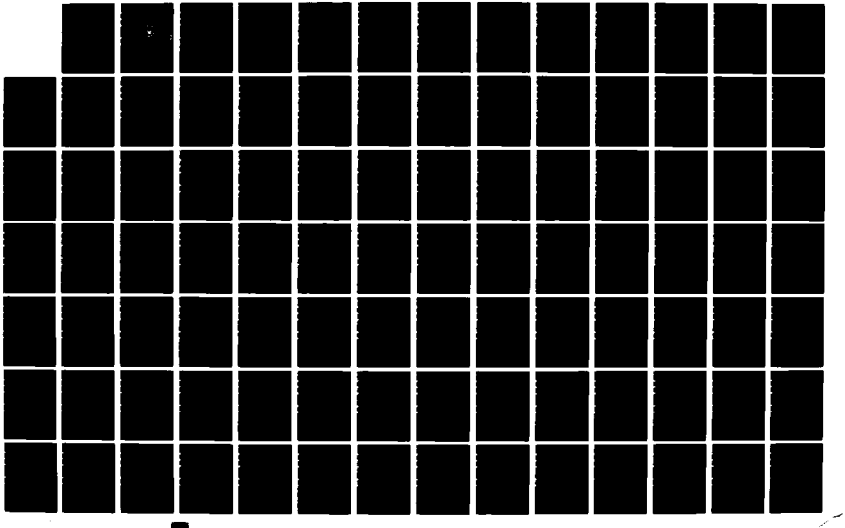
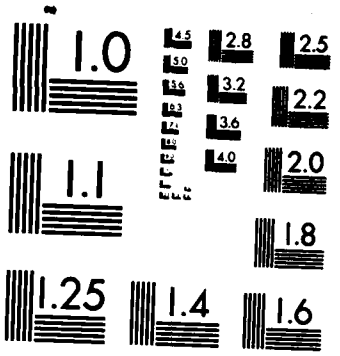
AD-A164 128 2-D SIGNAL GENERATION USING STATE-SPACE FORMULATION(U) 1/2
NAVAL POSTGRADUATE SCHOOL MONTEREY CA E THEOFILOU
DEC 85
UNCLASSIFIED F/G 9/2 NL

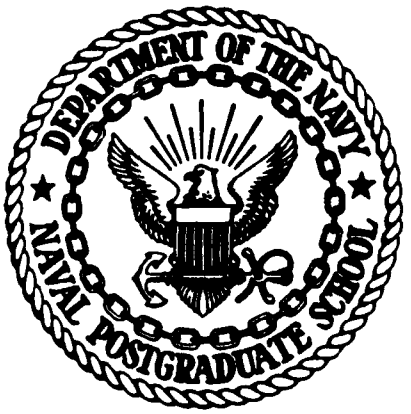




(2)

NAVAL POSTGRADUATE SCHOOL

Monterey, California



THESIS

2-D SIGNAL GENERATION USING
STATE-SPACE FORMULATION

by

Evangelos Theofilou

December 1985

Thesis Advisor: Sydney R. Parker

Approved for public release; distribution is unlimited

DISCLAIMER NOTICE

THIS DOCUMENT IS BEST QUALITY PRACTICABLE. THE COPY FURNISHED TO DTIC CONTAINED A SIGNIFICANT NUMBER OF PAGES WHICH DO NOT REPRODUCE LEGIBLY.

UNCLASSIFIED
SECURITY CLASSIFICATION OF THIS PAGE

ADA 164 128

REPORT DOCUMENTATION PAGE

1a. REPORT SECURITY CLASSIFICATION	1b. RESTRICTIVE MARKINGS
2a. SECURITY CLASSIFICATION AUTHORITY	3. DISTRIBUTION / AVAILABILITY OF REPORT Approved for public release; distribution is unlimited
2b. DECLASSIFICATION / DOWNGRADING SCHEDULE	
4. PERFORMING ORGANIZATION REPORT NUMBER(S)	5. MONITORING ORGANIZATION REPORT NUMBER(S)
6a. NAME OF PERFORMING ORGANIZATION: Naval Postgraduate School 6b. OFFICE SYMBOL (If applicable): Code 62	7a. NAME OF MONITORING ORGANIZATION: Naval Postgraduate School
6c. ADDRESS (City, State, and ZIP Code): Monterey, California 93943-5100	7b. ADDRESS (City, State, and ZIP Code): Monterey, California 93943-5100
8a. NAME OF FUNDING / SPONSORING ORGANIZATION 8b. OFFICE SYMBOL (If applicable)	9. PROCUREMENT INSTRUMENT IDENTIFICATION NUMBER
8c. ADDRESS (City, State, and ZIP Code)	10. SOURCE OF FUNDING NUMBERS PROGRAM ELEMENT NO. / PROJECT NO. / TASK NO. / WORK UNIT ACCESSION NO.

11. TITLE (Include Security Classification)
2-D SIGNAL GENERATION USING STATE-SPACE FORMULATION

12. PERSONAL AUTHOR(S)
Theofilou, Evangelos

13a. TYPE OF REPORT	13b. TIME COVERED	14. DATE OF REPORT (Year, Month, Day)	15. PAGE COUNT
Master's Thesis	FROM _____ TO _____	1985, December	171

16. SUPPLEMENTARY NOTATION

17 COSATI CODES			18. SUBJECT TERMS (Continue on reverse if necessary and identify by block number)
FIELD	GROUP	SUB-GROUP	2-D Signal Generation State-Space Formulation

19. ABSTRACT (Continue on reverse if necessary and identify by block number)

This thesis has dealt with various approaches to modelling 2-D data fields using state-space formulations. Computer simulation of these models has been carried out to generate simulated 2-D data which could then be used for various other signal processing operations. An interesting development that has resulted from this study is that of

20. DISTRIBUTION / AVAILABILITY OF ABSTRACT ☒ UNCLASSIFIED/UNLIMITED ☐ SAME AS RPT ☐ DTIC USERS	21. ABSTRACT SECURITY CLASSIFICATION Unclassified
22a. NAME OF RESPONSIBLE INDIVIDUAL Prof. Sydney R. Parker	22b. TELEPHONE (Include Area Code) (408) 646-2788 22c. OFFICE SYMBOL Code 62Px

DD FORM 1473, 84 MAR 83 APR edition may be used until exhausted
All other editions are obsolete.

SECURITY CLASSIFICATION OF THIS PAGE

#19 - ABSTRACT - (CONTINUED)

adaptation of the 1-D SSPACK software package for simulating 2-D linear systems as well as using one of the above state-variable models.

Approved for public release; distribution is unlimited.

2-D Signal Generation Using State-Space Formulation

by

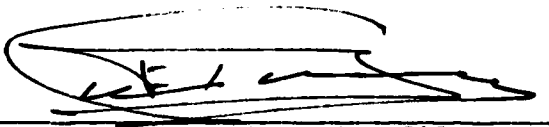
Evangelos Theofilou
Lieutenant, Greek Navy
B.S., Greek Naval Academy, 1975

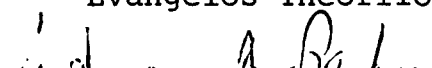
Submitted in partial fulfillment of the requirements for the degree of

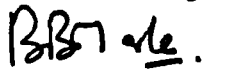
MASTER OF SCIENCE IN ELECTRICAL ENGINEERING

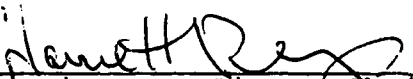
from the

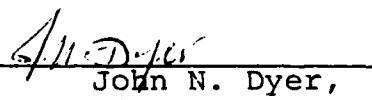
NAVAL POSTGRADUATE SCHOOL
December 1985

Author: 
Evangelos Theofilou

Approved by: 
Sydney R. Parker, Thesis Advisor


Bharat Madan, Second Reader


Harriett B. Rigas, Chairman,
Department of Electrical and Computer Engineering


John N. Dyer,
Dean of Science and Engineering

ABSTRACT

This thesis has dealt with various approaches to modelling 2-D data fields using state-space formulations. Computer simulation of these models has been carried out to generate simulated 2-D data which could then be used for various other signal processing operations. An interesting development that has resulted from this study is that of adaptation of the 1-D SSPACK software package for stimulating 2-D linear systems as well as using one of the above state-variable models.

TABLE OF CONTENTS

I. INTRODUCTION

A. THE MAIN IDEA

Image processing by nonoptical means has received extensive attention in the last few years. Several books and many papers have been published that have established nonoptical image processing as a viable area of research. A large portion of this research emphasizes the linear processing of images for two main reasons: 1) Many image processing tasks are linear in nature. These tasks include image enhancement, image restoration, picture coding, linear pattern recognition, and TV bandwidth reduction. 2) There are many known linear techniques that may be brought to bear in the treatment of linear image processing. These techniques include transform theory, matrix theory, filtering, signal modeling, etc. Several common operations involved in image processing include transfer function concepts, partial difference (recursive) equations, and convolution summations. For example, Vander Lugt [Refs. 1,2] has presented an extensive development of linear optics based on transfer functions. The transfer functions relate the two-dimensional Fourier transform of an output image to that of the input image. Complex optical systems are easily described by combinations of transfer functions that correspond to individual components of the optical system.

Partial difference equations are used by Habibi [Ref. 3] to describe a model for estimating images corrupted by noise. The model corresponds to a two-dimensional extension of Kalman filters. Convolution summations are discussed by Fryer and Richmond [Ref. 4] in work that involves simplifying a two-dimensional filter to a single dimensional filter.

The time-discrete state-space model offers great utility in the formulation and analysis of linear systems. Linear systems that are described by transfer functions, difference equations, or convolution summations are formulated into a state-space representation. Once formulated, many known techniques may be applied to systematically analyze the model. Consequently, the state space model is a general and powerful tool that is used to unify the research and the study of time-discrete linear systems.

This thesis develops the discrete model of Roesser [Ref. 5] for linear image processing which closely parallels the well-known state space model for time-discrete systems. Because it is parallel, many of the concepts that are known for the temporal model may be carried over to the spatial model. This is done by generalizing from a single coordinate in time to two coordinates in space. The spatial model will hopefully have some of the same utility for the study of two-dimensional linear systems as the temporal model for one-dimensional linear systems [Ref. 3]. However, not all of the properties of one-dimensional systems carry over into the multi-dimensional case.

One of the fundamental problems involved with recursive 2-Dimensional systems is that the order of the system (recursive memory) is not the same as the number of initial conditions (boundary conditions). In one-dimensional systems these are the same. Temporal systems are inherently nonanticipatory and are often treated as such for the sake of physical realizability in real time; whereas spatial systems do not have causality which is an inherent limitation. That is, an image processor may have right to left dependency as well as left to right dependency. Finally it is noted that stability criteria in one-dimensional recursive systems become much more difficult when carried over to the multidimensional case.

Causality is built into the temporal state-space model if an initial state is assumed to be fully specified. In order to establish a close parallel for the spatial model, the same built-in causality will be intentionally assumed, despite the fact that causality is not necessary for physical realizability in real space. Such an image processor is said to be unilateral. If the constraint of causality is removed, then the image processor is said to be bilateral [Ref. 5]. Concepts that are developed in this thesis for the latter case are:

1) Formulation of the state space model of Roesser. [Ref. 5]
2) The definition of state transition matrix.
3) A resulting computer program based on the above model.
4) An investigation of the class of 2-Dimensional transfer functions defined by this model.
5) Derivation of a general response formula.

6) Extension of Roesser's model of state variable equations to encompass a larger class of transfer functions.

7) Adaptation of the 1-D "SSPACK" program to produce 2-D data.

B. STATE SPACE REPRESENTATION

Toward the end of the 1950s, the concept of representing a discrete system by a set of first-order difference equations became a standard tool of the research engineer. These techniques have since become generally known as state-space representations. Such representations have become increasingly important during the intervening years because they often allow one to carry out a meaningful system design entirely in the discrete-time domain (in comparison to popular Z-transform methods). That this is important follows basically from these factors:

1. The system may be nonlinear so that transformation methods are not directly applicable.

2. Time-domain concepts often give one a better insight into the analysis and synthesis of the system (frequently with the aid of a digital computer).

3. Cases in which the initial conditions are non-zero may be handled straightforwardly.

A state space representation of a system differs from the conventional representation. In a conventional representation only the relationships between the input and output signals need be known. On the other hand, the state-space representation gives a total description of both the internal as well as the external signals of a system.

C. STATE-VARIABLE REALIZATIONS--THE CONCEPT OF STATE --

In 1-D linear systems theory and control theory, the concept of a filter state has played an important role. Basically the filter state at any point in time contains all the information necessary to compute the remainder of the filter output signal, given the input signal. One dimensional single-input, single-output filter realizations based on a state variable model can be written in the form:

$$x(k+1) = Ax(k) + Bu(k) \qquad \text{(I.1a)}$$

$$y(k) = Cx(k) + Du(k) \qquad \text{(I.1b)}$$

This form relates the input u(k) and the output y(k) through a state vector x(k). The state vector evolves in time according to equation (I.1a). The matrices A, B, and C and 1×1 matrix D govern the exact form of the input-output relationship. (In general these matrices may vary with the index (k) and the input and output signals may be vectors as well.) Quite often the components of the state vector are taken to be the constants of the Z^{-1} delay operators in a flowgraph representation of the 1-D filter.

A classic problem in state-variable theory representation is to find the matrices A, B, C and D which will realize a particular system function H(z) with a minimum number of state variables. A similar approach may be taken to develop a 2-D state-variable model.

A 2-D discrete system may be defined as a mathematical abstraction which utilizes three types of variables to represent or model the dynamics of a discrete-time process. The three variables are called the input, the output, and the state variable. The input variables u(i,j), serve as external forces which influence the dynamics or motion of the system. The output variables y(i,j) are the characteristic variables which are directly observable (measurable) by the external observer. The state variables x(i,j) characterize the internal dynamics of the system and are to be selected according to the following rule.

These variables are formulated in such a manner that, if one knows the values of the present state variables x(i,j) along with the values of the input variables u(i,j) then the output variables y(i,j) and the next state variables x(i,j) are completely determined. Moreover, the number of state variables used in a state-space representation must be minimized. A state-space representation may be visualized in block diagram form, as shown below.

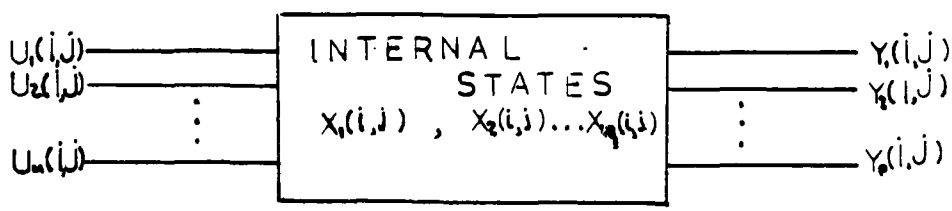


Figure 1.1

In Figure 1.1, m-inputs, p-outputs and n-state variables are represented. However, we will be mainly interested in those systems which have one input (m = 1) and one output (p = 1). It is important to note that the input and output variables appear external to the system, while the state variables are generally internal.

The different input variables will be represented by the input vector u(i,j) where,

$$u(i,j) = \begin{bmatrix} u_1(i,j) \\ u_2(i,j) \\ \vdots \\ u_m(i,j) \end{bmatrix} ,$$

the output vector y(i,j) where,

$$y(i,j) = \begin{bmatrix} y_1(i,j) \\ y_2(i,j) \\ \vdots \\ y_p(i,j) \end{bmatrix} ,$$

and the state vector x(i,j) where,

$$x(i,j) = \begin{bmatrix} x_1(i,j) \\ x_2(i,j) \\ \vdots \\ x_n(i,j) \end{bmatrix} .$$

For a given process the state space representation is not unique. However all such representations have one characteristic in common for a given system, namely the number of elements n is referred to as the order of the system.

II. ROESSER'S STATE-SPACE MODEL

A. THE FRAMEWORK

An image is a generalization of a temporal signal, in that it is defined over two spatial dimensions instead of a single temporal dimension. Consequently, two space coordinates i and j take the place of time, t. Also, two-state sets are introduced to replace the single-state set. The following definitions are made by the model:

- i An integer-valued vertical coordinate;
- j An integer-valued horizontal coordinate;
- {R} A set of n_1 real vectors which convey information horizontally;
- {S} A set of n_2 real vectors which convey information vertically;
- {u} A set of m real vectors that act as inputs;
- {y} A set of p real vectors that act as outputs.

A specific image processor is then defined as 6-tuple

$$<\{R\},\{S\},\{u\},\{y\},f,g> ,$$

where f is the next state function:

$$f: \{\{R\},\{S\},\{u\} \rightarrow \{\{R\},\{S\}\}$$

and y is the output function

$$g: \{\{R\},\{S\},\{u\}\} \rightarrow \{y\} .$$

Now since f and g are to be linear functions, they may be represented by the following matrix equations:

$$\begin{aligned} R(i+1,j) &= A_1R(i,j) + A_2S(i,j) + B_1u(i,j) \\ S(i,j+1) &= A_3R(i,j) + A_4S(i,j) + B_2u(i,j) \qquad \text{(II.1)} \\ y(i,j) &= C_1R(i,j) + C_2S(i,j) + Du(i,j) \quad i,j \geq 0 \end{aligned}$$

A_1, A_2, A_3, A_4, B_1 B_2, C_1, C_2, D are matrices of appropriate dimensions. Boundary conditions R(0,j) and S(i,0) and also the input u(i,j) are externally specified. In the next section a computational rule is obtained that uniquely determines the states R(i,j) and S(i,j) and also the output y(i,j) (for $i,j \geq 0$) from the boundary conditions (such as all zero). The equations produce a set of output vectors from the input vectors.

This formulation is general so that any discrete linear image process may be so represented. Notation is condensed somewhat by introducing the following matrices and vectors:

$$A = \begin{bmatrix} A_1 & A_2 \\ A_3 & A_4 \end{bmatrix} \qquad B = \begin{bmatrix} B_1 \\ B_2 \end{bmatrix} \qquad C = [C_1 \quad C_2]$$

$$T(i,j) = \begin{bmatrix} R(i,j) \\ S(i,j) \end{bmatrix} \qquad T'(i,j) = \begin{bmatrix} R(i+1,j) \\ S(i,j+1) \end{bmatrix}$$

$$T'(i,j) = AT(i,j) + Bu(i,j)$$

$$y(i,j) = CT(i,j) + Du(i,j)$$

B. GENERAL RESPONSE FORMULA

A state-transition matrix A is defined as follows:

$$A = \begin{bmatrix} A_1 & A_2 \\ A_3 & A_4 \end{bmatrix}$$

Then exponentiation $A^{i,j}$ is defined as,

$$A^{i,j} = A^{1,0}A^{i-1,j} + A^{0,1}A^{i,j-1} \qquad (i,j) > (0,0)$$

$$A^{0,0} = I \; ; \; A^{-i,j} = A^{i,-j} = 0 \quad \text{for} \quad j \geq 1, \; i \geq 1$$

Examination of this definition bears out that it is an effective recursive definition of $A^{i,j}$ for integer values of i and j such that either $i > 0$ or $j > 0$ or $(i,j) = (0,0)$. It parallels the definition of the time-discrete state-transition matrix $A^t = A\,A^{t-1}$.

It now remains to be shown that this state transition matrix $A^{i,j}$ may be used in expressions for the response of the model in terms of the inputs and boundary conditions. The term boundary conditions is used here to refer to the states along the edges of the model. Specifically, the set of boundary conditions consist of $R(0,j)$ for $j \geq 0$ and $S(i,0)$ for $i \geq 0$.

C. CHARACTERISTIC FUNCTION OF A MATRIX

If the primary inputs and outputs are dropped in the model equations (II.1), a representation arises for the state behavior of the system having the form

$$\begin{aligned} R(i+1,j) &= A_1R(i,j) + A_2S(i,j) \\ S(i,j+1) &= A_3R(i,j) + A_4S(i,j) \end{aligned} \qquad \text{(II.2)}$$

These equations are useful in the development of a form for a two-dimensional characteristic matrix of A. Operators are

first introduced that advance a particular coordinate of their operand.

Definition: Let E be an operator that has the effect of advancing the vertical coordinate or the first subscipt of the function upon which it is operating. Likewise, let F be an operator that has the effect of advancing the horizontal coordinate or second subscript of the function upon which it is operating. The effect of these operators on the state vectors is:

$$R(i+1,j) = ER(i,j)$$
$$S(i,j+1) = FS(i,j)$$

The state equations can be rewritten using these advance operators.

$$(EI-A_1)R(i,j) - A_2S(i,j) = 0$$

$$-A_3R(i,j) + (FI-A_4)S(i,j) = 0$$

These equations are equivalently represented in the overall matrix form.

$$\begin{bmatrix} (EI-A_1) & -A_2 \\ -A_3 & (FI-A_4) \end{bmatrix} T(i,j) = 0$$

The above equation represents a system of homogeneous equations in the elements of T(i,j). If the system is to have a non-trivial solution for T(i,j) then the transformation represented by the matrix must be singular. The above matrix is said to be the two-dimensonal characteristic matrix of the partitioned matrix A, where

$$A = \begin{bmatrix} A_1 & A_2 \\ A_3 & A_4 \end{bmatrix}$$

The characteristic matrix of A is denoted cm(A) and may be represented as

$$cm(A) = EI^{1,0} + FI^{0,1} - A$$

where,

$$I^{1,0} = \left[\begin{array}{c|c} I & 0 \\ \hline 0 & 0 \end{array}\right] \quad \text{and} \quad I^{0,1} = \left[\begin{array}{c|c} 0 & 0 \\ \hline 0 & I \end{array}\right]$$

Now since cm(A) must be singular, its determinant must be equal to zero. $|cm(A)| = 0$. If E and F are replaced in the above by general indeterminates x and y respectively, the result is an expression called the two-dimensional characteristic equation for A. The determinant of cm(A), and x and y replacing E and F, is called the two-dimensional characteristic function of the matrix and is denoted by

$$|cm(A)| = f(x,y) = 0$$

f(x,y) will be a monic polynomial in x and y with degree n_1 in x, and degree n_2 in y, where n_1 is the dimension of R and n_2 is the dimension of S. f(x,y) has the form

$$f(x,y) = \sum_{(0,0) \leq (i,j) \leq (n_1,n_2)} a_{i,j} x^i y^j$$

where $a_{i,j}$ denotes elements of A and $a_{n_1,n_2} = 1$.

D. CIRCUIT ELEMENTS AND THEIR REALIZATION

Let us consider the single 2-D IIR filter transfer function given by:

$$H(z_1,z_2) = \frac{b_{00}+b_{10}z_1^{-1}+b_{01}z_2^{-1}+b_{11}z_1^{-1}z_2^{-1}+b_{21}z_1^{-2}z_2^{-1}}{1-a_{10}z_1^{-1}-a_{01}z_2^{-1}-a_{11}z_1^{-1}z_2^{-1}-a_{21}z_1^{-2}z_2^{-1}} = \frac{B(z_1,z_2)}{1-A(z_1,z_2)}$$

A simple block diagram for $H(z_1,z_2)$ follows.

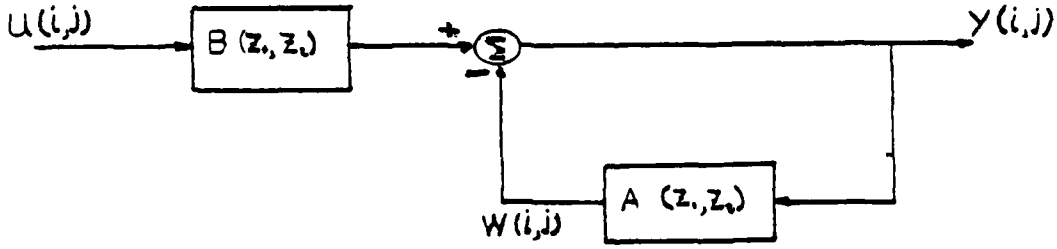


Figure 2.1

The input signal u(i,j) flows through a filter corresponding to the numerator transfer function $B(z_1,z_2)$. The resulting signal is added to the signal $-w(i,j)$ to produce the output signal y(i,j). The denominator transfer function $1-A(z_1,z_2)$ is realized by the feedback loop containing $A(z_1,z_2)$.

Since we are dealing with two dimensions, there are two fundamental shift operators which may occur along a signal flow path, the horizontal shift operator indicated by z_1^{-1} and the vertical shift indicated by z_2^{-1} [we shall omit from consideration the inverse shift operators z_1 and z_2]. In most cases of practical interest they can be eliminated by multiplying both the numerator and denominator polynomials of $H(z_1,z_2)$ by the appropriate powers of z_1^{-1} and z_2^{-1}. Let us look at a signal flowgraph representing the numerator polynomial:

$$B(z_1,z_2) = b_{00} + b_{10}z_1^{-1} + b_{01}z_2^{-1} + b_{11}z_1^{-1}z_2^{-1} + b_{21}z_1^{-2}z_2^{-1}$$

which is shown in Figure 2.2 below.

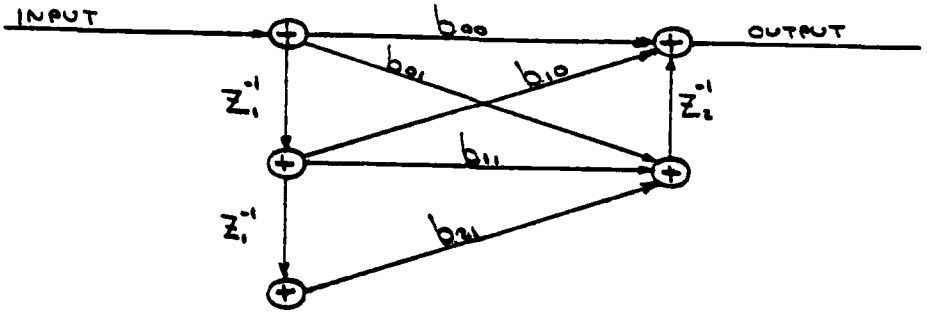


Figure 2.2

Note the chain of two z_1^{-1} operators descending on the left and the single z_2^{-1} operator ascending on the right. The nodes along these two vertical paths are connected by branches with the appropriate gains. If we label the nodes in both z_1^{-1} chains and the z_2^{-1} chain 0,1,2 and so on, from the top down, the ith node in the z_1^{-1} chain is connected to the jth node in the z_2^{-1} chain by a branch with a gain factor of b_{ij}.

Similarly the signal flowgraph for the polynomial $A(z_1,z_2)$ is shown in Figure 2.3.

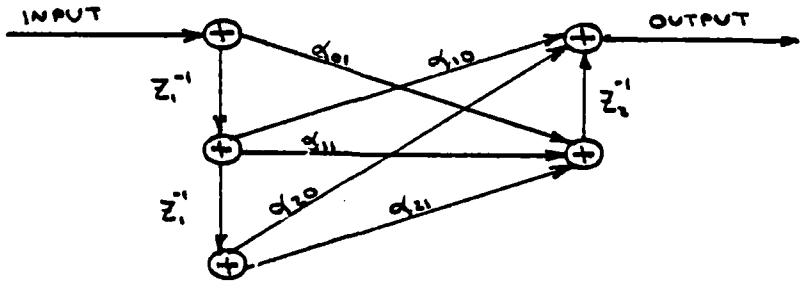


Figure 2.3

Since there is no a_{00} term, there is no direct connection between the input and output nodes of this signal flowgraph. Thus any path from the input node to the output node will encounter at least one z_1^{-1} or z_2^{-1} shift operator.

At this point it is appropriate to discuss realizations for the two shift operators z_1^{-1} and z_2^{-1}. At their simplest level, the shift operators merely select the "previous" S-tuple value in the horizontal or vertical direction. When the input to a z_1^{-1} operator is the S-tuple u(i,j) the output will be R(i-1,j). Similarly for a z_2^{-1} operator the output will be S(i,j-1) when the input is R(i,j) or S(i,j). Consequently a realization of either shift operator must embody the appropriate amount of memory to retain the "previous" S-tuple in the appropriate direction.

Interestingly enough, in the more general case where the numerator and denominator polynomials are considered jointly, the state variable realizations based on conventional signal flowgraphs may not be minimal in the sense that the transfer furnction can be realized with fewer coefficients. Consider,

$$H(z_1,z_2) = \frac{b_{10}z_1^{-1}+b_{01}z_2^{-1}+b_{11}z_1^{-1}z_2^{-1}}{1-a_{10}z_1^{-1}-a_{01}z_2^{-1}-a_{11}z_1^{-1}z_2^{-1}} \qquad \text{(II.3)}$$

The corresponding signal flow representation is shown in Figure 2.4 below:

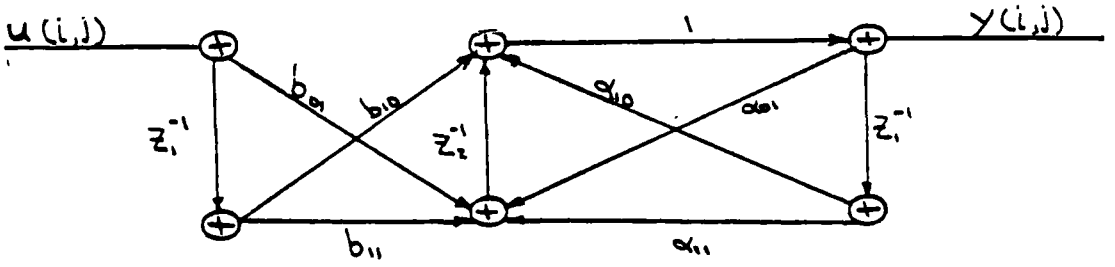


Figure 2.4

E. ANALYSIS OF ROESSER'S MODEL

Recalling from page 14 the equations of the model are:

$$R(i+1,j) = A_1R(i,j) + A_2S(i,j) + B_1u(i,j)$$

$$S(i,j+1) = A_3R(i,j) + A_4S(i,j) + B_2u(i,j)$$

$$y(i,j) = C_1R(i,j) + C_2S(i,j) + Du(i,j)$$

A_1, A_2, A_3, A_4, B_1, B_2, C_1, C_2, D are scalars or matrices of appropriate dimensions.

$$\begin{bmatrix} R(i+1,j) \\ S(i,j+1) \end{bmatrix} = \begin{bmatrix} A_1 & A_2 \\ A_3 & A_4 \end{bmatrix} \begin{bmatrix} R(i,j) \\ S(i,j) \end{bmatrix} + \begin{bmatrix} B_1 \\ B_2 \end{bmatrix} u(i,j) \qquad \text{(II.4)}$$

$$y(i,j) = [C_1 \quad C_2] \begin{bmatrix} R(i,j) \\ S(i,j) \end{bmatrix} + Du(i,j) \qquad \text{(II.5)}$$

$$R(i+1,j) = A_1R(i,j) + A_2S(i,j) + B_1u(i,j)$$

$$S(i+1,j) = A_3R(i,j) + A_4S(i,j) + B_2u(i,j)$$

And taking Z transforms:

$$z_1R(z_1,z_2) = A_1R(z_1,z_2) + Z_2S(z_1,z_2) + B_1u(z_1,z_2)$$

$$z_2S(z_1,z_2) = A_3R(z_1,z_2) + A_4S(z_1,z_2) + B_2u(z_1,z_2)$$

$$y(z_1,z_2) = [C_1 \quad C_2] \begin{bmatrix} R(z_1,z_2) \\ S(z_1,z_2) \end{bmatrix} + Du(z_1,z_2) \qquad \text{(II.6)}$$

or

$$z_1 R(z_1,z_2) - A_1 R(z_1,z_2) - A_2 S(z_1,z_2) = B_1 u(z_1,z_2)$$

$$z_2 S(z_1,z_2) - A_3 R(z_1,z_2) - A_4 S(z_1,z_2) = B_2 u(z_1,z_2)$$

or

$$R(z_1,z_2)[z_1 - A_1] - A_2 S(z_1,z_2) = B_1 u(z_1,z_2)$$

$$R(z_1,z_2)[-A_3] - [z_2 - A] S(z_1,z_2) = B_2 u(z_1,z_2]$$

or

$$\left[\begin{array}{c|c} z_1 - A_1 & -A_2 \\ \hline -A_3 & z_2 - A \end{array}\right] \left[\begin{array}{c} R(z_1,z_2) \\ S(z_1,z_2) \end{array}\right] = \left[\begin{array}{c} B_1 \\ B_2 \end{array}\right] u(z_1,z_2)$$

or

$$\left[\left[\begin{array}{c|c} z_1 & 0 \\ \hline 0 & z_2 \end{array}\right] - \left[\begin{array}{c|c} A_1 & A_2 \\ \hline A_3 & A_4 \end{array}\right]\right] \left[\begin{array}{c} R(z_1,z_2) \\ \hline S(z_1,z_2) \end{array}\right] = \left[\begin{array}{c} B_1 \\ B_2 \end{array}\right] u(z_1,z_2)$$

where $Z_1 = z_1 I$ and $Z_2 = z_2 I$, and

$$\left[\begin{array}{c} R(z_1,z_2) \\ S(z_1,z_2) \end{array}\right] = \left[\left[\begin{array}{c|c} Z_1 & 0 \\ \hline 0 & Z_2 \end{array}\right] - \left[\begin{array}{c|c} A_1 & A_2 \\ \hline A_3 & A_4 \end{array}\right]\right]^{-1} \left[\begin{array}{c} B_1 \\ B_2 \end{array}\right] u(z_1,z_2)$$

and after substitution in Equation (II.6)

$$y(z_1,z_2) = [C_1 \quad C_2] \left[\begin{array}{c|c} Z_1 & 0 \\ \hline 0 & Z_2 \end{array}\right] - \left[\begin{array}{c|c} A_1 & A_2 \\ \hline A_3 & A_4 \end{array}\right]^{-1} \left[\begin{array}{c} B_1 \\ B_2 \end{array}\right] u(z_1,z_2)$$

$$+ Du(z_1,z_2)$$

or

$$H(z_1,z_2) = \frac{y(z_1,z_2)}{u(z_1,z_2)}$$

$$= [C_1 \quad C_2] \left[\left[\begin{array}{c|c} z_1 & 0 \\ \hline 0 & z_2 \end{array}\right] - \left[\begin{array}{c|c} A_1 & A_2 \\ \hline A_3 & A_4 \end{array}\right] \right]^{-1} \left[\begin{array}{c} B_1 \\ \hline B_2 \end{array}\right]$$

$$+ D \qquad \text{(II.7)}$$

The submatrix Z_1 is simply z_1 times an identity matrix of the appropriate size. Similarly Z_2 is z_2 times an identity matrix. The objective of the state variable realization procedure is to find the matrices A, B, C, and D which yields an $F(z_1,z_2)$ that equals or approximates a desired system function $H(z_1,z_2)$. In essence, the equations of Roesser represent an implementation for which a design algorithm must be found. One choice for the state variables is the output signals from the shift operators.

Thus R(i,j) is a vector containing the output signals from the z_1^{-1} operators and S(i,j) contains the output signals from the z_2^{-1} operators. (Note that the output signal of a shift operator signal path is not necessarily the same as the nodal signal at the node to which the signal path points.) If a state variable corresponds to the output of a shift operator, the next value of that state variable must correspond to the input of the shift operator. To obtain the submatrices A_1, A_2, A_3, A_4 in equations of Roesser, we write the input signal of each shift operator in terms of the outputs of all the

shift operators, taking care to include all shift-free paths from ouput to input (see the following flowgraph).

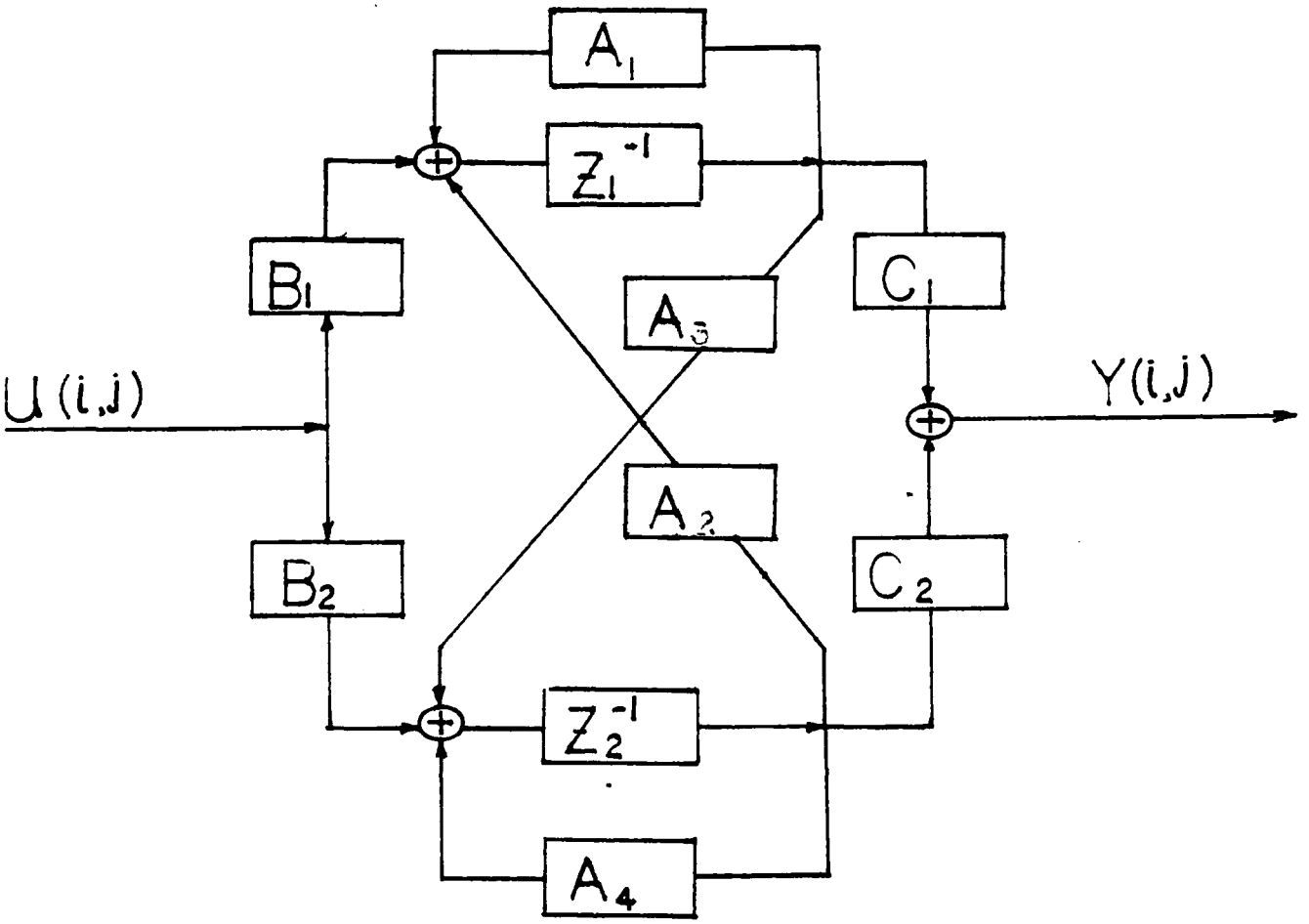


Figure 2.5

Expanding the form of Equation II-7, page 24, yields:

$$H(z_1,z_2) = [C_1 \quad C_2] \left[\begin{array}{c|c} z_1 & 0 \\ \hline 0 & z_2 \end{array}\right] - \left[\begin{array}{c|c} A_1 & A_2 \\ \hline A_3 & A_4 \end{array}\right]^{-1} \left[\begin{array}{c} B_1 \\ \hline B_2 \end{array}\right] + D$$

$$A^{-1} = \frac{1}{\det A} \text{ adj } A$$

$$A^{-1} = [C_1 \quad C_2] \left[\frac{1}{(z_1-A_1)(z_2-A_4)-A_2A_3} \left[\begin{array}{c|c} z_2-A_4 & A_2 \\ \hline A_3 & z_1-A_1 \end{array}\right] \left[\begin{array}{c} B_1 \\ \hline B_2 \end{array}\right] \right]$$

$$= [C_1 \quad C_2] \begin{bmatrix} \dfrac{(z_2-A_4)B_1}{(z_2-A_1)(A_2-z_4)-A_2A_3} & \dfrac{A_2B_2}{(z_1-A_1)(z_2-A_4)-A_2A_3} \\ \dfrac{A_3B_1}{(z_1-A_1)(z_2-A_4)-A_2A_3} & \dfrac{(z_1-A_1)B_2}{(z_1-A_1)(z_2-A_4)-A_2A_3} \end{bmatrix}$$

$$= [C_1 \quad C_2] \begin{bmatrix} \dfrac{(z_2-A_4)B_1 + A_2B_2}{(z_2-A_1)(z_2-A_4)-A_2A_3} \\ \dfrac{A_3B_1 + (z_1-A_1)B_2}{(z_1-A_1)(z_2-A_4)-A_2A_3} \end{bmatrix}$$

or

$$H(z_1,z_2) = \frac{C_1(Z_2-A_4)B_1 + C_1A_2B_2 + C_2A_3B_1 + C_2(Z_1-A_1)B_2}{(Z_1-A_1)(Z_2-A_4)-A_2A_3}$$

or

$$H(z_1,z_2) = \frac{C_1B_1Z_2-C_1B_1A_4+C_1A_2B_2+C_2A_3B_1+C_2B_2Z_1-C_2B_2A_1}{Z_1Z_2 - A_4Z_1 - A_1Z_2 - A_2A_3 + A_1A_4}$$

$$= \frac{(C_1A_2B_2+C_2A_3B_1-C_2B_2A_1-C_1B_1A_4)+(C_2B_2Z_1+C_1B_1Z_2)}{(A_1A_4-A_2A_3) - A_4Z_1 - A_1Z_2 + Z_1Z_2} \qquad \text{(II.8)}$$

Equating equation (II.8) with (II.3) on page 21 yields

$$\frac{(C_1A_2B_2+C_2A_2B_1-C_2B_2A_1-C_1B_1A_4)+C_2B_2Z_1+C_1B_1Z_2}{(A_1A_4-A_2A_3) - A_4Z_1 - A_1Z_2 + Z_1Z_2}$$

$$= \frac{b_{10}z_1^{-1}+b_{01}z_2^{-1}+b_{11}z_1^{-1}z_2^{-1}}{1-a_{10}z_1^{-1}-a_{01}z_2^{-1}-a_{11}z_1^{-1}z_2^{-1}}$$

For this example, $Z_1 = z_1$, $Z_2 = z_2$, all of the coefficients on the left hand side are scalars. Equation terms of equal powers of z_1 and z_2,

$$C_1A_2B_2+C_2A_3B_1-C_2B_2A_1-C_1B_1A_4 = b_{11} = 0 \quad \text{(II.9)}$$

$$C_2B_2 = b_{10} \quad \text{(II.10)}$$

$$C_1B_1 = b_{01} \quad \text{(II.11)}$$

$$A_1A_4 - A_2A_3 = 1 \quad \text{(II.12)}$$

$$A_4 = a_{10} \quad \text{(II.13)}$$

$$A_1 = a_{01} \quad \text{(II.14)}$$

$$a_{11} = -1 \quad \text{(II.15)}$$

From these equations, assuming that $B_1 = B_2 = 1$, it follows that:

$$C_1 = b_{10}$$

$$C_2 = b_{01}$$

$$A_1 = a_{01}$$

$$A_4 = a_{10}$$

From Equation (II-12):

$$A_1A_4 - A_2A_3 = 1 = -a_{11}$$

$$-A_2A_3 = -a_{11} - A_1A_4$$

$$A_2A_3 = a_{11} + a_{10}a_{01}$$

Let A_2 and A_3 take on particular values p and q respectively,

$$A_3 = q \qquad A_2 = p$$

or

$$pq = a_{11} + a_{10}a_{01} \tag{II.16}$$

From Equation (II-6):

$$C_1A_2B_2 + C_2A_3B_1 - C_2B_2A_1 - C_1B_1A_4 = b_{11}$$

or,

$$b_{01}p + b_{10}q - b_{10}a_{01} - b_{01}a_{10} - b_{11} = 0 \tag{II.17}$$

Substituting Equation (II.16) into Equation (II.17):

$$b_{01}\frac{a_{11}+a_{10}a_{01}}{q} + b_{10}q - b_{10}a_{01} - b_{01}a_{10} - b_{11} = 0$$

or

$$b_{10}q^2-(b_{10}a_{01}+b_{01}a_{10}+b_{11})] + (b_{01}a_{11}+b_{01}a_{10}a_{01}) = 0 . \quad (II.18)$$

The results are just the same as in [Ref. 8]. After the comparison between Roesser's model and the 2-D IIR filter, described by Equation (II.3), we have:

$$A_1 = a_{01}$$

$$A_2 = p \; ; \quad pq = a_{11} = a_{10}a_{01}$$

$$A_3 = q; \quad b_{10}q^2-(b_{10}a_{01}+b_{01}a_{10}+b_{11})q+(b_{01}a_{11}+b_{01}a_{10}a_{01}) = 0$$

$$A_4 = a_{10}$$

$$C_1 = b_{01} \qquad (II.19)$$

$$C_2 = b_{10}$$

$$B_1 = 1$$

$$B_2 = 1$$

$$D = 0$$

The foregoing equations relate the coefficients of the 2-D transfer function to the terms of the system matrices of the Roesser model, Equation (II.1).

Kung et al. [Ref. 2] have shown that the following state variable equations, which use only two shift operators, will also realize $H(z_1,z_2)$. For the foregoing example,

$$\begin{bmatrix} R(i+1,j) \\ S(i,j+1) \end{bmatrix} = \left[\begin{array}{c|c} a_{01} & p \\ \hline q & a_{10} \end{array}\right] \begin{bmatrix} R(i,j) \\ S(i,j) \end{bmatrix} + \begin{bmatrix} 1 \\ 1 \end{bmatrix} u(i,j)$$

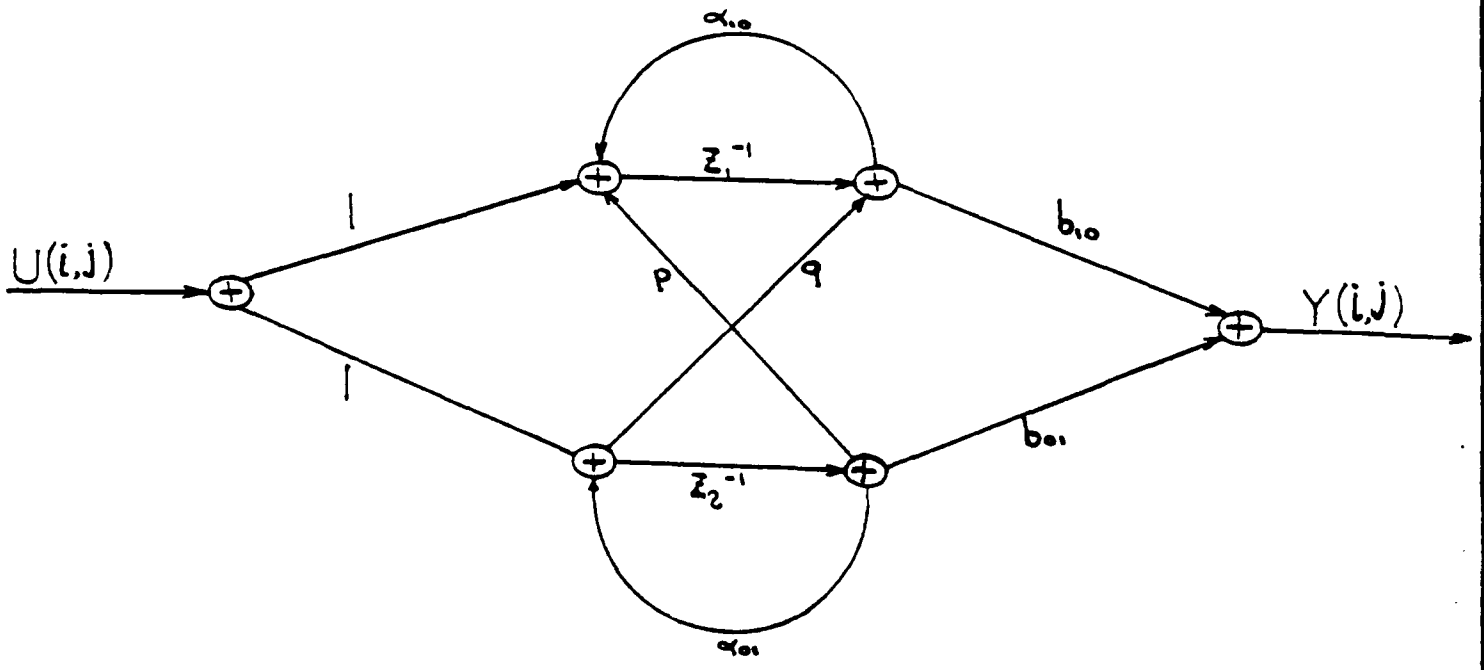
$$Y(i,j) = [b_{10} \quad b_{01}] \begin{bmatrix} R(i,j) \\ S(i,j) \end{bmatrix}$$

or

$$H(z_1,z_2) = [b_{10} \quad b_{01}] \begin{bmatrix} z_1-a_{10} & -p \\ -q & z_2-a_{01} \end{bmatrix}^{-1} \begin{bmatrix} 1 \\ 1 \end{bmatrix}.$$

We can construct a signal flowgraph with only two shift operators. It is an equivalent figure to that on page 25.

