1979

RECURSIVE DIGITAL FILTERS: DESIGN AND APPLICATIONS TO IMAGE PROCESSING.

APPANNA T. CHOTTERA
University of Windsor

RECURSIVE DIGITAL FILTERS: DESIGN AND APPLICATIONS TO IMAGE PROCESSING

by

© Appanna T. Chottera

A Dissertation
submitted to the Faculty of Graduate Studies
through the Department of Electrical Engineering
in partial fulfillment of the requirements
for the degree of Doctor of Philosophy at the
University of Windsor

Windsor, Ontario, Canada
1979

ABSTRACT

The phase characteristics of a digital filter is of importance in many applications; specifically, in image processing problems. Part of the work reported in this thesis is devoted to the design of recursive digital filters in which a simultaneous approximation of the desired magnitude and linear phase is performed, using linear programming. The use of linear programming is facilitated via the linearization of the inherently non-linear approximation problem. Using this approach, both one and two dimensional quarter plane recursive digital filters have been designed, the examples which are provided with the design algorithm. Linear stability constraints are proposed, which can easily be incorporated into the linear programming design procedure, to enable the design of stable filters. These stability constraints are sufficient conditions for stability, and therefore allow the design of a subclass of one and two dimensional quarter plane recursive digital filters.

Considering the computational advantage of recursive digital filters, compared to most convolutional methods of filtering (specifically, convolution filtering via the Fast Fourier Transform), the second part of this thesis examines applications of quarter plane two dimensional recursive digital filters in image processing. The thesis considers applications in the areas of image enhancement and restoration problems. The problems considered in image enhancement are: high frequency emphasis and edge enhancement. The

problems of image restoration are considered for the cases of motion, focus and atmospheric turbulance blurs.

ACKNOWLEDGEMENTS

I would like to express my sincere appreciation to my supervisor, Dr. G.A. Jullien for his invaluable advice, help and constructive criticisms during the course of this research. The valuable advice and comments of Dr. W.C. Miller and Dr. J.J. Soltis, are gratefully acknowledged. In addition, the help of many of the graduate students and the technicians, Mr. Novasad and Mr. Reiter, of the Electrical Engineering Department is sincerely appreciated.

To my parents, I extend my sincerest thanks. Without their help and love, though far away, this work would not have started.

Thanks are also due to Mrs. Barbara L. Denomey for her excellent typing of this thesis.

TABLE OF CONTENTS

		Page
ABSTRACT		i
ACKNOWLEDGEMENTS		iii
TABLE OF CONTENTS		iv
LIST OF ILLUSTRATIONS		vi
LIST OF APPENDICES		xi
CHAPTER I	INTRODUCTION	1
1.1	ONE AND TWO DIMENSIONAL SIGNAL PROCESSING	1
1.2	RECURSIVE DIGITAL FILTER DESIGN AND APPLICATIONS	3
1.3	PROBLEM STATEMENT	7
1.4	THESIS ORGANIZATION	8
CHAPTER II	RECURSIVE DIGITAL FILTER DESIGN USING LINEAR PROGRAMMING	9
2.1	INTRODUCTION	9
2.2	LINEAR PROGRAMMING	9
	2.2.a Linear Programming Theory	10
	2.2.b Dual Linear Programming	12
2.3	ONE DIMENSIONAL RECURSIVE FILTER DESIGN	14
	2.3.a Theory of Approximation	14
	2.3.b Stability Constraints in One Dimension	21
	2.3.c Design Procedure	25
	2.3.d Computational Considerations	30
	2.3.e Examples of Design	34
2.4	TWO DIMENSIONAL RECURSIVE FILTER DESIGN	48
	2.4.a Theory of Approximation	49
	2.4.b Stability Constraints in Two Dimensions	51
	2.4.c Design Procedure	52
	2.4.d Computational Considerations	56
	2.4.e Design Examples	61
2.5	SUMMARY	70

		Page
CHAPTER III	RECURSIVE DIGITAL FILTER APPLICATIONS...	71
3.1	INTRODUCTION.............................	71
3.2	APPLICATION OF RECURSIVE DIGITAL FILTERS IN IMAGE ENHANCEMENT............	73
3.2.a	High Frequency Emphasis or Crispening.......................	74
3.2.b	Edge Enhancement.................	78
3.3	APPLICATION OF RECURSIVE DIGITAL FILTERS IN IMAGE RESTORATION............	85
3.3.a	Restoration Filter Transfer Functions........................	87
3.3.b	Uniform Motion Blur...............	91
3.3.c	Focus Blur.......................	102
3.3.d	Atmospheric Turbulance Blur......	117
3.4	SUMMARY..................................	129
CHAPTER IV	DISCUSSION OF RESULTS AND EXTENSIONS....	131
4.1	INTRODUCTION.............................	131
4.2	DISCUSSION OF THE FILTER DESIGN METHOD..	131
4.3	DISCUSSION ON THE APPLICATION OF RECURSIVE DIGITAL FILTERS TO IMAGE PROCESSING............................	133
4.3.a	Image Enhancement Applications...	133
4.3.b	Image Restoration Applications...	134
4.4	EXTENSIONS...............................	137
4.4.a	On Two Dimensional Filter Design..........................	137
4.4.b	On Recursive Digital Filter Applications in Image Processing.	138
CHAPTER V	CONCLUSIONS..............................	139
REFERENCES	....	141
APPENDICES	....	145
VITA AUCTORIS	....	276

LIST OF ILLUSTRATIONS

Figure		Page
2.1	Graphical Explanation of the Linear Program	11
2.2	Constraint Regions for Poles of First Order Transfer Function	23
2.3	Range of τ_s in the Impulse Response	27
2.4	Desired Magnitude Characteristics	35
2.5	Designed Magnitude Characteristics	36
2.6	Designed Group Delay Characteristics	37
2.7	Designed Impulse Response	38
2.8	Pole-Zero Positions of the Designed Filter	39
2.9	Realization of Differentiator with Half Sample Delay	41
2.10	Desired Magnitude Characteristics	42
2.11	Designed Filter Magnitude Characteristics	43
2.12	Designed Filter Group Delay Characteristics	44
2.13	Error in the Magnitude	45
2.14	Impulse Response of the Designed Filter	46
2.15	Pole-Zero Positions of the Designed Filter	47
2.16 (a)	Desired Magnitude Specifications	63
(b)	Designed Magnitude Response	63
(c) & (d)	Designed Group Delays τ_1 and τ_2	63
2.17 (a)	First 16x16 Pixels of the Impulse Response	64
(b)	Designed Magnitude Response	64
2.18 (a)	Desired Magnitude Specifications	65
(b)	Designed Magnitude Response	65
(c) & (d)	Designed Group Delays τ_1 and τ_2	65
2.19 (a)	First 16x16 Pixels of the Impulse Response	66
(b)	Coefficients of the Designed Filter	66
2.20 (a)	Desired Magnitude Specification	68
(b)	Designed Magnitude Response	68
(c) & (d)	Designed Group Delays τ_1 and τ_2	68

Figure		Page
2.21(a)	First 16x16 Pixels of Impulse Response	69
(b)	Coefficients of the Designed Filter	69
3.1 (a)	Original Image	75
(b)	Enhanced Image	75
3.2 (a)	Designed High Pass Filter Magnitude Response	76
(b)&(c)	Designed Group Delay Responses τ_1 and τ_2	76
(d)	First 8x8 Pixels of the Impulse Response	76
(e)	Coefficients of the Designed Filter	77
3.3 (a)	Desired Magnitude Specifications	80
(b)	Designed Magnitude Response	80
(c)&(d)	Designed Group Delays τ_1 and τ_2	80
(e)	First 8x8 Pixels of the Designed Filter Impulse Response	81
(f)	Coefficients of the Designed Filter	81
3.4 (a)	Original Image	82
(b)	Edge Enhanced Image with Linear Phase Filter	82
(c)	Edge Enhanced Image with Non-Linear Phase Filter	82
3.5 (a)	Magnitude Response	83
(b)&(c)	Group Delays τ_1 and τ_2	83
(d)	Impulse Response	83
(e)	Coefficients of the Filter	84
3.6 (a)	Motion Blur Spatial Response	92
(b)	Motion Blur Transfer Function	92
(c)	Amount of Phase Shift Introduced by (b)	92
3.7 (a)	Blurring Transfer Function. Blur Length Equals 4 Pixels; Angle=45^o w.r.t. Horizontal	94
(b)	Original Image	94
(c)	Image Blurred by (a), With Noise, SNR=30 db	94
3.8 (a)	Restoration Filter Response	96
(b)	Restoration Filter Response Along $\theta=45^o$	96
(c)	Phase Shift Introduced by (b)	96
3.9 (a)	Recursive Filter Implementation for Motion Deblur	98
(b)	Frequency Response of (a)	98
3.10(a)	Impulse Response of Desired Restoration Filter	100
(b)	Impulse Response of Recursive Filter Implementation	100
(c)	Coefficients of the Recursive Filter $H_1(Z_1,Z_2)$	100

Figure		Page
3.11(a)	Original Image	101
(b)	Blurred Image with Noise, Blur Length=4, Angle=45^{o}, SNR=30 db	101
(c)	Image Restored Using FFT	101
(d)	Image Restored Using Recursive Implementation of Figure 3.9(a)	101
3.12(a)	Motion Blurring Transfer Function; Blur Length=8 Pixels, Angle=22.5^{o} w.r.t. Horizontal	103
(b)	PSEC Restoration Filter Corresponding to (a) and SNR=30 db	103
(c)	Recursive Filter Approximation to (b)	103
3.13(a)	Impulse Response of the Desired Restoration Filter	104
(b)	Impulse Response of Recursive Filter Implementation	104
3.14	Coefficients of the Recursive Filter $H_1(Z_1,Z_2)$	105
3.15(a)	Original Image	106
(b)	Motion Blurred Image With Noise	106
(c)	Restoration Using FFT	106
(d)	Restoration Using Recursive Filter Implementation of Figure 3.9(a)	106
3.16(a)	Focus Blur Point Spread Function	108
(b)	Transfer Function of Focus Blur	108
(c)	Phase Shift Introduced by (b)	108
3.17(a)	Focus Blurring Transfer Function Blur Radius=4 Pixels	110
(b)	Original Image	110
(c)	Focus Blurred Noisy Image; SNR=30 db	110
3.18(a)	Restoration Filter Response	111
(b)	Restoration Filter Response Along 45^{o} Radial Direction	111
(c)	Phase Shift Introduced by (b)	111
3.19(a)	Recursive Filter Implementation for Focus Deblur	113
(b)	Frequency Response of (a)	113
3.20	Coefficients of Two Dimensional Filter $H_1(Z_1,Z_2)$	114
3.21	Coefficients of Two Dimensional Filter $H_2(Z_1,Z_2)$	115
3.22(a)	Restoration by Convolution Via FFT	116
(b)	Restoration Using Recursive Filter Implementation of Figure 3.19(a)	116
3.23(a)	Atmospheric Turbulance Blur Transfer Function	119

Figure		Page
3.23(b)	One Dimensional Plot of the Transfer Function Along any Radial Direction.....	119
3.24(a)	Restoration Filter Response for Atmospheric Blur..........................	121
(b)	Plot of Magnitude of (a) Along 45° Radial Direction.......................	121
3.25(a)	Magnitude Response of the Designed Filter.................................	122
(b)	Coefficients of the Designed Filter.....	122
3.26(a)	Impulse Response Corresponding to Desired Magnitude Specification.........	123
(b)	Impulse Response of the Designed Filter...................................	123
3.27(a)	Original Image...........................	124
(b)	Blurred Noisy Image.......................	124
(c)	Restoration Using FFT....................	124
(d)	Restoration Using Recursive Filter......	124
3.28(a)	Recursive Filter Implementation for Atmospheric Turbulance Deblur...........	125
(b)	Desired Restoration Filter Frequency Response......................	125
(c)	Frequency Response Realized by (a)......	125
3.29(a)	Impulse Response Corresponding to Figure 3.28(b)..........................	126
(b)	Impulse Response Corresponding to Figure 3.28(c)..........................	126
(c)	Coefficients of $H_1(Z_1,Z_2)$..............	127
3.30(a)	Original Image...........................	128
(b)	Atmospheric Turbulance Blurred Image With Noise................................	128
(c)	Restoration Using FFT....................	128
(d)	Implementation of Figure 3.28(a)........	128
A1	An Example of Input and Output Masks to Calculate y(3,3) for Equation (A.1)..	149
A2	An Example of Input and Output Masks to Calculate y(2,2) for Equation (A.8)..	151
A3	Direction of Delays in Spatial Domain...	159
B1	Schematic of Image Formation System.....	165
B2	Block Diagram Representation of Image Formation and Recording.................	167
C1	Figure C1................................	172
D1	Solution Region Corresponding to Various Types of Constraints............	175
E1	Figure E1................................	180
G1	Block Diagram of Image Processing Facility.................................	191

Figure		Page
G2	Flying Spot Scanner System..............	192
G3	Timing and Control Circuit and Integrator Set-Up.......................	195
G4	Timing and Control Circuit..............	197
G5	Timing Diagram..........................	198
G6	Photomultiplier Tube Output Amplifier...	200
G7	Image Display and Reproduction Hardware................................	202
H1	Sampling of Image in a Rectangular Array...................................	238

Table		
4.1	Computations for Recursive Filtering....	136

LIST OF APPENDICES

		Page
APPENDIX A	Digital Filtering Fundamentals..........	145
APPENDIX B	Digital Image Processing Fundamentals...	163
APPENDIX C	Proof of Stability Criterion for One Dimensional Filter Design...............	169
APPENDIX D	On The Stability Criterion for Second Order One Dimensional Filter.....	173
APPENDIX E	Proof of the Sufficiency of the Two Dimensional Stability Constraint........	178
APPENDIX F	Two Dimensional Recursive Digital Quarter Plane Filter Design in Cascade Form...........................	181
APPENDIX G	Image Processing Hardware...............	189
APPENDIX H	Filter Design, Image Filtering and Other Additional Image Processing Computer Programs.......................	204

CHAPTER I

INTRODUCTION

1.1 One and Two Dimensional Signal Processing

In recent years there has been a rapid growth of interest in computer processing of both one and two dimensional digital signals. Examples of one dimensional signals of interest are speech, ECG (Electro Cardiogram) and EEG (Electro Encephalogram). Two dimensional signals of interest include photographic data, such as medical x-rays, aerial photographic data used for geographic purposes and non-pictorial data such as the data corresponding to seismic signals obtained in the exploration for oil and gas.

The purpose of processing such signals is manifold. In the case of speech signals, for example, it may be required to extract information corresponding to the identity of a speaker, or in the case of ECG it may be data compression, where the purpose is to reduce storage requirement to a minimum. In the two dimensional case examples are the enhancement of satellite pictures for improving image quality, or, in the case of ERTS (Earth Resource Technology Satellite) pictures, the purpose may be to obtain a classification of earth's resources.

In many cases the processing of such signals by analog techniques is unattractive. In the one dimensional signal processing, for example, consider an analog filter which requires twenty poles to accomplish a design objective. In this case even if the filter is realized actively with isolation between stages, it probably will be almost impossible to tune. In the two dimensional case an example of analog processing is the processing of photographic data by optical techniques. Although optical processing is a completely parallel process (and hence fast), digital processing is much more flexible and does not require the set up procedures normally associated with, for example, optical filtering. Optical filtering is normally restricted to handling only linear filtering problems. In an all digital environment, it is possible to carry out iterative processes and processes requiring tests and decisions as well as normal linear, and non-linear filtering algorithms. To carry out these processes one may employ a general purpose digital computer, along with some dedicated hardware to perform specific tasks.

One of the most common signal processing operations which can be performed digitally is linear filtering. As an example, the edge enhancement of a picture can be carried out by two dimensional high pass filtering. The need for such types of processing and the decreasing cost of performing these processes digitally has given rise to considerable research interest in the area of digital filter design and their applications. Digital filters are of two types, namely; a) Finite impulse response (FIR) and b) Infinite impulse response or Recursive. As a part of this research interest, this dissertation presents a method for the design of, a class of digital filters called the recursive (infinite impulse response) digital filters in one and two dimensions and examines their applications in certain image processing problems. A complete description of the types of digital filters and various other terminologies used in digital filtering and digital image processing is given in Appendices A and B.

1.2 Recursive Digital Filter Design and Applications

In recent years considerable work in the area of recursive digital filter design has been and is still being carried out, primarily because recursive digital filters offer greater speed of filtering, smaller memory requirements and easier implementations compared to FIR filters. Many of the one dimensional recursive digital filter design techniques can be found in [1] and many of the recent two dimensional design techniques are presented in [2,3,4,5,6, 7,8,9,10,11,12]. These techniques consider only the frequency domain approximation of given arbitrary specifications. The techniques of spatial domain approximations are not considered in this thesis because these design procedures do not incorporate constraints on filter coefficients, which are required for a stable design [13]. Due to the huge amount of literature in this area, a complete survey of frequency domain design techniques is not attempted here. However, a brief discussion of the existing design techniques with some comparisons are given in the following paragraphs.

Some of the one dimensional techniques given in [1] and the two dimensional design techniques of [10,11,12] are analytical in nature, i.e., the desired type of filters are obtained by using certain types of transformations (e.g., impulse invariance, bilinear, etc.) on a prototype filter. This, however, provides very little control over the response of the designed filter. Also, if the desired specifications are arbitrary, which is true in many applications such as speech and image processing, then in these situations these methods of design are of very little use. The remaining design techniques reported in [1,2,3,4,5,6,7,8,9] employ either linear or non-linear optimization procedures in designing stable recursive digital filters in which approximations can be carried out to arbitrary specifications. Many of these design techniques design filters to approximate only arbitrary magnitude characteristics; however, this is an incomplete specification of the filter since it is indicated in [1] and also shown by Huang [14], that phase characteristics should be given as much importance as magnitude characteristics. Also in many applications linear phase is important, where dispersion due to non-linear phase characteristics is harmful, specifically in image processing problems. In the case of FIR filters, linear phase filters are easily designed using the design procedures of [14,15], which use linear programming [16]. In the case of existing recursive digital filter design techniques, the linear phase is realized via group delay equalization [1,3], where a non-linear optimization procedure is employed for the approximation. The overall design procedure of [1,3] involves two steps: a) the approximation of magnitude followed by b) group delay equalization which compensates for the non-linearities in the group delay characteristics of the magnitude only filter. However, there are two drawbacks to using the non-linear optimization approach: a) it requires the specification of initial values for parameters of optimization, b) the optimum obtained in the approximation procedure is a local one rather than the global optimum. In addition to this, as indicated in [1, pp. 288-291], in situations where the group delay characteristics of a magnitude only design are highly non-linear, the equalization is impractical. Furthermore, in the two dimensional case, finding a good set of initial values for the parameters of optimization is much more difficult than in the one dimensional case. However, the non-linear optimization techniques of design given in [1,2,3,4,5,6,7,9] are still useful in situations where other types of approximations (e.g., approximating to the real part of the transfer function or a magnitude squared transfer function approximation) are involved, specifically in the two dimensional case.

In comparison, the use of a linear optimization approach, such as linear programming, not only overcomes the drawbacks of the non-linear optimization procedures, but also it is possible to approximate simultaneously linear phase and arbitrary magnitude characteristics. A characteristic of linear programming that is worth noting is that, if a solution to the problem exists, then it is unique and the optimum obtained is the absolute optimum consistent with the constraints of the problem. As indicated earlier, linear programming has been extensively used in designing FIR filters where both linear phase and magnitude approximations are carried out simultaneously. Due to its success in the case of FIR filters, linear programming has also been suggested [23] for recursive filter design where simultaneous approximations of linear phase and magnitude may be carried. However, the linear programming methods of designs presented so far [8,17,18,19,20,21] approximate either magnitude, magnitude squared or phase characteristics only. In this dissertation, a method is presented to design both one and two dimensional (quarter plane) recursive digital filters to simultaneously approximate both magnitude and linear phase characteristics, using linear programming. A preliminary investigation of this method by the author is reported in [22].

Another aspect of the research reported in this thesis deals with the applications in image processing area. In recent years, the use of FIR filters has become very common [23,24,25,26,27], specifically with the advent of the fast fourier transform (FFT) [28]. This has not been the case, however, for recursive filters; their uses have been few [29,30], because of difficulty encountered at the early attempts at designing stable filters. As indicated earlier, recent design techniques, including the method to be presented in this thesis, are able to design stable filters and at the same time able to approximate desired frequency domain characteristics. Considering the above, coupled with the computational advantage of the recursive filters compared to FIR filters [31], this dissertation sets out to investigate the use of quarter plane recursive digital filters in image processing. Some preliminary work carried out by the author, in the applications of recursive digital filters to image processing can be found in [32].

It should be noted here, that at the time this research was carried out, the theory and design of half plane filters were just being proposed and therefore neither the design nor the applications of half plane filters are considered in this thesis.

1.3 Problem Statement

Given an arbitrary magnitude characteristic, the problem of finding coefficients of the recursive digital transfer function using the linear programming approach, where the filter transfer function simultaneously approximates magnitude and linear phase characteristics, is considered in this thesis. The linear programming approach is first applied to the design of one dimensional filters, and then extended to the design of two dimensional quarter plane filters. Since the approximation procedure is based on linear programming, the constraints that are to be used for stable filter design are required to be linear. Therefore the design approach presented also considers various linear stability constraints that can be incorporated in the design technique for stable filters.

Given a blurred image, the thesis considers the inverse filtering problem using recursive filter implementation for restoration from motion, focus and atmospheric turbulance blurs. The thesis also considers the application of recursive digital filters to image enhancement applications where the problem involves the high frequency emphasis and enhancement of the edges of a given image.

1.4 Thesis Organization

In Chapter II, techniques are presented for the design of both one and two dimensional recursive digital filters. The techniques make use of linear programming for the simultaneous approximation of both magnitude and linear phase characteristics.

Chapter III presents the applications of two dimensional recursive digital filters to image processing problems. The problems considered in the applications area are restricted to image enhancement and restoration.

In Chapter IV, the discussions of the design method and results of applications of recursive digital filters to image processing are presented. In addition, some extensions to the research work presented in this thesis are also discussed.

Finally, Chapter V presents the conclusions that can be obtained from the research work presented in this thesis.

CHAPTER II

RECURSIVE DIGITAL FILTER DESIGN USING LINEAR PROGRAMMING

2.1 Introduction

Linear programming has been widely used in the design of both analog and digital filters. The design of analog filters, using linear programming, was considered by Mathew's et al [33]. Later on, the linear prorramming approach was used by Rabiner and Hu [25,26] in the design of FIR digital filters. Their design approach included the approximation of both magnitude and linear phase specifications and the designs were carried out in both one and two dimensions. Following this, the linear programming approach was used in the design of one dimensional [17,18,19,20,21] and two dimensional [8] recursive digital filters. However, these techniques designed filters that approximated magnitude or magnitude squared specifications [17,18,19,20] or phase only specification [21] using all pass filters and therefore these methods are of very little use in situations where both magnitude linear phase approximations are required.

In this chapter, a linear programming method for designing recursive digital filters is presented, where a simultaneous approximation of linear phase and arbitrary magnitude response is performed. For completeness, a discussion on the general linear programming technique is presented below.

2.2 Linear Programming [16]

This section briefly introduces the theory of linear programming so as to facilitate the understanding of the linear programming design technique. This includes the concept of dual linear programming, which in many cases becomes useful in reducing the number of varibles in the linear programming problem.

2.2.a Linear Programming Theory

A linear programming problem can be mathematically stated in the following form - find a vector $(w_1, w_2, w_3, \ldots w_M)$, subject to the constraints:

$$\begin{bmatrix} w_1, w_2, \ldots w_M \end{bmatrix} \cdot \begin{bmatrix} a_{11} & a_{12} & a_{13} \cdots & a_{1N} \\ a_{21} & a_{22} & a_{23} \cdots & a_{2N} \\ \cdot & & & \\ \cdot & & & \\ \cdot & & & \\ a_{M1} & \cdots\cdots & \cdots\cdots & a_{MN} \end{bmatrix} \leq \begin{bmatrix} c_1, c_2 \ldots c_N \end{bmatrix}; \quad (M<N) \tag{2.2.1}$$

such that,

$$g = \begin{bmatrix} w_1, w_2 \ldots, w_M \end{bmatrix} \cdot \begin{bmatrix} b_1 \\ b_2 \\ \vdots \\ b_M \end{bmatrix} \tag{2.2.2}$$

is maximized.

Here, the variables $(w_1, w_2, \ldots w_M)$ may be constrained; for example $w_i \geq 0$, $i=1,2,\ldots M$.

A characteristic of the linear program is that given there is a solution, it is guaranteed to be a unique solution and there are well defined procedures for arriving at this solution within (M+N) iterations. The procedures also determine if the solution is constrained or unconstrained.

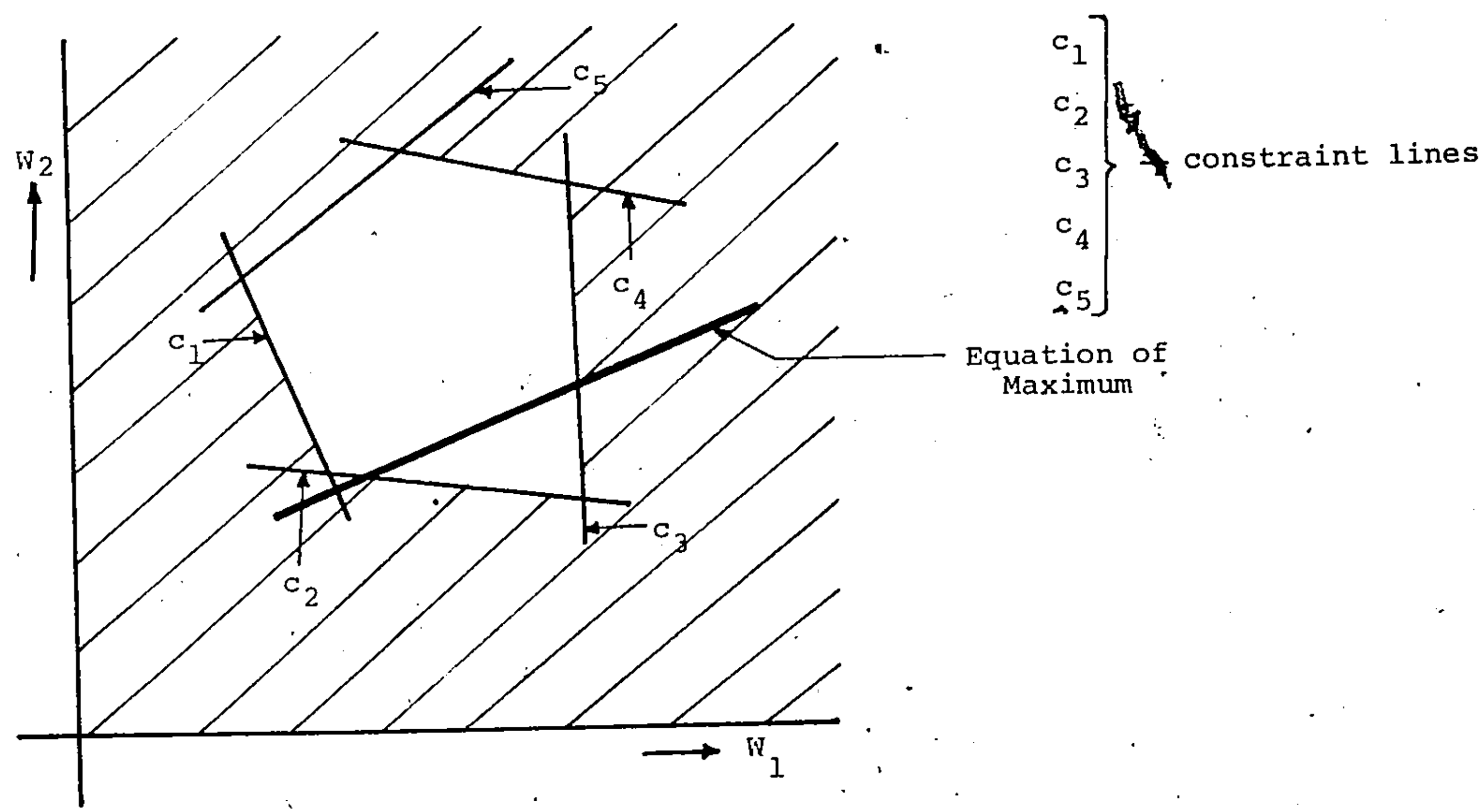


Figure 2.1 Graphical Explanation of the Linear Program.

Figure 2.1 shows the graphical interpretation of the linear program with two variables. Each constraint line (c_1-c_5) is a linear inequality in two variables, w_1, and w_2. Therefore a straight line can be drawn representing the linear equality, and part of the solution space (shown by shaded lines) is eliminated as a region that does not belong to the possible solution. When all the constraint lines have been drawn, only a small region as shown in Figure 2.1, is admissible as the solution region in which to find the maximum. It should be noted that an important property of the linear programming problem is that if there is to be a solution, the constraint equations should form a polyhedron, and that the maximum or the minimum value of the desired linear function occurs at an extreme point of the polyhedron. Thus the procedure is to compute the value of the objective function at each of the extreme points and choose an extreme point as the solution for which the objective function value is a maximum. The maximum, thus obtained, is the absolute maximum consistent with the constraints in the linear programming problem.

2.2.b Dual Linear Programming

The linear programming problem described above can be considered as the 'Primal Problem' of linear programming. Rewriting (2.2.2) and (2.2.1) in the matrix notation, we have:

Maximize

$$g = w^T b \tag{2.2.3}$$

Subject to constraints

$$w^T A \leqslant c^T \tag{2.2.4}$$ ‡

where,

$$w = \begin{bmatrix} w_1 \\ w_2 \\ \cdot \\ \cdot \\ \cdot \\ w_M \end{bmatrix}, \quad c = \begin{bmatrix} c_1 \\ c_2 \\ \cdot \\ \cdot \\ \cdot \\ c_N \end{bmatrix} \text{ and } b = \begin{bmatrix} b_1 \\ b_2 \\ \cdot \\ \cdot \\ \cdot \\ b_M \end{bmatrix}$$

Depending on whether w is constrained or unconstrained, the primal problem given by (2.2.3) and (2.2.4) can assume either of the two types of dual problems', indicated below:

If w is constrained, i.e.,

$$w \geqslant 0 \tag{2.2.5}$$

then the dual problem is a 'symmetrical dual linear programming problem' and is stated as:

Minimize

$$f = c^T x \tag{2.2.6}$$

Subject to

$$Ax \geqslant b \tag{2.2.7}$$

and

$$x \geqslant 0 \tag{2.2.8}$$

However, if w is unconstrained (i.e., it can assume both positive and negative values), then the dual problem is an 'unsymmetrical dual linear programming' problem, and is stated as:

‡ c^T refers to transpose of matrix vector c

Minimize

$$f = c^T x \tag{2.2.9}$$

Subject to

$$Ax = b \tag{2.2.10}$$

and

$$x \geq 0 \tag{2.2.11}$$

It should be noted that the solution of the primal and dual problem can be obtained in the solution of either of the problems because maximum g = minimum f.

The problem of interest here is the unsymmetrical primal-dual problem, since the design of digital filters can be formulated as a linear programming program given by (2.2.3) and (2.2.4), having no constraints on the variable vector w.

2.3 One Dimensional Recursive Filter Design

In this section, a frequency domain approximation procedure for one dimensional recursive digital filters, using linear programming, is presented. Suitable stability constraints on filter coefficients are indicated for stable designs and an algorithm for designing linear phase filters is described. Details of computations and design examples are also included.

2.3.a Theory of Approximation

Let H(Z) be the transfer function of a recursive digital filter. Assume H(Z) has the form,

$$H(Z) = \frac{P(Z)}{Q(Z)} = \frac{a_0 + a_1 Z + a_2 Z^2 + \ldots + a_N Z^N}{b_0 + b_1 Z + b_2 Z^2 + \ldots + b_M Z^M} \tag{2.3.1}$$

where $Z = e^{-j\Omega}$ and Ω is the normalized frequency variable. The degree of numerator is less than or equal to that of the denominator, i.e., $N \leq M$. The term b_0 can be set equal to 1.0 without any loss of generality.

Now, given arbitrary magnitude and phase specifications, it is desired to formulate the digital filter transfer function approximation problem into a linear programming problem such that the constraints of the linear program are in terms of the filter coefficients. This can be carried out as follows.

Let $R(\Omega_i)$ and $\phi(\Omega_i)$ be the given magnitude and phase specifications, respectively, specified at a discrete set of frequency points Ω_i, $i = 1,2,\ldots,L$. The real component $Y(\Omega_i)$ and the imaginary component $Y'(\Omega_i)$ of the frequency domain specifications can then be written as:

$$Y(\Omega_i) = R(\Omega_i) \cdot \left[\cos \phi(\Omega_i)\right] \tag{2.3.2}$$

$$Y'(\Omega_i) = R(\Omega_i) \cdot \left[\sin \phi(\Omega_i)\right] \tag{2.3.3}$$

Now, the problem of approximating the characteristics of a recursive digital filter to (2.3.2) and (2.3.3) can be formulated into a linear programming problem by following the procedure of Matthews et al [16].

Define $r(\Omega_i)$ as,

$$r(\Omega_i) = Y(\Omega_i) + j\, Y'(\Omega_i) - \frac{P(e^{-j\Omega_i})}{Q(e^{-j\Omega_i})} ;$$

$$\text{for } i = 1,2,\ldots,L \tag{2.3.4}$$

which is a complex error between the desired characteristics and the approximating filter. Multiplying (2.3.4) on both sides by $Q(e^{-j\Omega_i})$, a weighted error function (which is linear in terms of filter coefficients) is obtained, and is given by:

$$r(\Omega_i)Q(e^{-j\Omega_i}) = Y(\Omega_i)Q(e^{-j\Omega_i}) + j\ Y'(\Omega_i)Q(e^{-j\Omega_i})$$

$$- P(e^{-j\Omega_i}) \text{ for } i=1,2,\ldots,L \tag{2.3.5}$$

The quantities in the expression (2.3.5) can be separated into their real and imaginary components as:

$$r(\Omega_i)Q(e^{-j\Omega_i}) = e(\Omega_i) + j\ e'(\Omega_i) \tag{2.3.6}$$

$$P(e^{-j\Omega_i}) = P_R(\Omega_i) - j\ P_I(\Omega_i) \tag{2.3.7}$$

$$Q(e^{-j\Omega_i}) = Q_R(\Omega_i) - j\ Q_I(\Omega_i) \tag{2.3.8}$$

Substitution of (2.3.6), (2.3.7) and (2.3.8) into (2.3.5) results in the following:

$$e(\Omega_i) + j\ e'(\Omega_i) = Y(\Omega_i)\cdot\left[Q_R(\Omega_i) - j\ Q_I(\Omega_i)\right] + j\ Y'(\Omega_i)\cdot\left[Q_R(\Omega_i)\right.$$

$$\left. - j\ Q_I(\Omega_i)\right] - P_R(\Omega_i) + j\ P_I(\Omega_i) \tag{2.3.9}$$

Equating the real and imaginary parts in expression of (2.3.9) results in:

$$e(\Omega_i) = Y(\Omega_i)Q_R(\Omega_i) + Y'(\Omega_i)Q_I(\Omega_i) - P_R(\Omega_i) \tag{2.3.10}$$

$$e'(\Omega_i) = Y'(\Omega_i)Q_R(\Omega_i) - Y(\Omega_i)Q_I(\Omega_i) + P_I(\Omega_i) \tag{2.3.11}$$

where $e(\Omega_i)$ and $e'(\Omega_i)$ are linear in terms of filter coefficients.

It is now possible to put Equations (2.3.10) and (2.3.11) into a linear programming problem such that the weighted error in (2.3.6) is minimized.

Choose a quantity ε such that,

$$|e(\Omega_i)| \leqslant \varepsilon \tag{2.3.12}$$

and

$$|e'(\Omega_i)| \leqslant \varepsilon, \text{ for } i=1,2,\ldots L \tag{2.3.13}$$

Expressions (2.3.12) and (2.3.13) can be re-written as,

$$-\varepsilon \leqslant e(\Omega_i) \leqslant \varepsilon \tag{2.3.14}$$

and,

$$-\varepsilon \leqslant e'(\Omega_i) \leqslant \varepsilon \tag{2.3.15}$$

From the above, one can write four inequalities as follows:

$$\begin{aligned} e(\Omega_i) - \varepsilon &\leqslant 0 \\ e'(\Omega_i) - \varepsilon &\leqslant 0 \\ -e(\Omega_i) - \varepsilon &\leqslant 0 \\ -e'(\Omega_i) - \varepsilon &\leqslant 0 \end{aligned} \tag{2.3.16}$$

Let $\xi = -\varepsilon$, in which case the inequalities are as follows:

$$\begin{aligned} e(\Omega_i) + \xi &\leqslant 0 \\ e'(\Omega_i) + \xi &\leqslant 0 \\ -e(\Omega_i) + \xi &\leqslant 0 \\ -e'(\Omega_i) + \xi &\leqslant 0 \end{aligned} \tag{2.3.17}$$

After substitution of $e(\Omega_i)$ and $e'(\Omega_i)$ from (2.3.10) and (2.3.11) into (2.3.17), the linear programming problem for filter design can be stated in a form similar to (2.2.1) and (2.2.2), as,

Maximize $\qquad g = \xi$

Subject to,

$$Y(\Omega_i)Q_R(\Omega_i) + Y'(\Omega_i)Q_I(\Omega_i) - P_R(\Omega_i) + \xi \leq 0 \qquad (2.3.18)$$

$$Y'(\Omega_i)Q_R(\Omega_i) - Y(\Omega_i)Q_I(\Omega_i) + P_I(\Omega_i) + \xi \leq 0 \qquad (2.3.19)$$

$$-Y(\Omega_i)Q_R(\Omega_i) - Y'(\Omega_i)Q_I(\Omega_i) + P_R(\Omega_i) + \xi \leq 0 \qquad (2.3.20)$$

$$-Y'(\Omega_i)Q_R(\Omega_i) + Y(\Omega_i)Q_I(\Omega_i) - P_I(\Omega_i) + \xi \leq 0 \qquad (2.3.21)$$

Maximizing ξ, minimizes the weighted errors in (2.3.10) and (2.3.11).

It can be clearly seen from (2.3.17) that in the ideal situation, the maximum value of ξ can be equal to zero. Also, ξ cannot be positive, and if it does become positive, the solution is meaningless. Thus the upperbound on ξ is zero. A careful examination of the approximation procedure reveals that the minimization of the weighted errors in the error set,

$$\left[e(\Omega_i),\ e'(\Omega_i)\right] \quad ; \ i=1,2,\ldots L$$

is performed in the mini-max sense. Therefore the absolute value of ξ is equal to the magnitude of the largest deviation from zero of the real component $e(\Omega_i)$ or the imaginary component $e'(\Omega_i)$ of the weighted error.

The constraints, (2.3.18) through (2.3.21), are sufficient to carry out an approximation to the desired specifications. If the designed filter is desired to be stable, however, additional constraints will be needed. Since the approximation procedure involves linear programming, these additional constraints will have to be linear in form. The general form of these constraints could be as follows:

$$\sum_{m=0}^{M} f_m(b_m, \Omega_i) \leq 0; \; i=1,2,\ldots,L \tag{2.3.22}$$

The inequalities in (2.3.18) through (2.3.22) can be simplified as follows:

Let

$$Q_R(\Omega_i) = 1 + \sum_{m=1}^{M} b_m \cos(m\Omega_i) \tag{2.3.23}$$

$$Q_I(\Omega_i) = \sum_{m=1}^{M} b_m \sin(m\Omega_i) \tag{2.3.24}$$

$$P_R(\Omega_i) = 1 + \sum_{n=0}^{N} a_n \cos(n\Omega_i) \tag{2.3.25}$$

$$P_I(\Omega_i) = \sum_{n=0}^{N} a_n \sin(n\Omega_i) \tag{2.3.26}$$

$$\sum_{m=0}^{M} b_m f_m(\Omega_i) \leq 0 \tag{2.3.27}$$

Substituting inequalities (2.3.23) through (2.3.26) into inequalities (2.3.18) through (2.3.21) and simplifying, one can rewrite the linear programming problem as:

Maximize

$$g = \xi \qquad (2.3.28)$$

Subject to

$$\sum_{m=1}^{M} b_m\left\{Y(\Omega_i)\cos(m\Omega_i) + Y'(\Omega_i)\sin(m\Omega_i)\right\} - \sum_{n=0}^{N} a_n\cos(n\Omega_i) + \xi \leqslant -Y(\Omega_i) \qquad (2.3.29)$$

$$\sum_{m=1}^{M} b_m\left\{Y'(\Omega_i)\cos(m\Omega_i) - Y(\Omega_i)\sin(m\Omega_i)\right\} + \sum_{n=0}^{N} a_n\sin(n\Omega_i) + \xi \leqslant -Y'(\Omega_i) \qquad (2.3.30)$$

$$-\sum_{m=1}^{M} b_m\left\{Y(\Omega_i)\cos(m\Omega_i) + Y'(\Omega_i)\sin(m\Omega_i)\right\} + \sum_{n=0}^{N} a_n\cos(n\Omega_i) + \xi \leqslant Y(\Omega_i) \qquad (2.3.31)$$

$$-\sum_{m=1}^{M} b_m\left\{Y'(\Omega_i)\cos(m\Omega_i) - Y(\Omega_i)\sin(m\Omega_i)\right\} - \sum_{n=0}^{N} a_n\sin(n\Omega_i) + \xi \leqslant Y'(\Omega_i) \qquad (2.3.32)$$

and

$$\sum_{m=1}^{M} b_m f_m(\Omega_i) \leqslant 0$$

which completely defines the linear programming design problem for a one dimensional recursive digital filter.

2.3.b Stability Constraints in One Dimension

As indicated earlier, because of the use of linear programming in the approximation procedure, the constraints that can be used to design stable filters have to be linear in form. There are two types of constraints that readily satisfy the above requirement. These are as follows:

(a) Monotonicity of denominator filter coefficients [34],

$$\text{i.e., } b_0 > b_1 > b_2 > \ldots. > b_{n-1} > b_n > 0 \qquad (2.3.33)$$

where b_0 can be equal to 1. As shown in [34], the above constraint ensures that all the roots of Q(Z) lie outside the unit circle $|Z| = 1$ and hence ensures the stability of the transfer function H(Z).

(b) Real part of the denominator polynomial greater than zero on the unit circle [35],

$$\text{i.e., } \mathrm{Re}\{Q(Z)\} > 0 \text{ for } |Z| = 1 \qquad (2.3.34)$$

This also ensures stability of H(Z), if (2.3.33) is specified over the unit circle in the Z plane.

The constraint of (b) has been successfully used in the design of all pass digital filters [21]. It should, however, be noted that these constraints are just sufficient conditions for stability (the proof of the sufficiency of (b) is given in Appendix C). Also, since these constraints are just sufficient conditions for stability, they generate a subset of possible stable filter realizations.

From a closer examination of the two possible types of constraints that can be used in the design, it appears that the latter of the two constraints yields a larger subclass of stable filters compared to the other. This can at least be shown to be true in the case of filters whose order is less than or equal to 2.

Proof for the First Order Filter

Let

$$Q(Z) = 1 + b_1 Z \qquad (2.3.35)$$

where $Z=e^{-j\Omega}$ and Ω is the normalized frequency variable. Use of constraint (a) results in the following:

$$1 > b_1 > 0 \qquad (2.3.36)$$

Use of type (b) constraint results in,

$$1 + b_1 \cos\Omega > 0, \ 0 \leqslant \Omega \leqslant \pi \qquad (2.3.37)$$

From (3.3.37) it is clear that b_1 is constrained to be as follows:

$$-1 < b_1 < 1$$

Consider now a first order transfer function $H(Z)$, such that,

$$H(Z) = \frac{P(Z)}{Q(Z)}$$

where $Q(Z)$ is given by (2.3.35). Therefore the pole of $H(Z)$ is at $Z = -1/b$. From Figure 2.2 it can be observed that the application of the constraint (2.3.36) restricts the poles of the transfer function $H(Z)$ to the left half of the real axis in the Z plane. On the other hand, if (2.3.37) is used

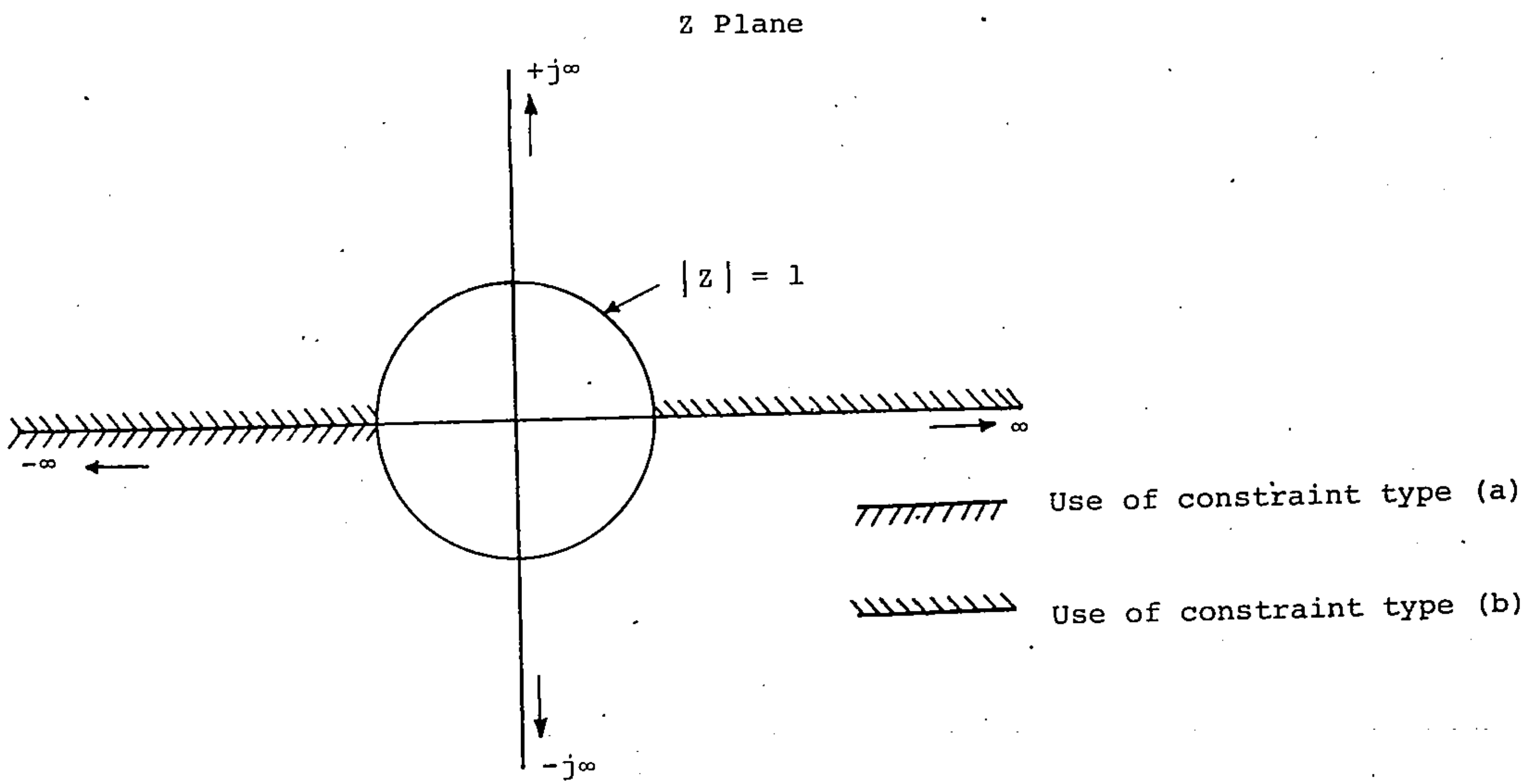


Figure 2.2 Constraint Regions for Poles of 1st Order Transfer Function.

then this results in H(Z) having poles on the entire real axis except for the portion which is inside and on the unit circle. Thus, constraint type (b) yields a larger class of stable filters compared to type (a) constraint.

In the second order example, (which is shown in Appendix D) although the proof is not so rigorous, it can still be seen that the type (b) constraint yields a larger subclass of stable filters compared to that of type (a). Therefore in all the design examples, the constraint type (b) has been used to design stable filters. The actual stability constraint that is used in the design is a slightly modified form of (2.3.34) and is given by,

$$\mathrm{Re}\{Q(Z)\} \geq \Delta C \text{ for } |Z| = 1 \qquad (2.3.38)$$

where ΔC is a small positive quantity.

Since $\mathrm{Re}\{Q(Z)\} = 1 + \sum_{m=1}^{M} b_m \cos(m\Omega_i)$, (3.3.38) can be rewritten as,

$$\sum_{m=1}^{M} b_m \cos(m\Omega_i) \geq \Delta C - 1; \quad 0 \leq \Omega_i \leq \pi \qquad (2.3.39)$$

or

$$\sum_{m=1}^{M} b_m \left[-\cos(m\Omega_i)\right] \leq 1 - \Delta C; \quad 0 \leq \Omega_i \leq \pi \qquad (2.3.40)$$

Proper choice of ΔC and a proper number of points at which the constraint (2.3.40) is specified can ensure a stable filter design.

2.3.c Design Procedure

The linear programming approach of filter design outlined in the previous section is now used in the design of linear phase filters. The desired linear phase characteristics can be specified in terms of the spatial delay.

Let $\phi(\Omega)$, be the phase characteristics in the frequency domain. As indicated in Appendix A, the delay characteristics $\tau(\Omega)$ is given by:

$$\tau(\Omega) = -\frac{d\phi(\Omega)}{d\Omega} \quad (2.3.41)$$

Therefore, if $\phi(\Omega)$ is desired to be linear with respect to the frequency Ω, then $\tau(\Omega)$ needs to be a constant. Letting $\tau(\Omega) = \tau_s$, where τ_s is a constant, the desired linear phase characteristics can be written as:

$$\phi(\Omega) = -\tau_s\Omega \quad (2.3.42)$$

Using (2.3.42), the real and imaginary components of the desired specifications given in (2.3.2) and (2.3.3) can be written as:

$$Y(\Omega_i) = R(\Omega_i)\cos(-\tau_s\Omega_i) \quad (2.3.43)$$

$$Y'(\Omega_i) = R(\Omega_i)\sin(-\tau_s\Omega_i); \; i=1,2,\ldots L \quad (2.3.44)$$

The linear programming problem can therefore be rewritten as:

Maximize

$$g = \xi$$

Subject to the following constraints:

$$\sum_{m=1}^{M} b_m R(\Omega_i)\left[\cos\{\Omega_i(m+\tau_s)\}\right] - \sum_{n=0}^{N} a_n \cos(n\Omega_i) + \xi$$

$$\leqslant -R(\Omega_i)\cos(-\tau_s\Omega_i) \tag{2.3.45}$$

$$\sum_{m=1}^{M} b_m R(\Omega_i)\left[-\sin\{\Omega_i(m+\tau_s)\}\right] + \sum_{n=0}^{N} a_n \sin(n\Omega_i) + \xi$$

$$\leqslant -R(\Omega_i)\sin(-\tau_s\Omega_i) \tag{2.3.46}$$

$$-\sum_{m=1}^{M} b_m R(\Omega_i)\left[\cos\{\Omega_i(m+\tau_s)\}\right] + \sum_{n=0}^{N} a_n \cos(n\Omega_i) + \xi$$

$$\leqslant R(\Omega_i)\cos(-\tau_s\Omega_i) \tag{2.3.47}$$

$$-\sum_{m=1}^{M} b_m R(\Omega_i)\left[-\sin\{\Omega_i(m+\tau_s)\}\right] - \sum_{n=0}^{N} a_n \sin(n\Omega_i) + \xi$$

$$\leqslant R(\Omega_i)\sin(-\tau_s\Omega_i); \quad i = 1,2,\ldots,L \tag{2.3.48}$$

Thus given an arbitrary magnitude characteristics, it is now possible to approximate these characteristics together with a linear phase specification (specified in terms of delay τ_s) by a recursive filter of specified order. A good design is normally obtained for a particular value of τ_s that lies within a certain range of values. This range can be determined by examining the impulse response corresponding to the given arbitrary magnitude characteristics. Figure 3.3 shows the range of τ_s in the impulse response

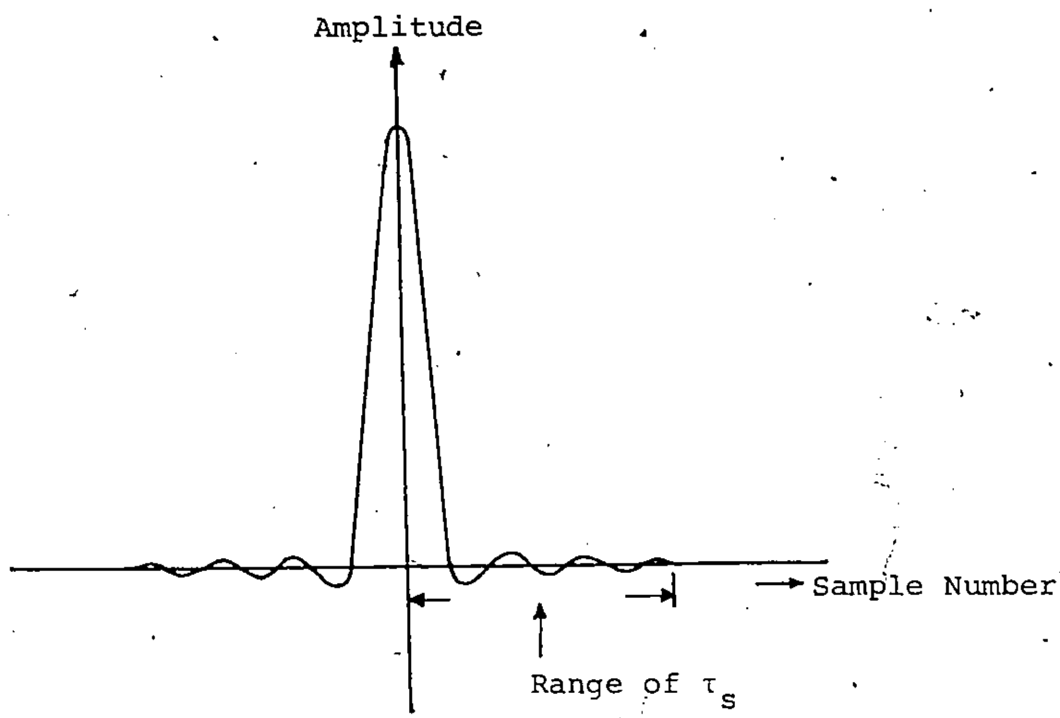


Figure 2.3 Range of τ_s in the Impulse Response.

corresponding to low pass type magnitude characteristics. Searching for a proper τ_s in the entire range may result in excessive design time. However, given the order of the filter, one can specify appropriate upper and lower limits for τ_s. Thus if K is chosen as the order of filter, then an appropriate range for τ_s is $(K-5) \leqslant \tau_s \leqslant K$. An argument as to why the upper limit for τ_s is equal to K can be understood from the following.

Consider an all pass transfer function of order K. A property of this function is that [1],

$$\int_0^{\pi} \tau(\Omega)\, d\Omega = K\pi$$

From the above, it can be seen that to realize a constant delay exactly equal to K, up to the nyquist rate, the filter order should at least be equal to K. Therefore, in the case of a general recursive digital filter, the order of the filter should at least be greater than or equal to K, if it is to realize, approximately, a constant delay K and also at the same time approximate given arbitrary magnitude specifications.

The lower limit for τ_s is chosen to be equal to (K-5), although the ideal lower limit can be equal to 1. However, choice of the lower limit, indicated above has not only reduced the excessive design time but also has provided consistently good results, as shown in the examples later.

A suitable procedure for designing linear phase filters is as follows:

1) Specify the magnitude characteristics; i.e., $R(\Omega)$;

2) Choose the filter order; say K and range of τ_s as indicated above;

3) Solve the linear programming problem for each τ_s starting from upper limit of K (in descending order) until a maximum in ξ is obtained. Denote the corresponding τ_s as $\tau_{s_{max}}$;

4) Choose the coefficients of the filter for which maximum ξ was obtained and compute frequency response and errors.

The error measures in step 4) are a) the squared error sum in magnitude, E_1, which is given by:

$$E_1 = \sum_{i=1}^{L} \Big[R(\Omega_i) - |H(\Omega_i)| \Big]^2 \qquad (2.3.49)$$

b) the squared error sum in the delay characteristics in the pass band of the filter, E_2, given by:

$$E_2 = \sum^{\text{(Pass Band)}} \Big[\tau_{s_{max}} - \tau(\Omega_i) \Big]^2 \qquad (2.3.50)$$

$R(\Omega_i)$ and $\tau_{s_{max}}$ are the desired magnitude characteristics and the constant group delay and $|H(\Omega_i)|$ and $\tau(\Omega_i)$ are the magnitude and overall group delay characteristics of the designed filter at the specified discrete frequency points Ω_i.

As indicated above, the error measure incorporates the error in delay characteristics rather than the error in phase. A point to note is that the degree of linearity of the phase can be observed much more clearly using group delay characteristics since they are the differential of phase characteristics. Also, the error E_2 in (2.3.50) is computed only over the pass band of the filter instead of computing over all the frequency points of specification, because, any non-linearity in phase or group delay of the filter is of little consequence outside the pass band region, since in this region the magnitude of the filter characteristics is not significant.

Using the procedure described above, a large number of filters have been designed and the examples of these designs are presented in Section 2.3.e.

2.3.d Computational Considerations

Reconsider the linear programming design problem of (2.3.27) through (2.3.32). This problem can be written in the matrix notation of section (2.2.a) as follows.

