\lambda_1^* = \frac{-\mathbf{S}_1^T(\mathbf{A}\mathbf{X}_1 + \mathbf{B})}{\mathbf{S}_1^T\mathbf{A}\mathbf{g}_1} = \frac{\mathbf{S}_1^T \nabla f_1}{\mathbf{S}_1^T\mathbf{A}\mathbf{S}_1} \tag{3.13}$$

Now express the second search direction as a linear combination of $\mathbf{S}_1$ and $-\nabla f_2$ as

$$\mathbf{S}_2 = -\nabla f_2 + \beta_2 \mathbf{S}_1 \tag{3.14}$$

where β_2 is to be chosen so as to make $\mathbf{S}_1$ and $\mathbf{S}_2$ conjugate. This requires

$$\mathbf{S}_1^T\mathbf{A}\mathbf{S}_2 = 0 \tag{3.15}$$

Substituting for $\mathbf{S}_1$ from Eq. (3.14), Eq. (3.15) becomes

$$\mathbf{S}_1^T \mathbf{A}(-\nabla f_2 + \beta_2 \mathbf{S}_1) = 0 \tag{3.16}$$

Since Eq. (3.11) gives

$$\mathbf{S}_1 = \frac{(\mathbf{X}_2 - \mathbf{X}_1)}{\lambda_1^*} \tag{3.17}$$

Eq. (3.16) can be written as

$$\mathbf{S}_1^T \mathbf{A}\, \mathbf{S}_2 = -\frac{(\mathbf{X}_2 - \mathbf{X}_1)^T}{\lambda_1^*} \mathbf{A}(\nabla f_2 - \beta_2 \mathbf{S}_1) = 0 \tag{3.18}$$

The difference of the gradients $(\nabla f_2 - \nabla f_1)$ is given by

$$(\nabla f_2 - \nabla f_1) = (\mathbf{A}\mathbf{X}_2 + \mathbf{B}) - (\mathbf{A}\mathbf{X}_1 + \mathbf{B}) = \mathbf{A}(\mathbf{X}_2 - \mathbf{X}_1) \tag{3.19}$$

From Eq. (3.19), Eq. (3.18) can be written as

$$(\nabla f_2 - \nabla f_1)^T (\nabla f_2 - \beta_2 \mathbf{S}_1) = 0 \tag{3.20}$$

$$\nabla f_2^{\mathrm{T}} \nabla f_2 - \nabla f_1^{\mathrm{T}} \nabla \mathrm{f}_2 - \beta_2 \nabla f_2^T \mathbf{S}_1 + \beta_2 \nabla f_1^T \mathbf{S}_1 = 0 \tag{3.21}$$

Since $\nabla f_1^T \nabla f_2 = -\mathbf{S}_1^T \nabla f_2 = 0$ from Eq.(3.5), this equation gives the value of β_2 as

$$\beta_2 = \frac{\nabla f_2^T \nabla f_2}{\nabla f_1^T \nabla f_1} \tag{3.22}$$

Next we let the third search direction as a linear combination of $\mathbf{S}_1, \mathbf{S}_2$ and $-\nabla f_3$ *as*

$$\mathbf{S}_3 = -\nabla f_3 + \beta_3 \mathbf{S}_2 + \delta_3 \mathbf{S}_1 \tag{3.23}$$

where the values of β_3 and δ_3 can be found by making $\mathbf{S}_3$ conjugate to $\mathbf{S}_1$ and $\mathbf{S}_2$.

First we consider

$$\mathbf{S}_1^T \mathbf{A}\, \mathbf{S}_3 = -\mathbf{S}_1^T \mathbf{A} \nabla f_3 + \beta_3 \mathbf{S}_1^T \mathbf{A} \mathbf{S}_2 + \delta_3 \mathbf{S}_1^T \mathbf{A} \mathbf{S}_1 = 0 \tag{3.24}$$

If we assume that $\mathbf{S}_1$ and $\mathbf{S}_2$ are already made conjugate, $\mathbf{S}_1^T \mathbf{A}\, \mathbf{S}_2 = 0$, and Eq .(3.24) gives

$$\boldsymbol{\delta}_3 = \frac{\mathbf{S}_1^T \mathbf{A} \nabla f_3}{\mathbf{S}_1^T \mathbf{A} \mathbf{S}_1} \tag{3.25}$$

From Eq. (3.17), δ_3 can be expressed as

$$\boldsymbol{\delta}_3 = \frac{(\mathbf{X}_2 - \mathbf{X}_1)}{\lambda_1^*} \frac{\mathbf{A}\nabla f_3}{\mathbf{S}_1^T \mathbf{A} \mathbf{S}_1} \tag{3.26}$$

By using Eq. (3.19), Eq. (3.26) can be rewritten as

$$\boldsymbol{\delta}_3 = \frac{1}{\lambda_1^*} \frac{(\nabla f_2 - \nabla f_1)^T \nabla f_3}{\mathbf{S}_1^T \mathbf{A} \mathbf{S}_1} \tag{3.27}$$

Since $\mathbf{S}_1 = -\nabla f_1$ from Eq.(3.10), and $\mathbf{S}_2 - \beta_2 \mathbf{S}_1 = -\nabla f_2$ from Eq.(3.14), we obtain

$$\nabla f_2 - \nabla f_1 = -\mathbf{S}_2 + \mathbf{S}_1(1 + \beta_2) \tag{3.28}$$

and Eq.(3.27) gives $$\boldsymbol{\delta}_3 = \frac{1}{\lambda_1^*} \frac{\{-\mathbf{S}_2 + \mathbf{S}_1(1 + \beta_2)\}^T \nabla f_3}{\mathbf{S}_1^T \mathbf{A} \mathbf{S}_1} \tag{3.29}$$

which we can deduce that Eq. (3.29) is equal to zero because of Eq. (3.9). Therefore

Eq. (3.23) becomes $$\mathbf{S}_3 = -\nabla f_3 + \beta_3 \mathbf{S}_2 \tag{3.30}$$

The value of β_3 can be found by making $\mathbf{S}_3$ conjugate to $\mathbf{S}_2$. However, instead of finding the value of a specific β, we can derive a general formula for $\beta_i, i = 2, 3, \ldots$

By generalizing Eq. (3.30), we can express the search direction in the ith step, $\mathbf{S}_i$, as a linear combination of $-\nabla f_i$ and $\mathbf{S}_{i-1}$, that is,

$$\mathbf{S}_i = -\nabla f_i + \beta_i \mathbf{S}_{i-1} \tag{3.31}$$

where the value of β_i can be found by making $\mathbf{S}_i$ conjugate to $\mathbf{S}_{i-1}$ as

$$\beta_i = \frac{\nabla f_i^T \nabla f_i}{\nabla f_{i-1}^T \nabla f_{i-1}} \tag{3.32}$$

The search directions that have been considered so far, Eq. (3.31), are exactly the directions used in the Fletcher–Reeves method.

3.3.3 The Fletcher–Reeves Algorithm

Using of Eqs. (3.31) and (3.32) for the minimization of general functions was first suggested by Fletcher and Reeves [10]. Let us state their algorithm briefly as follows:

(i) Begin with an arbitrary initial point $\mathbf{X}_1$

(ii) Let the first search direction $\mathbf{S}_1 = -\nabla f(\mathbf{X}_1) = -\nabla f_1$

(iii) Determine the point $\mathbf{X}_2$ according to the equation $\mathbf{X}_2 = \mathbf{X}_1 + \lambda_1^* \mathbf{S}_1$ where λ_1^* is the minimum step length in the direction of $\mathbf{S}_1$. Let i=2 and go to the next step.

(iv) Determine $\nabla f_i = \nabla f(\mathbf{X}_i)$, and let

$$\mathbf{S}_i = -\nabla f_i + (|\nabla f_i|^2/|\nabla f_{i-1}|^2)\,\mathbf{S}_{i-1} \tag{3.33}$$

(v) Calculate the minimum step length λ_i^* in the direction of $\mathbf{S}_i$, and determine the new point $\mathbf{X}_{i+1} = \mathbf{X}_i + \lambda_i^* \mathbf{S}_i$

(vi) Test if the point $\mathbf{X}_{i+1}$ is the minimum point. If the point is minimum, stop the process. Else, let the value of i = i + 1, and repeat steps (iv), (v) and (vi) until the convergence is achieved.

The Fletcher and Reeves method was originally proposed by Hestenes and Stiefel [15] as a method for solving systems of linear equations derived from the stationary conditions of a quadratic. Since the directions $\mathbf{S}_i$ used in this method are A–conjugate, the program should converge in n–times or less for a quadratic function. But, for ill–conditioned quadratics (i.e. which possess a highly eccentric and distorted contours), the program may take more than n–times to converge. The reason for this has been found to be the total effect of rounding errors. Since $\mathbf{S}_i$ is given by Eq.(3.33), any error as a consequence of the inaccuracies involved in finding λ_i^*, is rippled through the vector $\mathbf{S}_i$. Thus the search directions $\mathbf{S}_i$ will be progressively effected by these errors. Hence it is better to restart the method periodically after every $m=n+1$ steps, where n is the number of design variables, by taking

the new search direction as the steepest descent direction. Inspite of this, the Fletcher and Reeves algorithm is greatly superior to the steepest descent method, but it is rather less efficient than the quasi–Newton and the variable metric methods, which we will consider in the next chapter.

CHAPTER IV

QUASI–NEWTON METHODS

All the local minima $\mathbf{X}^*$ of a continuously differentiable function f satisfy the necessary conditions

$$\mathbf{g}(\mathbf{X}^*) \equiv \nabla f(\mathbf{X}^*) = 0 \tag{4.1}$$

Eq.(4.1) represents a set of n nonlinear equations which must be solved to get $\mathbf{X}^*$. One approach to the optimization of $f(\mathbf{X})$ is, to seek the solutions of the set of Eqs .(4.1) by including a provision to ensure that the solution found does indeed correspond to a local minimum. The oldest method for solving a set of nonlinear equations is the Newton's method. We shall consider this method briefly and then turn to a class of methods which can be called "Quasi–Newton" methods since they can be regarded as approximations to the Newton's method in some sense.

4.1 NEWTON'S METHOD

To solve the system of nonlinear Eqs. (4.1) by the Newton's method, we first linearize the set of equations about some point $\mathbf{X}_i$ (which is the ith approximation to the minimum point $\mathbf{X}^*$). Thus if $\mathbf{X}^*$ can be written as $\mathbf{X}^* = \mathbf{X}_i + \mathbf{S}$, the Taylor's series expansion of $\mathbf{g}(\mathbf{X}^*)$ gives

$$\mathbf{g}(\mathbf{X}^*) = g(\mathbf{X}_i + \mathbf{S}) = \mathbf{g}(\mathbf{X}_i) + \mathbf{J}_{\mathbf{X}_i}\mathbf{S} + \cdots \tag{4.2}$$

By neglecting the higher order terms in Eq. (4.2) and setting $\mathbf{g}(\mathbf{X}^*) = \mathbf{0}$, we obtain

$$\mathbf{g}_i + \mathbf{J}_i\mathbf{S} = \mathbf{0} \tag{4.3}$$

where $\mathbf{g}_i = \mathbf{g}(\mathbf{X}_i)$ and $\mathbf{J}_i = \mathbf{J}|_{\mathbf{X}_i}$ is the matrix of second partial derivatives of f evaluated

at the point $\mathbf{X}_i$.

If $\mathbf{J}_i$ is non–singular, the set of linear equations, Eqs. (4.3), can be easily solved for the vector $\mathbf{g}$, and the desired minimum can be obtained as $\mathbf{X}^* = \mathbf{X}_i + \mathbf{S}$. Thus, Eqs. (4.3) give

$$\mathbf{g} = -\mathbf{J}_i^{-1}\mathbf{S} \tag{4.4}$$

However, in general, the higher order terms in Eq. (4.2) are not negligible and hence an iterative procedure has to be used to find the improved approximations. The iterative scheme is given by

$$\mathbf{X}_{i+1} = \mathbf{X}_i + \mathbf{S}_i = \mathbf{X}_i - \mathbf{J}_i^{-1}\mathbf{g}_i \tag{4.5}$$

The sequence of points $\mathbf{X}_1, \mathbf{X}_2, \cdots, \mathbf{X}_{i+1}$ can be shown to converge to the actual solution $\mathbf{X}^*$ from any initial point $\mathbf{X}_1$ sufficiently close to the solution $\mathbf{X}^*$, provided that $\mathbf{J}_1$ is non–singular. These conditions are, however, very restrictive and the method frequently fails to converge. If $f(\mathbf{X})$ is a quadratic, we can find its minimum in a single step by using Eq. (4.5) since the Taylor's series expansion is exact. This can also be proved as follows:

If $$f(\mathbf{X}) = \frac{1}{2}\mathbf{X}^T\mathbf{A}\,\mathbf{X} + \mathbf{B}^T\mathbf{X} + C$$

the minimum of $f(\mathbf{X})$ is given by $\mathbf{X}^* = -\mathbf{A}^{-1}\mathbf{B}$. The iterative step of Eq. (4.5) gives

$$\mathbf{X}_{i+1} = \mathbf{X}_i - \mathbf{A}^{-1}(\mathbf{A}\,\mathbf{X}_i + \mathbf{B}) \tag{4.6}$$

where $\mathbf{X}_i$ is the starting point for the ith iteration. Thus Eq. (4.6) gives the exact solution $\mathbf{X}_{i+1} = \mathbf{X}^* = -\mathbf{A}^{-1}\mathbf{B}$.

Lets try to minimize the function $f(x_1,x_2) = x_1 - x_2 + 2x_1x_2 + 2x_1^2 + x_2^2$, using Newton's Method, by starting with the point $\mathbf{X}_1 = (\mathbf{0},\mathbf{0})$. To find $\mathbf{X}_2$ according to Eq. (4.5), we need $\mathbf{J}_1^{-1}$ where

$$\mathbf{J}_1 = \begin{bmatrix} \dfrac{\partial^2 f}{\partial x_1^2} & \dfrac{\partial^2 f}{\partial x_1 \partial x_2} \\ \dfrac{\partial^2 f}{\partial x_2 \partial x_1} & \dfrac{\partial^2 f}{\partial x_2^2} \end{bmatrix}_{\mathbf{X}_1} = \begin{bmatrix} 4 & 2 \\ 2 & 2 \end{bmatrix}$$

$$\therefore \mathbf{J}_1^{-1} = \frac{1}{4}\begin{bmatrix} +2 & -2 \\ -2 & +4 \end{bmatrix} = \begin{bmatrix} +0.5 & -0.5 \\ -0.5 & +1.0 \end{bmatrix}. \text{ As}$$

$$\mathbf{g}_1 = \begin{bmatrix} \partial f/\partial x_1 \\ \partial f/\partial x_2 \end{bmatrix}_{\mathbf{x}_1} = \begin{bmatrix} 1 + 4x_1 + 2x_2 \\ -1 + 2x_1 + 2x_2 \end{bmatrix}_{(0,0)} = \begin{bmatrix} 1 \\ -1 \end{bmatrix}, \text{ Equation (4.5) gives}$$

$$\mathbf{X_2} = \mathbf{X_1} - \mathbf{J}^{-1}\mathbf{g_1} = \begin{bmatrix} 0 \\ 0 \end{bmatrix} - \begin{bmatrix} 0.5 & -0.5 \\ -0.5 & 1 \end{bmatrix}\begin{bmatrix} 1 \\ -1 \end{bmatrix} = \begin{bmatrix} -1 \\ 1.5 \end{bmatrix}, \text{ To see whether } \mathbf{X}_2 \text{ is the}$$

optimum point or not, we evaluate

$$\mathbf{g}_2 = \begin{bmatrix} \partial f/\partial x_1 \\ \partial f/\partial x_2 \end{bmatrix}_{\mathbf{x}_2} = \begin{bmatrix} 1 + 4x_1 + 2x_2 \\ -1 + 2x_1 + 2x_2 \end{bmatrix}_{(-1,1.5)} = \begin{bmatrix} 0 \\ 0 \end{bmatrix}.$$

As $\mathbf{g_2} = \mathbf{0}$, $\mathbf{X_2}$ is the optimum point. Thus the method has converged in one iteration for this quadratic function.

If $f(\mathbf{X})$ is a non–quadratic function, the Newton's method may sometimes diverge, and it may converge to saddle points and relative maxima. The method can be improved considerably by modifying Eq. (4.5) as

$$\mathbf{X}_{i+1} = \mathbf{X}_i + \lambda_i^* \mathbf{S}_i = \mathbf{X}_i - \lambda_i^* \mathbf{J}_i^{-1}\mathbf{g}_i \tag{4.7}$$

where λ_i^* is the minimizing step length in the direction $\mathbf{S}_i = -\mathbf{J}_i^{-1}\mathbf{g}_i$.

Let us apply the same method to minimize this non–quadratic function of two variables: $f(x_1, x_2) = -1/(x_1^2 + x_1^2 + 2)$ from the starting point $\mathbf{X}_1 = (4, 0)$. The gradient $\mathbf{g}$ and the Hessian matrix $\mathbf{J}$ of f are given by

$$\mathbf{g} = \begin{bmatrix} \partial f/\partial x_1 \\ \partial f/\partial x_2 \end{bmatrix} = \frac{2}{\left(x_1^2 + x_2^2 + 2\right)^2}\begin{bmatrix} x_1 \\ x_2 \end{bmatrix} \quad \text{and}$$

$$\mathbf{J} = \frac{2}{\left(x_1^2 + x_2^2 + 2\right)^3}\begin{bmatrix} \left(-3x_1^2 + x_2^2 + 2\right) & \left(-4x_1x_2\right) \\ \left(-4x_1x_2\right) & \left(-3x_2^2 + x_1^2 + 2\right) \end{bmatrix} \text{ At } \quad \mathbf{X}_1 = \begin{bmatrix} 4 \\ 0 \end{bmatrix},$$

$$\mathbf{g} = \begin{bmatrix} 0.0247 \\ 0 \end{bmatrix}, \quad \mathbf{J} = \begin{bmatrix} -0.01580 & 0 \\ 0 & 0.00617 \end{bmatrix}, \text{ and}$$

$\mathbf{J}^{-1} = \frac{1}{(-0.0000975)}\begin{bmatrix} 0.00617 & 0 \\ 0 & -0.01580 \end{bmatrix} = \begin{bmatrix} -63.4 & 0 \\ 0 & 162.0 \end{bmatrix}$ Hence Eq. (4.5) gives

$\mathbf{X}_2 = \mathbf{X}_1 - \mathbf{J}_1^{-1}\mathbf{g}_1 = \begin{bmatrix} 5.57 \\ 0 \end{bmatrix}$. If we compare the values of f at $\mathbf{X}_1$ and $\mathbf{X}_2$, we find that $f_1 = -0.0556$ and $f_2 = -0.0303$. Thus f_2 is greater than f_1 and therefore the method is diverging (true minimum point is $\mathbf{X}^* = \begin{bmatrix} 0 \\ 0 \end{bmatrix}$ with $f^* = -0.5$). But, if we use Eq. (4.7) instead of Eq. (4.5), the method can be made to converge to the minimum point.There is a number of advantages to this modification. First, it will converge to the minimum point in steps less than the original method. Second, it converges to the minimum point in all the cases whereas the original method may not converge in some cases. Third, it doesn't converge to a saddle point or a maximum. This method seems to be the most powerful minimization method. But, in spite of these advantages, this method is not generally used in practice because of these problems.

