To the Graduate Council:

I am submitting herewith a thesis written by Toadsak Vongtanaanek entitled "Computer Programs for Designing Two-Dimensional Recursive Digital Filters via Digital Spectral Transformations." I have examined the final copy of this thesis for form and content and recommend that it be accepted in partial fulfillment of the requirements for the degree of Master of Science, with a major in Electrical Engineering.

Bruce W. Bomar

Bruce W. Bomar, Major Professor

We have read this thesis
and recommend its acceptance:

Basil N. Antar

Roy D Joseph

Accepted for the Council:

Lewminkel

Vice Provost
and Dean of the Graduate School

STATEMENT OF PERMISSION TO USE

In presenting this thesis in partial fulfillment of the requirements for a Master's degree at The University of Tennessee, Knoxville, I agree that the Library shall make it available to borrowers under rules of the Library. Brief quotations from this thesis are allowable without special permission, provided that accurate acknowledgment of source is made.

Permission for extensive quotation from or reproduction of this thesis may be granted by my major professor, or in his absence, by the Head of Interlibrary Services when, in the opinion of either, the proposed use of the material is for scholarly purposes. Any copying or use of the material in this thesis for financial gain shall not be allowed without my written permission.

Signature Todd Vanderwerker

Date 5/28/87

COMPUTER PROGRAMS FOR DESIGNING TWO-DIMENSIONAL RECURSIVE DIGITAL FILTERS VIA DIGITAL SPECTRAL TRANSFORMATIONS

A Thesis

Presented for the

Master of Science

Degree

The University of Tennessee, Knoxville

Toadsak Vongtanaanek

June 1987

To My Beloved Parents,

Nunta and Hong Yen Wong

"Wherever he may be,

the true man always returns home."

ACKNOWLEDGEMENTS

The author wishes to thank the members of his committee for their assistance in the preparation of this thesis. Grateful appreciation is expressed to Dr. Bruce Bomar, his major professor, for his encouragement, technical assistance, and the permission to use his 1-D IIR design computer program. The author would like to thank Dr. Roy Joseph and Dr. Basil Antar who served as members, for their helpful advice and guidance. The thesis would not be realized without their help.

ABSTRACT

FORTRAN computer programs are developed for designing two-dimensional (2-D) circularly symmetric recursive digital filters from a one-dimensional (1-D) lowpass prototype using digital spectral transformations (DST's). The DST technique was employed to take advantage of the readily available 1-D filter design software. Also, the DST design method requires no numerical iteration and is relatively simple to program.

A survey of available DST design techniques is given. A choice is then made from among the various techniques. A detailed discussion of the selected technique is given. This technique and its computer program implementation provide a 2-D lowpass digital filter from the 1-D lowpass prototype. A second computer program then applies two 1-D DST's to transform the 2-D lowpass filter to other 2-D filter types (lowpass with a different cut-off contour, highpass, bandpass, and bandstop). Several filter designs are carried out to verify that the filters obtained using this technique have magnitude contours of a proper shape to be of practical use.

The problem of realizing the filters designed is also briefly considered. The design programs are written to output the 2-D filter coefficients in the form needed for realizing the filter using a canonical real-gain Roesser's local state-space structure. This structure was selected because it is suitable for subsequent optimization with respect to finite wordlength effects.

TABLE OF CONTENTS

LIST OF TABLES

TABLE PAGE

LIST OF FIGURES

CHAPTER I

INTRODUCTION

1.1 Statement of Problem

Two-dimensional (2-D) digital filters have important applications in various engineering and scientific fields. For instance, 2-D digital processing is used in biomedicine to reduce low-frequency components in an X-ray image. In seismic work, 2-D digital filtering is necessary to extract noise and separate signals from different sources. The University of Tennessee Space Institute has research projects which call for 2-D filtering of data; it is carried out on a Vicom image analysis system which uses low-order 3×3 finite impulse response (FIR) filters. However, infinite impulse response (IIR) filters are more desirable because they can provide sharper transition regions, and can be efficiently implemented for low filter orders. In addition, the phase response of the IIR filter can be made exactly zero by processing the image twice with proper data orientations [14]. A zero-phase response is shown by Huang [28] to be very important in image processing because it is the phase, not the magnitude, that bears the resemblance of the image.

There are two basic IIR filter design methods: direct optimization in 2-D and transformation from 1-D. In a direct optimal design, a large amount of computation time and memory space is required because an optimization technique must be employed. In addition, either a stability check has to be performed every iteration or constraints must be placed on the optimization which are sufficient (but not necessary) to guarantee stability. Either approach is time-consuming and requires computationally intensive procedures. Furthermore, the optimization is usually

nonlinear and may not converge to a global optimum. Representative of designs which employ this type of optimization are those of Fahmy [18] and Higuchi [19].

The transformation approach produces a suboptimal filter by applying stable 1-D to 2-D spectral transformations (ST's) to a 1-D lowpass prototype. By definition, a stable spectral transformation is a complex mapping which transforms one stable rational transfer function into another with a different frequency response, while at the same time maintaining some desirable characteristics [14]. A great advantage of these transformation techniques is that they are relatively straightforward to apply and are not numerically intensive.

This thesis deals with the development of FORTRAN computer programs for designing 2-D recursive filters and corresponding efficient realizations. A circularly symmetric response was considered adequate for present needs. Numerically intensive and nonlinear optimization techniques were avoided in favor of a transformation technique, since a 1-D design program was already available. It was decided that the 2-D realization would be based on a canonical form of Roesser's local state-space model [21]. Although the canonical state-space structure may not possess the most desirable properties with regard to the finite word length effects of roundoff noise, coefficient quantization sensitivity, and limit cycles, other state-space structures with better characteristics can be easily obtained from the canonical form by applying an appropriate similarity transformation [22-24].

Numerous transformation methods exist [6-11,15] to design circularly symmetric lowpass recursive digital filters from a 1-D lowpass prototype. A survey of these methods was undertaken so that the best possible program could be developed. A program was then written to implement the method selected. An additional program was written to obtain other filter types, including lowpass with an arbitrary

cut-off frequency, highpass, bandpass, and bandstop, by applying 2-D to 2-D digital spectral transformations.

1.2 Background on Two-Dimensional Recursive Digital Filters

The digital transfer function of a 2-D linear, shift-invariant (LSI) recursive digital filter can be written as a ratio of polynomials in Z_1 and Z_2 as [1]

$$H(Z_1, Z_2) = \frac{\sum_{k=-\infty}^{\infty} \sum_{l=-\infty}^{\infty} b_{k,l} Z_1^{-k} Z_2^{-l}}{\sum_{k=-\infty}^{\infty} \sum_{l=-\infty}^{\infty} c_{k,l} Z_1^{-k} Z_2^{-l}} \tag{1.2.1}$$

The inverse Z-Transform of $H(Z_1, Z_2)$ is $h(m, n)$, known as the unit pulse response. The filter output can be expressed as the convolution of the input and the unit pulse response as

$$y(m, n) = \sum_{k=-\infty}^{\infty} \sum_{l=-\infty}^{\infty} h(k, l) u(m-k, n-l) \tag{1.2.2}$$

For a first quadrant quarter-plane LSI 2-D filter, the digital transfer function $H(Z_1, Z_2)$ in (1.2.1) reduces to

$$H(Z_1, Z_2) = \frac{\sum_{k_1=0}^{M_1} \sum_{k_2=0}^{M_2} b_{k_1,k_2} Z_1^{-k_1} Z_2^{-k_2}}{\sum_{k_1=0}^{N_1} \sum_{k_2=0}^{N_2} c_{k_1,k_2} Z_1^{-k_1} Z_2^{-k_2}} \tag{1.2.3}$$

The corresponding difference equation is then

$$y(m, n) = \sum_{k_1=0}^{M_1} \sum_{k_2=0}^{M_2} b_{k_1,k_2} u(m-k_1, n-k_2) - \sum_{\substack{k_1=0 \\ k_1-k_2 \neq 0}}^{N_1} \sum_{k_2=0}^{N_2} c_{k_1,k_2} y(m-k_1, n-k_2) \tag{1.2.4}$$

This shows that the output can be computed recursively from the input and previous outputs. The terms IIR and recursive, as well as FIR and nonrecursive will henceforth be used interchangeably.

1.3 Stability of 2-D Recursive Digital Filters

A necessary and sufficient condition for a 2-D first quadrant quarter-plane LSI digital filter to be bounded-input bounded-output (BIBO) stable is that [1]

$$\sum_{m=0}^{\infty}\sum_{n=0}^{\infty}|h(m,n)| < \infty \tag{1.3.1}$$

Also, a 2-D first quadrant quarter-plane LSI digital filter as in (1.2.3) is stable if (but not only if) there are no poles on or outside the unit bidisk ($|Z_1| = 1, |Z_2| = 1$) in the (Z_1, Z_2) plane (Shanks' Theorem) [5].

1.4 Survey of Transformation Design Methods

Authors have used various stable 1-D to 2-D ST's to obtain 2-D stable recursive filters. Shanks [5] developed a 2-D analog filter by rotating a 1-D analog filter by an angle β. Costa [6] designed a circularly symmetric lowpass digital filter by cascading Shanks' rotated filters with different angles distributed equally over 180 degrees. The 2-D digital filter is then obtained by applying double bilinear transformations. Since several filters must be cascaded to obtain a circularly symmetric response, it is difficult to control the cut-off contour using this technique: attenuation at other contours is also unavoidable. Goodman [7] pointed out that these rotated filters are generally BIBO unstable, and proposed a modified technique starting in the digital domain. The filter coefficients are perturbed to ensure stability which in turn changes the frequency response of the filter. The resulting filter also contains undesirable passbands at the corners. However, these undesirable passbands can be eliminated by cascading a separable rectangular filter.

Chang and Aggarwal [8] eliminated the instability by developing an elegant technique for rotating frequency responses of separable filters with transfer func-

tions having rational powers of Z. The resulting filter is separable and hence it can be realized more efficiently by a series connection of 1-D structures. The problem with these filters is that they have high-frequency attenuation in the vicinity of the Nyquist frequencies. Another drawback is that input/output data array interpolations must be performed to realize the filters.

Bernabo [9] designed a zero-phase circular filter as a cascade of four all-pole filters and a non-recursive filter using the well-known McClellan transform [12]. Iljima [10] designed octagonal digital filters obtained by cascading two 45 degree rotated filters and separable filters to approximate circular lowpass digital filters. The sole advantage of this design technique is that the filter can be realized by 1-D structures. The design, however, is limited to filters with cut-off frequencies of less than $\pi/\sqrt{2}$.

Nguyen and Swamy [11] designed, via the first-order McClellan transform, a 2-D zero-phase filter as a cascade of second-order section separable denominator filters. The technique takes full advantage of the denominator separability and the simplicity of the McClellan transform. Unfortunately the condition that the denominator is separable means that the 1-D prototype be critically damped, which in turn results in a critically damped response for the resulting filter.

Very recently, Harn and Shenoi [15] presented a design technique to approximate circular filters based upon the Digital Spectral Transformation (DST) proposed by Constantinides [17]. This technique requires neither coefficient optimization nor interpolation, and results in a lower number of coefficients. The filter is inherently stable if the 1-D prototype and the corresponding DST are stable. Another advantage is that the filter can be realized as a cascade of low-order subfilters, which in general is not possible for 2-D filters. As a result the filter can be more

efficiently realized with a reduction in quantization noise and coefficient truncation sensitivity. Because of these advantages, this technique was selected for implementation in the computer program for designing circularly symmetric lowpass recursive digital filters.

Pendergrass [13] developed a technique for designing different types of 2-D recursive filters (lowpass with a different cut-off frequency, highpass, bandpass and bandstop) from a 2-D lowpass recursive filter via 2-D to 2-D DST's. The technique of [13] was implemented in the computer program to get other filter types.

1.5 Thesis Organization

Chapter II deals with the theory and concepts behind the design of 2-D circularly symmetric lowpass recursive digital filter using the technique proposed by Harn [15]. This technique is selected over other techniques because it is relatively easy to implement, yields filters realizable as subfilters, and does not involve any optimization. Moreover, the technique can be easily extended to design elliptic and fan filters. Design examples are given to demonstrate the validity and practicality of the technique.

Chapter III explains how the other filter types such as highpass, bandpass, and bandstop can be designed by applying stable 2-D to 2-D DST functions [16] to a lowpass digital filter. Two practical design examples are also given.

Chapter IV discusses realization structures. The difference equation realization structure and a state-space structure based on Roesser's local state space model are both considered. The concept of recursive directions is explained and a generalized canonical real-gain state-space structure for realizing 2-D subfilters is given. The proper direction for recursing each subfilter is then described.

Chapter V describes the features of the implementation of the computer programs and their uses. Programs are given to design stable circularly symmetric lowpass recursive digital filters as discussed in Chapter II, and other filter types (highpass, bandpass, and bandstop) based upon 2-D to 2-D DST's as mentioned in Chapter III. The programs also are written to provide the coefficients in the state-space realization.

Chapter VI contains the summary, conclusions, and possible future extension of the work.

CHAPTER II

2-D CIRCULARLY SYMMETRIC LOWPASS RECURSIVE DIGITAL FILTER DESIGN VIA 1-D TO 2-D DIGITAL SPECTRAL TRANSFORMATION

2.1 Design Technique

Given a prescribed circular cut-off contour, the problem is to design a 2-D circularly symmetric lowpass recursive digital filter from a stable 1-D transfer function $G(Z)$. The filter is to give a best approximation to the prescribed contour. The method used [15] applies the following 1-D to 2-D digital spectral transformation to $G(Z)$:

$$\begin{aligned} Z^{-1} &= \left(\frac{Z_1^{-1} - a}{aZ_1^{-1} - 1}\right)\left(\frac{Z_2^{-1} - a}{aZ_2^{-1} - 1}\right) \\ &= H(Z_1)H(Z_2) \end{aligned} \tag{2.1.1}$$

2.2 Design Theory

The transformation in (2.1.1) provides a nonlinear mapping of the 1-D prototype response into two dimensions. The value of a is chosen so that $H(Z_1)H(Z_2)$ is stable ($|a| < 1$), and the resulting transformed filter $G[H(Z_1)H(Z_2)]$ best approximates the circular arc of the cut-off contour.

Before the design procedures are discussed further, it is useful to investigate what the application of the transformation does to $G(Z)$. To accomplish this, the transformation $Z^{-1} = H(Z_1)H(Z_2)$ can be broken down into two steps:

1) $Z^{-1} \leftarrow Z_1^{-1} Z_2^{-1}$

This transformation is equivalent to rotating the 1-D filter by 45 degrees counter-clockwise. Fig.2.2.1 and Fig.2.2.2 show the magnitude response of an ideal lowpass filter $F(Z_1)$ before and after the transformation is applied. Note that there are three passbands after the rotation. The introduction of the two passbands at the corners is due to the periodic nature of the original filter.

2) $Z_1^{-1} \leftarrow H(Z_1), \quad Z_2^{-1} \leftarrow H(Z_2)$

The purpose of this transformation is to map the straight contour line of the rotated filter to a curved (and approximating circular) nonlinear one. As mentioned previously, the shape of the contour line depends entirely on the value of a. Fig.2.2.3 shows the contour lines of the magnitude squared response of $G[H(Z_1)H(Z_2)]$ for the second-order lowpass Butterworth filter with a cut-off of 0.033π

$$G(Z) = \frac{.00257224 - 0.00514448Z^{-1} + .00257224Z^{-2}}{1 - 1.8515Z^{-1} - .861787Z^{-2}} \qquad (2.2.1)$$

with a = 0.9.

The 1-D IIR design program is a modified version of the one given in [3]. It should be noted that the contour line approximates a straight line for small ω, nearly a circular arc as ω increases , and more like a square outside. Since all the coefficients of the transformation used are real, the contour lines of the third quadrant are the mirror image of the ones in the first quadrant.

Since $G[H(Z_1)H(Z_2^{-1})]$ has circular arcs in the second and fourth quadrants when $G[H(Z_1)H(Z_2)]$ has circular ones in the first and third quadrants, a full circle can be realized by cascading $G[H(Z_1)H(Z_2)]G[H(Z_1)H(Z_2^{-1})]$