Find a vector $(b_1, b_2, \ldots, b_M, a_0, a_1, \ldots, a_N, \xi)$, subject to constraints

$$\left[b_1, b_2, \ldots, b_M, a_0, a_1, \ldots, a_N, \xi\right] \left[\quad D \quad\right] \leqslant \left[\; c \;\right] \tag{2.3.51}$$

such that

$$g = \left[b_1, b_2, \ldots, b_M, a_0, a_1, \ldots a_N, \xi\right] \cdot \begin{bmatrix} 0 \\ 0 \\ \cdot \\ \cdot \\ \cdot \\ 0 \\ 1 \end{bmatrix} \tag{2.3.52}$$

is maximized. The matrix D is made up of several submatrices as given below:

$$D = \left[\begin{array}{cc:cc:c} D_1 & D_2 & D_5 & D_6 & D_9 \\ \hdashline D_3 & D_4 & D_7 & D_8 & D_{10} \\ \hdashline \multicolumn{2}{c:}{D_{11}} & \multicolumn{2}{c:}{D_{12}} & D_{13} \end{array}\right] \tag{2.3.53}$$

Matrices D_1 and D_2 are of size M x L and their elements are given by:

$$d_{1_{ij}} = Y(\Omega_j)\cos\left[i\Omega_j\right] + Y(\Omega_j)\sin\left[i\Omega_j\right] \tag{2.3.54}$$

$$d_{2_{ij}} = Y'(\Omega_j)\cos\left[i\Omega_j\right] - Y(\Omega_j)\sin\left[i\Omega_j\right] \tag{2.3.55}$$

$$i=1,2,\ldots,M;\ j=1,2,\ldots,L$$

Matrices D_3 and D_4 are of size (N+1) x L and their elements are given by:

$$d_{3_{ij}} = -\cos\left[(i-1)\Omega_j\right] \tag{2.3.56}$$

$$d_{4_{ij}} = \sin\left[(i-1)\Omega_j\right] \tag{2.3.57}$$

$$i=1,2,\ldots,(N+1);\ j=1,2,\ldots,L$$

Matrix D_9 is the stability constraint matrix corresponding to the stability constraint given by (2.3.40). The size of the matrix is M x L_1, where L_1 may be greater than or equal to L. The elements of this matrix are given by:

$$d_{9_{ij}} = -\cos(i\Omega_j); \tag{2.3.58}$$

$$i=1,2,\ldots,M;\ j=1,2,\ldots,L_1$$

D_{11}, D_{12} and D_{13} are row matrices. Matrix D_{11} is of size 1 x (2·M) and is given by:

$$d_{11_i} = 1 \;;\; i=1,2,\ldots,(2\cdot M) \tag{2.3.59}$$

The matrices D_{10} and D_{13} which are of size (N+1) x L_1 and 1 x L_1 respectively, are equal to zero. The rest of the submatrices of D are as follows:

$$\begin{aligned} D_5 &= -D_1 \;;\; D_6 = -D_2 \\ D_7 &= -D_3 \;;\; D_8 = -D_4 \end{aligned} \tag{2.3.60}$$

and

$$D_{12} = D_{11} \tag{2.3.61}$$

The row matrix C also contains submatrices as shown below:

$$C = \left[\; c_1 \;\vdots\; c_2 \;\vdots\; c_3 \;\vdots\; c_4 \;\vdots\; c_5 \;\right] \tag{2.3.62}$$

Matrices c_1 through c_4 are of size 1 x L and matrix c_5 is of size 1 x L_1. Their elements are as follows:

$$c_{1_i} = -Y(\Omega_i)\,;\; c_{2_i} = -Y'(\Omega_i)\,;\quad i=1,2,\ldots,L \tag{2.3.63}$$

$$c_{5_i} = 1-\Delta C \qquad ;\; i=1,2,\ldots,L_1 \tag{2.3.64}$$

and

$$c_3 = -c_1 \text{ and } c_4 = -c_2 \tag{2.3.65}$$

where $Y(\Omega)$ and $Y'(\Omega)$ are real and imaginary components respectively of the desired specifications.

As can be seen from (2.3.60), (2.3.61) and (2.3.65), the coefficient matrix D, and matrix C have submatrices whose elements are redundant. Also submatrices D_{10} and D_{11} are null matrices. This property can therefore be made use of in reducing the storage requirements.

The linear programming problem of (2.3.51) and (2.3.52) may be solved by a straightforward application of the simplex [16] (or the revised simplex [16]) algorithm. However, this involves a large number of computations because of the large number of variables involved. In the straightforward problem, there are $4L_1$ inequalities with (M+N+2) unknowns where (M+N+1) are denominator and numerator coefficients of the filter. These coefficient variables are unconstrained and so they have to be replaced by the difference of two positive variables, as the simplex algorithm requires that the variables be greater than or equal to zero. Also each inequality in the problem is replaced by an equality by adding a slack variable. Thus the resulting linear program will have $(4L_1 + 2M + 2N + 1)$ variables. For large order filters this would involve very large amounts of computation. To avoid this problem, one can turn to the dual linear program. In the dual system, which is similar to the system of (2.2.10) through (2.2.12), after the addition of artificial variables to each equality, the result is $(4L_1 + M + N + 1)$ variables. This is a smaller number of variables than the primal problem and can be more easily handled.

Finally, there does not exist any rule for choosing the number of sample frequencies (i.e., the value for L_1) at which to specify the desired specifications, although it must be large enough to ensure that the response is well represented. Brophy and Salazar [21] suggest that L_1 be greater than four times the order of the filter. Therefore, if K is the order of the filter, then $L_1 \geq 4K$. The stability constraint is also specified over 4K (or greater) number of points, so as to reasonably ensure the stability of the filter.

3.3.e Examples of Design

In the examples presented here, the desired characteristics are specified over 81 equally spaced discrete frequency points. The stability constraint is also specified over the same frequency points. In plotting the frequency response of the designed filters, the phase response is eliminated in favour of the group delay response, since the non-linearities in phase can be observed more clearly in the group delay response.

Example 1 - Band Pass Filter

The desired specifications are as follows:

The lower pass band frequency $\Omega_{p_1} = 0.35\pi$

The upper pass band frequency $\Omega_{p_2} = 0.55\pi$

The complete magnitude specification $R(\Omega_i)$ is given by:

$$
\begin{aligned}
R(\Omega_i) &= e^{-k(\Omega p_1-\Omega i)^2} && \text{for } 0 \leq \Omega_i < \Omega_{p_1} \\
R(\Omega_i) &= 0 && \text{for } \Omega_{p_1} \leq \Omega_i \leq \Omega_{p_2} \\
R(\Omega_i) &= e^{-k(\Omega i-\Omega p_2)} && \text{for } \Omega_{p_2} < \Omega_i \leq \pi; i=1,2,\ldots,L
\end{aligned}
$$

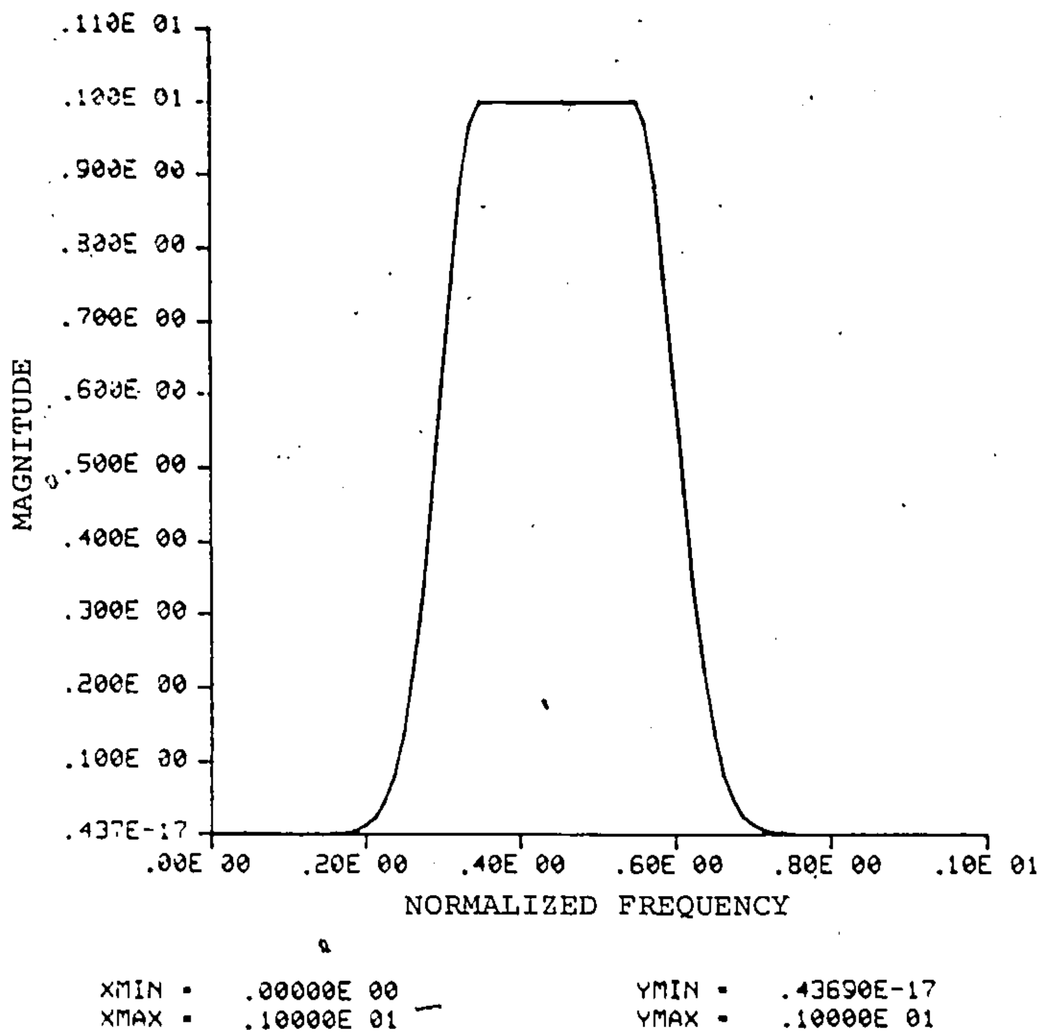


Figure 2.4 Desired Magnitude Characteristics.

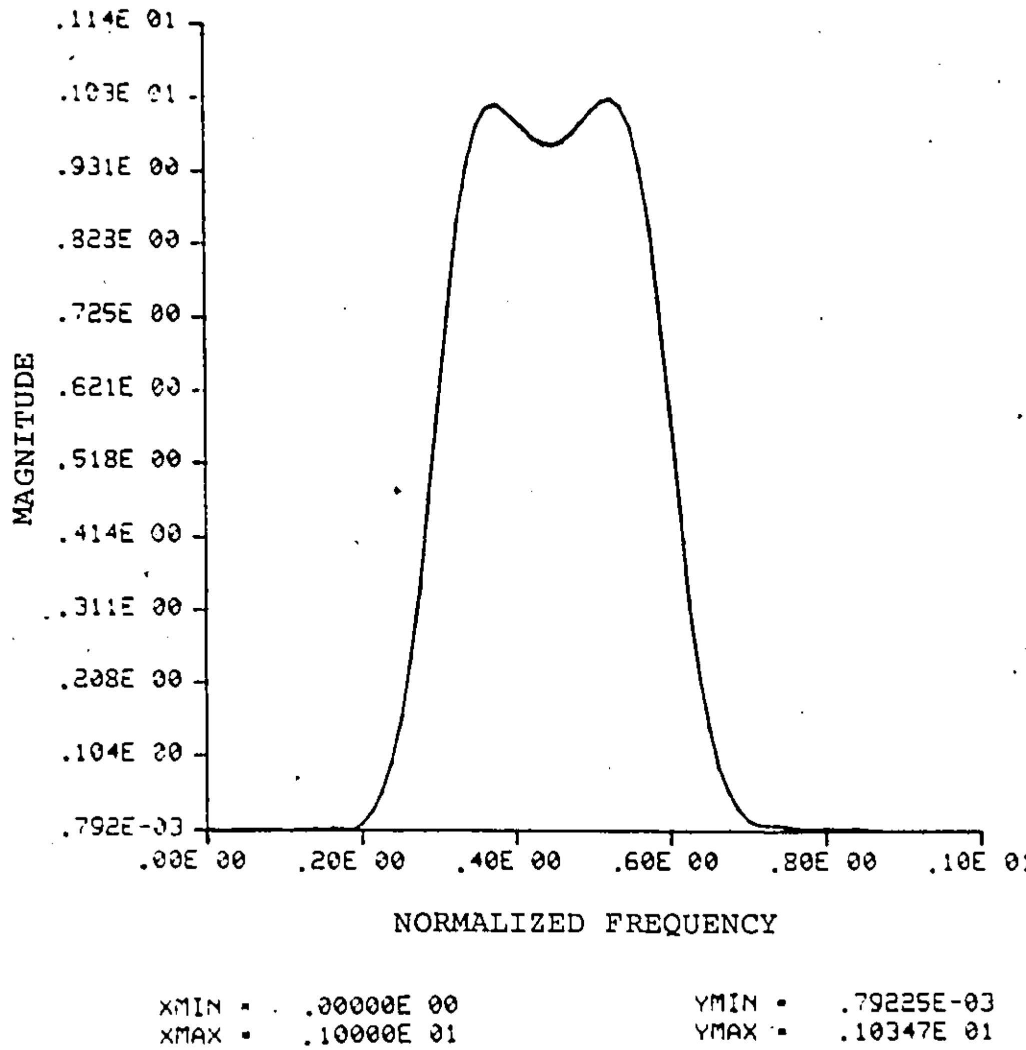


Figure 2.5 Designed Magnitude Characteristics

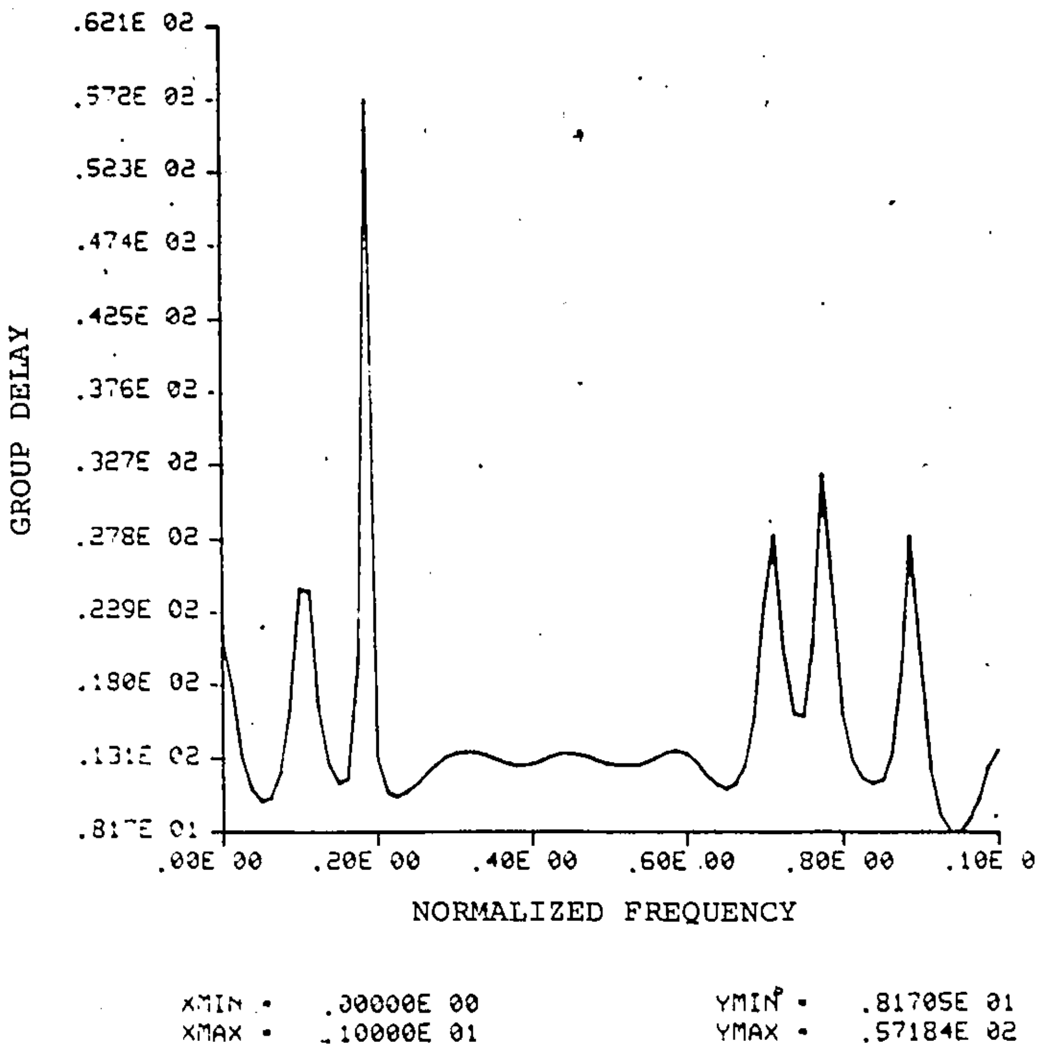


Figure 2.6 Designed Group Delay Characteristics

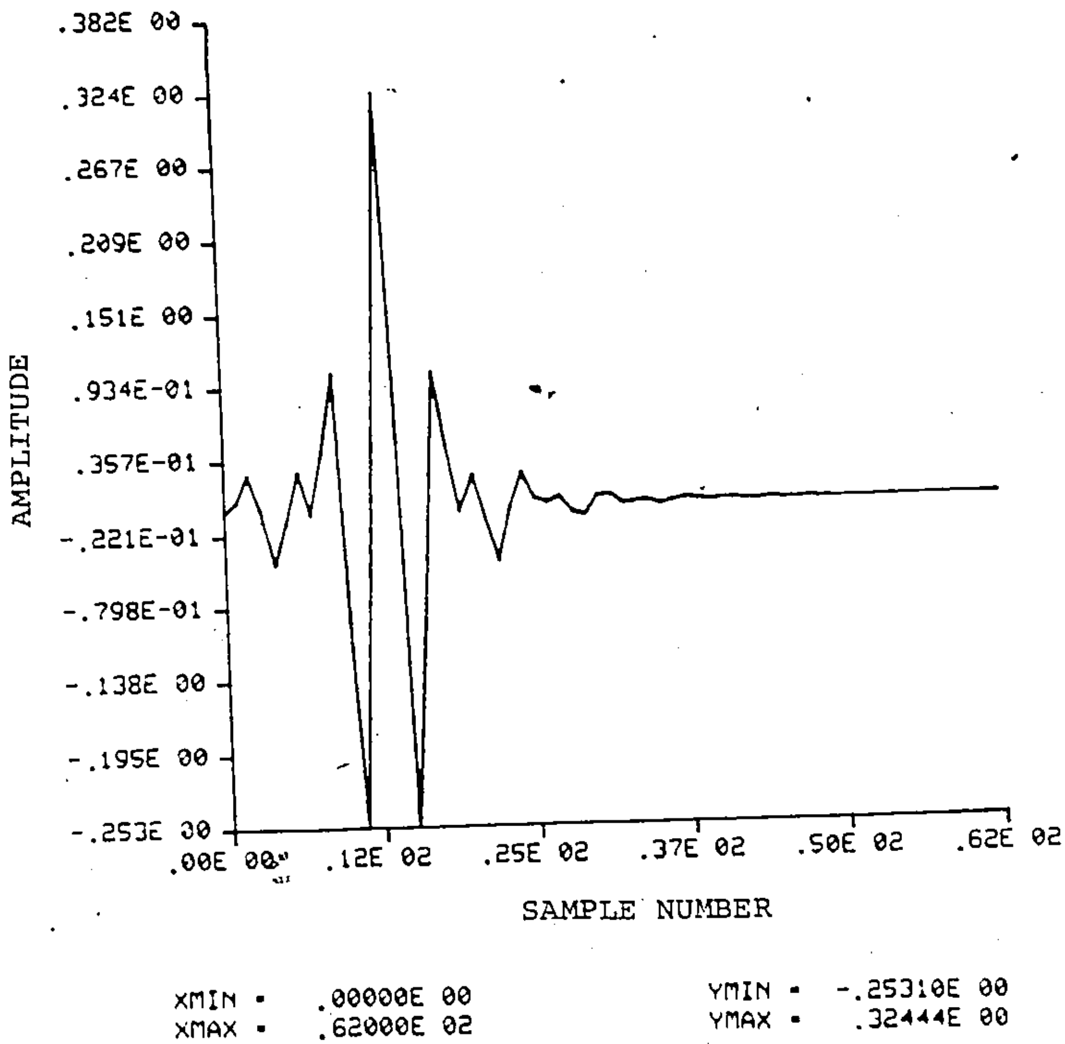


Figure 2.7 Designed Impulse Response

```
THE ZEROES OF THE FILTER ARE-

ZERO( 1)=     .3294084E 00 +J     5418085E 00
ZERO( 2)=      3294084E 00 +J   - 5418085E 00
ZERO( 3)=  -  8989047E 00 +J     0000000E 00
ZERO( 4)=      1203671E 00 +J     7476057E 00
ZERO( 5)=      1203671E 00 +J   - 7476057E 00
ZERO( 6)=  -  8963023E 00 +J     3264588E 00
ZERO( 7)=  -  8963023E 00 +J   - 3264588E 00
ZERO( 8)=      4703893E 00 +J     0000000E 00
ZERO( 9)=  -  5808388E 00 +J     7498324E 00
ZERO(10)=  -  5808388E 00 +J   - 7498324E 00
ZERO(11)=      9329610E 00 +J     0000000E 00
ZERO(12)=  -  7336164E 00 +J     6155835E 00
ZERO(13)=  -  7336164E 00 +J   - 6155835E 00
ZERO(14)=      8960102E 00 +J   - 3098137E 00
ZERO(15)=      8960102E 00 +J     3098137E 00
ZERO(16)=  -  4304536E 01 +J    .0000000E 00

THE POLES OF THE FILTER ARE-

POLE( 1)=      6356730E 00 +J   - 1080559E 01
POLE( 2)=      6356730E 00 +J     1080559E 01
POLE( 3)=  -  3536178E 00 +J   - 1139319E 01
POLE( 4)=  -  3536178E 00 +J     1139319E 01
POLE( 5)=      1257102E 01 +J   - 6273603E 00
POLE( 6)=      1257102E 01 +J     6273603E 00
POLE( 7)=  -  8697155E 00 +J   - 9954215E 00
POLE( 8)=  -  8697155E 00 +J     9954215E 00
POLE( 9)=  -  1533266E 00 +J   - 1326754E 01
POLE(10)=  -  1533266E 00 +J    .1326754E 01
POLE(11)=  -  1084559E 01 +J   - 6287196E 00
POLE(12)=  -  1084559E 01 +J     6287196E 00
POLE(13)=      2502317E 00 +J   - 1306255E 01
POLE(14)=      2502317E 00 +J     1306255E 01
POLE(15)=      7983543E 00 +J   - 9624366E 00
POLE(16)=      7983543E 00 +J    .9624366E 00
```

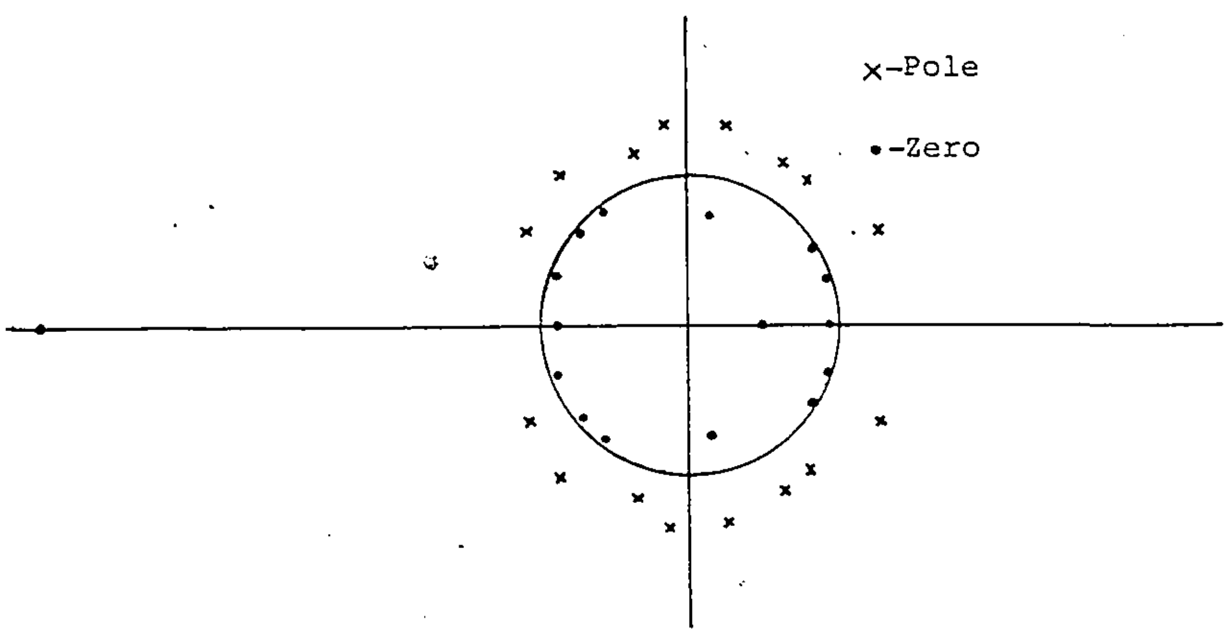


Figure 2.8 Pole-Zero Positions of the Designed Filter

The filter order was chosen to be 16. A maximum ξ was obtained for group delay value of 13. The value of ΔC (in the constraint of (3.3.38)) and k, chosen for this design are 1×10^{-6} and 20.0 respectively. The error measures of the design are as follows:

$$E_1 = 1.6 \times 10^{-2} , \quad E_2 = 1.48$$

The value of ξ (the variable that is maximized in the linear program) is equal to -7.8×10^{-3}. The desired magnitude specification is shown in Figure 2.4. The designed filter characteristics with its impulse response and pole-zero positions are shown in Figures 2.5 through 2.8.

Example 2 - Design of a Differentiater

The characteristics of an ideal differentiator is given by:

$$H(\Omega) = j(\frac{\Omega}{\Omega_n})$$

where Ω_n is half the sampling frequency in radians. From the magnitude-phase plot of $H(\Omega)$ shown in Figure 2.9(a) and (b), it can be seen that the phase characteristic is discontinuous at one-half the sampling frequency. This characteristic is difficult to realize because of the discontinuity at half the sampling frequency. However, with the addition of a one-half sample delay, this discontinuity can be eliminated [1] as shown in Figure 2.9(c) and (d). The discontinuity at the origin is of no consequence because the magnitude is zero at zero frequency. Thus the desired magnitude and phase characteristics

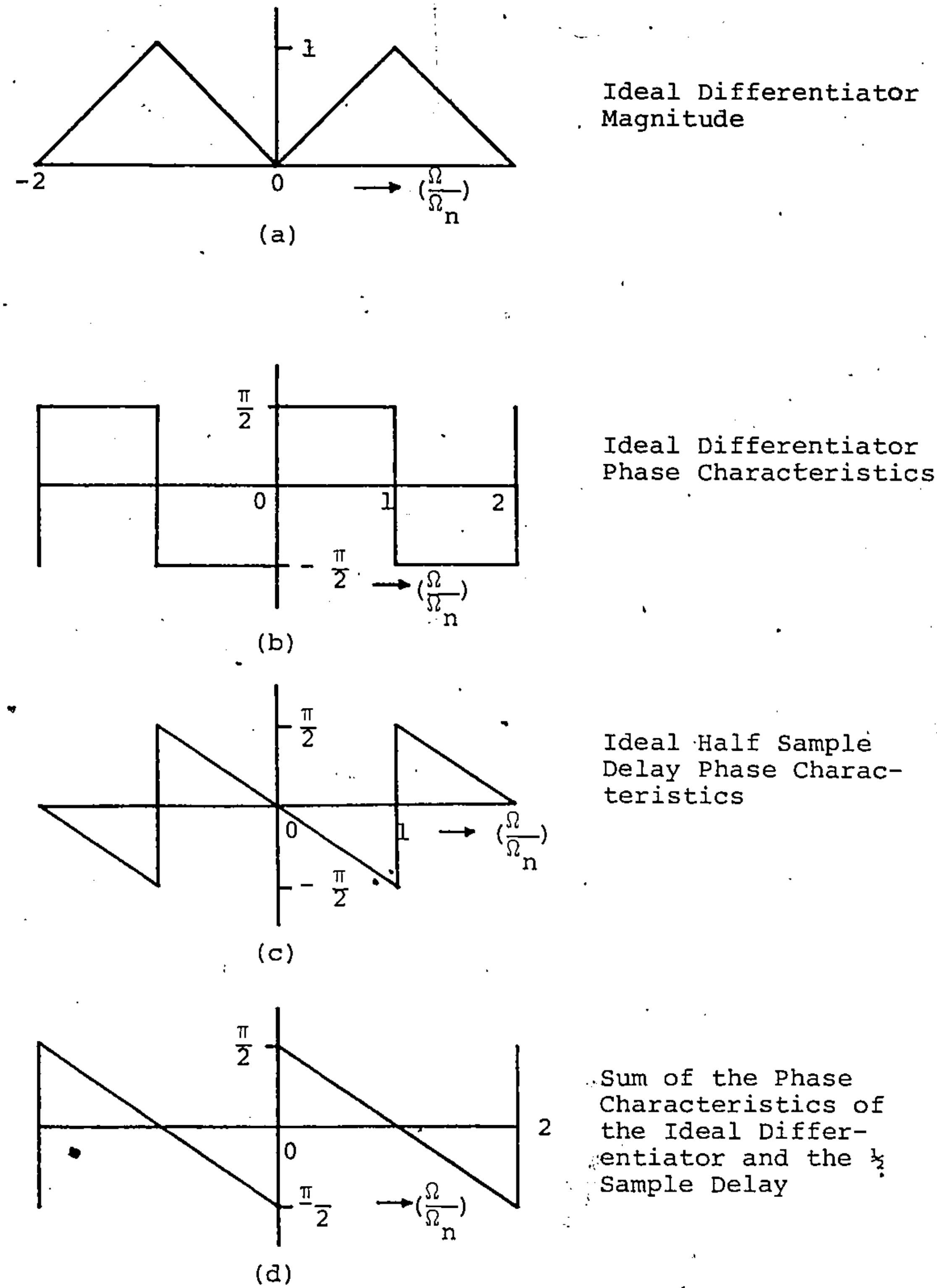


Figure 2.9 Realization of Differentiator With Half Sample Delay.

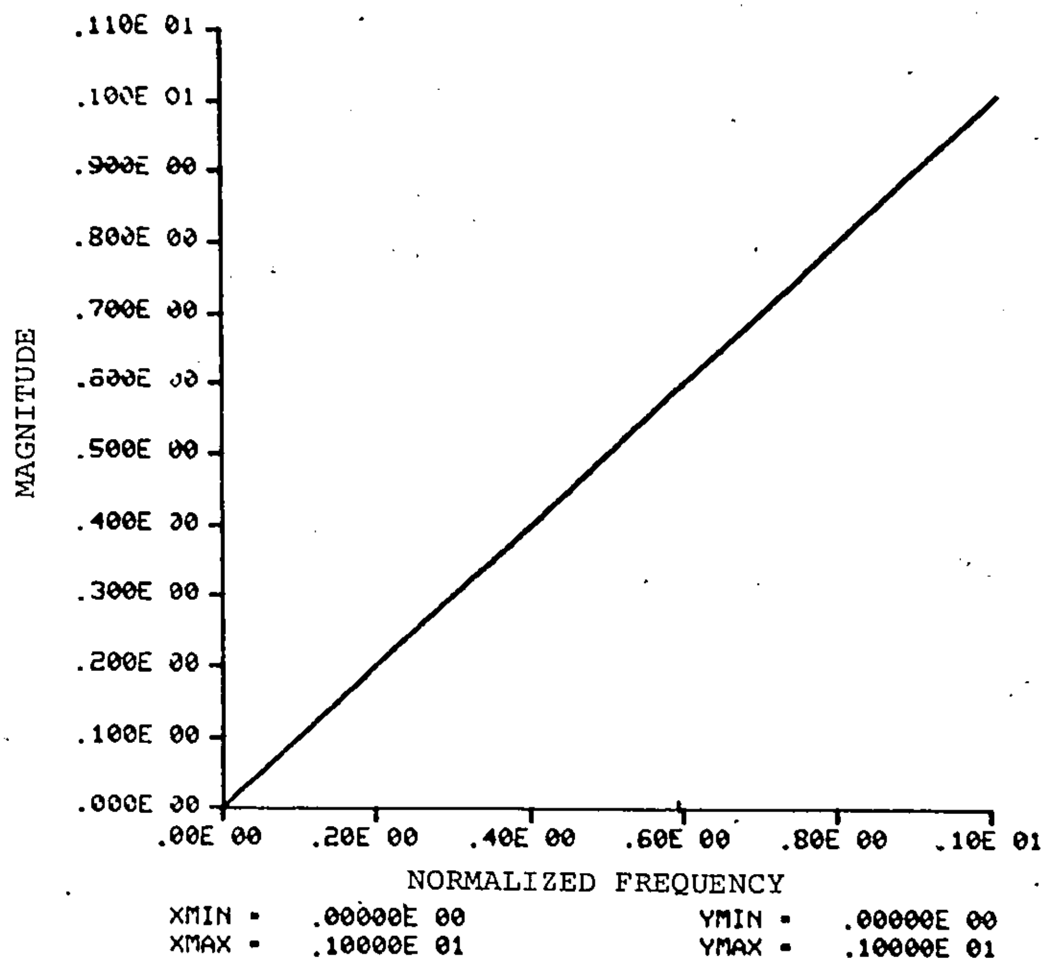


Figure 2.10 Desired Magnitude Characteristics.

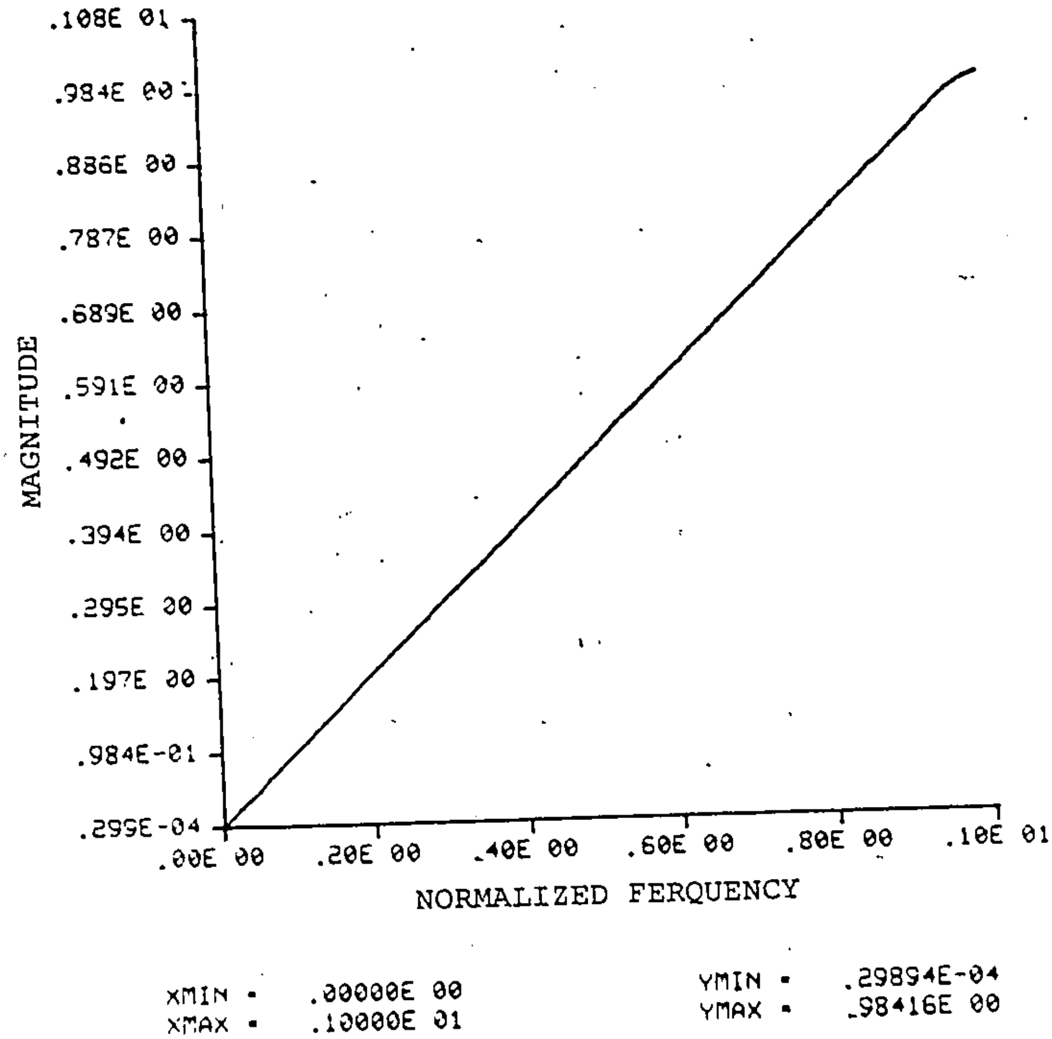


Figure 2.11 Designed Filter Magnitude Characteristics.

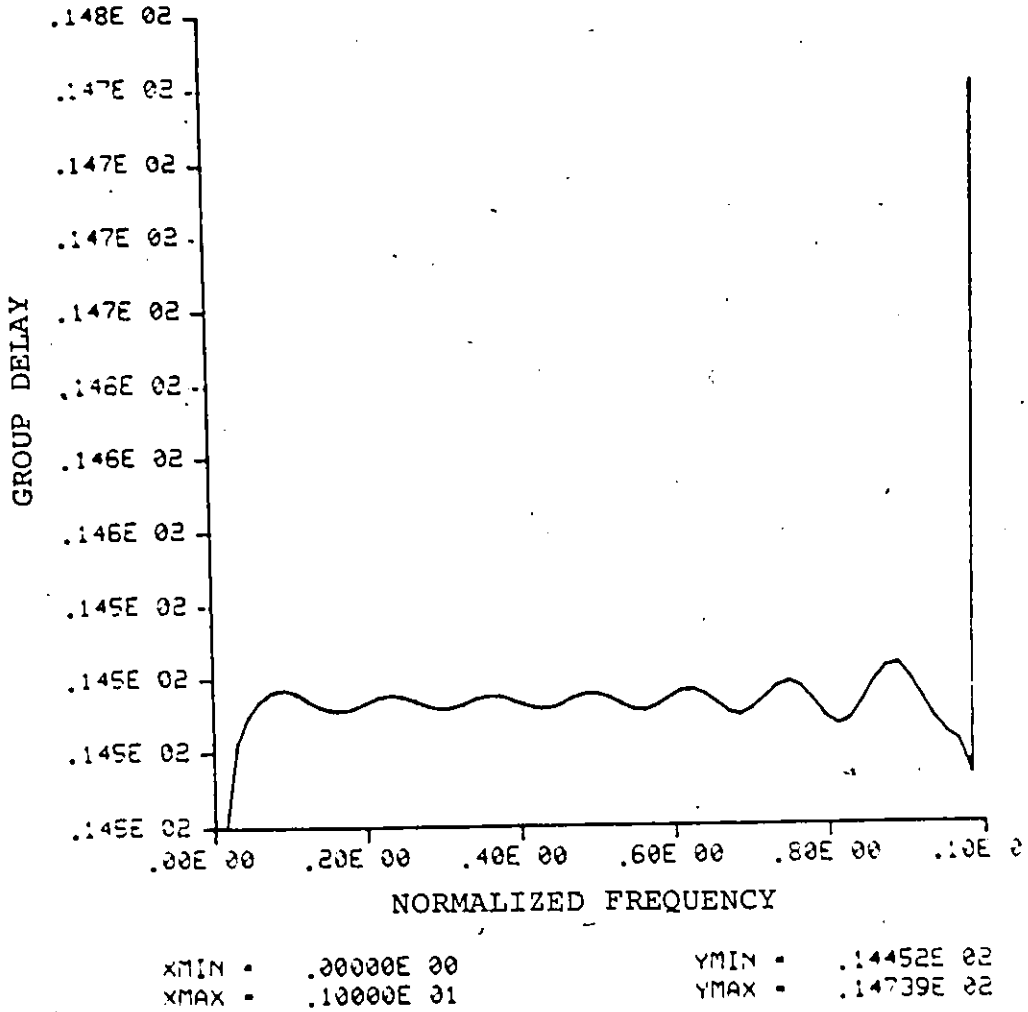


Figure 2.12 Designed Filter Group Delay Characteristics.

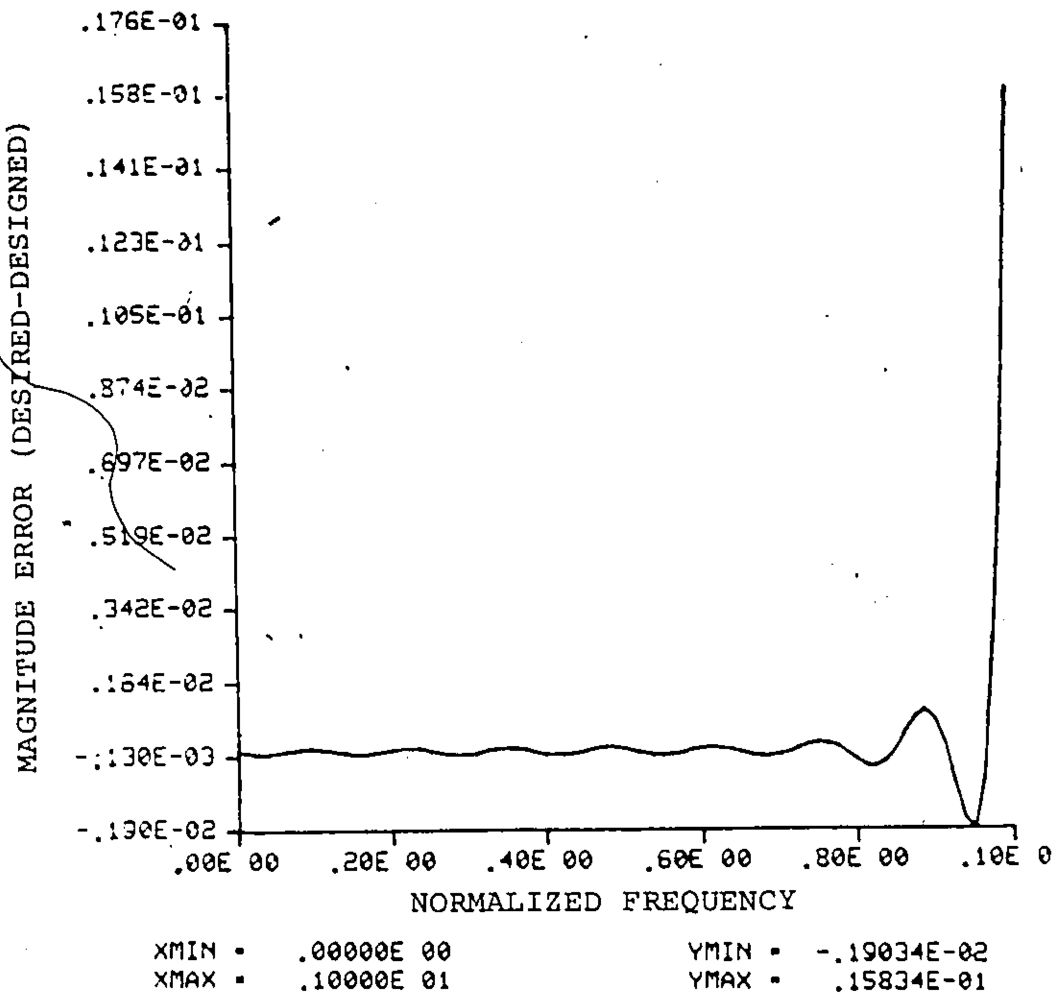


Figure 2.13 Error in the Magnitude

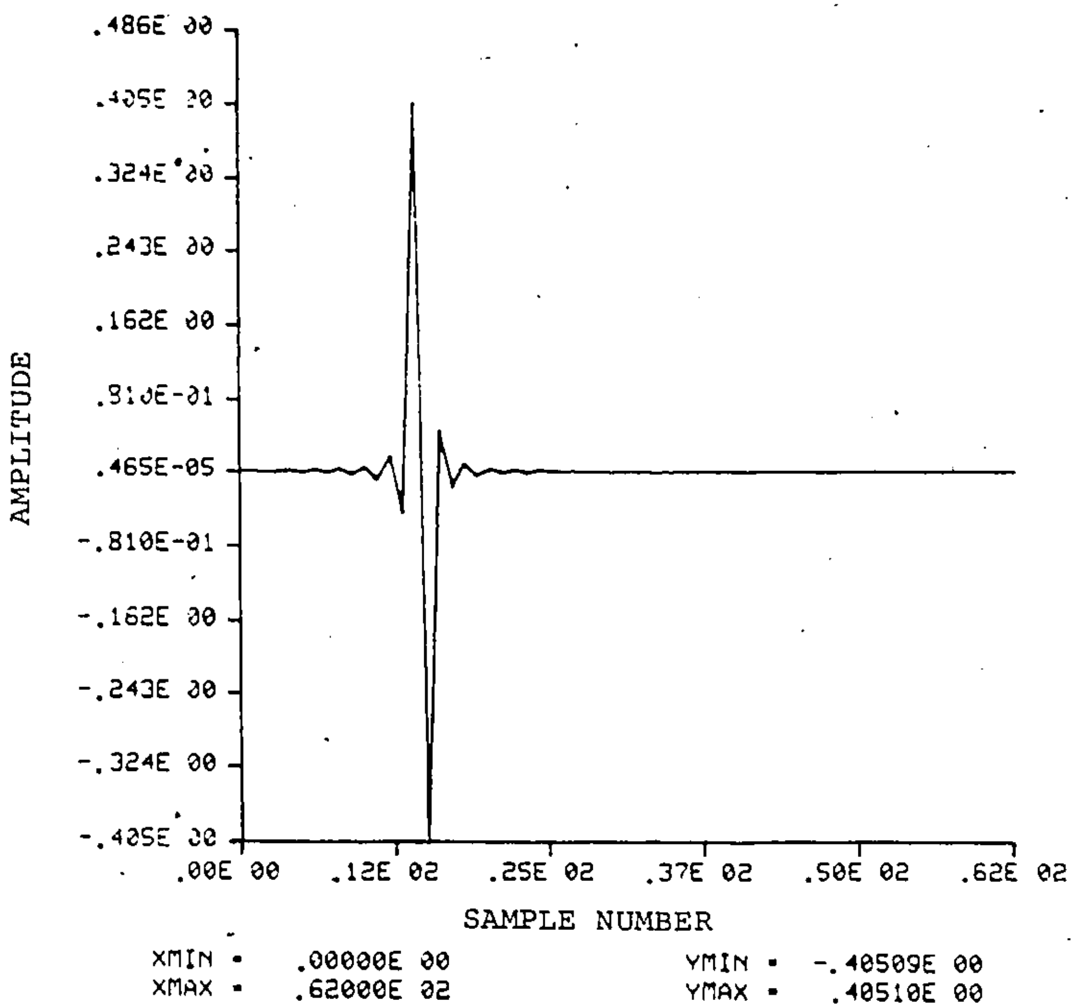


Figure 2.14 Impulse Response of the Designed Filter.

```
THE ZEROES OF THE FILTER ARE-

ZERO( 1)=     1000093E 01 +J     0000000E 00
ZERO( 2)= -   5861702E 00 +J     1718262E 00
ZERO( 3)= -   5861702E 00 +J  -  1718262E 00
ZERO( 4)=     5658592E 00 +J  -  1347634E 00
ZERO( 5)=     5658592E 00 +J     1347634E 00
ZERO( 6)=     2506263E 00 +J  -  5043780E 00
ZERO( 7)=     2506263E 00 +J     5043780E 00
ZERO( 8)= -   4302828E 00 +J  -  3951374E 00
ZERO( 9)= -   4302828E 00 +J     3951374E 00
ZERO(10)=     4431354E 00 +J  -  3580827E 00
ZERO(11)=     4431354E 00 +J     3580827E 00
ZERO(12)= -   2182096E 00 +J     5263062E 00
ZERO(13)= -   2182096E 00 +J  -  5263062E 00
ZERO(14)= -   1018166E 01 +J     0000000E 00
ZERO(15)=     1962210E-01 +J  -  5629674E 00
ZERO(16)=     1962210E-01 +J     5629674E 00
ZERO(17)= -   3175194E 01 +J     0000000E 00

THE POLES OF THE FILTER ARE-

POLE( 1)= -   1018085E 01 +J     0000000E 00
POLE( 2)= -   1530243E 01 +J     0000000E 00
POLE( 3)=     1760998E 01 +J     0000000E 00
POLE( 4)= -   1510073E 01 +J  -  6381251E 00
POLE( 5)= -   1510073E 01 +J     6381251E 00
POLE( 6)=     6182000E 00 +J  -  1703534E 01
POLE( 7)=     6182000E 00 +J     1703534E 01
POLE( 8)=     1220490E 01 +J  -  1329367E 01
POLE( 9)=     1220490E 01 +J     1329367E 01
POLE(10)= -   7249820E 00 +J  -  1613375E 01
POLE(11)= -   7249820E 00 +J     1613375E 01
POLE(12)=     1624080E 01 +J  -  7322979E 00
POLE(13)=     1624080E 01 +J     7322979E 00
POLE(14)= -   1230965E 01 +J  -  1194754E 01
POLE(15)= -   1230965E 01 +J     1194754E 01
POLE(16)= -   7138175E-01 +J  -  1800103E 01
POLE(17)= -   7138175E-01 +J     1800103E 01
```

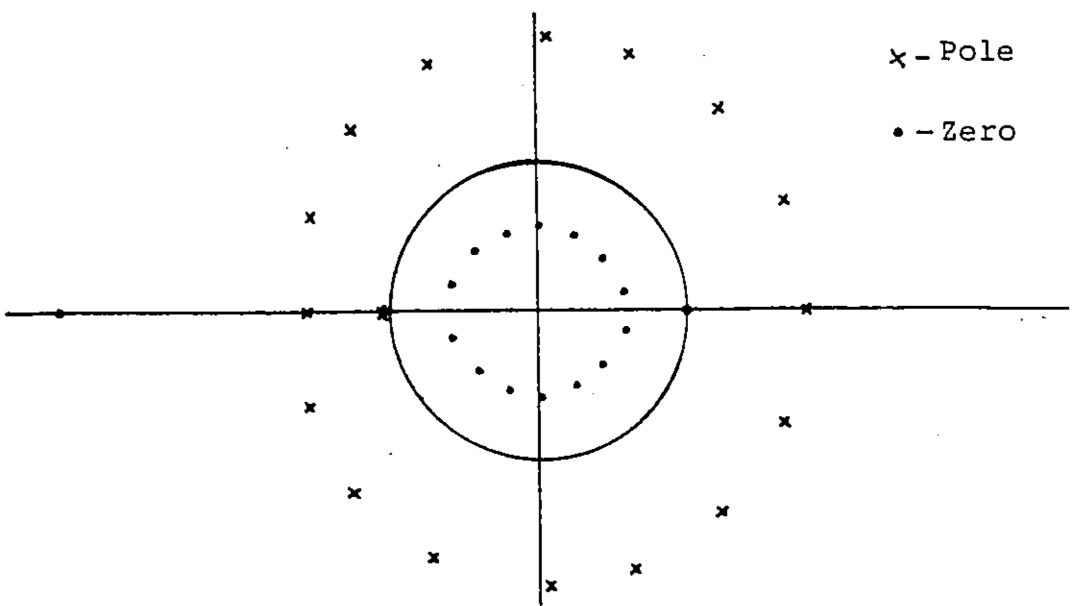


Figure 2.15 Pole-Zero Positions of the Designed Filter.

of the differentiator are:

$$R(\Omega_i) = \frac{\Omega_i}{\pi} \; ; \; 0 \leq \Omega_i \leq \pi$$

and the phase

$$\phi(\Omega_i) = 0.5\pi - (\tau_s + 0.5)\Omega_i \; ; \; 0 \leq \Omega_i \leq \pi$$

τ_s is the amount desired delay in samples for linear phase design. The total amount group delay realized is therefore equal to $(\tau_s + 0.5)$.

A filter of order 17 was chosen and ΔC was set equal to 1×10^{-6}. A maximum ξ was obtained for a group delay value of 14. In computing the error measure for group delay, the error at $\Omega = 0$ was disregarded, as it is of no consequence as the magnitude of the designed differentiator is almost zero at this frequency. The error measures for this design are as follows:

$$E_1 = 3.446 \times 10^{-4} \text{ and } E_2 = 6.8 \times 10^{-2}$$

The value of ξ is equal to 1.65×10^{-4}. The desired magnitude of the ideal differentiator is shown in Figure 2.10. The designed response, pole-zero position and various errors are shown in Figures 2.11 through 2.15.

2.4 Two-Dimensional Recursive Filter Design

In this section, the linear programming method of design discussed in the preceeding section for the one-dimensional filter is extended to the design of two dimensional recursive digital filters. As in the one dimensional case, the two dimensional approximation problem is made linear, so as to facilitate use of linear programming in the design. A linear stability constraint, similar to the one dimensional case, is proposed and is used in the design of stable filters.

2.4.a Theory of Approximation

Let $H(Z_1,Z_2)$ be the transfer function of a two dimensional recursive digital filter. Assume $H(Z_1,Z_2)$ is of the form:

$$H(Z_1,Z_2) = \frac{P(Z_1,Z_2)}{Q(Z_1,Z_2)} = \frac{\sum_{m=0}^{M1}\sum_{n=0}^{N1} a_{mn} Z_1^m Z_2^n}{\sum_{m=0}^{M2}\sum_{n=0}^{N2} b_{mn} Z_1^m Z_2^n} \tag{2.4.1}$$

where,

$$Z_1 = e^{-j\Omega_1} \quad\text{and}\quad Z_2 = e^{-j\Omega_2}$$

$$-\pi \leqslant \Omega_1 \leqslant \pi \text{ and } -\pi \leqslant \Omega_2 \leqslant \pi$$

Ω_1 and Ω_2 are the normalized frequency variables. The term b_{00} can be set equal to 1.0 without any loss of generality.

Now given an arbitrary magnitude and phase specification, it is possible, in a manner similar to the one dimensional case, to formulate a linear programming problem such that the constraints of the linear program are in terms of the filter coefficients. Let $R(\Omega_{1i}, \Omega_{2j})$ and $\phi(\Omega_{1i},\Omega_{2j})$ be the given magnitude and phase specifications respectively, specified over a frequency grid Ω_{1i}, Ω_{2j};

$i=1,2,\ldots,L1$, $j=1,2,\ldots,L2$. As before, the real components $Y(\Omega_{1i},\Omega_{2j})$ and the imaginary components $Y'(\Omega_{1i},\Omega_{2j})$ of the frequency domain specifications can be written as:

$$Y(\Omega_{1i},\Omega_{2j}) = R(\Omega_{1i},\Omega_{2j})\cos\left[\phi(\Omega_{1i},\Omega_{2j})\right] \quad (2.4.2)$$

$$Y'(\Omega_{1i},\Omega_{2j}) = R(\Omega_{1i},\Omega_{2j})\sin\left[\phi(\Omega_{1i},\Omega_{2j})\right] \quad (2.4.3)$$

From this point onwards, the mathematical manipulations are similar to the one dimensional design. Thus, in a similar manner, the two dimensional approximation problem can be stated as:

Maximize,

$$g = \xi$$

Subject to

$$Y(\Omega_{1i},\Omega_{2j})Q_R(\Omega_{1i},\Omega_{2j}) + Y'(\Omega_{1i},\Omega_{2j})Q_I(\Omega_{1i},\Omega_{2j}) - P_R(\Omega_{1i},\Omega_{2j}) + \xi \leqslant 0 \quad (2.4.5)$$

$$Y'(\Omega_{1i},\Omega_{2j})Q_R(\Omega_{1i},\Omega_{2j}) - Y(\Omega_{1i},\Omega_{2j})Q_I(\Omega_{1i},\Omega_{2j}) + P_I(\Omega_{1i},\Omega_{2j}) + \xi \leqslant 0 \quad (2.4.6)$$

$$-Y(\Omega_{1i},\Omega_{2j})Q_R(\Omega_{1i},\Omega_{2j}) - Y'(\Omega_{1i},\Omega_{2j})Q_I(\Omega_{1i},\Omega_{2j}) + P_R(\Omega_{1i},\Omega_{2j}) + \xi \leqslant 0 \quad (2.4.7)$$

$$-Y'(\Omega_{1i},\Omega_{2j})Q_R(\Omega_{1i},\Omega_{2j}) + Y(\Omega_{1i},\Omega_{2j})\,Q_I(\Omega_{1i},\Omega_{2j}) - P_I(\Omega_{1i},\Omega_{2j}) + \xi \leqslant 0\,; \quad (2.4.8)$$

$$i = 1,2,3,\ldots,L1;\ j = 1,2,\ldots,L2$$

where Q_R, Q_I and P_R, P_I are the real and imaginary components of Q and P respectively. Also, linear constraints on the denominator coefficients, of the form:

$$\sum_{m=0}^{M2} \sum_{n=0}^{N2} b_{mn} f_{mn}(\Omega_{i2}, \Omega_{2j}) \leqslant 0 \tag{2.4.9}$$

are incorporated to obtain stable designs.

Thus the above, completely defines the linear programming design problem for the two dimensional case.

2.4.b Stability Constraints in Two Dimensions

The stability constraint used in the two dimensional design procedure is similar to the constraint used in the one dimensional filter design. It is given by:

$$\mathrm{Re}\{Q(Z_1, Z_2)\} \geqslant 0 \quad \text{for} \quad |Z_1| = 1 \text{ and } |Z_2| = 1 \tag{2.4.10}$$

where $Q(Z_1, Z_2)$ is the denominator polynomial of the two-D transfer function. This constraint is linear in terms of the coefficients b(m,n) of the denominator polynomial $Q(Z_1, Z_2)$ and therefore it can be easily incorporated into the linear programming design procedure.

It should be noted here that the constraint of (2.4.10) is not a general stability constraint; however, it is a sufficient condition for $Q(Z_1, Z_2)$ to be a stable denominator polynomial of a two-D filter transfer function (the proof of the sufficienty is given in Appendix E) and therefore the use of this constraint in the filter design generates a subset of possible stable filters. This is not a drawback when compared to the existing design methods [2,3,4,5,6,7] where the designed filters also belong to a subclass of possible stable filters.