(a) we need to store the ($n \times n$) matrix J_i,

(b) it gets tedious and sometimes, impossible to compute the elements of the matrix J_i,

(c) it requires the inversion of the matrix J_i at each step,

(d) it requires the evaluation of the quantity $\mathbf{J}_i^{-1}\mathbf{g}_i$ at each step.

For a large number of variables of complicated functions, the above problems make the Newton's method impractical. Since the Newton's method uses the second derivatives of the function, the method is sometimes called a second order method. The methods that use the first order derivatives of the function are called first order method, like the steepest descent and Fletcher–Reeves method. Just like the differences in the function value contain information about the first derivatives, the differences in the gradient value contain information about the second derivatives. We can reduce the work by using the idea in-

volved in computing $\mathbf{J}_i$ in the Newton's method. If $\mathbf{X}_i$ and $\mathbf{X}_{i+1}$ are the approximations to the minimum obtained in two consecutive iterations, we can obtain, using Eq. (4.2),

$$\mathbf{g}(\mathbf{X}^*) = g(\mathbf{X}_i + \mathbf{S}) = \mathbf{g}(\mathbf{X}_i) + \mathbf{J}_{\mathbf{X}_i}\mathbf{S} \tag{4.8}$$

where $$\mathbf{X}^* = \mathbf{X}_i + \mathbf{S} \tag{4.9}$$

By denoting the new point found from the Newton's iteration as $\mathbf{X}_{i+1}$ instead of $\mathbf{X}^*$, we can change Eqs (4.8) and (4.9) to :

$$\mathbf{g}_{i+1} - \mathbf{g}_i = \mathbf{J}_i\mathbf{S} = \mathbf{J}_i\left(\mathbf{X}_{i+1} - \mathbf{X}_i\right) \tag{4.10}$$

If we define

$$\mathbf{G}_i = \mathbf{g}_{i+1} - \mathbf{g}_i \quad , \quad \mathbf{S}_i = \mathbf{X}_{i+1} - \mathbf{X}_i \tag{4.11}$$

and Eq. (4.10) becomes

$$\mathbf{G}_i = \mathbf{J}_i\mathbf{S}_i \quad \text{or} \quad \mathbf{S}_i = \mathbf{J}_i^{-1}\mathbf{G}_i \tag{4.12}$$

provided that the matrix $\mathbf{J}_i$ is non–singular. Equation (4.12) let us use the gradient differences to build up an approximation either to the matrix $\mathbf{J}_i$ or to its inverse $\mathbf{J}_i^{-1}$.

To form the iterative procedure of quasi–Newton methods, let

$$\mathbf{S}_i = \mathbf{H}_i\mathbf{G}_i, \quad i = 1, 2, \ldots, k \tag{4.13}$$

where $\mathbf{H}_i$ is the approximation to $\mathbf{J}_i^{-1}$ in the ith step (we can choose an appropriate $\mathbf{H}_1$ to start the iterative procedure). If we assume that Eq. (4.13) is satisfied in the $(k+1)$th step also, we have

$$\mathbf{S}_{k+1} = \mathbf{H}_{k+1}\mathbf{G}_{k+1} = \mathbf{H}_{k+1}\left(\mathbf{g}_{k+2} - \mathbf{g}_{k+1}\right) \tag{4.14}$$

such that if the point $\mathbf{X}_{k+2}$ found at the end of $(k+1)$th step is to be a stationary point, we need to have $\mathbf{g}_{k+2} = 0$, and Eq. (4.14) reduces to

$$\mathbf{S}_{k+1} = -\mathbf{H}_{k+1}\mathbf{g}_{k+1} \tag{4.15}$$

clearly the above assumption is not true, in general, and the point $\mathbf{X}_{k+2}$ may be a bad approximation to a stationary point. Therefore, we use Eq. (4.15) as a direction of

search and find the new point $\mathbf{X}_{k+2}$ as

$$\mathbf{X}_{k+2} = \mathbf{X}_{k+1} + \mathbf{S}_{k+1} \tag{4.16}$$

$$\text{where} \quad \mathbf{S}_{k+1} = -\lambda^*_{k+1}\, \mathbf{H}_{k+1}\, \mathbf{g}_{k+1} \tag{4.17}$$

where λ^*_{k+1} is the minimizing step length along the direction $-\mathbf{H}_{k+1}\,\mathbf{g}_{k+1}$. All the quasi–Newton methods are based on Eqs. (4.16) and (4.17) . They differ from one another only in the methods they use in constructing $\mathbf{H}_k$ to satisfy Eq. (4.13), and in choosing λ^*_k in Eq. (4.17). The method of constructing $\mathbf{H}_k$ completely eliminates the need for evaluating second derivatives and performing matrix inversions and yet the sequence of iterations converges to the minimum point $\mathbf{X}^*$. In addition to that, we can show that the matrix, $\mathbf{H}_k$, which is improved at each iteration, converges to $\mathbf{J}_i^{-1}$. In the following section, we will see one specific quasi–Newton method developed by Davidon, Fletcher, and Powell, also called the Variable Metric Method, is an optimization algorithm for finding the unconstrained minimum of a multivariable objective function of the form

$$\text{Objective} = F(x_1, x_2, \ldots, x_n)$$

derivatives of the objective function with respect to the independent variables are necessary. Since the algorithm is based on the assumption of unimodality (having one peak or a valley), several alternative starting points are recommended if the objective surface is suspected to be multimodal.

4.2 DAVIDON–FLETCHER–POWELL METHOD

Very important developments have happened in the area of descent techniques with the introduction of the variable metric method by Davidon [16]. This method was extended by Fletcher and Powell in 1963 [9]. This method is the best general purpose uncon-

strained optimization technique making use of the derivatives that is currently available. The iterative procedure of this method can be stated as follows:

(i) Start with an initial point $\mathbf{X}_1$ and a *n by n* positive definite symmetric matrix $\mathbf{H}_1$. Usually $\mathbf{H}_1$ is set at the start of the iterative process to equal the identity matrix $\mathbf{I}$. Set iteration number as i=1.

(ii) Compute the gradient of the function, ∇f_i, at the point $\mathbf{X}_i$, and set the search direction to :

$$\mathbf{S}_i = -\mathbf{H}_i \nabla f_i \tag{4.18}$$

(iii)Determine the optimal step length λ_i^* in the direction $\mathbf{S}_i$ and set the next point to:

$$\mathbf{X}_{i+1} = \mathbf{X}_i + \lambda_i^* \mathbf{S}_i \tag{4.19}$$

(iv)If the new point $\mathbf{X}_{i+1}$ optimal, stop the iterative procedure. Otherwise, go to step (v).

(v) Update the Hessian matrix $\mathbf{H}$ as

$$\mathbf{H}_{i+1} = \mathbf{H}_i + \mathbf{A}_i + \mathbf{B}_i \tag{4.20}$$

where

$$\mathbf{A}_i = \lambda_i^* \frac{\mathbf{S}_i \mathbf{S}_i^T}{\mathbf{S}_i^T \mathbf{Q}_i} \tag{4.21}$$

$$\mathbf{B}_i = -\frac{(\mathbf{H}_i \mathbf{Q}_i)(\mathbf{H}_i \mathbf{Q}_i)^T}{\mathbf{Q}_i^T \mathbf{H}_i \mathbf{Q}_i} \tag{4.22}$$

and

$$\mathbf{Q}_i = \nabla f(\mathbf{X}_{i+1}) - \nabla f(\mathbf{X}_i) = \nabla f_{i+1} - \nabla f_i \tag{4.23}$$

(vi)Set the new iteration index $k = i + 1$, and go to step (ii).

The above method of Davidon–Fletcher–Powell was originally considered to be a variable metric method by Davidon. It can be considered as a quasi–Newton method and

also as a conjugate gradient method. This method is very powerful and converges quadratically because it is a conjugate gradient method. It is very stable and continues to move towards the minimum even while minimizing very distorted and eccentric functions. This stability of the method can be owed to the fact that the information obtained in previous iterations is carried through the matrix $\mathbf{H}_i$.

This method can be put to the test against a special function of Rosenbrock called the Banana Function. Although it is simple mathematically, the Rosenbrock objective function

$$y = 100(x_2 - x_1^2)^2 + (1 - x_1)^2 \tag{4.24}$$

contains a curved valley as shown in Figure 4–1. its contour plot is also shown in Figure 4–2. The minimum location lies at the point (1.0, 1.0); however, if a starting value in the second quadrant is selected, convergence can sometimes be difficult to achieve. The algorithm uses a special scheme proposed by Davidon to perform the one–dimensional search for an optimum along the gradient line. This special search scheme is done in two phases.

4.2.1 Cubic Interpolation Technique

In the first phase extrapolation moves are made to bracket the location of the one–dimensional optimum. Then a cubic interpolation method based on a third–order polynomial approximation to the function is used to locate the best possible value within the brackets.

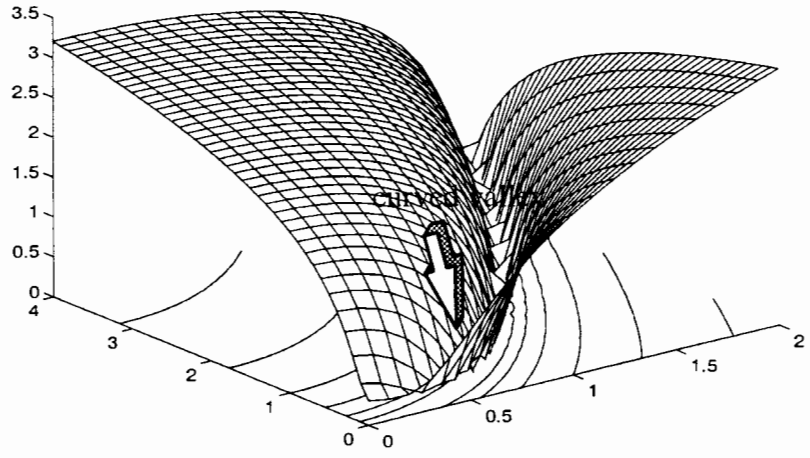

Figure 4–1 Graph of Rosenbrock (banana function)

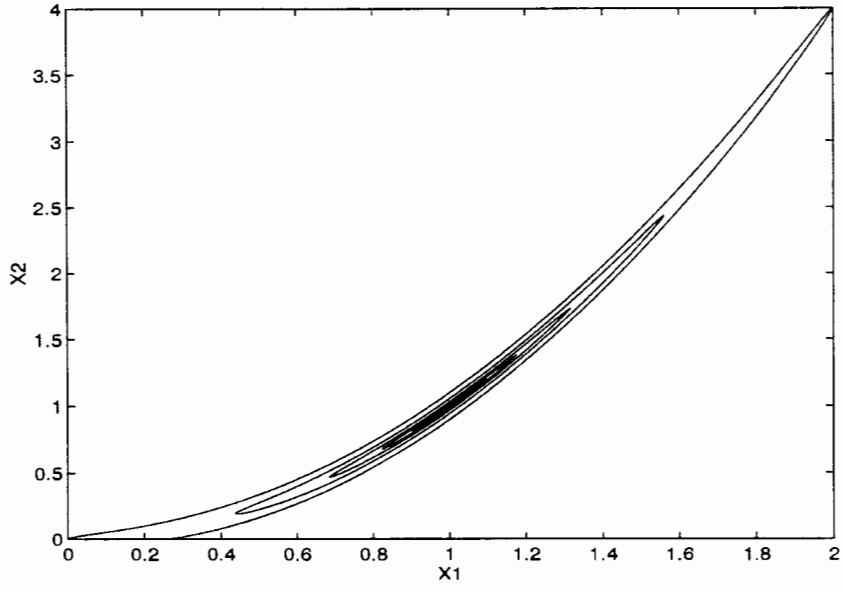

Figure 4–2 Contour Plot of Rosenbrock banana function

This technique [10] solves the problem of finding the minimizing step length λ^* in four stages. It makes use of the derivative

$$f'(\lambda) = \frac{df}{d\lambda} = \frac{d}{d\lambda} f(\mathbf{X} + \lambda \mathbf{g}) = \mathbf{g}^{\mathrm{T}} \nabla f(\mathbf{X} + \lambda \mathbf{g}) \tag{4.25}$$

The first stage normalizes the **g** vector so that a step size $\lambda = 1$ is acceptable. The second stage establishes bounds on λ^*, and the third stage finds the value of $\lambda^{\sim *}$ by approximating $f(\lambda)$ by a cubic polynomial $h(\lambda)$. If the $\lambda^{\sim *}$ found in stage three does not satisfy the prescribed convergence criteria, the cubic polynomial is refitted in the fourth stage.

Stage 1: Calculate $\Sigma = \max_i |g_i|$

where $|g_i|$ is the absolute value of the ith component of **g**, and divide each component of **g** by Σ. Another type of normalization is to find

$$\Sigma = (g_1^2 + g_2^2 + g_3^2 + \ldots + g_n^2)^{1/2} \tag{4.26}$$

and divide each component of **g** by Σ.

Stage 2: To establish lower and upper bounds on the optimal step size λ^*, we have to find two points A and B at which the slope $df/d\lambda$ has different signs. We know that at λ=0,

$$\left.\frac{df}{d\lambda}\right|_{\lambda=0} = \mathbf{g}^{\mathbf{T}}\nabla f(\mathbf{X}) < \mathbf{0}\,, \text{ Since } \mathbf{g} = -\nabla f(\mathbf{X}) \tag{4.27}$$

since **g** is assumed to be a direction of descent. i.e., the angle between the direction of steepest descent and **g** will be less than 90°. Hence, to start with, we can take $A = 0$ and try to find a point $\lambda = B$ at which the slope $df/d\lambda$ is positive. The point B can be taken as the first value out of t_0, 2 t_0, 3 t_0, 4 t_0, 8 t_0, . . . at which f' is nonnegative, where t_0 is a preassigned initial step size. It then follows that λ^* is bounded in the interval $A < \lambda^* \leq B$ as shown in (Fig 4–3).

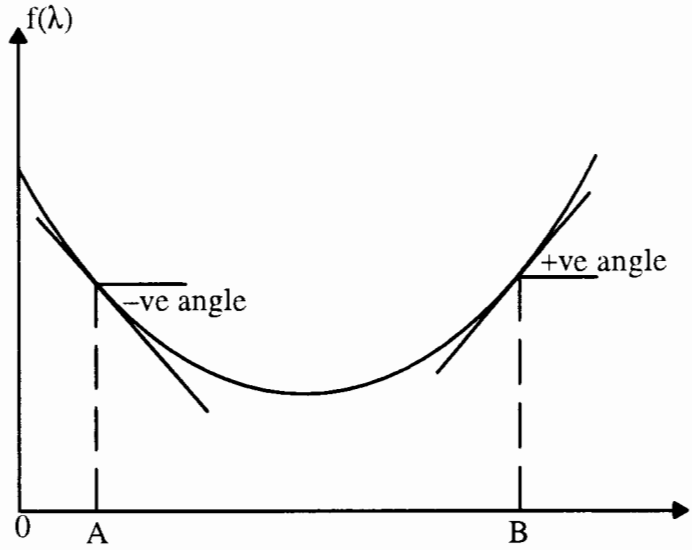

Figure 4–3 Minimum of $f(\lambda)$ lies between A and B.

Stage 3: If the cubic equation

$$h(\lambda) = a + b\lambda + c\lambda^2 + d\lambda^3 \tag{4.28}$$

is used to approximate the function $f(\lambda)$ between the points A and B, we have to find the values

$$f_A = f(\lambda = A),\ f_A' = \frac{df}{d\lambda}(\lambda = A),\ f_B = f(\lambda = B) \text{ and } f_B' = \frac{df}{d\lambda}(\lambda = B) \quad \text{in order}$$

to evaluate the constants, a, b, c and d in the cubic equation above Eq.(4.28). By assuming that $A \neq 0$, we can derive a general formula for $\tilde{\lambda}^*$. From Eq.(4.28), we have

$$\left.\begin{aligned} f_A &= a + bA + cA^2 + dA^3 \\ f_B &= a + bB + cB^2 + dB^3 \\ f_A' &= b + 2cA + 3dA^2 \\ f_B' &= b + 2cB + 3dB^2 \end{aligned}\right\} \tag{4.29}$$

Equations (4.29) can be solved to find the constants as

$$a = f_A - bA - cA^2 - dA^3 \tag{4.30}$$

where

$$b = \frac{1}{(A - B)^2}\left[B^2 f_A' + A^2 f_B' + 2ABZ\right] \tag{4.31}$$

$$c = -\frac{1}{(A - B)^2}\left[Bf_A' + Af_B' + (A + B)Z\right] \tag{4.32}$$

and

$$d = \frac{1}{3(A - B)^2}\left[f_A' + f_B' + 2Z\right] \tag{4.33}$$

where

$$Z = \frac{3(f_A - f_B)}{(B - A)} + \left[f_A' + f_B'\right] \tag{4.34}$$

The necessary condition for the minimum of $h(\lambda)$ given by Eq.(4.28) is that

$$\frac{dh}{d\lambda} = b + 2c\lambda + 3d\lambda^2 = 0 \tag{4.35}$$

i.e.,

$$\tilde{\lambda}^* = \frac{-c \pm (c^2 - 3bd)^{1/2}}{3d} \tag{4.36}$$

Application of the sufficiency condition for the minimum of $h(\lambda)$ leads to the condition

$$\left.\frac{d^2h}{d\lambda^2}\right|_{\tilde{\lambda}^*} = 2c + 6d\tilde{\lambda}^* > 0 \tag{4.37}$$

By substituting the expression for b, c and d given by Eqs.(4.30) to (4.33) into Eqs. (4.36) and (4.37), we obtain

$$\tilde{\lambda}^* = A + \frac{(f_A' + Z \pm Q)}{(f_A' + f_B' + 2Z)}(B - A) \tag{4.38}$$

where

$$Q = (Z^2 - f_A' f_B')^{1/2} \tag{4.39}$$

$$2(B - A)(2Z + f_A' + f_B')(f_A' + Z \pm Q)$$

$$- 2(B - A)(f_A'^2 + Zf_B' + 3Zf_A' + 2Z^2)$$

$$- 2(B + A)f_A' f_B' > 0 \tag{4.40}$$

By specializing the Eqs.(4.30) to (4.40) to the case where $A = 0$, we obtain

$$a = f_A$$

$$b = f_A'$$

$$\tilde{\lambda}^* = B\left(\frac{f_A' + Z \pm Q}{f_A' + f_B' + 2Z}\right) \tag{4.41}$$

and $$Q = (Z^2 - f_A' f_B')^{1/2} > 0 \tag{4.42}$$

where $$Z = \frac{3(f_A - f_B)}{B} + f_A' + f_B' \tag{4.43}$$

The two values of $\lambda^{\sim *}$ in Eqs.(4.38) and (4.41) correspond to the two possibilities for the vanishing of $h'(\lambda)$, i.e., at a maximum and at a minimum.

