SHORT TITLE

DESIGN OF RECURSIVE TWO-DIMENSIONAL DIGITAL FILTERS

DESIGN METHODS FOR RECURSIVE TWO-DIMENSIONAL DIGITAL FILTERS

by

Eric Dubois, B. Eng. (Honours)

A thesis submitted to the Faculty of Graduate Studies and Research in partial fulfillment of the requirements for the degree of Master of Engineering

Department of Electrical Engineering
McGill University,
Montreal, Canada.

August 1974

ABSTRACT

A technique is presented for the design of stable two-dimensional recursive digital filters. Stability of the resulting filters is guaranteed, eliminating the need for the repeated application of stability tests characteristic of most other mothods. Essentially, the one-dimensional bilinear transformation technique has been extended, where the transformation is applied to the transfer function of a two-variable passive circuit. The method has been applied to the design of lowpass filters whose magnitude characteristics must approximate circularly symmetric specifications.

ABRÈGÉ

On présente une Technique pour créer des filtrès récursifs digitaux stables à deux dimensions. La stabilité des filtres qui en résultent est garantie, ce qui élimine l'application répétitive des tests de stabilité néoessitée dans la plupart des autres méthodes. Essentiellement, c'est la technique de transformation bilinéaire à une dimension qui a été poussée plus loin, au point où la transformation est appliquée à la fonction de transfert d'un circuit passif bi-variable. Cette méthode a été appliquée à la création de filtres passe-basse dont les magnitude caractéristiques doivent s'approcher des spécifications symétriques circulaires.

ACKNOWLEDGEMENTS

The author wishes to thank Dr. M.L. Blostein, research director of this project, for his guidance and suggestions, and Dr. C.C. Paige for providing the optimization routine used in this work.

The financial assistance of the National Research Council is gratefully acknowledged.

A special thanks to Mrs. J. Dubois for her time and effort in producing the typescript.

TABLE OF CONTENTS

CHAPTER I

INTRODUCTION

Techniques for the processing of two-dimensional data have become of great interest to people in the fields of picture processing (2, 38) and geophysics (6, 15, 43). In some cases, this processing can be accomplished by means of coherent optical systems. Often, it is more desirable to perform the processing on sampled data, using a digital computer.

In the area of picture processing, the following types of filters might be required (21): equalizing filters for imaging system aberrations; notch or bandpass filters to remove or enhance systematic line structures; lowpass filters to reduce "snow noise"; highpass filters to remove contrast information while retaining edge information; high emphasis filters to enhance edge information; spatial matched filters to detect certain features. In geophysics, two-dimensional filters are used to process seismic records and potential field data such as the gravity and magnetic surveys used in exploration. In the latter case, it may be desired to separate the field data into various frequency components (6).

The processing is performed by a two-dimensional system, which can be represented as an operator $\mathcal{Q}$, acting on an input $f(x_1,x_2)$ to produce an output $g(x_1,x_2)$.

$$g(x_1,x_2) = \mathcal{Q}(f(x_1,x_2)) \tag{1.1}$$

The variables x_1 and x_2 are generally spatial, such as when $f(x_1,x_2)$

represents a picture. In some applications, we may have one spatial variable and one time variable, as in certain types of seismic records.

The field of two-dimensional digital filter design is relatively new, and, unlike one-dimensional filtering, the theory is largely incomplete. In the area of frequency domain design, which is the subject of the following chapters, very few practical design methods have been reported, and these gloss over some of the basic theoretical questions. A major difficulty is in the application of two-dimensional Chebyshev approximation theory to the problem (17, 37). The uniqueness and characterization properties of the one-dimensional case cannot be extended, even for the case of linear approximating functions, making it very difficult to identify a filter as optimal. Also, if the filter is being designed by means of the minimization of a performance functional, it is not unlikely that only a local minimum will be obtained.

As in one dimension, it may prove to be much more efficient to use recursive filtering in certain applications. However, a further difficulty encountered in the design of recursive two-dimensional digital filters is stability (23, 39). The property of one-variable polynomials of being factorable into first and second order factors does not extend to two variables. Thus, where a design in one dimension can be carried out on a magnitude squared function with no stability constraints, with only left hand plane poles being selected for the final design, this type of approach cannot be used in two dimensions and stability must be accounted for at each step of the design. As the two-dimensional stability test can be quite time consuming, this is a definite liability.

A class of filters where stability is not a question is the case of separable filters, where the processing in the two variables is independent. In this case the operator Q can be expressed as $Q = Q_1 Q_2$, where Q_1 and Q_2 represent one-dimensional systems in the variables x_1 and x_2 respectively. Although the problem of stability no longer presents a serious obstacle, the difficulties associated with two-dimensional approximation must still be dealt with. The design of the optimal separable filter is clearly much simpler than the general problem, but even this has received scant attention in the literature.

Some work has been done in recent years in the area of multi-dimensional circuit theory (3, 26, 32, 34), generally with the application of systems consisting of both lumped and distributed elements in mind. Some of these ideas have been used in this thesis to develop a design technique where a stability test is unnecessary, since the resulting filter is guaranteed to be stable. The method is an extension of the bilinear transformation design technique used in one dimension. Non-linear programming is used to select the parameters of a "two-dimensional circuit" which is analogous to one-dimensional passive lumped circuits, and a two-dimensional bilinear transformation is performed to obtain the digital filter. The basic structure of the thesis is then as follows.

In Chapter II, the basic mathematical structure used in the study of both continuous and discrete two-dimensional systems is presented. A discussion of the stability problems encountered in the design of two-dimensional recursive digital filters is given, along with various tests which have been devised to determine the stability of a filter. The theory of frequency domain design and two-dimensional approximation are discussed in Chapter III and a survey of the design methods reported in the literature is given. A class of filters which are guaranteed to be stable, based on the "two-dimensional circuit" analogy, is then described. Chapter IV outlines a frequency domain design algorithm, which has been termed the circuit analogy method, based on the class of stable filters described earlier, and an example of how the algorithm can be applied to the design of circularly symmetric lowpass filters is detailed. Experimental results are presented in Chapter V.

CHAPTER II

TWO-DIMENSIONAL SIGNAL PROCESSING

2.1 Two-Dimensional Continuous Systems

The theory of two-dimensional continuous systems has been discussed in detail by Papoulis (33), with particular reference to optical systems. This section presents some of the basic formulae which are used in the modelling of two-dimensional signal processing. The systems to be considered are linear, i.e. they satisfy the relation

$$\mathcal{Q}(a_1 f_1(x_1,x_2) + a_2 f_2(x_1,x_2)) = a_1\mathcal{Q}(f_1(x_1,x_2)) + a_2\mathcal{Q}(f_2(x_1,x_2)) \qquad (2.1)$$

The impulse response function of $\mathcal{Q}$ (also called the point spread function) is defined as

$$h(x_1,x_2;x_{10},x_{20}) = \mathcal{Q}(\,\delta(x_1-x_{10})\,\delta(x_2-x_{20})) \qquad (2.2)$$

The output of $\mathcal{Q}$ due to any input $f(x_1,x_2)$ can be determined in terms of h as

$$g(x_1,x_2) = \iint_{-\infty}^{\infty} f(\zeta,\eta)h(x_1,x_2;\zeta,\eta)d\zeta\, d\eta \qquad (2.3)$$

If the response of $\mathcal{Q}$ is independent of the location of the origin in the Cartesian coordinate system (x_1,x_2), the system is said to be shift invariant. Then, $\mathcal{Q}(f(x_1,x_2)) = g(x_1,x_2)$ implies that

$$\mathcal{Q}(f(x_1-\zeta,x_2-\eta)) = g(x_1-\zeta,x_2-\eta) \qquad (2.4)$$

It follows that the impulse response of a shift invariant system can be written as

$$h(x_1,x_2;x_{10},x_{20}) = h(x_1-x_{10},x_2-x_{20}) \tag{2.5}$$

Then, (2.3) becomes

$$\begin{aligned} g(x_1,x_2) &= \iint_{-\infty}^{\infty} f(\zeta,\eta)h(x_1-\zeta,x_2-\eta)d\zeta\, d\eta \\ &= \iint_{-\infty}^{\infty} f(x_1-\zeta,x_2-\eta)h(\zeta,\eta)d\zeta\, d\eta \\ &= f(x_1,x_2)**h(x_1,x_2) \end{aligned} \tag{2.6}$$

The symbol ** represents two-dimensional convolution. The system is said to be separable if $h(x_1,x_2) = h_1(x_1)h_2(x_2)$.

If the response of the system is independent of the orientation of the axes, the system is said to be rotation invariant. Then

$$\begin{aligned} &\mathcal{Q}(f(x_1\cos\theta + x_2\sin\theta, -x_1\sin\theta + x_2\cos\theta)) \\ &= g(x_1\cos\theta + x_2\sin\theta, -x_1\sin\theta + x_2\cos\theta) \quad 0 \le \theta < 2\pi \end{aligned} \tag{2.7}$$

It can be shown that this implies that

$$h(x_1,x_2) = h(x_1^2+x_2^2) \tag{2.8}$$

The two-dimensional Fourier transform of $f(x_1,x_2)$ is defined as

$$F(\omega_1,\omega_2) = \iint_{-\infty}^{\infty} f(x_1,x_2)e^{-j(\omega_1 x_1 + \omega_2 x_2)}\, dx_1 dx_2 \tag{2.9}$$

with the inversion formula

$$f(x_1,x_2) = \frac{1}{4\pi^2} \iint_{-\infty}^{\infty} F(\omega_1,\omega_2) e^{j(\omega_1 x_1 + \omega_2 x_2)} d\omega_1 d\omega_2 \qquad (2.10)$$

The two-dimensional Fourier transform has many properties similar to the conventional Fourier transform, including

$$f_1(x_1,x_2) ** f_2(x_1,x_2) \Leftrightarrow F_1(\omega_1,\omega_2) F_2(\omega_1,\omega_2) \qquad (2.11)$$

Thus, if we apply the input $f(x_1,x_2)$ to system $\mathcal{Q}$ with impulse response $h(x_1,x_2)$, the Fourier transformed output is

$$G(\omega_1,\omega_2) = H(\omega_1,\omega_2) F(\omega_1,\omega_2) \qquad (2.12)$$

where $H(\omega_1,\omega_2)$ is called the system function. If the system is separable, i.e. $h(x_1,x_2) = h_1(x_1)h_2(x_2)$, it follows that $H(\omega_1,\omega_2) = H_1(\omega_1)H_2(\omega_2)$.

Consider a shift invariant, rotation invariant system with impulse response $h(x_1^2 + x_2^2)$. Then

$$\begin{aligned} H(\omega_1,\omega_2) &= \iint_{-\infty}^{\infty} h(x_1^2 + x_2^2) e^{-j(\omega_1 x_1 + \omega_2 x_2)} dx_1 dx_2 \\ &= \int_{-\pi}^{\pi}\int_{0}^{\infty} h(r) e^{-j(\omega_1 \cos\theta + \omega_2 \sin\theta)r}\, r\, dr d\theta \\ &= 2\pi \int_{0}^{\infty} r h(r)\, J_0(\sqrt{\omega_1^2 + \omega_2^2}\, r) dr \\ &= H(\omega_1^2 + \omega_2^2) \end{aligned}$$

Where $J_0(x) = \frac{1}{2\pi} \int_{-\pi}^{\pi} e^{jx\cos(\theta - \alpha)} d\theta$

Similarly, it can be shown that if $H(\omega_1,\omega_2) = H(\omega_1^2 + \omega_2^2)$ then $h(x_1,x_2) = h(x_1^2 + x_2^2)$. These results are summarized by the expression

$$h(x_1^2 + x_2^2) \Leftrightarrow H(\omega_1^2 + \omega_2^2) \qquad (2.13)$$

In words, a circularly symmetric impulse response implies a circularly symmetric system function and vice versa. Thus both the impulse response and system function of a rotation invariant system are circularly symmetric.

The two-dimensional Laplace transform can be defined as

$$F_L(p_1,p_2) = \iint_{-\infty}^{\infty} f(x_1,x_2)e^{-(p_1x_1+p_2x_2)}dx_1dx_2 \tag{2.14}$$

Then

$$F_L(p_1,p_2)\Big|_{\substack{p_1=j\omega_1\\ p_2=j\omega_2}} = F(\omega_1,\omega_2) \tag{2.15}$$

and

$$|F(\omega_1,\omega_2)|^2 = F_L(p_1,p_2)F_L(-p_1,-p_2)\Big|_{\substack{p_1=j\omega_1\\ p_2=j\omega_2}} \tag{2.16}$$

If the input $f(x_1,x_2)$ to a linear, shift invariant system is the sinusoidal signal

$$f(x_1,x_2) = \cos(\omega_1 x_1+\omega_2x_2) = \mathrm{Re}(\exp(j(\omega_1x_1+\omega_2x_2)))$$

then the output is given by (2.6) to be

$$\begin{aligned} g(x_1,x_2) &= \mathrm{Re}\left(\iint_{-\infty}^{\infty}\exp(j(\omega_1(x_1-\zeta)+\omega_2(x_2-\eta)))h(\zeta,\eta)d\zeta d\eta\right) \\ &= \mathrm{Re}\left(\exp(j(\omega_1 x_1+\omega_2x_2))\iint_{-\infty}^{\infty}\exp(-j(\omega_1\zeta+\omega_2\eta))h(\zeta,\eta)d\zeta d\eta\right) \\ &= \mathrm{Re}(\exp(j(\omega_1x_1+\omega_2x_2))\quad H(\omega_1,\omega_2)) \\ &= |H(\omega_1,\omega_2)|\ \cos(\omega_1x_1+\omega_2x_2+\arg(H(\omega_1,\omega_2))) \end{aligned} \tag{2.17}$$

$|H(\omega_1,\omega_2)|$ is referred to as the magnitude response and $\arg(H(\omega_1,\omega_2))$ as the phase response. The phase term has the effect of shifting the surface $f(x_1,x_2)$ by an amount $\arg(H(\omega_1,\omega_2)) / \sqrt{\omega_1^2+\omega_2^2}$ in the direction perpendicular to the line $\omega_1 x_1 + \omega_2 x_2 = 0$. The direction of shift is different for each spectral component of the input, making it impossible to extend the concept of linear phase to two dimensions.

2.2 Two-Dimensional Discrete Systems

As mentioned previously, although the given data may be inherently continuous, it may be advantageous to process it digitally on a computer, and in this case, the signal must be sampled. If the continuous input $f(x_1,x_2)$ with Fourier transform $F(\omega_1,\omega_2)$ is band limited so that $F(\omega_1,\omega_2) = 0$ for $|\omega_1| > W_1$ and $|\omega_2| > W_2$, than it can be shown (12) that the sampling intervals must satisfy $T_1 \leq \frac{1}{2W_1}$ and $T_2 \leq \frac{1}{2W_2}$ to avoid aliasing. Henceforth, it will be assumed that the above requirement is satisfied and, for convenience, that $T_1 = T_2 = T$. The input function is now described by $f(mT,nT)$, $-\infty<m,n<\infty$, where m and n are integer, and will be labelled $f(m,n)$. The discrete system is represented by the operator Q_D, so that for an input $f(m,n)$, the output $g(m,n)$ is given by

$$g(m,n) = Q_D(f(m,n)) \tag{2.18}$$

Again, we deal with linear systems, as defined in (2.1). The unit pulse function is defined as

$$P(m,n) = \begin{cases} 1 & m = n = 0 \\ 0 & \text{elsewhere} \end{cases}$$

Then, the unit pulse response of the system is defined as

$$h(m,n;\, m_0,n_0) = \mathcal{Q}_D(P(m-m_0,n-n_0))$$

Any function f(m,n) can be written

$$f(m,n) = \sum_{k=-\infty}^{\infty} \sum_{\ell=-\infty}^{\infty} f(k,\ell)\, P(m-k,n-\ell)$$

Then $g(m,n) = \mathcal{Q}_D(f(m,n)) = \sum_{k=-\infty}^{\infty} \sum_{\ell=-\infty}^{\infty} f(k,\ell)\, \mathcal{Q}_D(P(m-k,n-\ell))$

$$= \sum_{k=-\infty}^{\infty} \sum_{\ell=-\infty}^{\infty} f(k,\ell)\, h(m,n;\, k,\ell) \tag{2.19}$$

The system is said to be shift invariant if

$$\mathcal{Q}_D(f(m-a,n-b)) = g(m-a,n-b) \tag{2.20}$$

If the system is shift invariant, than the unit pulse response can be written

$$h(m,n;\, m_0,n_0) = h(m-m_0,n-n_0) \tag{2.21}$$

and (2.19) becomes

$$g(m,n) = \sum_{k=-\infty}^{\infty} \sum_{\ell=-\infty}^{\infty} f(k,\ell)\, h(m-k,n-\ell) \tag{2.22}$$

This is the discrete convolution.

It is convenient to define the two-dimensional z-transform for dealing with discrete systems. The two-dimensional z-transform of an array f(m,n) is

$$F(z_1,z_2) = \sum_{m=-\infty}^{\infty} \sum_{n=-\infty}^{\infty} f(m,n) z_1^{\,m} z_2^{\,n} \tag{2.23}$$

Taking the z-transform of g(m,n) as given in (2.22) we obtain

$$G(z_1,z_2)=\sum_{m=-\infty}^{\infty}\sum_{n=-\infty}^{\infty}\left(\sum_{k=-\infty}^{\infty}\sum_{\ell=-\infty}^{\infty} f(k,\ell)h(m-k,n-\ell)\; z_1^{m}z_2^{n}\right)$$

$$=\sum_{k=-\infty}^{\infty}\sum_{\ell=-\infty}^{\infty}\left(\sum_{m=-\infty}^{\infty}\sum_{n=-\infty}^{\infty} h(m-k,n-\ell)z_1^{m-k}z_2^{n-\ell}\right) z_1^{k}z_2^{\ell}f(k,\ell)$$

The inner summation is $H(z_1,z_2)$ for all k,ℓ

$$\therefore\quad G(z_1,z_2) = H(z_1,z_2)\; F(z_1,z_2) \qquad (2.24)$$

Thus, the convolution and z-transform have the same duality for discrete systems as the convolution and Fourier transform have for continuous systems. $H(z_1,z_2)$ will be referred to as the transfer function of the discrete system. It can be seen that if the system is separable, then $h(m,n) = h_1(m)h_2(n)$ and $H(z_1,z_2) = H_1(z_1)H_2(z_2)$.

If $h(m,n) = 0$ for $M_1 \leq m \leq M_2$, $N_1 \leq n \leq N_2$ with $M_1,M_2,N_1,N_2 < \infty$ then $\mathcal{Q}_D$ is referred to as a finite impulse response (FIR) system.

Then, (2.22) becomes

$$g(m,n) = \sum_{k=m-M_2}^{m-M_1}\sum_{\ell=n-N_2}^{n-N_1} f(k,\ell)h(m-k,n-\ell) \qquad (2.25)$$

$$= \sum_{k=M_1}^{M_2}\sum_{\ell=N_1}^{N_2} f(m-k,n-\ell)h(k,\ell)$$

i.e. the output is just a linear combination of input values. If $M_1 = N_1 = 0$, then only values $f(i,j)$ such that $i \leq m$ and $j \leq n$ are used to compute g(m,n) and the system is referred to as causal.

If any of M_1,M_2,N_1, or N_2 are infinite, the system is said to be infinite impulse response (IIR). Again, the system is causal if $M_1 = N_1 = 0$. The most common form of IIR filter is when $H(z_1,z_2)$ is specified as the ratio of two polynomials in z_1 and z_2 .

$$H(z_1,z_2) = \frac{A(z_1,z_2)}{B(z_1,z_2)} \tag{2.26}$$

where $A(z_1,z_2) = (1 \; z_1 \; \ldots \; z_1^{M_A-1}) \; A \begin{bmatrix} 1 \\ z_2 \\ \cdot \\ \cdot \\ \cdot \\ z_2^{N_A-1} \end{bmatrix}$

(2.27)

and $B(z_1,z_2) = (1 \; z_1 \; \ldots \; z_1^{M_B-1}) \; B \begin{bmatrix} 1 \\ z_2 \\ \cdot \\ \cdot \\ \cdot \\ z_2^{N_B-1} \end{bmatrix}$

where A is an M_A by N_A matrix and B is M_B by N_B. For example, the matrix $B = \begin{bmatrix} 1. & .5 & .2 \\ -3. & 1.2 & 1. \\ 1. & .4 & -6. \end{bmatrix}$ corresponds to the polynomial

$$B(z_1,z_2) = (1 \; z_1 z_1^2) \begin{bmatrix} 1. & .5 & .2 \\ -3. & 1.2 & 1. \\ 1. & .4 & -6. \end{bmatrix} \begin{bmatrix} 1 \\ z_2 \\ z_2^2 \end{bmatrix}$$

$$= 1 + .5z_2 + .2z_2^2 - 3z_1 + 1.2z_1z_2 + z_1^2 + .4z_1^2z_2 - 6z_1^2z_2^2 + z_1z_2^2$$

The polynomial $B(z_1,z_2)$ and matrix B will be used interchangeably.

The above transfer function can be realized as a recursive causal filter (39)

$$g(m,n) = \sum_{i=1}^{M_A} \sum_{j=1}^{N_A} a_{ij}' \cdot f(m-i+1,n-j+1) - \sum_{k=1}^{M_B} \sum_{\substack{\ell=1 \\ k \cdot \ell \neq 1}}^{N_B} b_{k\ell}' \, g(m-k+1,n-\ell+1) \qquad (2.28)$$

$b_{k\ell}' = b_{k\ell}/b_{11}$ and $a_{k\ell}' = a_{k\ell}/b_{11}$

In this realization, the output depends on "previous" values of the output as well as "previous" inputs. In general, the number of parameters required to design a recursive filter is much less than the number required for a non-recursive filter with the same specifications, since previous output values contain information about all previous input values. There are three other possible recursive realizations for the same $H(z_1,z_2)$ (23) which recurse in different directions (the filter of (2.28) recurses in the $+m$, $+n$ direction).

Suppose the input to a filter $H(z_1,z_2) = A(z_1,z_2) / B(z_1,z_2)$ is the sampled sinusoid $f(m,n) = \cos(m\omega_1 T_1 + n\omega_2 T_2)$

$$= \mathrm{Re}\,(\exp(j(m\omega_1 T_1 + n\omega_2 T_2))).$$

From (2.22) it is clear that if the response to $f = \mathrm{Re}\, f + j\, \mathrm{Im}\, f$ is g, then the response to $\mathrm{Re}\, f$ is $\mathrm{Re}\, g$. Hence we find the response to

$$f' = \exp(j(m\omega_1 T_1 + n\omega_2 T_2)) \; .$$

$$F'(z_1,z_2) = \sum_{m=0}^{\infty} \sum_{n=0}^{\infty} \exp(j(m\omega_1 T_1 + n\omega_2 T_2))\, z_1^m z_2^n$$

$$= \sum_{m=0}^{\infty} (\exp(j\omega_1 T_1) z_1)^m \sum_{n=0}^{\infty} (\exp(j\omega_2 T_2) z_2)^n$$

$$= \frac{1}{(1-z_1 e^{j\omega_1 T_1})(1-z_2 e^{j\omega_2 T_2})} \tag{2.29}$$

$$G'(z_1,z_2) = \frac{A(z_1,z_2)}{B(z_1,z_2)(1-z_1 e^{j\omega_1 T_1})(1-z_2 e^{j\omega_2 T_2})}$$

$$= \frac{C(z_1,z_2)}{B(z_1,z_2)} + \frac{H(e^{-j\omega_1 T_1}, e^{-j\omega_2 T_2})}{(1-z_1 e^{j\omega_1 T_1})(1-z_2 e^{j\omega_2 T_2})} \tag{2.30}$$

If we assume the filter is stable (see section 2.3) then the response

$$C(z_1,z_2) / B(z_1,z_2) \longrightarrow 0 .$$

The steady state response is then given by

$$G'_{ss}(z_1,z_2) = \frac{H(e^{-j\omega_1 T_1}, e^{-j\omega_2 T_2})}{(1-z_1 e^{j\omega_1 T_1})(1-z_2 e^{j\omega_2 T_2})}$$

$$= H(e^{-j\omega_1 T_1}, e^{-j\omega_2 T_2})\, F(z_1,z_2) .$$

$$g'(m,n) = H(e^{-j\omega_1 T_1}, e^{-j\omega_2 T_2})\, e^{j(m\omega_1 T_1 + n\omega_2 T_2)} \tag{2.31}$$

and $g(m,n) = \left| H(e^{-j\omega_1 T_1}, e^{-j\omega_2 T_2}) \right| \cos(m\omega_1 T_1 + n\omega_2 T_2 +$

$$\arg H(e^{-j\omega_1 T_1}, e^{-j\omega_2 T_2})) .$$

$H(e^{-j\omega_1 T}, e^{-j\omega_2 T})$, which is referred to as the frequency response of the filter, is periodic in both the ω_1 and ω_2 directions as in Figure 2.1. Contours of $|H(e^{-j\omega_1 T}, e^{j\omega_2 T})|$ are shown.