2.4.c Design Procedure

The design procedure is similar to the one dimensional case. As in the one dimensional case, the linear phase characteristic is specified in terms of the spatial† delays. As indicated in Chapter I, in the two dimensional case, the spatial delays are defined as:

$$\tau_1(\Omega_1,\Omega_2) = -\frac{\partial\phi(\Omega_1,\Omega_2)}{\partial\Omega_1} \tag{2.4.11}$$

and,

$$\tau_2(\Omega_1,\Omega_2) = -\frac{\partial\phi(\Omega_1,\Omega_2)}{\partial\Omega_2} \tag{2.4.12}$$

where $\phi(\Omega_1,\Omega_2)$ is the two dimensional phase characteristics in the frequency domain. Thus if the desired phase characteristics is to be linear, then $\tau_1(\Omega_1,\Omega_2)$ and $\tau_2(\Omega_1,\Omega_2)$ are assigned constant values. By setting $\tau_1(\Omega_1,\Omega_2)$ equal to a constant τ_x and $\tau_2(\Omega_1,\Omega_2)$ equal to a constant τ_y, the desired linear phase characteristics can be written as:

$$\phi(\Omega_1,\Omega_2) = -(\tau_x\Omega_1 + \tau_y\Omega_2) \tag{2.4.13}$$

The specifications can be simplified by setting $\tau_x = \tau_y = \tau_c$; i.e., realizing the same amount of delay in both directions in the spatial domain. The phase characteristics is then given by:

---

† This is group delay referred to space rather than time.

$$\phi(\Omega_1,\Omega_2) = -\tau_c(\Omega_1 + \Omega_2) \tag{2.4.14}$$

Therefore, the real and imaginary components of the desired specifications can be rewritten as:

$$\left.\begin{aligned} Y(\Omega_{1i},\Omega_{2j}) &= R(\Omega_{1i},\Omega_{2j})\cos\left[-\tau_c(\Omega_{1i} + \Omega_{2j})\right] \\ Y'(\Omega_{1i},\Omega_{2j}) &= R(\Omega_{1i},\Omega_{2j})\sin\left[-\tau_c(\Omega_{1i} + \Omega_{2j})\right] \end{aligned}\right\} \tag{2.4.15}$$

$$\text{for } i=1,2,\ldots,L1;\ j=1,2,\ldots,L2$$

Now, with,

$$Q_R(\Omega_{1i},\Omega_{2j}) = 1 + \sum_{\substack{m=0 \\ m+n\neq 0}}^{M2} \sum_{n=0}^{N2} b_{mn}\cos(m\Omega_{1i} + n\Omega_{2j}) \tag{2.4.16}$$

$$Q_I(\Omega_{1i},\Omega_{2j}) = \sum_{\substack{m=0 \\ m+n\neq 0}}^{M2} \sum_{n=0}^{N2} b_{mn}\sin(m\Omega_{1i} + n\Omega_{2j}) \tag{2.4.17}$$

$$P_R(\Omega_{1i},\Omega_{2j}) = \sum_{m=0}^{M1} \sum_{n=0}^{N1} a_{mn}\cos(m\Omega_{1i} + n\Omega_{2j}) \tag{2.4.18}$$

$$P_I(\Omega_{1i},\Omega_{2j}) = \sum_{m=0}^{M1} \sum_{n=0}^{N1} a_{mn}\sin(m\Omega_{1i} + n\Omega_{2j}) \tag{2.4.19}$$

The linear programming design problem for two dimensional filter becomes:

Maximize

$$g = \xi$$

Subject to

$$\sum_{\substack{m=0\\m+n\neq 0}}^{M2}\sum_{n=0}^{N2} b_{mn}R(\Omega_{1i},\Omega_{2j})\ \cos\{(m-\tau_c)\Omega_{1i} + (n-\tau_c)\Omega_{2j}\}$$

$$-\sum_{m=0}^{M1}\sum_{n=0}^{N1} a_{mn}\cos\{m\Omega_{1i} + n\Omega_{2j}\} + \xi \leqslant -R(\Omega_{1i},\Omega_{2j})\cos\{-\tau_c(\Omega_{1i}+\Omega_{2j})\}$$

(2.4.20)

$$\sum_{\substack{m=0\\m+n\neq 0}}^{M2}\sum_{n=0}^{N2} b_{mn}R(\Omega_{1i},\Omega_{2j})\ -\sin\{(m+\tau_c)\Omega_{1i} + (n+\tau_c)\Omega_{2j}\}$$

$$+\sum_{m=0}^{M1}\sum_{n=0}^{N1} a_{mn}\sin\{m\Omega_{1i} + n\Omega_{2j}\} + \xi \leqslant -R(\Omega_{1i},\Omega_{2j})\sin\{-\tau_c(\Omega_{1i}+\Omega_{2j})\}$$

(2.4.21)

$$-\sum_{\substack{m=0\\m+n\neq 0}}^{M2}\sum_{n=0}^{N2} b_{mn}R(\Omega_{1i},\Omega_{2j})\ \cos\{(m-\tau_c)\Omega_{1i} + (n-\tau_c)\Omega_{2j}\}$$

$$+\sum_{m=0}^{M1}\sum_{n=0}^{N1} a_{mn}\cos\{m\Omega_{1i} + n\Omega_{2j}\} + \xi \leqslant R(\Omega_{1i},\Omega_{2j})\cos\{-\tau_c(\Omega_{1i}+\Omega_{2j})\}$$

(2.4.22)

$$-\sum_{\substack{m=0\\m+n\neq 0}}^{M2}\sum_{n=0}^{N2} b_{mn}R(\Omega_{1i},\Omega_{2j})-\sin\{(m-\tau_c)\Omega_{1i} + (n-\tau_c)\Omega_{2j}\}$$

$$+\sum_{m=0}^{M1}\sum_{n=0}^{N1} a_{mn}\sin\{m\Omega_{1i} + n\Omega_{2j}\} + \xi \leqslant R(\Omega_{1i},\Omega_{2j})\sin\{-\tau_c(\Omega_{1i}+\Omega_{2j})\}$$

(2.4.23)

and the stability constraint,

$$1 + \sum_{\substack{m=0 \\ m+n\neq 0}}^{M2} \sum_{n=0}^{N2} b_{mn}\cos(m\Omega_{1i} + n\Omega_{2j}) \geqslant \Delta C \qquad (2.4.24)$$

where ΔC is a small positive quantity.

The design procedure, which is similar to the one dimensional case, can be described as follows:

1) Specify the desired magnitude characteristics:

$$R(\Omega_{1i},\Omega_{2j}) \; ; \; i=1,2,\ldots,L1 \; ; \; j=1,2,\ldots,L2$$

2) Choose an order for the filter, say N; and choose a range for τ_c.

3) Solve the linear programming problem for each τ_c from its upper limit (in decending order) untill a maximum ξ is obtained. Denote the corresponding τ_c as $\tau_{c_{max}}$

4) Choose coefficients of the filter for which maximum ξ was obtained and compute frequency response and error measures.

The range of τ_c in Step 2 is chosen based on N. The larger the range for τ_c, the higher is the computational time for the design of the filter. A suitable range has been found to be $(N-2) \leqslant \tau_c \leqslant (N+1)^+$.

The error measures in Step 4 are: a) squared sum of magnitude errors E_1, and is given by:

$$E_1 = \sum_{i=1}^{L1} \sum_{j=2}^{L2} \{R(\Omega_{1i},\Omega_{2j}) - |H(\Omega_{1i},\Omega_{2j})|\}^2 \qquad (2.4.25)$$

+ If (N-2) is less than or equal to zero, the lower limit is to be set equal to 1.

b) the squared error sum in spatial delay $\tau_1(\Omega_1,\Omega_2)$ in the pass band of the filter, E_2, and is given by:

$$E_2 = \sum\sum^{(\text{Pass Band})} \{\tau_{c_{max}} - \tau_1(\Omega_{1i},\Omega_{2j})\}^2 \qquad (2.4.26)$$

and c) the squared error sum in spatial delay $\tau_2(\Omega_1,\Omega_2)$ in the pass band of the filter E_3, which is given by:

$$E_3 = \sum\sum^{(\text{Pass Band})} \{\tau_{c_{max}} - \tau_2(\Omega_{1i},\Omega_{2j})\}^2 \qquad (2.4.27)$$

3.4.d <u>Computational Considerations</u>

The computational simplifications, as well as the storage reductions in the two dimensional case are similar to the one dimensional case. The linear programming problem of two dimensional filter design, given in Section 2.4.c, can be written in a matrix form as:

$$\left[b_{01},b_{02},\ldots,b_{0N_2},b_{10},b_{12},\ldots,b_{1N_2},\ldots,b_{M2N2},a_{00},a_{01},\ldots a_{M1N1},\xi\right]\cdot$$

$$\begin{bmatrix} \\ D \\ \\ \end{bmatrix} \leq \begin{bmatrix} c \end{bmatrix} \qquad (2.4.28)$$

such that,

$$g = \left[b_{01},\ldots,b_{M2N2},a_{00},\ldots,a_{M1N1},\xi\right]\begin{bmatrix}0\\0\\\cdot\\\cdot\\\cdot\\0\\1\end{bmatrix} \qquad (2.4.29)$$

is maximized. The matrix D consists of several submatrices as given below:

$$D = \left[\begin{array}{cc:c:c:c} D_1 & D_2 & D_5 & D_6 & D_9 \\ \hdashline D_3 & D_4 & D_7 & D_8 & D_{10} \\ \hdashline \multicolumn{2}{c:}{D_{11}} & \multicolumn{2}{c:}{D_{12}} & D_{13} \end{array}\right], \qquad (2.4.30)$$

Matrices D_1 and D_2 are also made up of sub matrices:

$$D_1 = \begin{bmatrix} F_{i,1} & F_{1,2} \cdots F_{1,L2} \\ F_{2,1} & \\ \vdots & \\ F_{(M2+1),1} \cdots\cdots & F_{(M2+1),L2} \end{bmatrix} \qquad (2.4.31)$$

$$D_2 = \begin{bmatrix} G_{1,1} & G_{1,2} \cdots G_{1,L2} \\ G_{2,1} & \\ \vdots & \\ G_{(M2+1),1} \cdots\cdots & G_{(M2+1),L2} \end{bmatrix} \qquad (2.4.32)$$

The elements in the submatrices of D_1 are given by:

$$F_{i,j_{k\ell}} = Y(\Omega_{1j},\Omega_{2\ell})\{\cos\ (i-1)\Omega_{1j} + m\Omega_{2\ell}\ \}$$

$$+ Y'(\Omega_{1j},\Omega_{2\ell})\{\sin\ (i-1)\Omega_{1j} + m\Omega_{2\ell}\ \} \qquad (2.4.33)$$

where,

$$\left.\begin{array}{ll} & i = 1,2,\ldots,(M2+1) \\ & m = k \quad \text{and} \quad k = 1,2,\ldots,N2 \text{ for } i = 1 \\ \text{and} & \\ & m = (k-1) \quad \text{and} \quad k = 1,2,\ldots,(N2+1) \text{ for } i \neq 1 \\ \text{and} & \\ & j = 1,2,\ldots,L2 \; ; \; \ell = 1,2,\ldots,L1 \end{array}\right\} \quad (2.4.34)$$

The elements of the submatrices of D_2 are given by:

$$G_{i,j_{k\ell}} = Y'(\Omega_{1j},\Omega_{2\ell})\{\cos\left[(i-1)\Omega_{1j} + m\Omega_{2\ell}\right]\}$$

$$- Y(\Omega_{1j},\Omega_{2\ell})\{\sin\left[(i-1)\Omega_{1j} + m\Omega_{2\ell}\right]\} \quad (2.4.35)$$

where i,j,k,ℓ,m are as given in (2.4.34).

The submatrix D_9 is also made up of submatrices as follows:

$$D_9 = \begin{bmatrix} H_{1,1} & H_{1,2} \cdots H_{1,L2} \\ H_{2,1} & \\ \vdots & \\ H_{(M2+1),1} & \cdots\cdots H_{(M2+1)L2} \end{bmatrix} \quad (2.4.36)$$

and the elements of the submatrices $\{H_{i,j}\}$, which corresponds to the stability constraint given by (2.4.24), are:

$$H_{i,j_{k\ell}} = -\cos\left[(i-1)\Omega_{1j} + m\Omega_{2\ell}\right] \quad (2.4.37)$$

where, i,j,k,ℓ,m are as given in (2.4.34).

From the above, it can be seen that each of the submatrices, D_1, D_2 and D_9, are of size $(M2+1)(N2+1)-1 \times (L1 \cdot L2)$. Further, the first row of submatrices of D_1, D_2 and D_9 are of size N2 x L1, where as the rest of the submatrices of D_1, D_2 and D_9 are of size (N2+1) x L1. The submatrices D_3 and D_4 of D are further made up of submatrices as follows:

$$D_3 = \begin{bmatrix} I_{1,1} & I_{1,2} \cdots I_{1,L2} \\ I_{2,1} & \\ \vdots & \\ I_{(M1+1),1} \cdots\cdots & I_{(M1+1),L2} \end{bmatrix} \qquad (2.4.38)$$

$$D_4 \quad \begin{bmatrix} J_{1,1} & J_{1,2} \cdots J_{1,L2} \\ J_{2,1} & \\ \vdots & \\ J_{(M1+1),1} \cdots\cdots & J_{(M1+1),L2} \end{bmatrix} \qquad (2.4.39)$$

where elements of each of submatrices $I_{i,j}$'s are:

$$I_{i,j_{k\ell}} = -\cos\left[(i-1)\Omega_{1j} + (k-1)\Omega_{2\ell}\right] \qquad (2.4.40)$$

and the elements of $J_{i,j}$'s are:

$$J_{i,j_{k\ell}} = \sin\left[(i-1)\Omega_{1j} + (k-1)\Omega_{2\ell}\right] \qquad (2.4.41)$$

and $k=1,2,\ldots,(N1+1)$; $\ell=1,2,\ldots,L1$

The submatrix D_{11} is as follows:

$$D_{11} = \begin{bmatrix} 1 \; 1 \ldots . 1 \end{bmatrix}; \tag{2.4.42}$$

and is of size $1 \times (2 \cdot L1 \cdot L2)$. The remaining submatrices of D are as follows:

$$\begin{aligned} &D_5 = -D_1 \; ; \; D_6 = -D_2 \; ; \; D_7 = -D_3 \; ; \; D_8 = -D_4 \\ &D_{10} = 0 \; ; \; D_{12} = D_{11} \; ; \; D_{13} = 0 \end{aligned} \tag{2.4.43}$$

The row matrix, C, on the right hand side of (2.4.27) can be split into four submatrices, as shown below:

$$C = \quad C_1 \quad C_2 \quad C_3 \quad C_4 \quad C_5 \tag{2.4.44}$$

C_1 and C_2 are further made up of submatrices as follows:

$$C_1 = \begin{bmatrix} S_1, S_2, \ldots, S_{L2} \end{bmatrix}; \; C_2 = \begin{bmatrix} T_1, T_2, \ldots, T_{L2} \end{bmatrix} \tag{2.4.45}$$

The elements of the submatrices S_i's and T_i's are:

$$S_{i_j} = -Y(\Omega_{1i}, \Omega_{2j}) \; ; \; T_{i_j} = -Y'(\Omega_{1i}, \Omega_{2j}) \tag{2.4.46}$$

$$j = 1, 2, \ldots, L1$$

The elements of C_5 are as follows:

$$C_5 = \begin{bmatrix} (1-\Delta C), \ldots, (1-\Delta C) \end{bmatrix} \tag{2.4.47}$$

and it is of size $1 \times (L1 \cdot L2)$. The remaining submatrices of C are:

$$C_3 = -C_1 \; ; \; C_4 = -C_2 \tag{2.4.48}$$

It is clear from the foregoing explanation that there is redundancy in the coefficient matrix, and in matrix C. Because of this redundancy, the storage requirement can be almost reduced by half. Thus, instead of storing entire D and C matrices, it is required to store only submatrices D_1, D_2, D_3, D_4, D_{11}, D_9, C_1, C_2 and C_5. As in the one dimensional case, the solution is obtained by solving the dual of the linear programming problem; this involves a comparatively smaller number of variables compared to the primal problem.

Finally, a procedure similar to the one dimensional case, is adopted in order to choose the number of sample frequencies over the right half plane of the frequency domain. Thus, given the denominator of the transfer function $H(Z_1,Z_2)$, i.e.,

$$Q(Z_1,Z_2) = \sum_{m=0}^{M2} \sum_{n=0}^{N2} b_{mn} Z_1^{+m} Z_2^{+n} .$$

the frequency domain specifications are specified over L1 and L2 number of equally spaced points in the Ω_1 and Ω_2 frequency axes respectively, such that $L1 \geq 8.M2$ and $L2 \geq 4.N2$. The stability constraint is also specified over the same (or larger) number of points in order to reasonably ensure stability of the designed filter.

2.4.e Design Examples

In the design procedure, the specifications in the frequency domain are specified over equally spaced frequency points in the right half of $\Omega_1 - \Omega_2$ plane. The value of ΔC in the constraint equation of (2.4.24) was set equal to 1×10^{-3}.

Example 1 Low Pass Filter Design

The specifications are as follows:

$$R(\rho) = 1.0 \qquad \text{for } 0 \leqslant \rho \leqslant 0.2\pi$$

$$R(\rho) = e^{-k(\rho-0.2\pi)} \qquad \text{for otherwise}$$

where,

$$\rho = \sqrt{\Omega_1^2 + \Omega_2^2}$$

The transition and stop band characteristics are gaussian and the value of k is set at 5.0. The order of the filter was chosen such that M1=M2=N1=N2=3. The initial value of τ_c was set equal to 3 for which the design had maximum value of ξ . The approximation was carried out over a 32 x 17 grid of equally spaced points in the frequency domain. The error measures for this design were as follows:

$$E_1 = 0.9192 \; ; \; E_2 = 0.92 \; ; \; E_3 = 0.93 \; ; \; |\xi| = 3.12 \times 10^{-2}$$

The desired magnitude and the designed magnitude and group delays $\tau_1(\Omega_1,\Omega_2)$, $\tau_2(\Omega_1,\Omega_2)$ are shown in Figure 2.16. Also Figure 2.17 shows the first 16 x 16 pixels of the impulse response and the coefficients of the designed filter.

Example 2 Band Pass Filter

The specifications for the band pass filter are as follows:

$$R(\rho) = 1.0 \qquad \text{for } 0.3\pi \leqslant \rho \leqslant 0.4\pi$$

$$R(\rho) = e^{-k(0.3 \;\; - \;\; \rho)^2} \qquad \text{for } \quad 0 \leqslant \rho \leqslant 0.3\pi$$

$$R(\rho) = e^{-k(\rho \; - \; 0.4\pi)^2} \qquad \text{for otherwise (value of k=5.0)}$$

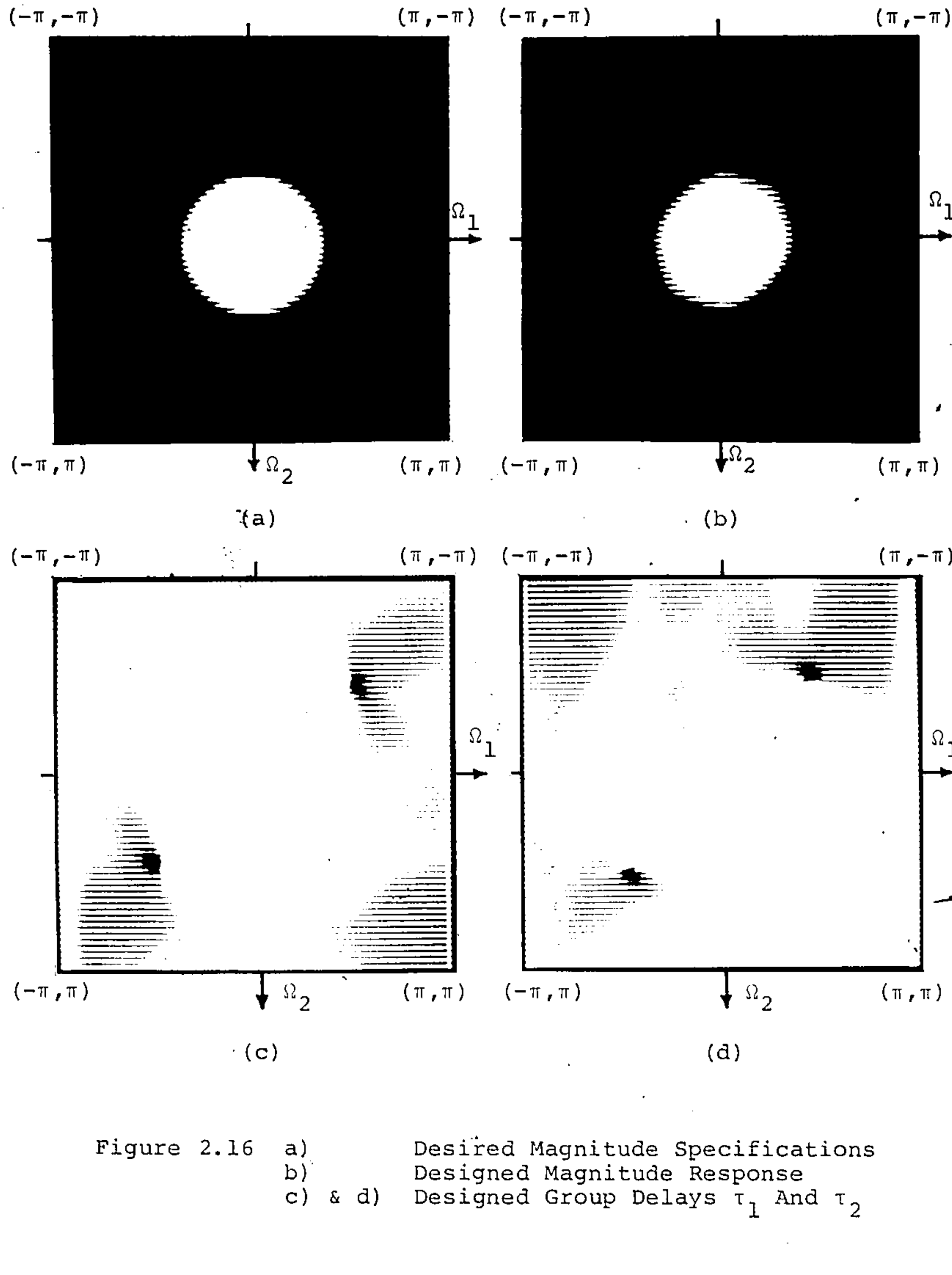
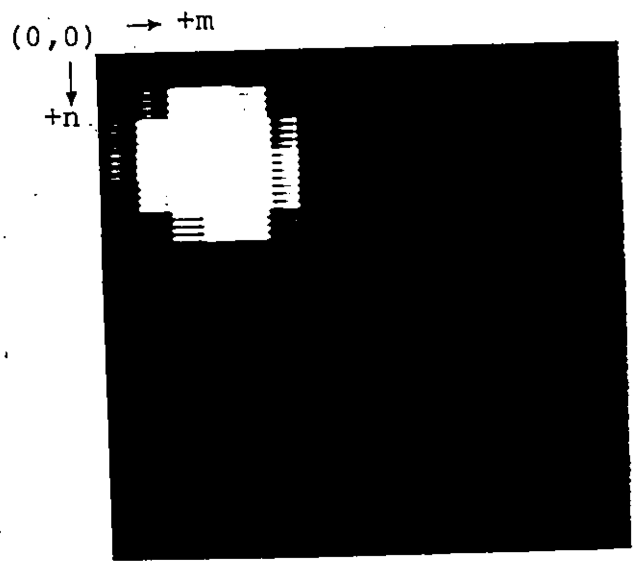


Figure 2.16 a) Desired Magnitude Specifications
b) Designed Magnitude Response
c) & d) Designed Group Delays τ_1 And τ_2



(a)

a_{00} =	-18972540E-02	b_{00} =	1.0000000
a_{01} =	21177270E-02	b_{01} =	- .84930090
a_{02} =	81653970E-02	b_{02} =	.44671370
a_{03} =	27611030E-02	b_{03} =	- .12157390
a_{10} =	21184090E-02	b_{10} =	- .84917370
a_{11} =	91720860E-02	b_{11} =	.51898170
a_{12} =	20013660E-01	b_{12} =	- .30581340
a_{13} =	17542880E-01	b_{13} =	.10858460
a_{20} =	81638170E-02	b_{20} =	.44659460
a_{21} =	20015080E-01	b_{21} =	- .30578520
a_{22} =	18486460E-01	b_{22} =	.24817360
a_{23} =	33414930E-01	b_{23} =	- .73503730E-01
a_{30} =	27604210E-02	b_{30} =	- .12151460
a_{31} =	17541230E-01	b_{31} =	.10852020
a_{32} =	33416200E-01	b_{32} =	- .73444780E-01
a_{33} =	19868710E-01	b_{33} =	.49963110E-02

Figuee 2.17 (a) First 16 x 16 Pixels of the Impulse Response+
(b) Coefficients of the Designed Filter

+Each square area of constant intensity represents one pixel.

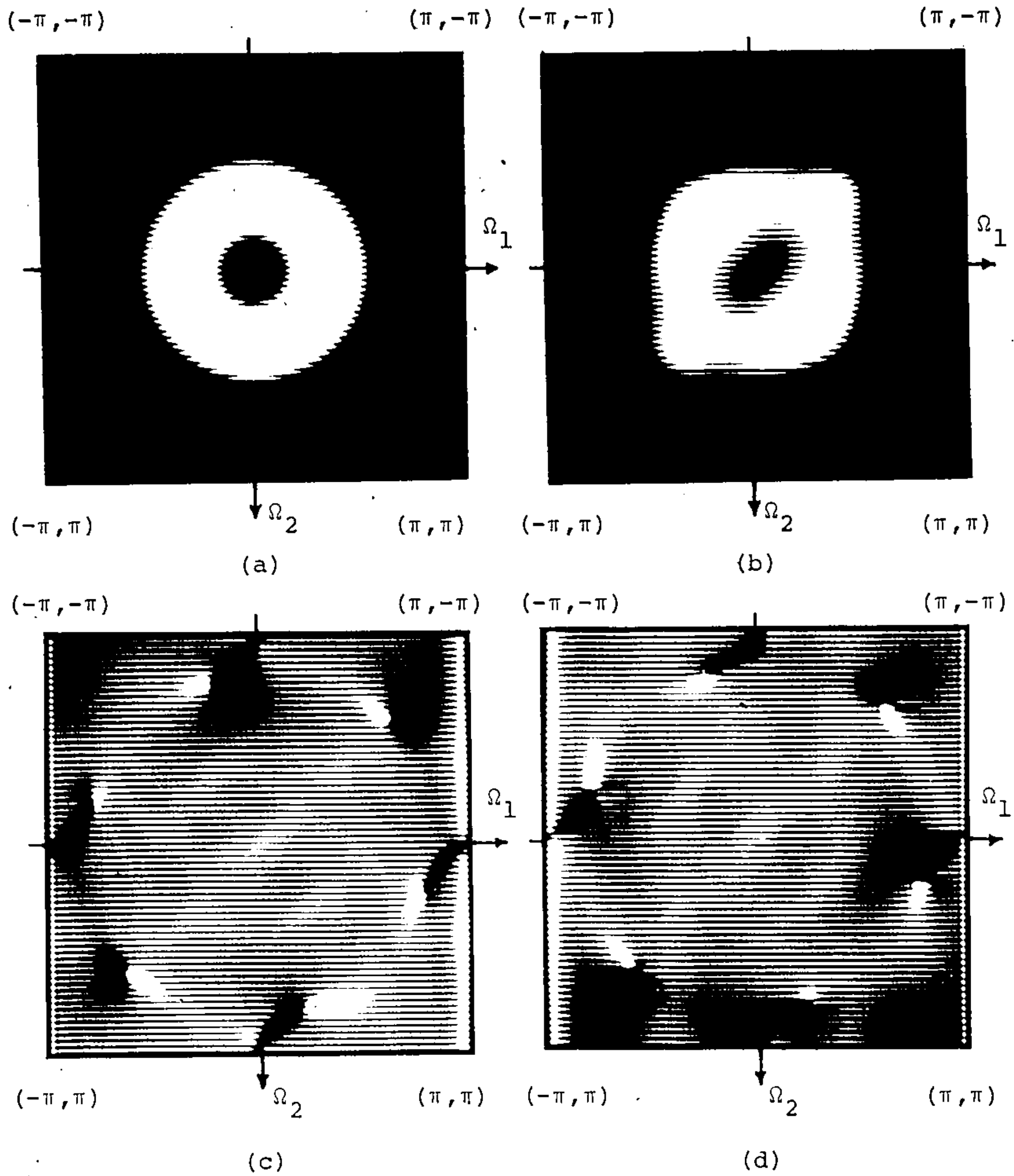
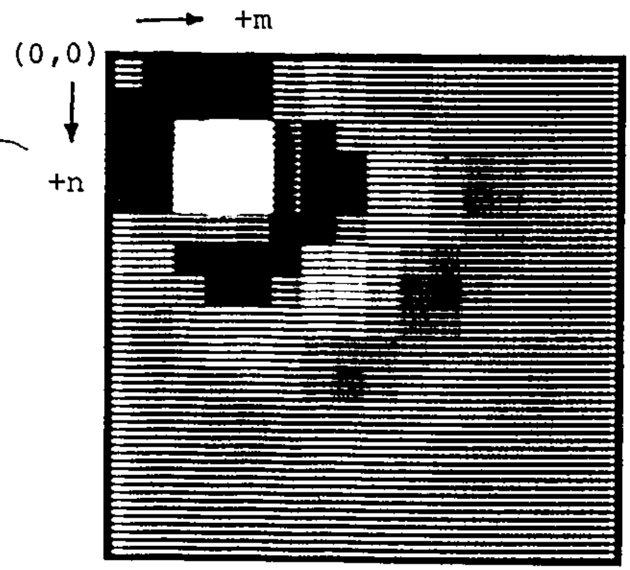


Figure 2.18 a) Desired Magnitude Specifications
b) Designed Magnitude Response
c) & d) Designed Group Delays τ_1 and τ_2



(a)

a_{00} = -57379310E-02	b_{00} = 1.0000000
a_{01} = -13393510E-01	b_{01} = - .75580170
a_{02} = -15518550E-01	b_{02} = .47858320
a_{03} = -30836500E-01	b_{03} = - .13298210
a_{10} = -13391570E-01	b_{10} = - .75583560
a_{11} = -34089480E-01	b_{11} = .44810160
a_{12} = 14897800E-01	b_{12} = - .27361320
a_{13} = -13460580E-01	b_{13} = .10086200
a_{20} = -15515840E-01	b_{20} = .47864050
a_{21} = 14900300E-01	b_{21} = - .27365880
a_{22} = 52191500E-01	b_{22} = .22460670
a_{23} = 8049360E-01	b_{23} = - .95416310E-01
a_{30} = -30835300E-01	b_{30} = - .13299140
a_{31} = -13458980E-01	b_{31} = .10087010
a_{32} = 80496430E-01	b_{32} = - .95393360E-01
a_{33} = 71583510E-01	b_{33} = .20017440E-01

(b)

Figure 2.19 a) First 16 x 16 Pixels of the Impulse Response[+]
b) Coefficients of the Designed Filter.

[+] Each square area of constant intensity represents one pixel.

The order of the filter is such that M1=M2=N1=N2=3. The initial value of τ_c was set at 3 and a lower limit of 1. A good design was obtained for $\tau_c = 3$ for which the value of ξ was a maximum . The error measures for this design are as follows:

$$E_1 = 2.27 \; ; \; E_2 = 0.81 \; ; \; E_3 = 0.67 \; ; \; |\xi| = 9.29 \times 10^{-2}$$

Figure 3.18 shows the desired magnitude and the designed magnitude, group delay responses. Figure 2.19 shows the first 16 x 16 pixels of the impulse response and the coefficients of the designed filter.

Example 3 High Pass Filter Design

The magnitude specifications for this design are as follows:

$$R(\rho) = e^{-k(0.5\pi - \rho)^2} \quad \text{for } 0 \leqslant \rho \leqslant 0.5\pi$$

$$R(\rho) = 1.0 \quad \text{for otherwise (value of } k=5.0)$$

The range of τ_c chosen was such that $1 \leqslant \tau_c \leqslant 3$. The order of the filter is such that M1=N1=M2=N2=3. The design corresponding to $\tau_c = 2$ resulted in a maximum in the value of ξ . The error measures are as follows:

$$E_1 = 0.447 \; ; \; E_2 = 0.613 \; ; \; E_3 = 0.57 \; ; \; |\xi| = 6.19 \times 10^{-2}$$

Figure 2.20 shows the desired magnitude and designed magnitude, group delay characteristics. Figure 2.21 shows the first 8 x 8 pixels of the impulse response and the coefficients of the designed filter.

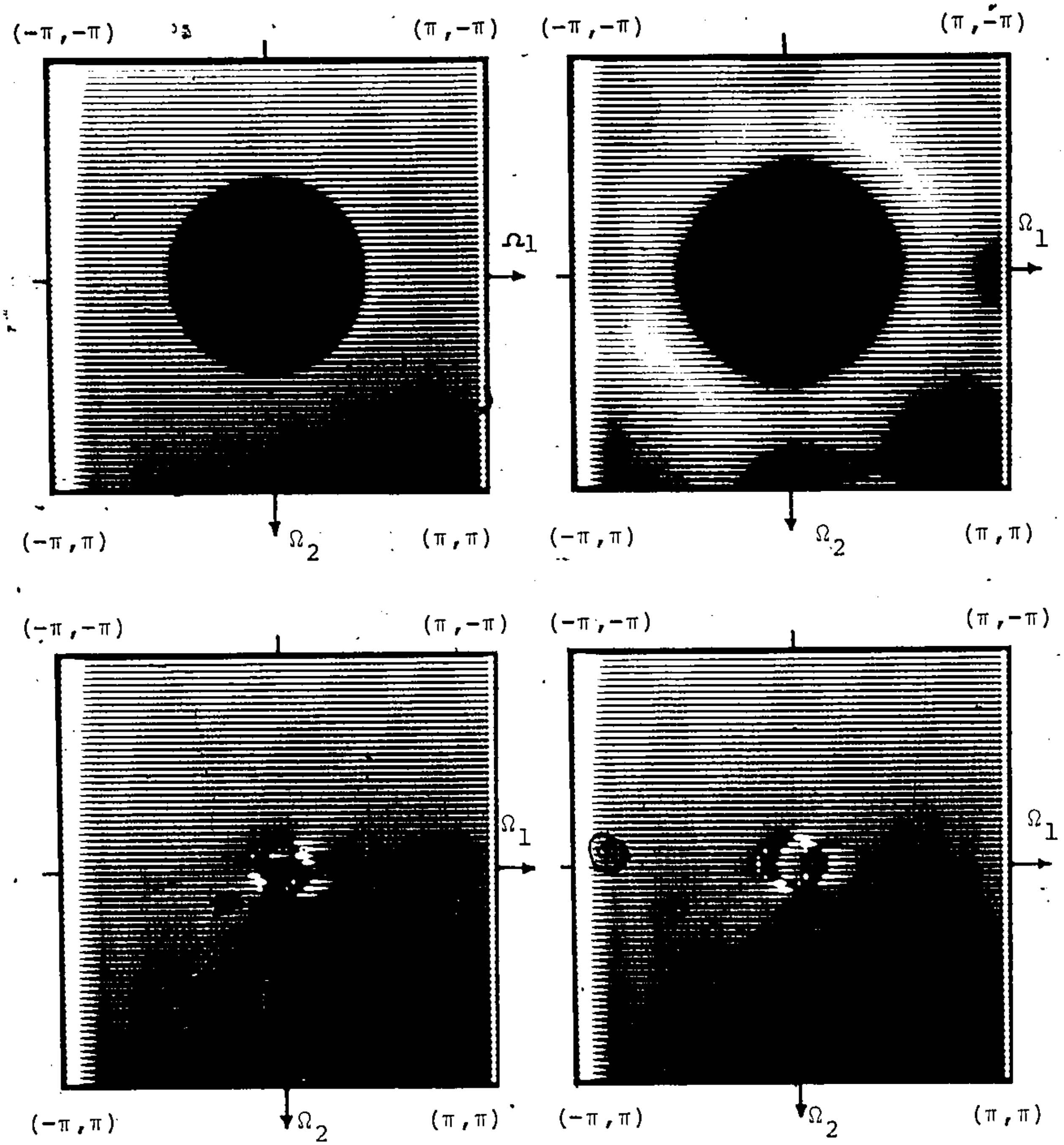
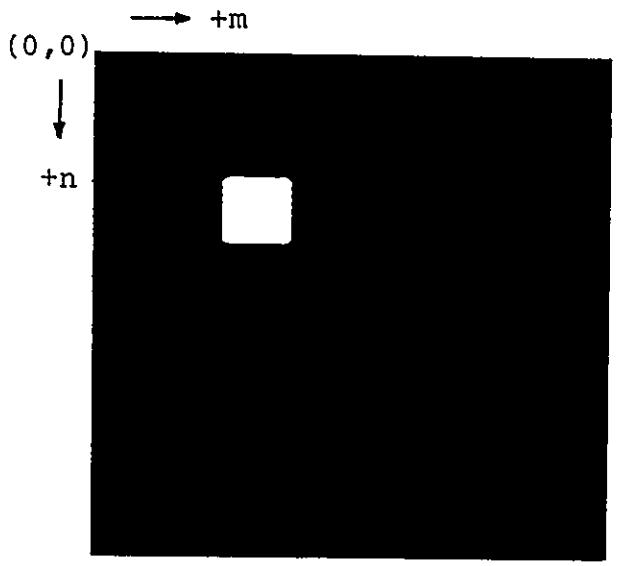


Figure 2.20 a) Desired Magnitude Specifications.
b) Designed Magnitude Response
c) & d) Designed Group Delays τ_1 and τ_2



(a)

a_{00} = -16757070E-01	b_{00} = 1.0000000
a_{01} = -31652120E-01	b_{01} = - .29278870
a_{02} = -37393000E-01	b_{02} = .25963300E-01
a_{03} = -21176130E-01	b_{03} = .73930880E-02
a_{10} = 031652610E-01	b_{10} = - .29278920
a_{11} = -36145740E-01	b_{11} = .75002130E-01
a_{12} = -46683850E-01	b_{12} = .64422040E-02
a_{13} = -38984960E-01	b_{13} = - .73239300E-03
a_{20} = -37387810E-01	b_{20} = .25962280E-01
a_{21} = -46688500E-01	b_{21} = .64420660E-02
a_{22} = 94498280	b_{22} = .27269780E-02
a_{23} = -32254580	b_{23} = - .18330530E-02
a_{30} = -21181930E-01	b_{30} = .73926150E-02
a_{31} = -38980000E-01	b_{31} = - .73249870E-03
a_{32} = -32254740	b_{32} = - .18328830E-02
a_{33} = 43960190E-01	b_{33} = - .41012050E-03

(b)

Figure 2.21 (a) First 16 x 16 Pixels of Impulse Response+
(b) Coefficients of the Designed Filter.

+ Each square area of constant intensity represents one pixel.

2.5 Summary

A linear programming method has been presented for the design of one and two dimensional recursive digital filters to approximate magnitude and phase characteristics. The variables of the linear program are the coefficients of the desired filter and the linear programming problem is set up such that they minimize the real and imaginary components of the complex weighted error.

The types of stability constraints, which are linear in form, are also indicated and are incorporated in the method to design stable filters. The constraints used are sufficient conditions for stability and proofs of their sufficiency are also presented.

The filters are designed in the direct form and examples of design are also presented.

CHAPTER III

RECURSIVE DIGITAL FILTER APPLICATIONS IN IMAGE PROCESSING

3.1 Introduction

Although recursive digital filters are computationally advantageous when compared to most convolutional methods of filtering, (specifically, convolution via FFT), little has been reported regarding the application of these filters in image processing. Lack of application of these filters in the past was primarily due to design and stability problems associated with these filters. In recent years, however, many of the stability and design problems have been overcome, specifically in the case of quarter plane filters and at the present time there are a large number of design methods available, including the one presented in Chapter II, and therefore this chapter considers in detail the applications of quarter plane recursive digital filters to image processing problems in the areas of enhancement and restoration.

A detailed analysis of the usefulness of various two dimensional recursive filter design techniques, indicated in Section 1.2, with regard to their ability to design recursive filters for image processing applications, will not be attempted here, since the task is beyond the scope of this thesis. However, a brief analysis is provided in the following paragraph.

As discussed in Section 1.2, in situations where simultaneous linear phase and arbitrary magnitude approximation are required, the linear programming technique, presented in Chapter II, is more advantageous to use than the group delay (phase) equalization technique of [3], which uses non-linear optimization for approximation. In situations where approximation of arbitrary specifications to the real part of a transfer function or to the magnitude squared transfer function, the non-linear optimization design procedures are more useful than the design techniques of [8] and the technique of Chapter II, which use linear programming. Considering magnitude squared approximations, the linear programming technique of Chapter II simplifies to the design technique of Dudgeon [8]. As discussed in [32], there are severe problems in implementation of the magnitude squared transfer function designed using the method of [8]. In the case of an approximation to the real part of the transfer function, Re $H(Z_1,Z_2)$ †, the coefficients of the numerator of Re $H(Z_1,Z_2)$ are in terms of the product of numerator and denominator coefficients of the original transfer function $H(Z_1,Z_2)$. Such an approximation is therefore, not in a linear form, and cannot be performed using linear programming techniques.

All of the non-linear optimization design techniques discussed in Section 1.2, design filters in cascades of lower order sections and therefore the filters obtained due to these design techniques belong to a subclass of all possible stable realizations.‡ Among these techniques, the most recent and the most improved, is the design procedure

---

† Re $H(Z_1,Z_2)$ refers to real part of filter transfer function, $H(Z_1,Z_2)$.

‡ This is due to the fact that the fundamental theorem of algebraic factorization does not exist in two dimensions.

of [4]. Compared to the remainder of the similar techniques, the technique of [4] designs filters with the general class of low order stable filter sections.

From the brief analysis above, it can be seen that suitable techniques for designing filters for image processing applications are the linear programming technique of Chapter II and the non-linear optimization design procedure of [4]. In the following sections on applications of recursive digital filters to image processing, the linear programming design technique is used in designing filters to simultaneous specification of magnitude and linear phase and the non-linear optimization procedure of [4], which is described in Appendix F, is made use of in approximating to the real part of a transfer function or to the magnitude squared transfer function.

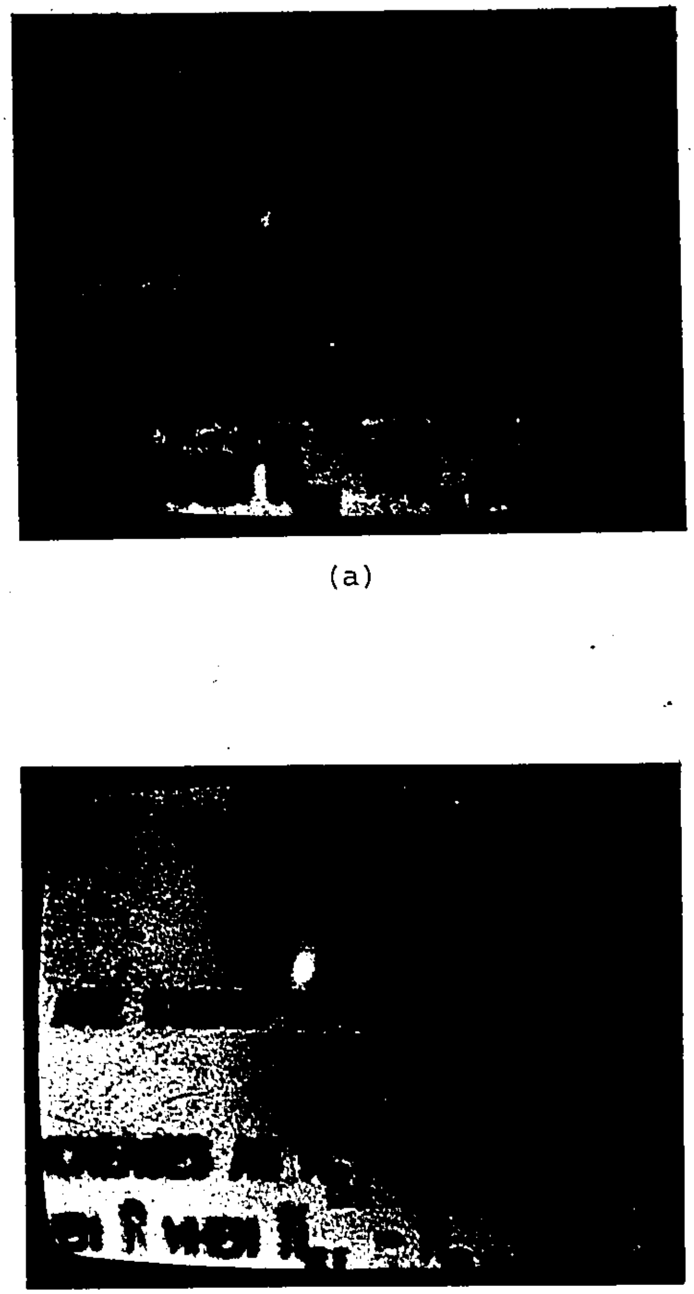
3.2 Application of Recursive Digital Filters to Image Enhancement

In image processing, the familiar enhancement problems are: high frequency emphasis or crispening and edge enhancement. The filters required for these applications have magnitude characteristics that are generally high pass in nature and linear phase characteristics over most of the frequency domain. As indicated earlier, the suitable design technique is the linear programming design procedure, with which the desired specifications can be approximated simultaneously. In the following subsections, the usefulness of this design technique is illustrated by filters designed to prescribed specifications and applied to image processing problems.

3.2.a High Frequency Emphasis or Crispening

Oftentimes it is required to make an image appear sharper. One way of achieving this objective is to pass the image through a high pass filter and thus emphasize the high frequency components of the image. Consider the image of Figure 3.1(a), which is sampled using the flying spot scanner. Referring to Appendix G (second section), the intensity of each pixel of the sampled image is the convolution of the original image with the CRT (cathode ray tube) beam spot of the flying spot scanner. The sampled image can also be thought of as the output of a filter, with the original image as the input and the transfer function of the filter equal to the fourier transform of CRT spot of light incident on the film. The filter transfer function is low pass in characteristics, as the intensity profile of CRT spot is gaussian in nature. The band width of this filter is directly related to the aperature of the CRT spot. If the CRT beam is out of focus, then the aperature of the spot incident on the film is larger than normal and therefore the band width of the low pass filter becomes very small. This causes severe attenuation of mid and higher frequency components of the image and results in a blurred image as shown in Figure 3.1(a).

Blurred images similar to the one shown in Figure 3.1(a) can be enhanced by passing the image through a high pass filter, which compensates for the attenuation of higher frequency components. For the image of Figure 3.1(a), a high pass



(a)

(b)

Figure 3.1 a) Original Image
b) Enhanced Image.

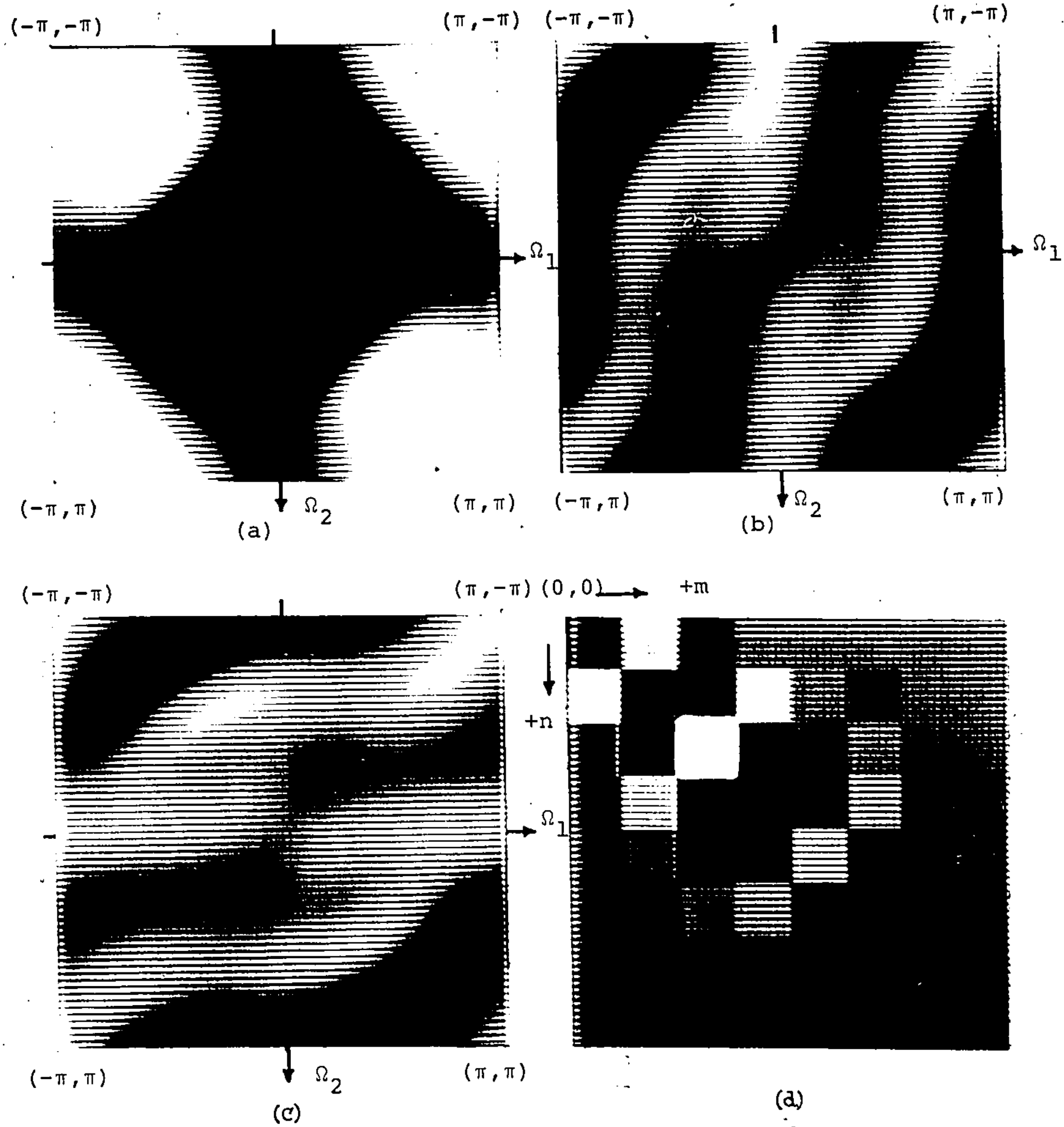


Figure 3.2 a) Designed High Pass Filter Magnitude Response
b) & c) Designed Group Delay Responses τ_1 and τ_2
d) First 8 x 8 Pixels of the Impulse Response+

+ Each square area of constant intensity represents one picture element.

$a_{00} = -.15575530E-01$	$b_{00} = 1.0000000$
$a_{01} = .46834400E-01$	b_{01} .22357600
$a_{02} = -.39941100E-02$	b_{02} .71496190E-01
$a_{10} = .49515360E-01$	b_{10} .22108698
$a_{11} = -.21031310E-01$	b_{11} .15445120
$a_{12} = -.22371150$	b_{12} .10571910
$a_{20} = .41128100E-02$	b_{20} .91730540E-01
$a_{21} = -.23362350$	b_{21} .10200290
$a_{22} = .70751300$	b_{22} .15625000E-01

Figure 3.2(e) Coefficients of the Designed Filter.

filter was designed with arbitrarily chosen specifications. The gain of the filter was specified to be equal to 1 beyond the radial frequency of 0.7π, and 0.1 below the radial frequency of 0.3π. The transition band in between 0.3π and 0.7π was gaussian. The magnitude, group delay and the impulse response of the designed filter are shown in Figure 3.2. The coefficients of the designed filter is shown in Figure 3.3. The high pass filtered image of 3.1(a) is shown in Figure 3.1(b). From this figure, it is clear that the image is much sharper compared to the original and the details which are blurred in the original are much clearer in the filtered version.

3.2.b Edge Enhancement

One of the approaches to picture segmentation is based on the detection of discontinuities; i.e., lines along which there is an abrupt change in gray level, indicating the end of one region and the beginning of another. Such discontinuities are called edges. Prior to the detection of these edges, it is required that they be enhanced. The filter that is often used for edge enhancement is the Laplacian [36,37]. In continuous space, it is defined as:

$$\nabla^2 f(x,y) = \frac{\partial^2 f(x,y)}{\partial x^2} + \frac{\partial^2 f(x,y)}{\partial y^2} \qquad (3.2.1)$$

where $f(x,y)$ is the image intensity which is a function of spatial coordinates x and y. The fourier transform of (3.2.1) is given by:

$$L(w_1,w_2) = -(w_1^2 + w_2^2)\cdot F(w_1,w_2) \qquad (3.2.2)$$

where $F(w_1,w_2)$ is the fourier transform of the image $f(x,y)$ and w_1 and w_2 are continuous frequency variables. Thus, the laplacian operator can be represented in terms of a linear shift invariant system possessing a transfer function:

$$H(w_1,w_2) = -(w_1^2 + w_2^2) \qquad (3.2.3)$$

Assuming w_s to be the sampling frequency at which the image is sampled along both the spatial directions, the frequency response of the laplacian filter can be written as:

$$H(\Omega_1,\Omega_2) = -(\Omega_1^2 + \Omega_2^2) \qquad (3.2.4)$$

where $\Omega_1 = w_1\pi/(w_{s/2})$ and $\Omega_2 = w_2\pi/w_{s/2})$ are the normalized frequency variables. The negative sign in (3.2.4) can be neglected, since it simply involves multiplying the input or the output by -1. Therefore the edge enhancement filter can be written as:

$$H(\Omega_1,\Omega_2) = (\Omega_1^2 + \Omega_2^2) \qquad (3.2.5)$$

The phase characteristics associated with the transfer function $H(\Omega_1,\Omega_2)$ of (3.2.5) are zero or linear phase. A filter was designed to approximate the magnitude characteristics of (3.2.5) with linear phase using the linear programming technique. The desired magnitude characteristics are shown in Figure 3.3(a). The designed filter magnitude, group delay and impulse response characteristics and the coefficients are shown in Figure 3.3. This filter was then applied to enhance the edges of the image shown in Figure 3.4(a). The result of enhancing the edges is shown in Figure 3.4(b). It is obvious that there is indiscriminate enhancement of all edges of the image; the desired result.

For the purpose of comparison, a recursive filter

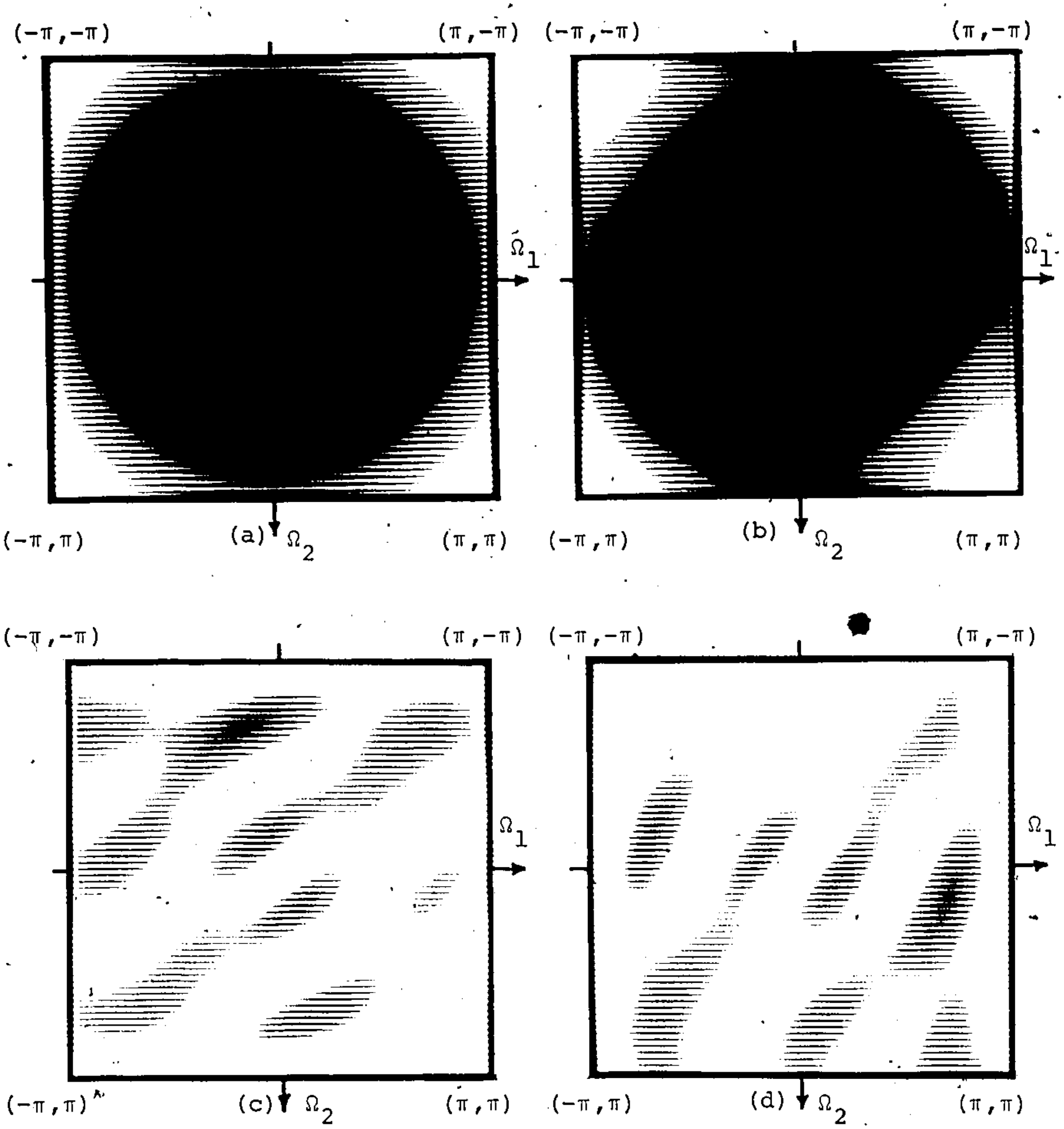
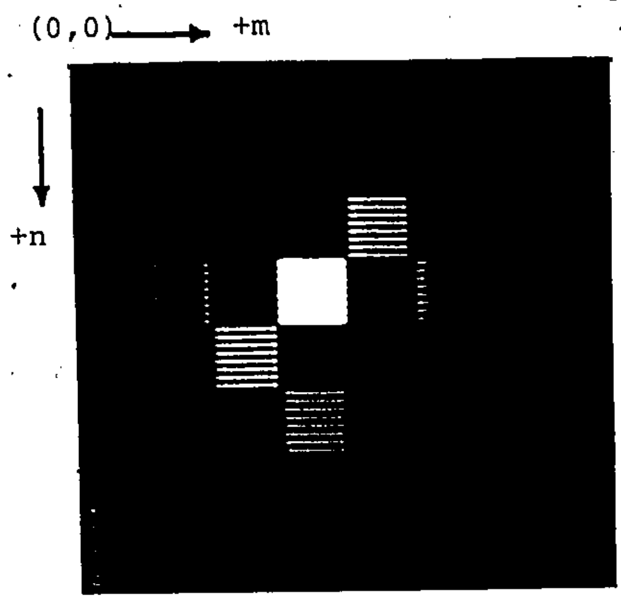


Figure 3.3 (a) Desired Magnitude Specifications
(b) Designed Magnitude Response
(c) & (d) Designed Group Delays, τ_1 and τ_2



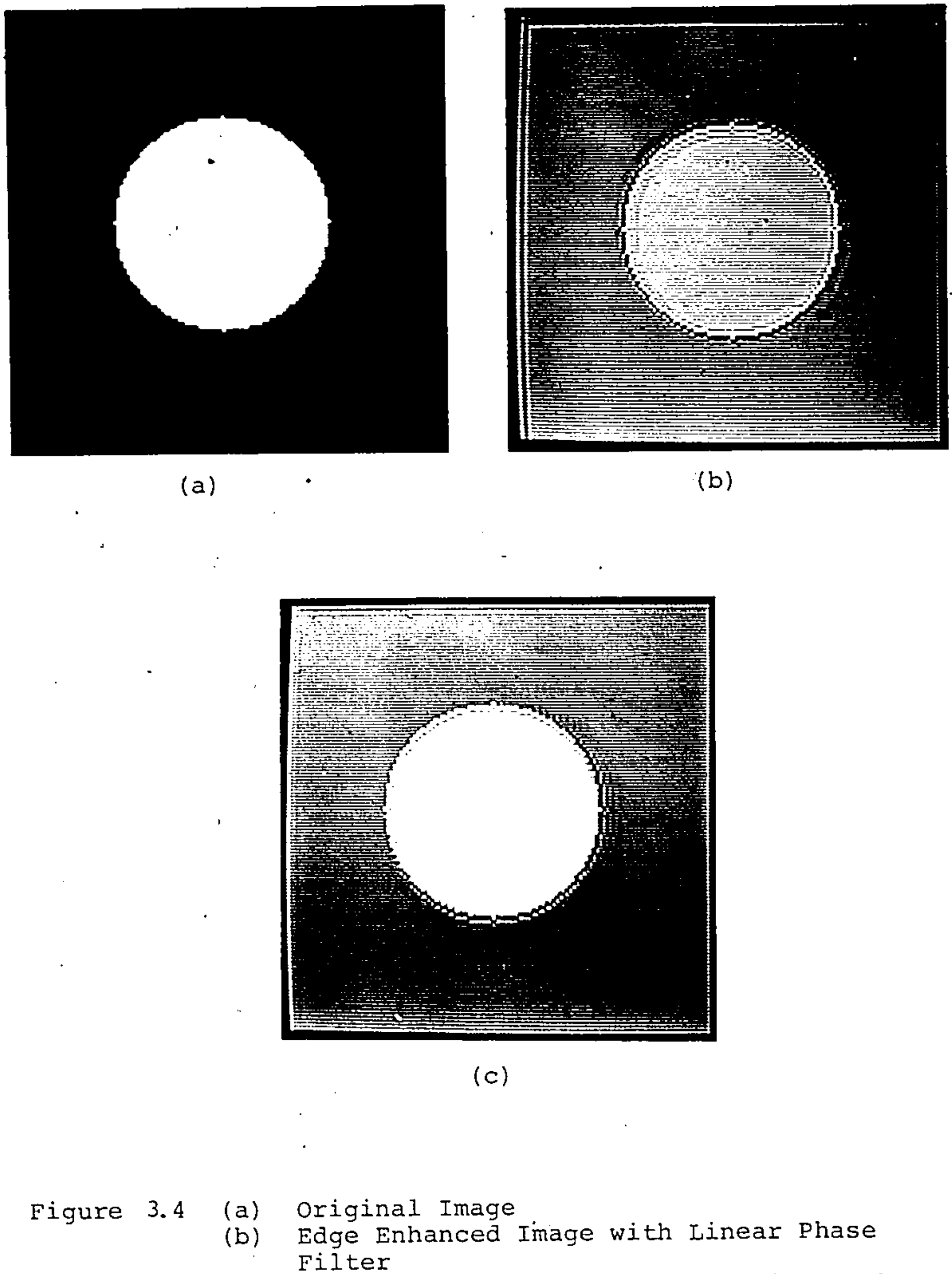
(e)

a_{00}	= -.38311810E-02	b_{00}	= 1.0000000
a_{01}	= -.22786600E-02	b_{01}	= .36874460
a_{02}	= .57360680E-02	b_{02}	= .56907850E-01
a_{03}	= -.38473030E-02	b_{03}	= .51248820E-01
a_{10}	= -.22166630E-02	b_{10}	= .36902940
a_{11}	= .92594360E-02	b_{11}	= .30360050
a_{12}	= .16063760E-04	b_{12}	= .10230900
a_{13}	= .17943920E-01	b_{13}	= .47874560E-01
a_{20}	= -.57588740E-02	b_{20}	= .56607170E-01
a_{21}	= -.76285960E-04	b_{21}	= .10248220
a_{22}	= .52557700E-03	b_{22}	= .10201330
a_{23}	= -.10184440	b_{23}	= .57075340E-01
a_{30}	= -.39055160E-02	b_{30}	= .51260240E-01
a_{31}	= .18055570E-01	b_{31}	= .48052420E-01
a_{32}	= -.10179080	b_{32}	= .57397140E-01
a_{33}	= .25666270	b_{33}	= .40963040E-01

(f)

Figure 3.3 (e) First 8 x 8 Pixels of the Designed Filter Impulse Response[+]
(f) Coefficients of the Designed Filter

[+]Each square area of constant intensity represents one pixel.