Kung et al. [Ref. 2] have also shown that state-variable realizations of the form of the equations above may be generalized for any system function $H(z_1,z_2)$ which satisfies the following three conditions:



1) The constant term in the numerator, $b_{00} = 0$, must be zero.

2) The largest powers of z_1^{-1}, in the numerator and denominator polynomials, must be equal, and

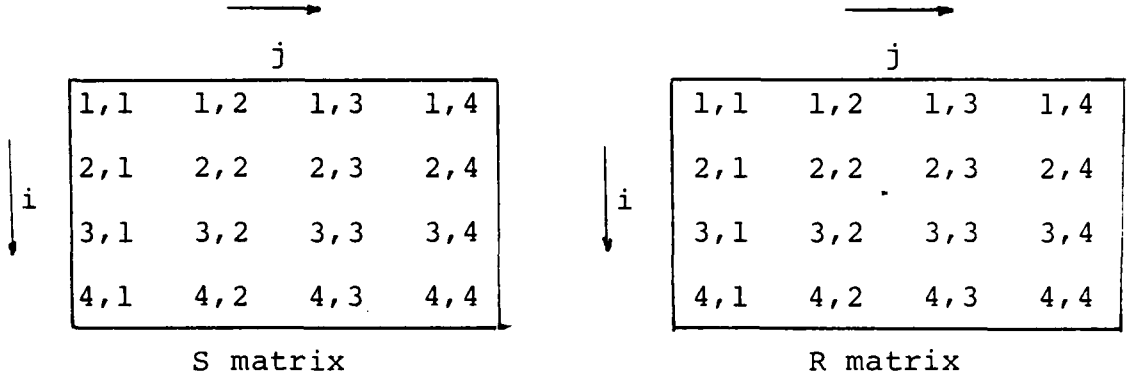
3) The largest powers of z_2^{-1} in the numerator and denominator polynomials must be equal.

There is one potential difficulty with state variable realizations of this type. The nonlinear equations defining p and q may result in complex values for these constants. For example, when $b_{10} = b_{01} = 1$, $b_{11} = 0$, $a_{10} = a_{01} = 2$ and $a_{11} = 1$, we get $p = q^* = 2 \pm j$.

III. THE PROGRAM OF ROESSER'S EQUATIONS WITH SCALAR COEFFICIENTS (FIRST ORDER)

A. AN EXAMPLE

For a 4 × 4 data field the S and R matrices are indexed as follows:



For 4 × 4 Matrices

The Initial Conditions are given by the values

$$R(1,1),\ R(2,1),\ R(3,1),\ R(4,1)$$

$$S(1,1),\ S(1,2),\ S(1,3),\ S(1,4)$$

The 2-D state variable equations can be written as:

$$R(i+1,j) = A_1R(i,j) + A_2S(i,j) + B_1u(i,j)$$

$$S(i,j+1) = A_3R(i,j) + A_4S(i,j) + B_2u(i,j)$$

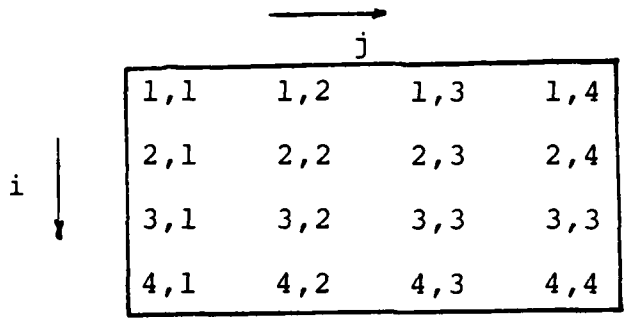
$$y(i,j) = [C_1 \quad C_2] \begin{bmatrix} R(i,j) \\ S(i,j) \end{bmatrix}$$

The input 2-D field is taken to be,

$$u(i,j) = 1, \quad \text{for} \quad i = j = 1$$

$$= 0, \quad \text{otherwise.}$$

The output data field is indexed as:



Y output matrix

B. THE 2-D FOURIER TRANSFORM

The 2-D discrete Fourier transform Y(m,n) of the output y(i,j) can be written as,

$$Y(m,n) = \sum_{\ell=0}^{M-1} \sum_{k=0}^{N-1} y(\ell,k) e^{-j2\pi\frac{\ell m}{M}} e^{-j2\pi\frac{kn}{N}}$$

or for convenience,

$$Y(m,n) = \sum_{\ell=1}^{M} \sum_{k=1}^{N} y(\ell,k) e^{j2\pi\frac{(\ell-1)(m-1)}{M}} e^{-j2\pi\frac{(k-1)(n-1)}{N}}$$

Y(m,n): 2-D D.F.T. {y(i,j)}

M × N: The dimension of the given data y(ℓ,k) and D.F.T. Y(m,n) also.

y(ℓ,k): Given data (The output as described above).

To develop the D.F.T. for two-dimensional signals we consider a finite area sequence y(ℓ,k) which is zero outside

the interval $0 \leq \ell \leq M-1$, $0 \leq k \leq N-1$, i.e., it is of area (M,N) and construct the periodic sequence:

$$\tilde{y}(\ell,k) = y[((\ell))_M((k))_N)]$$

The original sequence $y(\ell,k)$ is recovered by extracting one period of $\tilde{y}(\ell,k)$, i.e.,

$$y(\ell,k) = \tilde{y}(\ell,k) R_{M,N}(\ell,k)$$

$$R_{M,N}(\ell,k) = \begin{cases} 1, & 0 \leq \ell \leq M-1,\ 0 \leq k \leq N-1 \\ 0, & \text{otherwise .} \end{cases}$$

We then define the discrete Fourier transform of $y(\ell,k)$ to correspond to the Fourier series coefficients of $\tilde{y}(\ell,k)$. However, just as we did with one-dimensional sequences, we will maintain the duality between the time and frequency domains by interpreting the D.F.T. coefficients to also be a finite 2-D sequence. Thus with Y(m,n) denoting the D.F.T. of $y(\ell,k)$, we can write

$$Y(m,n) = \sum_{\ell=0}^{M-1} \sum_{k=0}^{N-1} y(\ell,k) e^{-j2\pi\frac{\ell m}{M}} e^{-j2\pi\frac{kn}{N}} R_{M,N}(m,n)$$

or

$$y(\ell,k) = \frac{1}{MN} \sum_{m=0}^{M-1} \sum_{y=0}^{N-1} Y(m,n)\ e^{j2\pi\frac{\ell m}{M}} e^{j2\pi\frac{kn}{N}} R_{M,N}(\ell,k)$$

or,

$$Y(m,n) = \sum_{\ell=1}^{M} \sum_{k=1}^{N} y_{i,j}(\ell,k) e^{-j2\pi\frac{(\ell-1)(m-1)}{N}} e^{-j2\pi\frac{(k-1)(n-1)}{N}}$$

As an example, consider the case for M = N = 5.

Given 2-D Data Sequence

$$y(\ell,k) = \begin{matrix} 1,1 & 1,2 & 1,3 & 1,4 & 1,5 \\ 2,1 & 2,2 & 2,3 & 2,4 & 2,5 \\ 3,1 & 3,2 & 3,3 & 3,4 & 3,5 \\ 4,1 & 4,2 & 4,3 & 4,4 & 4,5 \\ 5,1 & 5,2 & 5,3 & 5,4 & 5,5 \end{matrix}$$

i

j

Matrix 5×5

M=5 N=5

ℓ=1,2,3,4,5 k=1,2,3,4,5

Then,

$$Y(1,1)_{m=1,n=1} = \begin{matrix} y(1,1) + y(1,2) + y(1,3) + y(1,4) + y(1,5) \\ +y(2,1) + y(2,2) + y(2,3) + y(2,4) + y(2,5) \\ +y(3,1) + y(3,2) + y(3,3) + y(3,4) + y(3,5) \\ +y(4,1) + y(4,2) + y(4,3) + y(4,4) + y(4,5) \\ +y(5,1) + y(5,2) + y(5,3) + y(5,4) + y(5,5) \end{matrix}$$

$$
\begin{aligned}
Y(1,2) = {} & y(1,1) + y(1,2)e^{2(-j\frac{\pi}{5})} + y(1,3)e^{4(-j\frac{\pi}{5})} + y(1,4)e^{6(-j\frac{\pi}{5})} + y(1,5)e^{8(-j\frac{\pi}{5})} \\
& + y(2,1) + y(2,2)e^{2(-j\frac{\pi}{5})} + y(2,3)e^{4(-j\frac{\pi}{5})} + y(2,4)e^{6(-j\frac{\pi}{5})} + y(2,5)e^{8(-j\frac{\pi}{5})} \\
& + y(3,1) + y(3,2)e^{2(-j\frac{\pi}{5})} + y(3,3)e^{4(-j\frac{\pi}{5})} + y(3,4)e^{6(-j\frac{\pi}{5})} + y(3,5)e^{8(-j\frac{\pi}{5})} \\
& + y(4,1) + y(4,2)e^{2(-j\frac{\pi}{5})} + y(4,3)e^{4(-j\frac{\pi}{5})} + y(4,4)e^{6(-j\frac{\pi}{5})} + y(4,5)e^{8(-j\frac{\pi}{5})} \\
& + y(5,1) + y(5,2)e^{2(-j\frac{\pi}{5})} + y(5,3)e^{4(-j\frac{\pi}{5})} + y(5,4)e^{6(-j\frac{\pi}{5})} + y(5,5)e^{8(-j\frac{\pi}{5})}
\end{aligned}
$$

m=1,n=2

$$
\begin{aligned}
Y(1,3) = {} & y(1,1) + y(1,2)e^{4(-j\frac{\pi}{5})} + y(1,3)e^{8(-j\frac{\pi}{5})} + y(1,4)e^{12(-j\frac{\pi}{5})} + y(1,5)e^{16(-j\frac{\pi}{5})} \\
& + y(2,1) + y(2,2)e^{4(-j\frac{\pi}{5})} + y(2,3)e^{8(-j\frac{\pi}{5})} + y(2,4)e^{12(-j\frac{\pi}{5})} + y(2,5)e^{16(-j\frac{\pi}{5})} \\
& + y(3,1) + y(3,2)e^{4(-j\frac{\pi}{5})} + y(3,3)e^{8(-j\frac{\pi}{5})} + y(3,4)e^{12(-j\frac{\pi}{5})} + y(3,5)e^{16(-j\frac{\pi}{5})} \\
& + y(4,1) + y(4,2)e^{4(-j\frac{\pi}{5})} + y(4,3)e^{8(-j\frac{\pi}{5})} + y(4,4)e^{12(-j\frac{\pi}{5})} + y(4,5)e^{16(-j\frac{\pi}{5})} \\
& + y(5,1) + y(5,2)e^{4(-j\frac{\pi}{5})} + y(5,3)e^{8(-j\frac{\pi}{5})} + y(5,4)e^{12(-j\frac{\pi}{5})} + y(5,5)e^{16(-j\frac{\pi}{5})}
\end{aligned}
$$

m=1,n=3

$$
\begin{aligned}
Y(1,4) = {} & y(1,1) + y(1,2)e^{6(-j\frac{\pi}{5})} + y(1,3)e^{12(-j\frac{\pi}{5})} + y(1,4)e^{18(-j\frac{\pi}{5})} + y(1,5)e^{24(-j\frac{\pi}{5})} \\
& + y(2,1) + y(2,2)e^{6(-j\frac{\pi}{5})} + y(2,3)e^{12(-j\frac{\pi}{5})} + y(2,4)e^{18(-j\frac{\pi}{5})} + y(2,5)e^{24(-j\frac{\pi}{5})} \\
& + y(3,1) + y(3,2)e^{6(-j\frac{\pi}{5})} + y(3,3)e^{12(-j\frac{\pi}{5})} + y(3,4)e^{18(-j\frac{\pi}{5})} + y(3,5)e^{24(-j\frac{\pi}{5})} \\
& + y(4,1) + y(4,2)e^{6(-j\frac{\pi}{5})} + y(4,3)e^{12(-j\frac{\pi}{5})} + y(4,4)e^{18(-j\frac{\pi}{5})} + y(4,5)e^{24(-j\frac{\pi}{5})} \\
& + y(5,1) + y(5,2)e^{6(-j\frac{\pi}{5})} + y(5,3)e^{12(-j\frac{\pi}{5})} + y(5,4)e^{18(-j\frac{\pi}{5})} + y(5,5)e^{24(-j\frac{\pi}{5})}
\end{aligned}
$$

m=1,n=4

$$
\begin{aligned}
Y(2,1) = {} & y(1,1) + y(1,2) + y(1,3) + y(1,4) + y(1,5) \\
& + y(2,1)e^{2(-j\frac{\pi}{5})} + y(2,2)e^{2(-j\frac{\pi}{5})} + y(2,3)e^{2(-j\frac{\pi}{5})} + y(2,4)e^{2(-j\frac{\pi}{5})} + y(2,5)e^{2(-j\frac{\pi}{5})} \\
& + y(3,1)e^{4(-j\frac{\pi}{5})} + y(3,2)e^{4(-j\frac{\pi}{5})} + y(3,3)e^{4(-j\frac{\pi}{5})} + y(3,4)e^{4(-j\frac{\pi}{5})} + y(3,5)e^{4(-j\frac{\pi}{5})} \\
& + y(4,1)e^{6(-j\frac{\pi}{5})} + y(4,2)e^{6(-j\frac{\pi}{5})} + y(4,3)e^{6(-j\frac{\pi}{5})} + y(4,4)e^{6(-j\frac{\pi}{5})} + y(4,5)e^{6(-j\frac{\pi}{5})} \\
& + y(5,1)e^{8(-j\frac{\pi}{5})} + y(5,2)e^{8(-j\frac{\pi}{5})} + y(5,3)e^{8(-j\frac{\pi}{5})} + y(5,4)e^{8(-j\frac{\pi}{5})} + Y95,5)e^{8(-j\frac{\pi}{5})}
\end{aligned}
$$

m=2,n=1

In Appendix A we give a listing of the programs that have been written to generate y(i,j) and Y(m,n).

C. NUMERICAL EXAMPLES

Three numerical examples which depend on Equation (II.19) are used to demonstrate the program in Appendix A.

First example:

$$H(z_1,z_2) = \frac{.5(z_1^{-1} + z_2^{-1})}{1 - .2z_1^{-1} - .3z_2^{-1}}$$

yields

$$a_{11} = 0$$

$$A_1 = a_{01} = 0.3$$

$$A_4 = a_{10} = 0.2$$

$$C_1 = b_{10} = 0.5$$

$$C_2 = b_{01} = 0.5$$

$$B_1 = 1$$

$$B_2 = 1$$

$$D = 0$$

After substitution of these values in Eqs. (II.16) and (II.18) we identify

$$a_{11} = 0 \qquad b_{00} = 0 \qquad b_{11} = 0$$

$$A_1 = a_{01} = 0.3$$

$$A_2 = p = 0.2$$

$$A_3 = q = 0.3$$

$$A_4 = a_{10} = 0.2$$

$$C_1 = b_{01} = 0.5 \qquad \text{(III.1a)}$$

$$C_2 = b_{10} = 0.5$$

$$B_1 = 1$$

$$B_2 = 1$$

$$D = 0$$

Second example:

Proceeding in a similar way with

$$H(z_1,z_2) = \frac{0.25z_1^{-1}+0.3z_2^{-1}+0.2z_1^{-1}z_2^{-1}}{1-0.125z_1^{-1}-0.2z_2^{-1}-0.1z_1^{-1}z_2^{-1}}$$

yields

$$a_{11} = 0.1 \qquad b_{00} = 0$$

$$A_1 = a_{01} = 0.2 \qquad b_{11} = 0.2$$

$$A_2 = p = 0.125 \qquad = 0.83$$

$$A_3 = q = 1 \quad \text{or} \quad = 0.15 \qquad \text{(III.1b)}$$

$$A_4 = a_{10} = 0.125$$

$$C_1 = b_{01} = 0.3$$

$$C_2 = b_{10} = 0.25$$

$B_1 = 1$

$B_2 = 1$

$D = 0$

Third example:

Proceeding in a similar way with

$$H(z_1,z_2) = \frac{0.25z_1^{-1}+0.15z_2^{-1}+0.72z_1^{-1}z_2^{-1}}{1-0.135z_1^{-1}-0.25z_2^{-1}-0.15z_1^{-1}z_2^{-1}}$$

yields

$b_{00} = 0 \quad b_{11} = 0.25$

$a_{11} = 0.15$

$A_1 = a_{01} = 0.25$

$A_2 = p = 0.1312$

$A_3 = q = 1.4$ (III.1c)

$A_4 = a_{10} = 0.135$

$C_1 = b_{01} = 0.15$

$C_2 = b_{10} = 0.25$

$B_1 = 1$

$B_2 = 1$

$D = 0$

Zero initial conditions were assumed for all examples. The simulation results are presented in Figures 3-1, 3-2 and 3-3.

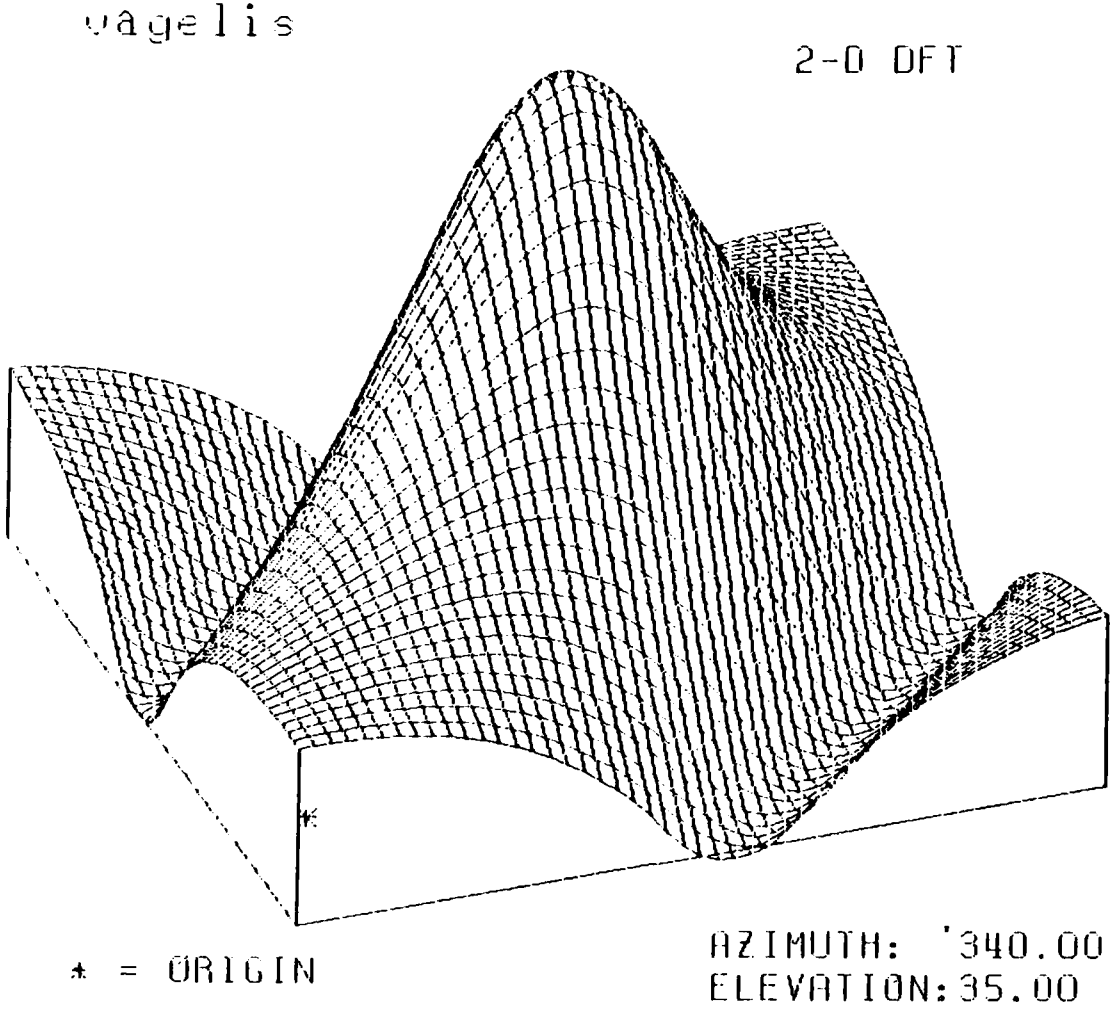


Figure 3-1a. 2-D D.F.T. Squence, Y(m,n) for Example 1

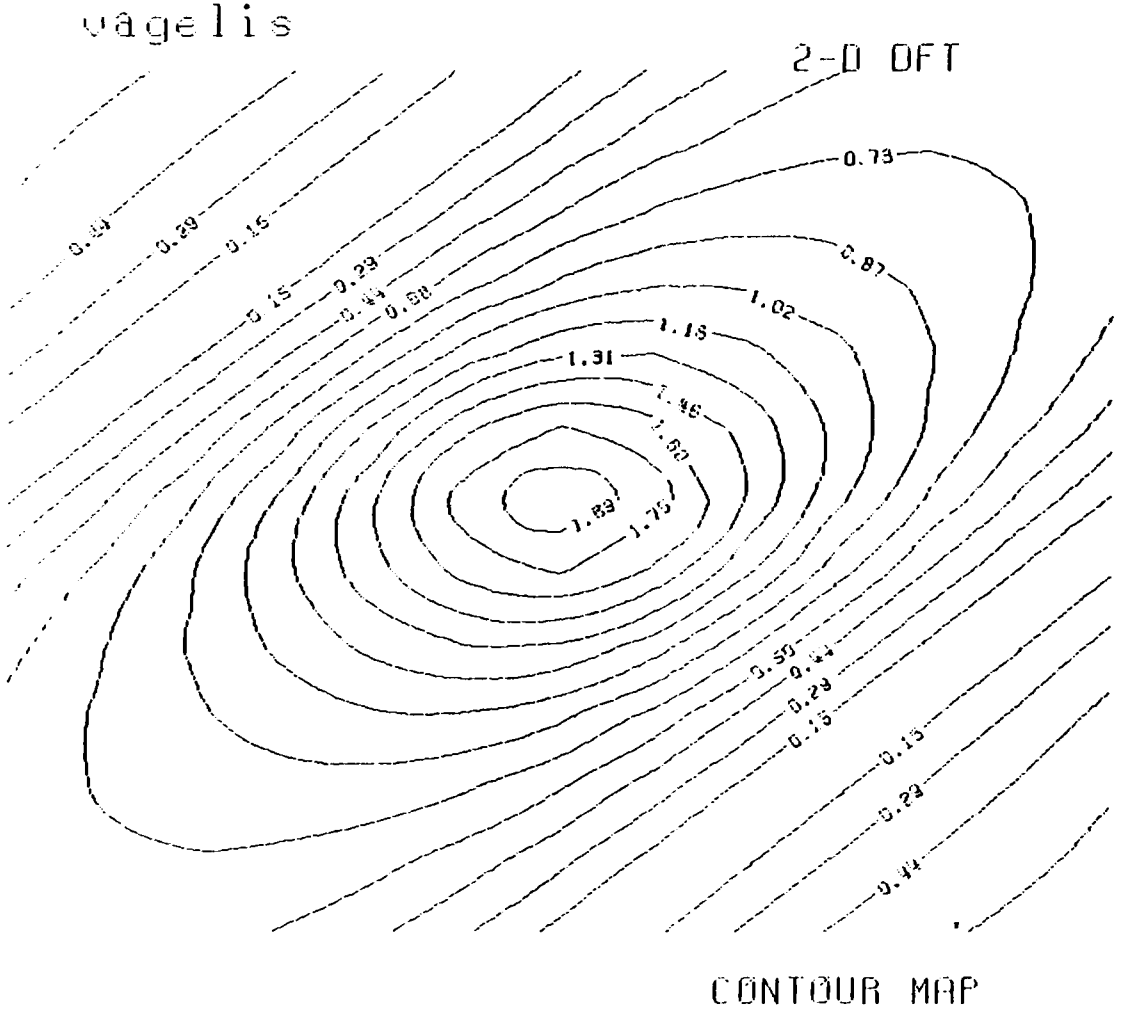


Figure 3-1b, Contour Map for Figure 3-1a

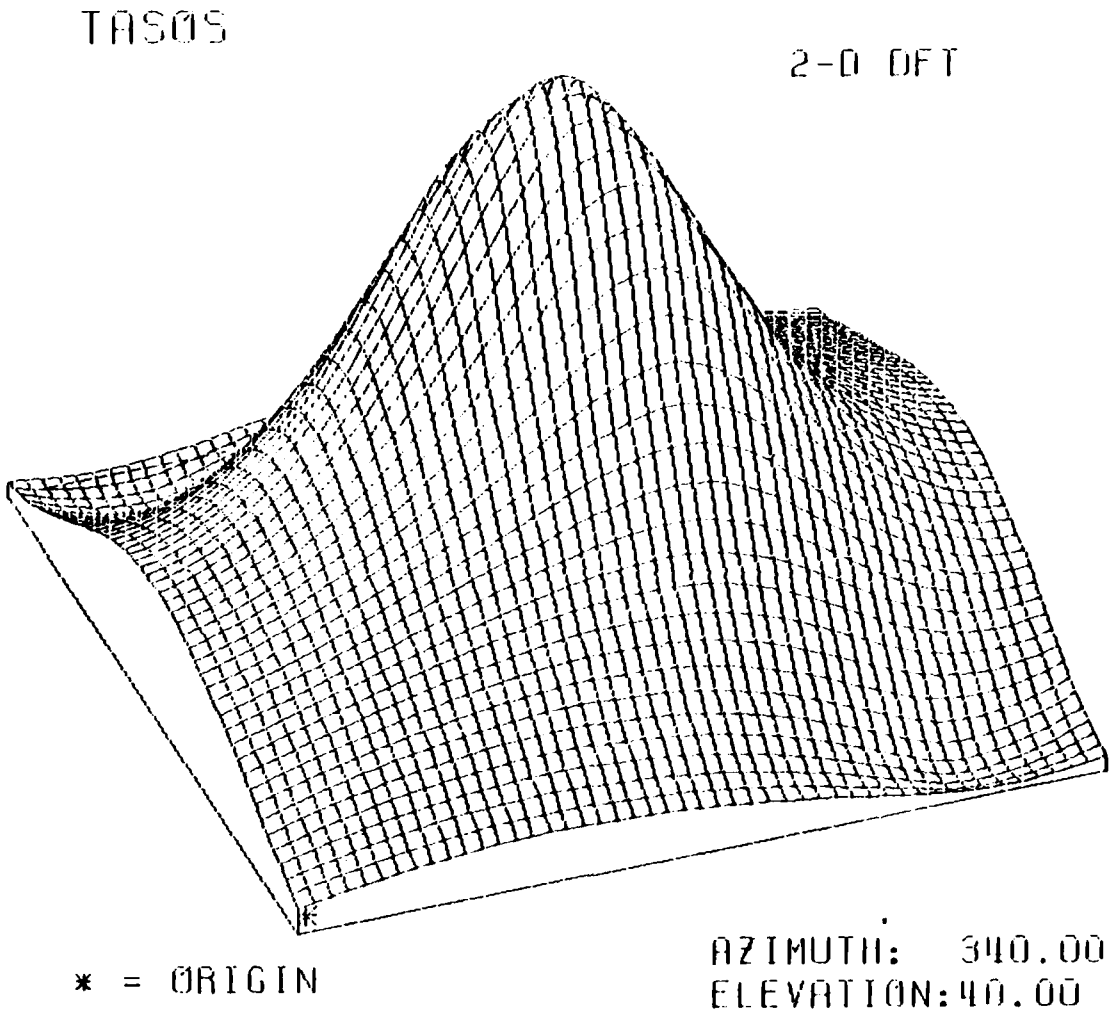


Figure 3-2a. 2-D D.F.T. Sequence, Y(m,n) for Example 2

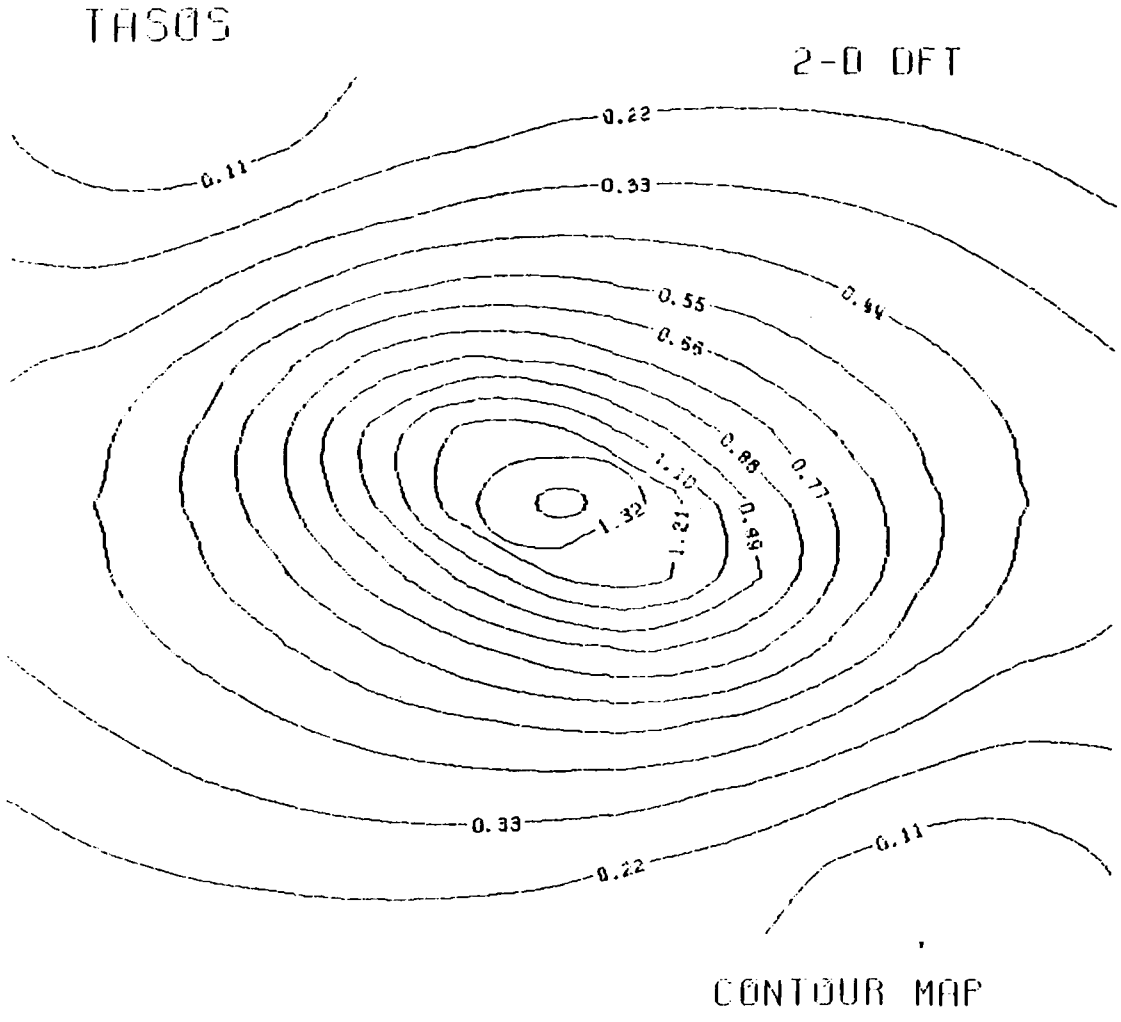


Figure 3-2b. Contour Map for Figure 3-2a

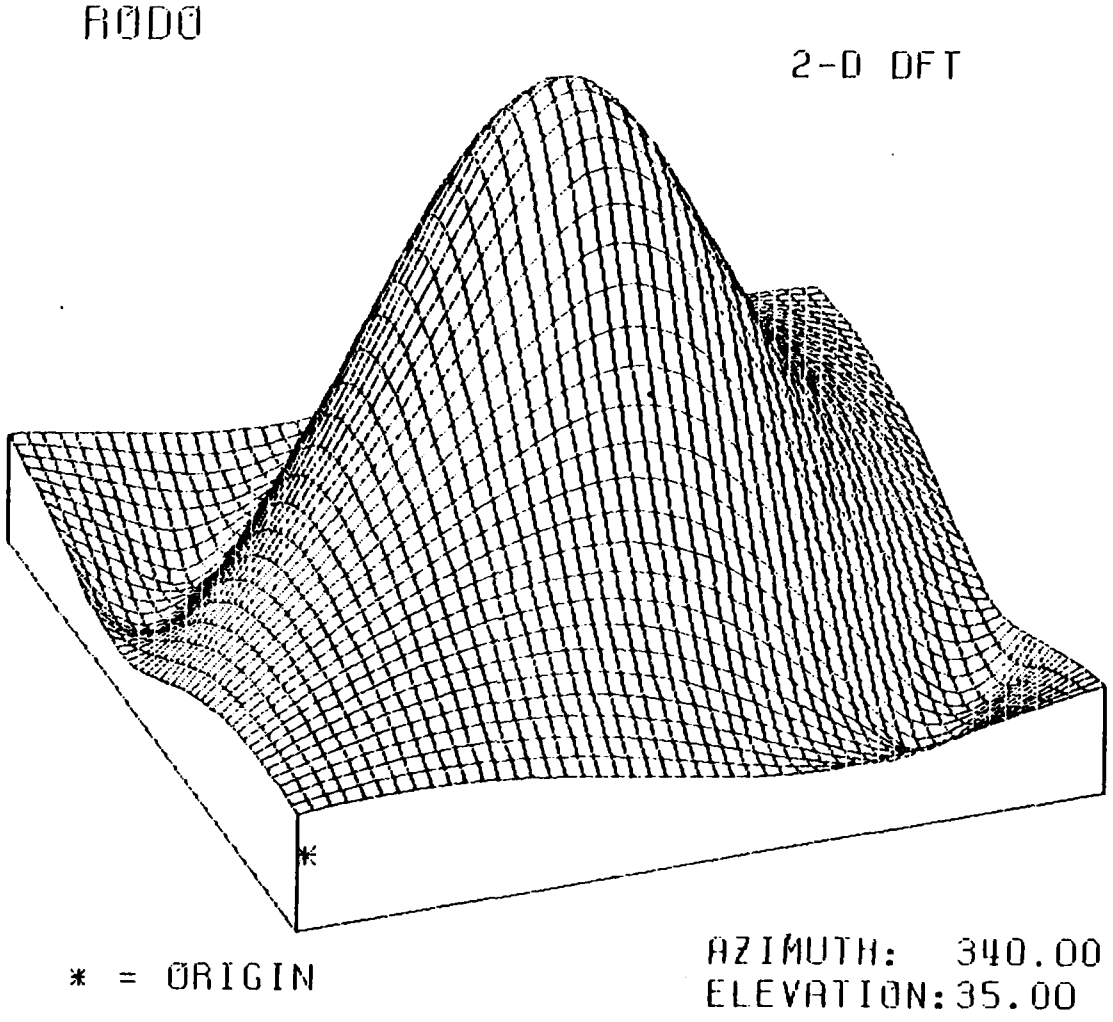


Figure 3-3a. 2-D D.F.T. Sequence, Y(m,n) for Example 3

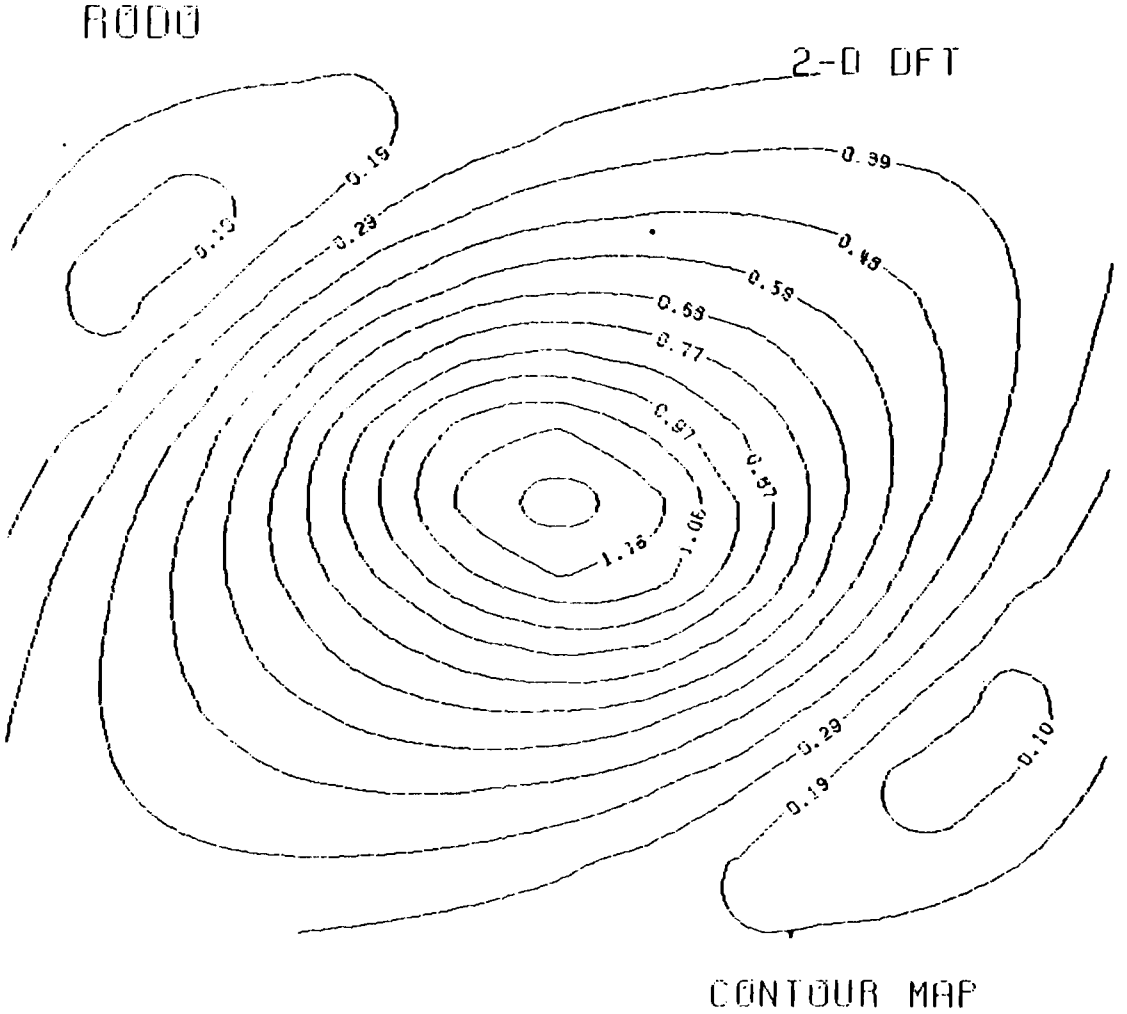


Figure 3-3b. Contour Map for Figure 3-3a

In order to verify the correctness of the output produced by Roesser, the 2-D D.F.T. Y(m,n) plots for these examples were compared with the corresponding $|H(z_1,z_2)|$. 2-D transfer function plots $|H(z_1,z_2)|$ for Examples 1, 2 and 3 are shown in Figs. 3-4a,b, 3-5a,b and 3-6a,b respectively. The listing of a program used to generate these plots can be found in Appendix B.

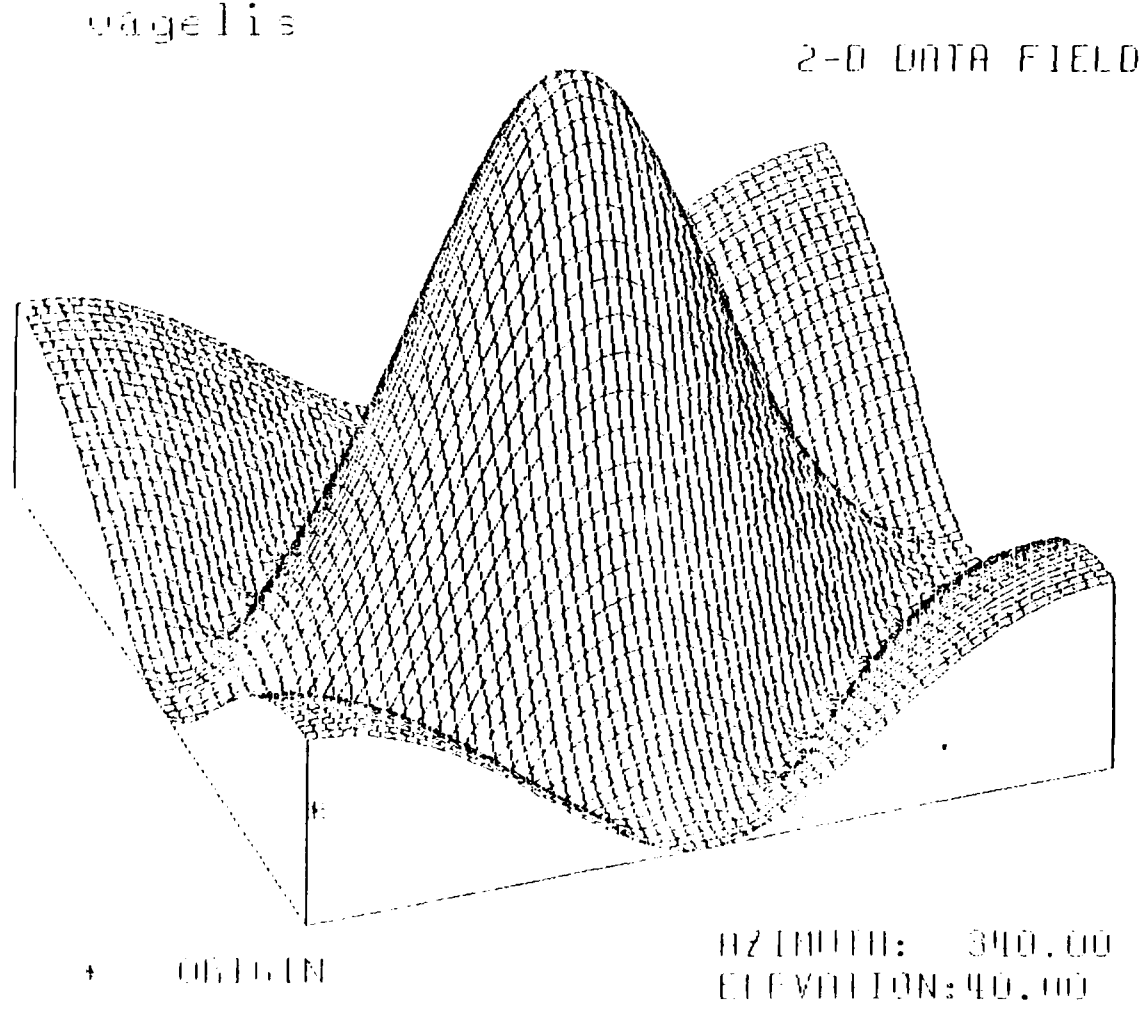


Figure 3-4a. Transfer Function $|H(z_1,z_2)|$, $z_1 = e^{j\omega_1}$, $z_2 = e^{j\omega_2}$ for Example 1

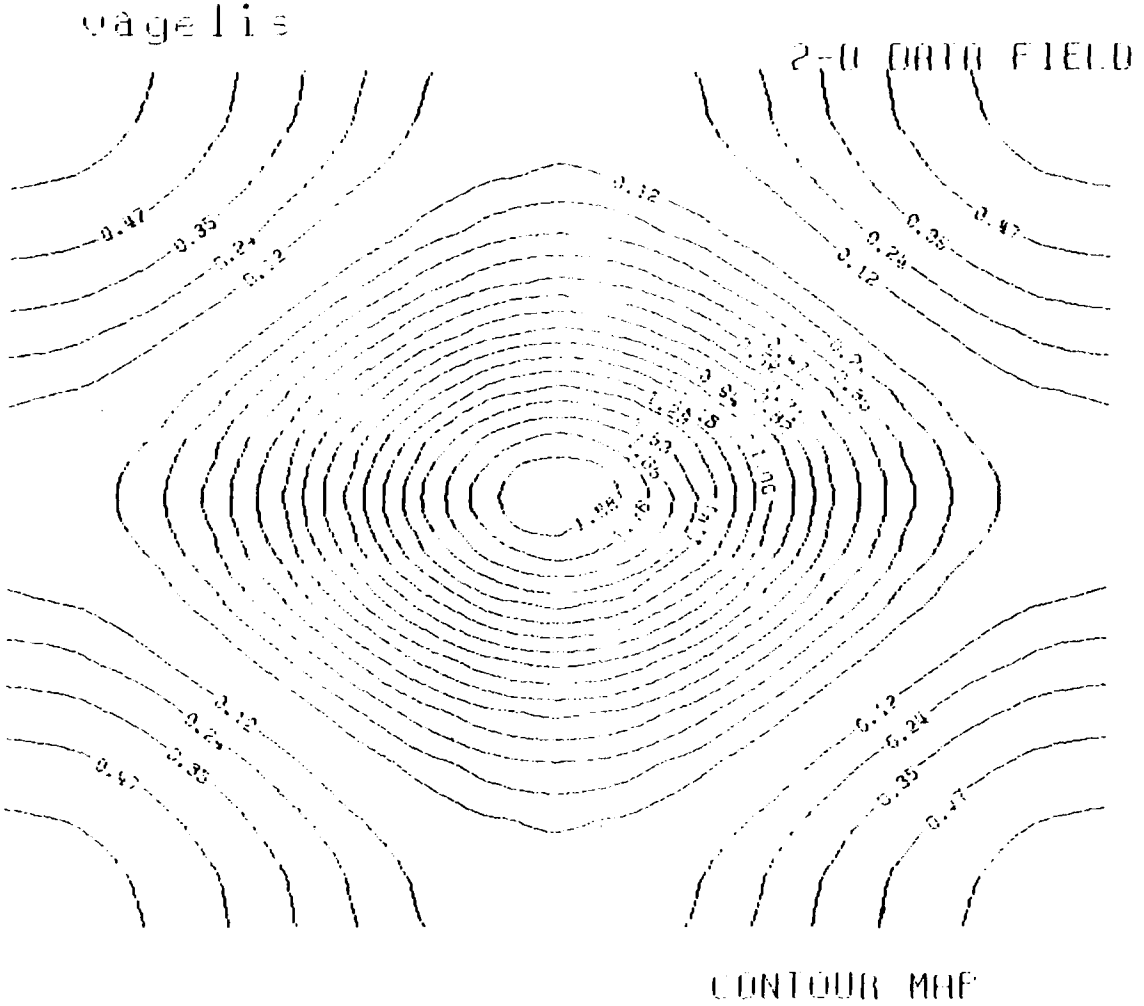


Figure 3-4b. Contour Map for Figure 3-4a

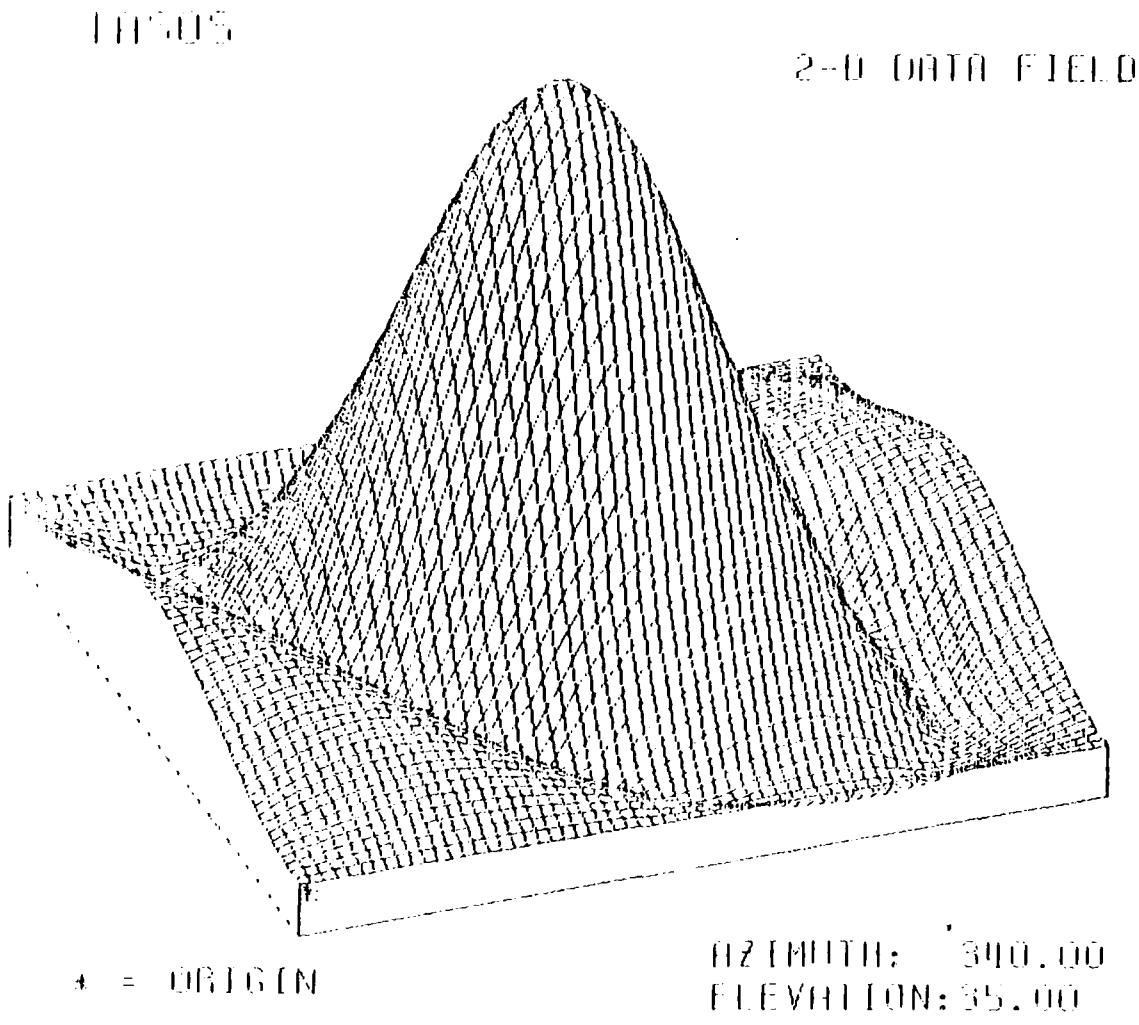


Figure 3-5a. Transfer Function $|H(z_1,z_2)|$, $z_1 = e^{j\omega_1}$, $z_2 = e^{j\omega_2}$ for Example 2

