NAVAL POSTGRADUATE SCHOOL
Monterey, California

THESIS

2-D SIGNAL GENERATION USING STATE-SPACE FORMULATION

by

Evangelos Theofilou

December 1985

Thesis Advisor: Sydney R. Parker

REPORT DOCUMENTATION PAGE

EPORT SECURITY CLASSIFICATION		1b. RESTRICTIVE MARKINGS	
ECURITY CLASSIFICATION AUTHORITY		3. DISTRIBUTION/AVAILABILITY OF REPORT Approved for public release; distribution is unlimited	
ECLASSIFICATION/DOWNGRADING SCHEDULE			
RFORMING ORGANIZATION REPORT NUMBER(S)		5. MONITORING ORGANIZATION REPORT NUMBER(S)	
AME OF PERFORMING ORGANIZATION val Postgraduate School	6b. OFFICE SYMBOL *(If applicable)* Code 62	7a. NAME OF MONITORING ORGANIZATION Naval Postgraduate School	
DDRESS *(City, State, and ZIP Code)* nterey, California 93943-5100		7b. ADDRESS *(City, State, and ZIP Code)* Monterey, California 93943-5100	
AME OF FUNDING/SPONSORING RGANIZATION	8b. OFFICE SYMBOL *(If applicable)*	9. PROCUREMENT INSTRUMENT IDENTIFICATION NUMBER	
DDRESS *(City, State, and ZIP Code)*		10. SOURCE OF FUNDING NUMBERS	

PROGRAM ELEMENT NO.	PROJECT NO.	TASK NO.	WORK UNIT ACCESSION NO.

TLE *(Include Security Classification)*
) SIGNAL GENERATION USING STATE-SPACE FORMULATION

ERSONAL AUTHOR(S)
eofilou, Evangelos

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

JPPLEMENTARY NOTATION

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

BSTRACT *(Continue on reverse if necessary and identify by block number)*

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

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

ORM 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.

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

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:

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

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

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.

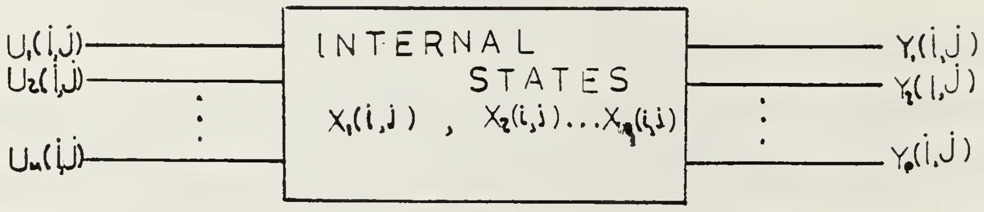


Figure 1.1

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

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

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.

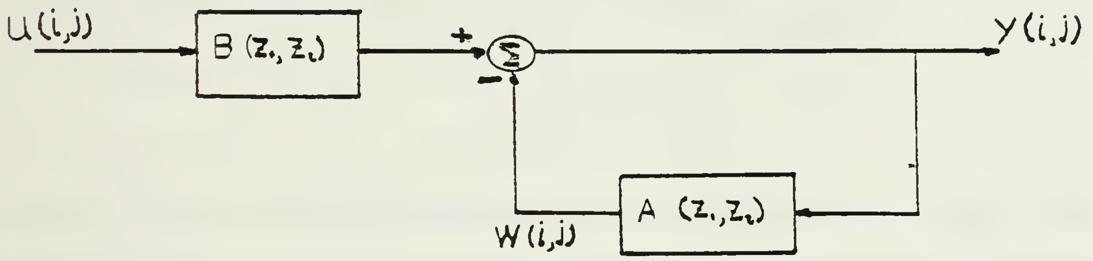


Figure 2.1

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.

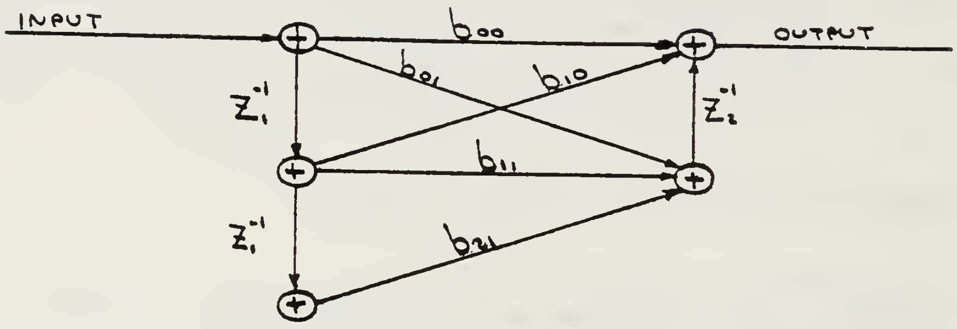


Figure 2.2

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.

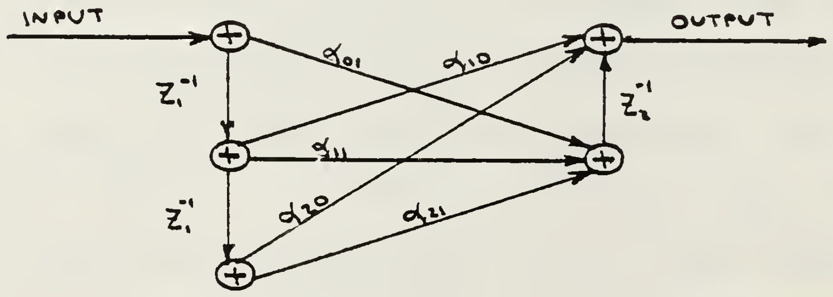


Figure 2.3

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:

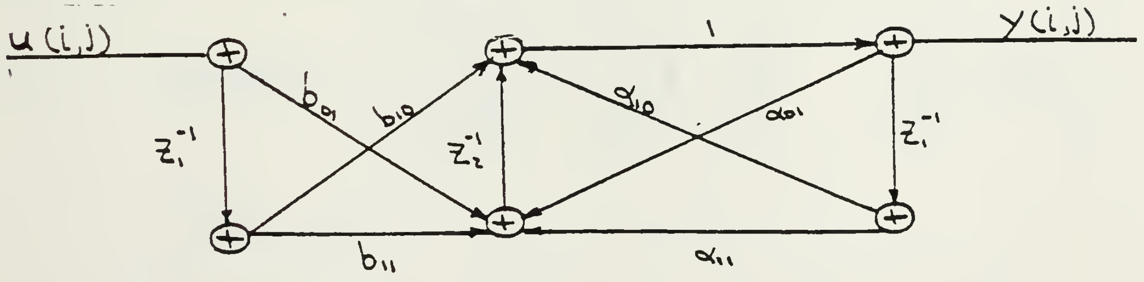


Figure 2.4

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

$$z_1R(z_1,z_2) - A_1R(z_1,z_2) - A_2S(z_1,z_2) = B_1u(z_1,z_2)$$

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

or

$$R(z_1,z_2)[z_1 - A_1] - A_2S(z_1,z_2) = B_1u(z_1,z_2)$$

$$R(z_1,z_2)[-A_3] - [z_2-A]S(z_1,z_2) = B_2u(z_1,z_2]$$

or

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

or

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

where $Z_1 = z_1I$ and $Z_2 = z_2I$, and

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

and after substitution in Equation (II.6)

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

$$+ Du(z_1,z_2)$$

or

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

$$= [C_1 \quad C_2] \left[\left[\begin{array}{c|c} Z_1 & 0 \\ \hline 0 & Z_2 \end{array}\right] - \left[\begin{array}{c|c} A_1 & A_2 \\ \hline A_3 & A_4 \end{array}\right] \right]^{-1} \begin{bmatrix} B_1 \\ B_2 \end{bmatrix}$$

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

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).

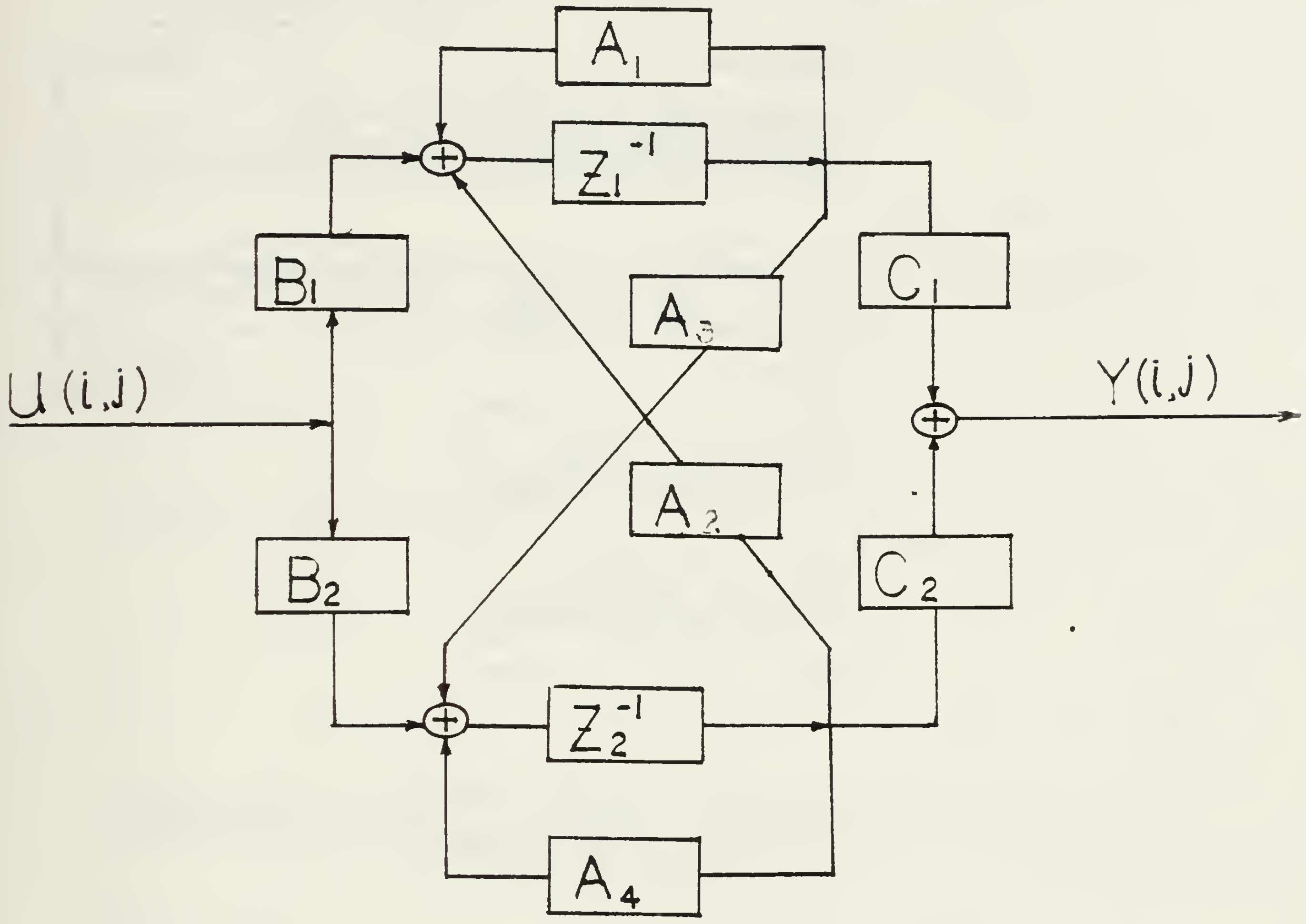


Figure 2.5

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

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

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

$$A^{-1} = [C_1 \quad C_2] \left[\frac{1}{(z_1-A_1)(z_2-A_4)-A_2A_3} \left[\begin{array}{c|c} Z_2-A_4 & A_2 \\ \hline A_3 & Z_1-A_1 \end{array}\right] \left[\begin{array}{c} B_1 \\ \hline B_2 \end{array}\right] \right]$$

$$= [C_1 \quad C_2] \begin{bmatrix} \dfrac{(Z_2-A_4)B_1}{(Z_2-A_1)(A_2-Z_4)-A_2A_3} & \dfrac{A_2B_2}{(Z_1-A_1)(Z_2-A_4)-A_2A_3} \\[2ex] \dfrac{A_3B_1}{(Z_1-A_1)(Z_2-A_4)-A_2A_3} & \dfrac{(Z_1-A_1)B_2}{(Z_1-A_1)(Z_2-A_4)-A_2A_3} \end{bmatrix}$$

$$= [C_1 \quad C_2] \begin{bmatrix} \dfrac{(Z_2-A_4)B_1 + A_2B_2}{(Z_2-A_1)(Z_2-A_4)-A_2A_3} \\[2ex] \dfrac{A_3B_1 + (Z_1-A_1)B_2}{(Z_1-A_1)(Z_2-A_4)-A_2A_3} \end{bmatrix}$$

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

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

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

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,

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

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

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

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

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

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

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

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:

$$\begin{aligned}
A_1 &= a_{01} \\
A_2 &= p \ ; \quad pq = a_{11} = a_{10}a_{01} \\
A_3 &= q; \quad b_{10}q^2-(b_{10}a_{01}+b_{01}a_{10}+b_{11})q+(b_{01}a_{11}+b_{01}a_{10}a_{01}) = 0 \\
A_4 &= a_{10} \\
C_1 &= b_{01} \qquad (II.19) \\
C_2 &= b_{10} \\
B_1 &= 1 \\
B_2 &= 1 \\
D &= 0
\end{aligned}$$

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,

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

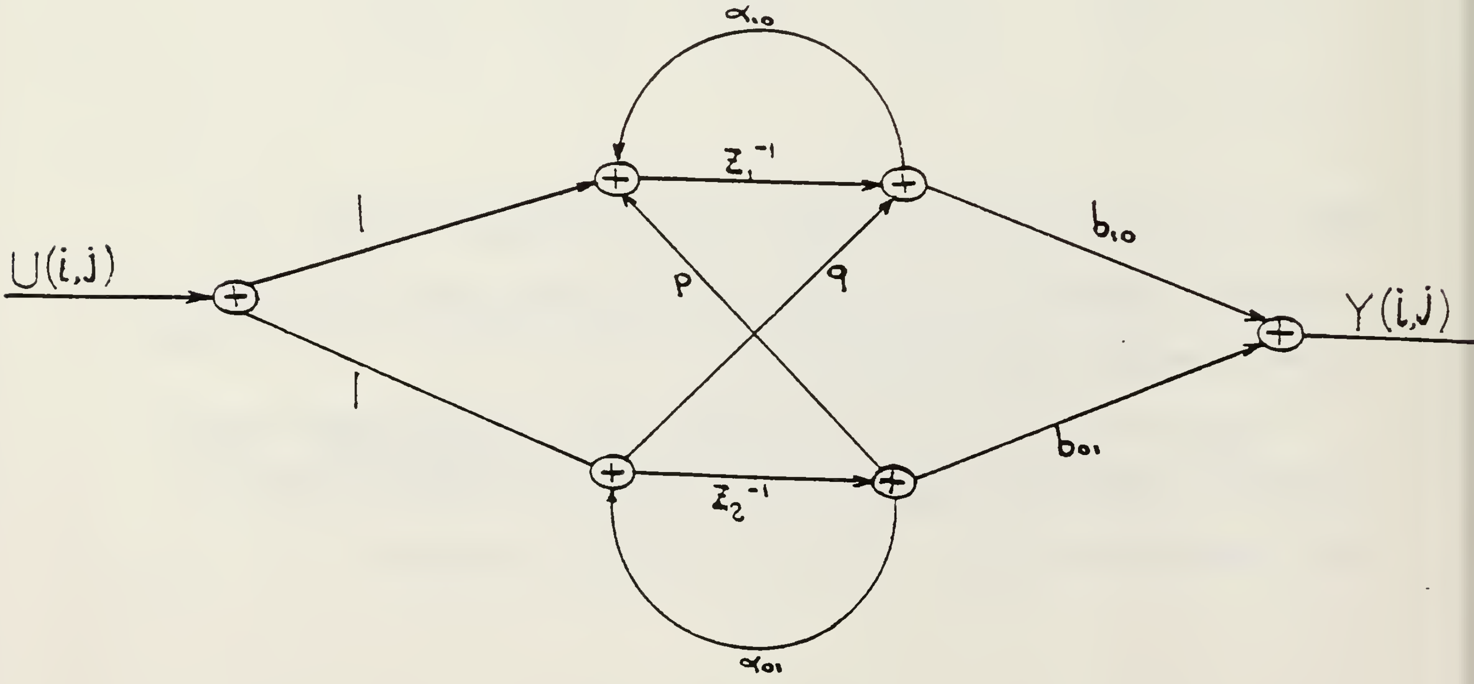
$$Y(i,j) = [b_{10} \quad b_{01}] \begin{bmatrix} R(i,j) \\ S(i,j) \end{bmatrix}$$

or

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

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:

$$\begin{array}{c|cccc|} & \multicolumn{4}{c}{\xrightarrow{\;j\;}} \\ \hline & 1,1 & 1,2 & 1,3 & 1,4 \\ & 2,1 & 2,2 & 2,3 & 2,4 \\ i\big\downarrow & 3,1 & 3,2 & 3,3 & 3,4 \\ & 4,1 & 4,2 & 4,3 & 4,4 \\ \hline \end{array} \qquad \begin{array}{c|cccc|} & \multicolumn{4}{c}{\xrightarrow{\;j\;}} \\ \hline & 1,1 & 1,2 & 1,3 & 1,4 \\ & 2,1 & 2,2 & 2,3 & 2,4 \\ i\big\downarrow & 3,1 & 3,2 & 3,3 & 3,4 \\ & 4,1 & 4,2 & 4,3 & 4,4 \\ \hline \end{array}$$

S matrix R matrix

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,

$$\begin{aligned} u(i,j) &= 1, \quad \text{for} \quad i = j = 1 \\ &= 0, \quad \text{otherwise.} \end{aligned}$$

The output data field is indexed as:

$$\begin{array}{c|cccc|} & & \overset{\longrightarrow}{j} & & \\ \hline & 1,1 & 1,2 & 1,3 & 1,4 \\ & 2,1 & 2,2 & 2,3 & 2,4 \\ i \downarrow & 3,1 & 3,2 & 3,3 & 3,3 \\ & 4,1 & 4,2 & 4,3 & 4,4 \\ \hline \end{array}$$

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. $\{y(i,j)\}$

$M \times N$: The dimension of the given data $y(\ell,k)$ and D.F.T. $Y(m,n)$ also.

$y(\ell,k)$: Given data (The output as described above).

To develop the D.F.T. for two-dimensional signals we consider a finite area sequence $y(\ell,k)$ 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

$$y(\ell,k) = \begin{array}{c|ccccc} & & & j & & \\ \hline & 1,1 & 1,2 & 1,3 & 1,4 & 1,5 \\ & 2,1 & 2,2 & 2,3 & 2,4 & 2,5 \\ & 3,1 & 3,2 & 3,3 & 3,4 & 3,5 \\ i & 4,1 & 4,2 & 4,3 & 4,4 & 4,5 \\ & 5,1 & 5,2 & 5,3 & 5,4 & 5,5 \end{array}$$

Matrix	5×5
M=5	N=5
ℓ=1,2,3,4,5	k=1,2,3,4,5

Then,

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

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

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

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

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

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

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

$$
\begin{aligned}
A_1 &= a_{01} = 0.3 \\
A_2 &= p = 0.2 \\
A_3 &= q = 0.3 \\
A_4 &= a_{10} = 0.2 \\
C_1 &= b_{01} = 0.5 \\
C_2 &= b_{10} = 0.5 \\
B_1 &= 1 \\
B_2 &= 1 \\
D &= 0
\end{aligned}
\tag{III.1a}
$$

<u>Second example:</u>

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

$$
\begin{aligned}
& a_{11} = 0.1 \qquad b_{00} = 0 \\
A_1 &= a_{01} = 0.2 \qquad b_{11} = 0.2 \\
A_2 &= p = 0.125 \qquad\quad = 0.83 \\
A_3 &= q = 1 \quad \text{or} \quad = 0.15 \\
A_4 &= a_{10} = 0.125 \\
C_1 &= b_{01} = 0.3 \\
C_2 &= b_{10} = 0.25
\end{aligned}
\tag{III.1b}
$$

$B_1 = 1$

$B_2 = 1$

$D = 0$

<u>Third example</u>:

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

$$\begin{aligned}
&b_{00} = 0 \quad b_{11} = 0.25\\
&a_{11} = 0.15\\
&A_1 = a_{01} = 0.25\\
&A_2 = p = 0.1312\\
&A_3 = q = 1.4\\
&A_4 = a_{10} = 0.135\\
&C_1 = b_{01} = 0.15\\
&C_2 = b_{10} = 0.25\\
&B_1 = 1\\
&B_2 = 1\\
&D = 0
\end{aligned} \tag{III.1c}$$