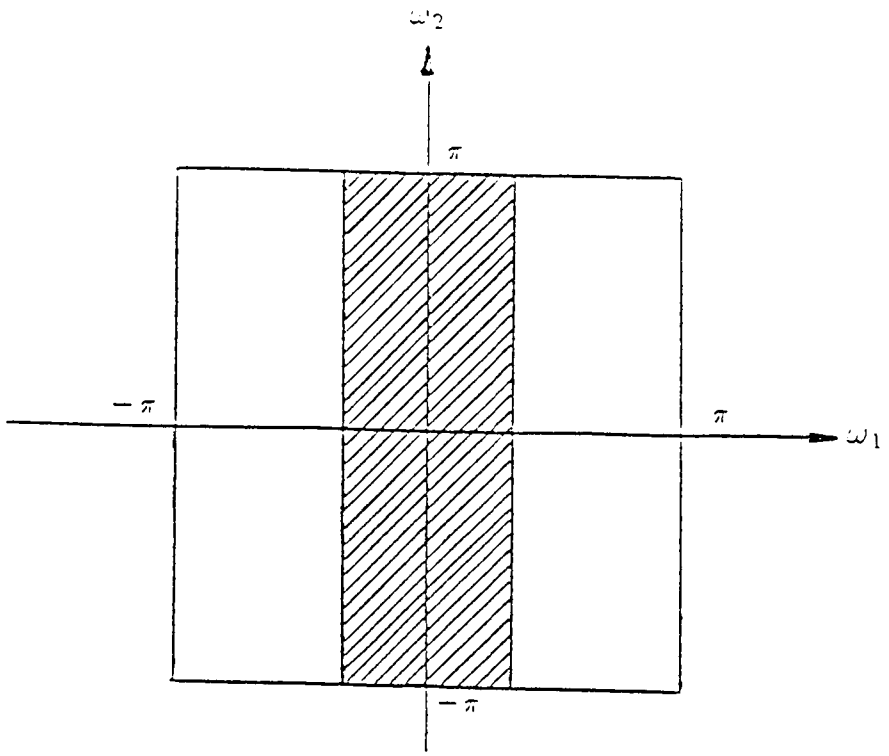


Fig. 2.2.1. 2-D Filter $F(Z_1)$ before Rotation

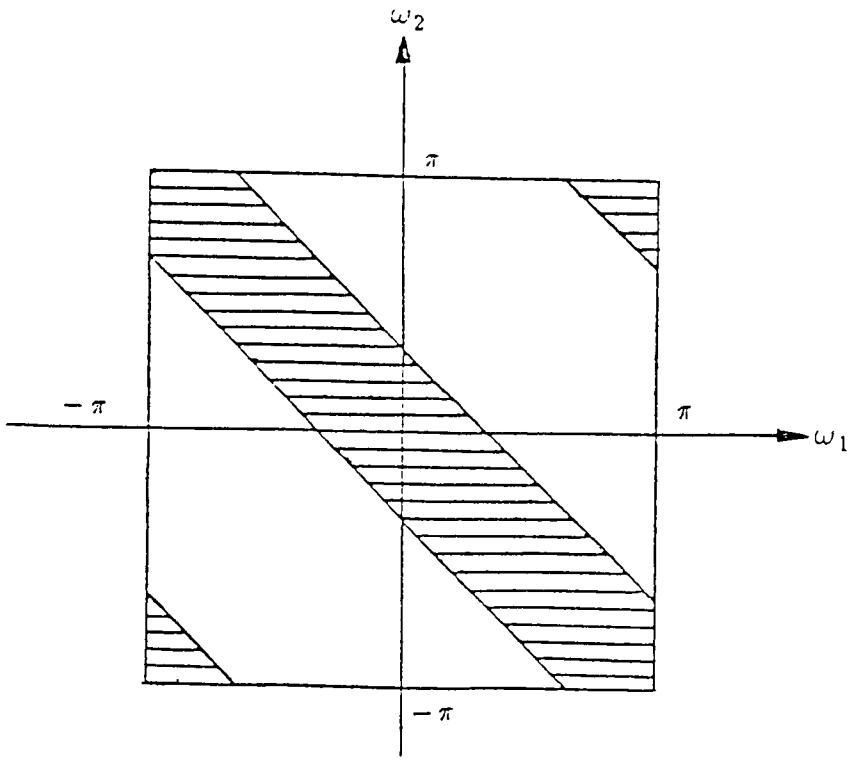


Fig. 2.2.2. 2-D Filter Rotated by 45 Degrees Counter-Clockwise, $F(Z_1, Z_2)$

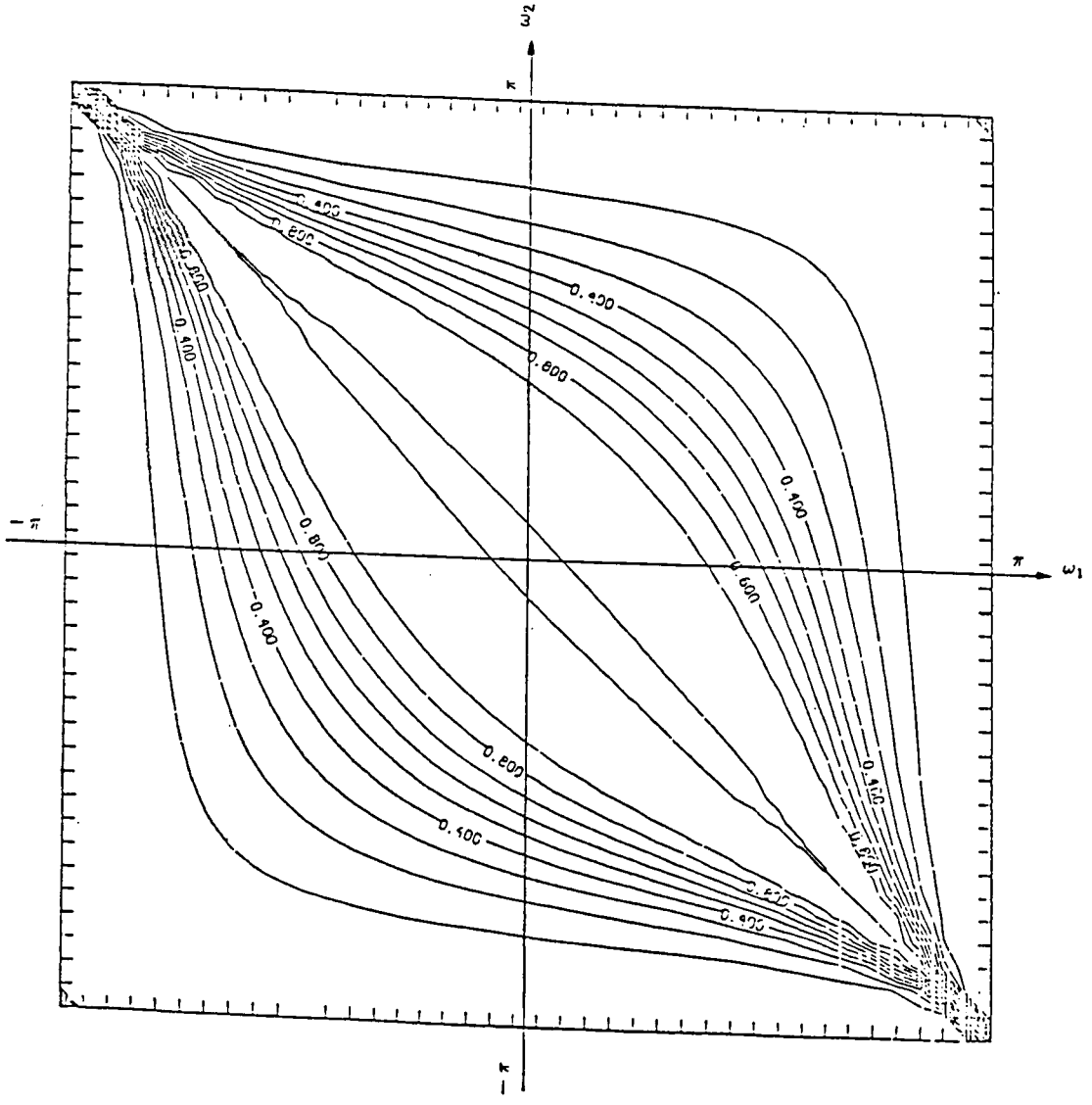


Fig. 2.2.3. Contour Lines of the Magnitude Squared Response of $G[H(Z_1)H(Z_2)]$

For example, Fig.2.2.4 shows the contour lines of the magnitude squared response of this product for the filter of (2.2.1). The reader should note that a stable implementation of $G[H(Z_1)H(Z_2^{-1})]$ is possible, as will be discussed in Chapter IV, by changing the direction of recursion in the second variable [32, pp.432-434].

The presence of the four small unwanted passbands at the corners in Fig.2.2.4 is due to the periodic nature of the filter as mentioned previously. They can be eliminated by cascading a separable filter $G[H(Z_1)]G[H(Z_2)]$. This gives the overall 2-D transfer function as:

$$T(Z_1, Z_2) = G[H(Z_1)H(Z_2)]G[H(Z_1)H(Z_2^{-1})]G[H(Z_1)]G[H(Z_2)] \qquad (2.2.2)$$

Fig.2.2.5 shows the contour lines of the magnitude squared response of $T(Z_1, Z_2)$ for the filter of (2.2.1). As we can see from Fig.2.2.5, in addition to the elimination of the undesirable corner passbands, the cascade of the separable filter also improves the overall response of the filter. A zero-phase digital filter, if desired, can be obtained by connecting $T(Z_1^{-1}, Z_2^{-1})$ with $T(Z_1, Z_2)$ either in parallel or cascade [14].

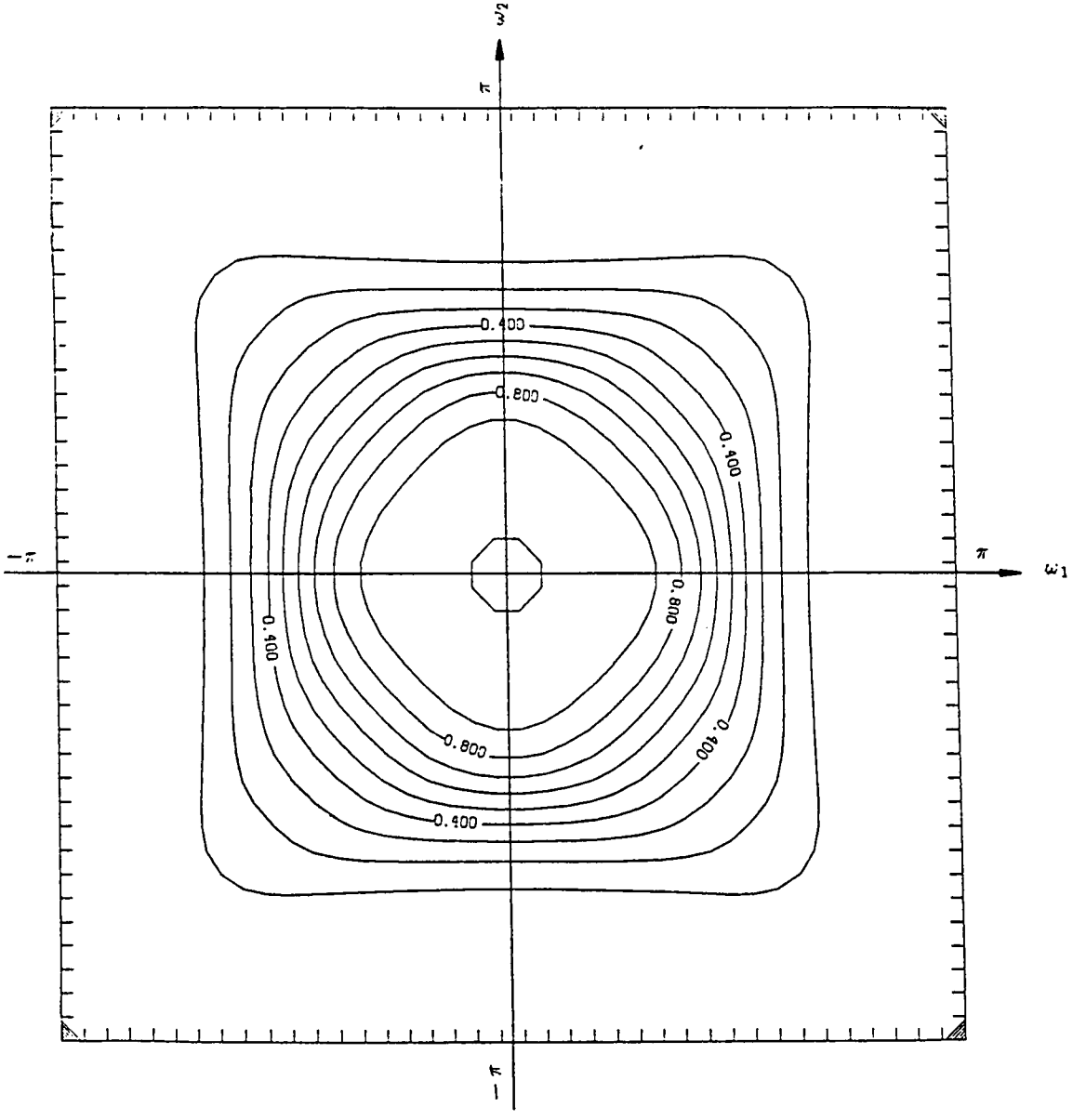


Fig. 2.2.4. Contour Lines of $G[H(Z_1)H(Z_2)]G[H(Z_1)H(Z_2^{-1})]$

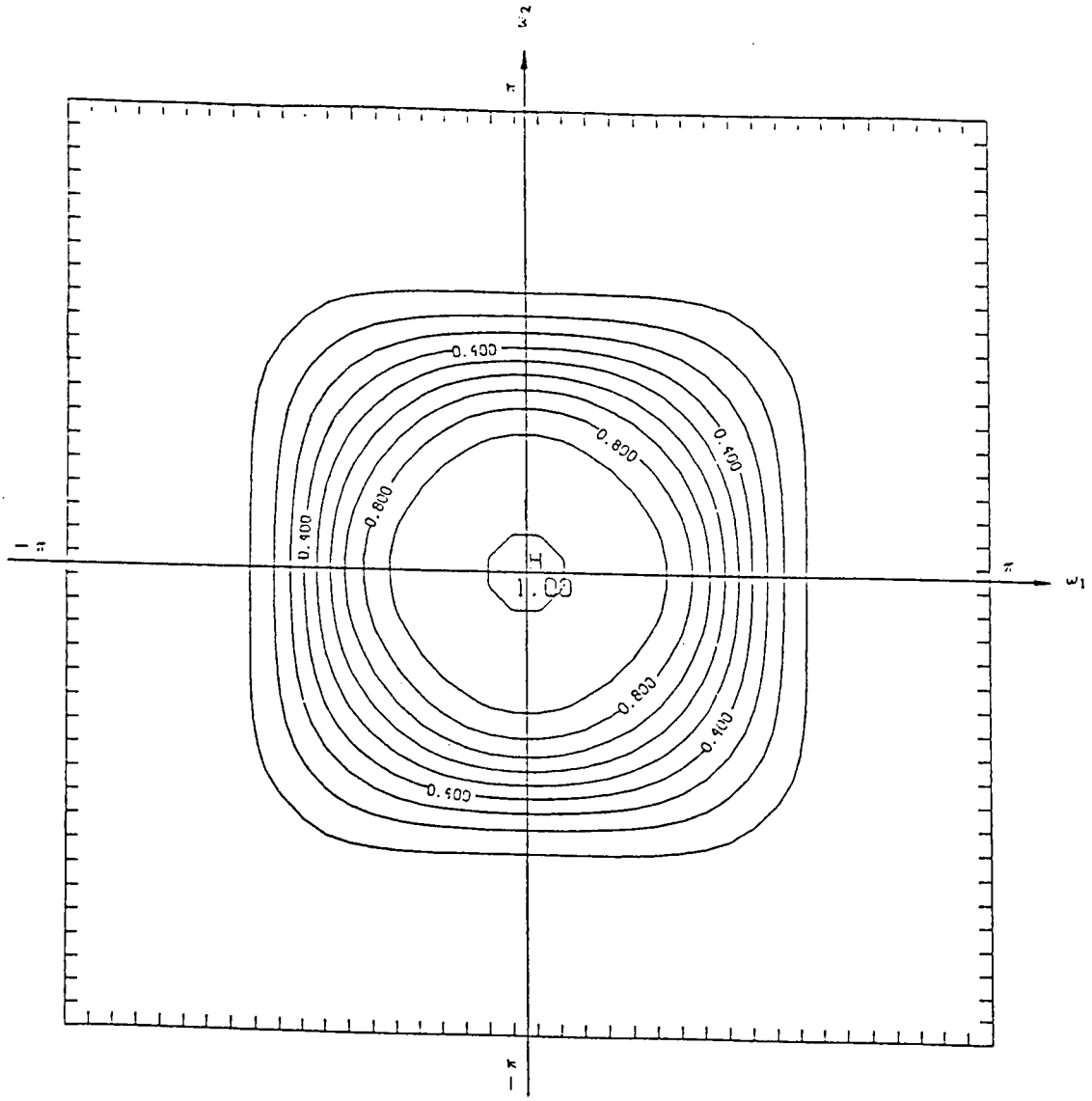


Fig. 2.2.5. Contour Lines of $T(Z_1, Z_2)$

2.3 Two-Dimensional and One-Dimensional Relationships

Up to this point we have not discussed how the value of a and the 1-D prototype cut-off frequency are chosen, given a desired cut-off contour, such that the best possible circular contour approximation at the cut-off is obtained.

From (2.1.1), the frequency mapping relationship between ω of the 1-D filter and (ω_1, ω_2) of the 2-D filter can be obtained as [15]

$$\omega = \omega_1 - \omega_2 - 2\left(tan^{-1}\frac{a\,sin\,\omega_1}{a\,cos\,\omega_1 + 1} + tan^{-1}\frac{a\,sin\,\omega_2}{a\,cos\,\omega_2 + 1}\right) \tag{2.3.1}$$

For any given circular contour, the contour line must pass through the ω_1-axis and ω_2-axis at the same distance, say ω_{c2}, from the origin. Thus, if ω_{c2} is the desired circular cut-off, the contour line must pass through (ω_{c2},0) and (0,ω_{c2}) for a constant bandwidth, ω_{c1}, of the 1-D prototype. Therefore the relation between the 1-D bandwidth, ω_{c1}, of the 1-D prototype and the 2-D circular cut-off contour ω_{c2}, for any given value of a, can be expressed as [15]

$$\omega_{c1} = \omega_{c2} - 2\,tan^{-1}\frac{a\,sin\,\omega_{c2}}{a\,cos\,\omega_{c2} - 1} \tag{2.3.2}$$

The next step is to find the value of a that gives the best approximation of a circular contour at the cut-off. To accomplish this, a search for the best a and the corresponding ω_{c1} is performed. Given ω_{c2}, different values of a are tried and the "best" is chosen. Recall that for $H(Z_1)$ to be stable, $|a| < 1$. Taking the effects of finite wordlength into account, the range of a to be searched is confined to $-0.9 \leq a \leq 0.9$. For each value of a, we find N points $\{(\omega_{1_i}, \omega_{2_i}), i = 1, 2, ..., N\}$, distributed at equal angular intervals along the contour of (2.3.1). a is searched such that the $\sqrt{\omega_1^2 + \omega_2^2}$ best approximates ω_{c2}. The error criteria used in the search

procedure is an average normalized error E, defined as

$$E(a) = \frac{\sum_{i=0}^{N} |\omega_{c2} - \sqrt{\omega_{1_i}^2 + \omega_{2_i}^2}|}{N\omega_{c2}} \qquad (2.3.3)$$

The program 1D2DWC which performs the search for a is given in Appendix C. Table I shows a_{opt}. (the "best" value of a), the corresponding 1-D cut-off frequency ω_{c1}, and the average normalized error of the circular contour approximation, for N = 20, for different values of specified ω_{c2}.

Table I. 1-D & 2-D Relationship

ω_{c2}/π	ω_{c1}/π	a_{opt}	Error(%)
.10	.00531	0.9	14.44
.15	.00804	0.9	14.21
.20	.01089	0.9	13.81
.25	.01388	0.9	13.41
.30	.01707	0.9	12.84
.35	.02053	0.9	12.14
.40	.02433	0.9	11.28
.45	.02860	0.9	10.25
.50	.03348	0.9	9.03
.55	.03918	0.9	7.58
.60	.04604	0.9	5.86
.65	.05454	0.9	3.85
.70	.06553	0.9	1.49
.75	.25640	0.7	1.34
.80	.50814	0.5	1.19
.85	.60264	0.5	1.94
.90	.71761	0.5	3.14
.95	.90761	0.3	3.40

It is clear from Table I that the normalized error is large for a small contour approximation. However, in actual design this is not as bad as it might seem, as

mentioned previously, since the response will be improved by cascading the separable filter. Two design examples will be given in Section 2.5 to justify this statement.

2.4 Algebraic Expression of 2-D Second-Order Transfer Function

Since a cascade of low-order filters can be more efficiently realized than the direct realization of a high-order one, we will start with $G(Z)$ expressed as the product of second-order sections. Thus it is appropriate to have general expressions for the 1-D and 2-D transformed transfer functions as a function of the 1-D second-order lowpass prototype sections. Only $G[H(Z_1)H(Z_2)]$ and $G[H(Z_i)]$ will be generalized, since $G[H(Z_1)H(Z_2^{-1})]$ can be implemented by a simple data array transformation as will be discussed in Chapter IV.

Consider a 1-D second-order transfer function $G_1(Z)$

$$G_1(Z) = \frac{p_0 + p_1 Z^{-1} + p_2 Z^{-2}}{q_0 + q_1 Z^{-1} + q_2 Z^{-2}} \tag{2.4.1}$$

By directly substituting $H(Z_i) = \frac{(Z_i^{-1} - a)}{(aZ_i^{-1} - 1)}$ into Z^{-1}, for i = 1, 2, we get

$$G_1[H(Z_i)] = \frac{p_0 + p_1 H(Z_i) + p_2 H^2(Z_i)}{q_0 + q_1 H(Z_i) + q_2 H^2(Z_i)}$$

Multiplying out and collecting the common terms, we obtain

$$G_1[H(Z_i)] = \frac{b_0 + b_1 Z_i^{-1} + b_2 Z_i^{-2}}{c_0 + c_1 Z_i^{-1} + c_2 Z_i^{-2}} \tag{2.4.2}$$

where

$$b_0 = p_0 + ap_1 + a^2 p_2$$

$$b_1 = 2ap_0 + (1 + a^2)p_1 + 2ap_2$$

$$b_2 = a^2 p_0 + ap_1 + p_2$$

$$c_0 = q_o + aq_1 + a^2q_2$$

$$c_1 = 2aq_o + (1 + a^2)q_1 + 2aq_2$$

$$c_2 = a^2q_o + aq_1 + q_2$$

To obtain $G_1[H(Z_1)H(Z_2)]$ we replace Z^{-1} by $H(Z_1)H(Z_2)$ in $G_1(Z)$. By simply expanding and collecting the common terms, we get

$$G_1[H(Z_1)H(Z_2)] = \frac{\begin{bmatrix}1 & Z_1^{-1} & Z_1^{-2}\end{bmatrix}\begin{bmatrix}b_{11} & b_{12} & b_{13}\\ b_{21} & b_{22} & b_{23}\\ b_{31} & b_{32} & b_{33}\end{bmatrix}\begin{bmatrix}1\\ Z_2^{-1}\\ Z_2^{-1}\end{bmatrix}}{\begin{bmatrix}1 & Z_1^{-1} & Z_1^{-2}\end{bmatrix}\begin{bmatrix}c_{11} & c_{12} & c_{13}\\ c_{21} & c_{22} & c_{23}\\ c_{31} & c_{32} & c_{33}\end{bmatrix}\begin{bmatrix}1\\ Z_2^{-1}\\ Z_2^{-2}\end{bmatrix}} \qquad (2.4.3)$$

where

$$b_{11} = p_0 - a^2p_1 - a^4p_2$$

$$b_{12} = p_{21} = 2ap_0 + (a - a^3)p_1 - 2a^3p_2$$

$$b_{13} = p_{31} = a^2(p_0 - p_1 - p_2)$$

$$b_{22} = 4a^2p_0 - (1 + 2a^2 - a^4)p_1 - 4a^2p_2$$

$$b_{23} = p_{32} = 2a^3p_0 - (a + a^3)p_1 - 2ap_2$$

$$b_{33} = a^4p_0 + a^2p_1 - p_2$$

$$c_{11} = q_0 + a^2q_1 + a^4q_2$$

$$c_{12} = q_{21} = 2aq_0 + (a + a^3)q_1 + 2a^3q_2$$

$$c_{13} = q_{31} = a^2(q_0 + q_1 + q_2)$$

$$c_{22} = 4a^2q_0 - (1 - 2a^2 - a^4)q_1 - 4a^2q_2$$

$$c_{23} = q_{32} = 2a^3 q_0 + (a + a^3)q_1 + 2aq_2$$

$$c_{33} = a^4 q_0 + a^2 q_1 + q_2$$

2.5 Numerical Examples

Example 2.5.1

In this example we design a 2-D circularly symmetric lowpass digital filter with a cut-off contour of 0.3π. From Table I we determined that the best a = 0.9. 1-D bandwidth $\omega_{c1} = 0.01707\pi$. Hence

$$H(Z_i) = \frac{Z_i^{-1} - 0.9}{0.9Z_i^{-1} - 1}, \quad i = 1, 2$$

The 1-D prototype is a second-order Butterworth filter whose transfer function is

$$G(Z) = 10^{-3} \frac{0.692402 - 1.384806Z^{-1} + 0.692402Z^{-2}}{1 - 1.9242Z^{-1} - 0.926972Z^{-2}} \tag{2.5.1}$$

The transformed subfilters then become

$$G[H(Z_i)] = \frac{0.249957 + 0.499914Z_i^{-1} - 0.249957Z_i^{-2}}{1.906731 - 1.425238Z_i^{-1} - 0.519199Z_i^{-2}}, \quad i = 1, 2 \tag{2.5.2}$$

$$G[H(Z_1)H(Z_2)] = \frac{\begin{bmatrix} 1 & Z_1^{-1} & Z_1^{-2} \end{bmatrix} \begin{bmatrix} 0.226838 & 0.451169 & 0.224338 \\ 0.451169 & 0.902352 & 0.451169 \\ 0.226838 & 0.451169 & 0.226838 \end{bmatrix} \begin{bmatrix} 1 \\ Z_2^{-1} \\ Z_2^{-2} \end{bmatrix}}{\begin{bmatrix} 1 & Z_1^{-1} & Z_1^{-2} \end{bmatrix} \begin{bmatrix} 4.958436 & 1.700327 & 0.224532 \\ 1.700327 & -6.048263 & -0.797232 \\ 0.224532 & -0.797232 & 2.447003 \end{bmatrix} \begin{bmatrix} 1 \\ Z_2^{-1} \\ Z_2^{-2} \end{bmatrix}} \tag{2.5.3}$$

Figures 2.5.1 and 2.5.2 show the magnitude squared response and the contour plot of the filter respectively. The approximation at the 3-db contour is seen to be very close to a circle.