In order to avoid imaginary values for Q, we should ensure the satisfaction of the condition

$$Z^2 - f_A' f_B' \geqq 0 \tag{4.44}$$

in Eq. (4.39). This inequality is automatically ensured as we are assuming $f_A' < 0$ and $f_B' \geq 0$. Furthermore, sufficiency condition when $A = 0$ requires that $Q > 0$, which is already satisfied. Now, we calculate $\lambda^{\sim *}$ according to the formula (4.41), and proceed to the next stage.

Stage 4: The value of $\lambda^{\sim *}$ found in stage 3 is the true minimum of $h(\lambda)$ and may not be close to the minimum of $f(\lambda)$. Hence the following convergence criteria can be used before taking λ^* approximately as $\lambda^{\sim *}$.

$$\left|\frac{h(\tilde{\lambda}^*) - f(\tilde{\lambda}^*)}{f(\tilde{\lambda}^*)}\right| \leq \epsilon_1 \tag{4.45}$$

$$\left|\frac{df}{d\lambda}\Big|_{\tilde{\lambda}^*}\right| = \left|\mathbf{g}^T \nabla f|_{\tilde{\lambda}^*}\right| \leq \epsilon_2 \tag{4.46}$$

where ε_1 and ε_2 are small numbers whose values depend on the accuracy desired.

The latter criterion can be stated in non–dimensional form as

$$\left|\frac{\mathbf{g}^T \nabla f}{|\mathbf{g}|\,|\nabla f|}\right|_{\tilde{\lambda}^*} \leq \epsilon_2 \tag{4.47}$$

If the criteria stated in Eqs. (4.45) and (4.47) are not satisfied, a new cubic equation

$$h'(\lambda) = a' + b'\lambda + c'\lambda^2 + d'\lambda^3 \tag{4.58}$$

can be used to approximate $f(\lambda)$. The constants a', b', c', and d' can be determined by using the function derivative values at the best two points out of the three points currently available, namely, A, B, and $\tilde{\lambda}^*$. Now the general formula given by Eq. (4.38) has to be used for finding the optimal step length $\tilde{\lambda}^*$. If $f'(\tilde{\lambda}^*) < 0$, the new points A and B are taken as $\tilde{\lambda}^*$ and B respectively; otherwise (if $f'(\tilde{\lambda}^*) > 0$), the new points A and B are taken as A and $\tilde{\lambda}^*$, and Eq. (4.38) is applied to find the new value of $\tilde{\lambda}^*$. Equations (4.45) and (4.47) are again used to test for the convergence of $\tilde{\lambda}^*$. If the convergence is achieved, $\tilde{\lambda}^*$ is taken as λ^* and the procedure is stopped. Otherwise, the whole procedure is repeated until the desired convergence.

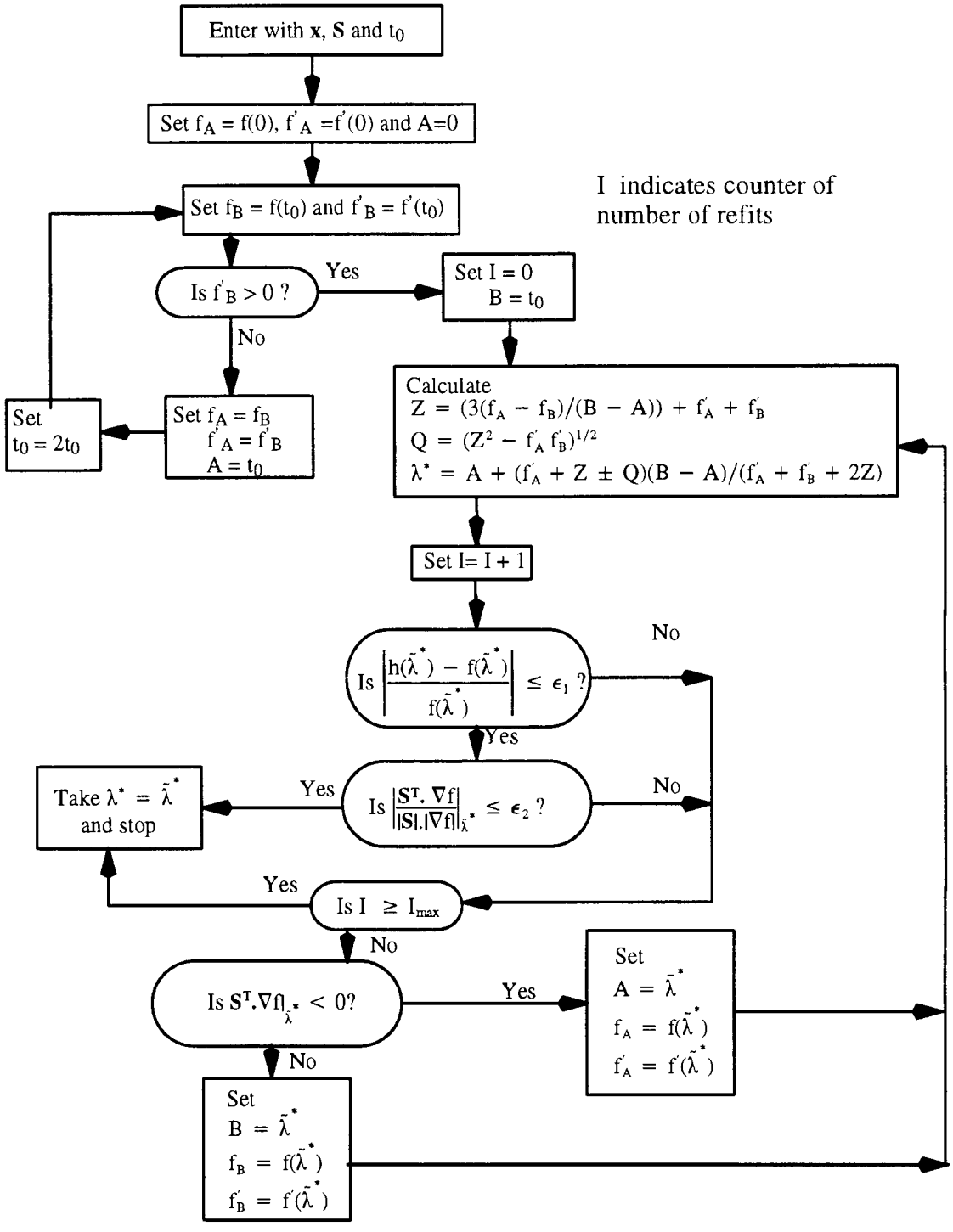

Figure 4–4 Flow chart for cubic interpolation method.

4.2.2 Application Of Cubic Interpolation

Find the minimum of $f(\lambda) = \lambda^5 - 5\lambda^3 - 20\lambda + 5$ by cubic interpolation method.

Solution

We take $A = 0$ and find that

$$\frac{df}{d\lambda}(\lambda = A = 0) = 5\lambda^4 - 15\lambda^2 - 20|_{\lambda=0} = -20 < 0$$

To find B at which $df/d\lambda$ is nonnegative, we start with $t_0 = 0.4$ and evaluate the derivative at t_0, $2\,t_0$, $4\,t_0$, ... This gives

$$f'(t_0 = 0.4) = 5(0.4)^4 - 15(0.4)^2 - 20.0 = -22.272$$
$$f'(2t_0 = 0.8) = 5(0.8)^4 - 15(0.8)^2 - 20.0 = -27.552$$
$$f'(4t_0 = 1.6) = 5(1.6)^4 - 15(1.6)^2 - 20.0 = -25.632$$
$$f'(8t_0 = 3.2) = 5(3.2)^4 - 15(3.2)^2 - 20.0 = 350.688$$

Thus we find that

$$A = 0.0,\ f_A = 5.0, \qquad f'_A = -20.0$$
$$B = 3.2,\ f_B = 113.0, \qquad f'_B = 350.688$$

and $A < \lambda^* < B$.

Iteration 1

To find the value of $\tilde{\lambda}^*$ and to test the convergence criteria, we first compute Z and Q as:

$$Z = \frac{3(5.0 - 113.0)}{3.2} - 20.0 + 350.688 = 229.588$$

$$Q = \left[229.588^2 + (20.0)(350.688)\right]^{1/2} = 244.0$$

Hence

$$\tilde{\lambda}^* = 3.2\left(\frac{-20.0 + 229.588 \pm 244.0}{-20.0 + 350.688 + 459.176}\right) = 1.84 \text{ or } -0.1396$$

By discarding the negative value, we have $\tilde{\lambda}^* = 1.84$

Convergence criterion: If $\tilde{\lambda}^*$ is close to the true minimum, λ^*, then $f'(\tilde{\lambda}^*) = df(\tilde{\lambda}^*)/d\lambda$ should be approximately zero

Since $f' = 5\lambda^4 - 15\lambda^2 - 20$,

$$f'(\tilde{\lambda}^*) = 5(1.84)^4 - 15(1.84)^2 - 20 = -13.0$$

Since this is not too small, we go to the next iteration or refitting. As

$$f'(\tilde{\lambda}^*) < 0, \text{ we take } A = \tilde{\lambda}^*$$

and

$$f_A = f(\tilde{\lambda}^*) = (1.84)^5 - 5(1.84)^3 - 20(1.84) + 5 = -41.70$$

Therefore

$$A = 1.84, \quad f_A = -41.70, \quad f'_A = -13.0,$$
$$B = 3.2, \quad f_B = 113.0, \quad f'_B = 350.688,$$

$$A < \tilde{\lambda}^* < B.$$

Iteration 2

$$Z = \frac{3(-41.7 - 113.0)}{(3.20 - 1.84)} - 13.0 + 350.688 = -3.312$$

$$Q = \left[(-3.312)^2 + (13.0)(350.688)\right]^{1/2} = 67.5$$

Hence

$$\tilde{\lambda}^* = 1.84 + \left(\frac{-13.0 - 3.312 \pm 67.5}{-13.0 + 350.688 - 6.624}\right)(3.2 - 1.84) = 2.05$$

Since this value is large, we go to the next iteration with $B = \lambda\sim^* = 2.05 (\text{as } f'(\lambda\sim^*) > 0)$

and

$$f_B = (2.05)^5 - 5.0(2.05)^3 - 20.0(2.05) + 5.0 = -42.90$$

Therefore

$$A = 1.84, f_A = -41.70, f'_A = -13.00,$$

$$B = 2.05, f_B = -42.90, f'_B = 5.35,$$

and

$$A < \lambda^* < B.$$

Iteration 3

$$Z = \frac{3.0(-41.70 + 42.90)}{(2.05 - 1.84)} - 13.00 + 5.35 = 9.49$$

$$Q = \left[(9.49)^2 + (13.0)(5.35)\right]^{1/2} = 12.61$$

$$\therefore \quad \tilde{\lambda}^* = 1.84 + \left(\frac{-13.00 + 9.49 \pm 12.61}{-13.00 + 5.35 + 18.98}\right)(2.05 - 1.84) = 2.0086$$

convergence criterion:

$$f'(\tilde{\lambda}^*) = 5.0(2.0086)^4 - 15.0(2.0086)^2 - 20.0 = 0.855$$

Assuming that this value is close to zero, we can stop the iterative process and take $\lambda^* \simeq \tilde{\lambda}^* = 2.0086$.

CHAPTER V

SIMULATION RESULTS

This chapter presents a variety of nonlinear phase filters, that are cascaded with an all–pass filter to equalize their phase using the variable metric method of Davidon–Fletcher–Powell algorithm. The Program is first applied to the equalization of the group delay of a 4th–order low–pass elliptic filter that Deczky [18] used in his 1972 paper of Synthesis of Recursive Digital Filters Using the Minimum p–Error Criterion. The original group delay distortion of this filter was about $10T$ seconds and its specifications were as follows:

5.1 Equalization of a 4th–Order Elliptic Filter

Maximum passband ripple is : 0.5 dB $0 \leq \phi \leq 0.5 f_n$

Minimum stopband attenuation: 32.0 dB $0.6 f_n \leq \phi \leq f_n$

where $f_n = 1/2T$, The Nyquist frequency coefficients as follows:

$$k_0 = 1.47295 * 10^{-01}$$

$$a_{11} = 1.62178, \quad a_{12} = 0.71895, \quad a_{21} = 1.0, \quad a_{22} = 1.0$$

$$b_{11} = -0.403133, \quad b_{12} = 0.051401, \quad b_{21} = 0.233280, \quad b_{22} = 0.797295$$

The passband group delay of this filter was next equalized using all–pass sections of the form:

$$H_E(z) = \prod_{j=1}^{M} \frac{\left(C_{0j} z^2 + C_{1j} z + 1\right)}{\left(z^2 + C_{1j} z + C_{0j}\right)} \tag{5.1}$$

The distance function used in this case was

$$\mathbf{E}_{2q}(\mathbf{X}) = \frac{1}{2q}\sum_{i=1}^{L}|\mathbf{e}_i(\mathbf{X})|^{2q} \tag{5.2}$$

$$\mathbf{e}_i(\mathbf{X}) = \tau_f(\omega_i) + \tau_e(\mathbf{X}, \omega_i) - \tau_0 \tag{5.3}$$

where $\boldsymbol{\tau}_f$ is the group delay of the original filter, $\boldsymbol{\tau}_e$ is that of the equalizer, and τ_0 is a constant delay whose value is to be determined by minimizing $\mathbf{E}_{2q}(\mathbf{X})$ with respect to τ_0.

$$\mathbf{X} = \left[C^T, \tau_0\right]^T, \qquad \tau_0 = \frac{\tau}{T} \tag{5.4}$$

$\mathbf{C} = \left[C_{01}, C_{11}, C_{02}, C_{12}, \ldots, C_{1M}\right]^T$ is the vector matrix of the coefficients of an All–pass filter that will be used to equalize a low–pass IIR filter. Using a one–section equalizer which means (M = 1), an index 2q = 2, L= 15 sampling points, and the starting set of poles (in polar coordinates using normalized angles) are

$r_{p1} = 0.8 \quad \phi_{p1} = 0.18 \quad \tau_0 = 0.0.$

The program converged to the solution (with an error criterion $\epsilon = 10^{-5}$ in Fletcher Powell algorithm, and CPU time = 6min) to

$X_{min} = \left[R_p, \phi_p, \tau\right]$, where

$R_p = 0.3225, \ \phi_p = -0.4597, \ \tau = 4.4414$

The group delay of this filter before and after equalization compared to Deczky's is shown in Figure. 5–1. The Poles–Zeros plot of the final All–pass filter (used to equalize the 4th–order elliptic filter) and the overall plot of Poles–Zeros of final filter after equalization are shown in figures. 5–2 and 5–3 , respectively. The final filter has linear phase and it is stable since its poles are located inside the unit circle in the z–domain plot.

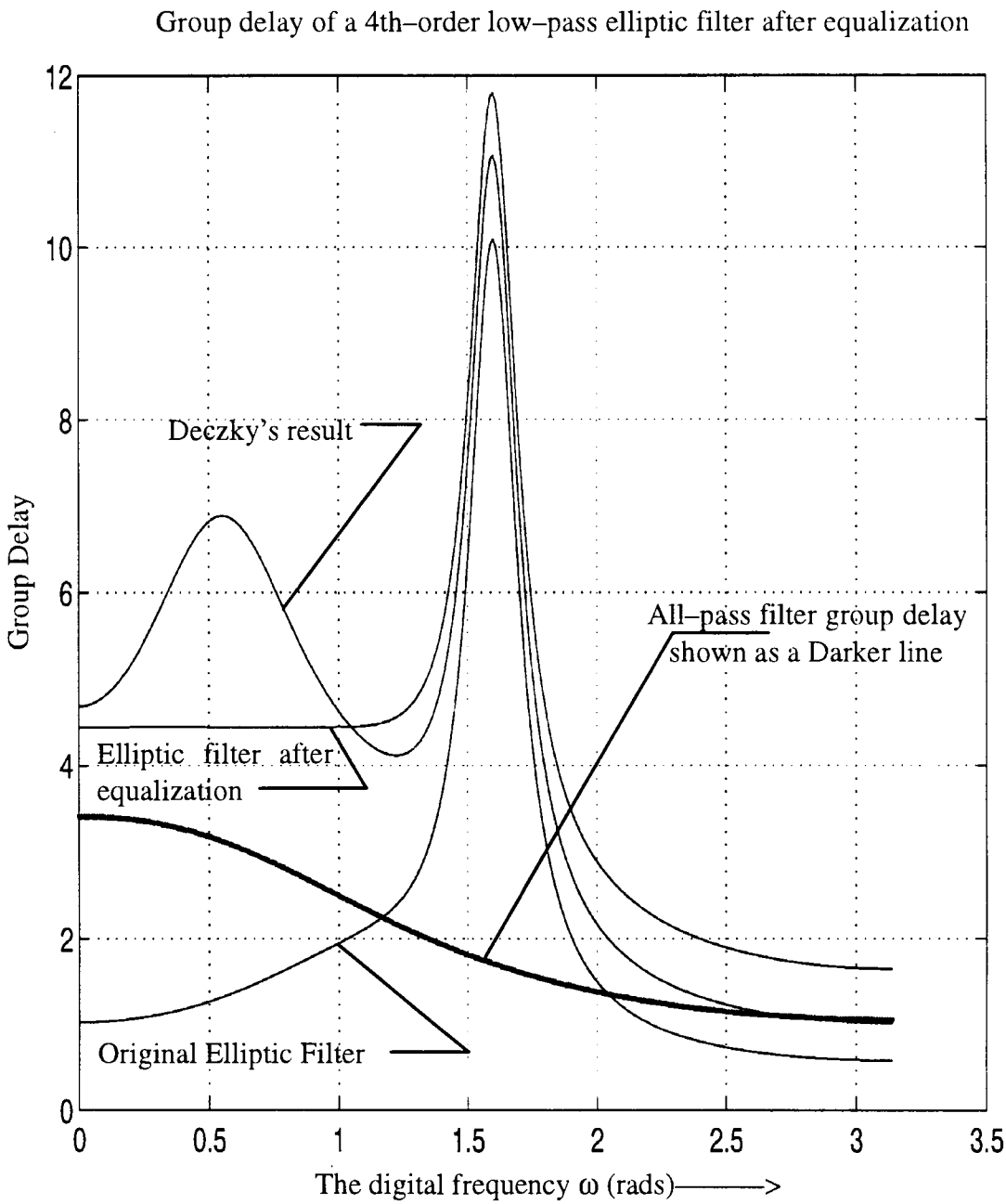

Figure 5–1 Group delay of the fourth–order low–pass elliptic filter after equalization by a one–section all–pass with index 2q=2, and L=15 samples/period.