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

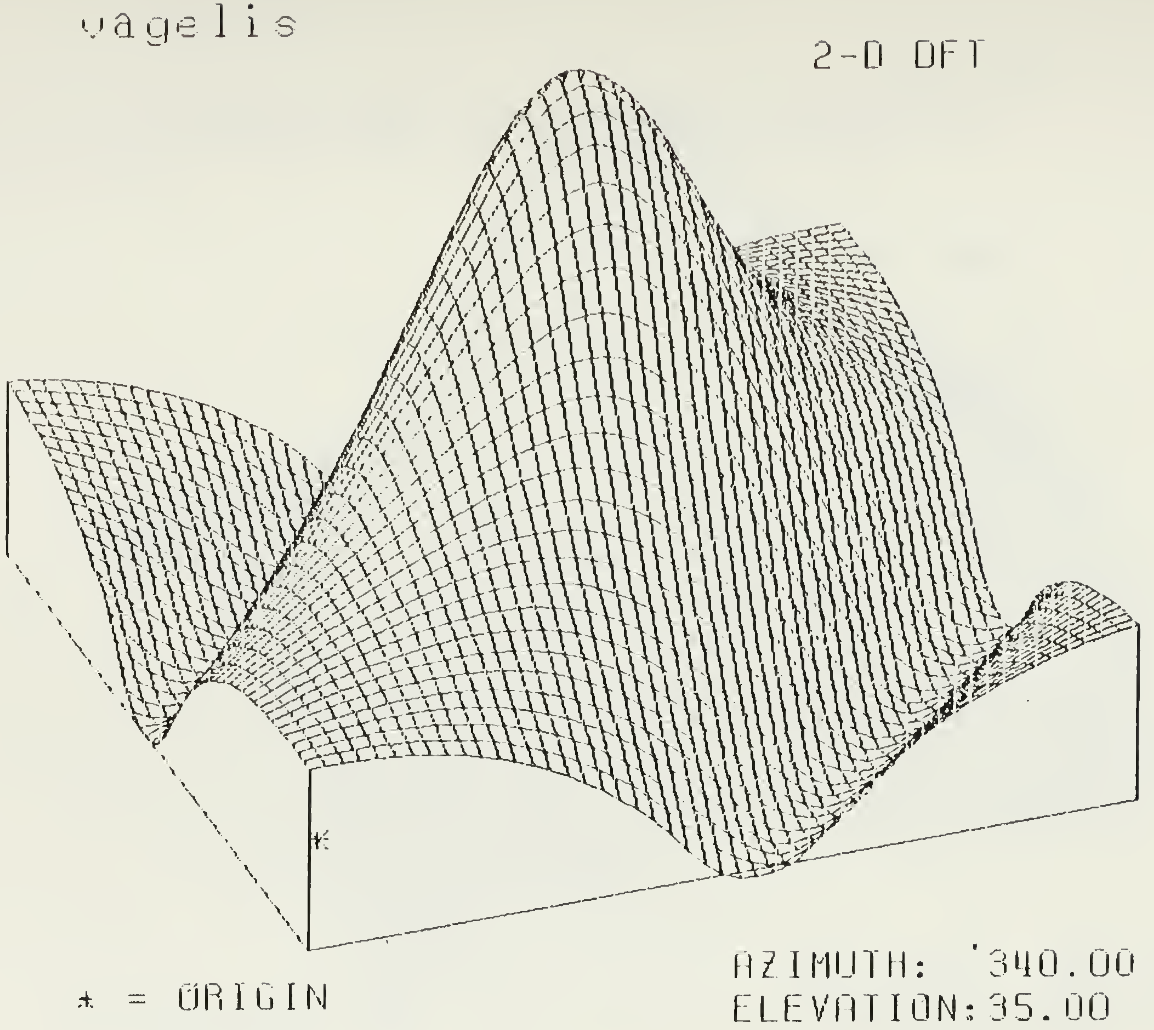


Figure 3-1a. 2-D D.F.T. Squence, Y(m,n) for Example 1

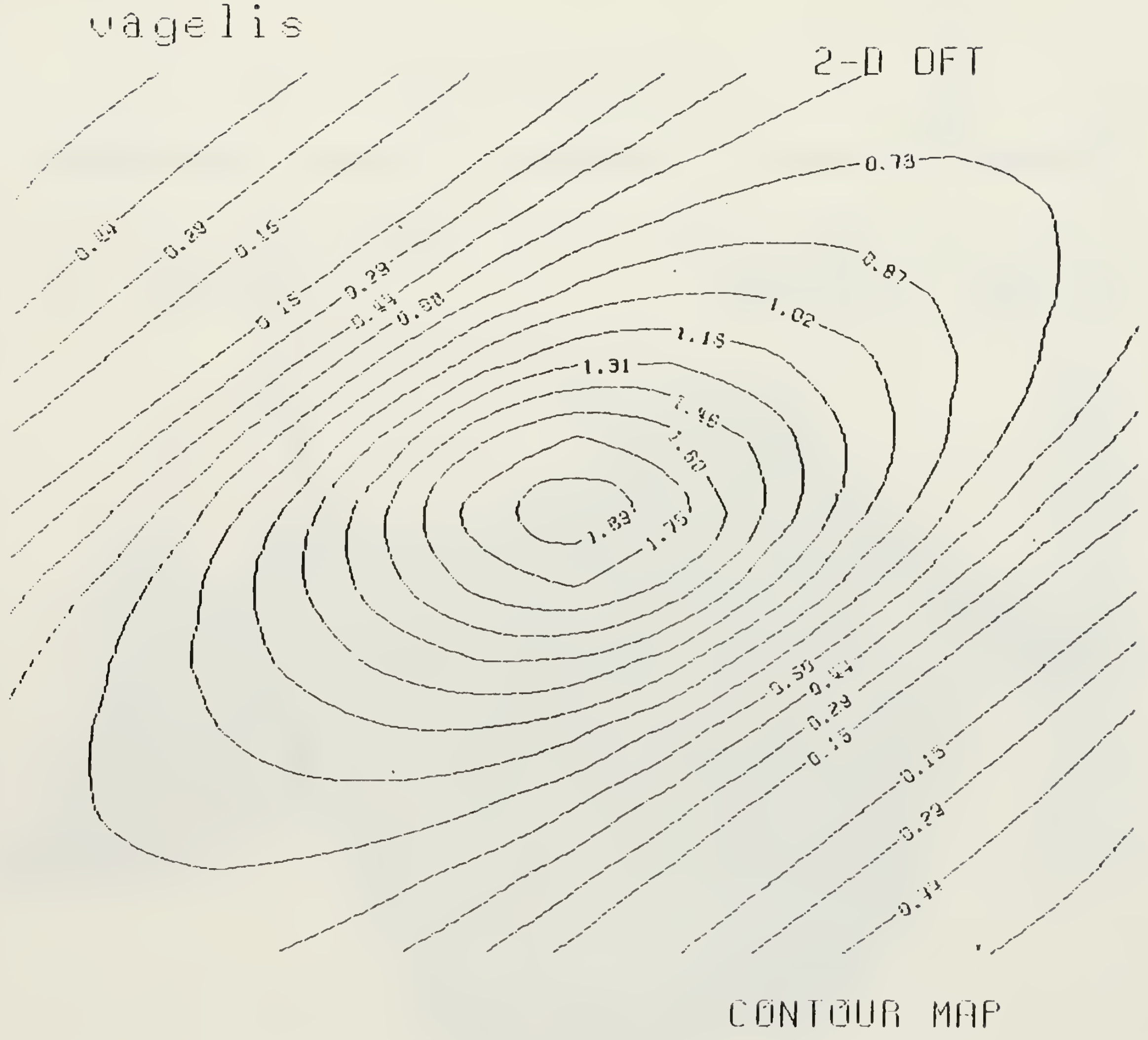


Figure 3-1b, Contour Map for Figure 3-1a

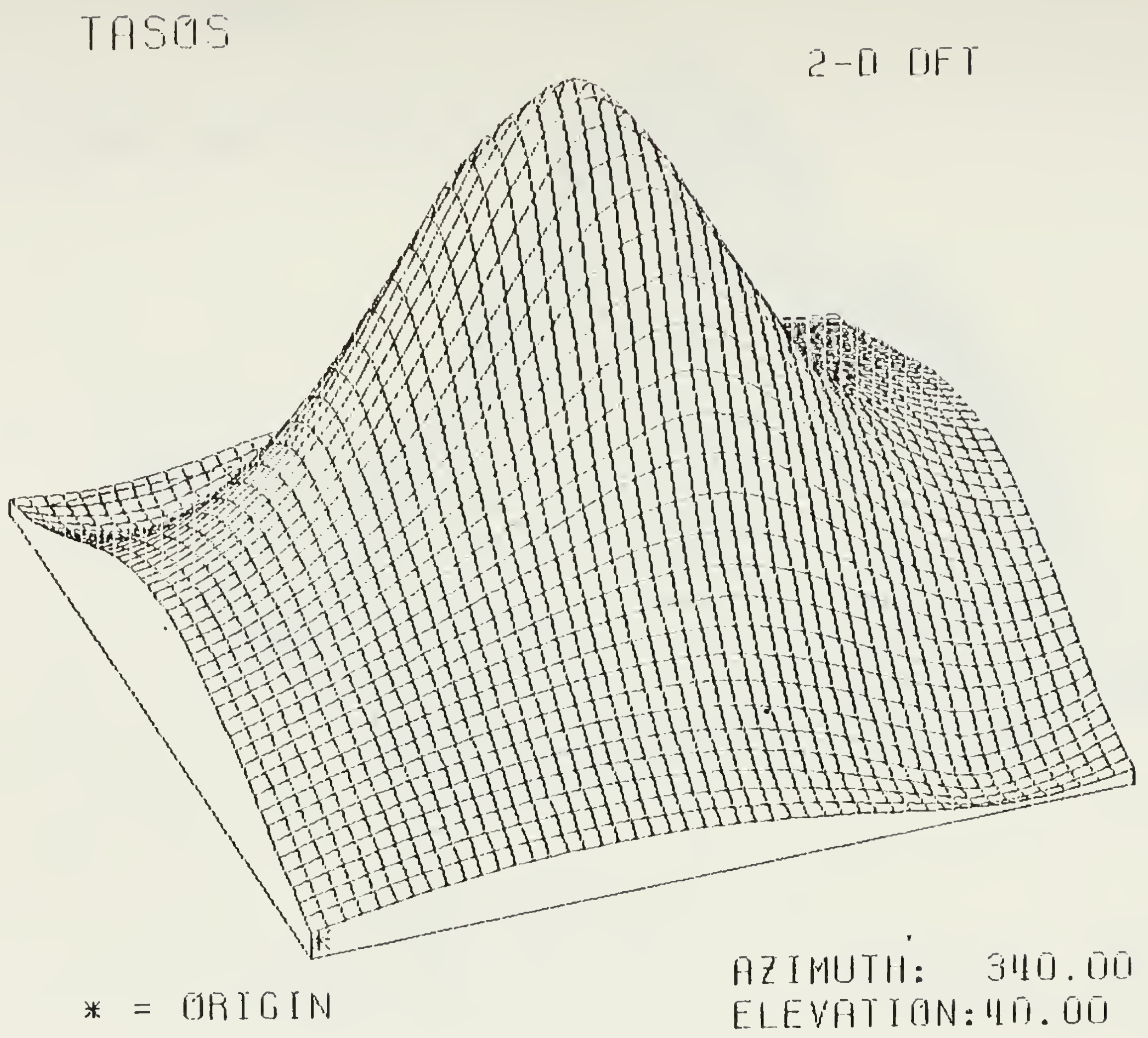


Figure 3-2a. 2-D D.F.T. Sequence, Y(m,n) for Example 2

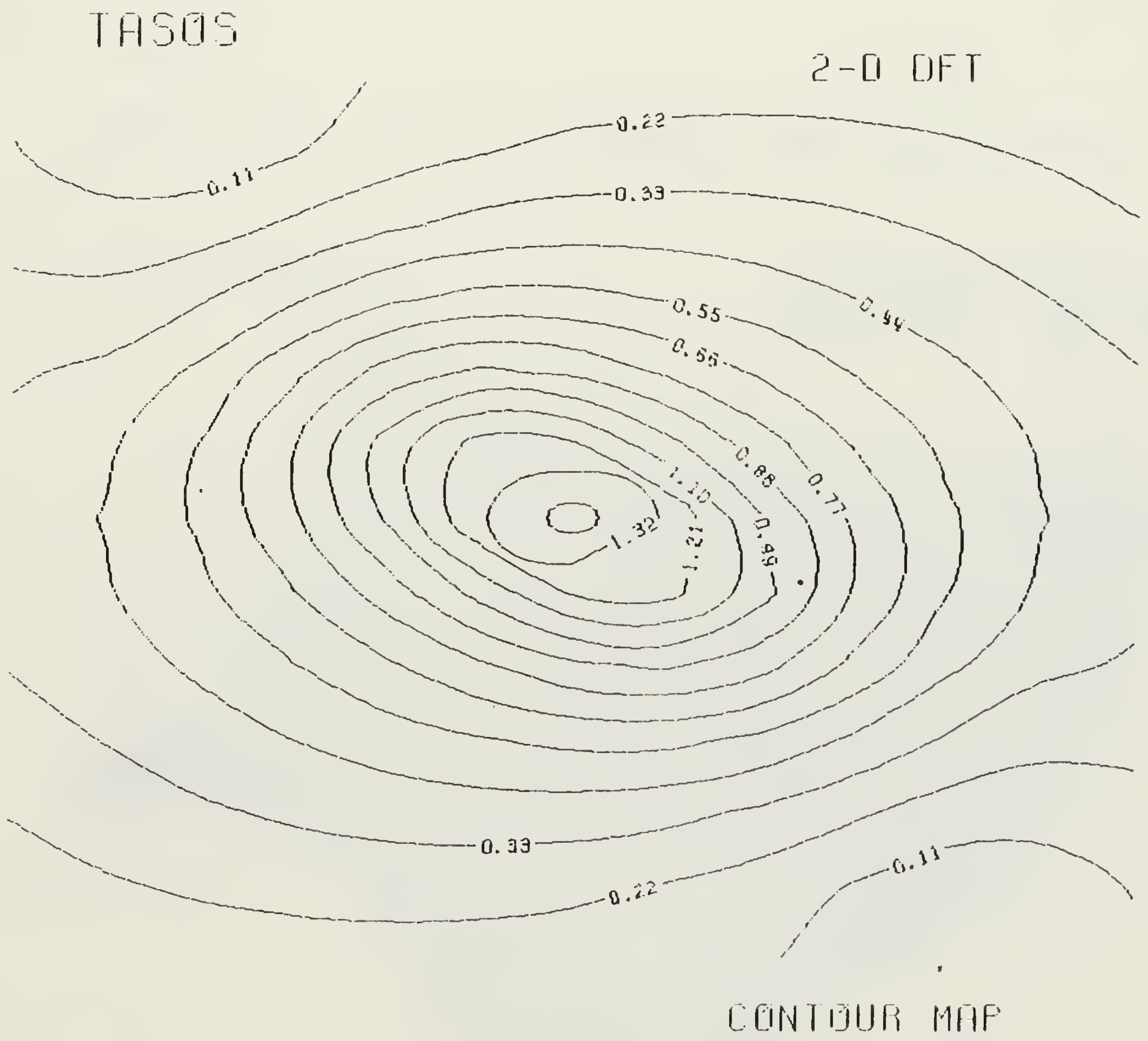


Figure 3-2b. Contour Map for Figure 3-2a

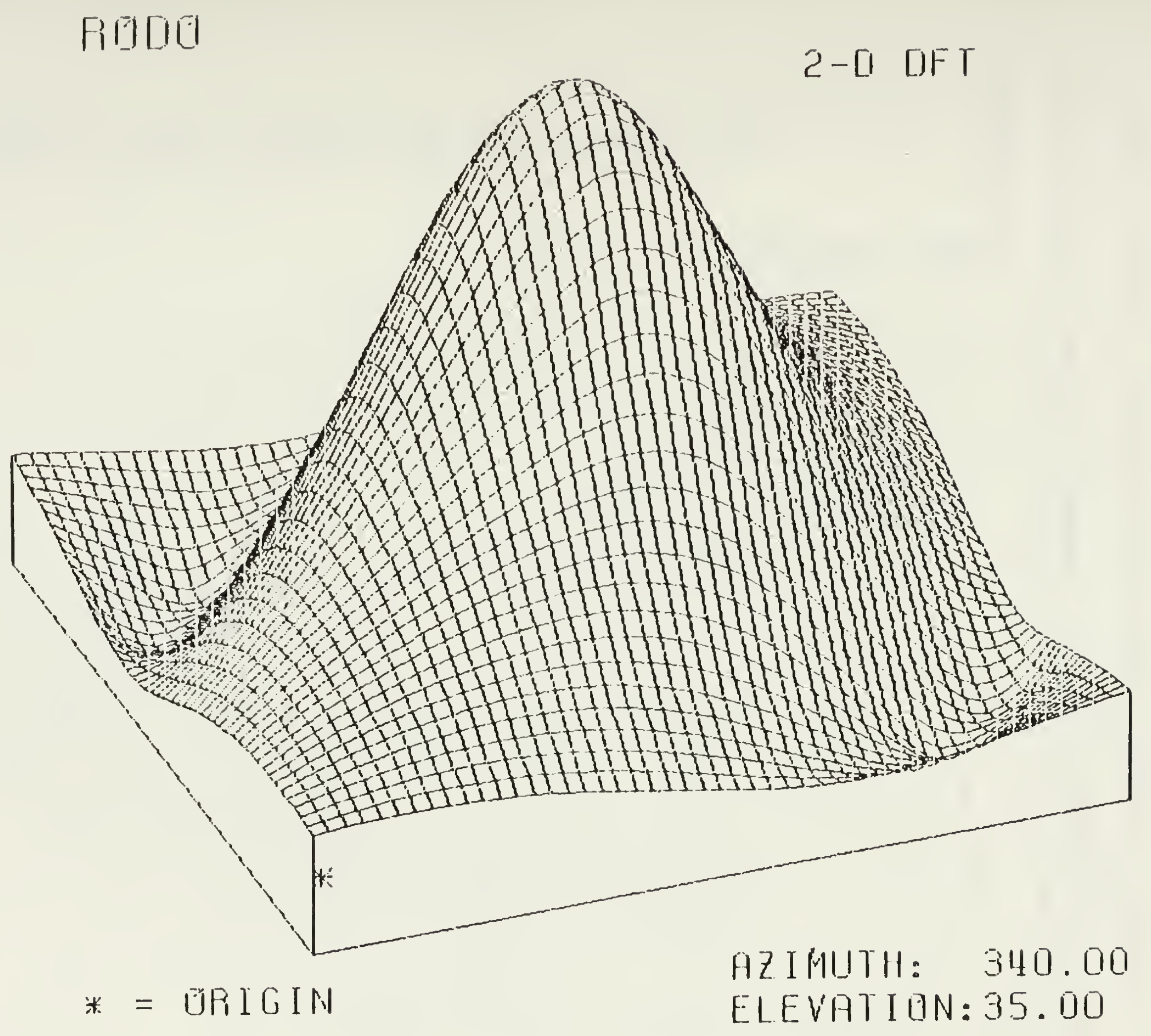


Figure 3-3a. 2-D D.F.T. Sequence, Y(m,n) for Example 3

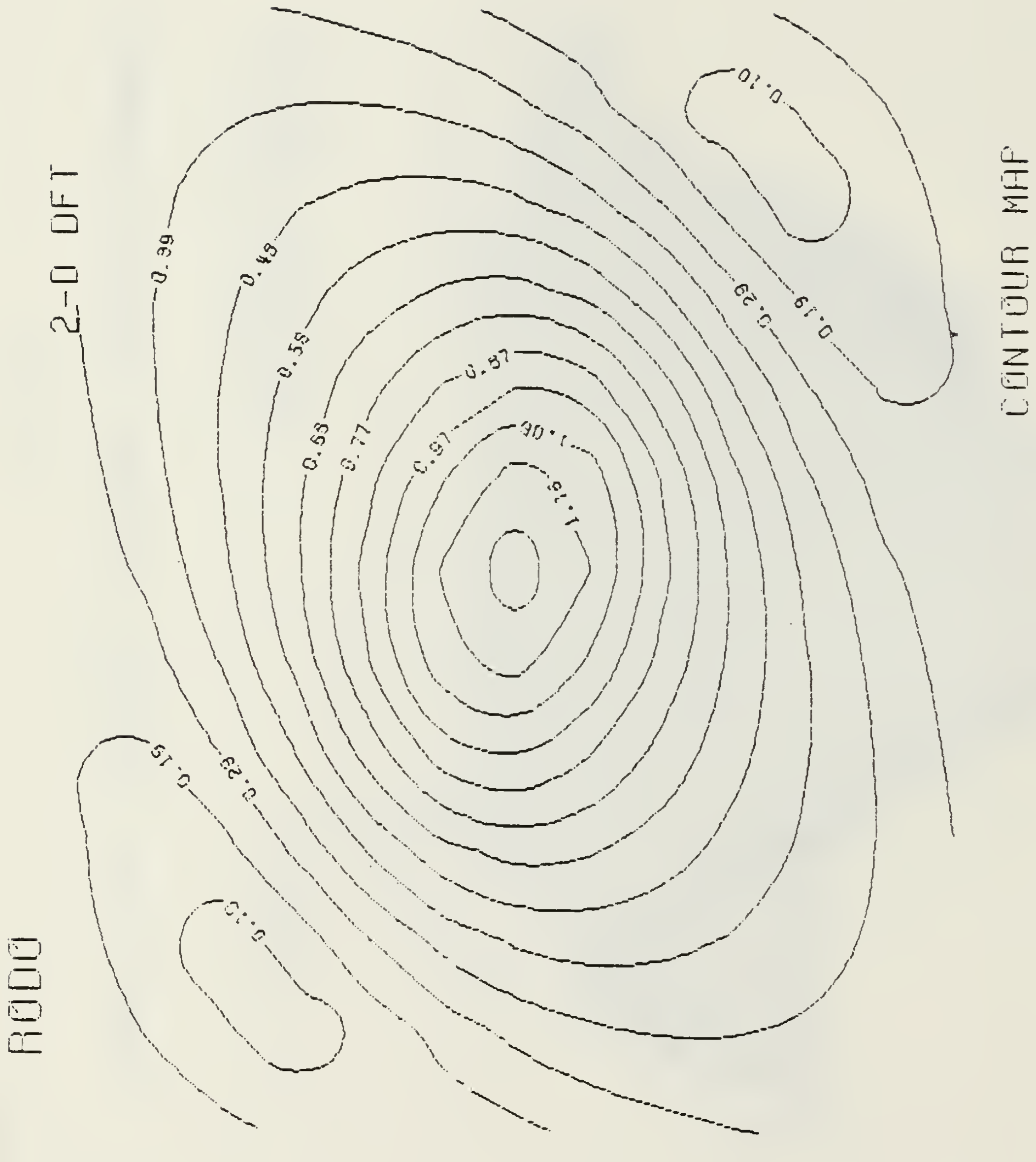


Figure 3-3b. Contour Map for Figure 3-3a

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.

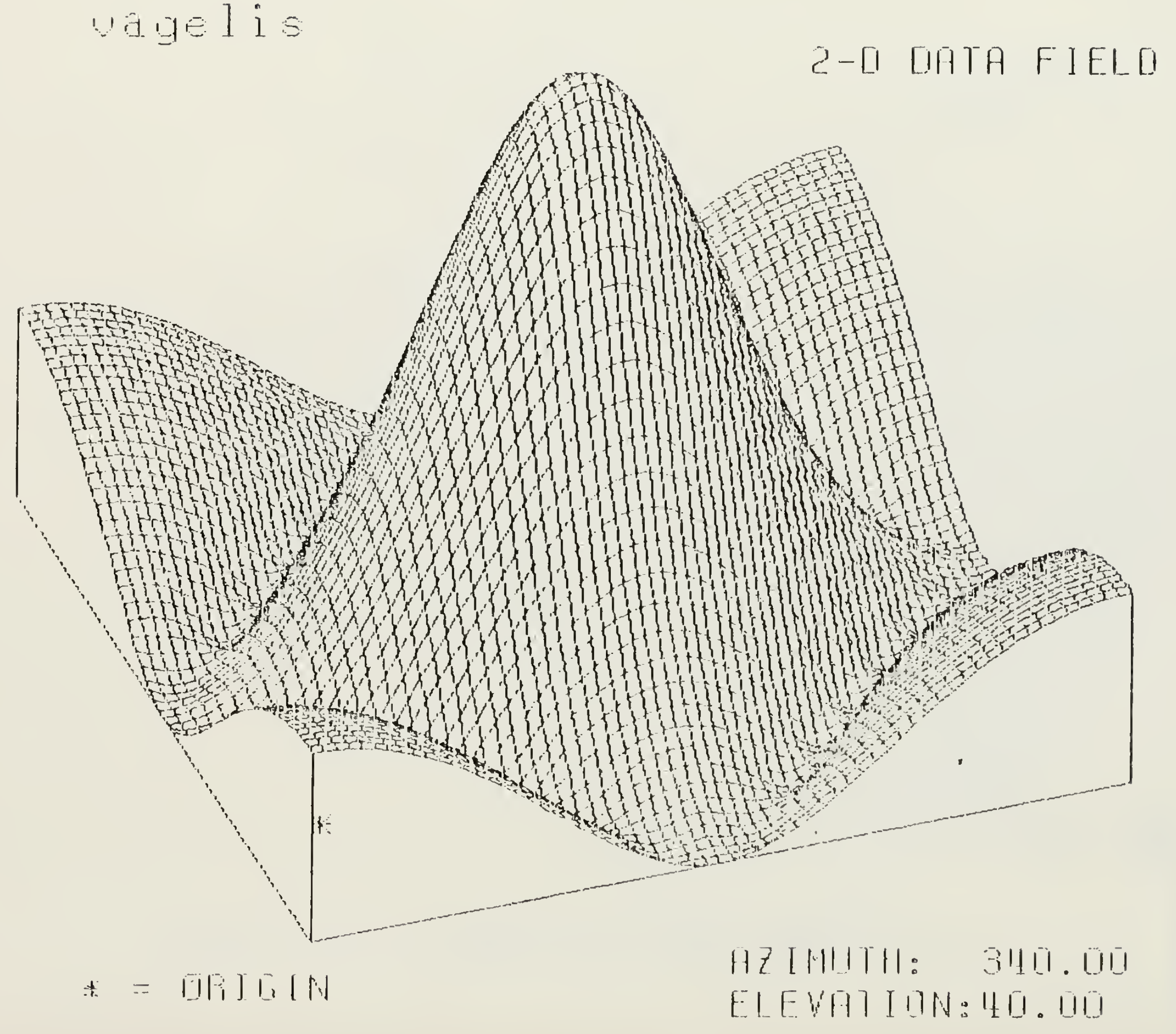


Figure 3-4a. Transfer Function $|H(z_1,z_2)|$, $z_1 = e^{j\omega_1}$, $z_2 = e^{j\omega_2}$ for Example 1

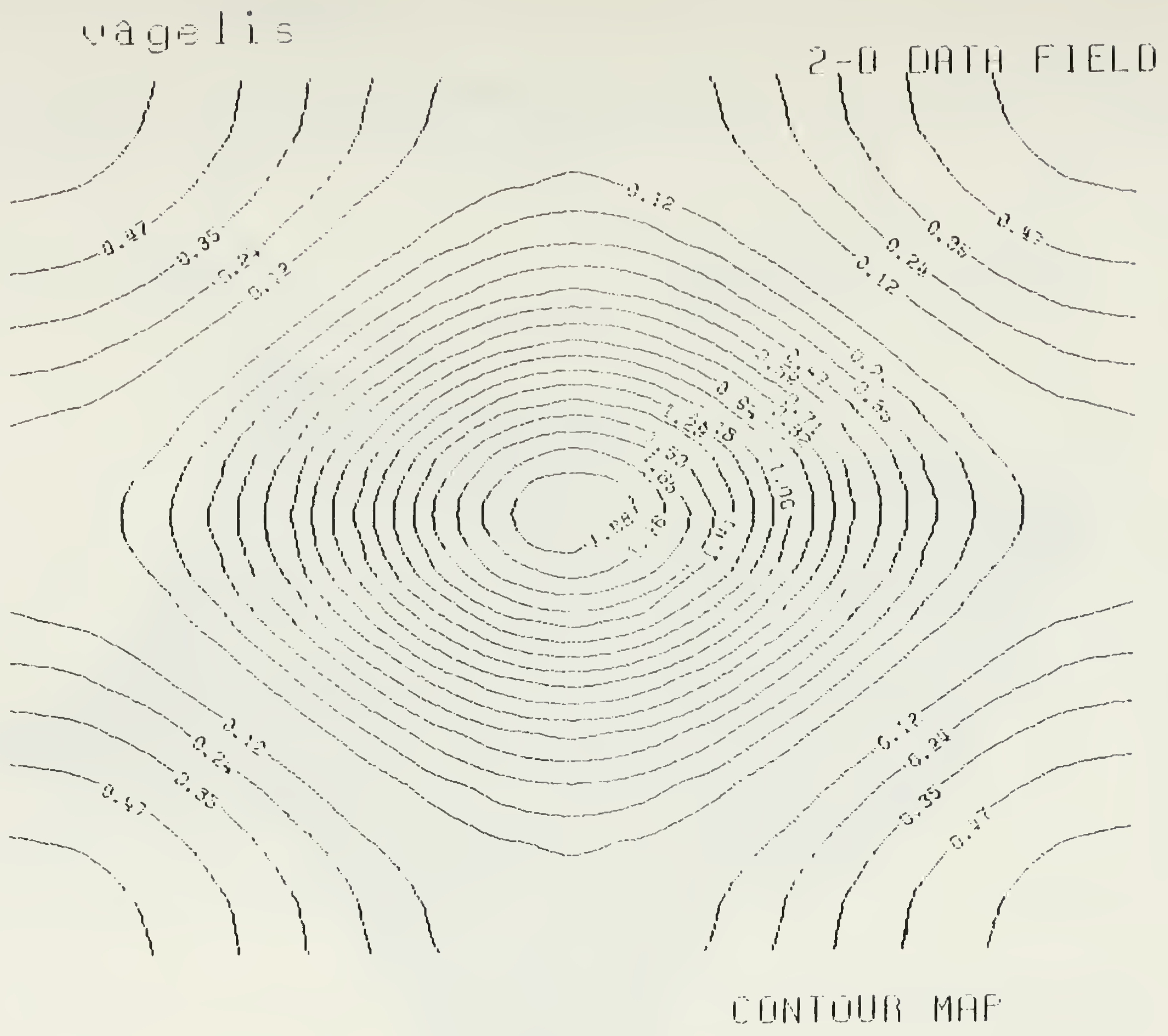


Figure 3-4b. Contour Map for Figure 3-4a

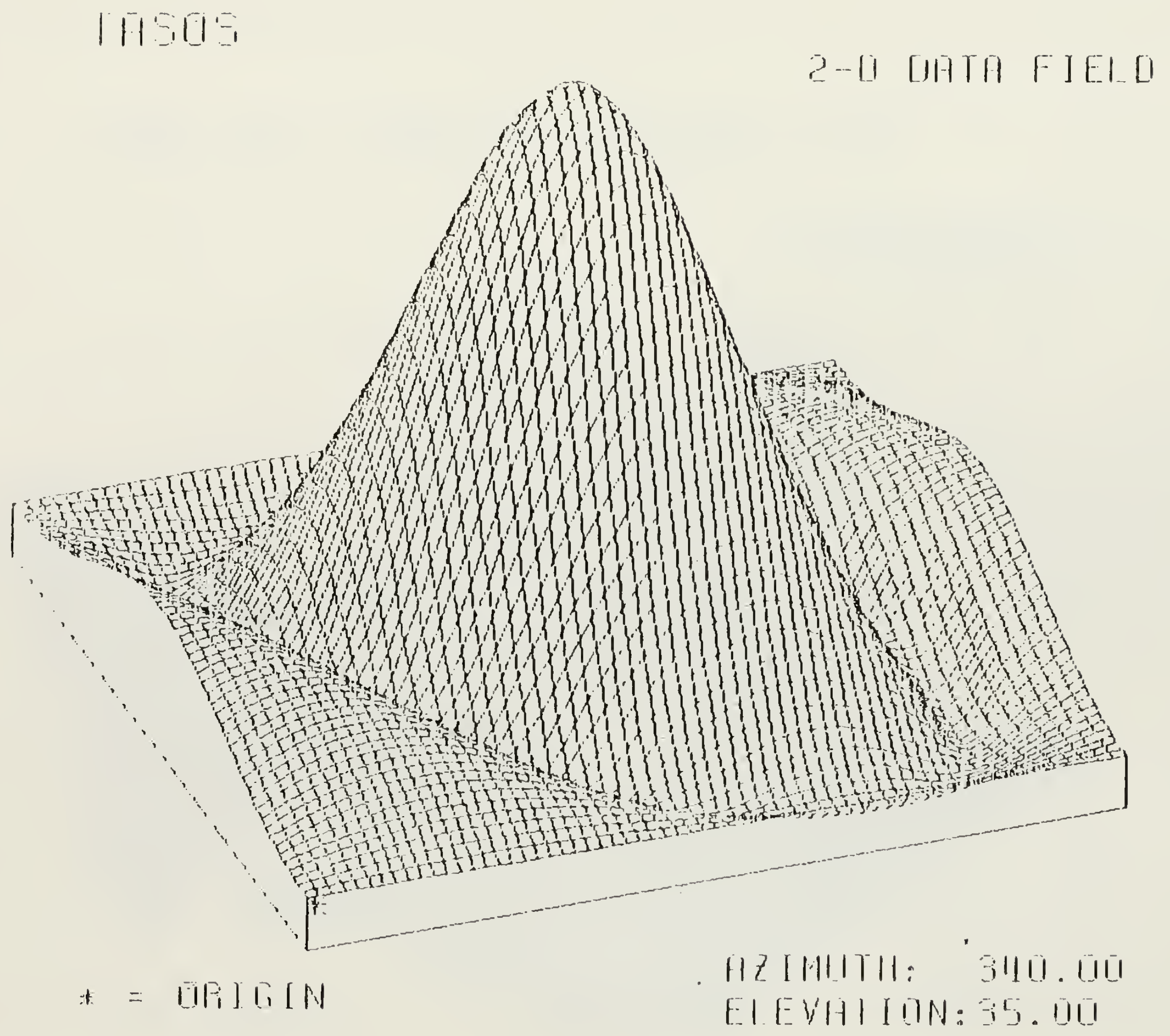


Figure 3-5a. Transfer Function $|H(z_1,z_2)|$, $z_1 = e^{j\omega_1}$, $z_2 = e^{j\omega_2}$ for Example 2

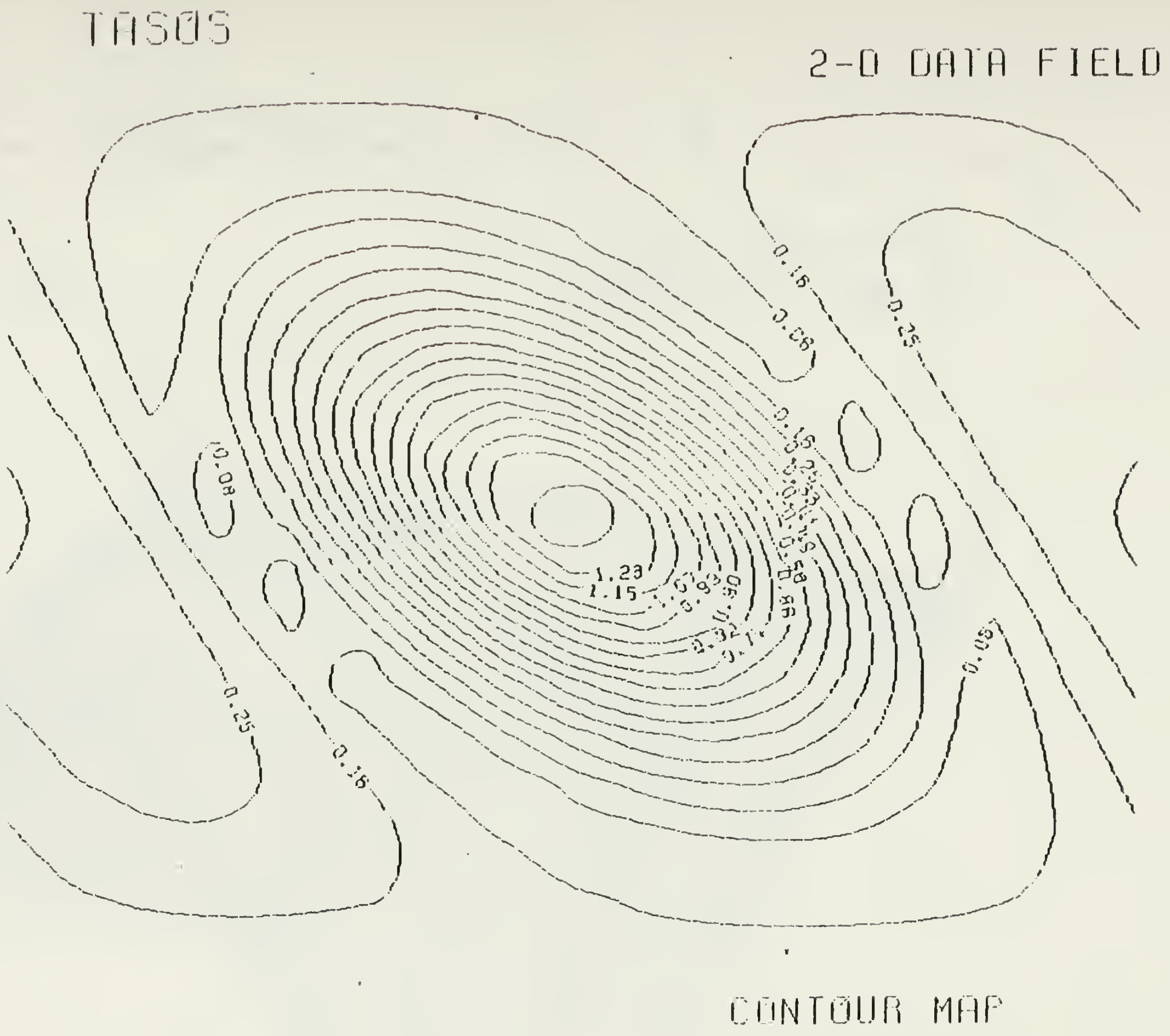


Figure 3-5b. Contour Map for Figure 3-5a

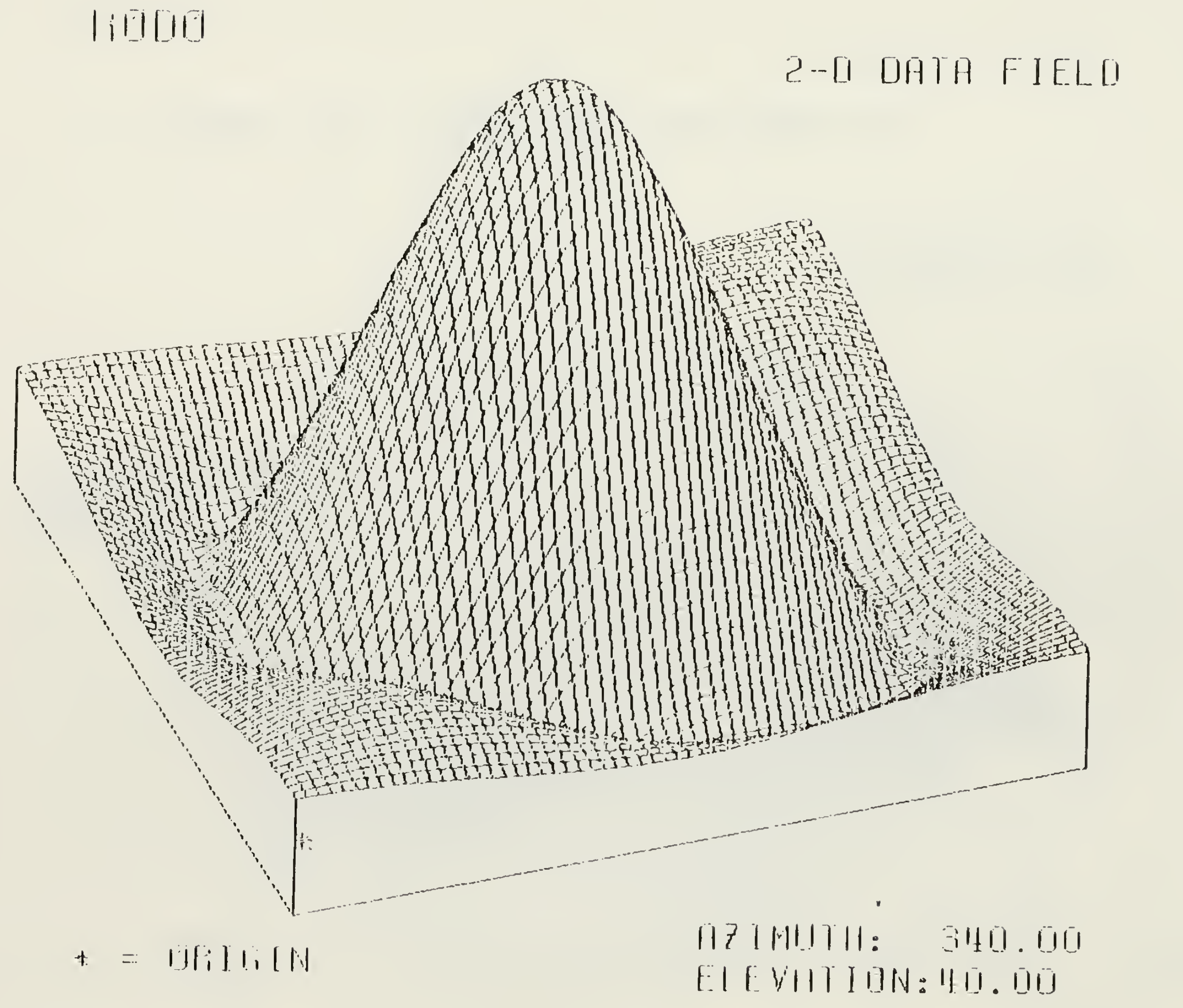


Figure 3-6a. Transfer Function $|H(z_1,z_2)|$, $z_1 = e^{j\omega_1}$, $z_2 = e^{j\omega_2}$ for Example 3

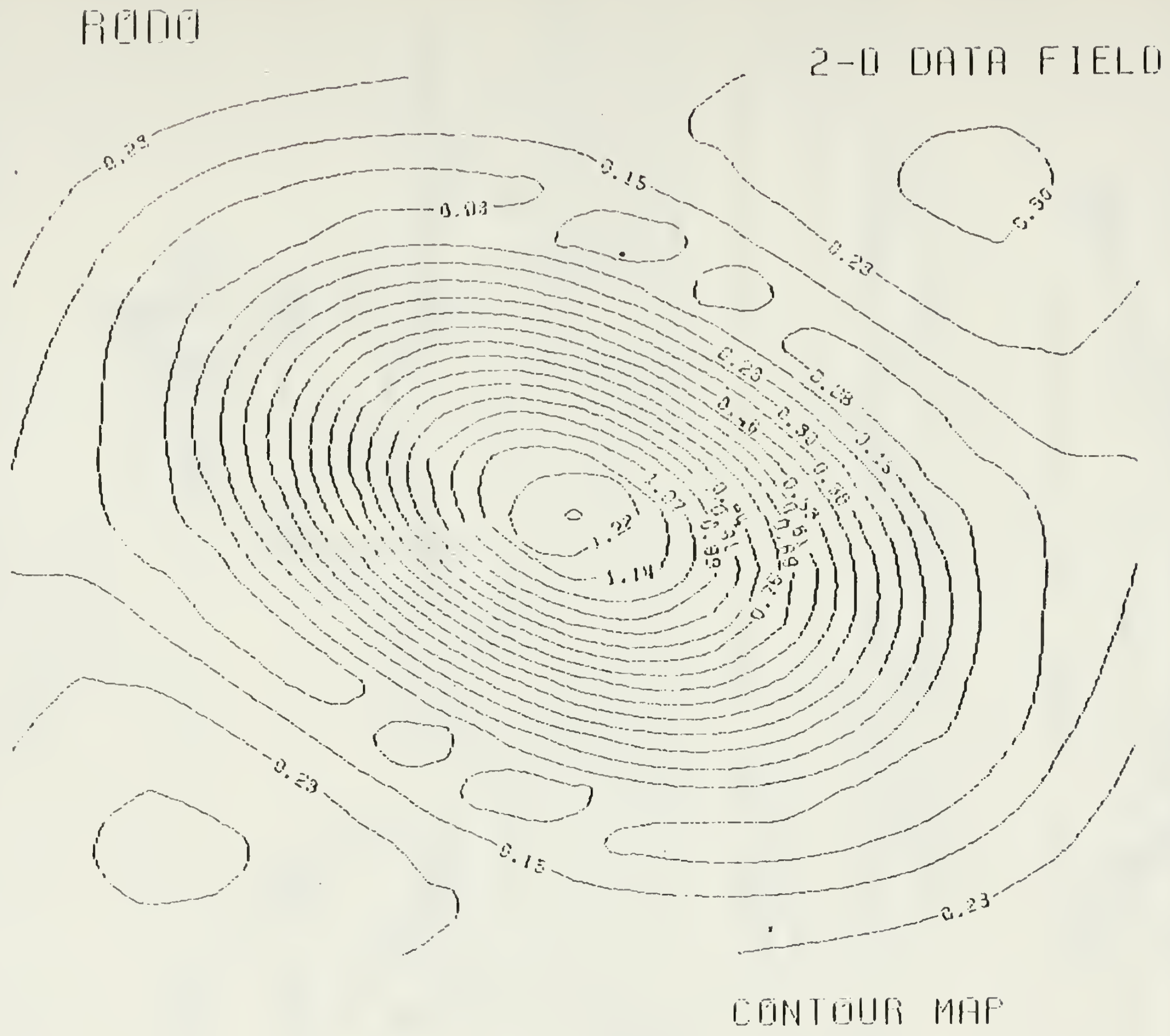


Figure 3-6b. Contour Map for Figure 3-6a

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

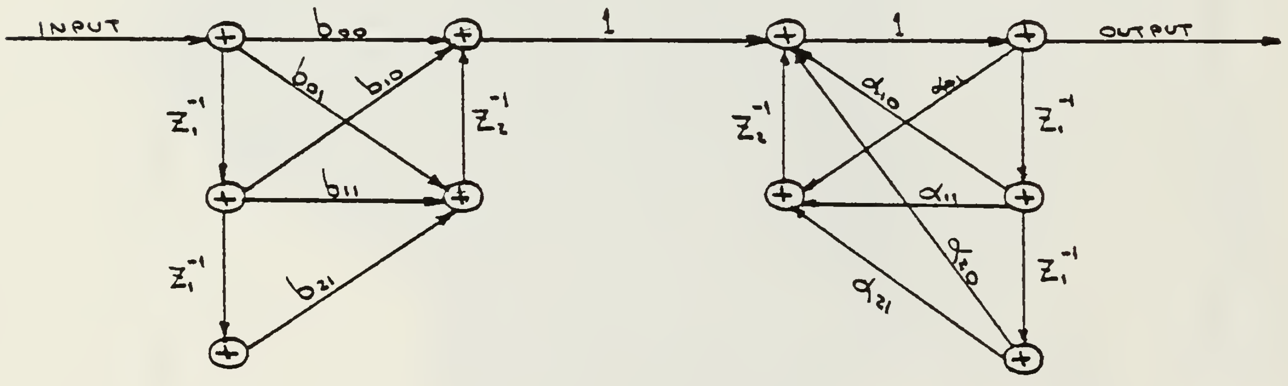


Figure 4-1

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.

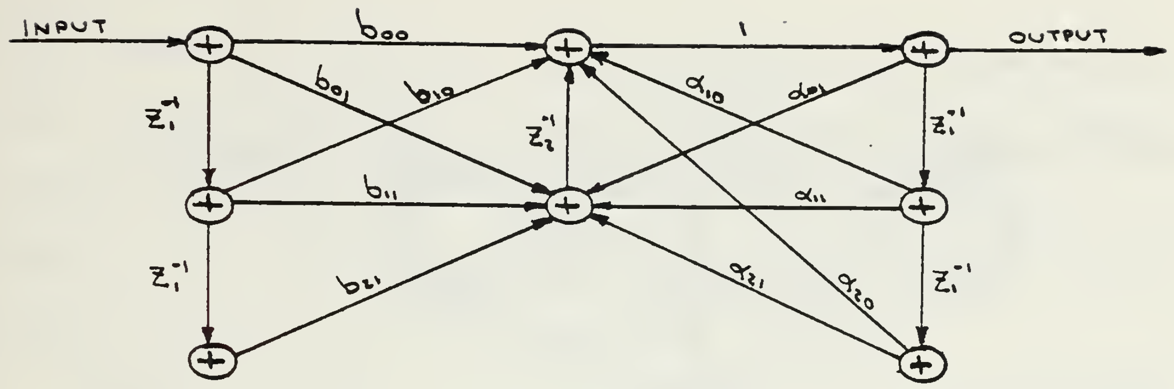


Figure 4-2

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:

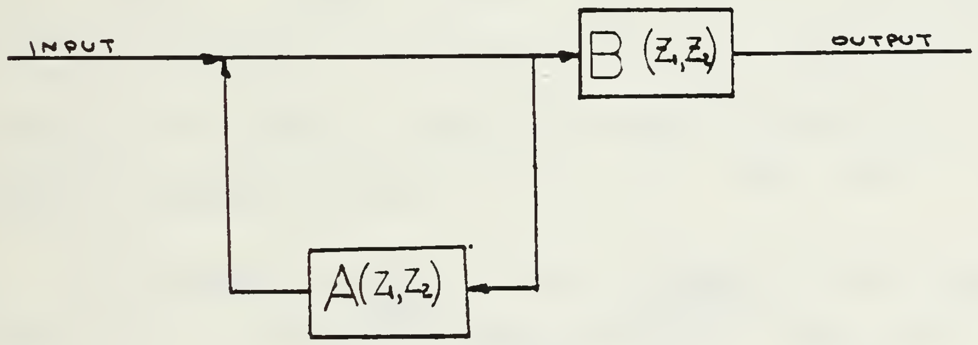


Figure 4-3

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.

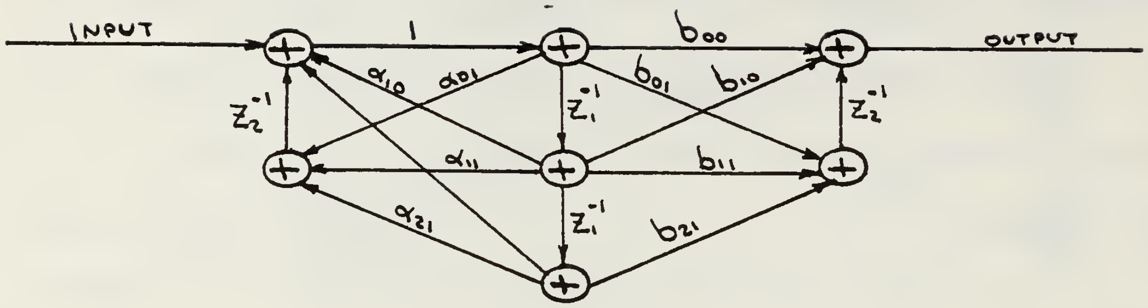


Figure 4-4

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.