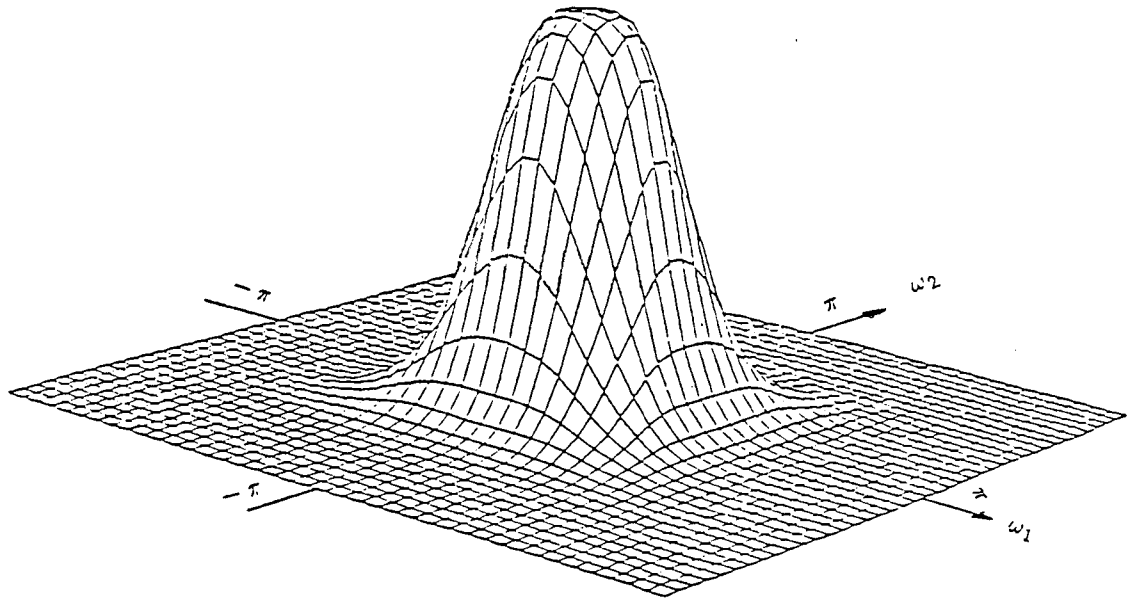


Fig. 2.5.1. Magnitude Squared Response of Lowpass Filter with $.3\pi$ Cut-off

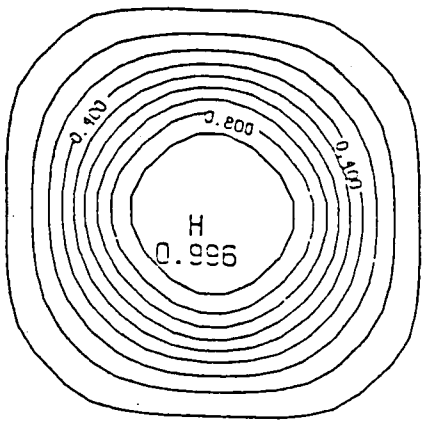


Fig. 2.5.2. Contour Plot of Lowpass Filter with $.3\pi$ Cut-off

Example 2.5.2

In this example we design a 2-D fourth-order circularly symmetric Butterworth lowpass digital filter with a cut-off contour of 0.55π. From Table I we have $a = 0.9$, $\omega_{c1} = 0.03918\pi$. The 1-D prototype is expressed as a cascade of two second-order sections

$$G(Z) = G_1(Z)G_2(Z)$$

where

$$G_1(Z) = 10^{-4}\frac{0.516108 + 1.032216Z^{-1} + 0.516108Z^{-2}}{1 - 1.78265Z^{-1} + 0.796243Z^{-2}} \tag{2.5.4}$$

$$G_2(Z) = \frac{0.237877 + 0.475754Z^{-1} + 0.237877Z^{-2}}{1 - 1.89579Z^{-1} + 0.910245Z^{-2}} \tag{2.5.5}$$

The transformed subfilters then become

$$G_1[H(Z_i)] = \frac{0.018632 + 0.037263Z_i^{-1} + 0.018632Z_i^{-2}}{4.057183 - 0.664092Z_i^{-1} - 0.185799Z_i^{-2}}, \quad i = 1, 2 \tag{2.5.6}$$

$$G_1[H(Z_1)H(Z_2)] = \frac{\begin{bmatrix}1 & Z_1^{-1} & Z_1^{-2}\end{bmatrix}\begin{bmatrix}0.016908 & 0.033630 & 0.016722\\ 0.033630 & 0.067260 & 0.033630\\ 0.016722 & 0.033630 & 0.016908\end{bmatrix}\begin{bmatrix}1\\ Z_2^{-1}\\ Z_2^{-2}\end{bmatrix}}{\begin{bmatrix}1 & Z_1^{-1} & Z_1^{-2}\end{bmatrix}\begin{bmatrix}7.846857 & 5.698535 & 1.101033\\ 5.698535 & -2.031261 & -1.269957\\ 1.101033 & -1.269957 & 0.839652\end{bmatrix}\begin{bmatrix}1\\ Z_2^{-1}\\ Z_2^{-2}\end{bmatrix}} \tag{2.5.7}$$

$$G_2[H(Z_i)] = \frac{0.858736 - 1.717472Z_i^{-1} - 0.858736Z_i^{-2}}{0.031087 - 0.007061Z_i^{-1} - 0.014034Z_i^{-2}}, \quad i = 1, 2 \tag{2.5.8}$$

$$G_2[H(Z_1)H(Z_2)] = \frac{\begin{bmatrix} 1 & Z_1^{-1} & Z_1^{-2} \end{bmatrix} \begin{bmatrix} 0.779309 & 1.550006 & 0.770721 \\ 1.550006 & 3.100060 & 1.550006 \\ 0.770721 & 1.550006 & 0.779309 \end{bmatrix} \begin{bmatrix} 1 \\ Z_2^{-1} \\ Z_2^{-2} \end{bmatrix}}{\begin{bmatrix} 1 & Z_1^{-1} & Z_1^{-2} \end{bmatrix} \begin{bmatrix} 0.061622 & 0.038895 & 0.011709 \\ 0.038895 & -0.021604 & 0.008199 \\ 0.011709 & 0.008199 & 0.030755 \end{bmatrix} \begin{bmatrix} 1 \\ Z_2^{-1} \\ Z_2^{-2} \end{bmatrix}} \tag{2.5.9}$$

The magnitude squared response and contour plot of the resulting filter are shown in Figures 2.5.3 and 2.5.4 respectively.

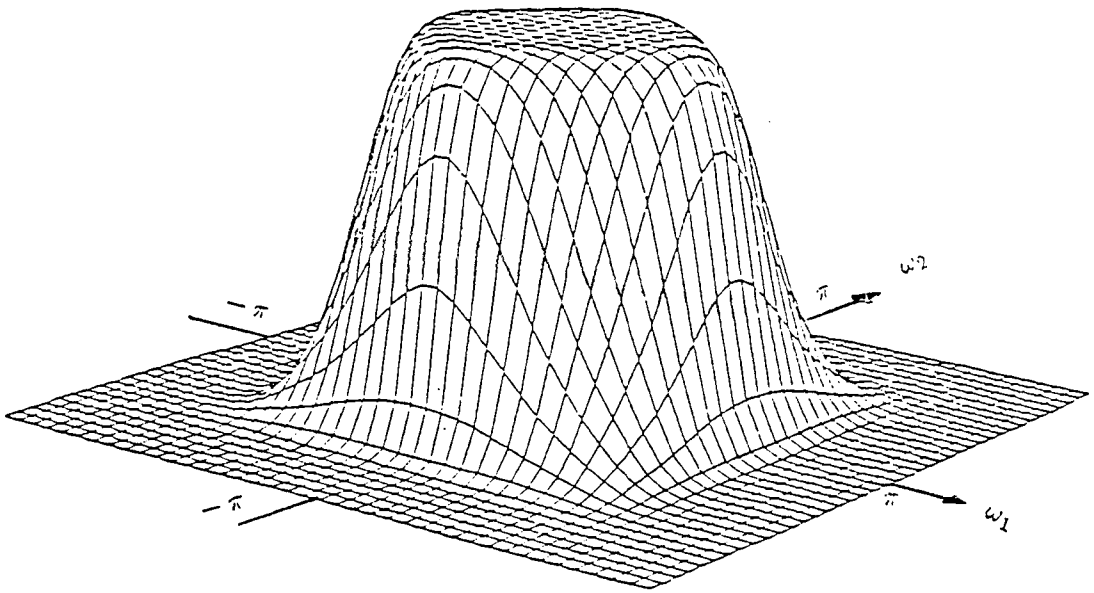


Fig. 2.5.3. Magnitude Squared Response of Lowpass Filter with .55π Cut-off

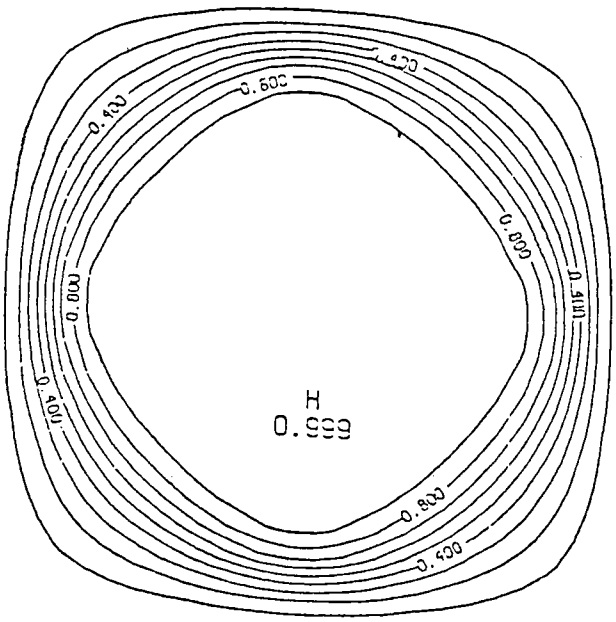


Fig. 2.5.4. Contour Plot of Lowpass Filter with .55π Cut-off

CHAPTER III

DESIGN OF GENERAL 2-D RECURSIVE DIGITAL FILTERS VIA 2-D TO 2-D DIGITAL SPECTRAL TRANSFORMATION

3.1 Background

For 1-D IIR filters, given a lowpass digital filter, another lowpass filter with a different cut-off frequency, a highpass, a bandpass, or a bandstop filter can be quickly obtained by applying one of the four digital transformations proposed by Constantinides [17] to the lowpass prototype filter. However, in the 2-D digital filter case, the cut-off contour as well as the shape of the filter are involved. Pendergrass [13] has proposed a technique that transforms a 2-D lowpass digital filter to the other filter types by independently applying the four Constantinides' DST's to Z_1 and Z_2. This is equivalent to a mapping of one point on Z_1 to another on Z_1 and one point on Z_2 to another on Z_2. Since only one point on the contour is controlled and because of the nonlinear nature of the transformation, the shape of the resulting filter is transformed in an unknown way. However, practical design examples, as will be seen later, show that an acceptable degree of shape factor is often preserved in the transformed filter.

As far as the stability of the transformations is concerned, Constantinides [17] and Pendergrass [13] have correctly pointed out that the transformed filter is stable if the DST's are stable. However, neither of them proved the stability of their transformations. Harn [16] recently has proven that the lowpass-to-highpass transformation in [17] is not stable. However, since the transformation is an allpass function, a stable one is simply obtained from the reciprocal of the unstable one.

3.2 Stable 2-D to 2-D Digital Spectral Transformations

In this thesis, the modified stable DST's in [16] are used. These transformations are called single variable transformations [13]. The four stable DST's are shown in Table II.

Table II. The Four Stable Digital Spectral Transformations

Type	Equation	Parameters	Frequencies
LP → LP	$Z^{-1} = \frac{Z^{-1}+b}{bZ^{-1}+1}$	$b = \sin(\frac{\omega_c-\beta}{2})/\sin(\frac{\omega_c+\beta}{2})$	$\beta \rightarrow \omega_c$
LP → HP	$Z^{-1} = -\frac{bZ^{-1}+1}{Z^{-1}+b}$	$b = \cos(\frac{\omega_c-\beta}{2})/\cos(\frac{\omega_c+\beta}{2})$	$\beta \rightarrow -\omega_c$
LP → BP	$Z^{-1} = -\frac{Z^{-2}+c_1Z^{-1}-c_2}{c_2Z^{-2}+c_1Z^{-1}+1}$	$c_1 = -2h\frac{k}{k+1}$ $c_2 = \frac{k-1}{k+1}$ $k = \cot(\frac{\omega_{c1}-\omega_{c2}}{2})\tan(\frac{\beta}{2})$ $h = \cos(\frac{\omega_{c1}+\omega_{c2}}{2})/\cos(\frac{\omega_{c1}-\omega_{c2}}{2})$	$\beta \rightarrow \omega_{c1}$ $-\beta \rightarrow \omega_{c2}$ $\omega_{c1} > \omega_{c2}$
LP → BS	$Z^{-1} = -\frac{Z^{-2}-c_1Z^{-1}+c_2}{c_2Z^{-2}+c_1Z^{-1}+1}$	$c_1 = -2h\frac{1}{k+1}$ $c_2 = \frac{1-k}{1+k}$ $k = \tan(\frac{\omega_{c1}-\omega_{c2}}{2})\tan(\frac{\beta}{2})$ $h = \cos(\frac{\omega_{c1}+\omega_{c2}}{2})/\cos(\frac{\omega_{c1}-\omega_{c2}}{2})$	$-\beta \rightarrow \omega_{c1}$ $\beta \rightarrow \omega_{c2}$ $\omega_{c1} > \omega_{c2}$

The stability of these transformations is proven in [16]. As mentioned previously, Z_1 and Z_2 are transformed independently. Therefore we can design a filter with different types in Z_1 and Z_2. The effect of the possible combinations of the transformations was tabulated in [13].

From Table II, we see that the lowpass-to-bandpass and lowpass-to-bandstop are a one-to-two mapping. Therefore the filter order will be doubled in each direction so transformed. This is similar to the 1-D filter case. After all, the 2-D to 2-D DST is just a product of two 1-D to 1-D DST's.

3.3 Numerical Examples

Example 3.3.1

In this example we design a 2-D digital recursive filter with the following specifications:

LowPass with cut-off frequency of $.55\pi$ for ω_1

HighPass with cut-off frequency of $.4\pi$ for ω_2

A cascade of the transfer functions in (2.5.6), (2.5.7), (2.5.8), (2.5.9) are used as a prototype.

The transformations on Z_1 and Z_2 are

$$Z_1^{-1} = Z_1^{-1} \tag{3.3.1a}$$

$$Z_2^{-1} = \frac{-1 - 12.393356 Z_2^{-1}}{-12.393356 - Z_2^{-1}} \tag{3.3.1b}$$

The resulting subfilters are

$$T_1(Z_2) = \frac{0.033422 - 0.066843 Z_2^{-1} - 0.033422 Z_2^{-2}}{6.315802 - 2.0783502 Z_2^{-1} - 0.408253 Z_2^{-2}} \tag{3.3.2}$$

$$T_1(Z_1, Z_2) = 10^{-1} \frac{\begin{bmatrix} 1 & Z_1^{-1} & Z_1^{-2} \end{bmatrix} \begin{bmatrix} 0.303053 & -0.603256 & 0.300210 \\ 0.603256 & -1.206525 & 0.603256 \\ 0.300210 & -0.603256 & 0.303053 \end{bmatrix} \begin{bmatrix} 1 \\ Z_2^{-1} \\ Z_2^{-2} \end{bmatrix}}{\begin{bmatrix} 1 & Z_1^{-1} & Z_1^{-2} \end{bmatrix} \begin{bmatrix} 12.769652 & -11.027554 & 2.475843 \\ 8.488240 & 2.042536 & -2.14535 \\ 1.542141 & 1.482262 & 1.143285 \end{bmatrix} \begin{bmatrix} 1 \\ Z_2^{-1} \\ Z_2^{-2} \end{bmatrix}} \tag{3.3.3}$$

$$T_2(Z_2) = \frac{154.041764 - 308.083527 Z_2^{-1} - 154.041764 Z_2^{-2}}{4.876430 - 2.210027 Z_2^{-1} - 2.274153 Z_2^{-2}} \tag{3.3.4}$$

$$T_2(Z_1,Z_2)=\frac{\begin{bmatrix}1 & Z_1^{-1} & Z_1^{-2}\end{bmatrix}\begin{bmatrix}139.678660 & -278.043851 & 138.368270\\ 278.043852 & -556.093879 & 278.043852\\ 138.368270 & -278.043851 & 139.678660\end{bmatrix}\begin{bmatrix}1\\ Z_2^{-1}\\ Z_2^{-2}\end{bmatrix}}{\begin{bmatrix}1 & Z_1^{-1} & Z_1^{-2}\end{bmatrix}\begin{bmatrix}9.958580 & -7.830633 & 2.342042\\ 5.714569 & 2.172583 & 1.030471\\ 1.930746 & -2.320057 & 4.837163\end{bmatrix}\begin{bmatrix}1\\ Z_2^{-1}\\ Z_2^{-2}\end{bmatrix}} \tag{3.3.5}$$

The magnitude squared response and the contour plot of

$$T_1(Z_1)T_1(Z_2)T_1(Z_1,Z_2)T_1(Z_1,Z_2^{-1})T_2(Z_1)T_2(Z_2)T_2(Z_1,Z_2)T_2(Z_1,Z_2^{-1})$$

are shown in Figures 3.3.1 and 3.3.2 respectively.

Example 3.3.2

In this example we design a 2-D bandpass recursive digital filter with cutoff band edges of $.7\pi$ and $.3\pi$. The lowpass prototype is a cascade of (2.5.2) and (2.5.3). The transformations on Z_1 and Z_2 are

$$Z_i^{-1}=\frac{0.17557-10^{-7}(0.44544)Z_i^{-1}-Z_i^{-2}}{1-10^{-7}(0.44544)Z_i^{-1}-0.17557Z_i^{-2}},\quad i=1,2 \tag{3.3.6}$$

The magnitude squared response and the contour plot of the resulting bandpass filter are shown in Figures 3.3.3. and 3.3.4 respectively. Filter coefficients are obtained in a similar fashion as in Example 3.3.1. They are omitted to conserve available space.