(a)

(b)

(c)

Figure 3.4 (a) Original Image
(b) Edge Enhanced Image with Linear Phase Filter
(c) Edge Enhanced Image with Non-Linear Phase Filter.

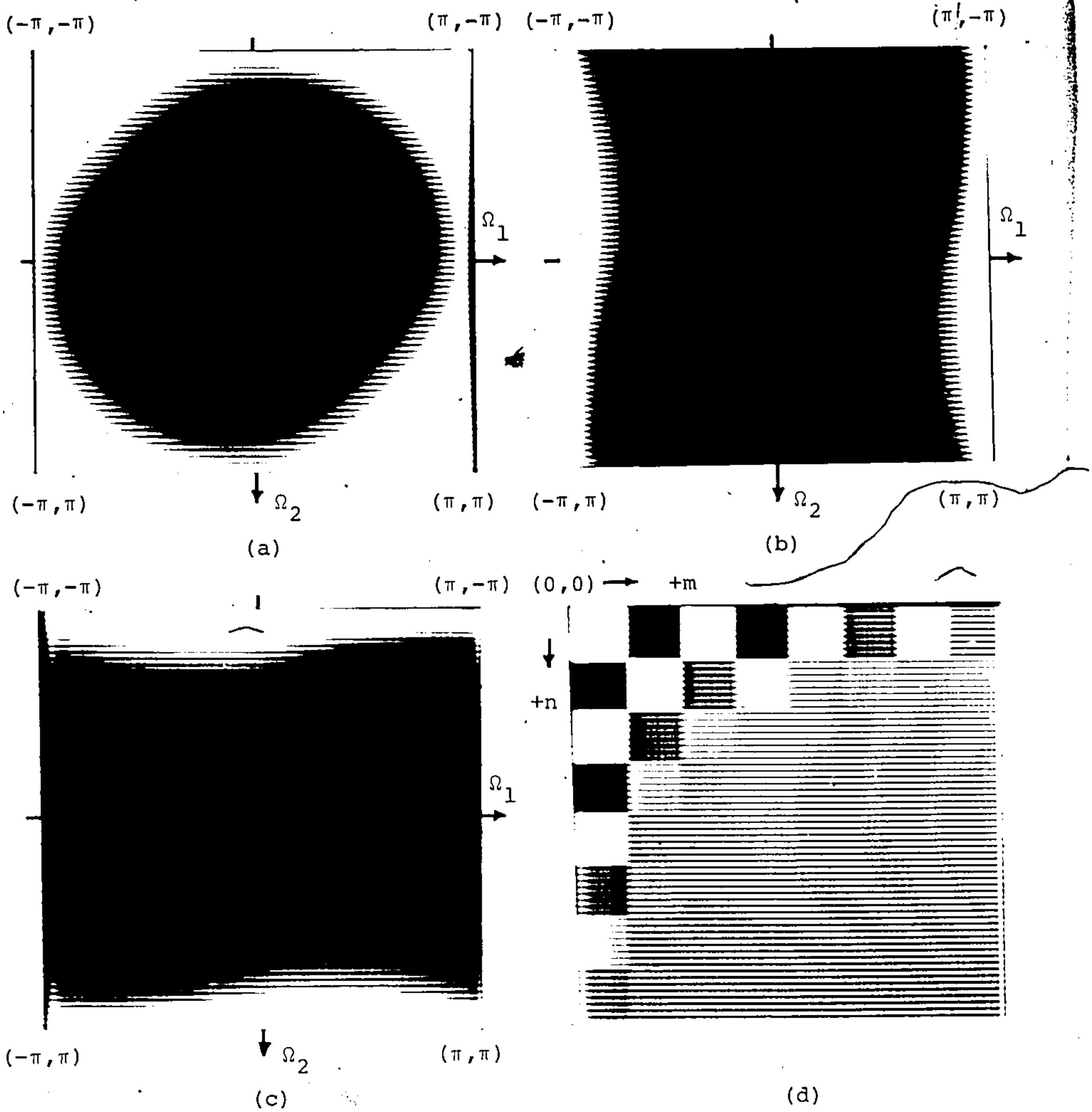


Figure 3.5 a) Magnitude Response
b) & c) Group Delays τ_1 and τ_2
d) Impulse Response[+]

[+]Each square area of constant intensity represents one pixel.

a_{00}	= 1.0000000	b_{00}	= 1.0000000
a_{01}	= .33264320	b_{01}	= .93398490
a_{02}	= − .14728560	b_{02}	= .17348960
a_{10}	= .33264320	b_{10}	= .93398490
a_{11}	= − .44744180E-01	b_{11}	= .90195800
a_{12}	= − .11984560	b_{12}	= .18074860
a_{20}	= − .14728560	b_{20}	= .17348960
a_{21}	= − .11984560	b_{21}	= .18074860
a_{22}	= − .63393340E-02	b_{22}	= .42728630E-01

Figure 3.5(e) Coefficients of the Filter.

was designed in which the magnitude alone was given consideration. The characteristics of the designed filter and the coefficients are shown in Figures 3.5. Using this filter, the image of Figure 3.4(a) was filtered to obtain an enhanced image of 3.4(c). The consequences of neglecting the phase characteristics in the design are obvious in the enhanced image.

3.3 Application of Recursive Digital Filters in Image Restoration

The problem of image restoration of degraded images deals with the removal of the sources of degradation. The type of degradation can be spatially varying or spatially invariant. In this section, we will consider using recursive filter implementations for image restoration, where image degradations are of the latter type. The most common types of degradations are:

a) Motion Blur

b) Focus Blur

c) Atmospheric Turbulance Blur

With the assumption of spatial invariance, the degrading system, corresponding to the three types of blur can be modeled as shown in Figure B2 of Appendix B. In this figure, g(x,y) is the blurred image and h(x,y) is the point spread function corresponding to the type of blur in question.

In order to restore blurred images, the first task is to identify the blur and the second to generate the corresponding restoration or inverse filter transfer function. The details of the blur point spread function identification will not be given here, since a complete description of identification of the three different types of blurs is found in the work of Cannon [23]. The generation of restoration filter response is presented in the next section. In the work reported by Cannon and others, the inverse filtering is carried out by using non-recursive methods, specifically, by the use of fast fourier transform (FFT). Here, we consider performing inverse filtering using recursive digital filters whose response has been approximated to the desired restoration filter characteristics. For the purpose of comparison, the inverse filtering was also carried out via convolution using the FFT. The restoration filter (i.e., inverse filter) impulse response was windowed by a two dimensional Hamming window, thus restricting the convolution kernal to the chosen size of the window. For the extent of blurs considered in all the examples in this thesis, a kernal size of 32 x 32 was found to be appropriate. Larger kernal sizes did not appear to improve the restorations significantly. However, if the extent of blur is to be increased to larger values than what is considered here, the kernal sizes would have to be increased further. The convolution via the FFT was implemented using the overlap and save method [1], with an FFT array size equal to 64 x 64. Since the input to the FFT is complex, the convolution can be carried out for two 32 x 32 sections of image data at a time.

The original images used in the examples were sampled from photographs. The blurring of the images was performed by the computer, from known blurring functions and computer generated noise (wide band gaussian) was added to the blurred images, with a specified signal to noise ratio (SNR). The two dimensional recursive digital filters used in the inverse filtering of the blurred images were designed using either the cascaded design procedure (described in Appendix F) or the linear programming technique described in Chapter II, depending on the type of approximation at hand.

3.3.a Restoration Filter Transfer Functions

Consider, the block diagram of Figure B2 in Appendix B. Let b(x,y) be the intensity of the blurred image. Therefore, from B3:

$$b(x,y) = h(x,y) \circledast i(x,y) + n(x,y) \qquad (3.3.1)$$

where h(x,y) is the impulse response of the blurring system. The fourier transform of the blurred image can therefore be written as:

$$B(w_1,w_2) = H(w_1,w_2) \cdot I(w_1,w_2) + N(w_1,w_2) \qquad (3.3.2)$$

where w_1 and w_2 are the frequency variables in the two dimensional frequency domain. B, H, I and N are the fourier transforms of b, h, i and n respectively. If w_s is the frequency at which the image is sampled along both the spatial directions, then (3.3.2) can be written as:

$$B(\Omega_1,\Omega_2) = H(\Omega_1,\Omega_2) \cdot I(\Omega_1,\Omega_2) + N(\Omega_1,\Omega_2) \qquad (3.3.3)$$

where $\Omega_1 = w_1\pi/(w_{s/2})$ and $\Omega_2 = w_2\pi/(w_{s/2})$ are the normalized frequency variables. From (3.3.3), the power spectrum of $b(x,y)$ can be written as:

$$P_B(\Omega_1,\Omega_2) = |H(\Omega_1,\Omega_2)|^2 \cdot P_I(\Omega_1,\Omega_2) + P_N(\Omega_1,\Omega_2) \tag{3.3.4}$$

where P_B, P_I and P_N are the power spectra of b, i and n respectively.

Consider now, a blurred image that is free of noise. For this case, $N(\Omega_1,\Omega_2)$ in (3.3.3) is equal to zero. Therefore, the spectrum of the original image can be obtained in a straight forward manner, as:

$$I(\Omega_1,\Omega_2) = B(\Omega_1,\Omega_2)/H(\Omega_1,\Omega_2) \tag{3.3.5}$$

provided $H(\Omega_1,\Omega_2)$ is known and invertible. Thus the simple restoration, or inverse filter, would be:

$$H_R(\Omega_1,\Omega_2) = \frac{1}{H(\Omega_1,\Omega_2)} \tag{3.3.6}$$

With the knowledge of the type of blur, the simple restoration filter $H_R(\Omega_1,\Omega_2)$ can be computed and in the ideal situation (when the blurred image is noise free), the original image can be obtained from the blurred image. However, in the presence of noise, the simple restoration filter would yield an inferior restoration as it tends to amplify the noise which dominates the higher frequency components of the image. Thus, one has to consider restoration filters which take into account the presence of noise in the image.

In the literature, the restoration filters, for deblurring blurred noisy image, have been derived based on two criteria. These are the minimum mean squared error criteria (MMSE) and the power spectrum equivalization criteria (PSEC). The former criterion can be written as:

Minimize

$$E\,\{i(x,y) - \hat{i}(x,y)\}^{2\dagger} \quad \text{over all } x \text{ and } y \tag{3.3.7}$$

where $i(x,y)$ is the original image and $\hat{i}(x,y)$ is the restored image estimate. A restoration filter based on this criterion is given by [38,39].

$$H_{R_1}(\Omega_1,\Omega_2) = \frac{H^*(\Omega_1,\Omega_2)\cdot P_I(\Omega_1,\Omega_2)}{|H(\Omega_1,\Omega_2)|^2\cdot P_I(\Omega_1,\Omega_2) + P_N(\Omega_1,\Omega_2)} \tag{3.3.8}$$

where P_I and P_N are the power spectra of the original image and noise. The second criterion suggested and used by Cannon and Stockham [23,39], can be stated as:

$$P_B(\Omega_1,\Omega_2) \cdot |H_{R_2}(\Omega_1,\Omega_2)|^2 = P_I(\Omega_1,\Omega_2) \tag{3.3.9}$$

where H_{R_2} is the desired restoration filter and P_B and P_I are the power spectra of the blurred and original images respectively. The restoration filter can then be written as:

$$H_{R_2}(\Omega_1,\Omega_2) = \sqrt{\frac{P_I(\Omega_1,\Omega_2)}{P_B(\Omega_1,\Omega_2)}} \cdot e^{-j\phi} \quad ;$$

$$\phi = \tan^{-1}\left(\frac{\text{Imag } H(\Omega_1,\Omega_2)}{\text{Real } H(\Omega_1,\Omega_2)}\right) \tag{3.3.10}$$

†E{ } refers to the expected value of the quantity inside the parenthesis.

* denotes conjugation.

and as indicated earlier, P_B is given by:

$$P_B(\Omega_1,\Omega_2) = P_I(\Omega_1,\Omega_2) \cdot |H(\Omega_1,\Omega_2)|^2 + P_N(\Omega_1,\Omega_2)$$

In (3.3.8) and (3.3.10), the function $H(\Omega_1,\Omega_2)$ represents the blurring transfer function. Also, it should be noted that in (3.3.8) and (3.3.10) the phase response of $H_{R_1}(\Omega_1,\Omega_2)$ and $H_{R_2}(\Omega_1,\Omega_2)$ are identical to the phase response of $H(\Omega_1,\Omega_2)$.

In the examples considered in this thesis, since the image blurrings are performed by the computer, the quantities on the right hand side of (3.3.8) and (3.3.10) are readily available to compute the restoration filters H_{R_1} and H_{R_2}. In actual situation, where only the blurred image is available, the blurring transfer function $H(\Omega_1,\Omega_2)$ has to be determined from the blurred image itself and the details of this procedure can be found in [23]. Also the power spectrum $P_I(\Omega_1,\Omega_2)$ of the original image is required in computing H_{R_1} and H_{R_2}, and is unknown, when only the blurred image is available. However, it has been suggested, and shown, by Cannon [23], that good restorations can be obtained where the power spectrum of an image, which is statistically similar to the original image, is substituted for $P_I(\Omega_1,\Omega_2)$. It has also been shown by Cannon [23], that the restoration filter H_{R_2} (based on PSEC) provides restorations that are somewhat superior to that of H_{R_1} (based on MMSEC), for the case of motion and focus blurs. However, in the case of atmospheric turbulance blur, H_{R_1} has been shown to handle the situation more effectively compared to H_{R_2}. Therefore, in the application examples shown, H_{R_2} has been used for motion and focus blur restorations and H_{R_1} for atmospheric turbulance blur restorations.

3.3.b Uniform Motion Blur

Subjecting a point source of light to a camera which is operated while in uniform motion produces a streak in the resulting picture. Plotting the intensity of this image, as a function of spatial coordinates, results in a rectangular function in the direction of the blur; as shown in Figure 3.6. This function is the impulse response (point spread function) of the blurring system and is given by:

$$
\begin{aligned}
h(r,\phi) &= \frac{1}{d} \; ; \text{ for } r \leqslant \frac{d}{2} \text{ and } \phi = \theta \\
& \qquad\qquad r = \sqrt{x^2 + y^2} \qquad\qquad (3.3.11) \\
&= 0 \text{ otherwise}
\end{aligned}
$$

where x and y are spatial coordinates and θ is the angle with respect to the x axis along which the camera motion has taken place. The fourier transform of this function has the form of a sinx/x function in the direction of blur and is constant in the direction perpendicular to it. The fourier transform is given by:

$$H_M(w_1,w_2) = \frac{\sin(wd/2)}{(wd/2)} \quad ; \; w = w_1\cos\theta + w_2\sin\theta \qquad (3.3.12)$$

where w_1 and w_2 are spatial frequency variables. If we let Δ be the sampling interval, chosen for sampling the images along both the spatial directions, then the blur length d can be expressed in terms of Δ as:

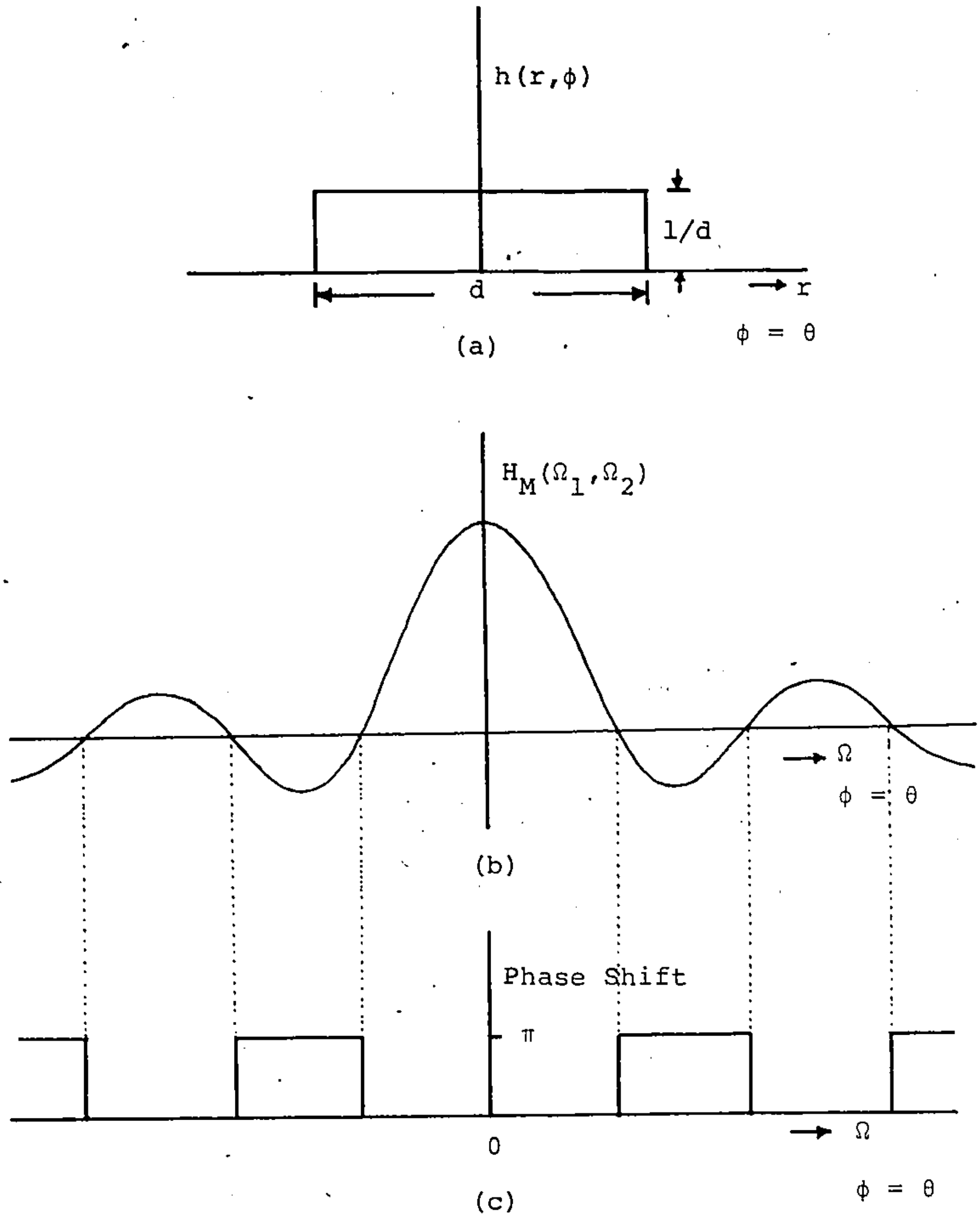


Figure 3.6 (a) Motion Blur Spatial Response
(b) Motion Blur Transfer Function
(c) Amount of Phase Shift Introduced by (b)

$$d = k\Delta \tag{3.3.13}$$

where k is any positive number. Let the sampling frequency in radians be w_s, so that:

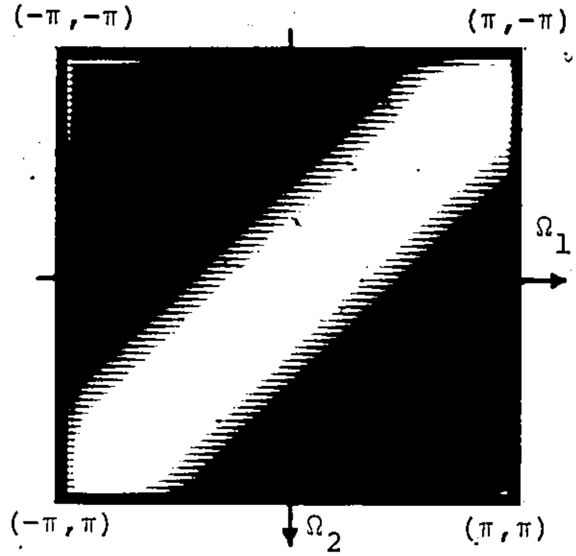
$$w_s = \frac{2\pi}{\Delta} \tag{3.3.14}$$

Therefore (3.3.2) can be rewritten as:

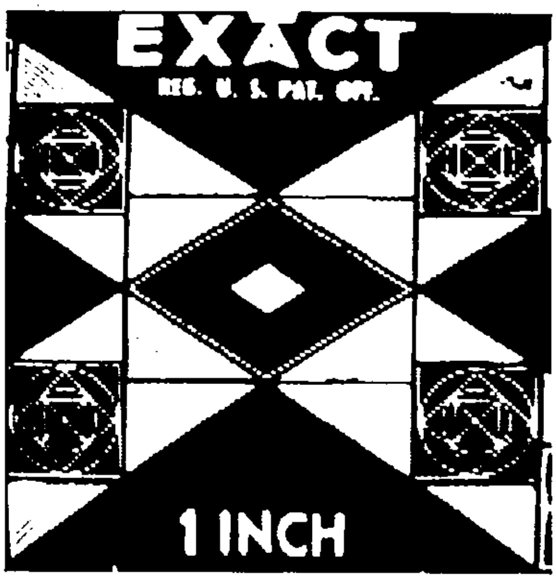
$$H_M(\Omega_1,\Omega_2) = \frac{\sin \Omega k/2}{\Omega k/2} \; ; \; \Omega = \Omega_1 \cos\theta + \Omega_2 \sin\theta \tag{3.3.15}$$

where $\Omega_1 = \frac{w1\pi}{(w_{s/2})}$ and $\Omega_2 = \frac{w2\pi}{(w_{s/2})}$ are the normalized frequency variables. The plot of $H_M(\Omega_1,\Omega_2)$ along the direction of the blur is shown in Figure 3.6(b). In (3.3.13), k is referred to as the blur length in pixels . The motion blurring transfer function of (3.3.15) is purely real and it attenuates the higher frequency components of the image. Also, the alternate side lobes of the transfer function are negative and this introduces a phase shift of π radians to the frequencies that lie in those negative regions, as shown in Figure 3.6(c).

Figure 3.7(a) shows the computer generated motion blurring transfer function for a blur length of 4 pixels. The blur is along a direction which makes an angle $\theta = 45^o$ with respect to the x axis. An original image shown in Figure 3.7(b) is blurred using the above transfer function. The blurring is performed by convolution via the FFT, with kernal size of 32 x 32 pixels. Noise is then added to the blurred image such that the SNR is equal to 30 db. The resulting



(a)

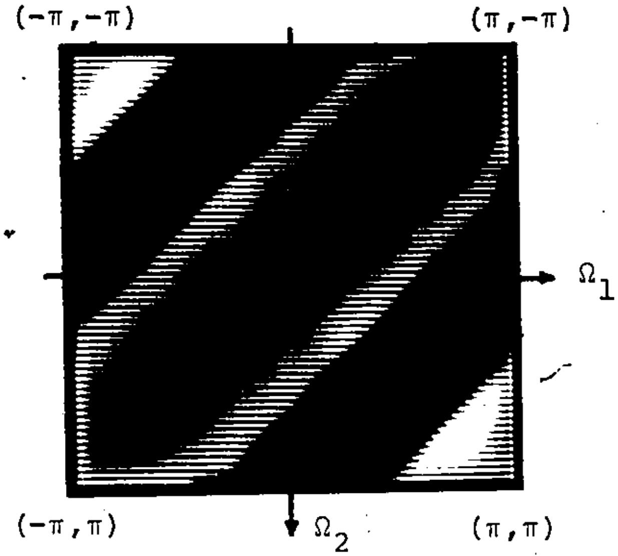


(b) (c)

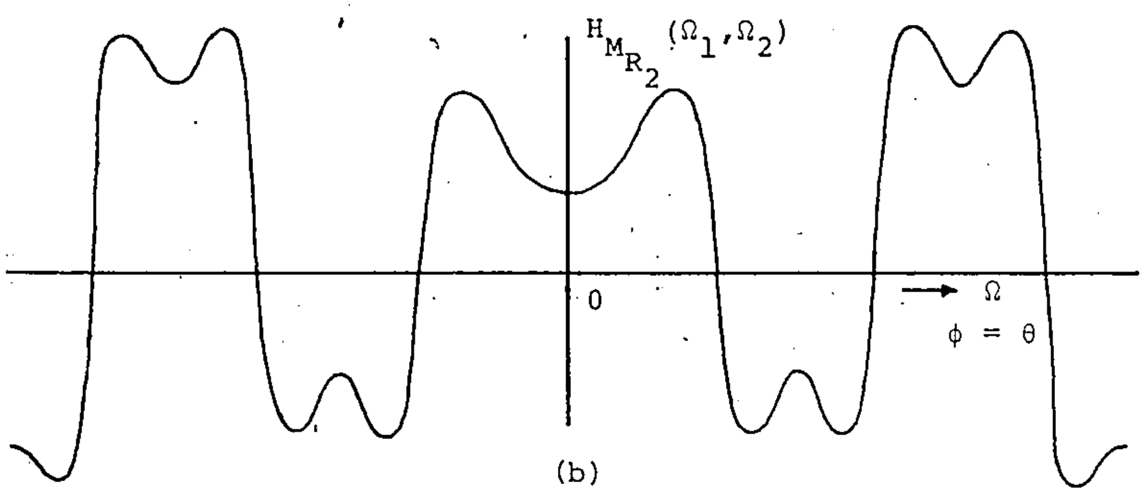
Figure 3.7 (a) Blurring Transfer Function. Blur Length=4 Pixels, Angle=45 degrees w.r.t. Horizontal.
(b) Original Image
(c) Image Blurred by (a), With Noise, SNR=30 db.

blurred, noisy image is shown in Figure 3.7(c). For this blurred image, a restoration filter response $H_{MR_2}(\Omega_1,\Omega_2)$ was computed based on the power spectrum equalization criteria. The response is purely real and is shown in Figure 3.8(a). An approximate form of the response $H_{MR_2}(\Omega_1,\Omega_2)$, of the restoration filter along the direction $\theta = 45^o$ is shown in Figure 3.8(b). In Figure 3.8(a), the intense dark regions and intense bright regions correspond to the most positive and most negative values in Figure 3.8(b). The restoration filter magnitude characteristics is simply the absolute value of $H_{MR_2}(\Omega_1,\Omega_2)$ and the phase characteristics is as shown in Figure 3.8(c), which is identical to .6(c).

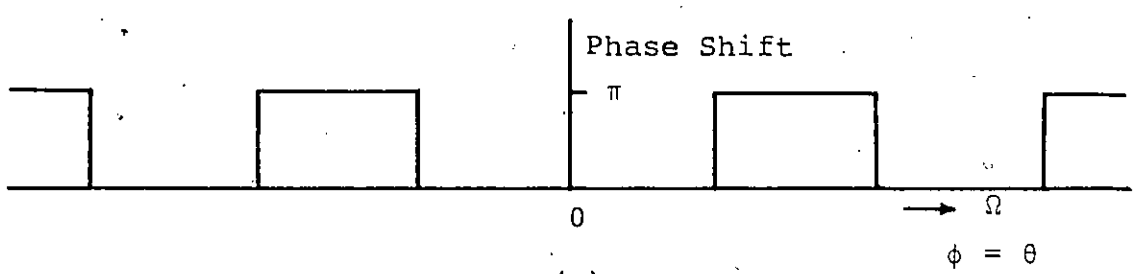
From Figure 3.8(c), it is clear that the phase characteristics is discontinuous in nature; therefore, a two dimensional recursive filter implementation having causal impulse response cannot be used to realize the motion blur restoration filter. This would become clear if the spatial delay characteristics corresponding to the desired phase response are considered. From Figure 3.8(c), at points where the phase characteristics is discontinuous, the corresponding spatial delay would be an impulse and such a delay characteristics cannot be realized by a two dimensional recursive filter. A more successful approach, to realize the desired restoration filter response of Figure 3.8(a) by a recursive filter is to use the implementation of Figure 3.9(a) where $H_1(Z_1,Z_2)$ is a two dimensional recursive digital filter transfer function. The overall frequency response;



(a)



(b)



(c)

Figure 3.8 (a) Restoration Filter Response (Origin is at the Centre of the Image)
(b) Restoration Filter Response Along $\theta = 45^{\circ}$
(c) Phase Shift Introduced by (b)

$H(\Omega_1,\Omega_2)$ of this implementation is given by:

$$H(\Omega_1,\Omega_2) = H_1(Z_1,Z_2) + H_1^*(Z_1,Z_2)^{\dagger} \qquad (3.3.16)$$

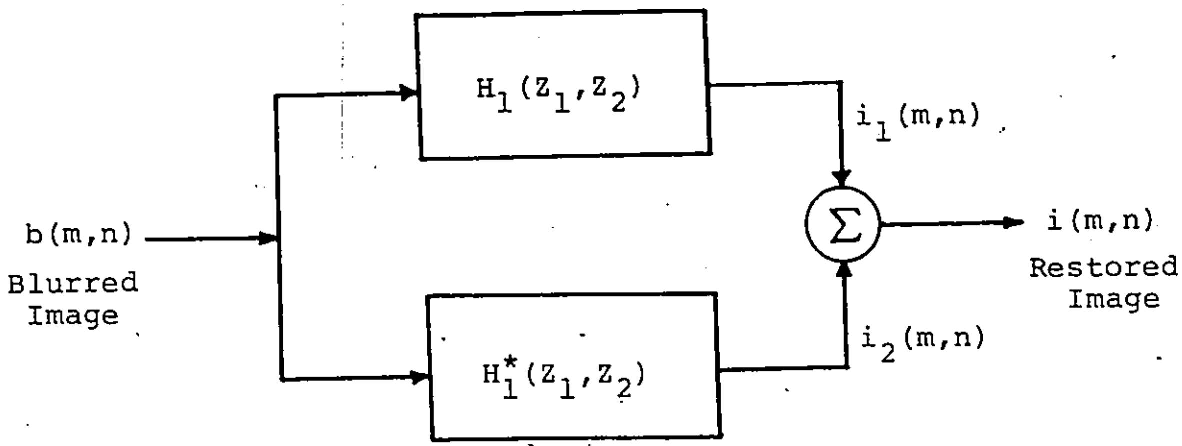
$$= 2\ \text{Real}\{H_1(Z_1,Z_2)\}; \qquad (3.3.17)$$

$$Z_1 = e^{-j\Omega 1}\ ,\ Z_2 = e^{-j\Omega 2}$$

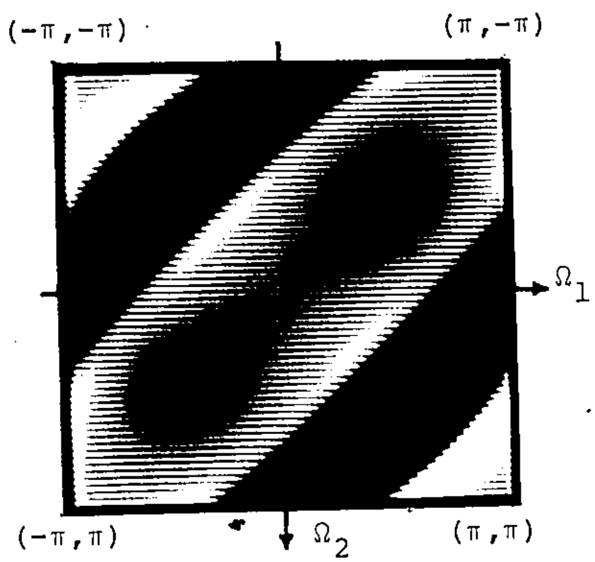
$H(\Omega_1,\Omega_2)$ is purely real and can assume both positive and negative values; therefore, it is now possible to approximate the desired restoration filter response by the recursive filter implementation of Figure 3.9(a). In (3.3.16), the impulse response of $H_1(Z_1,Z_2)$ is causal and the impulse response of $H_1^*(Z_1,Z_2)$ is the same as that of $H_1(Z_1,Z_2)$, but non-causal. Therefore, the overall impulse response of $H(\Omega_1,\Omega_2)$ is non-causal.

The restored image, i, is the sum of the outputs i_1 and i_2 of $H_1(Z_1,Z_2)$ and $H_1^*(Z_1,Z_2)$, respectively. The output array, i_1, is obtained by filtering the input array, b, with $H_1(Z_1,Z_2)$, starting from the upper left hand corner of the input array. As shown in Chapter I, the filtering operation is carried out via the difference equation relating to $H_1(Z_1,Z_2)$. The input array, i_2, is also obtained by filter- the input b by $H_1(Z_1,Z_2)$, but starting from the diagonally opposite corner of the input array. The i_2 thus obtained is

† * denotes conjugation



(a)



(b)

Figure 3.9 (a) Recursive Filter Implementation for Motion Deblur
(b) Frequency Response of (a) (Origin is at the Centre).

equivalent to filtered output of $H_1^*(Z_1,Z_2)$.†

Figure 3.9(b) shows the approximation of $H(\Omega_1,\Omega_2)$ to the desired restoration filter response of Figure 3.8(a). The impulse response corresponding to the desired restoration filter is shown in Figure 3.10(b)‡. Figure 3.10(c) shows the coefficients of the two dimensional recursive digital filter $H_1(Z_1,Z_2)$. The filter was designed using the cascaded design procedure (the design approach is described in Appendix F). The linear programming method, however, is not suitable in this situation because of the type of approximation involved.

The result of using the recursive filter implementation to restore the motion blurred, noisy image is shown in Figure 3.11. Figures 3.11(a) and (b) are the original and blurred images shown earlier in Figures 3.7(b) and (c). Figure 3.11(d) shows the restoration from motion blur, where the inverse

---

† Let $B(Z_1,Z_2)$ and $I_2(Z_1,Z_2)$ be the Z transforms of $b(m,n)$ and $i_2(m,n)$, respectively. Then:

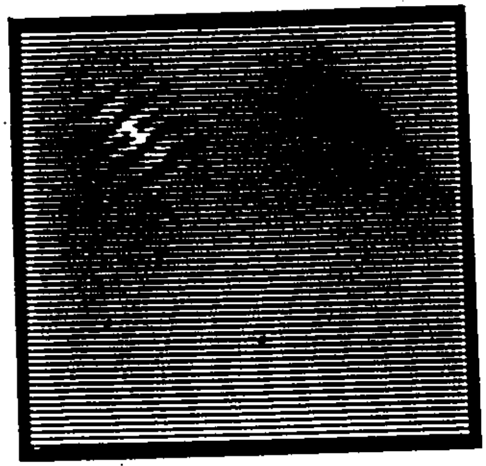
$$I_2(Z_1,Z_2) = B(Z_1,Z_2)\cdot H_1^*(Z_1,Z_2) = B^*(Z_1,Z_2)\cdot H_1(Z_1,Z_2)^*$$

Let $I_A(Z_1,Z_2) = B^*(Z_1,Z_2)$ and let $i_A(m,n)$ be the inverse Z transform of $I_A(Z_1,Z_2)$. The inverse Z transform of $B^*(Z_1,Z_2)$ is an array $b_1(m,n)$, such that:

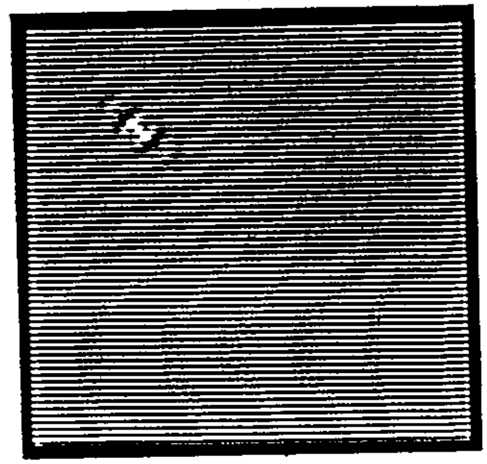
$$b_1(m,n) = b(M-m,N-n)\ ;\ m=0,1,\ldots,M\ ;\ n=0,1,\ldots,N\ .$$

From the above it can be seen that b_1 is obtained by rotating b through 180° about its minor diagonal. i_A is now the filtered output of H_1, with b_1 as the input. As before i_2 is obtained by rotating i_A through 180° about its minor diagonal.

‡ The impulse responses are shifted from the origin by 16 pixels in both spatial directions.



(a)



(b)

```
# OF 1ST ORD CASCADES= 1 , # OF 2ND ORD CASCADES= 1

A0=  .26440229051581550-01

CASCADE SECTION # = 1

A(0.0)=  .1000000000000000D 01      B(0.0)=  .1000000000000000D 01
A(0.1)= -.1003083863699968D 00      B(0.1)=  .6644838711353664D-01
A(1.0)=  .7341402965489785D 00      B(1.0)=  .7057847194698069D 00
A(1.1)=  .1678596599355020D-01      B(1.1)=  .1973196455158322D 00


CASCADE SECTION # = 2

C(0.0)=  .1000000000000000D 01      D(0.0)=  .1000000000000000D 01
C(0.1)= -.1110481494867157D 01      D(0.1)=  .1162602227188408D 00
C(0.2)=  .3037569657003432D 01      D(0.2)= -.1882630060641928D-01
C(1.0)= -.2603153037754572D 00      D(1.0)=  .1766902440322087D 00
C(1.1)=  .1430236633523622D 02      D(1.1)=  .5481103347570174D 00
C(1.2)=  .1493566466746765D 01      D(1.2)=  .1002558734330278D 00
C(2.0)=  .2833783317482269D 01      D(2.0)= -.2427145028110564D-01
C(2.1)= -.4036324129371472D 00      D(2.1)=  .1249117326508150D 00
C(2.2)= -.2715113184155007D 01      D(2.2)=  .4501940746669472D 00
```

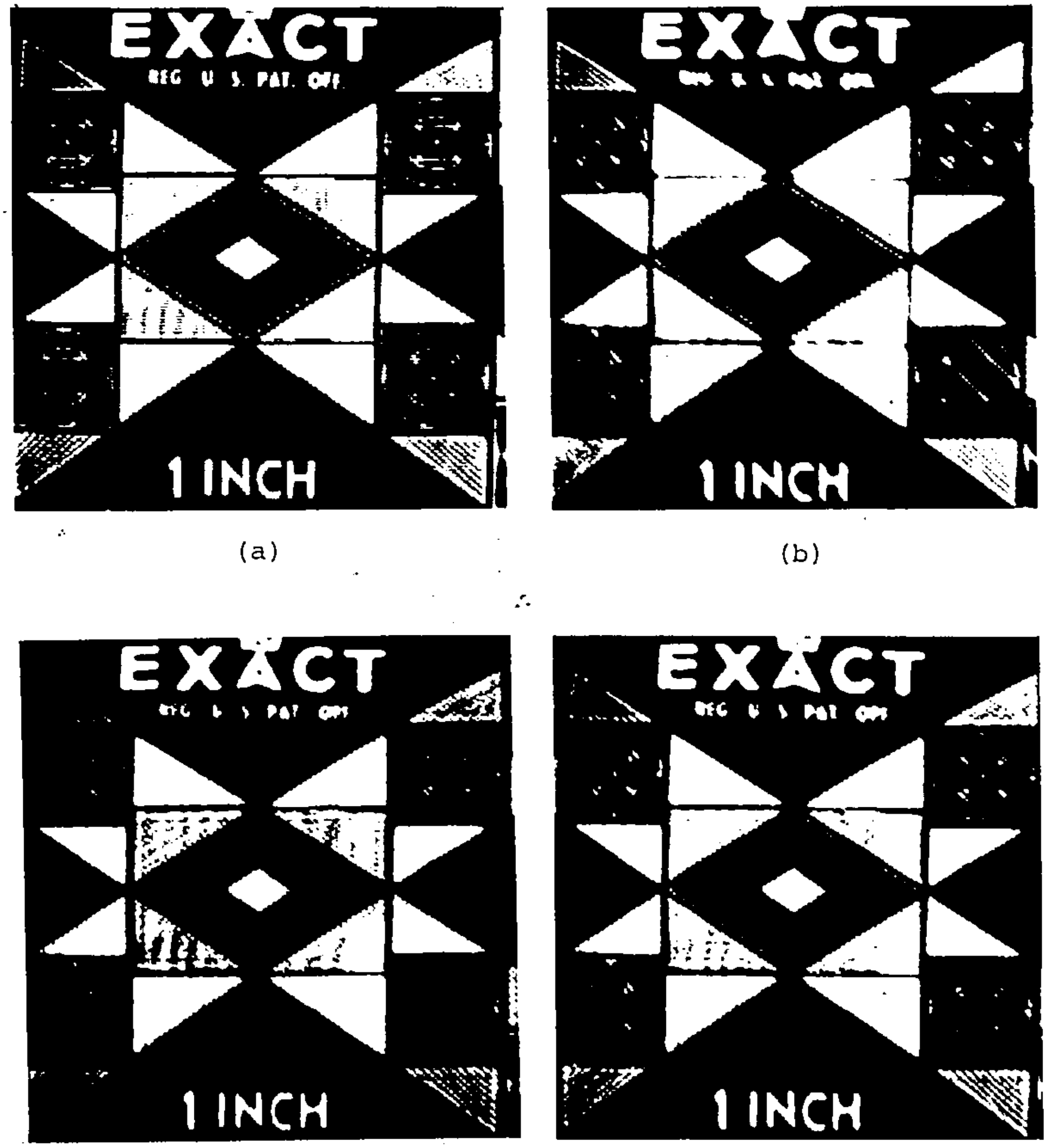
(c)

Figure 3.10 (a) Impulse Response of Desired Restoration Filter ‡.

(b) Impulse Response of Recursive Filter Implementation ‡.

(c) Coefficients of the Recursive Filter $H_1(Z_1,Z_2)$

‡ Origin is at the upper left hand corner. Size = 64x64 pixels.



(a) (b)

(c) (d)

Figure 3.11 (a) Original Image
(b) Blurred Image with Noise; Blur Length=4, Angle=45°, SNR=30 db
(c) Image Restored Using FFT
(d) Image Restored Using Recursive Implementation of Figure 3.9(a).

filtering was carried out using the recursive filter implementation of Figure 3.9(a). The restoration that was carried out via convolution using the FFT is shown in Figure 3.11(c). In another example considered, the blur length is increased to 8 pixels and the angle of blur is set equal to 22.5° with respect to the horizontal. The corresponding blurring transfer function frequency response is shown in Figure 3.12(a). The original image and the motion blurred noisy image, where the blurring is performed using the transfer function of Figure 3.12(a), is shown in Figure 3.15(a) and (b) respectively. The restoration filter specifications, computed based on the power spectrum equivalization criteria, is shown in Figure 3.12(b). Figure 3.12(c) shows the frequency response $H(\Omega_1,\Omega_2)$ of the recursive filter implementation of Figure 3.9(a) after the approximation. The impulse responses† corresponding to Figures 3.12(b) and (c) are shown in Figures 3.13(a) and (b) respectively. The coefficients of the designed recursive filter are shown in Figure 3.14. The results of using the recursive filter implementation for deblurring the motion blurred image of Figure 3.15(b) is shown in Figure 3.15(d). For the purpose of comparison, the restoration carried out by convolution via the FFT is shown in Figure 3.15(c).

3.3.c Focus Blur

When a camera lens system is out of focus, then the image of a point source of light is not a point; instead, it is a circular disc of constant intensity. This is the

† the impulse responses are shifted from origin by 16 pixels in each spatial direction.

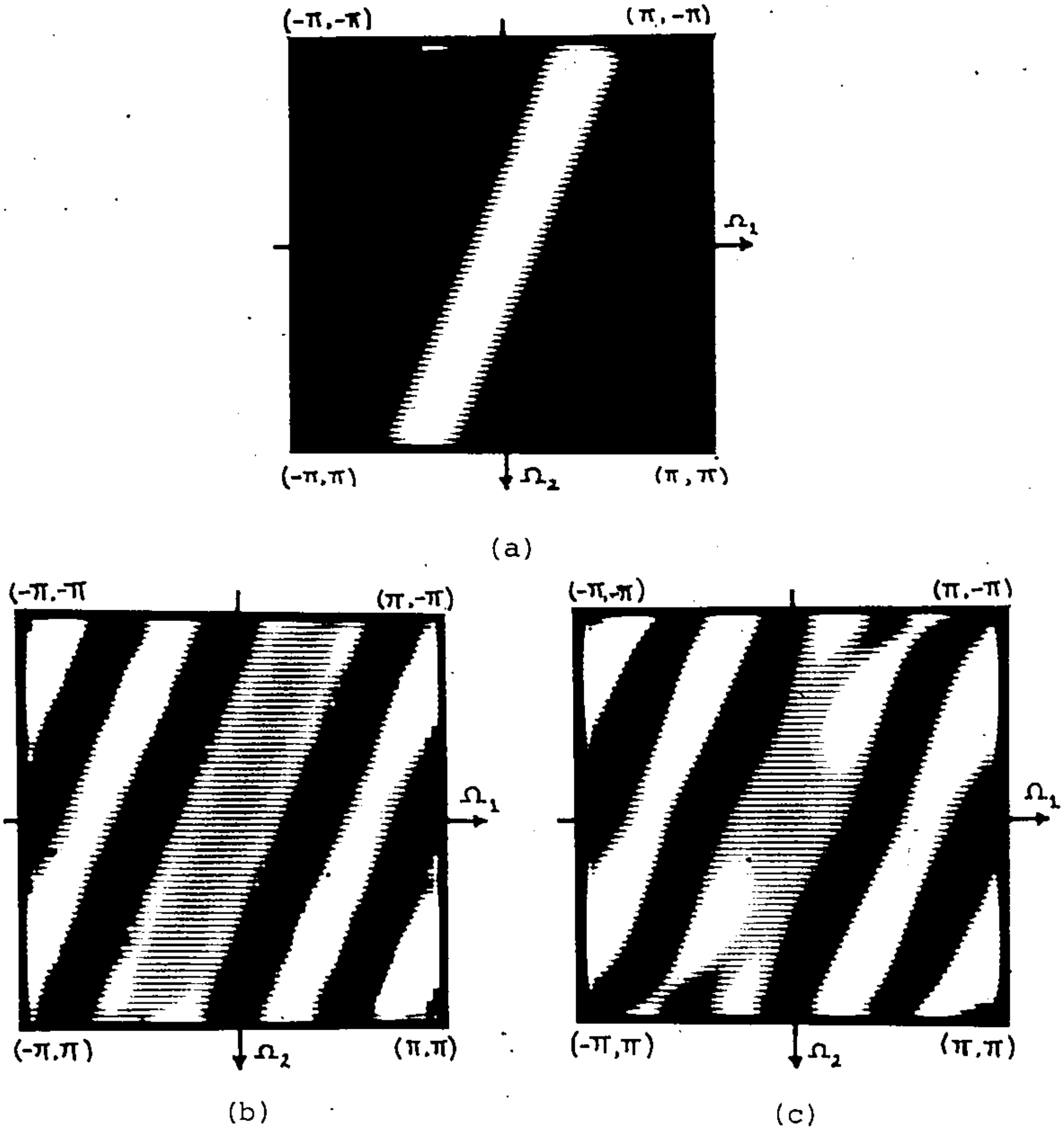
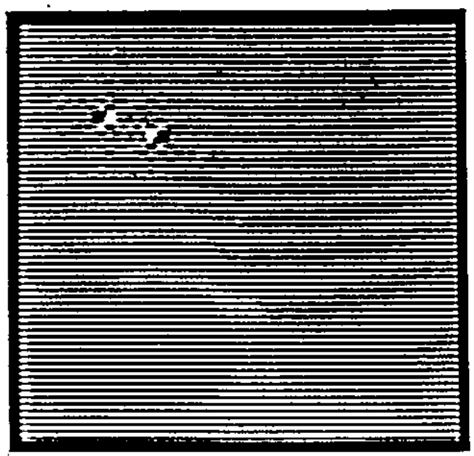
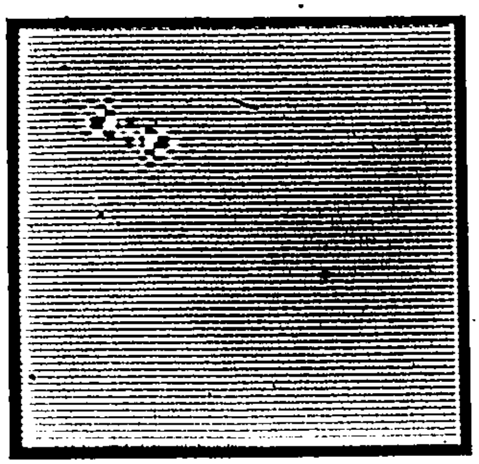


Figure 3.12 (a) Motion Blurring Transfer Function; Blur Length=8 pixels, Angle=22.5° with respect to horizontal
(b) PSEC Restoration Filter Corresponding to (a) and SNR=30 db.
(c) Recursive Filter Approximation to (b).



(a)



(b)

Figure 3.13 (a) Impulse Response of Desired Restoration Filter ‡
(b) Impulse Response of Recursive Filter Implementation ‡

‡ Origin is at the upper left hand corner. Size=64x64 pixels.

```
# OF 1ST ORD CASCADES= 2 , # OF 2ND ORD CASCADES= 1

A0= .2422416858840913D-01

CASCADE SECTION # = 1

A(0.0)=  .1000000000000000D 01     B(0.0)=  .1000000000000000D 01
A(0.1)= -.4972509122866367D 00     B(0.1)= -.4103727140135029D 00
A(1.0)=  .3674239992364412D 00     B(1.0)= -.2405544455206474D 00
A(1.1)= -.6533632115090327D 00     B(1.1)=  .2663596135895748D 00


CASCADE SECTION # = 2

A(0.0)=  .1000000000000000D 01     B(0.0)=  .1000000000000000D 01
A(0.1)= -.9607166305801180D 00     B(0.1)= -.6862261413011388D-02
A(1.0)= -.1758633247230895D 01     B(1.0)=  .2342174747700759D 00
A(1.1)=  .5648634983254563D 01     B(1.1)= -.3733908668970402D 00


CASCADE SECTION # = 3

C(0.0)=  .1000000000000000D 01     D(0.0)=  .1000000000000000D 01
C(0.1)=  .2103011228745327D 01     D(0.1)=  .8428940615734177D-02
C(0.2)=  .3929397613648437D 01     D(0.2)=  .1401852781064118D 00
C(1.0)=  .1045248343517323D 00     D(1.0)=  .2508090800462715D 00
C(1.1)=  .2538266120462690D 00     D(1.1)=  .1069113289132319D 00
C(1.2)=  .1227549568585194D 01     D(1.2)=  .5149778110380243D-01
C(2.0)= -.4737383517005235D 00     D(2.0)= -.3707212898722911D 00
C(2.1)= -.1165890319356933D 01     D(2.1)= -.1187656445252706D 00
C(2.2)= -.2511052402437726D 01     D(2.2)= -.4587766559607870D-01
```

Figure 3.14 Coefficients of the Recursive Filter $H_1(Z_1,Z_2)$.

(a) (b)

(c) (d)

Figure 3.15 (a) Original Image
(b) Motion Blurred Image with Noise; Blur Length≑8 pixels; Angle=22.5°, with respect to horizontal; SNR=30 db
(c) Restoration Using FFT
(d) Restoration Using Recursive Filter Implementation of Figure 3.9(a).

point spread function of a focus blurring system. Although the true point spread function is actually related to the fourier transform of the aperature of the lens system, a cylindrical approximation is a good one and is also mathematically tractable. Thus, the focus blurring point spread function can be written as:

$$\left.\begin{array}{ll} h(r) = 0 & ;\ r > R \\ h(r) = 1/(\pi R^2) & ;\ r \leqslant R\ ;\ r = \sqrt{x^2 + y^2} \end{array}\right\} \qquad (3.3.18)$$

The plot of (3.3.18) is shown in Figure 3.16(a). The fourier transform of this point spread function is a bessel function of first order and it is of the form:

$$H_F(w) = 2 \cdot J_1(Rw)/(Rw) \quad ;\ w = \sqrt{w_1^2 + w_2^2} \qquad (3.3.19)$$

where w_1 and w_2 are the two dimensional frequency variables. As in Section 3.3.b, let Δ and w_s be the sampling interval and the sampling frequency respectively. The radius R of the blur point spread function can then be written as:

$$R = k\Delta \qquad (3.3.20)$$

where k is referred to as the radius of blur in pixels. Therefore, (3.3.19) can be rewritten as:

$$H_F(\Omega) = 2 \cdot J_1(k\Omega)/(k\Omega) \quad ;\ \Omega = \sqrt{\Omega_1^2 + \Omega_2^2} \qquad (3.3.21)$$

where $\Omega_1 = \frac{w1\pi}{(w_{s/2})}$ and $\Omega_2 = \frac{w2\pi}{(w_{s/2})}$ are the normalized frequency variables. $H_F(\Omega)$ is a radially symmetric function with alternating positive and negative side lobes. An image that is filtered through $H_F(\Omega)$, will not only experience an attenuation

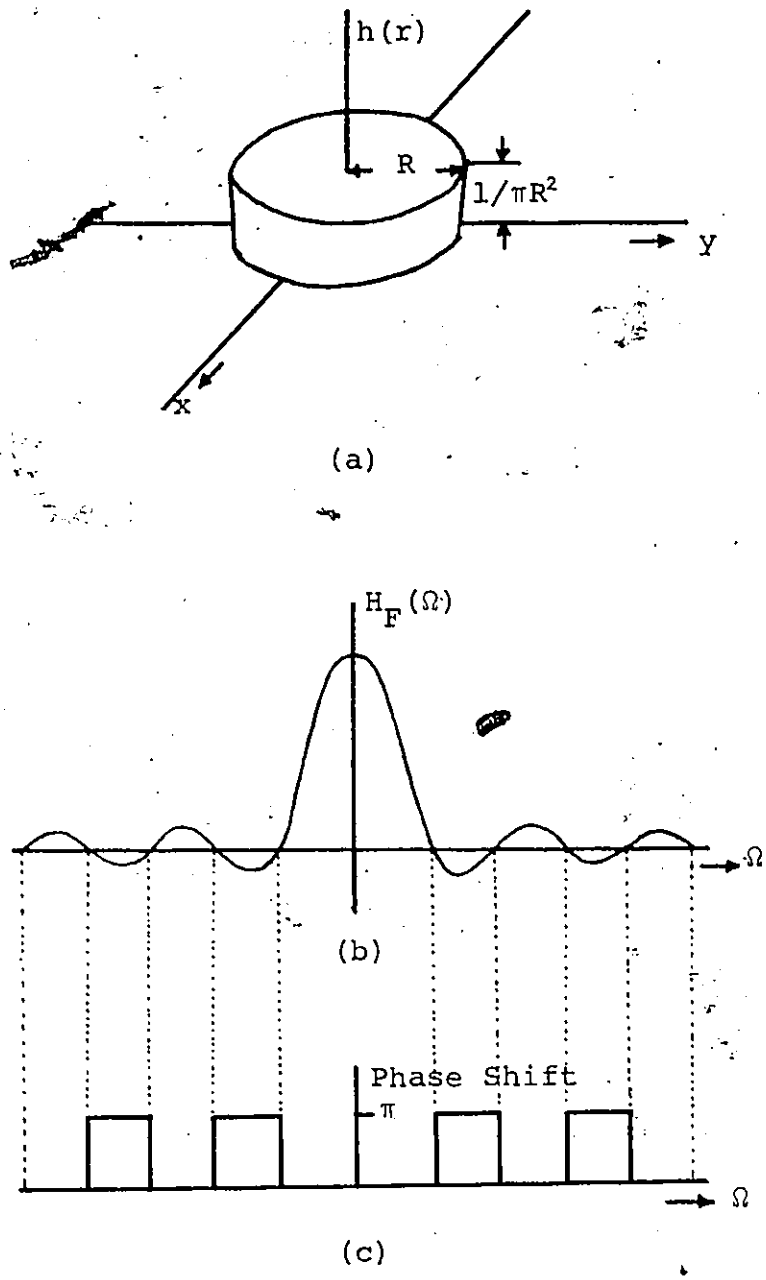
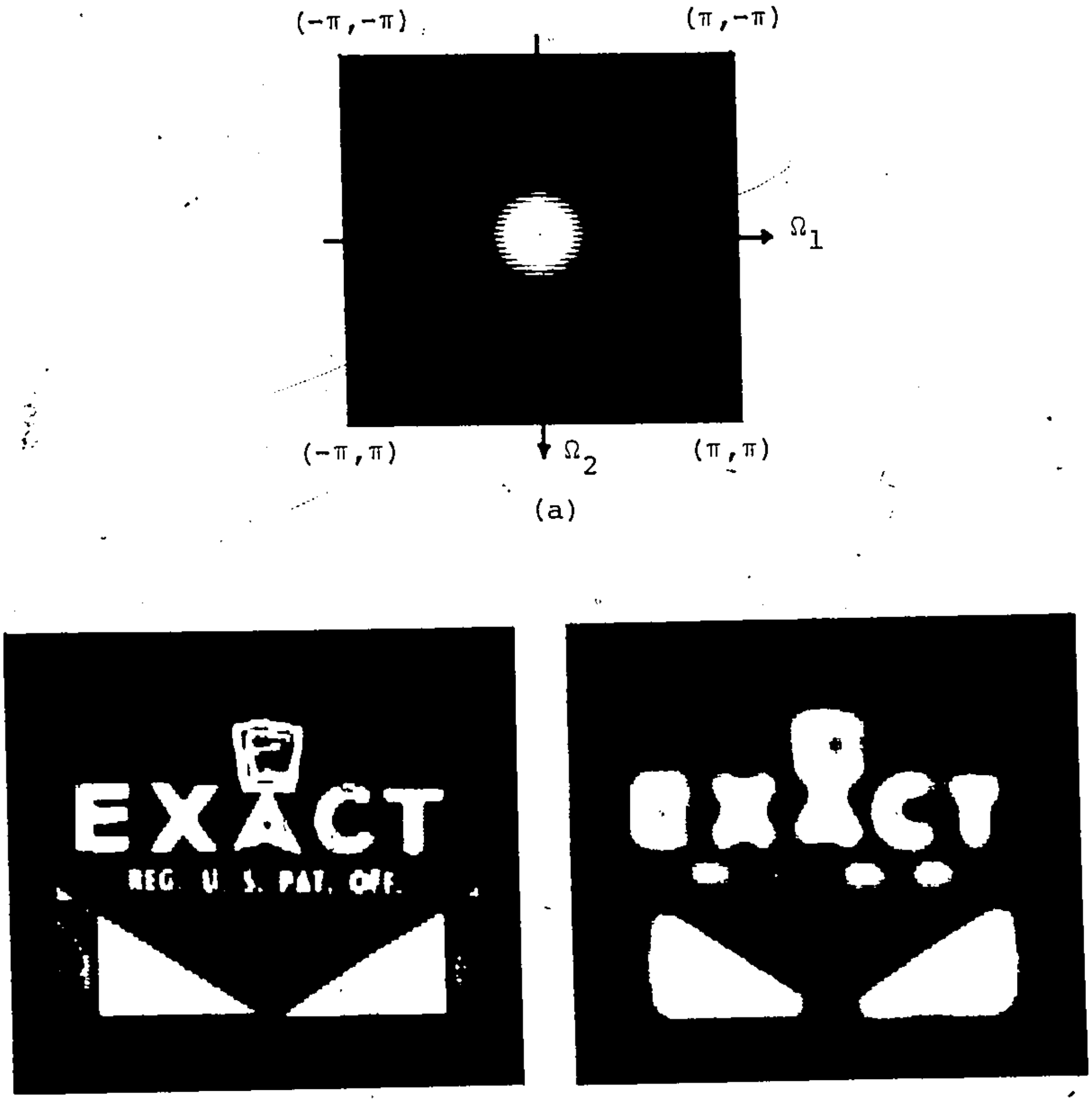


Figure 3.16 (a) Focus Blur Point Spread Function
(b) Transfer Function of Focus Blur
(c) Phase Shift Introduced by (b)

of higher frequency components, but also the frequencies in the negative side lobe regions will experience a phase shift of π radians. The plot of $H_F(\Omega)$ along any radial direction is shown in Figure 3.16(b). Figure 3.16(c) shows the phase shift experienced by various frequency components of the input.