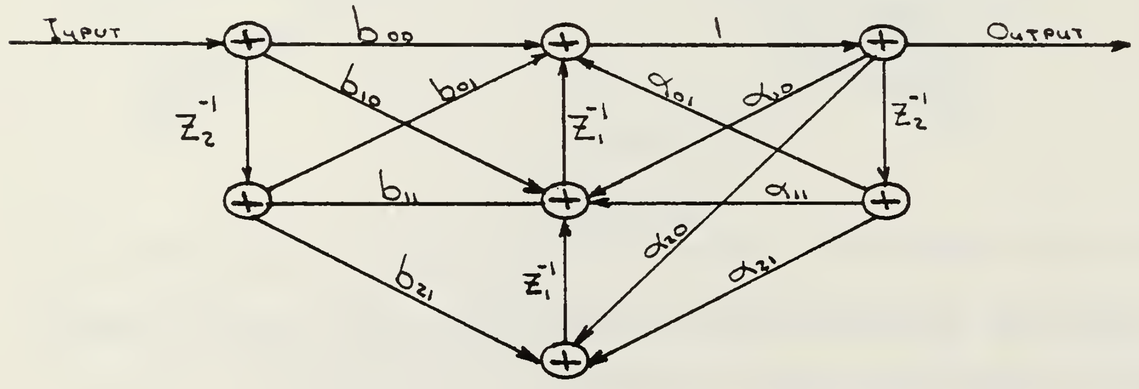


Figure 4-5

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 micro-processor 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 micro-processors 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:

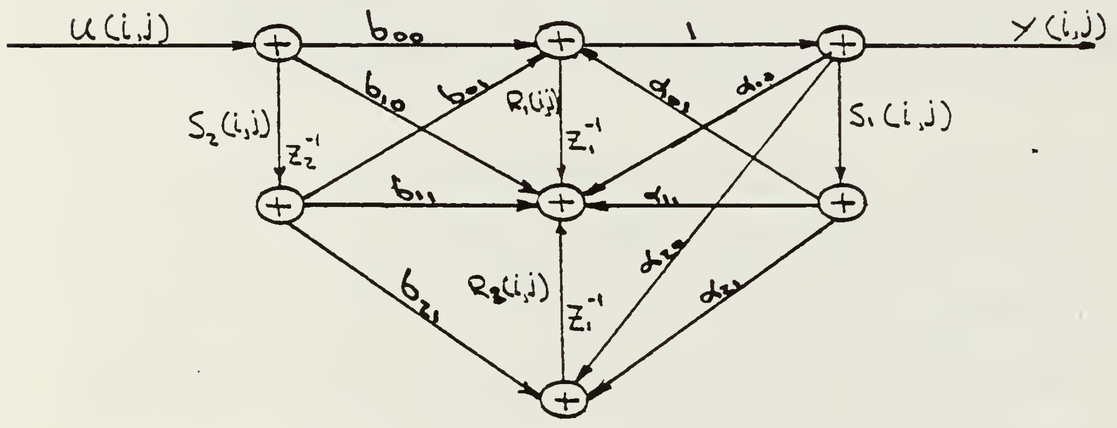


Figure 4-6

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

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

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

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

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):

$$\begin{array}{lllllll}
R_1(i+1,j) = & a_{10}R_1(i,j) + & R_2(i,j) + & (b_{11}+b_{01}a_{10})S_1'(i,j) + & (a_{11}+a_{01}a_{10})S_1^2(i,j) + & (b_{10}+b_{00}a_{10})u(i,j) \\
R_2(i+1,j) = & a_{20}R_1(i,j) + & 0 + & (b_{21}+b_{01}a_{20})S_1'(i,j) + & (a_{11}+a_{01}a_{20})S_1^2(i,j) + & (b_{00}a_{20})u(i,j) \\
S_1'(i,j+1) = & 0 & 0 & 0 & 0 + & 1u(i,j) \\
S_1^2(i,j+1) = & R_1(i,j) + & 0 & b_{01}S_1'(i,j) + & a_{01}S_1^2(i,j) + & b_{00}u(i,j)
\end{array} \qquad \text{(IV-4)}$$

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

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

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

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

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

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

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

$|Y(m,n)|$ 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.

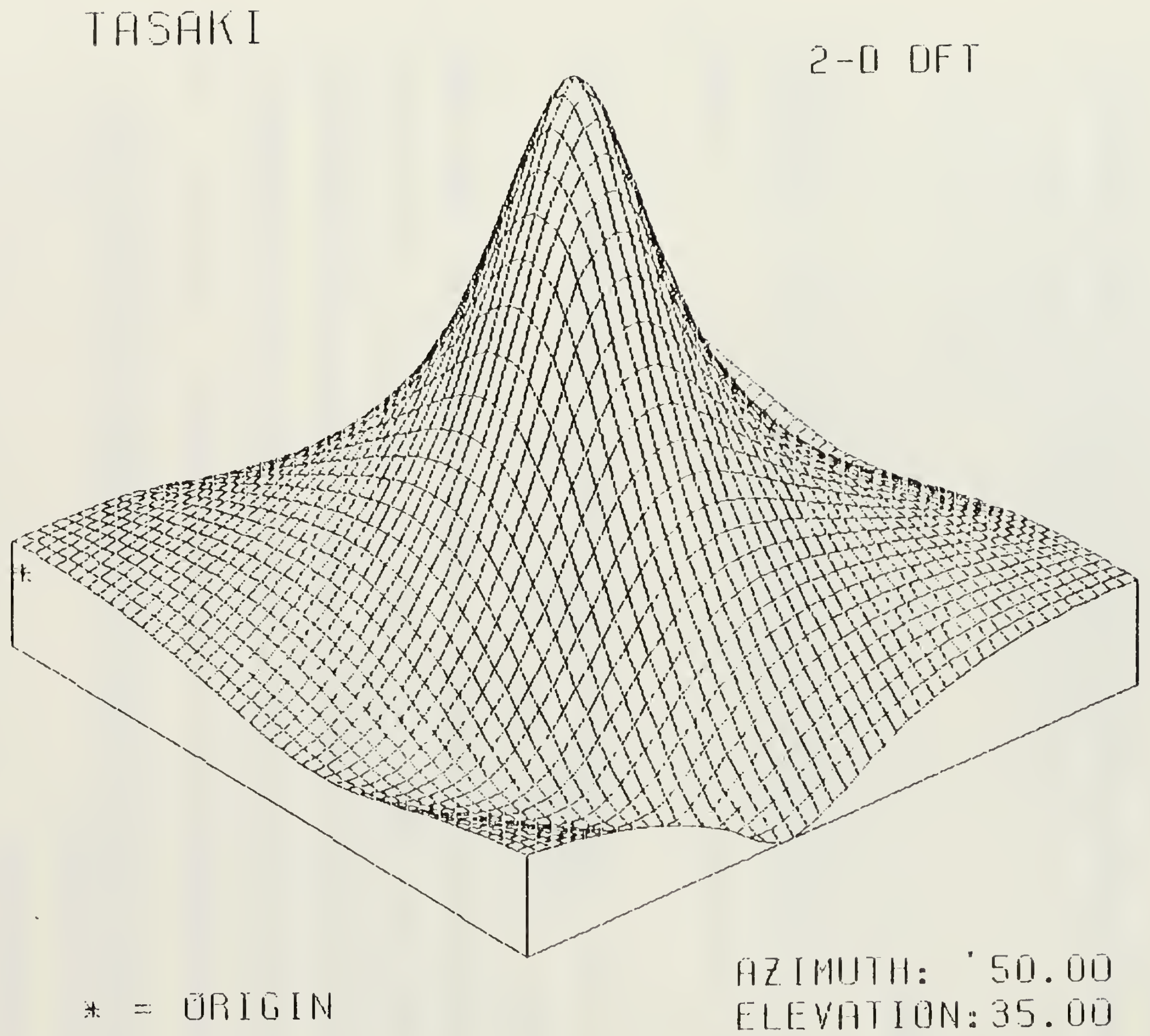


Figure 4-1a. 2-D D.F.T. Sequences, Y(m,n) for Example 4

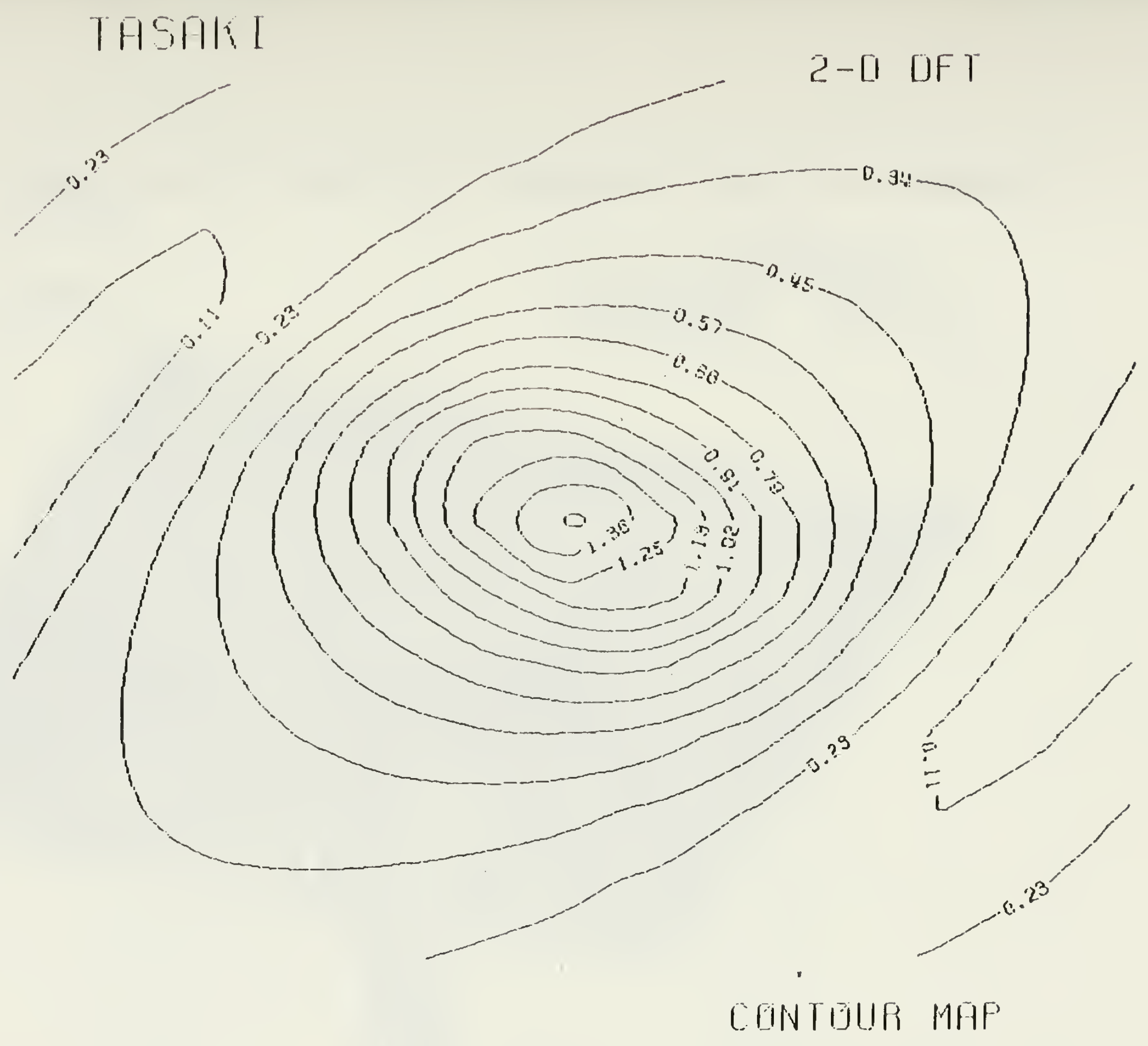


Figure 4-1b. Contour Map for Figure 4-1a

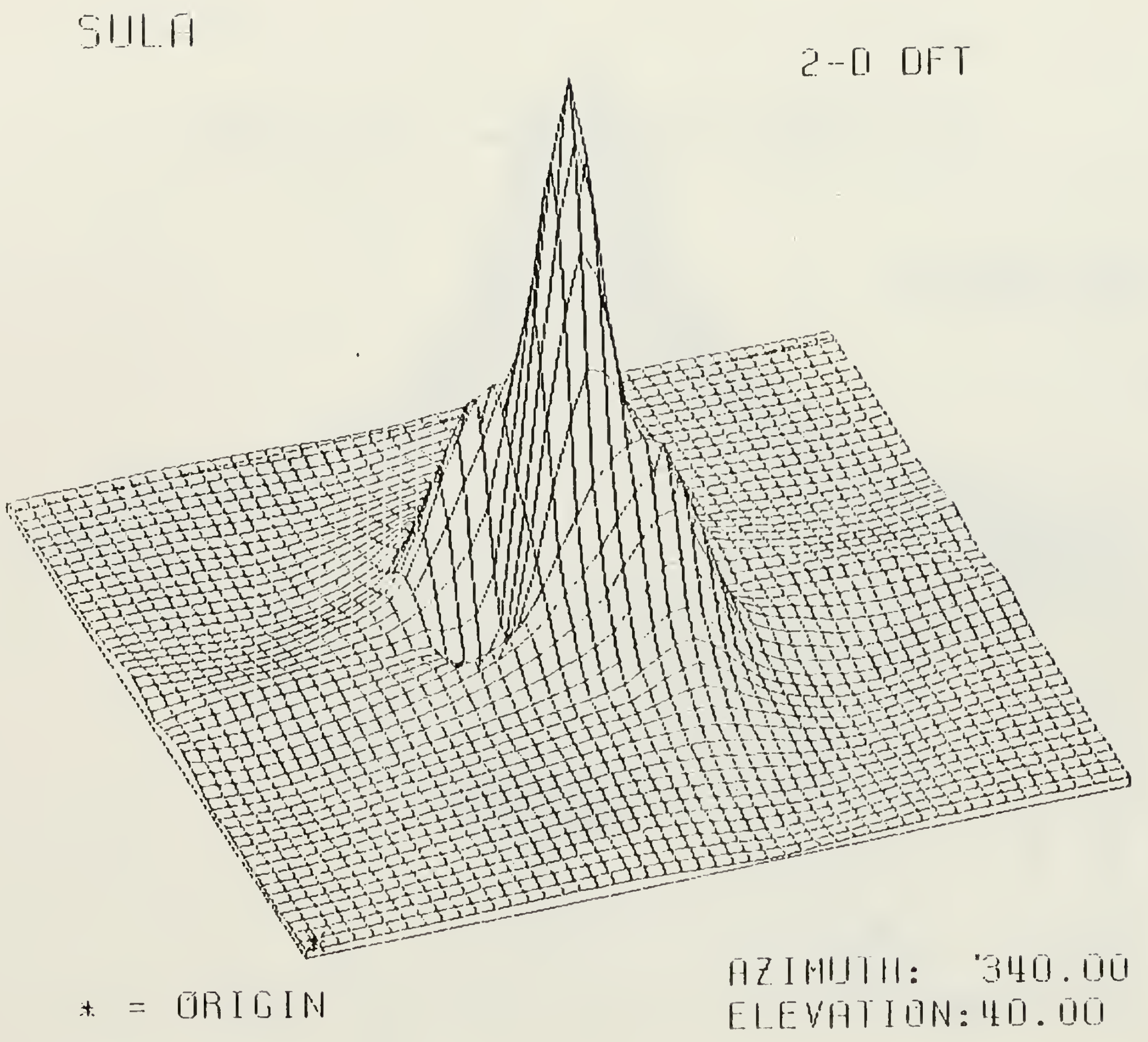


Figure 4-2a. 2-D D.F.T. Sequences Y(m,n) for Example 5

SULA

2-D DFT

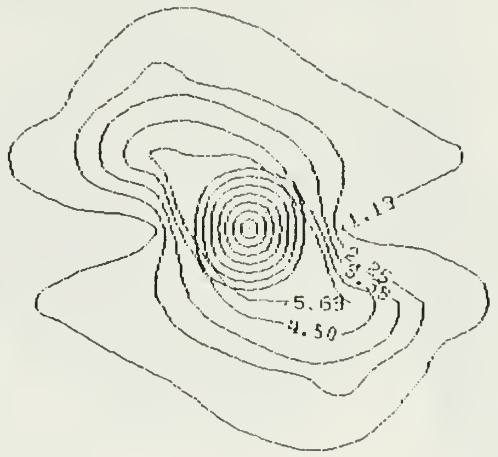


Figure 4-2b. Contour Map for Figure 4-2a

CONTOUR MAP

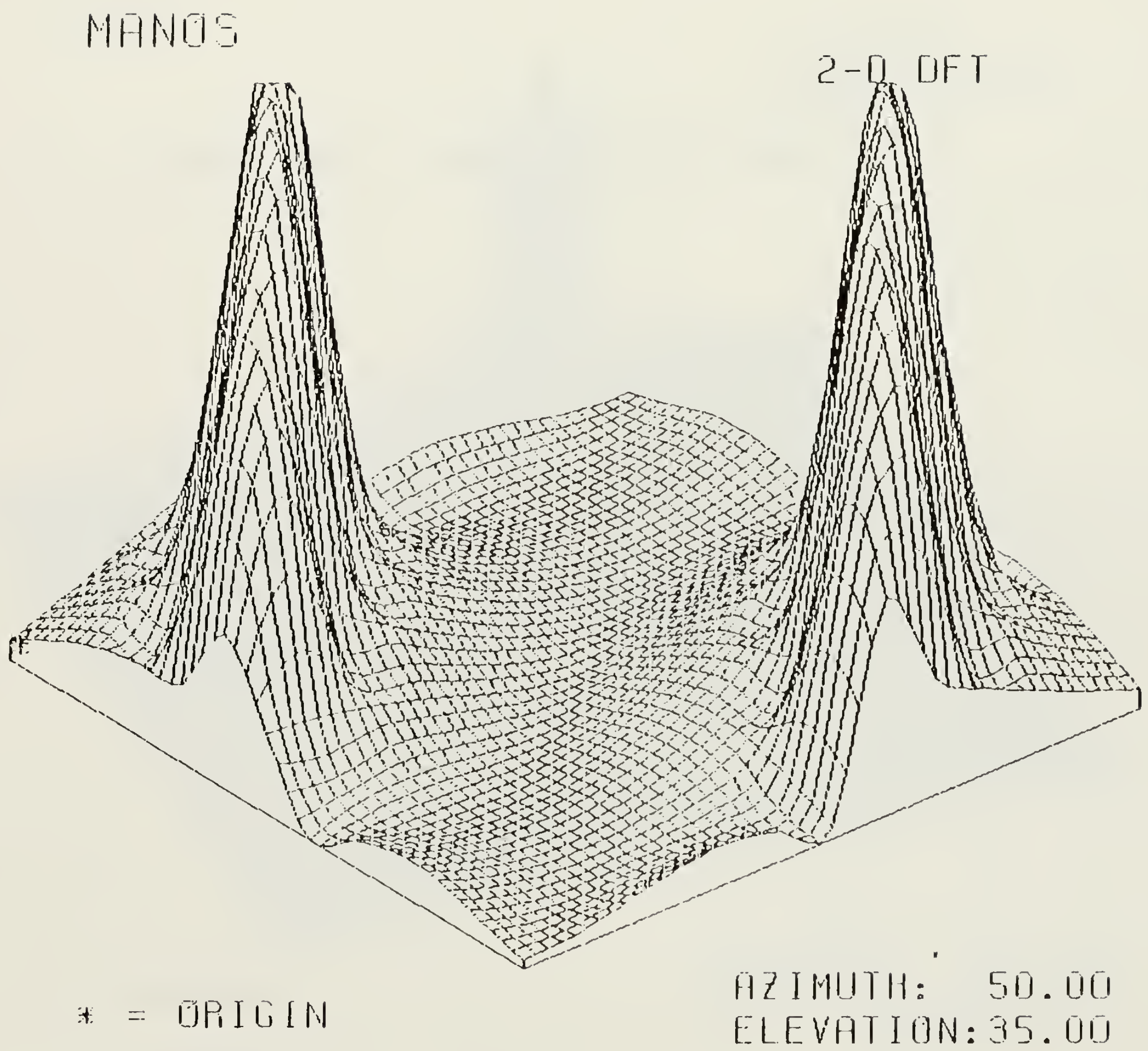


Figure 4-3a. 2-D D.F.T. Squences, Y(m,n) for Example 6

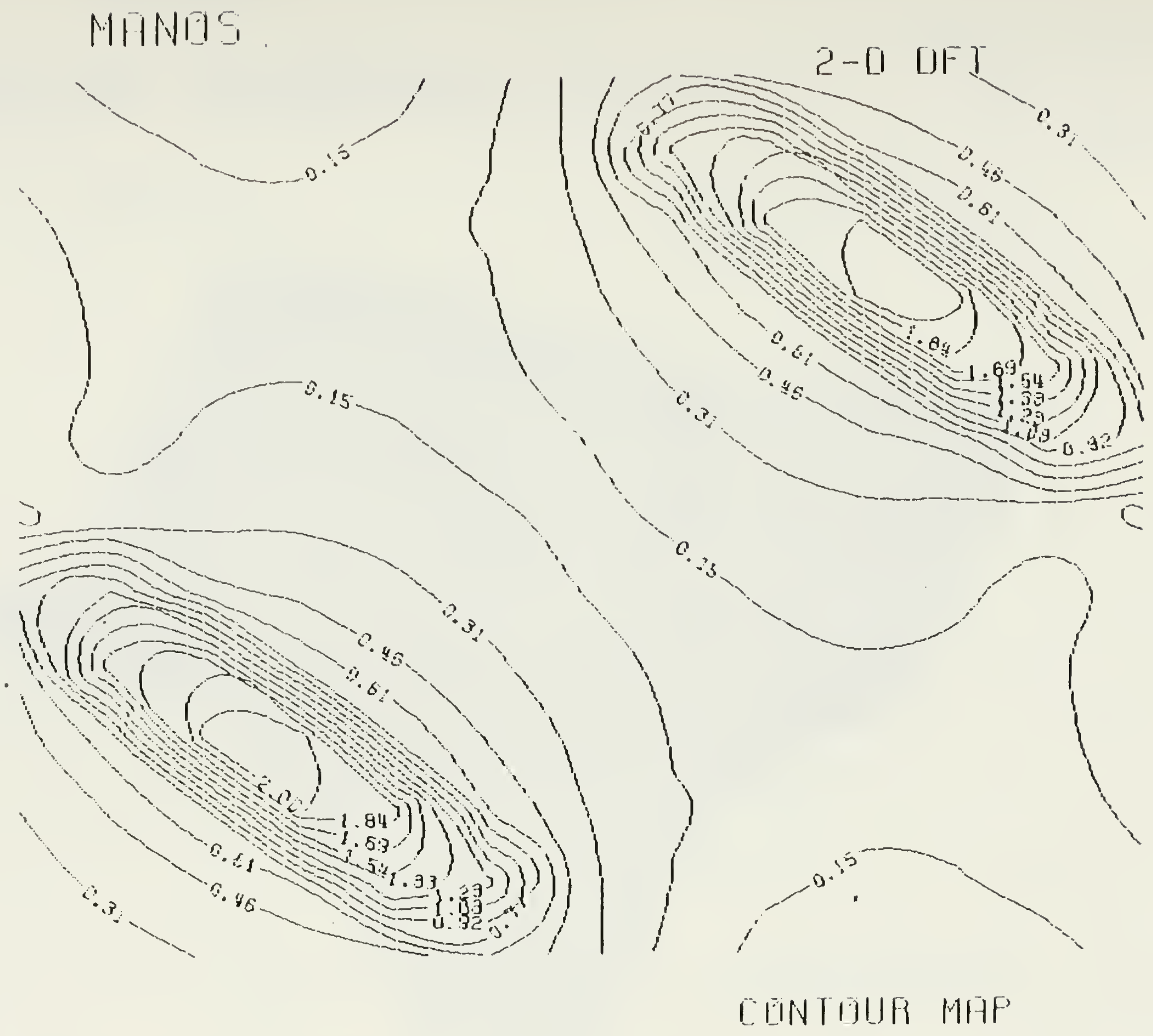


Figure 4-3b. Contour Map for Figure 4-3a

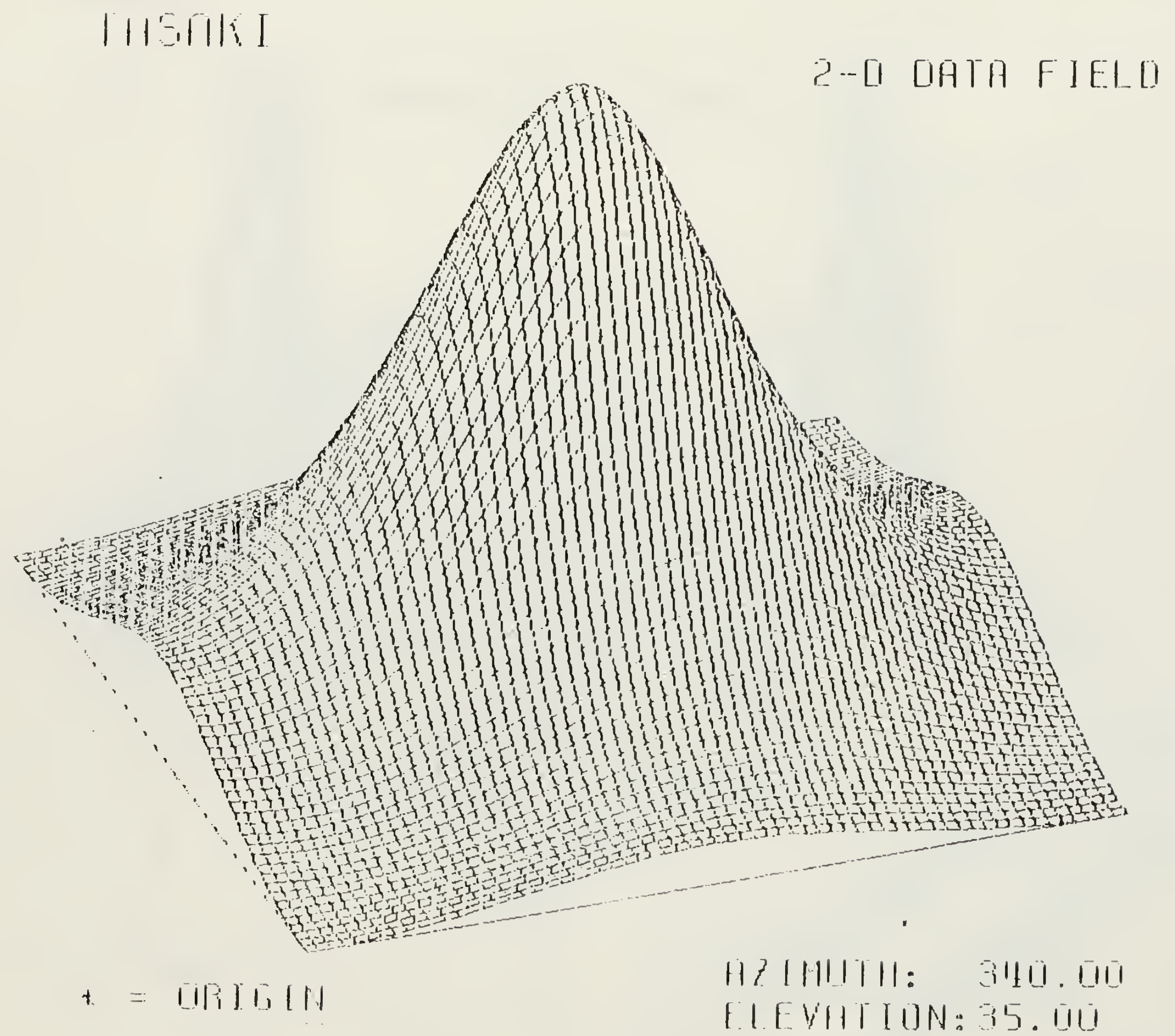


Figure 4-4a. Transfer Function $|H(z_1,z_2)|$, $z_1 = e^{j\omega_1}$, $z_2 = e^{j\omega_2}$ for Example 4

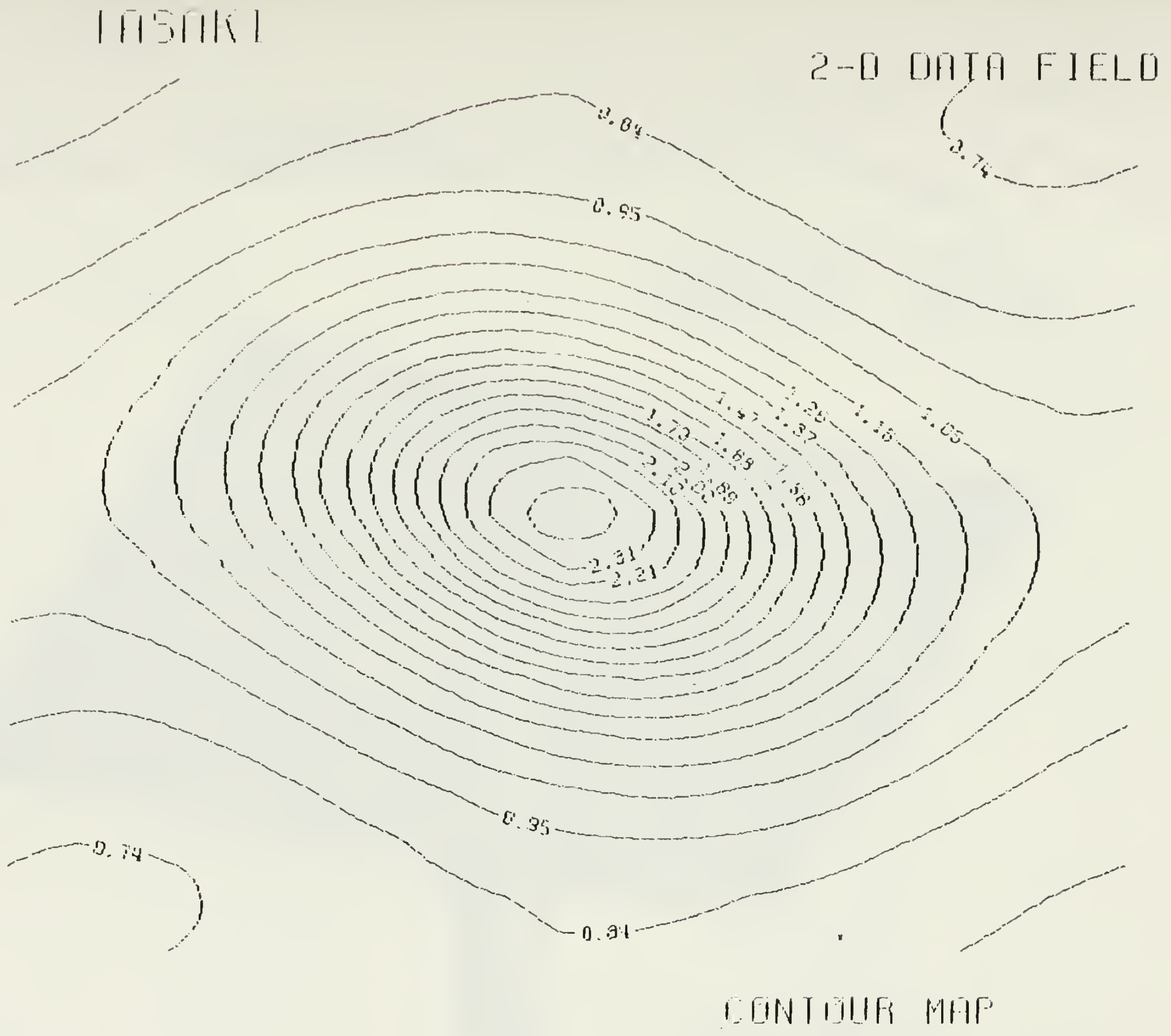


Figure 4-4b. Contour Map for Figure 4-4a

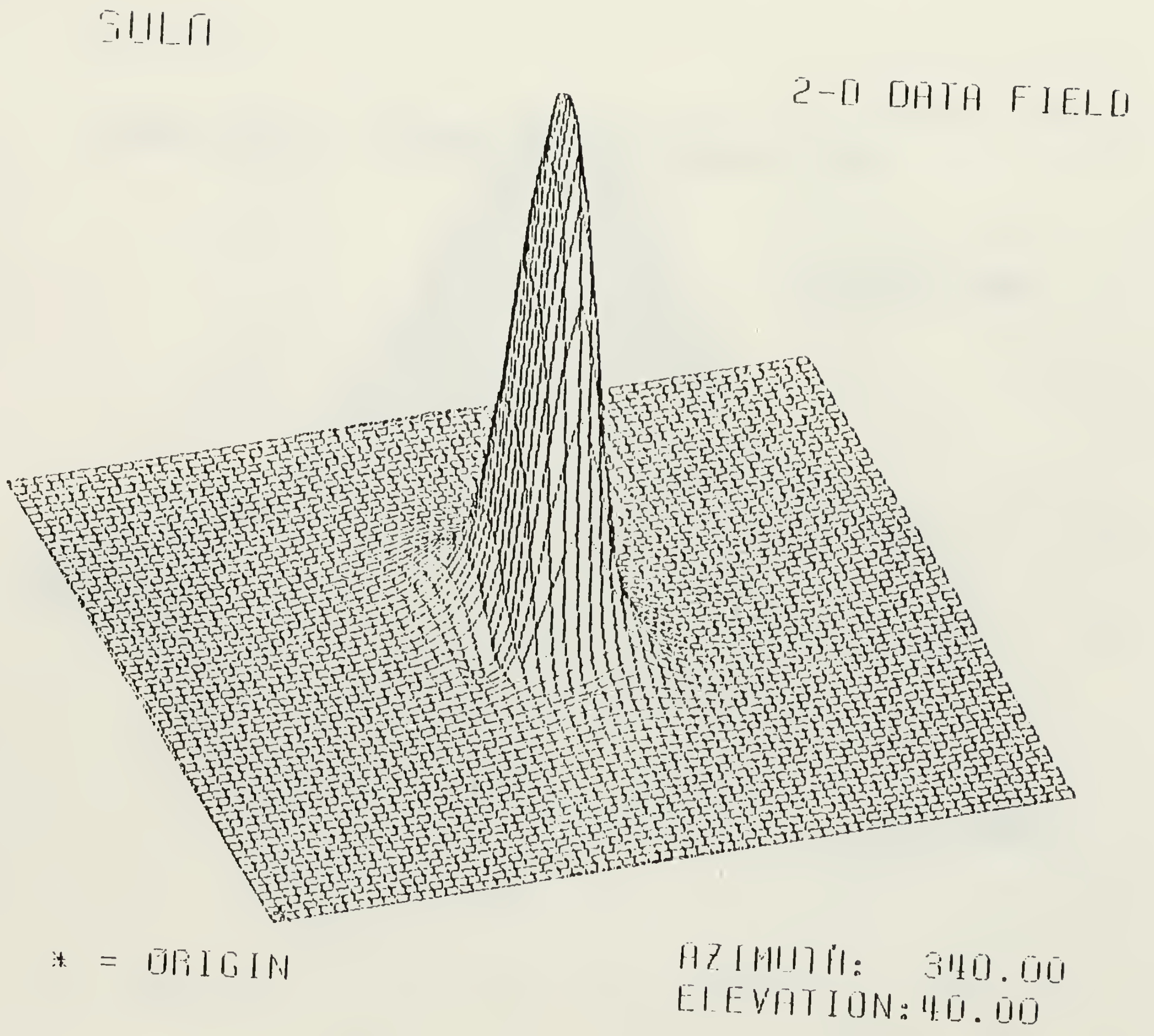


Figure 4-5a. Transfer Function $|H(z_1,z_2)|$, $z_1 = e^{j\omega_1}$, $z_2 = e^{j\omega_2}$ for Example 5 .