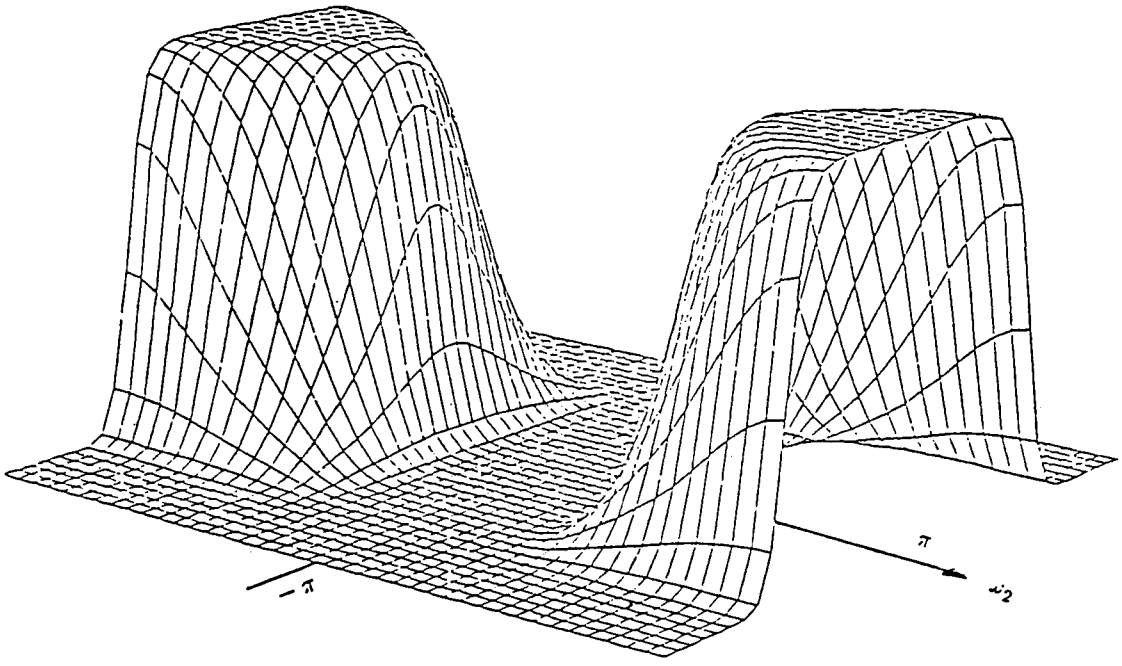


Fig. 3.3.1. Magnitude Squared Response of Transformed Filter of Ex. 2.5.2

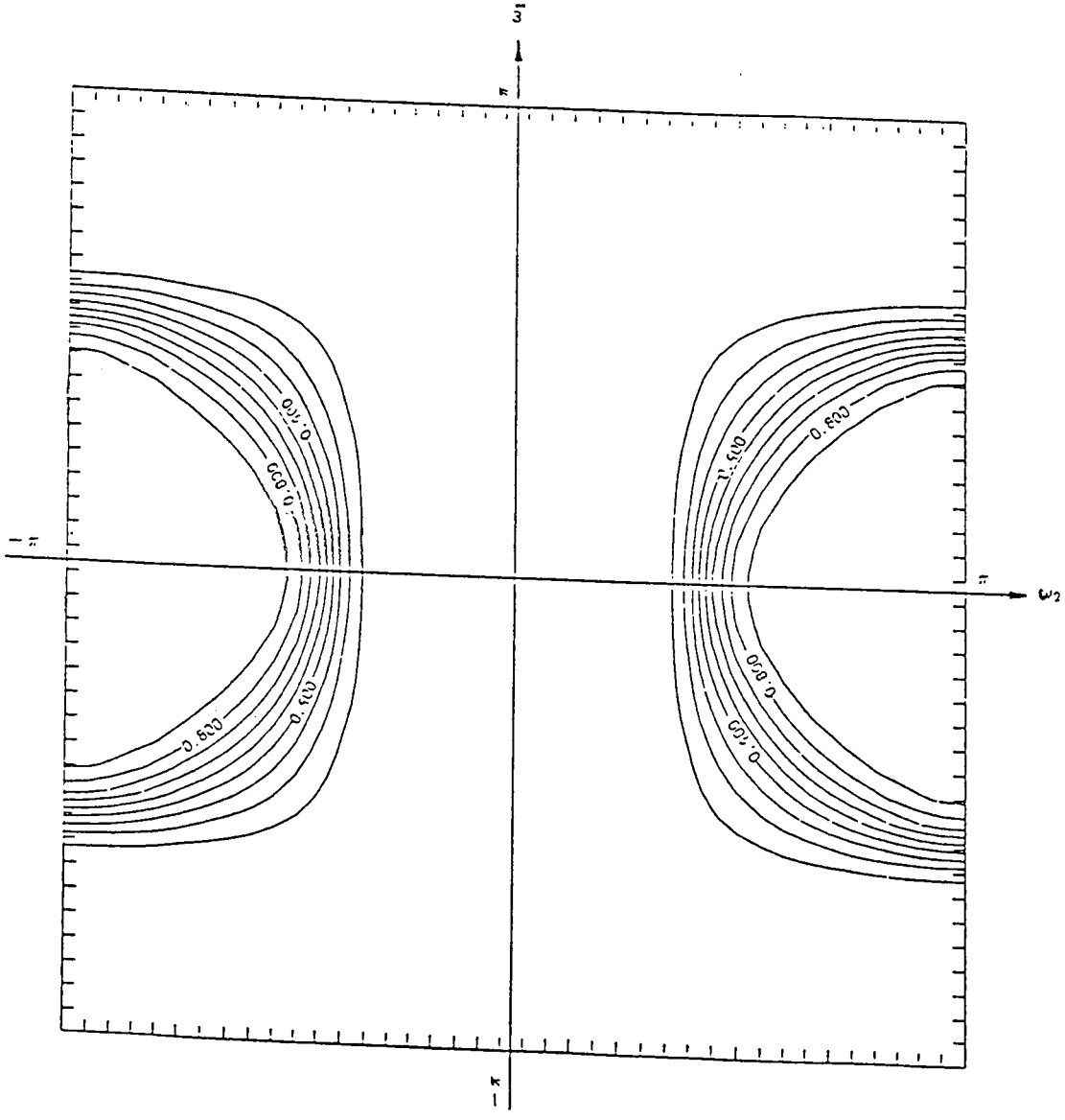


Fig. 3.3.2. Contour Plot of Transformed Filter of Ex. 2.5.2

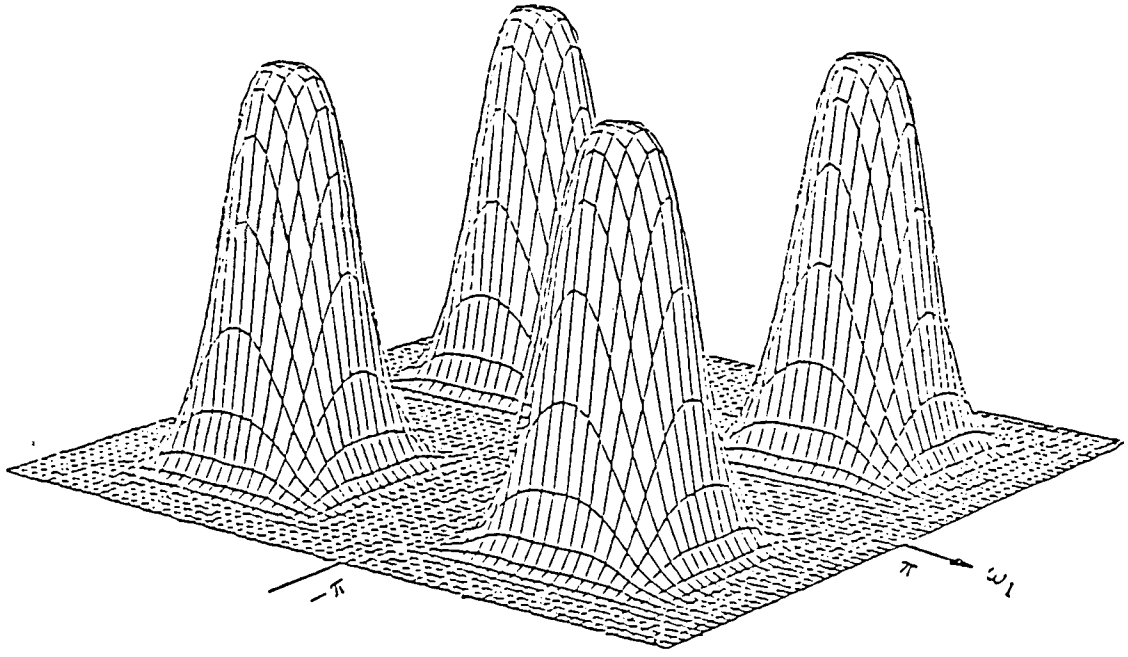


Fig. 3.3.3. Magnitude Squared Response of Transformed Filter of Ex. 2.5.1

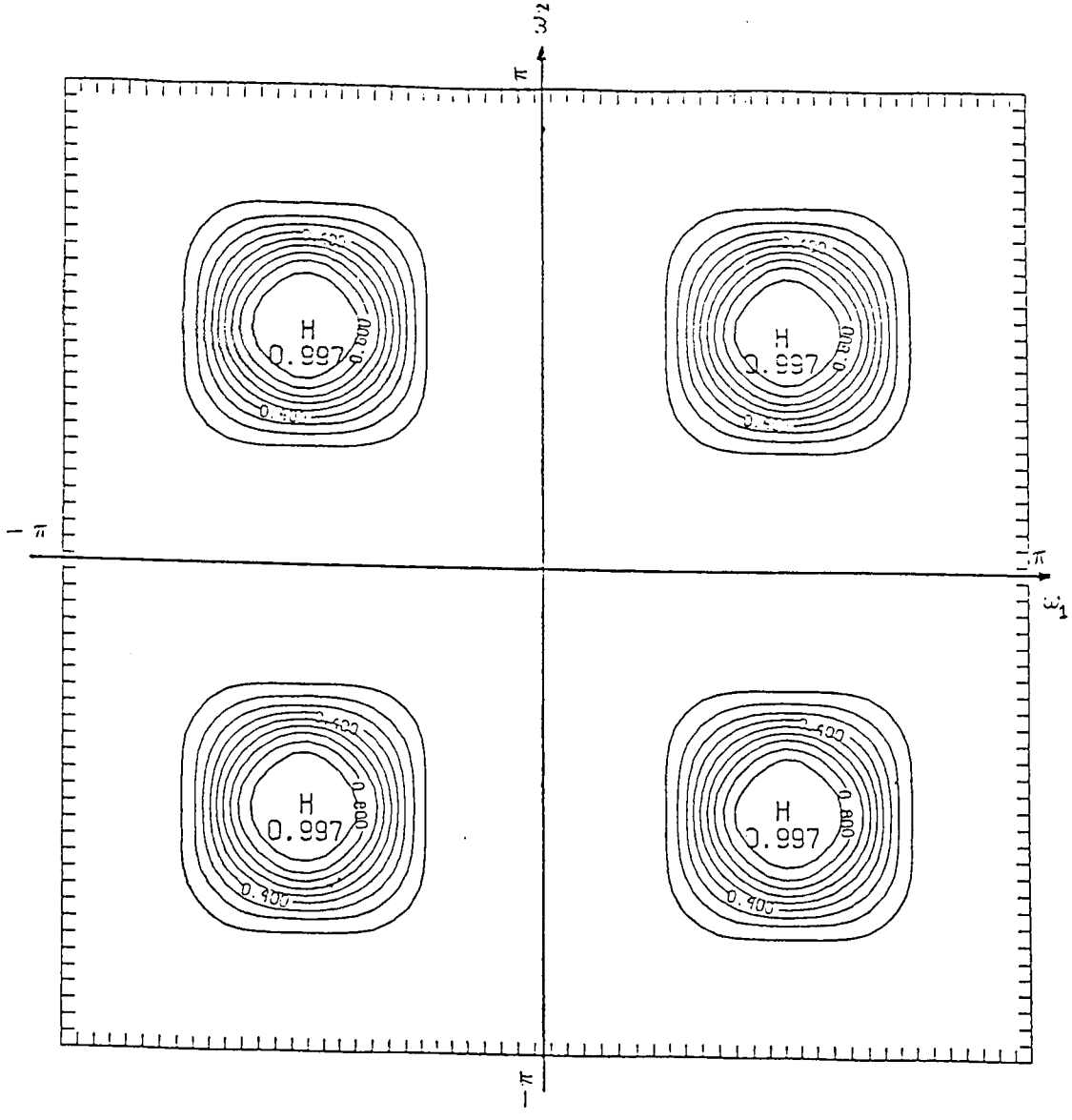


Fig. 3.3.4. Contour Plot of Transformed Filter of Ex. 2.5.1

CHAPTER IV

REALIZATION OF 2-D RECURSIVE DIGITAL FILTERS

4.1 2-D Transfer Function and Its Difference Equation

Before any realization structure is discussed, a comment on the 2-D structure suggested by Harn [15] should be made. Harn [15] suggested a 2-D realization structure by directly replacing Z^{-1} in the 1-D direct structure by the following DST function

$$Z^{-1} \leftarrow \frac{(Z_1^{-1} - a)(Z_2^{-1} - a)}{(aZ_1^{-1} - 1)(aZ_2^{-1} - 1)}$$

By this direct substitution of the DST function, however, a delay-free loop is introduced into the resulting structure. Therefore, the filter is not directly realizable [2]. Figures 4.1.1 and 4.1.2 show the 1-D direct structure used in [15] and the suggested 2-D structure after the substitution of the DST function.

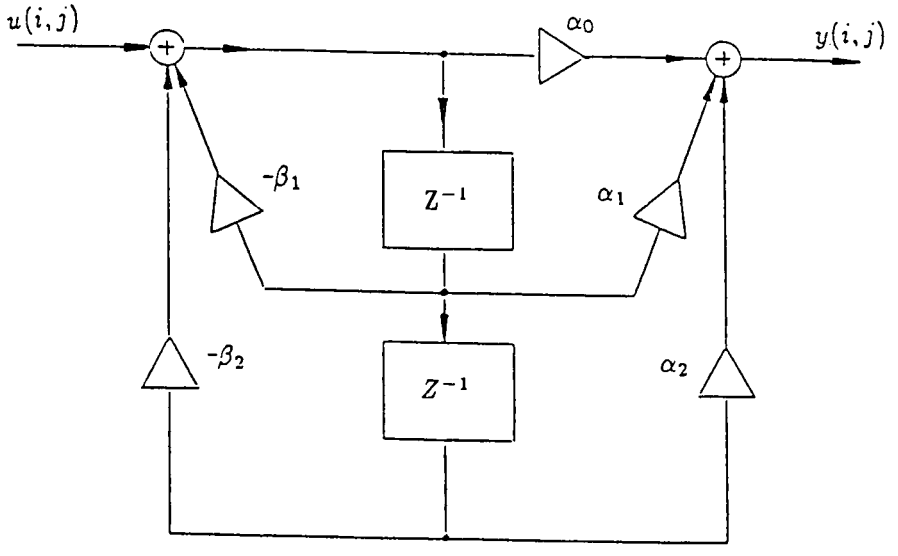


Fig. 4.1.1. 1-D Second-Order Direct Structure Realization

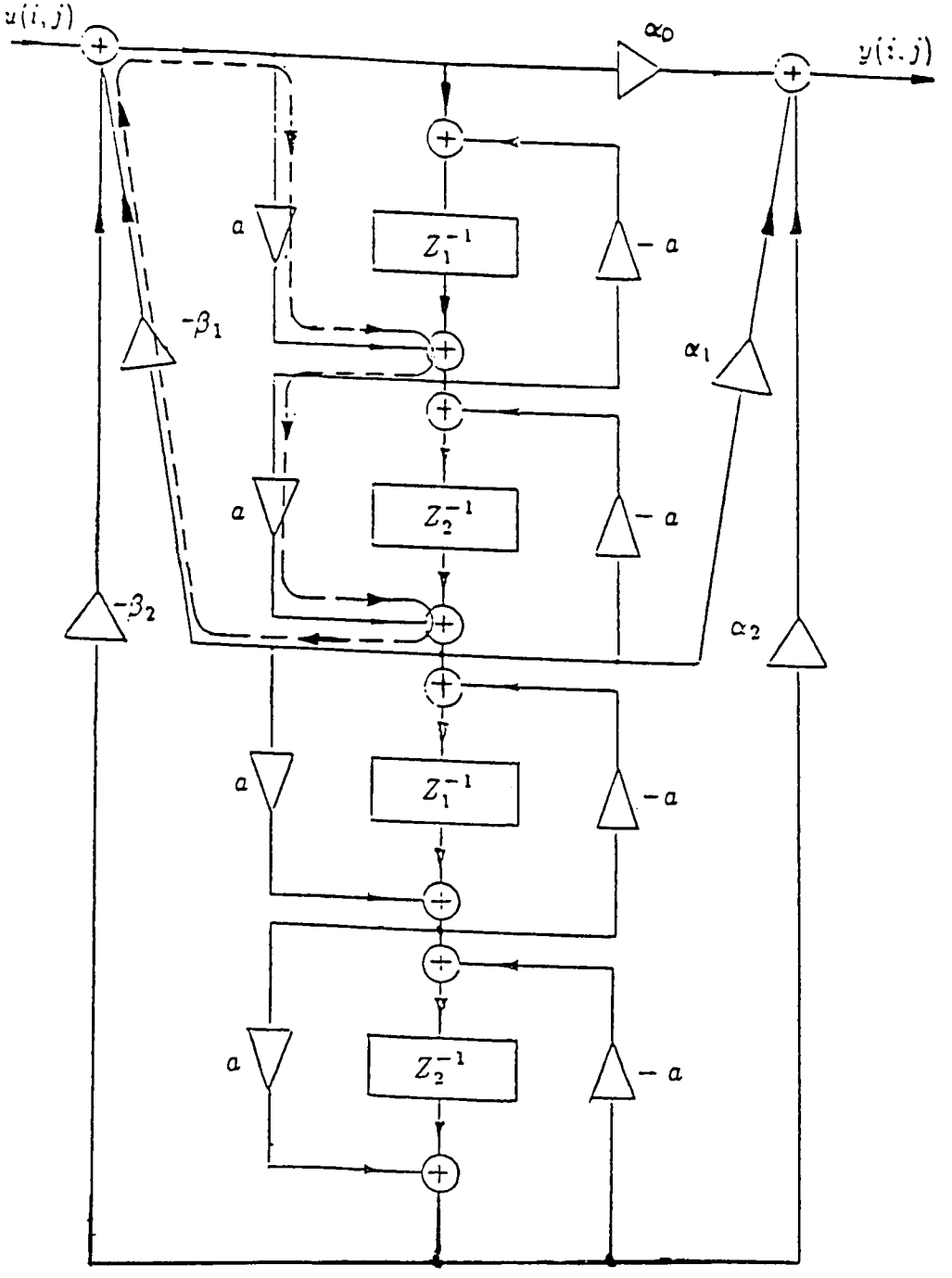


Fig. 4.1.2. 2-D Second-Order Structure via Direct DST Substitution

The transfer function of any first quadrant (causal) quarter-plane LSI 2-D digital filter, $G(Z_1, Z_2)$ is given as

$$G(Z_1, Z_2) = \frac{\sum_{r_1=0}^{M_1} \sum_{r_2=0}^{M_2} b(r_1, r_2) Z_1^{-r_1} Z_2^{-r_2}}{\sum_{k_1=0}^{N_1} \sum_{k_2=0}^{N_2} c(k_1, k_2) Z_1^{-k_1} Z_2^{-k_2}} \tag{4.1.1}$$

The difference equation associated with (4.1.1) can be written as

$$y(n_1, n_2) = \sum_{r_1=0}^{M_1} \sum_{r_2=0}^{M_2} b(r_1, r_2) u(n_1 - r_1, n_2 - r_2) - \sum_{\substack{k_1=0 \\ k_1+k_2 \neq 0}}^{N_1} \sum_{k_2=0}^{N_2} c(k_1, k_2) y(n_1 - k_1, n_2 - k_2) \tag{4.1.2}$$

From (4.1.2) it is seen that the output at a particular point can be computed recursively as a weighted sum of inputs and previous output points (and initial conditions). The output mask, which is determined by the coefficient array $c(k_1, k_2)$, of a recursively computable first-quadrant filter is drawn in Fig.4.1.3. The realization of (4.1.2) is known as the difference equation realization or direct structure.

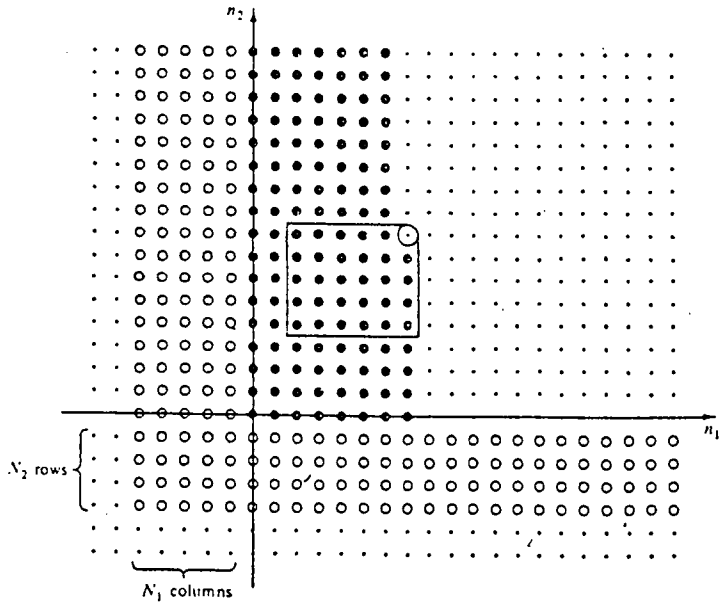


Fig. 4.1.3. Output-Mask of Recursively Computable 1^{st} Quadrant Quarter-Plane

4.2 Two-Dimensional State-Space Structure Realization

It is well known that an IIR digital filter implemented directly by the difference equation, particularly with fixed-point arithmetic, results in a filter with high coefficient sensitivity and roundoff noise. The state-space structure realization has been studied as an alternative approach of realization. Over the past years, several authors have proposed different state-space models. Kung [20] discussed these different models, and showed that the Roesser model [21] is the most general 2-D state-space model.

The governing equations of Roesser's local state-space structure for a single-input, single-output [21] are described as follows

$$\begin{aligned}\begin{bmatrix} x^h(i-1,j) \\ x^v(i,j-1) \end{bmatrix} &= \begin{bmatrix} A_1 & A_2 \\ A_3 & A_4 \end{bmatrix} \begin{bmatrix} x^h(i,j) \\ x^v(i,j) \end{bmatrix} - \begin{bmatrix} b_1 \\ b_2 \end{bmatrix} u(i,j) \\ &= Ax + bu \end{aligned} \tag{4.2.1a}$$

$$\begin{aligned} y(i,j) &= \begin{bmatrix} c_1^t & c_2^t \end{bmatrix} \begin{bmatrix} x^h(i,j) \\ x^v(i,j) \end{bmatrix} + du(i,j), \quad i,j \geq 0 \\ &= c^t x + du \end{aligned} \tag{4.2.1b}$$

where u is the input, and y is the output. x^h and x^v are local horizontal and vertical state vectors respectively. The dimensions of x^h and x^v are referred as the order of the filter (not necessarily the same as the order of the transfer function). The problem here is to find matrices A, b, c, and d which will realize a particular transfer function $G(Z_1, Z_2)$ with the least possible number of delay elements.