In Figure 3.17(a) is shown the transfer function corresponding to focus blur, where the radius of the blur is 4 pixels. Using this transfer function, the original image of Figure 3.17(b) is blurred and noise is added to the image such that the SNR is equal to 30 db. The blurred noisy image is as shown in Figure 3.17(c). For this blurred image, a restoration filter response $H_{FR_2}(\Omega_1,\Omega_2)$ was computed based on the power spectrum equivalization criterion. The response is purely real and it is as shown in Figure 3.18(a). An approximate form of the response along a radial direction of 45^o with respect to the horizontal is shown in Figure 3.18(b). In Figure 3.18(a), the intense dark regions and the intense bright regions correspond to the most negative and most positive values, respectively of the function in Figure 3.18(b). The restoration filter magnitude characteristic is the absolute value of $H_{FR_2}(\Omega_1,\Omega_2)$ and the phase characteristics is as shown in Figure 3.18(c).

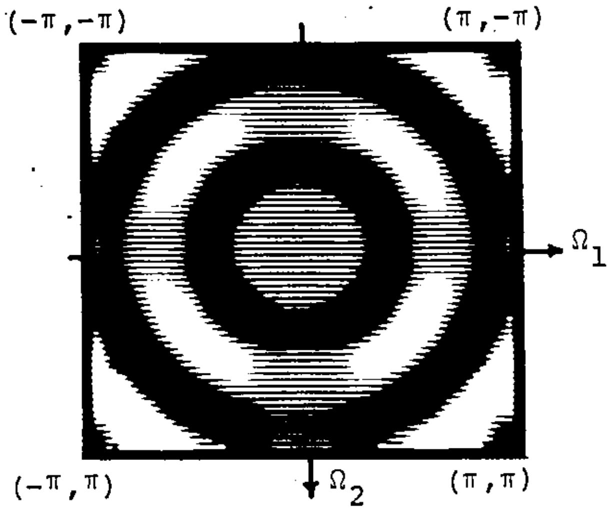
As in the case of motion blur, the desired restoration filter phase characteristics is discontinuous in nature. Such a response cannot be realized by a two dimensional recursive digital filter having causal impulse response. However, a possible recursive filter implementation to realize



(a) (b) (c)

Figure 3.17 (a) Focus Blurring Transfer Function⁺, Blur Radius=4 Pixels
(b) Original Image (Size 128x128 Pixels)
(c) Focus Blurred Noisy Image; SNR=30db

⁺Origin is at the centre of the image.



(a)

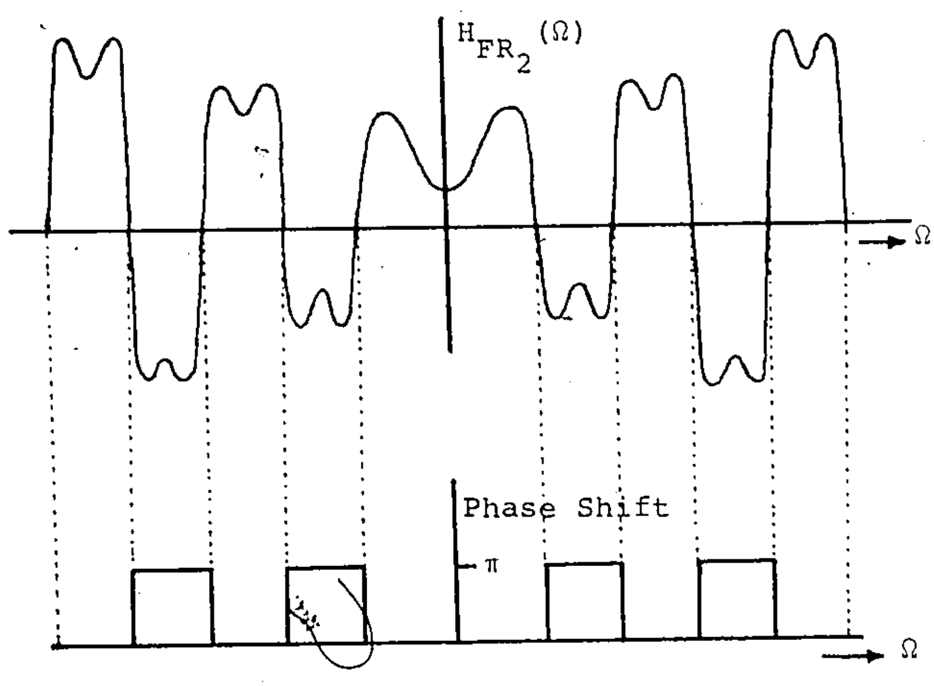


Figure 3.18 (a) Restoration Filter Response†
(b) Restoration Filter Response Along 45° Radial Direction
(c) Phase Shift Introduced by (b)

† Origin is at the centre.

the desired response is as shown in Figure 3.19(a). The implementation consists of two recursive filters in parallel having non-causal impulse response. The overall frequency response $H(\Omega_1,\Omega_2)$ of the recursive filter implementation is given by:

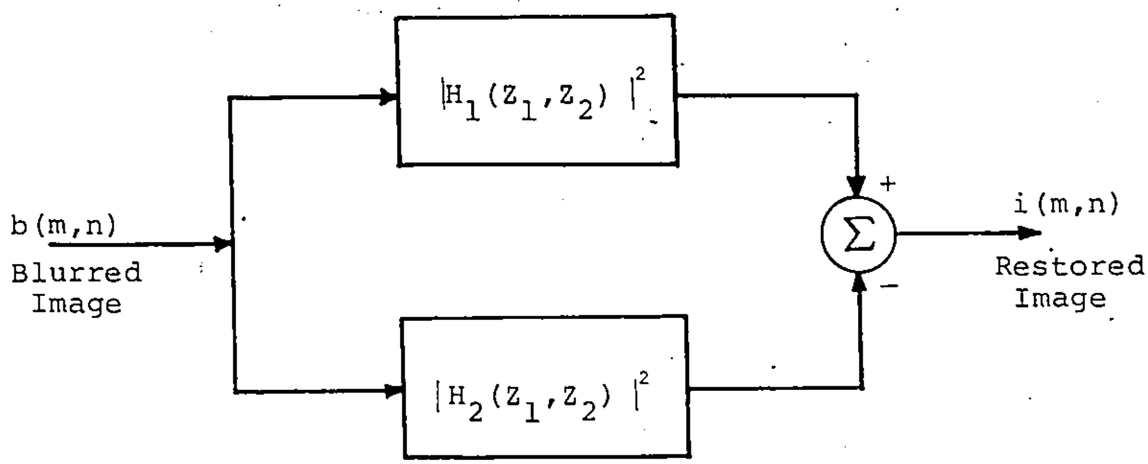
$$H(\Omega_1,\Omega_2) = |H_1(Z_1,Z_2)|^2 - |H_2(Z_1,Z_2)|^2 ; \qquad (3.3.22)$$
$$Z_1 = e^{-j\Omega_1} , \quad Z_2 = e^{-j\Omega_2}$$

where

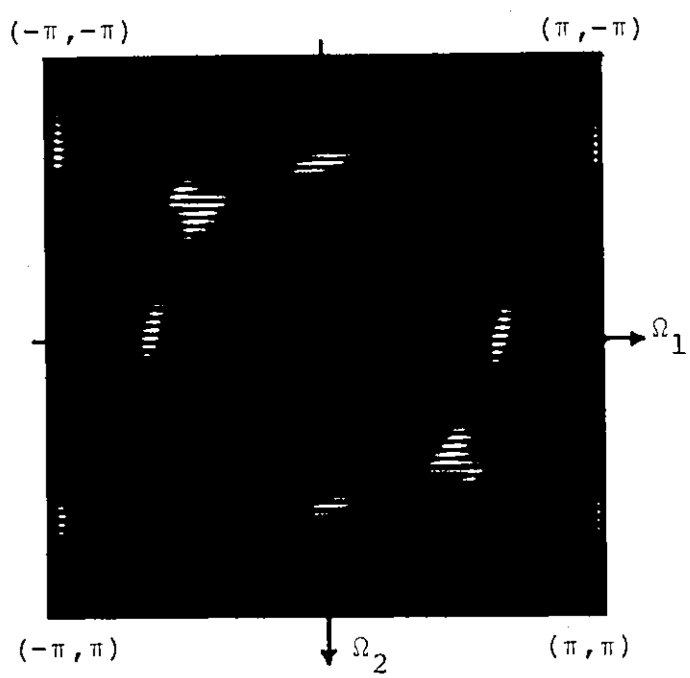
$$|H_1(Z_1,Z_2)|^2 = H_1(Z_1,Z_2) \cdot H_1^*(Z_1,Z_2)$$
$$\qquad (3.3.23)$$
$$|H_2(Z_1,Z_2)|^2 = H_2(Z_1,Z_2) \cdot H_2^*(Z_1,Z_2)$$

The impulse response of the overall filter $H(\Omega_1,\Omega_2)$ is non-causal since $H_1^*(Z_1,Z_2)$ and $H_2^*(Z_1,Z_2)$ are filters with non-causal impulse response. The filtering by $H_1^*(Z_1,Z_2)$ can be carried out in a manner described in Section 3.3.b. The frequency response realized by $H(\Omega_1,\Omega_2)$, after the approximation, is shown in Figure 3.19(b). The coefficients of the filter are shown in Figures 3.20 and 3.21. The filter was designed using the cascaded design procedure (described in Appendix F).

The result of restoration via convolution using FFT is shown in Figure 3.22(a) and Figure 3.22(b) shows the restoration using the recursive filter implementation. It is obvious that the restoration due to recursive filter implementation is a failure. The cause of this failure can



(a)



(b)

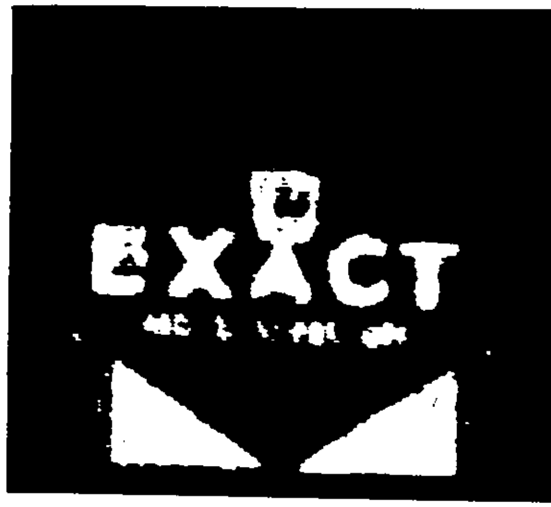
Figure 3.19 (a) Recursive Filter Implementation for Focus Deblur.
(b) Frequency Response of (a) (Origin Is At Centre).

```
# OF 2ND ORD CASCADES= 2 , # OF 1ST ORD CASCADES= 2

A0=  .1025773477444603D 00

CASCADE SECTION # = 1

A(0,0)=  .1000000000000002D 01      B(0,0)=  .1000000000000000D 01
A(0,1)=  .1024342150442620D 00      B(0,1)= -.3480120503340452D 00
A(1,0)=  .1024342150442620D 00      B(1,0)= -.3480120503340452D 00
A(1,1)= -.3767544302462180D 00      B(1,1)= -.6116171967936076D-01


CASCADE SECTION # = 2

A(0,0)=  .1000000000000000D 01      B(0,0)=  .1000000000000000D 01
A(0,1)=  .5585437832608302D 00      B(0,1)= -.1616978969843114D-01
A(1,0)=  .5585437832608302D 00      B(1,0)= -.1616978969843114D-01
A(1,1)=  .1159661479435546D 01      B(1,1)= -.1620512800125240D 00


CASCADE SECTION # = 3

C(0,0)=  .1000000000000000D 01      D(0,0)=  .1000000000000000D 01
C(0,1)= -.9084443738128608D 00      D(0,1)=  .1625681799934077D 00
C(0,2)=  .1610651630894878D 01      D(0,2)=  .3077035110253783D 00
C(1,0)= -.9084443738128608D 00      D(1,0)=  .1625681799934077D 00
C(1,1)= -.4430101634717109D 00      D(1,1)=  .2021560263391447D 00
C(1,2)=  .3335720033241048D-02      D(1,2)=  .8365967993611407D-02
C(2,0)=  .1610651630894878D 01      D(2,0)=  .3077035110253783D 00
C(2,1)=  .3335720033241048D-02      D(2,1)=  .8365967993611407D-02
C(2,2)=  .7858278930375544D 00      D(2,2)= -.4425695101938034D-01

CASCADE SECTION # = 4

C(0,0)=  .1000000000000000D 01      D(0,0)=  .1000000000000000D 01
C(0,1)= -.1404001360697000D 00      D(0,1)=  .4716106700625655D 00
C(0,2)=  .1895189425542409D 00      D(0,2)=  .3497712560888230D 00
C(1,0)= -.1404001360697000D 00      D(1,0)=  .4716106700625655D 00
C(1,1)=  .2656130979150225D 00      D(1,1)=  .3508731781905086D 00
C(1,2)= -.3324512425822108D 00      D(1,2)=  .3624763840247807D 00
C(2,0)=  .1895189425542409D 00      D(2,0)=  .3497712560888230D 00
C(2,1)= -.3324512425822108D 00      D(2,1)=  .3624763840247807D 00
C(2,2)=  .5462900662707240D 00      D(2,2)=  .5627035622789205D 00
```

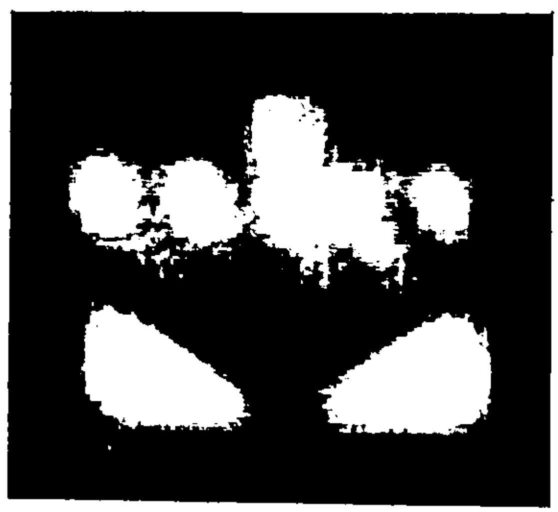
Figure 3.20 Coefficients of Two Dimensional Filter $H_1(Z_1,Z_2)$.

```
# OF 2ND ORD CASCADES= 2  # OF 1ST ORD CASCADES= 2

AO= .2560094172179390D 00

CASCADE SECTION # = 1

A(0.0)=  .1000000000000000D 01     B(0.0)=  .1000000000000000D 01
A(0.1)=  .2554556675250177D-01     B(0.1)=  .4750741851718458D-01
A(1.0)=  .2554556675250177D-01     B(1.0)=  .4750741851718458D-01
A(1.1)= -.4597090063746180D 00     B(1.1)= -.7001788014266632D 00


CASCADE SECTION # = 2

A(0.0)=  .1000000000000000D 01     B(0.0)=  .1000000000000000D 01
A(0.1)= -.5482699821488491D-01     B(0.1)=  .2912752161940441D 00
A(1.0)= -.5482699821488491D-01     B(1.0)=  .2912752161940441D 00
A(1.1)= -.4357426178074469D 00     B(1.1)=  .6901991762259232D 00


CASCADE SECTION # = 3

C(0.0)=  .1000000000000000D 01     D(0.0)=  .1000000000000000D 01
C(0.1)=  .1545887255426392D 00     D(0.1)=  .1721814682245648D-01
C(0.2)=  .3088158569994695D 00     D(0.2)=  .5941255292995205D 00
C(1.0)=  .1545887255426392D 00     D(1.0)=  .1721814682245648D-01
C(1.1)= -.2095429069733181D-01     D(1.1)= -.1672262752936182D 00
C(1.2)=  .4000538806468211D 00     D(1.2)=  .1899735569141059D 00
C(2.0)=  .3088158569994695D 00     D(2.0)=  .5941255292995205D 00
C(2.1)=  .4000538806468211D 00     D(2.1)=  .1899735569141059D 00
C(2.2)=  .1995700259014226D 00     D(2.2)=  .5001965714888430D 00


CASCADE SECTION # = 4

C(0.0)=  .1000000000000000D 01     D(0.0)=  .1000000000000000D 01
C(0.1)=  .9565814897624177D 00     D(0.1)=  .4201989509201866D 00
C(0.2)=  .1016472874156251D 01     D(0.2)=  .2017354689701443D 00
C(1.0)=  .9565814897624177D 00     D(1.0)=  .4201989509201866D 00
C(1.1)=  .1045954406105385D 00     D(1.1)= -.5655794327644990D 00
C(1.2)=  .9075819640262747D 00     D(1.2)=  .1749242081455408D 00
C(2.0)=  .1016472874156251D 01     D(2.0)=  .2017354689701443D 00
C(2.1)=  .9075819640262747D 00     D(2.1)=  .1749242081455408D 00
C(2.2)=  .1160859244345993D 01     D(2.2)=  .4859593858039368D 00
```

Figure 3.21 Coefficients of Two Dimensional Filter $H_2(Z_1,Z_2)$.



(a)



(b)

Figure 3.22 (a) Restoration by Convolution Via FFT.
(b) Restoration Using Recursive Filter Implementation of Figure 3.19(a).

be attributed to the recursive filters $H_1(Z_1,Z_2)$ and $H_2(Z_1,Z_2)$, which are quarter plane recursive filters. As pointed out by Ekstrom and Woods [40], this class of filters cannot realize general magnitude characteristics. As can be seen from Figure 3.19(b), the approximation of the recursive filter implementation to the desired response is inadequate.

The class of filters, called half plane filters, appear to be more suitable for this type of application, and their use is suggested for further work in this area. Design techniques for half plane filters were not available at the time this work was performed and therefore the application of this class of filters is not considered here. At the time of completion of this thesis, several techniques for half plane filter design have been demonstrated in the literature, [41,42] which it is suggested, can be used to adequately approximate the required focus deblurring filter response.

3.3.d Atmospheric Turbulance Blur

The cause of this type of blur is attributed to the variation in the refractive index of the atmosphere. Since the atmosphere is thermally non-uniform, its refractive index varies as function of both time and space. Therefore, in a strict sense, the point spread function corresponding to atmospheric turbulance blur is not only space variant, but also a function of time. However, it has been shown by Horner [43], that the image of a point source of light coming through atmosphere, when averaged over a period of time, has the form of a gaussian function and therefore, one can write the point spread function corresponding to atmospheric turbulance blur as:

$$h(r) = e^{-kr^2} \quad ; \; r = \sqrt{x^2 + y^2} \qquad (3.3.24)$$

where x and y are spatial coordinates and k is a constant. The fourier transform of the point spread function is also gaussian and it is given by:

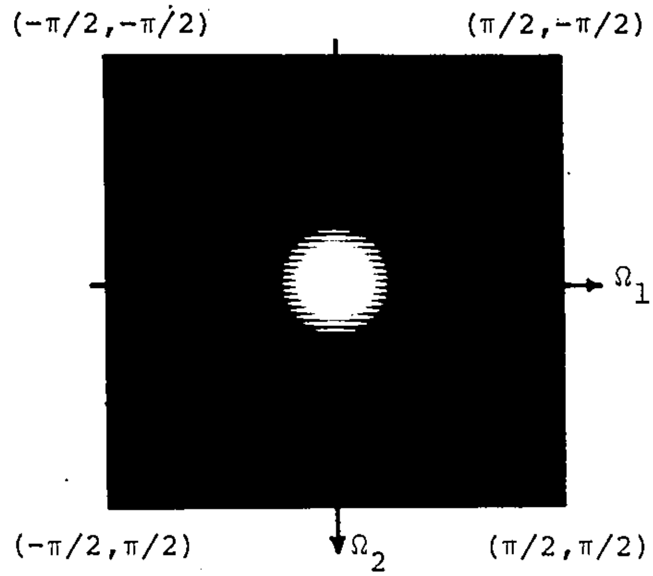
$$H_A(w) = \frac{\pi}{k} e^{-(w^2/4k)} \quad ; \; w = \sqrt{w_1^2 + w_2^2} \qquad (3.3.25)$$

where w_1 and w_2 are spatial frequency variables. As before, choosing w_s as the sampling frequency in both spatial directions, the atmospheric turbulance blur transfer function can be written as:

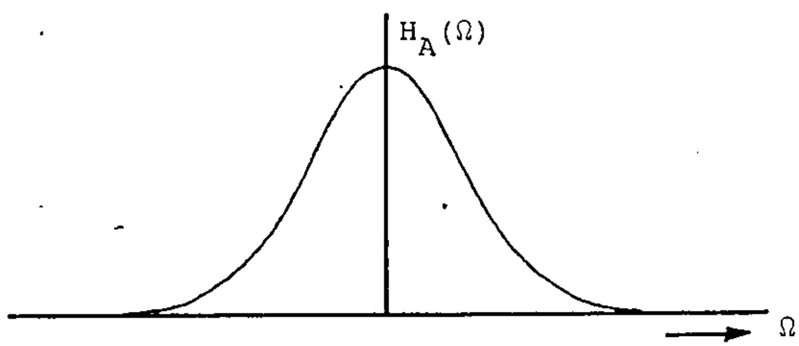
$$H_A(\Omega) = \frac{\pi}{k} e^{-(\Omega^2/4k)} \quad ; \; \Omega = \sqrt{\Omega_1^2 + \Omega_2^2} \qquad (3.3.26)$$

where $\Omega_1 = (\frac{w1\pi}{w_{s/2}})$ and $\Omega_2 = (\frac{w2\pi}{w_{s/2}})$ are the normalized frequency variables. The computer generated atmospheric turbulance blur transfer function for a value of k = 2 is shown in Figure 3.23(a) and a plot of the cross section of the transfer function is shown in Figure 3.23(b). This transfer function is used to blur the original image, shown in Figure 3.27(a). Noise is added to the blurred image such that the signal to noise ratio is 30 db. The noisy blurred image is shown in Figure 3.27(b).

For the blurred image of 3.27(b), a restoration filter, $H_{A_R}(\Omega)$, was computed based on the minimum mean squared error criterion. The intensity plot of the specifications is shown in Figure 3.24(a). An approximate variation of restoration



(a)



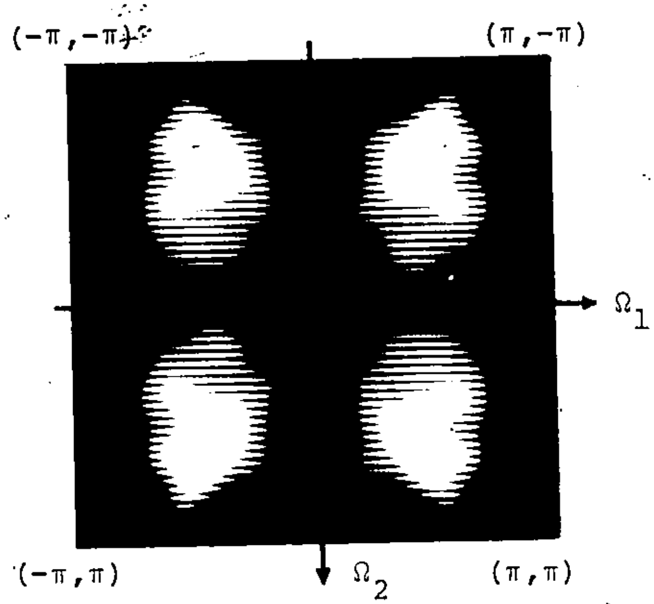
(b)

Figure 3.23 (a) Atmospheric Turbulance Blur Transfer Function‡
(b) One Dimensional Plot of the Transfer Function Along Any Radial Direction

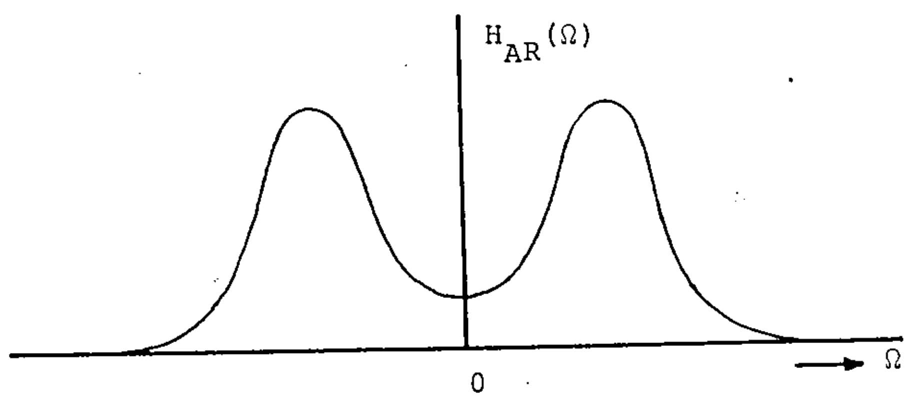
‡ Origin is at the centre.

filter specifications, along 45° radial direction is shown in Figure 3.24(b). The intense dark and the intense bright regions in 3.24(a) correspond to the zero and most positive values of the function shown in Figure 3.24(b). Since the blurring transfer function has zero or linear phase characteristics, the restoration filter also has zero or linear phase characteristics. Therefore, the linear programming technique was used in designing a causal recursive digital filter to the desired restoration filter magnitude and linear phase characteristics. The magnitude characteristics of the designed filter, and its coefficients, are shown in Figure 3.25(a) and (b). The impulse responses corresponding to the restoration filter frequency domain specifications and the recursive digital filter, designed using linear programming, are shown in Figure 3.26(a) and (b). The result of using the recursive filter for the purpose of restoration is shown in Figure 3.27(d). The restoration performed via convolution using FFT is shown in Figure 3.27(c).

One more example of atmospheric turbulance blur is considered, in which the image is blurred using the transfer function of (3.3.26) with $k = 0.5$. The SNR in the blurred noisy image is 20db. The original and the blurred noisy images are shown in Figures 3.30(a) and (b). In this example, a recursive filter implementation whose impulse response is non-causal, is used as opposed to the causal implementation of the previous example, in the restoration of the blurred image. The recursive filter imple-



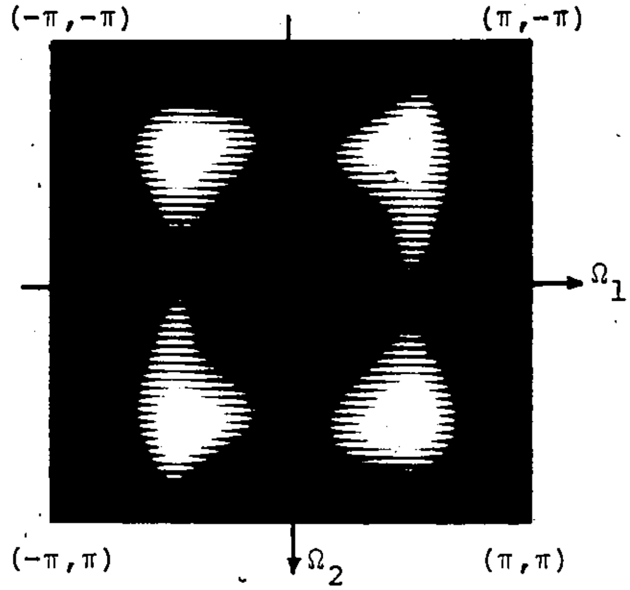
(a)



(b)

Figure 3.24 (a) Restoration Filter Response for Atmospheric Blur+
(b) Plot of Magnitude of (a) Along 45° Radial Direction

+Origin is at the centre of the image.



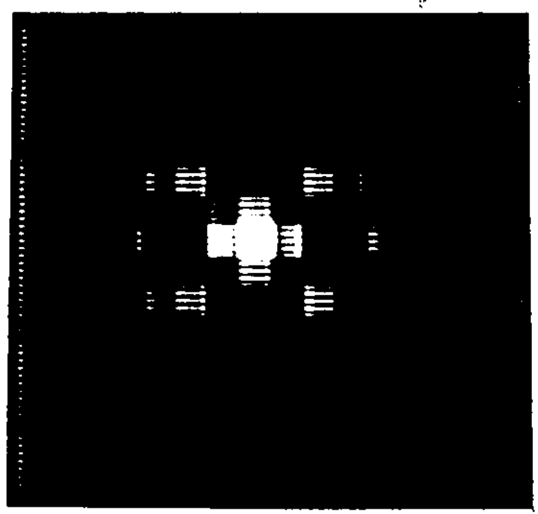
(a)

N(1. 1)=	.25498670E-02	D(1. 1)=	1.0000000
N(1. 2)=	.74899490E-02	D(1. 2)=	-.50561400
N(1. 3)=	.41056050E-02	D(1. 3)=	.53115590
N(1. 4)=	-.24494110E-02	D(1. 4)=	-.18774210
N(2. 1)=	.24467150E-01	D(2. 1)=	-.24585170
N(2. 2)=	.25440030E-01	D(2. 2)=	.10674080
N(2. 3)=	-.23722040E-01	D(2. 3)=	-.12841990
N(2. 4)=	-.60396460E-01	D(2. 4)=	.81305800E-01
N(3. 1)=	-.10464750E-01	D(3. 1)=	.37625940
N(3. 2)=	-.18416250E-01	D(3. 2)=	-.19782750
N(3. 3)=	.14736130E-01	D(3. 3)=	.22916440
N(3. 4)=	.47165410E-01	D(3. 4)=	-.11484400
N(4. 1)=	-.16275610E-01	D(4. 1)=	-.19099720
N(4. 2)=	-.62952880E-01	D(4. 2)=	.93058290E-01
N(4. 3)=	.76140580E-01	D(4. 3)=	-.86800810E-01
N(4. 4)=	.12656440	D(4. 4)=	.93072410E-01

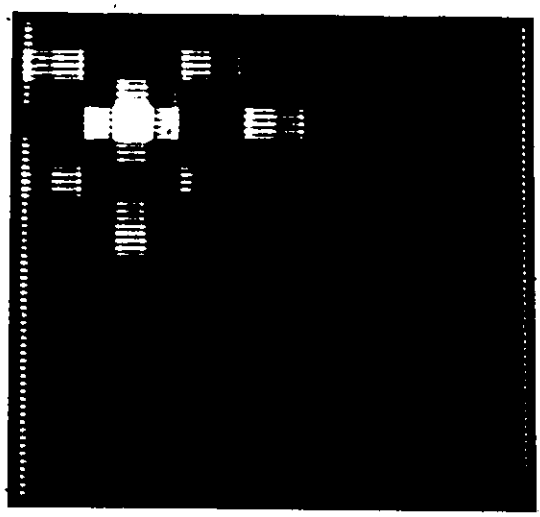
(b)

Figure 3.25 (a) Magnitude Response of the Designed Filter
(b) Coefficients of the Designed Filter +

+The array N corresponds to numerator coefficients and array D corresponds to denominator coefficients.



(a)



(b)

Figure 3.26 (a) Impulse Response Corresponding to Desired Magnitude Specifications (16x16 pixels)
(b) Impulse Response of the Designed Filter (the first 16x16 pixels)

Note: For (a) the origin is at the centre of the image; for (b) the origin is at upper left hand corner. In (a) and (b) each square area of constant intensity represents one pixel.

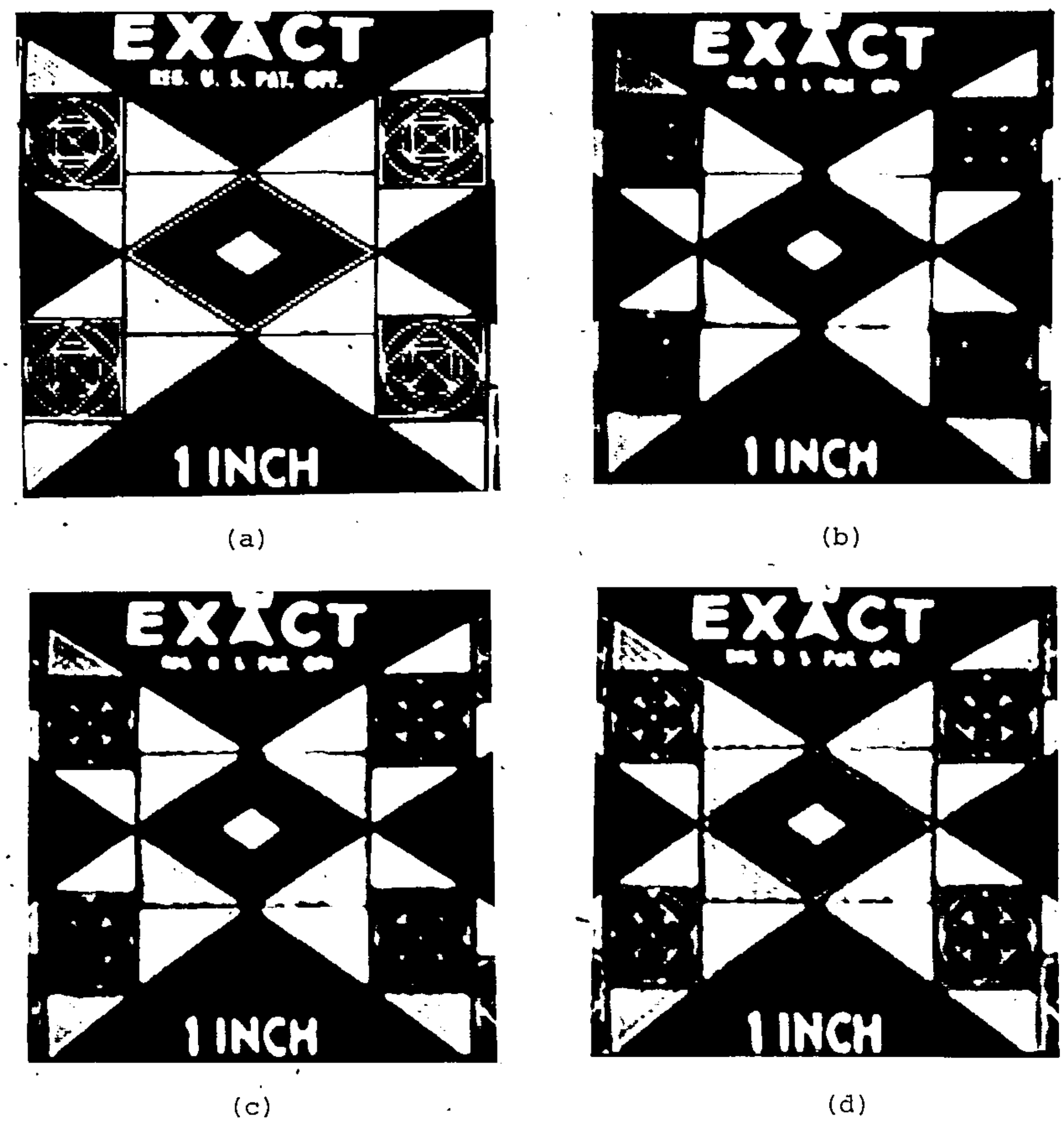


Figure 3.27 (a) Original Image
(b) Blurred Noisy Image; SNR=30 db
(c) Restoration Using FFT
(d) Restoration Using Recursive Filter

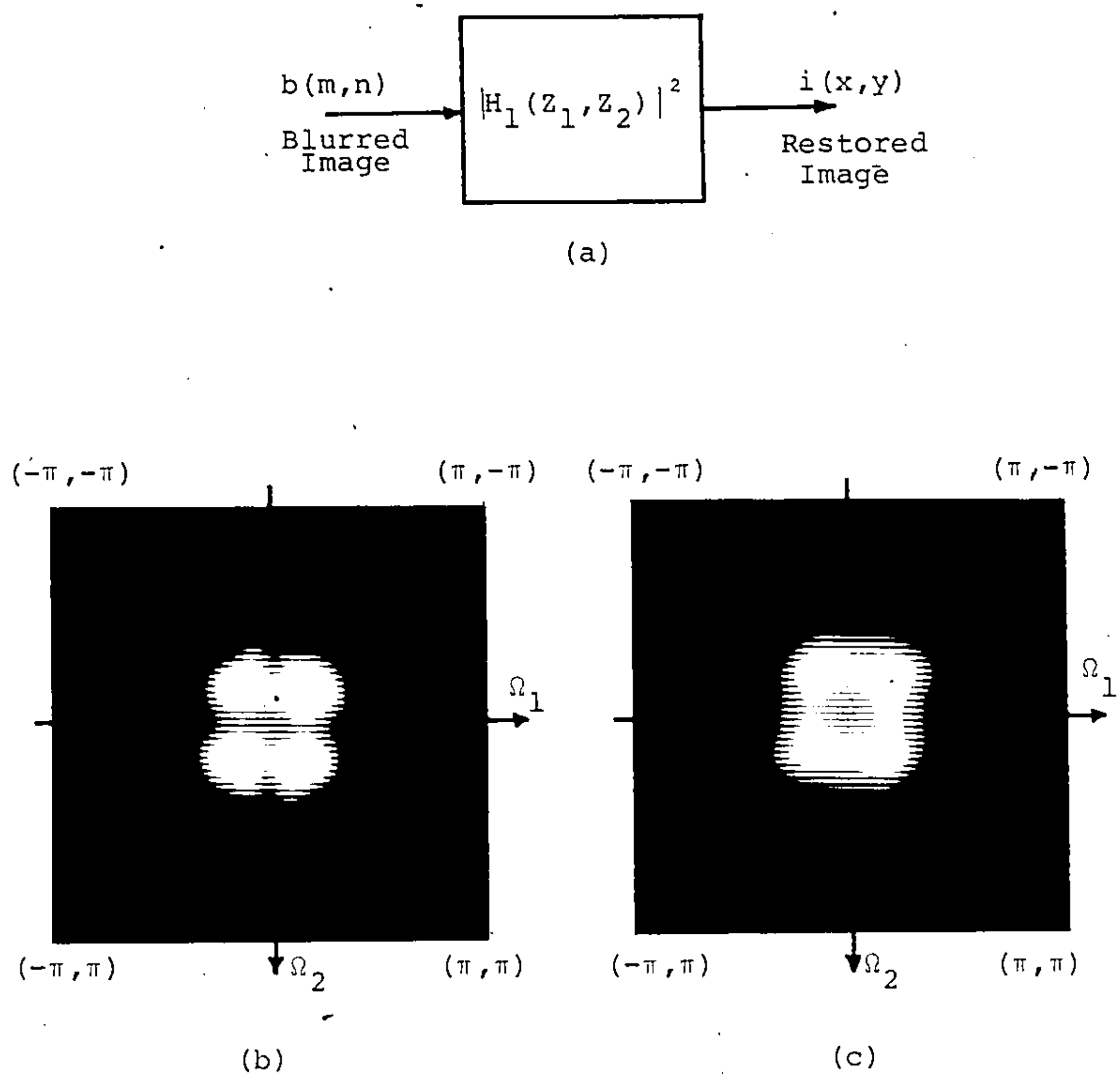
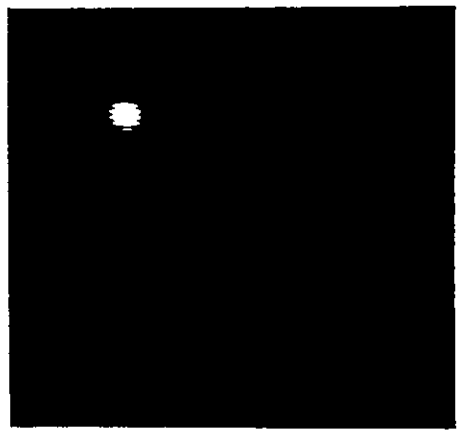
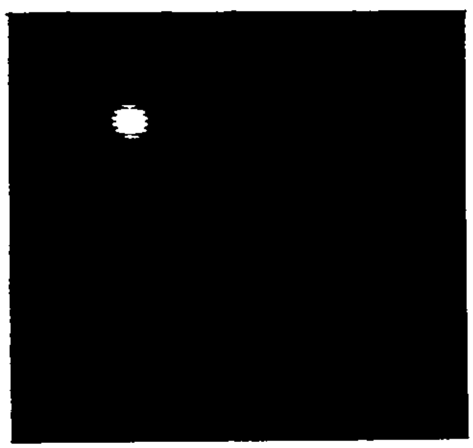


Figure 3.28 (a) Recursive Filter Implementation for Atmospheric Turbulance Deblur
(b) Desired Restoration Filter Frequency Response
(c) Frequency Response Realized by (a)



(a)



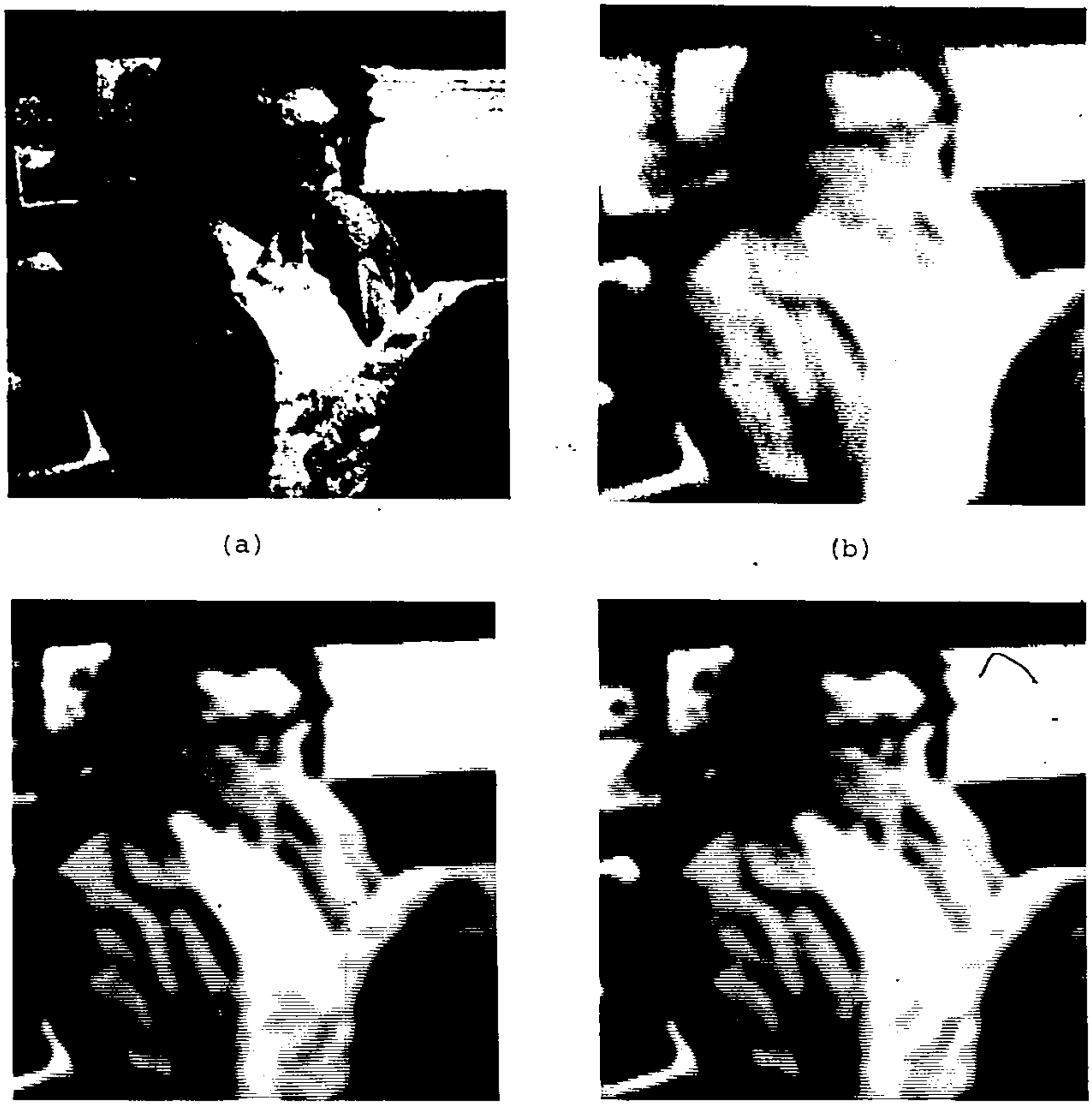
(b)

Figure 3.29 (a) Impulse Response Corresponding to Figure 3.28(b)+ (64 x 64 Pixels)
(b) Impulse Response Corresponding to Figure 3.28(c)+ (64 x 64 Pixels)

+ Origin is at upper left hand corner.

```
AO= .1227548808825765D-01

CASCADE SECTION # = 1

C(0,0)=  .1000000000000000D 01
C(0,1)=  .3706583492246172D 01
C(0,2)=  .1394659257816349D 01
C(1,0)=  .3639675365119787D 01
C(1,1)= -.1235379612695683D 01
C(1,2)=  .4705701202775955D 01
C(2,0)= -.3005763355293158D 01
C(2,1)=  .3618879697731638D 01
C(2,2)= -.1181942282898142D 01


D(0,0)=  .1000000000000000D 01
D(0,1)= -.9309267746238101D 00
D(0,2)=  .3720848963395903D 00
D(1,0)= -.1001351716909383D 01
D(1,1)=  .9110536675183730D 00
D(1,2)= -.3245630923013647D 00
D(2,0)=  .4551665866647211D 00
D(2,1)= -.4096692213601412D 00
D(2,2)=  .1435237900884555D 00
```

Figure 3.29 (c) Coefficients of $H_1(Z_1,Z_2)$



(a) (b)

(c) (d)

Figure 3.30 (a) Original Image
(b) Atmospheric Turbulance Blurred Image With Noise
(c) Restoration Using FFT
(d) Restoration Using Recursive Filter Implementation of Figure 3.28(a)

mentation is shown in Figure 3.28(a). The overall frequency response $H(\Omega_1,\Omega_2)$ of the implementation is:

$$
\begin{aligned}
H(\Omega_1,\Omega_2) &= |H_1(Z_1,Z_2)|^2 \\
&= H_1(Z_1,Z_2)\cdot H_1^*(Z_1,Z_2);\; Z_1=3^{-j\Omega_1},\; Z_2=e^{-j\Omega_2} \qquad (3.3.27)
\end{aligned}
$$

$H(\Omega_1,\Omega_2)$ is purely real and positive and has zero phase characteristics, and therefore, it can be used to approximate the desired restoration filter response. The filtering of data by $H_1^*(Z_1,Z_2)$ is performed in a manner described in Section 3.3.b.

The desired restoration filter response as shown in Figure 3.28(b) and Figure 3.28(c) shows the response realized by the recursive filter implementation, after the approximation. The recursive filter was designed using the cascaded design procedure. The impulse responses corresponding to the desired restoration filter specifications, and the recursive filter implementation is shown in Figure 3.29(a) and (b)+. The coefficients of the recursive filter $H_1(Z_1,Z_2)$ are shown in Figure 3.29(c). The restoration achieved using the recursive filter implementation is shown in Figure 3.30(d). The restoration that was carried out via convolution using the FFT is shown in Figure 3.30(c).

3.4 Summary

This chapter has presented the application of quarter plane recursive digital filters to problems in image processing. The applications considered are in the areas of image

---

+ The impulse responses are shifted by 16 pixels in each direction from the origin.

enhancement and image restoration. The problems of enhancement considered were high frequency emphasis and edge enhancement. In the case of restoration, the use of recursive filters are examined for motion, focus and atmospheric turbulance blurs. In the case of focus blur, the restoration appears to be a failure due to the inability of quarter plane filters to approximate the required filter response. However, in the case of motion and atmospheric tubuulance blurs, it is shown that successful restorations can be obtained with recursive filter implementations.

CHAPTER IV

DISCUSSIONS OF RESULTS AND EXTENSIONS

4.1 Introduction

In Chapter II, the design of both one dimensional and quarter plane two dimensional recursive digital filters, using linear programming, was presented where a simultaneous approximation is carried out to both the desired magnitude and linear phase characteristics. The application of quarter plane two dimensional recursive digital filters to image processing problems were considered in Chapter III. This chapter presents a discussion of some of the problems and the advantages associated with the linear programming design method for recursive digital filters and also the problems and the computational efficiencies associated with restorations of blurred images using recursive digital filter implementations. Also some extensions of the research work are discussed towards the end of the chapter.

4.2 Discussion of the Filter Design Method

The design examples presented earlier have shown that linear programming can be successfully used to design recursive filters which approximate given frequency domain magnitude response with linear phase characteristic. However, as in the case of many other linear programming methods, there are some problems that limit the range of filters that can be designed using this method. The type of stability constraints given by (2.3.34) and (2.4.10), used in the one and two dimensional recursive filter design method respectively, are not general stability constraints. Therefore, the filters that can be designed using this method belong to a subclass of all possible stable filter realizations. This is not a drawback in the case of two dimensional filters, since for many of the existing methods, the designed filters are also of a subclass of all possible stable filter realizations. However, the storage requirements for linear programming design of two dimensional filters are very large. Also, the computation times are quite high because of the large number of constraints involved in the linear program. Because of the high storage requirement, the range of the order of the two dimensional filters that can be designed using this method is limited. Thus, in this work, with 32 x 17 points of frequency domain specifications, the largest possible filter order that can be realized is M1=M2=N1=N2=3, (i.e., 16 coefficients in the numerator and 16 in the denominator). However, in the case of one dimension, this is not a problem, and filters of order 16 have been designed using this design procedure.

The design time for both one and two dimensional filters vary with the number of desired specification points and also the order of the filter. On the average, the time taken for one trial value of τ_s in the one dimensional filter design is approximately one minute. In the case of two dimensional filter design, the average time taken for one trial value of τ_c is approximately two minutes for specifications over a grid of points less than or equal to 21 x 11. However, for specifications over the grid points 32 x 17, the time for one trial value of τ_c is quite high (about ten minutes). All the designs presented in the examples were carried out using an IBM 360/65 computer in single precision arithmetic.

In spite of the problems mentioned earlier, the method has advantages compared to non-linear optimization and some of the other linear programming design methods. In the case of methods that use non-linear optimization [2,3,4,5,6,7,9] they require the specification of initial values of the parameters, to start the optimization, and also the final design is dependent on ~~the~~ initial values of the parameters. In addition, the convergence of the optimization is not guaranteed and even when the procedure converges, it generally converges to a local optimum. In comparison, the linear programming method of design does not require the specification of initial parameter values and also the optimum obtained is the absolute optimum consistent with the constraints of the problem. Compared to many of the linear programming methods [8,17,18,19,20,21], where it is possible to approximate only a magnitude squared specifications, this method can approximate to both magnitude and linear phase specifications.

4.3 Discussion on the Applications of Recursive Digital Filters to Image Processing

4.3.a Image Enhancement Applications

In image enhancement applications, such as high frequency emphasis and edge enhancement, the filter phase characteristics are zero or linear over most of the spatial frequency domain. In Chapter III, it is demonstrated that the linear programming design can be successfully employed in designing the desired filters with linear phase characteristics. It is also demonstrated that the linear phase characteristics are critical in edge enhancement applications. It should be noted that zero phase can also be achieved in the filters used for enhancement, by using the magnitude squared transfer function (non-causal) implementation shown in Figure 3.28(a). However, this results in almost twice the number of computations than that of the causal filter implementation.

4.3.b Image Restoration Applications

In Chapter III, suitable recursive filter implementations are shown for restorations due to motion, focus and atmospheric turbulance blurs. It is shown that the phase characteristics of the restoration filters are discontinuous in nature, in the case of motion and focus blurs, and therefore a non-causal impulse response filter implementation is required. However, in the case of atmospheric turbulance blur, the phase characteristics are either zero or linear phase and both causal and non-causal impulse response implementations are possible.

Prior to using recursive filter implementations for restoration purposes, the restorations were also carried out by convolution via FFT. The restorations so obtained, were used in judging the performance of the recursive filter implementations. In the case of motion blur, it can be seen from the examples provided that the restorations obtained are successful and are comparable to those obtained by convolution via FFT. In the case of atmospheric turbulance, it can be observed from the examples that both causal and non-causal impulse response recursive filter implementations provide good restorations, and are almost equivalent to those obtained by convolution via FFT. The usefulness of the linear programming design method is also demonstrated in designing the restoration filter for causal impulse response recursive filter implementation. In addition to providing good restorations, the recursive filter implementations also provide significant computational savings compared to convolution via FFT. A brief comparison of the two types of implementations follows.

The size of kernal used in the restoration by convolution via the FFT is restricted to 32 x 32. Since the FFT array size is set to 64 x 64, the filtering can be performed for two 32 x 32 sections of the image at a time, where the input for the FFT is complex. Hence, according to 31 , the number of complex operations involved for filtering an image of size 256 x 256 pixels is equal to 6,422,528 complex operations.+ In comparison, the number of real operations involved for the recursive filter implementation used, in the case of motion and atmospheric turbulance blurs, where successful restorations were obtained, are as in Table 4.1. It can be seen from this table that the number of computations required for recursive filtering is very much less than that required for the convolution via FFT. The average time required for filtering image data of size 256 x 256 using FFT was approximately equal to 32 minutes compared to an average of 8 minutes for the recursive filter implementations, using the Data

+ One complex operation is defined as one complex multiplication and an addition.

Type of Blur	Order of the Two Dimensional Recursive Digital Filter	Number of Real+ Operations for Filtering
MOTION BLUR		
Example 1: Blur Length=4 Pixels Angle=45^{o} w.r.t. Horizontal	One 1st and One 2nd Order Filter Sections.	3,145,728
Example 2: Blur Length=8 Pixels Angle=22.5^{o} w.r.t. Horizontal	Two 1st and One 2nd Order Filter Sections.	4,063,232
ATMOSPHERIC TURBULANCE BLUR		
Example 1: k=2 In Equation (4.4.26)	One 3rd Order Filter	2,162,688
Example 2: k=0.5 In Equation (4.4.26)	One 2nd Order Filter	2,228,224

Image Size = 256x256 Pixels

Table 4.1: Computations For Recursive Filtering (+One Real Operation Is Defined As One Real Multiplication and Addition. The Calculations Are Obtained According to 31).

General NOVA-840 mini-computer.

In the case of focus blur, the restoration is a failure. The cause of this can be clearly seen in Figure 3.19(b), which is an approximation obtained by the recursive filter implementation to the desired restoration filter response of Figure 3.18(a). This indicates the inability of quarter plane recursive digital filters to approximate the nearly circularly symmetric restoration filter response. In the preliminary study reported in [32] the restorations for focus blur were obtained using the same type of recursive filter implementations as shown in this thesis. The restorations obtained were reasonably good in comparison to restoration by convolution via FFT; however, the original images used in the study were simple computer generated images which contained only 16 gray levels in comparison to the 256 levels of the original images used in this thesis.

4.4 Extensions

This section discusses the possible extensions and further study of the filter design technique, and the further applications of two dimensional recursive digital filters that are considered in this thesis.