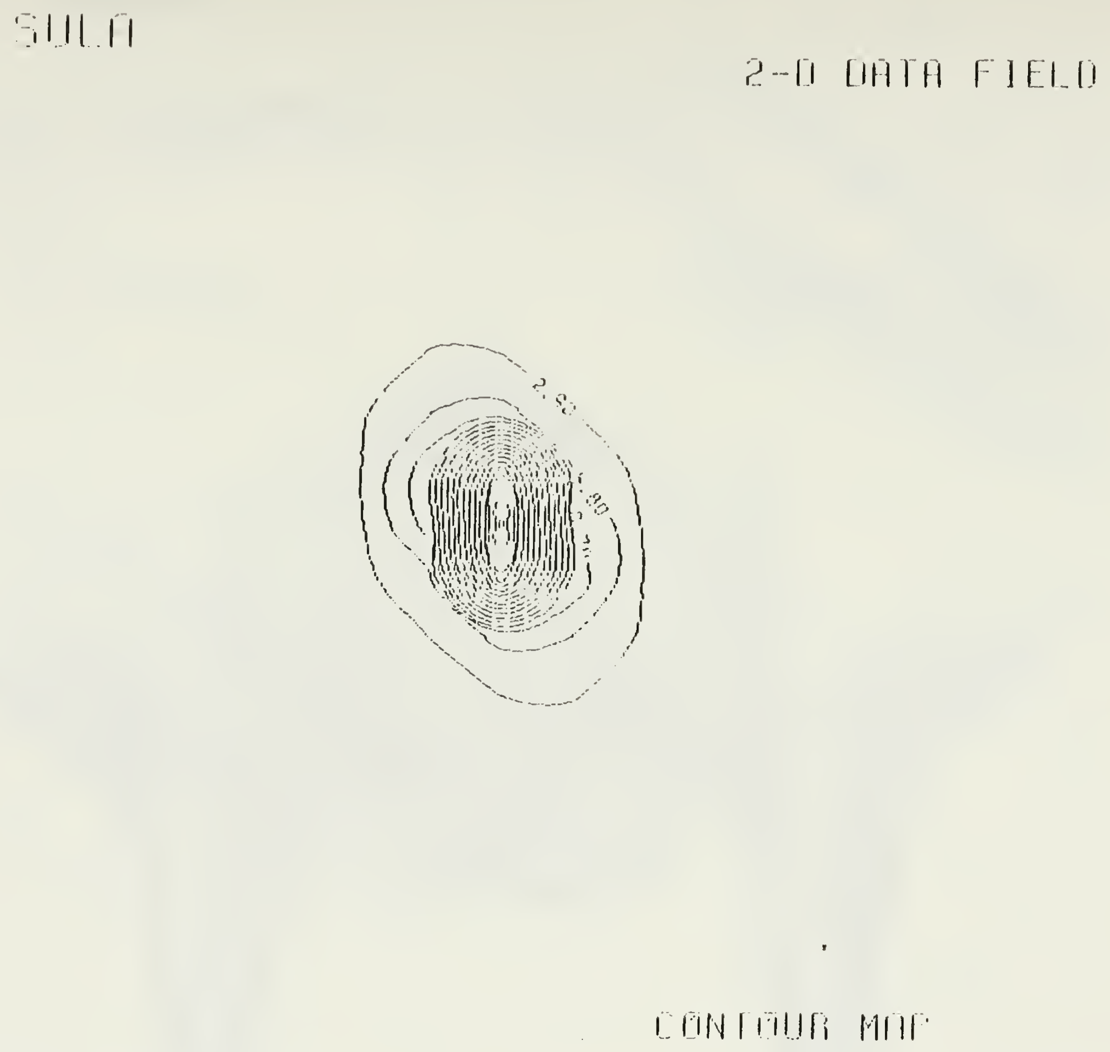


Figure 4-5b. Contour Map for Figure 4-5a

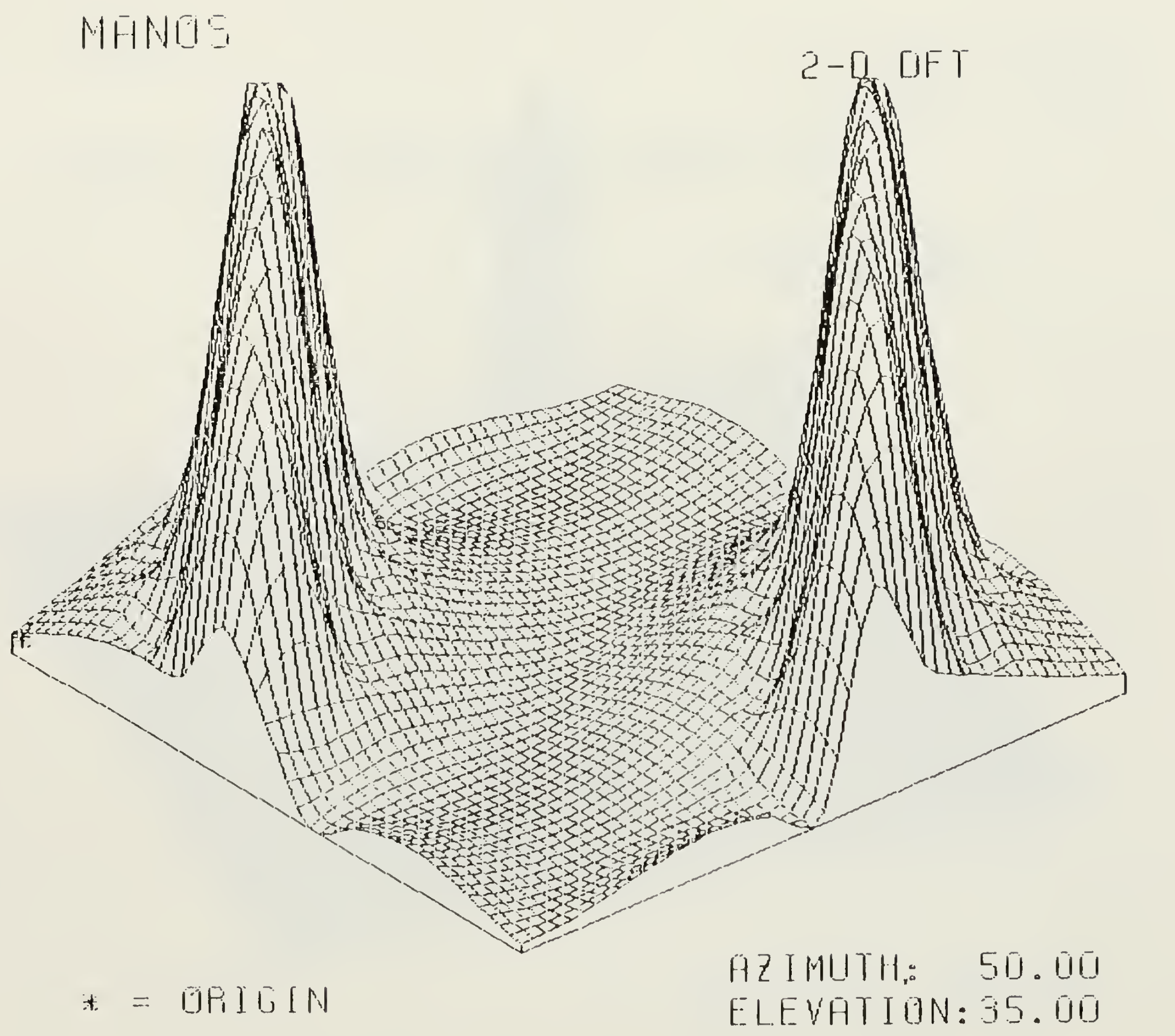


Figure 4-6a. Transfer Function $|H(z_1,z_2)|$, $Z_1 = e^{j\omega_1}$, $Z_2 = e^{j\omega_2}$ for Example 6

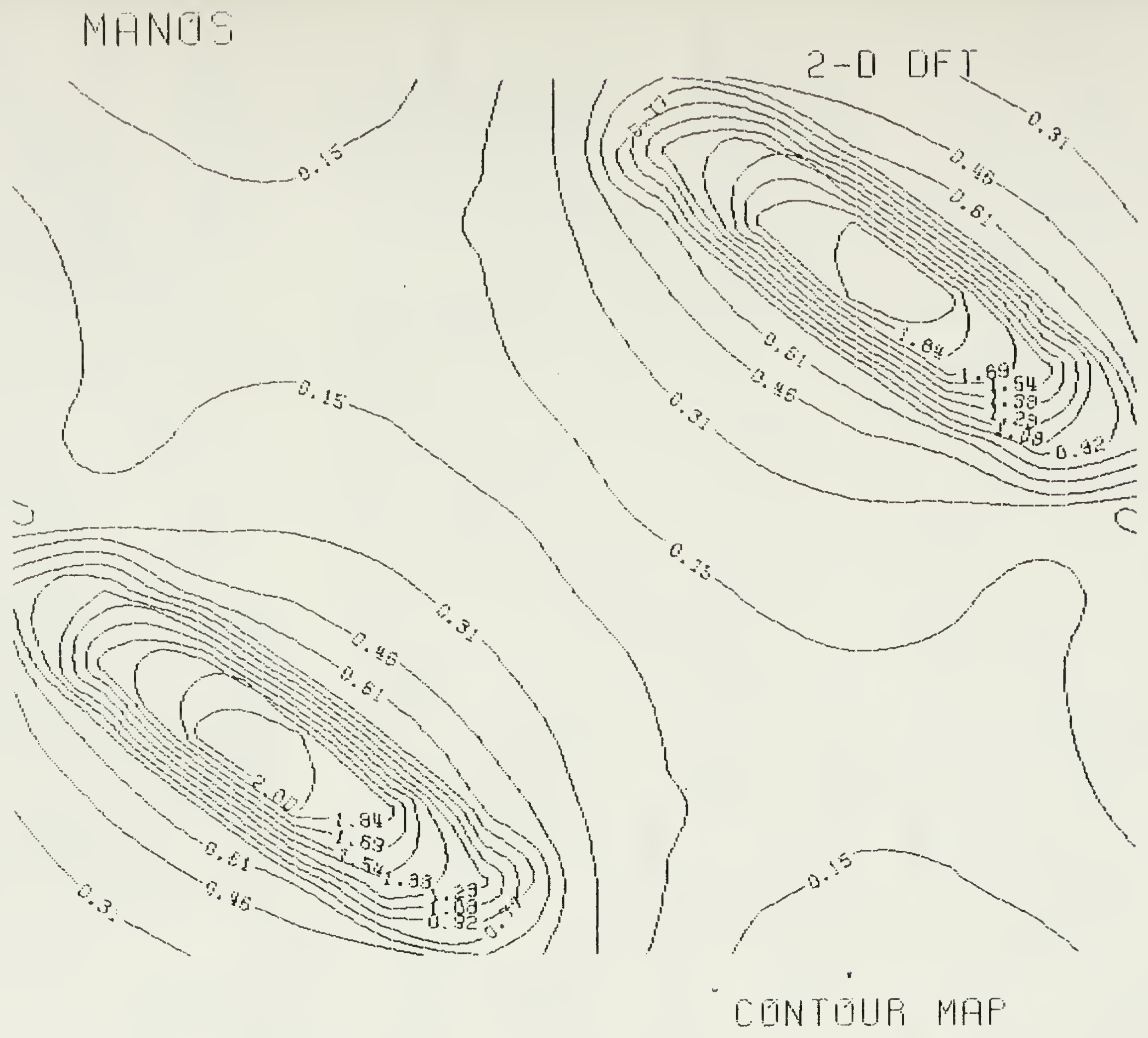


Figure 4-6b. Contour Map for Figure 4-6a

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.

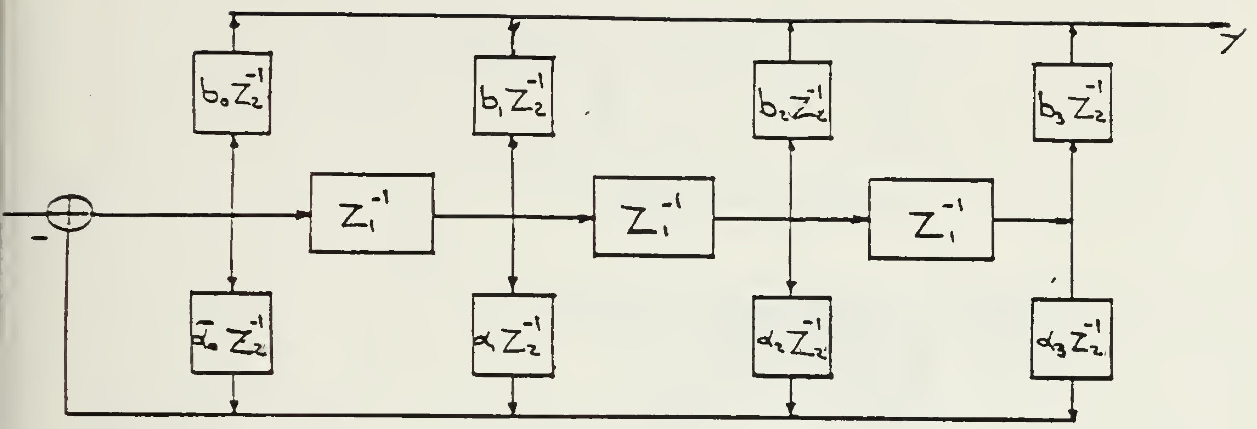


Figure 4-7

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.

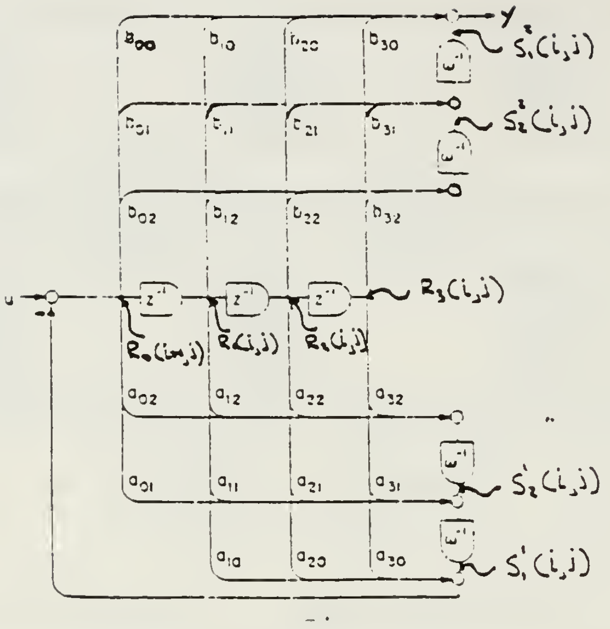


Figure 4-8

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

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

where:

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

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

$$
\text{Transition Matrix} \quad A = \left[\begin{array}{cccc:c:ccc:ccc}
-a_{10} & -a_{20} & \cdots & -a_{n0} & -1 & & & & & & \\
1 & & & & & & & & & & \\
& \ddots & & \bigcirc & & & \bigcirc & & & \bigcirc & \\
\bigcirc & & \ddots & 0 & & & & & & & \\
& & & 1 & & & & & & & \\
\hdashline
\tilde{a}_{11} & & & \tilde{a}_{n1} & -a_{01} & 1 & & & & & \\
& & & & \vdots & & \ddots & \bigcirc & & & \\
& \tilde{a}_{ij} & & & \vdots & & & \ddots & & \bigcirc & \\
& & & & \vdots & \bigcirc & & 1 & & & \\
\tilde{a}_{1m} & & & \tilde{a}_{nm} & -a_{0m} & & & 0 & & & \\
\hdashline
\tilde{b}_{11} & & & \tilde{b}_{n1} & -b_{01} & & & & 0 & 1 & \\
& & & & \vdots & & & & & \ddots & \bigcirc \\
& \tilde{b}_{ij} & & & \vdots & & \bigcirc & & & & \ddots \\
& & & & \vdots & & & & \bigcirc & & 1 \\
\tilde{b}_{1m} & & & \tilde{b}_{nm} & -b_{0m} & & & & & & 0
\end{array}\right]
$$

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

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

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

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

$$\begin{bmatrix} R'_1(i+1,j) \\ R'_2(i+1,j) \\ R'_3(i+1,j) \\ \vdots \\ R'_n(i+1,j) \\ S'_1(i,j+1) \\ S'_2(i,j+1) \\ S'_3(i,j+1) \\ \vdots \\ S'_M(i,j+1) \\ S^2_1(i,j+1) \\ S^2_2(i,j+1) \\ S^2_3(i,j+1) \\ \vdots \\ S^2_M(i,j+1) \end{bmatrix} = \begin{array}{c|ccccc:ccccc:ccccc} & \longleftarrow & & n & & \longrightarrow & \longleftarrow & & M & & \longrightarrow & \longleftarrow & & M & & \longrightarrow \\ \hline & -a_{10} & -a_{20} & -a_{30} & \cdots & -a_{n0} & -1 & 0 & 0 & \cdots & 0 & & & & & \\ & 1 & 0 & 0 & \cdots & 0 & 0 & 0 & 0 & \cdots & 0 & & & & & \\ n & 0 & 1 & 0 & \cdots & 0 & 0 & 0 & 0 & \cdots & 0 & & & O & & \\ & \vdots & \vdots & 1 & & \vdots & \vdots & & & & \vdots & & & & & \\ & 0 & 0 & \cdots & 1 & 0 & 0 & 0 & 0 & \cdots & 0 & & & & & \\ \hdashline & \tilde{\alpha}_{11} & \tilde{\alpha}_{21} & \tilde{\alpha}_{31} & \cdots & \tilde{\alpha}_{n1} & -\alpha_{01} & 1 & 0 & \cdots & 0 & & & & & \\ & \tilde{\alpha}_{12} & \tilde{\alpha}_{22} & \tilde{\alpha}_{32} & \cdots & \tilde{\alpha}_{n2} & -\alpha_{02} & 0 & 1 & \cdots & 0 & & & & & \\ M & \tilde{\alpha}_{13} & \tilde{\alpha}_{23} & \tilde{\alpha}_{33} & \cdots & \tilde{\alpha}_{n3} & -\alpha_{03} & 0 & 0 & 1 & 0 & & & O & & \\ & \vdots & \vdots & \vdots & & \vdots & \vdots & \vdots & \vdots & \vdots & 1 & & & & & \\ & \tilde{\alpha}_{1M} & \tilde{\alpha}_{2M} & \tilde{\alpha}_{3M} & \cdots & \tilde{\alpha}_{nM} & -\alpha_{0M} & 0 & 0 & 0 \cdots & 0 & & & & & \\ \hdashline & \tilde{b}_{11} & \tilde{b}_{21} & \tilde{b}_{31} & \cdots & \tilde{b}_{n1} & -b_{01} & 0 & 0 & \cdots & 0 & 0 & 1 & 0 & \cdots & 0 \\ & \tilde{b}_{12} & \tilde{b}_{22} & \tilde{b}_{32} & \cdots & \tilde{b}_{n2} & -b_{02} & 0 & 0 & \cdots & 0 & 0 & 0 & 1 & \cdots & 0 \\ M & \tilde{b}_{13} & \tilde{b}_{23} & \tilde{b}_{33} & \cdots & \tilde{b}_{n3} & -b_{03} & 0 & 0 & \cdots & 0 & 0 & 0 & 0 & 1 & 0 \\ & \vdots & \vdots & \vdots & & \vdots & \vdots & \vdots & \vdots & & & \vdots & \vdots & \vdots & \vdots & 1 \\ & \tilde{b}_{1M} & \tilde{b}_{2M} & \tilde{b}_{3M} & \cdots & \tilde{b}_{nM} & -b_{0M} & 0 & 0 & \cdots & 0 & 0 & 0 & 0 & 0 \cdots & 0 \end{array} \begin{bmatrix} R_1(i,j) \\ R_2(i,j) \\ R_3(i,j) \\ \vdots \\ R_n(i,j) \\ S'_1(i,j) \\ S'_2(i,j) \\ S'_3(i,j) \\ \vdots \\ S'_M(i,j) \\ S^2_1(i,j) \\ S^2_2(i,j) \\ S^2_3(i,j) \\ \vdots \\ S^2_M(i,j) \end{bmatrix} + \begin{bmatrix} 1 \\ 0 \\ 0 \\ \vdots \\ 0 \\ \alpha_{01} \\ \alpha_{02} \\ \alpha_{03} \\ \vdots \\ \alpha_{0M} \\ b_{01} \\ b_{02} \\ b_{03} \\ \vdots \\ b_{0M} \end{bmatrix} \qquad \text{(IV-12a)}$$

$$(n+2M) \times (n+2M)$$

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

(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

$R_1(1,1), R_2(1,1), R_3(1,1) R_4(1,1)$ $S^1_1(1,1), S^1_2(1,1), S^1_3(1,1)$ $S^2_1(1,1), S^2_2(1,1), S^2_3(1,1)$	$R_1(1,2), R_2(1,2), R_3(1,2), R_4(1,2)$ $S^1_1(1,2), S^1_2(1,2), S^1_3(1,2)$ $S^2_1(1,2), S^2_1(1,2), S^2_3(1,2)$	$R_1(1,3), R_2(1,3), R_3(1,3), R_4(1,3)$ $S^1_1(1,3), S^1_2(1,3), S^1_3(1,3)$ $S^2_1(1,3), S^2_2(1,3), S^2_3(1,3)$	$R_1(1,4), R_2(1,4), R_3(1,4), R_4(1,$ $S^1_1(1,4), S^1_2(1,4), S^1_3(1,4)$ $S^2_1(1,4), S^2_2(1,4), S^2_3(1,4)$
$R_1(2,1), R_2(2,1), R_3(2,1), R_4(2$ $S^1_1(2,1), S^1_2(2,1), S^1_3(2,1)$ $S^2_1(2,1), S^2_2(2,1), S^2_3(2,1)$	$R_1(2,2), R_2(2,2), R_3(2,2), R_4(2,2)$ $S^1_1(2,2), S^1_2(2,2), S^1_3(2,2)$ $S^2_1(2,2), S^2_2(2,2), S^2_3(2,2)$	$R_1(2,3), R_2(2,3), R_3(2,3), R_4(2,3)$ $S^1_1(2,3), S^1_2(2,3), S^1_3(2,3)$ $S^2_1(2,3), S^2_2(2,3), S^2_3(2,3)$	$R_1(2,4), R_2(2,4), R_3(2,4), R_4(2,4)$ $S^1_1(2,4), S^1_2(2,4), S^1_3(2,4)$ $S^2_1(2,4), S^2_2(2,4), S^2_3(2,4)$
$R_1(3,1), R_2(3,1), R_3(3,1), R_4(3,$ $S^1_1(3,1), S^1_2(3,1), S^1_3(3,1)$ $S^2_1(3,1), S^2_2(3,1), S^2_3(3,1)$	$R_1(3,2), R_2(3,2), R_3(3,2), R_4(3,2)$ $S^1_1(3,2), S^1_2(3,2), S^1_3(3,2)$ $S^2_1(3,2), S^2_2(3,2), S^2_3(3,2)$	$R_1(3,3), R_2(3,3), R_3(3,3), R_4(3,3)$ $S^1_1(3,3), S^1_2(3,3), S^1_3(3,3)$ $S^2_1(3,3), S^2_2(3,3), S^2_3(3,3)$	$R_1(3,4), R_2(3,4), R_3(3,4), R_4(3,4)$ $S^1_1(3,4), S^1_2(3,4), S^1_3(3,4)$ $S^2_1(3,4), S^2_2(3,4), S^2_3(3,4)$
$R_1(4,1), R_2(4,1), R_3(4,1), R_4(4,1)$ $S^1_1(4,1), S^1_2(4,1), S^1_3(4,1)$ $S^2_1(4,1), S^2_2(4,1), S^2_3(4,1)$	$R_1(4,2), R_2(4,2), R_3(4,2), R_4(4,2)$ $S^1_1(4,2), S^1_2(4,2), S^1_3(4,2)$ $S^2_1(4,2), S^2_2(4,2), S^2_3(4,2)$	$R_1(4,3), R_2(4,3), R_3(4,3), R_4(4,3)$ $S^1_1(4,3), S^1_2(4,3), S^1_3(4,3)$ $S^2_1(4,3), S^2_2(4,3), S^2_3(4,3)$	$R_1(4,4), R_2(4,4), R_3(4,4), R_4(4,4)$ $S^1_1(4,4), S^1_2(4,4), S^1_3(4,4)$ $S^2_1(4,4), S^2_2(4,4), S^2_3(4,4)$

$N = 4$ $M = 3$ MATRIX 4x4

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 ω_1 sin ω_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): -0.25
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
```

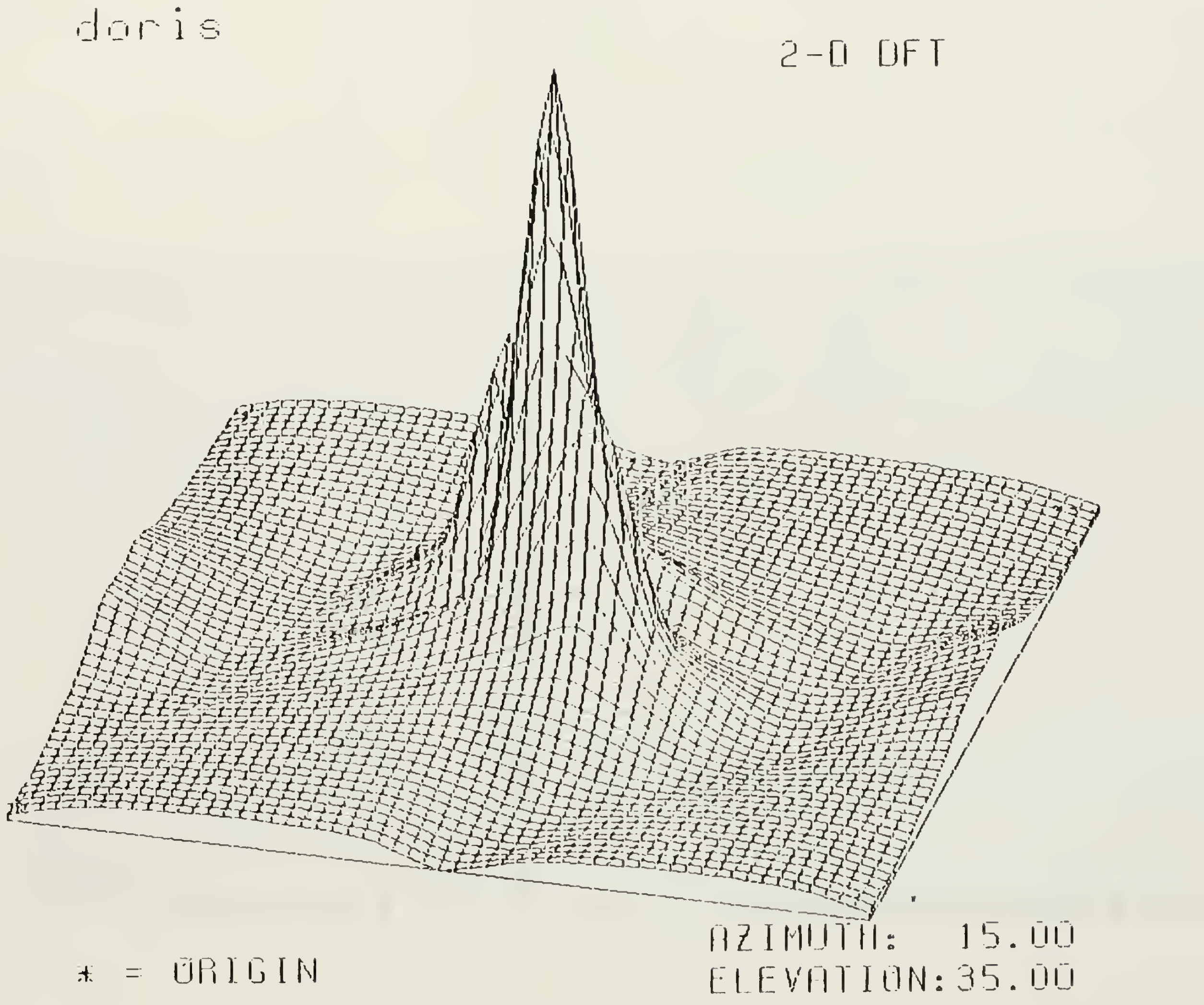


Figure 4-9a. 2-D D.F.T. Sequences, $|Y(m,n)|$ for Example 9

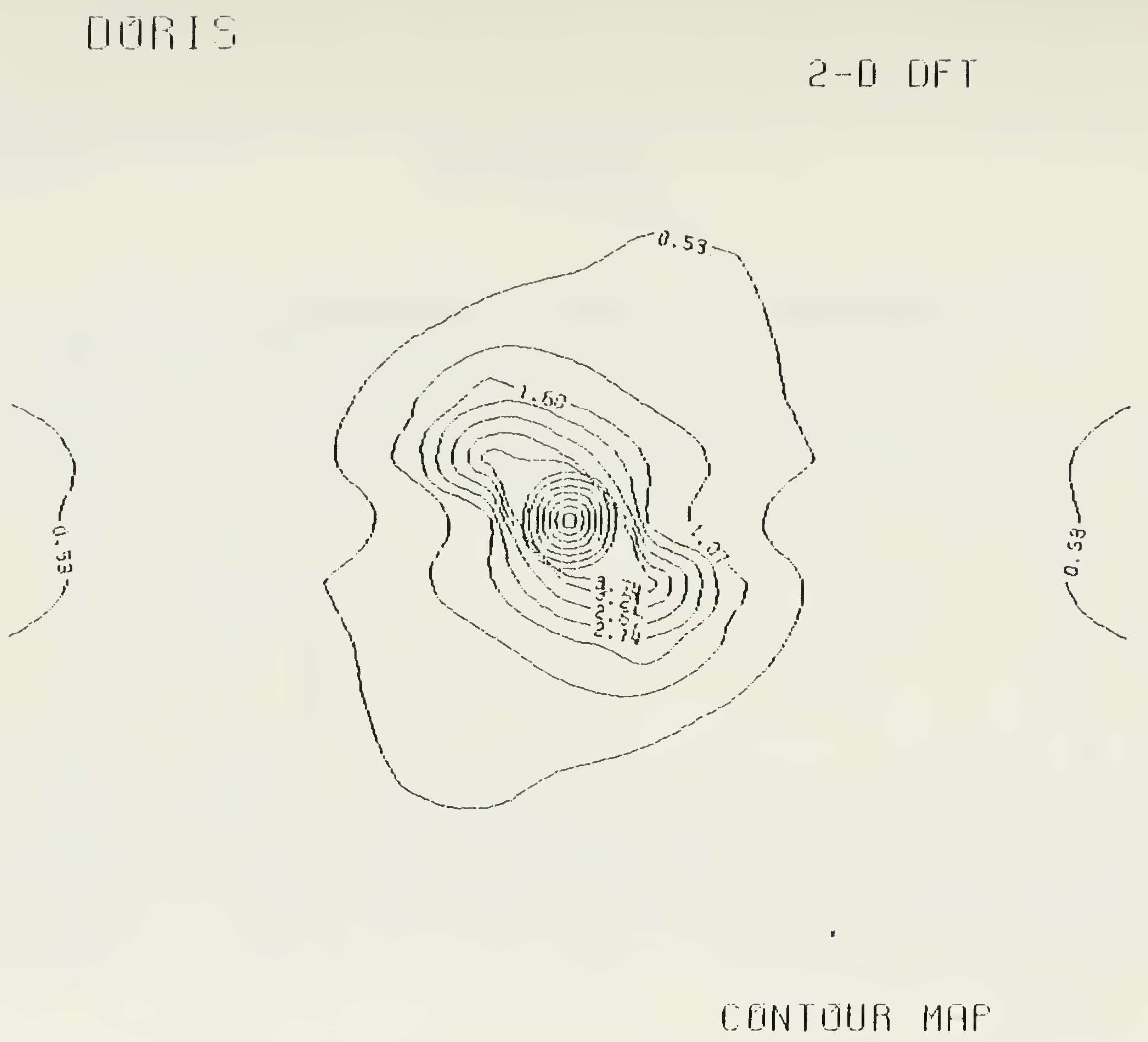


Figure 4-9b. Contour Map for Figure 4-9a

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): 0.125
  b( 10): 0
  b( 20): 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

 .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
```