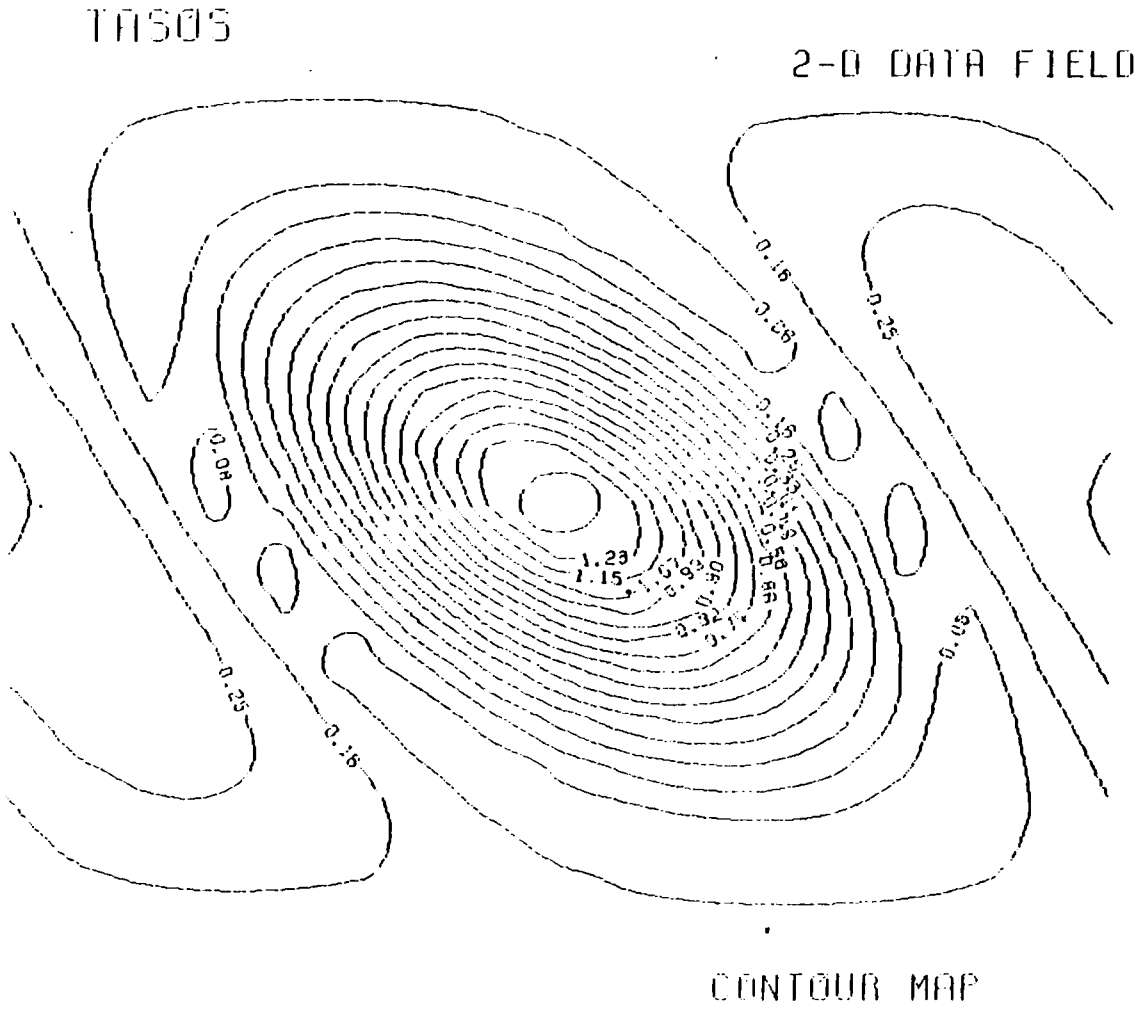


Figure 3-5b. Contour Map for Figure 3-5a

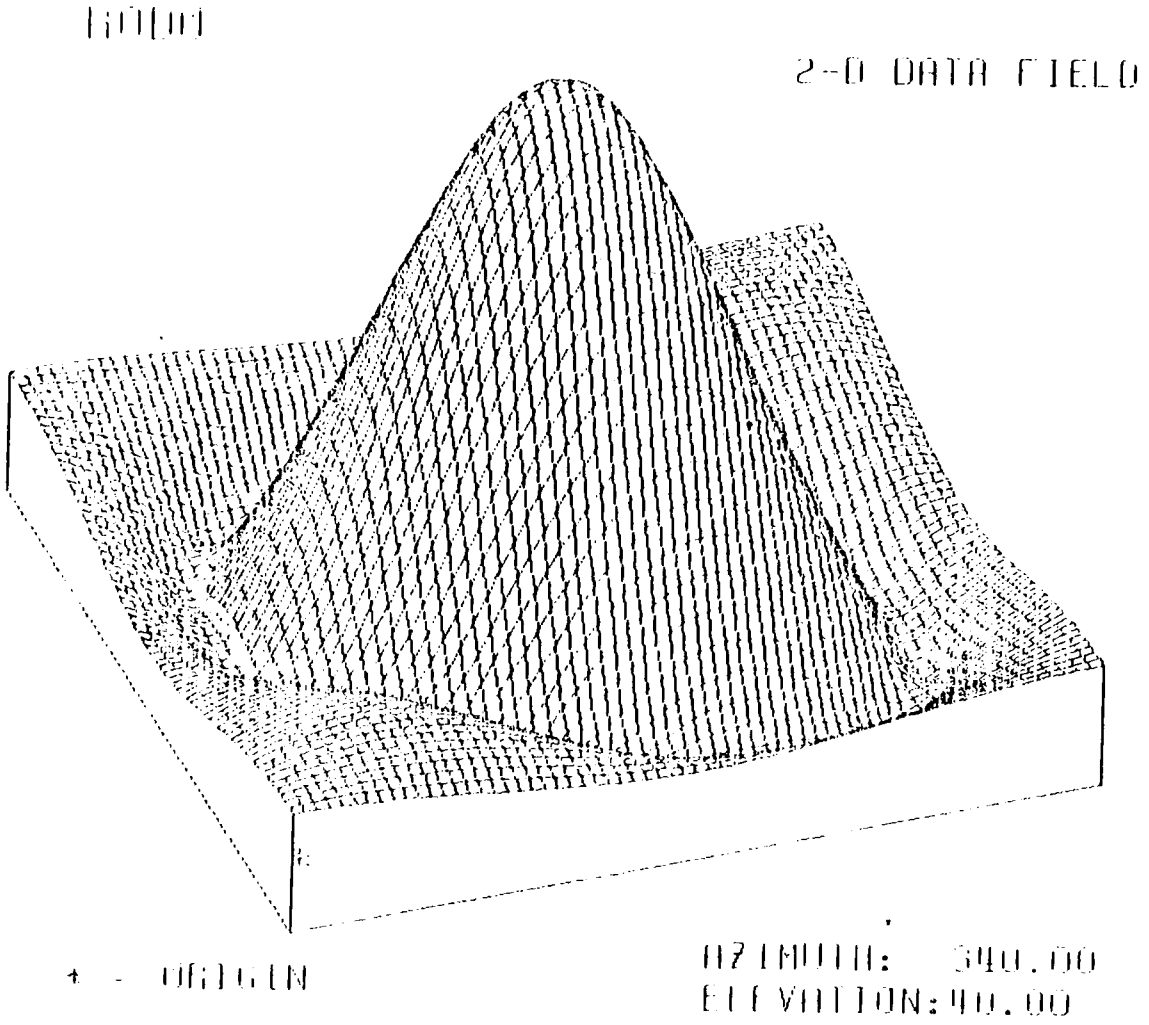


Figure 3-6a. Transfer Function $|H(z_1,z_2)|$, $z_1 = e^{j\omega_1}$, $z_2 = e^{j\omega_2}$ for Example 3

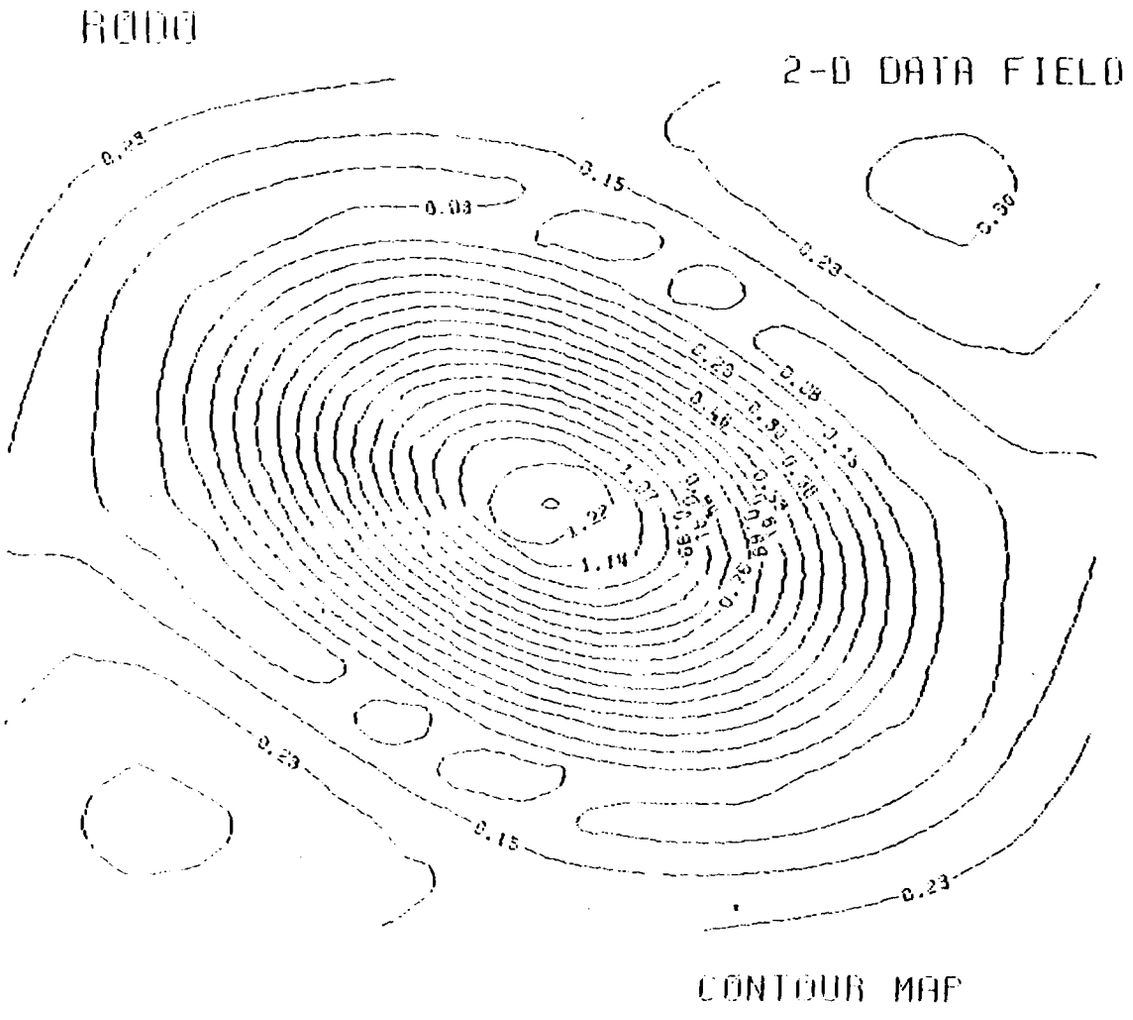


Figure 3-6b. Contour Map for Figure 3-6a

IV. EXTENSION OF ROESSER'S MODEL TO SECOND AND HIGHER ORDERS

A. MINIMIZING THE NUMBER OF SHIFT OPERATORS

In order to minimize the number of shift operators we follow the procedure given in Kung [Ref. 8]. Let us consider the simple 2-D IIR filter transfer function given by

$$H(z_1,z_2) = \frac{b_{00}+b_{10}z_1^{-1}+b_{01}z_2^{-1}+b_{11}z_1^{-1}z_2^{-1}+b_{21}z_1^{-2}z_2^{-1}}{1-a_{10}a_1^{-1}-a_{01}a_z^{-1}-a_{11}z_1^{-1}z_2^{-1}-a_{10}z_1^{-2}-a_{21}z_1^{-2}z_2^{-1}}$$

$$= \frac{B(z_1,z_2)}{1-A(z_1,z_2)} \qquad \text{(IV.1)}$$

Our problem will be drawing a detailed signal flowgraph for the system function $H(z_1,z_2)$. We can do this simply enough by combining the flowgraphs on Figures 2-2 and 2-3 to get the

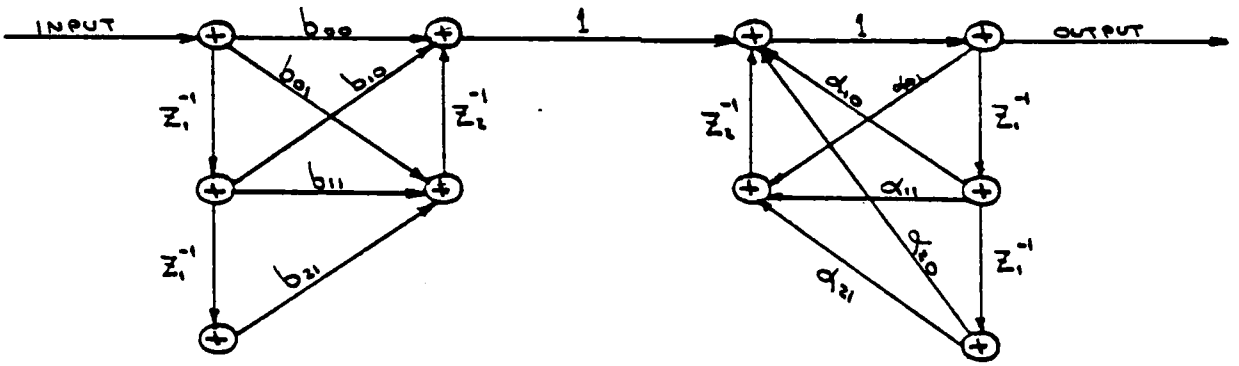


Figure 4-1

This flowgraph can be made even simpler because the shift operation is distributive over addition. We can combine the two z_2^{-1} operators into a single one, yielding the following flowgraph.

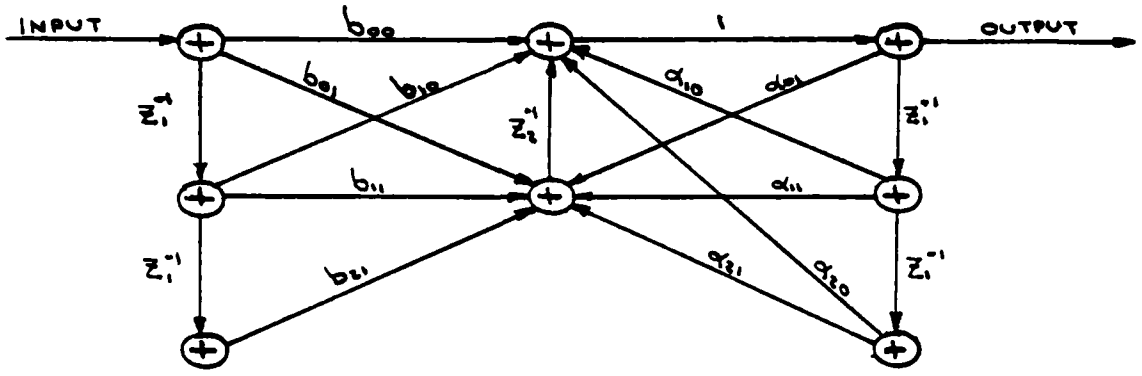


Figure 4-2

Doing so reduces the number of shift operators that need to be implemented and consequently the amount of storage necessary.

There are other signal flowgraphs which give rise to the desired system function $H(z_1,z_2)$. For example, we could invert the order of the $B(z_1,z_2)$ filter and the feedback loop containing $A(z_1,z_2)$ to obtain the block diagram:

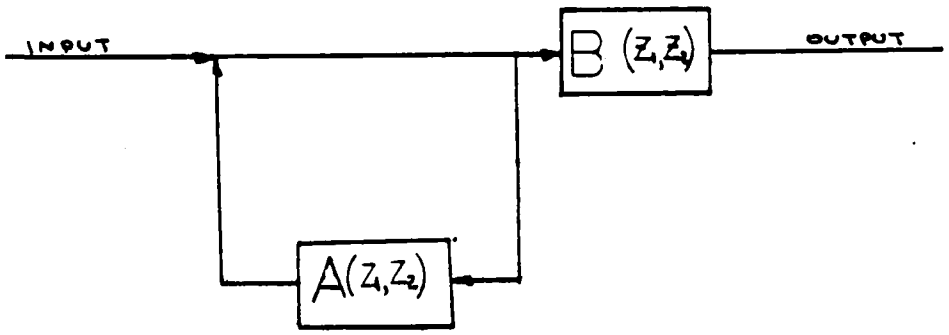


Figure 4-3

Then, when we substitute Figures 2-2 and 2-3 for the blocks as before, the two z_1^{-1} chains will contain the same data and can be merged to yield the signal flowgraph in Figure 4-4. This glowgraph has a total of four shift operators, and it minimizes the number of z_1^{-1} operators.

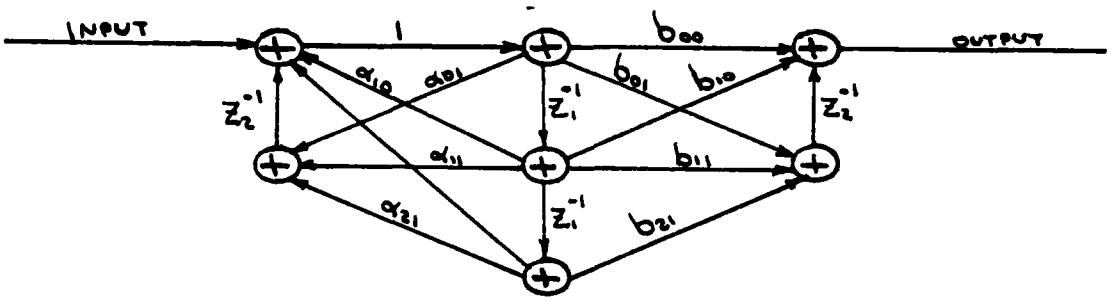


Figure 4-4

Another signal flowgraph that minimizes the number of z_1^{-1} operators may be obtained from Figure 4-4 by the 2-D transposition theorem to obtain a transposed network. Like its 1-D counterpart [Ref. 9], the 2-D transposition theorem states that the transposed network, which is obtained by reversing the directions of all the arrows in a signal flowgraph, will have the same system function as the original network. If we reverse the direction of all the arrows in Figure 4-4 and then redraw the flow graph with the input port on the left and the output port on the right, we get the flowgraph shown in Figure 4-5.

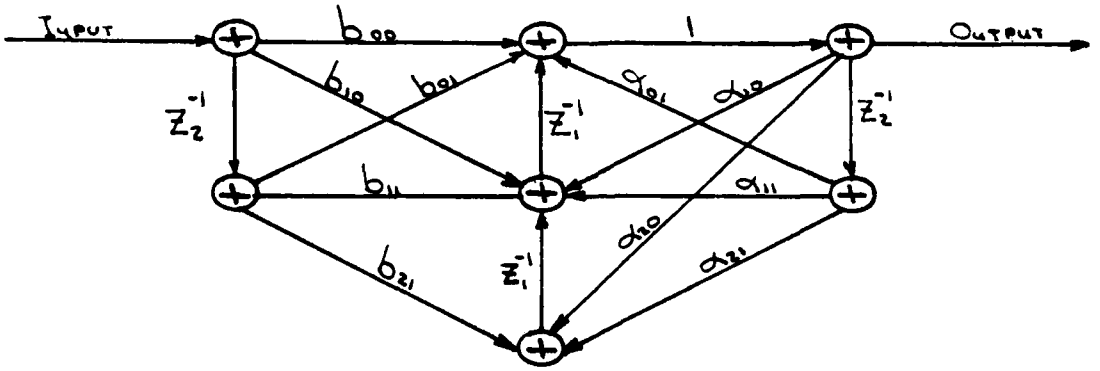


Figure 4-5

This transposed flowgraph may be preferred in implementations with limited wordlengths since the attenuation due to the "zeros" of $H(z_1,z_2)$ occurs before the gain due to the "poles" thus lessening somewhat the possibility of arithmetic overflow in the intermediate computations.

Using the notion of transposition at both the flowgraph level and the block diagram level (note that Figure 2-2 is the transpose of Figure 2-1) the flowgraph can be manipulated to yield a realization that minimizes the total number of shift operators.

As we saw earlier, however, a z_2^{-1} operator will require substantially more storage than a z_1^{-1} operator for a row-by-row ordering of input samples. Consequently, it may be more economical to minimize not the total number of shift operators (as in the 1-D case) but the number of z_2^{-1} operators.

If the filter is realized by using a separate microprocessor to compute samples of each node signal, storage may be less of an issue.

In this case, we may want to minimize the total number of nodes in a flowgraph in order to reduce the number of microprocessors in an implementation.

As digital technology progresses, the relative costs of storage, computation, and interconnectivity keep changing. In the future digital systems designers may have radically different criteria for optimizing a filter realization.

B. A SECOND ORDER MODEL

Looking at the flowgraph in Figure 4-5 and developing a state variable implementation from it, we shall call the output of the top z_1^{-1} operator $R_1(i,j)$, the output of the lower z_1^{-1} operator $R_2(i,j)$, the output of the left z_2^{-1} operator $S_1(i,j)$ and the output of the right z_2^{-1} operator $S_2(i,j)$ as indicated:

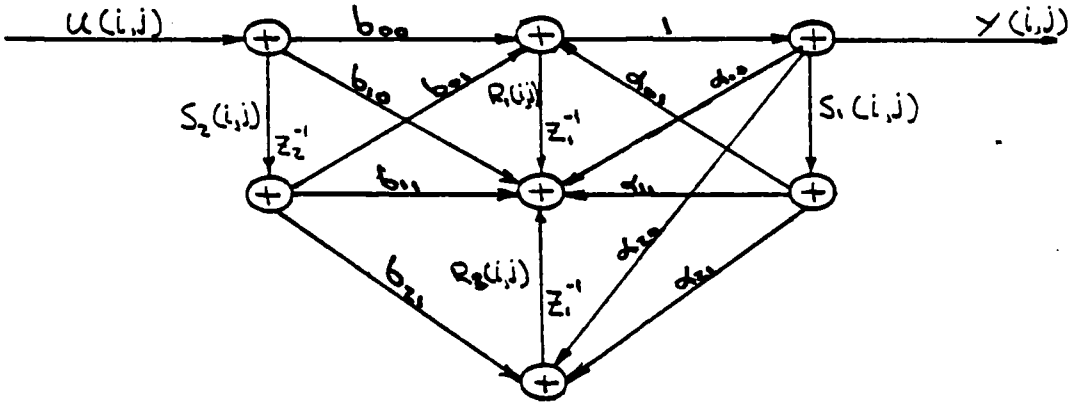


Figure 4-6

$$H(z_1,z_2) = \frac{b_{00}+b_{10}z_1^{-1}+b_{01}z_2^{-1}+b_{11}z_1^{-1}z_2^{-1}+b_{21}z_1^{-2}z_2^{-1}}{1-a_{10}z_1^{-1}-a_{01}z_1^{-1}-a_{11}z_1^{-1}z_2^{-1}-a_{20}z_1^{-2}-a_{21}z_1^{-2}z_2^{-1}}$$

$$\begin{bmatrix} R_1(i+1,j) \\ R_2(i+1,j) \\ S_1(i,j+1) \\ S_2(i,j+1) \end{bmatrix} = \begin{bmatrix} a_{10} & 1 & b_{11}+b_{01}a_{10} & a_{11}+a_{01}a_{10} \\ a_{20} & 0 & b_{21}+b_{01}a_{10} & a_{21}+a_{01}a_{20} \\ 0 & 0 & 0 & 0 \\ 1 & 0 & b_{01} & a_{01} \end{bmatrix} \begin{bmatrix} R_1(i,j) \\ R_2(i,j) \\ S_1(i,j) \\ S_2(i,j) \end{bmatrix}$$

$$+ \begin{bmatrix} b_{10}+b_{00}a_{10} \\ b_{00}a_{20} \\ 1 \\ b_{00} \end{bmatrix} u(i,j) \qquad \text{(IV.2)}$$

$$Y(i,j) = [1 \quad 0 \quad b_{01} \quad a_{01}] \begin{bmatrix} R_1(i,j) \\ R_2(i,j) \\ S_1(i,j) \\ S_2(i,j) \end{bmatrix} + [b_{00}]u(i,j) \qquad \text{(IV.3)}$$

Defining

$$\tilde{b}_{11} = b_{11} + a_{10}b_{01}$$

$$\tilde{a}_{11} = a_{11} + a_{10}a_{01}$$

$$\tilde{b}_{21} = b_{21} + a_{20}b_{01}$$

$$\tilde{a}_{21} = a_{21} + a_{20}a_{01}$$

In general the foregoing equations can be written as:

$$\tilde{b}_{ij} = b_{ij} + a_{i0}b_{0j}$$

$$\tilde{a}_{ij} = a_{ij} + a_{i0}a_{0j}$$

Now we can give an expanded version of (IV-2):

$$R_1(i+1,j) = a_{10}R_1(i,j) + R_2(i,j) + (b_{11}+b_{01}a_{10})s_1'(i,j) + (a_{11}+a_{01}a_{10})S_1^2(i,j) + (b_{10}+b_{00}a_{10})u(i,j)$$

$$R_2(i+1,j) = a_{20}R_1(i,j) + 0 + (b_{21}+b_{01}a_{20})S_1'(i,j) + (a_{11}+a_{01}a_{20})S_1^2(i,j) + (b_{00}a_{20})\ u(i,j) \qquad \text{(IV-4)}$$

$$S_1'(i,j+1) = 0 \quad 0 \quad 0 \quad 0 + lu(i,j)$$

$$S_1^2(i,j+1) = R_1(i,j) + 0 \quad b_{01}S_1'(i,j) + a_{01}S_1^2(i,j) + b_{00}u(i,j)$$

$$Y(i,j) = R(i,j) + 0 + b_{01}S_1'(i,j) + a_{01}S_1^2(i,j) + [b_{00}]u(i,j)$$

$$\tilde{a}_{ij} = a_{ij}+a_{i0}a_{0j} \qquad \text{(IV-5)}$$

$$\tilde{b}_{ij} = b_{ij}+a_{i0}b_{0j}$$

Equations (IV.2) and (IV.3) represent an algorithm for computing the samples of the output signal from the samples of the input signal. Just as in the preceding subsection, the amount of memory required to store the state variables depends on the order in which the output samples are to be computed. It is possible to envision a multiprocessor architecture for computing equation (IV.4) by assigning each processor the responsibility of computing the next value of a particular state variable given the current input value and the current state-variable values. Equation (IV.3) could be implemented by a filter microprocessor to generate the desired output signal values.

In such an architecture, minimization of the number of microprocessors corresponds to the minimization of the number of state variables, a problem studied thoroughly in the literature. Other state-variable forms with the same number of state variables can also be found that will realize the same system function $H(z_1,z_2)$ and may exhibit lower coefficients of sensitivity or round-off noise [Refs. 2,10].

For the special case of "all-pole" 2-D IIR filters, that is, filters with a system function of the form:

$$H(z_1,z_2) = \frac{b_{00}}{A(z_1,z_2)}$$

where b_{00} is a constant and $A(z_1,z_2)$ is a 2-D polynomial, it can be shown that state variable realizations based on signal

flowgraphs, using the output of the shift operators as the state variables, require the minimum number of state variables. They are minimal realizations [Ref. 2].

From the above equations corresponding to the second order Roesser model, the program in Appendix C, was written. This program uses the values of coefficients of $H(z_1,z_2)$ as inputs and it generates an output, y(i,j). Next, the program finds the 2-D Fourier transform of this output matrix, and compares it to the transfer function $H(\omega_1,\omega_2)$.

Numerical Example

In the following three examples (first and second orders), we use the coefficients of first and second order transfer functions. We consider the special case of "all-pole" 2-D IIR filters, i.e., filters with a transfer function of the form:

$$H(z_1,z_2) = \frac{b_{00}}{A(z_1,z_2)} = \frac{1}{A(z_1,z_2)}$$

where b_{00} is constant (unity in our case) and $A(z_1,z_2)$ is a 2-D polynomial. It can be shown that state variable realizations based on signal flowgraphs, using the output of the shift operators as the state variables, require the minimum number of state variables. They are minimal realizations [Ref. 2].

For the third program we have a graph for the case of a BP filter.

Example #4

$$H(z_1,z_2) = \frac{1}{1-0.2z_1^{-1}-0.5z_2^{-1}-0.1z_1^{-1}z_2^{-1}} \qquad \text{(IV.6)}$$

The $|Y(m,n)|$ for this example is plotted in Fig. 4-1a. The corresponding contour map of 2-D surface is shown in Figure 4-1b.

Example #5

$$H(z_1,z_2) = \frac{1}{1-0.25z_1^{-1}-0.345z_2^{-1}-0.125z_1^{-1}z_2^{-1}-0.1z_1^{-2}z_2^{-1}} \quad \text{(IV.7)}$$

The 2-D D.F.T. $|y(m,n)|$ of the output of this filter is shown in Fig. 4-2a. The corresponding contour map is shown in Fig. 4-2b.

Example #6

$$H(z_1,z_2) = \frac{-0.125+0.25z_1^{-1}+0.125z_2^{-1}-0.125z_1^{-1}z_2^{-1}+0.125z_1^{-2}z_2^{-1}}{1 + z_1^{-1}z_2^{-1}} \quad \text{(IV.8)}$$

$|Y(m,n)|$ for this examle and the corresponding contour map are shown in Figs. 4-3a and 4-3b, respectively.

For reasons of verification, as before, $Y(m,n)$ was compared to the actual transfer function $H(\omega_1,\omega_2)$ for examples 4,5,6. These transfer functions plots and the corresponding contour maps are shown in Fig. 4-4a,b, Fig. 4-5a,b and Fig. 4-6a,b for the examples 4,5,6, respectively.

C. EXTENSION OF THE 2-D STATE SPACE MODELS TO HIGHER ORDER TRANSFER FUNCTIONS

1. Introduction

During recent years, several authors (Attasi [Ref. 11, Fozmasimi and Mazchesini [Ref. 13], Givone and Roesser [Ref.

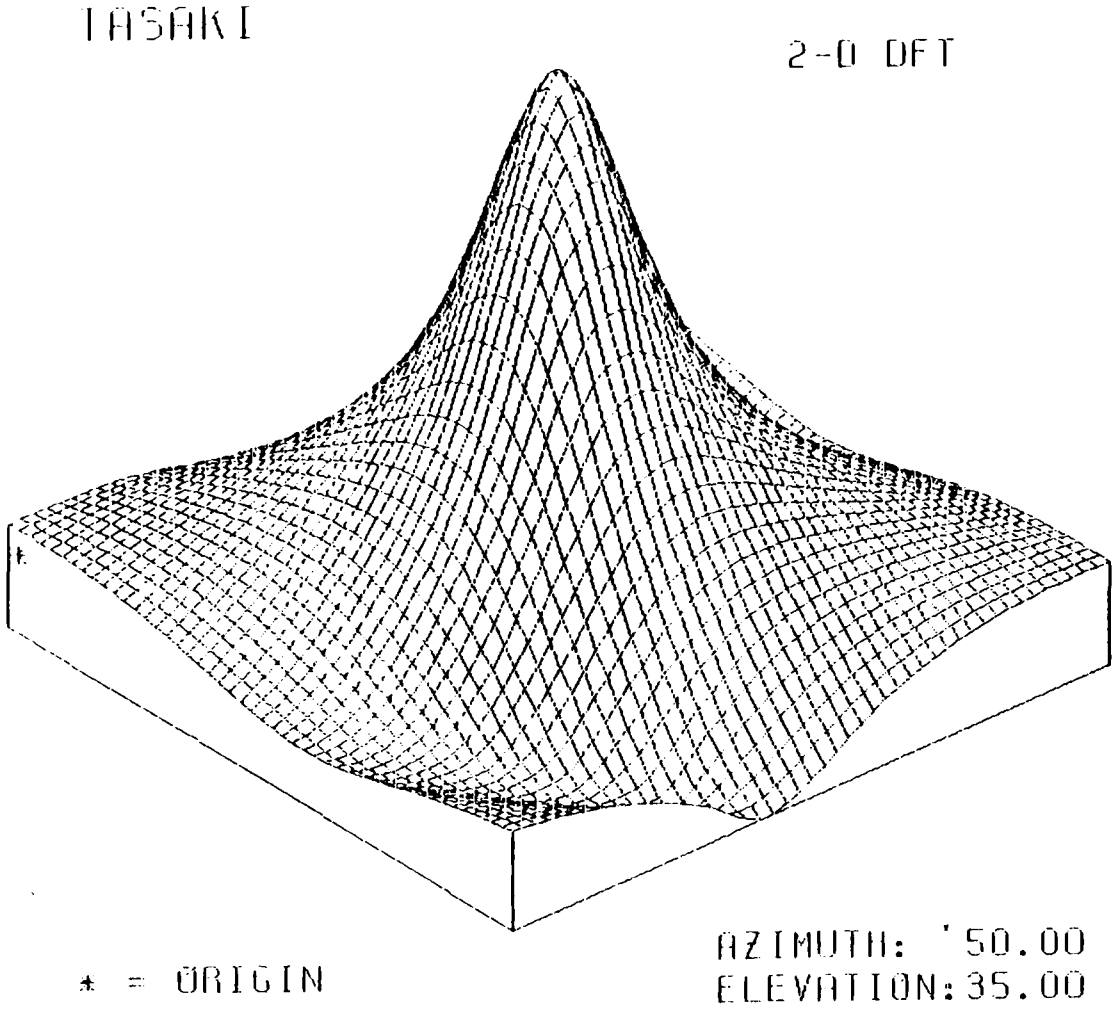


Figure 4-1a. 2-D D.F.T. Sequences, Y(m,n) for Example 4

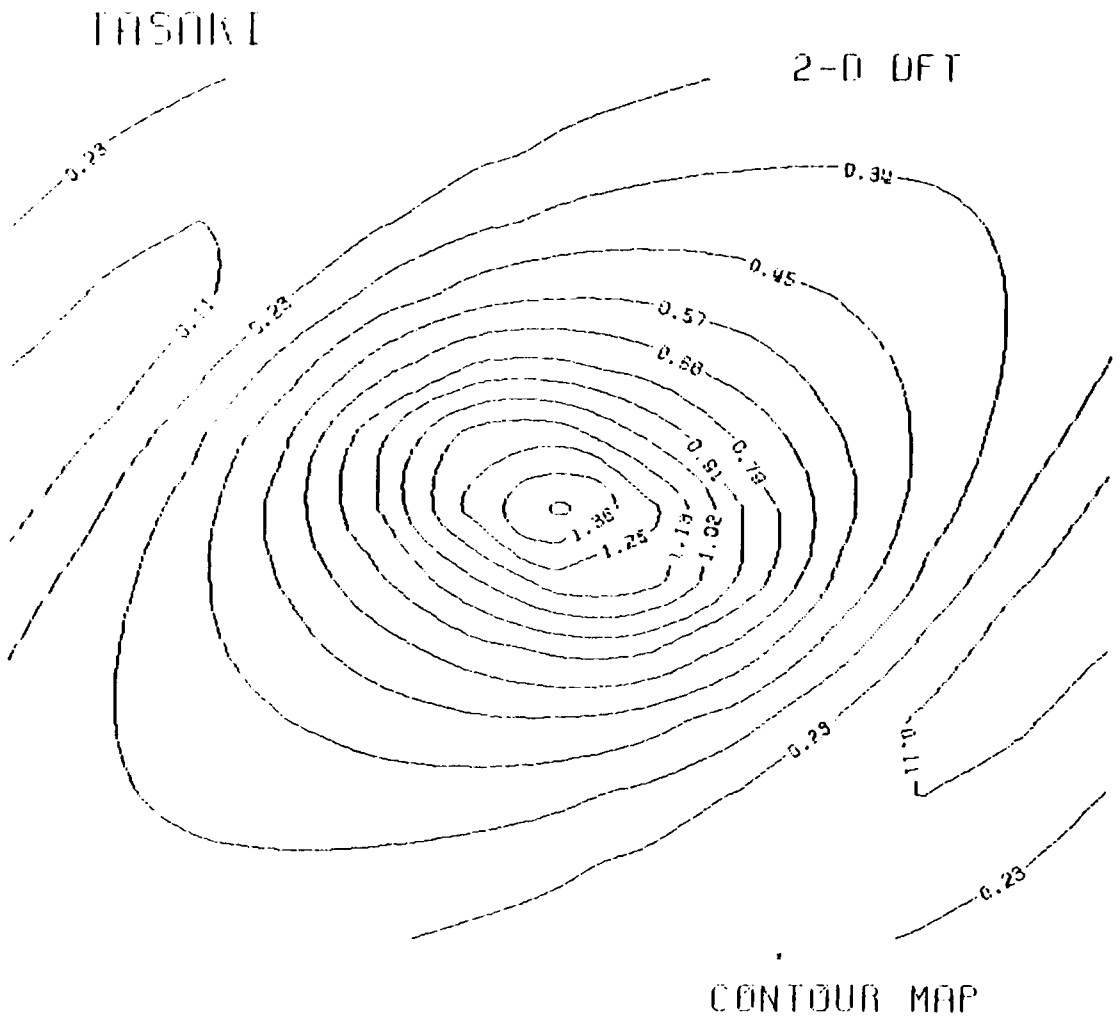


Figure 4-1b. Contour Map for Figure 4-1a

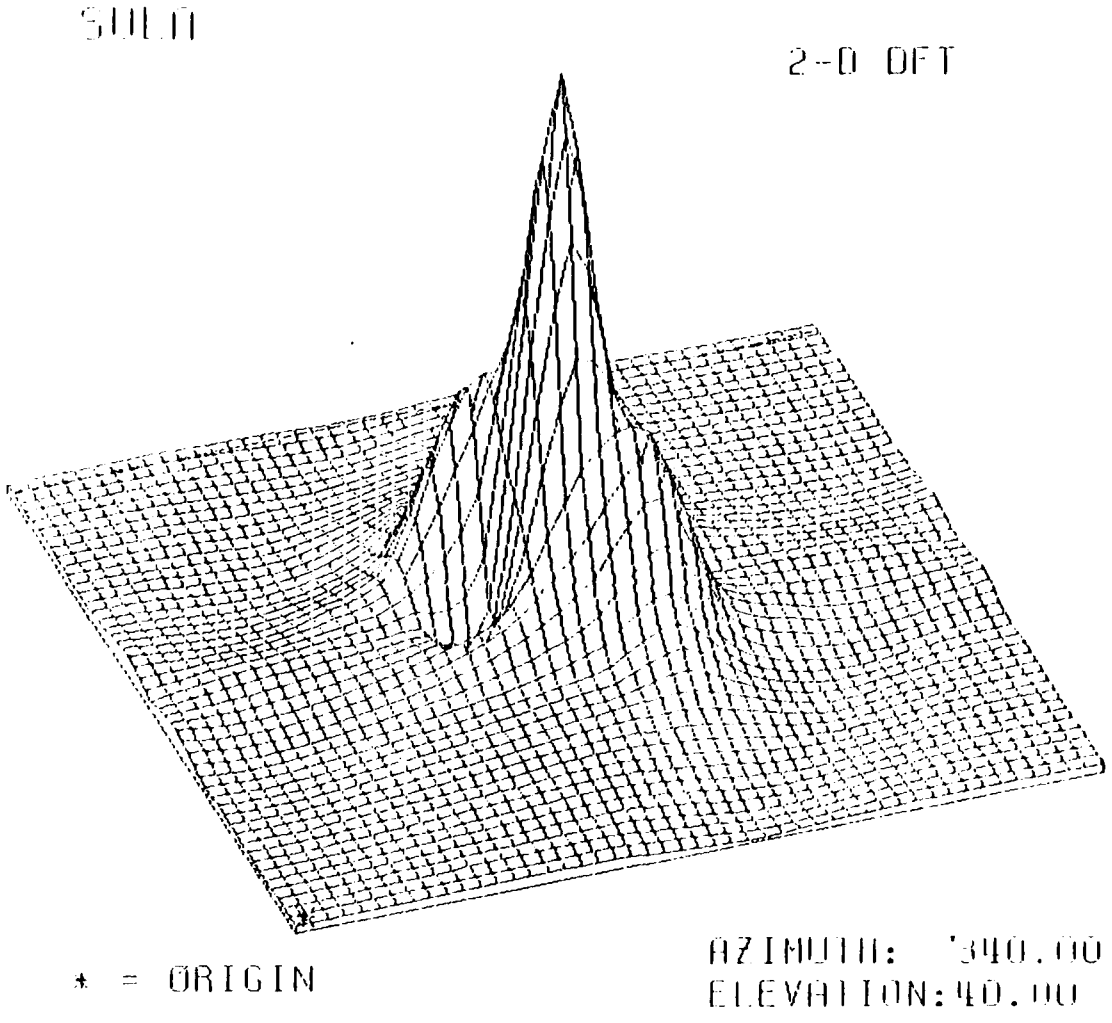


Figure 4-2a. 2-D D.F.T. Sequences Y(m,n) for Example 5

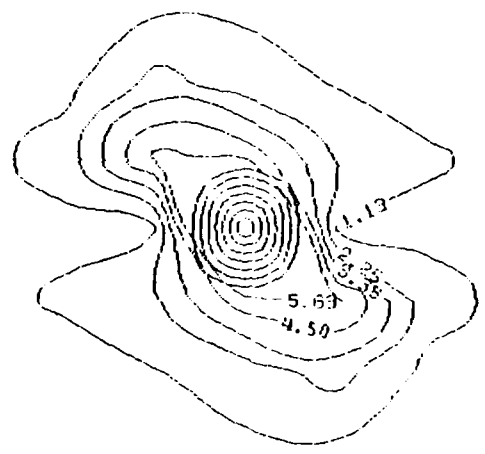


Figure 4-2b. Contour Map for Figure 4-2a

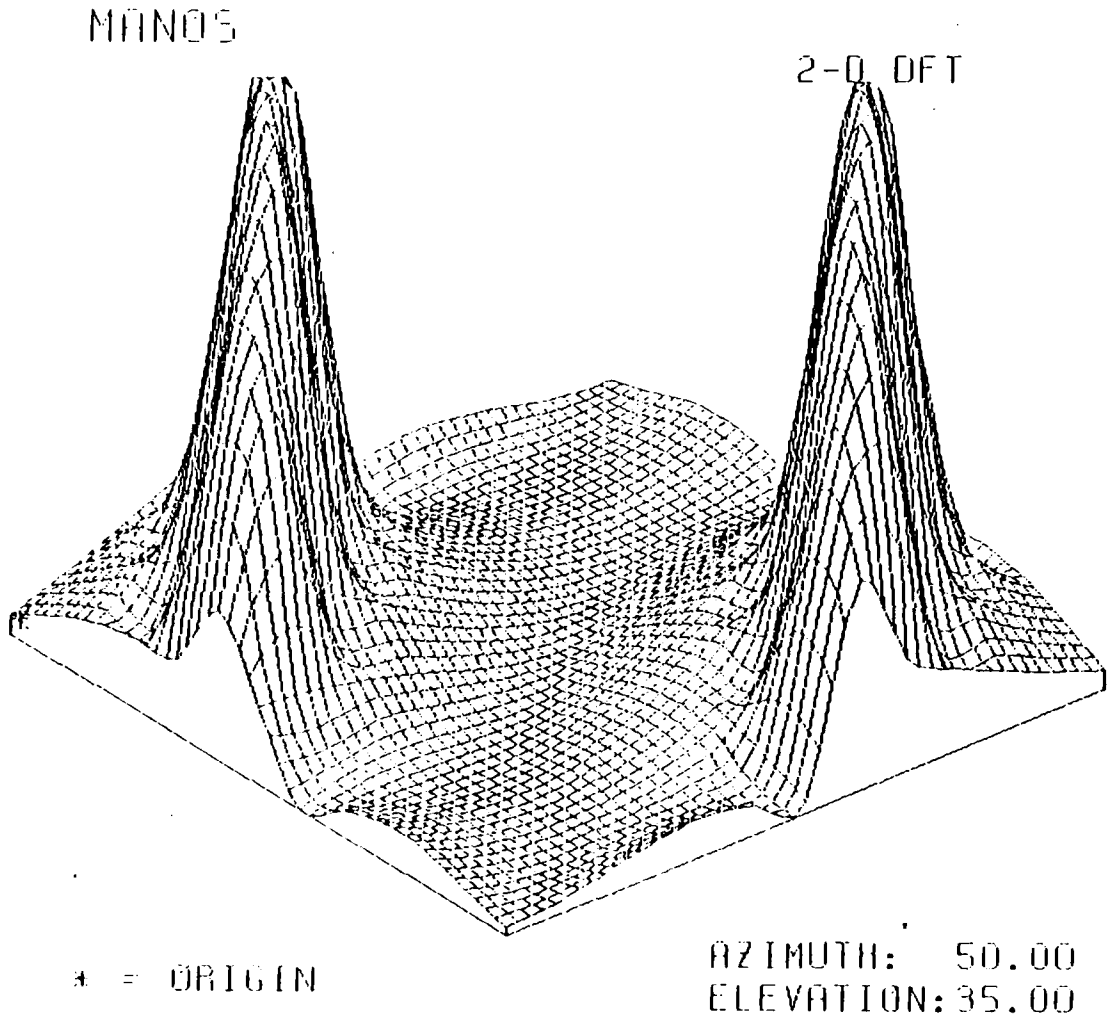


Figure 4-3a. 2-D D.F.T. Squences, Y(m,n) for Example 6

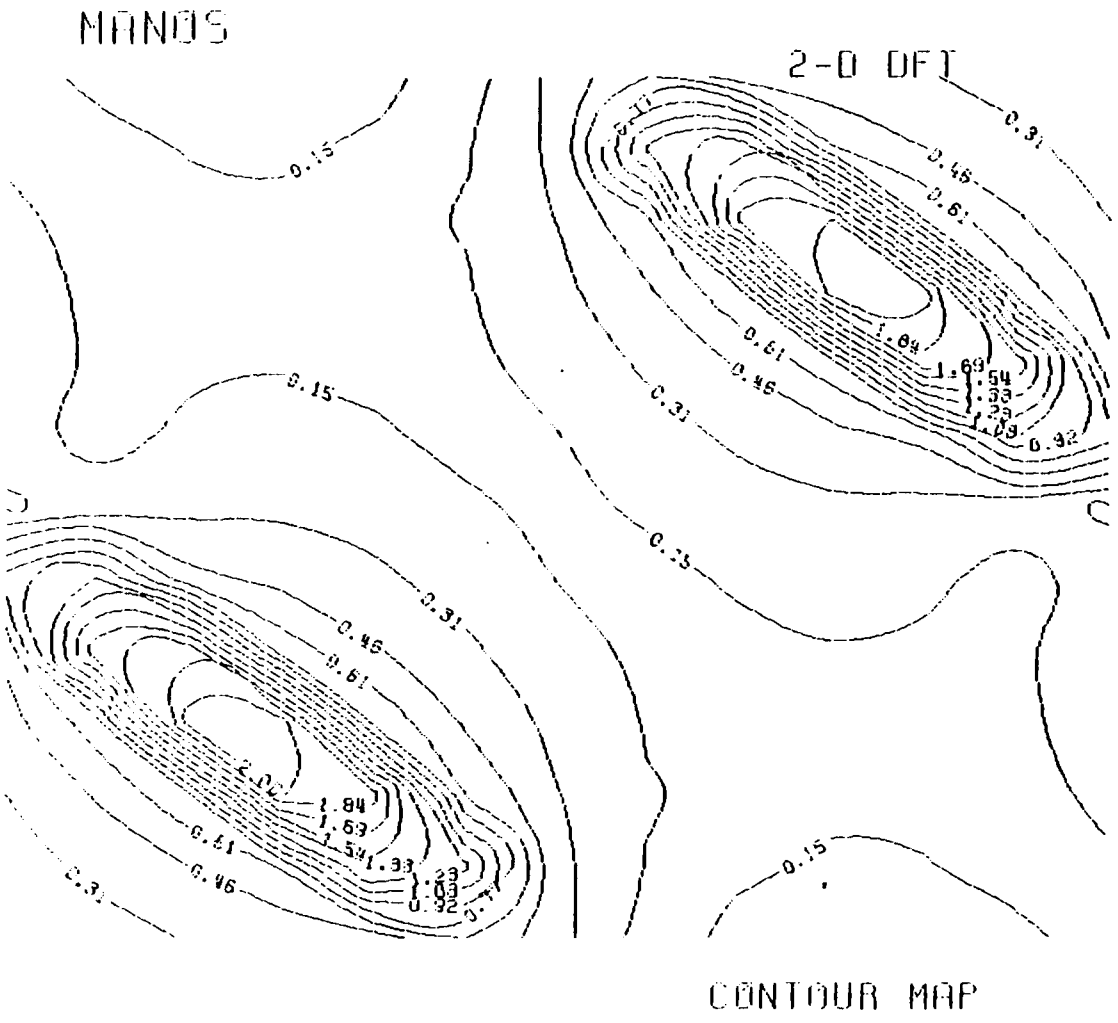


Figure 4-3b. Contour Map for Figure 4-3a

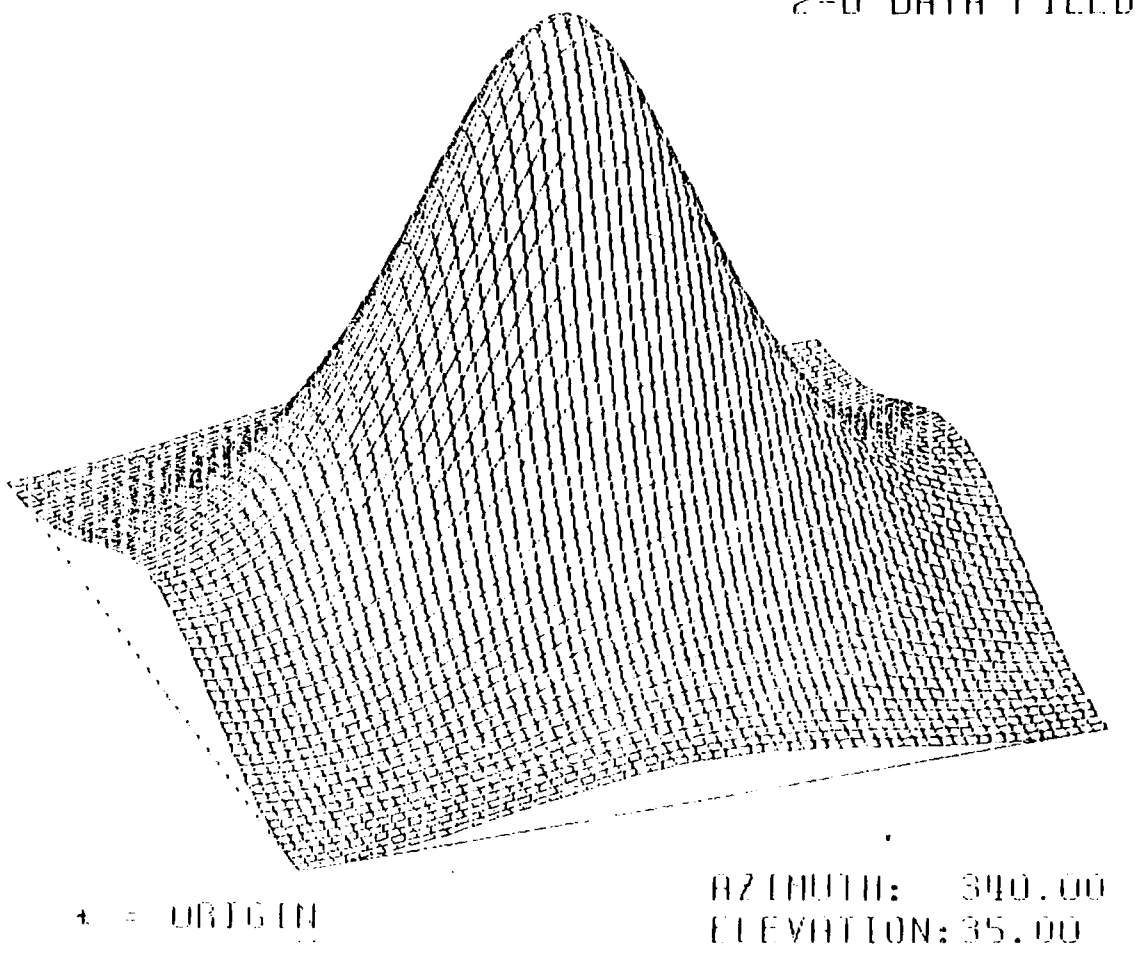


Figure 4-4a. Transfer Function $|H(z_1,z_2)|$, $z_1 = e^{j\omega_1}$, $z_2 = e^{j\omega_2}$ for Example 4