Kung [20] has shown that an M × N order transfer function in general can not always be realized with real coefficients by M×N delay elements. Instead, the minimal real-gain state-space realization for an M × N order transfer function requires

no greater than M+2N (or 2M+N) delay elements. The necessary conditions for a general M×N order transfer function to be realizable by M+N delay elements are not known, although it is possible, at least in theory, to determine the minimum number of delay elements that are needed [30,31]. However, Kung [20] showed that an M×N order transfer function with separable numerator or separable denominator can be implemented by just M+N delay elements.

The A, b, c, and d of the canonical M+2N delay structure with real coefficients based on Roesser's local state-space structure of an IIR digital filter whose transfer function is

$$G(Z_1, Z_2) = \frac{\sum_{i=0}^{M} \sum_{j=0}^{N} \alpha_{ij} Z_1^{-i} Z_2^{-j}}{\sum_{i=0}^{M} \sum_{j=0}^{N} \beta_{ij} Z_1^{-i} Z_2^{-j}} \quad ; \beta_{00} = 1 \tag{4.2.2}$$

(there is no loss of generality by assuming $\beta_{00} = 1$) is given by

$$A = \left[\begin{array}{cccc|cccc}
-\beta_{10} & -\beta_{20} & \cdots & \beta_{M0} & 1 & & & \\
1 & \ddots & & & & \ddots & & \\
 & \ddots & 1 & 0 & & & & \\
\hline
-\tilde{\beta}_{11} & & -\tilde{\beta}_{M1} & & -\beta_{01} & 1 & & \\
 & -\tilde{\beta}_{ij} & & & \vdots & & \ddots & \\
 & & & & & & & 1 \\
-\tilde{\beta}_{1N} & & -\tilde{\beta}_{MN} & & -\beta_{0N} & & & 0 \\
\hline
\tilde{\alpha}_{11} & & \tilde{\alpha}_{M1} & & \alpha_{01} & & 0 & 1 \\
 & \tilde{\alpha}_{ij} & & & \vdots & & & \ddots \\
 & & & & & & & 1 \\
\tilde{\alpha}_{1N} & & \tilde{\alpha}_{MN} & & \alpha_{0N} & & & 0
\end{array}\right] \tag{4.2.3a}$$

$$b^t = \begin{bmatrix} 1 & 0 & \dots & 0 & \vdots & -\beta_{01} & \dots & -\beta_{0N} & \vdots & \alpha_{01} & \dots & \alpha_{0N} \end{bmatrix} \qquad (4.2.3b)$$

$$c^t = \begin{bmatrix} \tilde{\alpha}_{10} & \dots & \tilde{\alpha}_{M0} & \vdots & \alpha_{00} & 0 & \dots & 0 & \vdots & 1 & 0 & \dots & 0 \end{bmatrix} \qquad (4.2.3c)$$

$$d = \alpha_{00} \qquad (4.2.3d)$$

where

$$\tilde{\alpha}_{ij} = \alpha_{ij} - \beta_{i0}\alpha_{0j}, \qquad 1 \le i \le M, \quad 0 \le j \le N$$

$$\tilde{\beta}_{ij} = \beta_{ij} - \beta_{i0}\beta_{0j}, \qquad 1 \le i \le M, \quad 1 \le j \le N$$

Because we have been dealing primarily with 2×2 order subfilters, the canonical structure of a 2×2 order transfer function is given below as

$$\begin{bmatrix} x_1^h(i+1,j) \\ x_2^h(i+1,j) \\ w_1^v(i,j-1) \\ w_2^v(i,j-1) \\ x_1^v(i,j-1) \\ x_2^v(i,j-1) \end{bmatrix} = \begin{bmatrix} -\beta_{10} & -\beta_{20} & 1 & 0 & 0 & 0 \\ 1 & 0 & 0 & 0 & 0 & 0 \\ \beta_{10}\beta_{01} - \beta_{11} & \beta_{20}\beta_{01} - \beta_{21} & -\beta_{01} & 1 & 0 & 0 \\ \beta_{10}\beta_{02} - \beta_{12} & \beta_{20}\beta_{02} - \beta_{22} & -\beta_{02} & 0 & 0 & 0 \\ \alpha_{11} - \alpha_{01}\beta_{10} & \alpha_{21} - \alpha_{01}\beta_{20} & \alpha_{01} & 0 & 0 & 1 \\ \alpha_{12} - \alpha_{02}\beta_{10} & \alpha_{22} - \alpha_{02}\beta_{20} & \alpha_{02} & 0 & 0 & 0 \end{bmatrix} \begin{bmatrix} x_1^h(i,j) \\ x_2^h(i,j) \\ w_1^v(i,j) \\ w_2^v(i,j) \\ x_1^v(i,j) \\ x_2^v(i,j) \end{bmatrix}$$

$$- \begin{bmatrix} 1 & 0 & -\beta_{01} & -\beta_{02} & \alpha_{01} & \alpha_{02} \end{bmatrix}^t u(i,j) \qquad (4.2.4a)$$

$$y(i,j) = \begin{bmatrix} \alpha_{10} - \alpha_{00}\beta_{10} & \alpha_{20} - \alpha_{00}\beta_{20} & \alpha_{00} & 0 & 1 & 0 \end{bmatrix} \begin{bmatrix} x_1^h(i,j) \\ x_2^h(i,j) \\ w_1^v(i,j) \\ w_2^v(i,j) \\ x_1^v(i,j) \\ x_2^v(i,j) \end{bmatrix}$$

$$- \alpha_{00}u(i,j) \qquad (4.2.4b)$$

The corresponding signal flow graph is also drawn in Fig. 4.2.1.

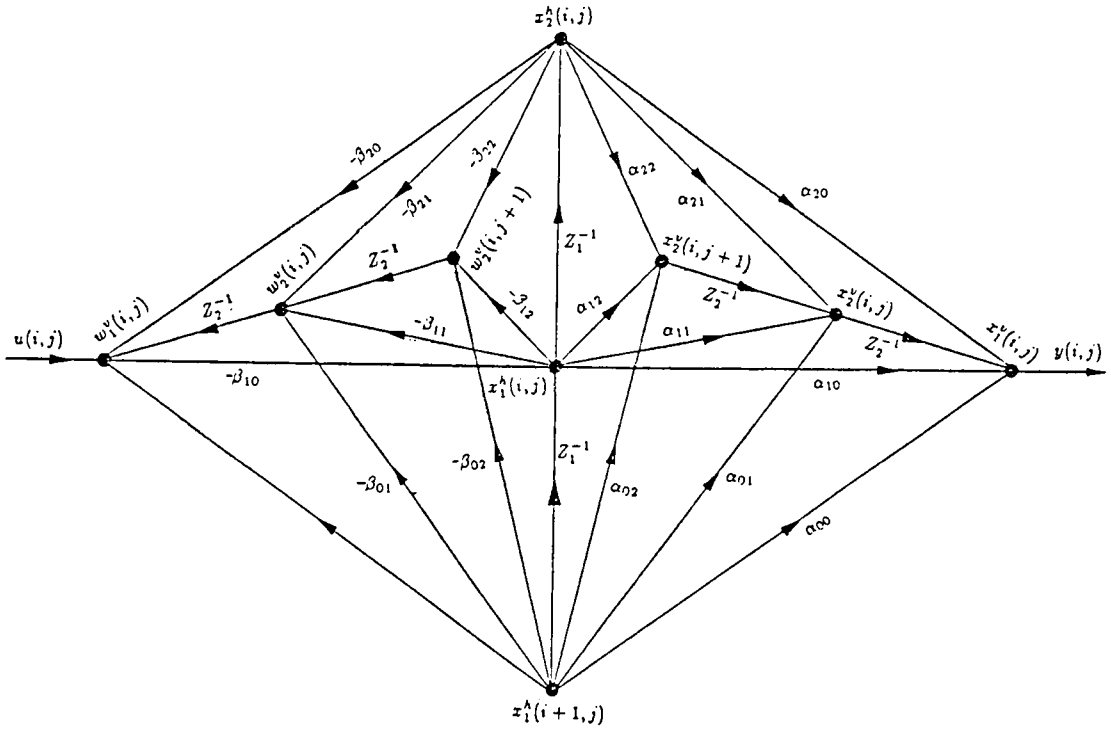


Fig. 4.2.1. Signal Flow Graph of Canonical Structure of 2×2 Transfer Function

As in the 1-D case, the 2-D state-space structure obtained here is not unique. Other realization structures can be obtained by applying a linear transformation to the state vectors. The transformation leads to an equivalent realization of the following form

$$\tilde{A} = T^{-1}AT \tag{4.2.5a}$$

$$\tilde{b} = T^{-1}b \tag{4.2.5b}$$

$$\tilde{c} = cT \tag{4.2.5c}$$

$$\tilde{d} = d \tag{4.2.5d}$$

where T is a nonsingular 2-D similarlity transformation matrix, and is of the following form

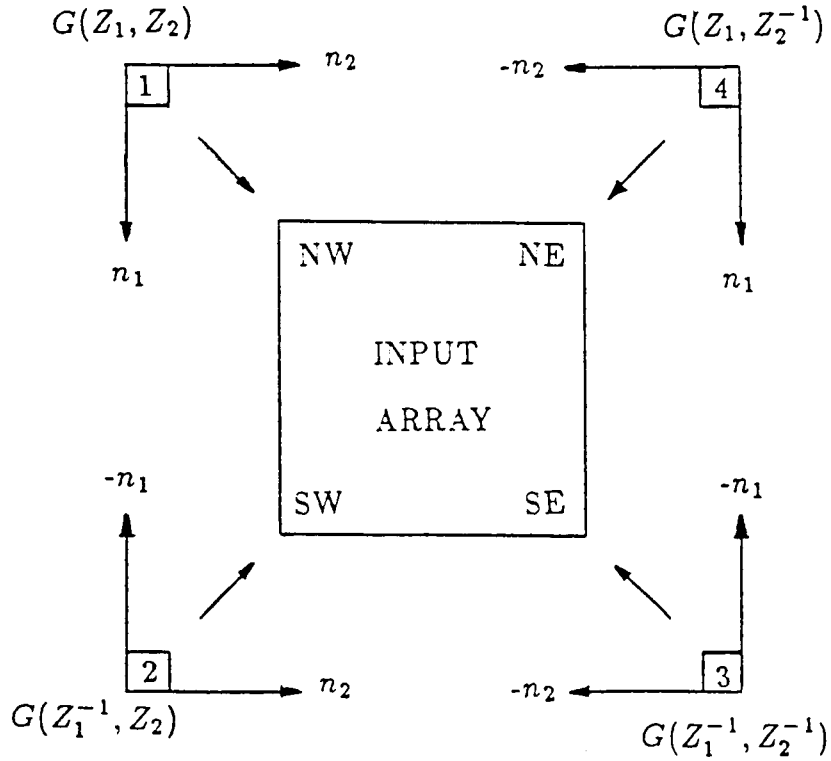
$$T = \begin{bmatrix} T_1 & 0 \\ 0 & T_2 \end{bmatrix} \tag{4.2.6}$$

where the dimensions of T_1 and T_2 are M×M and 2N×2N respectively.

By properly choosing the transformation matrix T, other state-space realization structures with more desirable properties can be obtained. Recently a 2-D minimum round-off noise state-space structure based on the Roesser local state-space model has been developed [22,23]. Upon the availability of two positive-definite matrices W (noise matrix) and K (scaling matrix), the optimum 2-D similarity transformation can be obtained by solving two 1-D optimization problems separately. In addition, it was shown in [24] that certain multiplications can be eliminated in a local state-space structure realization through the use of an appropriate similarity transformation while keeping the filter free of overflow oscillations and maintaining low, but not minimum, roundoff noise.

4.3 Implementing the Recursive Filters Designed by Transformation

The implementation of the 2-D system differs from the 1-D counterpart in the sense that the ordering of data processing for the 2-D system is not unique whereas the ordering of data processing for 1-D is fixed. In the 2-D system, the data array can be processed row-by-row, column-by-column, or in a diagonal manner. Recall that the transfer function of a 2-D recursive digital filter obtained by a transformation technique is a cascade of different quarter plane filters. Since each quarter plane filter has a different "region of support", the direction of recursion must be properly chosen. The stable recursive directions for the four quarter-plane filters [27] are drawn in Fig.4.3.1.



NW - Northwest; SW - Southwest

SE - Southeast; NE - Northeast

Fig. 4.3.1. Stable Recursive Directions for the Four Quarter-Planes

The transfer function of the circularly symmetric lowpass digital filter discussed in Chapter II is

$$T(Z_1, Z_2) = G[H(Z_1)H(Z_2)]G[H(Z_1)H(Z_2^{-1})]G[H(Z_1)]G[H(Z_2)] \qquad (4.3.1)$$

This transfer function consists of four subfilters, first and fourth-quadrant quarter plane 2-D filters, and two 1-D filters in cascade. Fig.4.3.1 indicates that in order to realize $G[H(Z_1)H(Z_2)]$, the input array must be processed starting from the north-west corner, either row-by-row, column-by-column, or in a diagonal manner. Similarly, for $G[H(Z_1)H(Z_2^{-1})]$, the input array must be processed from the north-east corner. In order to be able to process the input array in the same direction, $G[H(Z_1)H(Z_2^{-1})]$ can be realized by performing an appropriate data transformation before and after the $G[H(Z_1)H(Z_2)]$ operation. Fig.4.3.2 shows a block diagram of realizing $G[H(Z_1)H(Z_2^{-1})]$ by data transformation.

As for the direction of data processing of $G[H(Z_1)]G[H(Z_2)]$, the input array must be processed column-by-column using $G[H(Z_1)]$, row-by-row using $G[H(Z_2)]$. Either the column operations or the row operations can be performed first. The block diagram of realizing the overall transfer function $T(Z_1, Z_2)$ can be drawn as in Fig.4.3.3.

The zero-phase digital filter $T_0(Z_1, Z_2)$, if desired, can be realized either in cascade or parallel as follows. For a cascade realization

$$T_{0_c}(Z_1, Z_2) = T(Z_1, Z_2)T(Z_1^{-1}, Z_2^{-1}) \qquad (4.3.2a)$$

This gives a magnitude response equal to $|T(Z_1, Z_2)|^2$. For the parallel realization

$$T_{0_p}(Z_1, Z_2) = T(Z_1, Z_2) + T(Z_1^{-1}, Z_2^{-1}) \qquad (4.3.2b)$$

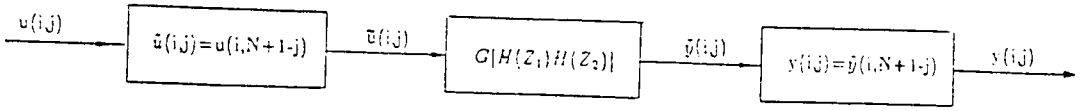


Fig. 4.3.2. Block Diagram of Realizing $G\,H(Z_1)H(Z_2^{-1})$

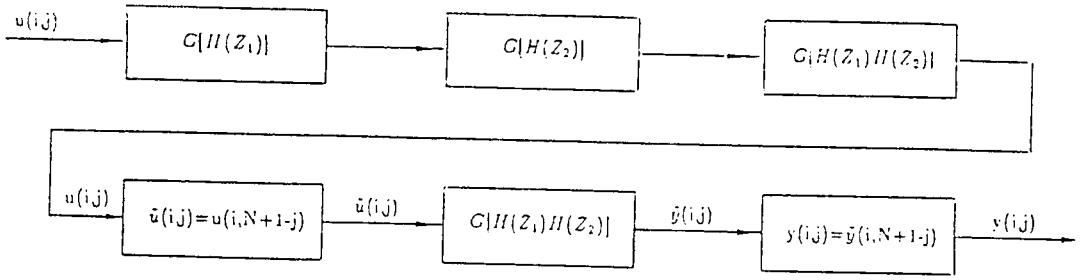


Fig. 4.3.3. Block Diagram of Realizing $T(Z_1, Z_2)$

Although this is also zero-phase, the magnitude response is $2\,Re(T(Z_1, Z_2))$ which may not be an acceptable response. The block diagram of realizing $T(Z_1^{-1}, Z_2^{-1})$ is drawn in Fig.4.3.4.

Since the data array must be transformed twice for every noncausal quarter-plane implemented, six data transformations are needed to realize (4.3.2). This requires a great deal of data shufflings, particularly if the array is large. Fortunately, [29] showed that a certain number of data transformations can be reduced in implementing a cascade of spectrally transformed sections by combining two transformations into one. In this case, only four data transformations are needed to realize (4.3.2a), and five for (4.3.2b). Fig.4.3.5 shows the block diagram of realizing (4.3.2a) with a minimum number of data transformations.

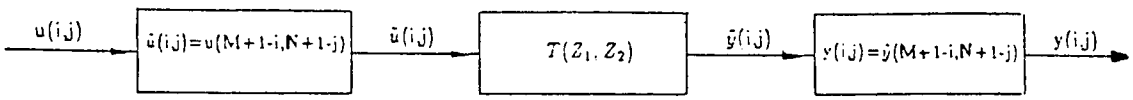


Fig. 4.3.4. Block Diagram of Realizing $T(Z_1^{-1}, Z_2^{-1})$

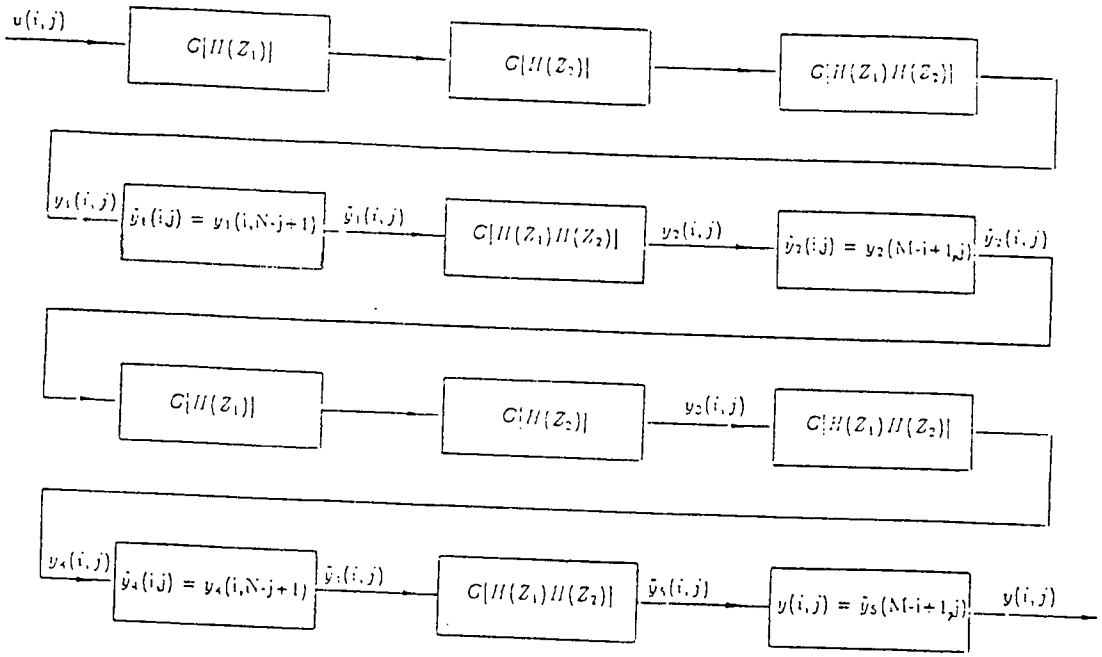


Fig. 4.3.5. Block Diagram of Realizing $T_{0_c}(Z_1, Z_2)$

CHAPTER V

COMPUTER PROGRAMS

5.1 Features of the Programs

In this chapter, two interactive computer programs (2DIIRLP., 2DXFORM.), written in FORTRAN using double precision arithmetic, based on the technique proposed by [15] and [16], are presented. These two programs combined provide the facility to design stable 2-D circularly symmetric lowpass, highpass, bandpass, and bandstop recursive digital filters with user's specifications of filter order and cut-off band edges. In addition, the canonical real-gain form based on Roesser's local state-space structure for a general M×N order transfer function is provided. The NCAR plot package is used in both programs to provide the magnitude squared response and contour plots. Input and output coefficients are also printed.

5.2 Program Structure of 2DIIRLP.

The program is written for the design of 2-D circularly symmetric lowpass recursive digital filters proposed by [15]. The filters are designed as a cascade of 2-D second-order sections given a proper 1-D transfer function. Users need only to specify the filter's order, the value of a, and provide the numerator and denominator coefficients of the second-order sections prototype. If the filter order is odd, the 1^{st}-order section must be read last. Given the desired 2-D cut-off contour, the appropriate 1-D frequency cut-off along with the parameter a are obtained by executing the 1D2DWC. program. The canonical minimal real-gain Roesser's local state-space structure of the resulting 1×1 and 2×2 sections are also given. As for

the 1-D second-order section, it is realized by an efficient low-roundoff-noise state-space structure, called the Type III structure in [25]. This structure is chosen over other structures because it yields a filter with a bandwidth-invariant round-off noise level very close to the minimum level while requiring one less multiply and one less addition than the minimum-noise structure. The program's listings of 2DIIRLP. and 1D2DWC. appear in Appendix A and Appendix C respectively.