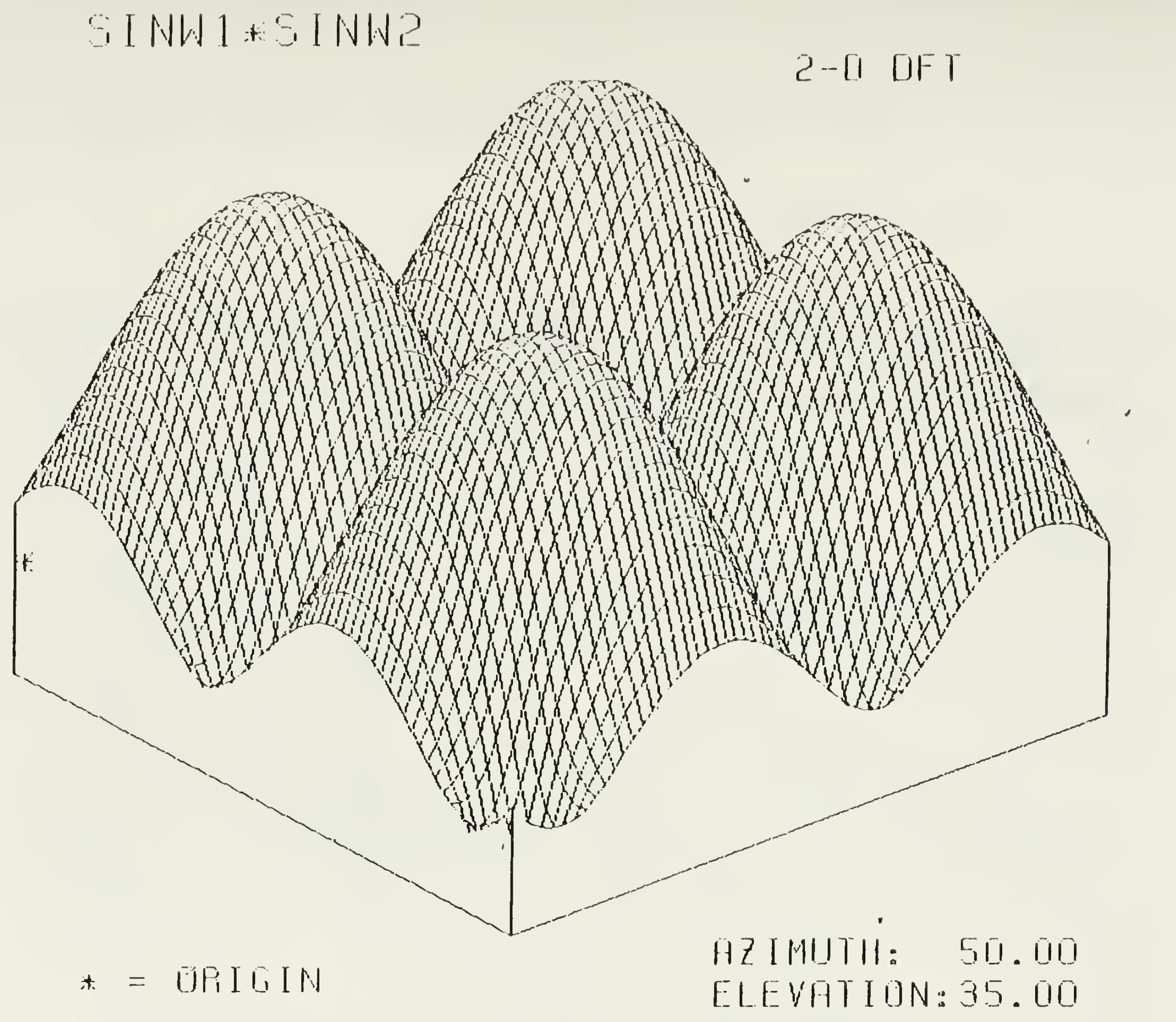


Figure 4-10a. 2-D D.F.T. Sequences, $|Y(m,n)|$, for Example 10

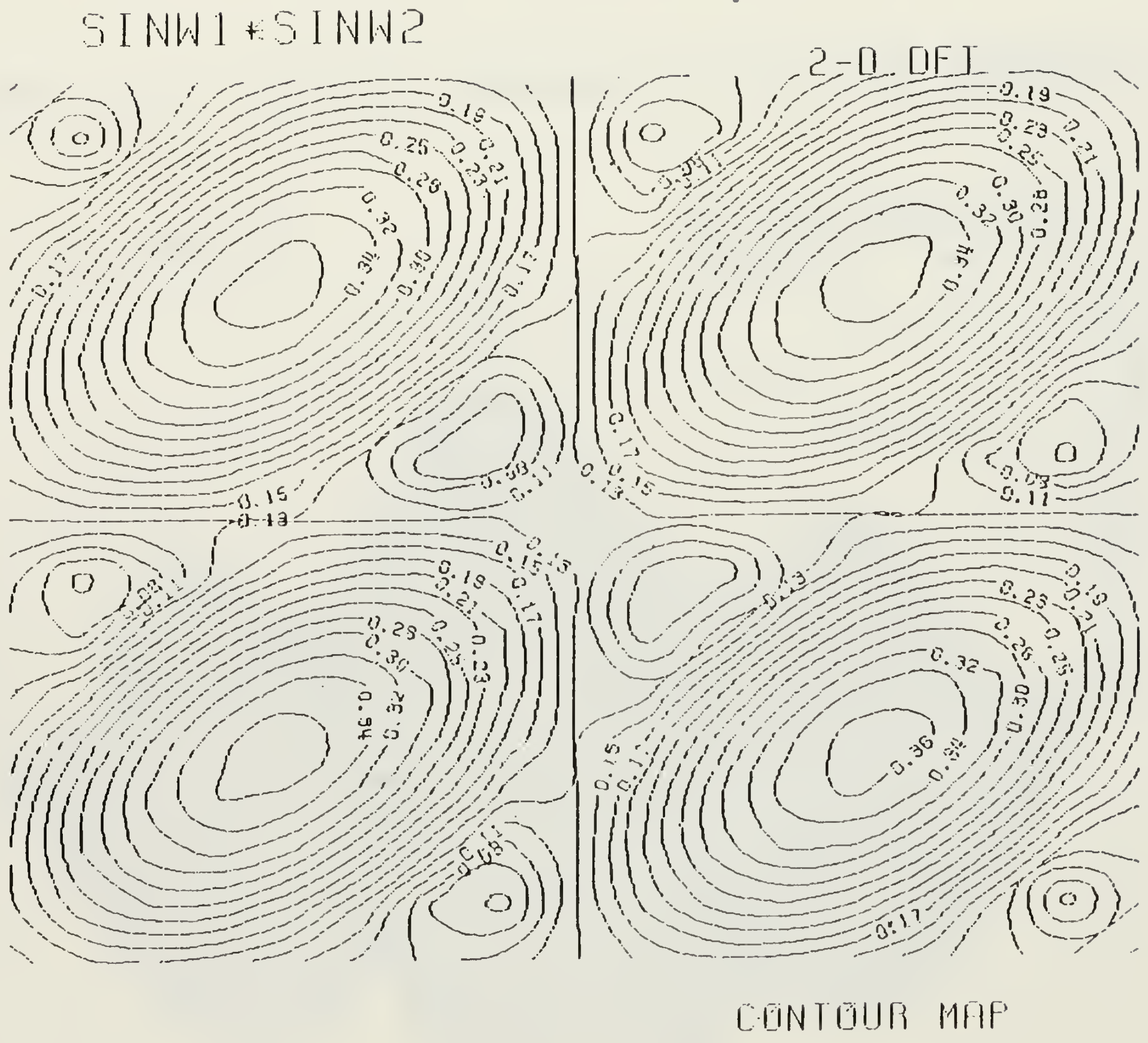


Figure 4-10b. Contour Map for Figure 4-10a

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(*.#)
    b(00): -0.125
   b( 10): 0
   b( 20): 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
```

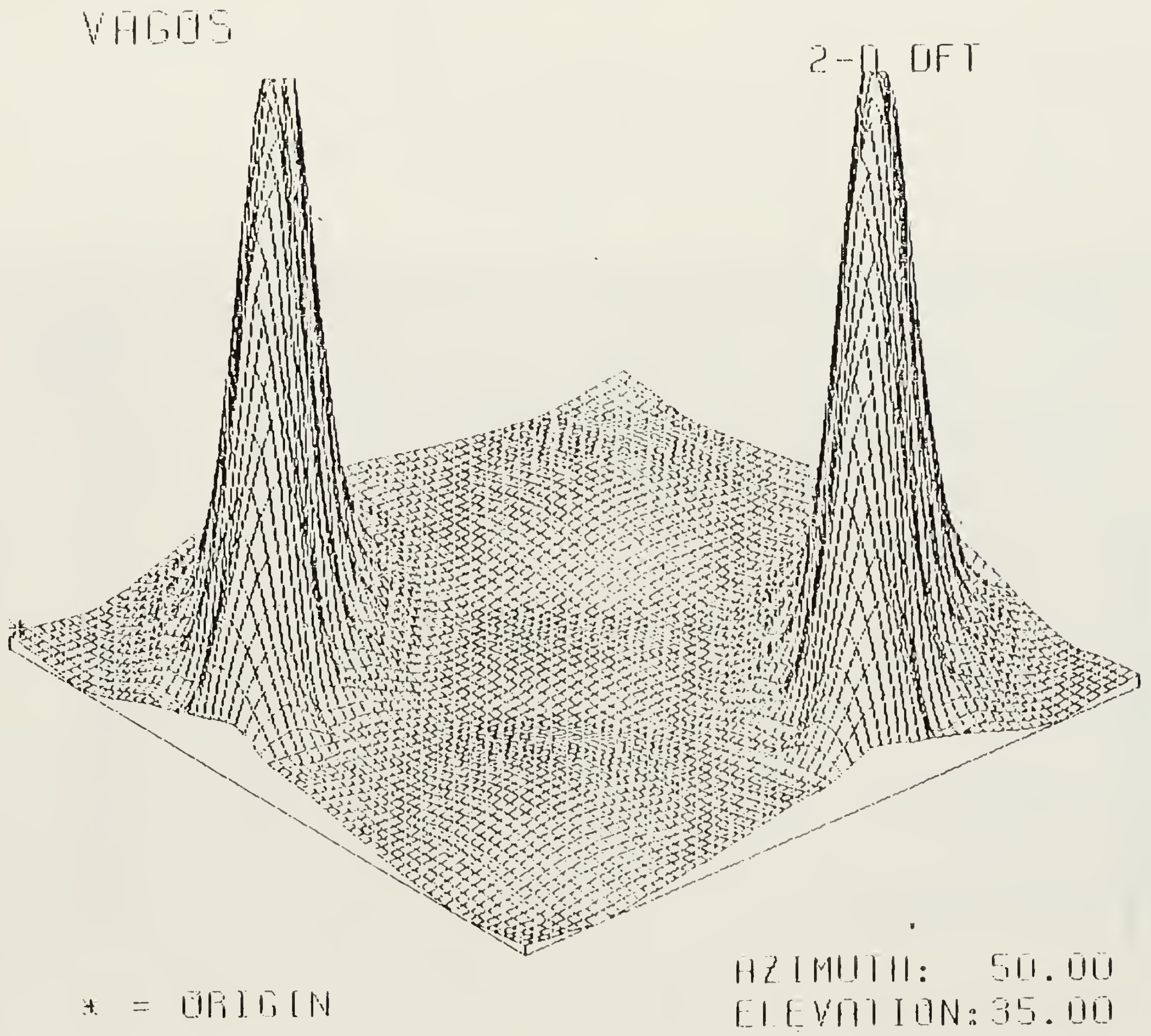


Figure 4-11a. 2-D D.F.T. Sequences, |Y(m,n)| for Example 11

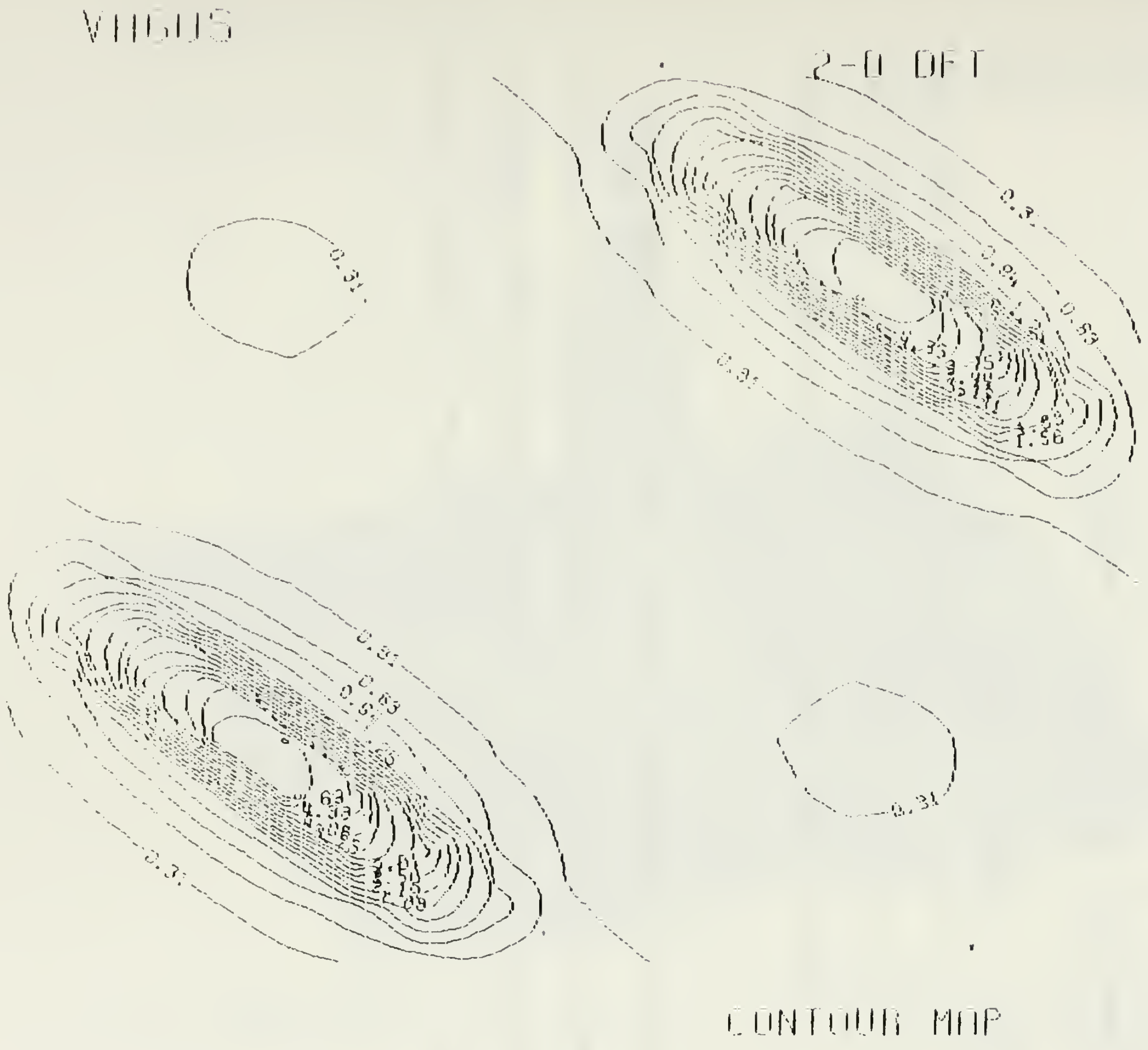


Figure 4-11b. Contour Map for Figure 4-11a

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.

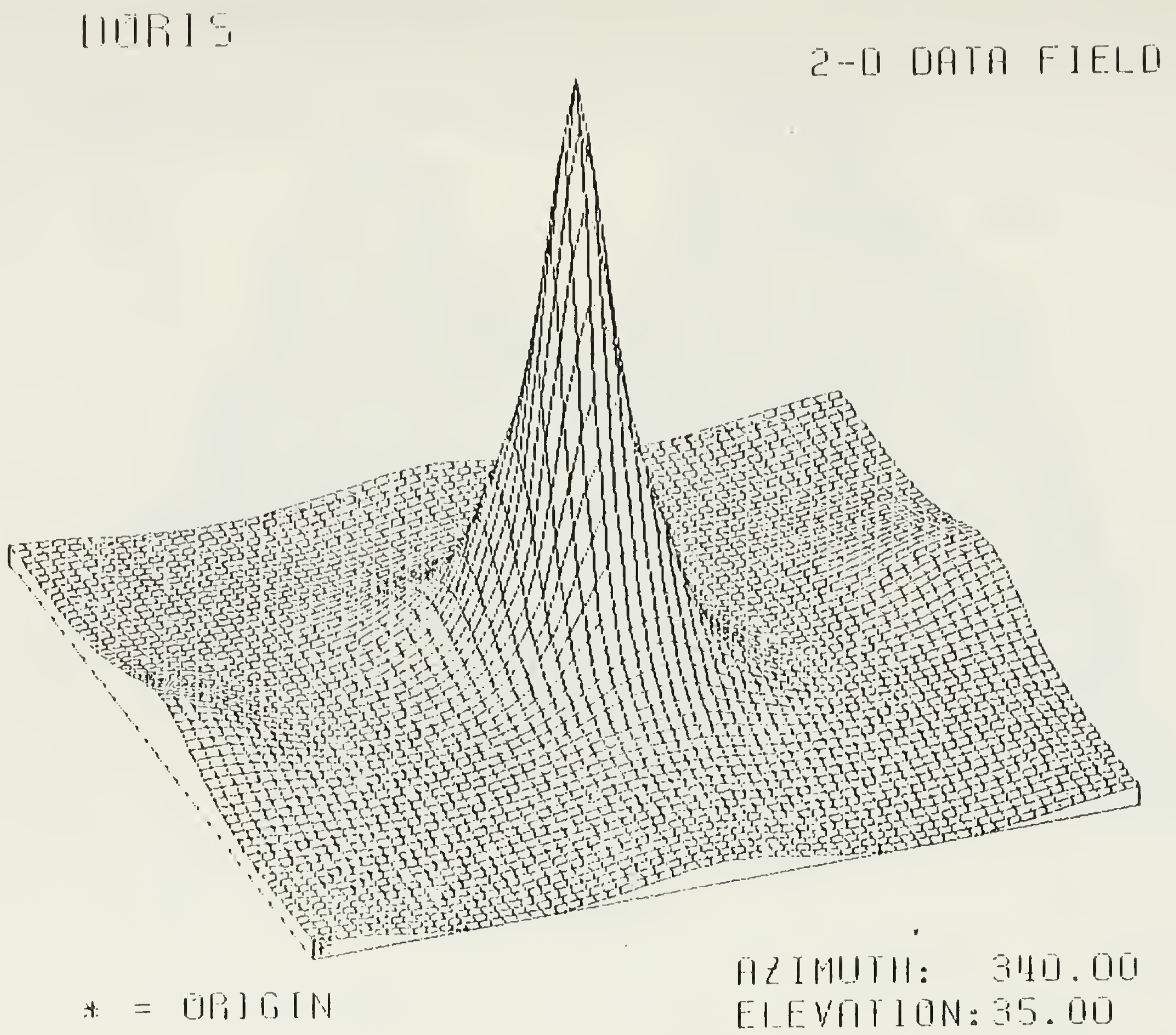


Figure 4-12a. Transfer Function $|H(z_1,z_2)|$, $z_1 = e^{j\omega_1}$, $z_2 = e^{j\omega_2}$ for Example 9

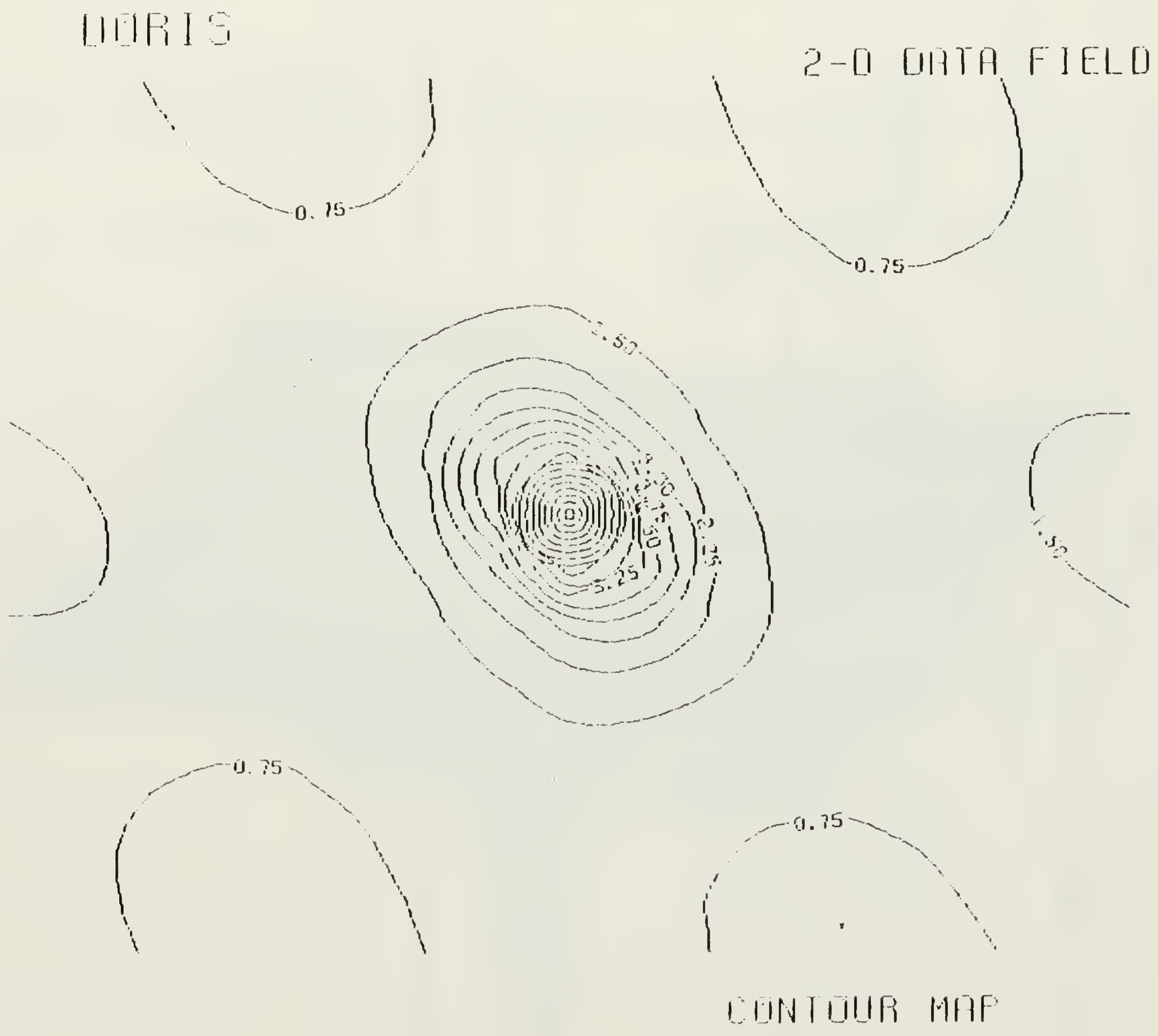


Figure 4-12b. Contour Map for Figure 4-12a

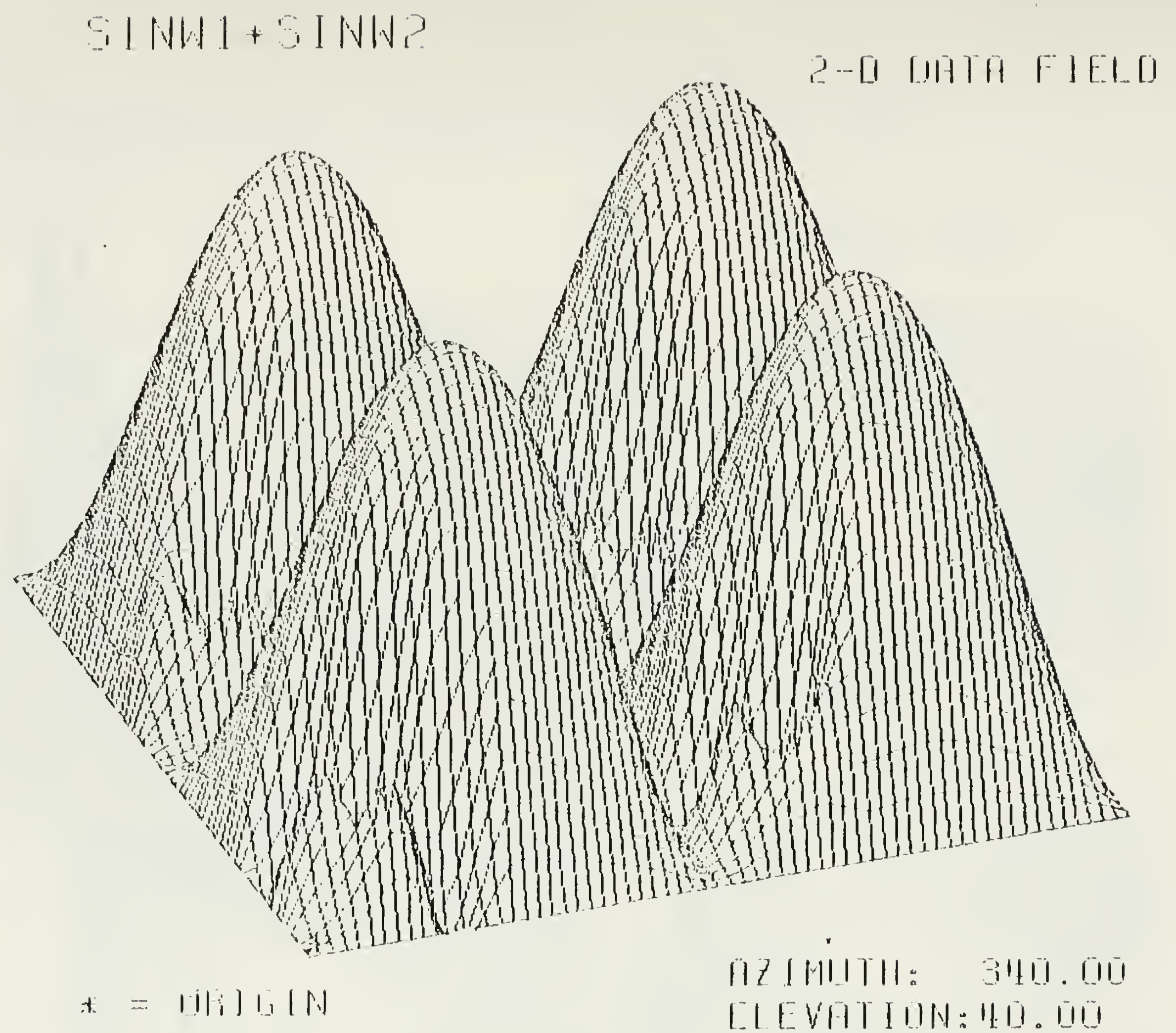


Figure 4-13a. Transfer Function $|H(z_1,z_2)|$, $z_1 = e^{j\omega_1}$, $z_2 = {}^{j\omega_2}$ for Example 10

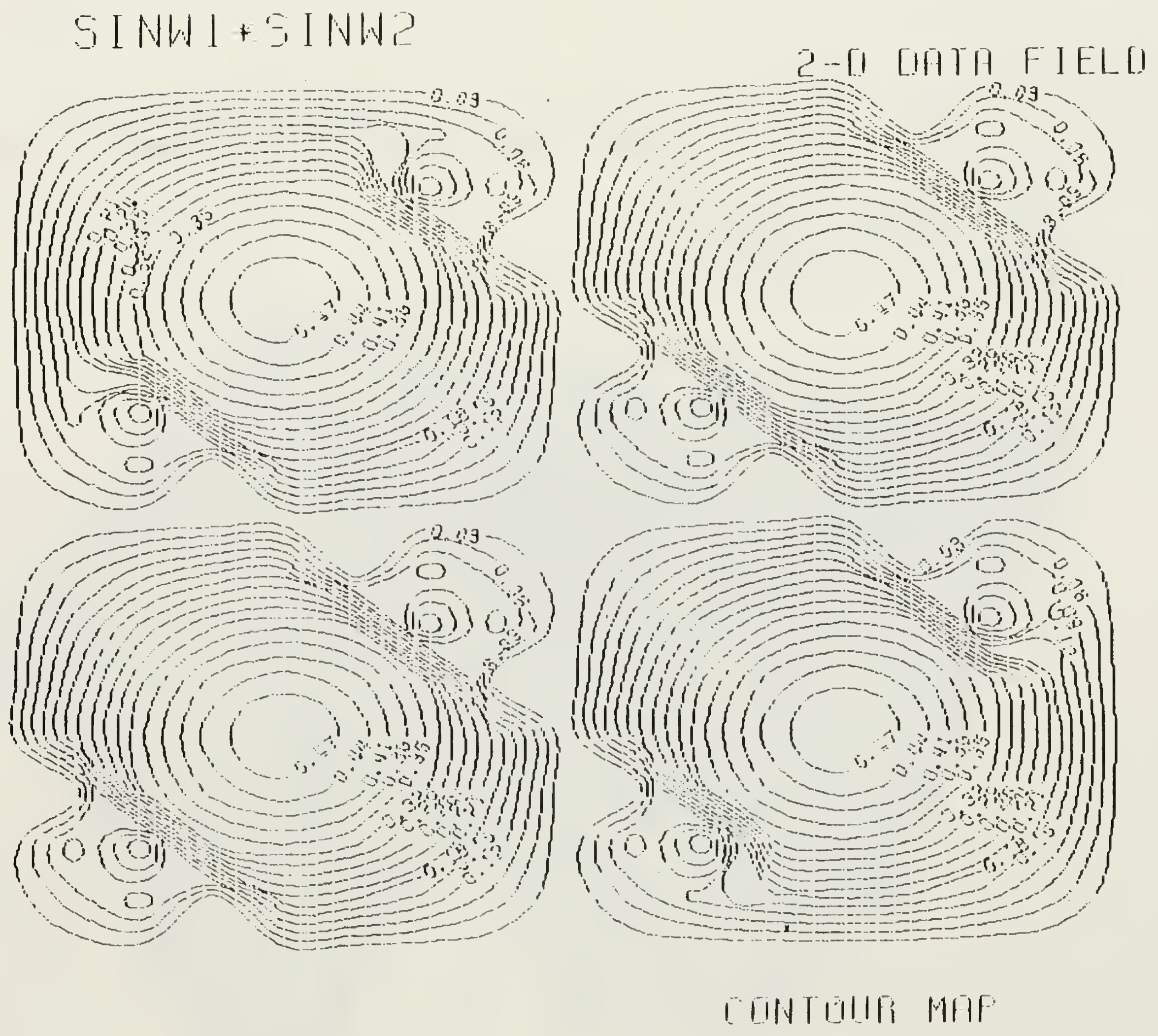


Figure 4-13b. Contour Map for Figure 4-13a

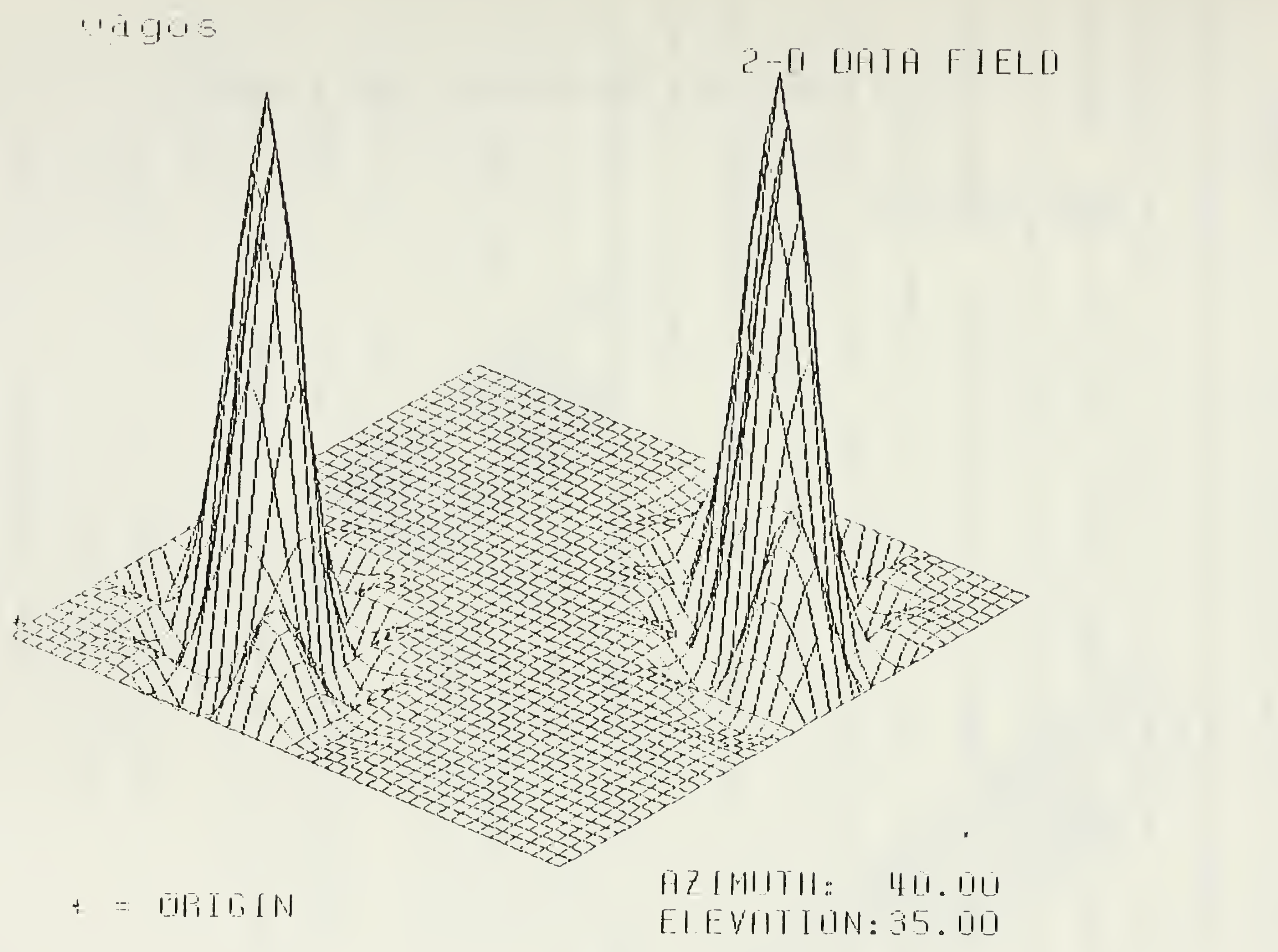


Figure 4-14a. Transfer Function $|H(z_1,z_2)|$, $Z_1 = e^{j\omega_1}$, $Z_2 = e^{j\omega_2}$ for Example #11

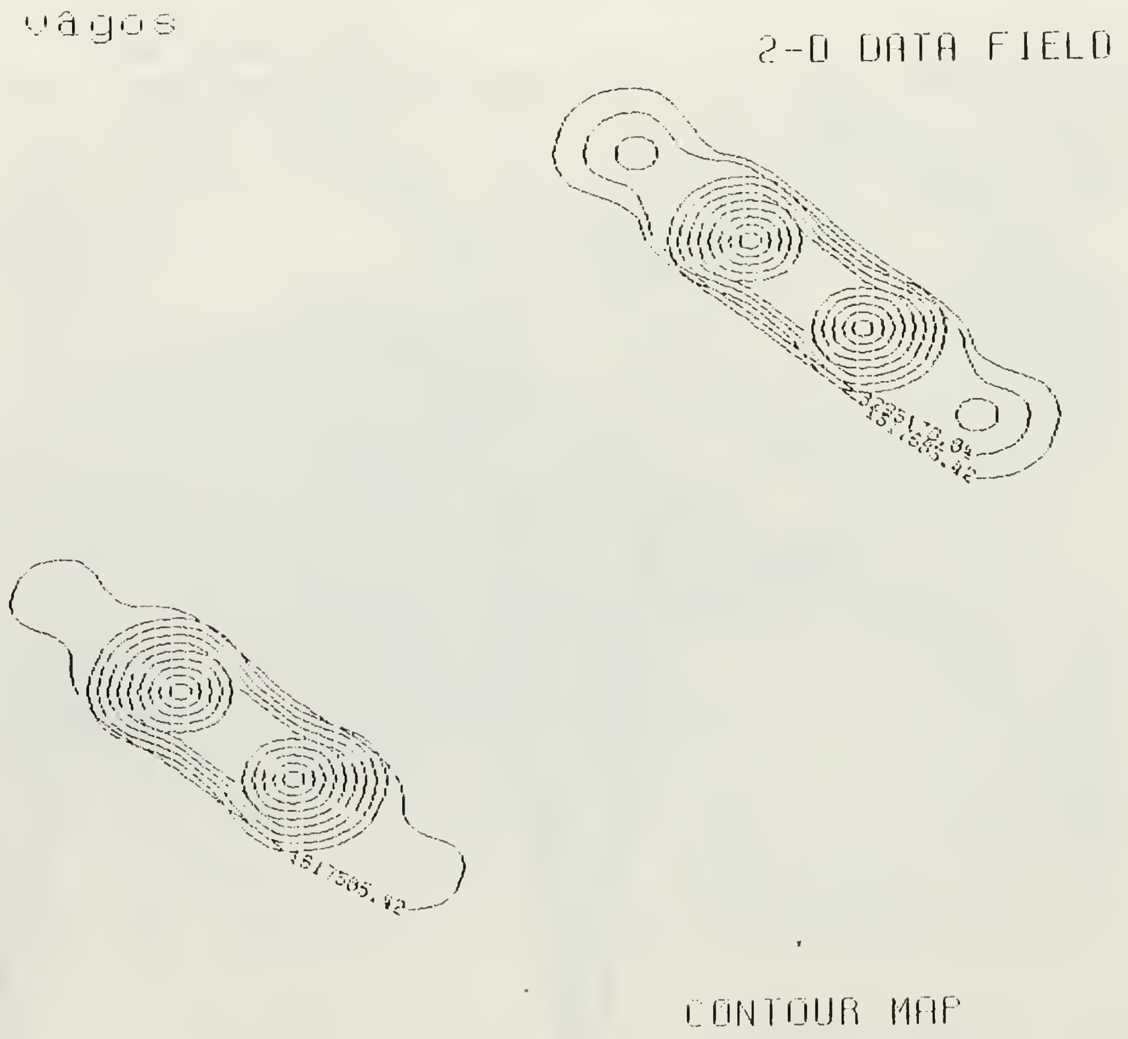


Figure 4-14b. Contour Map for Figure 4-14a

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.

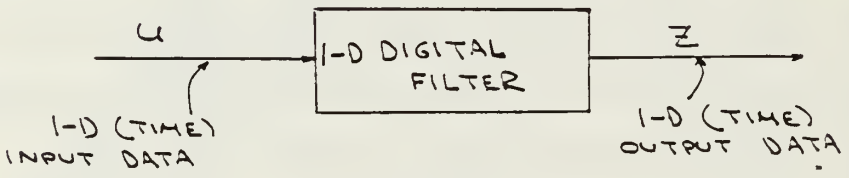


Figure 5-1

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

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

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,

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

yields

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

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).

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

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:

$$\begin{array}{c|ccccc|}
 & & & j & & \\
\hline
 & 0 & 1 & 2 & 3 & 4 \\
i & 5 & 6 & 7 & 8 & 9 \\
 & 10 & 11 & 12 & 13 & 14 \\
 & 15 & 16 & 17 & 18 & 19 \\
\hline
\end{array}$$

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

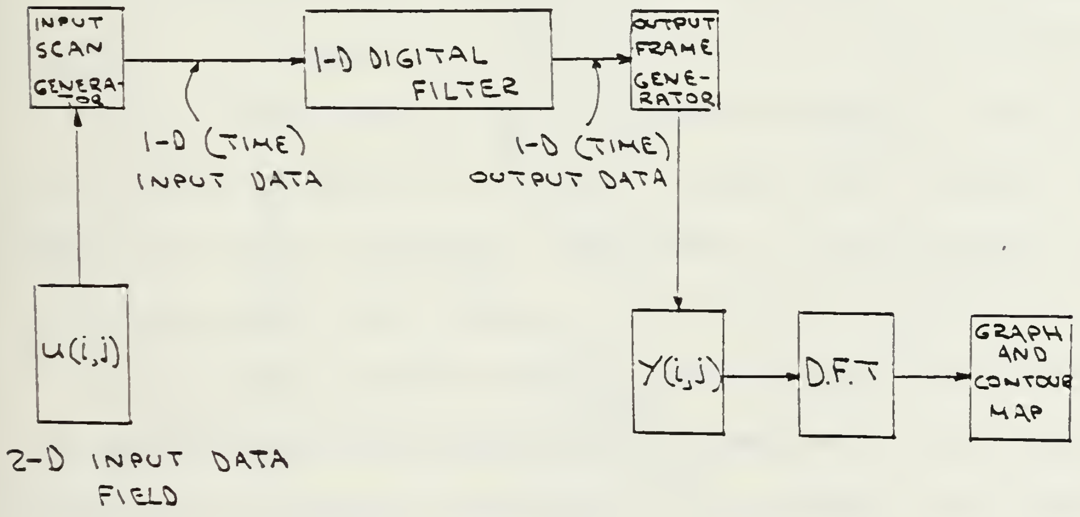
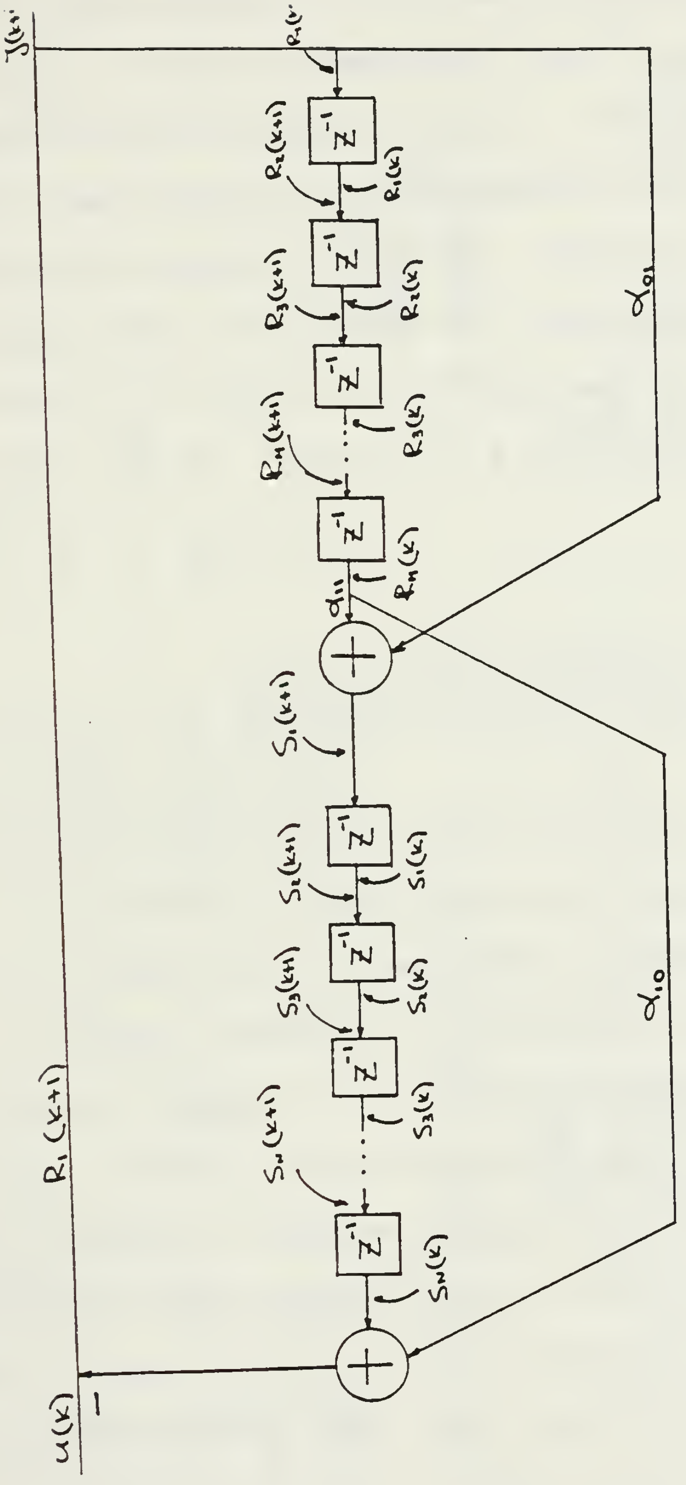


Figure 5-2

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



$J(k+1)$
$R_1(k+1)$
$R_2(k+1)$
$R_1(k)$
$R_3(k+1)$
$R_2(k)$
$R_4(k+1)$
$R_3(k)$
$R_N(k)$
α_{01}
α_{11}
$S_1(k+1)$
$S_2(k+1)$
$S_1(k)$
$S_3(k+1)$
$S_2(k)$
$S_N(k+1)$
$S_3(k)$
$S_N(k)$
α_{10}
$R_1(k+1)$
$u(k)$
$-$
z^{-1}

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,

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

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:

$M = 2;\ N = 2$

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

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

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

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

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

$$
\left[\begin{array}{c}
R_1(k+1) \\ R_2(k+1) \\ R_3(k+1) \\ \vdots \\ R_M(k+1) \\ \hline
S_1(k+1) \\ S_2(k+2) \\ S_3(k+3) \\ \vdots \\ S_N(k+1)
\end{array}\right]
=
\begin{array}{c}
\\ M \\ \\ \\ \\ \\ \\ N \\ \\ \\
\end{array}
\left[\begin{array}{ccccccccccc}
0 & 0 & 0 & \cdots & & -a_{1}0 & 0 & 0 & 0 & \cdots & -1 \\
1 & 0 & 0 & \cdots & & 0 & 0 & 0 & 0 & \cdots & 0 \\
0 & 1 & 0 & \cdots & & 0 & 0 & 0 & 0 & \cdots & 0 \\
\vdots & \vdots & \vdots & & & \vdots & \vdots & \vdots & \vdots & & \vdots \\
0 & 0 & 0 & \cdots & 1 & 0 & 0 & 0 & 0 & \cdots & 0 \\ \hline
0 & 0 & 0 & \cdots & & (a_{11}-a_{10}a_{01}) & 0 & 0 & 0 & \cdots & -a_{01} \\
0 & 0 & 0 & \cdots & & 0 & 1 & 0 & 0 & \cdots & 0 \\
0 & 0 & 0 & \cdots & & 0 & 0 & 1 & 0 & \cdots & 0 \\
\vdots & \vdots & \vdots & & & & \vdots & \vdots & \vdots & & \\
0 & 0 & 0 & \cdots & & 0 & 0 & 0 & 0 & \cdots\ 1, & 0
\end{array}\right]
\left[\begin{array}{c}
R_1(k) \\ R_2(k) \\ R_3(k) \\ \vdots \\ R_M(k) \\ \hline
S_1(k) \\ S_2(k) \\ S_3(k) \\ \vdots \\ S_N(k)
\end{array}\right]
\left[\begin{array}{c}
1 \\ 0 \\ 0 \\ \vdots \\ 0 \\ \hline
a_{01} \\ 0 \\ 0 \\ \vdots \\ 0
\end{array}\right]
U(k)
$$

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

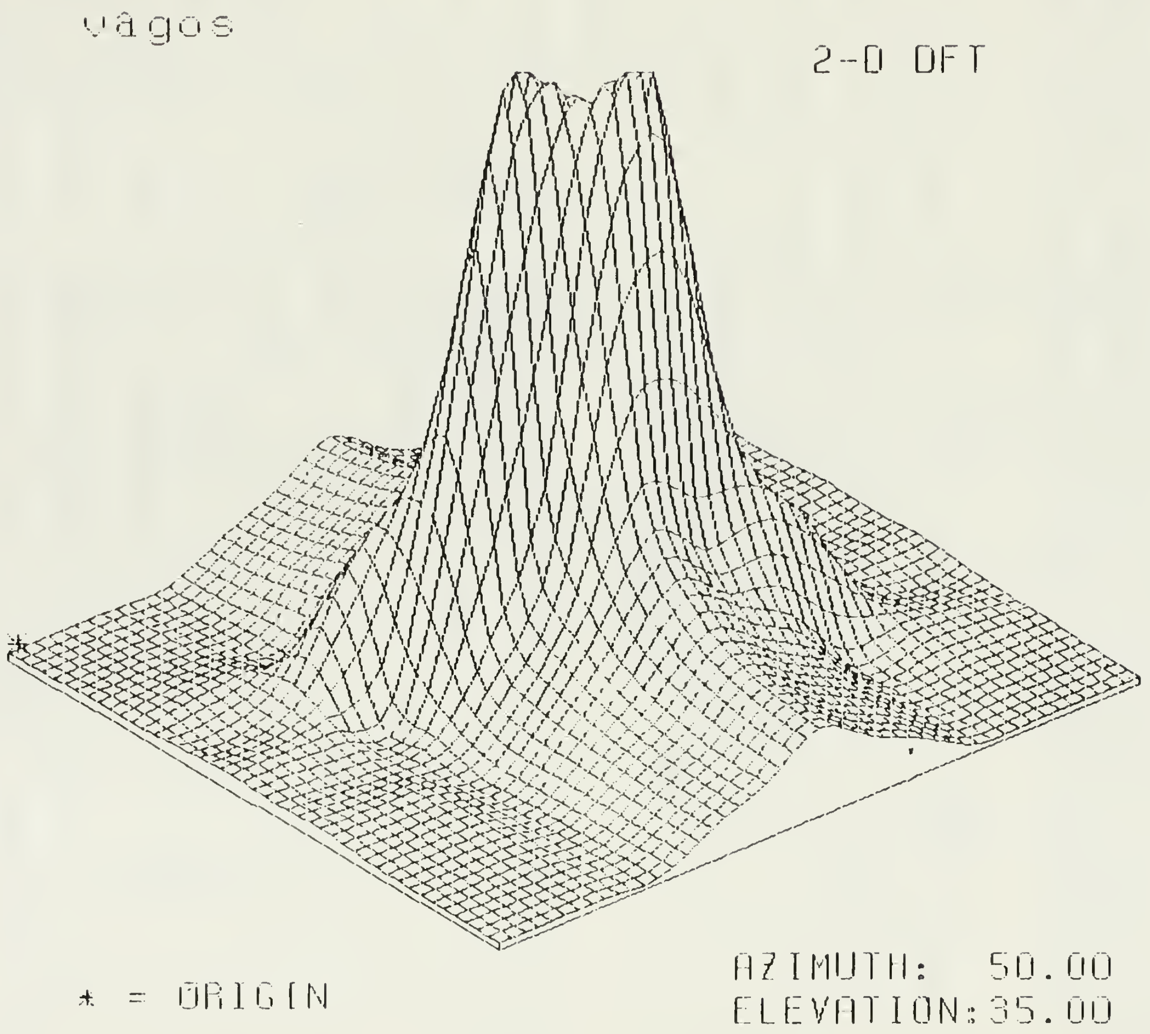


Figure 5-4a. 2-D D.F.T. Sequences, $|Y(m,n)|$ for Example 12

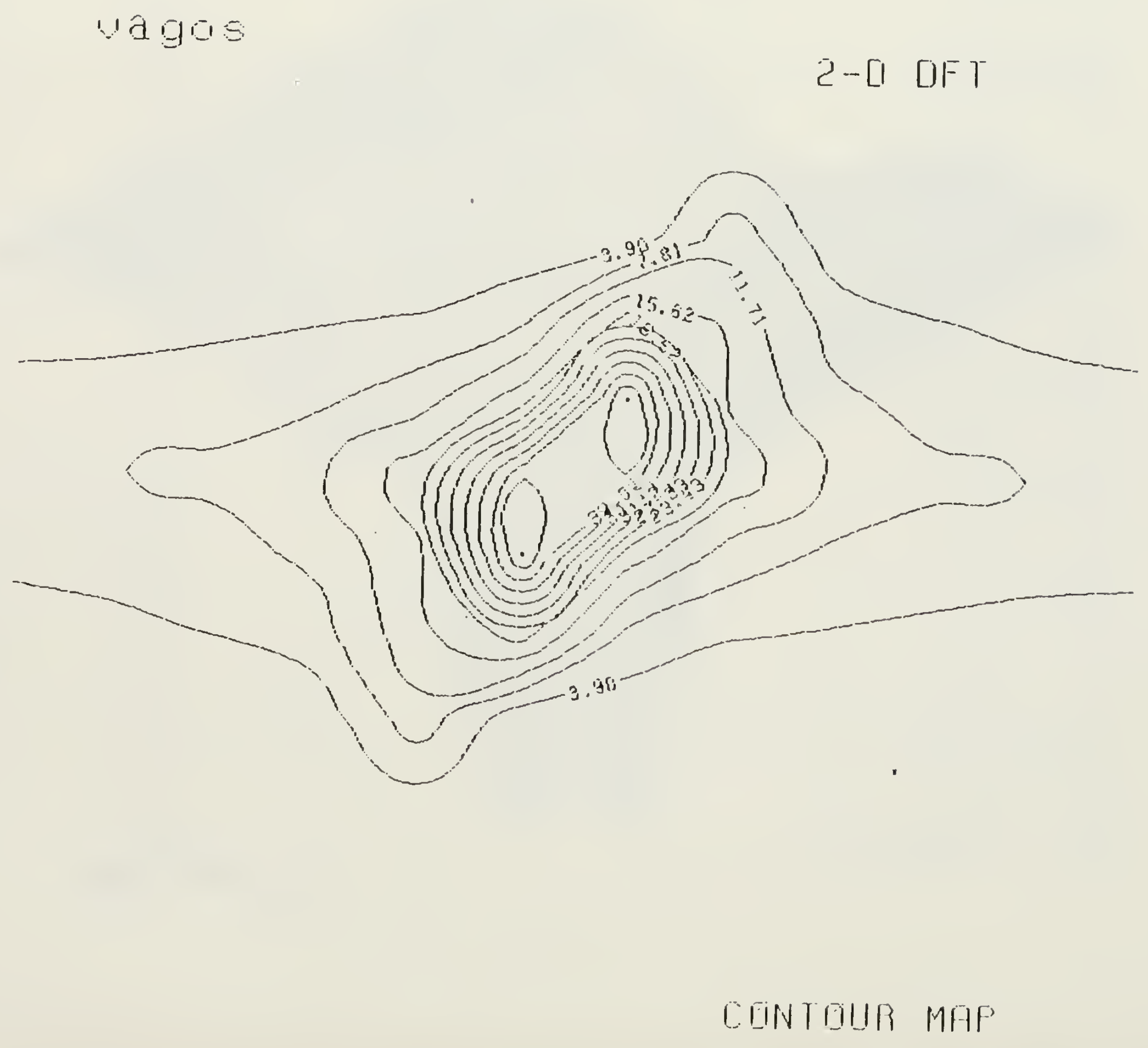


Figure 5-4b. Contour Map for Figure 5-4a

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 simulation 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