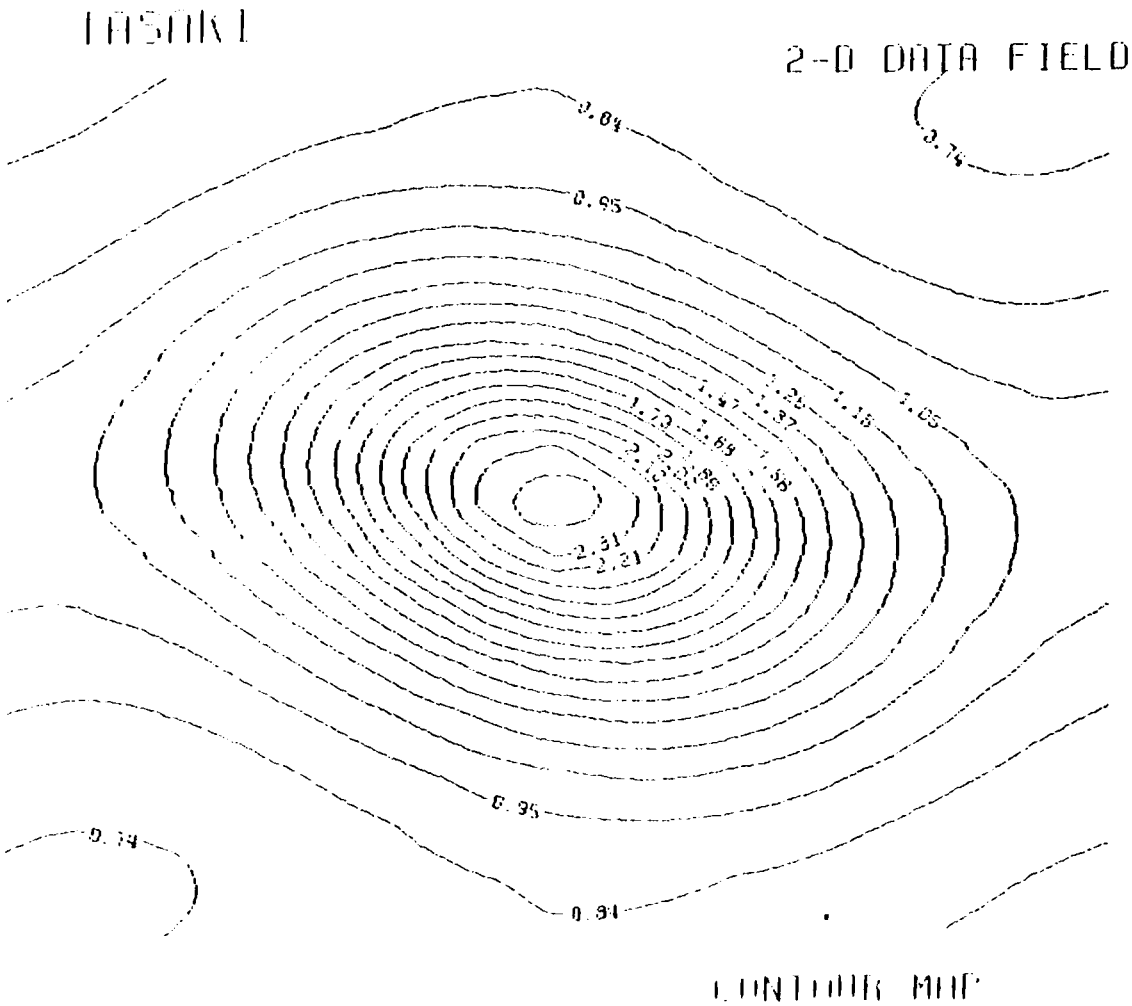


Figure 4-4b. Contour Map for Figure 4-4a

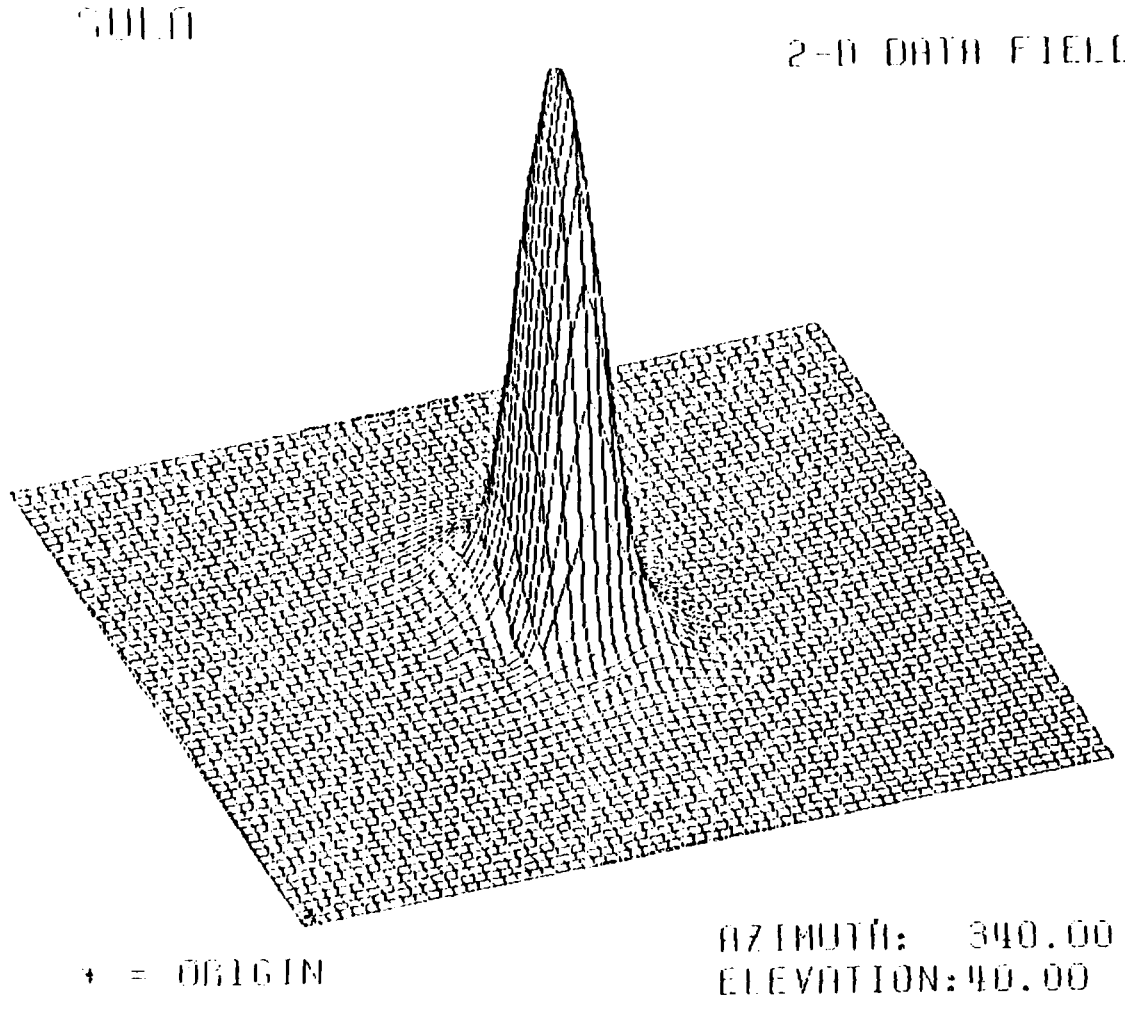


Figure 4-5a. Transfer Function $|H(z_1,z_2)|$, $z_1 = e^{j\omega_1}$, $z_2 = e^{j\omega_2}$ for Example 5

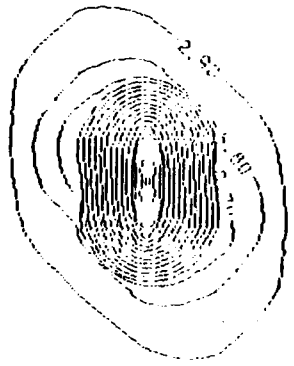


Figure 4-5b. Contour Map for Figure 4-5a

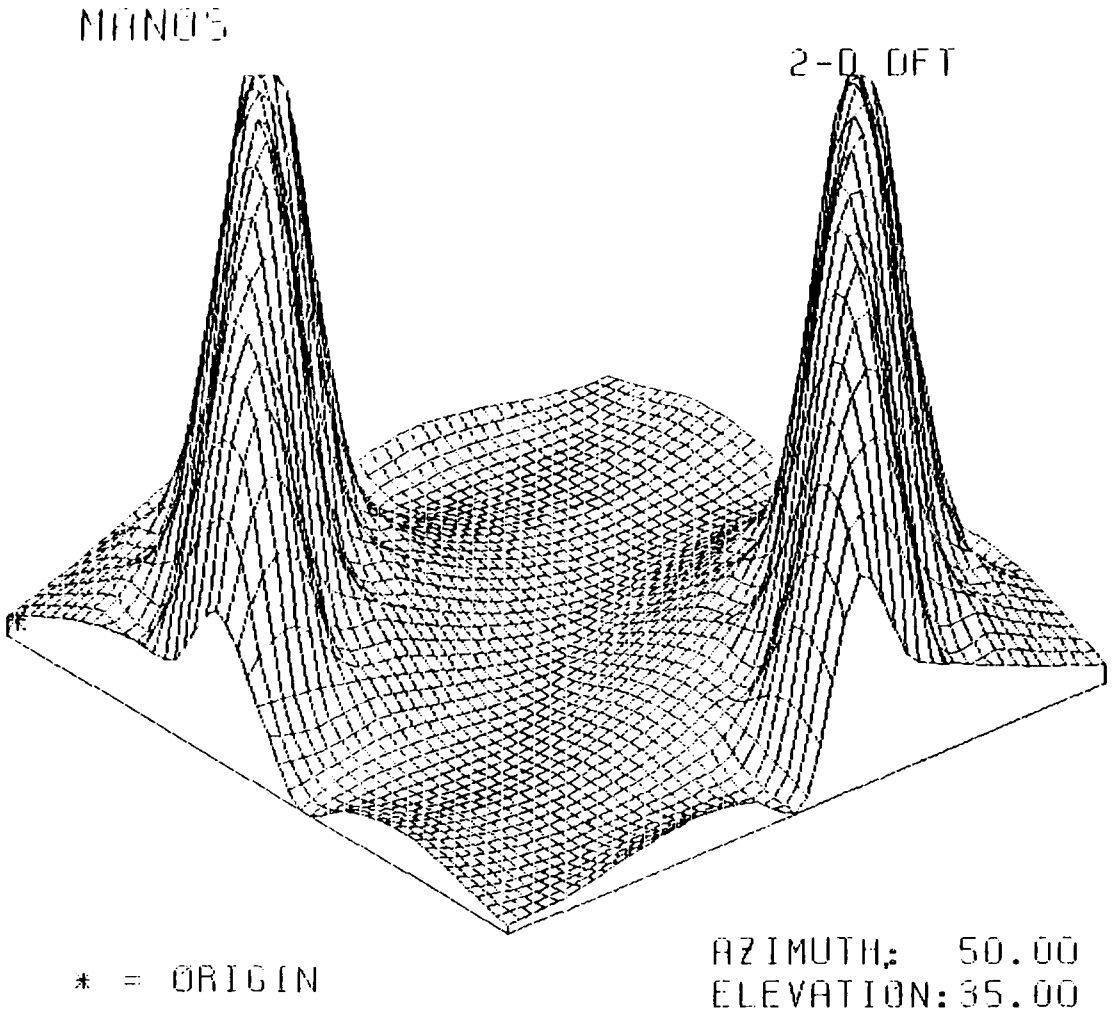


Figure 4-6a. Transfer Function $|H(z_1,z_2)|$, $z_1 = e^{j\omega_1}$, $z_2 = e^{j\omega_2}$ for Example 6

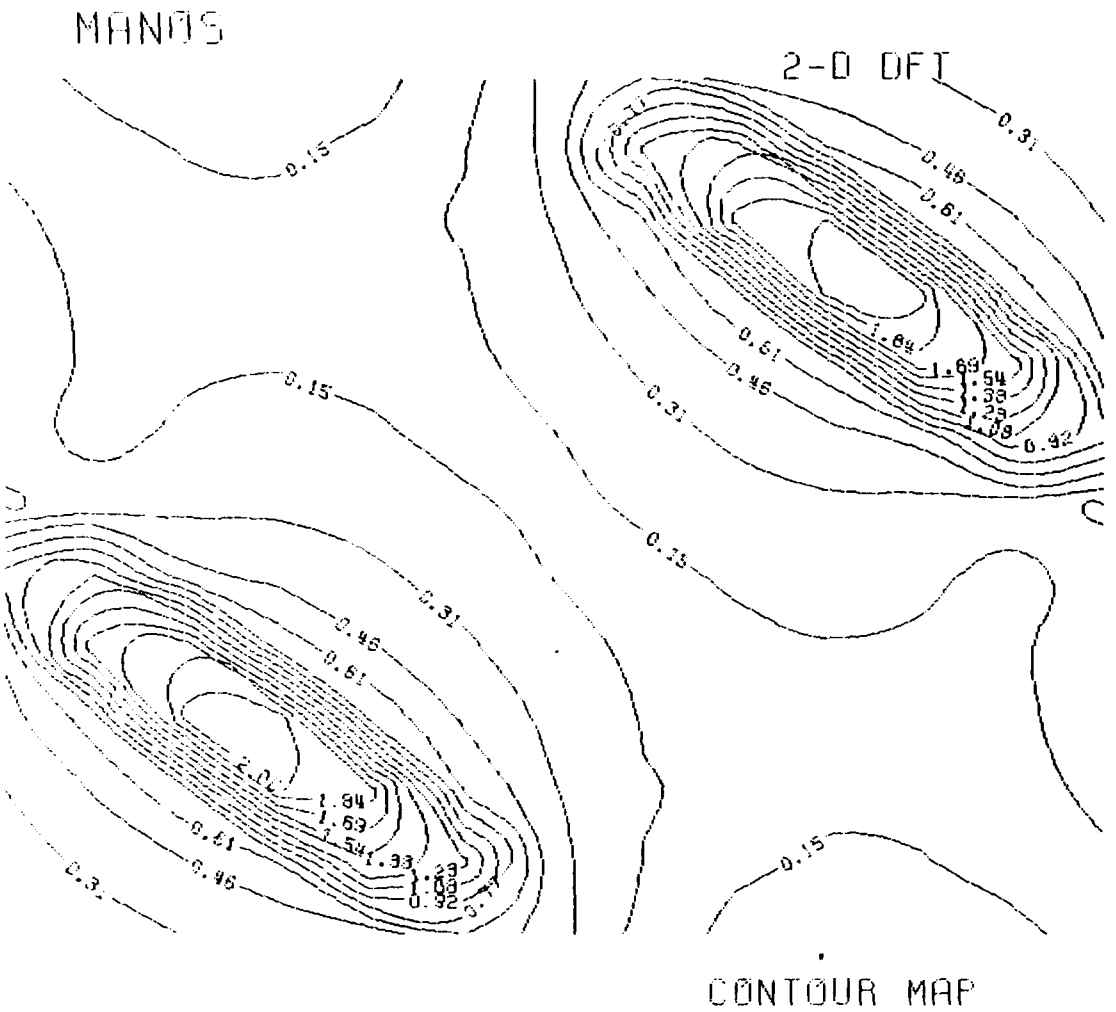


Figure 4-6b. Contour Map for Figure 4-6a

17]) have proposed different state space models for 2-D systems. They have also suggested some extensions of the usual 1-D notions of controllability, observability, and minimality to the 2-D case.

However, these results are not quite satisfactory. They either lack motivation for the state-space models introduced or the notion of state-space is improperly defined. In Chapter II we started with a comparison of all the current models based on a practical (circuit-oriented) point of view and on a proper definition of state. It is shown that the model of Roesser is the most satisfactory, in that it is also the most general since the Attasi and Fozmasimi Mazchesimi models can be imbedded in the Givone and Roesser model.

In Chapter II we pointed out that a major difference between 1-D and 2-D systems is that in the 2-D case a global state (which preserves all past information) and a local state (which gives us the size of the recursions of the 2-D filter) can be introduced.

2. Extension for 2-D Systems

In [Ref. 14], Fozmasimi and Mazchesimi use the algebraic point of view of "Nerode" equivalence. In this framework, the state space arises from the factorization of the 2-D input/output map. Fozmasimi and Mazchesini were the first to realize that a major difference between 1-D and 2-D systems is that we can introduce a global state and a local state in the 2-D case.

The global state (which is of infinite dimensions, in general) preserves all the past information while the local

state gives us the size of the recursions to be performed at each step by the 2-D filter. However, Fozmasimi and Mazchesini failed to exploit fully the structure of the global state and its relation to the local state, so that the state space model they introduced is unsatisfactory in the sense that what they introduce as the state is really only a "partial state" (as defined by Wololich [Ref. 15] for 1-D systems). Indeed, this partial state does not obey a first-order difference equation (the notion of first order difference equation for linear systems or partially ordered sets has been defined by Mullans and Elliot in [Ref. 16]). Attasi's model suffers from the same drawback as the Fozmasimi and Mazchesini one.

On the other hand, Givone and Roesser [Refs. 17,18,1] have used a "circuit approach" to the problem of state space realization for 2-D systems. They present a model in which the local state is divided into a horizontal and a vertical state which are propagated, respectively, horizontally and vertically by first-order difference equations. From this point of view the global state appears as the boundary condition necessary to propagate the state-space equations.

However, Roesser did not provide much motivation for the introduction of such a model and seemed unaware of the full circuit interpretation of their model since they were not able to implement an arbitrary 2-D transfer function, say

$$H(z_1,z_2) = \frac{b(z_1,z_2)}{a(z_1,z_2)}$$

Mitra et al gave an answer in [Ref. 19] by presenting an implementation method for 2-D transfer functions using delay elements z_1^{-1} and z_2^{-1}. We shall see below that this approach is consistent with Roesser's model. It is shown in [Ref. 8] that Roesser's model appears naturally as a way to describe the local state properties. For a (n,m) 2-D transfer function,

$$H(z_1,z_2) = \frac{b(z_1,z_2)}{a(z_1,z_2)} = \frac{\sum_{i=0}^{n}\sum_{j=0}^{m} b_{ij} z_1^{-i} z_2^{-j}}{\sum_{i=0}^{n}\sum_{j=0}^{m} a_{ij} z_1^{-i} z_2^{-j}} \qquad \text{(IV.9)}$$

exhibits some canonical state-space forms (controllability, observability), which can also be written as,

$$H(z_1,z_2) = \frac{\sum_{i=0}^{n} b_i(z_2^{-1}) z_1^{-i}}{\sum_{i=0}^{n} a_i(z_2^{-1}) z_1^{-i}} \qquad \text{(IV.10)}$$

Without loss of generality, we can assume $a_{00} = 1$ and we denote

$$\bar{a}_0(z_2^{-1}) = 1 + a_0(z_2^{-1})$$

Thus, using 1-D realization technique, $H(z_1,z_2)$ of Eq. (IV.10) can be used as shown below in Fig. 4-7.

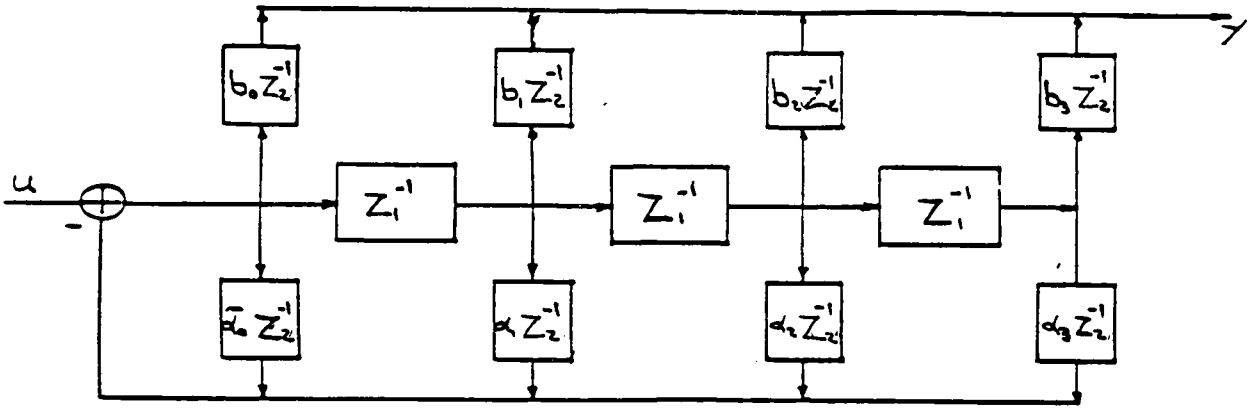


Figure 4-7

The realization is almost achieved: in addition to the n-horizontal delay elements, we need only m vertical delay elements to implement the feedback gains $\{a_i(z_2^{-1}),\ i = 0,1,\ldots,m\}$ and m other vertical delay elements to implement the readout gains $\{b_i(z_2^{-1}),\ i = 0,1,\ldots,m\}$. Thus the complete realization shown in Fig. 4-8 requires only n+2m dynamic elements. This realization is a standard (canonical) one; its structure is very simple and it involves only real gains. Note also that we need fewer dynamic elements than was suggested by the implementations of [Ref. 19].

As mentioned in Section (b), circuit implementations with delay elements z_1^{-1} and z_2^{-1} are in a one-to-one correspondence with state-space models of Roesser's type. The outputs of the z_1^{-1} delays are the horizontal states and the outputs of the z_2^{-1} delays are the vertical states. Thus the implementation of the following figure can be transformed readily into the following state-space model.

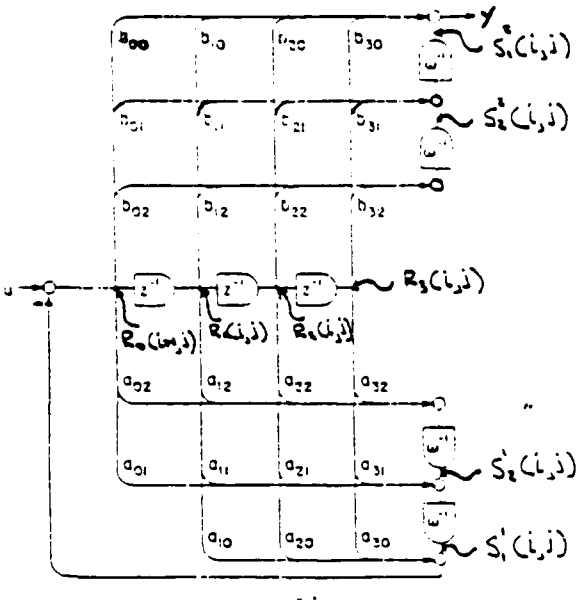


Figure 4-8

$$\begin{bmatrix} R(i+1,j) \\ S^1(i,j+1) \\ S^2(i,j+1) \end{bmatrix} = A \begin{bmatrix} R(i,j) \\ S^1(i,j) \\ S^2(i,j) \end{bmatrix} + bu(i,j) \qquad \text{(IV.11)}$$

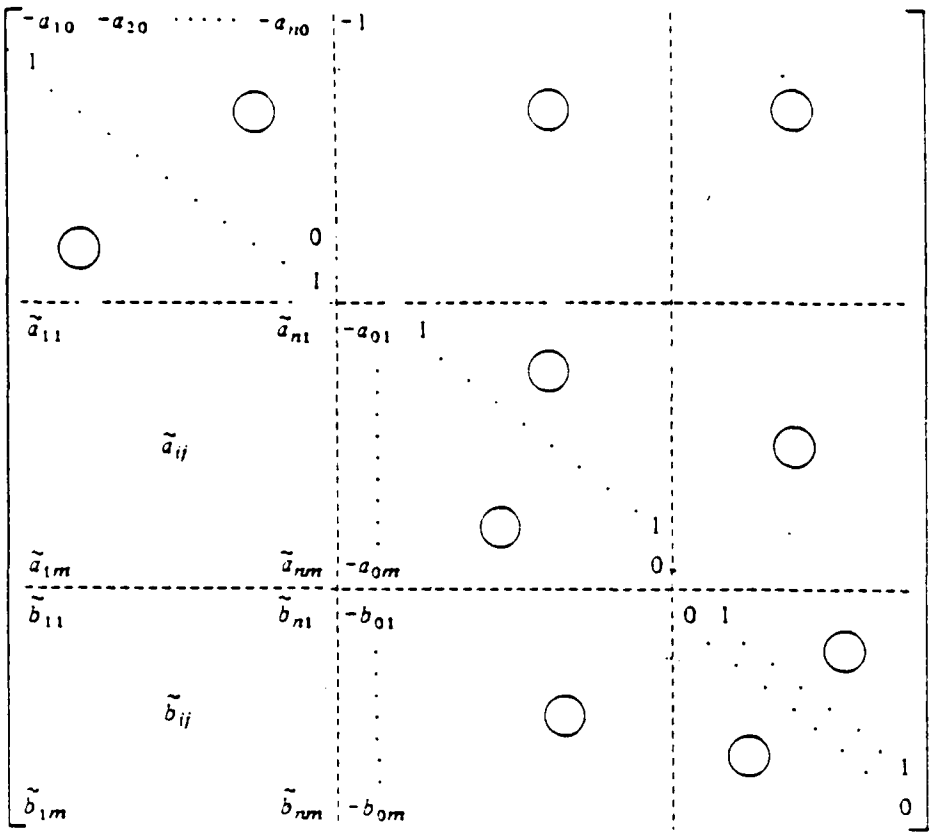
$$y(i,j) = C \begin{bmatrix} R(i,j) \\ S(i,j) \end{bmatrix}$$

where:

$$C = [b_{10} \cdots b_{n0} \quad -b_{00} \quad 0 \cdots 0 \quad 1 \quad 0 \quad \cdots \quad 0]$$

$$b^T = [1 \quad 0 \cdots 0 \quad a_{01} \cdots a_{0m} \quad b_{01} \cdots b_{0m}] \quad \text{(input vector)}$$

Transition Matrix $A =$

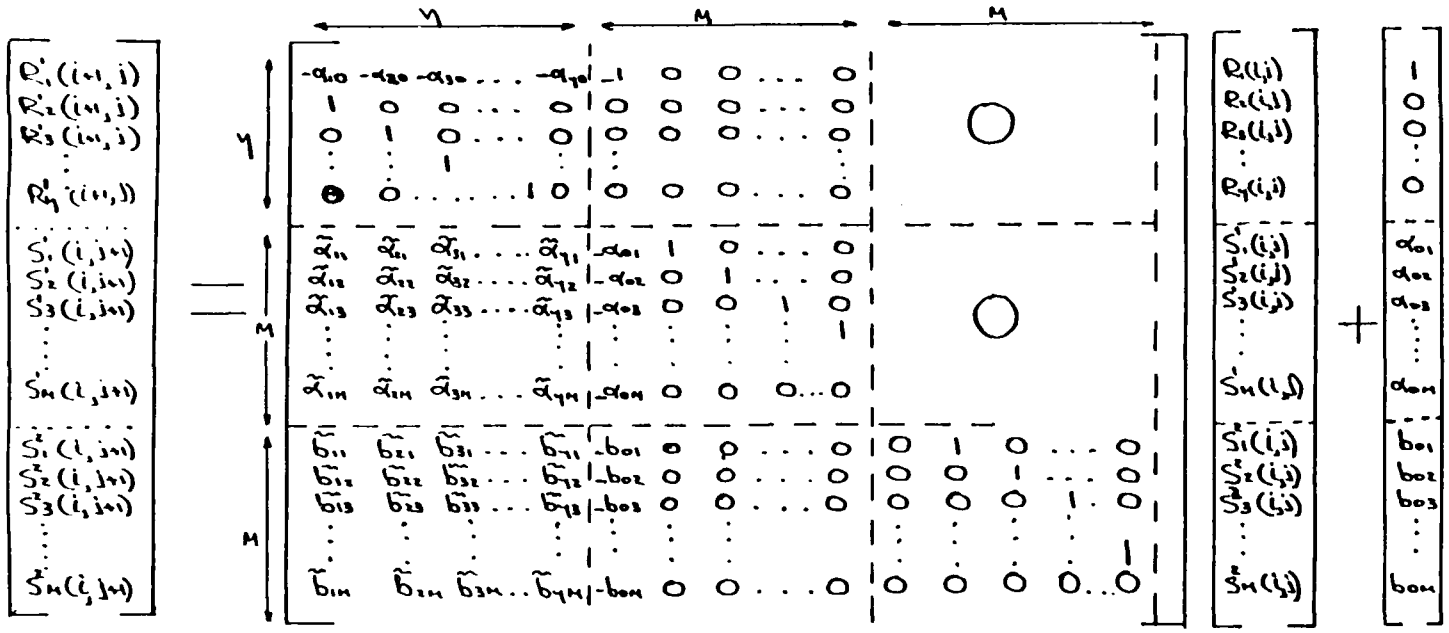


with: $a_{ij} = a_{ij} - a_{i0}a_{0j}$ $\quad b_{ij} = b_{ij} - a_{i0}b_{0j}$

$1 \leq i \leq n \quad 1 \leq j \leq m \qquad 1 \leq i \leq n \quad 0 \leq j \leq m$

The expanded form of Eq. IV-11 can now be shown as:

$$\begin{bmatrix} R(i+1,j) \\ S_1(i,j+1) \\ S_2(i,j+1) \end{bmatrix} = A \begin{bmatrix} R(i,j) \\ S_1(i,j) \\ S_2(i,j) \end{bmatrix} + bu(i,j) \qquad y(i,j) = C \begin{bmatrix} R(i,j) \\ S(i,j) \end{bmatrix}$$



$$(N+2M) \times (N+2M) \qquad \text{(IV-12a)}$$

$$Y(i,j) = [\tilde{b}_{10}\ \tilde{b}_{20}\ \tilde{b}_{30}\ \ldots\ \tilde{b}_{n0} \mid -b_{00}\ \ldots\ 0 \mid 1\ 0\ \ldots\ 0] \qquad \begin{matrix} R_1(i,j) \\ R_2(i,j) \\ R_3(i,j) \\ \vdots \\ R_4(i,j) \\ \hline S_1^1(i,j) \\ S_2^1(i,j) \\ S_3^1(i,j) \\ \vdots \\ S_m^1(i,j) \\ \hline S_1^2(i,j) \\ S_2^2(i,j) \\ S_3^2(i,j) \\ \vdots \\ S_m^2(i,j) \end{matrix} \qquad \text{(IV.12b)}$$

(output vector)

D. PROGRAM AND EXAMPLES FOR ROESSER'S EQUATIONS USING KUNG'S MODEL

This program (Appendix D) takes as initial conditions one horizontal state and two vertical states. The order of horizontal states is given by N and the order of the vertical states by M.

We give two examples, one for N = 2 and M = 2 (Example 7) (two orders for horizontal states and 2 orders for vertical states) and one for N = 4 and M = 3 (Example 8) (four orders for horizontal states and three orders for vertical states). The first example is for a matrix 2×2 and the second example 4×4.

N=2 M=2 MATRIX 2X2

$R_1(1,1)$ $R_2(1,1)$	$R_1(1,2)$ $R_2(1,2)$
$S_1^1(1,1)$ $S_2^1(1,1)$	$S_1^1(1,2)$ $S_2^1(1,2)$
$S_1^2(1,1)$ $S_2^2(1,1)$	$S_1^2(1,2)$ $S_2^2(1,2)$
$R_1(2,1)$ $R_2(2,1)$	$R_1(2,2)$ $R_2(2,2)$
$S_1^1(2,1)$ $S_2^1(2,1)$	$S_1^1(2,2)$ $S_2^1(2,2)$
$S_1^2(2,1)$ $S_2^2(2,1)$	$S_1^2(2,2)$ $S_2^2(2,2)$

State variables for example 7

$R_1(1,1), R_2(1,1), R_3(1,1), R_4(1,1)$	$R_1(1,2), R_2(1,2), R_3(1,2), R_4(1,2)$	$R_1(1,3), R_2(1,3), R_3(1,3), R_4(1,3)$	$R_1(1,4), R_2(1,4), R_3(1,4), R_4(1,4)$
$S_1^1(1,1), S_2^1(1,1), S_3^1(1,1)$	$S_1^1(1,2), S_2^1(1,2), S_3^1(1,2)$	$S_1^1(1,3), S_2^1(1,3), S_3^1(1,3)$	$S_1^1(1,4), S_2^1(1,4), S_3^1(1,4)$
$S_1^2(1,1), S_2^2(1,1), S_3^2(1,1)$	$S_1^2(1,2), S_2^2(1,2), S_3^2(1,2)$	$S_1^2(1,3), S_2^2(1,3), S_3^2(1,3)$	$S_1^2(1,4), S_2^2(1,4), S_3^2(1,4)$
$R_1(2,1), R_2(2,1), R_3(2,1), R_4(2,1)$	$R_1(2,2), R_2(2,2), R_3(2,2), R_4(2,2)$	$R_1(2,3), R_2(2,3), R_3(2,3), R_4(2,3)$	$R_1(2,4), R_2(2,4), R_3(2,4), R_4(2,4)$
$S_1^1(2,1), S_2^1(2,1), S_3^1(2,1)$	$S_1^1(2,2), S_2^1(2,2), S_3^1(2,2)$	$S_1^1(2,3), S_2^1(2,3), S_3^1(2,3)$	$S_1^1(2,4), S_2^1(2,4), S_3^1(2,4)$
$S_1^2(2,1), S_2^2(2,1), S_3^2(2,1)$	$S_1^2(2,2), S_2^2(2,2), S_3^2(2,2)$	$S_1^2(2,3), S_2^2(2,3), S_3^2(2,3)$	$S_1^2(2,4), S_2^2(2,4), S_3^2(2,4)$
$R_1(3,1), R_2(3,1), R_3(3,1), R_4(3,1)$	$R_1(3,2), R_2(3,2), R_3(3,2), R_4(3,2)$	$R_1(3,3), R_2(3,3), R_3(3,3), R_4(3,3)$	$R_1(3,4), R_2(3,4), R_3(3,4), R_4(3,4)$
$S_1^1(3,1), S_2^1(3,1), S_3^1(3,1)$	$S_1^1(3,2), S_2^1(3,2), S_3^1(3,2)$	$S_1^1(3,3), S_2^1(3,3), S_3^1(3,3)$	$S_1^1(3,4), S_2^1(3,4), S_3^1(3,4)$
$S_1^2(3,1), S_2^2(3,1), S_3^2(3,1)$	$S_1^2(3,2), S_2^2(3,2), S_3^2(3,2)$	$S_1^2(3,3), S_2^2(3,3), S_3^2(3,3)$	$S_1^2(3,4), S_2^2(3,4), S_3^2(3,4)$
$R_1(4,1), R_2(4,1), R_3(4,1), R_4(4,1)$	$R_1(4,2), R_2(4,2), R_3(4,2), R_4(4,2)$	$R_1(4,3), R_2(4,3), R_3(4,3), R_4(4,3)$	$R_1(4,4), R_2(4,4), R_3(4,4), R_4(4,4)$
$S_1^1(4,1), S_2^1(4,1), S_3^1(4,1)$	$S_1^1(4,2), S_2^1(4,2), S_3^1(4,2)$	$S_1^1(4,3), S_2^1(4,3), S_3^1(4,3)$	$S_1^1(4,4), S_2^1(4,4), S_3^1(4,4)$
$S_1^2(4,1), S_2^2(4,1), S_3^2(4,1)$	$S_1^2(4,2), S_2^2(4,2), S_3^2(4,2)$	$S_1^2(4,3), S_2^2(4,3), S_3^2(4,3)$	$S_1^2(4,4), S_2^2(4,4), S_3^2(4,4)$

$N = 4$ $\quad M = 3$ $\quad$ MATRIX 4×4

State variables for example 8

E. NUMERICAL EXAMPLES FOR KUNG'S MODEL

The following presents three examples. The first one corresponds to an "all-pole" 2-D low-pass filter. The second one is an "all-zero" 2-D band-pass filter ($\sin \omega_1 \sin \omega_2$). The third one is also a band-pass filter. All these examples are second order. The outputs of these examples are produced using Kung's [Ref. 8] state-space model. In this formulation, for a second order system, we require two horizontal states--R1(i,j) and R2(i,j) and four vertical states, S1(1)(i,j), S1(2)(i,j), S2(1)(i,j) and S2(2)(i,j). The program listing for implementing this model is given in Appendix D.

Example #9

The system parameters and the initial conditions chosen for this example are as listed in Table 4.1. The 2-D D.F.T. |Y(m,n)| of the output sequence y(i,j) produced by the program in Appendix D is shown in Fig. 4-9a. The corresponding contour map is shown in Fig. 4-9b.

Example #10

The parameter coefficients and the initial conditions for this example are listed in Table 4.2. The 2-D D.F.T. sequence |Y(m,n)| for this example is illustrated in Fig. 4-10a, and Fig 4.10b shows the associated contour map.

Example #11

The parameter coefficients and the initial conditions for this example are listed in Table 4.3. The 2-D D.F.T. sequence |Y(m,n)| for this example are illustrated in Fig. 4-11a and Figure 4-11b shows the associated contour map.