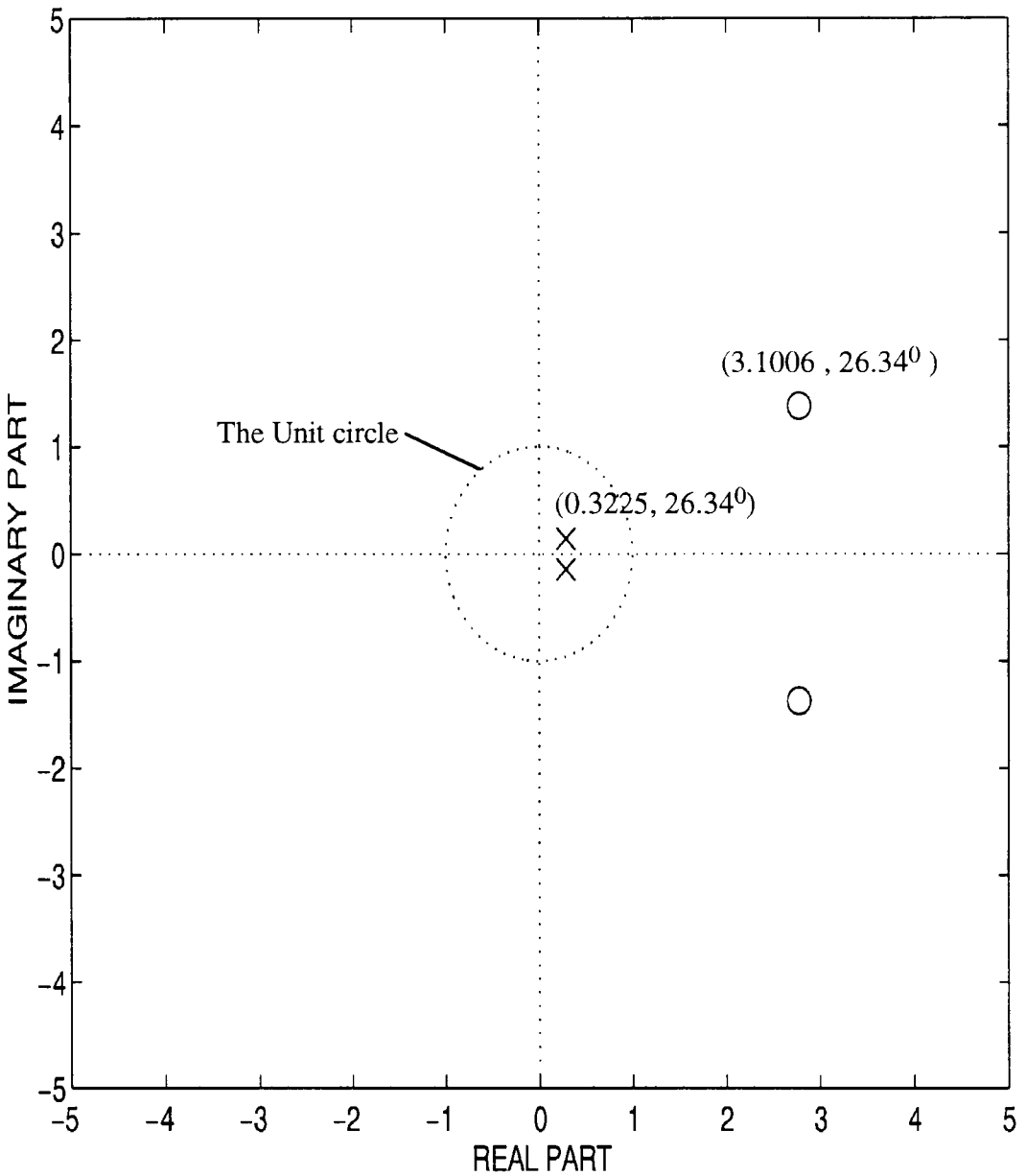

Figure 5–2 The Poles and Zeros Plot of second order all–pass filter used to equalize fourth–order elliptic filter.

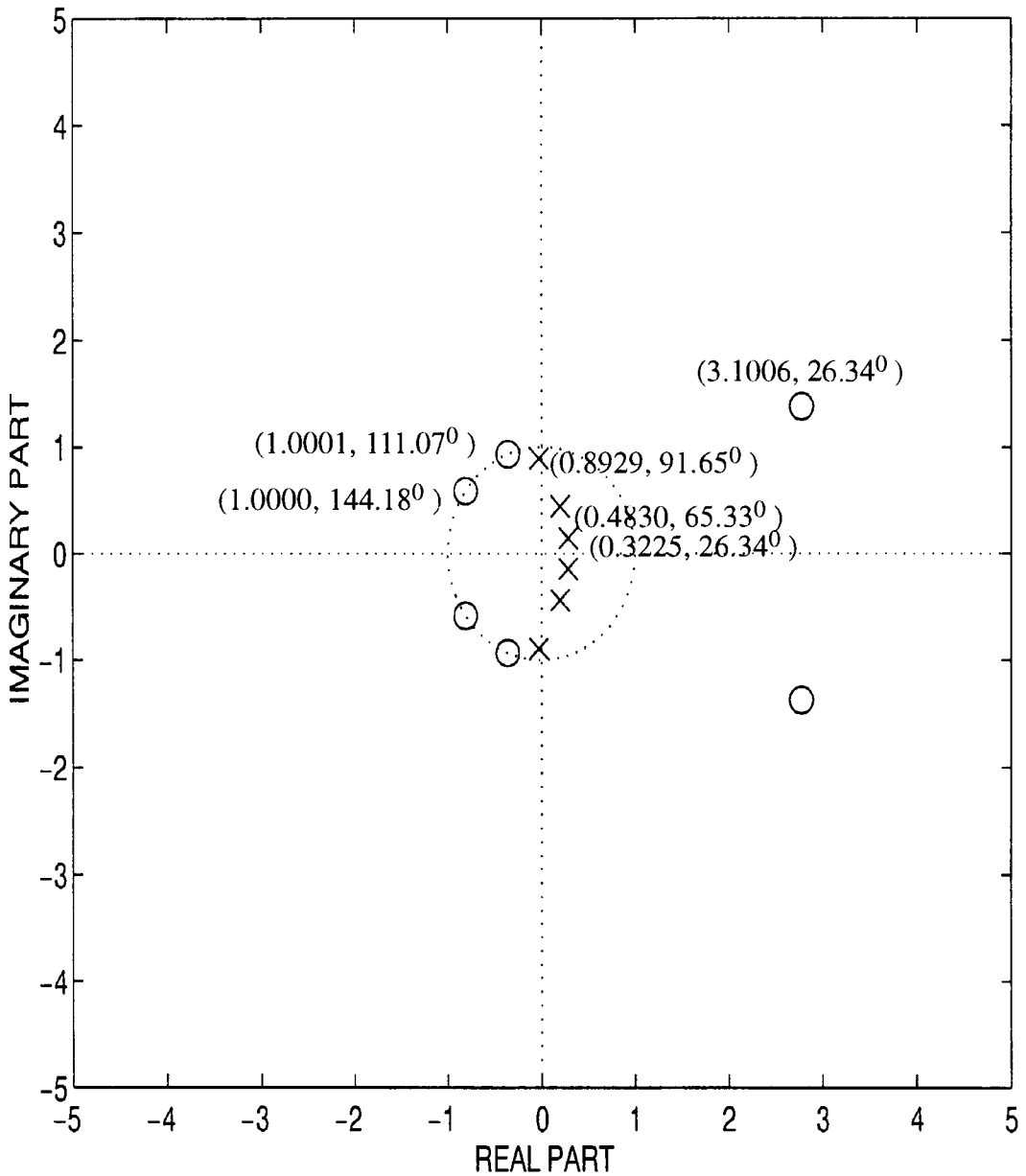

Figure 5–3 Poles and Zeros plot of a linear–phase 6–th order elliptic filter after being equalized by a one section second order all–pass filter

5.2 Equalization Of 14th–order Butterworth Filter

In this example we consider the equalization of a 14th–order Butterworth filter from the given design specifications:

$\delta_1 = 0.01$, Maximum variation or Ripple in the Pass–Band. $0 \leq \omega_p \leq 0.4\pi$.

$\delta_2 = 0.001$, Maximum variation or Ripple in the Stop–Band. $0.6\pi \leq \omega_s \leq \pi$.

Using a all–pass equalizer , index 2q=2, sampling points L=15, and initial set of poles(in polar coordinates using normalized angles):

$X_0 = [0.2, 0.12, 0.0]$. The converged after 5 minutes of CPU time ,with an error criterion $\epsilon = 1.0 * e^{-03}$, The Solution is :

$X_{min} = [0.5512, -0.3952, 10.3220]$, Minimum point reached.

$$H = \begin{bmatrix} 0.0083 & -0.0045 & 0.0307 \\ -0.0045 & 0.0072 & -0.0115 \\ 0.0307 & -0.0115 & 0.1881 \end{bmatrix}, \text{ The Hessian matrix.}$$

$$A = H^{-1} = \begin{bmatrix} 483.4978 & 201.9645 & -66.4354 \\ 201.9645 & 238.0672 & -18.3262 \\ -66.4354 & -18.3262 & 15.0000 \end{bmatrix}, \text{ the Jacobian of second derivatives}$$

$\nabla f|_{X_{min}} = 1.0e^{-04} * [0.1039, -0.1561, -0.0748]$, the gradient at the minimum point

the Poles/Zeros for the required all–pass filter after equalization:

$P_1 = [0.5087 \pm 0.2122i]$ and $Z_1 = [1.6743 \pm 0.6984i]$. The Solution is shown in the following graphs in Figures. (5–4) thru (5–10).

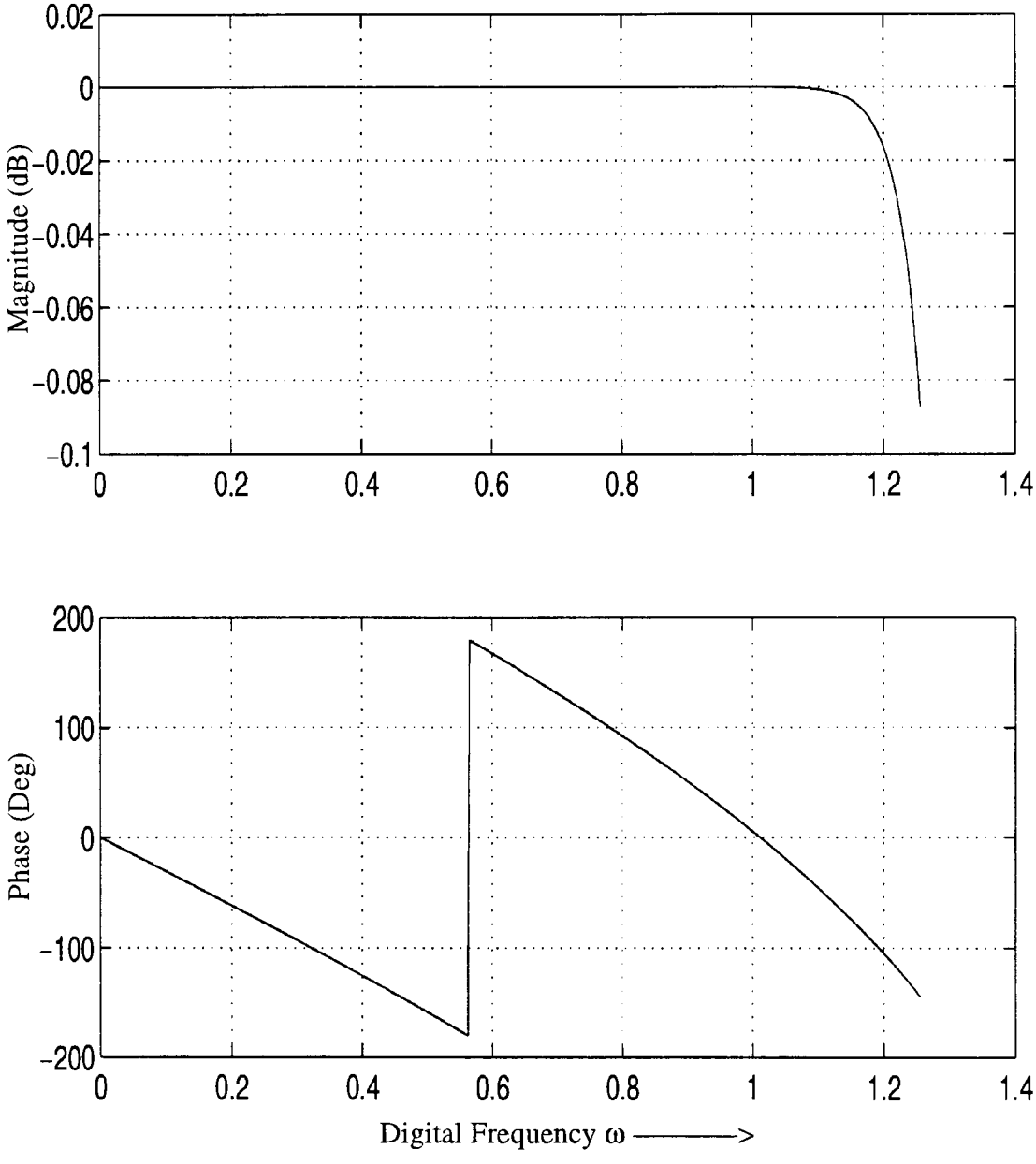

Figure. 5–4 The Magnitude and Phase response of a 14th–order low–pass Butterworth filter before equalization.