```
$STORAGE:  2
$LARGE

C     *******************************************************************
C     *                                                                 *
C     *  THE PURPOSE OF THIS PROGRAM IS  TO  COMPUTE  AND GRAPH THE     *
C     *  EQUATIONS OF ROBERT P.ROESSER IN THE "DISCRETE STATE-SPACE     *
C     *  MODEL FOR LINEAR IMAGE PROCESSING".                            *
C     *                                                                 *
C     *                         EVANGELOS THEOFILOU                     *
C     *******************************************************************
C     PROGRAM 2D-DATA-FIELD
.
C     ***** VARIABLE DECLARATIONS *****
      REAL         R(25,25),S(25,25),R1(2),R2(2),Z(31,31),
     *             RLPART,IMGPART,ZF(31,31),VERTEX(16),ZLEV(31)
      INTEGER      MASK(3000),LDIG(31),LWGT(31)
      CHARACTER*1  ANSWER
      CHARACTER*20 CTEXT

      DATA          XLOL/0.0/,YLOL/0.0/,XUPR/8.5/,YUPR/7.0/,
     *              ZLOW/1.0E35/,IPROJ/0/,NRNG/100/

C     *************  M A I N   P R O G R A M  *************

C     ***** ASK THE REQUIRED VALUES FOR THE MODEL *****
   10 WRITE (*,*) 'ENTER VALUES FOR THE FOLLOWING VARIABLES(#.#,..):'
      WRITE (*,399) 'A1: '
      READ (*,*) A1
      WRITE (*,399) 'A2: '
      READ (*,*) A2
      WRITE (*,399) 'A3: '
      READ (*,*) A3
      WRITE (*,399) 'A4: '
      READ (*,*) A4
      WRITE (*,399) 'B1: '
      READ (*,*) B1
      WRITE (*,399) 'B2: '
      READ (*,*) B2
      WRITE (*,399) 'C1: '
      READ (*,*) C1
      WRITE (*,399) 'C2: '
      READ (*,*) C2
    5 WRITE (*,402)
      READ  (*,*) N
      IF (N .GT. 25) GOTO 5

      WRITE (*,211) 'ENTER ',N,' INITIAL CONDITIONS FOR MATRIX R(#.#)'
      DO 99 I = 1,N
      WRITE (*,403) 'R(1,',I,'): '
      READ  (*,*) R(1,I)
   99 CONTINUE

      WRITE (*,211) 'ENTER ',N,' INITIAL CONDITIONS FOR MATRIX S(#.#)'
      DO 100 I = 1,N
      WRITE (*,404) 'S(',I,'1): '
      READ  (*,*) S(I,1)
  100 CONTINUE

```

```
      WRITE (*,419)
      READ  (*,200) ANSWER
      IF  ((ANSWER .EQ. 'Y') .OR. (ANSWER .EQ. 'y')) GOTO 10

      U = 1.0
C     ***** COMPUTE R AND S MATRICES *****
      DO 101 I = 1,N
         DO 101 J = 1,N
            IF  (I+1 .LE. N)  THEN
               R(I+1,J) = A1*R(I,J) + A2*S(I,J) + B1*U
            ENDIF
            IF  (J+1 .LE. N)  THEN
               S(I,J+1) = A3*R(I,J) + A4*S(I,J) + B2*U
            ENDIF
            U = 0.0
  101 CONTINUE

C     **** FILL 0's THE TWO DIMENTIONAL GRID OF CONTROL POINTS ****
      DO 102 I = 1,31
        DO 102 J = 1,31
          Z(I,J) =0.0
  102 CONTINUE

C     ***** COMPUTE Z MATRIX *****
      DO 103 I = 1,N
        DO 103 J = 1,N
          Z(I,J) = C1*R(J,I) + C2*S(J,I)
  103 CONTINUE

C     ***** OUTPUT THE Z MATRIX *****
      WRITE (*,205) '*****  Z  M A T R I X  ',N,' X ',N,'  *****'
      WRITE (*,212)
      DO 104 I = 1,N
        WRITE (*,300) (Z(I,J), J = 1,N)
        WRITE (*,210)
  104 CONTINUE
      WRITE (*,213)

      WRITE (*,418)
      READ  (*,200) ANSWER
      IF  ((ANSWER .NE. 'Y') .AND. (ANSWER .NE. 'y')) GOTO 18

C     ***** ASK THE PARAMETERS FOR THE GRAPH *****
   15 WRITE (*,210)
      WRITE (*,*) '***  E N T E R  P L O T  P A R A M E T E R S  ***'
      WRITE (*,405)
      READ  (*,*)  AZIM
      WRITE (*,406)
      READ  (*,*)  ELEV
      WRITE (*,408)
      READ  (*,*)  ITRIM
      WRITE (*,409)
      READ  (*,*)  IDIV
      WRITE (*,411)
      READ  (*,199)  CTEXT
      WRITE (*,401)
      READ  (*,200)  ANSWER

C     ***** INITIALIZE PLOT88 *****
```

```
      IF  ((ANSWER .EQ. 'Y') .OR. (ANSWER .EQ. 'y'))  THEN
        CALL PLOTS(0,0,2)
      ELSE
        CALL PLOTS(0,99,99)
      ENDIF

      CALL WINDOW(XLOL,YLOL,XUPR,YUPR)

C     ***** DRAW THE MESH SURFACE OF THE GRAPH *****
      CALL MESHS(Z,31,31,N,N,AZIM,ELEV,0.5,0.5,7.5,5.5,IDIV,0,
     *           3,IPROJ,1,ZLOW,3,ITRIM,MASK,VERTEX)
C     ***** ANNOTATION OF THE GRAPH *****
      CALL SYMBOL(5.5,0.3,0.2,'AZIMUTH:   ',0.0,10)
      CALL NUMBER(999.0,999.0,0.2,AZIM,0.0,2)
      CALL SYMBOL(5.5,0.0,0.2,'ELEVATION:',0.0,10)
      CALL NUMBER(999.0,999.0,0.2,ELEV,0.0,2)
      DY = (Z(1,1)/90.0) * ELEV
      CALL P3D2D(1.0,1.0,Z(1,1)-DY,XR,YR)
      CALL SYMBOL(XR,YR,0.25,'*',0.0,1)
      CALL SYMBOL(1.0,0.1,0.2,'* = ORIGIN',0.0,10)
      CALL SYMBOL(1.0,6.75,0.25,CTEXT,0.0,20)
      CALL SYMBOL(6.0,6.5,0.2,'2-D DATA FIELD',0.0,14)

C     ***** OUTPUT THE GRAPH *****
      CALL PLOT(0.0,0.0,999)
      WRITE (*,412)
      READ  (*,200) ANSWER
      IF  ((ANSWER .EQ. 'Y') .OR. (ANSWER .EQ. 'y')) GOTO 15

   18 WRITE(*,417)
      READ(*,200)  ANSWER
      IF  ((ANSWER .EQ. 'Y') .OR. (ANSWER .EQ. 'y')) THEN
C       **** FILL 0's THE TWO DIMENTIONAL GRID OF CONTROL POINTS ****
        DO 106 I = 1,31
          DO 106 J = 1,31
            ZF(I,J) =0.0
  106 CONTINUE
        ZFMAX = -9.9E20
        ZFMIN = 9.9E20
        DN = (N-1)/2.0
        P = 6.283185
        DO 107 I = 1,N
          DO 107 J = 1,N
            RLPART  = 0.0
            IMGPART = 0.0
            DO 108 L = 1,N
              DO 108 K = 1,N
                R1(1) = COS(-P*(L-1)*(I-DN-1)/N)
                R1(2) = SIN(-P*(L-1)*(I-DN-1)/N)
                R2(1) = COS(-P*(K-1)*(J-DN-1)/N)
                R2(2) = SIN(-P*(K-1)*(J-DN-1)/N)
                RLPART  = RLPART  + Z(L,K)*(R1(1)*R2(1)
     *                                     -R1(2)*R2(2))
                IMGPART = IMGPART + Z(L,K)*(R1(1)*R2(2)
     *                                     +R1(2)*R2(1))
  108       CONTINUE
            ZF(I,J) = SQRT(RLPART**2 + IMGPART**2)
            IF (ZF(I,J) .GT. ZFMAX) THEN
              ZFMAX = ZF(I,J)
```

```
D Line# 1      7                                         Microsoft FORTRAN77 V3.20 02/84
2                  ENDIF
2                  IF (ZF(I,J) .LT. ZFMIN) THEN
2                    ZFMIN = ZF(I,J)
2                  ENDIF
2         107    CONTINUE

        C         ***** OUTPUT THE ZF MATRIX *****
               WRITE (*,205) '***  FOURIER TRANSFORMATION  ',N,' X ',N,'   ***'
               WRITE (*,212)
               DO 109 I = 1,N
1                 WRITE (*,300) (ZF(I,J), J = 1,N)
1                 WRITE (*,210)
1         109  CONTINUE
               WRITE (*,213)

               WRITE (*,418)
               READ  (*,200) ANSWER
               IF  ((ANSWER .NE. 'Y') .AND. (ANSWER .NE. 'y')) GOTO 16

        C         ***** ASK THE PARAMETERS FOR THE GRAPH *****
          30      WRITE (*,210)
                  WRITE (*,*) '***  E N T E R  P L O T  P A R A M E T E R S  ***'
                  WRITE (*,405)
                  READ  (*,*)  AZIM
                  WRITE (*,406)
                  READ  (*,*)  ELEV
                  WRITE (*,408)
                  READ  (*,*)  ITRIM
                  WRITE (*,409)
                  READ  (*,*)  IDIV
                  WRITE (*,411)
                  READ  (*,199)  CTEXT
                  WRITE (*,401)
                  READ  (*,200)  ANSWER

        C         ***** INITIALIZE PLOT88 *****
                  IF  ((ANSWER .EQ. 'Y') .OR. (ANSWER .EQ. 'y'))  THEN
                    CALL PLOTS(0,0,2)
                  ELSE
                    CALL PLOTS(0,99,99)
                  ENDIF

                  WRITE (*,420)
                  READ  (*,200)  ANSWER

                  CALL WINDOW(XLOL,YLOL,XUPR,YUPR)

                  IF  ((ANSWER .EQ. 'Y') .OR. (ANSWER .EQ. 'y'))  THEN
                    DLEV = (ZFMAX-ZFMIN)/FLOAT(N)
                    CALL ZLEVEL(ZF,31,31,N,N,DLEV,ZLEV,N+1)
                    DO 110 I = 1,N+1
1                     LDIG(I) = 2
1                     LWGT(I) = 1
1         110       CONTINUE

        C           ***** DRAW THE CONTOUR MAP *****
                    CALL ZCNTUR(ZF,31,31,N,N,0.5,0.5,7.5,5.5,ZLEV,LDIG,LWGT,
             *                  N+1,0.10,10)
                    CALL SYMBOL(5.5,0.0,0.2,'CONTOUR MAP',0.0,11)
        ELSE
C         ***** DRAW THE MESH SURFACE OF THE GRAPH *****
          CALL MESHS(ZF,31,31,N,N,AZIM,ELEV,0.5,0.5,7.5,5.5,IDIV,0,
     *               3,IPROJ,1,ZLOW,3,ITRIM,MASK,VERTEX)
C         ***** ANNOTATION OF THE GRAPH *****
          CALL SYMBOL(5.5,0.3,0.2,'AZIMUTH:   ',0.0,10)
          CALL NUMBER(999.0,999.0,0.2,AZIM,0.0,2)
          CALL SYMBOL(5.5,0.0,0.2,'ELEVATION:',0.0,10)
          CALL NUMBER(999.0,999.0,0.2,ELEV,0.0,2)
          DY = (ZF(1,1)/90.0) * ELEV
          CALL P3D2D(1.0,1.0,ZF(1,1)-DY,XR,YR)
          CALL SYMBOL(XR,YR,0.25,'*',0.0,1)
          CALL SYMBOL(1.0,0.1,0.2,'* = ORIGIN',0.0,10)
        ENDIF

        CALL SYMBOL(1.0,6.75,0.25,CTEXT,0.0,20)
        CALL SYMBOL(6.0,6.5,0.2,'2-D DFT',0.0,7)

C       ***** OUTPUT THE GRAPH *****
        CALL PLOT(0.0,0.0,999)
        WRITE (*,412)
        READ  (*,200) ANSWER
        IF  ((ANSWER .EQ. 'Y') .OR. (ANSWER .EQ. 'y')) GOTO 30
  16  ENDIF
      WRITE (*,413)
      READ  (*,200) ANSWER
      IF  ((ANSWER .EQ. 'Y') .OR. (ANSWER .EQ. 'y')) GOTO 10
      STOP

  199 FORMAT(A20)
  200 FORMAT(A)
  205 FORMAT(/,20X,A25,I2,A3,I2,A8,/)
  210 FORMAT(/)
  211 FORMAT(/,A8,I2,A47)
  212 FORMAT(/,2X,'(AZIMUTH 320.0)',46X,'(AZIMUTH 230.0)',/)
  213 FORMAT(/,2X,'(AZIMUTH 050.0)',46X,'(AZIMUTH 140.0)',/)
  300 FORMAT(10(F7.2,1X))
  399 FORMAT(/,5X,A4,\)
  400 FORMAT(9X,\)
  401 FORMAT(/,5X,'SEND GRAPH TO THE PRINTER(Y or N): ',\)
  402 FORMAT(/,5X,'NUMBER OF ROWS/COLUMNS FOR R AND S(1 to 25): ',\)
  403 FORMAT(5X,A4,I2,A3,\)
  404 FORMAT(5X,A2,I2,A5,\)
  405 FORMAT(/,5X,'AZIMUTH(0.0 to 360.0 DEGREES): ',\)
  406 FORMAT(/,5X,'ELEVATION(90.0 to -90.0 DEGREES): ',\)
  408 FORMAT(/,5X,'TRIM(0=NO, 1=Xs, 2=Ys): ',\)
  409 FORMAT(/,5X,'2,4 OR 8 SUBGRIDS: ',\)
  411 FORMAT(/,5X,'TITLE OF GRAPH(UP TO 20 CHAR): ',\)
  412 FORMAT(/,5X,'DO YOU WANT TO CHANGE PARAMETERS? ',\)
  413 FORMAT(/,5X,'DO YOU WANT TO REPEAT THE PROCESS? ',\)
  417 FORMAT(/,5X,'DO YOU WANT FOURIER TRANSFORMATION ? ',\)
  418 FORMAT(/,5X,'DO YOU WANT TO MAKE GRAPH ? ',\)
  419 FORMAT(/,5X,'DO YOU WANT TO CORRECT ? ',\)
  420 FORMAT(/,5X,'DO YOU WANT CONTOUR MAP ? ',\)
      END
```

```
Name     Type         Offset P Class

A1      REAL              26
```

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

```
                                                                         Page    7
                                                                         09-25-85
                                                                         12:34:27
D Line# 1      7                                Microsoft FORTRAN77 V3.20 02/84
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

```
                                                      Page    1
                                                      09-26-85
                                                      21:09:36
D Line# 1      7                   Microsoft FORTRAN77 V3.20 02/84
$STORAGE:  2
$PAGESIZE:58
C     ***************************************************************
C     *                                                             *
C     *  THE PURPOSE OF THIS PROGRAM IS  TO  COMPUTE  AND GRAPH THE *
C     *  FREQUENCY RESPONSE OF A 2-D DIGITAL FILTER.                *
C     *                                                             *
C     *                        EVANGELOS THEOFILOU                  *
C     ***************************************************************
C     PROGRAM 2D-DATA-FIELD

C     ***** VARIABLE DECLARATIONS *****
      REAL          A(7,7),B(7,7),R1(7,7,2),R2(7,7,2),
     *              RLPART, IMGPART, Z(51,51),
     *              VERTEX(16),ZLEV(51)
      INTEGER       MASK(3000),LDIG(51),LWGT(51)
      CHARACTER*1 ANSWER
      CHARACTER*20 CTEXT

      DATA           XLOL/0.0/,YLOL/0.0/,XUPR/8.5/,YUPR/7.0/,
     *               ZLOW/1.0E35/,IPROJ/0/,NRNG/100/

C     *************  M A I N   P R O G R A M  *************

   10 WRITE (*,401)
      READ  (*,*)    IT
      IF (IT .GT. 25) GOTO 10
      WRITE (*,402)
      READ  (*,*) K
      K = K + 1
      WRITE(*,*) '       ENTER VALUES OF COEFFICIENTS:'
      DO 100 I  = 0,K-1
        DO 100 J  = 0,K-1
          WRITE(*,404) 'B(',I,',',J,'): '
          READ  (*,*) B(I+1,J+1)
  100 CONTINUE

      DO 101 I  = 0,K-1
        DO 101 J  = 0,K-1
          WRITE(*,404) 'A(',I,',',J,'): '
          READ  (*,*) A(I+1,J+1)
  101 CONTINUE

      WRITE (*,419)
      READ  (*,200) ANSWER
      IF  ((ANSWER .EQ. 'Y') .OR. (ANSWER .EQ. 'y')) GOTO 10

C     **** FILL 0's THE TWO DIMENTIONAL GRID OF CONTROL POINTS ****
      DO 107 I = 1,51
        DO 107 J = 1,51
          Z(I,J)  = 0.0
```