TABLE 4.1

```
NUMBER OF HORIZONTAL STATES(N=1to4): 2

NUMBER OF VERTICAL STATES(M=1to4): 2

DIMENSION OF OUTPUT(1to25): 15

ENTER INITIAL CONDITIONS FOR HORIZONTAL R(#,#)
R 1(1, 1): 0
R 2(1, 1): 0
R 1(1, 2): 0
R 2(1, 2): 0
R 1(1, 3): 0
R 2(1, 3): 0
R 1(1, 4): 0
R 2(1, 4): 0
R 1(1, 5): 0
R 2(1, 5): 0
R 1(1, 6): 0
R 2(1, 6): 0
R 1(1, 7): 0
R 2(1, 7): 0
R 1(1, 8): 0
R 2(1, 8): 0
R 1(1, 9): 0
R 2(1, 9): 0
R 1(1,10): 0
R 2(1,10): 0
R 1(1,11): 0
R 2(1,11): 0
R 1(1,12): 0
R 2(1,12): 0
R 1(1,13): 0
R 2(1,13): 0
R 1(1,14): 0
R 2(1,14): 0
R 1(1,15): 0
R 2(1,15): 0

ENTER INITIAL CONDITIONS FOR VERTICAL S1(#,#)
S1( 1)( 1,1): 0
S1( 2)( 1,1): 0
S1( 1)( 2,1): 0
S1( 2)( 2,1): 0
S1( 1)( 3,1): 0
S1( 2)( 3,1): 0
S1( 1)( 4,1): 0
S1( 2)( 4,1): 0
S1( 1)( 5,1): 0
S1( 2)( 5,1): 0
S1( 1)( 6,1): 0
S1( 2)( 6,1): 0
S1( 1)( 7,1): 0
S1( 2)( 7,1): 0
S1( 1)( 8,1): 0
S1( 2)( 8,1): 0
S1( 1)( 9,1): 0
S1( 2)( 9,1): 0
S1( 1)(10,1): 0
S1( 2)(10,1): 0
S1( 1)(11,1): 0
S1( 2)(11,1): 0
S1( 1)(12,1): 0
```

```
S1( 1)(13,1): 0
S1( 2)(13,1): 0
S1( 1)(14,1): 0
S1( 2)(14,1): 0
S1( 1)(15,1): 0
S1( 2)(15,1): 0

ENTER INITIAL CONDITIONS FOR VERTICAL S2(#,#)
S2( 1)( 1,1): 0
S2( 2)( 1,1): 0
S2( 1)( 2,1): 0
S2( 2)( 2,1): 0
S2( 1)( 3,1): 0
S2( 2)( 3,1): 0
S2( 1)( 4,1): 0
S2( 2)( 4,1): 0
S2( 1)( 5,1): 0
S2( 2)( 5,1): 0
S2( 1)( 6,1): 0
S2( 2)( 6,1): 0
S2( 1)( 7,1): 0
S2( 2)( 7,1): 0
S2( 1)( 8,1): 0
S2( 2)( 8,1): 0
S2( 1)( 9,1): 0
S2( 2)( 9,1): 0
S2( 1)(10,1): 0
S2( 2)(10,1): 0
S2( 1)(11,1): 0
S2( 2)(11,1): 0
S2( 1)(12,1): 0
S2( 2)(12,1): 0
S2( 1)(13,1): 0
S2( 2)(13,1): 0
S2( 1)(14,1): 0
S2( 2)(14,1): 0
S2( 1)(15,1): 0
S2( 2)(15,1): 0

ENTER VALUES FOR THE INPUT VECTOR(#,#)
a(0 1): -0.35
a(0 2): 0
b(0 1): 0
b(0 2): 0

ENTER ELEMENTS OF THE TRANSITION MATRIX(#,#)
a( 10): -0.125
a( 20): -0.25
a( 1 1): -0.1
a( 2 1): 0
a( 1 2): 0
a( 2 2): -0.1
b( 1 1): 0
b( 2 1): 0
b( 1 2): 0
b( 2 2): 0

ENTER VALUES FOR THE OUTPUT VECTOR(#,#)
 b(00): 1
b( 10): 0
b( 20): 0
```

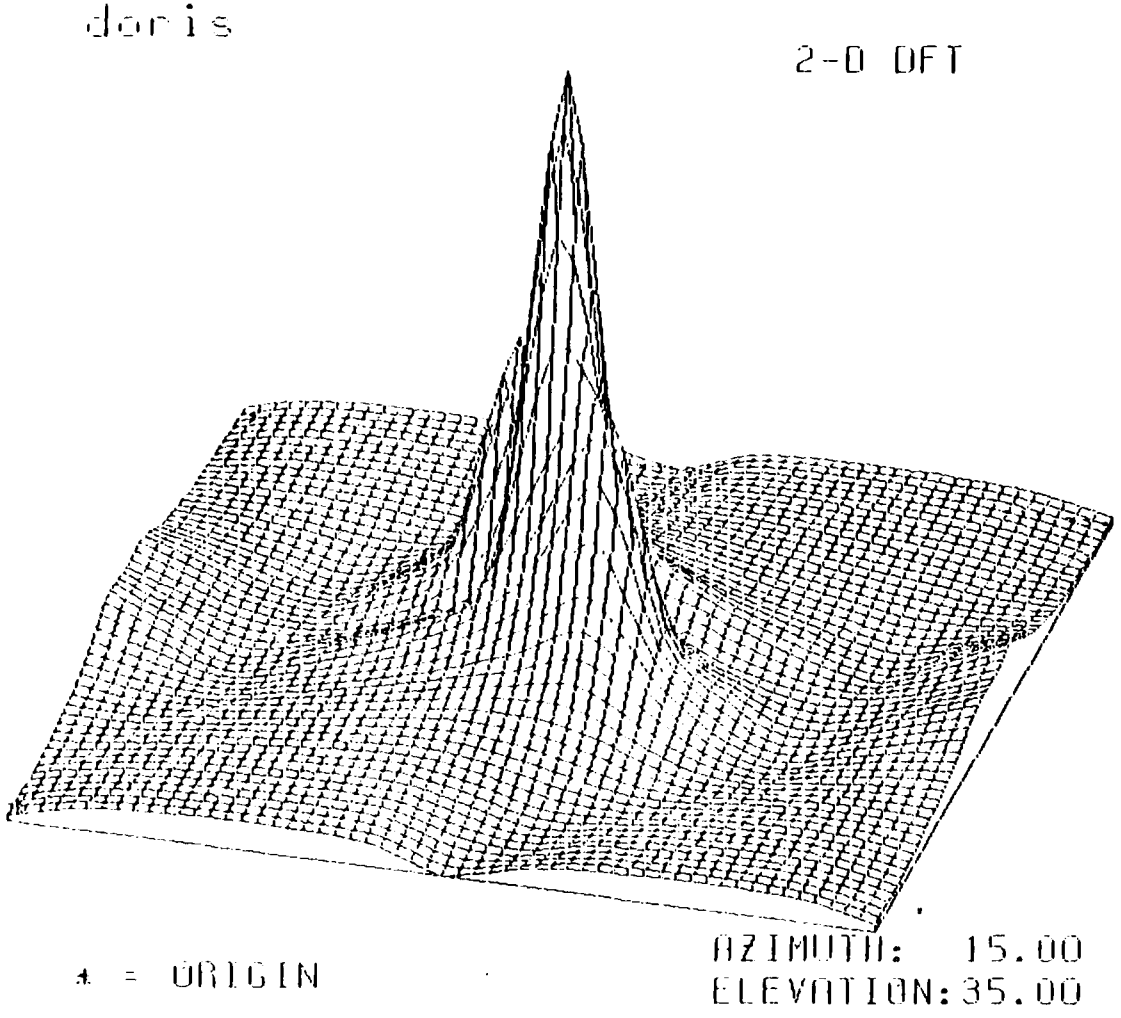


Figure 4-9a. 2-D D.F.T. Sequences, |Y(m,n)| for Example 9

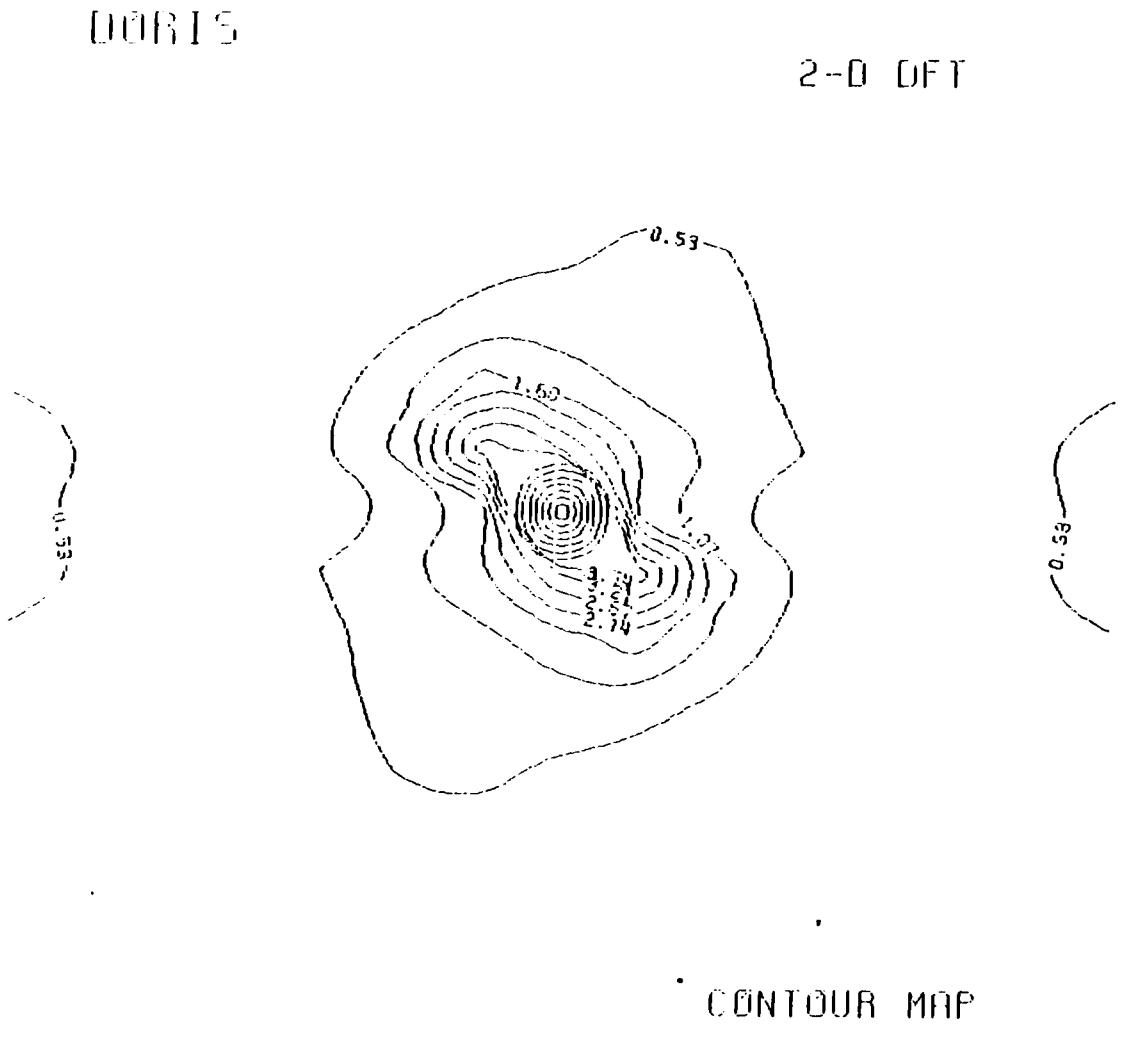


Figure 4-9b. Contour Map for Figure 4-9a

TABLE 4.2

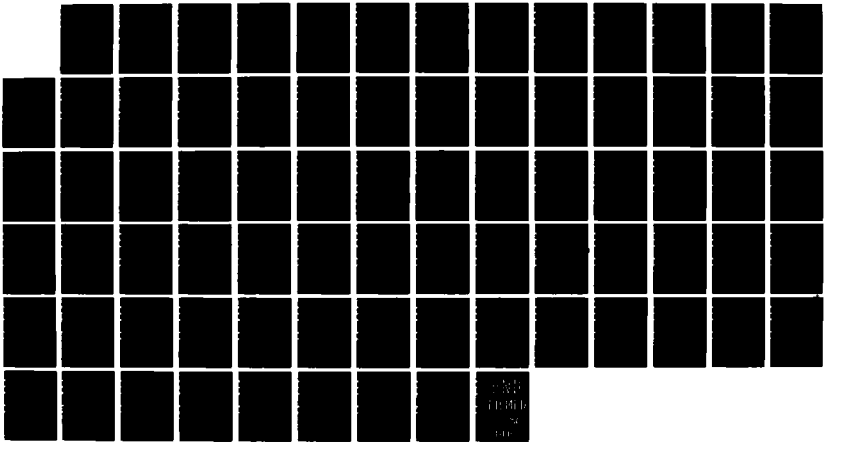
```
NUMBER OF HORIZONTAL STATES(N=1to4): 2

NUMBER OF VERTICAL STATES(M=1to4): 2

DIMENSION OF OUTPUT(1to25): 17

ENTER INITIAL CONDITIONS FOR HORIZONTAL R(#.#)
R 1(1, 1): 0
R 2(1, 1): 0
R 1(1, 2): 0
R 2(1, 2): 0
R 1(1, 3): 0
R 2(1, 3): 0
R 1(1, 4): 0
R 2(1, 4): 0
R 1(1, 5): 0
R 2(1, 5): 0
R 1(1, 6): 0
R 2(1, 6): 0
R 1(1, 7): 0
R 2(1, 7): 0
R 1(1, 8): 0
R 2(1, 8): 0
R 1(1, 9): 0
R 2(1, 9): 0
R 1(1,10): 0
R 2(1,10): 0
R 1(1,11): 0
R 2(1,11): 0
R 1(1,12): 0
R 2(1,12): 0
R 1(1,13): 0
R 2(1,13): 0
R 1(1,14): 0
R 2(1,14): 0
R 1(1,15): 0
R 2(1,15): 0
R 1(1,16): 0
R 2(1,16): 0
R 1(1,17): 0
R 2(1,17): 0

ENTER INITIAL CONDITIONS FOR VERTICAL S1(#.#)
S1( 1)( 1,1): 0
S1( 2)( 1,1): 0
S1( 1)( 2,1): 0
S1( 2)( 2,1): 0
S1( 1)( 3,1): 0
S1( 2)( 3,1): 0
S1( 1)( 4,1): 0
S1( 2)( 4,1): 0
S1( 1)( 5,1): 0
S1( 2)( 5,1): 0
S1( 1)( 6,1): 0
S1( 2)( 6,1): 0
S1( 1)( 7,1): 0
S1( 2)( 7,1): 0
S1( 1)( 8,1): 0
S1( 2)( 8,1): 0
S1( 1)( 9,1): 0
S1( 2)( 9,1): 0
S1( 1)(10,1): 0
S1( 2)(10,1): 0
S1( 1)(11,1): 0
```

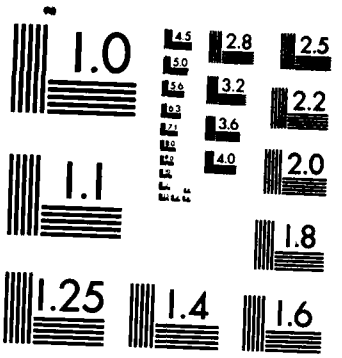
```
S1( 2)(12,1): 0
S1( 1)(13,1): 0
S1( 2)(13,1): 0
S1( 1)(14,1): 0
S1( 2)(14,1): 0
S1( 1)(15,1): 0
S1( 2)(15,1): 0
S1( 1)(16,1): 0
S1( 2)(16,1): 0
S1( 1)(17,1): 0
S1( 2)(17,1): 0

ENTER INITIAL CONDITIONS FOR VERTICAL S2(#,#)
S2( 1)( 1,1): 0
S2( 2)( 1,1): 0
S2( 1)( 2,1): 0
S2( 2)( 2,1): 0
S2( 1)( 3,1): 0
S2( 2)( 3,1): 0
S2( 1)( 4,1): 0
S2( 2)( 4,1): 0
S2( 1)( 5,1): 0
S2( 2)( 5,1): 0
S2( 1)( 6,1): 0
S2( 2)( 6,1): 0
S2( 1)( 7,1): 0
S2( 2)( 7,1): 0
S2( 1)( 8,1): 0
S2( 2)( 8,1): 0
S2( 1)( 9,1): 0
S2( 2)( 9,1): 0
S2( 1)(10,1): 0
S2( 2)(10,1): 0
S2( 1)(11,1): 0
S2( 2)(11,1): 0
S2( 1)(12,1): 0
S2( 2)(12,1): 0
S2( 1)(13,1): 0
S2( 2)(13,1): 0
S2( 1)(14,1): 0
S2( 2)(14,1): 0
S2( 1)(15,1): 0
S2( 2)(15,1): 0
S2( 1)(16,1): 0
S2( 2)(16,1): 0
S2( 1)(17,1): 0
S2( 2)(17,1): 0

ENTER VALUES FOR THE INPUT VECTOR(#,#)
a(0 1): 0
a(0 2): 0
b(0 1): 0
b(0 2): 0.125

ENTER ELEMENTS OF THE TRANSITION MATRIX(#,#)
a( 10): 0
a( 20): 0
a( 1 1): 0
a( 2 1): 0
a( 1 2): 0
a( 2 2): 0
b( 1 1): 0
b( 2 1): 0
b( 1 2): 0
b( 2 2): -0.125
```

```
 ENTER VALUES FOR THE OUTPUT VECTOR(#.#)
  b(00): 0.135
  b( 10): 0
  b( 20): 0.135

                        ***** INPUT VECTOR *****
1.00      .00      .00      .00      .00      .13

                        ***** OUTPUT VECTOR *****
 .00      .13     -.13      .00     1.00      .00

                     ***** TRANSITION MATRIX *****
 .00      .00    -1.00      .00      .00      .00

1.00      .00      .00      .00      .00      .00

 .00      .00      .00     1.00      .00      .00

 .00      .00      .00      .00      .00      .00

 .00      .00      .00      .00      .00     1.00

 .00     -.13     -.13      .00      .00      .00
```

AD-A164 128 2-D SIGNAL GENERATION USING STATE-SPACE FORMULATION(U) 2/2
NAVAL POSTGRADUATE SCHOOL MONTEREY CA E THEOFILOU
DEC 85

UNCLASSIFIED F/G 9/2 NL





MICROCOPY RESOLUTION TEST CHART
NATIONAL BUREAU OF STANDARDS-1963-A

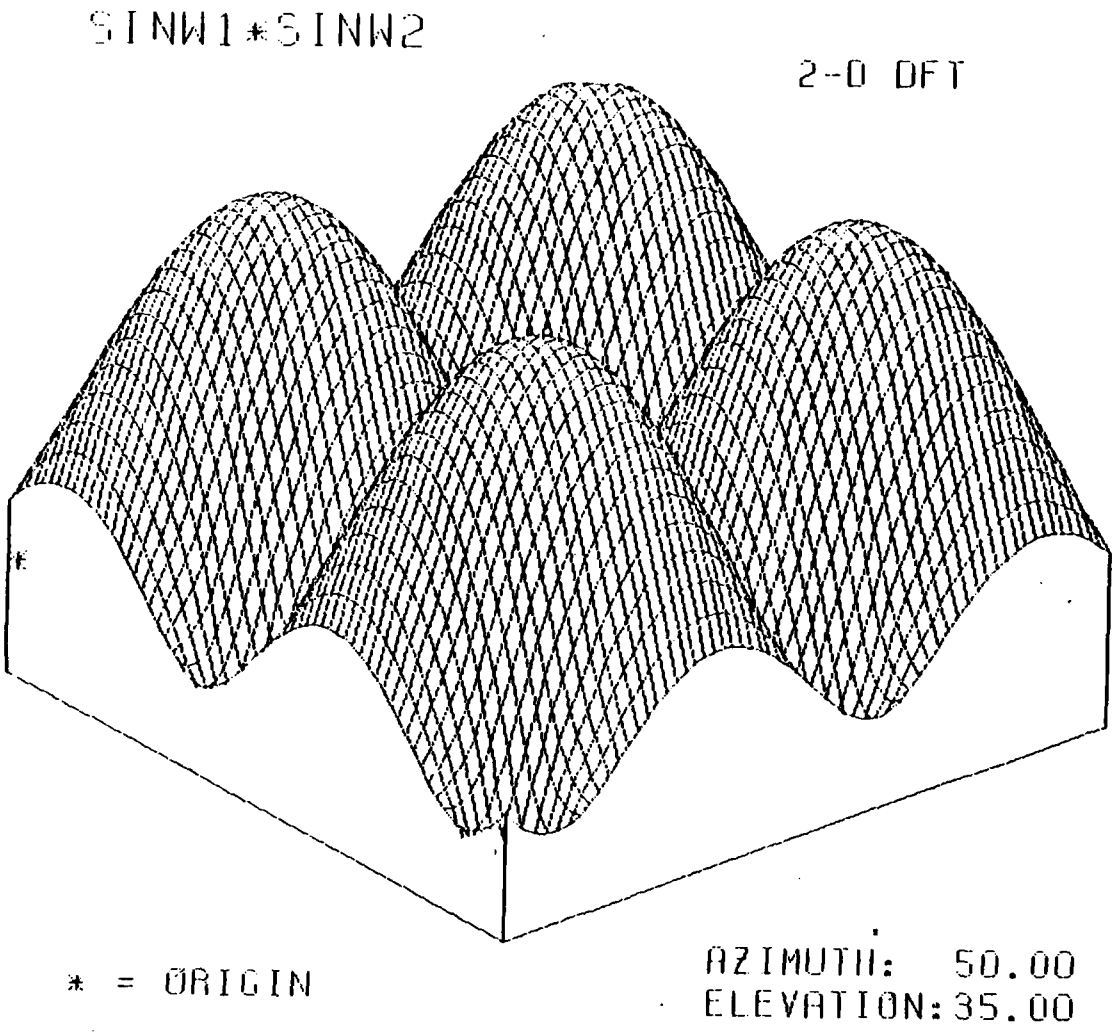


Figure 4-10a. 2-D D.F.T. Sequences, |Y(m,n)|, for Example 10

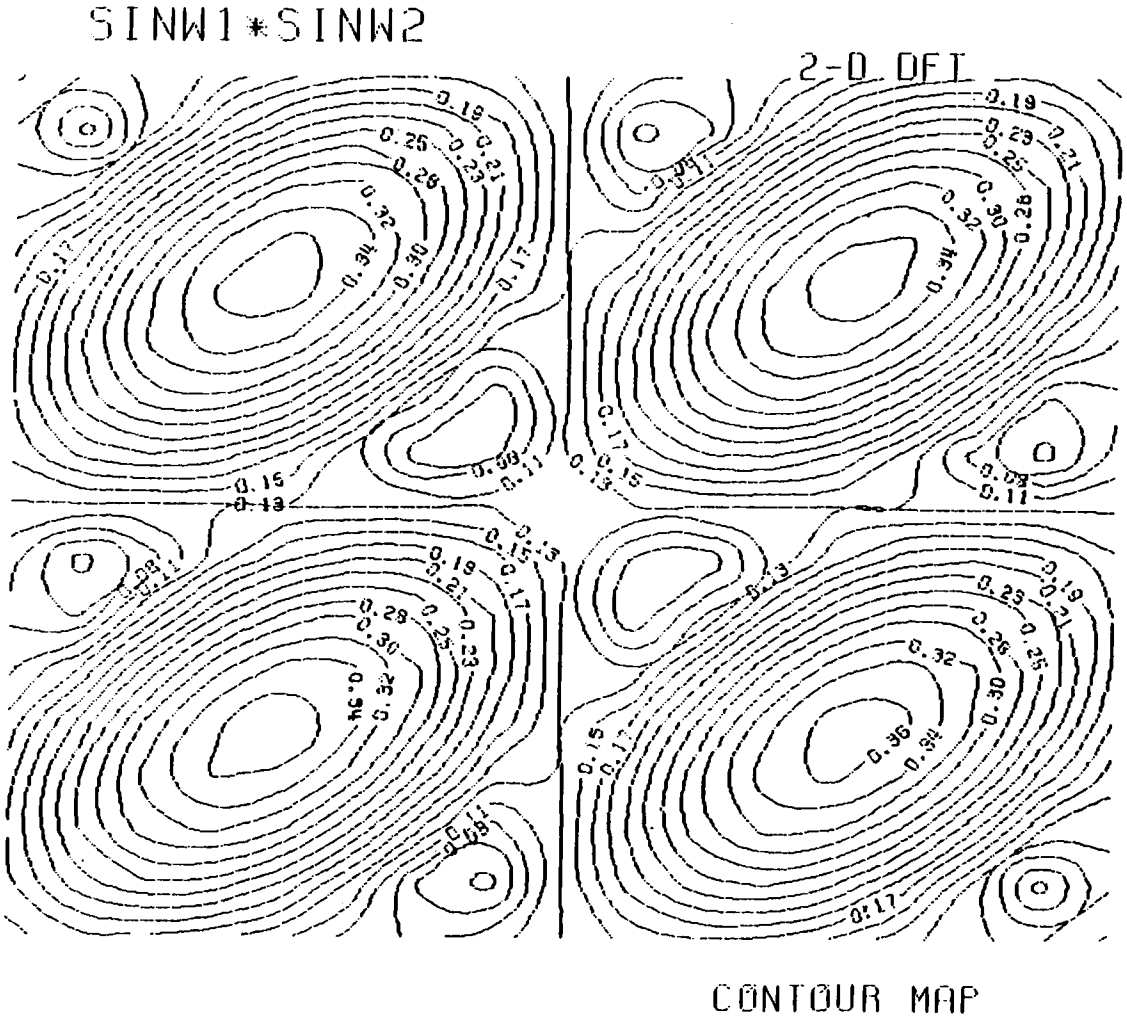


Figure 4-10b. Contour Map for Figure 4-10a

TABLE 4.3

```
NUMBER OF HORIZONTAL STATES(N=1to4): 2

NUMBER OF VERTICAL STATES(M=1to4): 2

DIMENSION OF OUTPUT(1to25): 17

ENTER INITIAL CONDITIONS FOR HORIZONTAL R(#,#)
R 1(1, 1): 0
R 2(1, 1): 0
R 1(1, 2): 0
R 2(1, 2): 0
R 1(1, 3): 0
R 2(1, 3): 0
R 1(1, 4): 0
R 2(1, 4): 0
R 1(1, 5): 0
R 2(1, 5): 0
R 1(1, 6): 0
R 2(1, 6): 0
R 1(1, 7): 0
R 2(1, 7): 0
R 1(1, 8): 0
R 2(1, 8): 0
R 1(1, 9): 0
R 2(1, 9): 0
R 1(1,10): 0
R 2(1,10): 0
R 1(1,11): 0
R 2(1,11): 0
R 1(1,12): 0
R 2(1,12): 0
R 1(1,13): 0
R 2(1,13): 0
R 1(1,14): 0
R 2(1,14): 0
R 1(1,15): 0
R 2(1,15): 0
R 1(1,16): 0
R 2(1,16): 0
R 1(1,17): 0
R 2(1,17): 0

ENTER INITIAL CONDITIONS FOR VERTICAL S1(#,#)
S1( 1)( 1,1): 0
S1( 2)( 1,1): 0
S1( 1)( 2,1): 0
S1( 2)( 2,1): 0
S1( 1)( 3,1): 0
S1( 2)( 3,1): 0
S1( 1)( 4,1): 0
S1( 2)( 4,1): 0
S1( 1)( 5,1): 0
S1( 2)( 5,1): 0
S1( 1)( 6,1): 0
S1( 2)( 6,1): 0
S1( 1)( 7,1): 0
S1( 2)( 7,1): 0
S1( 1)( 8,1): 0
S1( 2)( 8,1): 0
S1( 1)( 9,1): 0
S1( 2)( 9,1): 0
S1( 1)(10,1): 0
S1( 2)(10,1): 0
```

```
S1( 2)(11, 1): 0
S1( 1)(12, 1): 0
S1( 2)(12, 1): 0
S1( 1)(13, 1): 0
S1( 2)(13, 1): 0
S1( 1)(14, 1): 0
S1( 2)(14, 1): 0
S1( 1)(15, 1): 0
S1( 2)(15, 1): 0
S1( 1)(16, 1): 0
S1( 2)(16, 1): 0
S1( 1)(17, 1): 0
S1( 2)(17, 1): 0

ENTER INITIAL CONDITIONS FOR VERTICAL S2(#,#)
S2( 1)( 1, 1): 0
S2( 2)( 1, 1): 0
S2( 1)( 2, 1): 0
S2( 2)( 2, 1): 0
S2( 1)( 3, 1): 0
S2( 2)( 3, 1): 0
S2( 1)( 4, 1): 0
S2( 2)( 4, 1): 0
S2( 1)( 5, 1): 0
S2( 2)( 5, 1): 0
S2( 1)( 6, 1): 0
S2( 2)( 6, 1): 0
S2( 1)( 7, 1): 0
S2( 2)( 7, 1): 0
S2( 1)( 8, 1): 0
S2( 2)( 8, 1): 0
S2( 1)( 9, 1): 0
S2( 2)( 9, 1): 0
S2( 1)(10, 1): 0
S2( 2)(10, 1): 0
S2( 1)(11, 1): 0
S2( 2)(11, 1): 0
S2( 1)(12, 1): 0
S2( 2)(12, 1): 0
S2( 1)(13, 1): 0
S2( 2)(13, 1): 0
S2( 1)(14, 1): 0
S2( 2)(14, 1): 0
S2( 1)(15, 1): 0
S2( 2)(15, 1): 0
S2( 1)(16, 1): 0
S2( 2)(16, 1): 0
S2( 1)(17, 1): 0
S2( 2)(17, 1): 0

ENTER VALUES FOR THE INPUT VECTOR(#,#)
a(0 1): 0
a(0 2): 0
b(0 1): 0
b(0 2): 0.125

ENTER ELEMENTS OF THE TRANSITION MATRIX(#,#)
a( 10): 0
a( 20): 0
a( 1 1): 1
a( 2 1): 0
a( 1 2): 0
a( 2 2): 0
b( 1 1): 0
b( 2 1): 0
```

```
ENTER VALUES FOR THE OUTPUT VECTOR(0,m)
 b(00): -0.125
 b( 10): 0
 b( 30): 0.125

                    ***** INPUT VECTOR *****
1.00      .00      .00      .00      .00      .13

                    ***** OUTPUT VECTOR *****
 .00      .13      .13      .00     1.00      .00

                  ***** TRANSITION MATRIX *****
 .00      .00    -1.00      .00      .00      .00

1.00      .00      .00      .00      .00      .00

1.00      .00      .00     1.00      .00      .00

 .00      .00      .00      .00      .00      .00

 .00      .00      .00      .00      .00     1.00

 .00     -.13     -.13      .00      .00      .00
```

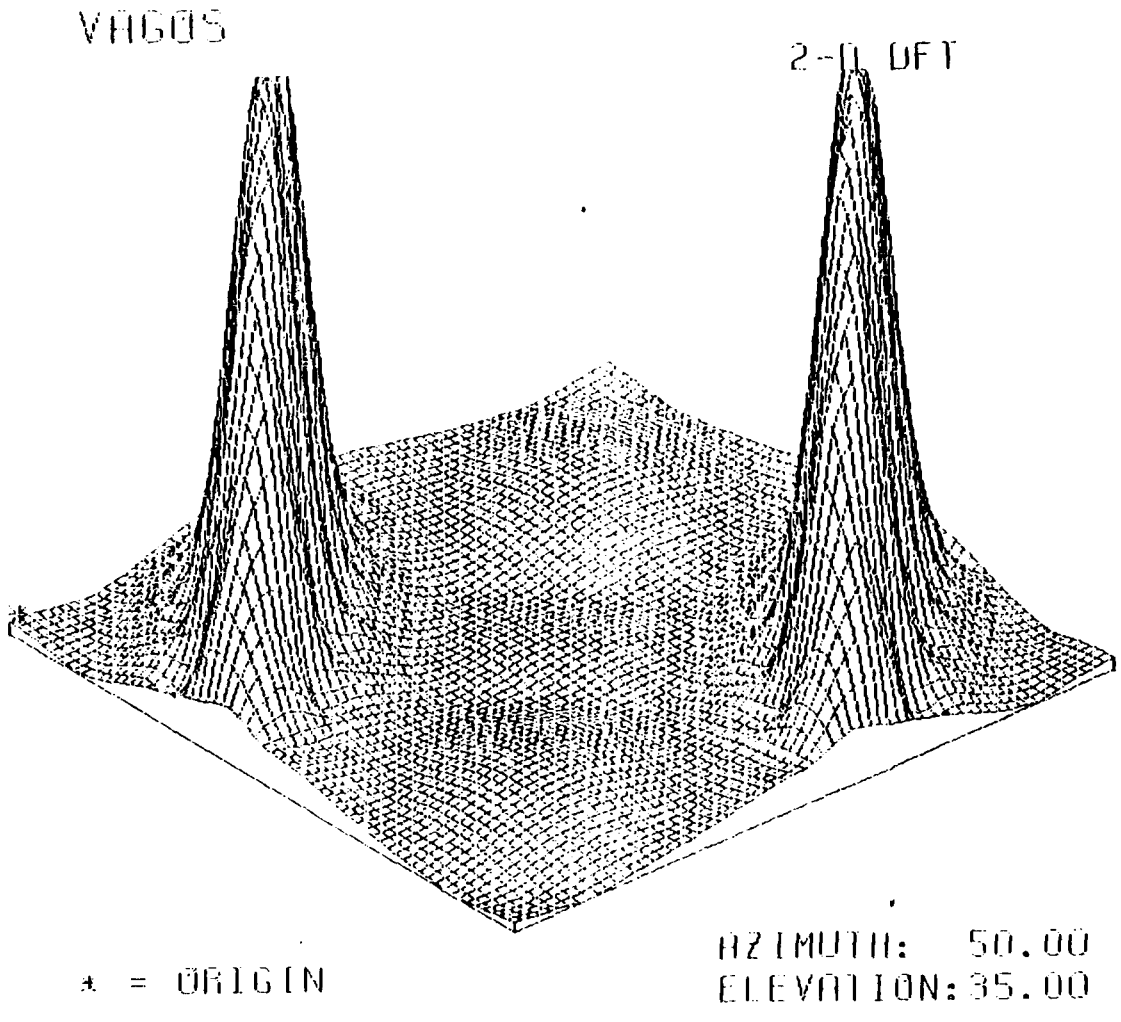


Figure 4-11a. 2-D D.F.T. Sequences, |Y(m,n)| for Example 11

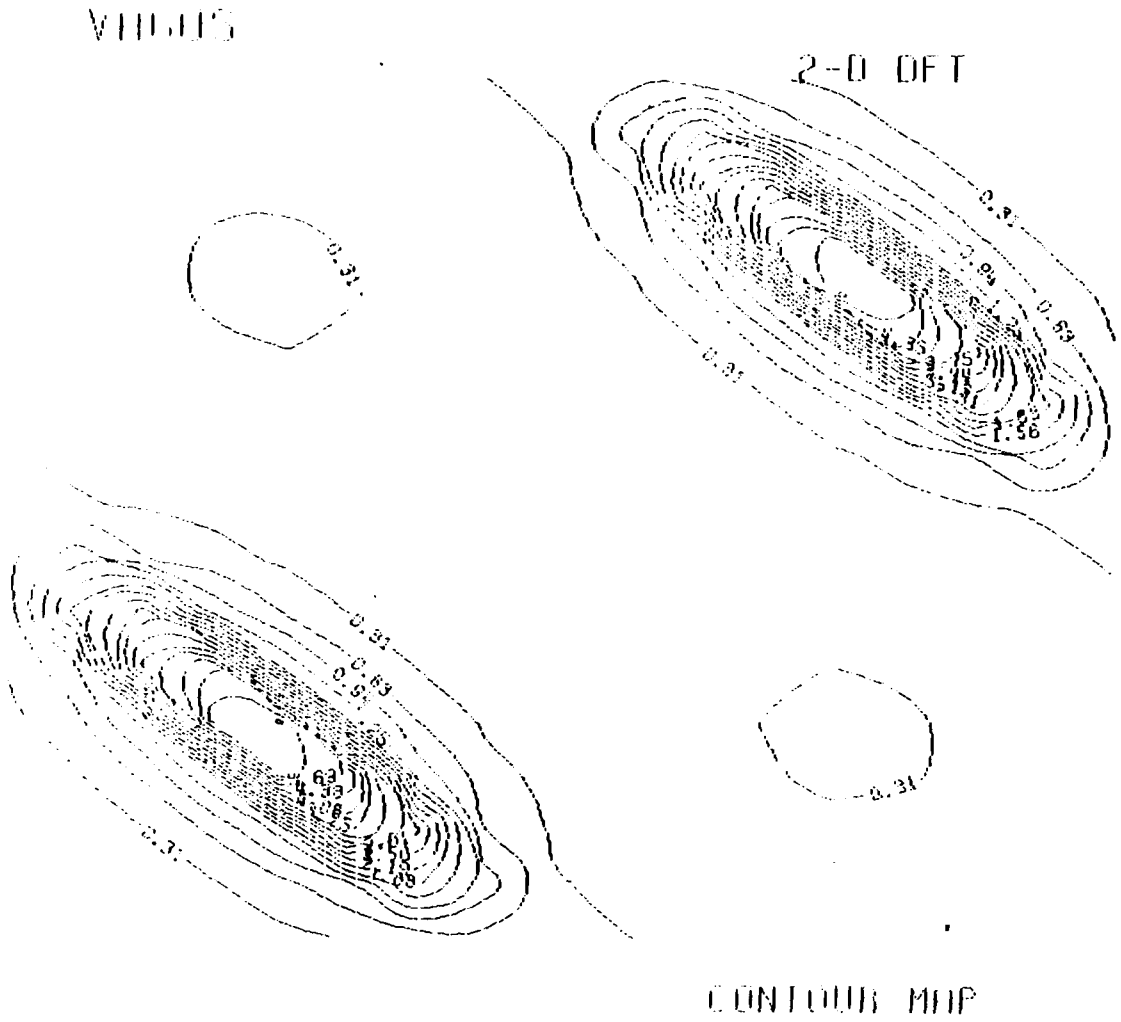


Figure 4-11b. Contour Map for Figure 4-11a

Once again, to verify the correctness of our program, the D.F.T. $|Y(m,n)|$ was compared to $|H(\omega_1,\omega_2)|$. $H(\omega_1,\omega_2)$ and the corresponding contour maps are shown in Fig. 4-12a,b, Fig. 4-13a,b and Fig. 4-14a,b for examples 9, 10 and 11, respectively.

F. SUMMARY OF PROGRAMS DEVELOPED

The programs which have been written, cover the following orders based upon the different models.

Appendix	Order	Model	# of States
A	1st	Roesser	1 horizontal, 1 vertical
C	2nd	Roesser	2 horizontal, 1 vertical
D	Multi-order	Kung	η horizontal, 2η vertical

In order to check the program listing, the same first order example was used on all programs. Identical results were obtained. Similarly, identical second order examples were used in Programs C and D and produced identical outputs.

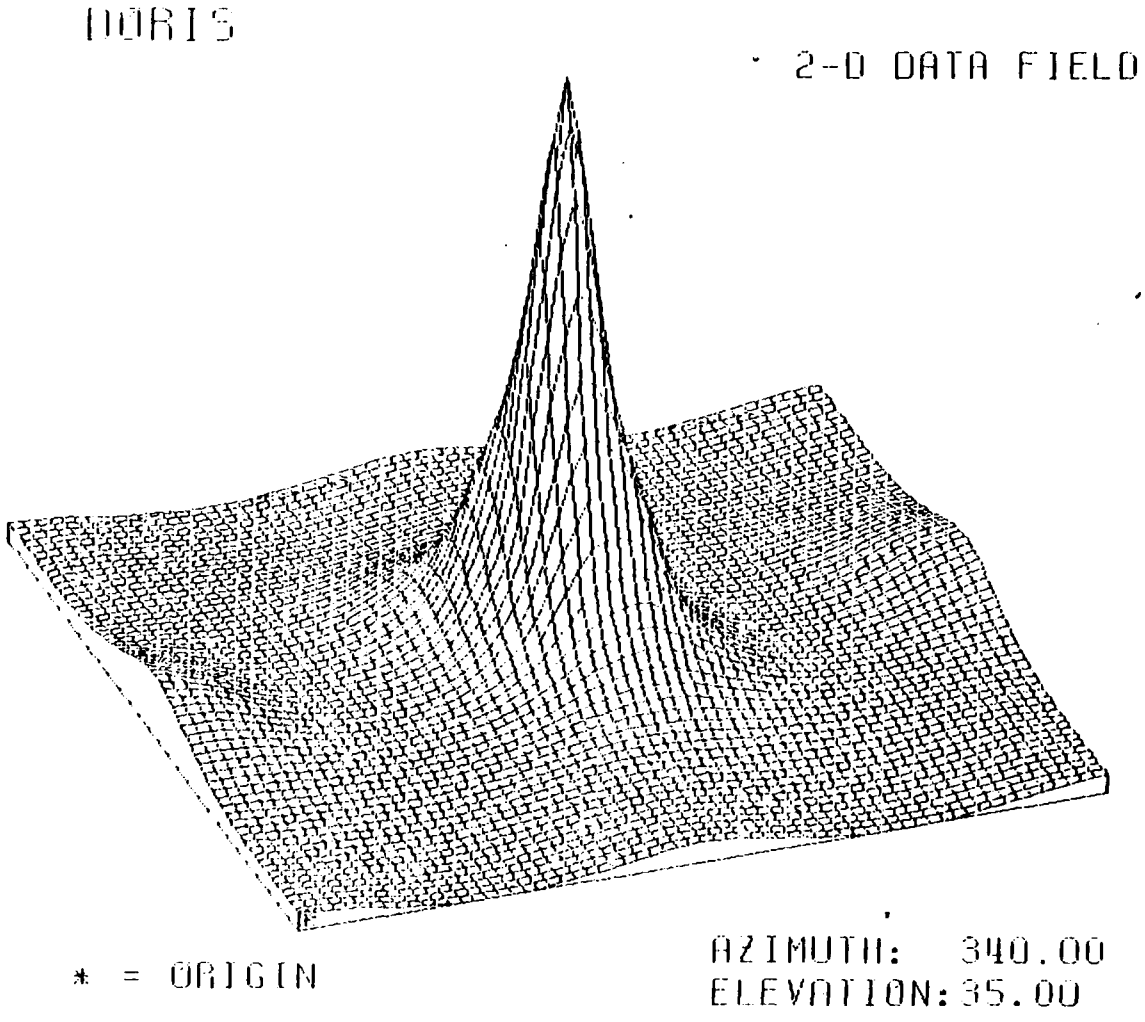


Figure 4-12a. Transfer Function $|H(z_1,z_2)|$, $z_1 = e^{j\omega_1}$, $z_2 = e^{j\omega_2}$ for Example 9

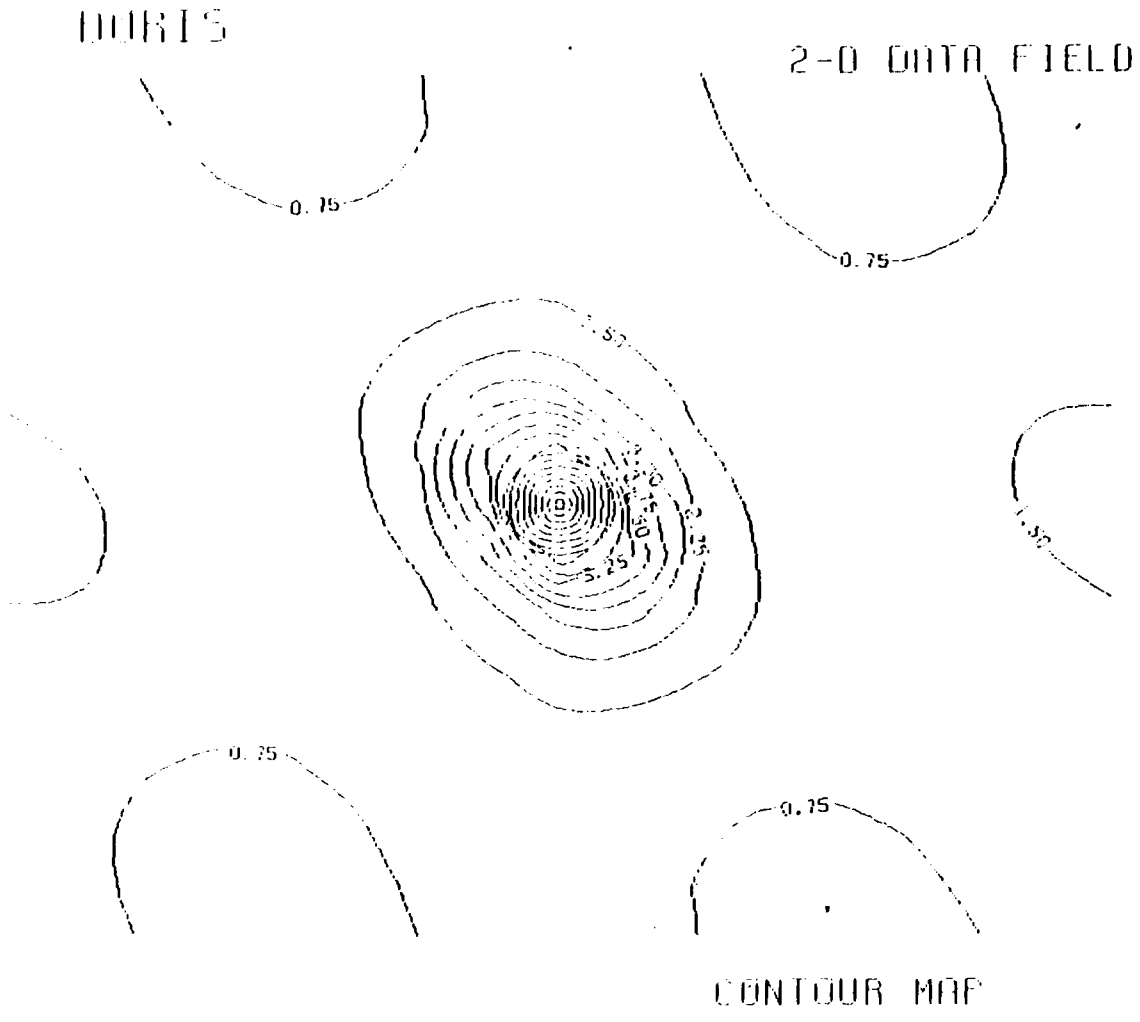


Figure 4-12b. Contour Map for Figure 4-12a

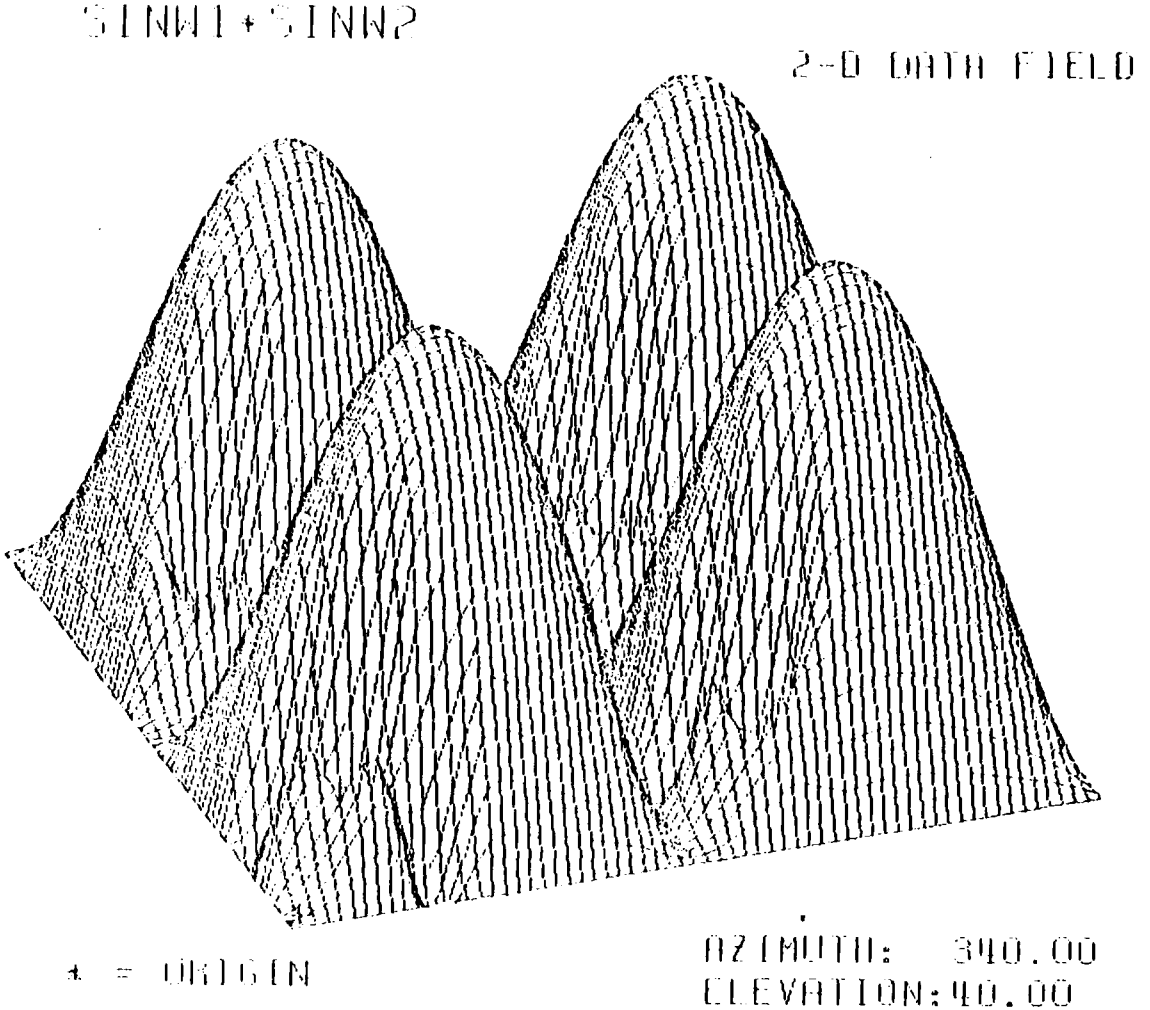


Figure 4-13a. Transfer Function $|H(z_1,z_2)|$, $z_1 = e^{j\omega_1}$, $z_2 = e^{j\omega_2}$ for Example 10

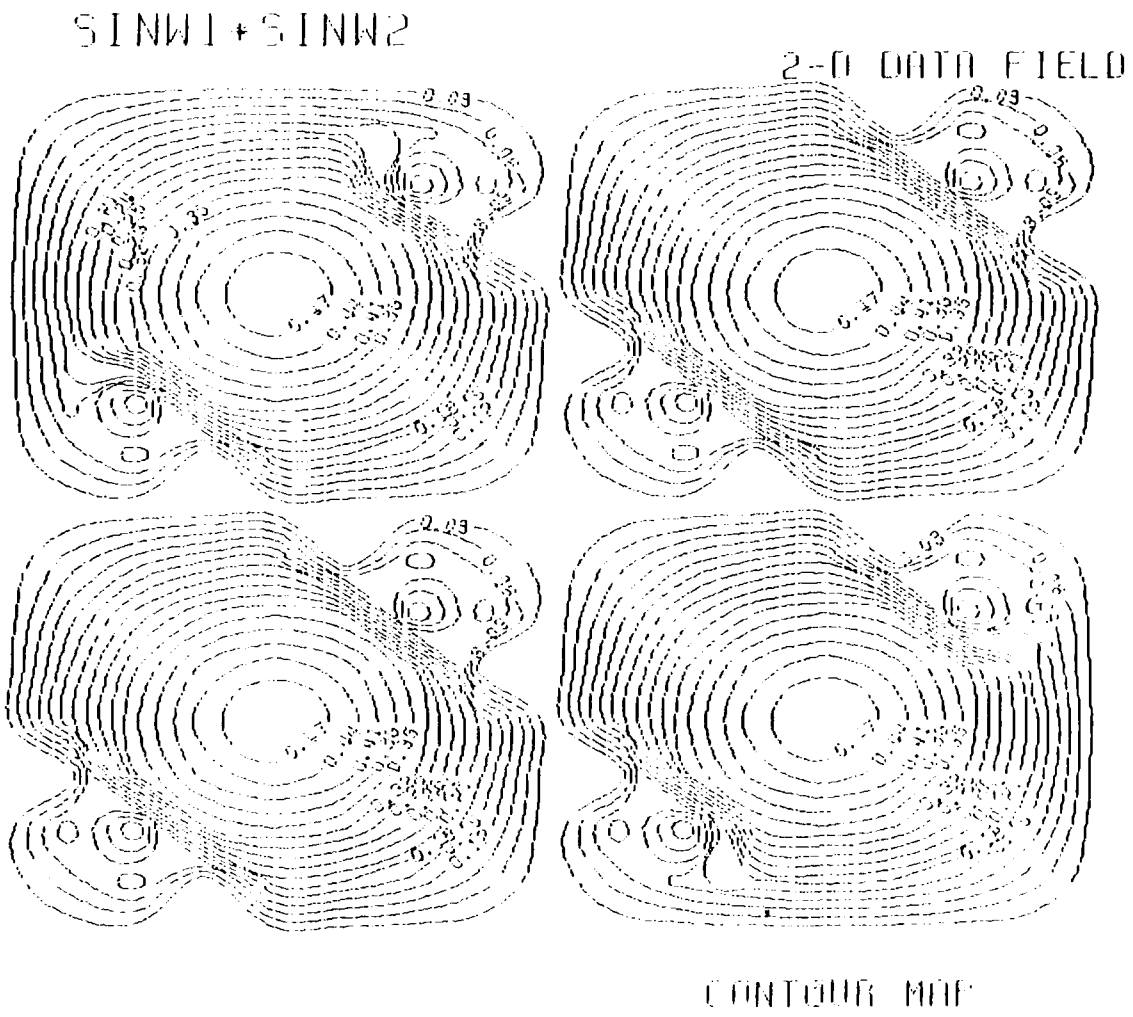


Figure 4-13b. Contour Map for Figure 4-13a

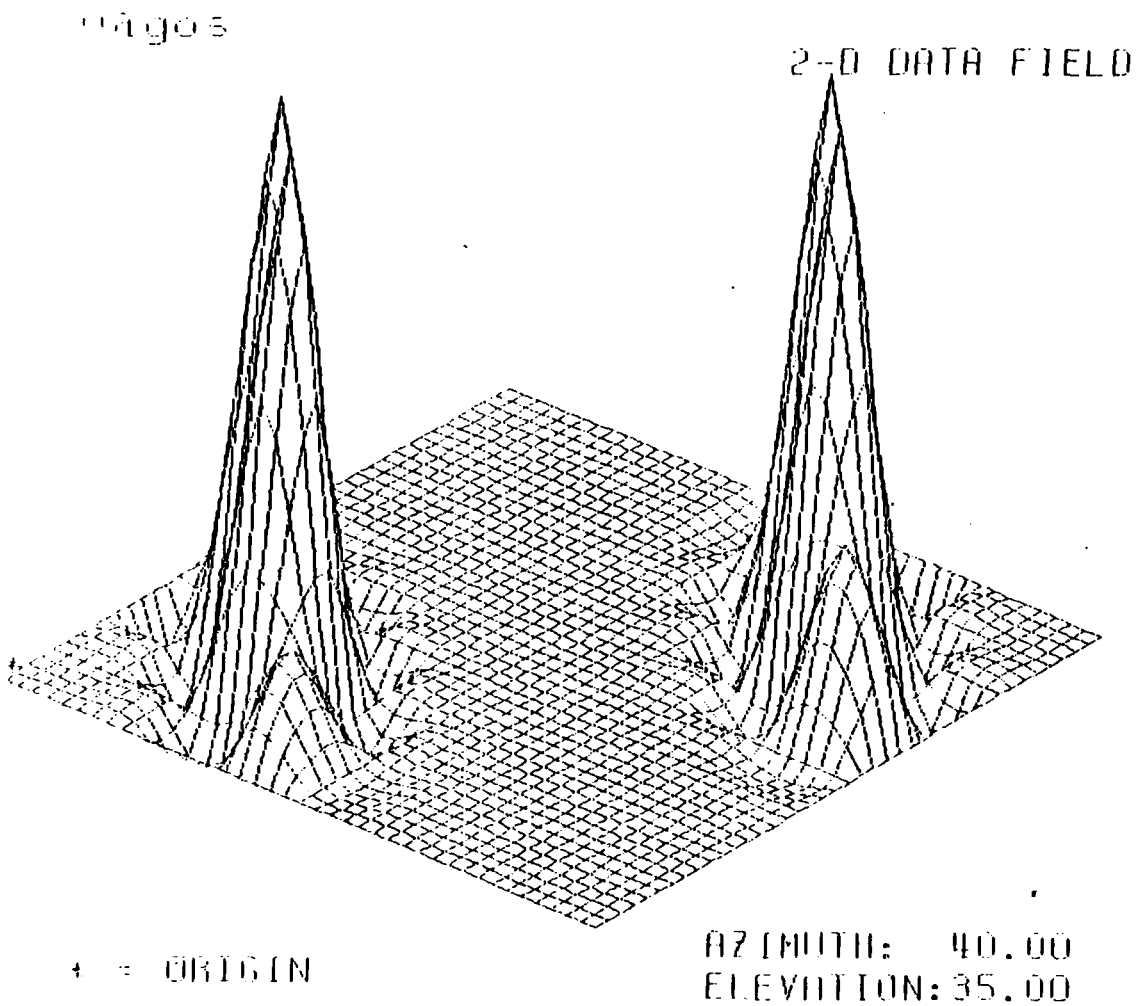


Figure 4-14a. Transfer Function $|H(z_1,z_2)|$, $z_1 = e^{j\omega_1}$, $z_2 = e^{j\omega_2}$ for Example #11

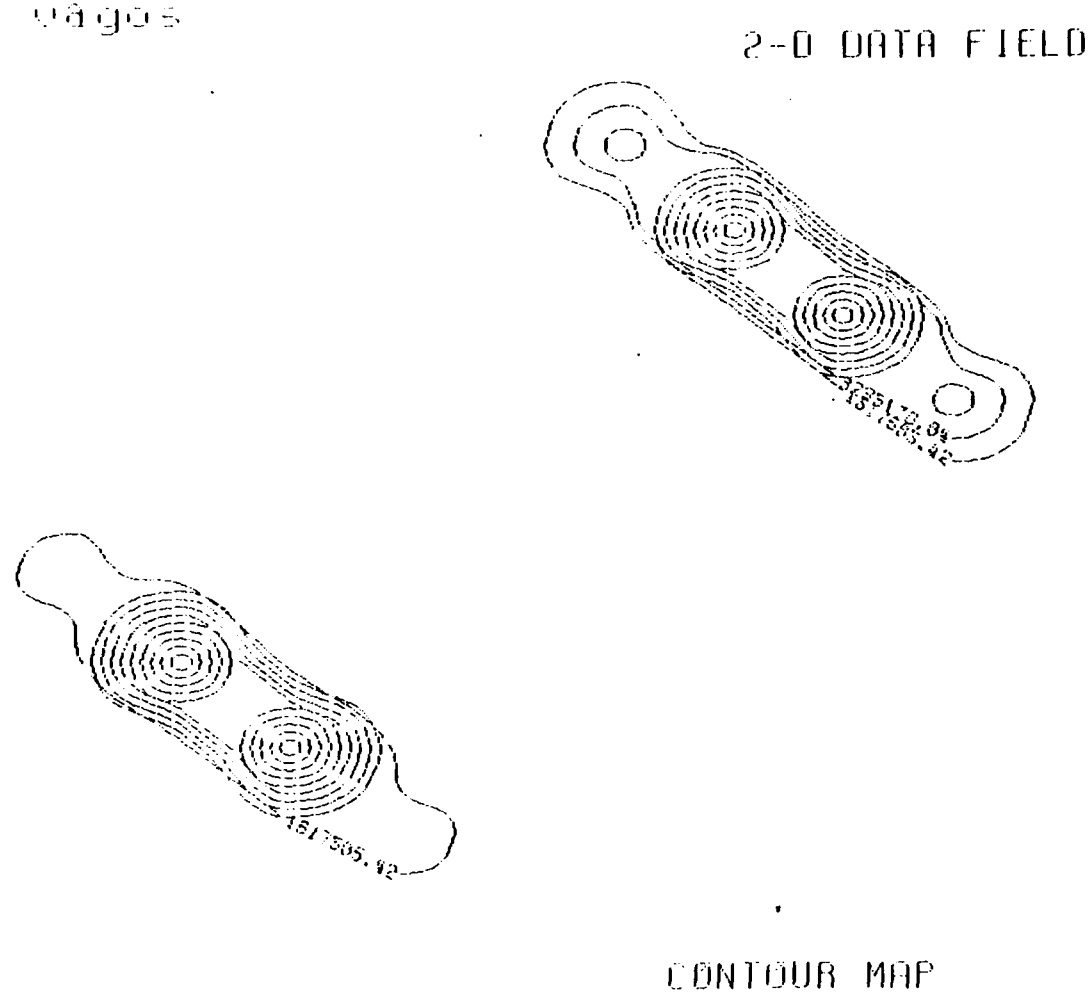


Figure 4-14b, Contour Map for Figure 4-14a

V. USE OF SSPACK PACKAGE

A. SSPACK

"SSPACK" is a "state space system package," [Ref. 21] that is an interactive, state-of-the-art, software package for the analysis, design, and display of one-dimensional state-space systems. The work which follows adapts this program so that it can be used to produce 2-D data fields from state space formulations. A brief description of SSPACK follows.