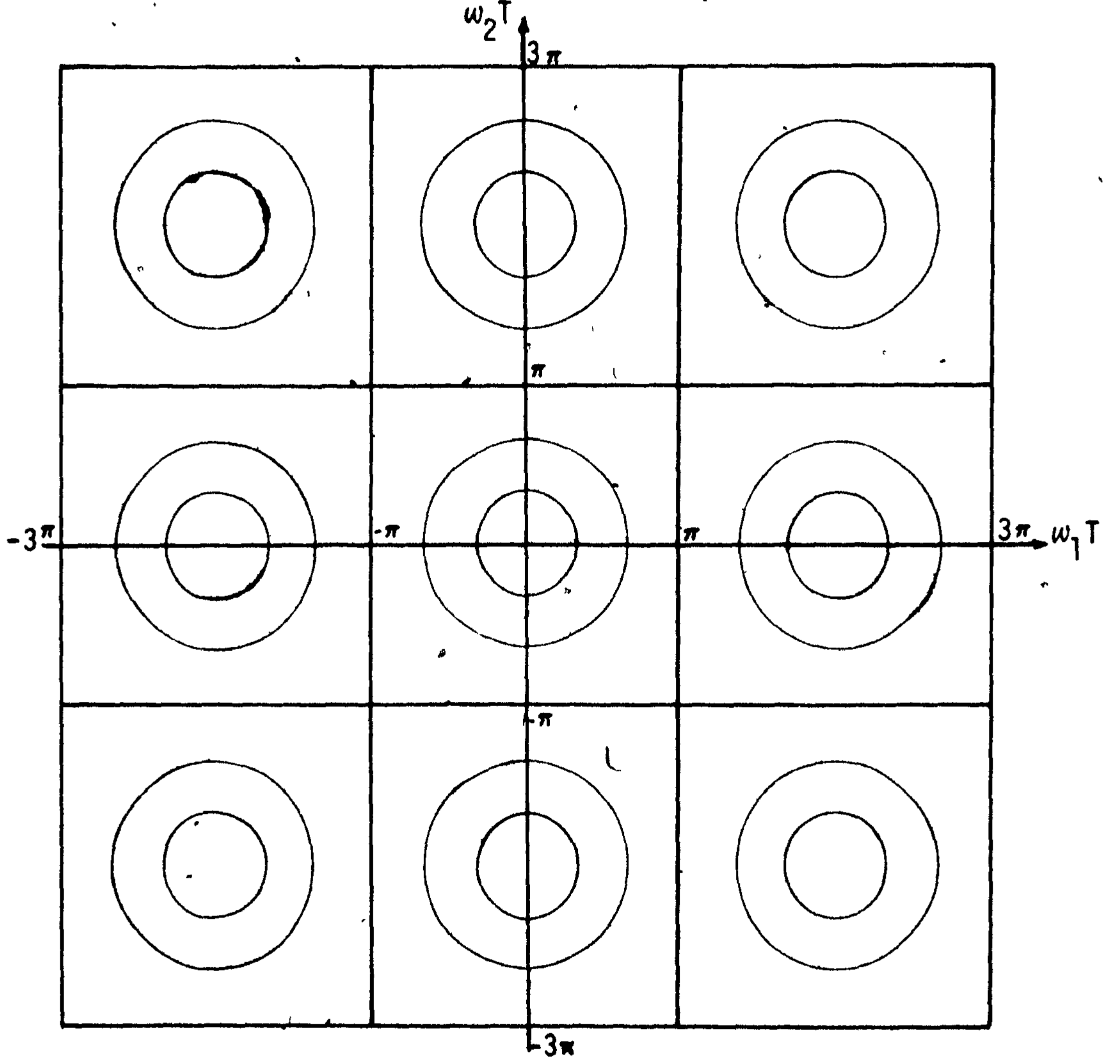


FIGURE 2.1

2.3 Stability Criteria for Two-Dimensional Filters

One of the difficulties in the extension of one-dimensional design techniques to two dimensions is the question of stability. In the one-dimensional case, it is simply required that all poles of the transfer function be outside the unit circle $|z| = 1$. For example, if a magnitude squared function is designed, reciprocal poles will occur inside and outside the unit circle, each contributing the same amount to the response. Hence those poles outside the unit circle can be chosen, giving a stable filter with the desired response. Such methods cannot be used in the two-dimensional case, as will be seen from the following discussion.

A causal, two-dimensional digital filter with transfer function

$$H(z_1,z_2) = \sum_{i=1}^{\infty} \sum_{j=1}^{\infty} h_{ij} z_1^{\,i-1} z_2^{\,j-1}$$

is said to be bounded-input bounded-output (BIBO) stable if and only if for any bounded input, the output is bounded. It can be shown by a trivial extension of a theorem in (11) that $H(z_1,z_2)$ is BIBO stable if and only if there exists a finite k such that

$$\sum_{i=1}^{\infty} \sum_{j=1}^{\infty} |h_{ij}| \leq k < \infty \tag{2.32}$$

It is clear that the problem of stability does not exist for FIR filters, since in this case there exists only a finite number of terms in (2.32). For recursive filters this is not true and stability is a consideration. The following theorem gives the conditions under which (2.32) holds for a recursive filter with transfer function

$$H(z_1,z_2) = \frac{A(z_1,z_2)}{B(z_1,z_2)} = \frac{\sum_{i=1}^{M_A} \sum_{j=1}^{N_A} a_{ij} z_1^{i-1} z_2^{j-1}}{\sum_{i=1}^{M_B} \sum_{j=1}^{N_B} b_{ij} z_1^{i-1} z_2^{j-1}} \quad (2.33)$$

Theorem 2.1 (39): A causal, recursive filter with transfer function $H(z_1,z_2) = A(z_1,z_2) / B(z_1,z_2)$, where A and B are polynomials in z_1 and z_2, is BIBO stable if and only if there exists no values of z_1 and z_2 such that $B(z_1,z_2) = 0$ for $|z_1| \leq 1$ and $|z_2| \leq 1$ simultaneously.

Although any one-variable polynomial can be factored into a product of second order terms, no such factorization exists for a general two-variable polynomial. Furthermore, as no methods currently exist for finding the continuum of zeroes of a polynomial in two variables, the above theorem is difficult to test directly. The maximum-modulus theorem can be invoked to give a simplified test procedure due to Huang.

Theorem 2.2 (23): A causal, recursive filter $H(z_1,z_2) = A(z_1,z_2) / B(z_1,z_2)$ is BIBO stable if and only if:

1) the map of $\partial d_1 = \{z_1 : |z_1| = 1\}$ to the z_2 plane by $B(z_1,z_2) = 0$ lies outside $d_2 = \{z_2 : |z_2| \leq 1\}$;

2) no point in $d_1 = \{z_1 : |z_1| \leq 1\}$ is mapped to $z_2 = 0$ by $B(z_1,z_2) = 0$.

The test procedure is then to solve $b(e^{j\phi},z_2) = 0$ for $0 \leq \phi < 2\pi$ and see that no roots exist with $|z_2| \leq 1$. Also, no roots of $B(z_1,0)$

must exist with $|z_1| \leq 1$, which can be checked by Jury's method (25). This procedure is still infinite in the sense that in condition 1, $B(e^{j\phi},z_2) = 0$ must be solved for all values of ϕ on $(0,2\pi)$. A procedure by which condition 1 of theorem 2.2 can be tested in a finite number of steps has been given in (1). The test involves the construction of the Schur-Cohn matrix C, which is an M_A+1 by M_A+1 matrix whose elements are of the form $\sum_i c_i \cos(i\phi)$. Condition 1 will hold if C is negative definite for all ϕ, i.e. if the leading principal minors of C have certain signs. This latter condition could be verified by a series of Sturm tests. Calculating determinants of polynomial matrices can become quite time consuming and the addition of the Sturm tests make the stability check become quickly infeasible, as the order increases, even on a computer. This is especially true if a design algorithm requires the stability check to be performed repeatedly.

Alternatively, theorem 2.2 can be framed in a form suitable for Hurwitz type testing. A bilinear transformation is applied to both z_1 and z_2.

$$p_1 = \frac{1-z_1}{1+z_1} \qquad p_2 = \frac{1-z_2}{1+z_2} \tag{2.34}$$

The bilinear transformation $p = (1-z)/(1+z)$ maps the region $|z| \leq 1$ to $\mathrm{Re}(p) \geq 0$ and $z = 0$ to $p = 1$. Using (2.34), $H(z_1,z_2)$ can be written

$$H(z_1,z_2) = H'(p_1,p_2) = \frac{A'(p_1,p_2)}{B'(p_1,p_2)} \tag{2.35}$$

and the conditions of theorem 2.2 can be given in terms of $B'(p_1,p_2)$.

<u>Theorem 2.3 (23)</u>: A causal, recursive filter $H'(p_1,p_2) = A'(p_1,p_2) / B'(p_1,p_2)$ is BIBO stable if and only if:

1) $B'(j\omega,p_2)$ has no zeroes in $Re(p_2) \geq 0$ for all finite ω;

2) $B'(p_1,1)$ has no zeroes in $Re(p_1) \geq 0$.

Condition 2 can be checked by a Hurwitz test and condition 1 can be tested using a Hermite test followed by a series of Sturm tests. Both forms of the stability test require considerable computation.

In general, a two-variable polynomial $B(z_1,z_2)$ cannot be factored into a product of stable and unstable parts, from which a stable filter with the desired characteristics can be derived. Hence methods based on the ability to factor one-variable polynomials cannot be extended, and most design techniques must directly incorporate one of the stability tests discussed previously.

CHAPTER III

FREQUENCY DOMAIN DESIGN OF TWO-DIMENSIONAL DIGITAL FILTERS

3.1 The Approximation Problem

In the design of a recursive filter $H(z_1,z_2) = A(z_1,z_2) / B(z_1,z_2)$, A and B must be chosen so that B is stable and H performs some desired filtering operation. In frequency domain design, some function of $H(e^{-j\omega_1 T},e^{-j\omega_2 T})$ must be made to approximate in some specified way an ideal response $\hat{H}(\omega_1,\omega_2)$. The functions usually dealt with are magnitude and phase, and the desired function here will be labelled $H_1(x,\omega)$, where x represents the parameters to be varied (e.g. elements of A and B) and $\omega = (\omega_1,\omega_2)$. The set of values over which ω ranges is W, which in theory is the continuous set $\{\omega_1,\omega_2 : -\pi/T \leq \omega_1,\omega_2 \leq \pi/T\}$, but for practical implementations will be a finite point set.

A strategy often used is to choose x to minimize the L_p norm of $r(x,\omega) = \hat{H}(\omega) - H_1(x,\omega)$, such that H remains stable. If S_x represents the set of all x such that H is stable, the optimum choice of x, denoted by x_p, is defined by

$$\|r(x_p,\omega)\|_p = \inf_{x \in S_x} \|r(x,\omega)\|_p = \inf_{x \in S_x} \left(\int_W |r(x,\omega)|^p \, d\omega \right)^{1/p} \quad (3.1)$$

If W is a finite point set, the integral is replaced by a summation. The limiting case of the L_p norm as $p \rightarrow \infty$ is the L_∞ or Chebyshev norm, and in this case x_∞ is defined by

$$\|r(x_\infty,\omega)\|_\infty = \inf_{x \in S_x} \max_{\omega \in W} |r(x,\omega)| \quad (3.2)$$

Chebyshev approximation is usually desired as it minimizes the maximum deviation of the approximating function from the ideal. The problem defined by (3.2) will be termed P1.

The case of linear Chebyshev approximation, where $H_1(x,\omega) = \sum_{i=1}^{n} x_i\phi_i(\omega)$, has been discussed in the literature (37, 17). It is found that many properties of one-dimensional Chebyshev approximation cannot be extended to the two-variable case. Rice (37) has shown that the lack of non-trivial Chebyshev sets $\phi_i(\omega)$ in two dimensions leads to the lack of a uniqueness property. In fact, there may be an infinite number of optimal approximations x_∞ yielding the same minimum norm $\| r(x_\infty,\omega)\|_\infty$. Furthermore, there is no effective characterization of the best Chebyshev approximation, as in the one-dimensional case where the error curve alternates n+1 times. Thus, methods based on the characterization of the error curve, such as the second method of Remez, cannot be used for two-dimensional approximation. Also, an attempt to use gradient techniques to minimize $\|r(x\ ,\omega)\|_\infty$ as a function of x may break down because the gradient will in general be non-zero and discontinuous at the optimum.

A technique for obtaining the Chebyshev approximation without reference to characterization is the Pólya algorithm, which states that if the Chebyshev approximation is unique, then for any sequence $\{x_{p_k} : p_k \to \infty \text{ as } k \to \infty\}$, $\lim_{k\to\infty} x_{p_k} = x_\infty$. For approximation on a finite point set, then $\lim x_{p_k} = \hat{x}_\infty$, the strict approximation (37). Although the Chebyshev approximation may not be unique, the strict approximation, described by Rice as "the best of the best", is unique. No similar result has been established for approximation on a continuum. Fletcher et.al. (17) describe a technique for extrapolating several L_p approximations to obtain the L_∞ approximation. This requires much lower values of p for equivalent accuracy than if one just uses the L_p approximation with a large value of p.

The above comments are valid only for linear approximation, and can probably be extended to rational approximation. However, for the case of general non-linear approximation, little can be said, and these statements can only serve as plausibility arguments as to what may be expected in the more general case.

For many applications, $\hat{H}(\omega)$ will have the form

$$\hat{H}(\omega) = \begin{cases} 1 & \omega \in P \\ 0 & \omega \in S \end{cases}$$

A transition region T may exist where $\hat{H}(\omega)$ is not defined. This arises in the design of Pass-Stop filters, and a different approach is generally required. The problem can be stated as a constrained optimization, namely to minimize

$$\max_{\omega \in S} \left| H_1(x,\omega) \right|$$

subject to the constraint

$$1-\epsilon_1 \leq \left| H_1(x,\omega) \right| \leq 1+\epsilon_2 \qquad \omega \in P$$

This constrains the response to lie within a certain passband tolerance, and obtains the best stopband performance for that tolerance. This problem will be termed P2.

The following section discusses techniques which have been proposed for the solution of P1 and P2.

3.2 Frequency Domain Design Methods

As mentioned in section 2.3, stability is not a consideration in the design of FIR filters and thus one dimensional design methods can be extended in a straight forward manner. Hu and Rabiner (22) have used linear programming to solve a problem similar to P2. The technique is to minimize δ subject to the constraints

$$1-\alpha\delta \le H_1(x,\omega) \le 1+\alpha\delta \qquad \omega \in P$$

$$-\delta \le H_1(x,\omega) \le \delta \qquad \omega \in S$$

The parameters x represent the DFT coefficients of the filter in the transition band T, and appear linearly in $H_1(x,\omega)$, allowing linear programming to be used. Although the actual passband tolerance is not specified as in P2, the free parameter α can be used to adjust the ratio of maximum passband deviation to maximum stopband deviation.

In recursive filtering, stability becomes one of the major considerations in any design technique. As a simplistic first solution, Shanks et.al. (39) proposed a method whereby a stable one-dimensional filter $F_2(p_2)$ is rotated by an angle β via the transformation

$$p_1 = p_1' \cos\beta + p_2' \sin\beta$$

$$p_2 = p_2' \cos\beta - p_1' \sin\beta$$

The bilinear transformation (2.34) is applied to the resulting filter $F'(p_1', p_2')$ to obtain the equivalent digital filter $F(z_1, z_2)$. This technique yields filters with a strong directionality, with no filtering at all along the axis at angle β from the original.

The above method was adapted by Costa and Venetsanopoulos (14) to obtain lowpass filters with essentially circularly symmetric magnitude characteristics. Several filters designed by the method of Shanks et.al. but with different angles of rotation β are cascaded. By spacing the angles equally on 0^0 to 360^0, the resulting characteristic can be circularly symmetric to a very good approximation. The authors show that due to stability considerations, only $-90^0 \leq \beta \leq 0^0$ is allowed, and thus certain transformations must be performed on the data to avoid stability problems. Although the resulting characteristics possess good circular symmetry, they are not steep. For example, a 12th order filter designed by this method satisfies the same pass-stop specification as a 4th order filter designed by nonlinear programming (Chapter 4).

Maria and Fahmy (29) have used an L_p approach, working with a cascade of 2nd and 4th order sections, thus limiting the size of the stability test required

$$H(z_1, z_2) = \frac{\prod_{i=1}^{k_1} A_i(z_1, z_2)}{\prod_{i=1}^{k_2} B_i(z_1, z_2)}$$

The coefficients of the polynomials A_i and B_i are chosen to minimize the L_p norm of $\hat{H}(\omega) - \left|H(e^{-j\omega_1 T}, e^{-j\omega_2 T})\right|$ on a finite point set. The value of p used by the authors varies between 2 and 10. Minimization of the performance functional is accomplished using Newton's method (Appendix A). At each iteration, a Newton step is taken and stability of each of the B_i is checked. If unstable, the incremental change in the coefficients of the unstable sections is successively reduced by half until stability is achieved. Convergence is assumed if a stationary point of the functional is found or if the step size is reduced below a certain level. Although the error due to quantization and finite word length may be smaller for cascade realizations than for direct realization, a general two-variable polynomial cannot be factored as a product of 2nd and 4th order sections. Thus, the optimal cascade filter of a given order may be far worse than the optimal general filter of that order.

A commonly used technique for rational L_∞ approximation is the differential correction algorithm (5). Dudgeon (16) has demonstrated the use of this algorithm for one-dimensional recursive filter design, and has shown how it can be extended to the two-dimensional case. His method, however, does not take stability into account and once a solution has been obtained, the discrete Hilbert transform is used to obtain a stable filter (35). Such stabilization algorithms modify the frequency response causing significant degradation and defeat the entire method.

Bednar and Farmer (7) have adapted the differential correction algorithm for the solution of P1 in a way which accounts for the stability problem. The structure of the algorithm is as follows:

define

$$\Delta(x) = \max_{\omega} \left| \hat{H}(\omega) - H_1(x,\omega) \right| = \| r(x,\omega) \|_{\infty}$$

with $$\Delta^* = \| r(x_{\infty},\omega) \|_{\infty}$$

Choose a starting point x^0 such that $|x^0{}_i| < 1$ and B is stable. Then for each $k \geq 1$ define the auxiliary function

$$\delta_k(x) = \max(\left| \hat{H}(\omega)B(x,\omega) - A(x,\omega) \right| - \Delta(x^k)\left| B(x,\omega) \right|)$$

and select x^{k+1} to minimize this function in the cube $|x_i| < 1$. If $B(x^{k+1},\omega)$ is unstable then x^k is a solution i.e. $x^k = x_{\infty}$, and $B(x^k,\omega)$ is stable. Otherwise $\Delta(x^k)$ decreases monotonically to Δ^* as $k \to \infty$. Hence, each iteration requires an optimization of a complicated function ($\delta_k(x)$ is the Chebyshev norm of some function and may have to be approximated by an L_p norm for computational purposes) along with a stability test, and thus this method could be prohibitively time consuming.

The following section describes a class of filters where stability is guaranteed, making possible a design algorithm not requiring repeated stability tests.

3.3 A "Two-Dimensional Circuit" Analogy

A method commonly used for one-dimensional recursive digital filter design is the bilinear transformation method. A bilinear transformation is applied to a continuous filter transfer function with the desired characteristics to obtain the digital filter transfer function.

The rationale for this technique is that a considerable body of knowledge has been built up in one-dimensional continuous filter theory, and hence it is desirable to exploit this knowledge. Although little work has been done on two-dimensional frequency domain design, there has been much work done recently in developing a two-dimensional circuit theory. Bearing in mind that it would be very desirable to have a class of network functions which are guaranteed to be stable, thus avoiding repeated stability tests, the following "two-dimensional circuit" analogy has been developed. Once a design is obtained in the continuous domain, a two-dimensional bilinear transformation is performed to obtain the two-dimensional recursive digital filter. The method is analogous to the one-dimensional case, but the underlying motivation is different (guaranteed stability rather than previous experience).

Definition (26): A finite, passive network of two variables is a network composed of finite numbers of two-terminal elements whose impedances are proportional to p_1, $1/p_1$, p_2 and $1/p_2$ with positive coefficients, positive resistors, ideal transformers, and ideal gyrators. Figure 3.1 gives an example of such a network

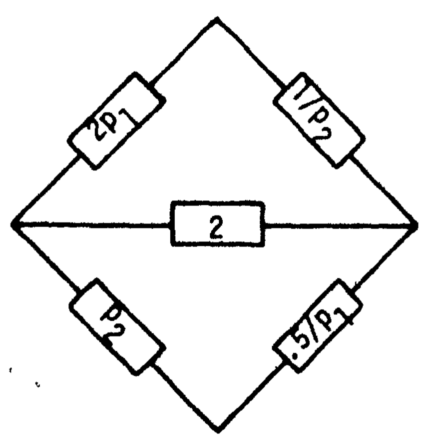


FIGURE 3.1

Given a passive, two-variable n-port network N, the usual network functions such as driving point impedance and voltage transfer can be defined. Given such a function $H'(p_1,p_2)$, the transfer function of a digital filter $H(z_1,z_2)$ can be obtained by means of the bilinear transformation (2.34). The following statement can be made about the resulting digital filter.

<u>Assertion</u>: The digital filter with transfer function $H(z_1,z_2)$ obtained by performing a bilinear transformation on a network function $H'(p_1,p_2)$ of a two-variable passive network is marginally stable.

<u>Proof</u>: From theorem 2.3, $H(z_1,z_2)$ is BIBO stable if and only if

1) $H'(j\omega,p_2)$ has no poles in $\mathrm{Re}(p_2) \geq 0$ for all positive ω;

2) $H'(p_1,1)$ has no poles in $\mathrm{Re}(p_1) \geq 0$.

$H'(j\omega,p_2)$ represents the corresponding network function of a one-dimensional passive filter with imaginary elements, and thus has no poles in $\mathrm{Re}(p_2) > 0$ (8). Similarly $H'(p_1,1)$ has no poles in $\mathrm{Re}(p_1) > 0$. Hence only marginal instability can occur, namely if $H'(j\omega,p_2)$ or $H'(p_1,1)$ has j-axis poles. This can generally be avoided by choosing N to be a lossy network.

Two-variable networks have been used in the study of networks consisting of both lumped and distributed elements (26, 34). Koga has shown (26) that an arbitrarily prescribed n by n positive real matrix of two variables is realizable as the impedance or admittance matrix of a finite passive n-port of two variables. However, it is not known if all stable transfer functions can be obtained as a network function of some two-variable circuit. This is a much more difficult problem, and its solution would indicate whether or not the class of transfer functions obtained in this way is restrictive.

The previous discussion can shed light on the method of Shanks discussed in section 3.2. Suppose $F_2(p_2)$ has a passive network realization. Then $F'(p_1',p_2')$ is obtained by replacing each inductor L with the series connection shown in Figure 3.2 (a) and each capacitor C with the parallel connection shown in Figure 3.2 (b). The resulting filter will be stable if all the new elements are positive i.e. $\cos\beta \geq 0$ and $\sin\beta \geq 0$ or $-90^0 \leq \beta \leq 0^0$. This agrees with the result of Costa and Venetsanopoulos (14).

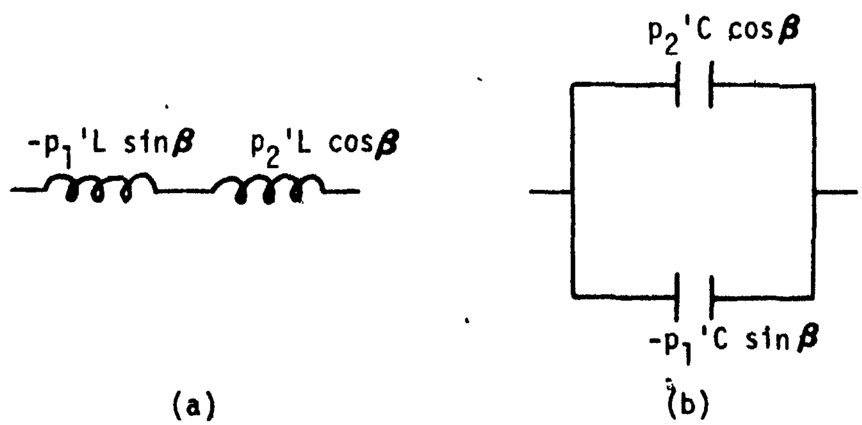


FIGURE 3.2

CHAPTER IV

THE CIRCUIT ANALOGY DESIGN TECHNIQUE

4.1 The Design Procedure

This section describes how the two-dimensional circuit analogy of section 3.3 can be applied to the design of stable two-dimensional recursive digital filters. It is assumed that the configuration of a two-variable passive network having a response of the general form desired can be obtained. This network should possess assymptotic behaviour (i.e. as $\omega \rightarrow 0$ and $\omega \rightarrow \infty$) compatible with the specification. Experience with the frequency behaviour of one-dimensional circuits would be extremely valuable in making this initial choice. The values of the circuit elements and transformer and gyrator parameters form the vector of variables x to be adjusted in the design procedure. The response of the chosen network is denoted by $H_c(x,p)$, where $p = (p_1,p_2)$.

The Chebyshev design will be carried out on a discrete set of frequency points W chosen on $-\pi/T \leq \omega_1,\omega_2 \leq \pi/T$. The number of points required depends on the order of the filter and must be sufficiently large to adequately represent the circuit response.