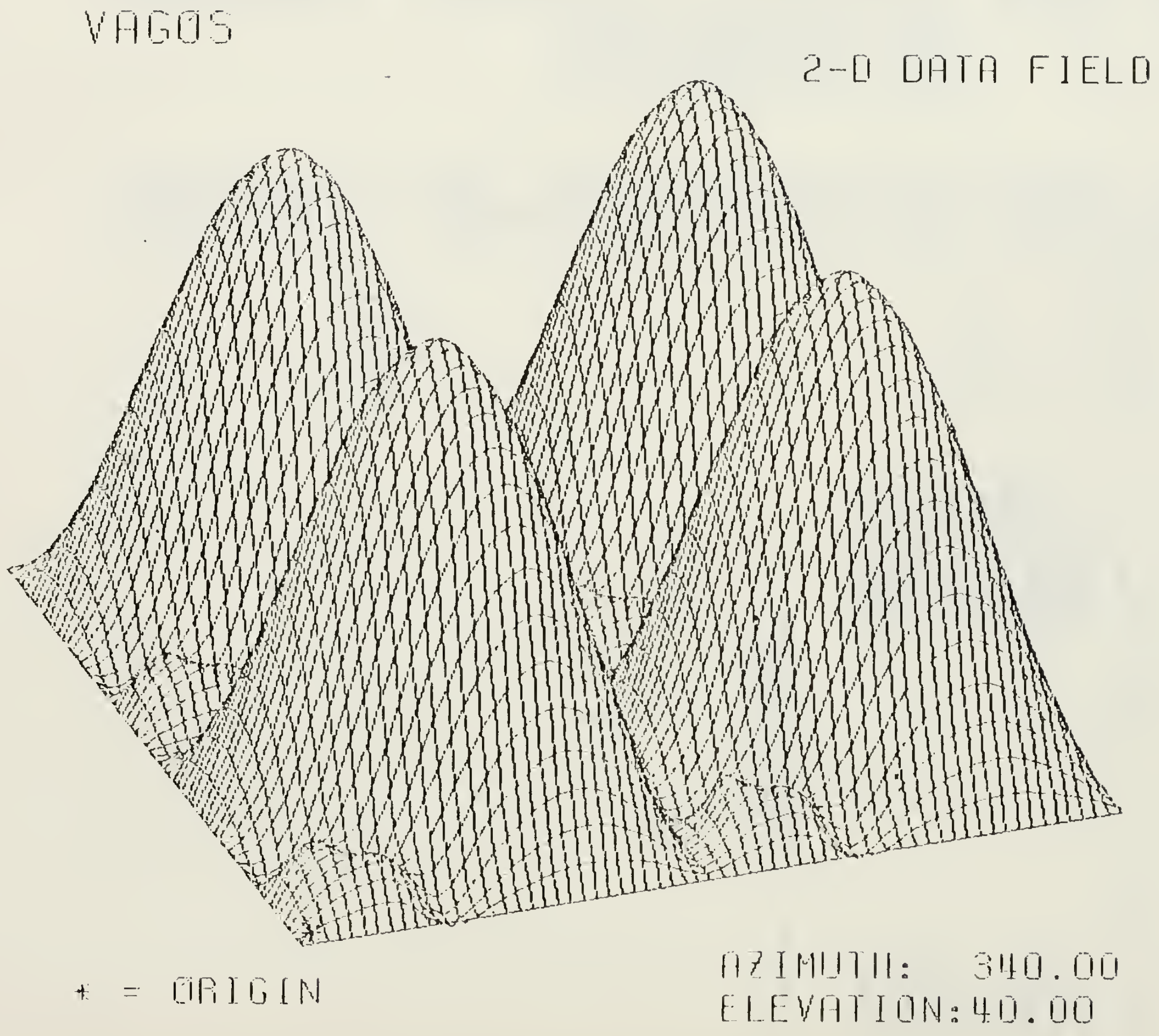
```
D Line# 1      7                                      Microsoft FORTRAN77 V3.20 02/84
2        107 CONTINUE

             ZMIN = 9.9E20
             ZMAX = -9.9E20
             P  = 3.14159
             STEP = 2*P / (IT-1)
             W1 = -P - STEP
             L  = 0
             DO 102 I = 1,IT
1              W1 = W1 + STEP
1              W2 = -P - STEP
1              DO 103 J = 1,IT
2                L = L + 1
2                W2 = W2 + STEP
2                DO 104 M = 0,K-1
3                  DO 104 N = 0,K-1
4                    R1(M+1,N+1,1) = COS(-M * W1)
4                    R1(M+1,N+1,2) = SIN(-M * W1)
4                    R2(M+1,N+1,1) = COS(-N * W2)
4                    R2(M+1,N+1,2) = SIN(-N * W2)
4        104     CONTINUE
2                RLNOM  = 0.0
2                IMGNOM = 0.0
2                RLDEN  = 0.0
2                IMGDEN = 0.0
2                DO 105 M = 0,K-1
3                  DO 105 N = 0,K-1
4                    RLNOM  = RLNOM+B(M+1,N+1)*(R1(M+1,N+1,1)*R2(M+1,N+1,1)
4           *                                - R1(M+1,N+1,2)*R2(M+1,N+1,2))
4                    IMGNOM = IMGNOM+B(M+1,N+1)*(R1(M+1,N+1,1)*R2(M+1,N+1,2)
4           *                                  + R2(M+1,N+1,1)*R1(M+1,N+1,2))
4                    RLDEN  = RLDEN+A(M+1,N+1)*(R1(M+1,N+1,1)*R2(M+1,N+1,1)
4           *                                - R1(M+1,N+1,2)*R2(M+1,N+1,2))
4                    IMGDEN = IMGDEN+A(M+1,N+1)*(R1(M+1,N+1,1)*R2(M+1,N+1,2)
4           *                                  + R2(M+1,N+1,1)*R1(M+1,N+1,2))
4        105     CONTINUE
2                ELEMENT = SQRT(RLNOM**2 + IMGNOM**2) /
2           *              SQRT(RLDEN**2 + IMGDEN**2)
2                Z(I,J) = ELEMENT
2                IF (Z(I,J) .GT. ZMAX) THEN
2                  ZMAX = Z(I,J)
2                ENDIF
2                IF (Z(I,J) .LT. ZMIN) THEN
2                  ZMIN = Z(I,J)
2                ENDIF
2        103   CONTINUE
1        102 CONTINUE

         C.  ***** OUTPUT THE Z MATRIX *****
             WRITE (*,205) '*****  Z  M A T R I X  ',IT,' X ',IT,'  *****'
             WRITE (*,212)
```

```
       DO 106 I = 1,IT
         WRITE (*,300) (Z(I,J), J = 1,IT)
         WRITE (*,210)
  106 CONTINUE
      WRITE (*,213)

      WRITE (*,418)
      READ  (*,200) ANSWER
      IF  ((ANSWER .NE. 'Y') .AND. (ANSWER .NE. 'y')) GOTO 15


C     ***** ASK THE PARAMETERS FOR THE GRAPH *****
  20  WRITE (*,210)
      WRITE (*,*) '****  E N T E R  P L O T  P A R A M E T E R S  ****'
      WRITE (*,410)
      READ  (*,*)  AZIM
      WRITE (*,411)
      READ  (*,*)  ELEV
      WRITE (*,413)
      READ  (*,*)  ITRIM
      WRITE (*,414)
      READ  (*,*)  IDIV
      WRITE (*,415)
      READ  (*,199)  CTEXT
      WRITE (*,451)
      READ  (*,200)  ANSWER

C     ***** INITIALIZE PLOT88 *****
      IF  ((ANSWER .EQ. 'Y') .OR. (ANSWER .EQ. 'y'))  THEN
        CALL PLOTS(0,0,2)
      ELSE
        CALL PLOTS(0,99,99)
      ENDIF

      WRITE (*,420)
      READ (*,200) ANSWER

      CALL WINDOW(XLOL,YLOL,XUPR,YUPR)

      IF  ((ANSWER .EQ. 'Y') .OR. (ANSWER .EQ. 'y'))  THEN
        DLEV = (ZMAX-ZMIN)/FLOAT(IT)
        CALL ZLEVEL(Z,51,51,IT,IT,DLEV,ZLEV,IT+1)
        DO 108 I = 1,IT+1
          LDIG(I) = 2
          LWGT(I) = 1
  108   CONTINUE
C       ***** DRAW THE CONTOUR MAP *****
        CALL ZCNTUR(Z,51,51,IT,IT,0.5,0.5,8.25,6.5,ZLEV,LDIG,LWGT,
     *              IT+1,0.10,10)
        CALL SYMBOL(5.5,0.0,0.2,'CONTOUR MAP',0.0,11)
      ELSE
```

```
C       ***** DRAW THE MESH SURFACE OF THE GRAPH *****
        CALL MESHS(Z,51,51,IT,IT,AZIM,ELEV,0.5,0.5,8.25,6.5,IDIV,0,
     *             3,IPROJ,1,ZLOW,3,ITRIM,MASK,VERTEX)
C       ***** ANNOTATION OF THE GRAPH *****
        CALL SYMBOL(5.5,0.3,0.2,'AZIMUTH:   ',0.0,10)
        CALL NUMBER(999.0,999.0,0.2,AZIM,0.0,2)
        CALL SYMBOL(5.5,0.0,0.2,'ELEVATION:',0.0,10)
        CALL NUMBER(999.0,999.0,0.2,ELEV,0.0,2)
        DY = (Z(1,1)/90.0) * ELEV
        CALL P3D2D(1.0,1.0,Z(1,1)-DY,XR,YR)
        CALL SYMBOL(XR,YR,0.25,'*',0.0,1)
        CALL SYMBOL(1.0,0.1,0.2,'* = ORIGIN',0.0,10)
      ENDIF

      CALL SYMBOL(1.0,6.75,0.25,CTEXT,0.0,20)
      CALL SYMBOL(6.0,6.5,0.2,'2-D DATA FIELD',0.0,14)

C     ***** OUTPUT THE GRAPH *****
      CALL PLOT(0.0,0.0,999)

      WRITE (*,416)
      READ  (*,200) ANSWER
      IF  ((ANSWER .EQ. 'Y') .OR. (ANSWER .EQ. 'y')) GOTO 20
 15   WRITE (*,417)
      READ  (*,200) ANSWER
      IF  ((ANSWER .EQ. 'Y') .OR. (ANSWER .EQ. 'y')) GOTO 10
      STOP

  199 FORMAT(A20)
  200 FORMAT(A)
  205 FORMAT(/,20X,A25,I2,A3,I2,A8,/)
  210 FORMAT()
  211 FORMAT(/,5X,A60)
  212 FORMAT(/,2X,'(AZIMUTH 320.0)',46X,'(AZIMUTH 230.0)',/)
  213 FORMAT(/,2X,'(AZIMUTH 050.0)',46X,'(AZIMUTH 140.0)',/)
  300 FORMAT(10(F7.2,1X))
  400 FORMAT(9X,\)
  451 FORMAT(/,5X,'SEND GRAPH TO THE PRINTER(Y or N): ',\)
  401 FORMAT(/,5X,'DIMENSION OF OUTPUT MATRIX(1 to 25): ',\)
  402 FORMAT(/,5X,'ORDER OF TRANSFER FUNCTION(0 to 4): ',\)
  404 FORMAT(5X,A2,I1,A,I1,A3,\)
  410 FORMAT(/,5X,'AZIMUTH(0.0 to 360.0 DEGREES): ',\)
  411 FORMAT(/,5X,'ELEVATION(90.0 to -90.0 DEGREES): ',\)
  413 FORMAT(/,5X,'TRIM(0=NO,1=Xs,2=Ys): ',\)
  414 FORMAT(/,5X,'2,4 OR 8 SUBGRIDS: ',\)
  415 FORMAT(/,5X,'TITLE OF GRAPH(UP TO 20 CHAR): ',\)
  416 FORMAT(/,5X,'DO YOU WANT TO CHANGE PARAMETERS ? ',\)
  417 FORMAT(/,5X,'DO YOU WANT TO REPEAT THE PROCESS ? ',\)
  418 FORMAT(/,5X,'DO YOU WANT TO MAKE GRAPH ? ',\)
  419 FORMAT(/,5X,'DO YOU WANT TO CORRECT ? ',\)
  420 FORMAT(/,5X,'DO YOU WANT CONTOUR MAP ? ',\)
```

```
                                                                    Page    3
                                                                    09-26-85
                                                                    21:09:36
D Line# 1      7                              Microsoft FORTRAN77 V3.20 02/84
    205        END

Name     Type          Offset P Class

A        REAL               2
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
```

```
                                                                                     Page    6
                                                                                     09-26-85
                                                                                     21:09:36
D Line# 1      7                                    Microsoft FORTRAN77 V3.20 02/84
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

```
                                                            Page   1
                                                            09-26-85
                                                            20:42:32
D Line# 1     7                     Microsoft FORTRAN77 V3.20 02/84
$STORAGE:  2
$PAGESIZE:58

C     ****************************************************************
C     *                                                              *
C     *  THE PURPOSE OF THIS PROGRAM IS  TO  COMPUTE  AND GRAPH THE  *
C     *  EQUATIONS OF ROBERT P.ROESSER IN THE "DISCRETE STATE-SPACE  *
C     *  MODEL FOR LINEAR IMAGE PROCESSING". IT TRANSFORMS ALSO THE  *
C     *  OUTPUT MATRIX Y ACCORDING TO FOURIER ANALYSIS.              *
C     *                                                              *
C     *                     EVANGELOS THEOFILOU                      *
C     ****************************************************************
C     PROGRAM 2D-DATA-FIELD

C     ***** VARIABLE DECLARATIONS *****
      REAL         R1(26,26),R2(26,26),S1(26,26),S2(26,26),
     +             FR1(2),FR2(2),TRM(4,4),IV(4),OV(4),IMGPART
      CHARACTER*1  ANSWER

C     **** VARIABLE DECLARATIONS FOR PLOT88 *****
      CHARACTER*20 CTEXT
      COMMON        /WORK /Z(26,26),ZF(26,26),ZLEV(26),LDIG(26),
     +              LWGT(26),MASK(3000),VERTEX(16)

      DATA          XLOL/0.0/,YLOL/0.0/,XUPR/8.5/,YUPR/7.0/,
     +              ZLOW/1.0E35/,IPROJ/0/,NRNG/100/

C     *************  M A I N   P R O G R A M  *************

C     ***** ASK THE REQUIRED VALUES FOR THE MODEL *****
   10 WRITE  (*,403)
      READ   (*,*) KK
      IF ((KK .LT. 3) .OR. (KK .GT. 25)) GOTO 10

      DO 100 I = 1,KK+1
        DO 100 J = 1,KK+1
          R1(I,J)  = 0.0
          R2(I,J)  = 0.0
          S1(I,J)  = 0.0
          S2(I,J)  = 0.0
  100 CONTINUE

      DO 101 I = 1,4
         DO 101 J = 1,4
            TRM(I,J)  = 0.0
  101 CONTINUE

      DO 102 I = 1,4
         IV(I)  = 0.0
         OV(I)  = 0.0
  102 CONTINUE
```

```

      WRITE (*,211) 'ENTER INITIAL CONDITIONS FOR HORIZONTAL R1(#.#)'
      DO 103 I = 1,KK
        WRITE (*,404) 'R1(1,',I,'): '
        READ  (*,*) R1(1,I)
  103 CONTINUE

      WRITE (*,211) 'ENTER INITIAL CONDITIONS FOR HORIZONTAL R2(#.#)'
      DO 104 I = 1,KK
        WRITE (*,404) 'R2(1,',I,'): '
        READ  (*,*) R2(1,I)
  104 CONTINUE

      WRITE (*,211) 'ENTER INITIAL CONDITIONS FOR VERTICAL S1(#.#) '
      DO 105 I = 1,KK
        WRITE (*,405) 'S1(',I,',1): '
        READ  (*,*) S1(I,1)
  105 CONTINUE

      WRITE (*,211) 'ENTER INITIAL CONDITIONS FOR VERTICAL S2(#.#) '
      DO 106 I = 1,KK
        WRITE (*,405) 'S2(',I,',1): '
        READ  (*,*) S2(I,1)
  106 CONTINUE

      WRITE (*,211) 'ENTER VALUES FOR THE OUTPUT VECTOR(#.#)        '
      OV(1) = 1
      WRITE (*,409) 'b01: '
      READ  (*,*) OV(3)
      WRITE (*,409) 'a01: '
      READ  (*,*) OV(4)

      WRITE (*,211) 'ENTER ELEMENTS OF THE TRANSITION MATRIX(#.#)   '
      TRM(1,2) = 1
      TRM(4,1) = 1
      WRITE (*,409) 'a10: '
      READ  (*,*) TRM(1,1)
      WRITE (*,409) 'a20: '
      READ  (*,*) TRM(2,1)
      WRITE (*,409) 'b11: '
      READ  (*,*) TEMP
      TRM(1,3) = TEMP + OV(3)*TRM(1,1)
      WRITE (*,409) 'a11: '
      READ  (*,*) TEMP
      TRM(1,4) = TEMP + OV(4)*TRM(1,1)
      WRITE (*,409) 'b21: '
      READ  (*,*) TEMP
      TRM(2,3) = TEMP + OV(3)*TRM(2,1)
      WRITE (*,409) 'a21: '
      READ  (*,*) TEMP
      TRM(2,4) = TEMP + OV(4)*TRM(2,1)
```

```
D Line# 1      7                               Microsoft FORTRAN77 V3.20 02/84
              TRM(4,3) = OV(3)
              TRM(4,4) = OV(4)

              WRITE (*,211) 'ENTER VALUES FOR THE INPUT VECTOR(#.#)'
              IV(3) = 1
              WRITE (*,409) 'b00: '
              READ (*,*) IV(4)
              WRITE (*,409) 'b10: '
              READ (*,*) TEMP
              IV(1) = TEMP + IV(4)*TRM(1,1)
              IV(2) = IV(4)*TRM(2,1)

              U = 1.0
              DO 107 I = 1,KK
1               DO 107 J = 1,KK
2               R1(I+1,J) = TRM(1,1)*R1(I,J) + R2(I,J) + TRM(1,3)*S1(I,J) +
2            *                                     TRM(1,4)*S2(I,J) + IV(1)*U
2               R2(I+1,J) = TRM(2,1)*R1(I,J) + TRM(2,3)*S1(I,J) +
2            *                               TRM(2,4)*S2(I,J) + IV(2)*U
2               S1(I,J+1) = U
2               S2(I,J+1) = R1(I,J) + OV(3)*S1(I,J) + OV(4)*S2(I,J) + IV(4)*U
2               U = 0.0
2         107 CONTINUE

              WRITE (*,205) '***** INPUT VECTOR *****'
              WRITE (*,300) (IV(I),I = 1,4)

              WRITE (*,205) '***** OUTPUT VECTOR *****'
              WRITE (*,300) (OV(I),I = 1,4)

              WRITE (*,205) '***** TRANSITION MATRIX *****'
              DO 108 I = 1,4
1               WRITE (*,300) (TRM(I,J),J = 1,4)
1               WRITE (*,210)
1         108 CONTINUE

        C     **** FILL 0's THE TWO DIMENTIONAL GRID OF CONTROL POINTS ****
              DO 109 I = 1,26
1               DO 109 J = 1,26
2                 Z(I,J) =0.0
2         109 CONTINUE

              DO 110 I = 1,KK
1               DO 110 J = 1,KK
2                 Z(I,J) = R1(I,J) + OV(3)*S1(I,J) + OV(4)*S2(I,J)
2         110 CONTINUE

              WRITE (*,205) '***** R1  M A T R I X  ',KK,' X ',KK,' *****'
              DO 111 I = 1,KK
1               WRITE (*,300) (R1(I,J), J = 1,KK)
1               WRITE (*,210)
```

```
1    111 CONTINUE

         WRITE (*,205) '***** R2   M A T R I X  ',KK,' X ',KK,' *****'
         DO 112 I = 1,KK
1          WRITE (*,300) (R2(I,J), J = 1,KK)
1          WRITE (*,210)
1    112 CONTINUE

         WRITE (*,205) '***** S1   M A T R I X  ',KK,' X ',KK,' *****'
         DO 113 I = 1,KK
1          WRITE (*,300) (S1(I,J), J = 1,KK)
1          WRITE (*,210)
1    113 CONTINUE

         WRITE (*,205) '***** S2   M A T R I X  ',KK,' X ',KK,' *****'
         DO 114 I = 1,KK
1          WRITE (*,300) (S2(I,J), J = 1,KK)
1          WRITE (*,210)
1    114 CONTINUE

     C   ***** OUTPUT THE Y MATRIX *****
         WRITE (*,205) '*****   Z   M A T R I X  ',KK,' X ',KK,'  *****'
         WRITE (*,212)
         DO 115 I = 1,KK
1          WRITE (*,300) (Z(I,J), J = 1,KK)
1          WRITE (*,210)
1    115 CONTINUE
         WRITE (*,213)

         WRITE(*,419)
         READ  (*,200) ANSWER
         IF  ((ANSWER .NE. 'Y') .AND. (ANSWER .NE. 'y')) GOTO 21

     C   ***** ASK THE PARAMETERS FOR THE GRAPH *****
      20 WRITE (*,210)
         WRITE (*,*) '****  E N T E R  P L O T  P A R A M E T E R S  ****'
         WRITE (*,410)
         READ  (*,*)  AZIM
         WRITE (*,411)
         READ  (*,*)  ELEV
         WRITE (*,413)
         READ  (*,*)  ITRIM
         WRITE (*,414)
         READ  (*,*)  IDIV
         WRITE (*,415)
         READ  (*,199)  CTEXT
         WRITE (*,451)
         READ  (*,200)  ANSWER

     C   ***** INITIALIZE PLOT88 *****
         IF  ((ANSWER .EQ. 'Y') .OR. (ANSWER .EQ. 'y'))  THEN
         CALL PLOTS(0,0,2)
      ELSE
         CALL PLOTS(0,99,99)
      ENDIF

      CALL WINDOW(XLOL,YLOL,XUPR,YUPR)

C     ***** DRAW THE MESH SURFACE OF THE GRAPH *****
      CALL MESHS(Z,26,26,KK,KK,AZIM,ELEV,0.5,0.5,8.25,6.5,IDIV,0,
     *             3,IPROJ,1,ZLOW,3,ITRIM,MASK,VERTEX)

C     ***** ANNOTATION OF THE GRAPH *****
      CALL SYMBOL(1.0,6.75,0.25,CTEXT,0.0,20)
      CALL SYMBOL(6.0,6.5,0.2,'2-D DATA FIELD',0.0,14)
      CALL SYMBOL(5.5,0.3,0.2,'AZIMUTH:   ',0.0,10)
      CALL NUMBER(999.0,999.0,0.2,AZIM,0.0,2)
      CALL SYMBOL(5.5,0.0,0.2,'ELEVATION:',0.0,10)
      CALL NUMBER(999.0,999.0,0.2,ELEV,0.0,2)
      DY = (Z(1,1)/90.0) * ELEV
      CALL P3D2D(1.0,1.0,Z(1,1)-DY,XR,YR)
      CALL SYMBOL(XR,YR,0.25,'*',0.0,1)
      CALL SYMBOL(1.0,0.1,0.2,'* = ORIGIN',0.0,10)

C     ***** OUTPUT THE GRAPH *****
      CALL PLOT(0.0,0.0,999)
      WRITE (*,416)
      READ  (*,200) ANSWER
      IF  ((ANSWER .EQ. 'Y') .OR. (ANSWER .EQ. 'y')) GOTO 20

 21   WRITE(*,418)
      READ(*,200)  ANSWER
      IF  ((ANSWER .EQ. 'Y') .OR. (ANSWER .EQ. 'y')) THEN

C        **** FILL 0's THE TWO DIMENTIONAL GRID OF CONTROL POINTS ****
         DO 116 I = 1,26
           DO 116 J = 1,26
             ZF(I,J) =0.0
 116     CONTINUE

         ZFMAX = -9.9E20
         ZFMIN = 9.9E20
         DK = (KK - 1) / 2.0
         P = 3.141592
         DO 117 M = 1,KK
           DO 117 N = 1,KK
             RLPART  = 0.0
             IMGPART = 0.0
             DO 118 L = 1,KK
               DO 118 K = 1,KK
                 FR1(1) = COS(-2*P*(L-1)*(M-DK-1)/KK)
                 FR1(2) = SIN(-2*P*(L-1)*(M-DK-1)/KK)
```

```
4                    FR2(1) = COS(-2*P*(K-1)*(N-DK-1)/KK)
4                    FR2(2) = SIN(-2*P*(K-1)*(N-DK-1)/KK)
4                    RLPART  = RLPART  + Z(L,K)*(FR1(1)*FR2(1)
4            *                                  -FR1(2)*FR2(2))
4                    IMGPART = IMGPART + Z(L,K)*(FR1(1)*FR2(2)
4            *                                  +FR1(2)*FR2(1))
4        118       CONTINUE
2                  ZF(M,N) = SQRT(RLPART**2 + IMGPART**2)
2                  IF (ZF(M,N) .GT. ZFMAX) THEN
2                  ZFMAX = ZF(M,N)
2                ENDIF
2                IF (ZF(M,N) .LT. ZFMIN) THEN
2                  ZFMIN = ZF(M,N)
2                ENDIF
2        117   CONTINUE

       C       ***** OUTPUT THE ZF MATRIX *****
               WRITE (*,205) '*** FOURIER TRANSFORMATION   ',KK,' X ',KK,' ***'
               WRITE (*,212)
               DO 119 I = 1,KK
1                WRITE (*,300) (ZF(I,J), J = 1,KK)
1                WRITE (*,210)
1        119   CONTINUE
               WRITE (*,213)

               WRITE(*,419)
               READ  (*,200) ANSWER
               IF  ((ANSWER .NE. 'Y') .AND. (ANSWER .NE. 'y')) GOTO 22

       C       ***** ASK THE PARAMETERS FOR THE GRAPH *****
         30    WRITE (*,210)
               WRITE (*,*) '***  E N T E R  P L O T  P A R A M E T E R S  ***'
               WRITE (*,410)
               READ  (*,*)  AZIM
               WRITE (*,411)
               READ  (*,*)  ELEV
               WRITE (*,413)
               READ  (*,*)  ITRIM
               WRITE (*,414)
               READ  (*,*)  IDIV
               WRITE (*,415)
               READ  (*,199)  CTEXT
               WRITE (*,451)
               READ  (*,200)  ANSWER

       C       ***** INITIALIZE PLOT88 *****
               IF  ((ANSWER .EQ. 'Y') .OR. (ANSWER .EQ. 'y'))  THEN
                 CALL PLOTS(0,0,2)
               ELSE
                 CALL PLOTS(0,99,99)
               ENDIF
```

```

        WRITE (*,420)
        READ (*,200) ANSWER

        CALL WINDOW(XLOL, YLOL, XUPR, YUPR)

        IF ((ANSWER .EQ. 'Y') .OR. (ANSWER .EQ. 'y')) THEN
          DLEV = (ZFMAX-ZFMIN)/FLOAT(KK)
          CALL ZLEVEL(ZF,26,26,KK,KK,DLEV,ZLEV,KK+1)
          DO 136 I = 1,KK+1
            LDIG(I) = 2
            LWGT(I) = 1
  136     CONTINUE
          CALL ZCNTUR(ZF,26,26,KK,KK,0.5,0.5,8.25,6.5,ZLEV,LDIG,LWGT,
     *                KK+1,0.10,10)
          CALL SYMBOL(5.5,0.0,0.2,'CONTOUR MAP',0.0,11)
        ELSE
C         ***** DRAW THE MESH SURFACE OF THE GRAPH *****
          CALL MESHS(ZF,26,26,KK,KK,AZIM,ELEV,0.5,0.5,8.25,6.5,IDIV,0,
     *               3,IPROJ,1,ZLOW,3,ITRIM,MASK,VERTEX)

C         ***** ANNOTATION OF THE GRAPH *****
          CALL SYMBOL(5.5,0.3,0.2,'AZIMUTH:   ',0.0,10)
          CALL NUMBER(999.0,999.0,0.2,AZIM,0.0,2)
          CALL SYMBOL(5.5,0.0,0.2,'ELEVATION:',0.0,10)
          CALL NUMBER(999.0,999.0,0.2,ELEV,0.0,2)
          DY = (ZF(1,1)/90.0) * ELEV
          CALL P3D2D(1.0,1.0,ZF(1,1)-DY,XR,YR)
          CALL SYMBOL(XR,YR,0.25,'*',0.0,1)
          CALL SYMBOL(1.0,0.1,0.2,'* = ORIGIN',0.0,10)
        ENDIF
        CALL SYMBOL(1.0,6.75,0.25,CTEXT,0.0,20)
        CALL SYMBOL(6.0,6.5,0.2,'2-D DFT',0.0,7)

C       ***** OUTPUT THE GRAPH *****
        CALL PLOT(0.0,0.0,999)
        WRITE (*,416)
        READ  (*,200) ANSWER
        IF  ((ANSWER .EQ. 'Y') .OR. (ANSWER .EQ. 'y')) GOTO 30
 22   ENDIF
      WRITE (*,417)
      READ  (*,200) ANSWER
      IF  ((ANSWER .EQ. 'Y') .OR. (ANSWER .EQ. 'y')) GOTO 10
      STOP

  199 FORMAT(A20)
  200 FORMAT(A)
  205 FORMAT(/,18X,A29,I2,A3,I2,A8,/)
  210 FORMAT()
  211 FORMAT(/,5X,A56)
  212 FORMAT(/,2X,'(AZIMUTH 320.0)',46X,'(AZIMUTH 230.0)',/)
```