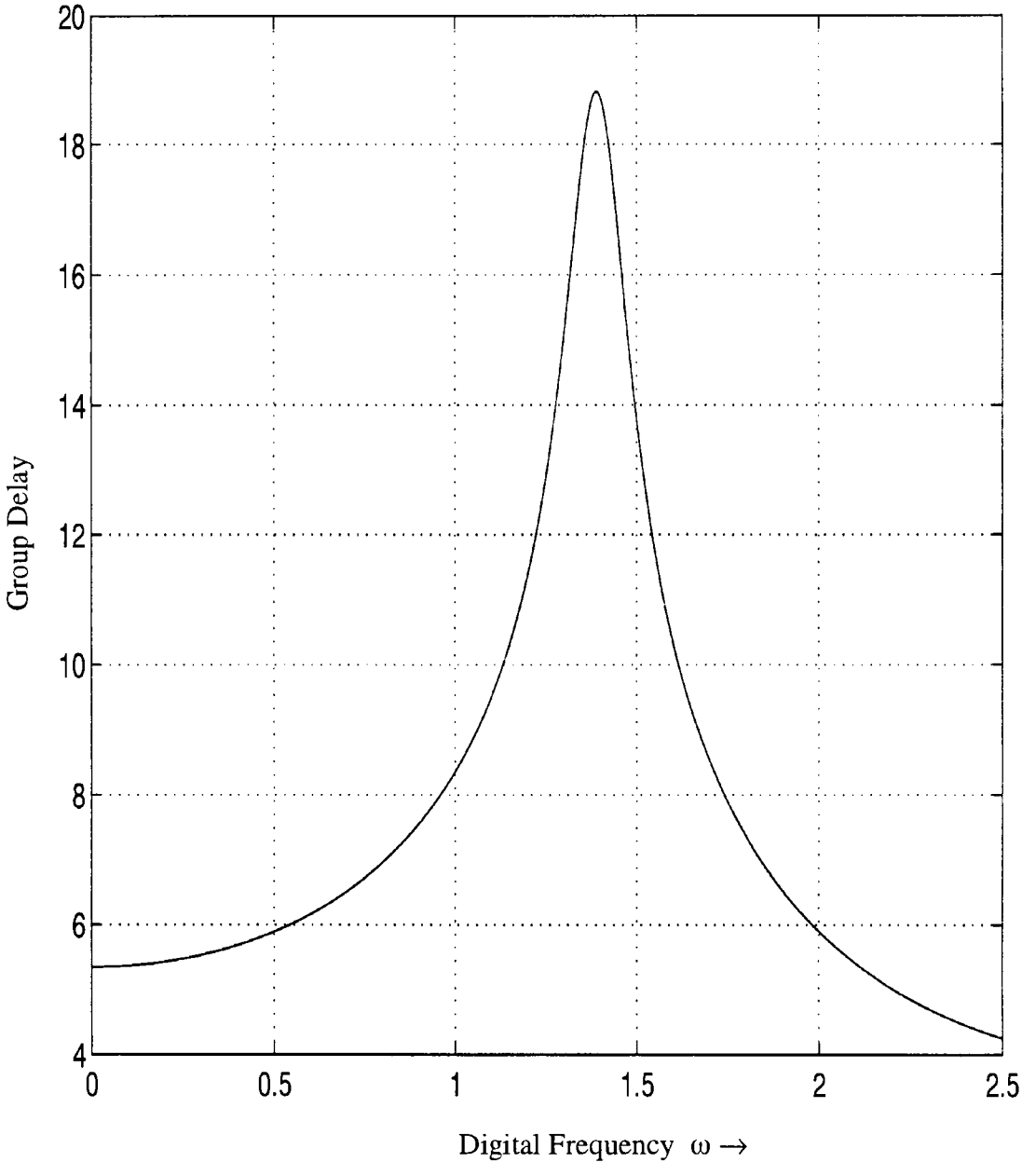

Figure 5–5 Group Delay of a 14th–order Butterworth IIR Filter

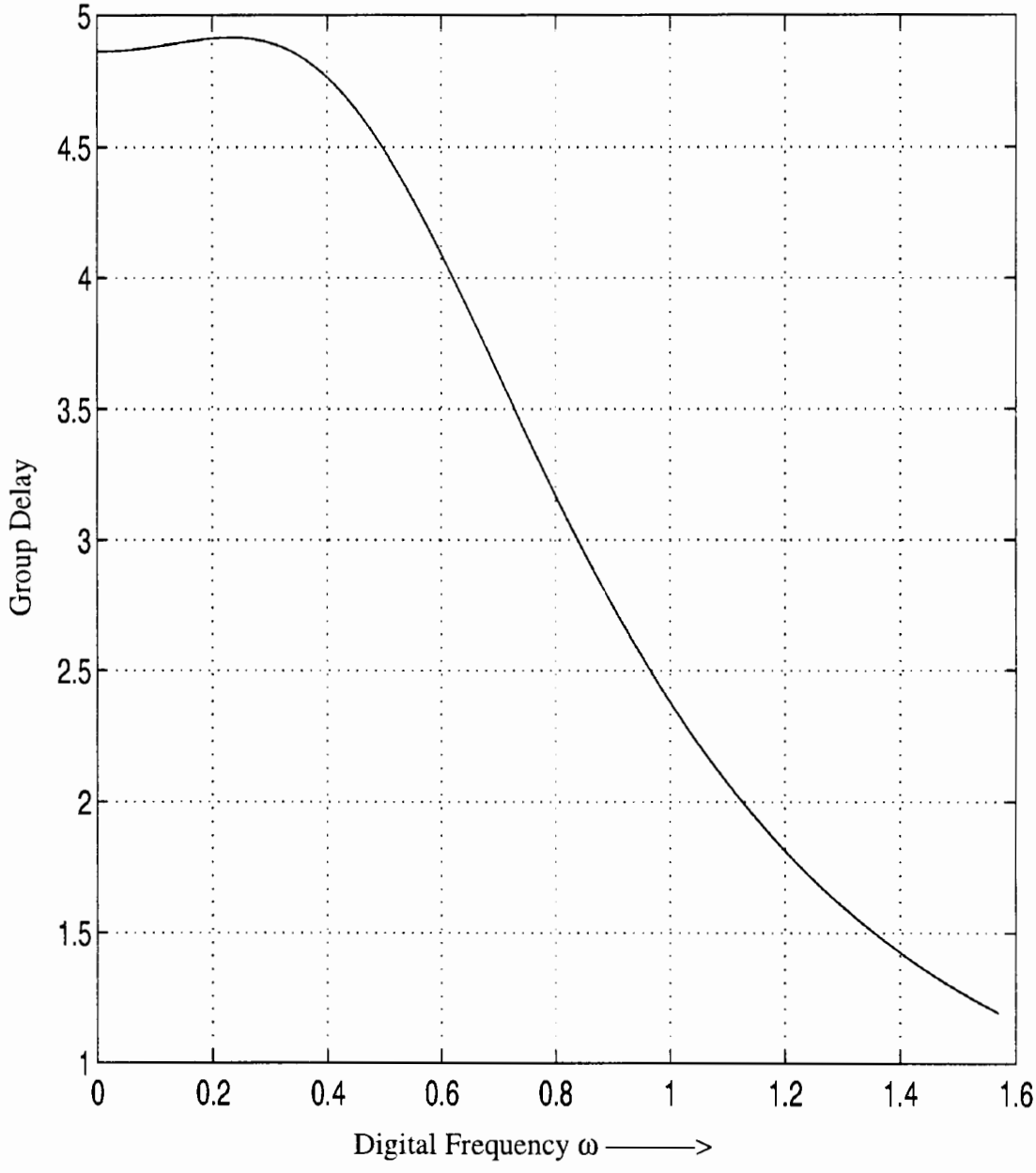

Figure 5–6 The Resulting Group delay of 2nd–order all–pass filter to equalize the 14th–order Butterworth filter

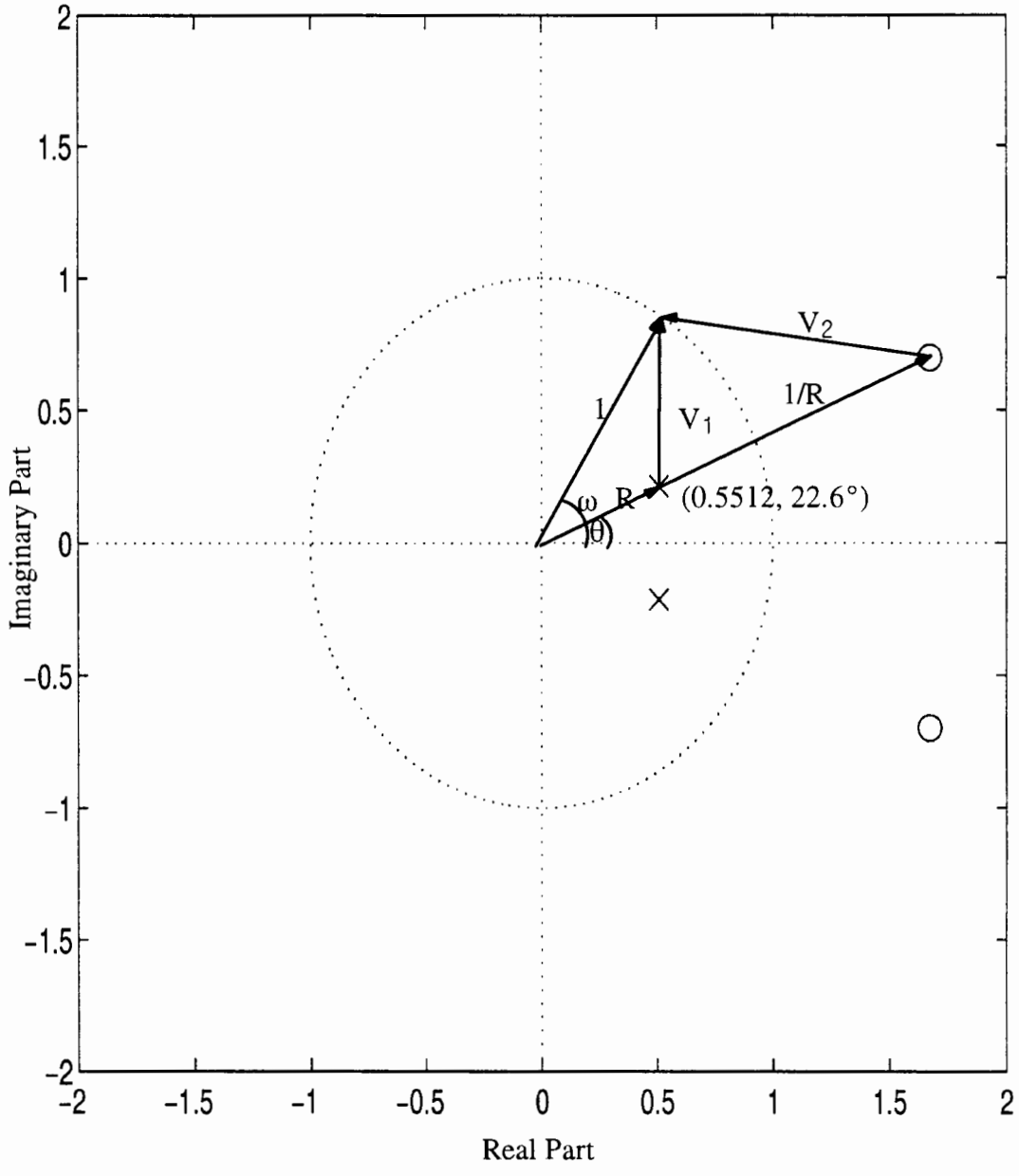

Figure 5–7 Poles–Zeros plot for an All–pass filter

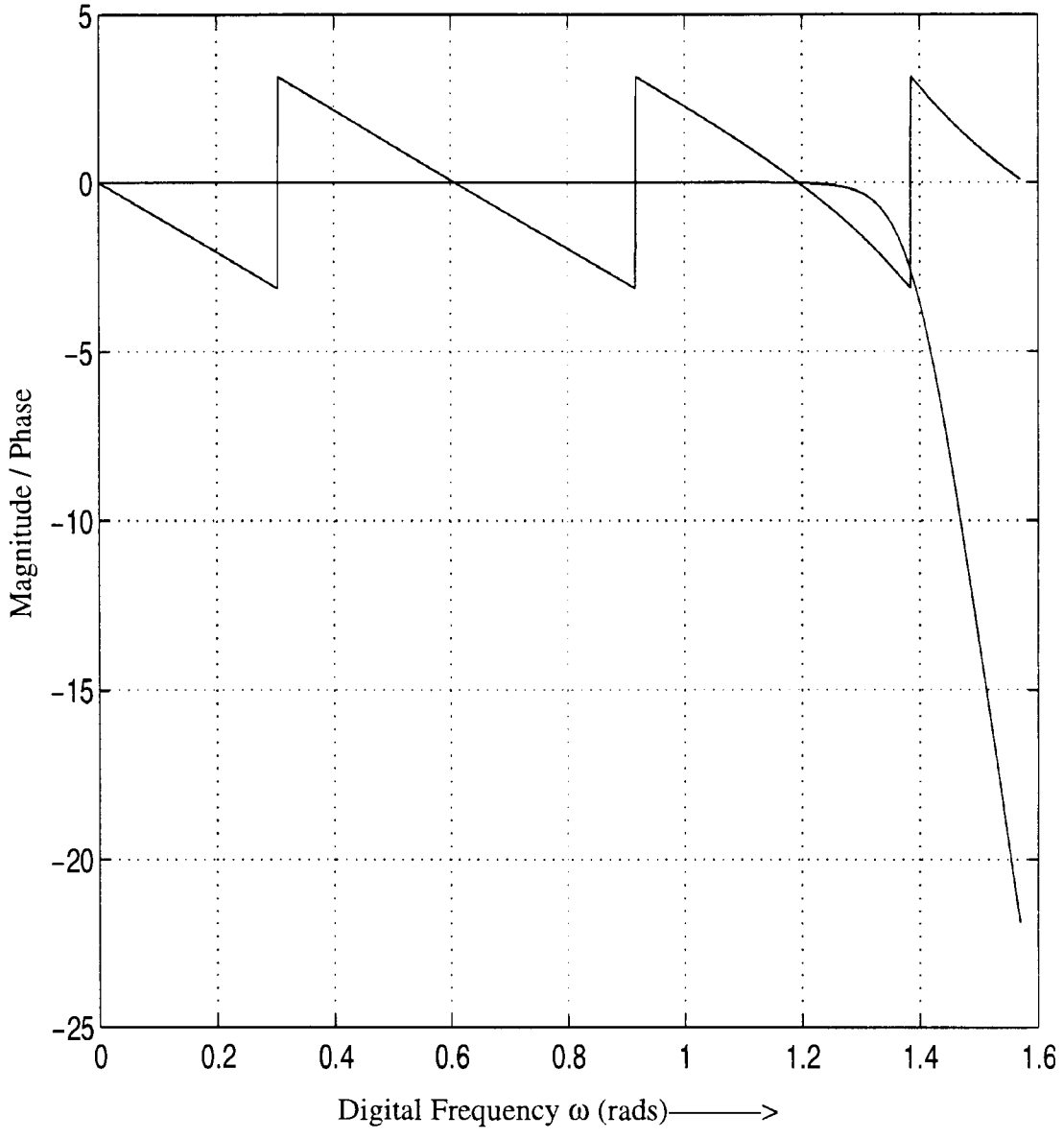

Figure 5–8 Magnitude and Phase of linear–phase 14th–order Butterworth filter after equalization by a 2nd–order all–pass filter

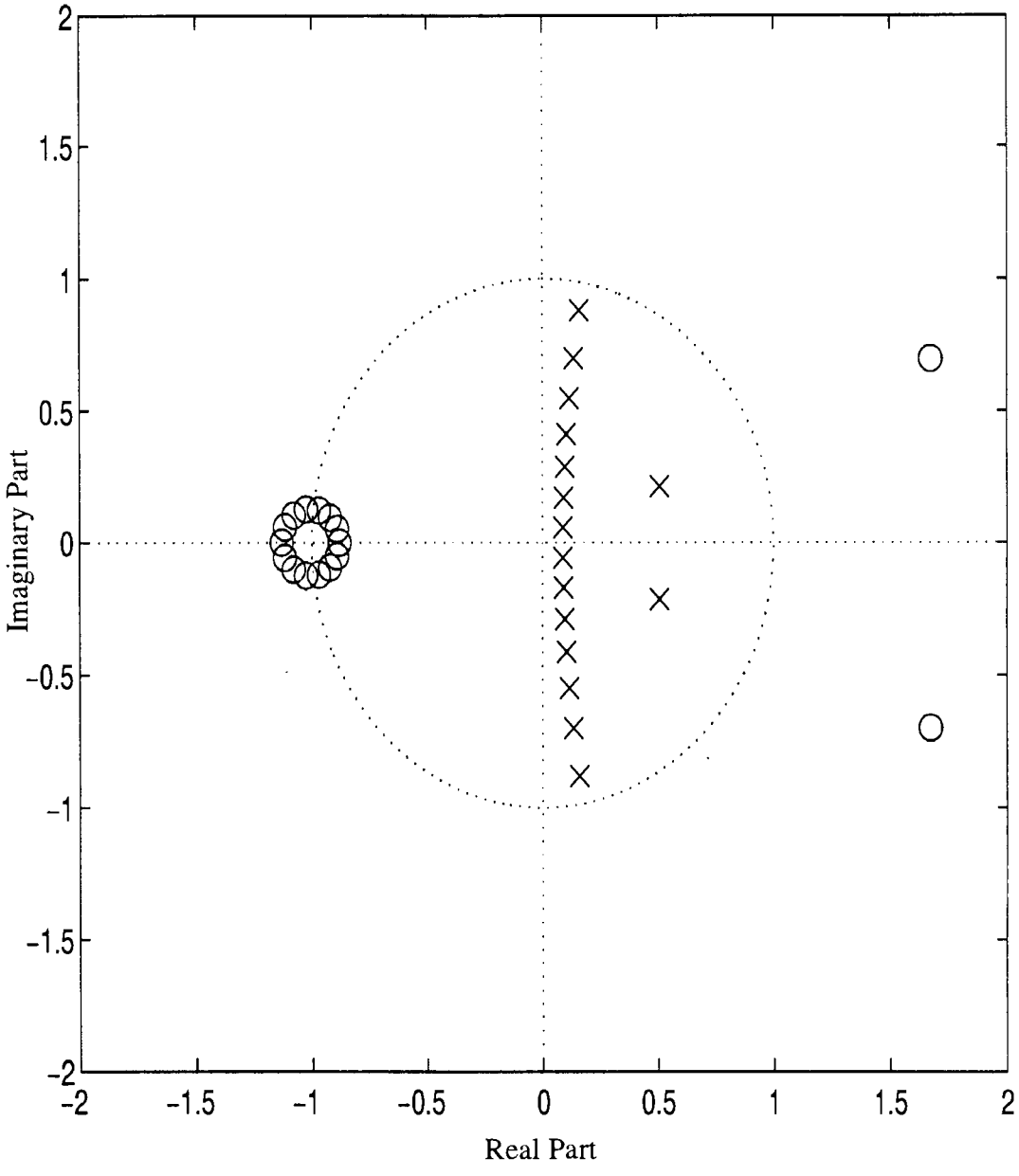

Figure 5–9 Zero–Pole plot of a 16th–order Butterworth IIR filter after being equalized by one–section all–pass filter.

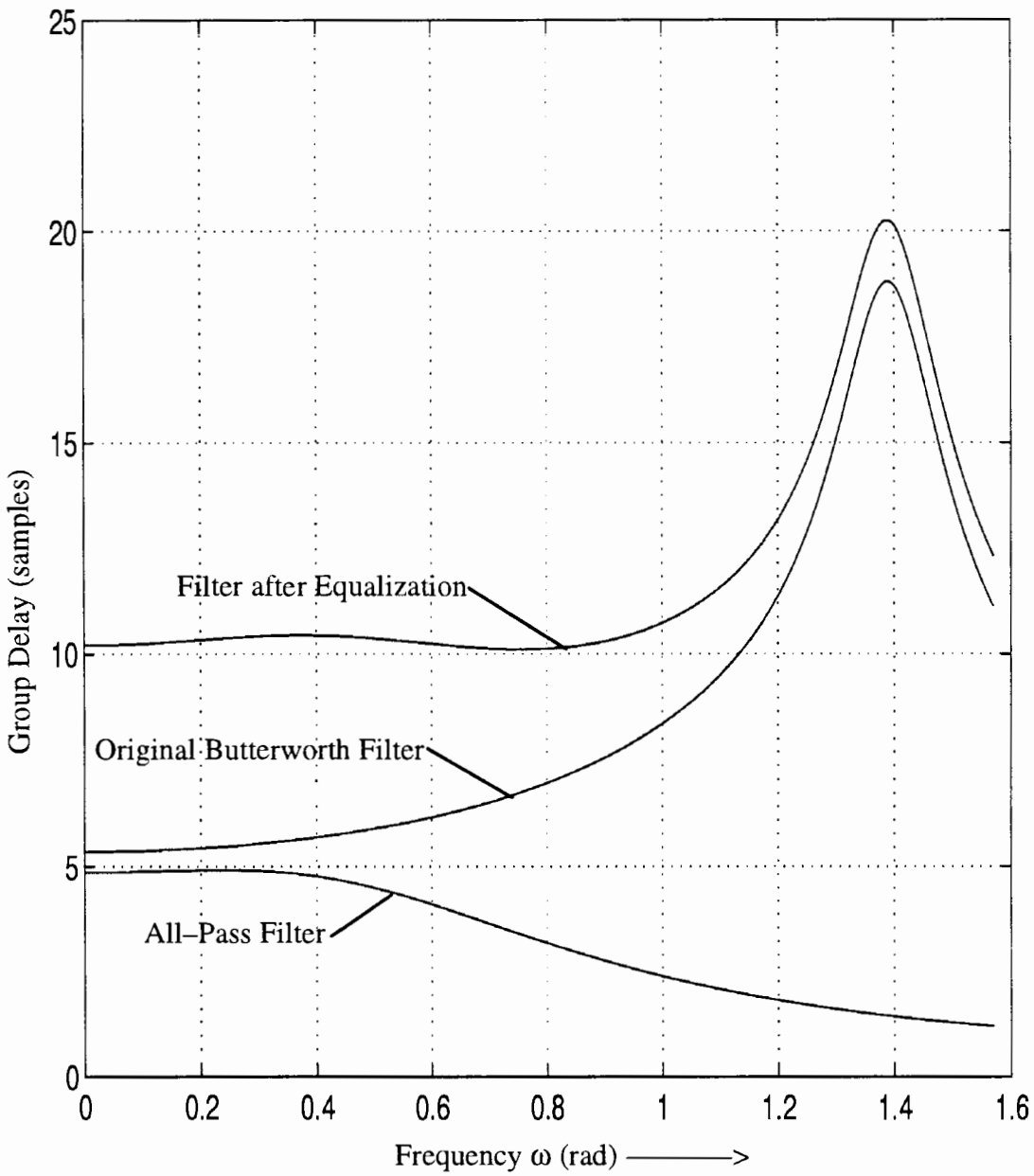

Figure 5–10 The Group Delay of 14th–order Butterworth filter after equalization by one–section All–pass filter.