The desired digital filter $H(z_1,z_2)$ is obtained from $H_c(x,p)$ by means of the bilinear transformation (2.34). It is not necessary however to actually perform the transformation to get $H(e^{-j\omega_1 T},e^{-j\omega_2 T})$ from $H_c(x,j\omega)$ if it is noted that

$$p = \frac{1-z}{1+z} = \frac{1 - e^{-j\omega T}}{1 + e^{-j\omega T}} = \frac{j\ \sin\omega T}{1+\cos\omega T}$$

Using a standard trigonometric identity this becomes

$$p = j\tan\frac{\omega T}{2}$$

and the relationship between H and H_c is

$$H(e^{-j\omega_1 T}, e^{-j\omega_2 T}) = H_c(j\tan\frac{\omega_1 T}{2}, j\tan\frac{\omega_2 T}{2})$$

$$= H_c(j\omega_{c1}, j\omega_{c2}) \tag{4.1}$$

Thus, given a point $(\omega_1, \omega_2) \in W$, the response at that point is the response of the continuous filter at $(\tan\frac{\omega_1 T}{2}, \tan\frac{\omega_2 T}{2})$.

As in section 3.1 $H_1(x,\omega)$ is the function of $H(e^{-j\omega_1 T}, e^{-j\omega_2 T})$ which must approximate the ideal response $\hat{H}(\omega)$, and $r(x,\omega) = \hat{H}(\omega) - H_1(x,\omega)$. The solution of P1 is that x which minimizes the Chebyshev norm of $r(x,\omega)$. To allow more flexibility, a weighting function $u(\omega)$ is introduced, so that the problem is now to minimize the Chebyshev norm of $r'(x,\omega) = r(x,\omega)u(\omega)$. Then

$$\|r'(x_\infty,\omega)\|_\infty = \inf_{x \in S} \max_{\omega \in W} |r'(x,\omega)| \tag{4.2}$$

The solution to P1 is obtained when $u(\omega) = 1$.

From the Pólya algorithm it is known that on the finite point set W, x_p converges to the strict approximation as $p \to \infty$, for linear approximation. With this as motivation, the L_p norm with a large value of p is used, rather than the Chebyshev norm, although the problem is nonlinear. The optimal L_p approximation x_p is given by

$$\|r'(x_p,\omega)\|_p = \inf_{x \in S} \|r'(x,\omega)\|_p = \inf_{x \in S} \left(\sum_{\omega \in W} |r'(x,\omega)|^p\right)^{1/p} \tag{4.3}$$

x_p is obtained by using nonlinear programming to minimize

$$J(x) = \Big(\sum_{\omega \in W} |r'(x,\omega)|^p \Big)^{1/p} \tag{4.4}$$

If one of the standard derivative methods is to be used, the gradient $\nabla J(x)$ is required. Assuming p is an even integer,

$$\nabla J(x) = \Big(\sum_{\omega \in W} (r'(x,\omega))^p \Big)^{1/p-1} \Big(\sum_{\omega \in W} (r'(x,\omega))^{p-1} \nabla r'(x,\omega) \Big) \tag{4.5}$$

where

$$\begin{aligned} \nabla r'(x,\omega) &= \nabla (\hat{H}(\omega) - H_1(x,\omega))u(\omega) \\ &= - u(\omega) \nabla H_1(x,\omega) \end{aligned} \tag{4.6}$$

A case of particular interest is when $H_1(x,\omega) = |H(e^{-j\omega_1 T}, e^{-j\omega_2 T})|$, the magnitude response. Assuming $\nabla H(e^{-j\omega_1 T}, e^{-j\omega_2 T})$ is available, $\nabla H_1(x,\omega)$ can be obtained in the following way.

$$H = |H| \, e^{j\theta_H}$$

$$\frac{\partial H}{\partial x_i} = \frac{\partial |H|}{\partial x_i} e^{j\theta_H} - j \, |H| \, e^{j\theta_H} \frac{\partial \theta_H}{\partial x_i}$$

$$\frac{1}{H} \frac{\partial H}{\partial x_i} = \frac{1}{|H|} \frac{\partial |H|}{\partial x_i} - j \frac{\partial \theta_H}{\partial x_i}$$

Since $|H|$ and θ_H are real, it is clear that

$$\frac{1}{|H|} \frac{\partial |H|}{\partial x_i} = \mathrm{Re} \frac{1}{H} \frac{\partial H}{\partial x_i}$$

and in vector form

$$\nabla |H| = |H| \quad \mathrm{Re}\Big(\frac{1}{H} \nabla H\Big) \tag{4.7}$$

$H_c(x,p)$ must be available explicitly in symbolic form in order to compute $H(z_1,z_2)$ via the bilinear transformation. Thus H and ∇H can be calculated directly from this explicit form. However, methods of computer aided network analysis can be invoked to calculate H, and ∇H, which represents sensitivities with respect to network parameters, can be obtained with little extra computation (9).

In the approach in which the coefficients of A and B make up the parameter vector x, the set S_x is very complicated, being defined by the stability conditions of section 2.3. When applying the method of this section, S_x becomes a very simple set, in which the only requirement is $x_i > 0$. Although constrained optimization methods can be used, if the optimum lies in the non-feasible region, a constrained method would yield a solution with $x_i = 0$ for some of the x_i. Such a solution would indicate that the circuit chosen is not really suitable for the desired application. Thus unconstrained gradient methods, such as the Quasi-Newton methods outlined in Appendix A, can be used to minimize J(x).

The following section gives an example of how this method can be applied for a lowpass filter design with circular symmetry.

4.2 Example for Lowpass Filter Design

As an example of a pass-stop characteristic often encountered in two-dimensional filter design, the lowpass filter with circular symmetry will be considered. As discussed in section 2.1, a circularly symmetric frequency response implies that the filtering does not depend on the relative orientation of the data, a condition which is usually desirable.

The specification $\hat{H}(\omega)$ which we will try to approximate by $|H(e^{-j\omega_1 T}, e^{-j\omega_2 T})|$ is shown in Figure 4.1. It must of course be kept in mind that the actual response will not be exactly circularly symmetric.

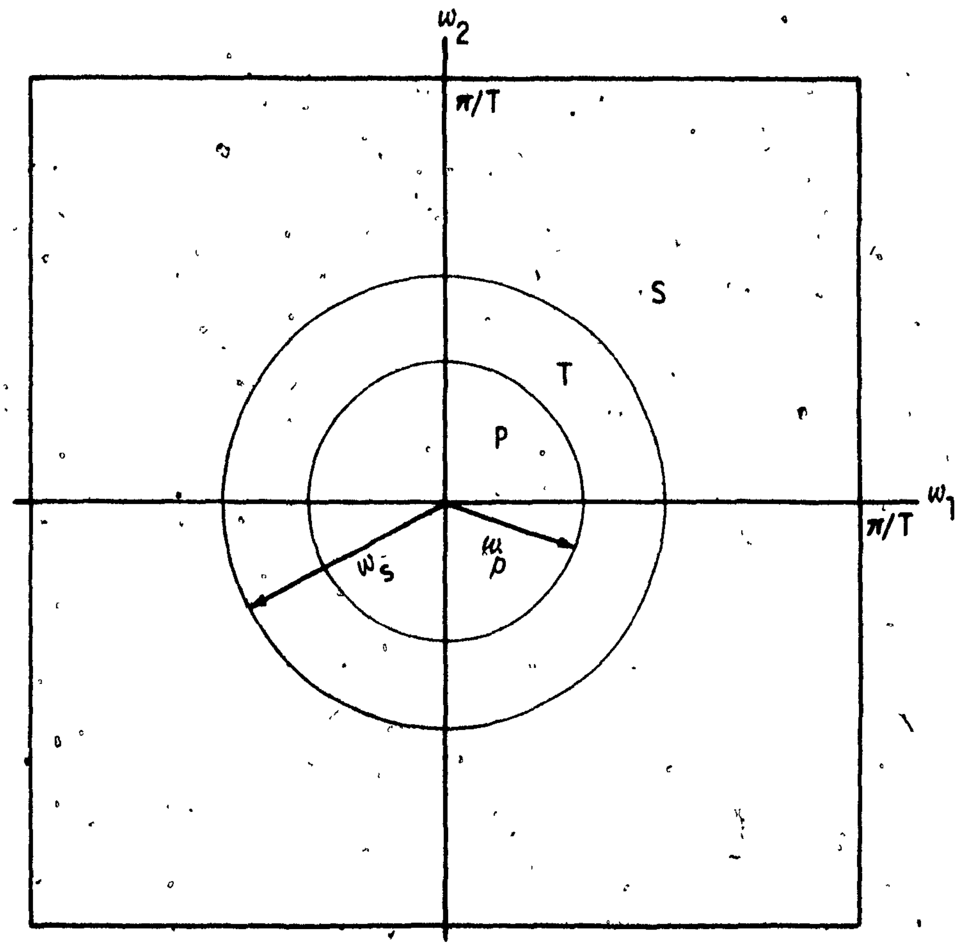


FIGURE 4.1

$$\hat{H}(\omega) = \begin{cases} 1 & \omega \in P = \{\omega_1, \omega_2 : \omega_1^2 + \omega_2^2 \leq \omega_p^2\} \\ 0 & \omega \in S = \{\omega_1, \omega_2 : \omega_1^2 + \omega_2^2 \geq \omega_s^2\} \end{cases} \tag{4.8}$$

The design procedure of section 4.1 can be modified to obtain a solution to either P1 or P2.

Drawing from one-dimensional filter theory, it is known that the L-C ladder has the properties of a lowpass filter. Thus, the configuration of Figure 4.2 is postulated for use in designing a two-dimensional lowpass filter.

Referring to Figure 4.2, the vector x is given by $x = (L_1, C_2, L_3, \ldots, L_{n-2}, C_{n-1}, G_n)^T$, and $i_k = 1$ or 2 for k = 1, n-1. The i_k associate one of the frequency variables p_1 or p_2 with each circuit element.

The transfer function $H_c(x,p)$ can be obtained by any of the standard methods of circuit analysis. Johnson (24) gives a simple arithmetic procedure which involves the evaluation of the determinant of a nearly diagonal matrix of order (n-1)/2 and the product of the shunt impedances. Once $H_c(x,p)$ is available, $H(z_1,z_2)$ is obtained via the bilinear transformation (2.34). (10) presents an algorithm to perform an n-dimensional bilinear transformation.

Many possible choices of the i_k can be immediately ruled out as impractical. Since $\hat{H}(\omega)$ is symmetric about $\omega_1 = \omega_2$, $H_c(x,p)$ should be of the same order in p_1 and p_2, and so there should be an equal number of $i_k = 1$ and $i_k = 2$. As another example, if all series arms have $i_k = 1$ and all shunt arms have $i_k = 2$, there will be no filtering at all along $\omega_1 = 0$ and $\omega_2 = 0$. The merits of other possibilities are discussed in Chapter V.

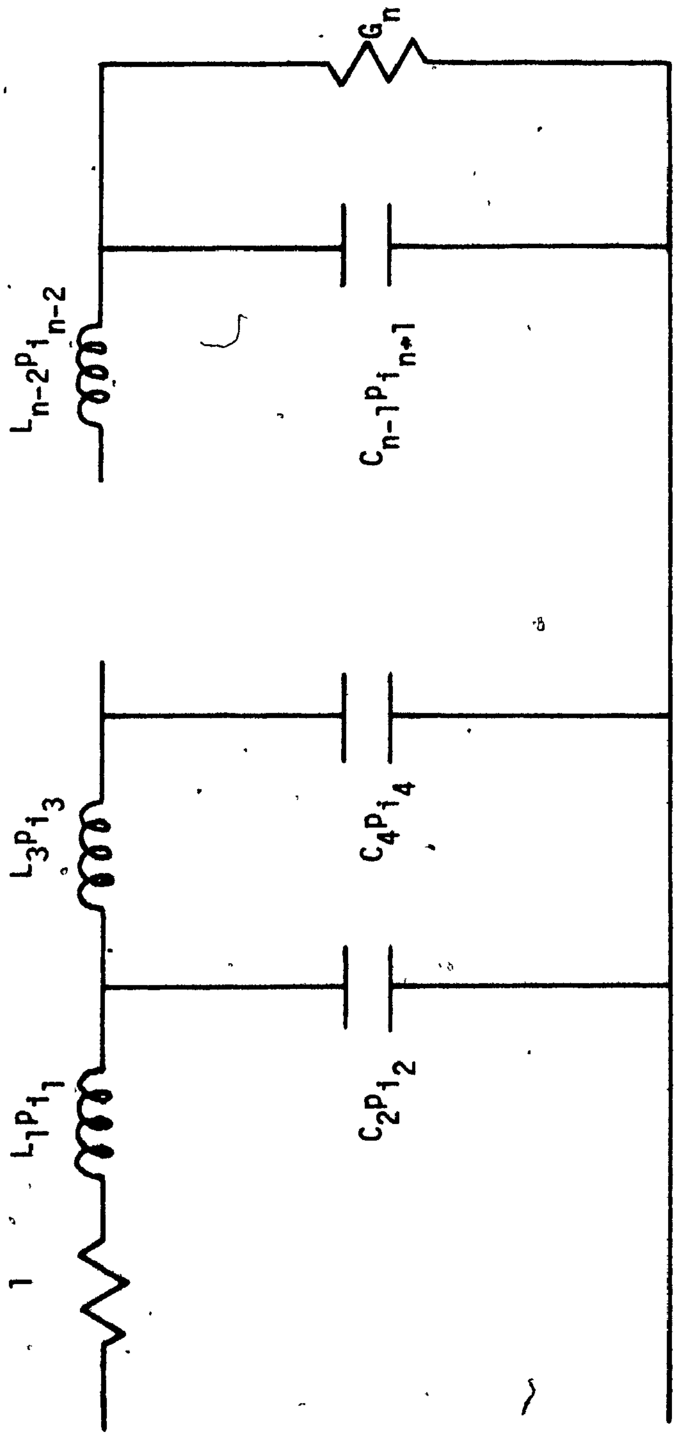


FIGURE 4.2

Since the specification $\hat{H}(\omega)$ is circularly symmetric, it is convenient to choose the points of W on radii $\omega_1^2 + \omega_2^2 = \omega_i^2$. Also, $H(\omega_1,\omega_2) = H(-\omega_1,-\omega_2)$, and thus points need only be in the region $-\pi/T \leq \omega_2 \leq \pi/T$, $0 \leq \omega_1 \leq \pi/T$. The response of the circuit of Figure 4.2 decreases monotonically in the stopband, and hence only one radial at $\omega = \omega_s$ is required to monitor stopband performance. In the passband, it is generally found that the response changes more rapidly near the band edge and thus the frequency radii should be denser near ω_p. A convenient formula for choosing these radii, taken from one-dimensional filter design, is

$$\omega_i = -\omega_p \cos \frac{\pi}{2} (1-\frac{i}{n_p}) \quad i = 1,2,\ldots n_p \tag{4.9}$$

where n_p is the number of radials desired. (4.9) relates to the pole locations of an all pole Chebyshev filter in the one-dimensional case, but is simply a convenient device here. The number of radials and the number of points per radial should be large enough to adequately represent the response surface. n_p should be comparable to the filter order, and the spacing of points on a radial should be about equal to the corresponding spacing between radials.

To evaluate $|H(e^{-j\omega_1 T},e^{-j\omega_2 T})|$ and $\nabla H(e^{-j\omega_1 T},e^{-j\omega_2 T})|$ at the points of W, we need to evaluate $|H_c(j\omega_{c1},j\omega_{c2})|$ and $\nabla H_c(j\omega_{c1},j\omega_{c2})|$ at the points of W_c, as in (4.1). H_c and ∇H_c can be obtained for the ladder structure quite simply in the following way. The general ladder structure of NS stages is shown in Figure 4.3.

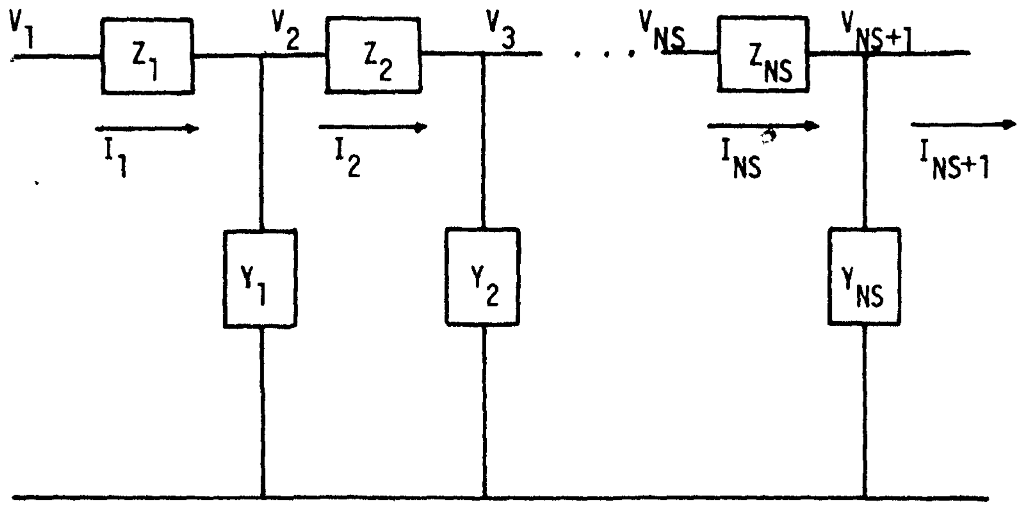


FIGURE 4.3

Using the recursive relations $V_i = V_{i+1} + Z_i I_i$ and $I_i = Y_i V_{i+1} + I_{i+1}$, with conditions $V_{NS+1} = 1$, $I_{NS+1} = 0$, the currents and voltages I_i and V_i can be found. The transfer function H_c is given by $1/V_1$, and the currents and voltages due to a unit voltage input $V_1 = 1$ are V_i/V_1 and I_i/I_1. These are denoted by V_{fi} and I_{fi}. Using the adjoint network (9) ∇H_c is obtained as follows. The voltages and currents, V_{ri} and I_{ri} due to a unit current $I_{NS+1} = 1$ are obtained in a similar manner. Then, it can be shown that $\partial H_c/\partial Z_i = -I_{fi} I_{ri}$ and $\partial H_c/\partial Y_i = V_{fi} V_{ri}$. Using this, ∇H_c can be calculated using the form of the Z_i and Y_i, and (4.7).

The objective function (4.4) can be modified to increase speed of computation and improve accuracy. The resulting function is similar to one used by Bandler and Charalambous (4). A function $\xi(w)$ is selected, and then only those points in W are chosen such that $|r(x,w)| > \xi(w)$.

If a solution to P1 is desired, then $\xi(\omega)$ is a constant, set initially to an estimate of $\|r(x_{\infty},\omega)\|_{\infty}$, and $u(\omega) = 1$. The set W' is defined as

$$W' = \{\omega \in W : |r(x,\omega)| > \xi(\omega)\}$$

and the modified performance functional is

$$J'(x) = (\sum_{\omega \in W'} ((|r(x,\omega)| - \xi(\omega))u(\omega))^p)^{1/p}$$

$$= (\sum_{\omega \in W'} (d(x,\omega))^p)^{1/p} \qquad (4.10)$$

To reduce the problem of ill conditioning in the evaluation of $d(x,\omega)^p$ for large p, the following equivalent performance functional is used.

$$J'(x) = M(\sum_{\omega \in W'} (\frac{d(x,\omega)}{M})^p)^{1/p} \qquad (4.11)$$

where

$$M = \max_{\omega \in W'} d(x,\omega) \qquad (4.12)$$

If at some stage of the optimization procedure M becomes negative, then $\xi(\omega)$ can be reduced by some factor and the procedure restarted.

If a solution to P2 is desired, the following approach can be used. For points in the passband

$$\xi(\omega) = \epsilon \qquad \omega \in P,$$

where ϵ represents the maximum passband ripple desired. $\xi(\omega)$ will remain fixed in the passband throughout the procedure. $\xi(\omega)$ in the stopband is a constant and set to an estimate of the maximum stopband ripple for the given passband ripple. Of course if the estimate proves excessive, it can be reduced during the optimization. By weighting points in the passband much more heavily than those in the stopband, the passband ripple is forced to be near ϵ, and the stopband ripple is then minimized for that ϵ.

The circuit of Figure 4.2 is an all pole network. Noting that the numerator of a recursive filter cannot introduce instability, the transfer function of the circuit can be multiplied by the term

$$A(\omega_1,\omega_2) = \sum_{m=1}^{M_A} \sum_{n=1}^{N_A} a_{mn}\, e^{-j\omega_1 Tm}\, e^{-j\omega_2 Tn}$$

and the a_{mn} can be included in the parameter vector $\underline{x}$. The gradient components of the a_{mn} can be trivially calculated. The only modification to the previous discussion is that a dense set of points in the stopband must now be included.

CHAPTER V

EXPERIMENTAL RESULTS

5.1 Implementation of Ladder Design

The design technique of section 4.1 has been tested using the lowpass filter example of section 4.2. The computer programs are written in Algol W for use with the Stanford Algol compiler on an IBM 360 system. Procedure QNMDER (Appendix A) for unconstrained function minimization is available as an Algol procedure requiring supplementary function and gradient evaluation procedures. The structure of the program is shown in Figure 5.1.

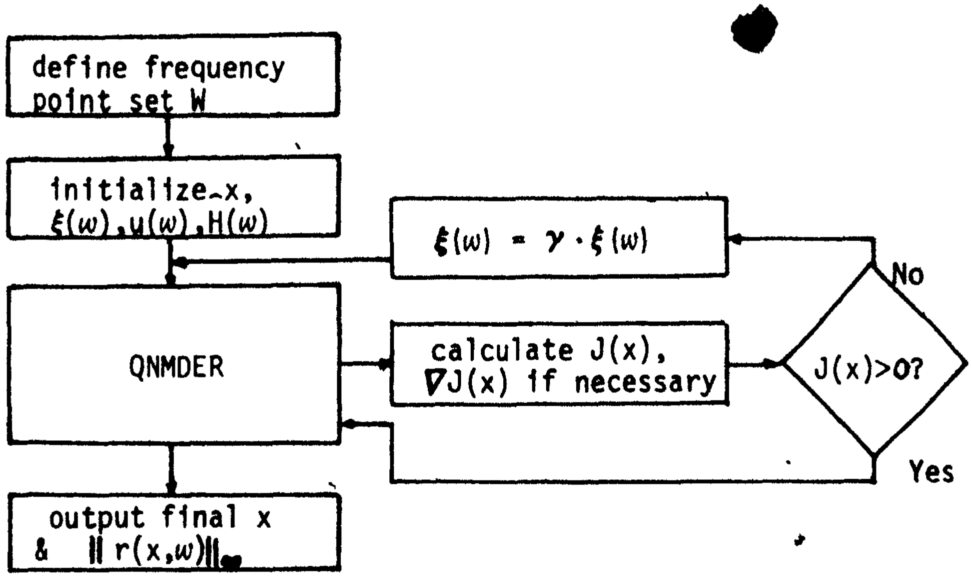


FIGURE 5.1

Procedure FGBAN evaluates the performance functional $J'(x)$ (4.11), and if logical variable GRADYESNO = TRUE, it evaluates $\nabla J'(x)$ also. The flowchart of procedure FGBAN is given in Figure 5.2.

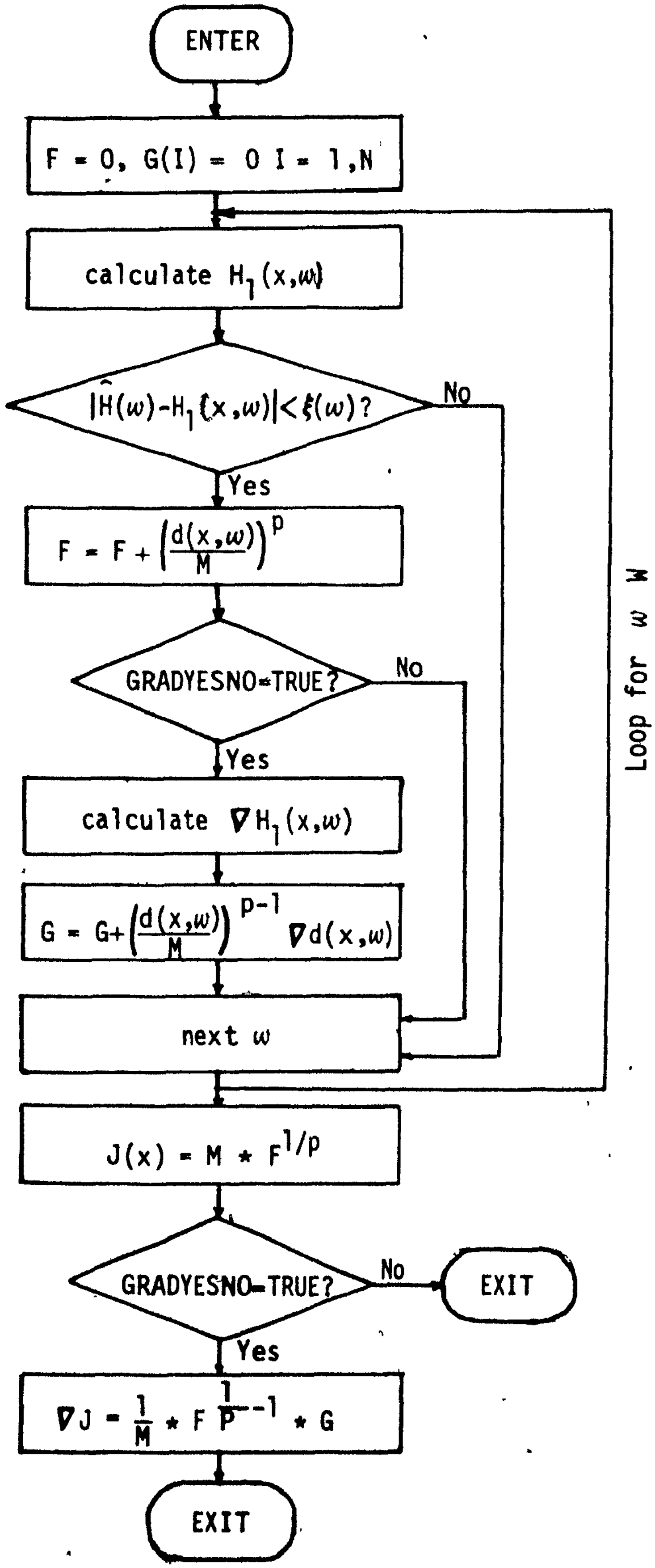

ENTER
F = 0, G(I) = 0 I = 1,N
calculate H1(x,ω)
|Ĥ(ω)-H1(x,ω)|<ξ(ω)?
No
Yes
F = F + (d(x,ω)/M)^p
GRADYESNO=TRUE?
No
Yes
calculate ∇H1(x,ω)
G = G+(d(x,ω)/M)^(p-1) ∇d(x,ω)
next ω
Loop for ω ∈ W
J(x) = M * F^(1/p)
GRADYESNO=TRUE?
No
EXIT
Yes
∇J = 1/M * F^(1/p - 1) * G
EXIT

FIGURE 5.2

The value of $J'(x)$ is theoretically independant of the value of M. Since M is not known until $d(x,\omega)$ has been calculated for all $\omega \in W'$, the value of M from the previous function evaluation can be used, using $M = 1$ for the first function evaluation. A listing of procedure FGBAN is presented in Appendix B.