SSPACK is useful for a variety of applications in signal processing and control [Ref. 22]. The package consists of a supervisor which controls the operation of the software and a set of independent programs which communicate using disk files. The core of the package are the pre- and post-processors. The state-space pre-processor (SSPREP) program aids in preparing files for the individual algorithm programs. [Refs. 23,24] The state space post-processor (SSPOST) program displays and analyzes the output from the algorithms. SSPREP prompts with a series of questions in a menu format.

SSPOST is an interactive command-drive processor. It is designed to help interpret the output of the various SSPACK algorithms, and display time histories:

A is the Nx by Nx state transition matrix;

B is the Nx by Nu input transition matrix;

C is the Nz by Nx measurement matrix;

D is the Nz by Nu feedthru matrix;

W is the Nx by Nw process noise matrix;

V is the Nz by Nv measurement noise matrix.

The SSPACK works in multi-order form, using the transfer function of the 1-D digital filter.

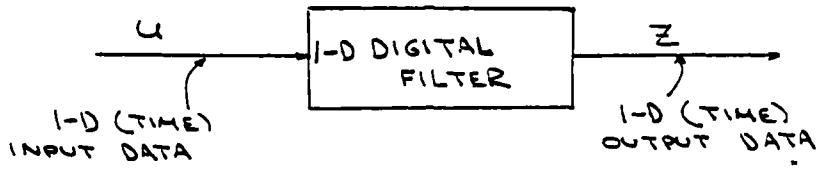


Figure 5-1

The present objective is to use SSPACK with a 2-D input data field and through the same transfer function, 1-D digital filter, to accomplish 2-D output data field.

B. DESIGN OF 2-D DIGITAL FILTERS USING 1-D DIGITAL FILTER STRUCTURES

The idea of using two types of dynamic elements is not very abstract; it is very natural in delay-differential systems. However, before considering its practical applications to image systems, two remarks have to be made. The first is because the "spatial" dynamic elements seem unimplementable, and we need to replace them by time-delay elements. Secondly, in order to have a finite order, we shall only consider a bounded frame system, i.e., we assume that the picture frame of interest is an $M \times N$ frame (with vertical width M and horizontal length N). Note that in order to use time delay elements, we need

first to find a way to code a 2-D spatial system into a 1-D (discrete time) system and vice versa.

Thus we propose the following system, composed of three subsystems in series:

i) The Input Scan Generator codes the 2-D spatial input into 1-D time data according to the mapping function

$$t(\cdot,\cdot) \qquad t(i,j) = iM + jN, \quad \begin{matrix} 0 \le i \le N-1 \\ 0 \le j \le M-1 \end{matrix} \qquad (V.1)$$

where M and N are relatively prime integers. For example, we consider a 2-D input data u(i,j):

$$u(i,j) = \begin{bmatrix} (0,0) & (0,1) & (0,2) & (0,3) & \cdots & (0,M-1) \\ (1,0) & (1,1) & & & & (1,M-1) \\ \vdots & \vdots & & & & \vdots \\ (N-1,0) & (N-1,1) & & & & (N-1,M-1) \end{bmatrix}$$

Scanning

The data field u(i,j) is scanned to produce u(t) as follows:

$$u(i,j) = (0,0),(0,1),(0,2),\ldots,(0,M-1),(1,0),(1,1),\ldots,(1,M-1),(N-1,0)\ldots(N-1,M-1)$$

$$\{u(t)\},\ t = 0,\ 1,\ 2,\ M,\ M+1,\ M+2,\ \ldots,\ (M-1)(N-1),\quad t = iM + jN$$

For example,

$$u(i,j) = \begin{bmatrix} 1 & 0 & 0 & \cdots & 0 \\ 0 & 0 & 0 & \cdots & 0 \\ \vdots & & & & \\ 0 & 0 & 0 & \cdots & 0 \end{bmatrix}$$

yields

$$y(t) = [1 \quad 0 \quad 0 \quad 0 \quad 0 \quad \cdots \quad 0 \quad 0 \quad 0 \quad 0]$$

ii) A 1-D (discrete time) digital filter processes the 1-D data generated. This subsystem is implemented by replacing z_1^{-1} by δ, z_2^{-1} by Δ in a 2-D circuit realization (e.g., 2-D controller form). δ and Δ are chosen as:

$$\delta = D^M = \text{M-units delay element}$$
$$\Delta = D^N = \text{N-units delay element}$$

iii) The Output Frame Generator decodes the 1-D (discrete time) output of the 1-D digital filter described above into a 2-D (discrete-spatial) picture according to the inverse mapping of (V.1).

$$(i(t),j(t)) = Pt \text{ Mod } N, [t-(Pt \text{ Mod } N)M]/N) \qquad (V.2)$$

where P is a unique integer such that PM-PN = 1 and $0 < P < N$. This formula is given in [Ref. 2]. Alternately, we can compute (i,j) as

$$i = t \text{ Mod } N$$

and

$$j = \text{Quotient } (t/N)$$

For example we suppose t = 19 with N = 10 and M = 9. The corresponding value in the 2-D case will be i = Remainder$\{\frac{19}{10}\}$ = 9 and j = Quotient $\{\frac{19}{10}\}$ = 1. So in the 2-D case we will have (i,j) = (9,1).

Another Example: For M = 4 and N = 5, the single index t will be mapped into (i,j) as:

i \ j	0	1	2	3	4
	5	6	7	8	9
	10	11	12	13	14
	15	16	17	18	19

The procedure for implementing 2-D filters using 1-D filter structures is as shown below in Fig. 5-2.

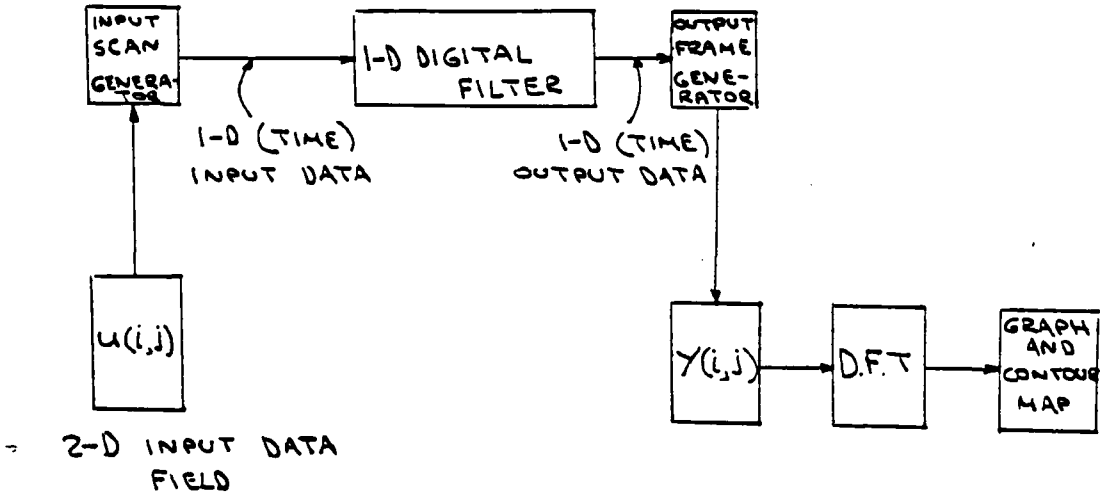


Figure 5-2

The index scanning is required for the input data so SSPACK can be carried out simply because the input is assumed to be a 2-D unit pulse. This is followed by implementing the corresponding 1-D filter of Fig. 5-3 using SSPACK to convert the 1-D output from SSPACK to a program for output index mapping--written as shown in Appendix E. The 2-D Fourier transform of the resulting 2-D field is then computed.

Considering a bounded frame ($M \times N$) system it is interesting to know the dimension of the global state (or initial conditions) needed to process the $M \times N$ future data field. Since vertical states convey information vertically, all the vertical states along the X-axis are necessary initial conditions and their dimension is mN. Similarly, all the horizontal states along the Y-axis are necessary initial conditions (with dimension nM). They convey information horizontally.

Therefore, in the bounded frame case a total number of mN+nM are needed to summarize the "past" information. This very same idea can be used again from a computational point of view. Indeed, the number of required storage elements for recursive computations is also equal to mN+mN if initial conditions are not zero. However, it is quite often the case that the system starts with zero initial conditions; the size of storage required is reduced to mN (respectively, nM) which is used to store the updated data row by row (respectively, column by column). No storage is needed for the rest of the initial conditions--nM horizontal states (respectively, mN vertical states) since they are assumed to be zero. This is consistent

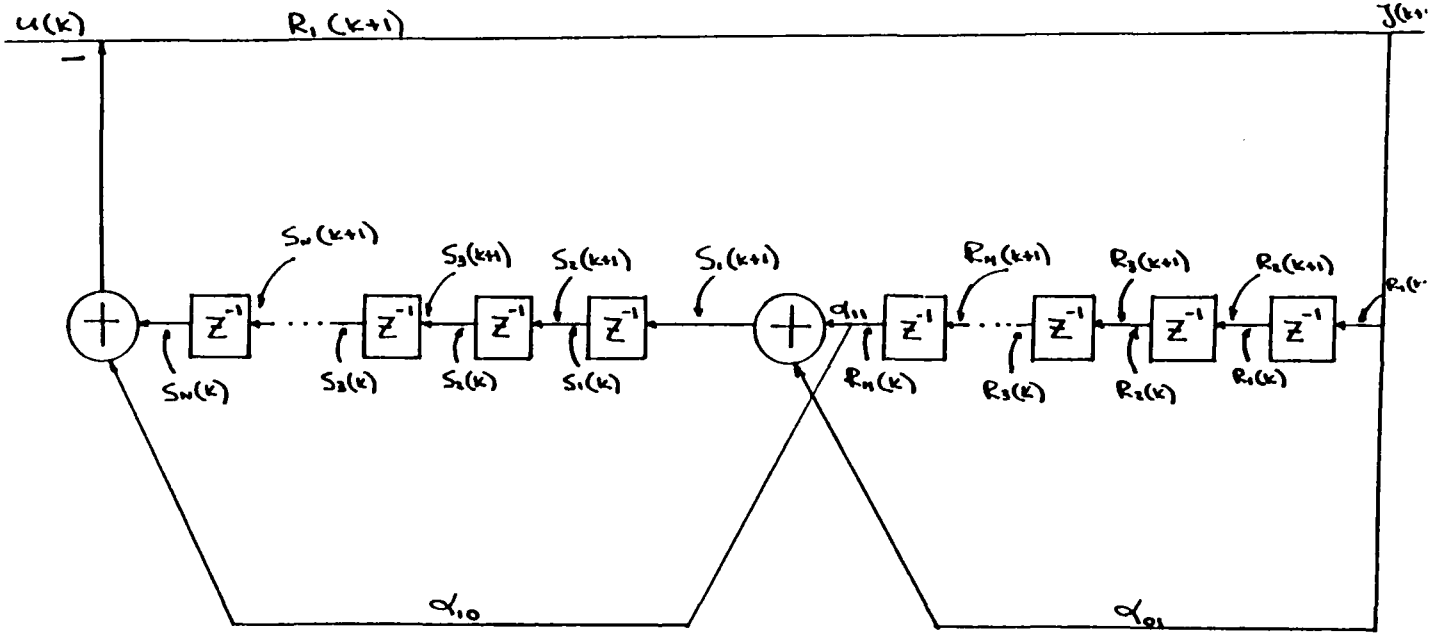

u(k)
R1(k+1)
J(k+1)
−
Sn(k+1)
S3(k+1)
S2(k+1)
S1(k+1)
Rn(k+1)
R3(k+1)
R2(k+1)
R1(k+1)
Z-1
Sn(k)
S3(k)
S2(k)
S1(k)
Rn(k)
R3(k)
R2(k)
R1(k)
α11
α10
α01

Figure 5-3

with the results of Read [Ref. 24] derived from a direct polynomial approach.

Another interesting observation concerns the dimension of the 1-D figital filter contained by our 2-D digital filter design discussed above. Since it needs nM unit-delays and mN-unit delays, the corresponding 1-D state-space also has a dimension equal to nM+mN. Note that, despite the high dimension of the corresponding 1-D filter, its high sparsity is very encouraging for further studies. In short, following the above method of designing a 2-D filter, for the first order case,

$$H(z_1,z_2) = \frac{1}{1+a_{10}z_1^{-1}+a_{01}z_2^{-1}+a_{11}z_1^{-1}z_2^{-1}} \qquad (V.3)$$

Using the above approach we get the 1-D filter realization for this 2-D filter which turns out to be as shown in Fig. 5-3.

The detailed matrix equations for realizing Eq. (V.3) using SSPACK can be written as, The SSPACK produces a 1-D sequence, which converted into a 2-D sequence using the output index mapping formulae discussed earlier. The listing of a program which does this mapping is shown in Appendix E.

After obtaining the valid 2-D output data sequence y(i,j) we next compute its 2-D D.F.T. to produce |Y(m,n)| which for this example is plotted in Fig. 5-4a. The corresponding contour map is as shown in Fig. 5-4b.

For a specific example, #12, we consider the following values:

$M = 2;\ N = 2$

$a_{11} - a_{10}a_{01} = -0.1 - 0.06 = -0.04$

$a_{01} = -.03 \quad a_{10} = -0.2 \quad a_{11} = -0.1$

$$\begin{bmatrix} R_1(1) \\ R_2(1) \\ \hline S_1(1) \\ S_2(1) \\ S_3(1) \end{bmatrix} = \left[\begin{array}{ccccc} 0 & -a_{10} & 0 & 0 & -1 \\ 1 & 0 & 0 & 0 & 0 \\ \hline 0 & (a_{11}-a_{10}a_{01}) & 0 & 0 & -a_{01} \\ 0 & 0 & 1 & 0 & 0 \\ 0 & 0 & 0 & 1 & 0 \end{array}\right] \begin{bmatrix} R_1(0) \\ R_2(0) \\ \hline S_1(0) \\ S_2(0) \\ S_3(0) \end{bmatrix} + \begin{bmatrix} 1 \\ 0 \\ \hline a_{01} \\ 0 \\ 0 \end{bmatrix} U(k)$$

$R_1(0), R_2(0), S_1(0), S_2(0), S_3(0)$ are the initial conditions. $U(k) = \begin{cases} 1 & \text{when } k = 0 \\ 0 & \text{otherwise} \end{cases}$

$$Y(k) = R_1(k) = C \begin{bmatrix} R_1(k) \\ R_2(k) \\ S_1(k) \\ S_2(k) \\ S_3(k) \end{bmatrix} = \underbrace{\begin{bmatrix} 1 & 0 & 0 & 0 & 0 \end{bmatrix}}_{M+N} \begin{bmatrix} R_1(1) \\ R_2(1) \\ \hline S_1(1) \\ S_2(1) \\ S_3(1) \end{bmatrix}$$

$$\begin{bmatrix} R_1(0) \\ R_2(0) \\ S_1(0) \\ S_2(0) \\ S_3(0) \end{bmatrix} = 0 \quad \text{initial conditions}$$

$$
\begin{bmatrix} R_1(k+1) \\ R_2(k+1) \\ R_3(k+1) \\ \vdots \\ R_M(k+1) \end{bmatrix}
\quad M
$$

$$
\begin{bmatrix}
0 & 0 & 0 & \cdots & -a_1 0 & 0 & 0 & 0 & \cdots & -1 \\
1 & 0 & 0 & \cdots & 0 & 0 & 0 & 0 & \cdots & 0 \\
0 & 1 & 0 & \cdots & 0 & 0 & 0 & 0 & \cdots & 0 \\
\vdots & \vdots & \vdots & & \vdots & \vdots & \vdots & \vdots & & \vdots \\
0 & 0 & 0 & \cdots & 1 \; 0 & 0 & 0 & 0 & \cdots & 0
\end{bmatrix}
\begin{bmatrix} R_1(k) \\ R_2(k) \\ R_3(k) \\ \vdots \\ R_M(k) \end{bmatrix}
+
\begin{bmatrix} 1 \\ Q \\ 0 \\ \vdots \\ 0 \end{bmatrix}
U(k)
$$

$$
\begin{bmatrix} S_1(k+1) \\ S_2(k+2) \\ S_3(k+3) \\ \vdots \\ S_N(k+1) \end{bmatrix}
\quad N
$$

$$
\begin{bmatrix}
0 & 0 & 0 & \cdots & (a_{11} - a_{10}a_{01}) & 0 & 0 & 0 & \cdots & -a_{01} \\
0 & 0 & 0 & \cdots & 0 & 1 & 0 & 0 & \cdots & 0 \\
0 & 0 & 0 & \cdots & 0 & 0 & 1 & 0 & \cdots & 0 \\
\vdots & \vdots & \vdots & & & \vdots & \vdots & \vdots & & \\
0 & 0 & 0 & \cdots & 0 & 0 & 0 & 0 & \cdots & 1 \; 0
\end{bmatrix}
\begin{bmatrix} S_1(k) \\ S_2(k) \\ S_3(k) \\ \vdots \\ S_N(k) \end{bmatrix}
+
\begin{bmatrix} a_{01} \\ 0 \\ 0 \\ \vdots \\ 0 \end{bmatrix}
$$

$$
Y(k) = R_1(k+1)
$$

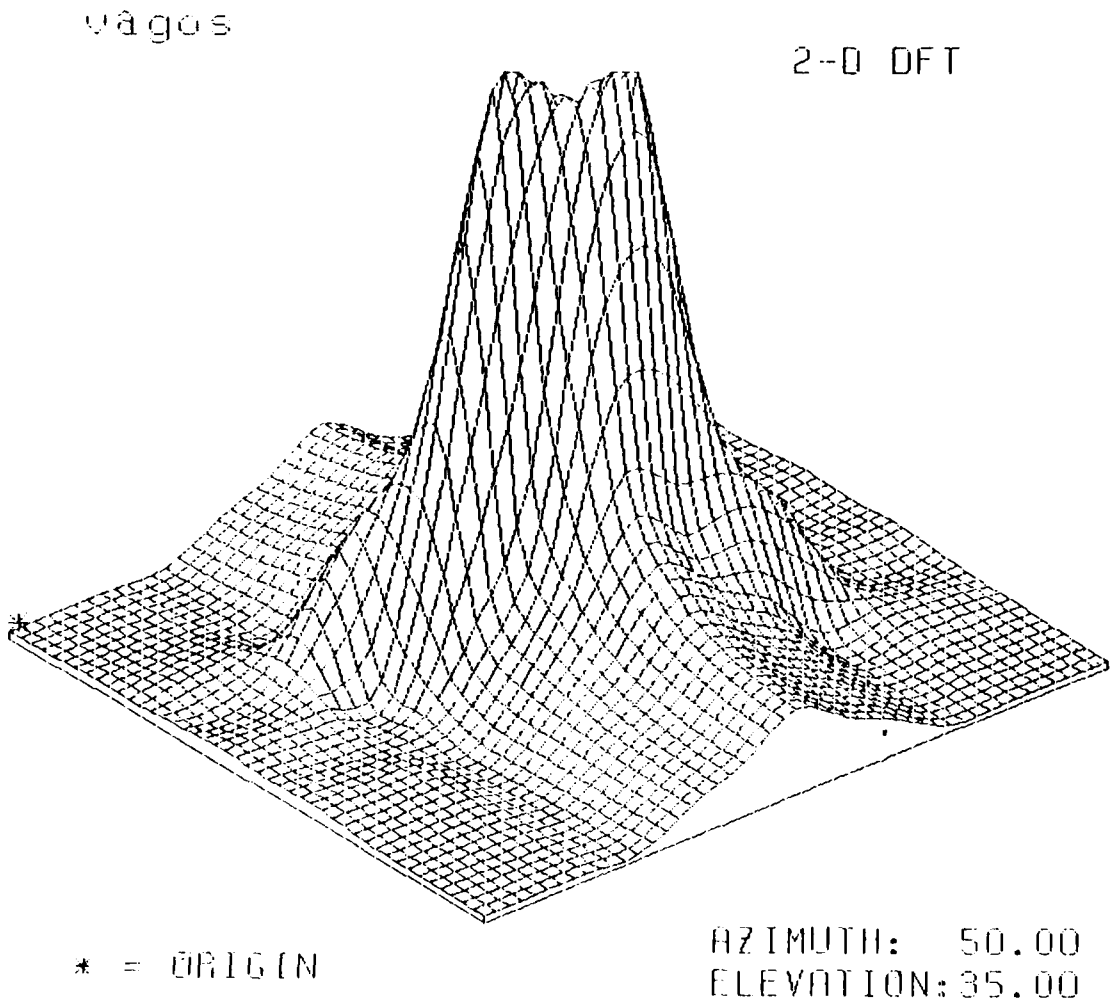


Figure 5-4a. 2-D D.F.T. Sequences, |Y(m,n)| for Example 12

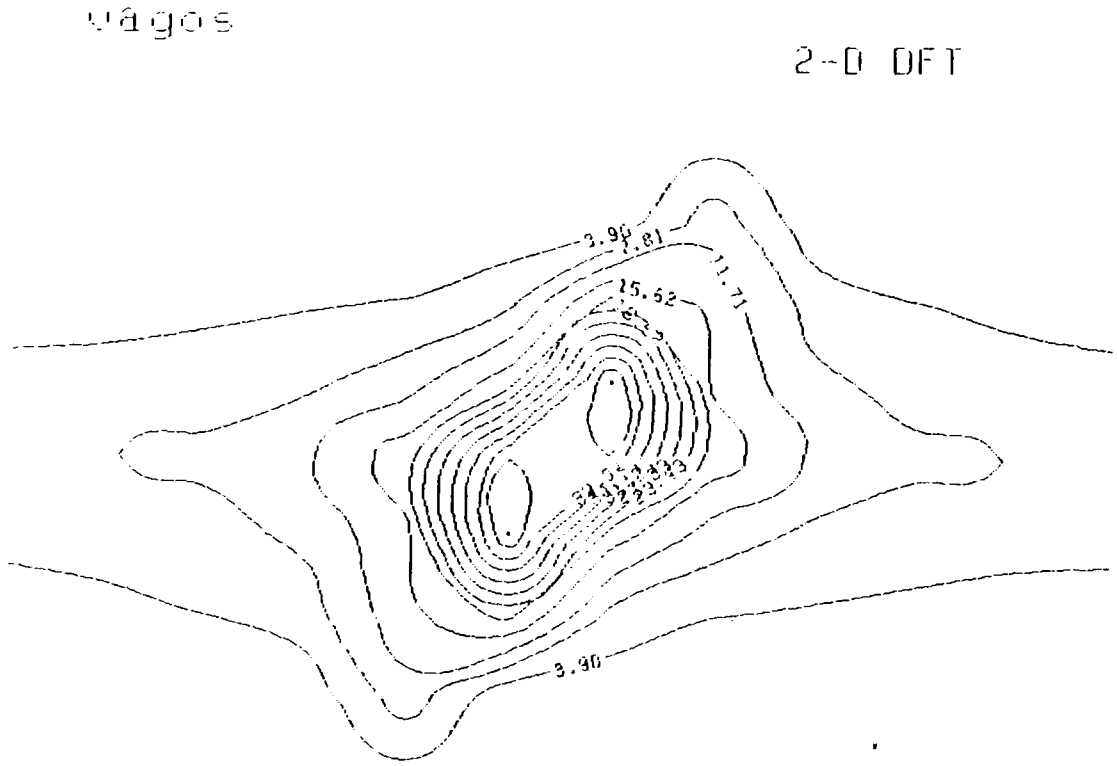


Figure 5-4b. Contour Map for Figure 5-4a

VI. CONCLUSIONS

This thesis has dealt with the problem of modelling 2-D data fields in the state-space domain. First of all we have pointed out the main problems associated with the extension of 1-D time-discrete state-space models to 2-D data fields. The remaining part of the thesis has been divided primarily in 3 parts.

In the first part we describe Roesser's [Ref. 5] approach to modelling 2-D systems in the state space domain. Extensive computer simulation results are presented to verify the functioning of this approach. This modelling approach has been tried out for the scalar (1×1) as well as for higher order (2×2) etc., 2-D systems.

The second part deals with a modification of Roesser's approach as described by Kung [Ref. 7]. The main advantage of this approach is that the 2-D state-space model can be realized as a 2-D circuit. More importantly, this 2-D circuit realization can be implemented as a 1-D digital filter. Computer simultation studies that have been carried out substantiate the making of this model. The 1-D filter realization obtained in this part turns out to be a very convenient starting point for the nezt part of our effort, dealing with the use of the 1-D SSPACK commercial software package designed for dynamic system simulation.

In the final part of the thesis, we make use of the 1-D filter realization of 2-D state-space model obtained in the second

part, and implement this filter using SSPACK. Some additional programming effort reuiqred for input and output mapping was necessary. Programs for converting 2-D input and output sequences to 1-D have been written separately. In this fashion we have succeeded in extending the applicability of the SSPACK to simulating 2-D linear systems as well. Once again, detailed computer simulations have been carried out to verify the functioning of this modification of the SSPACK.

APPENDIX A


```
D Line#  1     7                                  Microsoft FORTRAN77 V3.30 02/84
      1 $STORAGE:  2
      2 $LARGE
      3
      4 C     ***************************************************************
      5 C     *                                                             *
      6 C     *  THE PURPOSE OF THIS PROGRAM IS  TO  COMPUTE  AND GRAPH THE     *
      7 C     *  EQUATIONS OF ROBERT P.ROESSER IN THE "DISCRETE STATE-SPACE     *
      8 C     *  MODEL FOR LINEAR IMAGE PROCESSING".                        *
      9 C     *                                                             *
     10 C     *                          EVANGELOS THEOFILOU                *
     11 C     ***************************************************************
     12 C     PROGRAM 2D-DATA-FIELD
     13 .
     14 C     ***** VARIABLE DECLARATIONS *****
     15       REAL         R(25,25),S(25,25),R1(2),R2(2),Z(31,31),
     16      *             RLPART,IMGPART,ZF(31,31),VERTEX(16),ZLEV(31)
     17       INTEGER      MASK(3000),LDIG(31),LWGT(31)
     18       CHARACTER*1 ANSWER
     19       CHARACTER*20 CTEXT
     20
     21       DATA          XLOL/0.0/,YLOL/0.0/,XUPR/8.5/,YUPR/7.0/,
     22      *              ZLOW/1.0E35/,IPROJ/0/,NRNG/100/
     23
     24 C     ************  M A I N   P R O G R A M  ************
     25
     26 C     ***** ASK THE REQUIRED VALUES FOR THE MODEL *****
     27    10 WRITE (*,*) 'ENTER VALUES FOR THE FOLLOWING VARIABLES(#.#,..):'
     28       WRITE (*,399) 'A1: '
     29       READ (*,*) A1
     30       WRITE (*,399) 'A2: '
     31       READ (*,*) A2
     32       WRITE (*,399) 'A3: '
     33       READ (*,*) A3
     34       WRITE (*,399) 'A4: '
     35       READ (*,*) A4
     36       WRITE (*,399) 'B1: '
     37       READ (*,*) B1
     38       WRITE (*,399) 'B2: '
     39       READ (*,*) B2
     40       WRITE (*,399) 'C1: '
     41       READ (*,*) C1
     42       WRITE (*,399) 'C2: '
     43       READ (*,*) C2
     44     5 WRITE (*,402)
     45       READ  (*,*) N
     46       IF (N .GT. 25) GOTO 5
     47
     48       WRITE (*,211) 'ENTER ',N,' INITIAL CONDITIONS FOR MATRIX R(#.#)'
     49       DO 99 I = 1,N
  1  50       WRITE (*,403) 'R(1,',I,'): '
  1  51       READ  (*,*) R(1,I)
  1  52    99 CONTINUE
     53
     54       WRITE (*,211) 'ENTER ',N,' INITIAL CONDITIONS FOR MATRIX S(#.#)'
     55       DO 100 I = 1,N
  1  56       WRITE (*,404) 'S(',I,'1): '
  1  57       READ  (*,*) S(I,1)
  1  58   100 CONTINUE
     59
```

```
D Line# 1      7                                    Microsoft FORTRAN77 V3.20 02/84
      60       WRITE (*,413)
      61       READ  (*,200) ANSWER
      62       IF  ((ANSWER .EQ. 'Y') .OR. (ANSWER .EQ. 'y')) GOTO 10
      63
      64       U = 1.0
      65 C     ***** COMPUTE R AND S MATRICES *****
      66       DO 101 I = 1,N
1     67          DO 101 J = 1,N
2     68             IF  (I+1 .LE. N)  THEN
2     69                R(I+1,J) = A1*R(I,J) + A2*S(I,J) + B1*U
2     70             ENDIF
2     71             IF  (J+1 .LE. N)  THEN
2     72                S(I,J+1) = A3*R(I,J) + A4*S(I,J) + B2*U
2     73             ENDIF
2     74             U = 0.0
2     75   101 CONTINUE
      76
      77 C     **** FILL 0's THE TWO DIMENTIONAL GRID OF CONTROL POINTS ****
      78       DO 102 I = 1,31
1     79         DO 102 J = 1,31
2     80           Z(I,J) =0.0
2     81   102 CONTINUE
      82
      83 C     ***** COMPUTE Z MATRIX *****
      84       DO 103 I = 1,N
1     85         DO 103 J = 1,N
2     86           Z(I,J) = C1*R(J,I) + C2*S(J,I)
3     87   103 CONTINUE
      88
      89 C     ***** OUTPUT THE Z MATRIX *****
      90       WRITE (*,205) '*****  Z  M A T R I X  ',N,' X ',N,'  *****'
      91       WRITE (*,212)
      92       DO 104 I = 1,N
1     93         WRITE (*,300) (Z(I,J), J = 1,N)
1     94         WRITE (*,210)
1     95   104 CONTINUE
      96       WRITE (*,213)
      97
      98       WRITE (*,418)
      99       READ  (*,200) ANSWER
     100       IF  ((ANSWER .NE. 'Y') .AND. (ANSWER .NE. 'y')) GOTO 18
     101
     102 C     ***** ASK THE PARAMETERS FOR THE GRAPH *****
     103   15  WRITE (*,210)
     104       WRITE (*,*) '***  E N T E R  P L O T  P A R A M E T E R S  ***'
     105       WRITE (*,405)
     106       READ  (*,*)  AZIM
     107       WRITE (*,406)
     108       READ  (*,*)  ELEV
     109       WRITE (*,408)
     110       READ  (*,*)  ITRIM
     111       WRITE (*,409)
     112       READ  (*,*)  IDIV
     113       WRITE (*,411)
     114       READ  (*,199)  CTEXT
     115       WRITE (*,401)
     116       READ  (*,200)  ANSWER
     117
     118 C     ***** INITIALIZE PLOT88 *****
```

```
D Line# I      7                                  Microsoft FORTRAN77 V3.20 02/84
     119       IF  ((ANSWER .EQ. 'Y') .OR. (ANSWER .EQ. 'y'))  THEN
     120         CALL PLOTS(0,0,2)
     121       ELSE
     122         CALL PLOTS(0,99,99)
     123       ENDIF
     124
     125       CALL WINDOW(XLOL,YLOL,XUPR,YUPR)
     126
     127 C     ***** DRAW THE MESH SURFACE OF THE GRAPH *****
     128       CALL MESHS(Z,31,31,N,N,AZIM,ELEV,0.5,0.5,7.5,5.5,IDIV,0,
     129      *           3,IPROJ,1,ZLOW,3,ITRIM,MASK,VERTEX)
     130 C     ***** ANNOTATION OF THE GRAPH *****
     131       CALL SYMBOL(5.5,0.3,0.2,'AZIMUTH:   ',0.0,10)
     132       CALL NUMBER(999.0,999.0,0.2,AZIM,0.0,2)
     133       CALL SYMBOL(5.5,0.0,0.2,'ELEVATION:',0.0,10)
     134       CALL NUMBER(999.0,999.0,0.2,ELEV,0.0,2)
     135       DY = (Z(1,1)/90.0) * ELEV
     136       CALL P3D2D(1.0,1.0,Z(1,1)-DY,XR,YR)
     137       CALL SYMBOL(XR,YR,0.25,'*',0.0,1)
     138       CALL SYMBOL(1.0,0.1,0.2,'* = ORIGIN',0.0,10)
     139       CALL SYMBOL(1.0,6.75,0.25,CTEXT,0.0,20)
     140       CALL SYMBOL(6.0,6.5,0.2,'2-D DATA FIELD',0.0,14)
     141
     142 C     ***** OUTPUT THE GRAPH *****
     143       CALL PLOT(0.0,0.0,999)
     144       WRITE (*,412)
     145       READ  (*,200) ANSWER
     146       IF  ((ANSWER .EQ. 'Y') .OR. (ANSWER .EQ. 'y')) GOTO 15
     147
     148   18  WRITE(*,417)
     149       READ(*,200)  ANSWER
     150       IF  ((ANSWER .EQ. 'Y') .OR. (ANSWER .EQ. 'y')) THEN
     151 C       **** FILL 0's THE TWO DIMENTIONAL GRID OF CONTROL POINTS ****
     152         DO 106 I = 1,31
1    153           DO 106 J = 1,31
2    154             ZF(I,J) =0.0
2    155   106 CONTINUE
     156         ZFMAX = -9.9E30
     157         ZFMIN = 9.9E30
     158         DN = (N-1)/2.0
     159         P = 6.283185
     160         DO 107 I = 1,N
1    161           DO 107 J = 1,N
2    162             RLPART  = 0.0
2    163             IMGPART = 0.0
2    164             DO 108 L = 1,N
3    165               DO 108 K = 1,N
4    166                 R1(1) = COS(-P*(L-1)*(I-DN-1)/N)
4    167                 R1(2) = SIN(-P*(L-1)*(I-DN-1)/N)
4    168                 R2(1) = COS(-P*(K-1)*(J-DN-1)/N)
4    169                 R2(2) = SIN(-P*(K-1)*(J-DN-1)/N)
4    170                 RLPART  = RLPART  + Z(L,K)*(R1(1)*R2(1)
4    171      *                                     -R1(2)*R2(2))
4    172                 IMGPART = IMGPART + Z(L,K)*(R1(1)*R2(2)
4    173      *                                     +R1(2)*R2(1))
4    174   108         CONTINUE
2    175             ZF(I,J) = SQRT(RLPART**2 + IMGPART**2)
2    176             IF (ZF(I,J) .GT. ZFMAX) THEN
3    177               ZFMAX = ZF(I,J)
```

```
D Line# 1      7
2   178              ENDIF
2   179              IF (ZF(I,J) .LT. ZFMIN) THEN
2   180                ZFMIN = ZF(I,J)
2   181              ENDIF
2   182    107   CONTINUE
    183
    184 C          ***** OUTPUT THE ZF MATRIX *****
    185          WRITE (*,205) '***  FOURIER TRANSFORMATION  ',N,' X ',N,'  ***'
    186          WRITE (*,212)
    187          DO 109 I = 1,N
1   188             WRITE (*,300) (ZF(I,J), J = 1,N)
1   189             WRITE (*,210)
1   190    109 CONTINUE
    191          WRITE (*,213)
    192
    193          WRITE (*,418)
    194          READ  (*,200) ANSWER
    195          IF  ((ANSWER .NE. 'Y') .AND. (ANSWER .NE. 'y')) GOTO 16
    196
    197 C          ***** ASK THE PARAMETERS FOR THE GRAPH *****
    198    30    WRITE (*,210)
    199          WRITE (*,*) '***  E N T E R  P L O T  P A R A M E T E R S  ***'
    200          WRITE (*,405)
    201          READ  (*,*)  AZIM
    202          WRITE (*,406)
    203          READ  (*,*)  ELEV
    204          WRITE (*,408)
    205          READ  (*,*)  ITRIM
    206          WRITE (*,409)
    207          READ  (*,*)  IDIV
    208          WRITE (*,411)
    209          READ  (*,199)  CTEXT
    210          WRITE (*,401)
    211          READ  (*,200)  ANSWER
    212
    213 C          ***** INITIALIZE PLOT88 *****
    214          IF  ((ANSWER .EQ. 'Y') .OR. (ANSWER .EQ. 'y'))  THEN
    215            CALL PLOTS(0,0,2)
    216          ELSE
    217            CALL PLOTS(0,99,99)
    218          ENDIF
    219
    220          WRITE (*,420)
    221          READ  (*,200)  ANSWER
    222
    223          CALL WINDOW(XLOL,YLOL,XUPR,YUPR)
    224
    225          IF  ((ANSWER .EQ. 'Y') .OR. (ANSWER .EQ. 'y'))  THEN
    226            DLEV = (ZFMAX-ZFMIN)/FLOAT(N)
    227            CALL ZLEVEL(ZF,31,31,N,N,DLEV,ZLEV,N+1)
    228            DO 110 I = 1,N+1
1   229              LDIG(I) = 2
1   230              LWGT(I) = 1
1   231    110     CONTINUE
    232
    233 C          ***** DRAW THE CONTOUR MAP *****
    234            CALL ZCNTUR(ZF,31,31,N,N,0.5,0.5,7.5,5.5,ZLEV,LDIG,LWGT,
    235     *                  N+1,0.10,10)
    236            CALL SYMBOL(5.5,0.0,0.2,'CONTOUR MAP',0.0,11)
```

```
D Line# 1      7                                    Microsoft FORTRAN77 V3.20 02/84
     237          ELSE
     238 C          ***** DRAW THE MESH SURFACE OF THE GRAPH *****
     239            CALL MESHS(ZF,31,31,N,N,AZIM,ELEV,0.5,0.5,7.5,5.5,IDIV,0,
     240        *                   3,IPROJ,1,ZLOW,3,ITRIM,MASK,VERTEX)
     241 C          ***** ANNOTATION OF THE GRAPH *****
     242            CALL SYMBOL(5.5,0.3,0.2,'AZIMUTH:    ',0.0,10)
     243            CALL NUMBER(999.0,999.0,0.2,AZIM,0.0,2)
     244            CALL SYMBOL(5.5,0.0,0.2,'ELEVATION:',0.0,10)
     245            CALL NUMBER(999.0,999.0,0.2,ELEV,0.0,2)
     246            DY = (ZF(1,1)/90.0) * ELEV
     247            CALL P3D2D(1.0,1.0,ZF(1,1)-DY,XR,YR)
     248            CALL SYMBOL(XR,YR,0.25,'*',0.0,1)
     249            CALL SYMBOL(1.0,0.1,0.2,'* = ORIGIN',0.0,10)
     250          ENDIF
     251
     252          CALL SYMBOL(1.0,6.75,0.25,CTEXT,0.0,20)
     253          CALL SYMBOL(6.0,6.5,0.2,'2-D DFT',0.0,7)
     254
     255 C        ***** OUTPUT THE GRAPH *****
     256          CALL PLOT(0.0,0.0,999)
     257          WRITE (*,412)
     258          READ  (*,200) ANSWER
     259          IF  ((ANSWER .EQ. 'Y') .OR. (ANSWER .EQ. 'y')) GOTO 30
     260   16   ENDIF
     261        WRITE (*,413)
     262        READ  (*,200) ANSWER
     263        IF  ((ANSWER .EQ. 'Y') .OR. (ANSWER .EQ. 'y')) GOTO 10
     264        STOP
     265
     266   199 FORMAT(A20)
     267   200 FORMAT(A)
     268   205 FORMAT(/,20X,A25,I2,A3,I2,A8,/)
     269   210 FORMAT(/)
     270   211 FORMAT(/,A8,I2,A47)
     271   212 FORMAT(/,2X,'(AZIMUTH 320.0)',46X,'(AZIMUTH 230.0)',/)
     272   213 FORMAT(/,2X,'(AZIMUTH 050.0)',46X,'(AZIMUTH 140.0)',/)
     273   300 FORMAT(10(F7.2,1X))
     274   399 FORMAT(/,5X,A4,\)
     275   400 FORMAT(9X,\)
     276   401 FORMAT(/,5X,'SEND GRAPH TO THE PRINTER(Y or N): ',\)
     277   402 FORMAT(/,5X,'NUMBER OF ROWS/COLUMNS FOR R AND S(1 to 25): ',\)
     278   403 FORMAT(5X,A4,I2,A3,\)
     279   404 FORMAT(5X,A2,I2,A5,\)
     280   405 FORMAT(/,5X,'AZIMUTH(0.0 to 360.0 DEGREES): ',\)
     281   406 FORMAT(/,5X,'ELEVATION(90.0 to -90.0 DEGREES): ',\)
     282   408 FORMAT(/,5X,'TRIM(0=NO,1=Xs,2=Ys): ',\)
     283   409 FORMAT(/,5X,'2,4 OR 8 SUBGRIDS: ',\)
     284   411 FORMAT(/,5X,'TITLE OF GRAPH(UP TO 20 CHAR): ',\)
     285   412 FORMAT(/,5X,'DO YOU WANT TO CHANGE PARAMETERS? ',\)
     286   413 FORMAT(/,5X,'DO YOU WANT TO REPEAT THE PROCESS? ',\)
     287   417 FORMAT(/,5X,'DO YOU WANT FOURIER TRANSFORMATION ? ',\)
     288   418 FORMAT(/,5X,'DO YOU WANT TO MAKE GRAPH ? ',\)
     289   419 FORMAT(/,5X,'DO YOU WANT TO CORRECT ? ',\)
     290   420 FORMAT(/,5X,'DO YOU WANT CONTOUR MAP ? ',\)
     291        END

Name      Type         Offset P Class

A1        REAL           26
```

D Line# 1 7

A2	REAL	30	
A3	REAL	34	
A4	REAL	38	
ANSWER	CHAR*1	74	
AZIM	REAL	114	
B1	REAL	42	
B2	REAL	46	
C1	REAL	50	
C2	REAL	54	
COS			INTRINSIC
CTEXT	CHAR*20	126	
DLEV	REAL	216	
DN	REAL	166	
DY	REAL	146	
ELEV	REAL	118	
FLOAT			INTRINSIC
I	INTEGER*2	60	
IDIV	INTEGER*2	124	
IMGPAR	REAL	190	
IPROJ	INTEGER*2	22	
ITRIM	INTEGER*2	122	
J	INTEGER*2	86	
K	INTEGER*2	202	
L	INTEGER*2	194	
LDIG	INTEGER*2	6000	LARGE
LWGT	INTEGER*2	6062	LARGE
MASK	INTEGER*2	0	LARGE
N	INTEGER*2	58	
NRNG	INTEGER*2	24	
P	REAL	170	
R	REAL	0	LARGE
R1	REAL	0	LARGE
R2	REAL	8	LARGE
RLPART	REAL	186	
S	REAL	2500	LARGE
SIN			INTRINSIC
SQRT			INTRINSIC
U	REAL	76	
VERTEX	REAL	0	LARGE
XLOL	REAL	2	
XR	REAL	150	
XUPR	REAL	10	
YLOL	REAL	6	
YR	REAL	154	
YUPR	REAL	14	
Z	REAL	5000	LARGE
ZF	REAL	8844	LARGE
ZFMAX	REAL	158	
ZFMIN	REAL	162	
ZLEV	REAL	12688	LARGE
ZLOW	REAL	18	

Name	Type	Size	Class
MAIN			PROGRAM
MESHS			SUBROUTINE
NUMBER			SUBROUTINE

```
D Line# 1      7
P3D2D                          SUBROUTINE
PLOT                           SUBROUTINE
PLOTS                          SUBROUTINE
SYMBOL                         SUBROUTINE
WINDOW                         SUBROUTINE
ZCNTUR                         SUBROUTINE
ZLEVEL                         SUBROUTINE

 Pass One     No Errors Detected
             291 Source Lines

A>
```