4.4.a On Two Dimensional Filter Design

The thesis has presented the linear programming technique for quarter plane two dimensional recursive filter design. Therefore, one might consider the extension of this technique to design half plane filters that were discussed in Chapter I of this thesis. The basic design procedure remains unaltered; however, the linear stability constraint may require some modifications. These modifications will have to take into account the change in the stability criteria from quarter plane filters to that of half plane filters.

4.4.b On Recursive Digital Filter Applications In Image Processing

Chapter IV of the thesis considers the application of recursive digital filters in image processing; in particular, the application of quarter plane filters. However, the case of focus blur indicates the failure of quarter plane filter implementations. Hence, some further work is required, specifically in the area of half plane filter applications to image processing. It may also be worthwhile, not only to consider half plane filter application in focus deblurring, but in all the problems of image processing that have been studied in this thesis.

CHAPTER V

CONCLUSIONS

A new technique of designing one and two dimensional recursive digital filter transfer functions, which can approximate simultaneously linear phase and arbitrary magnitude specifications is presented. The technique sets up a linear programming problem which is in terms of the filter coefficients and the desired constant group delay (linear phase), which is then solved iteratively for the best approximation to the given specifications. Since the approximation is performed using the linear programming technique which is a linear optimization procedure, the constraints on the filter coefficients that can be used to obtain stable filters, are required to be linear in form. In Chapter II, such stability constraints are proposed and proofs of the sufficiency of these constraints are also provided. A wide range of design examples are provided in both one and two dimensional cases and some limitations of the linear programming design technique are also discussed.

The thesis also has presented the application of recursive digital filters to image processing. A brief evaluation of various two dimensional recursive digital filter techniques are carried out and it is shown that the linear programming technique of Chapter II and the non-linear optimization design technique of [4] are useful in designing filters for image processing applications. The applications of recursive digital filters are considered in the areas of a) image enhancement and b) image restorations. Examples of enhancements using recursive filters are shown for the cases of high frequency emphasis and edge enhancement and importance of linear phase characteristics is stressed by way of an edge enhancement example. The applications of recursive digital filters in image restoration is considered for the cases of motion, focus and atmospheric turbulance blurs. Using suitable restoration filter specification, it is shown, with examples, that recursive digital filter implementations cannot only provide good restorations, but are also computationally advantageous compared to restoration by convolution via FFT, for the cases of motion and atmospheric turbulance blurs. In the case of focus blur, the restoration is a failure and the cause of the failure is attributed to the inability of quarter plane filters to adequately meet the circularly symmetric specifications of the desired restoration filter.

REFERENCES

1. L.R. Rabiner and B. Gold, Theory and Applications of Digital Signal Processing, Englewood Cliffs, N.J., Prentice Hall, 1975.

2. G.A. Maria and M.M. Fahmy, "An ℓp Design Technique for Two Dimensional Digital Recursive Filters," IEEE Trans. on ASSP, Vol. 22, No. 1, Feb. 1974.

3. ______________________, Lp Approximation of the Group Delay Response of One and Two Dimensional Filters," IEEE Trans. on Circuits and Systems, Vol. 21, No. 3, pp. 431-436, May 1974.

4. P.A. Ramamoorthy and L.T. Bruton, "Frequency Domain Approximation of Multidimensional Discrete Recursive Filters," Proc. IEEE Symposium on Circuits and Systems, pp. 654-657, 1977.

5. ______________________, "Design of Stable Two Dimensional Recursive Filters and Application in Image Processing," Proc. 20th Midwest Symposium on Circuits and System, Aug. 1977.

6. M.A. Sid-Ahmed, "Analysis and Synthesis of One and Two Dimensional Recursive Digital Filters," Ph.D. dissertation, Dept. of Electrical Engineering, U. of Windsor, Windsor, Ontario, Canada, 1974.

7. G.A. Jullien, W.C. Miller, A. Chottera, "Design Procedures for a Class of Stable Two Dimensional Recursive Digital Filters," Proc. 18th Mid-West Symposium on Circuits and Systems, pp. 309-313, August 1975.

8. D.E. Dudgeon, "Two Dimensional Recursive Filter Design Using Differential Correction," IEEE Trans. On ASSP, Vol. 23, No. 3, pp. 264-267, June 1975.

9. R.E. Twogood and S.K. Mitra, "Computer-Aided Design of Separable Two Dimensional Digital Filters," IEEE Trans. on ASSP, Vol. 25, No. 2, pp. 165-169, April 1977.

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

11. N.A. Pendergrass, S.K. Mitra and E.I. Jury, "Spectral Transformation for Two Dimensional Digital Filters," IEEE Trans. on Circuits and Systems, Vol. 23, No. 1, pp. 26-35, Jan. 1976.

12 H. Chang and J.K. Aggarwal, "Design of Two Dimensional Recursive Filters by Interpolation," IEEE Trans. on Circuits and Systems, Vol. 24, No. 26, pp. 281-291, June 1977.

13 B. Nowrozian, M. Ahmadi and R.A. King, "On the Space Domain Design Techniques of N-Dimensional Recursive Digital Filters," Proc. IEEE Int. Conf. on ASSP, pp. 531-534, April 1977.

14 T.S. Huang, Picture Processing and Digital Filtering, New York, Springer-Verlag, 1975.

15 J.V. Hu and L.R. Rabiner, "Design Techniques for Two Dimensional Digital Filters," IEEE Trans. on Audio and Electroacoustics, Vol. 20, No. 4, pp. 249-257, Oct. 1972.

16 S.I. Gauss, Linear Programming; Methods and Applications, 3rd Edition, McGraw-Hill, N.Y. 1969.

17 P. Thajchayapong and P.J.W. Rayner, "Recursive Digital Filter Design by Linear Programming," IEEE Trans. on Audio and Electroacoustics, Vol. 21, No. 2, pp. 107-112, April 1973.

18 D. Swanton, "Linear Programming Design of Recursive Digital Filters," Masters Thesis, Dept. of Electrical Engg., McGill University, Canada, March, 1973.

19 L.R. Rabiner, N.Y. Graham and H.D. Helms, "Linear Programming Design of IIR Digital Filters with Arbitrary Magnitude Function," IEEE Trans. on ASSP, Vol. 22, No. 2, pp. 117-123, April, 1974.

20 D.E. Dudgeon, "Recursive Filter Design Using Differential Corrections," IEEE Trans. on ASSP, Vol. 22, No. 6, pp. 443-448, Dec. 1974.

21 F.J. Brophy and A.C. Salazar, "Two Design Techniques for Digital Phase Networks," Bell System Technical Journal, Vol. 54, No. 4, pp. 767-781, April 1975.

22 A. Chottera and G.A. Jullien, "Designing Near Linear Phase Recursive Filter Using Linear Programming," Proc. of IEEE Int. Conf. on ASSP, pp. 88-92, May 1977.

23 T.M. Cannon, "Digital Image Deblurring by Non-Linear Homomorphic Filtering," Ph.D. dissertation, Dept. of Electrical Engineering, Division of Computer Science, University of Utah, U.S.A., August 1974.

24 R.J. Arguello, H.R. Sellner and J.A. Stuller, "Transfer Function Compensation of Sampled Imagery," IEEE Trans. on Computers, Vol. 21, No. 7, pp. 812-818, July 1972.

25 B.R. Hunt, "Computational Considerations in Digital Image Enhancement," Proc. Two-Dimensional Digital Signal Processing Conference, U. of Missouri, Columbia, Mo., U.S.A., pp. 1.4.1-1.4.7, Oct. 1971.

26 ________, "Data Structures and Computational Organization in Digital Image Enhancement," Proc. of IEEE, Vol. 60, No. 7, pp. 884-887, July 1972.

27 D.S. Gooden and C.H. Farmer, "Two Dimensional Filtering for Image Enhancement in Nuclear Medicine," Proc. of Two Dimensional Digital Signal Processing Conf. U. of Missouri, Columbia, Mo., U.S.A., pp. 2.2.1-2.2.15, Oct. 1971.

28 J.W. Cooley, P. Lewis and P.D. Welch, "The East Fourier Transform and Its Applications," IEEE Trans. on Education, Vol. 12, pp. 27-34, March 1969.

29 E.L. Hall and A. Kahveci, "High Resolution Image Enhancement Techniques," Proc. of Two Dimensional Digital Signal Processing Conf., U. of Missouri, Columbia, Mo., U.S.A., pp. 1.5.1-1.5.3, Oct. 1971.

30 J.W. Modestino and R.W. Fries, "Edge Detection in Noisy Images Using Recursive Digital Filtering," Computer Graphics and Image Processing, Vol. 6, No. 5, Oct. 1977.

31 E.L. Hall, "A Comparison of Computations for Spatial Frequency Filtering," Proc. of IEEE, Vol. 60, No. 7, pp. 887-891, July 1972.

32 A. Chottera and G.A. Jullien, "Recursive Digital Filters in Image Processing," Proc. of IEEE Int. Conf. on ASSP, pp. 757-760, April 1978.

33 G.A. Matthews, F. Davis, J.C. McKee, R.C. Cavin and G.M. Sibbles, "Approximating Transfer Functions by Linear Programming," Proc. of Fist Annual Allerton Conf. on Circuits and Systems, pp. 191-204, 1963.

34 E.I. Jury, Theory and Applications of the Z Transform Method, John Wiley & Sons, Inc., New York, 1964.

35 E.A. Robinson, Statistical Communication and Detection, Hafner, New York, 1967.

36 H.C. Andrews, Computer Techniques in Image Processing, Academic Press, New York, 1970.

37 A. Rosenfeld and A.C. Kak, Digital Picture Processing, Academic Press, New York, 1976.

38 B.R. Hunt, "Digital Image Processing," Proc. of IEEE, Vol. 63, No. 4, pp. 693-708, April 1975.

39 T.G. Stockham, T.M. Cannon and R.B. Ingebretsen, "Blind Deconvolution Through Digital Signal Processing," Proc. of IEEE, Vol. 63, No. 4, pp. 678-692, Arpil 1975.

40. M.P. Ekstrom and J.W. Woods, "Two Dimensional Spectral Factorization with Applications to Recursive Digital Filtering," IEEE Trans. on ASSP, Vol. 24, pp. 115-127, April 1976.

41 H. Chang and J.K. Aggarwal, "Design of Semi-Causal Two Dimensional Recursive Filters," Proc. IEEE Int. Conf. on ASSP, pp. 777-781, April 1978.

42 M.P. Ekstrom, R.E. Twogood and J.W. Woods, "Design of Stable 2-D Half Plane Recursive Filters Using Spectral Factorization," Proc. IEEE Int. Conf. on ASSP, pp. 761-764, April 1978.

43 J.L. Horner, "Optical Restoration of Images Blurred by Atmospheric Turbulance," Applied Optics, Vol. 9, No. 1, January 1970.

APPENDIX A

Digital Filtering Fundamentals

One and Two Dimensional Filtering

Digital filters fall into two classes. Filters whose spatial response contains a finite number of non-zero samples are called Finite Impulse Response (FIR) filters, and those whose spatial response contains, in general, an infinite number of non-zero samples†, are called Infinite Impulse Response (IIR) or recursive filters.

In the one dimensional case, the output sequence $y(m)$, of a FIR filter, assuming an input sequence $x(m)$, is given by [1]:

$$y(m) = \sum_{k=0}^{K-1} h(k)x(m-k) \tag{A.1}$$

where $h(k)$ is the impulse response defined over the interval $0 \leq k \leq K-1$. Similarly, in the two dimensional case, the output array $y(m,n)$ can be written as [1]:

$$y(m,n) = \sum_{k=0}^{K-1} \sum_{\ell=0}^{L-1} h(k,\ell) \cdot x(m-k,n-\ell) \tag{A.2}$$

where $x(m,n)$ is the input array and $h(k,\ell)$ is the impulse response defined over the interval $0 \leq k \leq (K-1)$, $0 \leq \ell \leq (L-1)$.

A one dimensional recursive filter is characterized by the difference equation [1]:

$$y(m) = \sum_{k=0}^{N} a(k)x(m-k) - \sum_{\ell=1}^{M} b(\ell)y(m-\ell) \tag{A.3}$$

---

† This definition is used loosely, since convergent properties are required of the filter.

Once again x(m) and y(m) are the input and output respectively, and y(m) is computable (i.e., the IIR filter is said to be recursible) for $m \geq 0$. The impulse response, h(m), corresponding to (1.2.3) is causal, i.e., $h(m) = 0$ for $m < 0$ and it extends up to infinity for positive values of m. However an extension of similar characteristics to the two dimensional case yields only a specific class of two dimensional recursive filters 40 . This will be shown in the following discussion.

In the two dimensional case, a general difference equation that characterizes a two dimensional IIR filter can be written as 40 :

$$y(m,n) = \sum_{(k,\ell)\in R_a}\sum a(k,\ell)x(m-k,n-\ell) - \sum_{\substack{(i,j)\in R_b \\ i+j\neq 0}}\sum b(i,j)y(m-i,n-j) \qquad (A.4)$$

where x and y are the input and output arrays respectively, and R_a and $R_b \in I_s^2$, where I_s is the set of integers. Before discussing further about the difference equation (A.4), it is important to understand about the recursibility of a two dimensional recursive digital filter.

Definition : A two dimensional IIR filter is said to be recursible if for every output point (m,n), the output mask covers only points which have been previously computed.

In the above definition, the term output mask, simply refers to a rectangle that encloses the appropriate samples of the output array y for computation of an output point $y(m,n)$ (an input mask can also be defined in a similar manner). Now, consider the following case of R_a and R_b, such that:

$$R_a = \{(k,\ell) \mid 0 \leqslant k \leqslant K1,\ 0 \leqslant \ell \leqslant L1\}$$

$$R_b = \{(i,j) \mid 0 \leqslant i \leqslant I1,\ 0 \leqslant j \leqslant J1\} \tag{A.5}$$

The difference equation corresponding to this case can be written as

$$y(m,n) = \sum_{k=0}^{K1} \sum_{\ell=0}^{L1} a(k,\ell)x(m-k,n-\ell)$$

$$- \sum_{i=0}^{I1} \sum_{j=0}^{J1} b(i,j)y(m-i,n-j);\quad i+j \neq 0 \tag{A.6}$$

In (A.6), if I1, J1, K1, L1 are all chosen to be equal to 2, then the recursive filtering operation corresponding to this case can be described by Figure 1.1. In this figure, to produce the value of $y(m,n)$, which is the desired output hole, the input and output masks are placed over the appropriate samples of the input and output arrays respectively and each sample is multiplied by the coefficient associated with that position in the mask and the products are summed. From the type of the output mask, it can be seen that the recursive filter associated with Equation (A.6), is recursible either column or rowwise. Consider, for example, another case of R_a and R_b, such that,

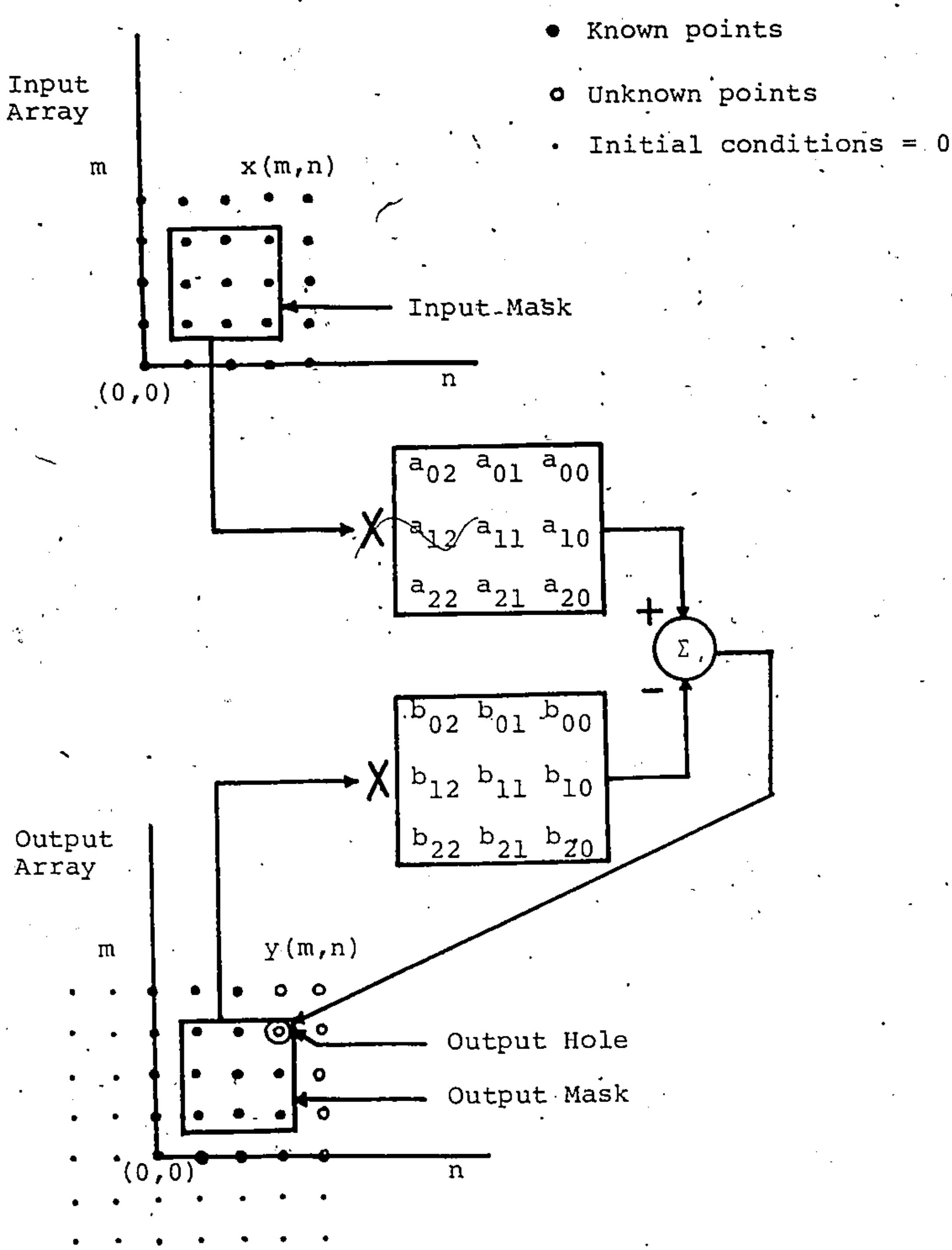


Figure A1 An Example of Input and Output Masks to Calculate y(3,3) for Equation (1.2.6).

$$R_a = (k,\ell) \mid 0 \leq k \leq K2 \text{ if } 0 \leq \ell \leq L2$$
$$\text{and} \qquad\qquad (A.7)$$
$$-K4 \leq k \leq -K3 \text{ if } 0 < \ell \leq L2$$

$$R_b = (i,j) \mid 0 \leq i \leq I2 \text{ if } 0 \leq j \leq J2$$
$$\text{and}$$
$$-I4 \leq i \leq -I3 \text{ if } 0 < j \leq J2$$

This results in an output sequence y(m,n) given by:

$$y(m,n) = \sum_{k=0}^{K2} \sum_{\ell=0}^{L2} a(k,\ell)x(m-k,n-\ell) + \sum_{k=-K3}^{-K4} \sum_{\ell=1}^{L2} a(k,\ell)x(m-k,n-\ell)$$
$$- \sum_{\substack{i=0 \\ i+j\neq 0}}^{I2} \sum_{j=0}^{J2} b(i,j)y(m-i,n-j) - \sum_{i=-I3}^{-I4} \sum_{j=1}^{J2} b(i,j)y(m-i,n-j) \qquad (A.8)$$

Here again, if I2=J2=I3=I4=K2=L2=K4=K3=1, then the recursive filtering operation given in (A.8) can be described by Figure A2. An examination of the output mask for this case indicates that this filter is recursible only columnwise.

According to Huang 14 , a two dimensional recursive digital filter is said to be causal if the impulse response h(m,n) is zero for m or n less than zero. Returning back to Figure A1, one can see that impulse response of the IIR filter (A.6) is spread over only the upper quadrant of the right half plane in the spatial domain and therefore it is a causal filter. This type of filter is also referred to as a quarter plane recursive filter, since its impulse response

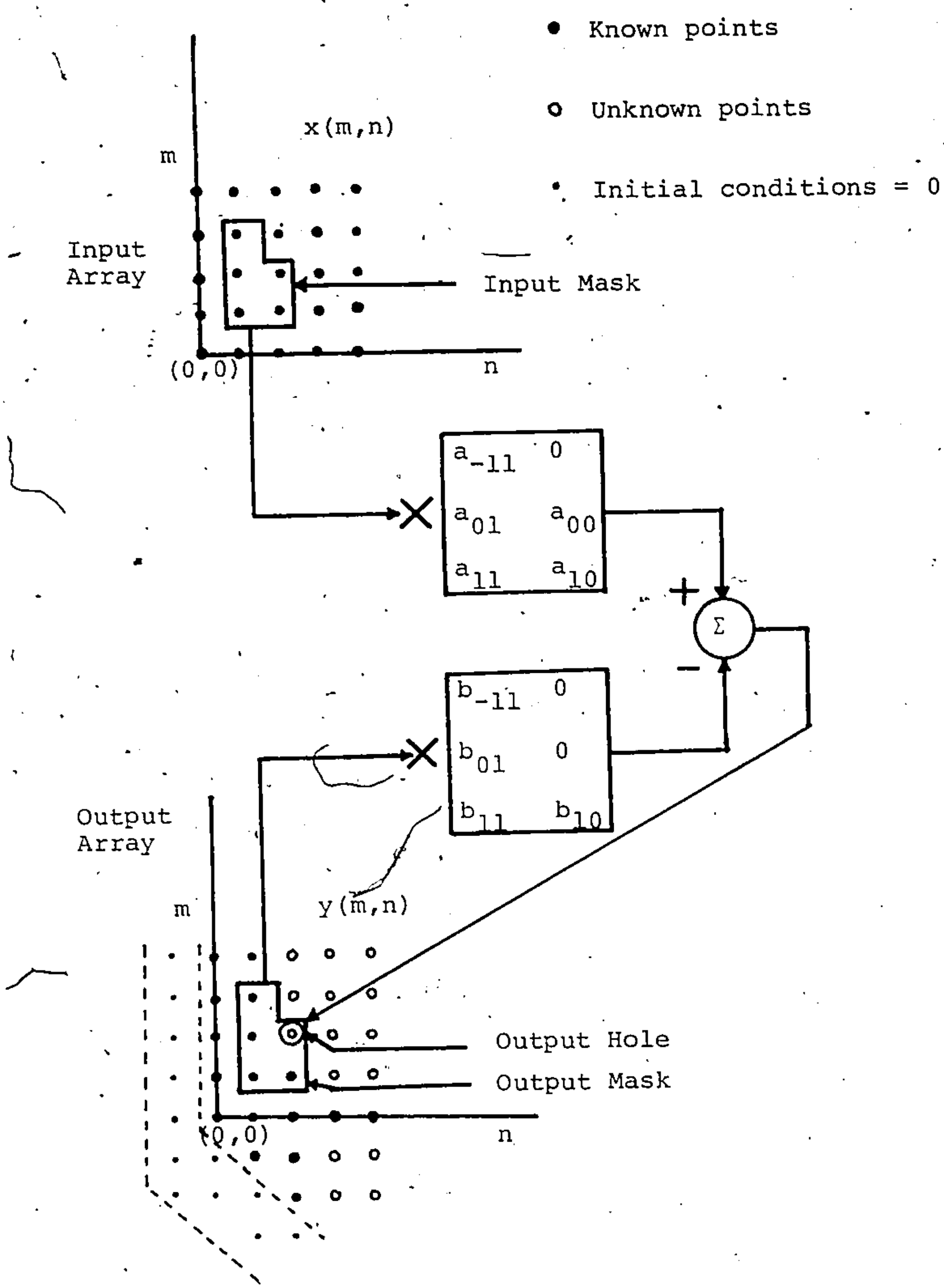


Figure A2 An Example of Input and Output Masks to Calculate y(2,2) for the Equation of (A.8).

is confined to one quarter of the spatial domain. Compared to this, however, from Figure A2, it can be seen that the recursive filter of (A.8) has its impulse response spread over the right half plane of the spatial domain and it is referred to as a semicausal recursive filter with respect to the definition of causal filters. Because the impulse response of this type of filter spreads over half the spatial domain, it is also referred to as a half plane filter.

The work reported in this thesis deals with only the quarter plane filters and therefore, from here on, any reference to two dimensional recursive digital filter should be assumed to be a reference to a quarter plane two dimensional recursive digital filter, unless specified. For more detailed information on half plane filters, the interested reader should refer to recent works reported in 40,41,42 .

Z Transforms and Filter Transfer Functions

Given a sequence x(n), defined for all n, the Z transform is defined as,

$$x(Z) = \sum_{n=-\infty}^{\infty} x(n)Z^{n}; \; Z = e^{-ST} \qquad (A.9)$$

where S and Z are complex variables. In (A.9) x(n) is a sequence that is obtained by sampling a continuous signal x(t) once every T units of time. T represents the sampling period and its choice is based on the sampling theorem (see [1]). When the complex variable S and the sampling period T are such that

$$S = jw; \; T = \frac{2\pi}{w_s} \qquad (A.10)$$

where w is the continuous frequency variable and w_s is the sampling frequency, then the complex variable Z can be written as:

$$Z = e^{-j\Omega} \qquad (A.11)$$

In (1.3.3) $\Omega = (\pi w/w_{s/2})$ is referred to as the normalized frequency variable†. For various values of Ω, Z takes on values on the unit circle in the Z plane. Using (A.11) in (A.9), one finds that the evaluation of the Z transform on the unit circle results in:

$$X(Z)\Big|_{Z=e^{-j\Omega}} = X(e^{-j\Omega}) = \sum_{n=-\infty}^{\infty} x(n)e^{-j\Omega n}$$

which is the fourier transform of the sampled sequence x(n).

Using the above definition, it is now possible to derive the transfer functions for FIR and recursive digital filters. Thus for a causal FIR filter whose impulse response h(n) is zero for values of n outside of the range $0 < n < (N-1)$, the transfer function can be written as:

$$H(Z) = \sum_{n=0}^{(N-1)} h(n)Z^n \qquad (A.12)$$

Similarly, the transfer function correspon ing to the causal recursive (IIR) filter is obtained as:

$$H(Z) = \sum_{n=0}^{\infty} h(n)Z^n = \frac{\sum_{i=0}^{N} a(i)Z^i}{1 + \sum_{j=1}^{M} b(j)Z^j} \qquad (A.13)$$

†Expressed in radians.

where we assume that no roots of the denominator are cancelled by roots of the numerator. Most often N is less than or equal to M and the filter is referred to as an M^{th} order recursive filter. If $N > M$, then the filter can be taken to be an M^{th} order recursive filter with an $(N-M)^{th}$ order FIR filter. Another important quantity in filter theory is the frequency response $H(\Omega)$ of the filter. This is obtained by evaluating the transfer function $H(Z)$ on the unit circle in the Z plane, i.e., $Z = e^{-j\Omega}$.

A two dimensional Z transform can be defined in exactly the same manner as the one dimensional Z transform. Hence, for a two dimensional sequence $x(m,n)$, defined for all m and n, the Z transform is defined as:

$$X(Z_1,Z_2) = \sum_{m=-\infty}^{\infty} \sum_{n=-\infty}^{\infty} x(m,n)\ Z_1^m\ Z_2^n;\ Z_1=e^{-S_1T_1},\ Z_2=e^{-S_2T_2} \tag{A.14}$$

where S_1, S_2, Z_1 and Z_2 are complex variables.

$x(m,n)$ is a sequence obtained by sampling a continuous two dimensional signal $x(t_1,t_2)$ at intervals of T_1 and T_2 units in spatial directions x and y. With complex variables S_1, S_2 and the sampling periods T_1 and T_2, so that:

$$S_1=jw_1,\ S_2=jw_2,\ T_1 = (2\pi/w_{s_1}),\ T_2 = (2\pi/w_{s_2}) \tag{A.15}$$

where w_1 and w_2 are continuous spatial frequency variables and w_{s_1} and w_{s_2} are the frequencies at which the signal $x(t_1,t_2)$ is sampled in x and y spatial directions, the complex variables Z_1 and Z_2 can be written as:

$$\left.\begin{aligned} Z_1 &= e^{-j\Omega_1} \; ; \; \Omega_1 = \pi w_1/(w_{s_{1/2}}) \\ Z_2 &= e^{-j\Omega_2} \; ; \; \Omega_2 = \pi w_2/(w_{s_{2/2}}) \end{aligned}\right\} \tag{A.16}$$

In (A.17) Ω_1 and Ω_2 are referred to as normalized spatial frequency variables in radians and for various values of Ω_1 and Ω_2, Z_1 and Z_2 take on values on the unit circles in the Z_1 and Z_2 planes respectively. Therefore from (A.14) the evaluation of Z transform on the unit bidisc[+] results in:

$$X(Z_1, Z_2)\Big|_{\substack{Z_1 = e^{-j\Omega_1} \\ Z_2 = e^{-j\Omega_2}}} \cancel{= x(e^{-j\Omega_1}, e^{-j\Omega_2})}$$

$$= \sum_{m=-\infty}^{\infty} \sum_{n=-\infty}^{\infty} x(m,n) e^{-j(m\Omega_1 + n\Omega_2)} \tag{A.17}$$

which is the two dimensional fourier transform of the sequence $x(m,n)$.

It follows from the above discussion, that given a FIR filter whose impulse response $h(m,n)$ is zero outside the region $0 \leq m \leq M1$ and $0 \leq n \leq N1$, the two dimensional transfer function can be obtained as:

$$H(Z_1, Z_2) = \sum_{m=0}^{M1} \sum_{n=0}^{N1} h(m,n) Z_1^m Z_2^n \tag{A.18}$$

For a two dimensional recursive digital filter, the transfer function can be obtained from the difference equation of (A.4) as:

[+] Unit bidisc $\triangleq \{Z_1, Z_2 : |Z_1| < 1 \text{ and } |Z_2| > 1\}$

$$H(Z_1,Z_2) = \frac{\sum\sum_{(k,\ell)\ R_a} a(k,\ )Z_1^k\ Z_2}{1 + \sum\sum_{\substack{(i,j)\ R_b \\ i+j\neq 0}} b(i,j)Z_1^i\ Z_2^j} . \qquad \text{(A.19)}$$

Depending upon the choice of R_a and R_b the filter of (A.19) can either be a half plane or a quarter plane digital filter. The frequency response $H(\Omega_1,\Omega_2)$ of (A.18) or (A.19) can be obtained by evaluating the respective transfer functions around the unit bidisc.

Order of Two-Dimensional Recursive Digital Filter

Unlike the one dimensional case, the general approach in the two dimensional case has been to specify the number of filter coefficients, instead of an order for the fflter. However, as in the approach of some authors 2,3,4,5 , it is convenient to specify an order for the two dimensional filter for a specific case of the sets R_a and R_b of (A.19). Hence if R_a and R_b are as indicated in (A.5), where K1, L1, I1 and J1 are all chosen to be equal to an integer K, then the filter of (A.19) is referred to as a quarter plane filter of order K. A straightforward extension of this can be carried out to the case of half plane filters. The above notation is adopted, throughout this thesis, in referring to the order of a two dimensional recursive digital filter. In situations when R_a and R_b do not belong to the specific case indicated above (which is not very common), the filter is referred to in terms of the number of numerator and denominator coefficients.

Group Delay of a Filter 1

The group delay of a filter is a measure of average spatial or time delay as a function of frequency.

The transfer function H(Z) of a one dimensional filter can be written in the form:

$$H(Z) = |H(Z)|\, e^{j\beta(Z)} \tag{A.20}$$

where $|H(Z)|$ is the magnitude and $\beta(Z)$ is the phase response of H(Z). The phase response of the filter is defined as:

$$\beta(\Omega) = \beta(Z)\Big|_{Z=e^{j\Omega}} = \tan^{-1}\left\{\frac{\text{Im}\ \ H(Z)}{\text{Re}\ \ H(Z)}\right\}\Bigg|_{Z=e^{-j\Omega}} \tag{A.21}$$ [+]

The group delay is now defined as:

$$\tau(\Omega) = -\frac{d\beta(\Omega)}{d\Omega} \tag{A.22}$$

$\tau(\Omega)$ can be expressed in terms of H(Z) as:

$$\tau(\Omega) = \text{Re}\left\{\frac{Z}{H(Z)}\ ,\ \frac{dH(Z)}{dZ}\right\}\Bigg|_{Z=e^{-j\Omega}} \tag{A.23}$$

The group delays for two dimensional filters can be defined in a manner similar to the one dimensional case. For a two dimensional filter, there exists group delays in each of the spatial directions and are functions of both the spatial frequencies.

Consider a two dimensional filter transfer function

+ Im{ }and Re{ }refers to imaginary part of and real part of, respectively.

expressed in a form similar to (A.20) as:

$$H(Z_1,Z_2) = |H(Z_1,Z_2)|\ e^{j\beta(Z_1,Z_2)} \qquad (A.24)$$

As before, the phase response is defined as:

$$\beta(\Omega_1,\Omega_2) = \beta(Z_1,Z_2)\Big|_{\substack{Z_1=e^{-j\Omega_1}\\ Z_2=e^{-j\Omega_2}}}$$

$$= \tan^{-1}\left\{\frac{\mathrm{Im}\ H(Z_1,Z_2)}{\mathrm{Re}\ H(Z_1,Z_2)}\right\}\Bigg|_{\substack{Z_1=e^{-j\Omega_1}\\ Z_2=e^{-j\Omega_2}}} \qquad (A.25)$$

The group delays can now be defined as (9):

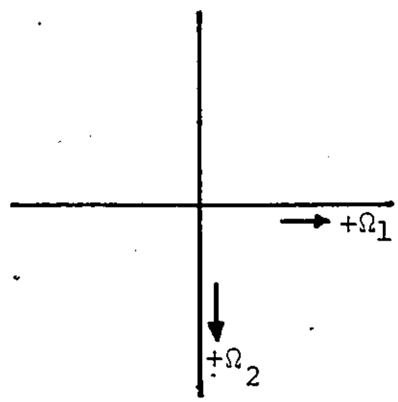
$$\tau_1(\Omega_1,\Omega_2) = -\frac{\partial\beta(\Omega_1,\Omega_2)}{\partial\Omega_1}$$

$$\tau_2(\Omega_1,\Omega_2) = -\frac{\partial\beta(\Omega_1,\Omega_2)}{\partial\Omega_2} \qquad (A.26)$$

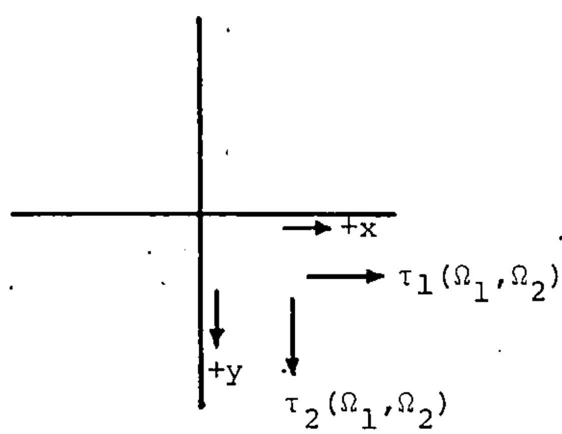
shows the spatial frequency domain and the corresponding delay directions in the spatial domain.

$\tau_1(\Omega_1,\Omega_2)$ and $\tau_2(\Omega_1,\Omega_2)$ can be expressed in terms of the two dimensional filter transfer function $H(Z_1,Z_2)$ as:

$$\tau_1(\Omega_1,\Omega_2) = \mathrm{Re}\left\{\frac{Z_1}{H(Z_1,Z_2)}\cdot\frac{\partial H(Z_1,Z_2)}{\partial Z_1}\right\}\Bigg|_{\substack{Z_1=e^{-j\Omega_1}\\ Z_2=e^{-j\Omega_2}}} \qquad (A.27)$$



(a) Frequency Domain



(b) Spatial Domain

Figure A3 Directions of Delays in Spatial Domain.

$$\tau_2(\Omega_1,\Omega_2) = \mathrm{Re}\left\{\frac{Z_2}{H(Z_1,Z_2)} \quad \frac{\partial H(Z_1,Z_2)}{\partial Z_2}\right\}\Bigg|_{\substack{Z_1=e^{-j\Omega_1}\\ Z_2=e^{-j\Omega_2}}} \tag{A.28}$$

Stability of IIR Digital Filters

One Dimensional Case

In the case of one dimensional filters, a necessary and sufficient condition for stability is:

$$\sum_{n=0}^{\infty} |h(n)| < \infty \tag{A.29}$$

where h(n) is the impulse response of the given filter.

FIR filters satisfy the above condition. Since their impulse response is defined only over a short period, i.e., h(n) is defined for $N_1 < n < N_2$. However, in the case of recursive filters, the impulse response extends up to infinity and in order to meet the stability criterion, it is required that poles of the transfer function be outside the unit circle.

Thus given an IIR filter,

$$H(Z) = \frac{P(Z)}{Q(Z)} = \frac{\sum_{n=0}^{N} a(n)Z^n}{\sum_{m=0}^{M} b(m)Z^m}, \quad Z=e^{-j\Omega} \tag{A.30}$$

in order for this to be stable, the singularities (often referred to as zeros) of Q(Z) should lie outside the unit circle in the Z-plane. Therefore in designing recursive filters of the form given in (A.30), proper stability constraints have to be imposed on the coefficients b(m), so that the zeros of Q(Z) lie outside the unit circle in the Z-plane.

Two Dimensional Case

In the two dimensional case, conditions for stability become more involved. The following definition 41, is useful in the statement of stability theorems for two dimensional recursive filters.

Definition 41: In the two dimensional complex space, we define the following spatial regions by:

$$T^2 = \{(Z_1,Z_2) \mid |Z_1| = 1, \ |Z_2| = 1\};$$

$$D{+}{+} = \{(Z_1,Z_2) \mid |Z_1| \leqslant 1, \ |Z_2| \leqslant 1\};$$

$$D{-}{+} = \{(Z_1,Z_2) \mid |Z_1| \geqslant 1, \ |Z_2| \leqslant 1\}; \qquad (A.31)$$

$$D{-}{-} = \{(Z_1,Z_2) \mid |Z_1| \geqslant 1, \ |Z_2| \geqslant 1\};$$

$$D{+}{-} = \{(Z_1,Z_2) \mid |Z_1| \leqslant 1, \ |Z_2| \geqslant 1\}$$

T^2 is generally referred to as the unit bicircle.

Now, let $H(Z_1,Z_2)$ be a rational two dimensional transfer function such that:

$$H(Z_1,Z_2) = \frac{A(Z_1,Z_2)}{B(Z_1,Z_2)} = \sum_{(k,\ell)\in R_a} a(k,\ell)\, Z_1^k\, Z_2^\ell \Bigg/ \sum_{(i,j)\in R_b} b(i,j)\, Z_1^i\, Z_2^j \qquad (A.32)$$

where $Z_1=e^{-j\Omega_1}$ and $Z_2=e^{-j\Omega_2}$ and R_a, R_b are as defined in (1.2.5) and (1.2.7). The stability theorems can then be stated as follows:

Theorem 1 41 : A causal (quarter plane) filter of definition given in 14 is stable if $B(Z_1,Z_2) \neq 0$ for (Z_1,Z_2) D++.

The above theorem is the same as Shanks' stability theorem for causal filters 14 .

Theorem 2 41 : A semicausal filter is said to be stable if $B(Z_1,Z_2) \neq 0$ for (Z_1,Z_2) (D++ D-+) and $B(Z_1,0) \neq 0$ for $|Z_1| < 1$.

APPENDIX B

Digital Image Processing Fundamentals

Digital Image Processing

Image processing is a broad area that deals with manipulation of pictorial data, which are inherently two dimensional in nature. It encompasses various areas such as image enhancement, restoration, pictorial pattern recognition and efficient picture coding for picture transmission and storage.

In order to understand various techniques of digital image processing, first it is important to understand some of the basic concepts in digital image processing such as image formation, recording and sampling. A brief description of the above follows.

The principle elements of an image formation system can be described as shown in Figure B1. The black box in Figure 1.4 acts upon a radiant energy component of the object to generate the image. Thus the image at point (x,y) can be considered as a function of contributions in a (possibly infinite) neighbourhood of (x_1,y_1). If $g_i(x,y)$ represents image radiant energy distribution and $i(x,y)$ represents object radiant energy distribution, then a general description of g_i is given by:

$$g_i(x,y) = \int_{-\infty} \int_{-\infty} h(x,y,x_1,y_1)\ i(x_1,y_1)dx_1dy_1 \qquad (B.1)$$

In (B.1), h is assumed to merely weigh the object distribution as a scalar multiplier and h is referred to as a point spread function. As such h in (B.1) represents a space variant point spread function. If h is made position invariant however, then g_i can be expressed as:

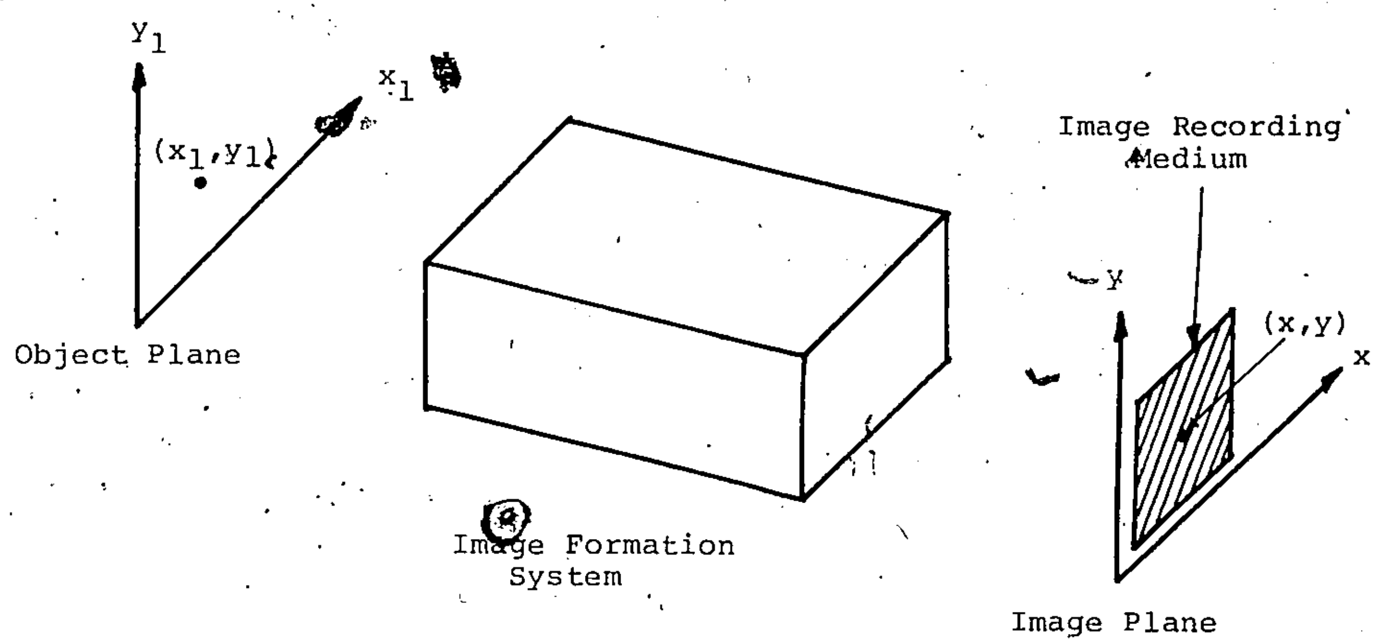


Figure B1 Schematic of Image Formation System.

$$g_i(x,y) \doteq \int_{-\infty} \int_{-\infty} h(x-x_1,y-y_1)\ i(x_1,y_1)dx_1dy_1 \qquad \text{(B.2)}$$

This is true in situations such as optical image formation systems.

The two major means of recording images are photochemical and photoelectric. Example of photoelectric recording is a television camera and in the case of photochemical, photographic film. Part of this dissertation dealing with image processing is concerned with images that are recorded in a photographic film and therefore the process of image formation and recording associated with this case can be described by a model shown in Figure B2. This is a simplified block diagram that shows the image formation system with spatial response h. The intensity of the recorded image is given by:

$$g(x,y) = h(x,y) \circledast i(x,y) + n(x,y)^{+} \qquad \text{(B.3)}$$

where n(x,y) is noise, which is modeled as additive. Although the modelling of noise as additive may not always reflect reality (an example is film grain noise [38], which is multiplicative) it does however appear to provide good results in most image processing problems [23].

The intensities g(x,y), recorded in the film can now be obtained by projecting a spot of light in a raster scanning sequence, e.g., left-to-right, top-to-bottom and sampling the intensity of the transmitted light at given coordinate spacings. In a system such as flying spot scanner,

+ $\circledast$ indicates convolution

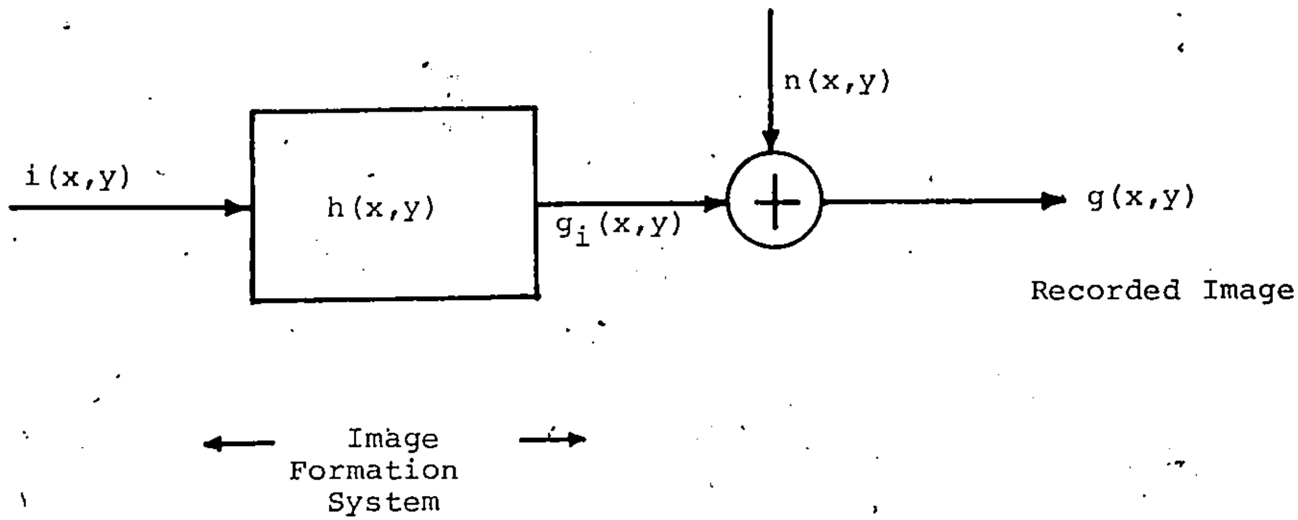


Figure B2 Block Diagram Representation of Image Formation and Recording.

the light that is transmitted through the film is received by a photomultiplier tube, which generates an equivalent electrical signal at its output. This electrical output is converted to an integer number, via an analog to digital converter (A/D) which is then stored in a digital computer. Thus a digital image is formed as a two dimensional array of numbers representing individual brightness values at given co-ordinate spacings of the original continuous image.

APPENDIX C

Proof of Stability Criterion For One Dimensional Filter Design

Proof for the sufficiency of the one dimensional stability criterion:

As indicated in Chapter III, the stability constraint used in the design of one dimensional filter design is:

$$\mathrm{Re}\{Q(Z)\} > 0 \quad \text{for} \quad |Z| = 1 \qquad \text{(C.1)}$$

where Q(Z) is the denominator polynomial of the recursive digital filter transfer function given by:

$$H(Z) = \frac{P(Z)}{Q(Z)} = \frac{a_0 + a_1 Z + a_2 Z^2 + \ldots a_N Z^n}{b_0 + b_1 Z + b_2 Z^2 + \ldots b_M Z^M}$$

It is required to prove that if (C.1) is given to be true, then Q(Z) is non-zero (i.e., it has no zeroes) inside the unit circle in the Z plane.

Proof:

Consider the Figure C1. From Cauchy's integral formula, at any point Z_0 inside the unit circle in Z plane, $Q(Z_0)$ is given by:

$$Q(Z_0) = \int_{|Z|=1} \frac{Q(Z)}{(Z-Z_0)} \, dZ \qquad \text{(C.2)}$$

Reparameterizing the contour of integration, Figure C1, i.e., with $Z = e^{it}$ and also with $Z_0 = re^{i\phi}$, where $r < 1$, (C.2) can be rewritten as:

$$Q(re^{i\phi}) = \frac{1}{2\pi i} \int_0^{2\pi} \frac{Q(e^{it})}{e^{it} - re^{i\phi}} \, ie^{it} dt$$

which can be rewritten as:

$$Q(re^{i\phi}) = \frac{1}{2\pi} \int_0^{2\pi} \frac{Q(e^{it})}{1-re^{i(\phi-t)}} dt \qquad (C.3)$$

(C.3) can also be expressed as:

$$Q(re^{i\phi}) = \frac{1}{2\pi} \int_0^{2\pi} Q(e^{it}) \cdot \sum_{n=0}^{\infty} r^n e^{i(\phi-t)n} dt \qquad (C.4)$$

$Q(e^{it})$ is given by:

$$Q(e^{it}) = \sum_{m=0}^{M} b_m e^{imt} = b_0 + b_1 e^{it} + b_2 e^{i2t} + \ldots + b_M e^{iMt} \qquad (C.5)$$

in which case it is true that:

$$\frac{1}{2\pi} \int_0^{2\pi} Q(e^{it}) \cdot \sum_{n=-1}^{-\infty} r^{-n} e^{i(\phi-t)n} dt = 0 \qquad (C.6)$$

Adding (C.6) to the right hand side of (C.3) does not alter the value of $Q(re^{i\phi})$ and therefore $Q(re^{i\phi})$ can be expressed as:

$$Q(re^{i\phi}) = \frac{1}{2\pi} \int_0^{2\pi} Q(e^{it}) \sum_{n=0}^{\infty} r^n e^{i(\phi-t)n} + \sum_{n=-1}^{-\infty} r^{-n} e^{i(\phi-t)n} dt \qquad (C.7)$$

In (C.7), the quantity:

$$\sum_{n=0}^{\infty} r^n e^{i(\phi-t)n} + \sum_{n=-1}^{-\infty} r^{-n} e^{i(\phi-t)n} = P_r(\phi-t) \qquad (C.8)$$

where $P_r(\phi-t)$ is called the Poisson Kernel. One of the properties of the Poisson Kernel is that it is always positive for $r < 1$.

$$\text{i.e.,} \quad P_r(\phi - t) > 0 \quad \forall\, r < 1 \tag{C.9}$$

Therefore, in (C.7), one can see that, if

$$\mathrm{Re}\{Q(Z)\} = \mathrm{Re}\{Q(e^{it})\} > 0 \quad 0 \leqslant t \leqslant 2\pi \tag{C.10}$$

then,

$$Q(re^{i\phi}) > 0 \quad \forall \quad r < 1$$

Therefore, it is proven that if,

$$\mathrm{Re}\{Q(Z)\} > 0 \text{ for } |Z| = 1$$

then $Q(Z)$ has no zeros inside the unit circle in the Z plane.

Z Plane

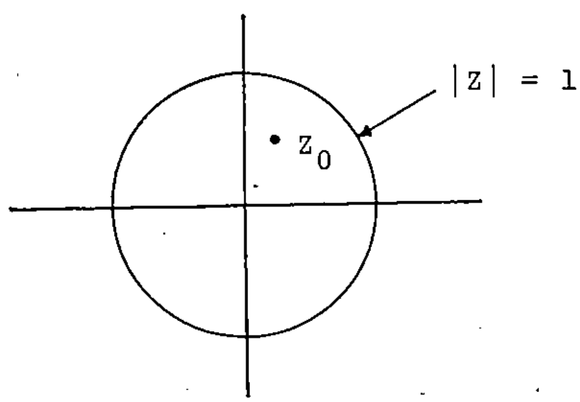


Figure C1

APPENDIX D

On The Stability Criterion For Second Order One Dimensional Filter

Comparison of two types of stability constraints for the second order case:

Consider a second order filter transfer function:

$$H(Z) = \frac{P(Z)}{Q(Z)} = \frac{a_0 + a_1 Z + a_2 Z^2}{1 + b_1 Z + b_2 Z^2}; \quad Z = e^{-j\Omega} \qquad (D.1)$$

where Ω is the normalized frequency variable.

Consider now, the type (a) constraint. Accordingly, the coefficients of Q(Z) would be constrained as follows:

$$1 > b_1 > b_2 > 0 \qquad (D.2)$$

Inequality (D.2) can be split into several inequalities as follows:

$$b_1 < 1 \qquad (D.3)$$

$$b_1 - b_2 > 0 \qquad (D.4)$$

$$b_2 > 0 \qquad (D.5)$$

The solution region for the inequalities (D.3) through (D.5) is shown in Figure D1(a). As one can see, the solution region is dependent upon b_1. The solution region gets smaller and smaller as b_1 decreases and therefore the type of stable filters one can design is very limited. Now consider the type (b) constraint. The coefficients of Q(Z), according to type (b) constraint, would be constrained as follows:

$$1 + b_1\cos\Omega + b_2\cos 2\Omega > 0 \; ; \; 0 \leqslant \Omega \leqslant \pi \qquad (D.6)$$

Substituting various values for Ω we can get several inequa-

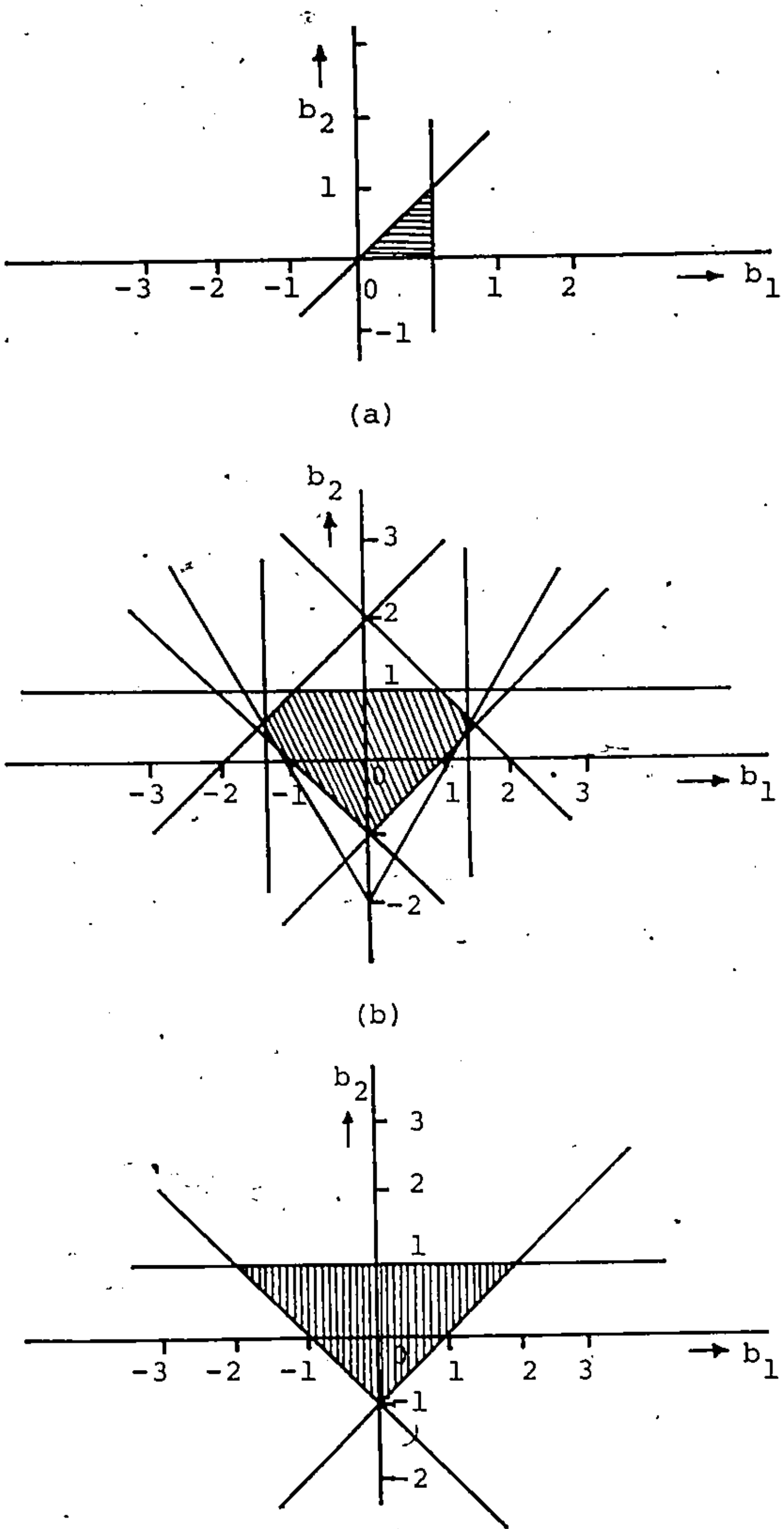


Figure D 1 Solution Region Corresponding to Various Types of Constraints.

qualities, which, to a considerable extent, is sufficient to determine the convex polyhedron solution region. Thus (D.6) for various values of Ω becomes:

$\Omega = 0$	$1 + b_1 + b_2 > 0$; i.e.,	$b_1 + b_2 > -1$	(D.7)
$\Omega = \frac{\pi}{6}$	$1 + \frac{3b_1}{2} + \frac{b_2}{2} > 0$	$3b_1 + b_2 > -2$	(D.8)
$\Omega = \frac{\pi}{3}$	$1 + \frac{b_1}{2} - \frac{b_2}{2} > 0$	$b_1 - b_2 > -2$	(D.9)
$\Omega = \frac{\pi}{4}$	$1 + \frac{b_1}{2} > 0$	$b_1 > -2$	(D.10)
$\Omega = \frac{\pi}{2}$	——— > 0	$b_2 < 1$	(D.11)
$\Omega = \frac{2\pi}{3}$	$1 - \frac{b_1}{2} - \frac{b_2}{2} > 0$	$b_1 + b_2 < 2$	(D.12)
$\Omega = \frac{3\pi}{4}$	$1 - \frac{b_1}{2} > 0$	$b_1 < 2$	(D.13)
$\Omega = \pi$	$1 - b_1 + b_2 > 0$	$-b_1 + b_2 > -1$	(D.14)

Constraints on b_1 and b_2 for other values of Ω do not seem to alter the result significantly and Figure D1(b) shows the solution region for inequalities of (D.7) through (D.14).

Finally, according to Jury [34], the general constraints on coefficients of Q(Z) for a stable filter is as follows:

$$b_1 + b_2 > -1 \tag{D.15}$$

$$-b_1 + b_2 > -1 \tag{D.16}$$

$$b_2 < 1 \tag{D.17}$$

The solution region corresponding to these constraints is shown in Figure D1(c).