For the designs tested, the values of ω_p and ω_s in the specification of Figure 4.1 were $\omega_p = .14\,\pi/T$ and $\omega_s = .26\,\pi/T$. The all pole configuration of Figure 4.2 was tested for 2nd, 4th, and 8th order, and the results of these experiments are discussed in the following section.

5.2 Circuit Analogy Design Results

The second order ladder network is shown in Figure 5.3.

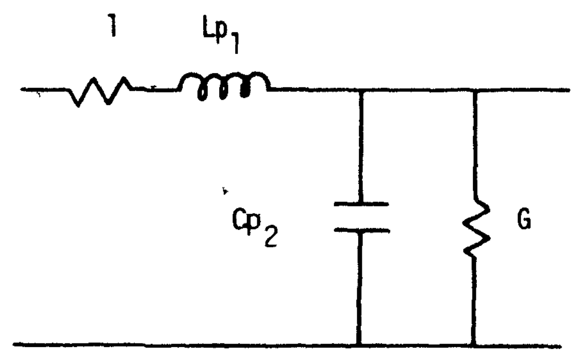


FIGURE 5.3

The transfer function of this filter is given by

$$H(p_1,p_2) = \frac{K}{1 + \frac{C}{1+G}\,p_1 + \frac{LG}{1+G}\,p_2 + \frac{LC}{1+G}\,p_1 p_2} \qquad (5.1)$$

where the value of K does not affect stability and can be set to one for unity transmission at the origin, or it can be made a parameter of the optimization procedure. The latter choice was used for all design examples.

If $G \gg 1$, then $H(p_1,p_2)$ can be written

$$H(p_1,p_2) \approx \frac{K}{(1+\frac{C}{G}p_1)(1+Lp_2)} \qquad (5.2)$$

In this case we say that the transfer function is separable, i.e. $H(p_1,p_2) = H_1(p_1)H_2(p_2)$.

As a starting point for the optimization procedure, the parameters of a filter with a Butterworth characteristic along $p_1 = p_2$ with cutoff at $\omega = .2\pi/T$ can be used, namely $L = C = 6.18$, $G = 1$. For these values, $\|r(x,\omega)\|_\infty = .627$, and $\|\nabla J(x)\|_2 = 6.4 \times 10^{-2}$. Applying the optimization procedure, a limiting solution is obtained: $L = 5.244$ and $C/G = 5.244$ as $C \to \infty$ For example, $L = 5.244$, $C = 1.406 \times 10^{-5}$, $G = 2.681 \times 10^4$ gives $\|r(x,\omega)\|_\infty = .40335$ and $\|\nabla J(x)\|_2 = 1.46 \times 10^{-5}$, while $L = 5.244$, $C = 5.244 \times 10^{-9}$, $G = 10^9$ gives $\|r(x,\omega)\|_\infty = .40329$ and $\|\nabla J(x)\|_2 = 4.05 \times 10^{-11}$. The resulting transfer function is

$$H(p_1,p_2) = \frac{1}{(1+5.24p_1)(1+5.24p_2)} \qquad (5.3a)$$

or, performing the bilinear transformation

$$H(z_1,z_2) = \frac{(1/39)(1+z_1)(1+z_2)}{(1-.68z_1)(1-.68z_2)} \qquad (5.3b)$$

The response of this filter is given in Figure 5.4. This plot and subsequent ones give contours of the transmission of the filter spaced by .1 .

Performing the optimization on the general transfer function

$$H(p_1,p_2) = \frac{1}{1+x_1p_2+x_2p_1+x_3p_1p_2}$$

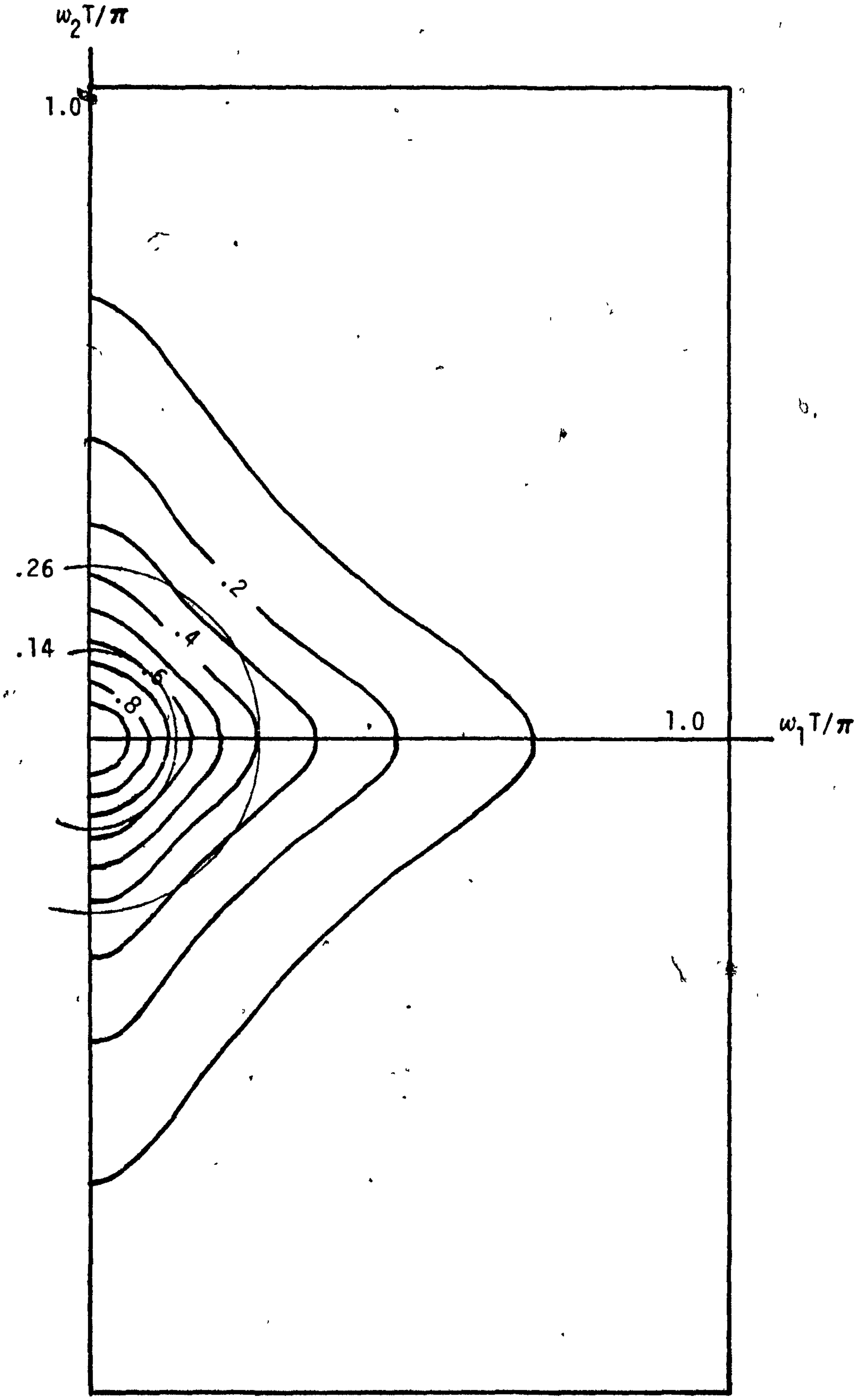


FIGURE 5.4

with starting point x = 1 and no constraints on the coefficients yields the same transfer function as in (5.3), i.e. $x_1 = 5.244$, $x_2 = 5.244$, $x_3 = 27.47$, suggesting that the optimal second order filter is separable. This would account for the singular solution obtained using the circuit analogy method, since the circuit of Figure 5.3 does not possess a separable transfer function for any finite values of the parameters L, C, and G.

In the above example, $\xi(\omega) = .4$ was used, with $u(\omega) = 1$, yielding a solution to P1. It turned out that ξ was a good estimate of $\| r(x_\infty,\omega)\|_\infty$. A solution to P2 with a maximum passband ripple of $\epsilon = .2$ can be obtained by using $\xi(\omega) = .2$, and a high pass to stop weighting. For example, for

$$u(\omega) = \begin{cases} 10 & \omega \in P \\ 1 & \omega \in S \end{cases}$$

the algorithm gives a solution with a maximum passband ripple of .236 and stopband ripple of .549. For

$$u(\omega) = \begin{cases} 100 & \omega \in P \\ 1 & \omega \in S \end{cases}$$

these values become .204 and .583 respectively. In this case the transfer function is

$$H(z_1,z_2) = \frac{(1/17.8)(1+z_1)(1+z_2)}{(1-.527z_1)(1-.527z_2)}$$

with response given in Figure 5.5.

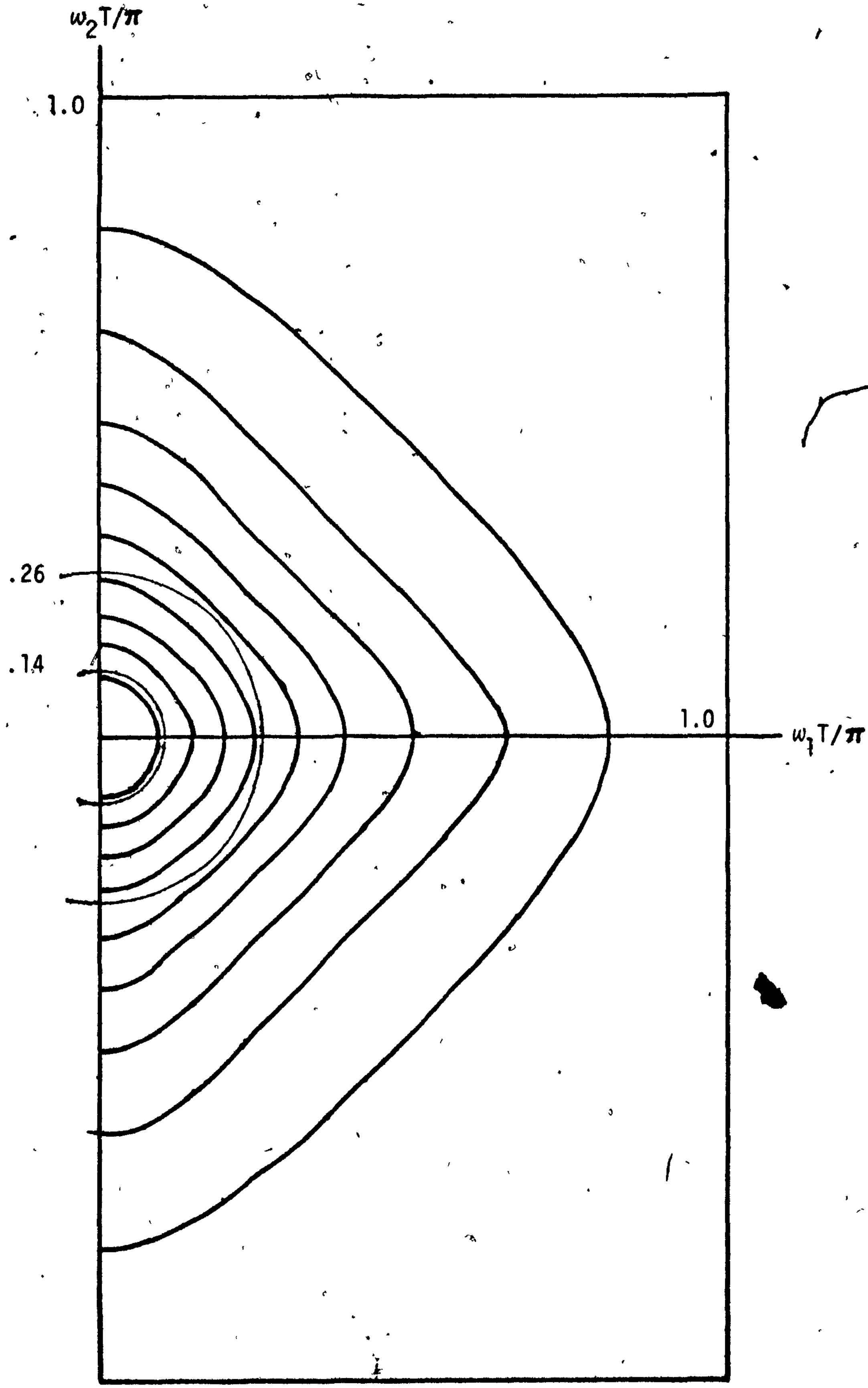


FIGURE 5.5

Figure 5.6 shows the 4th order version of Figure 4.1.

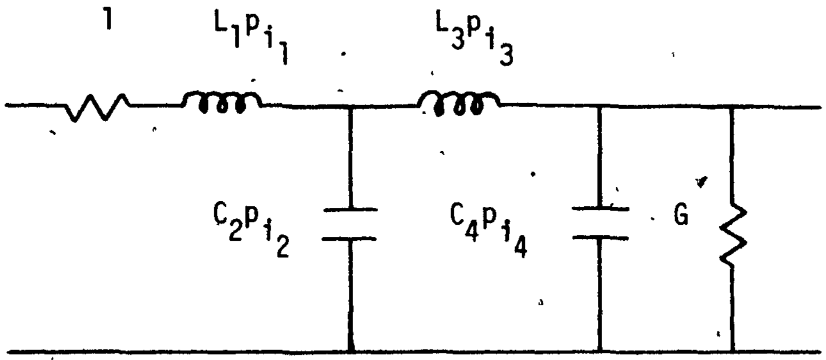


FIGURE 5.6

Eliminating certain choices of the i_k as discussed in section 4.2, there are two possible alternatives: $i_1 = i_2$, $i_3 = i_4$ and $i_1 = i_4$, $i_2 = i_3$. The first case tested was $i_1 = i_2 = 1$, $i_3 = i_4 = 2$, for which the transfer function is given by

$$H(p_1,p_2) = K/(1+G+(L_1G+C_2)p_1+(L_3G+C_4)p_2+(L_1C_4+GL_3C_2)p_1p_2+$$
$$L_1C_2p_1^2+L_3C_4p_2^2+GL_1C_2L_3p_1^2p_2+C_2L_3C_4p_1p_2^2+$$
$$L_1C_2L_3C_4p_1^2p_2^2)$$

The same tests as for the 2nd order case were performed using this circuit. Solving P1 with $\xi(\omega) = .25$ and $u(\omega) = 1$, a separable solution was again obtained, having transfer function

$$H(p_1,p_2) = \frac{1}{(1+5.636p_1+29.24p_1^2)(1+5.636p_2+29.24p_2^2)}$$

or

$$H(z_1,z_2) = \frac{(1/34.88)^2\,(1+z_1)^2\,(1+z_2)^2}{(1-1.62z_1+.705z_1^2)(1-1.62z_2+.705z_2^2)}\ .$$

This gives $\|r(x,\omega)\|_\infty = .2545$, with the response shown in Figure 5.7.

The optimization was performed directly on the general transfer function

$$H(p_1,p_2) = \frac{1}{(1\ p_1\ p_1^2)B\begin{bmatrix}1\\ p_2\\ p_2^2\end{bmatrix}}$$

with no constraints on the coefficients of B. The resulting solution was

$$B=\begin{bmatrix}1.000 & 5.684 & 26.36\\ 5.684 & 51.40 & 174.3\\ 26.36 & 174.3 & 5484.\end{bmatrix},$$

which is not separable. $\|r(x,\omega)\|_\infty = .2391$, as compared with .2545 for the separable case, a slight improvement. However, an application of theorem 2.2 shows this filter to be unstable. This suggests that the optimal separable filter is not far from the "best" in this case.

P2 was solved for $\epsilon = .1$ using $\xi = .1$ and a 100 to 1 pass to stop weighting. The resulting transfer function is

$$H(z_1,z_2)= \frac{(1/23.85)^2\ (1+z_1)^2\ (1+z_2)^2}{(1-1.415z_1 + .584z_1^2)(1-1.415z_2 + .584z_2^2)}$$

with a maximum passband ripple of .103 and a stopband ripple of .407. The response is shown in Figure 5.8.