APPENDIX B


```
D Line#  1      7                          Microsoft FORTRAN77 V3.20 82/8-
       1 $STORAGE:  2
       2 $PAGESIZE:59
       3 C     ********************************************************************
       4 C     *                                                                  *
       5 C     *  THE PURPOSE OF THIS PROGRAM IS  TO  COMPUTE  AND GRAPH THE     *
       6 C     *  FREQUENCY RESPONSE OF A 2-D DIGITAL FILTER.                     *
       7 C     *                                                                  *
       8 C     *                          EVANGELOS THEOFILOU                     *
       9 C     ********************************************************************
      10 C     PROGRAM 2D-DATA-FIELD
      11
      12 C     ***** VARIABLE DECLARATIONS *****
      13       REAL         A(7,7),B(7,7),R1(7,7,2),R2(7,7,2),
      14      *             RLPART,IMGPART,Z(51,51),
      15      *             VERTEX(16),ZLEV(51)
      16       INTEGER      MASK(3000),LDIG(51),LWGT(51)
      17       CHARACTER*1 ANSWER
      18       CHARACTER*20 CTEXT
      19
      20       DATA          XLOL/0.0/,YLOL/0.0/,XUPR/8.5/,YUPR/7.0/,
      21      *              ZLOW/1.0E35/,IPROJ/0/,NRNG/100/
      22
      23 C     ************  M A I N   P R O G R A M  *************
      24
      25   10  WRITE (*,401)
      26       READ  (*,*)   IT
      27       IF (IT .GT. 25) GOTO 10
      28       WRITE (*,402)
      29       READ  (*,*) K
      30       K = K + 1
      31       WRITE(*,*) '       ENTER VALUES OF COEFFICIENTS:'
      32       DO 100 I = 0,K-1
1     33         DO 100 J = 0,K-1
2     34           WRITE(*,404) 'B(',I,',',J,'): '
2     35           READ (*,*) B(I+1,J+1)
2     36   100 CONTINUE
      37
      38       DO 101 I = 0,K-1
1     39         DO 101 J = 0,K-1
2     40           WRITE(*,404) 'A(',I,',',J,'): '
2     41           READ (*,*) A(I+1,J+1)
2     42   101 CONTINUE
      43
      44       WRITE (*,419)
      45       READ  (*,200) ANSWER
      46       IF  ((ANSWER .EQ. 'Y') .OR. (ANSWER .EQ. 'y')) GOTO 10
      47
      48 C     **** FILL 0's THE TWO DIMENTIONAL GRID OF CONTROL POINTS ****
      49       DO 107 I = 1,51
1     50         DO 107 J = 1,51
2     51           Z(I,J) = 0.0
```

```
D Line# 1      7
3     52    107 CONTINUE
      53
      54        ZMIN = 9.9E20
      55        ZMAX = -9.9E20
      56        P  = 3.14159
      57        STEP = 2*P / (IT-1)
      58        W1 = -P - STEP
      59        L  = 0
      60        DO 102 I = 1,IT
1     61          W1 = W1 + STEP
1     62          W2 = -P - STEP
1     63          DO 103 J = 1,IT
2     64            L = L + 1
2     65            W2 = W2 + STEP
2     66            DO 104 M = 0,K-1
3     67              DO 104 N = 0,K-1
4     68                R1(M+1,N+1,1) = COS(-M * W1)
4     69                R1(M+1,N+1,2) = SIN(-M * W1)
4     70                R2(M+1,N+1,1) = COS(-N * W2)
4     71                R2(M+1,N+1,2) = SIN(-N * W2)
4     72    104       CONTINUE
2     73              RLNOM  = 0.0
2     74              IMGNOM = 0.0
2     75              RLDEN  = 0.0
2     76              IMGDEN = 0.0
2     77              DO 105 M = 0,K-1
3     78                DO 105 N = 0,K-1
4     79                  RLNOM  = RLNOM+B(M+1,N+1)*(R1(M+1,N+1,1)*R2(M+1,N+1,1)
4     80       *                                   - R1(M+1,N+1,2)*R2(M+1,N+1,2))
4     81                  IMGNOM = IMGNOM+B(M+1,N+1)*(R1(M+1,N+1,1)*R2(M+1,N+1,2)
4     82       *                                    + R2(M+1,N+1,1)*R1(M+1,N+1,2))
4     83                  RLDEN  = RLDEN+A(M+1,N+1)*(R1(M+1,N+1,1)*R2(M+1,N+1,1)
4     84       *                                   - R1(M+1,N+1,2)*R2(M+1,N+1,2))
4     85                  IMGDEN = IMGDEN+A(M+1,N+1)*(R1(M+1,N+1,1)*R2(M+1,N+1,2)
4     86       *                                    + R2(M+1,N+1,1)*R1(M+1,N+1,2))
4     87    105       CONTINUE
2     88              ELEMENT = SQRT(RLNOM**2 + IMGNOM**2) /
2     89       *                SQRT(RLDEN**2 + IMGDEN**2)
2     90              Z(I,J) = ELEMENT
2     91              IF (Z(I,J) .GT. ZMAX) THEN
2     92                ZMAX = Z(I,J)
2     93              ENDIF
2     94              IF (Z(I,J) .LT. ZMIN) THEN
2     95                ZMIN = Z(I,J)
2     96              ENDIF
2     97    103     CONTINUE
1     98    102   CONTINUE
      99
    100 C       ***** OUTPUT THE Z MATRIX *****
    101         WRITE (*,205) '*****  Z  M A T R I X  ',IT,' X ',IT,'  *****'
    102         WRITE (*,212)
```

```
D Line# I      7                                  Microsoft FORTRAN77 V3.30 03/84
     103          DO 106 I = 1,IT
1    104            WRITE (*,300) (Z(I,J), J = 1,IT)
1    105            WRITE (*,210)
1    106     106  CONTINUE
     107          WRITE (*,213)
     108
     109          WRITE (*,418)
     110          READ  (*,200) ANSWER
     111          IF  ((ANSWER .NE. 'Y') .AND. (ANSWER .NE. 'y')) GOTO 15
     112
     113
     114 C        ***** ASK THE PARAMETERS FOR THE GRAPH *****
     115     20   WRITE (*,210)
     116          WRITE (*,*) '****  E N T E R  P L O T  P A R A M E T E R S  ****'
     117          WRITE (*,410)
     118          READ  (*,*)  AZIM
     119          WRITE (*,411)
     120          READ  (*,*)  ELEV
     121          WRITE (*,413)
     122          READ  (*,*)  ITRIM
     123          WRITE (*,414)
     124          READ  (*,*)  IDIV
     125          WRITE (*,415)
     126          READ  (*,199)  CTEXT
     127          WRITE (*,451)
     128          READ  (*,200)  ANSWER
     129
     130 C        ***** INITIALIZE PLOT88 *****
     131          IF  ((ANSWER .EQ. 'Y') .OR. (ANSWER .EQ. 'y'))  THEN
     132            CALL PLOTS(0,0,2)
     133          ELSE
     134            CALL PLOTS(0,99,99)
     135          ENDIF
     136
     137          WRITE (*,420)
     138          READ (*,200) ANSWER
     139
     140          CALL WINDOW(XLOL,YLOL,XUPR,YUPR)
     141
     142          IF  ((ANSWER .EQ. 'Y') .OR. (ANSWER .EQ. 'y'))  THEN
     143            DLEV = (ZMAX-ZMIN)/FLOAT(IT)
     144            CALL ZLEVEL(Z,51,51,IT,IT,DLEV,ZLEV,IT+1)
     145            DO 108 I = 1,IT+1
1    146              LDIG(I) = 2
1    147              LWGT(I) = 1
1    148     108    CONTINUE
     149 C          ***** DRAW THE CONTOUR MAP *****
     150            CALL ZCNTUR(Z,51,51,IT,IT,0.5,0.5,3.25,6.5,ZLEV,LDIG,LWGT,
     151           *            IT+1,0.10,10)
     152            CALL SYMBOL(5.5,0.0,0.2,'CONTOUR MAP',0.0,11)
     153          ELSE
```

```
D Line# 1      7                                    Microsoft FORTRAN77 V3.20 02/84
     154 C        ***** DRAW THE MESH SURFACE OF THE GRAPH *****
     155          CALL MESHS(Z,51,51,IT,IT,AZIM,ELEV,0.5,0.5,8.25,6.5,IDIV,0,
     156         *           3,IPROJ,1,ZLOW,3,ITRIM,MASK,VERTEX)
     157 C        ***** ANNOTATION OF THE GRAPH *****
     158          CALL SYMBOL(5.5,0.3,0.2,'AZIMUTH:    ',0.0,10)
     159          CALL NUMBER(999.0,999.0,0.2,AZIM,0.0,2)
     160          CALL SYMBOL(5.5,0.0,0.2,'ELEVATION:',0.0,10)
     161          CALL NUMBER(999.0,999.0,0.2,ELEV,0.0,2)
     162          DY = (Z(1,1)/90.0) * ELEV
     163          CALL P3D2D(1.0,1.0,Z(1,1)-DY,XR,YR)
     164          CALL SYMBOL(XR,YR,0.25,'*',0.0,1)
     165          CALL SYMBOL(1.0,0.1,0.2,'* = ORIGIN',0.0,10)
     166        ENDIF
     167
     168        CALL SYMBOL(1.0,6.75,0.25,CTEXT,0.0,80)
     169        CALL SYMBOL(6.0,6.5,0.2,'3-D DATA FIELD',0.0,14)
     170
     171 C      ***** OUTPUT THE GRAPH *****
     172        CALL PLOT(0.0,0.0,999)
     173
     174        WRITE (*,416)
     175        READ  (*,300) ANSWER
     176        IF  ((ANSWER .EQ. 'Y') .OR. (ANSWER .EQ. 'y')) GOTO 20
     177  15    WRITE (*,417)
     178        READ  (*,300) ANSWER
     179        IF  ((ANSWER .EQ. 'Y') .OR. (ANSWER .EQ. 'y')) GOTO 10
     180        STOP
     181
     182   199 FORMAT(A20)
     183   300 FORMAT(A)
     184   305 FORMAT(/,20X,A25,I2,A3,I2,A8,/)
     185   310 FORMAT()
     186   311 FORMAT(/,5X,A60)
     187   312 FORMAT(/,2X,'(AZIMUTH 320.0)',46X,'(AZIMUTH 230.0)',/)
     188   313 FORMAT(/,2X,'(AZIMUTH 050.0)',46X,'(AZIMUTH 140.0)',/)
     189   300 FORMAT(10(F7.2,1X))
     190   400 FORMAT(9X,\)
     191   451 FORMAT(/,5X,'SEND GRAPH TO THE PRINTER(Y or N): ',\)
     192   401 FORMAT(/,5X,'DIMENSION OF OUTPUT MATRIX(1 to 25): ',\)
     193   402 FORMAT(/,5X,'ORDER OF TRANSFER FUNCTION(0 to 4): ',\)
     194   404 FORMAT(5X,A2,I1,A,I1,A3,\)
     195   410 FORMAT(/,5X,'AZIMUTH(0.0 to 360.0 DEGREES): ',\)
     196   411 FORMAT(/,5X,'ELEVATION(90.0 to -90.0 DEGREES): ',\)
     197   413 FORMAT(/,5X,'TRIM(0=NO,1=Xs,2=Ys): ',\)
     198   414 FORMAT(/,5X,'2,4 OR 8 SUBGRIDS: ',\)
     199   415 FORMAT(/,5X,'TITLE OF GRAPH(UP TO 80 CHAR): ',\)
     200   416 FORMAT(/,5X,'DO YOU WANT TO CHANGE PARAMETERS ? ',\)
     201   417 FORMAT(/,5X,'DO YOU WANT TO REPEAT THE PROCESS ? ',\)
     202   418 FORMAT(/,5X,'DO YOU WANT TO MAKE GRAPH ? ',\)
     203   419 FORMAT(/,5X,'DO YOU WANT TO CORRECT ? ',\)
     204   420 FORMAT(/,5X,'DO YOU WANT CONTOUR MAP ? ',\)
```

D Line# 1 7 Microsoft FORTRAN77 V5.10 08/04
305 END

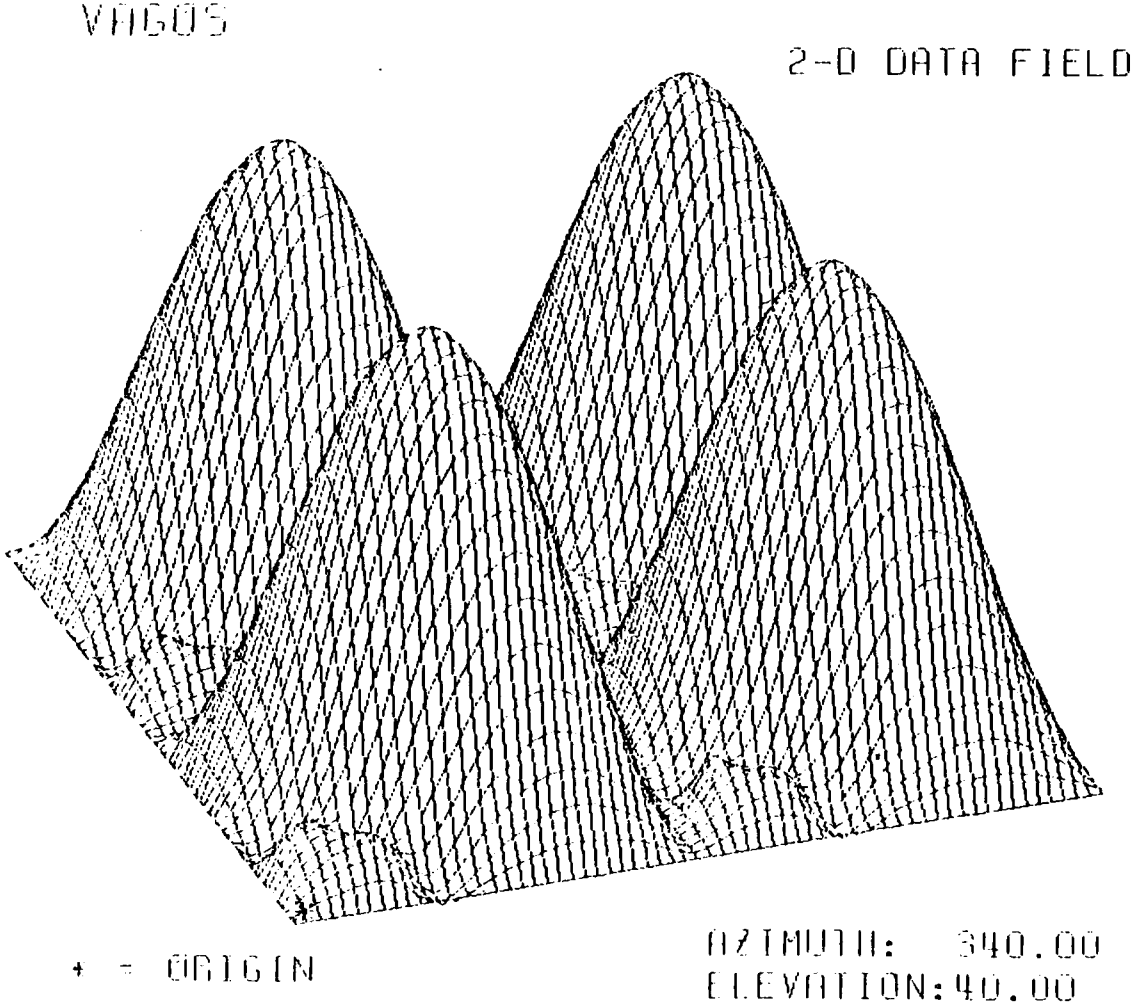
Name	Type	Offset	P	Class
A	REAL	3		
ANSWER	CHAR*1	18110		
AZIM	REAL	18202		
B	REAL	198		
COS				INTRINSIC
CTEXT	CHAR*20	18214		
DLEV	REAL	18234		
DY	REAL	18244		
ELEMEN	REAL	18190		
ELEV	REAL	18206		
FLOAT				INTRINSIC
I	INTEGER*2	18082		
IDIV	INTEGER*2	18212		
IMGDEN	INTEGER*2	18176		
IMGNOM	INTEGER*2	18170		
IMGPAR	REAL	*****		
IPROJ	INTEGER*2	18074		
IT	INTEGER*2	18078		
ITRIM	INTEGER*2	18210		
J	INTEGER*2	18090		
K	INTEGER*2	18080		
L	INTEGER*2	18132		
LDIG	INTEGER*2	17850		
LWGT	INTEGER*2	17952		
M	INTEGER*2	18150		
MASK	INTEGER*2	11850		
N	INTEGER*2	18158		
NRNG	INTEGER*2	18076		
P	REAL	18120		
R1	REAL	394		
R2	REAL	786		
RLDEN	REAL	18172		
RLNOM	REAL	18166		
RLPART	REAL	*****		
SIN				INTRINSIC
SQRT				INTRINSIC
STEP	REAL	18124		
VERTEX	REAL	11786		
W1	REAL	18128		
W2	REAL	18140		
XLOL	REAL	18054		
XR	REAL	18248		
XUPR	REAL	18062		
YLOL	REAL	18058		
YR	REAL	18252		
YUPR	REAL	18066		
Z	REAL	1178		

D Line# 1 7

ZLEV	REAL	11582
ZLOW	REAL	18070
ZMAX	REAL	18116
ZMIN	REAL	18112

Name	Type	Size	Class
MAIN			PROGRAM
MESHS			SUBROUTINE
NUMBER			SUBROUTINE
P3D2D			SUBROUTINE
PLOT			SUBROUTINE
PLOTS			SUBROUTINE
SYMBOL			SUBROUTINE
WINDOW			SUBROUTINE
ZCNTUR			SUBROUTINE
ZLEVEL			SUBROUTINE

Pass One No Errors Detected
205 Source Lines


VAGOS
2-D DATA FIELD
+ = ORIGIN
AZIMUTH: 340.00
ELEVATION: 40.00

APPENDIX C

D Line# 1 7 Microsoft FORTRAN77 V3.30 02/84

```
      1 $STORAGE:  2
      2 $PAGESIZE:58
      3
      4 C     ***************************************************************
      5 C     *                                                             *
      6 C     *  THE PURPOSE OF THIS PROGRAM IS  TO  COMPUTE  AND GRAPH THE *
      7 C     *  EQUATIONS OF ROBERT P.ROESSER IN THE "DISCRETE STATE-SPACE  *
      8 C     *  MODEL FOR LINEAR IMAGE PROCESSING". IT TRANSFORMS ALSO THE *
      9 C     *  OUTPUT MATRIX Y ACCORDING TO FOURIER ANALYSIS.             *
     10 C     *                                                             *
     11 C     *                        EVANGELOS THEOFILOU                  *
     12 C     ***************************************************************
     13 C     PROGRAM 2D-DATA-FIELD
     14
     15 C     ***** VARIABLE DECLARATIONS *****
     16       REAL         R1(26,26),R2(26,26),S1(26,26),S2(26,26),
     17      *             FR1(2),FR2(2),TRM(4,4),IV(4),OV(4),IMGPART
     18       CHARACTER*1 ANSWER
     19
     20 C     **** VARIABLE DECLARATIONS FOR PLOT88 *****
     21       CHARACTER*20 CTEXT
     22       COMMON       /WORK /Z(26,26),ZF(26,26),ZLEV(26),LDIG(26),
     23      *             LWGT(26),MASK(3000),VERTEX(16)
     24
     25       DATA         XLOL/0.0/,YLOL/0.0/,XUPR/8.5/,YUPR/7.0/,
     26      *             ZLOW/1.0E35/,IPROJ/0/,NRNG/100/
     27
     28 C     *************  M A I N   P R O G R A M  *************
     29
     30 C     ***** ASK THE REQUIRED VALUES FOR THE MODEL *****
     31    10 WRITE (*,403)
     32       READ  (*,*) KK
     33       IF ((KK .LT. 3) .OR. (KK .GT. 25)) GOTO 10
     34
     35       DO 100 I = 1,KK+1
1    36         DO 100 J = 1,KK+1
2    37           R1(I,J) = 0.0
2    38           R2(I,J) = 0.0
2    39           S1(I,J) = 0.0
2    40           S2(I,J) = 0.0
2    41   100 CONTINUE
     42
     43       DO 101 I = 1,4
1    44         DO 101 J = 1,4
2    45           TRM(I,J) = 0.0
2    46   101 CONTINUE
     47
     48       DO 102 I = 1,4
1    49         IV(I) = 0.0
1    50         OV(I) = 0.0
1    51   102 CONTINUE
```

D Line# 1 7 Microsoft FORTRAN77 V3.30 02/84

```
      52
      53        WRITE (*,211) 'ENTER INITIAL CONDITIONS FOR HORIZONTAL R1(#.#)'
      54        DO 103 I = 1,KK
1     55          WRITE (*,404) 'R1(1,',I,'): '
1     56          READ  (*,*) R1(1,I)
1     57    103 CONTINUE
      58
      59        WRITE (*,211) 'ENTER INITIAL CONDITIONS FOR HORIZONTAL R2(#.#)'
      60        DO 104 I = 1,KK
1     61          WRITE (*,404) 'R2(1,',I,'): '
1     62          READ  (*,*) R2(1,I)
1     63    104 CONTINUE
      64
      65        WRITE (*,211) 'ENTER INITIAL CONDITIONS FOR VERTICAL S1(#.#) '
      66        DO 105 I = 1,KK
1     67          WRITE (*,405) 'S1(',I,',1): '
1     68          READ  (*,*) S1(I,1)
1     69    105 CONTINUE
      70
      71        WRITE (*,211) 'ENTER INITIAL CONDITIONS FOR VERTICAL S2(#.#) '
      72        DO 106 I = 1,KK
1     73          WRITE (*,405) 'S2(',I,',1): '
1     74          READ  (*,*) S2(I,1)
1     75    106 CONTINUE
      76
      77        WRITE (*,211) 'ENTER VALUES FOR THE OUTPUT VECTOR(#.#)       '
      78        OV(1) = 1
      79        WRITE (*,409) 'b01: '
      80        READ  (*,*) OV(3)
      81        WRITE (*,409) 'a01: '
      82        READ  (*,*) OV(4)
      83
      84        WRITE (*,211) 'ENTER ELEMENTS OF THE TRANSITION MATRIX(#.#)  '
      85        TRM(1,2) = 1
      86        TRM(4,1) = 1
      87        WRITE (*,409) 'a10: '
      88        READ  (*,*) TRM(1,1)
      89        WRITE (*,409) 'a20: '
      90        READ  (*,*) TRM(2,1)
      91        WRITE (*,409) 'b11: '
      92        READ  (*,*) TEMP
      93        TRM(1,3) = TEMP + OV(3)*TRM(1,1)
      94        WRITE (*,409) 'a11: '
      95        READ  (*,*) TEMP
      96        TRM(1,4) = TEMP + OV(4)*TRM(1,1)
      97        WRITE (*,409) 'b21: '
      98        READ  (*,*) TEMP
      99        TRM(2,3) = TEMP + OV(3)*TRM(2,1)
     100        WRITE (*,409) 'a21: '
     101        READ  (*,*) TEMP
     102        TRM(2,4) = TEMP + OV(4)*TRM(2,1)
```

```
D Line# 1      7                                   Microsoft FORTRAN77 V3.20 02/84
     103       TRM(4,3) = OV(3)
     104       TRM(4,4) = OV(4)
     105
     106       WRITE (*,211) 'ENTER VALUES FOR THE INPUT VECTOR(#.#)'
     107       IV(3) = 1
     108       WRITE (*,409) 'b00: '
     109       READ (*,*) IV(4)
     110       WRITE (*,409) 'b10: '
     111       READ (*,*) TEMP
     112       IV(1) = TEMP + IV(4)*TRM(1,1)
     113       IV(2) = IV(4)*TRM(2,1)
     114
     115       U = 1.0
     116       DO 107 I = 1,KK
1    117         DO 107 J = 1,KK
2    118         R1(I+1,J) = TRM(1,1)*R1(I,J) + R2(I,J) + TRM(1,3)*S1(I,J) +
2    119      *                                           TRM(1,4)*S2(I,J) + IV(1)*U
2    120         R2(I+1,J) = TRM(2,1)*R1(I,J) + TRM(2,3)*S1(I,J) +
2    121      *                                       TRM(2,4)*S2(I,J) + IV(2)*U
2    122         S1(I,J+1) = U
2    123         S2(I,J+1) = R1(I,J) + OV(3)*S1(I,J) + OV(4)*S2(I,J) + IV(4)*U
2    124         U = 0.0
2    125   107 CONTINUE
     126
     127       WRITE (*,205) '***** INPUT VECTOR *****'
     128       WRITE (*,300) (IV(I),I = 1,4)
     129
     130       WRITE (*,205) '***** OUTPUT VECTOR *****'
     131       WRITE (*,300) (OV(I),I = 1,4)
     132
     133       WRITE (*,205) '***** TRANSITION MATRIX *****'
     134       DO 108 I = 1,4
1    135         WRITE (*,300) (TRM(I,J),J = 1,4)
1    136         WRITE (*,210)
1    137   108 CONTINUE
     138
     139 C     **** FILL 0's THE TWO DIMENTIONAL GRID OF CONTROL POINTS ****
     140       DO 109 I = 1,26
1    141         DO 109 J = 1,26
2    142           Z(I,J) =0.0
2    143   109 CONTINUE
     144
     145       DO 110 I = 1,KK
1    146         DO 110 J = 1,KK
2    147           Z(I,J) = R1(I,J) + OV(3)*S1(I,J) + OV(4)*S2(I,J)
2    148   110 CONTINUE
     149
     150       WRITE (*,205) '***** R1  M A T R I X  ',KK,' X ',KK,' *****'
     151       DO 111 I = 1,KK
1    152         WRITE (*,300) (R1(I,J), J = 1,KK)
1    153         WRITE (*,210)
```

```
D Line# 1      7                                  Microsoft FORTRAN77 V3.20 02/84
1    154    111 CONTINUE
     155
     156        WRITE (*,205) '***** R2  M A T R I X  ',KK,' X ',KK,' *****'
     157        DO 112 I = 1,KK
1    158          WRITE (*,300) (R2(I,J), J = 1,KK)
1    159          WRITE (*,210)
1    160    112 CONTINUE
     161
     162        WRITE (*,205) '***** S1  M A T R I X  ',KK,' X ',KK,' *****'
     163        DO 113 I = 1,KK
1    164          WRITE (*,300) (S1(I,J), J = 1,KK)
1    165          WRITE (*,210)
1    166    113 CONTINUE
     167
     168        WRITE (*,205) '***** S2  M A T R I X  ',KK,' X ',KK,' *****'
     169        DO 114 I = 1,KK
1    170          WRITE (*,300) (S2(I,J), J = 1,KK)
1    171          WRITE (*,210)
1    172    114 CONTINUE
     173
     174 C      ***** OUTPUT THE Y MATRIX *****
     175        WRITE (*,205) '*****  Z  M A T R I X  ',KK,' X ',KK,'  *****'
     176        WRITE (*,212)
     177        DO 115 I = 1,KK
1    178          WRITE (*,300) (Z(I,J), J = 1,KK)
1    179          WRITE (*,210)
1    180    115 CONTINUE
     181        WRITE (*,213)
     182
     183        WRITE(*,419)
     184        READ  (*,200) ANSWER
     185        IF  ((ANSWER .NE. 'Y') .AND. (ANSWER .NE. 'y')) GOTO 21
     186
     187 C      ***** ASK THE PARAMETERS FOR THE GRAPH *****
     188    20  WRITE (*,210)
     189        WRITE (*,*) '****  E N T E R  P L O T  P A R A M E T E R S  ****'
     190        WRITE (*,410)
     191        READ  (*,*)  AZIM
     192        WRITE (*,411)
     193        READ  (*,*)  ELEV
     194        WRITE (*,413)
     195        READ  (*,*)  ITRIM
     196        WRITE (*,414)
     197        READ  (*,*)  IDIV
     198        WRITE (*,415)
     199        READ  (*,199)  CTEXT
     200        WRITE (*,451)
     201        READ  (*,200)  ANSWER
     202
     203 C      ***** INITIALIZE PLOT88 *****
     204        IF  ((ANSWER .EQ. 'Y') .OR. (ANSWER .EQ. 'y'))  THEN
```

```
D Line# 1      7                                   Microsoft FORTRAN77 V3.20 02/84
     205          CALL PLOTS(0,0,2)
     206        ELSE
     207          CALL PLOTS(0,99,99)
     208        ENDIF
     209
     210        CALL WINDOW(XLOL,YLOL,XUPR,YUPR)
     211
     212 C      ***** DRAW THE MESH SURFACE OF THE GRAPH *****
     213        CALL MESHS(Z,26,26,KK,KK,AZIM,ELEV,0.5,0.5,8.25,6.5,IDIV,0,
     214       *           3,IPROJ,1,ZLOW,3,ITRIM,MASK,VERTEX)
     215
     216 C      ***** ANNOTATION OF THE GRAPH *****
     217        CALL SYMBOL(1.0,6.75,0.25,CTEXT,0.0,20)
     218        CALL SYMBOL(6.0,6.5,0.2,'2-D DATA FIELD',0.0,14)
     219        CALL SYMBOL(5.5,0.3,0.2,'AZIMUTH:    ',0.0,10)
     220        CALL NUMBER(999.0,999.0,0.2,AZIM,0.0,2)
     221        CALL SYMBOL(5.5,0.0,0.2,'ELEVATION:',0.0,10)
     222        CALL NUMBER(999.0,999.0,0.2,ELEV,0.0,2)
     223        DY = (Z(1,1)/90.0) * ELEV
     224        CALL P3D2D(1.0,1.0,Z(1,1)-DY,XR,YR)
     225        CALL SYMBOL(XR,YR,0.25,'*',0.0,1)
     226        CALL SYMBOL(1.0,0.1,0.2,'* = ORIGIN',0.0,10)
     227
     228 C      ***** OUTPUT THE GRAPH *****
     229        CALL PLOT(0.0,0.0,999)
     230        WRITE (*,416)
     231        READ  (*,200) ANSWER
     232        IF  ((ANSWER .EQ. 'Y') .OR. (ANSWER .EQ. 'y')) GOTO 20
     233
     234  21    WRITE(*,418)
     235        READ(*,200)  ANSWER
     236        IF  ((ANSWER .EQ. 'Y') .OR. (ANSWER .EQ. 'y')) THEN
     237
     238 C        **** FILL 0's THE TWO DIMENTIONAL GRID OF CONTROL POINTS ****
     239          DO 116 I = 1,26
1    240            DO 116 J = 1,26
2    241              ZF(I,J) =0.0
2    242   116      CONTINUE
     243
     244          ZFMAX = -9.9E20
     245          ZFMIN = 9.9E20
     246          DK = (KK - 1) / 2.0
     247          P = 3.141592
     248          DO 117 M = 1,KK
1    249            DO 117 N = 1,KK
2    250              RLPART  = 0.0
2    251              IMGPART = 0.0
2    252              DO 118 L = 1,KK
3    253                DO 118 K = 1,KK
4    254                  FR1(1) = COS(-2*P*(L-1)*(M-DK-1)/KK)
4    255                  FR1(2) = SIN(-2*P*(L-1)*(M-DK-1)/KK)
```

```
4   256                FR2(1) = COS(-2*P*(K-1)*(N-DK-1)/KK)
4   257                FR2(2) = SIN(-2*P*(K-1)*(N-DK-1)/KK)
4   258                RLPART  = RLPART  + Z(L,K)*(FR1(1)*FR2(1)
4   259       *                                   -FR1(2)*FR2(2))
4   260                IMGPART = IMGPART + Z(L,K)*(FR1(1)*FR2(2)
4   261       *                                   +FR1(2)*FR2(1))
4   262    118       CONTINUE
2   263              ZF(M,N) = SQRT(RLPART**2 + IMGPART**2)
2   264              IF (ZF(M,N) .GT. ZFMAX) THEN
2   265              ZFMAX = ZF(M,N)
2   266            ENDIF
2   267            IF (ZF(M,N) .LT. ZFMIN) THEN
2   268              ZFMIN = ZF(M,N)
2   269            ENDIF
2   270    117   CONTINUE
    271
    272 C         ***** OUTPUT THE ZF MATRIX *****
    273           WRITE (*,205) '*** FOURIER TRANSFORMATION  ',KK,' X ',KK,' ***'
    274           WRITE (*,212)
    275           DO 119 I = 1,KK
1   276             WRITE (*,300) (ZF(I,J), J = 1,KK)
1   277             WRITE (*,210)
1   278    119    CONTINUE
    279           WRITE (*,213)
    280
    281           WRITE(*,419)
    282           READ  (*,200) ANSWER
    283           IF  ((ANSWER .NE. 'Y') .AND. (ANSWER .NE. 'y')) GOTO 22
    284
    285 C         ***** ASK THE PARAMETERS FOR THE GRAPH *****
    286    30     WRITE (*,210)
    287           WRITE (*,*) '***  E N T E R  P L O T  P A R A M E T E R S  ***'
    288           WRITE (*,410)
    289           READ  (*,*)  AZIM
    290           WRITE (*,411)
    291           READ  (*,*)  ELEV
    292           WRITE (*,413)
    293           READ  (*,*)  ITRIM
    294           WRITE (*,414)
    295           READ  (*,*)  IDIV
    296           WRITE (*,415)
    297           READ  (*,199)  CTEXT
    298           WRITE (*,451)
    299           READ  (*,200)  ANSWER
    300
    301 C         ***** INITIALIZE PLOT88 *****
    302           IF  ((ANSWER .EQ. 'Y') .OR. (ANSWER .EQ. 'y'))  THEN
    303             CALL PLOTS(0,0,2)
    304           ELSE
    305             CALL PLOTS(0,99,99)
    306           ENDIF
```

D Line# 1 7 Microsoft FORTRAN77 V3.30 02/84

```
    307
    308          WRITE (*,420)
    309          READ (*,200) ANSWER
    310
    311          CALL WINDOW(XLOL,YLOL,XUPR,YUPR)
    312
    313          IF ((ANSWER .EQ. 'Y') .OR. (ANSWER .EQ. 'y')) THEN
    314            DLEV = (ZFMAX-ZFMIN)/FLOAT(KK)
    315            CALL ZLEVEL(ZF,26,26,KK,KK,DLEV,ZLEV,KK+1)
    316            DO 136 I = 1,KK+1
1   317              LDIG(I) = 2
1   318              LWGT(I) = 1
1   319   136      CONTINUE
    320            CALL ZCNTUR(ZF,26,26,KK,KK,0.5,0.5,8.25,6.5,ZLEV,LDIG,LWGT,
    321       *                KK+1,0.10,10)
    322            CALL SYMBOL(5.5,0.0,0.2,'CONTOUR MAP',0.0,11)
    323          ELSE
    324 C          ***** DRAW THE MESH SURFACE OF THE GRAPH *****
    325            CALL MESHS(ZF,26,26,KK,KK,AZIM,ELEV,0.5,0.5,8.25,6.5,IDIV,0,
    326       *               3,IPROJ,1,ZLOW,3,ITRIM,MASK,VERTEX)
    327
    328 C          ***** ANNOTATION OF THE GRAPH *****
    329            CALL SYMBOL(5.5,0.3,0.2,'AZIMUTH:   ',0.0,10)
    330            CALL NUMBER(999.0,999.0,0.2,AZIM,0.0,2)
    331            CALL SYMBOL(5.5,0.0,0.2,'ELEVATION:',0.0,10)
    332            CALL NUMBER(999.0,999.0,0.2,ELEV,0.0,2)
    333            DY = (ZF(1,1)/90.0) * ELEV
    334            CALL P3D2D(1.0,1.0,ZF(1,1)-DY,XR,YR)
    335            CALL SYMBOL(XR,YR,0.25,'*',0.0,1)
    336            CALL SYMBOL(1.0,0.1,0.2,'* = ORIGIN',0.0,10)
    337          ENDIF
    338          CALL SYMBOL(1.0,6.75,0.25,CTEXT,0.0,20)
    339          CALL SYMBOL(6.0,6.5,0.2,'2-D DFT',0.0,7)
    340
    341 C        ***** OUTPUT THE GRAPH *****
    342          CALL PLOT(0.0,0.0,999)
    343          WRITE (*,416)
    344          READ  (*,200) ANSWER
    345          IF  ((ANSWER .EQ. 'Y') .OR. (ANSWER .EQ. 'y')) GOTO 30
    346   22   ENDIF
    347        WRITE (*,417)
    348        READ  (*,200) ANSWER
    349        IF  ((ANSWER .EQ. 'Y') .OR. (ANSWER .EQ. 'y')) GOTO 10
    350        STOP
    351
    352   199 FORMAT(A20)
    353   200 FORMAT(A)
    354   205 FORMAT(/,18X,A29,I2,A3,I2,A8,/)
    355   210 FORMAT()
    356   211 FORMAT(/,5X,A56)
    357   212 FORMAT(/,2X,'(AZIMUTH 320.0)',46X,'(AZIMUTH 230.0)',/)
```

```
D Line# 1      7                                    Microsoft FORTRAN77 V3.30 08/34
    358   213 FORMAT(/,2X,'(AZIMUTH 050.0)',46X,'(AZIMUTH 140.0)',/)
    359   300 FORMAT(10(F7.2,1X))
    360   400 FORMAT(9X,\)
    361   403 FORMAT(/,5X,'DIMENSION OF OUTPUT(K=1to20): ',\)
    362   404 FORMAT(5X,A5,I2,A3,\)
    363   405 FORMAT(5X,A3,I2,A5,\)
    364   406 FORMAT(5X,A2,I2,A4,\)
    365   407 FORMAT(5X,A2,I2,I2,A3,\)
    366   408 FORMAT(5X,A3,I2,A3,\)
    367   409 FORMAT(5X,A5,\)
    368   410 FORMAT(/,5X,'AZIMUTH(0.0 to 360.0 DEGREES): ',\)
    369   411 FORMAT(/,5X,'ELEVATION(90.0 to -90.0 DEGREES): ',\)
    370   413 FORMAT(/,5X,'TRIM(0=NO,1=Xs,2=Ys): ',\)
    371   414 FORMAT(/,5X,'2,4 OR 8 SUBGRIDS: ',\)
    372   415 FORMAT(/,5X,'TITLE OF GRAPH(UP TO 20 CHAR): ',\)
    373   416 FORMAT(/,5X,'DO YOU WANT TO CHANGE PARAMETERS? ',\)
    374   417 FORMAT(/,5X,'DO YOU WANT TO REPEAT THE PROCESS? ',\)
    375   418 FORMAT(/,5X,'DO YOU WANT FOURIER TRANSFORMATION ? ',\)
    376   419 FORMAT(/,5X,'DO YOU WANT TO MAKE GRAPH ? ',\)
    377   420 FORMAT(/,5X,'DO YOU WANT CONTOUR MAP ? ',\)
    378   451 FORMAT(/,5X,'SEND GRAPH TO THE PRINTER(Y or N): ',\)
    379       END

Name     Type         Offset P Class

ANSWER   CHAR*1        11060
AZIM     REAL          11062
COS                            INTRINSIC
CTEXT    CHAR*20       11074
DK       REAL          11114
DLEV     REAL          11168
DY       REAL          11094
ELEV     REAL          11066
FLOAT                          INTRINSIC
FR1      REAL          10882
FR2      REAL          10890
I        INTEGER*2     10956
IDIV     INTEGER*2     11072
IMGPAR   REAL          11142
IPROJ    INTEGER*2     10950
ITRIM    INTEGER*2     11070
IV       REAL          10898
J        INTEGER*2     10964
K        INTEGER*2     11154
KK       INTEGER*2     10954
L        INTEGER*2     11146
LDIG     INTEGER*2      5512   /WORK  /
LWGT     INTEGER*2      5564   /WORK  /
M        INTEGER*2     11122
MASK     INTEGER*2      5616   /WORK  /
N        INTEGER*2     11130
```

D Line# 1 7

NRNG	INTEGER*2	10952	
OV	REAL	10914	
P	REAL	11118	
R1	REAL	2	
R2	REAL	2706	
RLPART	REAL	11138	
S1	REAL	5410	
S2	REAL	8114	
SIN			INTRINSIC
SQRT			INTRINSIC
TEMP	REAL	10996	
TRM	REAL	10818	
U	REAL	11000	
VERTEX	REAL	11616	/WORK /
XLOL	REAL	10930	
XR	REAL	11098	
XUPR	REAL	10938	
YLOL	REAL	10934	
YR	REAL	11102	
YUPR	REAL	10942	
Z	REAL	0	/WORK /
ZF	REAL	2704	/WORK /
ZFMAX	REAL	11106	
ZFMIN	REAL	11110	
ZLEV	REAL	5408	/WORK /
ZLOW	REAL	10946	

Name	Type	Size	Class
MAIN			PROGRAM
MESHS			SUBROUTINE
NUMBER			SUBROUTINE
P3D2D			SUBROUTINE
PLOT			SUBROUTINE
PLOTS			SUBROUTINE
SYMBOL			SUBROUTINE
WINDOW			SUBROUTINE
WORK		11680	COMMON
ZCNTUR			SUBROUTINE
ZLEVEL			SUBROUTINE