The assertion from Figure D1 is therefore the constraint type (b) offers the design of larger subclass of stable filter than type (a) constraint.

APPENDIX E

Proof of the Sufficiency of the Two Dimensional Stability Constraint

The stability constraint used in the two dimensional design method is:

$$\mathrm{Re}\{B(Z_1,Z_2)\} > 0 \text{ for } |Z_1| = 1, \ |Z_2| = 1 \qquad (E.1)$$

where $B(Z_1,Z_2)$ is the denominator of a two dimensional recursive digital filter transfer function, given by:

$$H(Z_1,Z_2) = \frac{P(Z_1,Z_2)}{Q(Z_1,Z_2)} = \frac{\sum_{m=0}^{M1}\sum_{n=0}^{N1} a_{mn} Z_1^m Z_2^n}{\sum_{m=0}^{M2}\sum_{n=0}^{N2} b_{mn} Z_1^m Z_2^n}$$

It is required to show that if (E.1) is true, then $B(Z_1,Z_2)$ satisfies the Shank's stability theorum 14 ; i.e.,

$$B(Z_1,Z_2) \neq 0 \text{ for } |Z_1| \leq 1 \text{ and } |Z_2| \leq 1$$

Proof:

Consider a point, $Z_2 = Z_{2a}$, on the unit circle in the Z_2 plane of Figure E1. Now, for any $Z_2 = Z_{2a}$; $|Z_2| = 1$, $B(Z_1,Z_2)$ is a polynomial in Z_1 only, i.e., $f_1(Z_1) = B(Z_1,Z_{2a})$, where the coefficients of $f_1(Z_1)$ are, in general complex.

From (A.1), it is then true, that,

$$\mathrm{Re}\{f_1(Z_1)\} > 0 \ \forall \ |Z_1| = 1 \text{ and for any } Z_{2a}; \ |Z_{2a}| = 1 \qquad (E.2)$$

Therefore, using the proof for the one dimensional case from Appendix C, it can be clearly seen that $f_1(Z_1)$ is non-zero inside the unit circle in Z_1 plane, i.e.,

$$\mathrm{Re}\{f_1(Z_{10})\} > 0 \quad \text{for any } Z_{10};\ |Z_{10}| < 1 \qquad \text{(E.3)}$$

Therefore, it is true that:

$$B(Z_1,Z_2) \neq 0 \quad \text{for } |Z_1| \leq 1 \text{ and } |Z_2| = 1 \qquad \text{(E.4)}$$

Consider now, the function $f_2(Z_2)$ such that:

$$f_2(Z_2) = B(Z_{10},Z_2) \quad \forall \quad |Z_{10}| < 1 \text{ and } |Z_2| = 1 \qquad \text{(E.5)}$$

From (E.3), it is clear that:

$$\mathrm{Re}\{f_2(Z_2)\} > 0 \quad \forall \quad |Z_2| = 1 \qquad \text{(E.6)}$$

Therefore, once again, using the one dimensional proof, it can be said that for any Z_{20}; $|Z_{20}| < 1$.

$$f_2(Z_{20}) \neq 0 \qquad \text{(E.7)}$$

i.e., $B(Z_{10},Z_{20}) \neq 0$ for any Z_{10}; $|Z_{10}| < 1$; and

for any Z_{20}; $|Z_{20}| < 1$ (E.8)

Thus, it is proved that:

$$B(Z_1,Z_2) \neq 0 \quad \forall \quad |Z_1| \leq 1 \text{ and } |Z_2| \leq 1$$

$$\text{if} \quad \mathrm{Re}\{B(Z_1,Z_2)\} > 0 \quad \forall \quad |Z_1| = 1 \text{ and } |Z_2| = 1$$

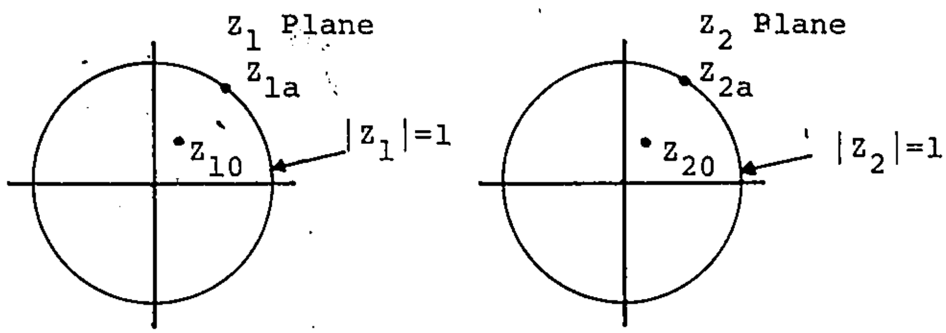


Figure E1

APPENDIX F

Two Dimensional Recursive Digital Quarter Plane Filter Design in Cascade Form

This appendix describes the frequency domain approximation of two dimensional recursive digital filters, using the technique of Ramamoorthy and Bruton [4]. The filters designed by this approach are realized in cascade form.

Form of the Recursive Filter:

The two dimensional recursive filter transfer function used in the approximation procedure is of the form:

$$H(Z_1,Z_2) = A \cdot \prod_{k=1}^{K} H_{1_k}(Z_1,Z_2) \cdot \prod_{\ell=1}^{L} H_{2_\ell}(Z_1,Z_2) \quad \text{(F.1)}$$

where $Z_1 = e^{-j\Omega_1}$, $Z_2 = e^{-j\Omega_2}$ and Ω_1, Ω_2 are the normalized frequency variables. $H_{1_k}(Z_1,Z_2)$ and $H_{2_\ell}(Z_1,Z_2)$ are the first and second order filter sections, and A is the gain factor. The first and second order filter sections are of the form:

$$H_{1_k}(Z_1,Z_2) = \frac{\sum_{m=0}^{1}\sum_{n=0}^{1} a_k(m,n) Z_1^m Z_2^n}{\sum_{m=0}^{1}\sum_{n=0}^{1} b_k(m,n) Z_1^m Z_2^n} \quad \text{(F.2)}$$

and

$$H_{2_\ell}(Z_1,Z_2) = \frac{\sum_{m=0}^{2}\sum_{n=0}^{2} c_\ell(m,n) Z_1^m Z_2^n}{\sum_{m=0}^{2}\sum_{n=0}^{2} d_\ell(m,n) Z_1^m Z_2^n} \quad \text{(F.3)}$$

Central to the technique of [10] is the problem of obtaining the filter transfer functions of (F.2) and (F.3) in the stable form. The procedure required to obtain stable $H_{1_k}(Z_1,Z_2)$ and $H_{2_\ell}(Z_1,Z_2)$ is described in the following section.

A. Stable First and Second Order Transfer Functions

As shown in [4], a stable first order filter transfer function $H_{1_k}(Z_1,Z_2)$ or a stable second order transfer function $H_{2_\ell}(Z_1,Z_2)$ can be obtained by transforming a two dimensional analog transfer function $H(S_1,S_2)$ of corresponding order, via the double bilinear transformation, where:

$$H(S_1,S_2) = \frac{P(S_1,S_2)}{Q(S_1,S_2)} \text{ and } Q(S_1,S_2) \neq 0 \; ; \tag{F.4}$$

$$\text{for } \mathrm{Re}\{S_1\} > 0, \; \mathrm{Re}\{S_2\} > 0; \; S_1 = jw_1, \; S_2 = jw_2$$

The double bilinear transformation is given by:

$$S_1 = \frac{1 - Z_1}{1 + Z_1} \; ; \; S_2 = \frac{1 - Z_2}{1 + Z_2} \tag{F.5}$$

First Order Case:

Consider a first order analog transfer function, given by:

$$H_{1_k}(S,S_2) = \frac{P_{1_k}(S_1,S_2)}{Q_{1_k}(S_1,S_2)} = \frac{p_{1_k} + p_{2_k}S_2 + p_{3_k}S_1 + S_1S_2}{q_{1_k} + q_{2_k}S_2 + q_{3_k}S_1 + S_1S_2} \tag{F.6}$$

In order that (F.6) satisfy the condition given in (F.4), it is required that the coefficients of $Q_{1_k}(S_1,S_2)$ be expressed in terms of a set of non-zero variables $(x_{1_k}, x_{2_k}, x_{3_k})$ such that:

$$q_{1_k} = x_{1_k}^2 ; \; q_{2_k} = x_{2_k}^2 ; \text{ and } q_{3_k} = x_{3_k}^2 \tag{F.7}$$

Using the double bilinear transform on (F.6) and also using (F.7), the coefficients of the stable $H_{1_k}(Z_1,Z_2)$, which is of the form given by (F.2), can be written as follows:

$$\begin{aligned}
a_k(0,0) &= (p_{1_k}+p_{2_k}+p_{3_k}+1) & b_k(0,0) &= (x_{1_k}^2+x_{2_k}^2+x_{3_k}^2+1)\\
a_k(0,1) &= (p_{1_k}-p_{2_k}+p_{3_k}-1) & b_k(0,1) &= (x_{1_k}^2-x_{2_k}^2+x_{3_k}^2-1)\\
a_k(1,0) &= (p_{1_k}+p_{2_k}-p_{3_k}-1) & b_k(1,0) &= (x_{1_k}^2+x_{2_k}^2-x_{3_k}^2-1)\\
a_k(1,1) &= (p_{1_k}-p_{2_k}-p_{3_k}+1) & b_k(1,1) &= (x_{1_k}^2-x_{3_k}^2-x_{3_k}^2+1)
\end{aligned} \tag{F.8}$$

Thus the first order filter $H_{1_k}(Z_1,Z_2)$ becomes a function of the parameter vector, given by:

$$X_{1_k} = (p_{1_k},p_{2_k},p_{3_k},x_{1_k},x_{2_k},x_{3_k}) \tag{F.9}$$

where x_{1_k}, x_{2_k} and x_{3_k} are non-zero variables.

Second Order Case:

Consider an analog two dimensional filter, given by:

$$H_{2_\ell}(S_1,S_2) = \frac{P_{2_\ell}(S_1,S_2)}{Q_{2_\ell}(S_1,S_2)}$$

$$= \frac{U_{1_\ell}+U_{2_\ell}S_2+U_{3_\ell}S_2^2+U_{4_\ell}S_1+U_{5_\ell}S_1S_2+U_{6_\ell}S_1S_2^2+U_{7_\ell}S_1^2+U_{8_\ell}S_1^2S_2+S_1^2S_2^2}{V_{1_\ell}+V_{2_\ell}S_2+V_{3_\ell}S_2^2+V_{4_\ell}S_1+V_{5_\ell}S_1S_2+V_{6_\ell}S_1S_2^2+V_{7_\ell}S_1^2+V_{8_\ell}S_1^2S_2+S_1^2S_2^2} \tag{F.10}$$

In order that (F.10) satisfy (F.4), it is required that the coefficients of $Q_2(S_1,S_2)$ be expressed in terms of a set of non-zero variables $(y_1, y_2, y_3, y_4, y_5, y_6, y_7, y_8, y_9, y_{10})$ such that:

$$v_{1_\ell} = (y_{5_\ell} y_{10_\ell} - y_{6_\ell} y_{9_\ell} + y_{7_\ell} y_{8_\ell})^2$$

$$v_{2_\ell} = (y_{1_\ell} y_{8_\ell} - y_{2_\ell} y_{6_\ell} + y_{3_\ell} y_{5_\ell})^2 + (y_{1_\ell} y_{9_\ell} - y_{2_\ell} y_{7_\ell} + y_{4_\ell} y_{5_\ell})^2$$

$$v_{3_\ell} = y_{5_\ell}^2$$

$$v_{4_\ell} = (y_{1_\ell} y_{10_\ell} - y_{3_\ell} y_{7_\ell} + y_{4_\ell} y_{6_\ell})^2 + (y_{2_\ell} y_{10_\ell} - y_{3_\ell} y_{9_\ell} + y_{4_\ell} y_{8_\ell})^2$$

$$v_{5_\ell} = (y_{6_\ell}^2 + y_{7_\ell}^2 + y_{8_\ell}^2 + y_{9_\ell}^2)$$

$$v_{6_\ell} = (y_{1_\ell}^2 + y_{2_\ell}^2)$$

$$v_{7_\ell} = y_{10_\ell}^2$$

$$v_{8_\ell} = y_{3_\ell}^2 + y_{4_\ell}^2$$

(F.11)

Using the double bilinear transform on (F.10), the coefficients of the stable $H_{2_\ell}(z_1, z_2)$ which is of the form given by (F.3), can be written as:

$$c_\ell(0,0) = (u_{1_\ell} + u_{2_\ell} + u_{3_\ell} + u_{4_\ell} + u_{5_\ell} + u_{6_\ell} + u_{7_\ell} + u_{8_\ell} + 1)$$

$$c_\ell(0,1) = 2(u_{1_\ell} - u_{3_\ell} + u_{4_\ell} - u_{6_\ell} + u_{7_\ell} - 1)$$

$$c_\ell(0,2) = (u_{1_\ell} - u_{2_\ell} + u_{3_\ell} + u_{4_\ell} - u_{5_\ell} + u_{6_\ell} + u_{7_\ell} - u_{8_\ell} + 1)$$

$$c_\ell(1,0) = 2(u_{1_\ell} + u_{2_\ell} - u_{3_\ell} - u_{7_\ell} - u_{8_\ell} - 1)$$

(F.12)
(cont'd)

$$c_\ell(1,1) = 4(u_{1_\ell} - u_{3_\ell} - u_{7_\ell} + 1)$$

$$c_\ell(1,2) = 2(u_{1_\ell} - u_{2_\ell} + u_{3_\ell} - u_{7_\ell} + u_{8_\ell} - 1)$$

$$c_\ell(2,0) = (u_{1_\ell} + u_{2_\ell} + u_{3_\ell} - u_{4_\ell} - u_{5_\ell} - u_{6_\ell} + u_{7_\ell} + u_{8_\ell} + 1)$$

$$c_\ell(2,1) = 2(u_{1_\ell} - u_{3_\ell} - u_{4_\ell} + u_{6_\ell} + u_{7_\ell} - 1)$$

$$c_\ell(2,2) = (u_{1_\ell} - u_{2_\ell} + u_{3_\ell} - u_{4_\ell} + u_{5_\ell} - u_{6_\ell} + u_{7_\ell} - u_{8_\ell} + 1) \qquad \text{(F.12)}$$

Similarly $d_\ell(m,n)$ is expressed in terms of $(v_{1_\ell}, v_{2_\ell}, \ldots, v_{8_\ell})$ where $(v_{1_\ell}, v_{2_\ell}, \ldots v_{8_\ell})$ are in turn expressed in terms of non-zero variables $(y_{1_\ell}, y_{2_\ell} \ldots, y_{10_\ell})$, as given in (F.11). Thus the second order filter $H_{2_\ell}(Z_1, Z_2)$ becomes a function of the parameter vector:

$$X_{2_\ell} = (u_{1_\ell}, u_{2_\ell}, u_{3_\ell}, \ldots, u_{8_\ell}, y_{1_\ell}, y_{2_\ell}, y_{3_\ell}, \ldots, y_{10_\ell}) \qquad \text{(F.13)}$$

where $y_1, y_2, \ldots, y_{10}$ are non-zero variables.

B. Approximation Procedure

The procedure involves the minimization of an L_{2_p} norm of the form:

$$L_{2_p}(X) = \sum_{i=1}^{I} \sum_{j=1}^{J} \{f(X, \Omega_{1i}, \Omega_{2j}) - f_d(\Omega_{1i}, \Omega_{2j})\}^{2p}$$

where X is a parameter vector with respect to which the minimization is performed. The minimization is carried out over a set of discrete frequency points Ω_{1i} and Ω_{2j} in the right half of the frequency domain such that:

$H_B(Z_1,Z_2)$ is chosen to be equal to that of $H_A(Z_1,Z_2)$.

Case 3: Atmospheric turbulance blur (2nd example):

$$f(X,\Omega_1,\Omega_2) = |H(X,Z_1,Z_2)|^2 \Big|_{\substack{Z_1=e^{-j\Omega_1} \\ Z_2=e^{-j\Omega_2}}} \qquad (F.19)$$

where X consists of parameter vectors X_{1_k} and X_{2_ℓ} of each first and second filter sections respectively of $H(Z_1,Z_2)$.

The minimization of the L_{2_p} norm of (F.14) is performed by the widely used optimization procedure of Fletcher and Powell[+].

[+] R. Fletcher and M.J.D. Powell "A rapidly convergent decent method for minimization", Comput. J., Vol. 6, pp. 163-168, 1963.

$$-\pi < \Omega_{1i} < \pi \quad \text{and} \quad 0 < \Omega_{2j} < \pi \tag{F.15}$$

$f_d(\Omega_{1i},\Omega_{2j})$ is the desired specification. $f(x,\Omega_{1i},\Omega_{2j})$ is the approximating function whose form depends on the type of recursive filter implemented. The form of $f(x,\Omega_{1i},\Omega_{2j})$ for various types of implementations considered in this thesis, is as follows:

Case 1: Recursive filter implementation for Motion deblur:

$$f(x,\Omega_1,\Omega_2) = 2\cdot\text{Real}\{H(X,Z_1,Z_2)\}\Big|_{\substack{Z_1=e^{-j\Omega_1}\\ Z_2=e^{-j\Omega_2}}} \tag{F.16}$$

where $H(Z_1,Z_2)$ is of the form given by (F.1). The parameter vector X consists of parameter vectors X_{1_k} and X_{2_ℓ} of each first and second order filter sections respectively. X_{1_k} and X_{2_ℓ} are as indicated in (F.9) and (F.13) respectively.

Case 2: Recursive filter implementation for Focus deblur:

$$f(x,\Omega_1,\Omega_2) = |H_A(X_A,Z_1,Z_2)|^2 - |H_B(X_B,Z_1,Z_2)|^2\Big|_{\substack{Z_1=e^{-j\Omega_1}\\ Z_2=e^{-j\Omega_2}}} \tag{F.17}$$

where parameter vector X is given by:

$$X = \{X_A,X_B\} \tag{F.18}$$

In (F.k8), X_A of $H_A(Z_1,Z_2)$ and X_B of $H_B(Z_1,Z_2)$ each consist of parameter vectors X_{1_k} and X_{2_k} of each first and second order filter sections respectively. In (F.17), the order of

APPENDIX G

Image Processing Hardware

This appendix describes the image processing facility that was developed at the Signals and Systems Laboratory of the Department of Electrical Engineering, at the University of Windsor, which makes use of a Flying Spot Scanner System 14 for image sampling and a Colorado Video Instrument display unit for image display. The block diagram of Figure G1 shows the complete hardware set-up. The facility is centred around a Data General Corporation NOVA-840 minicomputer with 128K, 16 bit words of core memory and moving head disk storage, with a total storage of 2.4 million 16-bit words. There is also an A/D and digital to analog (D/A) converters, with a general purpose high speed data channel interface.

Flying Spot Scanner System

The flying spot scanner system, used for sampling images, is shown in Figure G2. It consists of a cathode ray tube (CRT), objective lens, transmitting film, a condensing lens and photomultiplier tube. The x and y outputs from the digital to analog converter generates a raster scanned electron beam across the CRT. The resulting light pattern is focused by the objective lens, onto the transmitting film. The light that is transmitted by the film is received by a photomultiplier tube which is placed behind the film and the condenser lens system. This generates an equivalent electrical signal corresponding to the amount of light received.

In order to reduce the effects of internal and 60HZ beam supply noise, the output of the phototube is integrated over a time period before being sampled by the analog to digital converter.

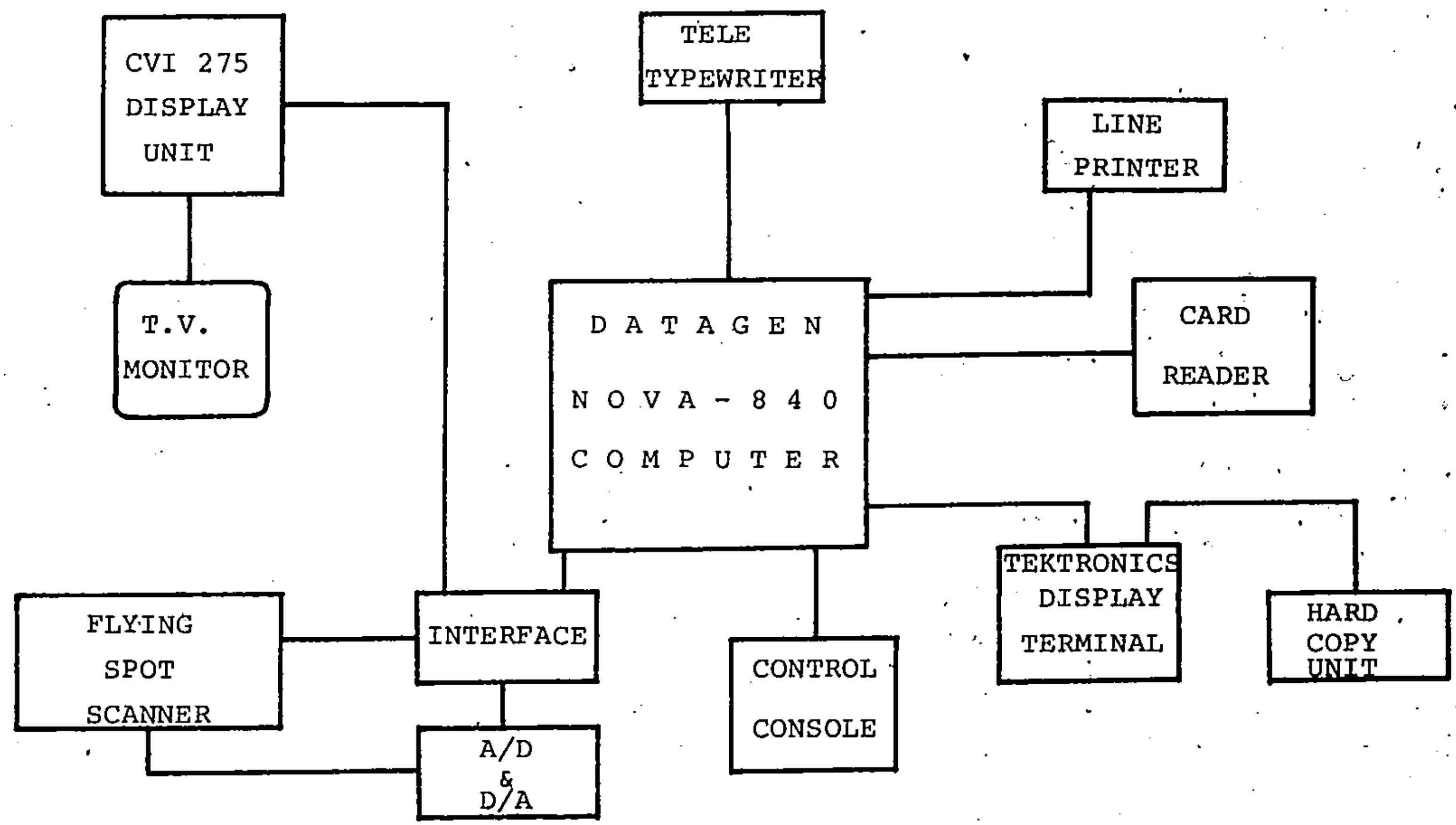


Figure G1 Block Diagram of the Image Processing Facility.

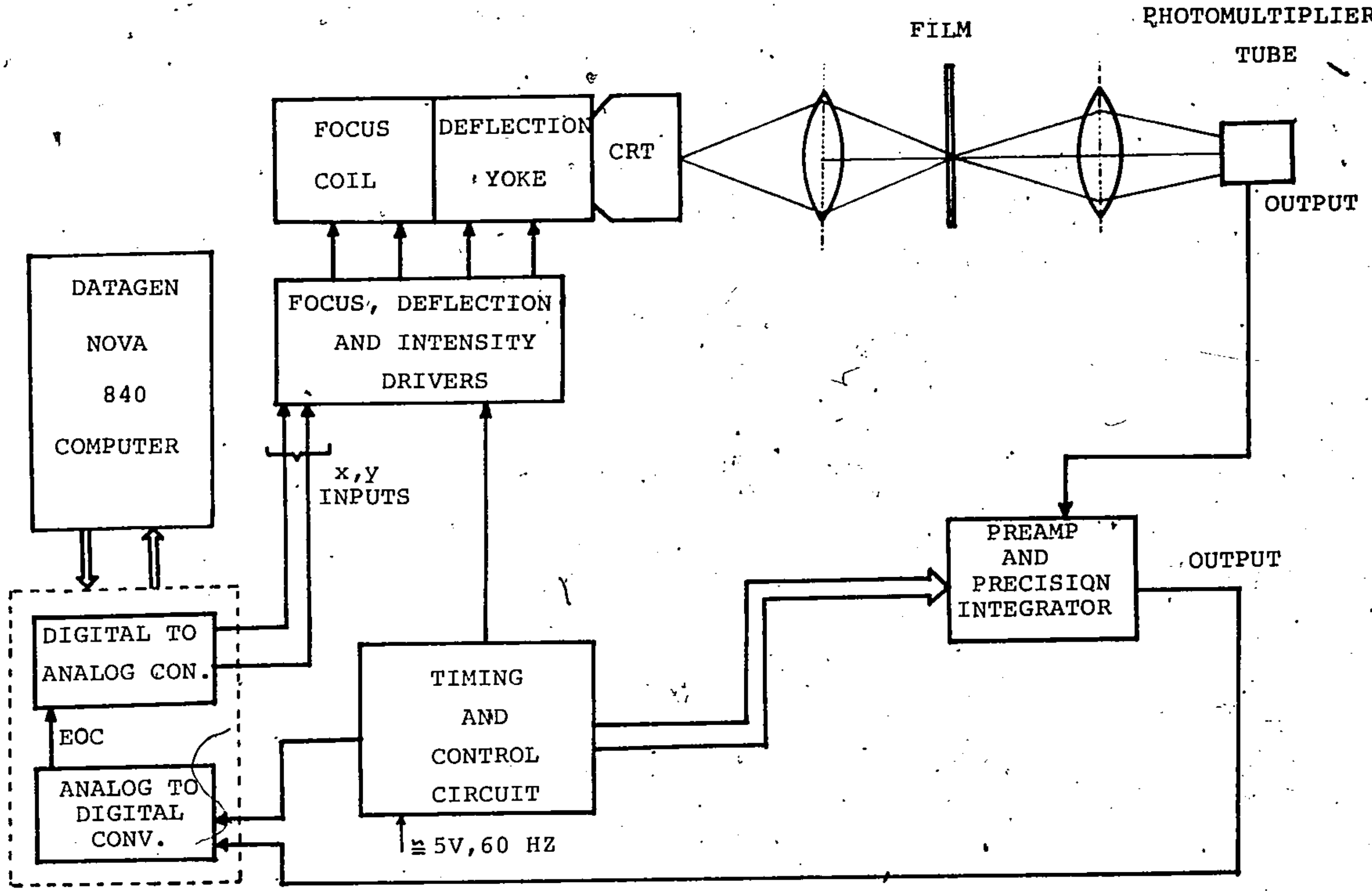


Figure G2 Flying Spot Scanner System.

The output of the analog digital converter is a digital number corresponding to the light that was transmitted through the film. It should be noted here that the sampled image does not excctly correspond to the original image that was recorded on the photographic film. The image being sampled is convolved with the CRT spot that scans across the film. The mathematical model that describes the light transmitted through the film is given by 38 :

$$g_1(x,y) = \int_{-\infty}^{\infty} \int_{-\infty}^{\infty} h_a(x-x_1,y-y_1)g(x_1,y_1)dx_1dy_1 \quad (G.1)$$

where g is the image recorded on the photographic film. $g_1(j\Delta x,k\Delta y)$ are the matrix samples for $j=0,1,\ldots,(N-1)$ and $k=0,1,2,\ldots,(M-1)$. The function h_a, describes the intensity profile of the CRT spot that is projected onto the film and this is the effective aperature through which the film is observed. The frequency domain equivalent of Equation (G.1) is given by:

$$G_1(w_1,w_2) = H_a(w_1,w_2)\cdot G(w_1,w_2) \quad (G.2)$$

where G_1 is the fourier transform of g_1, etc. If the aperature projected on the film is infinitesimally small, i.e., a two dimensional impulse, then $G_1=G$. In practice, such an aperature is impossible. From the viewpoint of aliasing+, such a response may be undesirable. Considering Equation (G.1), it can be seen that a suitable aperature can perform analog prefiltering and this may, in turn, reduce the problem

+ Aliasing is the phenomenon in which the overlap of the fourier spectrum takes place around the folding frequency due to incorrect choice of sampling interval.

of aliasing since the prefiltering is low pass in nature. This will also help to reduce the high frequency noise present in the image.

Image Digitization Scheme

As mentioned in the previous section, the output of the photomultiplier tube is integrated over a certain period of time. This period of time was chosen to be half the period of 60 HZ power supply (8 m.sec) since 60 HZ interference is present at the output of the photomultiplier tube.

With the flying spot scanner on, the timing and control circuit generates control pulses to control the integrater and A/D sampling. A block diagram of the timing and control circuit connections to the integrator is shown in Figure G3. The details of the timing and control circuit, along with the integrator circuit and the timing diagram can be found in the next section. The integrator is a precision integrator, with control inputs to allow reset, integrate and hold.

The entire image is sampled by scanning one line at a time, moving vertically downward after each line. This is carried out using a program 'PICSAMP' given in Appendix H. This program sets up two buffer storage areas in the memory. One buffer contains the x and y co-ordinate values for one line of the image to be scanned and the other is reserved for storing the sampled image values for the line that is being scanned.

The timing and control circuit sends out control signals soon after the power is turned on. The execution of the pro-

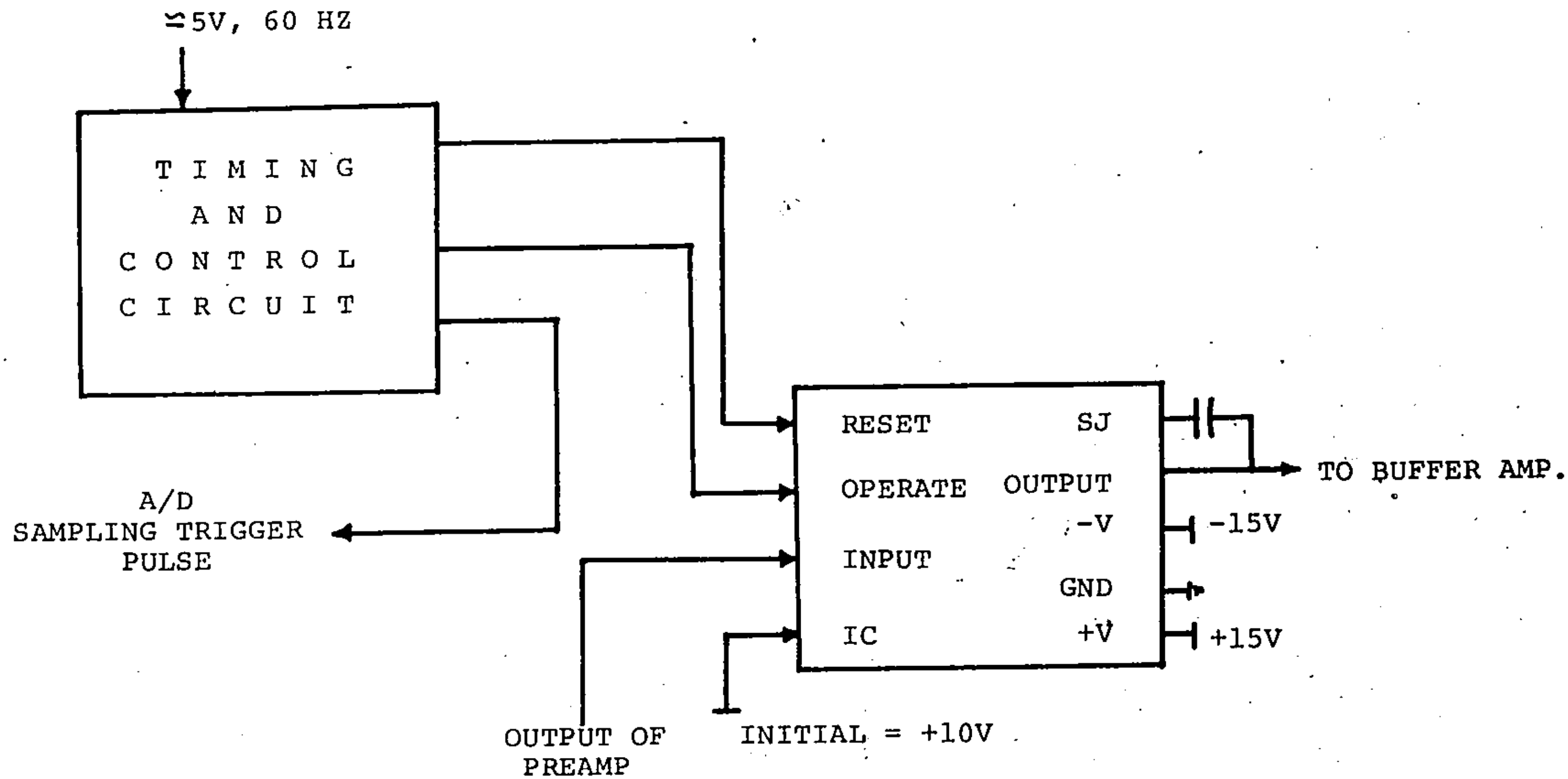


Figure G3 Timing and Control Circuit and Integrator SetUp.

gram then enables the D/A and A/D to start scanning and sampling the image. The A/D waits for the operate pulse to go high (+5v), at which point the integration of the PMT output ends. After a small time delay, the A/D receives a sampling pulse via a specified trigger bus. The A/D converts the output of the integrator (which is held constant during the conversion) and stores the data in the buffer memory. Following the conversion, the A/D converter generates an end of conversion (EOC) pulse, which is sent to the D/A trigger bus. This generates a simultaneous D/A conversion of two data corresponding to the next x and y co-ordinates of the image. After a short time delay following the A/D sampling pulse, the timing and control circuit sends out a reset pulse to reset the output of the integrator to the appropriate initial condition.

Using the above procedure, images can be sampled with array sizes up to 512 x 512. The image scan can have its origin at any point on the image with different sampling grid sizes.

Details of Timing and Control Circuit and the Integrator Set Up

The main circuit is shown in Figure G4. The zero crossings of the 60 HZ supply are detected by two operational amplifiers (Motorola 741), operated in an open loop configuration. The outputs of OP1 and OP2 are square waves, with a period equal to that of the 60 HZ waveform, and are 180° out-of-phase with each other. These outputs are inputs to two monostables (74121) which are connected in such a way that they respond to only the positive transitions in the input. The outputs of

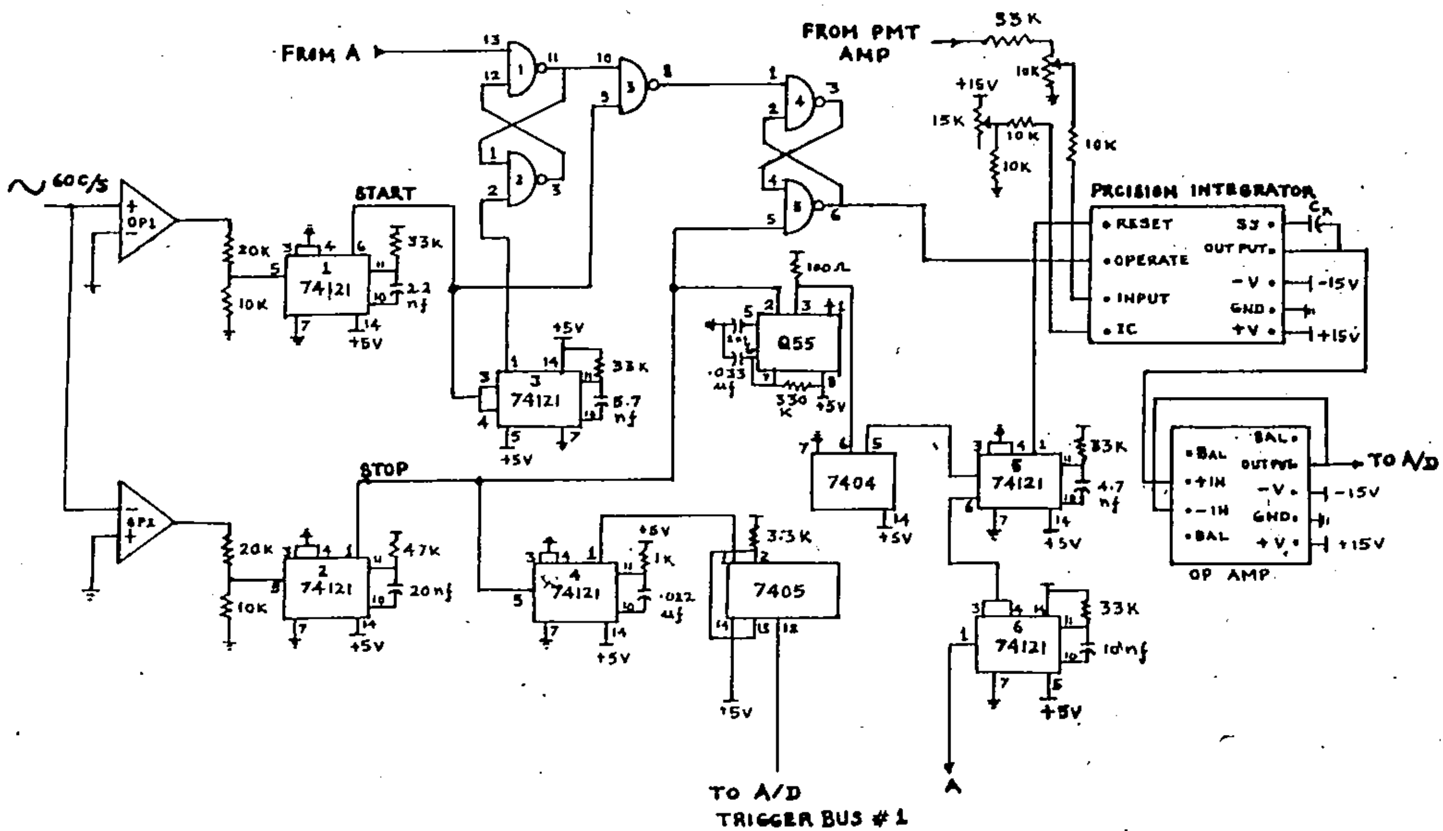


Figure G4 Timing and Control Circuit

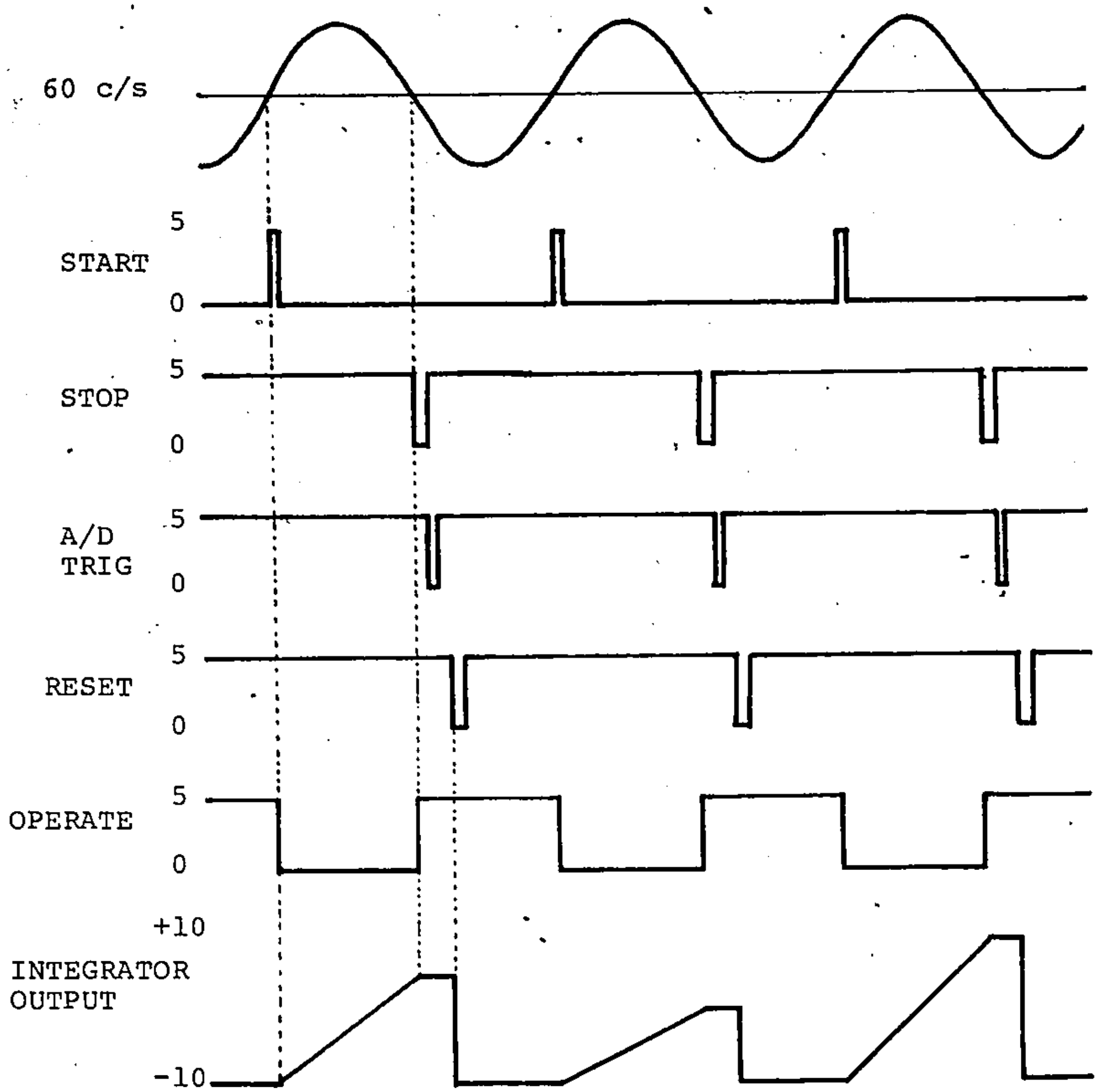


Figure G5 Timing Diagram.

these monostables are the 'start' and 'stop' pulses. Through some extra circuitry, these pulses are then used to set or reset the 'operate flip-flop', which is made up of two nand gates; namely, 4 and 5 shown in Figure . From the stop pulse, the A/D sampling trigger pulse is midway between the 'stop' and 'reset' pulse, thus ensuring correct sampling of the integrator output. The 'reset' pulse is generated by another network of delays as shown in Figure G4, and this pulse resets the integrator output to the desired initial condition (IC). The timing diagram corresponding to the timing and control circuit is shown in Figure G5.

The integrator shown in Figure G4 is a model 9018 precision integrator from +Optical Electronics Inc. As indicated earlier (Chapter II), it has digital control inputs, enable or disable integration and reset the output to the given initial condition. As can be seen from the timing diagram, the integration takes place only when the OPERATE input is low (0 volts) and the RESET input is high (+5 volts). The output is reset when the OPERATE is high and RESET is low and is held constant when both are high. The output voltage swing of the integrator is 10V. This range of output is maintained by proper choice of capacitor Cx, the initial condition and proper input range.

The output of the photomultiplier tube is amplified prior to feeding it to the integrator. The amplifier circuit is shown in Figure G6. The output from the integrator is fed

---

+ Optical Electronics Inc., P.O. Box 1140, Tucson, Arizona, U.S.A., 85734.

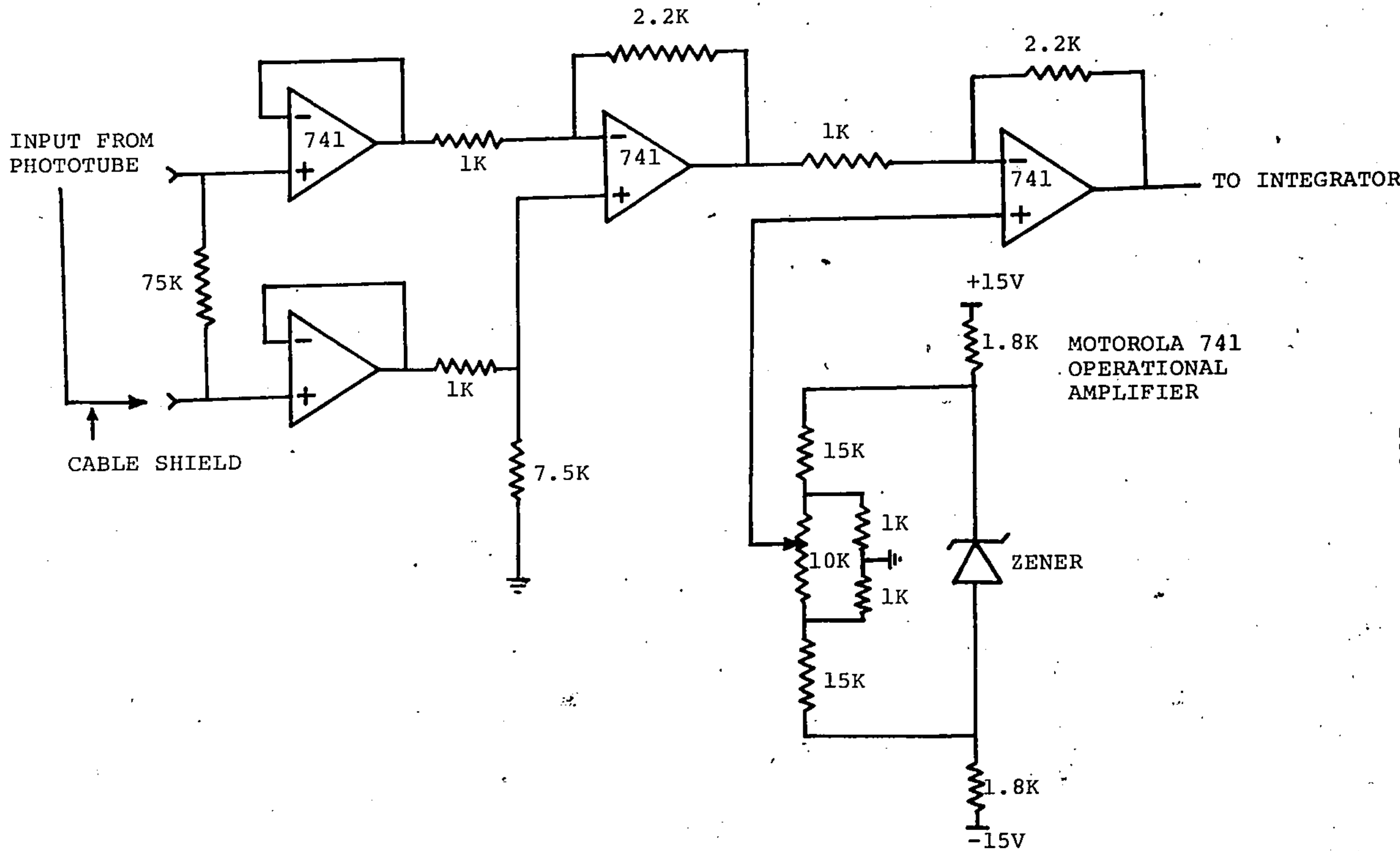


Figure G6 Photomultiplier Tube Output Amplifier.

into the A/D converter through a buffer which is a model +P201, FET operational amplifier.

Image Display and Reproduction

The display of sampled and processed images is carried out using the Colorado Video-270 A Instrument (CVI270A) Expander* which has Z modulation capability. The block diagram is shown in Figure G7.

In this display system, there exists a common interface with the NOVA computer, for both the D/A converter and the CVI270A Expander. In other words, the CVI270A and the D/A converter are considered as one peripheral device. The output latch for both the D/A converter and the CVI270A are the same. In order to transfer data to the CVI270A, the D/A converter has to be activated, and while the D/A conversion takes place, data is also transferred to the CVI270A.

The data transfer is accomplished via a handshaking operation between the D/A converter interface and the CVI270A. Considering Figure G7 - to display one pixel of the image requires the transfer of three data; namely, the x and y coordinates and the intensity value. The data is transferred one at a time. Thus for each pixel of the image, the D/A converter interface circuitry first presents the x data at the output latch, and sends a strobe pulse 'ACCEPT DATA', to the CVI270A, indicating that data is available, and the CVI-270A in return sends back a pulse to D/A converter interface circuitry, indicating that it is ready for the next data.

+ Polytron Devices Inc., P.O. Box 398, Paterson, N.J. 07524.

* Instruction Manual Model 270A Video Digitizer, Colorado Video Incorporated, Boulder, Colorado, Jan. 1977.

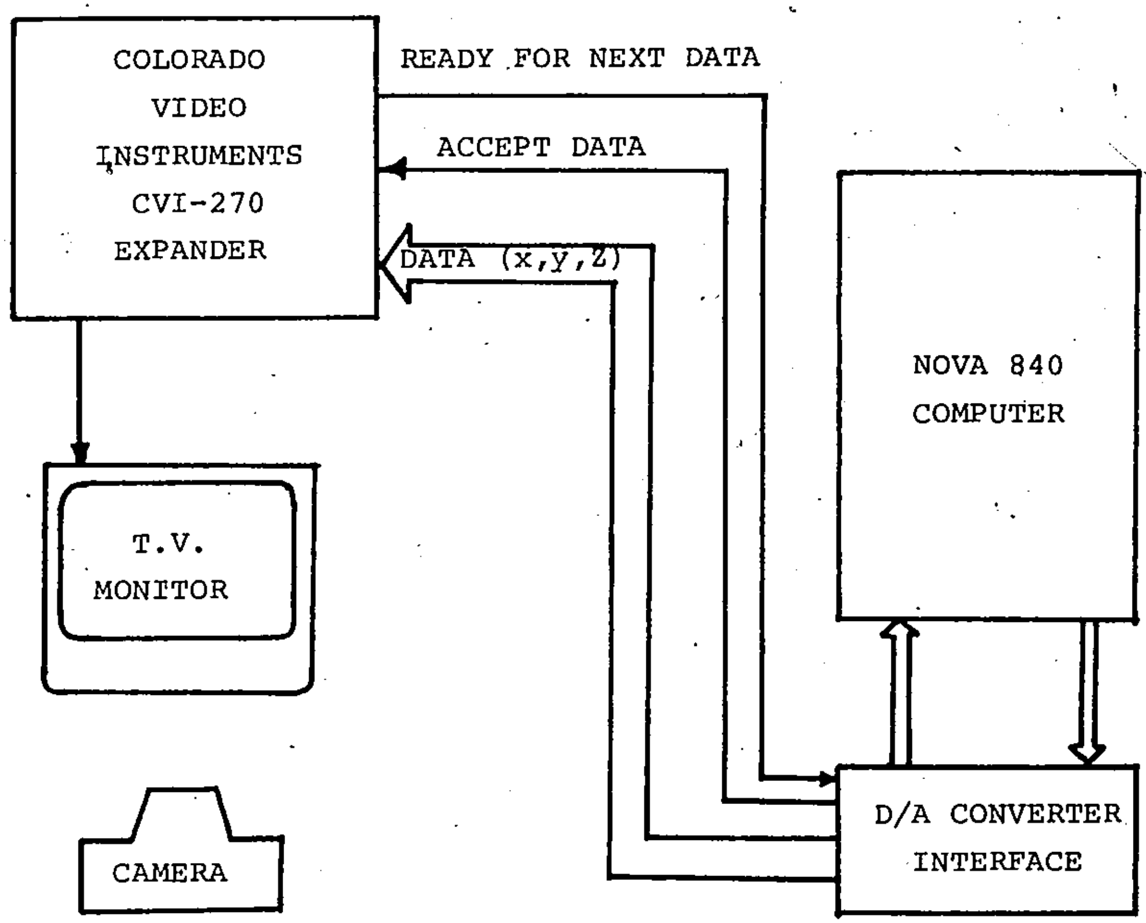


Figure G7 Image Display and Reproduction Hardware.

In a similar manner, y and Z (intensity) values are transferred.

Because of limited memory, the CVI270A can display images only up to size 256 x 256 pixels+. It can display intensities up to 256 gray levels, i.e., intensity data is 8 bits in length. The recording of the reproduced image on the T.V. monitor is carried out using a conventional camera, such as the Cannon FTb. For good results, one has to use a slow speed film, with exposure time set at ¼th of a second.

+ Pixel indicates a picture element.