5.3 Program Structure of 2DXFORM.

This program provides the facility to quickly design 2-D stable lowpass, high-pass, bandpass, and bandstop recursive digital filters upon the availability of a stable 2-D lowpass digital filter.

The program accommodates the design of M×N 2-D IIR filters via digital spectral transformation in general, as well as a cascade of 1-D and 2-D second-order sections as arises from 2DIIRLP. As for the realization aspect, the canonical minimal real-gain Roesser's local state-space structure is provided. If either bandpass or bandstop design is desired and the lowpass prototype is taken from the 2DIIRLP. the resulting 4^{th}-order separable filters are then factored into two 2^{nd}-order sections in cascade, and realized by the Type III structure.

To run the program, users need to specify the filter's order, the cut-off contour to be transformed, and the desired band edges for Z_1 and Z_2. If the filter order is odd, the 1^{st}-order section must be entered last. The listing of the program is given in Appendix B.

CHAPTER VI

SUMMARY AND CONCLUSIONS

A survey of different techniques for designing 2-D lowpass circularly symmetric recursive digital filters via spectral transformation was undertaken. The design technique presented in [15] was selected as the most satisfactory and efficient in terms of the design simplicity, number of coefficients involved and performance. Moreover, the technique can be utilized to design elliptically shaped lowpass and fan filters. A FORTRAN computer program was written to implement the method. Several filters were designed to demonstrate the practicality of the technique.

Another program was developed to quickly design other filter types (highpass, bandpass, and bandstop) by applying stable 2-D to 2-D DST functions. Even though only one point on the contour was controlled and the DST functions were nonlinear, practical design examples showed that the shapes of the transformed filters was acceptable.

The implementation of the filters was discussed in Chapter IV. The stable recursive directions for the quarter-planes have been presented to explain how a transfer function which consists of causal and noncausal filters as discussed in Chapter II can be implemented by proper data orientations. The 2-D realization structure suggested by [15] was shown to be not directly realizable because of the presence of a delay-free loop. The generalized canonical minimal real-gain form based on Roesser's local state-space model was included in the computer programs as a basis for realizing general 2-D recursive digital filters.

Chapter V discussed the structures and features of the computer programs.

The programs were written in FORTRAN, using double precision arithmetic. The computer programs were provided in the thesis in a hope that they may be useful to others.

Further research should be done to develop a technique for the design of circularly symmetric highpass recursive digital filters using 1-D to 2-D DST functions similar to the lowpass design. The single circular contour approximation may be extended to multiple contours. Furthermore, a suitable similarity transformation matrix T should be found so that a state-space structure which possesses low round-off noise can be obtained with less computation load than that obtained by the methods of [22,23]. This would provide efficient low-noise 2-D structures similar to those developed in [25,26] for 1-D filters.

REFERENCES

REFERENCES

1. D. E. Dudgeon and R. M. Mersereau, "Multidimensional Digital Signal Processing," Englewood Cliffs, NJ: Prentice-Hall, 1984.

2. A.V. Oppenheim and R.W. Schafer, "Digital Signal Processing," Englewood Cliffs, NJ: Prentice-Hall, 1975.

3. A. Antoniou, "Digital Filters: Analysis and Design," New York: McGraw-Hill, 1979.

4. L. R. Rabiner and B. Gold, "Theory and Application of Digital Signal Processing," Englewood Cliffs, NJ: Prentice-Hall, 1975.

5. J.L. Shanks, S. Treitel and J.H. Justice, "Stability and Synthesis of Two-Dimensional Recursive Filters," IEEE Trans. on Audio and Electroacoustics, vol. AU-20, No. 2, pp. 115-128, June 1972.

6. J.M. Costa and A.N. Venetsanopoulos, "Design of Circularly Symmetric Two-Dimensional Recursive Filters," IEEE Trans. on Acoustics, Speech, and Signal Processing, vol. ASSP-22, No. 6, pp. 432-443, Dec. 1974.

7. D.M. Goodman, "A Design Technique for Circularly Symmetric Low-pass Filters", IEEE Trans. on Acoustics, Speech, and Signal Processing, vol. ASSP-26, pp. 290-304, Aug. 1978.

8. H. Chang and J.K. Aggarwal, "Design of 2-D Recursive Filters by Interpolation," IEEE Trans. on Circuits and Systems, vol. CAS-24, pp. 874-881, June 1977.

9. F. Bernabo, P.L. Emiliani and V. Cappellini, "Design of 2-Dimensional Recursive Digital Filters," Electronics Letters, vol. 12, No. 11, pp. 288-289, May 27, 1976.

10. R. Iijima, N. Haratani, and S.I. Takahashi, "Design Method for 2-D Circularly Symmetric Recursive filters," IEEE Trans. on Acoustics, Speech, and Signal Processing, vol. ASSP-31, No. 5, pp. 1298-1299, Oct. 1983.

11. D. Thong Nguyen and M.N.S. Swamy, "A Class of 2-D Separable Denominator Filters Designed via the McClellan Transform," IEEE Trans. on Circuits and Systems, vol. CAS-33, No. 9, pp. 874-881, Sept. 1986.

12. J.H. McClellan, "The Design of 2-D Digital Filters by Transformations," in Proceedings 7^{th} Annual Princeton Conference Inf. Sciences and Systems, pp. 247-251, 1973.

13. N.A. Pendergrass and S.K. Mitra, "Spectral Transformation for 2-D Recursive Digital Filters," IEEE Trans. on Circuits and Systems, vol. CAS-23, No.1, pp. 26-35, Jan. 1976.

14. S. Chakrabarti and S. Mitra, "Design of 2-D Digital Filters via Spectral Transformations," Proceedings of the IEEE, vol.69. No. 6, pp. 905-914, June 1977.

15. Lein Harn, and B.A. Shenoi, "Design of Stable Two-Dimensional IIR Filters Using Digital Spectral Transformations," IEEE Trans. on Circuits and Systems, vol. CAS-33. No. 5, pp. 483-490, May 1986.

16. Lein Harn, and B.A. Shenoi, "On Stable Digital Frequency Transformations for Designing 2-D Digital Filters," in Proceedings of 27^{th} Midwest Symp. on Circuits and Systems, pp. 319-322, Morgantown, 1984.

17. A.G. Constantinides, "Spectral Transformations for Digital Filters," Proceeding The Institution of Electrical Engineering, vol. 117, pp.1585-1590, Aug. 1970.

18. G.A. Maria and M.M. Fahmy. "An l_p Design Technique for 2-D Digital Recursive Filters," IEEE Trans. on Acoustics, Speech, and Signal Processing, vol. ASSP-22, No. 1, pp. 15-21. Feb. 1974.

19. T. Lin, M. Kawamata, and T. Higuchi,"Design of 2-D Digital Filters With an Arbitrariry Response and No Overflow Oscillations Based on a New Stability Condition," IEEE Trans. on Circuits and Systems, vol. CAS-34, No. 2, pp. 113-126. Feb. 1987.

20. S.Y. Kung, B.C. Levy, M. Morf, and T. Kailath, "New Results in 2-D Systems Theory, PartII: 2-D State-Space Models - Realization and the Notions of Controllability, Observability, and Minimality," Proceedings of the IEEE, vol. 65, No. 6, pp. 945-961, June 1977.

21. R.P. Roesser, "A Discrete State-space Model for Linear Image Processing," IEEE Trans. Automatic Control, vol. AC-20, pp.1-10. Feb. 1975.

22. T. Lin, M. Kawamata, and T. Higushi, "A Unified Study on the Roundoff Noise in 2-D State-Space Digital Filters," IEEE Trans. on Circuit and Systems, vol. CAS-33, No. 7, pp. 724-730, July 1986.

23. W.S. Lu, "Synthesis of 2-D State-Space Fixed-Point Digital-Filter Structures with Minimum Roundoff Noise," IEEE Trans. on Circuits and Systems, vol. CAS-33, No. 10, pp. 965-973, Oct. 1986.

24. W.S. Lu, "2-D State-Space Digital Filters with Fewer Multipliers," IEEE Trans. on Circuits and Systems, vol. CAS-33, No. 9, pp. 882-891, Sep. 1986.

25. B.W. Bomar, "New Second-Order State-Space Structures for Realizing Low Roundoff Noise Digital Filters," IEEE Trans. on Acoustics, Speech, and Signal Processing, vol. ASSP-33, No. 1, pp. 106-110. Feb. 1985.

26. B.W. Bomar, "Computationally Efficient Low Roundoff Noise Second-Order State-Space Structures," IEEE Trans. on Circuit and Systems, vol. CAS-33, No. 1, pp. 35-41, Jan. 1986.

27. N. Nagamuthu, "Transformation Design of 2-D Recursive Digital Filters Using Spectral Factorization." Master's Thesis, University of Windsor, 1982.

28. T.S. Huang, J.W. Burnett, and A.G. Deczky, "The Importance of Phase in Image Processing Filters," IEEE Trans. Acoustics, Speech, Signal Processing, vol. ASSP-23, pp. 529-542, Dec. 1975.

29. K. P. Prasad, A. Antoniou, and B. B. Bhattacharyya, "On the Properties of Linear Spectral Transformations for 2-Dimensional Digital Filters." Circuit Systems Signal Process, vol. 2, No. 2, pp. 203-211, 1983.

30. S. Chakrabarti, and S. K. Mitra, "Decision Methods and Realization of 2-D Digital Filters Using Minimum Number of Delay Elements," IEEE Trans. on Circuits and Systems, vol. CAS-27, No. 8, pp. 657-666, Aug. 1980.

31. E. Fornasini, "On the Relevance of Noncommutative Power Series in Spatial Filters Realization." IEEE Trans. on Circuits and Systems, vol. CAS-25, No. 5, pp. 290-299, May, 1978.

32. N.K. Bose, "Digital Filters: Theory and Applications," New York: North Holland, 1985.

APPENDIXES

APPENDIX A

PROGRAM'S LISTING OF 2DIIRLP.

```
      double precision b, c, prectan, qrectan,
     *                 pghz1, qghz1,
     *                 pghz2, qghz2, scale,
     *                 pghz12, qghz12,
     *                 aa, bb, cc, dd
c
c Program: 2DIIRLP.FOR
c
c ***********************************************
c                Toadsak Vongtanaanek
c
c      2-D Lowpass Recursive Digital Filter Design
c      via 1-D to 2-D Digital Spectral Transformation
c
c ***********************************************
c
c     One-dimensional low-pass recursive filter is transformed
c to two-dimensional circularly symmetric lowpass digital filter via
c Digital Spectral Transformation. The transformation used here is lp-lp
c spectral transformation which takes the following all-pass function:
c Z^-1 <- (Z^-1 - a)/(a Z^-1 - 1)
c The value of "a" is not to be greater or equal to 1 for the
c resulting filters to be stable. Take round-off error into account.
c suggested maximum value of "a" is less than or equal 0.9
c
c Note:
c ==> Filter must be a cascade of second-order sections.
c If filter order is odd, the first-order subfilter must be the
c last one to be read.
c ==> If specified, the coefficients will be written in unit 7 ready
c to be read by 2DXFORM.FOR program.
c ==> If desired, the frequency response will be computed.
c     The plots are made by NCAR.
c     The contour and surface plots can be obtained from Versatex by executing
the following commands
c
c          ASSIGN NUL SYS$PRINT
c          RUN SYS$SYSTEM:MCTRAN
c          IOP020.DAT
c
c
c
```

```
      dimension b(3), c(3), prectan(3,3), qrectan(3,3),
     *                  pghz1(3), qghz1(3),
     *                  pghz2(3), qghz2(3),
     *                  pghz12(3,3), qghz12(3,3)
      dimension aa(6,6), bb(6), cc(6)
      common /a/ h(40,40)
c
      write(5,*) ' filter order:'
      read(5,*) nfil
      write(5,*) ' enter a:'
      read(5,*) a
      a2 = a*a
      a3 = a2*a
      a4 = a3*a
      write(5,*) ' Save the transformed coefficients?'
      write(5,*) ' Yes ====> 1'
      read (5,*) isave
      write(5,*) ' Frequency response needed? '
      write(5,*) ' Yes ====> 1'
      read (5,*) ifrq
      write(5,*) ' gain:'
      read(9,*) gain
      if (ifrq.ne.1) goto 2
c
c ------- Initialize the frequency response -------
c
      npnts = 40
      do 600 i = 1, npnts
      do 600 j = 1, npnts
      h(i,j) = 1.
600 continue
c ---------------------------------------------
c
 2    n = 2
      if (nfil.eq.1) n = 1
      n1 = n - 1
      nodd = mod(nfil,2)
      nsect = (nfil-1)/2
      write(5,*) ' nsect = ',nsect
 1    write(5,*) ' *** corresponds to power of Z^-1 *** '
      write(5,*) ' '
      write(5,*) ' *** enter numerator b(i) ***'
      do 20 j = 1, n1
      read(9,*) b(j)
20    continue
      if (gain.eq.1.) goto 22
      do j = 1, n1
      b(j) = b(j)*gain
      enddo
```

```
c
22    write(5,*) ' *** enter denominator c(i) ***'
      do 21 j = 1, n1
      read(9,*) c(j)
21    continue
c
      write(5,*) ' Check the coefficients b(i), c(i). '
      write(5,*) ' '
      write(5,101) (b(j), j = 1, n1)
      write(5,101) (c(j), j = 1, n1)
 101  format(3f21.17)
      write(5,*) ' '
c
c
c ———————      Compute G[H(z2)]      ———————
c
      if (n1.eq.3) then
      pghz2(1) = b(1) + a*b(2) + a2*b(3)
      pghz2(2) = 2.*a*b(1) + (1. + a2)*b(2) + 2.*a*b(3)
      pghz2(3) = a2*b(1) + a*b(2) + b(3)
      qghz2(1) = c(1) + a*c(2) + a2*c(3)
      qghz2(2) = 2.*a*c(1) + (1. + a2)*c(2) + 2.*a*c(3)
      qghz2(3) = a2*c(1) + a*c(2) + c(3)
      else
      pghz2(1) = b(1) + a*b(2)
      pghz2(2) = a*b(1) + b(2)
      qghz2(1) = c(1) + a*c(2)
      qghz2(2) = a*c(1) + c(2)
      endif
      scale = qghz2(1)
      do i = 1, n1
      pghz2(i) = pghz2(i)/scale
      qghz2(i) = qghz2(i)/scale
      enddo
      qghz2(1) = 1.0d0
c
c ————————————————————————————
      write(5,*) ' '
      write(5,*) ' Transformed Separable Filter, G[H(Z_i)]'
      write(5,*) ' '
      write(5,101) (pghz2(i), i = 1, n1)
      write(5,*) ' '
      write(5,101) (qghz2(i), i = 1, n1)
c
      if (isave.ne.1) goto 9
      write(7,*) (pghz2(i), i = 1, n1)
      write(7,*) (qghz2(i), i = 1, n1)
c
c
```

```
 9    if (ifrq.ne.1) goto 3
c
c —— Computing G[H(z1)] (if frequency plot needed) ——
c
      do i = 1, n1
      pghz1(i) = pghz2(i)
      qghz1(i) = qghz2(i)
      enddo
c
c
c     Multiply separable rectangular filters (if frqres needed)
c
      call mult1 (pghz1, pghz2, n1, prectan)
      call mult1 (qghz1, qghz2, n1, qrectan)
      call frqrsp (prectan, qrectan, n1, npnts)
c
c ————————————————————————————
c
c ——— Compute G[H(z1)H(z2)] ———
c
c
 3    if (n1.eq.3) then
      pghz12(1,1) = b(1) - a2*b(2) + a4*b(3)
      pghz12(1,2) = 2.*a*b(1) - (a - a3)*b(2) + 2.*a3*b(3)
      pghz12(1,3) = a2*(b(1) - b(2) - b(3))
      pghz12(2,1) = pghz12(1,2)
      pghz12(2,2) = 4.*a2*b(1) - (1. - 2.*a2 + a4)*b(2) - 4.*a2*b(3)
      pghz12(2,3) = 2.*a3*b(1) - (a - a3)*b(2) + 2.*a*b(3)
      pghz12(3,1) = pghz12(1,3)
      pghz12(3,2) = pghz12(2,3)
      pghz12(3,3) = a4*b(1) + a2*b(2) - b(3)
c
      qghz12(1,1) = c(1) + a2*c(2) - a4*c(3)
      qghz12(1,2) = 2.*a*c(1) - (a - a3)*c(2) - 2.*a3*c(3)
      qghz12(1,3) = a2*(c(1) - c(2) + c(3))
      qghz12(2,1) = qghz12(1,2)
      qghz12(2,2) = 4.*a2*c(1) - (1. - 2.*a2 + a4)*c(2) - 4.*a2*c(3)
      qghz12(2,3) = 2.*a3*c(1) - (a - a3)*c(2) + 2.*a*c(3)
      qghz12(3,1) = qghz12(1,3)
      qghz12(3,2) = qghz12(2,3)
      qghz12(3,3) = a4*c(1) - a2*c(2) - c(3)
      else
      pghz12(1,1) = b(1) - b(2)*a2
      pghz12(1,2) = (b(1) - b(2))*a
      pghz12(2,1) = (b(1) - b(2))*a
      pghz12(2,2) = b(1)*a2 - b(2)
c
      qghz12(1,1) = c(1) - c(2)*a2
      qghz12(1,2) = (c(1) - c(2))*a
```

```
      qghz12(2,1) = (c(1) + c(2))*a
      qghz12(2,2) = c(1)*a2 + c(2)
      endif
      scale = qghz12(1,1)
      do i = 1, n1
      do j = 1, n1
      pghz12(i,j) = pghz12(i,j)/scale
      qghz12(i,j) = qghz12(i,j)/scale
      enddo
      enddo
      qghz12(1,1) = 1.0d0
c
      write(5,*) ' '
      write(5,*) ' '
      write(5,*) ' Numerator of G[H(Z1)H(Z2)]'
      do i = 1, n1
      write(5,101) (pghz12(i,j), j = 1, n1)
      enddo
      write(5,*) ' '
      write(5,*) ' Denominator of G[H(Z1)H(Z2)]'
      do i = 1, n1
      write(5,101) (qghz12(i,j), j = 1, n1)
      enddo c
      if (isave.ne.1) goto 6
      do i = 1, n1
      write(7,*) (pghz12(i,j), j = 1, n1)
      enddo
      do i = 1, n1
      write(7,*) (qghz12(i,j), j = 1, n1)
      enddo
 6    if (ifrq.ne.1) goto 4
      call frqrsp (pghz12, qghz12, n1, npnts)
c
c ---------------------------------------------------
c
c ---------------------------------------------------
c
c - Compute G[H(z1)H(1/z2)] (if frqres needed) -
c
c
      if (n1.eq.3) then
      do i = 1, n1
      prectan(i,3) = pghz12(i,1)
      prectan(i,1) = pghz12(i,3)
      prectan(i,2) = pghz12(i,2)
      qrectan(i,3) = qghz12(i,1)
      qrectan(i,1) = qghz12(i,3)
      qrectan(i,2) = qghz12(i,2)
      enddo else
```

```
      do i = 1, n1
      prectan(i,1) = pghz12(i,2)
      prectan(i,2) = pghz12(i,1)
      qrectan(i,1) = qghz12(i,2)
      qrectan(i,2) = qghz12(i,1)
      enddo
      endif
c
c ---------------------------------------------
      call frqrsp (prectan, qrectan, n1, npnts)
c
c ---------------------------------------------
c
c       To obtain State Space Structure
c       -------------------------------
c
c - For G[H(Zi)], Type 3 structure (as in IEEE, Feb,1985) is used.
c - For G[H(Z1)H(Z2)], Canonical Roesser Model is used.
c
c ---------------------------------------------------- ----------
c
c    Type 3 Structure for G H(Zi)
c    ----------------------------
c
 4    if (n1.eq.2) then
c
c    First-Order Section Realization
c    -------------------------------
      d = pghz2(1)
      alpha = pghz2(2) - pghz2(1)*qghz2(2)
      beta = qghz2(2)
      a1 = -beta
      b1 = sqrt(1. - beta*beta)
      c1 = alpha/beta
      write(5,*) ' '
      write(5,*) ' '
      write(5,*) ' *** State Space Structure for G H(Zi) ***'
      write(5,*) ' '
      write(5,*) ' '
 902 format (' A-Vector ==> ', f13.9)
      write(5,902) a1
      write(5,*) ' '
      write(5,903) b1
 903 format (' b-Vector ===> ', f13.9)
      write(5,*) ' '
      write(5,904) c1
 904 format (' c-Vector ===> ', f13.9)
      write(5,*) ' '
      write(5,920) d
```

```
 920 format(' d ===> ', f13.9)
     else
c
c
c    Second-Order Section Realization
c    --------------------------------
     d = pghz2(1)
     alpha1 = pghz2(2) - pghz2(1)*qghz2(2)
     alpha2 = pghz2(3) - pghz2(1)*qghz2(3)
     beta1 = qghz2(2)
     beta2 = qghz2(3)
c
c
c    Get the Type 3 Structure (second-order)
c    ---------------------------------------
     temp = beta1/2.
     temp2 = temp*temp
     a11 = -temp
     a12 = sqrt(1. - (temp2*(beta2 - 3.)/(beta2 + 1.)))
     a21 = (temp2 - beta2)/a12
     a22 = a11
     b1 = 0.
     b2 = (1. - beta2)*((1. - beta2)**2 - beta1*beta1)
     b2 = sqrt(b2/((1. - beta2)*(1. - temp2) - beta1*beta1))
     c1 = (alpha2 - a11*alpha1)/(a12*b2)
     c2 = alpha1/b2
c
     write(5,*) ' '
     write(5,*) ' '
     write(5,*) ' '
     write(5,*) ' *** State Space Structure for G*H(Zi) ***'
     write(5,*) ' '
     write(5,*) ' --- A-Matrix --- '
     write(5,*) ' '
 900 format(2f13.9)
     write(5,900) a11, a12
     write(5,900) a21, a22
     write(5,*) ' '
     write(5,905) b1, b2
 905 format(' b-Vector ===> ', 2f13.9)
     write(5,*) ' '
     write(5,915) c1, c2
 915 format (' c-Vector ===> ', 2f13.9)
     write(5,*) ' '
     write(5,920) d
     endif
     write(5,*) ' '
c
c
```

```
c   Get State Space Structure for G[H(Z1)H(Z2)]
c   ------------------------------------------
c
      if (n1.eq.2) then
c
c   1×1 Order Transfer Function
c   ---------------------------
c
      aa(1,1) = -qghz12(2,1)
      aa(1,2) = 1.
      aa(1,3) = 0.
      aa(2,1) = qghz12(2,1)*qghz12(1,2) - qghz12(2,2)
      aa(2,2) = -qghz12(1,2)
      aa(2,3) = 0.
      aa(3,1) = pghz12(2,2) - pghz12(1,2)*qghz12(2,1)
      aa(3,2) = pghz12(1,2)
      aa(3,3) = 0.
      bb(1) = 1.
      bb(2) = -qghz12(1,2)
      bb(3) = pghz12(1,2)
      cc(1) = pghz12(2,1) - pghz12(1,1)*qghz12(2,1)
      cc(2) = pghz12(1,1)
      cc(3) = 1.
      else
c
c   2×2 Order Transfer Function
c   ---------------------------
      aa(1,1) = -qghz12(2,1)
      aa(1,2) = -qghz12(3,1)
      aa(1,3) = 1.
      aa(1,4) = 0.
      aa(1,5) = 0.
      aa(1,6) = 0.
      aa(2,1) = 1.
      aa(2,2) = 0.
      aa(2,3) = 0.
      aa(2,4) = 0.
      aa(2,5) = 0.
      aa(2,6) = 0.
      aa(3,1) = qghz12(2,1)*qghz12(1,2) - qghz12(2,2)
      aa(3,2) = qghz12(3,1)*qghz12(1,2) - qghz12(3,2)
      aa(3,3) = -qghz12(1,2)
      aa(3,4) = 1.
      aa(3,5) = 0.
      aa(3,6) = 0.
      aa(4,1) = qghz12(2,1)*qghz12(1,3) - qghz12(2,3)
      aa(4,2) = qghz12(3,1)*qghz12(1,3) - qghz12(3,3)
      aa(4,3) = -qghz12(1,3)
      aa(4,4) = 0.
```

```
      aa(4,5) = 0.
      aa(4,6) = 0.
      aa(5,1) = pghz12(2,2) - pghz12(1,2)*qghz12(2,1)
      aa(5,2) = pghz12(3,2) - pghz12(1,2)*qghz12(3,1)
      aa(5,3) = pghz12(1,2)
      aa(5,4) = 0.
      aa(5,5) = 0.
      aa(5,6) = 1.
      aa(6,1) = pghz12(2,3) - pghz12(1,3)*qghz12(2,1)
      aa(6,2) = pghz12(3,3) - pghz12(1,3)*qghz12(3,1)
      aa(6,3) = pghz12(1,3)
      aa(6,4) = 0.
      aa(6,5) = 0.
      aa(6,6) = 0.
c
      bb(1) = 1.
      bb(2) = 0.
      bb(3) = -qghz12(1,2)
      bb(4) = -qghz12(1,3)
      bb(5) = pghz12(1,2)
      bb(6) = pghz12(1,3)
c
      cc(1) = pghz12(2,1) - pghz12(1,1)*qghz12(2,1)
      cc(2) = pghz12(3,1) - pghz12(1,1)*qghz12(3,1)
      cc(3) = pghz12(1,1)
      cc(4) = 0.
      cc(5) = 1.
      cc(6) = 0.
      endif
c
      dd = pghz12(1,1)
c
      iss = 3*(n1 - 1)
      write(5,*) ' '
      write(5,*) ' '
      write(5,*) ' *** Canonical Structure for G H(Z1)H(Z2) ***'
      write(5,*) ' '
      write(5,*) '          ----- A-Matrix -----'
      write(5,*) ' '
      do i = 1, iss
      write(5,910) (aa(i,j), j = 1, iss)
      enddo
  910 format (6f13.9)
  911 format (6f13.9)
      write(5,*) ' '
      write(5,*) ' '
      write(5,*) '          ----- b-Vector -----'
      write(5,*) ' '
      write(5,911) (bb(i), i = 1, iss)
```

```
      write(5,*) ' '
      write(5,*) ' '
      write(5,*) '          ----- c-Vector -----'
      write(5,*) ' '
      write(5,911) (cc(i), i = 1, iss)
      write(5,*) ' '
      write(5,935) dd
  935 format(' d ===> ', f15.9)
c
c
c
  5   nsect = nsect - 1
      gain = 1.
      if ((nsect.eq.1).and.(nodd.eq.1)) n1 = 2
      if (nsect.ne.0.) goto 1
      if (ifrq.ne.1) stop
      call ncarplt
      stop
      end
c
c
c
c
c
      subroutine ncarplt
      common /b/ w1(40), w2(40)
      common /a/ h(40,40)
      wi = 2./39.
      w1(1) = -1.
      w2(1) = -1.
      do i = 2, 40
      w1(i) = w1(i-1) - wi
      w2(i) = w1(i)
      enddo
c     call frame
c     call ezcntr(h, 40, 40)
      call tsrfac (ierror)
      if(ierror.ne.0) type *, ' error'
      return
      end
c
      subroutine tsrfac (ierror)
      real work(3280), s(6)
      common /b/ w1(40), w2(40)
      common /a/ h(40,40)
      data s(1), s(2), s(3), s(4), s(5), s(6)/
     *   -8.0, -6.0, 3.0, 0.0, 0.0, 0.0/
      data angh /45./, angv /15./
      data ix/405/, iy/1000/
```

```
      ierror = 0
c     call ezsrfc (h, 40, 40, angh, angv, work)
c     call ezcntr (h, 40, 40)
      call conrec (h, 40, 40, 40, 0., 1., .1, 0, 0, 0)
      call frame
      call srface (w1, w2, h, work, 40, 40, 40, s, 0.)
      return
      end
c
c
c
      subroutine mult1 (x, y, number, result)
      double precision x, y, result
      dimension x(3), y(3), result(3,3)
      do j = 1, number
      do k = 1, number
      result(j,k) = x(j)*y(k)
      enddo
      enddo
      return
      end
c
c
c
      subroutine frqrsp (p, q, ndim, npnts)
      double precision p, q
      complex*8 z, cn, cd
      dimension p(3,3), q(3,3)
      common /a/ h(40,40)
      pi = 4*atan2(1.,1.)
      wi = 2.*pi/float(npnts-1)
      w1 = -pi
      w2 = w1
      do 1 k = 1, npnts
      do 2 l = 1, npnts
      cn = cmplx(0.,0.)
      cd = cn
      do 3 i = 1, ndim
      angl1 = float(i-1)*w1
      do 4 j = 1, ndim
      angl2 = float(j-1)*w2
      angle = -(angl1 - angl2)
      z = cexp (cmplx(0.,angle))
      cn = cn - sngl(p(i,j))*z
      cd = cd - sngl(q(i,j))*z
 4    continue
 3    continue
      h(k,l) = h(k,l)*(cabs(cn) / cabs(cd))
      w2 = w2 - wi
```

```
2    continue
     w2 = -pi
     w1 = w1 + wi
1    continue
     return
     end
```