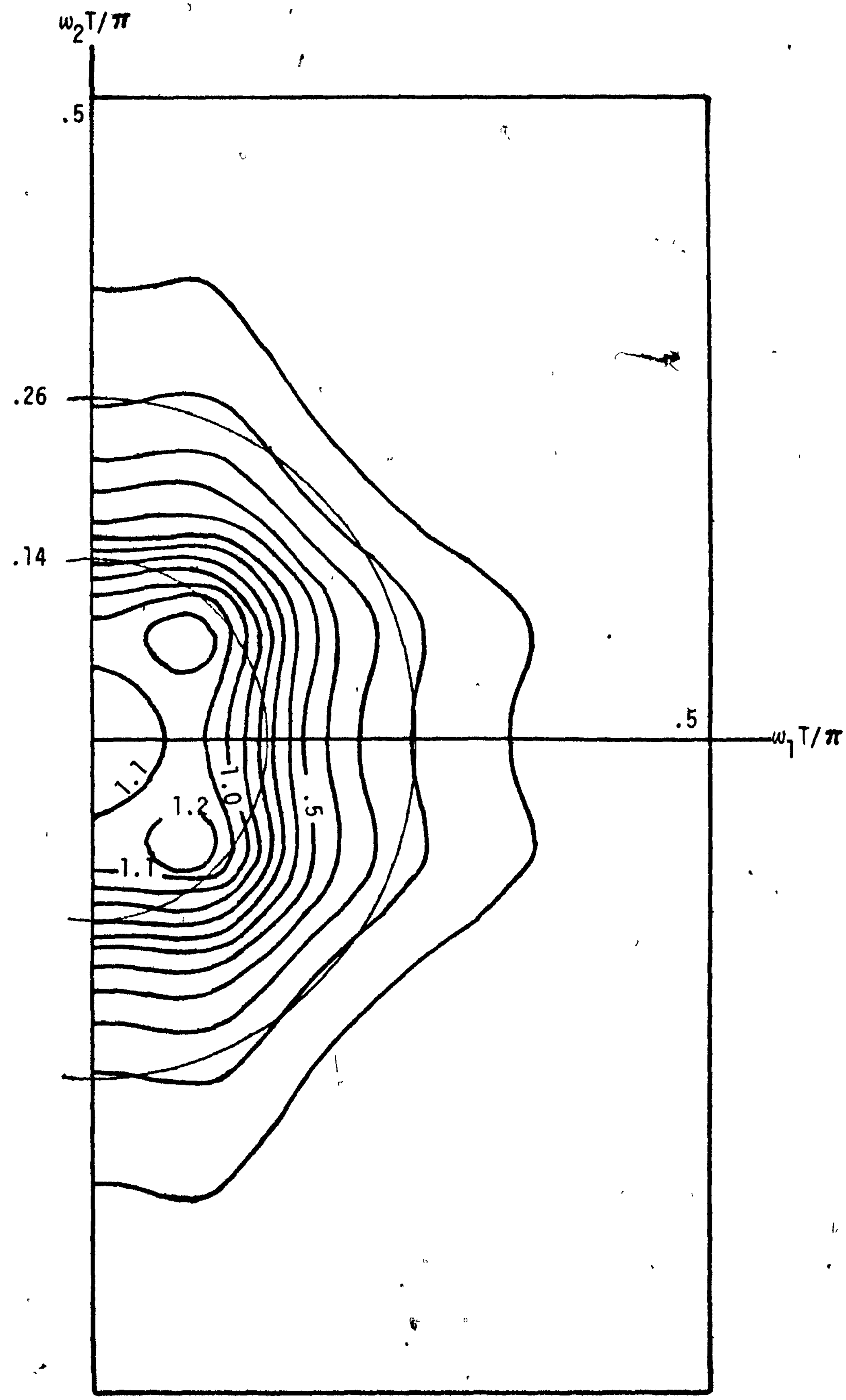


FIGURE 5.7

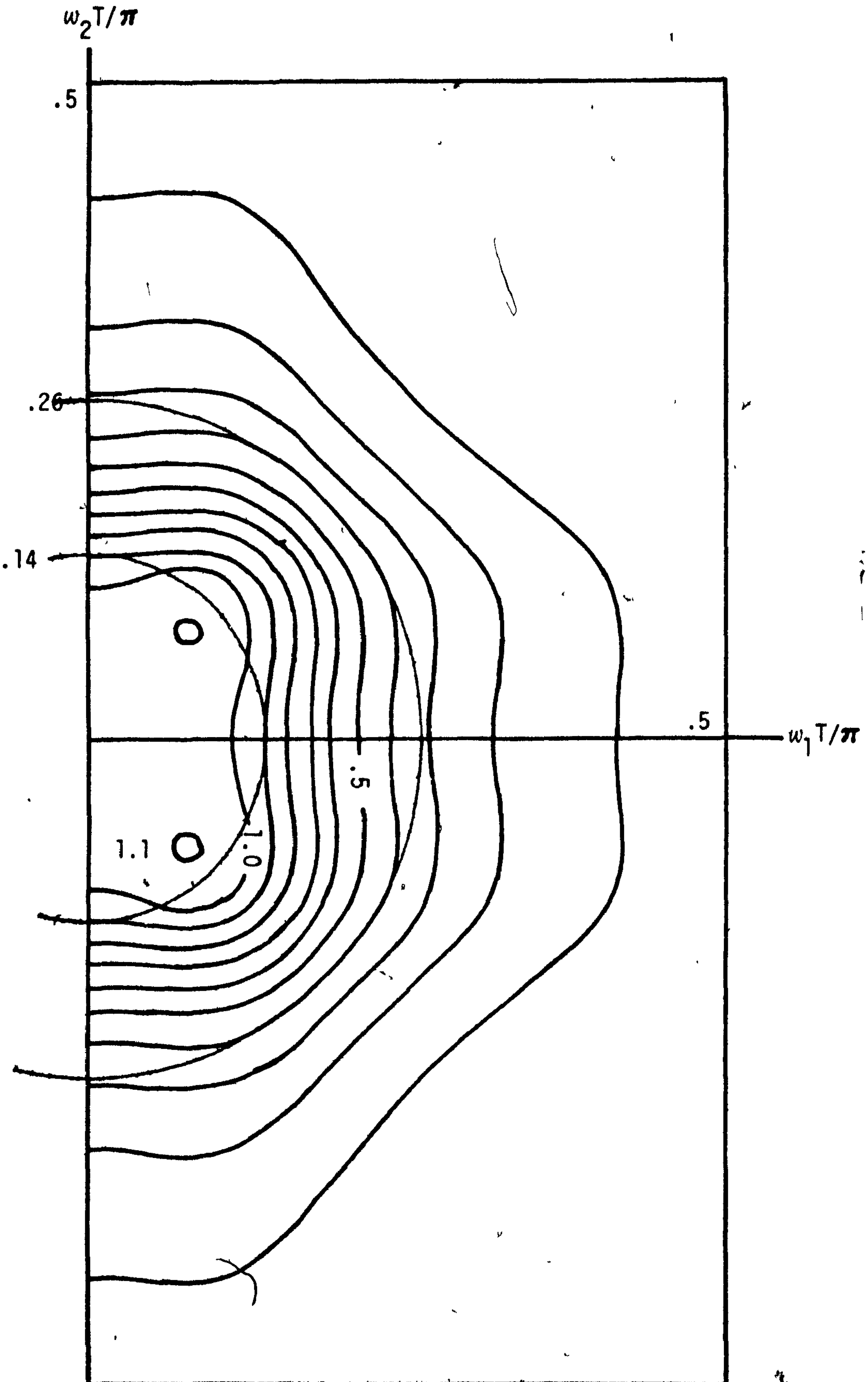


FIGURE 5.8

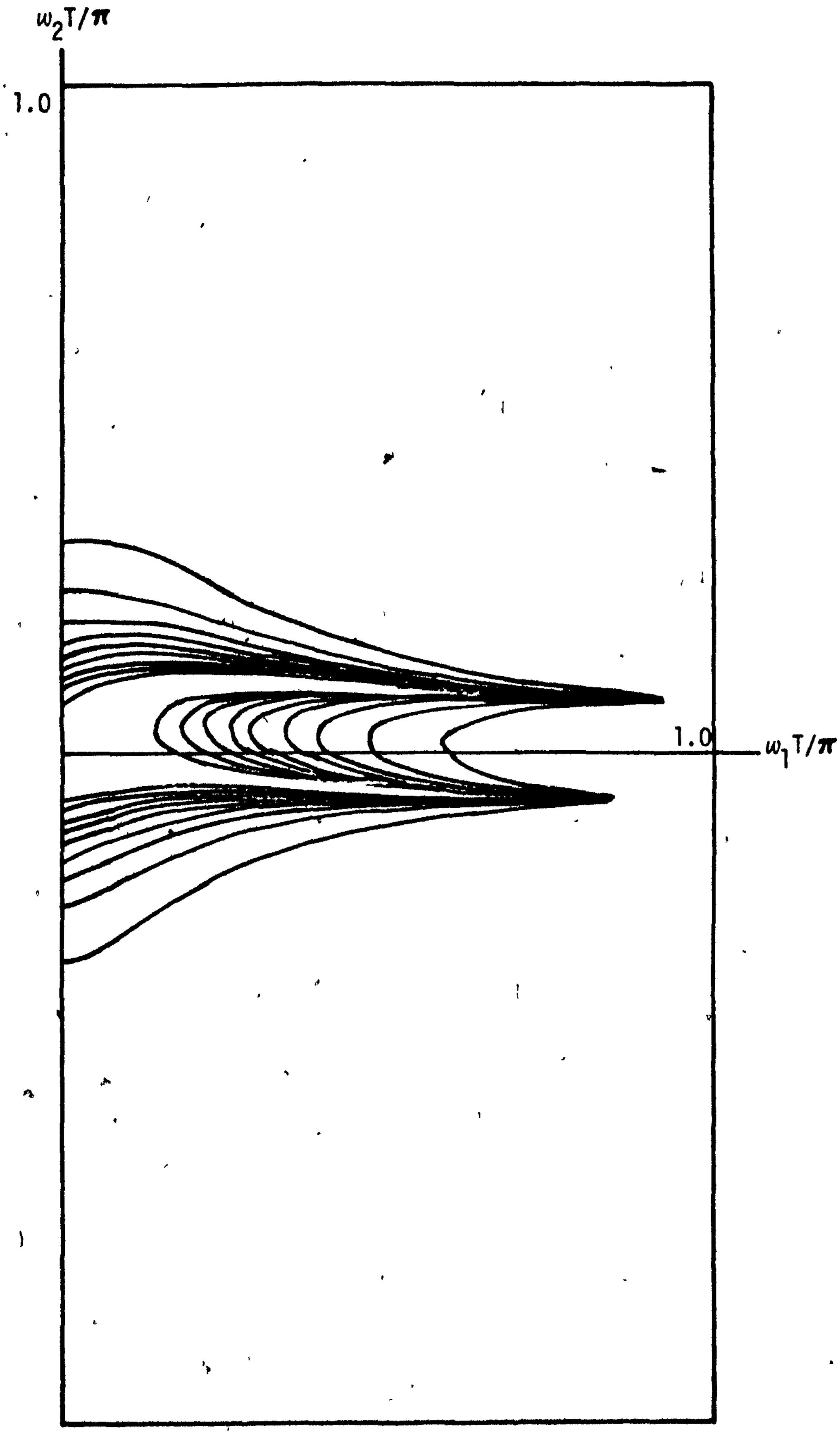


FIGURE 5.9

The other possible assignment of the i_k is $i_1 = i_4 = 1$, $i_2 = i_3 = 2$. This has not proven to be a useful configuration for the design of circularly symmetric lowpass filters. The starting point having a Butterworth characteristic along $p_1 = p_2$ has the response shown in Figure 5.9. The large spikes in the response indicate that this is not a useful configuration. In fact, when the optimization is performed, element values become negative and the resulting unstable network still does not have nearly as good performance as the separable filter.

Only one configuration of the four stage 8th order filter proved useful, namely $i_1 = i_2 = i_3 = i_4$ and $i_5 = i_6 = i_7 = i_8$, as shown in Figure 5.10.

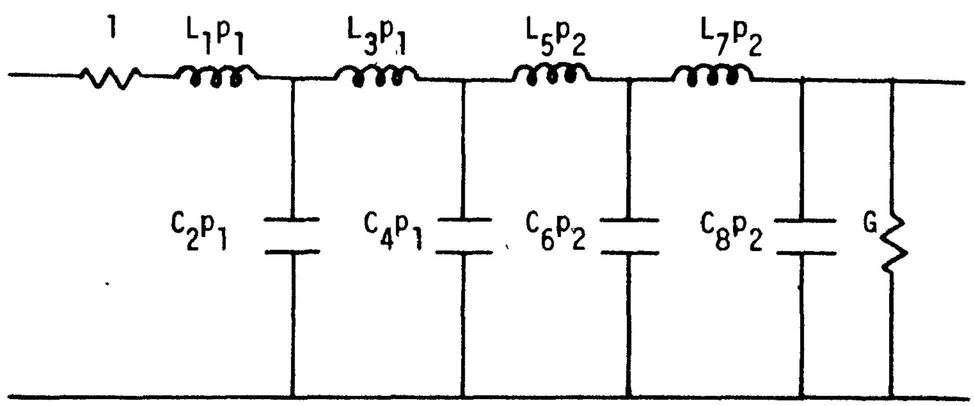


FIGURE 5.10

As in previous cases, a separable solution is obtained:

$$H(p_1,p_2) = 1/(1+12.9p_1 + 97.2p_1^2 + 317.5p_1^3 + 1205p_1^4) \times$$
$$(1+12.9p_2 + 97.2p_2^2 + 317.5p_2^3 + 1205p_2^4)$$

with $\|r(x,\omega)\|_\infty = .0708$. The response is given in Figure 5.11 .

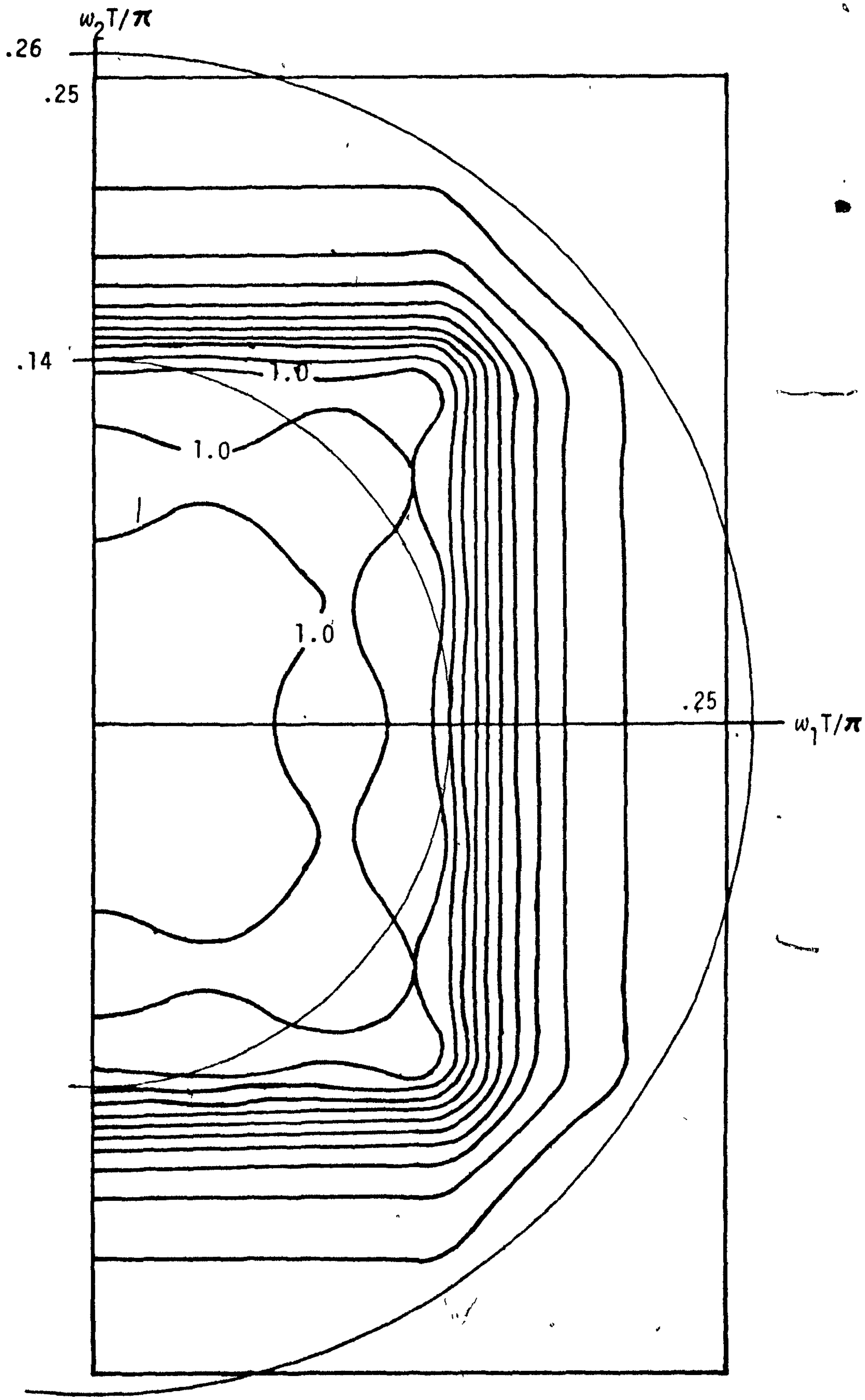


FIGURE 5.11

As mentioned in section 4.2, the numerator coefficients of the transfer function can be chosen freely without regard to stability. Hence, the response of an all pole circuit can be multiplied by the numerator term

$$A(p_1,p_2) = \sum_{m=1}^{M_A} \sum_{n=1}^{N_A} a_{mn} p_1^{m-1} p_2^{n-1}$$

where the a_{mn} become part of the parameter vector x. Some of the a_{mn} can be set to zero if desired, and by making the order of $A(p_1,p_2)$ less than the order of the denominator, zeroes at the Nyquist frequency can be achieved.

This technique was tested in conjunction with the 2nd and 4th order examples of Figure 5.3 and Figure 5.6. For the 2nd order case, the optimization gave $a_{11} = 1$, $a_{12} = a_{21} = a_{22} = 0$, yielding the same solution as before, i.e. with no zeroes. For the 4th order case, the solution obtained was

$$\frac{(1\,p_1\,p_1^2)\begin{bmatrix} 1 & 3.2\text{X}10^{-4} & 3.32 \\ -2.5\text{X}10^{-5} & .097 & -3.3\text{X}10^{-4} \\ 3.32 & -2\text{X}10^{-3} & -6.14 \end{bmatrix}\begin{bmatrix} 1 \\ p_2 \\ p_2^2 \end{bmatrix}}{(1\,p_1\,p_1^2)\begin{bmatrix} 1 \\ 4.45 \\ 21.2 \end{bmatrix}(1\ 4.45\ 21.2)\begin{bmatrix} 1 \\ p_2 \\ p_2^2 \end{bmatrix}}$$

The denominator is separable but the numerator is not. Neglecting small terms, the basic form of the numerator is $1 + a(p_1^2 + p_2^2) - bp_1^2 p_2^2$, which gives the locus of a zero in the stopband. The response of this filter is given in Figure 5.12 and the passband is shown magnified by two in Figure 5.13.

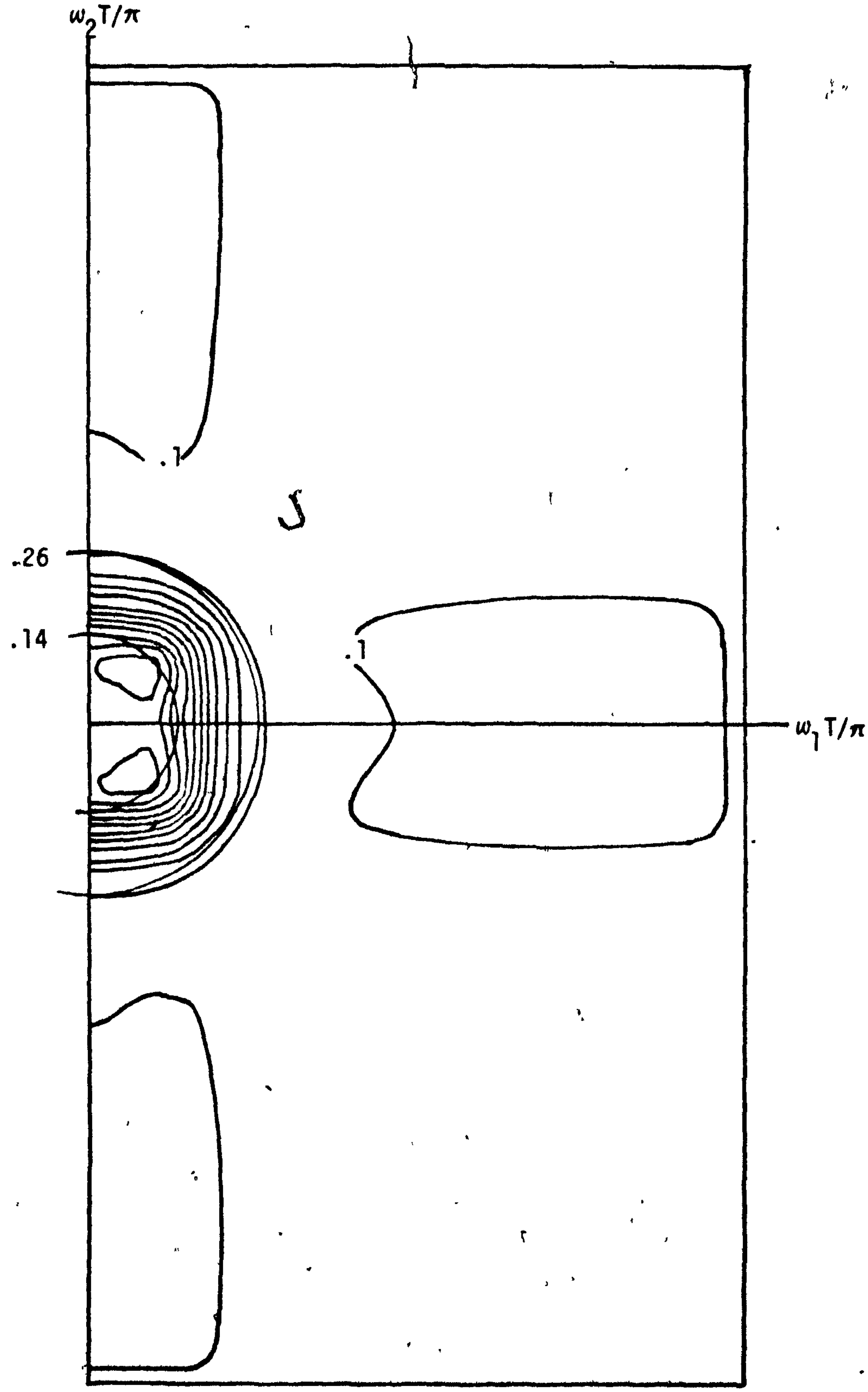


FIGURE 5.12

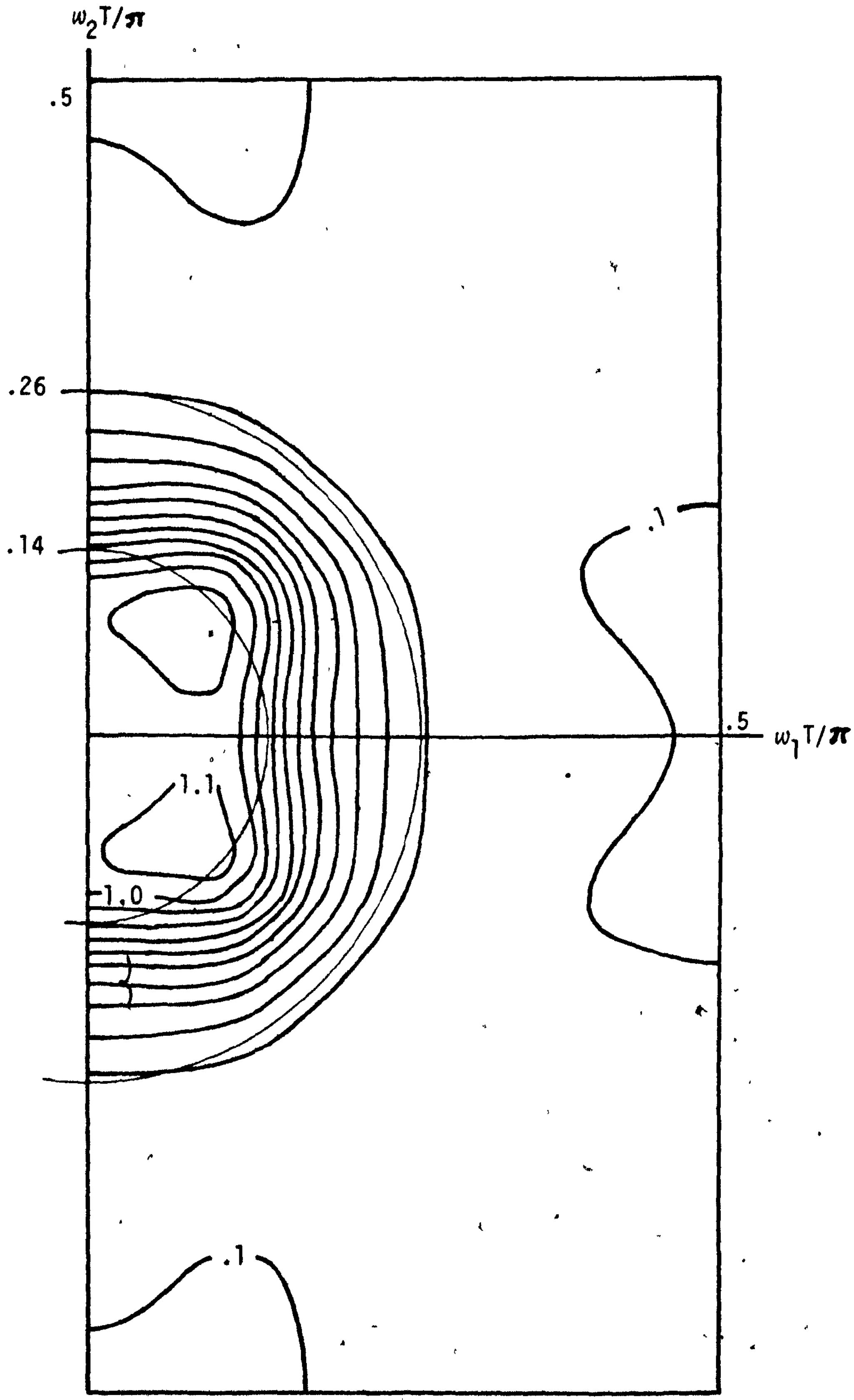


FIGURE 5.13

For the purpose of comparing this method with a design given in the literature, the pass-stop specifications of the example in (29) were used, namely $\omega_p T = .08$ and $\omega_s T = .12$. The solution obtained

$$H(p_1,p_2) = \frac{(1\, p_1\, p_1^2) \begin{bmatrix} 1 & .069 & 17.9 \\ .013 & .065 & .481 \\ 17.9 & 3.07 & -105.8 \end{bmatrix} \cdot \begin{bmatrix} 1 \\ p_2 \\ p_2^2 \end{bmatrix}}{(1 + 7.54p_1 + 79.2p_1^2)(1 + 7.54p_2 + 79.2p_2^2)}$$

The passband response is shown magnified by four in Figure 5.14. This design has a ripple of .3 whereas the design of Maria and Fahmy has a ripple of .35.

5.3 Discussion

From the examples that have been considered, it is evident that the circuit analogy algorithm which has been proposed is a feasible method for the design of two-dimensional recursive digital filters. Using the performance functional developed in Chapter IV, results which are as good as, or better than, the (very few) results which have been published in the literature have been obtained. However, it is also clear that it is not the best possible method for the example considered. Practical considerations make necessary the segregation of p_1 and p_2, and application of the optimization algorithm to this configuration leads to limiting solutions which can be represented by separable transfer functions. In fact, the same solution is obtained as when the optimization is performed directly on the separable transfer function, in which case convergence is faster and the problem of stability can be

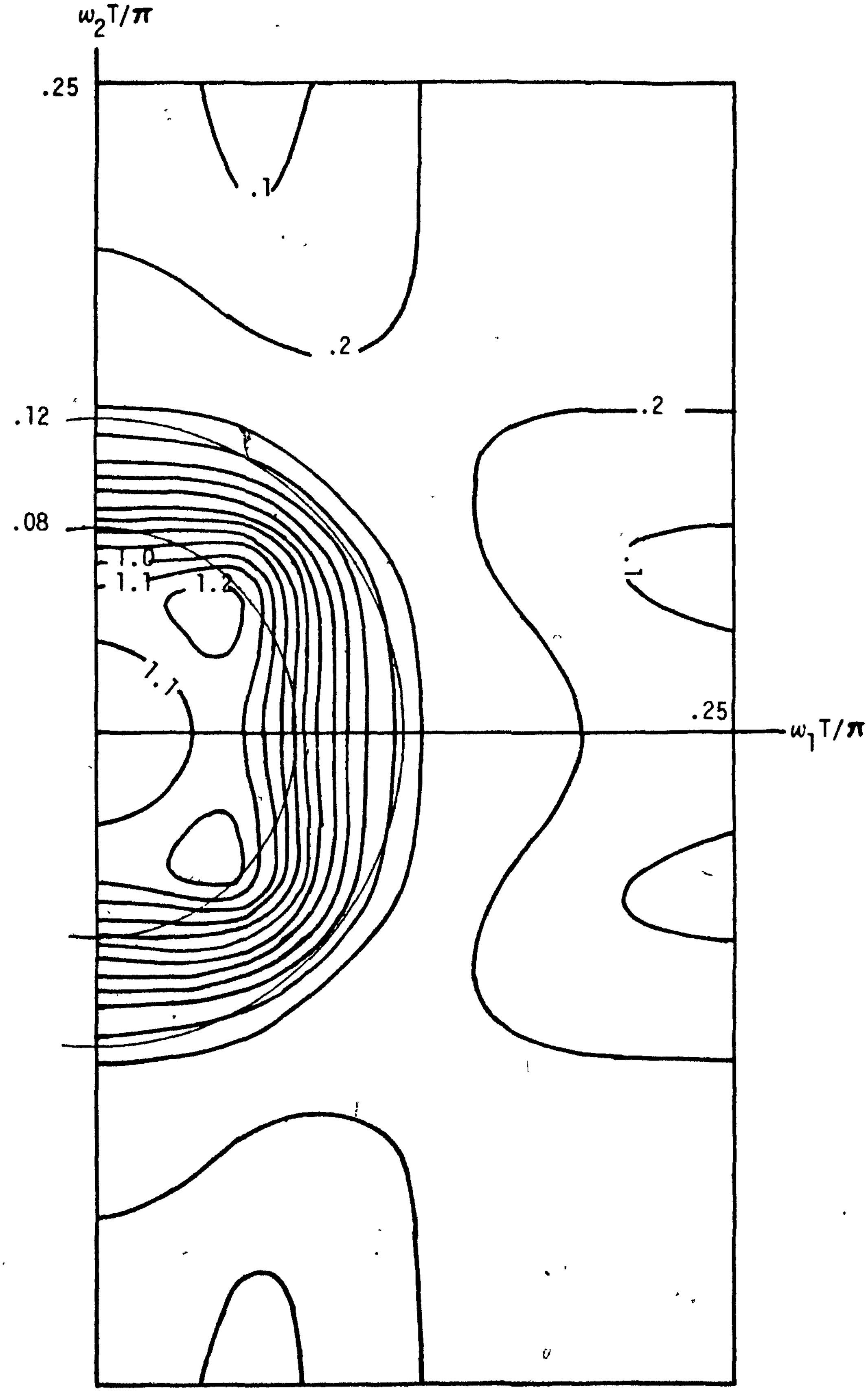


FIGURE 5.14

easily sidestepped. (It must still be accounted for though. A direct application of the optimization to an 8th order separable transfer function led to an unstable solution). Although the circuit analogy method does converge to a solution with no stability test required, the convergence is slow, since the use of gradient methods to minimize a function whose optimal point lies at infinity is not desirable. Of course these comments only apply to the circuit configuration which has been tested. Experience with one-dimensional filters has suggested use of the ladder structure but other configurations may exist giving the desired type of response and exhibiting better convergence in the optimization routine.

The optimization algorithm described in section 4.2 has proven to be a satisfactory method for obtaining minimax type solutions. For example, the response obtained in the 4th order case, shown in Figure 5.7, is seen to possess maximum positive and negative passband deviation and maximum stopband deviation all of the order of .25, making it, in a two-dimensional sense, "equiripple". The algorithm is not dependant on how the transfer function is obtained and can thus be used when the variable parameters are the coefficients of the transfer function itself. Thus, by the inclusion of a stability test, the algorithm could be used to solve the problem in the manner of Maria and Fahmy (29). Furthermore, by an appropriate choice of $\xi(\omega)$ and of the weighting $u(\omega)$, a design can be carried out in which the passband ripple is set to a prescribed value. Although this is the usual manner for the specification of one-dimensional filters, it has not been mentioned in the literature of two-dimensional recursive filtering. The 2nd and 4th order examples discussed previously show that with a high pass to stop weighting, this type of design can be carried out effectively. A pass to stop weighting ratio of 100 to 1 was found to give acceptable results.