5.3 Equalization Of a 6th–Order Chebyshev Filter

In this example we consider the equalization of a 6th–order Chebyshev filter from the given design specifications:

$\delta_1 = 0.01$, Maximum variation or Ripple in the Pass–Band. $0 \leq \omega_p \leq 0.4\pi$.

$\delta_2 = 0.001$, Maximum variation or Ripple in the Stop–Band. $0.6\pi \leq \omega_s \leq \pi$.

Using a all–pass equalizer , index 2q=2, sampling points L=15, and initial set of poles(in polar coordinates using normalized angles):

$X_0 = [0.2, 0.12, 0.0]$. The converged after 5 minutes of CPU time ,with an error criterion $\epsilon = 1.0 * e^{-03}$, The Solution is :

$X_{min} = [0.5938, -0.3448, 9.9993]$, Minimum point reached.

$$H = \begin{bmatrix} 0.0069 & -0.0030 & 0.0367 \\ -0.0030 & 0.0040 & -0.0108 \\ 0.0367 & -0.0108 & 0.2738 \end{bmatrix}, \text{ The Hessian matrix.}$$

$$A = H^{-1} = \begin{bmatrix} 797.5245 & 346.9652 & -93.3025 \\ 346.9652 & 430.3364 & -29.5085 \\ -93.3025 & -29.6090 & 15.0000 \end{bmatrix}, \text{ the Jacobian of second derivatives}$$

$\nabla f|_{X_{min}} = 1.0e^{-04} * [-0.2663, -0.1825, 0.0299]$, the gradient at the minimum point

the Poles/Zeros for the required all–pass filter after equalization:

$P_1 = [0.5588 \pm 0.2007i]$ and $Z_1 = [1.5951 \pm 0.5692i]$. The Solution is shown in the following graphs in Figures. (5–11) thru (5–15)

.

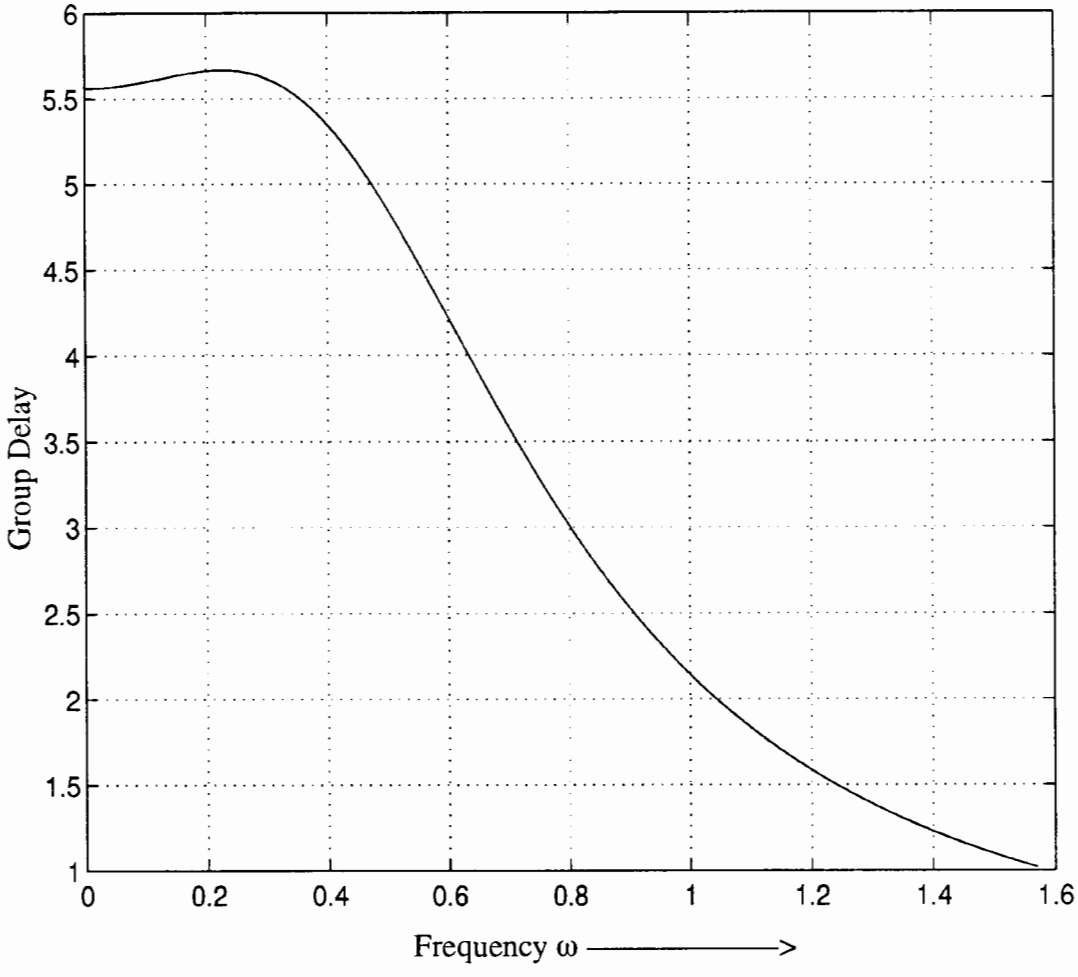

Figure 5–11 The Group–Delay of the required All–pass filter.

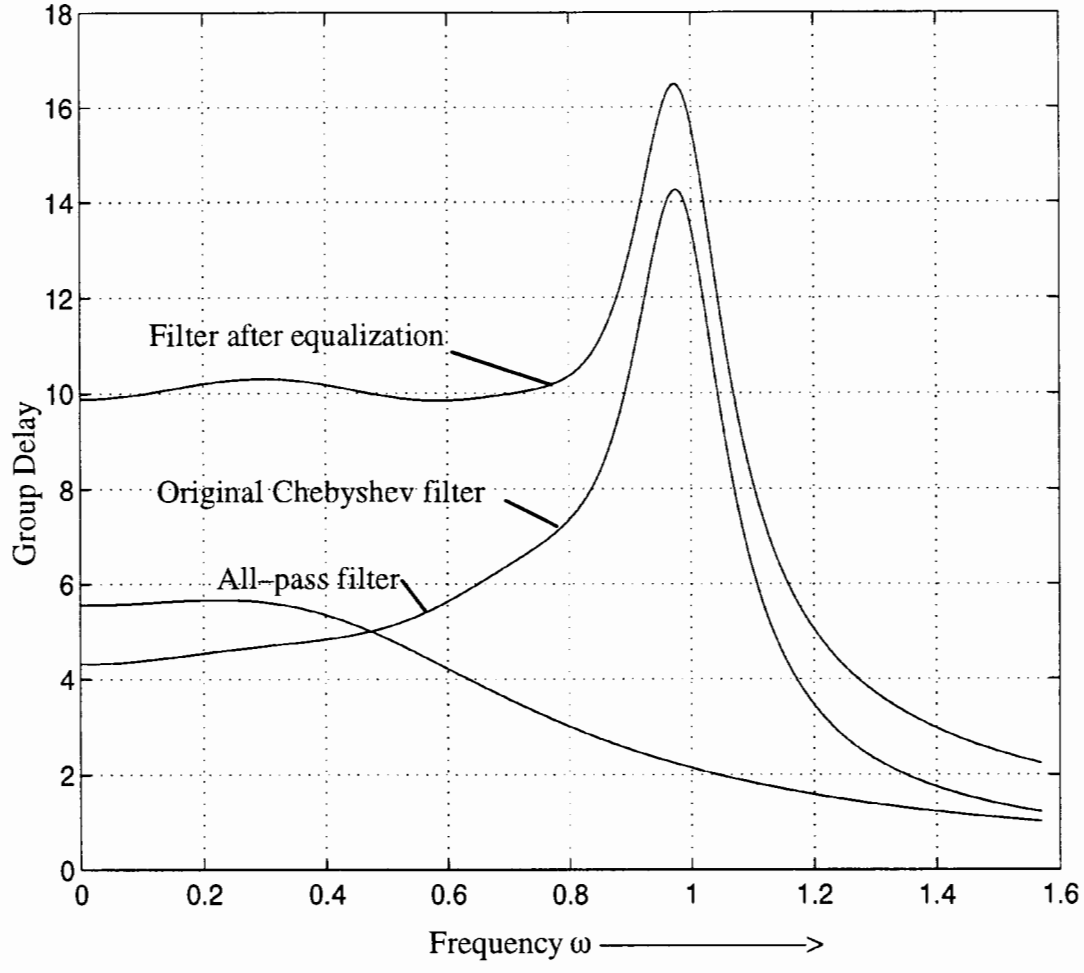

Figure 5–12 The group delay of a 6th–order Chebyshev filter before and after equalization by a second order All–pass filter.

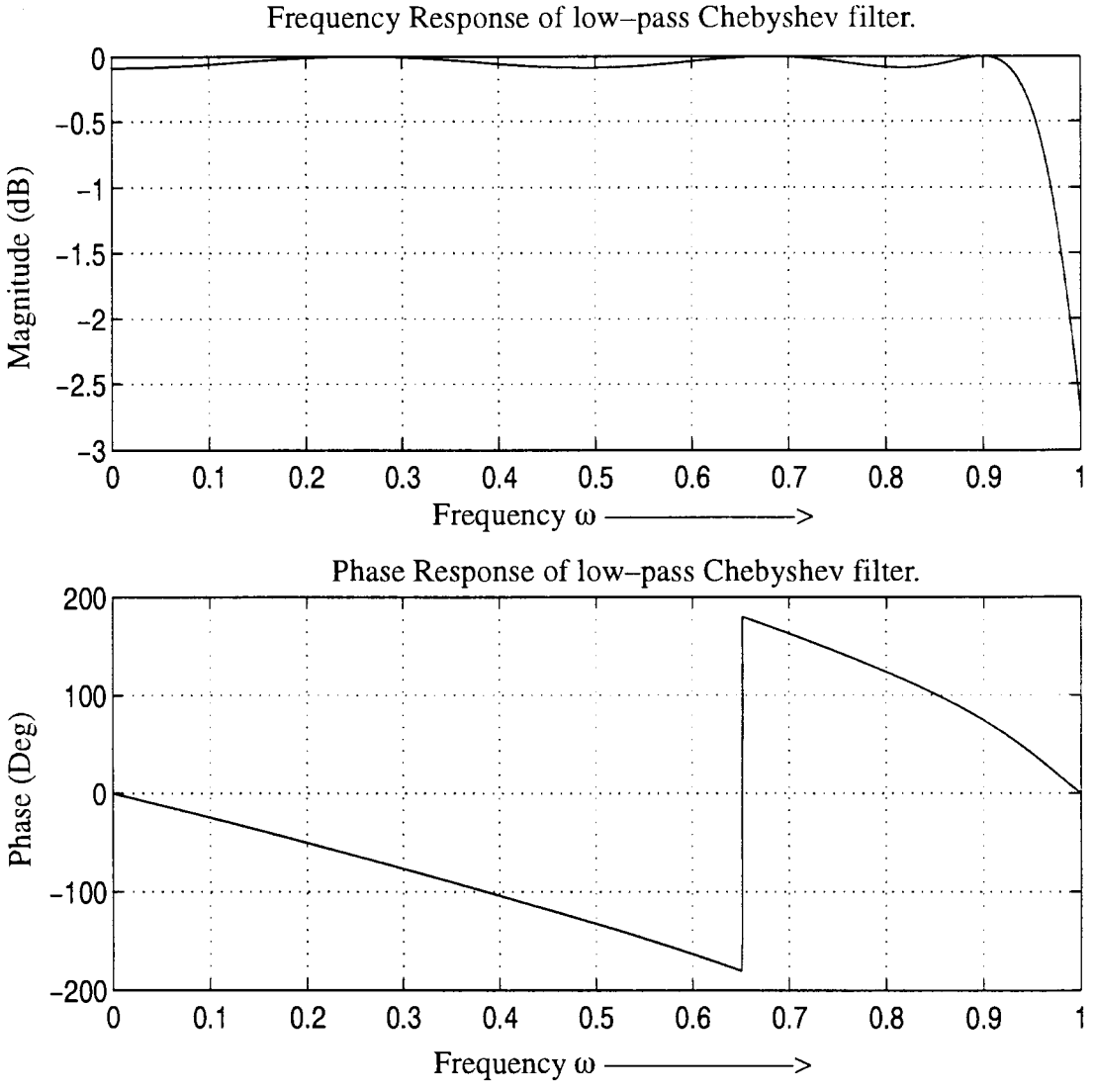

Figure 5–13 The Magnitude and Phase response of Chebyshev filter before equalization by All–pass filter

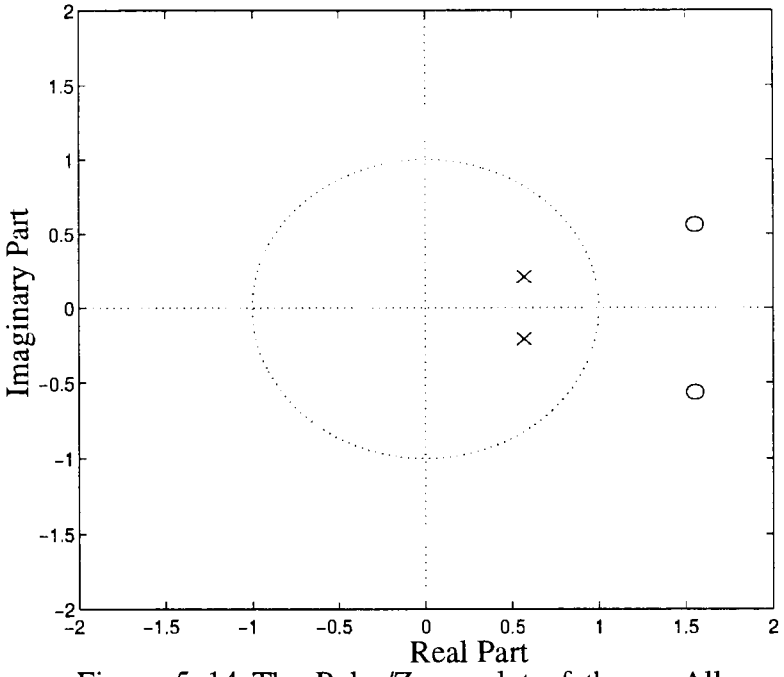

Figure 5–14 The Poles/Zeros plot of the an All–pass filter to equalize 6th–order low–pass Chebyshev filter.

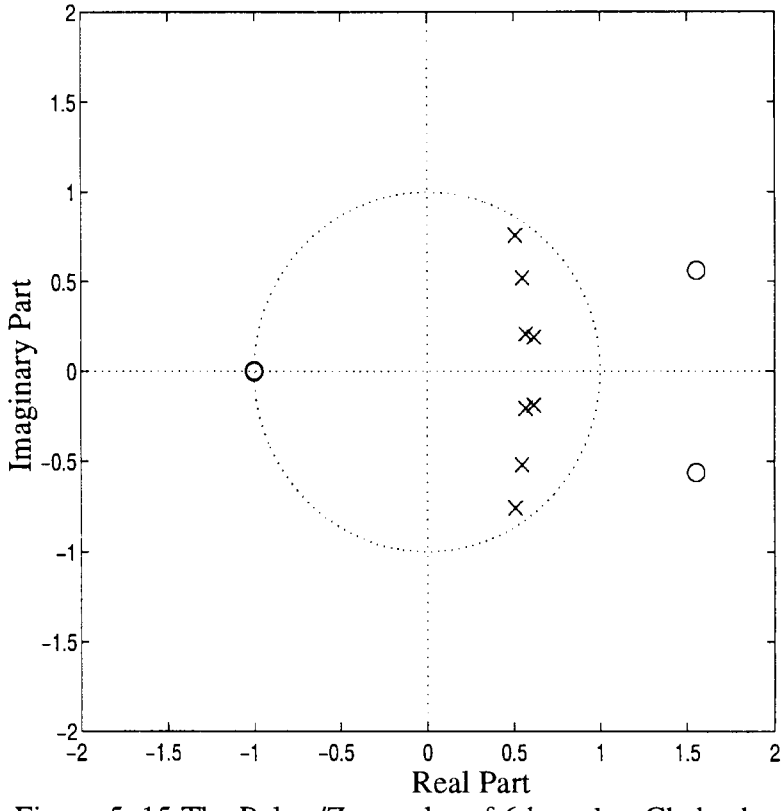

Figure 5–15 The Poles /Zeros plot of 6th–order Chebyshev filter and All–pass filter after equalization.

5.4 Equalization Of a Fullband Linear Delay All–Pass Filter

In this example we consider the equalization of a fullband linear delay all–pass filter discussed in [4]. The desired phase response is:

$P_T(\omega) = (8 - n)\omega - 8\omega^2/\pi$, i .e . , the group delay of $\tau(\omega) = n - 8 + 16\omega/\pi$ is expected to be equalized. An 80th–order Chirp filter group delay was equalized: $\delta_1 = 0.01$, Maximum variation or Ripple in the Pass–Band. $0 \leq \omega_p \leq 0.4\pi$.

$\delta_2 = 0.001$, Maximum variation or Ripple in the Stop–Band. $0.6\pi \leq \omega_s \leq \pi$.

Using a all–pass equalizer , index 2q=2, sampling points L=15, and initial set of poles(in polar coordinates using normalized angles): $X_0 = [0.2, 0.12, 0.0]$. The converged after 5 minutes of CPU time ,with an error criterion $\epsilon = 1.0 * e^{-03}$, The Solution is :

$X_{min} = [0.5938, -0.3448, 9.9993]$, Minimum point reached.

The Solution is shown in the following graphs in Figures. (5–16) thru (5–17).