```
D Line# 1       7                                   Microsoft FORTRAN77 V3.20 02/84
      213 FORMAT(/,2X,'(AZIMUTH 050.0)',46X,'(AZIMUTH 140.0)',/)
      300 FORMAT(10(F7.2,1X))
      400 FORMAT(9X,\)
      403 FORMAT(/,5X,'DIMENSION OF OUTPUT(K=1to20): ',\)
      404 FORMAT(5X,A5,I2,A3,\)
      405 FORMAT(5X,A3,I2,A5,\)
      406 FORMAT(5X,A2,I2,A4,\)
      407 FORMAT(5X,A2,I2,I2,A3,\)
      408 FORMAT(5X,A3,I2,A3,\)
      409 FORMAT(5X,A5,\)
      410 FORMAT(/,5X,'AZIMUTH(0.0 to 360.0 DEGREES): ',\)
      411 FORMAT(/,5X,'ELEVATION(90.0 to -90.0 DEGREES): ',\)
      413 FORMAT(/,5X,'TRIM(0=NO,1=Xs,2=Ys): ',\)
      414 FORMAT(/,5X,'2,4 OR 8 SUBGRIDS: ',\)
      415 FORMAT(/,5X,'TITLE OF GRAPH(UP TO 20 CHAR): ',\)
      416 FORMAT(/,5X,'DO YOU WANT TO CHANGE PARAMETERS? ',\)
      417 FORMAT(/,5X,'DO YOU WANT TO REPEAT THE PROCESS? ',\)
      418 FORMAT(/,5X,'DO YOU WANT FOURIER TRANSFORMATION ? ',\)
      419 FORMAT(/,5X,'DO YOU WANT TO MAKE GRAPH ? ',\)
      420 FORMAT(/,5X,'DO YOU WANT CONTOUR MAP ? ',\)
      451 FORMAT(/,5X,'SEND GRAPH TO THE PRINTER(Y or N): ',\)
          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

```
$STORAGE:  2
$PAGESIZE:58

C      **********************************************************************
C      *                                                                    *
C      *  THE PURPOSE OF THIS PROGRAM IS  TO  COMPUTE  AND GRAPH THE        *
C      *  EQUATIONS OF ROBERT P.ROESSER IN THE "DISCRETE STATE-SPACE        *
C      *  MODEL FOR LINEAR IMAGE PROCESSING". IT TRANSFORMS ALSO THE        *
C      *  OUTPUT MATRIX Y ACCORDING TO FOURIER ANALYSIS.                    *
C      *                                                                    *
C      *                     EVANGELOS THEOFILOU                            *
C      **********************************************************************
C      PROGRAM 2D-DATA-FIELD

C      ***** VARIABLE DECLARATIONS *****
       REAL          R(26,26,4),S1(26,26,4),S2(26,26,4),
      *              R1(2),R2(2),TRM(12,12),IV(12),OV(12),IMGPART
       CHARACTER*1 ANSWER

C      **** VARIABLE DECLARATIONS FOR PLOT88 *****
       CHARACTER*20 CTEXT
       COMMON        /WORK /Z(26,26),ZF(26,26),ZLEV(26),LDIG(26),
      *               LWGT(26),MASK(3000),VERTEX(16)

       DATA           XLOL/0.0/,YLOL/0.0/,XUPR/8.5/,YUPR/7.0/,
      *               ZLOW/1.0E35/,IPROJ/0/,NRNG/100/

C      *************   M A I N   P R O G R A M   ************

C      ***** ASK THE REQUIRED VALUES FOR THE MODEL *****
    10 WRITE (*,401)
       READ  (*,*)  N
       IF ((N .LT. 1) .OR. (N .GT. 4)) GOTO 10
     2 WRITE (*,402)
       READ  (*,*)  M
       IF ((M .LT. 1) .OR. (M .GT. 4)) GOTO 2
     3 WRITE (*,403)
       READ  (*,*)  KK
       IF (KK .GT. 25) GOTO 3

       DO 100 I = 1,KK+1
         DO 100 J = 1,KK+1
           DO 100 L = 1,N
             R(I,J,L)  = 0.0
             S1(I,J,L) = 0.0
             S2(I,J,L) = 0.0
   100 CONTINUE

       DO 101 I = 1,N+2*M
         DO 101 J = 1,N+2*M
           TRM(I,J) = 0.0
```

```
101 CONTINUE
      DO 102 I = 1,N+2*M
        IV(I) = 0.0
        OV(I) = 0.0
102 CONTINUE

      WRITE (*,211) 'ENTER INITIAL CONDITIONS FOR HORIZONTAL R(#.#)'
      DO 103 I = 1,KK
        DO 103 J = 1,N
          WRITE (*,404) 'R',J,'(1,',I,'): '
          READ  (*,*) R(1,I,J)
103 CONTINUE

      WRITE (*,211) 'ENTER INITIAL CONDITIONS FOR VERTICAL S1(#.#) '
      DO 104 I = 1,KK
        DO 104 J = 1,M
          WRITE (*,405) 'S1(',J,')(',I,',1): '
          READ  (*,*) S1(I,1,J)
104 CONTINUE

      WRITE (*,211) 'ENTER INITIAL CONDITIONS FOR VERTICAL S2(#.#) '
      DO 105 I = 1,KK
        DO 105 J = 1,M
          WRITE (*,405) 'S2(',J,')(',I,',1): '
          READ  (*,*) S2(I,1,J)
105 CONTINUE

      WRITE (*,211) 'ENTER VALUES FOR THE INPUT VECTOR(#.#)          '
      IV(1) = 1.0
      DO 106 I = 1,M
        WRITE (*,408) 'a(0',I,'): '
        READ (*,*) IV(N+I)
        TRM(N+I,N+1) = -IV(N+I)
106 CONTINUE
      DO 107 I = 1,M
        WRITE (*,408) 'b(0',I,'): '
        READ (*,*) IV(N+M+I)
        TRM(N+M+I,N+1) = -IV(N+M+I)
107 CONTINUE
      DO 108 I = 1,M-1
        TRM(N+I,N+I+1) = 1.0
108 CONTINUE
      DO 109 I = 1,M-1
        TRM(N+M+I,N+M+I+1) = 1.0
109 CONTINUE

      WRITE (*,211) 'ENTER ELEMENTS OF THE TRANSITION MATRIX(#.#)  '
      DO 110 I = 1,N
        WRITE (*,406) 'a(',I,'0): '
        READ  (*,*) TEMP
        TRM(1,I) = -TEMP
```

```
  110 CONTINUE
      TRM(1,N+1) = -1.0
      DO 111 I = 2,N
        TRM(I,I-1) = 1.0
  111 CONTINUE
      DO 112 I = 1,M
        DO 112 J = 1,N
          WRITE (*,407) 'a(',J,I,'): '
          READ  (*,*) TEMP1
          TRM(I+N,J) = TEMP1 + TRM(1,J) * IV(N+I)
  112 CONTINUE
      DO 113 I = 1,M
        DO 113 J = 1,N
          WRITE (*,407) 'b(',J,I,'): '
          READ  (*,*)  TEMP1
          TRM(I+N+M,J) = TEMP1 + TRM(1,J) * IV(N+M+I)
  113 CONTINUE

      WRITE (*,211) 'ENTER VALUES FOR THE OUTPUT VECTOR(#.#)         '
      WRITE (*,409) 'b(00): '
      READ  (*,*) TEMP
      OV(N+1) = -TEMP
      OV(N+M+1) = 1.0
      DO 114 I = 1,N
        WRITE (*,406) 'b(',I,'0): '
        READ  (*,*) TEMP1
        OV(I) = TEMP1 + TRM(1,I) * TEMP
  114 CONTINUE

      U = 1.0
      DO 115 I = 1,KK
        DO 115 J = 1,KK
          DO 116 II = 1,N+2*M

            IF  (II .LE. N)  THEN
              DO 117 JJ = 1,N+2*M
                IF  (JJ .LE. N)  THEN
                  R(I+1,J,II)=R(I+1,J,II)+TRM(II,JJ)*R(I,J,JJ)
                ENDIF
                IF  ((JJ .GT. N) .AND. (JJ .LE. N+M))  THEN
                  R(I+1,J,II)=R(I+1,J,II)+TRM(II,JJ)*S1(I,J,JJ-N)
                ENDIF
  117         CONTINUE
              R(I+1,J,II)=R(I+1,J,II) + IV(II) * U
            ENDIF

            IF  ((II .GT. N) .AND. (II .LE. N+M))  THEN
              DO 118 JJ = 1,N+2*M
                IF  (JJ .LE. N)  THEN
                  S1(I,J+1,II-N) = S1(I,J+1,II-N) + TRM(II,JJ) *
     *                                                 R(I,J,JJ)
```

```
4                    ENDIF
4                    IF ((JJ .GT. N).AND.(JJ .LE. N+M)) THEN
4                      S1(I,J+1,II-N) = S1(I,J+1,II-N) + TRM(II,JJ)*
4          *                                           S1(I,J,JJ-N)
4                    ENDIF
3     118          CONTINUE
3                  S1(I,J+1,II-N) = S1(I,J+1,II-N) + IV(II) * U
3                ENDIF
3
3                IF (II .GT. N+M) THEN
3                  DO 119 JJ = 1,N+2*M
4                    IF  (JJ .LE. N)  THEN
4                      S2(I,J+1,II-N-M) = S2(I,J+1,II-N-M) + TRM(II,JJ)
4          *                                                 * R(I,J,JJ)
4                    ENDIF
4                    IF  ((JJ .GT. N) .AND. (JJ .LE. N+M))  THEN
4                      S2(I,J+1,II-N-M) = S2(I,J+1,II-N-M) + TRM(II,JJ)
4          *                                              * S1(I,J,JJ-N)
4                    ENDIF
4                    IF  (JJ .GT. N+M)  THEN
4                      S2(I,J+1,II-N-M) = S2(I,J+1,II-N-M) + TRM(II,JJ)
4          *                                            * S2(I,J,JJ-N-M)
4                    ENDIF
4     119          CONTINUE
3                  S2(I,J+1,II-N-M) = S2(I,J+1,II-N-M) + IV(II) * U
3                ENDIF
3     116      CONTINUE
2               U = 0.0
2     115 CONTINUE

          WRITE (*,205) '***** INPUT VECTOR *****'
          WRITE (*,300) (IV(I),I = 1,N+2*M)

          WRITE (*,205) '***** OUTPUT VECTOR *****'
          WRITE (*,300) (OV(I),I = 1,N+2*M)

          WRITE (*,205) '***** TRANSITION MATRIX *****'
          DO 120 I = 1,N+2*M
1           WRITE (*,300) (TRM(I,J),J = 1,N+2*M)
1           WRITE (*,210)
1     120 CONTINUE

      C   **** FILL 0's THE TWO DIMENTIONAL GRID OF CONTROL POINTS ****
          DO 121 I = 1,26
1           DO 121 J = 1,26
2             Z(I,J) =0.0
2     121 CONTINUE

          DO 122 I = 1,KK
1           DO 122 J = 1,KK
2             DO 123 LL = 1,N+2*M
```

```
D Line# 1      7                                      Microsoft FORTRAN77 V3.20 02/84
3                    IF (LL .LE. N) THEN
3                      Z(I,J) = Z(I,J) + OV(LL) * R(I,J,LL)
3                    ENDIF
3                    IF ((LL .GT. N).AND.(LL .LE. N+M)) THEN
3                      Z(I,J) = Z(I,J) + OV(LL) * S1(I,J,LL-N)
3                    ENDIF
3                    IF (LL .GT. N+M)  THEN
3                      Z(I,J) = Z(I,J) + OV(LL) * S2(I,J,LL-N-M)
3                    ENDIF
3         123      CONTINUE
2         122 CONTINUE

              WRITE (*,205) '***** R  M A T R I X  ',KK,' X ',KK,' *****'
              DO 124 I = 1,KK
1               DO 125 L = 1,N
2                 WRITE (*,300) (R(I,J,L), J = 1,KK)
2         125   CONTINUE
1               WRITE (*,210)
1         124 CONTINUE

              WRITE (*,205) '***** S1  M A T R I X  ',KK,' X ',KK,' *****'
              DO 126 I = 1,KK
1               DO 127 L = 1,M
2                 WRITE (*,300) (S1(I,J,L), J = 1,KK)
2         127   CONTINUE
1               WRITE (*,210)
1         126 CONTINUE

              WRITE (*,205) '***** S2  M A T R I X  ',KK,' X ',KK,' *****'
              DO 128 I = 1,KK
1               DO 129 L = 1,M
2                 WRITE (*,300) (S2(I,J,L), J = 1,KK)
2         129   CONTINUE
1               WRITE (*,210)
1         128 CONTINUE

        C     ***** OUTPUT THE Z MATRIX *****
              WRITE (*,205) '*****  Z  M A T R I X  ',KK,' X ',KK,'  *****'
              WRITE (*,212)
              DO 130 I = 1,KK
1               WRITE (*,300) (Z(I,J), J = 1,KK)
1               WRITE (*,210)
1         130 CONTINUE
              WRITE (*,213)

              WRITE(*,419)
              READ  (*,200) ANSWER
              IF ((ANSWER .NE. 'Y') .AND. (ANSWER .NE. 'y')) GOTO 21

        C     ***** ASK THE PARAMETERS FOR THE GRAPH *****
          20  WRITE (*,210)
```

```
      WRITE (*,*) '****  E N T E R   P L O T   P A R A M E T E R S  ****'
      WRITE (*,410)
      READ  (*,*)  AZIM
      WRITE (*,411)
      READ  (*,*)  ELEV
      WRITE (*,413)
      READ  (*,*)  ITRIM
      WRITE (*,414)
      READ  (*,*)  IDIV
      WRITE (*,415)
      READ  (*,199)  CTEXT
      WRITE (*,451)
      READ  (*,200)  ANSWER

C     ***** INITIALIZE PLOT88 *****
      IF  ((ANSWER .EQ. 'Y') .OR. (ANSWER .EQ. 'y'))  THEN
         CALL PLOTS(0,0,2)
      ELSE
         CALL PLOTS(0,99,99)
      ENDIF

      CALL WINDOW(XLOL,YLOL,XUPR,YUPR)

C     ***** DRAW THE MESH SURFACE OF THE GRAPH *****
      CALL MESHS(Z,26,26,KK,KK,AZIM,ELEV,0.5,0.5,8.25,6.5,IDIV,0.
     *           3,IPROJ,1,ZLOW,3,ITRIM,MASK,VERTEX)

C     ***** ANNOTATION OF THE GRAPH *****
      CALL SYMBOL(1.0,6.75,0.25,CTEXT,0.0,20)
      CALL SYMBOL(6.0,6.5,0.2,'2-D DATA FIELD',0.0,14)
      CALL SYMBOL(5.5,0.3,0.2,'AZIMUTH:   ',0.0,10)
      CALL NUMBER(999.0,999.0,0.2,AZIM,0.0,2)
      CALL SYMBOL(5.5,0.0,0.2,'ELEVATION:',0.0,10)
      CALL NUMBER(999.0,999.0,0.2,ELEV,0.0,2)
      DY = (Z(1,1)/90.0) * ELEV
      CALL P3D2D(1.0,1.0,Z(1,1)-DY,XR,YR)
      CALL SYMBOL(XR,YR,0.25,'*',0.0,1)
      CALL SYMBOL(1.0,0.1,0.2,'* = ORIGIN',0.0,10)

C     ***** OUTPUT THE GRAPH *****
      CALL PLOT(0.0,0.0,999)
      WRITE (*,416)
      READ  (*,200) ANSWER
      IF  ((ANSWER .EQ. 'Y') .OR. (ANSWER .EQ. 'y')) GOTO 20

 21   WRITE(*,418)
      READ(*,200)  ANSWER
      IF  ((ANSWER .EQ. 'Y') .OR. (ANSWER .EQ. 'y')) THEN

C        **** FILL 0's THE TWO DIMENTIONAL GRID OF CONTROL POINTS ****
         DO 132 I = 1,26
```

```
        DO 132 J = 1,26
          ZF(I,J) =0.0
 132  CONTINUE

      ZFMAX = -9.9E20
      ZFMIN = 9.9E20
      DK = (KK - 1) / 2.0
      P = 3.141592
      DO 133 M = 1,KK
        DO 133 N = 1,KK
          RLPART  = 0.0
          IMGPART = 0.0
          DO 134 L = 1,KK
            DO 134 K = 1,KK
              R1(1)  = COS(-2*P*(L-1)*(M-DK-1)/KK)
              R1(2)  = SIN(-2*P*(L-1)*(M-DK-1)/KK)
              R2(1)  = COS(-2*P*(K-1)*(N-DK-1)/KK)
              R2(2)  = SIN(-2*P*(K-1)*(N-DK-1)/KK)
              RLPART  = RLPART  + Z(L,K)*(R1(1)*R2(1)
     *                                   -R1(2)*R2(2))
              IMGPART = IMGPART + Z(L,K)*(R1(1)*R2(2)
     *                                   +R1(2)*R2(1))
 134      CONTINUE
          ZF(M,N) = SQRT(RLPART**2 + IMGPART**2)
          IF (ZF(M,N) .GT. ZFMAX) THEN
          ZFMAX = ZF(M,N)
        ENDIF
        IF (ZF(M,N) .LT. ZFMIN) THEN
          ZFMIN = ZF(M,N)
        ENDIF
 133  CONTINUE

C     ***** OUTPUT THE ZF MATRIX *****
      WRITE (*,205) '*** FOURIER TRANSFORMATION  ',KK,' X ',KK,' ***'
      WRITE (*,212)
      DO 135 I = 1,KK
         WRITE (*,300) (ZF(I,J), J = 1,KK)
         WRITE (*,210)
 135  CONTINUE
      WRITE (*,213)

      WRITE(*,419)
      READ  (*,200) ANSWER
      IF  ((ANSWER .NE. 'Y') .AND. (ANSWER .NE. 'y')) GOTO 22

C     ***** ASK THE PARAMETERS FOR THE GRAPH *****
 30   WRITE (*,210)
      WRITE (*,*) '***  E N T E R  P L O T  P A R A M E T E R S  ***'
      WRITE (*,410)
      READ  (*,*)  AZIM
      WRITE (*,411)
      READ  (*,*)  ELEV
      WRITE (*,413)
      READ  (*,*)  ITRIM
      WRITE (*,414)
      READ  (*,*)  IDIV
      WRITE (*,415)
      READ  (*,199)  CTEXT
      WRITE (*,451)
      READ  (*,200)  ANSWER

C     ***** INITIALIZE PLOT88 *****
      IF  ((ANSWER .EQ. 'Y') .OR. (ANSWER .EQ. 'y'))  THEN
        CALL PLOTS(0,0,2)
      ELSE
        CALL PLOTS(0,99,99)
      ENDIF

      WRITE (*,420)
      READ (*,200) ANSWER

      CALL WINDOW(XLOL,YLOL,XUPR,YUPR)

      IF ((ANSWER .EQ. 'Y') .OR. (ANSWER .EQ. 'y')) THEN
        DLEV = (ZFMAX-ZFMIN)/FLOAT(KK)
        CALL ZLEVEL(ZF,26,26,KK,KK,DLEV,ZLEV,KK+1)
        DO 136 I = 1,KK+1
          LDIG(I) = 2
          LWGT(I) = 1
  136   CONTINUE
        CALL ZCNTUR(ZF,26,26,KK,KK,0.5,0.5,8.25,6.5,ZLEV,LDIG,LWGT,
     *              KK+1,0.10,10)
        CALL SYMBOL(5.5,0.0,0.2,'CONTOUR MAP',0.0,11)
      ELSE
C       ***** DRAW THE MESH SURFACE OF THE GRAPH *****
        CALL MESHS(ZF,26,26,KK,KK,AZIM,ELEV,0.5,0.5,8.25,6.5,IDIV,0,
     *             3,IPROJ,1,ZLOW,3,ITRIM,MASK,VERTEX)

C       ***** ANNOTATION OF THE GRAPH *****
        CALL SYMBOL(5.5,0.3,0.2,'AZIMUTH:   ',0.0,10)
        CALL NUMBER(999.0,999.0,0.2,AZIM,0.0,2)
        CALL SYMBOL(5.5,0.0,0.2,'ELEVATION:',0.0,10)
        CALL NUMBER(999.0,999.0,0.2,ELEV,0.0,2)
        DY = (ZF(1,1)/90.0) * ELEV
        CALL P3D2D(1.0,1.0,ZF(1,1)-DY,XR,YR)
        CALL SYMBOL(XR,YR,0.25,' *',0.0,1)
        CALL SYMBOL(1.0,0.1,0.2,'* = ORIGIN',0.0,10)
      ENDIF
      CALL SYMBOL(1.0,6.75,0.25,CTEXT,0.0,20)
      CALL SYMBOL(6.0,6.5,0.2,'2-D DFT',0.0,7)

C     ***** OUTPUT THE GRAPH *****
```

```
             CALL PLOT(0.0,0.0,999)
             WRITE (*,416)
             READ  (*,200) ANSWER
             IF  ((ANSWER .EQ. 'Y') .OR. (ANSWER .EQ. 'y')) GOTO 30
      22    ENDIF
           WRITE (*,417)
           READ  (*,200) ANSWER
           IF  ((ANSWER .EQ. 'Y') .OR. (ANSWER .EQ. 'y')) GOTO 10
           STOP

       199 FORMAT(A20)
       200 FORMAT(A)
       205 FORMAT(/,18X,A29,I2,A3,I2,A8,/)
       210 FORMAT()
       211 FORMAT(/,5X,A46)
       212 FORMAT(/,2X,'(AZIMUTH 320.0)',46X,'(AZIMUTH 230.0)',/)
       213 FORMAT(/,2X,'(AZIMUTH 050.0)',46X,'(AZIMUTH 140.0)',/)
       300 FORMAT(10(F7.2,1X))
       400 FORMAT(9X,\)
       451 FORMAT(/,5X,'SEND GRAPH TO THE PRINTER(Y or N): ',\)
       401 FORMAT(/,5X,'NUMBER OF HORIZONTAL STATES(N=1to4): ',\)
       402 FORMAT(/,5X,'NUMBER OF VERTICAL STATES(M=1to4): ',\)
       403 FORMAT(/,3X,'DIMENSION OF OUTPUT(1to25): ',\)
       404 FORMAT(5X,A1,I2,A3,I2,A3,\)
       405 FORMAT(5X,A3,I2,A2,I2,A5,\)
       406 FORMAT(5X,A2,I2,A4,\)
       407 FORMAT(5X,A2,I2,I2,A3,\)
       408 FORMAT(5X,A3,I2,A3,\)
       409 FORMAT(5X,A8,\)
       410 FORMAT(/,5X,'AZIMUTH(0.0 to 360.0 DEGREES): ',\)
       411 FORMAT(/,5X,'ELEVATION(90.0 to -90.0 DEGREES): ',\)
       413 FORMAT(/,5X,'TRIM(0=NO,1=Xs,2=Ys): ',\)
       414 FORMAT(/,5X,'2,4 OR 8 SUBGRIDS: ',\)
       415 FORMAT(/,5X,'TITLE OF GRAPH(UP TO 20 CHAR): ',\)
       416 FORMAT(/,5X,'DO YOU WANT TO CHANGE PARAMETERS? ',\)
       417 FORMAT(/,5X,'DO YOU WANT TO REPEAT THE PROCESS? ',\)
       418 FORMAT(/,5X,'DO YOU WANT FOURIER TRANSFORMATION ? ',\)
       419 FORMAT(/,5X,'DO YOU WANT TO MAKE GRAPH ? ',\)
       420 FORMAT(/,5X,'DO YOU WANT CONTOUR MAP ? ',\)
           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
```

```
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           33492
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

```
                                                                  Page    1
                                                                  09-27-85
                                                                  17:23:37
D Line# 1      7                                  Microsoft FORTRAN77 V3.20 02/84
$LARGE
$STORAGE:  2
$PAGESIZE:58

C      ******************************************************************
C      *                                                                *
C      *  THE PURPOSE OF THIS PROGRAM IS  TO  CODE THE 1-D (DISCRETE    *
C      *  TIME) SYSTEM TO A 2-D SPACIAL SYSTEM.                         *
C      *                                                                *
C      *                       EVANGELOS THEOFILOU                      *
C      ******************************************************************
C      PROGRAM 2D-DATA-FIELD

C      ***** VARIABLE DECLARATIONS *****
       REAL         R(25,625),S(25,625),R1(2),R2(2),TRM(50,50),IV(50),
      +             IMGPART
       CHARACTER*1 ANSWER

C      **** VARIABLE DECLARATIONS FOR PLOT88 *****
       CHARACTER*20 CTEXT
       COMMON       /WORK /Z(26,26),ZF(26,26),X(630),Y(630),ZLEV(26),
      +                    LDIG(26),LWGT(26),MASK(3000),VERTEX(16)

       DATA         XLOL/0.0/,YLOL/0.0/,XUPR/8.5/,YUPR/7.0/,
      +             ZLOW/1.0E35/,IPROJ/0/,NRNG/100/

C      *************  M A I N   P R O G R A M  *************

C      ***** ASK THE REQUIRED VALUES FOR THE MODEL *****

    2  WRITE (*,403)
       READ  (*,*) N
       IF ((N .LT. 3) .OR. (N .GT.25)) GOTO 2
    3  WRITE (*,404)
       READ  (*,*) M
       IF ((M .LT. 2) .OR. (M .GT. 25)) GOTO 3

       WRITE (*,401)
       READ (*,200) ANSWER
       IF  ((ANSWER .EQ. 'Y') .OR. (ANSWER .EQ. 'y')) THEN
C         **** FILL 0's THE TWO DIMENTIONAL GRID OF CONTROL POINTS ****
          DO 96 I = 1,26
            DO 96 J = 1,26
              Z(I,J) =0.0
   96     CONTINUE

C         **** ENTER VALUES FOR Y MATRIX ****
          DO 97 I = 1,M*N
            WRITE(*,408) 'Y(',I,'): '
            READ (*,*) R(1,I)
   97     CONTINUE
```

```
D Line# 1      7                                  Microsoft FORTRAN77 V3.20 02/84
               ENDIF
               IF  ((ANSWER .EQ. 'Y') .OR. (ANSWER .EQ. 'y'))  GOTO 4

               WRITE (*,402)
               READ (*,200) ANSWER
               IF  ((ANSWER .EQ. 'Y') .OR. (ANSWER .EQ. 'y')) THEN
      C          **** INITIALIZE THE TRANSITION MATRIX ****
                 DO 98 I = 1,M+N
1                  DO 98 J = 1,M+N
2                    TRM(I,J) =0.0
2         98     CONTINUE

      C          **** ENTER VALUES FOR TRANSITION MATRIX ****
                 DO 99 I = 1,M+N
1                  DO 99 J = 1,M+N
2                    WRITE(*,407) 'T(',I,',',J,'): '
2                    READ (*,*) TRM(I,J)
2         99     CONTINUE
               ENDIF

      C        ***** INITIALIZE R AND S ARRAYS *****
               DO 100 I = 1,25
1                DO 100 J = 1,625
2                    R(I,J) = 0.0
2                    S(I,J) = 0.0
2        100   CONTINUE

      C        ***** INITIALIZE INPUT VECTOR *****
               DO 101 I = 1,50
1                IV(I) = 0.0
1        101   CONTINUE

               WRITE (*,211) 'ENTER INITIAL CONDITIONS FOR HORIZONTAL R#'
               DO 102 I = 1,M
1                  WRITE (*,405) 'R',I,': '
1                  READ  (*,*) R(I,1)
1        102   CONTINUE

               WRITE (*,211) 'ENTER INITIAL CONDITIONS FOR VERTICAL S#'
               DO 103 I = 1,N
1                  WRITE (*,405) 'S',I,': '
1                  READ  (*,*) S(I,1)
1        103   CONTINUE

               WRITE (*,211) 'ENTER VALUES FOR THE INPUT VECTOR'
               IV(1) = 1.0
               WRITE (*,406) 'a01: '
               READ (*,*) IV(M+1)

               IF  ((ANSWER .EQ. 'Y') .OR. (ANSWER .EQ. 'y'))  GOTO 5
      C        ***** INITIALIZE TRANSITION MATRIX *****
      DO 104 I = 1,25
        DO 104 J = 1,25
          TRM(I,J) = 0.0
104   CONTINUE
      WRITE (*,211) 'ENTER ELEMENTS OF THE TRANSITION MATRIX'
      TRM(M+1,M+N) = -IV(M+1)
      WRITE (*,406) 'a10: '
      READ  (*,*) TEMP
      TRM(1,M) = -TEMP
      TRM(1,M+N) = -1.0
      WRITE (*,406) 'a11: '
      READ (*,*) TEMP
      TRM(M+1,M) = TEMP - TRM(1,M) * TRM(M+1,M+N)

      DO 105 I = 2,M
        TRM(I,I-1) = 1.0
105   CONTINUE

      DO 106 I = 2+M,M+N
        TRM (I,I-1) = 1.0
106   CONTINUE

 5    U = 1.0
      DO 107 I = 1,N*M
        DO 108 J = 1,M+N
          IF (J .LE. M) THEN
            DO 109 JJ = 1,M+N
              IF (JJ .LE. M) R(J,I+1) = R(J.I+1) +
     *                                  R(JJ,I)*TRM(J;JJ)
              IF (JJ .GT. M) R(J,I+1) = R(J,I+1) +
     *                                S(JJ-M,I)*TRM(J,JJ)
109         CONTINUE
            R(J,I+1) = R(J,I+1) + IV(J)*U
          ENDIF

          IF (J .GT. M) THEN
            DO 110 JJ = 1,M+N
              IF (JJ .LE. M) S(J-M,I+1) = S(J-M,I+1) +
     *                                      R(JJ,I)*TRM(J,JJ)
              IF (JJ .GT. M) S(J-M,I+1) = S(J-M,I+1) +
     *                                    S(JJ-M,I)*TRM(J,JJ)
110         CONTINUE
            S(J-M,I+1) = S(J-M,I+1) + IV(J)*U
          ENDIF
108     CONTINUE
        U = 0.0
107   CONTINUE

      WRITE (*,211) '***** INPUT VECTOR *****'
      WRITE (*,300) (IV(I),I = 1,M+N)

      WRITE (*,211) '***** TRANSITION MATRIX *****'
      DO 111 I = 1,M+N
        WRITE (*,300) (TRM(I,J),J = 1,M+N)
        WRITE (*,210)
  111 CONTINUE

      WRITE (*,211) '***** H O R I Z O N T A L  S T A T E S  R *****'
      DO 112 I = 1,M*N
        WRITE (*,300) (R(J,I), J = 1,M)
  112 CONTINUE

      WRITE (*,211) '  ***** V E R T I C A L  S T A T E S  S *****'
      DO 113 I = 1,M*N
        WRITE (*,300) (S(J,I), J = 1,N)
  113 CONTINUE

C     **** FILL 0's THE TWO DIMENTIONAL GRID OF CONTROL POINTS ****
      DO 114 I = 1,26
        DO 114 J = 1,26
          Z(I,J) =0.0
  114 CONTINUE

    4 DO 115 I = 1,M
        DO 115 J = 1,N
          Z(I,J) = R(1,(I-1)*N+J)
  115 CONTINUE

C     ***** OUTPUT THE Y ARRAY *****
      DO 119 I = 1,M*N
        WRITE (*,*) R(1,I)
  119 CONTINUE
C     ***** OUTPUT THE Z MATRIX *****
      WRITE (*,205) '*****  Z  M A T R I X  ',M,' X ',N,'  *****'
      WRITE (*,212)
      DO 116 I = 1,M
        WRITE (*,300) (Z(I,J), J = 1,N)
        WRITE (*,210)
  116 CONTINUE
      WRITE (*,213)

      WRITE(*,421)
      READ  (*,200) ANSWER
      IF  ((ANSWER .NE. 'Y') .AND. (ANSWER .NE. 'y')) GOTO 19
      DO 117 I = 1,630
        X(I) = 0.0
        Y(I) = 0.0
  117 CONTINUE

      DO 118 I = 1,M*N
        X(I) = I * 1.0
        Y(I) = R(1,I)
  118 CONTINUE

   18 WRITE (*,415)
      READ  (*,199)  CTEXT
      WRITE (*,451)
      READ  (*,200)  ANSWER

C     ***** INITIALIZE PLOT88 *****
      IF  ((ANSWER .EQ. 'Y') .OR. (ANSWER .EQ. 'y'))  THEN
         CALL PLOTS(0,0,2)
      ELSE
         CALL PLOTS(0,99,99)
      ENDIF

      CALL PLOT(1.0,1.0,-3)
      CALL SCALE(X,6.0,M*N,1)
      CALL SCALE(Y,4.0,M*N,1)
      CALL STAXIS(0.20,0.20,0.111,0.112,1)
      CALL AXIS(0.0,0.0,'X AXIS',-6,6.0,0.0,X(M*N+1),X(M*N+2))
      CALL AXIS(0.0,0.0,'Y AXIS',6,4.0,90.0,Y(M*N+1),Y(M*N+2))
      CALL LINE(X,Y,M*N,1,0,0)
      CALL PLOT(0.0,0.0,-3)
      CALL SYMBOL(1.0,6.75,0.25,CTEXT,0.0,20)
      CALL SYMBOL(6.0,6.5,0.2,'1-D DATA FIELD',0.0,14)

C     ***** OUTPUT THE GRAPH *****
      CALL PLOT(0.0,0.0,999)
      WRITE (*,416)
      READ  (*,200) ANSWER
      IF  ((ANSWER .EQ. 'Y') .OR. (ANSWER .EQ. 'y')) GOTO 18

   19 WRITE(*,419)
      READ  (*,200) ANSWER
      IF  ((ANSWER .NE. 'Y') .AND. (ANSWER .NE. 'y')) GOTO 21

C     ***** ASK THE PARAMETERS FOR THE GRAPH *****
   20 WRITE (*,210)
      WRITE (*,*) '****  E N T E R  P L O T  P A R A M E T E R S  ****'
      WRITE (*,410)
      READ  (*,*)  AZIM
      WRITE (*,411)
      READ  (*,*)  ELEV
      WRITE (*,413)
      READ  (*,*)  ITRIM
      WRITE (*,414)
      READ  (*,*)  IDIV
      WRITE (*,415)
      READ  (*,199)  CTEXT
      WRITE (*,451)
      READ  (*,200)  ANSWER

      C      ***** INITIALIZE PLOT88 *****
             IF  ((ANSWER .EQ. 'Y') .OR. (ANSWER .EQ. 'y'))  THEN
                CALL PLOTS(0,0,2)
             ELSE
                CALL PLOTS(0,99,99)
             ENDIF

             CALL WINDOW(XLOL,YLOL,XUPR,YUPR)

      C      ***** DRAW THE MESH SURFACE OF THE GRAPH *****
             CALL MESHS(Z,26,26,N,M,AZIM,ELEV,0.5,0.5,8.25,6.5,IDIV,0,
            *           3,IPROJ,1,ZLOW,3,ITRIM,MASK,VERTEX)

      C      ***** ANNOTATION OF THE GRAPH *****
             CALL SYMBOL(1.0,6.75,0.25,CTEXT,0.0,20)
             CALL SYMBOL(6.0,6.5,0.2,'2-D DATA FIELD',0.0,14)
             CALL SYMBOL(5.5,0.3,0.2,'AZIMUTH:    ',0.0,10)
             CALL NUMBER(999.0,999.0,0.2,AZIM,0.0,2)
             CALL SYMBOL(5.5,0.0,0.2,'ELEVATION:',0.0,10)
             CALL NUMBER(999.0,999.0,0.2,ELEV,0.0,2)
             DY = (Z(1,1)/90.0) * ELEV
             CALL P3D2D(1.0,1.0,Z(1,1)-DY,XR,YR)
             CALL SYMBOL(XR,YR,0.25,'*',0.0,1)
             CALL SYMBOL(1.0,0.1,0.2,'* = ORIGIN',0.0,10)

      C      ***** OUTPUT THE GRAPH *****
             CALL PLOT(0.0,0.0,999)
             WRITE (*,416)
             READ  (*,200) ANSWER
             IF  ((ANSWER .EQ. 'Y') .OR. (ANSWER .EQ. 'y')) GOTO 20

        21   WRITE(*,418)
             READ(*,200)  ANSWER
             IF  ((ANSWER .EQ. 'Y') .OR. (ANSWER .EQ. 'y')) THEN
      C        **** FILL 0's THE TWO DIMENTIONAL GRID OF CONTROL POINTS ****
               DO 132 I = 1,26
                 DO 132 J = 1,26
                   ZF(I,J) =0.0
       132     CONTINUE

               ZFMAX = -9.9E20
               ZFMIN = 9.9E20
               DN = (N-1)/2.0
               DM = (M-1)/2.0
               P = 3.141592
               DO 133 MM = 1,M
                 DO 133 NN = 1,N
                   RLPART  = 0.0
                   IMGPART = 0.0
                   DO 134 L = 1,M
                     DO 134 K = 1,N
```

```
D Line# 1      7                            Microsoft FORTRAN77 V3.20 02/84
4                 R1(1)  =  COS(-2*P*(L-1)*(MM-DM-1)/M)
4                 R1(2)  =  SIN(-2*P*(L-1)*(MM-DM-1)/M)
4                 R2(1)  =  COS(-2*P*(K-1)*(NN-DN-1)/N)
4                 R2(2)  =  SIN(-2*P*(K-1)*(NN-DN-1)/N)
4                 RLPART   = RLPART   + Z(L,K)*(R1(1)*R2(1)
4        *                                      -R1(2)*R2(2))
4                 IMGPART = IMGPART + Z(L,K)*(R1(1)*R2(2)
4        *                                      +R1(2)*R2(1))
4      134    CONTINUE
2             ZF(MM,NN) = SQRT(RLPART**2 + IMGPART**2)
2             IF (ZF(MM,NN) .GT. ZFMAX)  ZFMAX = ZF(MM,NN)
2             IF (ZF(MM,NN) .LT. ZFMIN)  ZFMIN = ZF(MM,NN)
2      133  CONTINUE

     C      ***** OUTPUT THE ZF MATRIX *****
            WRITE (*,205) '*** FOURIER TRANSFORMATION  ',M,' X ',N,' ***'
            WRITE (*,212)
            DO 135 I = 1,M
1              WRITE (*,300) (ZF(I,J), J = 1,N)
1              WRITE (*,210)
1      135  CONTINUE
            WRITE (*,213)

            WRITE(*,419)
            READ  (*,200) ANSWER
            IF  ((ANSWER .NE. 'Y') .AND. (ANSWER .NE. 'y')) GOTO 22

     C      ***** ASK THE PARAMETERS FOR THE GRAPH *****
        30  WRITE (*,210)
            WRITE (*,*) '***  E N T E R   P L O T   P A R A M E T E R S   ***'
            WRITE (*,410)
            READ  (*,*)  AZIM
            WRITE (*,411)
            READ  (*,*)  ELEV
            WRITE (*,413)
            READ  (*,*)  ITRIM
            WRITE (*,414)
            READ  (*,*)  IDIV
            WRITE (*,415)
            READ  (*,199)  CTEXT
            WRITE (*,451)
            READ  (*,200)  ANSWER

     C      ***** INITIALIZE PLOT88 *****
            IF  ((ANSWER .EQ. 'Y') .OR. (ANSWER .EQ. 'y'))  THEN
              CALL PLOTS(0,0,2)
            ELSE
              CALL PLOTS(0,99,99)
            ENDIF

            WRITE (*,420)
```

```
D Line# 1      7                                   Microsoft FORTRAN77 V3.20 02/84
                READ (*,200) ANSWER

                CALL WINDOW(XLOL,YLOL,XUPR,YUPR)

                IF ((ANSWER .EQ. 'Y') .OR. (ANSWER .EQ. 'y')) THEN
                  DLEV = (ZFMAX-ZFMIN)/FLOAT(M)
                  CALL ZLEVEL(ZF,26,26,M,N,DLEV,ZLEV,N)
                  DO 136 I = 1,N
1                   LDIG(I) = 2
1                   LWGT(I) = 1
1         136     CONTINUE
                  CALL ZCNTUR(ZF,26,26,M,N,0.5,0.5,8.25,6.5,ZLEV,LDIG,LWGT,
             *                N,0.10,10)
                  CALL SYMBOL(5.5,0.0,0.2,'CONTOUR MAP',0.0,11)
                ELSE
       C          ***** DRAW THE MESH SURFACE OF THE GRAPH *****
                  CALL MESHS(ZF,26,26,M,N,AZIM,ELEV,0.5,0.5,8.25,6.5,IDIV,0,
             *               3,IPROJ,1,ZLOW,3,ITRIM,MASK,VERTEX)

       C          ***** ANNOTATION OF THE GRAPH *****
                  CALL SYMBOL(5.5,0.3,0.2,'AZIMUTH:   ',0.0,10)
                  CALL NUMBER(999.0,999.0,0.2,AZIM,0.0,2)
                  CALL SYMBOL(5.5,0.0,0.2,'ELEVATION:',0.0,10)
                  CALL NUMBER(999.0,999.0,0.2,ELEV,0.0,2)
                  DY = (ZF(1,1)/90.0) * ELEV
                  CALL P3D2D(1.0,1.0,ZF(1,1)-DY,XR,YR)
                  CALL SYMBOL(XR,YR,0.25,'*',0.0,1)
                  CALL SYMBOL(1.0,0.1,0.2,'* = ORIGIN',0.0,10)
                ENDIF
                CALL SYMBOL(1.0,6.75,0.25,CTEXT,0.0,20)
                CALL SYMBOL(6.0,6.5,0.2,'2-D DFT',0.0,7)

       C        ***** OUTPUT THE GRAPH *****
                CALL PLOT(0.0,0.0,999)
                WRITE (*,416)
                READ  (*,200) ANSWER
                IF  ((ANSWER .EQ. 'Y') .OR. (ANSWER .EQ. 'y')) GOTO 30
         22   ENDIF
              WRITE (*,417)
              READ  (*,200) ANSWER
              IF  ((ANSWER .EQ. 'Y') .OR. (ANSWER .EQ. 'y')) GOTO 2
              STOP

          199 FORMAT(A20)
          200 FORMAT(A)
          205 FORMAT(/,18X,A29,I2,A3,I2,A8,/)
          210 FORMAT()
          211 FORMAT(/,5X,60A)
          212 FORMAT(/,2X,'(AZIMUTH 320.0)',46X,'(AZIMUTH 230.0)',/)
          213 FORMAT(/,2X,'(AZIMUTH 050.0)',46X,'(AZIMUTH 140.0)',/)
          300 FORMAT(10(F7.2,1X))
```

```
D Line# 1      7                                    Microsoft FORTRAN77 V3.20 02/84
    400 FORMAT(9X,\)
    401 FORMAT(/,5X,'DO YOU WANT TO FILL THE Z MATRIX ? ',\)
    402 FORMAT(/,5X,'DO YOU WANT TO FILL THE TRANSITION MATRIX ? ',\)
    403 FORMAT(/,5X,'COLUMNS OF OUTPUT FRAME(N=1to25): ',\)
    404 FORMAT(/,5X,'ROWS OF OUTPUT FRAME(M=1to25): ',\)
    405 FORMAT(5X,A1,I2,A2,\)
    406 FORMAT(5X,A5,\)
    407 FORMAT(5X,A2,I2,A1,I2,A3,\)
    408 FORMAT(5X,A2,I3,A3,\)
    409 FORMAT(5X,A8,\)
    410 FORMAT(/,5X,'AZIMUTH(0.0 to 360.0 DEGREES): ',\)
    411 FORMAT(/,5X,'ELEVATION(90.0 to -90.0 DEGREES): ',\)
    412 FORMAT(/,5X,'NUMBER OF SMOOTHINGS: ',\)
    413 FORMAT(/,5X,'TRIM(0=NO,1=Xs,2=Ys): ',\)
    414 FORMAT(/,5X,'2,4 OR 8 SUBGRIDS: ',\)
    415 FORMAT(/,5X,'TITLE OF GRAPH(UP TO 20 CHAR): ',\)
    416 FORMAT(/,5X,'DO YOU WANT TO CHANGE PARAMETERS? ',\)
    417 FORMAT(/,5X,'DO YOU WANT TO REPEAT THE PROCESS? ',\)
    418 FORMAT(/,5X,'DO YOU WANT FOURIER TRANSFORMATION ? ',\)
    419 FORMAT(/,5X,'DO YOU WANT TO MAKE GRAPH ? ',\)
    420 FORMAT(/,5X,'DO YOU WANT CONTOUR MAP ? ',\)
    421 FORMAT(/,5X,'DO YOU WANT TO DRAW CARVE ? ',\)
    451 FORMAT(/,5X,'SEND GRAPH TO THE PRINTER(Y or N): ',\)
        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                                          Microsoft FORTRAN77 V3.20 02/84
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
```

```
                                                                          Page  11
                                                                          09-27-85
                                                                          17:23:37
D Line# 1      7                              Microsoft FORTRAN77 V3.20 02/84
 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. Electron 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

[illegible]

215619

Thesis
T3637
c.1

Theofilou
2-D signal generation using state-space formulations.