In the previous discussions, the value of p used has not been mentioned. It was stated in section 3.1 that for large p, the solution using the L_p norm approaches the Chebyshev approximation (sometimes). To decide how large a value of p should be used, a 4th order separable problem was run with increasing values of p. As p increases, $\|r(x,\omega)\|_\infty$ becomes smaller, approaching a limiting value, as shown in Figure 5.15. The curve has more or less leveled off after p = 20, and this was the value used for all design examples.

Very few results for lowpass designs have been reported in the literature, rendering it difficult to make valid comparative statements about the results obtained here. A comparable result to that obtained by Maria and Fahmy was obtained for the 4th order case with zeroes, with of course no stability tests required. To obtain the quoted result, about 2½ minutes of computer time was required, the denominator once again being derived from a limiting solution. No results for the 8th order case or higher order have been published for comparison. Thus, although the method may by no means yield the optimal solution, it has produced designs comparable, or better, to those designed by other methods.

An alternative method for choosing the numerator coefficients, similar to a technique used by Swanton (41), might prove to be effective. Denominator and numerator coefficients could be chosen separately, iterating back and forth. With a fixed numerator, the best denominator could be chosen, and with that denominator, the best numerator would be chosen, considering only the stopband points in the performance functional, and so on until convergence is obtained. Furthermore, since the numerator is linear, linear programming could be used for that part of the procedure.

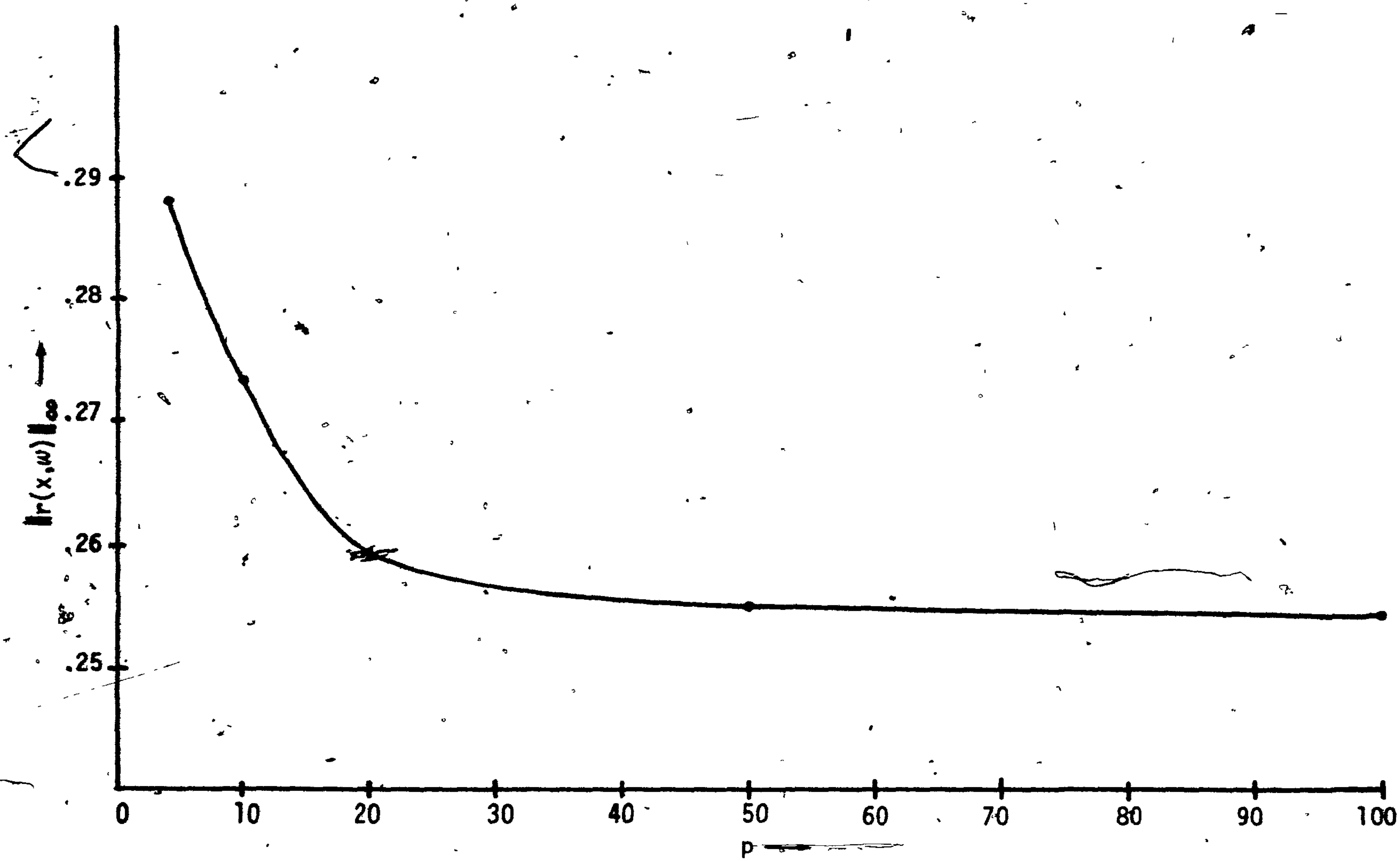


FIGURE 5.15

CHAPTER VI

CONCLUSION

A method for the design of stable two-dimensional recursive digital filters not requiring an explicit stability test has been proposed. In the application of this technique to the design of circularly symmetric lowpass filters, interesting results have been obtained which indicate further areas for investigation. In these designs, some circuit element values were found to tend to infinity or zero during the optimization procedure, while maintaining certain relationships among themselves, yielding separable denominators. However, the numerators did not become separable. Thus, an important question which must be answered is whether the optimal filter (of course "optimal" must be suitably defined) is characterized by a transfer function with a separable denominator. If this is the case, the design procedure could be greatly simplified, as standard one-dimensional techniques could now be applied. The conjecture has been verified empirically for the second order case but whether it applies for higher orders is open. At any rate, the circuit analogy method, as applied in Chapter V, is not the best tool for the solution of the circularly symmetric problem since the separable solution is always obtained, and so the simpler methods might just as well be used. It is possible that configurations other than the ladder structure may prove more useful for this problem. For the non-symmetric problem, however, where separable solutions are clearly not optimal, the method may prove useful.

The raison d'être of the circuit analogy method is its avoidance of the stability test, which may require considerable computational effort. Maria and Fahmy have tried to avoid this problem by dealing only with cascades of second and fourth order sections, thus limiting the size of the stability tests required. However, arguments for lower rounding and quantization noise with cascade forms notwithstanding, the resulting filter may be far from optimal, as general two-variable polynomials cannot be factored in this way. To ensure that the optimal solution can be obtained via the circuit analogy method, another question which must be answered is whether the optimal denominator is always representable as the transfer function of a two-variable passive circuit. Even if this were shown to be true, much work would have to be done in the area of how to select appropriate networks for the desired filter characteristics. If some positive results could be attained with regard to the above mentioned points, the circuit analogy method could prove to be a useful method for the design of two-dimensional recursive digital filters.

APPENDIX A

QUASI-NEWTON METHODS

In recent years, very powerful algorithms have been devised to find the unconstrained local minimum of a function f(x) of n variables, where $f(x) \in C^2$, with gradient vector $g(x) = \nabla f(x)$ and Hessian matrix $G(x) = \left[\frac{\partial^2 f(x)}{\partial x_i \partial x_j}\right]$. This section describes a general class of methods known as Quasi-Newton methods (31, Chapter 6), with particular reference to the implementation of Gill, Murray and Pitfield (18).

If f is quadratic in the neighbourhood of a point x, the minimum can be found in one step, as shown below.

$$f(x+h) = f(x) + h^T g(x) + \tfrac{1}{2} h^T G(x) h \qquad \text{(A-1)}$$

At the minimum $\frac{\partial f}{\partial h} = 0$, which gives the equation

$$g(x) + G(x)h = 0,$$

and solving this, the step is given by

$$h = -G(x)^{-1} g(x).$$

This forms the basis for the iterative algorithm known as Newton's method, where successive approximations to the minimum are given by

$$x_{k+1} = x_k - G(x_k)^{-1} g(x_k) \qquad \text{(A-2)}$$

It can be shown that $G(x_k)$ must be a positive definite matrix for (A-2) to represent a descent step. Although convergence of Newton's method is in general ultimately quadratic, it often fails to converge from a poor initial estimate. To overcome this problem, a linear search parameter α can be incorporated to ensure that $f(x_{k+1}) < f(x_k)$. In this case, the iteration is given by

$$x_{k+1} = x_k - \alpha_k G(x_k)^{-1} g(x_k) \tag{A-3}$$

where α_k is chosen to minimize the function of one variable

$$r(\alpha) = f(x_k - \alpha G(x_k)^{-1} g(x_k)).$$

Such linear searches play an important role in most optimization techniques now in use. The approach most often used is to interpolate $r(\alpha)$ by a second or third order polynomial and take the minimum of this polynomial as an estimate to the minimum of $r(\alpha)$. This is done iteratively until the desired accuracy is achieved. A detailed discussion of linear search techniques is given in (31, Chapter 1).

To implement (A-3), both the gradient and Hessian must be explicitly available, and a system of linear equations must be solved. If both $g(x)$ and $G(x)$ are unavailable, the search must be based on function values only and direct search techniques such as Rosenbrock's method or the conjugate direction method of Powell are in order. Some of the gradient methods, however, have been implemented using finite differences to approximate the gradient, and have proved quite effective. It will be assumed from here on that the gradient $g(x)$ is available but that the Hessian $G(x)$ is not.

The Quasi-Newton methods require only the gradient, and an approximation to the Hessian is constructed, being updated at each iteration. The procedure is basically as follows, where B_k is the approximate Hessian and C_k the update to the Hessian at the k^{th} iteration.

set x_o, B_o

$$\begin{aligned} &\text{solve } B_k p_k = -g(x_k) \\ &x_{k+1} = x_k + \alpha_k p_k \\ &B_{k+1} = B_k + C_k \\ &k = k + 1 \end{aligned} \qquad \text{(A-4)}$$

α_k is chosen by a line search along the direction p_k, as discussed previously. B_o can be initialized to the identity matrix. An equivalent algorithm, in which H_k, an approximation to the inverse Hessian, is constructed, can also be used. The iteration is then as follows.

set x_o, H_o

$$\begin{aligned} &p_k = H_k g(x_k) \\ &x_{k+1} = x_k + \alpha_k p_k \\ &H_{k+1} = H_k + E_k \\ &k = k + 1 \end{aligned} \qquad \text{(A-5)}$$

The solution of a set of linear equations has now been replaced by a matrix multiplication.

The basic difference between the various Quasi-Newton methods lies in the choice of C_k and E_k. In discussing this choice, the following notation will be used.

$$p_k = - B_k^{-1} g(x_k) \tag{A-6}$$

$$s_k = x_{k+1} - x_k \tag{A-7}$$

$$y_k = g(x_{k+1}) - g(x_k) \tag{A-8}$$

If f(x) is quadratic as in (A-1), then $g(x_{k+1}) = 0$. (A-2) can then be written

$$x_{k+1} - x_k = G(x_k)^{-1}(g(x_{k+1}) - g(x_k))$$

or in the above notation

$$s_k = G(x_k)^{-1} y_k \tag{A-9}$$

It is thus desirable that the approximate Hessian B_k satisfy this equation. However B_k is needed to compute s_k and y_k and so the following relation, known as the Quasi-Newton equation, is used.

$$B_k s_{k-1} = y_{k-1} \tag{A-10a}$$

Equivalently, for the iteration (A-5)

$$s_{k-1} = H_k y_{k-1} \tag{A-10b}$$

For all Quasi-Newton methods, C_k or E_k is chosen so that (A-10) is satisfied. The update should be simple, generally a rank 1 or rank 2 matrix, i.e.

$$C_k = \pi_1 w_1 w_1^T + \pi_2 w_2 w_2^T \tag{A-11}$$

where w_1 and w_2 are n-vectors and π_1 and π_2 are scalars. $\pi_2 = 0$ for a rank 1 update. A popular method which falls into this category is the Davidon-Fletcher-Powell (DFP) algorithm, which uses the rank 2 update

$$E_k = \frac{s_k s_k^T}{s_k^T y_k} - \frac{H_k y_k y_k^T H_k}{y_k^T H_k y_k} \tag{A-12}$$

It can easily be verified that (A-10b) is satisfied. The DFP algorithm is usually derived using the theory of conjugate directions, and it can be shown that if f(x) is quadratic, the minimum will be obtained in at most n iterations.

Another popular update is the complementary DFP (COMDFP) algorithm,

$$C_k = \frac{y_k y_k^T}{s_k^T y_k} + \frac{g_k g_k^T}{p_k^T g_k} \tag{A-13}$$

which can be shown to satisfy (A-10a). Gill and Murray (19) give a detailed discussion of this algorithm and its advantages. It has been implemented by Gill, Murray and Pitfield (18) in Algol procedure QNMDER (Quasi-Newton Method with DERivatives). The program is quite sophisticated, including checks on rounding error.

In the iteration (A-4), it is required to solve the set of linear equations

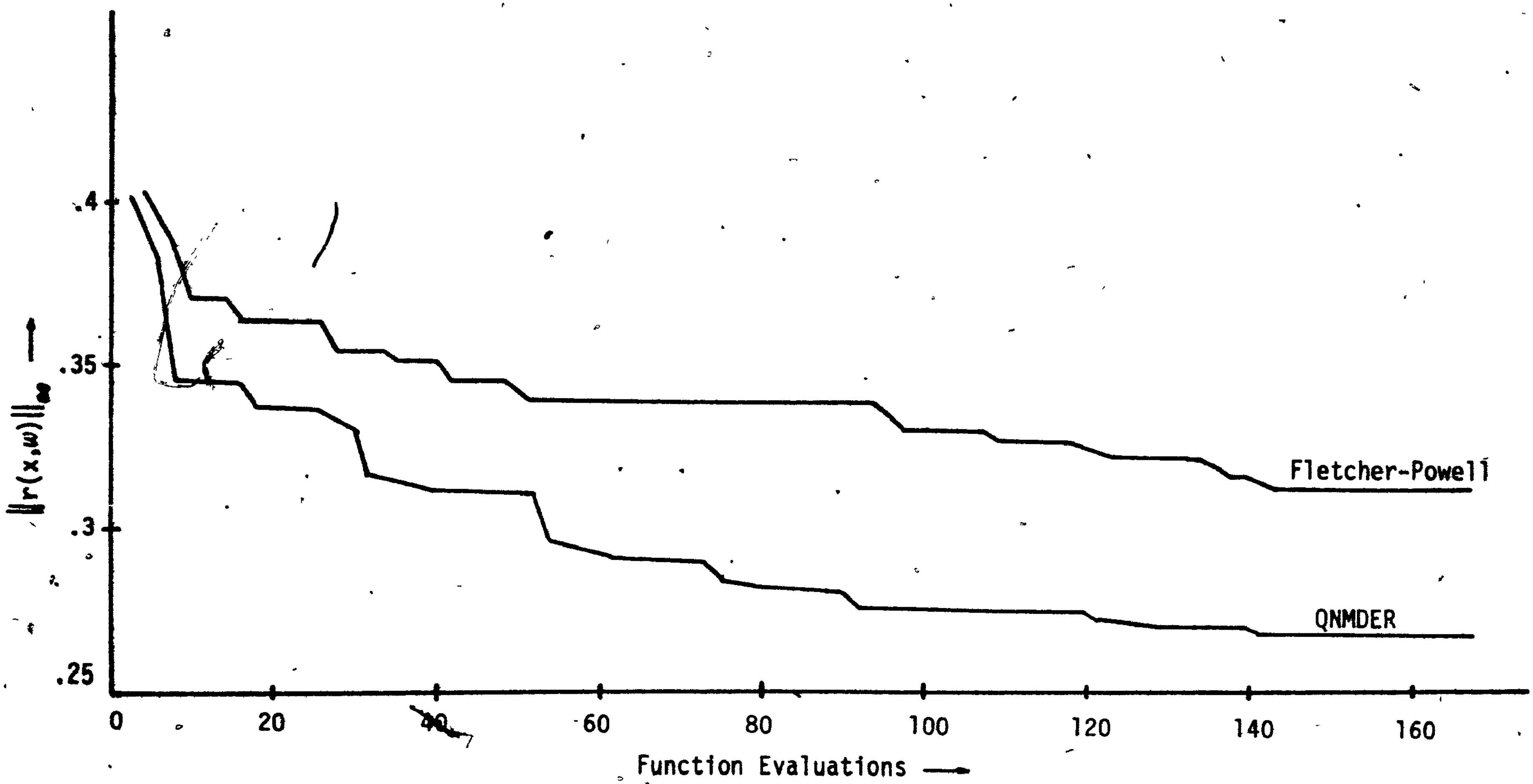
$$B_k p_k = -g(x_k)$$

This can be accomplished much more efficiently if B_k is available in the form

$$B_k = L_k D_k L_k^T \quad \text{(A-14)}$$

where L_k is unit lower diagonal and D_k is diagonal. Rather than updating B_k directly, the factors L_k and D_k can be updated, saving considerable computation time. The method used in QNMDER is numerically stable and guarantees that the new approximate Hessian is positive definite irrespective of rounding error.

As shown in (18), the performance of QNMDER on most standard test functions is equal or better than that of the other commonly used techniques, such as DFP. Since the performance of an algorithm is always problem dependent, both Fletcher-Powell (SSP subroutine) and QNMDER were used to perform a typical minimization of the performance functional J'(x) described in section 4.2. The behaviour of $\|r(x,\omega)\|_\infty$ versus the number of function evaluations is shown for both methods in Figure A.1, suggesting the superiority of QNMDER for this application.



.4
.35
.3
.25
$\|r(x,w)\|_\infty$
0
20
40
60
80
100
120
140
160
Function Evaluations
Fletcher-Powell
QNMDER