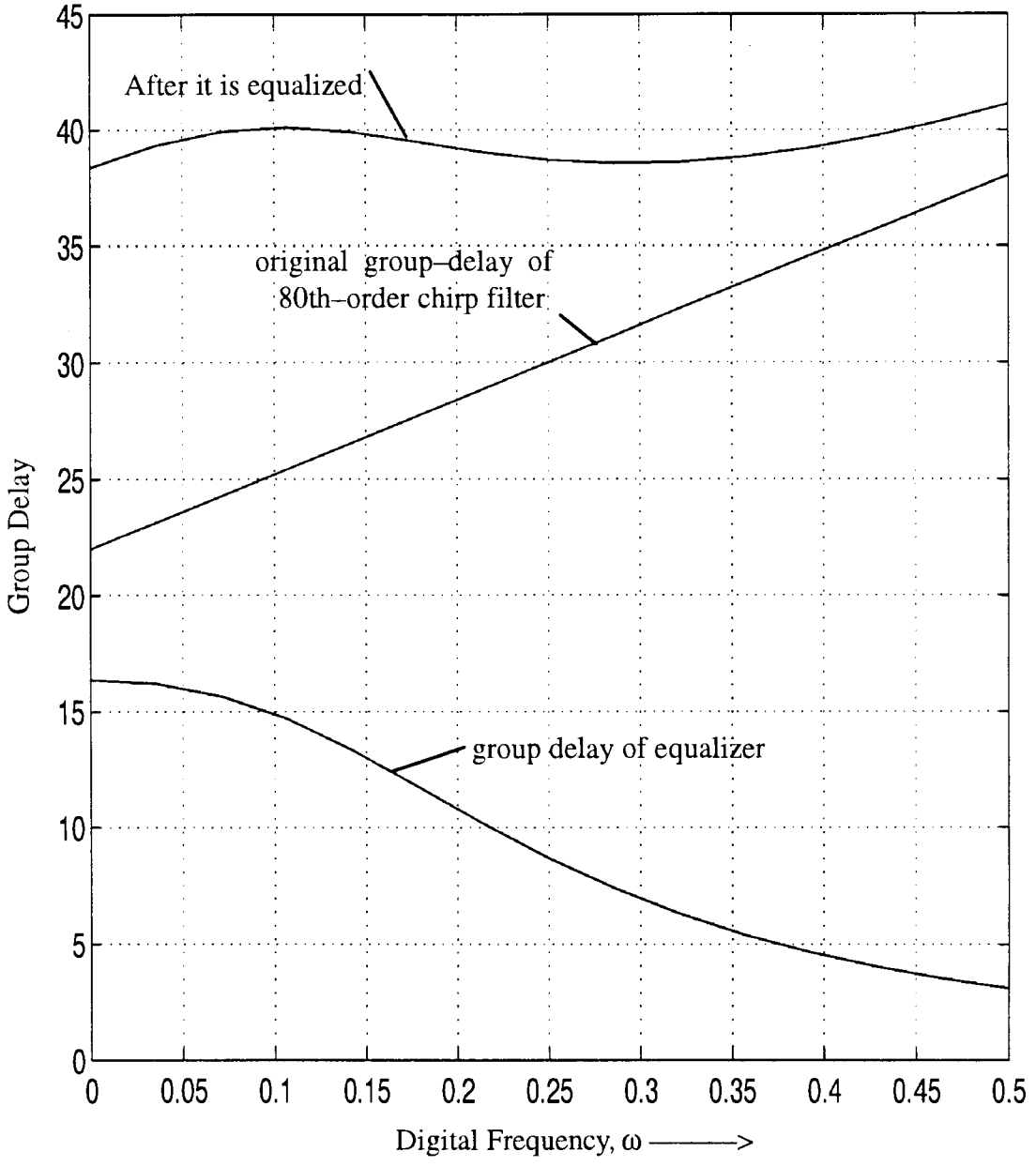

Figure 5–16 Group delay the fullband 80th–order chirp filter and the equalized group delay.

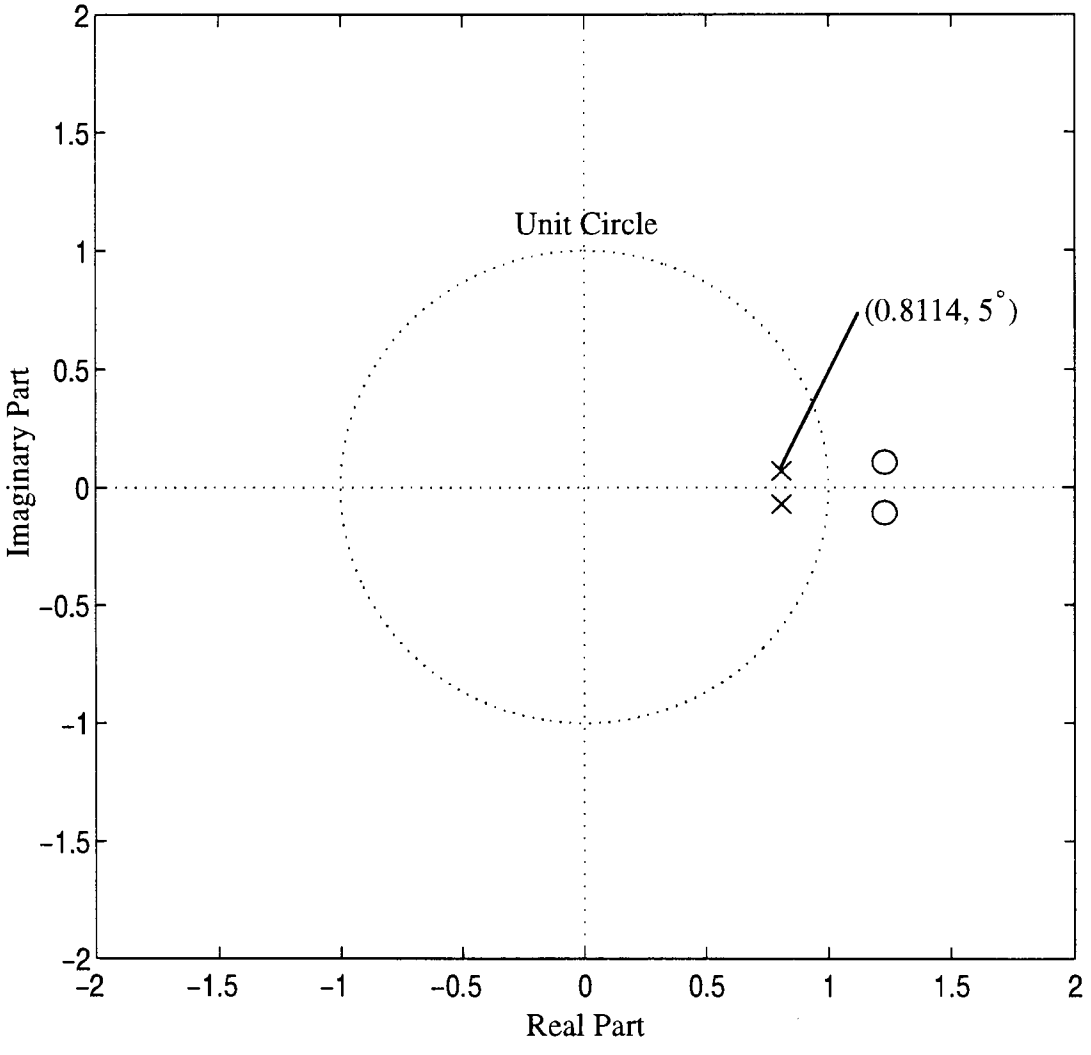

Figure 5–17 Poles–Zeros plot of all–pass filter that equalize an 80th–order fullband linear delay chirp filter.

5.5 Equalization Of a 7th–Order Butterworth Filter

In this example we consider the equalization of a 7th–order Chebyshev filter from the given design specifications:

$\delta_1 = 0.01$, Maximum variation or Ripple in the Pass–Band. $0 \leq \omega_p \leq 0.3\pi$.

$\delta_2 = 0.001$, Maximum variation or Ripple in the Stop–Band. $0.7\pi \leq \omega_s \leq \pi$.

Using a second–order all–pass equalizer , index 2q=2, sampling points L=25, and initial set of poles for second order all–pass equalizer (in polar coordinates using normalized angles): $X_0 = [0.1, 0.8, 0.2, 0.12, 0.0]$. with an error criterion $\epsilon = 1.0 * e^{-03}$, The minimum point reached after 20 minutes of CPU time :

$$X_{min} = [0.4087, 0.5097, 0.1121, 0.6895, 11.1584],$$

$$H = \begin{bmatrix} 0.7185 & 0.3215 & 1.8839 & 0.5020 & 3.1879 \\ 0.3215 & 0.3818 & -0.3570 & 0.1784 & 2.7704 \\ 1.8839 & -0.3570 & 11.4829 & 1.4239 & 1.7677 \\ 0.5020 & 0.1784 & 1.4239 & 0.3983 & 1.8875 \\ 3.1879 & 2.7704 & 1.7677 & 1.8875 & 21.9513 \end{bmatrix}, \text{ The Hessian Matrix}$$

the Poles/Zeros for the required all–pass filter after equalization:

$P_1 = [0.3932 \pm 0.3242i\,,\ 0.4061 \pm 0.0457i]$,and

$Z_1 = [1.5139 \pm 1.2482i\,,\ 2.4316 \pm 0.2737i]$. The Solution is shown in the following graphs in Figures. (5–18) thru (5–23).

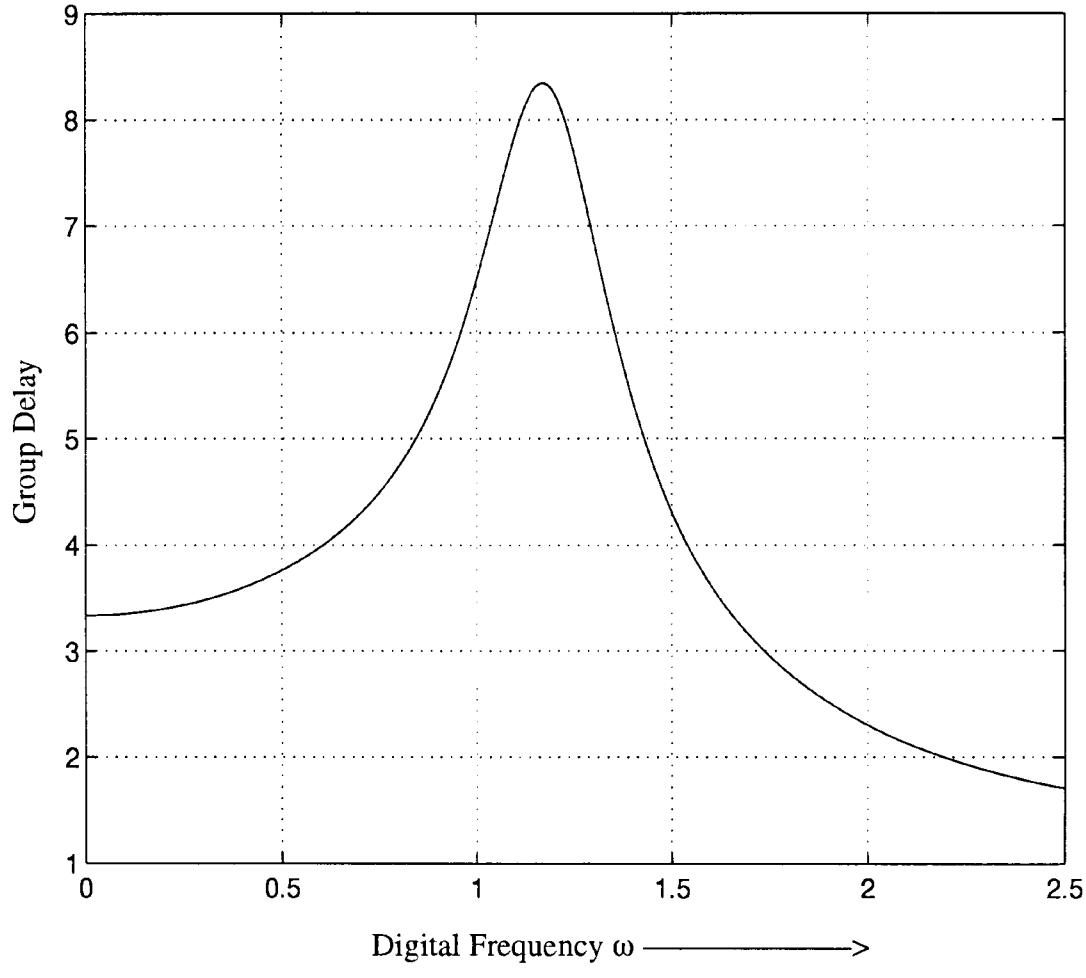

Figure 5–18 The Group delay of a 7th–order Butterworth low–pass IIR filter before equalization