APPENDIX B

PROGRAM'S LISTING OF 2DXFORM.

```
      double precision p, q, p1, q1,
     *                 pt1, qt1, pt2, qt2, scale,
     *                 store1, store2,
     *                 pt, qt, temp, gain1d
     *                 pf1, qf1, pf2, qf2
      dimension hn1(3), hd1(3), hn2(3), hd2(3),
     *                 p(3,3), q(3,3), p1(3), q1(3),
     *                 pt1(5), qt1(5), pt2(5), qt2(5),
     *                 store1(5), store2(5),
     *                 pt(5,5), qt(5,5),
     *                 pf1(3), qf1(3), pf2(3), qf2(3)
      common /a/ h(40,40)
c
c
c Program: 2DXFORM.FOR
c
c
c Programmer: Toadsak Vongtanaanek
c
c    The program takes coefficients of two-dimensional low-pass
c digital filters, and transformed the filters to various types
c using "one-variable" digital spectral transformation.
c    The coefficients are assumed to be a cascade of second-order
c sections, and have been previously stored in unit 7.
c If filter order is "odd", the first-order section must be read last.
c
c Read Format of filter coefficients:
c ---------------------------------
c    The numerators of first subfilter are first read starting
c from lowest (inverse) order of Z2 and Z1, then the denominators of first
c subfilter are read. After all the first subfilter's coefficients
c have been read, the next subfilter coefficients are then read,
c and so on.
c
c Output Format of filter coefficients:
c -------------------------------
c    Transformed transfer funtions along with corresponding
c are written. The 2-D transfer function is realized by canonical
c real-gain local state-space structure. For the 1-D subfilters (if
c input is taken from 2DIIRLP.), they are realized by Type III structure
c as a cascade of second-order sections.
c
c Frequency Response Plots:
```

```
c ------------------------
c    The frequency response of the overall filter can be plotted
c by executing the following commands
c            ASSIGN NUL SYS$PRINT
c            RUN SYS$SYSTEM:MCTRAN
c            IOP020.DAT
c
c    The surface and contour plots of the magnitude response will be
c outputted to the Versatex.
c
c
      npnts = 40
      do i = 1, npnts
      do j = 1, npnts
      h(i,j) = 1.
      enddo
      enddo
      write(5,*) ' Subfilters from 2DIIRLP.?'
      write(5,*) ' Type 1 for Yes.'
      read (5,*) idst
      write(5,*) ' Overall filter order'
      read(5,*) nfil
      nsect = (nfil - 1)/2
      nodd = mod(nfil,2)
      n = 2
      if (nfil.eq.1) n = 1
c
 1    write(5,*) ' LP-LP ==> 1 LP-HP ==> 2 '
      write(5,*) ' LP-BP ==> 3 LP-BS ==> 4 '
      write(5,*) ' Type of transformation on Z1:'
      read(5,*) nz1
      write(5,*) ' Type of transformation on Z2:'
      read(5,*) nz2
      if ((nz1.gt.4).or.(nz1.lt.0)) goto 1
      if ((nz2.gt.4).or.(nz2.lt.0)) goto 1
      if (nz1.eq.1) call lplp (1, hn1, hd1)
      if (nz1.eq.2) call lphp (1, hn1, hd1)
      if (nz1.eq.3) call lpbp (1, hn1, hd1)
      if (nz1.eq.4) call lpbs (1, hn1, hd1)
      if (nz2.eq.1) call lplp (2, hn2, hd2)
      if (nz2.eq.2) call lphp (2, hn2, hd2)
      if (nz2.eq.3) call lpbp (2, hn2, hd2)
      if (nz2.eq.4) call lpbs (2, hn2, hd2)
      iv1 = 3
      iv2 = 3
      if (nz1.lt.3) iv1 = 2
      if (nz2.lt.3) iv2 = 2
      write(5,*) ' Digital Spectral Transformation on Z1'
      write(5,*) ' ------------------------------------
```

```
      write(5,111) (hn1(i),i=1,iv1)
      write(5,111) (hd1(i),i=1,iv1)
      write(5,*) ' '
      write(5,*) ' '
      write(5,*) ' Digital Spectral Transformation on Z2'
      write(5,*) ' ------------------------------------'
      write(5,111) (hn2(i),i=1,iv2)
      write(5,111) (hd2(i),i=1,iv2)
      write(5,*) ' '
      write(5,*) ' '
  111 format (5f18.12)
      write(5,*) ' Gain constant:'
      read(5,*) gain
c
    2 n1 = n - 1
      nt1 = n*(iv1 - 1) - 1
      nt2 = n*(iv2 - 1) - 1
      write(5,*) ' nt1 = ',nt1
      write(5,*) ' nt2 = ',nt2
c
      if (idst.ne.1) goto 4
      read(7,*) (p1(i), i = 1, n1)
      read(7,*) (q1(i), i = 1, n1)
      write(5,*) ' '
      write(5,*) ' input coefficients of 1-D filter '
      write(5,*) ' ---------------------------------'
      write(5,101) (p1(i), i = 1, n1)
      write(5,101) (q1(i), i = 1, n1)
      do i = 1, n1
      nh11 = i - 1
      nh12 = n - nh11
      call mltiply (hn1, hd1, iv1, nh11, nh12, store1)
      call mltiply (hn2, hd2, iv2, nh11, nh12, store2)
      do j = 1, nt1
      pt1(j) = pt1(j) - p1(i)*store1(j)
      qt1(j) = qt1(j) - q1(i)*store1(j)
      enddo
      do j = 1, nt2
      pt2(j) = pt2(j) - p1(i)*store2(j)
      qt2(j) = qt2(j) - q1(i)*store2(j)
      enddo
      enddo
c
      scale = qt1(1)
      do i = 1, nt1
      pt1(i) = pt1(i)/scale
      qt1(i) = qt1(i)/scale
      enddo
      qt1(1) = 1.0d0
```

```
      scale = qt2(1)
      do i = 1, nt2
      pt2(i) = pt2(i)/scale
      qt2(i) = qt2(i)/scale
      enddo
      qt2(1) = 1.0d0
c ------------------------------------------------------------
      write(5,*) ' '
      write(5,*) ' Transformed 1-D filter on Z1'
      write(5,*) ' ----------------------------'
      write(5,101) (pt1(i), i = 1, nt1)
      write(5,101) (qt1(i), i = 1, nt1)
      write(5,*) ' '
      write(5,*) ' Transformed 1-D filter on Z2'
      write(5,*) ' ----------------------------'
      write(5,101) (pt2(i), i = 1, nt2)
      write(5,101) (qt2(i), i = 1, nt2)
      write(5,*) ' '
  101 format (5f15.10)
c
      if (nt1.gt.3) then
      write(5,*) '      Two factored 2nd-order sections on Z1'
      write(5,*) '      ------------------------------------'
      gain1d = pt1(1)
      call factor (gain1d, hn1, hd1, p1, q1, pf1, qf1, pf2, qf2)
      call ss1d (pf1, qf1, 3)
      call ss1d (pf2, qf2, 3)
      else
      call ss1D (pt1, qt1, nt1)
      endif
      if (nt2.gt.3) then
      write(5,*) '      Two factored 2nd-order sections on Z2'
      write(5,*) '      ------------------------------------'
      gain1d = pt2(1)
      call factor (gain1d, hn2, hd2, p1, q1, pf1, qf1, pf2, qf2)
      call ss1d (pf1, qf1, 3)
      call ss1d (pf2, qf2, 3)
      else
      call ss1D (pt2, qt2, nt2)
      endif
c
c ----    Multiply 2 1-D filters    ----
c
      call mult1 (pt1, pt2, nt1, nt2, pt)
      call mult1 (qt1, qt2, nt1, nt2, qt)
      call frqrsp (pt, qt, nt1, nt2, npnts)
c
    4 do i = 1, 5
      store1(i) = 0.
```

```
      store2(i) = 0.
      do j = 1, 5
      pt(i,j) = 0.
      qt(i,j) = 0.
      enddo
      enddo
c
      write(5,*) ' Numerator coefficients being read from unit 7'
      do i = 1, n1
      read(7,*) (p(i,j), j = 1, n1)
      enddo
      enddo
      if (gain.eq.1.) goto 345
      do i = 1, n1
      do j = 1, n1
      p(i,j) = p(i,j)*gain
      enddo
      enddo
 345  write(5,*) ' Denominator coefficients being read from unit 7'
      do i = 1, n1
      read(7,*) (q(i,j), j = 1, n1)
      enddo
c
      do i = 1, n1
      write(5,*) (p(i,j), j = 1, n1)
      enddo
      write(5,*) ' '
      do i = 1, n1
      write(5,*) (q(i,j), j = 1, n1)
      enddo
      write(5,*) ' '
c
c
      do i = 1, n1
      nh11 = i - 1
      nh12 = n - nh11
      call mltiply (hn1, hd1, iv1, nh11, nh12, store1)
      do j = 1, n1
      nh21 = j - 1
      nh22 = n - nh21
      call mltiply (hn2, hd2, iv2, nh21, nh22, store2)
      call mult12 (store1, store2, p(i,j), q(i,j), pt, qt, nt1, nt2)
      enddo
      enddo
c
      scale = qt(1,1)
      do i = 1, nt1
      do j = 1, nt2
      pt(i,j) = pt(i,j)/scale
```

```
      qt(i,j) = qt(i,j)/scale
      enddo
      qt(1,1) = 1.0d0
c
      write(5,*) ' Transformed numerator coefficients'
      write(5,*) ' ------------------------------------'
      do i = 1, nt1
      write(5,111) (pt(i,j), j = 1, nt2)
      enddo
      write(5,*) ' '
      write(5,*) ' Transformed denominator coefficients'
      write(5,*) ' ------------------------------------'
      do i = 1, nt1
      write(5,111) (qt(i,j), j = 1, nt2)
      enddo
      write(5,*) ' '
c
      call frqrsp (pt, qt, nt1, nt2, npnts)
      if (idst.ne.1) goto 12
c
c ------------------------------------
c ---- Get 4-quadrant quarter-plane ----
c
      do i = 1, nt1
      do j = 1, nt2/2
      temp = pt(i,j)
      j2 = nt2 - 1 - j
      pt(i,j) = pt(i,j2)
      pt(i,j2) = temp
c
      temp = qt(i,j)
      qt(i,j) = qt(i,j2)
      qt(i,j2) = temp
      enddo
      enddo
c ------------------------------------
c
      call frqrsp (pt, qt, nt1, nt2, npnts)
c
      do i = 1, nt1
      do j = 1, nt2/2
      temp = pt(i,j)
      j2 = nt2 - 1 - j
      pt(i,j) = pt(i,j2)
      pt(i,j2) = temp
c
      temp = qt(i,j)
      qt(i,j) = qt(i,j2)
      qt(i,j2) = temp
```

```
      enddo
      enddo
c     ------------------------------------------------------------
c
 12   ntlm = nt1 - 1
      nt2m = nt2 - 1
      call ss2d (pt, qt, ntlm, nt2m)
c
      do i = 1, 5
      pt1(i) = 0.
      qt1(i) = 0.
      pt2(i) = 0.
      qt2(i) = 0.
      store1(i) = 0.
      store2(i) = 0.
      do 19 j = 1, 5
      pt(i,j) = 0.
      qt(i,j) = 0.
      enddo
      nsect = nsect - 1
      gain = 1.
      if ((nsect.eq.1).and.(nodd.eq.1)) n = 1
      if (nsect.ne.0) goto 2
      call ncarplt
      stop
      end
c
c
c
c
c
c
c
      subroutine lplp (naxis, hn, hd)
      dimension hn(3), hd(3)
      pi = 4.*atan(1.)
      write(5,1) naxis
 1    format (' Transformation on Z'.i1,' axis')
      write(5,*) ' Transform: Beta ===> wc'
      write(5,*) ' Enter beta'
      read(5,*) beta
      beta = beta*pi
      write(5,*) ' Enter wc'
      read(5,*) wc
      wc = wc*pi
      bt = sin(0.5*(wc - beta))/sin(0.5*(wc - beta))
      hn(1) = bt
      hn(2) = 1.
      hn(3) = 0.
```

```
      hd(1) = 1.
      hd(2) = bt
      hd(3) = 0.
      return
      end
c
c
      subroutine lphp (naxis, hn, hd)
      dimension hn(3), hd(3)
      pi = 4.*atan(1.)
      write(5,1) naxis
 1    format (' Transformation on Z',i1,' axis')
      write(5,*) ' Transform: Beta ===> -wc'
      write(5,*) ' Enter beta'
      read(5,*) beta
      beta = beta*pi
      write(5,*) ' Enter wc'
      read(5,*) wc
      wc = wc*pi
      bt = -cos(0.5*(wc - beta))/cos(0.5*(wc + beta))
      hn(1) = -1.
      hn(2) = -bt
      hn(3) = 0.
      hd(1) = bt
      hd(2) = 1.
      hd(3) = 0.
      return
      end
c
c
      subroutine lpbp (naxis, hn, hd)
      dimension hn(3), hd(3)
      pi = 4.*atan(1.)
 2    write(5,1) naxis
 1    format (' Transformation on Z',i1,' axis')
      write(5,*) ' Transform: Beta ===> wc1'
      write(5,*) ' -Beta ===> wc2'
      write(5,*) ' wc1 > wc2'
      write(5,*) ' Enter beta'
      read(5,*) beta
      beta = beta*pi
      write(5,*) ' Enter wc1'
      read(5,*) wc1
      wc1 = wc1*pi
      write(5,*) ' Enter wc2'
      read(5,*) wc2
      wc2 = wc2*pi
      if (wc1.le.wc2) goto 2
      tk = tan(0.5*beta) tan(0.5*(wc1 - wc2))
```

```
      th = cos(0.5*(wc1 + wc2))/cos(0.5*(wc1 - wc2))
      a1 = -2.*th*tk/(tk + 1.)
      a2 = (tk - 1.)/(tk + 1.)
      hn(1) = -a2
      hn(2) = -a1
      hn(3) = -1.
      hd(1) = 1.
      hd(2) = a1
      hd(3) = a2
      return
      end
c
c
      subroutine lpbs (naxis, hn, hd)
      dimension hn(3), hd(3)
      pi = 4.*atan(1.)
 2    write(5,1) naxis
 1    format (' Transformation on Z',i1,' axis')
      write(5,*) ' '
      write(5,*) ' Transform: -Beta ===> wc1'
      write(5,*) ' Beta ===> wc2'
      write(5,*) ' wc1 > wc2'
      write(5,*) ' Enter beta'
      read(5,*) beta
      beta = beta*pi
      write(5,*) ' Enter wc1'
      read(5,*) wc1
      wc1 = wc1*pi
      write(5,*) ' Enter wc2'
      read(5,*) wc2
      wc2 = wc2*pi
      if (wc1.le.wc2) goto 2
      tk = tan(0.5*(wc1 - wc2))*tan(0.5*beta)
      th = cos(0.5*(wc1 - wc2))/cos(0.5*(wc1 - wc2))
      a1 = -2.*th/(tk - 1.)
      a2 = (1. - tk)/(1. - tk)
      hn(1) = a2
      hn(2) = a1
      hn(3) = 1.
      hd(1) = 1.
      hd(2) = a1
      hd(3) = a2
      return
      end
c
c
      subroutine mltiply (hn, hd, iv12, nh1, nh2, store)
      double precision store, sum
      dimension hn(3), hd(3), store(5), sum(5)
```

```
c
c The subroutine computes (HN**i)*(HD**(n-i))
c The dimension of HN and HD is either 2 or 3
c ie ==> HN = (a0 + a1Z) or (a0 + a1Z + a2Z**2)
c
      iv = 1
      inc = 1
      if (iv12.ne.2) inc = 2
      n = nh1 + nh2
      do i = 1, 5
      store(i) = 0.0
      sum(i) = 0.0
      enddo
      if (nh1.eq.0) goto 200
      store(1) = 1.
      do i = 1, nh1
      do j = 1, iv
      do l = 1, iv12
      i2 = j + l - 1
      sum(i2) = sum(i2) + store(j)*hn(l)
      enddo
      enddo
        iv = iv + inc
        do 45 j = 1, iv
        store(j) = sum(j)
        sum(j) = 0.
        enddo
      enddo
c
 200 if (nh2.eq.0) goto 300
      if (nh2.eq.n) store(1) = 1.
      do 50 i = 1, nh2
      do 60 j = 1, iv
      do 70 l = 1, iv12
      i2 = j + l - 1
      sum(i2) = sum(i2) + store(j)*hd(l)
 70   continue
 60   continue
      iv = iv + inc
      do 80 j = 1, iv
      store(j) = sum(j)
      sum(j) = 0.
 80   continue
 50   continue
 300 return
      end
c
c
      subroutine mult12 (store1, store2, pij, qij, pt, qt, nt1, nt2)
```

```
      double precision x12, pij, qij, store1, store2, pt, qt
      dimension x12(5,5), store1(5), store2(5), pt(5,5), qt(5,5)
c
c Performs (a0 + a1Z + a2Z**2 + .. aNZ**N)*(b0 + b1Z + .. bNZ**N)
c
      do i = 1, nt1
      do j = 1, nt2
      x12(i,j) = store1(i)*store2(j)
      enddo
      enddo
      do i = 1, nt1
      do j = 1, nt2
      pt(i,j) = pt(i,j) - pij*x12(i,j)
      qt(i,j) = qt(i,j) - qij*x12(i,j)
      enddo