FIGURE A-1

APPENDIX B

A listing of the Algol program used to obtain the experimental results discussed in Chapter V is presented in this appendix. Included is procedure QNMDER described in Appendix A. The exact implementation of Gill, Murray and Pitfield has been used. The general setup routine and procedure FGBAN for calculation of the performance functional and its gradient are also included.

```
BEGIN
LOGICAL PRINT;
INTEGER N,NTHMAX,N1;
INTEGER NR,NR1,NR2,IP,NCOUNT,LIMIT;
READ(N,N1);
READ(IP,NR1,NR2,NTHMAX);
NR:=NR1+NR2;
INTFIELDSIZE:=3;
PRINT:=FALSE;

BEGIN
LONG REAL ARRAY W1,W2,U(1::NR,1::NTHMAX);
LONG REAL ARRAY XMIN(1::N);
LONG REAL ARRAY HHAT(1::NR);
   LONG REAL ARRAY R(1::NR);
   LONG REAL ARRAY D,X,G(1::N);
LONG REAL ARRAY L(1::(N*(N-1)) DIV 2 +1);
INTEGER ARRAY IND(1::N);
INTEGER ARRAY NTH(1::NR);
LONG REAL XI,MMOLD;
LONG REAL DEV;
   LONG REAL TH,F,ETA,TOL;
   INTEGER ITNUM, GTOTAL, FTOTAL;
   LOGICAL LCADL, CONV;
INTEGER NE;
REAL ARRAY X1(1::20);
REAL FP,XNORM;

LONG REAL PROCEDURE DTAN(LONG REAL VALUE X);
   LONGSIN(X)/LONGCOS(X);


PROCEDURE QNMDER( INTEGER VALUE   N;                                   QNMDE001
                  INTEGER RESULT  FTOTAL, GTOTAL, ITNUM;               QNMDE002
                  LOGICAL VALUE   LCADL;                               QNMDE003
                  LOGICAL RESULT  CONV;                                QNMDE004
                  LONG REAL VALUE ETA, MACHEPS, TOL, MAXSTEP;          QNMDE005
                  LONG REAL VALUE RESULT F;                            QNMDE006
                  LONG REAL ARRAY X, L, D(*);                          QNMDE007
                   PROCEDURE  FUN , GRAD , FGBAN);
BEGIN                                                                  QNMDE009
                                                                       QNMDE010
   COMMENT: PROCEDURE QNMDER ACHIEVES FUNCTION MINIMIZATION            QNMDE011
            USING A REVISED QUASI-NEWTON METHOD WITH DERIVATIVES.      QNMDE012
            THE PROCEDURE ATTEMPTS TO FIND THE POINT X AT WHICH        QNMDE013
            THE TWICE CONTINUOUSLY DIFFERENTIABLE FUNCTION F(X)        QNMDE014
            ATTAINS ITS LEAST VALUE.  IDEALLY, THE VARIABLES SHOULD     QNMDE015
            BE SCALED SO THAT THE HESSIAN MATRIX AT THE SOLUTION IS    QNMDE016
            APPROXIMATELY ROW EQUILIBRATED, WITH THE FUNCTION MULTI-   QNMDE017
            PLIED BY A SCALAR SUCH THAT IT ACHIEVES A MAXIMUM VALUE    QNMDE018
            OF UNITY WITHIN A UNIT SPHERE SURROUNDING THE MINIMUM.     QNMDE019
            IT MAY NOT BE POSSIBLE TO FULFILL EITHER OF THESE RE-      QNMDE020
            QUIREMENTS.  GIVEN AN INITIAL APPROXIMATION TO THE         QNMDE021
            MINIMUM AND AN ESTIMATE OF THE MINIMUM VALUE, THE PRO-     QNMDE022
            CEDURE CALCULATES A LOWER FUNCTION VALUE AT EACH ITER-     QNMDE023
            ATION.  WHEN THE CONVERGENCE CRITERIA ARE SATSIFIED THE    QNMDE024
            PROCEDURE GIVES THE ESTIMATED POSITION OF THE MINIMUM,     QNMDE025
            THE FINAL FUNCTION VALUE, AND THE CHOLESKY FACTORIZATION   QNMDE026
```

```
              OF THE APPROXIMATE HESSIAN MATRIX;                          QNMDE027
                                                                          QNMDE028
    INTEGER    J, FCOUNT, GCOUNT, FNUM, GNUM, COUNT;                      QNMDE029
    LOGICAL    SUCCESSFULSEARCH;                                          QNMDE030
    LONG REAL  KBOUND, GTP1;                                              QNMDE031
    LONG REAL  ROOTMACHEPS, NORMP, GTP, FM, OLDF, NEWF, V, TOLSQ, ALPHA;  QNMDE032
    LONG REAL  ARRAY GK, GKPLUSONE, Y, P(1::N);                           QNMDE033
                                                                          QNMDE034
COMMENT: ===============================================================;QNMDE035
                                                                          QNMDE036
    PROCEDURE DELINSEARCH( INTEGER VALUE    N;                            QNMDE037
                           INTEGER RESULT   FNUM, GNUM;                   QNMDE038
                           LOGICAL RESULT   SUCCESSFULSEARCH;             QNMDE039
                           LONG REAL VALUE ETA, MACHEPS, MAXSTEP;         QNMDE040
                           LONG REAL VALUE RESULT F, ALPHA;               QNMDE041
                           LONG REAL ARRAY P, X, G(*);                    QNMDE042
                           PROCEDURE        FUN, GRAD ,FGRAN);            QNMDE043
    BEGIN                                                                 QNMDE044
                                                                          QNMDE045
    COMMENT: PROCEDURE DELINSEARCH FINDS AN APPROXIMATION ALPHA TO THE    QNMDE046
             POINT AT WHICH THE FUNCTION F(X + ALPHA*P) ATTAINS ITS       QNMDE047
             MINIMUM VALUE ALONG THE VECTOR P.  THE METHOD USED IS THAT   QNMDE048
             OF SUCCESSIVE CUBIC INTERPOLATION WITH SAFEGUARDS, DE-       QNMDE049
             SCRIBED IN SECTION 2.4.  THE PROCEDURE IS USED IN CONJUNC-   QNMDE050
             TION WITH PROCEDURE QNMDER, AND USES REAL PROCEDURE DOT;     QNMDE051
                                                                          QNMDE052
      INTEGER   KASE;                                                     QNMDE053
      LONG REAL MAXALPHA, S, T, X1, X2, XK, XMIN, F1, F2, FK, FMIN,       QNMDE054
                NEWALPHA, G1,G2,GK, LBOUND,UBOUND, OLDF, GTEST1,GTEST2;   QNMDE055
      LOGICAL GCALC;
      LONG REAL ARRAY Y, Z(1::N);                                         QNMDE056
                                                                          QNMDE057
      G1:= DOT( N, G, P );                                                QNMDE058
      NEWALPHA:= IF ALPHA> MAXSTEP THEN MAXSTEP                           QNMDE059
                                   ELSE ALPHA;                            QNMDE060
                                                                          QNMDE061
    COMMENT: ENSURE THAT THE INITIAL PROJECTED GRADIENT IS NEGATIVE       QNMDE062
             AND THE INITIAL STEP IS NON-ZERO.                            QNMDE063
                                                                          QNMDE064
      IF (G1 >= 0L) OR (NEWALPHA <= 0L) THEN                              QNMDE065
         BEGIN                                                            QNMDE066
            FNUM:= GNUM:= 0;                                              QNMDE067
            FMIN:= OLDF;                                                  QNMDE068
            GOTO TERMINATESEARCH;                                         QNMDE069
         END;                                                             QNMDE070
                                                                          QNMDE071
    COMMENT: FIND FIRST NEW POINT;                                        QNMDE072
                                                                          QNMDE073
      F1:= OLDF:= F;                                                      QNMDE074
      MAXALPHA:= MAXSTEP - NEWALPHA;                                      QNMDE075
      X1:= -NEWALPHA;                                                     QNMDE076
      GTEST1:= -1'-4*G1;                                                  QNMDE077
      GTEST2:= -ETA*G1;                                                   QNMDE078
      LBOUND:= XK:= X2:= 0L;                                              QNMDE079
                                                                          QNMDE080
    COMMENT: CALCULATE FUNCTION AT X + ALPHA*P;                           QNMDE081
                                                                          QNMDE082
      FOR I:= 1 UNTIL N DO  Z(I):= X(I) + NEWALPHA*P(I);                  QNMDE083
```

```
    FGRAN(N,Y,Z,FK,TRUE);
  IF(KCOUNT>LIMIT) THEN GO TO PREMATUREEJECTION;
    GCALC:=TRUE;
    F2:= FK;                                                          QNMDE085
    FNUM:= 1;                                                         QNMDE086
    GNUM:= 0;                                                         QNMDE087
                                                                      QNMDE088
ITERATE:                                                              QNMDE089
                                                                      QNMDE090
  COMMENT: SET UP INTERVAL BOUNDS, DETERMINE THE ARRANGEMENT OF POINTS,QNMDE091
           CALCULATE THE LOWEST POINT FMIN, AND THE STEP TO THE LOWEST QNMDE092
           POINT XMIN.                                                QNMDE093
           THE 6 POSSIBLE CASES ARE AS FOLLOWS:                       QNMDE094
              KASE = 1:   G1 < 0,   G2 < 0,   F1 > F2                 QNMDE095
              KASE = 2:   G1 > 0,   G2 > 0,   F1 < F2                 QNMDE096
              KASE = 3:   G1 < 0,   G2 > 0,   F1 > F2                 QNMDE097
              KASE = 4:   G1 < 0,   G2 > 0,   F1 < F2                 QNMDE098
              KASE = 5:   G1 > 0,   G2 > 0,   F1 > F2                 QNMDE099
              KASE = 6:   G1 < 0,   G2 < 0,   F1 < F2                 QNMDE100
                                                                      QNMDE101
           CALCULATE THE GRADIENT VECTOR:           ;                 QNMDE102
                                                                      QNMDE103
    IF(¬GCALC) THEN GRAD(N,Z,Y);
    GCALC:=FALSE;
    GNUM:= GNUM + 1;                                                  QNMDE105
    GK:=.DOT( N, Y, P );                                              QNMDE106
    IF XK = X1 THEN G1:= GK                                           QNMDE107
               ELSE G2:= GK;                                          QNMDE108
                                                                      QNMDE109
  COMMENT: OVERWRITE ARRAY G(I), I=1(1)N, WITH THE GRADIENT AT XK;    QNMDE110
                                                                      QNMDE111
    IF (FK <= F1) AND (FK <= F2) THEN                                 QNMDE112
       FOR I:= 1 UNTIL N DO  G(I):= Y(I);                             QNMDE113
    IF (G1 < OL ) AND (G2 > OL ) THEN                                 QNMDE114
       BEGIN                                                          QNMDE115
          KASE:= IF F1 > F2 THEN 3 ELSE 4;                            QNMDE116
          LBOUND:= X1;                                                QNMDE117
          UBOUND:= X2;                                                QNMDE118
       END                                                            QNMDE119
    ELSE IF G1 > OL THEN                                              QNMDE120
       BEGIN                                                          QNMDE121
          IF F1 < F2 THEN                                             QNMDE122
             BEGIN                                                    QNMDE123
                KASE:= 2;                                             QNMDE124
                UBOUND:= X1;                                          QNMDE125
             END                                                      QNMDE126
          ELSE                                                        QNMDE127
             KASE:= 5;                                                QNMDE128
       END                                                            QNMDE129
    ELSE                                                              QNMDE130
       BEGIN                                                          QNMDE131
          IF F1 > F2 THEN                                             QNMDE132
             BEGIN                                                    QNMDE133
                KASE:= 1;                                             QNMDE134
                LBOUND:= X2;                                          QNMDE135
                IF X2 > UBOUND THEN UBOUND:= X2;                      QNMDE136
             END                                                      QNMDE137
          ELSE                                                        QNMDE138
```

```
            BEGIN
               KASE:= 6;
               LBOUND:= X1;
               UBOUND:= X2;
            END;
      END;

COMMENT: SCALE X1, X2, LBOUND, AND UBOUND, WITH RESPECT TO XK;

   X1:= X1 - XK;
   X2:= X2 - XK;
   LBOUND:= LBOUND - XK;
   LBOUND:= UBOUND - XK;
   MAXALPHA:= MAXALPHA - XK;

   IF F1 > F2 THEN
      BEGIN
         XMIN:= X2;
         FMIN:= F2;
         T:= G2;
      END
   ELSE
      BEGIN
         XMIN:= X1;
         FMIN:= F1;
         T:= G1;
      END;

COMMENT: TERMINATION CRITERIA;

   IF ( (KASE>1) OR (UBOUND~=X2) ) AND ( UBOUND-LBOUND<=MACHEPS )
   OR ( (MAXALPHA<=X2) AND (G2<=0L) )
   OR ( (KASE>4) OR (ABS T < GTEST2) )
      AND ( CLDF-FMIN > GTEST1*(NEWALPHA + XMIN) )
   THEN GOTO TERMINATESEARCH;

COMMENT: CALCULATE TRIAL XK;

   IF KASE > 4 THEN
      BEGIN
         COMMENT: NON-UNIMODAL FUNCTION;
         XK:= (X1 + X2)/2L;
         GOTO EVALUATEFK;
      END;

   T:= 3L * (F2 - F1)/(X2 - X1) - G1 - G2;
   XK:= 1L - G2*G1/T/T;
   S:= LONGSQRT( ABS XK ) * ABS T;
   XK:=  IF XK > 0L THEN  X1 + (X2 - X1)*(S - G1 - T)/
                          (G2 - G1 + 2L*S)
         ELSE IF KASE = 1 THEN X2 + 4L*(X2 - X1)
         ELSE IF KASE = 2 THEN LBOUND
         ELSE (X1 + X2)/2L;

COMMENT: CHECK THAT ROUNDING ERROR HAS NOT CAUSED XK TO
         LIE OUTSIDE PRESCRIBED BOUNDS;

   IF (XK < LBOUND) AND (KASE ~= 2) THEN
```

```
         XK:= (IF KASE = 1 THEN X2 + 4L*(X2 - X1)
               ELSE (X1 + X2)/2L)
      ELSE IF (XK > UBOUND) AND (KASE ¬= 1) THEN
         XK:= (IF KASE = 2 THEN LBOUND
               ELSE (X1 + X2)/2L);

CALCULATEACCEPTABLEXK:

      T:= X2 - X1;
      IF (XK= X1) OR ( XK = X2) THEN
         BEGIN
            IF (CLDF - FMIN) > GTEST1*(NEWALPHA + XMIN) THEN
               GOTO TERMINATESEARCH;
            XK:= (X1 + X2)/2L;
         END
      ELSE IF XK < X1 THEN
         BEGIN
            COMMENT: EXTRAPOLATION IN DIRECTION X2-X1;
            S:= X1 - LBOUND;
            T:= IF T < S THEN X1 - 0.5L*LONGSQRT(T*S)
                ELSE X1 - (5L*(0.1L + S/T)*S)/11L;
            IF XK < T THEN XK:= T;
         END
      ELSE IF XK > X2 THEN
         BEGIN
            COMMENT: EXTRAPOLATION IN DIRECTION X1-X2;
            S:= IF UBOUND = X2 THEN 64L*T
                ELSE UBOUND - X2;
            T:= IF T < S THEN X2 + 0.5L*LONGSQRT(T*S)
                ELSE X2 + (5L*(0.1L + S/T)*S)/11L;
            IF XK > T THEN XK:= T;
         END;

EVALUATEFK:

      IF XK > MAXALPHA THEN XK:= MAXALPHA;
      T:= NEWALPHA + XK;
      FOR I:= 1 UNTIL N DO  Z(I):= X(I) + T*P(I);
      FUN( N, Z, FK );
      FNUM:= FNUM + 1;

   COMMENT: ORDER X1 AND X2;

      IF XK < X1 THEN
         BEGIN
            IF FK <= F2 THEN
               BEGIN
                  X2:= X1;
                  F2:= F1;
                  G2:= G1;
                  X1:= XK;
                  F1:= FK;
               END
            ELSE
               BEGIN
                  LBOUND:= XK;
                  GOTO CALCULATEACCEPTABLEXK;
               END
```

```
            END
        ELSE IF XK > X2 THEN
           BEGIN
              IF FK <= F1 THEN
                 BEGIN
                    X1:= X2;
                    F1:= F2;
                    G1:= G2;
                    X2:= XK;
                    F2:= FK;
                 END
              ELSE
                 BEGIN
                    UBOUND:= XK;
                    GOTO CALCULATEACCEPTABLEXK;
                 END
           END
        ELSE IF F2 > F1 THEN
           BEGIN
              X2:= XK;
              F2:= FK;
           END
        ELSE
           BEGIN
              X1:= XK;
              F1:= FK;
           END;

        NEWALPHA:= T;
        GOTO ITERATE;

  TERMINATESEARCH:

        IF FMIN >= OLDF THEN SUCCESSFULSEARCH:= FALSE
        ELSE
           BEGIN
              COMMENT: LOWER POINT FOUND;
              NEWALPHA:= ALPHA:= IF MAXALPHA = XMIN THEN MAXSTEP
                                 ELSE NEWALPHA + XMIN;
              FOR I:= 1 UNTIL N DO  X(I):= X(I) + NEWALPHA*P(I);
              F:= FMIN;
              SUCCESSFULSEARCH:= TRUE;
           END;
     END DELINSEARCH;

  COMMENT: ****************************************************************;

     LONG REAL PROCEDURE DOT( INTEGER VALUE   N;
                               LONG REAL ARRAY A, B(*) );
     BEGIN

     COMMENT: PROCEDURE DOT CALCULATES THE INNER PRODUCT OF THE
              1*N VECTORS A, B.  THE BODY OF THIS PROCEDURE SHOULD
              BE WRITTEN IN MACHINE CODE;

        LONG REAL SUM;

        SUM:= 0L;
```

```
      FOR I:= 1 UNTIL N DO  SUM:= SUM + A(I)*B(I);
      SUM
   END DOT;

COMMENT: ***************************************************************;

   PROCEDURE INITIALIZEALPHA;
   BEGIN

   COMMENT: CALCULATES INITIAL STEP FOR THE LINEAR SEARCH PROCEDURE;

      ALPHA:= 2L*ABS ((NEWF - FM)/GTP);
      IF (ALPHA > 1L) OR (NEWF - FM < MACHEPS) THEN ALPHA:= 1L;

   END INITIALIZEALPHA;

COMMENT: ***************************************************************;

   PROCEDURE LDLTSOL( INTEGER VALUE   N;
                      LONG REAL ARRAY L, D, B, X(*) );
   BEGIN

   COMMENT: PROCEDURE LDLTSOL SOLVES  LDUX = B, WHERE D IS DIAGONAL, AND
            U IS THE TRANSPOSE OF THE UNIT LOWER TRIANGULAR MATRIX L.
            L IS STORED BY ROWS WITH ITS DIAGONAL ELEMENTS OMITTED
            IN THE I=N(N-1)/2 ARRAY L(I), I=1(1)N(N-1)/2.  THE MATRIX D
            OCCUPIES THE N ELEMENTS OF THE ARRAY D(I), I= 1(1)N.  THE
            SOLUTION AND RIGHT-HAND-SIDE VECTORS ARE STORED IN X(I) AND
            B(I) RESPECTIVELY, WHERE I=1(1)N.  THE BODY OF THIS PROCED-
            URE SHOULD BE WRITTEN IN MACHINE CODE;

      INTEGER   R, S, T;
      LONG REAL SUM;

      R:= 1;
      FOR I:= 1 UNTIL N DO
         BEGIN
            SUM:= B(I);
            T:= I - 1;
            FOR K:= 1 UNTIL T DO
               BEGIN
                  SUM:= SUM - X(K)*L(R);
                  R:= R + 1;
               END KLOOP;
            X(I):= SUM;
         END ILOOP;

      FOR I:= N STEP -1 UNTIL 1 DO
         BEGIN
            S:= R;
            R:= R - 1;
            T:= I + 1;
            SUM:= X(I)/D(I);
            FOR K:= N STEP -1 UNTIL T DO
               BEGIN
                  SUM:= SUM - X(K)*L(S);
                  S:= S + 2 - K;
               END KLOOP;
```

```
            X(I):= SUM;                                               ONMDE371
         END IDOOP;                                                   ONMDE372
      END LDLTSOL;                                                    ONMDE373
                                                                      ONMDE374
COMMENT: ============================================================;ONMDE375
                                                                      ONMDE376
      PROCEDURE MODIFYCHOLESKYFACTORS( INTEGER VALUE   N;             ONMDE377
                                       LONG REAL VALUE S;             ONMDE378
                                       LONG REAL ARRAY Z, L, D(*) );  ONMDE379
      BEGIN                                                           ONMDE380
                                                                      ONMDE381
      COMMENT: THIS PROCEDURE FORMS THE CHOLESKY FACTORIZATION OF THE ONMDE382
               MATRIX LDU + SZW, WHERE U IS THE TRANSPOSE OF L,       ONMDE383
                                       W IS THE TRANSPOSE OF Z,       ONMDE384
                                       S IS A SCALAR, AND             ONMDE385
                                       D IS A DIAGONAL MATRIX.        ONMDE386
               THE MATRIX L IS STORED ROW BY ROW IN THE 1*N(N-1)/2    ONMDE387
               ARRAY L(I), I=1(1)N(N-1)/2, WITH THE UNIT DIAGONAL OMITTED. ONMDE388
               THE MATRIX D AND VECTOR Z ARE STORED IN THE 1*N ARRAYS D(I) ONMDE389
               AND Z(I), I=1(1)N, RESPECTIVELY.  BOTH L AND D ARE OVER-  ONMDE390
               WRITTEN WITH THE CORRESPONDING FACTORS OF THE MODIFIED ONMDE391
               MATRICES.  THE VALUES OF S AND Z ARE NOT RETAINED.  THE ONMDE392
               PROCEDURE ENSURES THAT THE NEW MATRIX IS POSITIVE DEFINITE. ONMDE393
               THIS PROCEDURE BODY SHOULD BE WRITTEN IN MACHINE CODE; ONMDE394
                                                                      ONMDE395
         INTEGER   J, IA, IC;                                         ONMDE396
         LONG REAL XA, XB, XC, XD, XE, XF, XG, D1, P1, PTP, BETA, SIGMA; ONMDE397
         LONG REAL ARRAY W(1::N);                                     ONMDE398
                                                                      ONMDE399
         PTP:= 0L;                                                    ONMDE400
         IA :=  1;                                                    ONMDE401
         FOR IB:= 1 UNTIL N DO                                        ONMDE402
            BEGIN                                                     ONMDE403
               IC:= IB - 1;                                           ONMDE404
               XC:= Z(IB);                                            ONMDE405
               FOR ID:= 1 UNTIL IC DO                                 ONMDE406
                  BEGIN                                               ONMDE407
                     XC:= XC - W(ID)*L(IA);                           ONMDE408
                     IA:= IA + 1;                                     ONMDE409
                  END IDLOOP;                                         ONMDE410
                                                                      ONMDE411
               W(IB):= XC;                                            ONMDE412
               PTP:= PTP + XC*XC/D(IB);                               ONMDE413
            END IBLOOP;                                               ONMDE414
                                                                      ONMDE415
      COMMENT: IF S*PTP + 1 < 0, THE MODIFIED MATRIX IS INDEFINITE.  SIGMA ONMDE416
               IS REPLACED BY A QUANTITY WHICH ENSURES THAT THE MODIFIED ONMDE417
               MATRIX IS POSITIVE DEFINITE REGARDLESS OF SUBSEQUENT   ONMDE418
               ROUNDING ERROR;                                        ONMDE419
                                                                      ONMDE420
         SIGMA:= S*PTP;                                               ONMDE421
         IF SIGMA < -1L THEN SIGMA:= -SIGMA;                          ONMDE422
         SIGMA:= -S/( 1L + LONGSQRT( 1L + SIGMA ) );                  ONMDE423
         IA:= 0;                                                      ONMDE424
         FOR I:= 1 UNTIL N DO                                         ONMDE425
            BEGIN                                                     ONMDE426
               IA:= IA + I;             D1:= D(I);                    ONMDE427
               P1:= W(I);               XF:= P1/D1;                   ONMDE428
```

```
            XC:= XF*P1;            XB:= SIGMA*PTP;                        QNMDE429
            XA:= SIGMA*XC;         XG:= XA - 1L;                          QNMDE430
            PTP:= PTP - XC;        XE:= XA*(XB - XA);                     QNMDE431
                                                                          QNMDE432
            IF XE > 0L THEN                                               QNMDE433
               BEGIN                                                      QNMDE434
                  XE:= XE + XG*XG;                                        QNMDE435
                  XD:= IF XA <= 1L THEN -LONGSQRT(XE)                     QNMDE436
                                   ELSE  LONGSQRT(XE);                    QNMDE437
               END                                                        QNMDE438
            ELSE                                                          QNMDE439
               BEGIN                                                      QNMDE440
                  XE:= XG*XG;                                             QNMDE441
                  XD:= XG;                                                QNMDE442
               END;                                                       QNMDE443
                                                                          QNMDE444
            BETA:= (XB - 2L)*SIGMA*XF/XE;                                 QNMDE445
            SIGMA:= SIGMA*(1L - XD)/(XE + XD*XG);                         QNMDE446
            J:= IA;                                                       QNMDE447
            D(I):= XE*D1;                                                 QNMDE448
                                                                          QNMDE449
            FOR IB:= I+1 UNTIL N DO                                       QNMDE450
               BEGIN                                                      QNMDE451
                  XC:= L(J);                                              QNMDE452
                  XF:= Z(IB):= Z(IB) - P1*XC;                             QNMDE453
                  L(J):= BETA*XF + XC;                                    QNMDE454
                  J:= J + IB - 1;                                         QNMDE455
               END IBLOOP;                                                QNMDE456
         END ILOOP;                                                       QNMDE457
   END MODIFYCHOLESKYFACTORS;                                             QNMDE458
                                                                          QNMDE459
COMMENT: ============================================================;QNMDE460
                                                                          QNMDE461
   PROCEDURE MODIFYCONDITIONNUMBEROFDIAGONAL;                             QNMDE462
   BEGIN                                                                  QNMDE463
                                                                          QNMDE464
   COMMENT: THIS PROCEDURE BOUNDS THE SPECTRAL CONDITION NUMBER OF THE    QNMDE465
            DIAGONAL MATRIX D ASSOCIATED WITH THE CHOLESKY FACTORIZATIONQNMDE466
            OF THE APPROXIMATE HESSIAN;                                   QNMDE467
                                                                          QNMDE468
      LONG REAL LB;                                                       QNMDE469
                                                                          QNMDE470
      LB:= D(1);                                                          QNMDE471
      FOR I:= 2 UNTIL N DO                                                QNMDE472
         IF D(I) > LB THEN LB:= D(I);                                     QNMDE473
      LB:= LB/KBOUND;                                                     QNMDE474
      FOR I:= 1 UNTIL N DO                                                QNMDE475
         IF D(I) < LB THEN D(I):= LB;                                     QNMDE476
   END MODIFYCONDITIONNUMBEROFDIAGONAL;                                   QNMDE477
                                                                          QNMDE478
COMMENT: ============================================================;QNMDE479
                                                                          QNMDE480
                                                                          QNMDE481
                                                                          QNMDE482
COMMENT: ==================START OF MAIN PROCEDURE===================;QNMDE483
                                                                          QNMDE484
   COMMENT: FORM UNIT MATRIX IN L IF REQUIRED;                            QNMDE485
                                                                          QNMDE486
```

```
    IF LOADL THEN COMMENT: FORM UNIT MATRIX;                                ONMDE487
       BEGIN                                                                ONMDE488
          FOR I:= 1 UNTIL N DO D(I):= 1L;                                   ONMDE489
          J:= ( N*(N - 1) ) DIV 2;                                          ONMDE490
          FOR I:= 1 UNTIL J DO L(I):= 0L;                                   ONMDE491
       END;                                                                 ONMDE492
                                                                            ONMDE493
STARTOFMINIMIZATION:                                                        ONMDE494
                                                                            ONMDE495
    TOLSQ:= TOL*TOL;                                                        ONMDE496
    ALPHA:= 0L;                                                             ONMDE497
    ROOTMACHEPS:= LONGSQRT(MACHEPS);                                        ONMDE498
    CONV:= TRUE;                                                            ONMDE499
    FM:= F;                                                                 ONMDE500
    KBOUND:= 1L/(100L*LONGSQRT(N)*MACHEPS);                                 ONMDE501
    FGRAN(N,GKPLUSONE,X,NEWF,TRUE);
    FCOUNT:= GCOUNT:= 1;                                                    ONMDE504
    COUNT := 0;                                                             ONMDE505
                                                                            ONMDE506
CALCULATEDIRECTIONOFSEARCH:                                                 ONMDE507
                                                                            ONMDE508
    OLDF:= NEWF;                                                            ONMDE509
    FOR I:= 1 UNTIL N DO  GK(I):= -GKPLUSONE(I);                            ONMDE510
    LDLTSOL( N, L, D, GK, P );                                              ONMDE511
    NORMP:= LONGSQRT( DOT(N, P, P) ) + MACHEPS**2;                          ONMDE512
    GTP:= DOT( N, GKPLUSONE, P );                                           ONMDE513
                                                                            ONMDE514
IF PRINT THEN                                                               ONMDE515
MONITOR( N, COUNT, FCOUNT, NEWF, ALPHA, X, GKPLUSONE, L, D, P );            ONMDE516
                                                                            ONMDE517
    INITIALIZEALPHA;                                                        ONMDE518
                                                                            ONMDE519
    DELINSEARCH( N, FNUM, GNUM, SUCCESSFULSEARCH, ETA,                      ONMDE520
                 ROOTMACHEPS*'-1/NORMP, MAXSTEP/NORMP,                      ONMDE521
                 NEWF, ALPHA, P, X, GKPLUSONE, FUN, GRAD ,FGRAN);           ONMDE522
                                                                            ONMDE523
    FCOUNT:= FCOUNT + FNUM;                                                 ONMDE524
    GCOUNT:= GCOUNT + GNUM;                                                 ONMDE525
    COUNT := COUNT + 1;                                                     ONMDE526
    IF ¬SUCCESSFULSEARCH THEN GOTO SETCONV;                                 ONMDE527
                                                                            ONMDE528
COMMENT: COMDFP MODIFICATION RULE;                                          ONMDE529
                                                                            ONMDE530
    GTP1:= DOT( N, GKPLUSONE, P );                                          ONMDE531
    FOR I:= 1 UNTIL N DO  P(I):= GKPLUSONE(I) + GK(I);                      ONMDE532
    MODIFYCHOLESKYFACTORS( N, 1L/(ALPHA*(GTP1 - GTP)), P, L, D );           ONMDE533
    MODIFYCHOLESKYFACTORS( N, 1L/GTP, GK, L, D );                           ONMDE534
    MODIFYCONDITIONNUMBEROFDIAGONAL;                                        ONMDE535
                                                                            ONMDE536
COMMENT: OVERALL CONVERGENCE CRITERION;                                     ONMDE537
                                                                            ONMDE538
    IF DOT(N, GKPLUSONE, GKPLUSONE) < TOLSQ THEN GOTO PERFORMLOCALSEARCH;ONMDE539
    IF OLDF > NEWF THEN GOTO CALCULATEDIRECTIONOFSEARCH;                    ONMDE540
                                                                            ONMDE541
SETCONV:                                                                    ONMDE542
                                                                            ONMDE543
    CONV:= FALSE;                                                           ONMDE544
                                                                            ONMDE545
```

```
PERFORMLOCALSEARCH:

MONITOR( N, COUNT, FCOUNT, NEWF, ALPHA, X, GKPLUSONE, L, D, P );
   WRITE("LOCAL SEARCH STARTED");
   OLDF:= NEWF;

COMMENT: TAKE RANDOM STEP;

   FOR I:= 1 UNTIL N DO  Y(I):= X(I) + ROOTMACHEPS;
   FGRAN(N,GK,Y,NEWF,TRUE);
   IF(NCOUNT>LIMIT) THEN GO TO PREMATUREEJECTION;
   FCOUNT:= FCOUNT + 1;   GCOUNT:= GCOUNT + 1;

COMMENT: CALCULATE ORTHOGONAL DIRECTION AT Y;

   P(1):= ROOTMACHEPS;
   FOR I:= 2 UNTIL N DO  P(I):= -P(I-1);
   IF ODD(N) THEN P(N):= 0L;

   FOR J:= 1 UNTIL 2 DO
      BEGIN
         IF J = 2 THEN
            BEGIN
               FOR I:= 1 UNTIL N DO  P(I):= X(I) - Y(I);
               IF NEWF > OLDF THEN
                  BEGIN
                     NEWF:= OLDF;
                     FOR I:= 1 UNTIL N DO
                        BEGIN
                           GK(I):= GKPLUSONE(I);
                           Y(I):= X(I);
                        END ILOOP;
                  END
            END;
         GTP:= DOT( N, GK, P );

COMMENT: ASCERTAIN DOWNHILL DIRECTION FOR LINEAR SEARCH;

         IF GTP > 0L THEN
            BEGIN
               GTP:= -GTP;
               FOR I:= 1 UNTIL N DO  P(I):= -P(I);
            END;
         NORMP:= LONGSQRT( DOT(N,P,P) ) + MACHEPS**2;

         INITIALIZEALPHA;
         DELINSEARCH( N, FNUM, GNUM, SUCCESSFULSEARCH, 0L,
                      ROOTMACHEPS**-1/NORMP, MAXSTEP/NORMP,
                      NEWF, ALPHA, P, Y, GK, FUN, GRAD ,FGRAN);

         FCOUNT:= FCOUNT + FNUM;
         GCOUNT:= GCOUNT + GNUM;
         COUNT := COUNT + 1;
      END JLOOP;

   IF NEWF < OLDF THEN
      BEGIN
         FOR I:= 1 UNTIL N DO
```

```
            BEGIN                                            QNMDE604
               X(I):= Y(I);                                  QNMDE605
               GKPLUSONE(I):= GK(I);                         QNMDE606
            END;                                             QNMDE607
         OLDF:= NEWF;                                        QNMDE608
         IF ( DOT(N,GK,GK) > TOLSQ ) OR ¬CONV THEN           QNMDE609
            BEGIN                                            QNMDE610
               CONV:= TRUE;                                  QNMDE611
               GOTO CALCULATEDIRECTIONOFSEARCH;              QNMDE612
            END;                                             QNMDE613
      END;                                                   QNMDE614
                                                             QNMDE615
   F:= OLDF;                                                 QNMDE616
   FTOTAL:= FCOUNT;                                          QNMDE617
   GTOTAL:= GCOUNT;                                          QNMDE618
   ITNUM := COUNT ;                                          QNMDE619
END QNMDER;                                                  QNMDE620
                                                             QNMDE621
```

```
PROCEDURE FUN(INTEGER VALUE N;
               LONG REAL ARRAY X(*);
               LONG REAL RESULT F);
BEGIN
LONG REAL ARRAY G(1::N);
LOGICAL GRADYESNO;
GRADYESNO:=FALSE;
FGBAN(N,G,X,F,GRADYESNO);
   IF(NCOUNT>LIMIT) THEN GO TO PREMATUREEJECTION;
END;

PROCEDURE GRAD(INTEGER VALUE N;
                LONG REAL ARRAY X,G(*) );
BEGIN
LONG REAL F;
LOGICAL GRADYESNO;
GRADYESNO:=TRUE;
FGBAN(N,G,X,F,GRADYESNO);
   IF(NCOUNT>LIMIT) THEN GO TO PREMATUREEJECTION;
END;

PROCEDURE FGBAN( INTEGER VALUE N;
                  LONG REAL ARRAY G,X(*);
                  LONG REAL RESULT F;
                  LOGICAL VALUE GRADYESNO );
BEGIN

   COMMENT: PROCEDURE FGBAN COMPUTES THE PERFORMANCE FUNCTION F
            FOR A TWO VARIABLE LADDER FILTER, FOR USE IN OPTIMIZATION
            PROCEDURE CNMOFR. THE PERFORMANCE CRITERION IS LEAST P'TH
            AS DEFINED BY BANDLER AND CHARALAMBOUS. THE GRADIENT IS ALSO
            COMPUTED IF LOGICAL VARIABLE GRADYESNO=TRUE;

   INTEGER NS,ISPEC,IQ;
   LONG REAL A,MOLD,M,VM,PM1,EP1,HP35,HP80,F1;
   LOGICAL BNSAT;
   LONG COMPLEX ARRAY S(1::2);
   LONG COMPLEX ARRAY Y,Z(1::(N1+1) DIV 2);
   LONG COMPLEX ARRAY VFOR,IFOR,VREV,IREV(1::(N1+1) DIV 2+1);
   LONG COMPLEX ARRAY SENS(0::N+1);
   LONG COMPLEX V,I1,VTRANS,ITRANS,B;
   LONG REAL ARRAY RSENS(1::N);
   LONG COMPLEX ARRAY TG,T(1::2);
   LONG COMPLEX SUM,SUM1,SUM2;
   REAL RN;
   INTEGER M1;
   LONG REAL SUMX;
   LONG COMPLEX FACT;

   COMMENT: INITIALIZE CONSTANTS AND VARIABLES;
   NS:=(N1-1) DIV 2;
   NCOUNT:=NCOUNT+1;
   A:=1L+X(N1);
   MOLD:=MMOLD;

   COMMENT: LOGICAL VARIABLE BNSAT INDICATES WHETHER OR NOT RESPONSE
            BOUNDS HAVE BEEN SATISFIED. BNSAT=TRUE INDICATES THAT
            THEY HAVE NOT;
```

```
BNSAT:=(MMOLD>0L);
ISPEC:=0;
IF(¬BNSAT) THEN ISPEC:=1;

EVALUATEF:

M:=-'75L;
IQ:=IF(MOLD>=0L) THEN IP ELSE -IP;
F:=0L;
IF GRADYESNO THEN
   FOR I:=1 UNTIL N DO G(I):=0L;

COMMENT: PERFORM SUMMATION OVER FRECUENCY POINTS;

FOR I:=1 UNTIL NR DO BEGIN
FOR J:= 1 UNTIL NTH(I) DO
BEGIN

   COMMENT: DEFINE COMPLEX FRECUENCIES;
   S(1):=LONGIMAG(W1(I,J));
   S(2):=LONGIMAG(W2(I,J));
IF(N1 > 1) THEN BEGIN
RN:=N-N1+1;
M1:=ROUND(SQRT(RN));
SUM2:=0L;
FOR KL:= 0 UNTIL M1-1 DO
BEGIN
SUM1:=0L;
FOR KN:= 0 UNTIL M1-2 DO
SUM1:=(SUM1+X(N-KL*M1-KN))*S(2);
IF(KL<M1-1) THEN
SUM2:=(SUM2+SUM1+X(N-KL*M1-M1+1))*S(1);
ELSE SUM2:=SUM2+SUM1+1L;
END;END;
ELSE SUM2:=1L;
   COMMENT: COMPUTE ARM IMPEDANCES AND ADMITTANCES;
   FOR NUM:=1 UNTIL NS DO
   BEGIN
      Z(NUM):=S(IND(2*NUM-1))*X(2*NUM-1);
      Y(NUM):=S(IND(2*NUM))*X(2*NUM)
   END;
   Z(1):=1L+Z(1);
Y(NS):=Y(NS)+X(N1);

   COMMENT: COMPUTE MAGNITUDE;
   VFOR(NS+1):=1L;  I1:=0L;
   FOR K:=1 UNTIL NS DO
   BEGIN
      COMMENT: CALCULATE FORWARD CURRENTS AND VOLTAGES;
      IFOR(NS-K+1):=VFOR(NS-K+2)*Y(NS-K+1)+I1;
      I1:=IFOR(NS-K+1);
      VFOR(NS-K+1):=VFOR(NS-K+2)+IFOR(NS-K+1)*Z(NS-K+1);
   END;
VTRANS:=SUM2/VFOR(1);
   VM:=LONGSQRT(LONGREALPART(VTRANS)**2+LONGIMAGPART(VTRANS)**2)*A;

   COMMENT: INCREMENT FUNCTION TERM;
   FOR K:= 1,2 DO
```

```
    BEGIN

       COMMENT: NO LOWER RESPONSE BOUND IN STOP BAND;
       IF( (K=2) AND (HHAT(I)=0L) ) THEN GO TO XIT;
       PM1:=(-1L)**K;

       COMMENT: CALCULATE ERROR FUNCTION AND UPDATE M;
       EP1:=PM1*U(I,J)*(VM-HHAT(I)+PM1*XI);
       M:=IF(M>(-EP1)) THEN M ELSE -EP1;

       COMMENT: IF BOUNDS HAVE BEEN SATISFIED BUT M>0, REDO FUNCTION
                EVALUATION WITH MOLD=.5;
       IF¬(BNSAT OR (M<0L)) THEN
       BEGIN
          BNSAT:=TRUE;
          MOLD:=.5L;
          GO TO EVALUATEF;
       END;

       COMMENT: IF RESPONSE BOUNDS HAVE NOT BEEN SATISFIED, IGNORE
                POINTS THAT SATISFY THE BOUND;
       IF(BNSAT AND (EP1>=0L) ) THEN GO TO XIT;

       COMMENT; INCREMENT FUNCTION TERM;
       HP35:=-EP1/MOLD;
       HP80:=HP35**(IC-1);
       F:=F+HP80*HP35;

       IF GRADYESNO THEN BEGIN
 IF( N1>1 ) THEN BEGIN
 SENS(N1):=1L/SUM2;
 FOR NUM:=N1+1 UNTIL N1+M1-1 DO
 SENS(NUM):=SENS(NUM-1)*S(2);
 FOR NUM:=N1+M1 UNTIL N DO
 SENS(NUM):=SENS(NUM-M1)*S(1);
 FOR NUM:= N1+1 UNTIL N DO
 RSENS(NUM):=VM*LONGREALPART(SENS(NUM));
 END;
          COMMENT: CALCULATE SENSITIVITIES AND INCREMENT GRADIENT
                   TERMS;
          IREV(1):=1L;  V:=0L;
          FOR L:= 1 UNTIL NS DO BEGIN
             COMMENT: CALCULATE REVERSE CURRENTS AND VOLTAGES;
             VREV(L):=-IREV(L)*Z(L)+V;
             V:=VREV(L);
             IREV(L+1):=IREV(L)-VREV(L)*Y(L)
          END;
          ITRANS:=1L/IREV(NS+1);

          COMMENT; CALCULATE NORMALIZED SENSITIVITIES;
          FOR L:= 1 UNTIL NS DO BEGIN
             SENS(2*L):=VFOR(L+1)*VREV(L)*ITRANS;
             SENS(2*L-1):=-IFOR(L)*IREV(L)*ITRANS;
          END;

 FOR NUM:=1 UNTIL N1-1 DO
          RSENS(NUM):=VM*LONGREALPART(SENS(NUM)*S(INC(NUM)));
          RSENS(N1):=VM*LONGREALPART(SENS(N1-1)+1L/A);
```

```
            FOR NUM:= 1 UNTIL N DO
            G(NUM):=G(NUM)-HPRO*U(I,J)*RSENS(NUM)*PM1;
         END GRADCALC;
            ISPEC:=ISPEC+1;
         XIT:
      END INCREMENT;
   END INNERSUMLOOP;
   END OUTERSUMLOOP;

   COMMENT: HAVE RESPONSE BOUNDS BEEN SATISFIED ON THIS ITERATION?
            IF SO, PRINT CURRENT SOLUTION AND REDO FUNCTION EVALUATION.
            WITH NEW VALUE OF MOLD;
   IF(ISPEC=0) THEN BEGIN
      MMOLD:=IF(M<MMOLD) THEN M ELSE MMOLD;
      MOLD:=MMOLD;
      BNSAT:=FALSE;
      FOR I:= 1 UNTIL N DO XMIN(I):=X(I);
      WRITE(" "); WRITE("RESPONSE BOUNDS EXCEEDED"); WRITE(" ");
   WRITE(" ");WRITE("     CURRENT SOLUTION           GRADIENT");
   FOR I:= 1 UNTIL N DO WRITE(X(I),G(I));
   WRITE(" ");WRITE("FUNCTION VALUE=",F);
   WRITE("MAXIMUM DEVIATION FROM DESIRED RESPONSE=",XI+MMOLD);
   MMOLD:=MMOLD+XI*.1L;
   XI:=XI*.9L;
   WRITE("NEW VALUE OF XI=",XI);WRITE(" ");
WRITE("NUMBER OF FUNCTION EVALUATIONS=",ACCUNT);
   ONMDER( N, FTOTAL, GTOTAL, ITNUM, LEADL, CONV, ETA, LONGEPSILON,
         TOL, *1, F, X, L, D, FUN, GRAD, FGRAN );
   GO TO PREMATUREEJECTION;
   END;
   COMMENT: CALCULATE F;
   F1:=LONGEXP((1L/IC-1L)*LONGLN(F));
   F:=MOLD*F1*F;
   IF GRADYESNO THEN
   BEGIN
      FOR NUM:= 1 UNTIL N DO G(NUM):=G(NUM)*F1;
   END;
   MMOLD:=IF(M<MMOLD) THEN M ELSE MMOLD;
   IF(MMOLD=M) THEN
      FOR I:= 1 UNTIL N DO XMIN(I):=X(I);
END FGRAN;


PROCEDURE MONITOR( INTEGER VALUE N;
                   INTEGER VALUE COUNT,FCOUNT;
                   LONG REAL VALUE NEWF,ALPHA;
                   LONG REAL ARRAY X,GKPLUSONE,L,C,P(*) );
BEGIN
   INTEGER K;
   LONG REAL DEV;

   WRITE("STATUS AT ITERATION #", COUNT);  WRITE(" ");
   WRITE("     CURRENT SOLUTION           GRADIENT                 ");
   WRITEON("DIRECTION OF SEARCH");
   FOR I:= 1 UNTIL N DO WRITE( X(I), GKPLUSONE(I), P(I) );
   WRITE(" ");
   WRITE("APPROXIMATE MINIMUM VALUE, NEWF, =",NEWF); WRITE(" ");
   DEV:=XI+MMOLD;
```

```
    WRITE("MAXIMUM DEVIATION FROM IDEAL RESPONSE=",DEV);WRITE(" ");
    WRITE("NUMBER OF FUNCTION EVALUATICNS,FCOUNT, =",FCOUNT);
    WRITE(" ");
    WRITE("STEP LENGTH, ALPHA, =",ALPHA);

    WRITE(" ");
END MONITOR;


ETA:=.99L;
FOR I:=1 UNTIL N DO READON(X(I));
READ(R(NR1),R(NR1+1));
FOR I:= 1 UNTIL NR DO READON (NTH(I));
FOR I:=1 UNTIL N1-1 DO READON(IND(I));
READ(XI);
READ(LIMIT);
READ(TOL);
FOR I:= 1 UNTIL NR1-1 DC
BEGIN
R(I):=-R(NR1)*LONGCOS((1L+I/NR1)*PI/2L);
HHAT(I):=1L;
END;
HHAT(NR1):=1L; HHAT(NR1+1):=0L;
FOR I:= 1 UNTIL NR2 DO
BEGIN
R(NR1+I):=R(NR1+1)+(I-1)*(.99L-R(NR1+1))/(NR2-1);
HHAT(NR1+I):=0L;
END;
    LOADL:=TRUE;

MMOLD:=1L;
NCOUNT:=0;
FOR I:= 1 UNTIL NR DO
BEGIN
    FOR J:= 1 UNTIL NTH(I) DO
    BEGIN
       TH:=PI/2L+PI*(1-J)/NTH(I);
    U(I,J):=1L;
       W1(I,J):=DTAN(R(I)*PI*LONGCOS(TH)/2L);
       W2(I,J):=DTAN(R(I)*PI*LONGSIN(TH)/2L);
    END;
END;

FGBAN(N,G,X,F,TRUE);
WRITE(" ");WRITE("LEAST P'TH APPROXIMATICN FOR TWO DIMENSIONAL LOW PASS
FILTER: P=",IP);
WRITE("BANDLER'S PERFORMANCE FUNCTIONAL WITH XI=",XI);
WRITE(" ");WRITE("FUNCTION EVALUATIONS DONE AT FOLLOWING POINTS");
WRITE(" ");WRITE("          RADIAL          NTH          WEIGHTING");
FOR I:= 1 UNTIL NR DO WRITE(R(I),NTH(I),U(I,1));
WRITE(" ");WRITE("     STARTING POINT          GRADIENT");
FOR I:= 1 UNTIL N DO WRITE(X(I),G(I));
WRITE(" ");WRITE("STARTING FUNCTION VALUE=",F);
WRITE("MAXIMUM DEVIATION FROM DESIRED RESPONSE=",MMOLD+XI); WRITE(" ");
WRITE(" ");WRITE("TOL=",TOL);WRITE(" ");
F:=0L;
    WRITE( "INITIAL ESTIMATE OF FMIN =", F ); WRITE(" ");
    WRITE( "LINEAR SEARCH CRITERION, ETA, =", ETA); WRITE(" ");
```

```
    CNMDER( N, FTOTAL, GTOTAL, ITNUM, LOADL, CONV, ETA, LONGEPSILON,
           TOL, '11, F, X, L, D, FUN, GRAD, FGBAN );
GO TO LOOPOUT;
    PREMATUREEJECTION:
    FGBAN(N,G,XMIN,F,TRUE);
    DEV:=XI+MMOLD;
    WRITE(" ");WRITE("TERMINATED DUE TO EXCESSIVE FUNCTION EVALUATIONS");
    WRITE(" ");WRITE("NUMBER OF FUNCTION EVALUATIONS WAS ", NCOUNT);
    WRITE(" ");WRITE("    CURRENT SOLUTION          GRADIENT");
    FOR I:= 1 UNTIL N DO WRITE(XMIN(I),G(I));
    WRITE(" ");WRITE("FUNCTION VALUE=",F);
    WRITE("MAXIMUM DEVIATION FROM DESIRED RESPONSE=",DEV);
LOOPOUT:
WRITE(" ");WRITE("OPTIMIZATION TIME IN SECONDS=",TIME(1)/60.);
IOCONTROL(3);
END;
END.
```