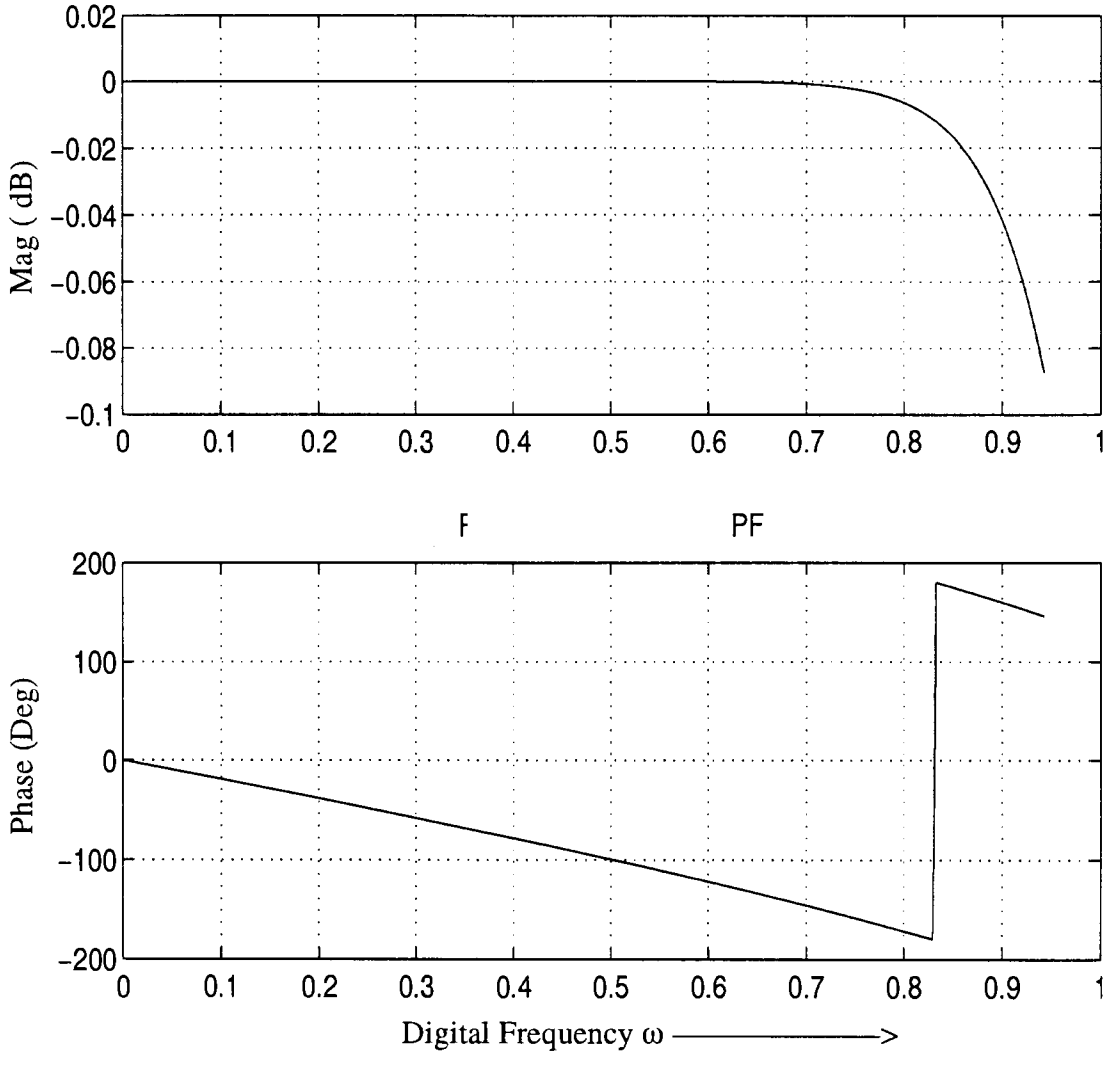

Figure 5–19 Plots of Mag and Phase of the 7th–order Butterworth filter

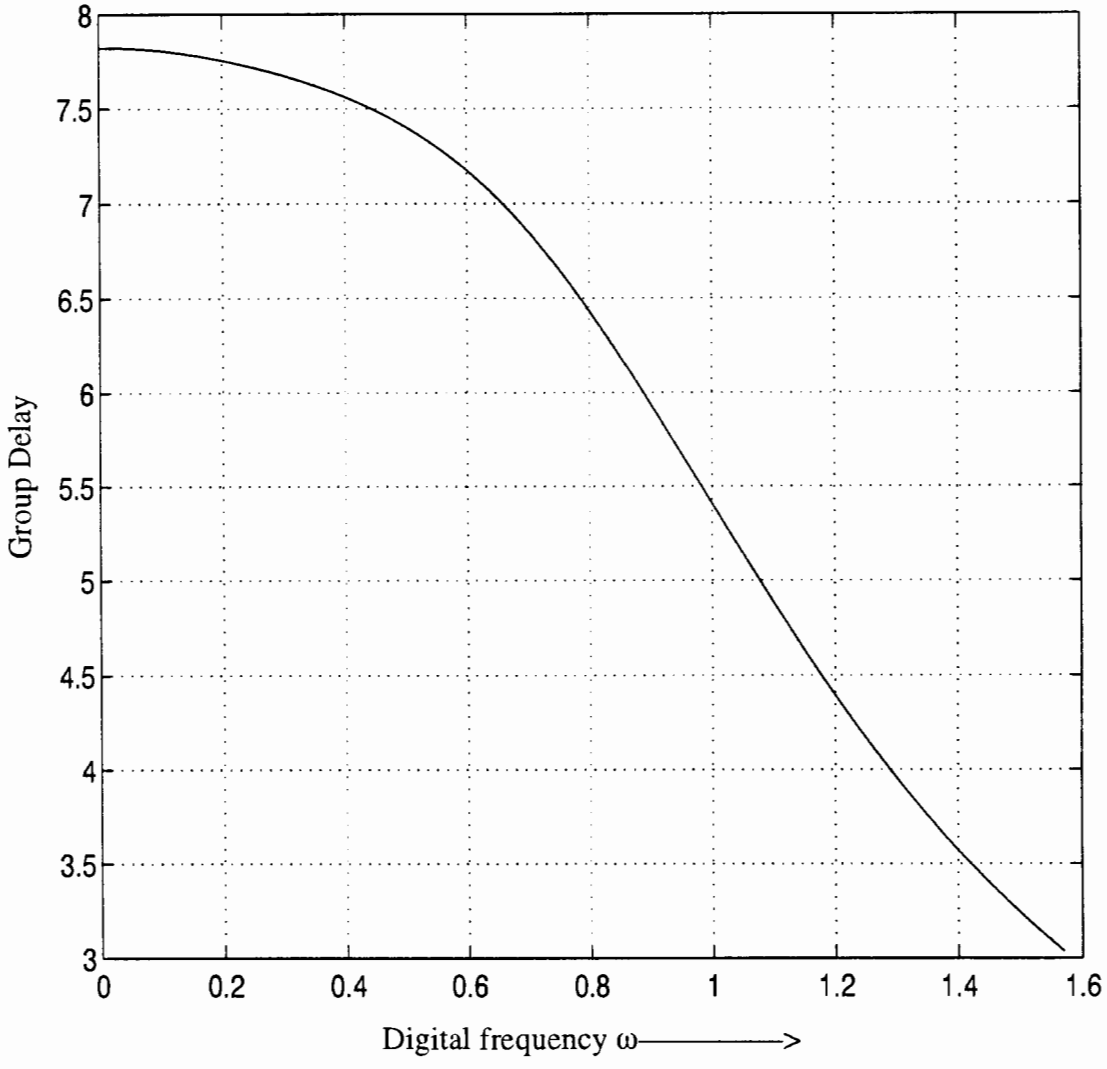

Figure 5–20 Plot of desired group delay of a fourth– order all–pass filter

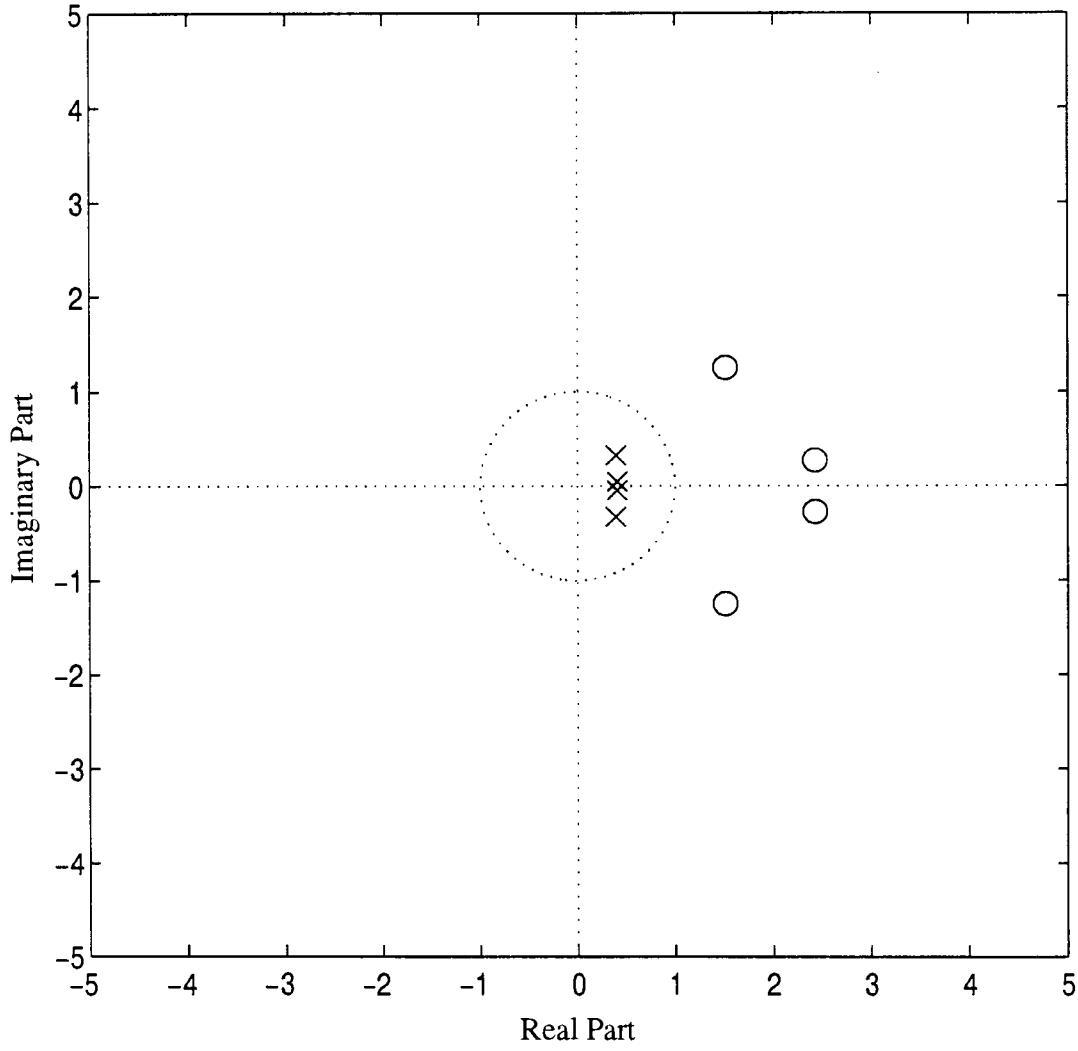

Figure 5–21 Poles/Zeros plots of the desired 2–sections all–pass filter

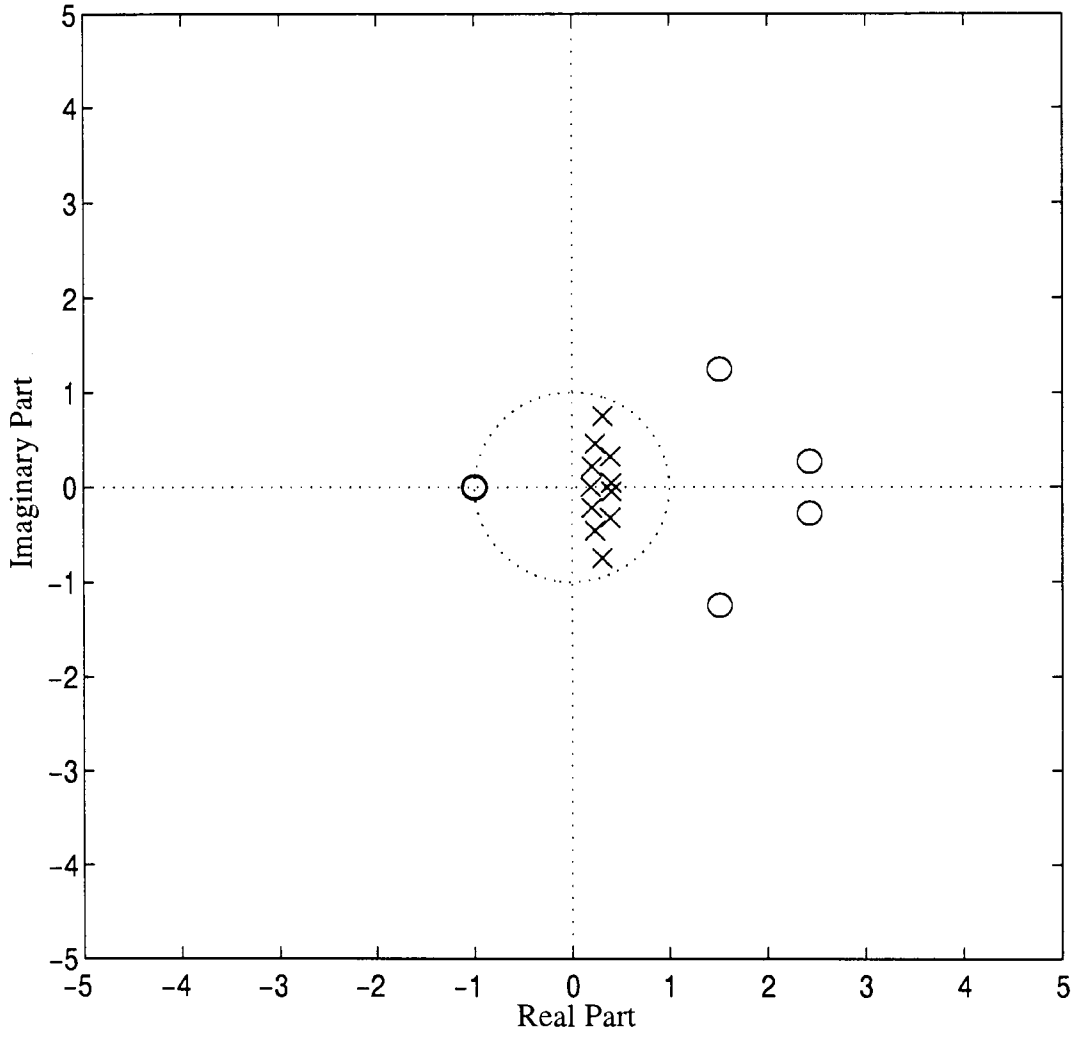

Figure 5–22 Poles/Zeros plots of overall Butterworth filter after it is equalized by a two section all–pass filter.

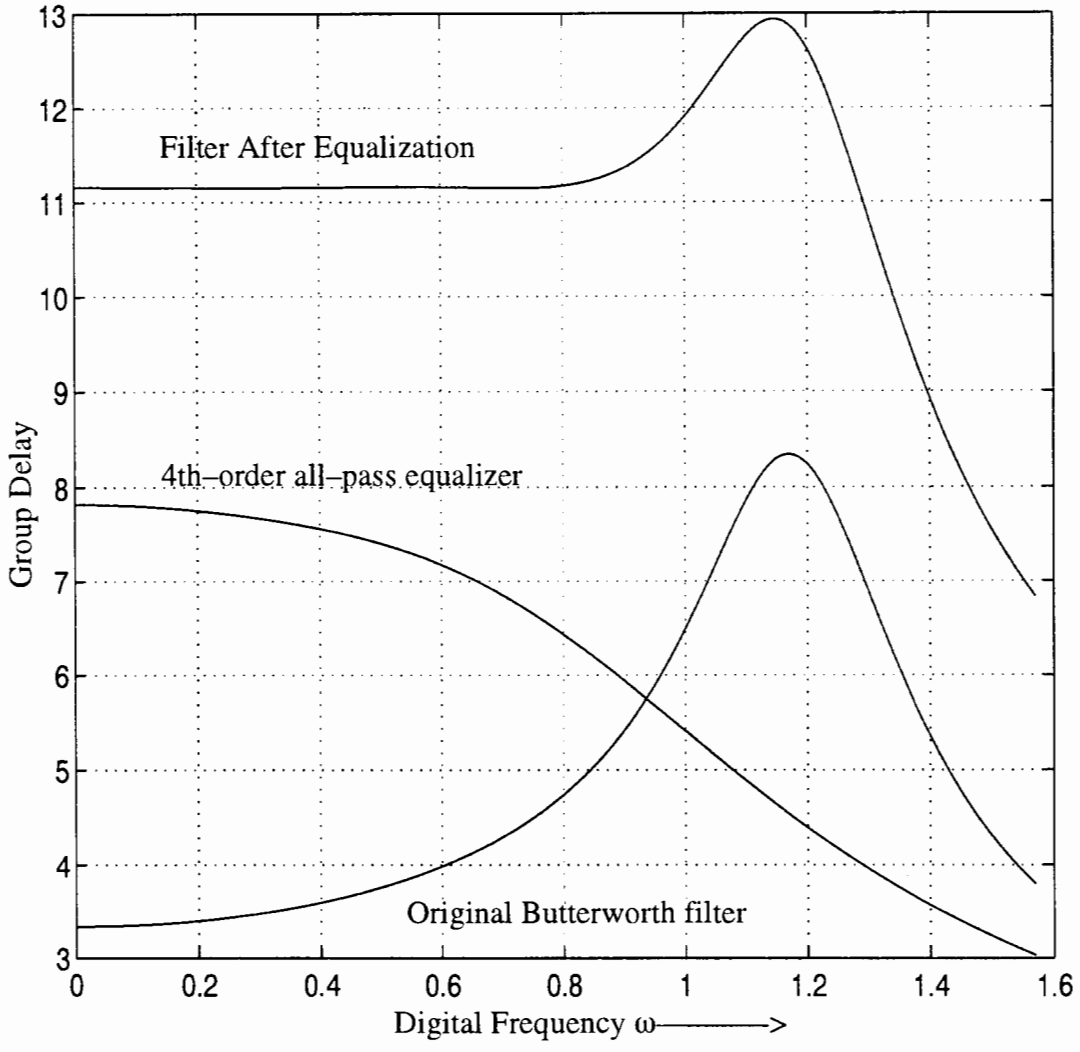

Figure 5–23 Plot of Butterworth low–pass filter after equalization by a two–section all–pass filter.

CHAPTER VI

CONCLUSION AND FUTURE WORK

6.1 Conclusion

A rapidly convergent descent method for minimization has been developed which based on a procedure described by Davidon (1959). The powerful iterative descent method by Fletcher and Powell discussed for finding a local minimum of a function of several variables applied to various equalizations of different types of filters. The type of filter examples show clearly that the type of method given by Davidon is considerably superior to other methods previously available. The simple form of the algorithm makes it easier to write a program and seems not to hinder the speed of convergence. still needs some improvements from local to global search method. The equalization of IIR filters is very important in Digital Signal Processing as far as non–linear phase filters are concerned that distort the signals that they pass through. The use of an all–pass filter is a good thing for designers because of its property of only effecting the phase of filters.

6.2 Future improvements

There is many ways to improve Fletcher and Powell method by modifying it from being a local search to a global search method , in this way , we do not have to worry about the choice of a feasible initial point. I think the most powerful improvement would be the elimination of the use derivatives in calculating gradients, because we may encounter places where derivatives are not defined. In fact, there are some methods that do not use derivatives, but not for global minimization.

REFERENCES

[1] Brent, R.P. 1973, *Algorithms for Minimization without Derivatives* (Englewood Cliffs, Nj: Prentice–Hall), Chapter 7.

[2a] Acton, F.S.1970, *Numerical Methods That Work*, 1990, corrected edition (Was hington: Mathematical Association of America), pp. 464–467.

[2b] Acton, F.S.1970, *Numerical Methods That Work*, 1990, corrected edition (Was hington: Mathematical Association of America), pp. 467–468.

[3] Jacobs, D.A.H. (ed.) 1977, *The State of the Art in Numerical Analysis* (London: Academic Press), pp. 259–262.

[4] Polak, E. 1971, *Computational Methods in Optimization* (New York: Academic Press), §2.3.

[5] Stoer, J., and Bulirsch, R. 1980, *Introduction to Numerical Analysis* (New York: Springer–Verlag), §8.7.

[6] Nelder, J.A., and Mead, R. 1965, *Computer Journal*, vol. 7, pp. 308–313.

[7] Jacoby, S.L.S, Kowalik, J.S., and Pizzo, J.T. 1972, *Iterative Methods for Nonlin ear Optimization Problems* (Englewood Cliffs, NJ: Prentice–Hall).

[8] M.J.D. Powell, "An Efficient method for finding the minimum of a function of several variables without calculating derivatives", *The Computer Journal*, Vol.7, pp. 155–162, July 1964.

[9] Fletcher, R., and Powell, M. J. D., "A Rapidly Convergent Descent Method for Minimization," *Computer J.,* Vol. 6, No. 2, pp. 163–168, 1963.

[10] R. Fletcher and C.M. Reeves, "Function Minimization by conjugate Gradients", The computer Journal, Vol. 7. pp. 149–154, July 1964.

[11] B. Carnahan, H.A. Luther and J.O. Wilkes, *Applied Numerical Methods*, Wiley, New York, 1969.

[12] D.J. Wilde, *Optimum Seeking methods*, Prentice–Hall, Englewood Cliffs, New Jersey, 1964.

[13] A.I. Cohen, *Stepsize Analysis for Descent Methods*, Journal of Optimization Theory and Applications, Vol. 33, pp. 187–205, 1981.

[14] D.A. Pierre, "Search Techniques and nonlinear programming", *Optimization Theory and Applications*, Wiley, New York, 1969.

[15] M.R. Hestenes and E. Stiefel, *Methods of Conjugate Gradients for Solving Linear Systems*, Report No. 1659, National Bureau of Standards, 1952.

[16] W.C. Davidon, *Variable Metric Method of Minimization*, Argonne National Labra tory Report No. ANL–5990, 1959.

[17] B.V. Shah, R.J. Buehler and O. Kempthorne, " Some Algorithms for minimizing a function of several variables", *SIAM journal*, Vol. 12, p. 74, 1964.

[18] A. G. Deczky, : "Synthesis of recursive digital filters using the minimum p–error criterion, " *IEEE Trans. Audio Electroacoustic.*, vol. AU–20,pp. 257–263, 1972.

[19] K. Steiglitz, "Design of FIR digital phase networks," *IEEE Trans. Acoust., Speech, Signal Processing,* vol. ASSP–29, pp. 171–176, 1981.

[20] L. R. Rabiner and B. Gold, *Theory and Applications of Digital Signal Processing.* Englewood Cliffs, NJ: Prentice–Hall, 1975.

[21] R. E. Bogner and A. G. Constantinides, *Introduction to Digital Filtering* (Wiley, New York and London, 1975) chapter 6.

[22] Antoniou, A., *Digital Filters: Analysis and Design,* McGraw–Hill Book Company, New York, 1979 and 1984.

[23] Lam, H. Y.–F., *Analog and Digital filters: Design and Realization,* Prentice–Hall, Englewood Cliffs, NJ, 1979.

[24] O. Herrmann and H.W. Schuessler, "Design of Nonrecursive Digital Filters", *Elec tronic Lett.,* 6 (1970) 329–30.

[25] Shoup, T. E., *Optimization Methods with Applications for Personal Computers*, Prentice–Hall, Inc., Englewood Cliffs, N.J., 1987.

[26] G.E. Forsythe and T.S. Motzkin, "Asymptotic properties of the optimum gradient method (abstract)", *American mathematical society bulletin,* Vol. 57, p. 183, 1951.

[27] H.W. Sorenson, "Comparison of some conjugate direction procedure for function minimization", *Journal of the Franklin Institute,* Vol. 288, p. 421, 1969.