Pass One No Errors Detected
379 Source Lines

APPENDIX D


```
D Line#  1      7                          Microsoft FORTRAN77 V3.30 02/84
       1 $STORAGE:  2
       2 $PAGESIZE:58
       3
       4 C      ************************************************************
       5 C      *                                                          *
       6 C      *  THE PURPOSE OF THIS PROGRAM IS  TO  COMPUTE  AND GRAPH THE    *
       7 C      *  EQUATIONS OF ROBERT P.ROESSER IN THE "DISCRETE STATE-SPACE     *
       8 C      *  MODEL FOR LINEAR IMAGE PROCESSING". IT TRANSFORMS ALSO THE     *
       9 C      *  OUTPUT MATRIX Y ACCORDING TO FOURIER ANALYSIS.                 *
      10 C      *                                                          *
      11 C      *                          EVANGELOS THEOFILOU             *
      12 C      ************************************************************
      13 C      PROGRAM 2D-DATA-FIELD
      14
      15 C      ***** VARIABLE DECLARATIONS *****
      16        REAL          R(26,26,4),S1(26,26,4),S2(26,26,4),
      17       *              R1(2),R2(2),TRM(12,12),IV(12),OV(12),IMGPART
      18        CHARACTER*1 ANSWER
      19
      20 C      **** VARIABLE DECLARATIONS FOR PLOT88 *****
      21        CHARACTER*20 CTEXT
      22        COMMON        /WORK /Z(26,26),ZF(26,26),ZLEV(26),LDIG(26),
      23       *              LWGT(26),MASK(3000),VERTEX(16)
      24
      25        DATA          XLOL/0.0/,YLOL/0.0/,XUPR/8.5/,YUPR/7.0/,
      26       *              ZLOW/1.0E35/,IPROJ/0/,NRNG/100/
      27
      28 C      *************  M A I N   P R O G R A M  *************
      29
      30 C      ***** ASK THE REQUIRED VALUES FOR THE MODEL *****
      31     10 WRITE (*,401)
      32        READ  (*,*) N
      33        IF ((N .LT. 1) .OR. (N .GT. 4)) GOTO 10
      34     2  WRITE (*,402)
      35        READ  (*,*) M
      36        IF ((M .LT. 1) .OR. (M .GT. 4)) GOTO 2
      37     3  WRITE (*,403)
      38        READ  (*,*) KK
      39        IF (KK .GT. 25) GOTO 3
      40
      41        DO 100 I = 1,KK+1
1     42          DO 100 J = 1,KK+1
2     43            DO 100 L = 1,N
3     44              R(I,J,L) = 0.0
3     45              S1(I,J,L) = 0.0
3     46              S2(I,J,L) = 0.0
3     47    100 CONTINUE
      48
      49        DO 101 I = 1,N+2*M
1     50          DO 101 J = 1,N+2*M
2     51            TRM(I,J) = 0.0
```

```
3     52   101 CONTINUE
      53       DO 102 I = 1,N+2*M
1     54         IV(I) = 0.0
1     55         OV(I) = 0.0
1     56   102 CONTINUE
      57
      58       WRITE (*,211) 'ENTER INITIAL CONDITIONS FOR HORIZONTAL R(#.#)'
      59       DO 103 I = 1,KK
1     60         DO 103 J = 1,N
2     61           WRITE (*,404) 'R',J,'(1,',I,'): '
2     62           READ  (*,*) R(1,I,J)
2     63   103 CONTINUE
      64
      65       WRITE (*,211) 'ENTER INITIAL CONDITIONS FOR VERTICAL S1(#.#) '
      66       DO 104 I = 1,KK
1     67         DO 104 J = 1,M
2     68           WRITE (*,405) 'S1(',J,')(',I,',1): '
2     69           READ  (*,*) S1(I,1,J)
2     70   104 CONTINUE
      71
      72       WRITE (*,211) 'ENTER INITIAL CONDITIONS FOR VERTICAL S2(#.#) '
      73       DO 105 I = 1,KK
1     74         DO 105 J = 1,M
2     75           WRITE (*,405) 'S2(',J,')(',I,',1): '
2     76           READ  (*,*) S2(I,1,J)
2     77   105 CONTINUE
      78
      79       WRITE (*,211) 'ENTER VALUES FOR THE INPUT VECTOR(#.#)        '
      80       IV(1) = 1.0
      81       DO 106 I = 1,M
1     82         WRITE (*,408) 'a(0',I,'): '
1     83         READ (*,*) IV(N+I)
1     84         TRM(N+I,N+1) = -IV(N+I)
1     85   106 CONTINUE
      86       DO 107 I = 1,M
1     87         WRITE (*,408) 'b(0',I,'): '
1     88         READ (*,*) IV(N+M+I)
1     89         TRM(N+M+I,N+1) = -IV(N+M+I)
1     90   107 CONTINUE
      91       DO 108 I = 1,M-1
1     92         TRM(N+I,N+I+1) = 1.0
1     93   108 CONTINUE
      94       DO 109 I = 1,M-1
1     95         TRM(N+M+I,N+M+I+1) = 1.0
1     96   109 CONTINUE
      97
      98       WRITE (*,211) 'ENTER ELEMENTS OF THE TRANSITION MATRIX(#.#)  '
      99       DO 110 I = 1,N
1    100         WRITE (*,406) 'a(',I,'0): '
1    101         READ  (*,*) TEMP
1    102         TRM(1,I) = -TEMP
```

```
D Line# 1      7                                   Microsoft FORTRAN77 V3.20 82/84
1    103   110 CONTINUE
      104       TRM(1,N+1) = -1.0
      105       DO 111 I = 2,N
1     106         TRM(I,I-1) = 1.0
1     107   111 CONTINUE
      108       DO 112 I = 1,M
1     109         DO 112 J = 1,N
2     110           WRITE (*,407) 'a(',J,I,'): '
2     111           READ  (*,*) TEMP1
2     112           TRM(I+N,J) = TEMP1 + TRM(1,J) * IV(N+I)
2     113   112 CONTINUE
      114       DO 113 I = 1,M
1     115         DO 113 J = 1,N
2     116           WRITE (*,407) 'b(',J,I,'): '
2     117           READ  (*,*)  TEMP1
2     118           TRM(I+N+M,J) = TEMP1 + TRM(1,J) * IV(N+M+I)
2     119   113 CONTINUE
      120
      121       WRITE (*,211) 'ENTER VALUES FOR THE OUTPUT VECTOR(#.#)        '
      122       WRITE (*,409) 'b(00): '
      123       READ  (*,*) TEMP
      124       OV(N+1) = -TEMP
      125       OV(N+M+1) = 1.0
      126       DO 114 I = 1,N
1     127         WRITE (*,406) 'b(',I,'0): '
1     128         READ  (*,*) TEMP1
1     129         OV(I) = TEMP1 + TRM(1,I) * TEMP
1     130   114 CONTINUE
      131
      132       U = 1.0
      133       DO 115 I = 1,KK
1     134         DO 115 J = 1,KK
2     135           DO 116 II = 1,N+2*M
3     136
3     137             IF  (II .LE. N)  THEN
3     138               DO 117 JJ = 1,N+2*M
4     139                 IF  (JJ .LE. N)  THEN
4     140                   R(I+1,J,II)=R(I+1,J,II)+TRM(II,JJ)*R(I,J,JJ)
4     141                 ENDIF
4     142                 IF  ((JJ .GT. N) .AND. (JJ .LE. N+M))  THEN
4     143                   R(I+1,J,II)=R(I+1,J,II)+TRM(II,JJ)*S1(I,J,JJ-N)
4     144                 ENDIF
4     145   117           CONTINUE
3     146               R(I+1,J,II)=R(I+1,J,II) + IV(II) * U
3     147             ENDIF
3     148
3     149             IF  ((II .GT. N) .AND. (II .LE. N+M))  THEN
3     150               DO 118 JJ = 1,N+2*M
4     151                 IF  (JJ .LE. N)  THEN
4     152                   S1(I,J+1,II-N) = S1(I,J+1,II-N) + TRM(II,JJ) *
4     153      *                                                 R(I,J,JJ)
```

```
D Line# 1      7                                   Microsoft FORTRAN77 V3.20 02/84
4    154                ENDIF
4    155                IF ((JJ .GT. N).AND.(JJ .LE. N+M)) THEN
4    156                  S1(I,J+1,II-N) = S1(I,J+1,II-N) + TRM(II,JJ)*
4    157       *                                              S1(I,J,JJ-N)
4    158                ENDIF
4    159    118       CONTINUE
3    160              S1(I,J+1,II-N) = S1(I,J+1,II-N) + IV(II) * U
3    161            ENDIF
3    162
3    163            IF (II .GT. N+M) THEN
3    164              DO 119 JJ = 1,N+2*M
4    165                IF  (JJ .LE. N)  THEN
4    166                  S2(I,J+1,II-N-M) = S2(I,J+1,II-N-M) + TRM(II,JJ)
4    167       *                                                * R(I,J,JJ)
4    168                ENDIF
4    169                IF  ((JJ .GT. N) .AND. (JJ .LE. N+M))  THEN
4    170                  S2(I,J+1,II-N-M) = S2(I,J+1,II-N-M) + TRM(II,JJ)
4    171       *                                               * S1(I,J,JJ-N)
4    172                ENDIF
4    173                IF  (JJ .GT. N+M)  THEN
4    174                  S2(I,J+1,II-N-M) = S2(I,J+1,II-N-M) + TRM(II,JJ)
4    175       *                                              * S2(I,J,JJ-N-M)
4    176                ENDIF
4    177    119       CONTINUE
3    178              S2(I,J+1,II-N-M) = S2(I,J+1,II-N-M) + IV(II) * U
3    179            ENDIF
3    180    116   CONTINUE
2    181        U = 0.0
2    182    115 CONTINUE
     183
     184        WRITE (*,205) '***** INPUT VECTOR *****'
     185        WRITE (*,300) (IV(I),I = 1,N+2*M)
     186
     187        WRITE (*,205) '***** OUTPUT VECTOR *****'
     188        WRITE (*,300) (OV(I),I = 1,N+2*M)
     189
     190        WRITE (*,205) '***** TRANSITION MATRIX *****'
     191        DO 120 I = 1,N+2*M
1    192          WRITE (*,300) (TRM(I,J),J = 1,N+2*M)
1    193          WRITE (*,210)
1    194    120 CONTINUE
     195
     196 C      **** FILL 0's THE TWO DIMENTIONAL GRID OF CONTROL POINTS ****
     197        DO 121 I = 1,26
1    198          DO 121 J = 1,26
2    199            Z(I,J) =0.0
2    200    121 CONTINUE
     201
     202        DO 122 I = 1,KK
1    203          DO 122 J = 1,KK
2    204            DO 123 LL = 1,N+2*M
```

```
3   205              IF (LL .LE. N) THEN
3   206                Z(I,J) = Z(I,J) + OV(LL) * R(I,J,LL)
3   207              ENDIF
3   208              IF ((LL .GT. N).AND.(LL .LE. N+M)) THEN
3   209                Z(I,J) = Z(I,J) + OV(LL) * S1(I,J,LL-N)
3   210              ENDIF
3   211              IF (LL .GT. N+M)  THEN
3   212                Z(I,J) = Z(I,J) + OV(LL) * S2(I,J,LL-N-M)
3   213              ENDIF
3   214    123     CONTINUE
2   215    122 CONTINUE
    216
    217        WRITE (*,205) '***** R  M A T R I X  ',KK,' X ',KK,' *****'
    218        DO 124 I = 1,KK
1   219          DO 125 L = 1,N
2   220            WRITE (*,300) (R(I,J,L), J = 1,KK)
2   221    125   CONTINUE
1   222          WRITE (*,210)
1   223    124 CONTINUE
    224
    225        WRITE (*,205) '***** S1  M A T R I X  ',KK,' X ',KK,' *****'
    226        DO 126 I = 1,KK
1   227          DO 127 L = 1,M
2   228            WRITE (*,300) (S1(I,J,L), J = 1,KK)
2   229    127   CONTINUE
1   230          WRITE (*,210)
1   231    126 CONTINUE
    232
    233        WRITE (*,205) '***** S2  M A T R I X  ',KK,' X ',KK,' *****'
    234        DO 128 I = 1,KK
1   235          DO 129 L = 1,M
2   236            WRITE (*,300) (S2(I,J,L), J = 1,KK)
2   237    129   CONTINUE
1   238          WRITE (*,210)
1   239    128 CONTINUE
    240
    241 C      ***** OUTPUT THE Z MATRIX *****
    242        WRITE (*,205) '*****  Z  M A T R I X  ',KK,' X ',KK,'  *****'
    243        WRITE (*,212)
    244        DO 130 I = 1,KK
1   245          WRITE (*,300) (Z(I,J), J = 1,KK)
1   246          WRITE (*,210)
1   247    130 CONTINUE
    248        WRITE (*,213)
    249
    250        WRITE(*,419)
    251        READ  (*,200) ANSWER
    252        IF  ((ANSWER .NE. 'Y') .AND. (ANSWER .NE. 'y')) GOTO 21
    253
    254 C      ***** ASK THE PARAMETERS FOR THE GRAPH *****
    255   20   WRITE (*,210)
```

```
D Lines# 1      7                                   Microsoft FORTRAN77 V3.20 02/84
     256        WRITE (*,*) '****  E N T E R  P L O T  P A R A M E T E R S  ****'
     257        WRITE (*,410)
     258        READ  (*,*)  AZIM
     259        WRITE (*,411)
     260        READ  (*,*)  ELEV
     261        WRITE (*,413)
     262        READ  (*,*)  ITRIM
     263        WRITE (*,414)
     264        READ  (*,*)  IDIV
     265        WRITE (*,415)
     266        READ  (*,199)  CTEXT
     267        WRITE (*,451)
     268        READ  (*,200)  ANSWER
     269
     270 C      ***** INITIALIZE PLOT88 *****
     271        IF  ((ANSWER .EQ. 'Y') .OR. (ANSWER .EQ. 'y'))  THEN
     272           CALL PLOTS(0,0,2)
     273        ELSE
     274           CALL PLOTS(0,99,99)
     275        ENDIF
     276
     277        CALL WINDOW(XLOL,YLOL,XUPR,YUPR)
     278
     279 C      ***** DRAW THE MESH SURFACE OF THE GRAPH *****
     280        CALL MESHS(Z,26.26,KK,KK,AZIM,ELEV,0.5,0.5,8.25,6.5,IDIV,0.
     281       *           3,IPROJ,1,ZLOW,3,ITRIM,MASK,VERTEX)
     282
     283 C      ***** ANNOTATION OF THE GRAPH *****
     284        CALL SYMBOL(1.0,6.75,0.25,CTEXT,0.0,20)
     285        CALL SYMBOL(6.0,6.5,0.2,'2-D DATA FIELD',0.0,14)
     286        CALL SYMBOL(5.5,0.3,0.2,'AZIMUTH:   ',0.0,10)
     287        CALL NUMBER(999.0,999.0,0.2,AZIM,0.0,2)
     288        CALL SYMBOL(5.5,0.0,0.2,'ELEVATION:',0.0,10)
     289        CALL NUMBER(999.0,999.0,0.2,ELEV,0.0,2)
     290        DY = (Z(1,1)/90.0) * ELEV
     291        CALL P3D2D(1.0,1.0,Z(1,1)-DY,XR,YR)
     292        CALL SYMBOL(XR,YR,0.25,'*',0.0,1)
     293        CALL SYMBOL(1.0,0.1,0.2,'* = ORIGIN',0.0,10)
     294
     295 C      ***** OUTPUT THE GRAPH *****
     296        CALL PLOT(0.0,0.0,999)
     297        WRITE (*,416)
     298        READ  (*,200) ANSWER
     299        IF  ((ANSWER .EQ. 'Y') .OR. (ANSWER .EQ. 'y')) GOTO 20
     300
     301  21    WRITE(*,418)
     302        READ(*,200)  ANSWER
     303        IF  ((ANSWER .EQ. 'Y') .OR. (ANSWER .EQ. 'y')) THEN
     304
     305 C        **** FILL 0's THE TWO DIMENTIONAL GRID OF CONTROL POINTS ****
     306          DO 132 I = 1,26
```

```
1   307            DO 132 J = 1,26
2   308              ZF(I,J) =0.0
2   309    132   CONTINUE
    310
    311          ZFMAX = -9.9E20
    312          ZFMIN = 9.9E20
    313          DK = (KK - 1) / 2.0
    314          P = 3.141592
    315          DO 133 M = 1,KK
1   316            DO 133 N = 1,KK
2   317              RLPART  = 0.0
2   318              IMGPART = 0.0
2   319              DO 134 L = 1,KK
3   320                DO 134 K = 1,KK
4   321                  R1(1) = COS(-2*P*(L-1)*(M-DK-1)/KK)
4   322                  R1(2) = SIN(-2*P*(L-1)*(M-DK-1)/KK)
4   323                  R2(1) = COS(-2*P*(K-1)*(N-DK-1)/KK)
4   324                  R2(2) = SIN(-2*P*(K-1)*(N-DK-1)/KK)
4   325                  RLPART  = RLPART  + Z(L,K)*(R1(1)*R2(1)
4   326       *                                     -R1(2)*R2(2))
4   327                  IMGPART = IMGPART + Z(L,K)*(R1(1)*R2(2)
4   328       *                                     +R1(2)*R2(1))
4   329    134         CONTINUE
2   330              ZF(M,N) = SQRT(RLPART**2 + IMGPART**2)
2   331              IF (ZF(M,N) .GT. ZFMAX) THEN
2   332              ZFMAX = ZF(M,N)
2   333            ENDIF
2   334            IF (ZF(M,N) .LT. ZFMIN) THEN
2   335              ZFMIN = ZF(M,N)
2   336            ENDIF
2   337    133   CONTINUE
    338
    339 C        ***** OUTPUT THE ZF MATRIX *****
    340          WRITE (*,305) '*** FOURIER TRANSFORMATION  ',KK,' X ',KK,' ***'
    341          WRITE (*,212)
    342          DO 135 I = 1,KK
1   343             WRITE (*,300) (ZF(I,J), J = 1,KK)
1   344             WRITE (*,210)
1   345    135   CONTINUE
    346          WRITE (*,213)
    347
    348          WRITE(*,419)
    349          READ  (*,200) ANSWER
    350          IF  ((ANSWER .NE. 'Y') .AND. (ANSWER .NE. 'y')) GOTO 22
    351
    352 C        ***** ASK THE PARAMETERS FOR THE GRAPH *****
    353    30    WRITE (*,210)
    354          WRITE (*,*) '***  E N T E R  P L O T  P A R A M E T E R S  ***'
    355          WRITE (*,410)
    356          READ  (*,*)  AZIM
    357          WRITE (*,411)
```

```
D Line# 1      7                                   Microsoft FORTRAN77 V3.20 02/84
     358          READ  (*,*)  ELEV
     359          WRITE (*,413)
     360          READ  (*,*)  ITRIM
     361          WRITE (*,414)
     362          READ  (*,*)  IDIV
     363          WRITE (*,415)
     364          READ  (*,199)  CTEXT
     365          WRITE (*,451)
     366          READ  (*,200)  ANSWER
     367
     368 C        ***** INITIALIZE PLOT88 *****
     369          IF  ((ANSWER .EQ. 'Y') .OR. (ANSWER .EQ. 'y'))  THEN
     370            CALL PLOTS(0,0,2)
     371          ELSE
     372            CALL PLOTS(0,99,99)
     373          ENDIF
     374
     375          WRITE (*,420)
     376          READ (*,200) ANSWER
     377
     378          CALL WINDOW(XLOL,YLOL,XUPR,YUPR)
     379
     380          IF ((ANSWER .EQ. 'Y') .OR. (ANSWER .EQ. 'y')) THEN
     381            DLEV = (ZFMAX-ZFMIN)/FLOAT(KK)
     382            CALL ZLEVEL(ZF,26,26,KK,KK,DLEV,ZLEV,KK+1)
     383            DO 136 I = 1,KK+1
1    384              LDIG(I) = 2
1    385              LWGT(I) = 1
1    386    136     CONTINUE
     387            CALL ZCNTUR(ZF,26,26,KK,KK,0.5,0.5,8.25,6.5,ZLEV,LDIG,LWGT,
     388           *                  KK+1,0.10,10)
     389            CALL SYMBOL(5.5,0.0,0.2,'CONTOUR MAP',0.0,11)
     390          ELSE
     391 C          ***** DRAW THE MESH SURFACE OF THE GRAPH *****
     392            CALL MESHS(ZF,26,26,KK,KK,AZIM,ELEV,0.5,0.5,8.25,6.5,IDIV,0,
     393           *                3,IPROJ,1,ZLOW,3,ITRIM,MASK,VERTEX)
     394
     395 C          ***** ANNOTATION OF THE GRAPH *****
     396            CALL SYMBOL(5.5,0.3,0.2,'AZIMUTH:   ',0.0,10)
     397            CALL NUMBER(999.0,999.0,0.2,AZIM,0.0,2)
     398            CALL SYMBOL(5.5,0.0,0.2,'ELEVATION:',0.0,10)
     399            CALL NUMBER(999.0,999.0,0.2,ELEV,0.0,2)
     400            DY = (ZF(1,1)/90.0) * ELEV
     401            CALL P3D2D(1.0,1.0,ZF(1,1)-DY,XR,YR)
     402            CALL SYMBOL(XR,YR,0.25,'*',0.0,1)
     403            CALL SYMBOL(1.0,0.1,0.2,'* = ORIGIN',0.0,10)
     404          ENDIF
     405          CALL SYMBOL(1.0,6.75,0.25,CTEXT,0.0,20)
     406          CALL SYMBOL(6.0,6.5,0.2,'2-D DFT',0.0,7)
     407
     408 C        ***** OUTPUT THE GRAPH *****
```

```
D Line# 1      7                                    Microsoft FORTRAN77 V3.20 02/84
     409         CALL PLOT(0.0,0.0,999)
     410         WRITE (*,416)
     411         READ  (*,200) ANSWER
     412         IF  ((ANSWER .EQ. 'Y') .OR. (ANSWER .EQ. 'y')) GOTO 30
     413  32     ENDIF
     414        WRITE (*,417)
     415        READ  (*,200) ANSWER
     416        IF  ((ANSWER .EQ. 'Y') .OR. (ANSWER .EQ. 'y')) GOTO 10
     417        STOP
     418
     419   199 FORMAT(A20)
     420   200 FORMAT(A)
     421   205 FORMAT(/,18X,A29,I2,A3,I2,A8,/)
     422   210 FORMAT()
     423   211 FORMAT(/,5X,A46)
     424   212 FORMAT(/,2X,'(AZIMUTH 320.0)',46X,'(AZIMUTH 230.0)',/)
     425   213 FORMAT(/,2X,'(AZIMUTH 050.0)',46X,'(AZIMUTH 140.0)',/)
     426   300 FORMAT(10(F7.2,1X))
     427   400 FORMAT(9X,\)
     428   451 FORMAT(/,5X,'SEND GRAPH TO THE PRINTER(Y or N): ',\)
     429   401 FORMAT(/,5X,'NUMBER OF HORIZONTAL STATES(N=1to4): ',\)
     430   402 FORMAT(/,5X,'NUMBER OF VERTICAL STATES(M=1to4): ',\)
     431   403 FORMAT(/,5X,'DIMENSION OF OUTPUT(1to25): ',\)
     432   404 FORMAT(5X,A1,I2,A3,I2,A3,\)
     433   405 FORMAT(5X,A3,I2,A2,I2,A5,\)
     434   406 FORMAT(5X,A2,I2,A4,\)
     435   407 FORMAT(5X,A2,I2,I2,A3,\)
     436   408 FORMAT(5X,A3,I2,A3,\)
     437   409 FORMAT(5X,A8,\)
     438   410 FORMAT(/,5X,'AZIMUTH(0.0 to 360.0 DEGREES): ',\)
     439   411 FORMAT(/,5X,'ELEVATION(90.0 to -90.0 DEGREES): ',\)
     440   413 FORMAT(/,5X,'TRIM(0=NO,1=Xs,2=Ys): ',\)
     441   414 FORMAT(/,5X,'2,4 OR 8 SUBGRIDS: ',\)
     442   415 FORMAT(/,5X,'TITLE OF GRAPH(UP TO 30 CHAR): ',\)
     443   416 FORMAT(/,5X,'DO YOU WANT TO CHANGE PARAMETERS? ',\)
     444   417 FORMAT(/,5X,'DO YOU WANT TO REPEAT THE PROCESS? ',\)
     445   418 FORMAT(/,5X,'DO YOU WANT FOURIER TRANSFORMATION ? ',\)
     446   419 FORMAT(/,5X,'DO YOU WANT TO MAKE GRAPH ? ',\)
     447   420 FORMAT(/,5X,'DO YOU WANT CONTOUR MAP ? ',\)
     448         END

Name     Type         Offset P Class

ANSWER CHAR*1          33434
AZIM   REAL            33436
COS                           INTRINSIC
CTEXT  CHAR*20         33448
DK     REAL            33488
DLEV   REAL            33536
DY     REAL            33468
ELEV   REAL            33440
```

D Line# 1 7

FLOAT			INTRINSIC
I	INTEGER*2	33168	
IDIV	INTEGER*2	33446	
II	INTEGER*2	33336	
IMGPAR	REAL	33512	
IPROJ	INTEGER*2	33158	
ITRIM	INTEGER*2	33444	
IV	REAL	33042	
J	INTEGER*2	33176	
JJ	INTEGER*2	33344	
K	INTEGER*2	33522	
KK	INTEGER*2	33166	
L	INTEGER*2	33184	
LDIG	INTEGER*2	5512	/WORK /
LL	INTEGER*2	33384	
LWGT	INTEGER*2	5564	/WORK /
M	INTEGER*2	33164	
MASK	INTEGER*2	5616	/WORK /
N	INTEGER*2	33162	
NRNG	INTEGER*2	33160	
OV	REAL	33090	
P	REAL	33498	
R	REAL	2	
R1	REAL	33026	
R2	REAL	33034	
RLPART	REAL	33508	
S1	REAL	10818	
S2	REAL	21634	
SIN			INTRINSIC
SQRT			INTRINSIC
TEMP	REAL	33276	
TEMP1	REAL	33298	
TRM	REAL	32450	
U	REAL	33320	
VERTEX	REAL	11616	/WORK /
XLOL	REAL	33138	
XR	REAL	33472	
XUPR	REAL	33146	
YLOL	REAL	33142	
YR	REAL	33476	
YUPR	REAL	33150	
Z	REAL	0	/WORK /
ZF	REAL	2704	/WORK /
ZFMAX	REAL	33480	
ZFMIN	REAL	33484	
ZLEV	REAL	5408	/WORK /
ZLOW	REAL	33154	

D Line# 1 7 Microsoft FORTRAN77 V3.20 02/84

Name	Type	Size	Class
MAIN			PROGRAM
MESHS			SUBROUTINE
NUMBER			SUBROUTINE
P3D2D			SUBROUTINE
PLOT			SUBROUTINE
PLOTS			SUBROUTINE
SYMBOL			SUBROUTINE
WINDOW			SUBROUTINE
WORK		11680	COMMON
ZCNTUR			SUBROUTINE
ZLEVEL			SUBROUTINE

Pass One No Errors Detected
448 Source Lines

APPENDIX E


```
D Line#  1     7                                  Microsoft FORTRAN77 V3.20 08/84
      1 $LARGE
      2 $STORAGE:  2
      3 $PAGESIZE:58
      4
      5 C     ************************************************************************
      6 C     *                                                                      *
      7 C     *  THE PURPOSE OF THIS PROGRAM IS  TO  CODE THE 1-D (DISCRETE          *
      8 C     *  TIME) SYSTEM TO A 2-D SPACIAL SYSTEM.                               *
      9 C     *                                                                      *
     10 C     *                          EVANGELOS THEOFILOU                         *
     11 C     ************************************************************************
     12 C     PROGRAM 2D-DATA-FIELD
     13
     14 C     ***** VARIABLE DECLARATIONS *****
     15       REAL          R(25,625),S(25,625),R1(2),R2(2),TRM(50,50),IV(50),
     16      *              IMGPART
     17       CHARACTER*1 ANSWER
     18
     19 C     **** VARIABLE DECLARATIONS FOR PLOT88 *****
     20       CHARACTER*20 CTEXT
     21       COMMON        /WORK /Z(26,26),ZF(26,26),X(630),Y(630),ZLEV(26),
     22      *                     LDIG(26),LWGT(26),MASK(3000),VERTEX(16)
     23
     24       DATA          XLOL/0.0/,YLOL/0.0/,XUPR/8.5/,YUPR/7.0/,
     25      *              ZLOW/1.0E35/,IPROJ/0/,NRNG/100/
     26
     27 C     *************  M A I N   P R O G R A M  *************
     28
     29 C     ***** ASK THE REQUIRED VALUES FOR THE MODEL *****
     30
     31    2  WRITE (*,403)
     32       READ  (*,*) N
     33       IF ((N .LT. 3) .OR. (N .GT.35)) GOTO 2
     34    3  WRITE (*,404)
     35       READ  (*,*) M
     36       IF ((M .LT. 2) .OR. (M .GT. 25)) GOTO 3
     37
     38       WRITE (*,401)
     39       READ (*,200) ANSWER
     40       IF  ((ANSWER .EQ. 'Y') .OR. (ANSWER .EQ. 'y')) THEN
     41 C       **** FILL 0's THE TWO DIMENTIONAL GRID OF CONTROL POINTS ****
     42         DO 96 I = 1,26
  1  43           DO 96 J = 1,26
  2  44             Z(I,J) =0.0
  2  45    96     CONTINUE
     46
     47 C       **** ENTER VALUES FOR Y MATRIX ****
     48         DO 97 I = 1,M*N
  1  49           WRITE(*,408) 'Y(',I,'): '
  1  50           READ (*,*) R(1,I)
  1  51    97     CONTINUE
```

```
D Line# I      7                                  Microsoft FORTRAN77 V3.20 02/84
      52        ENDIF
      53        IF  ((ANSWER .EQ. 'Y') .OR. (ANSWER .EQ. 'y'))  GOTO 4
      54
      55        WRITE (*,402)
      56        READ (*,200) ANSWER
      57        IF  ((ANSWER .EQ. 'Y') .OR. (ANSWER .EQ. 'y')) THEN
      58 C        **** INITIALIZE THE TRANSITION MATRIX ****
      59          DO 98 I = 1,M+N
1     60            DO 98 J = 1,M+N
2     61              TRM(I,J) =0.0
2     62     98     CONTINUE
      63
      64 C        **** ENTER VALUES FOR TRANSITION MATRIX ****
      65          DO 99 I = 1,M+N
1     66            DO 99 J = 1,M+N
2     67              WRITE(*,407) 'T(',I,',',J,'): '
2     68              READ (*,*) TRM(I,J)
2     69     99     CONTINUE
      70        ENDIF
      71
      72 C      ***** INITIALIZE R AND S ARRAYS *****
      73        DO 100 I = 1,25
1     74          DO 100 J = 1,625
2     75              R(I,J) = 0.0
2     76              S(I,J) = 0.0
2     77    100 CONTINUE
      78
      79 C      ***** INITIALIZE INPUT VECTOR *****
      80        DO 101 I = 1,50
1     81          IV(I) = 0.0
1     82    101 CONTINUE
      83
      84        WRITE (*,211) 'ENTER INITIAL CONDITIONS FOR HORIZONTAL R#'
      85        DO 102 I = 1,M
1     86            WRITE (*,405) 'R',I,': '
1     87            READ  (*,*) R(I,1)
1     88    102 CONTINUE
      89
      90        WRITE (*,211) 'ENTER INITIAL CONDITIONS FOR VERTICAL S#'
      91        DO 103 I = 1,N
1     92            WRITE (*,405) 'S',I,': '
1     93            READ  (*,*) S(I,1)
1     94    103 CONTINUE
      95
      96        WRITE (*,211) 'ENTER VALUES FOR THE INPUT VECTOR'
      97        IV(1) = 1.0
      98        WRITE (*,406) 'a01: '
      99        READ (*,*) IV(M+1)
     100
     101        IF  ((ANSWER .EQ. 'Y') .OR. (ANSWER .EQ. 'y'))  GOTO 5
     102 C      ***** INITIALIZE TRANSITION MATRIX *****
```

```
D Line# 1      7                                  Microsoft FORTRAN77 V3.30 08/84
     103       DO 104 I = 1,25
1    104         DO 104 J = 1,25
2    105           TRM(I,J) = 0.0
2    106   104 CONTINUE
     107       WRITE (*,211) 'ENTER ELEMENTS OF THE TRANSITION MATRIX'
     108       TRM(M+1,M+N) = -IV(M+1)
     109       WRITE (*,406) 'a10: '
     110       READ  (*,*) TEMP
     111       TRM(1,M) = -TEMP
     112       TRM(1,M+N) = -1.0
     113       WRITE (*,406) 'a11: '
     114       READ (*,*) TEMP
     115       TRM(M+1,M) = TEMP - TRM(1,M) * TRM(M+1,M+N)
     116
     117       DO 105 I = 2,M
1    118         TRM(I,I-1) = 1.0
1    119   105 CONTINUE
     120
     121       DO 106 I = 2+M,M+N
1    122         TRM (I,I-1) = 1.0
1    123   106 CONTINUE
     124
     125    5  U = 1.0
     126       DO 107 I = 1,N*M
1    127         DO 108 J = 1,M+N
2    128           IF (J .LE. M) THEN
2    129             DO 109 JJ = 1,M+N
3    130               IF (JJ .LE. M) R(J,I+1) = R(J,I+1) +
3    131     *                                   R(JJ,I)*TRM(J,JJ)
3    132               IF (JJ .GT. M) R(J,I+1) = R(J,I+1) +
3    133     *                                   S(JJ-M,I)*TRM(J,JJ)
3    134   109         CONTINUE
2    135             R(J,I+1) = R(J,I+1) + IV(J)*U
2    136           ENDIF
2    137
2    138           IF (J .GT. M) THEN
2    139             DO 110 JJ = 1,M+N
3    140               IF (JJ .LE. M) S(J-M,I+1) = S(J-M,I+1) +
3    141     *                                   R(JJ,I)*TRM(J,JJ)
3    142               IF (JJ .GT. M) S(J-M,I+1) = S(J-M,I+1) +
3    143     *                                   S(JJ-M,I)*TRM(J,JJ)
3    144   110         CONTINUE
2    145             S(J-M,I+1) = S(J-M,I+1) + IV(J)*U
2    146           ENDIF
2    147   108   CONTINUE
1    148         U = 0.0
1    149   107 CONTINUE
     150
     151       WRITE (*,211) '***** INPUT VECTOR *****'
     152       WRITE (*,300) (IV(I),I = 1,M+N)
     153
```

```
D Line# 1      7                                  Microsoft FORTRAN77 V3.20 08/84
     154       WRITE (*,211) '***** TRANSITION MATRIX *****'
     155       DO 111 I = 1,M+N
1    156         WRITE (*,300) (TRM(I,J),J = 1,M+N)
1    157         WRITE (*,210)
1    158   111 CONTINUE
     159
     160       WRITE (*,211) '***** H O R I Z O N T A L  S T A T E S  R *****'
     161       DO 112 I = 1,M*N
1    162         WRITE (*,300) (R(J,I), J = 1,M)
1    163   112 CONTINUE
     164
     165       WRITE (*,211) '  ***** V E R T I C A L  S T A T E S  S *****'
     166       DO 113 I = 1,M*N
1    167         WRITE (*,300) (S(J,I), J = 1,N)
1    168   113 CONTINUE
     169
     170 C     **** FILL 0's THE TWO DIMENTIONAL GRID OF CONTROL POINTS ****
     171       DO 114 I = 1,26
1    172         DO 114 J = 1,26
2    173           Z(I,J) =0.0
2    174   114 CONTINUE
     175
     176    4  DO 115 I = 1,M
1    177         DO 115 J = 1,N
2    178           Z(I,J) = R(1,(I-1)*N+J)
2    179   115 CONTINUE
     180
     181 C     ***** OUTPUT THE Y ARRAY *****
     182       DO 119 I = 1,M*N
1    183         WRITE (*,*) R(1,I)
1    184   119 CONTINUE
     185 C     ***** OUTPUT THE Z MATRIX *****
     186       WRITE (*,205) '*****  Z  M A T R I X  ',M,' X ',N,'  *****'
     187       WRITE (*,212)
     188       DO 116 I = 1,M
1    189         WRITE (*,300) (Z(I,J), J = 1,N)
1    190         WRITE (*,210)
1    191   116 CONTINUE
     192       WRITE (*,213)
     193
     194       WRITE(*,421)
     195       READ  (*,200) ANSWER
     196       IF  ((ANSWER .NE. 'Y') .AND. (ANSWER .NE. 'y')) GOTO 19
     197       DO 117 I = 1,630
1    198         X(I) = 0.0
1    199         Y(I) = 0.0
1    200   117 CONTINUE
     201
     202       DO 118 I = 1,M*N
1    203         X(I) = I * 1.0
1    204         Y(I) = R(1,I)
```

```
D Line# 1      7                                  Microsoft FORTRAN77 V3.20 02/84
1    205   118 CONTINUE
     206
     207    18  WRITE (*,415)
     208        READ  (*,199)  CTEXT
     209        WRITE (*,451)
     210        READ  (*,200)  ANSWER
     211
     212 C      ***** INITIALIZE PLOT88 *****
     213        IF  ((ANSWER .EQ. 'Y') .OR. (ANSWER .EQ. 'y'))  THEN
     214           CALL PLOTS(0,0,2)
     215        ELSE
     216           CALL PLOTS(0,99,99)
     217        ENDIF
     218
     219        CALL PLOT(1.0,1.0,-3)
     220        CALL SCALE(X,6.0,M*N,1)
     221        CALL SCALE(Y,4.0,M*N,1)
     222        CALL STAXIS(0.20,0.20,0.111,0.112,1)
     223        CALL AXIS(0.0,0.0,'X AXIS',-6,6.0,0.0,X(M*N+1),X(M*N+2))
     224        CALL AXIS(0.0,0.0,'Y AXIS',6,4.0,90.0,Y(M*N+1),Y(M*N+2))
     225        CALL LINE(X,Y,M*N,1,0,0)
     226        CALL PLOT(0.0,0.0,-3)
     227        CALL SYMBOL(1.0,6.75,0.25,CTEXT,0.0,20)
     228        CALL SYMBOL(6.0,6.5,0.2,'1-D DATA FIELD',0.0,14)
     229
     230 C      ***** OUTPUT THE GRAPH *****
     231        CALL PLOT(0.0,0.0,999)
     232        WRITE (*,416)
     233        READ  (*,200) ANSWER
     234        IF  ((ANSWER .EQ. 'Y') .OR. (ANSWER .EQ. 'y')) GOTO 18
     235
     236    19  WRITE(*,419)
     237        READ  (*,200) ANSWER
     238        IF  ((ANSWER .NE. 'Y') .AND. (ANSWER .NE. 'y')) GOTO 21
     239
     240 C      ***** ASK THE PARAMETERS FOR THE GRAPH *****
     241    20  WRITE (*,210)
     242        WRITE (*,*) '****  E N T E R  P L O T  P A R A M E T E R S  ****'
     243        WRITE (*,410)
     244        READ  (*,*)  AZIM
     245        WRITE (*,411)
     246        READ  (*,*)  ELEV
     247        WRITE (*,413)
     248        READ  (*,*)  ITRIM
     249        WRITE (*,414)
     250        READ  (*,*)  IDIV
     251        WRITE (*,415)
     252        READ  (*,199)  CTEXT
     253        WRITE (*,451)
     254        READ  (*,200)  ANSWER
     255
```

D Line# 1 7 Microsoft FORTRAN77 V3.30 08/84

```
    256 C      ***** INITIALIZE PLOT88 *****
    257        IF  ((ANSWER .EQ. 'Y') .OR. (ANSWER .EQ. 'y'))  THEN
    258           CALL PLOTS(0,0,2)
    259        ELSE
    260           CALL PLOTS(0,99,99)
    261        ENDIF
    262
    263        CALL WINDOW(XLOL,YLOL,XUPR,YUPR)
    264
    265 C      ***** DRAW THE MESH SURFACE OF THE GRAPH *****
    266        CALL MESHS(Z,26,26,N,M,AZIM,ELEV,0.5,0.5,8.25,6.5,IDIV,0,
    267       *           3,IPROJ,1,ZLOW,3,ITRIM,MASK,VERTEX)
    268
    269 C      ***** ANNOTATION OF THE GRAPH *****
    270        CALL SYMBOL(1.0,6.75,0.35,CTEXT,0.0,20)
    271        CALL SYMBOL(6.0,6.5,0.2,'2-D DATA FIELD',0.0,14)
    272        CALL SYMBOL(5.5,0.3,0.2,'AZIMUTH:   ',0.0,10)
    273        CALL NUMBER(999.0,999.0,0.2,AZIM,0.0,2)
    274        CALL SYMBOL(5.5,0.0,0.2,'ELEVATION:',0.0,10)
    275        CALL NUMBER(999.0,999.0,0.2,ELEV,0.0,2)
    276        DY = (Z(1,1)/90.0) * ELEV
    277        CALL P3D2D(1.0,1.0,Z(1,1)-DY,XR,YR)
    278        CALL SYMBOL(XR,YR,0.25,'*',0.0,1)
    279        CALL SYMBOL(1.0,0.1,0.2,'* = ORIGIN',0.0,10)
    280
    281 C      ***** OUTPUT THE GRAPH *****
    282        CALL PLOT(0.0,0.0,999)
    283        WRITE (*,416)
    284        READ  (*,200) ANSWER
    285        IF  ((ANSWER .EQ. 'Y') .OR. (ANSWER .EQ. 'y')) GOTO 20
    286
    287  21    WRITE(*,418)
    288        READ(*,200)  ANSWER
    289        IF  ((ANSWER .EQ. 'Y') .OR. (ANSWER .EQ. 'y')) THEN
    290 C        **** FILL 0's THE TWO DIMENTIONAL GRID OF CONTROL POINTS ****
    291          DO 132 I = 1,26
  1 292            DO 132 J = 1,26
  2 293              ZF(I,J) =0.0
  2 294   132      CONTINUE
    295
    296          ZFMAX = -9.9E30
    297          ZFMIN = 9.9E30
    298          DN = (N-1)/2.0
    299          DM = (M-1)/2.0
    300          P = 3.141592
    301          DO 133 MM = 1,M
  1 302            DO 133 NN = 1,N
  2 303              RLPART  = 0.0
  2 304              IMGPART = 0.0
  2 305              DO 134 L = 1,M
  3 306                DO 134 K = 1,N
```

```
D Line# 1      7                                  Microsoft FORTRAN77 V3.20 08/84
4    307                R1(1) = COS(-2*P*(L-1)*(MM-DM-1)/M)
4    308                R1(2) = SIN(-2*P*(L-1)*(MM-DM-1)/M)
4    309                R2(1) = COS(-2*P*(K-1)*(NN-DN-1)/N)
4    310                R2(2) = SIN(-2*P*(K-1)*(NN-DN-1)/N)
4    311                RLPART  = RLPART  + Z(L,K)*(R1(1)*R2(1)
4    312       *                                   -R1(2)*R2(2))
4    313                IMGPART = IMGPART + Z(L,K)*(R1(1)*R2(2)
4    314       *                                   +R1(2)*R2(1))
4    315    134       CONTINUE
2    316              ZF(MM,NN) = SQRT(RLPART**2 + IMGPART**2)
2    317              IF (ZF(MM,NN) .GT. ZFMAX)  ZFMAX = ZF(MM,NN)
2    318              IF (ZF(MM,NN) .LT. ZFMIN)  ZFMIN = ZF(MM,NN)
2    319    133   CONTINUE
     320
     321 C         ***** OUTPUT THE ZF MATRIX *****
     322           WRITE (*,205) '*** FOURIER TRANSFORMATION  ',M,' X ',N,' ***'
     323           WRITE (*,212)
     324           DO 135 I = 1,M
1    325              WRITE (*,300) (ZF(I,J), J = 1,N)
1    326              WRITE (*,210)
1    327    135   CONTINUE
     328           WRITE (*,213)
     329
     330           WRITE(*,419)
     331           READ  (*,200) ANSWER
     332           IF  ((ANSWER .NE. 'Y') .AND. (ANSWER .NE. 'y')) GOTO 22
     333
     334 C         ***** ASK THE PARAMETERS FOR THE GRAPH *****
     335    30     WRITE (*,210)
     336           WRITE (*,*) '***  E N T E R  P L O T  P A R A M E T E R S  ***'
     337           WRITE (*,410)
     338           READ  (*,*)  AZIM
     339           WRITE (*,411)
     340           READ  (*,*)  ELEV
     341           WRITE (*,413)
     342           READ  (*,*)  ITRIM
     343           WRITE (*,414)
     344           READ  (*,*)  IDIV
     345           WRITE (*,415)
     346           READ  (*,199)  CTEXT
     347           WRITE (*,451)
     348           READ  (*,200)  ANSWER
     349
     350 C         ***** INITIALIZE PLOT88 *****
     351           IF  ((ANSWER .EQ. 'Y') .OR. (ANSWER .EQ. 'y'))  THEN
     352             CALL PLOTS(0,0,2)
     353           ELSE
     354             CALL PLOTS(0,99,99)
     355           ENDIF
     356
     357           WRITE (*,420)
```

```
D Line# 1      7                                   Microsoft FORTRAN77 V3.30 02/84
     358          READ (*,200) ANSWER
     359
     360          CALL WINDOW(XLOL,YLOL,XUPR,YUPR)
     361
     362          IF ((ANSWER .EQ. 'Y') .OR. (ANSWER .EQ. 'y')) THEN
     363            DLEV = (ZFMAX-ZFMIN)/FLOAT(M)
     364            CALL ZLEVEL(ZF,26,26,M,N,DLEV,ZLEV,N)
     365            DO 136 I = 1,N
1    366              LDIG(I) = 2
1    367              LWGT(I) = 1
1    368    136     CONTINUE
     369            CALL ZCNTUR(ZF,26,26,M,N,0.5,0.5,8.25,6.5,ZLEV,LDIG,LWGT,
     370       *                N,0.10,10)
     371            CALL SYMBOL(5.5,0.0,0.2,'CONTOUR MAP',0.0,11)
     372          ELSE
     373 C          ***** DRAW THE MESH SURFACE OF THE GRAPH *****
     374            CALL MESHS(ZF,26,26,M,N,AZIM,ELEV,0.5,0.5,8.25,6.5,IDIV,0,
     375       *               3,IPROJ,1,ZLOW,3,ITRIM,MASK,VERTEX)
     376
     377 C          ***** ANNOTATION OF THE GRAPH *****
     378            CALL SYMBOL(5.5,0.3,0.2,'AZIMUTH:   ',0.0,10)
     379            CALL NUMBER(999.0,999.0,0.2,AZIM,0.0,2)
     380            CALL SYMBOL(5.5,0.0,0.2,'ELEVATION:',0.0,10)
     381            CALL NUMBER(999.0,999.0,0.2,ELEV,0.0,2)
     382            DY = (ZF(1,1)/90.0) * ELEV
     383            CALL P3D2D(1.0,1.0,ZF(1,1)-DY,XR,YR)
     384            CALL SYMBOL(XR,YR,0.25,'*',0.0,1)
     385            CALL SYMBOL(1.0,0.1,0.2,'* = ORIGIN',0.0,10)
     386          ENDIF
     387          CALL SYMBOL(1.0,6.75,0.25,CTEXT,0.0,20)
     388          CALL SYMBOL(6.0,6.5,0.2,'3-D DFT',0.0,7)
     389
     390 C        ***** OUTPUT THE GRAPH *****
     391          CALL PLOT(0.0,0.0,999)
     392          WRITE (*,416)
     393          READ  (*,200) ANSWER
     394          IF  ((ANSWER .EQ. 'Y') .OR. (ANSWER .EQ. 'y')) GOTO 30
     395   22    ENDIF
     396        WRITE (*,417)
     397        READ  (*,200) ANSWER
     398        IF  ((ANSWER .EQ. 'Y') .OR. (ANSWER .EQ. 'y')) GOTO 2
     399        STOP
     400
     401   199 FORMAT(A20)
     402   200 FORMAT(A)
     403   205 FORMAT(/,18X,A29,I2,A3,I2,A8,/)
     404   210 FORMAT()
     405   211 FORMAT(/,5X,60A)
     406   212 FORMAT(/,2X,'(AZIMUTH 320.0)',46X,'(AZIMUTH 230.0)',/)
     407   213 FORMAT(/,2X,'(AZIMUTH 050.0)',46X,'(AZIMUTH 140.0)',/)
     408   300 FORMAT(10(F7.2,1X))
```

D Line# 1 7 Microsoft FORTRAN77 V3.20 02/84

```
    409   400 FORMAT(9X,\)
    410   401 FORMAT(/,5X,'DO YOU WANT TO FILL THE Z MATRIX ? ',\)
    411   402 FORMAT(/,5X,'DO YOU WANT TO FILL THE TRANSITION MATRIX ? ',\)
    412   403 FORMAT(/,5X,'COLUMNS OF OUTPUT FRAME(N=1to25): ',\)
    413   404 FORMAT(/,5X,'ROWS OF OUTPUT FRAME(M=1to25): ',\)
    414   405 FORMAT(5X,A1,I2,A2,\)
    415   406 FORMAT(5X,A5,\)
    416   407 FORMAT(5X,A2,I2,A1,I2,A3,\)
    417   408 FORMAT(5X,A2,I3,A3,\)
    418   409 FORMAT(5X,A8,\)
    419   410 FORMAT(/,5X,'AZIMUTH(0.0 to 360.0 DEGREES): ',\)
    420   411 FORMAT(/,5X,'ELEVATION(90.0 to -90.0 DEGREES): ',\)
    421   412 FORMAT(/,5X,'NUMBER OF SMOOTHINGS: ',\)
    422   413 FORMAT(/,5X,'TRIM(0=NO,1=Xs,2=Ys): ',\)
    423   414 FORMAT(/,5X,'2,4 OR 8 SUBGRIDS: ',\)
    424   415 FORMAT(/,5X,'TITLE OF GRAPH(UP TO 20 CHAR): ',\)
    425   416 FORMAT(/,5X,'DO YOU WANT TO CHANGE PARAMETERS? ',\)
    426   417 FORMAT(/,5X,'DO YOU WANT TO REPEAT THE PROCESS? ',\)
    427   418 FORMAT(/,5X,'DO YOU WANT FOURIER TRANSFORMATION ? ',\)
    428   419 FORMAT(/,5X,'DO YOU WANT TO MAKE GRAPH ? ',\)
    429   420 FORMAT(/,5X,'DO YOU WANT CONTOUR MAP ? ',\)
    430   421 FORMAT(/,5X,'DO YOU WANT TO DRAW CARVE ? ',\)
    431   451 FORMAT(/,5X,'SEND GRAPH TO THE PRINTER(Y or N): ',\)
    432       END
```

Name	Type	Offset	P	Class
ANSWER	CHAR*1	30		
AZIM	REAL	194		
COS				INTRINSIC
CTEXT	CHAR*20	174		
DLEV	REAL	284		
DM	REAL	230		
DN	REAL	226		
DY	REAL	206		
ELEV	REAL	198		
FLOAT				INTRINSIC
I	INTEGER*2	32		
IDIV	INTEGER*2	204		
IMGPAR	REAL	258		
IPROJ	INTEGER*2	22		
ITRIM	INTEGER*2	202		
IV	REAL	0		LARGE
J	INTEGER*2	34		
JJ	INTEGER*2	110		
K	INTEGER*2	270		
L	INTEGER*2	262		
LDIG	INTEGER*2	10552		/WORK /
LWGT	INTEGER*2	10604		/WORK /
M	INTEGER*2	28		
MASK	INTEGER*2	10656		/WORK /

D Line# 1	7		
MM	INTEGER*2	238	
N	INTEGER*2	26	
NN	INTEGER*2	246	
NRNG	INTEGER*2	24	
P	REAL	234	
R	REAL	0	LARGE
R1	REAL	0	LARGE
R2	REAL	8	LARGE
RLPART	REAL	254	
S	REAL	0	LARGE
SIN			INTRINSIC
SQRT			INTRINSIC
TEMP	REAL	78	
TRM	REAL	0	LARGE
U	REAL	94	
VERTEX	REAL	16656	/WORK /
X	REAL	5408	/WORK /
XLOL	REAL	2	
XR	REAL	210	
XUPR	REAL	10	
Y	REAL	7928	/WORK /
YLOL	REAL	6	
YR	REAL	214	
YUPR	REAL	14	
Z	REAL	0	/WORK /
ZF	REAL	2704	/WORK /
ZFMAX	REAL	218	
ZFMIN	REAL	222	
ZLEV	REAL	10448	/WORK /
ZLOW	REAL	18	

Name	Type	Size	Class
AXIS			SUBROUTINE
LINE			SUBROUTINE
MAIN			PROGRAM
MESHS			SUBROUTINE
NUMBER			SUBROUTINE
P3D2D			SUBROUTINE
PLOT			SUBROUTINE
PLOTS			SUBROUTINE
SCALE			SUBROUTINE
STAXIS			SUBROUTINE
SYMBOL			SUBROUTINE
WINDOW			SUBROUTINE
WORK		16720	COMMON
ZCNTUR			SUBROUTINE
ZLEVEL			SUBROUTINE

D Line# 1 7
Pass One No Errors Detected
432 Source Lines

LIST OF REFERENCES

1. Lugt, A. Vander, "Orientational Notation for the Analysis and Synthesis of Optical Data-processing Systems," Proc. of IEEE, Vol. 84, pp. 1055-1063, August 1966.

2. Yu, F.T.S., Introduction to Diffraction Information Processing and Holography, MIT Press, Cambridge, Massachusetts, 1973.

3. Habibi, A., "Two-dimensional Bayesian Estimate of Images," Proc. IEEE (Special Issues on Digital Nature Processing) Vol. 60, pp. 878-883, July 1972.

4. Fryez, W.D. and Richmond, G.E., "Two-dimensional Spatial Filtering and Computers," in Proc. Nat. Elèctron Conf. 19 October 1962.

5. Roeeser, Robert P., "A Discrete State-Space Model for Linear Image Processing," IEEE Trans. Automatic Control, AC-20, 70.1 (Feb 1975).

6. Cadzow, James A., Discrete Time Systems, An Introduction with Interdisciplinary Applications, McGraw Hill Company, New York, New York, 1973.

7. Kung, Sun Yuan, Levy, Bernard C., Morf, Martin, Kailatn, Thomas, "New Results in 2-D Systems Theory, Part II: 2-D State-Space Models, Realization and the Notions of Controllability, Observability and Minimality," Proceedings of the IEEE, No. 6, pp. 945-6, June 1977.

8. Oppenheim, Alan V., Schafer, Ronald W., Digital Signal Processing, Prentice Hall, Inc., Englewood Cliffs, New Jersey, 1975.

9. Dudgeon, Dan E., Mersereau, Russell M., Multidimensional Digital Signal Processing, Prentice-Hall, Inc., Englewood Cliffs, New Jersey, 1984.

10. Chou, David S.K., "A Simple Derivation of Minimal and Near-Minimal Realizations of 2-D Transfer Functions," Proc IEEE, Vol. 66, No. 4 (April 1978), pp. 515-516.

11. Attasi, S., "Systemes Linearizes Homogenes A Deux Indices," Raport Laboria, No. 31, September 1973.

12. "Modelisation et Traitement Des Suites A Deux Indices," Rapport Laboria, September 1975.

13. Fozuasimi, E. and G. Mazchesini, "Algebraic Realization Theory of Two-Dimensional Filters," in A. Ruberti and R. Molder (Eds.), Variable Struture Systems, Springer Verlag (Springer Lecture Notes in Economics and Mathematical Systems), 1975.

14. "State-Space Realization Theory of Two Dimensional Filters," IEEE Trans Automatic Control, Pp. 484-492, August 1976.

15. Wolovick, W.A., Linear Multivariable Systems, Springer-Verlag, New York, 1974.

16. Mullans, R.E. and D.L. Elliot, "Linear Systems on Partially Ordered Time Sets," Proc 1974 IEEE Control Decisions and Control, pp. 334,337, 1973.

17. Givone, D.D. and Roesser, R.P., "Minimization of Multi-dimensional Linear Iterative Circuits," IEEE Trans. Comment, Vol. C-22, pp. 673-678, July 1973.

18. Givone, D.D. and Roesser, P.R., "Multidimensional Linear Iterative Circuits, General Properties," IEEE Trans. Comment, Vol. C-21, pp. 1067-1073, October 1972.

19. Mitra, S.K., Sagar, A.D. and Pendergass, N.A., "Realizations of Two-Dimensional Recursive Digital Filters," IEEE Trans. Circuits and Systems, Vol. CAS-22, pp. 177-184, March 1975.

20. Chev, C.T., Introduction to Linear System Theory, Holt, Rinehart, and Winston, New York, 1970.

21. Technical Software Systems, SSPACK: State-Space Systems Software Package, 1983.

22. Jazwinski, A., Stochastic Processes and Filtering Theory, Academic Press: New York, 1970.

23. Bierman, G., Factorization Methods For Discrete Sequential Estimation, Academic Press: New York, 1977.

24. Read, R.R., Shanks, J. and Ttel, S., "Two-Dimensional Recursive Filtering," in Topics in Applied Physics, Springer-Verlag, 975, Vol. 6, pp. 137-176.

INITIAL DISTRIBUTION LIST

		No. Copies
1.	Defense Technical Information Center Cameron Station Alexandria, Virginia 22304-6145	2
2.	Library, Code 0142 Naval Postgraduate School Monterey, California 93943-5100	2
3.	Department Chairman, Code 62 Department of Electrical and Computer Engineering Naval Postgraduate School Monterey, California 93943-5100	1
4.	Dr. Sydney Parker, Code 62Px Department of Electrical and Computer Engineering Naval Postgraduate School Monterey, California 93943-5100	5
5.	Dr. Bhazat B. Maday Department of Computer Sciences and Engineering I.I.T. New Delphi - 110016 INDIA	1
6.	Lt. E.A. Theofilou Spartis 26 Nikea 184.54 Piraeys GREECE	7
7.	Professor Spiro Lekas 222 Laine St. Monterey, California 93940	1

END

FILMED

3 -86

DTIC