REFERENCES

1. Anderson, B.D.O.; Jury, E.I.; - "Stability Test for Two-Dimensional Recursive Filters", IEEE Trans. on Audio and Electroacoustics, Vol. AU-21, #4, August 1973.

2. Andrews, H.C.; - "Computer Techniques in Picture Processing", New York, Academic Press 1970.

3. Ansell, H.G.; - "On Certain Two-Variable Generalizations of Circuit Theory, With Applications to Networks of Transmission Lines & Lumped Reactances", IEEE Trans. Circuit Theory, Vol. CT-11, pp. 214-223, June 1964.

4. Bandler, J.W.; Charalambous, C: - "Practical Least p^{th} Optimization of Networks", IEEE Trans. on Microwave Theory & Techniques, Vol. MTT-20, #12, pp. 834-840, December 1972.

5. Barrodale, I.; Powell, M.J.D.; Roberts, F.D.K.; - "The Differential Correction Algorithm for Rational L_{∞} Approximation", SIAM J. Numer. Anal., Vol. 9, #3, September 1972, pp. 493-504.

6. Battacharyya, B.K.; - "Two-dimensional Harmonic Analysis as a Tool for Magnetic Interpretation", Geophysics, Vol. 30, pp. 829-857, 1965.

7. Bednar, J.B.; Farmer, C.H.; - "Implementation of Chebyshev Digital Filter Design". Private communication.

8. Belevitch, V; - "Classical Network Theory", Holden-Day, 1968 (San Francisco).

9. Blostein, M.L.; - "Sensitivity Analysis of Linear Systems in the Frequency Domain". McGill report.

10. Bose, N.K.; Jury, E.I.; - "Positivity and Stability Test for Multidimensional Filters (Discrete-Continuous)" IEEE Trans. on Acoustics, Speech, and Signal Processing, Vol. ASSP-22, #3, June 1974, pp. 174-180.

11. Chen, C.T.; - "Introduction to Linear System Theory". Holt Rinehart and Winston, New York; p. 313.

12. Clement, W.G.; - "Basic Principles of Two-Dimensional Digital Filtering", Geophysical Prospecting, Vol. 21, pp. 125-145, 1973.

13. Corrington, M.S.; - "Design of Two-Dimensional Recursive Filters", Proc. 1973, IEEE Symposium on Circuit Theory.

14. Costà, J.M.; Venetsanopoulos, A.N.; - "Two-Dimensional Digital Filtering Applied to Picture Processing", U of T Dept. of Elect. Eng. Comm. Group, Technical Report #5, July 1973.

15. Darley, E.K.; Davies, E.B.; - "The Analysis and Design of Two-Dimensional Filters for Two-Dimensional Data", Geophysical Prospecting, Vol. 15, pp. 383-406, 1967.

16. Dudgeon, D.E.; - "Recursive Filter Design Using Differential Correction", ScD Thesis, M.I.T., 1974, Ch. 8.

17. Fletcher, R; Grant, J.A.; Hebden, M.D.; - "Linear Minimax Approximation as the Limit of Best L_p - Approximations", SIAM J. Numer. Anal., Vol. 11, #1, March 1974, pp. 123-136.

18. Gill, P.E.; Murray, W.; Pitfield, R.A.; - "The Implementation of Two Revised Quasi-Newton Algorithms for Unconstrained Optimization" National Physical Laboratory - Report NAC 11, April 1972.

19. Gill, P.E.; Murray, W.; - "Quasi-Newton Methods for Unconstrained Optimization", J. Inst. Math & Appl. (GB), Vol. 9, #1, pp. 91-108 February 1972.

20. Gold, B; Rader, C.M.; - "Digital Processing of Signals", Lincoln Laboratory Publications, McGraw Hill, 1969.

21. Hall, E.L.; - "A Comparison of Computations for Spatial Frequency Filtering", Proc. IEEE, Vol. 60, #7, July 1972, pp. 887-891.

22. Hu, J.V.; Rabiner, L.R.; - "Design Techniques for Two-Dimensional Filters" IEEE Trans. Audio and Electroacoustics, Vol. AU-20, #4, October 1972, p. 249.

23. Huang, T.S.; - "Stability of Two-Dimensional Recursive Filters", IEEE Trans. Audio & Electro Acoustics, Vol. AU-20, #2, June 1972, p. 158.

24. Johnson, E.C.; - "Analyzing Ladder-Type Networks by a Quick Arithmetic Procedure", Electronics, Vol. 46, #23, November 8, 1973, pp. 113-116.

25. Jury, E.I.; - "Theory and Application of the Z-Transform Method", New York, Wiley, 1964.

26. Koga, T.; - "Synthesis of Finite Passive n-ports with Prescribed Positive Real Matrices of Several Variables", IEEE Trans. Circuit Theory, Vol. CT-15, pp. 2-23, March 1968.

27. Kuo, F.F.; Magnuson, W.G.; - "Computer Oriented Circuit Design", Prentice Hall, 1969.

28. Manry, M.T.; Aggarwal, J.K.; - "Picture Processing Using One-Dimensional Implementations of Discrete Planar Filters", IEEE Trans. on Acoustics, Speech, and Signal Processing, Vol. ASSP-22, #3, June 1974, pp. 164-173.

29. Maria, G.A.; Fahmy, M.M.; - "An l_p Design Technique for Two-Dimensional Digital Recursive Filters", IEEE Trans. on Acoustics, Speech and Signal Processing, Vol. ASSP-22, #1, pp. 15-22, February 1974.

30. Maria, G.A.; Fahmy, M.M.; - "On the Stability of Two-Dimensional Digital Filters", IEEE Trans. on Audio & Electro Acoustics, Vol. AU-21, #5, p. 470-2.

31. Murray, W., ed.; - "Numerical Methods for Unconstrained Optimization" Academic Press, 1972.

32. Ozaki, H.; - "Positive Real Functions of Several Variables and their Applications to Variable Networks", IRE Trans. Circuit Theory, Vol. CT-7, pp. 251-260, September 1960.

33. Papoulis, A.; - "Systems & Transforms with Applications in Optics", New York, McGraw-Hill, 1968.

34. Ramachandran, V; Rao, A.S.; - "A Multivariable Array and its Applications to Ladder Networks", IEEE Trans. Circuit Theory, Vol. CT-20, #5, September 1973, pp. 511-518.

35. Read, R.R.; Treitel, S.; - "The Stabilization of Two-Dimensional Recursive Filters Via the Discrete Hilbert Transform", IEEE Trans. on Geoscience Electronics, July 1973, GE-11, #3, pp. 153-160.

36. Read, R.R.; Treitel, S.; - "Addendum to The Stabilization of Two-Dimensional Recursive Filters Via the Discrete Hilbert Transform", October 1973, GE-11, #4, pp. 205-207.

37. Rice, J.R.; - "The Approximation of Functions", Vol. 2, Addison-Wesley 1969, Ch. 12.

38. Rosenfeld, A.; - "Picture Processing by Computer", Academic Press, N.Y. 1969.

39. Shanks, J.L.; Treitel, S.; Justice, J.H.; - "Stability and Synthesis of Two-Dimensional Recursive Filters", IEEE Trans. Audio & Electroacoustics, Vol. AU-20, #2, p. 115, June 1972.

40. Shanks, J.L.; Read, R.R.; - "Two-Dimensional Recursive Filters for Digital Processing", Proc. of 1973, IEEE International Symposium on Circuit Theory.

41. Swanton, D.; - "Linear Programming Design of Recursive Digital Filters", M.Eng. Thesis, McGill University, 1973.

42. Treitel, S.; Shanks, J.L.; - "The Design of Multistage Separable Planar Filters", IEEE Trans. Geosci. Elect. Vol. GE-9, p. 10, January 1971.

43. Zurflueh, E.G.; - "Applications of Two-Dimensional Linear Wavelength Filtering" Geophysics 32, pp. 1015-1035.