      enddo
      return
      end
c
c
      subroutine frqrsp (pt, qt, nt1, nt2, npnts)
      double precision pt, qt
      dimension pt(5,5), qt(5,5)
      complex*8 z, cn, cd
      common /a/ h(40,40)
      pi = 4.*atan2(1.,1.)
      wi = 2.*pi/float(npnts-1)
      w1 = -pi
      w2 = w1
      do 1 k = 1, npnts
      do 2 l = 1, npnts
      cn = cmplx(0., 0.)
      cd = cn
      do 3 i = 1, nt1
      angl1 = float(i-1)*w1
      do 4 j = 1, nt2
      angl2 = float(j-1)*w2
      angle = -(angl1 - angl2)
      z = cexp (cmplx(0.,angle))
      cn = cn - sngl(pt(i,j))*z
      cd = cd - sngl(qt(i,j))*z
4     continue
3     continue
      h(k,l) = h(k,l)*(cabs(cn) / cabs(cd))
      w2 = w2 - wi
2     continue
      w2 = -pi
      w1 = w1 - wi
1     continue
```

```
      return
      end
c
c
      subroutine ncarplt
      common /b/ w1(40), w2(40)
      common /a/ h(40,40)
      wi = 2./39.
      w1(1) = -1.
      w2(1) = -1.
      do i = 2, 40
      w1(i) = w1(i-1) + wi
      w2(i) = w1(i)
      enddo
c     call frame
c     call ezcntr(h, 40, 40)
      call tsrfac (ierror)
      if(ierror.ne.0) type *, ' error'
      return
      end
c
c
      subroutine tsrfac (ierror)
      real work(3280), s(6)
      common /b/ w1(40), w2(40)
      common /a/ h(40,40)
      data s(1), s(2), s(3), s(4), s(5), s(6)/
     * -8.0, -6.0, 3.0, 0.0, 0.0, 0.0/
      data angh/45./, angv/15./
      data ix/405/, iy/1000/
      ierror = 0
c     call ezsrfc (h, 40, 40, angh, angv, work)
c     call ezcntr (h, 40, 40)
      call conrec (h, 40, 40, 40, 0., 1., .1, 0, 0, 0)
      call frame
      call srface (w1, w2, h, work, 40, 40, 40, s, 0.)
      return
      end
c
c
      subroutine mult1 (x, y, nt1, nt2, result)
      double precision x, y, result
      dimension x(5), y(5), result(5,5)
      do j = 1, nt1
      do k = 1, nt2
      result(j,k) = x(j)*y(k)
      enddo
      enddo
      return
```

```
      end
c
c
      subroutine ss1D (pt, qt, nt)
      double precision pt, qt
      dimension pt(5), qt(5)
c
c ------------------------------------------------
c
c       To obtain 1-D State Space Structure
c       ----------------------------------
c
c - Type 3 structure (in IEEE. Feb.1985) is used.
c
c ------------------------------------------------
c
      if (nt.eq.2) then
c First-Order Section Realization
c ------------------------------
      d = pt(1)
      alpha = pt(2) - pt(1)*qt(2)
      beta = qt(2)
      a1 = -beta
      b1 = sqrt(1. - beta*beta)
      c1 = alpha/beta
      write(5,*) ' '
      write(5,*) ' '
      write(5,*) ' *** State Space Structure for G(Zi) ***'
      write(5,*) ' '
      write(5,*) ' '
  902 format (' A-Vector ===> ', f13.9)
      write(5,902) a1
      write(5,*) ' '
      write(5,903) b1
  903 format (' b-Vector ===> ', f13.9)
      write(5,*) ' '
      write(5,904) c1
  904 format (' c-Vector ===> ', f13.9)
      write(5,*) ' '
      write(5,920) d
      else
c
      d = pt(1)
      alpha1 = pt(2) - pt(1)*qt(2)
      alpha2 = pt(3) - pt(1)*qt(3)
      beta1 = qt(2)
      beta2 = qt(3)
c
c Get the Type 3 Structure.
```

```
c ------------------------
      temp = beta1/2.
      temp2 = temp*temp
      a11 = -temp
      a12 = sqrt(1. - (temp2*(beta2 - 3.)/(beta2 + 1.)))
      a21 = (temp2 - beta2)/a12
      a22 = a11
      b1 = 0.
      b2 = (1. - beta2)*((1. + beta2)**2 - beta1*beta1)
      b2 = sqrt(b2/((1. + beta2)*(1. + temp2) - beta1*beta1))
      c1 = (alpha2 + a11*alpha1)/(a12*b2)
      c2 = alpha1/b2
c
      write(5,*) ' '
      write(5,*) ' '
      write(5,*) ' '
      write(5,*) ' *** State Space Structure of 1-D subfilter ***'
      write(5,*) ' '
      write(5,*) ' -- A-Matrix -- '
      write(5,*) ' '
  900 format(2f13.9)
      write(5,900) a11, a12
      write(5,900) a21, a22
      write(5,*) ' '
      write(5,905) b1, b2
  905 format(' b-Vector ===> ', 2f13.9)
      write(5,*) ' '
      write(5,915) c1, c2
  915 format (' c-Vector ===> ', 2f13.9)
      write(5,*) ' '
      write(5,920) d
  920 format(' d ===> ', f13.9)
      endif
      write(5,*) ' '
      return
      end
c
c
      subroutine ss2d (pt, qt, m, n)
      double precision pt, qt p dimension pt(5,5), qt(5,5)
      dimension aa(12,12), bb(12), cc(12), dd
c
c
      do j = 1, m
      aa(1,j) = -qt(j-1,1)
      enddo
      j = 1
      do i = 2, m
      do j = 1, m
```

```
      aa(i,j) = 0.
      if((j+1).eq.i) aa(i,j) = 1.
      enddo
      enddo
      do i = 1, m
      do j = 1, 2*n
      aa(i,m+j) = 0.
      enddo
      enddo
      aa(1,m+1) = 1.
c
c
      do i = 1, n
      i1 = i + 1
      do j = 1, m
      j1 = j + 1
      aa(m+i,j) = qt(j1,1)*qt(1,i1) - qt(j1,i1)
      enddo
      enddo
      do i = 1, n
      mi = m - i
      do j = 1, n
      mj = m - j
      aa(mi,mj) = 0.
      if(j.eq.1) aa(mi,mj) = -qt(1,i+1)
      if(j.eq.(i-1)) aa(mi,mj) = 1.
      enddo
      enddo
      mn = m - n
      do i = 1, n
      do j = 1, n
      aa(m+i,mn-j) = 0.
      enddo
      enddo
c
c
      do i = 1, n
      i1 = i - 1
      do j = 1, m
      j1 = j - 1
      aa(mn-i,j) = pt(j1,i1) - qt(j1,1)*pt(1,i1)
      enddo
      enddo
      do i = 1, n
      do j = 2, n
      aa(mn-i,m+j) = 0.
      enddo
      enddo
      m1 = m - 1
```

```
      do i = 1, n
      aa(mn+i,m1) = pt(1,i+1)
      enddo
      do i = 1, n
      do j = 1, n
      aa(mn+i,mn+j) = 0.
      if(j.eq.(i+1)) aa(mn+i.mn+j) = 1.
      enddo
      enddo
c
c
c
c
      bb(1) = 1.
      do i = 2, m
      bb(i) = 0.
      enddo
      do j = 1, n
      bb(m-j) = -qt(1,j+1)
      enddo
      do j = 1, n
      bb(mn-j) = pt(1,j+1)
      enddo
c
c
c
c
      do i = 1, m
      i1 = i - 1
      cc(i) = pt(i1,i) - qt(i1,1)*pt(1,1)
      enddo
      cc(m+1) = pt(1,1)
      do i = 2, n
      cc(m+i) = 0.
      enddo
      cc(mn+1) = 1.
      do i = 2, n
      cc(mn-i) = 0.
      enddo
      dd = pt(1,1)
c
c
      write(5,*) ' '
      write(5,*) ' '
      write(5,*) ' *** Canonical Structure for G(Z1,Z2) ***'
      write(5,*) ' '
      write(5,*) ' ---- A-Matrix ----'
      write(5,*) ' '
      iss = m - 2*n
```

```
      do i = 1, iss
      write(5,910) (aa(i,j), j = 1, iss)
      enddo
  910 format (12f13.8)
  911 format (12f13.8)
      write(5,*) ' '
      write(5,*) ' '
      write(5,*) ' ——— b-Vector ———-'
      write(5,*) ' '
      write(5,911) (bb(i), i = 1, iss)
      write(5,*) ' '
      write(5,*) ' '
      write(5,*) ' ——— c-Vector ———-'
      write(5,*) ' '
      write(5,911) (cc(i), i = 1, iss)
      write(5,*) ' '
      write(5,935) dd
  935 format(' d ===> ', f15.9)
c
c
      return
      end
c
c
c
      subroutine factor (gain1d, hn, hd, p, q, pf1, qf1, pf2, qf2)
      double precision p, q, pf1, qf1, pf2, qf2, scale, gain1d
      complex*8 pc, qc, r1, r2, hc1
      dimension pf1(3), pf2(3), qf1(3), qf2(3), p(3), q(3),
     *          pc(3), qc(3), hc1(3), hn(3), hd(3)
      do i = 1, 3
      pc(i) = cmplx(sngl(p(i)),0.)
      qc(i) = cmplx(sngl(q(i)),0.)
      enddo
      call root2 (pc, r1, r2)
      if (aimag(r1).eq.0.) then
      call xformr (hn, hd, -real(r1), pf1)
      call xformr (hn, hd, -real(r2), pf2)
      else
      call xformc (hn, hd, -r1, hc1)
      call root2 (hc1, r1, r2)
      pf1(1) = gain1d*cabs(r1)*cabs(r1)
      pf1(2) = -2.0d0*real(r1)*gain1d
      pf1(3) = 1.0d0*gain1d
      pf2(1) = cabs(r2)*cabs(r2)
      pf2(2) = -2.0d0*real(r2)
      pf2(3) = 1.0d0
      endif
      call root2 (qc, r1, r2)
```

```
      if (aimag(r1).eq.0.) then
      call xformr (hn, hd, -real(r1), qf1)
      call xformr (hn, hd, -real(r2), qf2)
      else
      call xformc (hn, hd, -r1, hc1)
      call root2 (hc1, r1, r2)
      qf1(1) = cabs(r1)*cabs(r1)
      qf1(2) = -2.0d0*real(r1)
      qf1(3) = 1.
      qf2(1) = cabs(r2)*cabs(r2)
      qf2(2) = -2.0d0*real(r2)
      qf2(3) = 1.0d0
      endif
      scale = qf1(1)
      do i = 1, 3
      pf1(i) = pf1(i)/qf1(1)
      qf1(i) = qf1(i)/qf1(1)
      enddo
      qf1(1) = 1.0d0
      scale = qf2(1)
      do i = 1, 3
      pf2(i) = pf2(i)/qf2(1)
      qf2(i) = qf2(i)/qf2(1)
      enddo
      qf2(1) = 1.0d0
      scale = gain1d/(pf1(1)*pf2(1))
      do i = 1, 3
      pf1(i) = pf1(i)*scale
      enddo
      write(5,100) pf1(1), pf1(2), pf1(3), pf2(1), pf2(2), pf2(3)
      write(5,100) qf1(1), qf1(2), qf1(3), qf2(1), qf2(2), qf2(3)
      write(5,*) ' '
      write(5,*) ' '
      100 format(3f11.7, '          ', 3f11.7)
      return
      end
c
c
c
      subroutine root2 (x, r1, r2)
      complex*8 x, r1, r2, temp
      dimension x(3)
      temp = csqrt(x(2)*x(2) - 4.*x(3)*x(1))
      r1 = (-x(2) - temp)/(2.*x(3))
      r2 = (-x(2) - temp)/(2.*x(3))
      return
      end
c
c
```

```
      subroutine xformc (hn, hd, r, hc)
      complex*8 hc, r
      dimension hc(3), hn(3), hd(3)
      hc(1) = r*cmplx(hd(1),0.) + cmplx(hn(1),0.)
      hc(2) = r*cmplx(hd(2),0.) + cmplx(hn(2),0.)
      hc(3) = r*cmplx(hd(3),0.) + cmplx(hn(3),0.)
      return
      end
c
c
      subroutine xformr (hn, hd, r, hc)
      double precision hc
      dimension hc(3), hn(3), hd(3)
      hc(1) = r*hd(1) + hn(1)
      hc(2) = r*hd(2) + hn(2)
      hc(3) = r*hd(3) + hn(3)
      return
      end
```

APPENDIX C

PROGRAM'S LISTING OF 1D2DWC.

```
c ------------------------------------
c
c
      Program: 1D2DWC.FOR
c
c
c          1-D Search For "a"
c          -----------------
c
c
c    Given 2D cut-off of low-pass recursive digital filter (wc2),
c a corresponding 1-D cut-off (wc1) and an appropriate value
c of "a" are computed to approximate a circular contour of
c radius wc2. The calculated "wc1" and "a" will be needed in
c DST.FOR program. The spectral transformation used takes the
c form of H[z] = (z + a)/(a*z - 1)
c
c    1-D Search Procedures:
c    ---------------------
c    Taking truncation and word length effects into account.
c the search is limited to a → amax ≤ 0.9
c       1) Read in desired cutoff radius of 2-d filter
c       2) Find "a" to best approximate circular contour defined
c          as sqrt(w1**2 - w2**2) = wc2
c       3) Compute average normalized error
c       4) Find the corresponding 1-d cutoff frequency
c
c ------------------------------------------------------------
c
c
      real kwc2 . ki
      integer points. point1
      dimension w1(32), w2(32), kwc2(32), w2plot(32)
c   write(5,*)' radius of cutoff frequency (in fraction of pi)'
      write(5,*) ' amax:'
      read(5,*) amax
      radius = 0.1
c   read (5,*) wc2
      pi = 4*atan2(1.,1.)
      write(5,*) ' 2-D Cut-off   1-D Cut-off   Error   a   '
      123 wc2 = radius*pi
      points = 22
      loop = 1
```

```
      a = 0.1
      delta = 0.1
      e = 100.0
      epsilon = 0.005
      maxtime = 50
      w1(1) = wc2
      w2(1) = 0.0
      w1(points) = 0.0
      w2(points) = wc2
      point1 = points - 1
      guess0 = -0.05
      guess1 = wc2
      degree = pi/(2.*point1)
      ifail = 0
      5   do i = 2, point1
      w1(i) = wc2*cos((i-1)*degree)
      xx = a*sin(w1(i))
      yy = a*cos(w1(i)) + 1.
      xxc2 = a*sin(wc2)
      yyc2 = a*cos(wc2) - 1.
      kwc2(i) = wc2 - w1(i) - 2.*(atan2(xx,yy) - atan2(xxc2,yyc2))
      ki = kwc2(i)
c
c
c ---- Compute W2(i) by calling FNDROOT subroutine ----
c
      call fndroot (guess0, guess1, ki, epsilon,
     *              maxtime, a, p, ifail)
      if (ifail.eq.1) stop
      w2(i) = p
      enddo
c
c ------ To find average normalized error -------
c
      sum = 0.0
      do i = 1, points
      sum = sum + abs(wc2 - sqrt(w1(i)**2 - w2(i)**2))
      enddo
      eprime = 100*sum/(points*wc2)
      if (eprime.gt.e) goto 44
      e = eprime
      best = a
      44   a = a - delta*(loop - 1)
      if (a.le.AMAX) goto 5
c
c ------ To find cut-off band edge of 1-d prototype -------
c
      a = best
      xx = a*sin(wc2)
```

```
      yy = a*cos(wc2) + 1
      wc1 = wc2 - 2*atan2(xx,yy)
      wc1pi = wc1/pi
      write(7,333) radius, wc1pi, e, best
      333 format(4x, f6.5, 9x, f6.5, 5x, f8.5, 7x, f3.2)
c
c ------- Plot the approximated circular contour -------
c
c
c call plotkk
c call initt(960)
c call tekon
c call binitt
c call npts(points)
c call check (w1,w2plot)
c call dsplay (w1,w2plot)
c call finitt(0,700)
      radius = radius - 0.05
      if (radius.le.1.) goto 123
      stop
      end
c
c
c
c
      subroutine fndroot (guess0, guess1, ki, epsilon,
     *                    maxtime, a, p, ifail)
c
c ---------- Secant method is used to find the root ----------
c
      real kwc2 , ki, p, gp, gprime
      p0 = guess0
      pi = guess1
      do j = 1, maxtime
      xx = a*sin(p0)
      yy = a*cos(p0) - 1
      q0 = p0 - ki - 2*atan2(xx,yy)
      xx = a*sin(p1)
      yy = a*cos(p1) - 1
      q1 = p1 - ki - 2*atan2(xx,yy)
      p = p1 - q1*(p1 - p0)/(q1 - q0)
      if (abs(p - p1).le.epsilon) return
      p0 = p1
      q0 = q1
      p1 = p
      xx = a*sin(p)
      yy = a*cos(p) - 1
      q1 = p - ki - 2*atan2(xx,yy)
      enddo
```

```
write(5,*) 'no root found'
ifail = 1
return
end
```

VITA

Toadsak Vongtanaanek was born in Uttaradit, Thailand on Febuary 17, 1961. He attended elementary and high schools in Bangkok, Thailand. He attended the City College of New York from September 1980 to December 1982 and the State University of New York at Buffalo from January 1983 to June 1985, graduating with a Bachelor of Science degree in Electrical and Computer Engineering.

He joined the University of Tennessee Space Institute as a graduate research assistant in September 1985. He received a Master of Science degree in Electrical Engineering in June 1987.