APPENDIX H

Filter Design, Image Filtering and Other Additional Image Processing Computer Programs

```
      DIMENSION W(101),G(50),DM(101),DP(101),XCOF(50),
     *COF(50),ROOTR(50),ROOTI(50),U(52,52),DMA(101)
C     ***************************************************
C
C     MAIN PROGRAM - DESIGNING ONE-DIMENSIONAL RECURSIVE
C     DIGITAL FILTER TO DESIRED MAGNITUDE AND LINEAR
C     PHASE CHARACTERISTICS
C     THE INPUT PARAMETERS ARE :-
C     M1 : # OF FREQUENCY POINTS IN THE FREQUENCY AXIS
C     IFT : TYPE OF DESIRED FILTER
C           LPF - LOW PASS ; BPF - BAND PASS
C           HPF - HIGH PASS ; DIF - DIFFERENTIATOR
C     WC1, WC2 : LOWER AND UPPER PASS BAND FREQUENCIES
C                RESPECTIVELY . FOR THE CASE OF
C                DIFFERENTIATOR , WC1= 0.0 AND WC2= 1.0
C     ITRIAL : # OF TRIALS , WHERE FOR EACH TRIAL A FILTER
C              OF DIFFERENT ORDER IS SPECIFIED
C     ISETS  : FOR EACH TRIAL VALUE OF ITRIAL , ISETS # OF
C              GROUP DELAY VALUES ARE TRIED
C     NORD : ORDER OF THE FILTER
C     THE DIMENSIONS OF THE ARRAYS ARE AS FOLLOWS :-
C     W(M1),G(NORD),DM(M1),DP(M1),XCOEF(NORD+1),COF(NORD+1)
C     ROOTR(NORD),ROOTI(NORD),U((2*NORD+4),(2*NORD+4)),
C     DMA(M1)
C
C     SUBROUTINES REQUIRED :- SPECM, SIMPTB, RSIMP, PLOTMP,
C     POLRT( FOR OBTAINING POLES AND ZEROES OF THE FILTER)
C
C     ***********************************************************
C
      COMMON/B12/DM,DP,W,GDLY,WC1,WC2
      COMMON/B30/NNC,NDC,MFRQ
      DATA PY,DIF/3.141592,'DIF '/
      KTIME=0
      READ(5,60) M1
   60 FORMAT(I3)
      DW=1./(M1-1)
      W(1)=0.
      DP(1)=0.
      DO 123 I=2,M1
  123 W(I)=W(I-1)+DW
      READ(5,655) IFT
  655 FORMAT(A4)
      READ(5,201) WC1,WC2
  201 FORMAT(2E12.5)
      WRITE(6,511)
  511 FORMAT(' ',2X,'DESIRED SPECIFICATIONS - (FREQUENCIES
     * ARE IN FRACTIONS OF NYQUIST RATE)',/)
      CALL SPECM(M1,IFT)
      MFRQ=M1
      READ(5,515) ISETS,ITRIAL
  515 FORMAT(2I2)
```

```
 40 FORMAT(I2)
    WRITE(6,50) M1,DW
 50 FORMAT(' ',2X,'NO. OF POINTS ON FREQ AXIS= ',I3,5X,'FRE
   *Q INTERVAL= ',E12.5,/)
    WRITE(6,512)
512 FORMAT(' ',4X,'FREQUENCY',3X,'MAGNITUDE',/)
    DO 444 I=1,M1
    DMA(I)=DM(I)
444 WRITE(6,20) W(I),DM(I)
 20 FORMAT(' ',2(2X,F10.6))
    CALL PLOT3(W,DMA,M1)
    DIFPH=0.5*PY
    DO 2000 ITRL=1,ITRIAL
    READ(5,40) NORD
    NNC=NORD+1
    NDC=NORD
    GDLY=NORD
    WRITE(6,52) NORD
 52 FORMAT(/,' ',2X,'ORDER OF THE FILTER:------ ',I2,//)
    DO 1000 KTIMES=1,ISETS
C
C   GENERATION OF LINEAR PHASE CHARACTERISTICS
C
    DO 2 I=2,M1
    DP(I)=-GDLY*W(I)*PY
    IF(IFT.NE.DIF) GO TO 505
    DP(I)=DP(I)+DIFPH-0.5*W(I)*PY
505 CONTINUE
    IF(DP(I).LT.-PY) DP(I)=DP(I)+2.0*PY
    IF(DP(I).LT.-PY) GO TO 505
  2 CONTINUE
    WRITE(6,513) KTIMES,GDLY
513 FORMAT(' ',2X,'SET NUMBER=',I2,3X,'GROUP-DELAY=',
   *E12.5,/)
    CALL SIMPTB(M1,NCR,NVR)
    CALL RSIMP(U,NCR,NVR,KSIM)
    IF(KSIM.EQ.1) GO TO 10002
    NRS=NCR+2
    DO70 I=1,NDC
 70 G(I)=-U(NCR+1,I)
    DO 71 I=1,NNC
 71 G(NDC+I)=-U(NCR+1,NDC+I)
    IPQ=1
    IPK=NDC+1
    COE=1.
    WRITE(6,900) IPQ,G(IPK),IPQ,COE
    DO 901 IK=1,NDC
    IK2=IK+1
    IK1=NDC+IK+1
    IF(IK1.GT.NPR) G(IK1)=0.
    WRITE(6,900) IK2,G(IK1),IK2,G(IK)
900 FORMAT(' ',2X,'COEFN(',I2,')=',G14.7,5X,'COEFD(',I2,')=
   *',G14.7,/)
```

```
  901 WRITE(6,800)
      DO 902 I=1,NNC
  902 XCOF(I)=G(NDC+I)
      MNC=NORD
      NNCP=NNC
      IF(MNC.EQ.0) GO TO 910
      CALL POLRT(XCOF,COF,MNC,ROOTR,ROOTI,IER,NNCP)
      WRITE(6,800)
      WRITE(6,903) IER
  903 FORMAT(' ',2X,'IER= ',I1,/)
      GO TO 911
  910 ROOTR(1)=XCOF(1)
      ROOTI(1)=0.
      MNC=1
  911 WRITE(6,800)
      WRITE(6,904)
  904 FORMAT(' ',2X,'THE ZEROES OF THE FILTER ARE-',//)
      WRITE(6,905) (I,ROOTR(I),ROOTI(I),I=1,MNC)
  905 FORMAT(' ',2X,'ZERO(',I2,')=',E14.7,' +J ',E14.7,/)
      XCOF(1)=1.
      DO 906 I=1,NDC
  906 XCOF(I+1)=G(I)
      NDCP=NDC
      CALL POLRT(XCOF,COF,NDCP,ROOTR,ROOTI,IER,NNCP)
      WRITE(6,800)
      WRITE(6,903) IER
      WRITE(6,800)
      WRITE(6,907)
  907 FORMAT(' ',2X,'THE POLES OF THE FILTER ARE-',//)
      WRITE(6,908) (I,ROOTR(I),ROOTI(I),I=1,NDC)
  908 FORMAT(' ',2X,'POLE(',I2,')=',E14.7,' +J ',E14.7,/)
      WRITE(6,800)
      KST=0
      DO 600 I=1,NDC
      STB=ROOTR(I)**2+ROOTI(I)**2
  600 IF(STB.LE.1.) KST=1
      IF(KST.EQ.0) WRITE(6,601)
      IF(KST.EQ.1) WRITE(6,602)
      WRITE(6,800)
      DO 912 I=1,NNC
  912 COF(I)=G(NDC+I)
      KTIME=KTIME+1
      CALL PLOTMP(M1,COF,XCOF,NNCP,NDCP,KTIME)
10002 CONTINUE
      GDLY=GDLY-1
 1000 CONTINUE
 2000 CONTINUE
  800 FORMAT(' ',//)
  601 FORMAT(' ',2X,'THE FILTER IS STABLE',/)
  602 FORMAT(' ',2X,'THE FILTER IS UNSTABLE',/)
      STOP
      END
```

```
      SUBROUTINE SPECM(M1,IFT)
C
C     SUBROUTINE TO GENERATE THE MAGNITUDE SPECS
C     CORRESPONDING TO THE DESIRED TYPE OF FILTER
C     THE CHARACTERISTICS BEYOND THE PASS BAND IS
C     GENERATED ACCORDING TO THE FUNCTION
C     FN=EXP(-FK*((WP-WX)**2)), WHERE FK CAN BE
C     VARIED. FK IS OBTAINED IN TERMS OF FK2.
C     SHARPER TRANSITION BAND IN THE SPECIFICATIONS
C     CAN BE GENERATED BY SPECIFYING SMALLER VALUES
C     FOR FK2. USUALLY LESS THAN 1.0
C
      DIMENSION DM(101),DP(101),W(101)
      INTEGER BP,HP,DIF
      COMPLEX*16 TP1,TP2,TP3,TP4,TP5
      COMMON/B12/DM,DP,W,GDLY,WP1,WP2
      DATA BP/'BPF '/,HP/'HPF '/,DIF/'DIF '/
      DATA PY/3.141592/
      DATA TP1/'LOW  PASS FILTER'/
      DATA TP2/'BAND PASS FILTER'/
      DATA TP3/'HIGH PASS FILTER'/
      DATA TP4/'DIFFERENTIATOR  '/
      FN(FK,WP,WX)=EXP(-FK*((WP-WX)**2))
      IF(IFT.EQ.DIF) GO TO 103
      FK2=0.7
      FK=100.0/(FK2*FK2)
      IF(IFT.EQ.BP) GO TO 100
      IF(IFT.EQ.HP) GO TO 101
      TP5=TP1
      DO 1 I=1,M1
      IF(W(I).GT.WP2) GO TO 2
      DM(I)=1.0
      GO TO 1
    2 DM(I)=FN(FK,WP2,W(I))
    1 CONTINUE
      GO TO 102
  100 TP5=TP2
      DO 3 I=1,M1
      IF(W(I).LT.WP1) GO TO 4
      IF(W(I).GT.WP2) GO TO 5
      DM(I)=1.0
      GO TO 3
    4 DM(I)=FN(FK,WP1,W(I))
      GO TO 3
    5 DM(I)=FN(FK,WP2,W(I))
    3 CONTINUE
      GO TO 102
  101 TP5=TP3
      DO 6 I=1,M1
      IF(W(I).LT.WP1) GO TO 7
      DM(I)=1.0
      GO TO 6
    7 DM(I)=FN(FK,WP1,W(I))
```

```
    6 CONTINUE
      GO TO 102
  103 TP5=TP4
      DO 8 I=1,M1
    8 DM(I)=W(I)
  102 WRITE(6,202) TP5
      WRITE(6,203) WP1,WP2
  202 FORMAT(/,' ',2X,'FILTER TYPE :---   ',A16,/)
  203 FORMAT(' ',2X,'PASS BAND FREQUENCIES +---',2X,'WP1= ',
     *E12.5,2X,'WP2= ',E12.5,/)
      RETURN
      END
```

```
      SUBROUTINE SIMPTB(M1,NTV,NVR)
C
C     SUBROUTINE TO GENERATE THE SIMPLEX TABLEAU.
C     THE DIMENSIONS OF THE ARRAYS ARE AS FOLLOWS
C     A(2*NORD+1,2*M1),B(2*NORD+4),R1(2,M1)
C     R2(2,2*M1),CT(NORD,M1)
C     NORD AND M1 ARE AS INDICATED IN THE
C     MAIN PROGRAM.
C
      DIMENSION A(49,202),B(52),R1(2,101),R2(2,202),
     *CT(24,101),W(101),DP(101),DM(101)
      COMMON/B30/NNC,NDC,MFREQ
      COMMON/B11/M2,DW2
      COMMON/B12/DM,DP,W,GDLY,WC1,WC2
      COMMON/B35/R1,R2,CT,A,B
      PY=3.1415926
      NTA=NDC+NNC
      NTV=NTA+1
      NTV1=NTA+3
      K1=M1
      KK1=K1+K1
      KK2=KK1+K1
      KK3=KK2+K1
      KK4=KK3+M2
      ICT=1
```

```
      NVR=KK4
      IF(ICT.EQ.0) NVR=KK3
      EPS=1.E-6
      DO 1 I=1,M1
      YR=DM(I)*COS(DP(I))
      YI=DM(I)*SIN(DP(I))
      SUM1=0.
      SUM2=0.
      SUM3=0.
      SUM4=0.
      XA=PY*W(I)
      DO 2 J=1,NDC
      XX=COS(J*XA)
      YY=SIN(J*XA)
      A(J,I)=YR*XX+YI*YY
      A(J,K1+I)=YI*XX-YR*YY
      SUM1=SUM1+A(J,I)
      SUM2=SUM2+A(J,K1+I)
    2 CONTINUE
      DO 3 K=1,NNC
      XK=COS((K-1)*XA)
      YK=SIN((K-1)*XA)
      A(NDC+K,I)=-XK
      A(NDC+K,K1+I)=YK
      SUM1=SUM1+A(NDC+K,I)
      SUM2=SUM2+A(NDC+K,K1+I)
    3 CONTINUE
      R2(1,I)=-YR
      R2(1,K1+I)=-YI
      R2(2,I)=-SUM1-1.
      R2(2,K1+I)=-SUM2-1.
    1 CONTINUE
      DO 5 I=1,NTA
    5 B(I)=0.
      B(NTV)=1.
      B(NTV+1)=0.
      B(NTV1)=-1.
      DO 16 I=1,M1
      SUM1=0.
      XA=PY*W(I)
      DO 17 J=1,NDC
      XX=COS(J*XA)
      CT(J,I)=-XX
   17 SUM1=SUM1+CT(J,I)
      R1(1,I)=1.-EPS
   16 R1(2,I)=-SUM1
      RETURN
      END
```

```
      SUBROUTINE RSIMP(U,NCR,NVR,KSIM)
C
C     SUBROUTINE TO SOLVE THE LINEAR PROGRAMMING
C     PROBLEM USING THE REVISED SIMPLEX METHOD
C     THIS PROGRAM USES THE SIMPLEX TABLEAU
C     GENERATED IN THE ROUTINE SIMPTB.
C     X , E , AX ARE WORK VECTORS. THEIR SIZES
C     ARE AS FOLLOWS.
C     X(2*NORD+4),E(2*NORD+4),AX(2*M1)
C
      DIMENSION A(49,202),B(52),U(52,52),X(52),E(52),
     *R1(2,101),R2(2,202),CT(24,101),AX(202)
      COMMON/B30/NNC,NDC,M1
      COMMON/B35/R1,R2,CT,A,B
      REAL*8 X1,Y1,X2
      REAL*8 XLIM
      KSIM=0
      KK1=M1+M1
      KK2=KK1+KK1
      NCR1=NCR-1
      NEQ=NCR+2
      IKT=2
      SF=2.0
      DO 2 I=1,NEQ
      DO 2 J=1,NEQ
    2 U(I,J)=0.0
      DO 4 I=1,NEQ
    4 U(I,I)=1.0
      ITR=1
      EMU=-1.E-04
      EMV=-EMU
      IX=0
      XLIM=1.D+75
   24 XMIN=1.E+75
      IOVER=0
      DO 5 I=1,NVR
      SUM=0.
      IF(I.GT.KK1) GO TO 100
      DO 6 J=1,NCR1
      IF(U(NEQ,J).EQ.0.0) GO TO 6
      X1=U(NEQ,J)
      Y1=A(J,I)
      X2=X1*Y1
      IF(X2.GE.XLIM) IOVER=IOVER+1
      IF(IOVER.GE.5) GO TO 37
      X11=SNGL(X2)
      SUM=SUM+X11
    6 CONTINUE
      AX(I)=SUM
      DELJ=SUM+U(NEQ,NCR)+R2(IKT,I)
      GO TO 102
  100 IF(I.GT.KK2) GO TO 101
      DELJ=-AX(I-KK1)+U(NEQ,NCR)-(R2(IKT,I-KK1)+SF)
```

```
      GO TO 102
  101 IKK2=I-KK2
      DO 66 J=1,NDC
      IF(U(NEQ,J).EQ.0.0) GO TO 66
      SUM=SUM+U(NEQ,J)*CT(J,IKK2)
   66 CONTINUE
      DELJ=SUM+R1(IKT,IKK2)
  102 IF(XMIN.LE.DELJ) GO TO 5
      XMIN=DELJ
      KK=I
    5 CONTINUE
      IF(IX.EQ.0) GO TO 21
      IF(XMIN.GE.-.1E-04) GO TO 22
   21 IF(KK.GT.KK2) GO TO 200
      IF(KK.GT.KK1) GO TO 201
      KKCOL=KK
      GO TO 202
  201 KKCOL=KK-KK1
  202 DO 7 I=1,NCR
      X(I)=0.0
      DO 8 J=1,NCR1
      IF(U(I,J).EQ.0.0) GO TO 8
      X(I)=X(I)+U(I,J)*A(J,KKCOL)
    8 CONTINUE
      IF(KK.GT.KK1) X(I)=-X(I)
    7 X(I)=X(I)+U(I,NCR)
      DO 77 LZ=1,IKT
      LZ1=NCR+LZ
      X(LZ1)=0.
      DO 88 JZ=1,NCR1
   88 X(LZ1)=X(LZ1)+U(LZ1,JZ)*A(JZ,KKCOL)
      IF(KK.GT.KK1) GO TO 301
      X(LZ1)=X(LZ1)+U(LZ1,NCR)+R2(LZ,KKCOL)
      GO TO 77
  301 IF(LZ.EQ.1) X(LZ1)=-X(LZ1)+U(LZ1,NCR)-R2(LZ,KKCOL)
      IF(LZ.EQ.2) X(LZ1)=-X(LZ1)+U(LZ1,NCR)-(R2(LZ,KKCOL)+SF)
   77 CONTINUE
      GO TO 203
  200 KKCOL=KK-KK2
      KNT=0
      DO 80 I=1,NEQ
      X(I)=0.0
      DO 81 J=1,NDC
      IF(U(I,J).EQ.0.0) GO TO 81
      X(I)=X(I)+U(I,J)*CT(J,KKCOL)
   81 CONTINUE
      IF(I.LE.NCR) GO TO 80
      KNT=KNT+1
      X(I)=X(I)+R1(KNT,KKCOL)
   80 CONTINUE
  203 XMIN=1.E70
      KKJ=0
      DO 10 I=1,NCR
```

```
      IF(X(I).LE.0.) GO TO 10
      THETA=B(I)/X(I)
      IF(XMIN.LT.THETA) GO TO 10
      XMIN=THETA
      KKJ=I
   10 CONTINUE
      IF(KKJ.EQ.0) GO TO 35
      DO 12 I=1,NEQ
   12 E(I)=-(X(I)/X(KKJ))
      E(KKJ)=1./X(KKJ)
      Y1=B(KKJ)
      DO 14 I=1,NEQ
   14 B(I)=B(I)+E(I)*Y1
      B(KKJ)=E(KKJ)*Y1
      DO 15 I=1,NEQ
      Y1=U(KKJ,I)
      DO 16 J=1,NEQ
   16 U(J,I)=U(J,I)+E(J)*Y1
   15 U(KKJ,I)=E(KKJ)*Y1
      ITR=ITR+1
      IF(IX.EQ.1) GO TO 19
      IF(B(NEQ).GE.EMU.AND.B(NEQ).LE.EMV) GO TO 18
      IF(B(NEQ).GT.EMV) GO TO 20
      GO TO 19
   18 IX=1
      NEQ=NEQ-1
      IKT=IKT-1
      SF=0.
   19 GO TO 24
   35 PRINT 36
   36 FORMAT(' ',3X,'NO LOWER BOUND FOR OPTIMUM',/)
      KSIM=1
      GO TO 32
   20 PRINT 41
   41 FORMAT(' ',3X,'NO FEASIBLE SOLUTION',/)
      KSIM=1
      WRITE(6,505) IX,ITR,B(NEQ)
  505 FORMAT(' ',3X,'PHASE - ',I1,3X,'NO.  OF ITR= ',I3,3X,
     *'OBJ.  FN.  VAL= ',E14.7,/)
      GO TO 32
   37 WRITE(6,52)
   52 FORMAT(//////,' ',2X,'OVER FLOW IN SIMPLEX ROUTINE',//)
      KSIM=1
      GO TO 32
   22 WRITE(6,42) ITR
   42 FORMAT(' ',3X,'OPTIMAL SOLUTION FOUND.....',3X,
     *'NO.  OF ITERATIONS= ',I6,/)
      WRITE(6,72) B(NCR+1)
   72 FORMAT(/,' ',3X,'THE OPTIMUM VALUE OF THE OBJECTIVE
     *FUNCTION=',E14.7,//)
   32 CONTINUE
      RETURN
      END
```

```
C
      SUBROUTINE PLOTMP(M1,COEF1,COEF2,NNC,NDC,KTIMES)
C
C     SUBROUTINE TO COMPUTE, PRINT AND PLOT OUT THE
C     MAGNITUDE AND GROUP-DELAY CHARACTERSITICS OF THE
C     DESIGNED FILTER. THIS ALSO COMPUTES THE
C     SQUARED ERROR BETWEEN THE DESIRED AND DESIGNED
C     CHARACTERISTICS. H IS THE ARRAY TO STORE
C     COMPUTED MAGNITUDE AND P IS THE ARRAY TO STORE
C     THE GROUP-DELAY VALUES.
C     THE SIZES OF ARRAYS LOCAL TO THIS ROUTINE ARE
C     Z(M1),H(M1),P(M1)
C     M1 IS AS INDICATED IN THE MAIN PROGRAM
C
      DIMENSION H(101),P(101),COEF1(50),COEF2(50),W(101),
     *Z(101),DP(101),DM(101)
      COMPLEX Z,HN,HD,HZ,CMPLX,ZZ,CEXP
      COMPLEX HN1,HD1
      COMMON/B12/DM,DP,W,GDLY,WP1,WP2
      DATA PY/3.141592/
      ERM=0.
      ERG=0.
      WRITE(6,88)
   88 FORMAT(' ',///)
      WRITE(6,87)
   87 FORMAT(' ',5X,'FREQUENCY',4X,'MAGNITUDE',4X,
     *'GROUP-DELAY',/)
      DO 4 I=1,M1
      IF(KTIMES.GE.2) GO TO 500
      ZZ=CMPLX(0.0,-W(I)*PY)
      Z(I)=CEXP(ZZ)
  500 HN1=(0.,0.)
      HD1=(0.,0.)
      HN=(0.0,0.0)
      HD=(0.0,0.0)
      DO 5 J=1,NNC
      HN=HN+COEF1(J)*(Z(I)**(J-1))
    5 HN1=HN1+(J-1)*COEF1(J)*(Z(I)**(J-1))
      DO 6 J=1,NDC
      HD=HD+COEF2(J)*(Z(I)**(J-1))
    6 HD1=HD1+(J-1)*COEF2(J)*(Z(I)**(J-1))
      HZ=HN/HD
      H(I)=CABS(HZ)
      P(I)=REAL((HN1/HN)-(HD1/HD))
    4 WRITE(6,82) W(I),H(I),P(I)
   82 FORMAT(' ',3(2X,E12.5))
      MT=M1
      KB=0
      DO 7 I=1,MT
      KB=KB+1
      ERM=ERM+(DM(I)-H(KB))**2
      IF(W(KB).LT.WP1.OR.W(KB).GT.WP2) GO TO 7
      ERG=ERG+(GDLY-P(KB))**2
    7 CONTINUE
```

```
      WRITE(6,90) ERM,ERG
   90 FORMAT(' ',2X,'SUM OF SQUARED ERROR IN MAGNITUDE IS=',
     *E12.5,/,' ',2X,'SUM OF SQUARED ERROR IN GROUP-DELAY IN
     * PASS BAND IS= ',E12.5)
      WRITE(6,50) M1,DW
      CALL PLOT3(W,H,M1)
      WRITE(6,60) M1,DW
      CALL PLOT3(W,P,M1)
   50 FORMAT(1H1,'DESIGNED MAGNITUDE RESPONSE',2X,'NO. OF
     * POINTS ON FREQ AXIS=',I2,2X,'FREQ INTERVAL=',E12.5)
   60 FORMAT(1H1,'DESIGNED GROUP DELAY RESPONSE',2X,'NO. OF
     * POINTS ON FR AXIS=',I2,2X,'FREQ INTERVAL=',E12.5)
      RETURN
      END
```

```
      DIMENSION DM(33,17),DP(33,17),W1(33),W2(17),
     *AC(6,6),BC(6,6),U(34,34)
C
C     ***********************************************************
C
C     MAIN PROGRAM - DESIGNING TWO-DIMENSIONAL RECURSIVE
C     DIGITAL FILTER TO DESIRED MAGNITUDE AND LINEAR
C     PHASE CHARACTERISTICS.
C     THE INPUT PARAMETERS ARE :-
C     M1 : # OF FREQUENCY POINTS ALONG THE W1-AXIS
C     N1 : # OF FREQUENCY POINTS ALONG THE W2-AXIS
C     IFT : DESIRED FILTER TYPE
C           LPF :- LOW PASS ; BPF :- BAND PASS
C           HPF :- HIGH PASS
C     WP1 , WP2 :- LOWER AND UPPER RADIAL PASS BAND
C                  FREQUENCIES
C     ISETS : # OF GROUP DELAY VALUES TO BE TRIED
C     NP : HIGHEST POWER OF COMPLEX FREQUENCY VARIABLE Z1
C          IN THE NUMERATOR OF THE TRANSFER FUNCTION
C     NQ : HIGHEST POWER OF COMPLEX FREQUENCY VARIABLE Z2
C          IN THE NUMERATOR OF THE TRANSFER FUNCTION
C     SIMILERLY MP & MQ ARE FOR THE DENOMINATOR OF THE
C     DIGITAL FILTER TRANSFER FUNCTION
C     LET  -  IT=(NP+1)*(NQ+1)+(MP+1)*(MQ+1)-1
C     THEN THE DIMENSIONS OF THE ARRAYS ARE AS FOLLOWS :-
C     W1(M1),W2(N1),DM(M1,N1),DP(M1,N1),AC(NP,NQ),
C     BC(MP,MQ),U(IT+3,IT+3)
C
C     ****************************************************************
C
      COMMON/B8/DM,DP,W1,W2,WP1,WP2
      COMMON/B7/MP1,MQ1,NP1,NQ1,NN,ND,MUNU
      PY=3.1415926
      PY2=PY+PY
      READ(5,800) M1,N1
      DW1=2./(M1-1)
      DW2=1./(N1-1)
      WRITE(6,908)
      WRITE(6,909)
      WRITE(6,910) M1,N1
      WRITE(6,911) DW1,DW2
      WRITE(6,901)
      W1(1)=-1.
      DO 1 I=2,M1
    1 W1(I)=W1(I-1)+DW1
      W2(1)=0.
      DO 2 I=2,N1
    2 W2(I)=W2(I-1)+DW2
      KTIME=1
      READ(5,603) IFT
      READ(5,604) WP1,WP2
  603 FORMAT(A4)
```

```
  604 FORMAT(2E12.5)
      CALL SPECM(M1,N1,IFT)
      READ(5,802) ISETS
      READ(5,801) NP,NQ,MP,MQ
      MP1=MP+1
      MQ1=MQ+1
      NP1=NP+1
      NQ1=NQ+1
      WRITE(6,904) NP,NQ
      WRITE(6,907) MP,MQ
      GDLY2=MQ
      GDLY1=MP
      WRITE(6,902)
      CALL OUTPUT(DM,M1,N1)
      WRITE(6,901)
      DO 300 KTIMES=1,ISETS
      WRITE(6,912) KTIMES
      WRITE(6,913) GDLY1,GDLY2
C
C     GENERATION OF LINEAR PHASE CHARACTERISTICS
C
      DO 6 I=1,M1
      PH=GDLY1*W1(I)*PY
      DO 6 J=1,N1
      DP(I,J)=-(PH+GDLY2*W2(J)*PY)
  505 IF(DP(I,J).LT.-PY) DP(I,J)=DP(I,J)+PY2
      IF(DP(I,J).LT.-PY) GO TO 505
  506 IF(DP(I,J).GT.PY) DP(I,J)=DP(I,J)-PY2
      IF(DP(I,J).GT.PY) GO TO 506
    6 CONTINUE
      CALL SIMPTB(M1,N1,NCR,NVR)
      CALL RSIMP(U,NCR,NVR,KSIM)
      IF(KSIM.EQ.1) GO TO 1000
      NT=NN+ND
      NA=NT
      NT1=NA+1
      NT2=NA+2
      NT3=NA+3
      BC(1,1)=1.
      KM=0
      DO 10 I=1,MP1
      KI=1
      IF(I.EQ.1) KI=2
      DO 10 J=KI,MQ1
      KM=KM+1
   10 BC(I,J)=-U(NT2,KM)
      KM=NT+ND
      L=0
      DO 11 I=1,NP1
      DO 11 J=1,NQ1
      L=L+1
   11 AC(I,J)=-U(NT2,ND+L)
      WRITE(6,901)
      WRITE(6,900) (U(NT2,I),I=1,NT3)
```

```
      WRITE(6,901)
      WRITE(6,905)
      DO 12 I=1,MP1
      DO 12 J=1,MQ1
      IF(I.GT.NP1.OR.J.GT.NQ1) AC(I,J)=0.
   12 WRITE(6,906) I,J,AC(I,J),I,J,BC(I,J)
      CALL PLOTMP(M1,N1,AC,BC,KTIME,GDLY1,GDLY2)
      KTIME=KTIME+1
 1000 CONTINUE
      WRITE(6,901)
      GDLY1=GDLY1-1
      GDLY2=GDLY2-1
  300 CONTINUE
  800 FORMAT(2I2)
  801 FORMAT(4I2)
  802 FORMAT(I2)
  900 FORMAT(' ',2X,11(F10.4,1X))
  901 FORMAT(' ',////)
  902 FORMAT(' ',2X,'DESIRED MAGNITUDE SPECIFICATI
     *ONS',/)
  904 FORMAT(' ',2X,'NUMERATOR INDICES P=',I2,2X,'&
     *  Q=',I2,//)
  905 FORMAT(' ',2X,'THE COEFFICIENTS OF THE FILTER
     * ARE:-------------',/)
  906 FORMAT(' ',2X,'N(',I2,',',I2,')=',G15.8,5X,'
     *D(',I2,',',I2,')=',G15.8,/)
  907 FORMAT(' ',2X,'DENOMINATOR INDICES P=',I2,2X,
     *'&  Q=',I2,//)
  908 FORMAT(' ',2X,'DESIRED SPECS (FREQUENCIES ARE
     *IN FRACTIONS OF NYQUIST RATE) .----',/).
  909 FORMAT(' ',2X,'********************************',//)
  910 FORMAT(' ',2X,'NO. OF SPECS-POINTS ALONG W1-A
     *XIS=',I2,3X,'NO. OF SPECS-POINTS ALONG W2-AXI
     *S=',I2,//)
  911 FORMAT(' ',2X,'FREQ-INTERVAL ALONG W1-AXIS=',
     *E12.5,3X,'FREQ-INTERVAL ALONG W2-AXIS=',E12.5
     *,////)
  912 FORMAT(' ',2X,'SET NO.=',I2,/)
  913 FORMAT(' ',2X,'GROUP-DELAY ALONG W1-AXIS=',E1
     *2.5,5X,'GROUP-DELAY ALONG W2-AXIS=',E12.5,/)
      STOP
      END
```

```
      SUBROUTINE SPECM(M1,N1,IFT)
C
C     SUBROUTINE TO GENERATE THE MAGNITUDE SPECS
C     CORRESPONDING TO THE DESIRED TYPE OF FILTER
C     THE CHARACTERISTICS BEYOND THE PASS BAND IS
C     GENERATED ACCORDING TO THE FUNCTION
C     FN=EXP(-FK*((WP-WX)**2)) WHERE FK CAN BE
C     VARIED. FK IS OBTAINED IN TERMS OF FK2.
C     SHARPER TRANSITION BAND IN THE SPECIFICATIONS
C     CAN BE GENERATED BY SPECIFYING SMALLER VALUES
C     FOR FK2. THE ARRAY PB(M1,N1) IS A LOGICAL
C     ARRAY WHICH IS TRUE IN THE PASS BAND AREA OF
C     DESIRED SPECIFICATION
C
C     FUCTION SUBPROGRAM REQUIRED :- FN
C
      DIMENSION DM(33,17),RP(33,17),W1(33),W2(17),
     *PB(33,17)
      COMPLEX*16 TP1,TP2,TP3,TP4
      LOGICAL*1 AX/.TRUE./,BX/.FALSE./,PB
      INTEGER BP,HP
      COMMON/B8/DM,RP,W1,W2,WP1,WP2
      COMMON/B30/PB
      DATA TP1/'LOW  PASS FILTER'/
      DATA TP2/'BAND PASS FILTER'/
      DATA TP3/'HIGH PASS FILTER'/
      DATA BP/'BPF '/,HP/'HPF '/,PY/3.141592/
      DO 15 I=1,M1
      DO 15 J=1,N1
   15 PB(I,J)=BX
      FK2=2.0
      FK=100./(FK2**2)
      DO 1 I=1,M1
      R1=W1(I)*W1(I)
      DO 1 J=1,N1
    1 RP(I,J)=SQRT(R1+W2(J)*W2(J))
      IF(IFT.EQ.BP) GO TO 100
      IF(IFT.EQ.HP) GO TO 101
      TP4=TP1
      DO 2 I=1,M1
      DO 2 J=1,N1
      IF(RP(I,J).GT.WP2) GO TO 3
      PB(I,J)=AX
      DM(I,J)=1.0
      GO TO 2
    3 DM(I,J)=FN(FK,WP2,RP(I,J))
    2 CONTINUE
      GO TO 102
  100 TP4=TP2
      DO 4 I=1,M1
      DO 4 J=1,N1
      IF(RP(I,J).LT.WP1) GO TO 5
      IF(RP(I,J).GT.WP2) GO TO 6
```

```
      PB(I,J)=AX
      DM(I,J)=1.0
      GO TO 4
    5 DM(I,J)=FN(FK,WP1,RP(I,J))
      GO TO 4
    6 DM(I,J)=FN(FK,WP2,RP(I,J))
    4 CONTINUE
      GO TO 102
  101 TP4=TP3
      DO 7 I=1,M1
      DO 7 J=1,N1
      IF(RP(I,J).GE.WP1) GO TO 8
      DM(I,J)=FN(FK,WP1,RP(I,J))
      GO TO 7
    8 PB(I,J)=AX
      DM(I,J)=1.0
    7 CONTINUE
  102 CONTINUE
      WRITE(6,500) TP4
      WRITE(6,501) WP1,WP2
  500 FORMAT(/,' ',2X,'FILTER TYPE :--- ',A16,/)
  501 FORMAT(/,' ',2X,'PASS BAND FREQUENCIES:--  WP1=
     *',E12.5,2X,'WP2=',E12.5,/)
      RETURN
      END
      FUNCTION FN(FK,WP,WX)
C
C     FUNCTION SUBPROGRAM REQUIRED BY THE SUB-
C     ROUTINE SPECM
C
      ARG=FK*((WP-WX)**2)
      FN=0.0
      IF(ARG.GT.157.0) GO TO 501
      FN=EXP(-ARG)
  501 RETURN
      END
C
      SUBROUTINE SIMPTB(M1,N1,MP,M0,NP,N0,NCR,NVR)
C
C     SUBROUTINE TO GENERATE THE SIMPLEX TABLEAU
C     IT,M1,N1 : IS AS DEFINED IN THE MAIN PROGRAM
C     LET ITX= MP1*MQ1-1 & ITY=M1*N1
C     THE DIMENSIONS OF THE ARRAYS ARE AS FOLLOWS
C     A(IT,2*ITY),B(IT+3),CT(ITX,ITY),R1(2,ITY)
C     R2(2,2*ITY)
C
      DIMENSION A(31,1122),B(34),DM(33,17),DP(33,17),
     *W1(33),W2(17),CT(15,561),R1(2,561),R2(2,1122)
      DIMENSION YR(33,17),YI(33,17)
      COMMON/B8/DM,DP,W1,W2,WP1,WP2
      COMMON/B7/MP1,MQ1,NP1,NQ1,NN,ND,IC
      COMMON/B35/A,B,CT,R1,R2
      PY=3.1415926
      ICT=1
```

```
      ND=MP1*MQ1-1
      IC=M1*N1
      NN=NP1*NQ1
      IC1=IC+IC
      IB2=ND+NN
      IC2=IC1+IC
      IC4=IC2+IC
      IC5=IC4+IC
      EPS=1.E-03
      IL=0
      IL1=IC
      IL2=IC4
      DO 1 L=1,M1
      IK=0
      DO 2 K=1,MP1
      IK1=K-1
      IMQ1=MQ1
      IF(K.EQ.1) IMQ1=MQ1-1
      DO 3 J=1,N1
      IM=0
      IF(K.EQ.1) IM=1
      DO 3 I=1,IMQ1
      XX=(IK1*W1(L)+IM*W2(J))*PY
      AX2=COS(XX)
      AY2=SIN(XX)
      AY1=SIN(DP(L,J))
      AX1=COS(DP(L,J))
      A(IK+I,IL1+J)=DM(L,J)*(AY1*AX2-AX1*AY2)
      A(IK+I,IL+J)=DM(L,J)*(AX1*AX2+AY1*AY2)
      CT(IK+I,IL+J)=-COS(XX)
    3 IM=IM+1
      IK=IK+MQ1
    2 IF(K.EQ.1) IK=IK-1
      IL=IL+N1
      IL1=IL1+N1
    1 IL2=IL2+N1
      IL=0
      IL1=IC
      IL2=IC4
      DO 5 L=1,M1
      IK=ND
      DO 6 K=1,NP1
      IK1=K-1
      DO 7 J=1,N1
      DO 7 I=1,NQ1
      IM=I-1
      XX=(IK1*W1(L)+IM*W2(J))*PY
      A(IK+I,IL+J)=-COS(XX)
    7 A(IK+I,IL1+J)=SIN(XX)
    6 IK=IK+NQ1
      IL=IL+N1
      IL1=IL1+N1
    5 IL2=IL2+N1
      IN1=IB2+1
```

```
      IN2=IB2+2
      IN3=IB2+3
      EPS1=1.-EPS
      IL=0
      DO 12 I=1,M1
      IL1=IL+IC
      DO 14 J=1,N1
      R2(1,IL+J)=-DM(I,J)*COS(DP(I,J))
   14 R2(1,IL1+J)=-DM(I,J)*SIN(DP(I,J))
   12 IL=IL+N1
      DO 15 I=1,IB2
   15 B(I)=0.
      B(IN1)=1.
      B(IN2)=0.
      B(IN3)=-1.
      DO 20 I1=1,IC1
      SUM=0.0
      DO 21 I2=1,IB2
   21 SUM=SUM+A(I2,I1)
   20 R2(2,I1)=-(SUM+1.0)
      DO 22 I1=1,IC
      SUM=0.0
      DO 23 I2=1,ND
   23 SUM=SUM+CT(I2,I1)
      R1(1,I1)=EPS1
   22 R1(2,I1)=-SUM
      NVR=IC5
      IF(ICT.EQ.0) NVR=IC4
      NCR=IN1
      RETURN
      END
C
      SUBROUTINE RSIMP(U,NCR,NVR,KSIM)
C
C     SUBROUTINE TO SOLVE THE LINEAR PROGRAMMING
C     PROBLEM USING THE REVISED SIMPLEX METHOD
C     THIS PROGRAM USES THE SIMPLEX TABLEAU
C     GENERATED IN THE ROUTINE SIMPTB.  X,E,AX
C     ARE WORK VECTORS.  THEIR SIZES ARE AS
C     FOLLOWS :-
C     X(IT+3) , E(IT+3) , AX(2*ITY)
C     IT :- IS AS DEFINED IN THE MAIN PROGRAM
C     ITY :-  IS AS DEFINED IN THE ROUTINE SIMPTB
C
      DIMENSION A(31,1122),B(34),U(34,34),X(34),E(34),
     *CT(15,561),R1(2,561),R2(2,1122),AX(1122)
      COMMON/B7/MP1,MQ1,NP1,NQ1,NNC,NDC,M1
      COMMON/B35/A,B,CT,R1,R2
      REAL*8 X1,Y1,X2
      REAL*8 XLIM
      KSIM=0
      KK1=M1+M1
      KK2=KK1+KK1
      NCR1=NCR-1
```

```
      NEQ=NCR+2
      IKT=2
      SF=2.0
      DO 2 I=1,NEQ
      DO 2 J=1,NEQ
    2 U(I,J)=0.0
      DO 4 I=1,NEQ
    4 U(I,I)=1.0
      ITR=1
      EMU=-1.E-03
      EMV=-EMU
      IX=0
      XLIM=1.D+75
   24 XMIN=1.E+75
      IOVER=0
      DO 5 I=1,NVR
      SUM=0.
      IF(I.GT.KK1) GO TO 100
      DO 6 J=1,NCR1
      IF(U(NEQ,J).EQ.0.0) GO TO 6
      X1=U(NEQ,J)
      Y1=A(J,I)
      X2=X1*Y1
      IF(X2.GE.XLIM) IOVER=IOVER+1
      IF(IOVER.GE.5) GO TO 27
      X11=SNGL(X2)
      SUM=SUM+X11
    6 CONTINUE
      AX(I)=SUM
      DELJ=SUM+U(NEQ,NCR)+R2(IKT,I)
      GO TO 102
  100 IF(I.GT.KK2) GO TO 101
      DELJ=-AX(I-KK1)+U(NEQ,NCR)-(R2(IKT,I-KK1)+SF)
      GO TO 102
  101 IKK2=I-KK2
      DO 66 J=1,NDC
      IF(U(NEQ,J).EQ.0.0) GO TO 66
      SUM=SUM+U(NEQ,J)*CT(J,IKK2)
   66 CONTINUE
      DELJ=SUM+R1(IKT,IKK2)
  102 IF(XMIN.LE.DELJ) GO TO 5
      XMIN=DELJ
      KK=I
    5 CONTINUE
      IF(IX.EQ.0) GO TO 21
      IF(XMIN.GE.-.1E-04) GO TO 22
   21 IF(KK.GT.KK2) GO TO 200
      IF(KK.GT.KK1) GO TO 201
      KKCOL=KK
      GO TO 202
  201 KKCOL=KK-KK1
  202 DO 7 I=1,NCR
      X(I)=0.0
      DO 8 J=1,NCR1
```

```
      IF(U(I,J).EQ.0.0) GO TO 8
      X(I)=X(I)+U(I,J)*A(J,KKCOL)
    8 CONTINUE
      IF(KK.GT.KK1) X(I)=-X(I)
    7 X(I)=X(I)+U(I,NCR)
      DO 77 LZ=1,IKT
      LZ1=NCR+LZ
      X(LZ1)=0.
      DO 88 JZ=1,NCR1
   88 X(LZ1)=X(LZ1)+U(LZ1,JZ)*A(JZ,KKCOL)
      IF(KK.GT.KK1) GO TO 301
      X(LZ1)=X(LZ1)+U(LZ1,NCR)+R2(LZ,KKCOL)
      GO TO 77
  301 IF(LZ.EQ.1) X(LZ1)=-X(LZ1)+U(LZ1,NCR)-R2(LZ,KKCOL)
      IF(LZ.EQ.2) X(LZ1)=-X(LZ1)+U(LZ1,NCR)-(R2(LZ,KKCOL)+SF)
   77 CONTINUE
      GO TO 203
  200 KKCOL=KK-KK2
      KNT=0
      DO 80 I=1,NEQ
      X(I)=0.0
      DO 81 J=1,NDC
      IF(U(I,J).EQ.0.0) GO TO 81
      X(I)=X(I)+U(I,J)*CT(J,KKCOL)
   81 CONTINUE
      IF(I.LE.NCR) GO TO 80
      KNT=KNT+1
      X(I)=X(I)+R1(KNT,KKCOL)
   80 CONTINUE
  203 CONTINUE
      XMIN=1.E70
      KKJ=0
      DO 10 I=1,NCR
      IF(X(I).LE.0.) GO TO 10
      THETA=B(I)/X(I)
      IF(XMIN.LT.THETA) GO TO 10
      XMIN=THETA
      KKJ=I
   10 CONTINUE
      IF(KKJ.EQ.0) GO TO 35
      DO 12 I=1,NEQ
   12 E(I)=-(X(I)/X(KKJ))
      E(KKJ)=1./X(KKJ)
      Y1=B(KKJ)
      DO 14 I=1,NEQ
   14 B(I)=B(I)+E(I)*Y1
      B(KKJ)=E(KKJ)*Y1
      DO 15 I=1,NEQ
      Y1=U(KKJ,I)
      DO 16 J=1,NEQ
   16 U(J,I)=U(J,I)+E(J)*Y1
   15 U(KKJ,I)=E(KKJ)*Y1
      ITR=ITR+1
      IF(IX.EQ.1) GO TO 19
```

```
      IF(B(NEQ).GE.EMU.AND.B(NEQ).LE.EMV) GO TO 18
      IF(B(NEQ).GT.EMV) GO TO 20
      GO TO 19
   18 IX=1
      NEQ=NEQ-1
      IKT=IKT-1
      SF=0.
   19 GO TO 24
   35 PRINT 36
   36 FORMAT(' ',3X,'NO LOWER BOUND FOR OPTIMUM',/)
      KSIM=1
      GO TO 32
   20 PRINT 41
   41 FORMAT(' ',3X,'NO FEASIBLE SOLUTION',/)
      KSIM=1
      WRITE(6,505) IX,ITR,B(NEQ)
  505 FORMAT(' ',3X,'PHASE - ',I1,3X,'NO.  OF ITR= ',I3,
     *3X,'OBJ.  FN. VAL= ',E14.7,/)
      GO TO 32
   37 WRITE(6,52)
   52 FORMAT(//////,' ',2X,'OVER FLOW IN SIMPLEX ROUTIN
     *E',//)
      KSIM=1
      GO TO 32
   22 WRITE(6,42) ITR
   42 FORMAT(' ',3X,'OPTIMAL SOLUTION FOUND.....',3X,'N
     *O.  OF ITERATIONS= ',I6,/)
      WRITE(6,72) B(NCR+1)
   72 FORMAT(/,' ',3X,'THE OPTIMUM VALUE OF THE OBJECTI
     *VE FUNCTION=',E14.7,//)
   32 CONTINUE
      RETURN
      END
C
      SUBROUTINE PLOTMP(M1,N1,A,B,KTIME,GDLY1,GDLY2)
C
C     SUBROUTINE TO COMPUTE AND PRINT THE MAGNITUDE AND
C     GROUP-DELAY CHARACTERISTICS OF THE TWO-DIMENSIONAL
C     RECURSIVE DIGITAL FILTER.  THIS PROGRAM ALSO
C     COMPUTES THE SQUARED ERROR BETWEEN THE DESIRED &
C     DESIGNED CHARACTERISTICS.  H IS THE ARRAY TO
C     STORE THE MAGNITUDE , T1 IS THE ARRAY TO STORE
C     THE GROUP-DELAY IN +VE VERTICAL DIRECTION AND T2
C     IS THE ARRAY TO STORE GROUP-DELAY IN THE +VE
C     HORIZONTAL DIRECTION.  THE SIZE OF ARRAYS LOCAL
C     TO THIS SUBROUTINE ARE AS FOLLOWS :-
C     H(M1,N1) , T1(M1,N1) , T2(M1,N1) , Z1(M1)
C     Z2(N1)
C
      DIMENSION A(6,6),B(6,6),Z1(33),Z2(17),W1(33),W2(17),
     *H(33,17),T1(33,17),T2(33,17)
      DIMENSION PB(33,17),DM(33,17),DP(33,17)
      LOGICAL *1 PB,BZ
      COMPLEX Z1,Z2,ZZ,CEXP,CMPLX,HN,HD,HZ,HN1,HN2,HD1,HD2,
```

```
     *HNT,ZA,ZA
      COMMON/B7/MP1,MQ1,NP1,NQ1,NN,ND,MUNU
      COMMON/B8/DM,DP,W1,W2,WP1,WP2
      COMMON/B30/PB
      PY=3.1415926
      IF(KTIME.GE.2) GO TO 6
      Z1(1)=(-1.0,0.0)
      DO 1 I=2,M1
      ZZ=CMPLX(0.,W1(I)*PY)
    1 Z1(I)=CEXP(-ZZ)
      Z2(1)=(1.,0.)
      DO 2 J=2,N1
      ZZ=CMPLX(0.,W2(J)*PY)
    2 Z2(J)=CEXP(-ZZ)
    6 CONTINUE
      ERM=0.
      ERT1=0.
      ERT2=0.
      DO 3 I=1,M1
      DO 3 J=1,N1
      BZ=PB(I,J)
      HN=(0.,0.)
      HD=(0.,0.)
      HN1=(0.,0.)
      HD1=(0.,0.)
      HN2=(0.,0.)
      HD2=(0.,0.)
      DO 4 K=1,NP1
      ZA=Z1(I)**(K-1)
      DO 4 L=1,NQ1
      HNT=A(K,L)*(ZA*(Z2(J)**(L-1)))
      HN=HN+HNT
      HN1=HN1+(K-1)*HNT
    4 HN2=HN2+HNT*(L-1)
      DO 5 K=1,MP1
      ZA=Z1(I)**(K-1)
      DO 5 L=1,MQ1
      HNT=B(K,L)*(ZA*(Z2(J)**(L-1)))
      HD=HD+HNT
      HD1=HD1+(K-1)*HNT
    5 HD2=HD2+(L-1)*HNT
      H(I,J)=CABS(HN/HD)
      T1(I,J)=REAL((HN1/HN)-(HD1/HD))
      T2(I,J)=REAL((HN2/HN)-(HD2/HD))
      ERM=ERM+(DM(I,J)-H(I,J))**2
      IF(.NOT.BZ) GO TO 3
      ERT1=ERT1+(GDLY1-T1(I,J))**2
      ERT2=ERT2+(GDLY2-T2(I,J))**2
    3 CONTINUE
      WRITE(6,700)
      WRITE(6,701)
      CALL OUTPUT(H,M1,N1)
      WRITE(6,704) ERM
      WRITE(6,700)
```

```
      WRITE(6,702)
      CALL OUTPUT(T1,M1,N1)
      WRITE(6,705) ERT1
      WRITE(6,700)
      WRITE(6,703)
      CALL OUTPUT(T2,M1,N1)
      WRITE(6,706) ERT2
  700 FORMAT(' ',///)
  701 FORMAT(' ',2X,'THE DESIGNED MAGNITUDE RESPONSE',/)
  702 FORMAT(' ',2X,'GD1:- THE GROUP-DELAY ALONG VERT-DI
     *RECT',/)
  703 FORMAT(' ',2X,'GD2:- THE GROUP-DELAY ALONG HORZ-DI
     *RECT',/)
  704 FORMAT(//,' ',2X,'SUM OF THE SQUARED ERRORS IN MAG
     *= ',E12.5)
  705 FORMAT(//,' ',2X,'SUM OF THE SQUARED ERRORS IN GD1
     *= ',E12.5)
  706 FORMAT(//,' ',2X,'SUM OF THE SQUARED ERRORS IN GD2
     *= ',E12.5)
      RETURN
      END
C
      SUBROUTINE OUTPUT(A,M,N)
C
C     SUBROUTINE TO PRINT OUT THE OUTPUT
C     REQUIRED BY SUBROUTINE PLOTMP
C
      DIMENSION A(33,17)
      NCOL=11
      N2=(N/11)+1
      NREM=N-(N2-1)*11
      IF(N.GT.11) GO TO 1
      N2=1
      NREM=N
    1 CONTINUE
      KCOL=0
      DO 6 J=1,N2
      IF(J.EQ.N2) NCOL=NREM
      DO 7 I=1,M
    7 WRITE(6,500) (A(I,KCOL+K),K=1,NCOL)
      WRITE(6,501)
      KCOL=KCOL+NCOL
    6 CONTINUE
  500 FORMAT(' ',2X,11(F10.4,1X))
  501 FORMAT(' ',//)
      RETURN
      END
```

```
      EXTERNAL FUNCT
      DIMENSION X(100),DQ(100),W1(41),W2(21),Z1(41),Z2(21)
C
C     MAIN PROGRAM :- DESIGNING TWO DIMENSIONAL RECURSIVE
C     DIGITAL FILTERS IN CASCADES OF 1ST AND 2ND ORDER
C     SECTIONS USING NON-LINEAR OPTIMIZATION PROCEDURE
C     IN THIS PROGRAM GIVEN DATA CAN BE APPROXIMATED TO
C     THE REAL PART OR THE MAGNITUDE SQUARED TRANSFER
C     FUNCTION OF THE RECURSIVE DIGITAL FILTER
C     THE INPUT DATA ARE :-
C     M1 -- # OF FREQUENCY POINTS IN W1 AXIS
C     M2 - # OF FREQUENCY POINTS IN W2 AXIS
C     KS,KF - # OF 2ND AND 1ST ORDER CASCADES RESPECTIVELY
C     LIMIT - # OF ITERATIONS , ER - DESIRED FREQUENCY
C     DOMAIN SPECS , X - INITIAL PARAMETER VALUES (ARE TO
C     BE NON-ZERO VALUES) , EST - ESTIMATED MINIMUM OF
C     THE OBJECTIVE FUNCTION , EPS - STEP SIZE , LP - THE
C     EXPONENTIAL FACTOR OF THE ERROR (VALUES SHOULD BE
C     2,4,6, ETC.)
C     THE METHOD USES THE FLECTHER AND POWELLS OPTIMIZATION
C     PROCEDURE.
C     SUBROUTINES REQUIRED :- FUNCT , OUTPUT ,DREAL , DIMAG
C     AND DFMFP(FLECTHER AND POWELS OPTIMIZATION SCHEME)
C
C     THE DIMENSIONS OF THE VARIOUS ARRAYS ARE :-
C     X(N),DQ(N),W1(M1),W2(M2),Z1(M1),Z2(M2),ZA(M1),ZB(M1)
C     YS(KS,16),YF(KF,6),IR(M2),S(9*KS),F(KF*4),TA(M1),
C     TB(M2),TD(M2),TC(M1),ER(M1,M2),RP(M1,M2),TE(M1,M2)
C     TF(M1,M2),TG(M1,M2),HC(N*(N+7)/2) WHERE
C     N EQUALS KS*18+KF*6+1 WHICH IS EQUAL TO THE
C     # OF PARAMETERS IN THE DESIGN PROBLEM
C
     *,ZA(41),ZB(21),YS(4,16),YF(4,6),IR(41)
      DIMENSION S(55),F(32),R(8),TA(41),TB(21),TC(41),TD(21),
     *ER(41,21),RP(41,21),TE(41,21),TF(41,21),TG(41,21),HC(400)
      REAL*8 DABS,DSQRT,DEXP
      REAL*8 X,DQ,HC,Q,YS,YF,DREAL,S,F,R
      COMPLEX ZZ,CMPLX,CEXP,Z1,Z2,ZA,ZB,HN,HD,POLY,P
      LOGICAL*1 ITYF,IREP,IMGS
      COMMON/B1/YS,YF,ER,RP,TA,TB,TC,TD,TE,TF,TG,AO
      COMMON/B2/KOUNT,M1,M2,KS,KF,LP
      COMMON/B3/ITYF
      DATA IREP/.TRUE./,IMGS/.FALSE./
      PY=3.141593
      PY2=PY/2.
      KOUNT=0
C
C     SET ITYF=IREP FOR REAL PART APPROXIMATION
C     IF NOT SET ITYF=IMGS FOR MAGNITUDE SQUARED
C     APPROXIMATION .
C
      ITYF=IREP
```

```
C
C     LP IS TO BE SET EQUAL TO 2 ,  4 ,  6 ETC I.E.
C     ANY EVEN NUMBERED INTEGER
C
      LP=2
C
C     READ IN # OF FREQUENCY POINTS ALONG W1 AND W2
C     AXIS IN THE FREQUENCY DOMAIN
C
      READ(5,500) M1,M2
C
C     READ IN THE # OF 2ND AND 1ST ORDER SECTIONS
C     RESPECTIVELY
C
      READ(5,500) KS,KF
C
      DW1=2./M1
      DW2=1./(M2-1)
      W1(1)=-1.
      W2(1)=0.
      DO 1 I=2,M1
    1 W1(I)=W1(I-1)+DW1
      DO 2 I=2,M2
    2 W2(I)=W2(I-1)+DW2
      W1(1)=W1(1)+0.0125
      W2(M2)=W2(M2)-0.0125
      DO 3 I=1,M1
      ZZ=CMPLX(0.,W1(I)*PY)
      TA(I)=TAN(W1(I)*PY2)
      TC(I)=TA(I)*TA(I)
      Z1(I)=CEXP(-ZZ)
    3 ZA(I)=Z1(I)*Z1(I)
      DO 4 I=1,M2
      TB(I)=TAN(W2(I)*PY2)
      TD(I)=TB(I)*TB(I)
      ZZ=CMPLX(0.,W2(I)*PY)
      Z2(I)=CEXP(-ZZ)
    4 ZB(I)=Z2(I)*Z2(I)
      DO 100 I=1,M1
      DO 100 J=1,M2
      TE(I,J)=TA(I)*TB(J)
      TF(I,J)=TC(I)*TB(J)
  100 TG(I,J)=TA(I)*TD(J)
      YMAX1=0.
C
C     READ IN THE FREQUENCY DOMAIN SPECIFICATINS
C     INTO THE ARRAY ER AND NORMALIZE THE DATA .
C
      DO 554 I=1,M1
  554 READ(5,556) (IR(K),K=1,M2)
      DO 554 J=1,M2
      YMAX=ABS(FLOAT(IR(J)))
      YMAX1=AMAX1(YMAX1,YMAX)
  554 ER(I,J)=IR(J)
```

```
  556 FORMAT(16I5)
      DO 557 I=1,M1
      DO 557 J=1,M2
  557 ER(I,J)=ER(I,J)/YMAX1
C
      WRITE(6,558) YMAX1
  558 FORMAT(' ',1X,'MAXIMUM VALUE IN INPUT DATA= ',E12.5)
      NCS=KS*13
      NCF=KF*6
      N=NCS+NCF+1
C
C     READ IN THE INITIAL PARAMETER VALUES
C
      READ(5,555) (X(I),I=1,N)
  555 FORMAT(3D23.16)
C
      IF(.NOT.ITYF) GO TO 27
      WRITE(6,650)
      GO TO 28
   27 WRITE(6,29)
   29 FORMAT(' ',1X,'THIS IS A MAGNITUDE SQUARED APPROXIMATION',/)
   28 WRITE(6,600) M1,M2
      WRITE(6,601) KS,KF
      WRITE(6,602)
      WRITE(6,603) X(N)
      N1=N/2
      DO 19 I=1,N1
      NX1=N1+I
   19 WRITE(6,604) I,X(I),NX1,X(NX1)
      WRITE(6,618)
      WRITE(6,605)
      CALL OUTPUT(ER,M1,M2)
      EST=0.1
      EPS=1.E-5
      LIMIT=40
C
      CALL DFMFP(FUNCT,N,X,Q,DQ,EST,EPS,LIMIT,IER,HC)
C
      WRITE(6,607) KOUNT
      WRITE(6,608) IER
      IF(IER)10,11,12
   11 WRITE(6,610)
      GO TO 10
   12 WRITE(6,609)
   10 WRITE(6,612) Q
      WRITE(6,613)
      DO 20 I=1,N1
      NX1=N1+I
   20 WRITE(6,604) I,X(I),NX1,X(NX1)
      WRITE(6,618)
      WRITE(6,614)
      WRITE(6,615)
      WRITE(6,616)
      WRITE(6,617)
```

```
      WRITE(6,621)
      WRITE(6,622)
      WRITE(6,623)
      WRITE(6,624)
      WRITE(6,603) X(N)
      WRITE(6,606)
      IF(KS.EQ.0) GO TO 700
      KK=-8
      KK1=-9
      DO 141 I=1,KS
      DO 141 J=1,2
      KK=KK+8
      KK1=KK1+9
      DO 142 L=1,8
  142 R(L)=YS(I,KK+L)
      S(KK1+1)=R(1)+R(2)+R(3)+R(4)+R(5)+R(6)+R(7)+R(8)+1.D0
      S(KK1+2)=2.D0*(R(1)-R(3)+R(4)-R(6)+R(7)-1.D0)
      S(KK1+3)=R(1)-R(2)+R(3)+R(4)-R(5)+R(6)+R(7)-R(8)+1.D0
      S(KK1+4)=2.D0*(R(1)+R(2)+R(3)-R(7)-R(8)-1.D0)
      S(KK1+5)=4.D0*(R(1)-R(3)-R(7)+1.D0)
      S(KK1+6)=2.D0*(R(1)-R(2)+R(3)-R(7)+R(8)-1.D0)
      S(KK1+7)=R(1)+R(2)+R(3)-R(4)-R(5)-R(6)+R(7)+R(8)+1.D0
      S(KK1+8)=2.D0*(R(1)-R(3)-R(4)+R(6)+R(7)-1.D0)
  141 S(KK1+9)=R(1)-R(2)+R(3)-R(4)+R(5)-R(6)+R(7)-R(8)+1.D0
      KK=-18
      DO 14 I=1,KS
      WRITE(6,625) I
      KK=KK+18
      DO 15 J=1,9
      JJ=J-1
   15 WRITE(6,619) JJ,S(KK+J),JJ,S(KK+9+J)
   14 WRITE(6,618)
  700 IF(KF.EQ.0) GO TO 701
      KK=-3
      KK1=-4
      DO 161 I=1,KF
      DO 161 J=1,2
      KK=KK+3
      KK1=KK1+4
      DO 162 L=1,3
  162 R(L)=YF(I,KK+L)
      F(KK1+1)=R(1)+R(2)+R(3)+1.D0
      F(KK1+2)=R(1)-R(2)+R(3)-1.D0
      F(KK1+3)=R(1)+R(2)-R(3)-1.D0
  161 F(KK1+4)=R(1)-R(2)-R(3)+1.D0
      KK=-8
      DO 16 I=1,KF
      II=KS+I
      KK=KK+8
      WRITE(6,625) II
      DO 17 J=1,4
      JJ=J-1
   17 WRITE(6,619) JJ,F(KK+J),JJ,F(KK+4+J)
```

```
   16 WRITE(6,618)
  701 CONTINUE
      DO 25 I=1,M1
      DO 25 J=1,M2
      IF(.NOT.ITYF) GO TO 26
      RP(I,J)=2*A0*RP(I,J)
      GO TO 25
   26 RP(I,J)=A0*RP(I,J)
   25 CONTINUE
      WRITE(6,626)
      CALL OUTPUT(RP,M1,M2)
  500 FORMAT(2I2)
  650 FORMAT(' ',1X,'THIS IS A REAL PART APPROXIMATION',/)
  600 FORMAT(' ',1X,'NO. OF FREQ POINTS ALONG W1=',I2,2X,'ALONG W2=',I2
     *,/)
  601 FORMAT(' ',1X,'NO. OF SEC-ORDER CASCADES=',I2,2X,'NO. OF FIRST-ORD
     *ER CASCADES=',I2,//)
  602 FORMAT(' ',1X,'INITIAL VALUES OF PARAMETERS:-',/)
  603 FORMAT(' ',1X,'A0=',D23.16,/)
  604 FORMAT(' ',1X,'X(',I2,')=',D23.16,3X,'X(',I2,')=',D23.16)
  605 FORMAT(' ',1X,'DESIRED SPECIFICATIONS:-',/)
  607 FORMAT(' ',1X,'NO. OF FUNCTION EVALUATIONS=',I6,/)
  608 FORMAT(' ',1X,'IER=',I2,/)
  609 FORMAT(5X,'CONVERGENCE NOT OBTAINED IN LIMIT ITERATIONS',/)
  610 FORMAT(5X,'CONVERGENCE OBTAINED IN LIMIT ITERATIONS',/)
  612 FORMAT(5X,'SUM OF THE SQUARED ERRORS BETWEEN DESIRED AND DESIGNED
     * SPECIFICATIONS=',D15.8,//)
  613 FORMAT(' ',1X,'FINAL VALUES OF THE PARAMETERS:-',/)
  606 FORMAT(' ',35X,'NUMERATOR COEFS',20X,'DENOMINATOR COEFS',/)
  614 FORMAT(' ',1X,'THE SECOND ORDER SECTION IS OF THE FORM:-',/)
  615 FORMAT(' ',11X,'A(0)+A(1)*Z2+A(2)*Z2**2+A(3)*Z1+A(4)*Z1*Z2+A(5)*Z1
     ***Z2**2+A(6)*Z1**2+A(7)*Z1**2*Z2+A(8)*Z1**2*Z2**2')
  616 FORMAT(' ',1X,'H(Z1,Z2)=---------------------------------------------
     *-----------------------------------------------------------------')
  617 FORMAT(' ',11X,'B(0)+B(1)*Z2+B(2)*Z2**2+B(3)*Z1+B(4)*Z1*Z2+B(5)*Z1
     ***Z2**2+B(6)*Z1**2+B(7)*Z1**2*Z2+B(8)*Z1**2*Z2**2',/)
  621 FORMAT(' ',1X,'THE FIRST ORDER SECTION IS OF THE FORM:-',/)
  622 FORMAT(' ',11X,'A(0)+A(1)*Z2+A(2)*Z1+A(3)*Z1*Z2')
  623 FORMAT(' ',1X,'H(Z1,Z2)=------------------------------------------')
  624 FORMAT(' ',11X,'B(0)+B(1)*Z2+B(2)*Z1+B(3)*Z1*Z2',/)
  618 FORMAT(' ',//)
  619 FORMAT(' ',32X,'A(',I1,')=',D23.16,15X,'B(',I1,')=',D23.16)
  625 FORMAT(' ',1X,'CASCADE SECTION NO.=',I2,/)
  626 FORMAT(' ',1X,'DESIGNED CHARACTERISTICS:-',/)
      STOP
      END
```

```
      SUBROUTINE FUNCT(N,X,Q,DQ)
C
C     SUBROUTINE THAT CALCULATES THE ERROR FUNCTION VALUE
C     Q AND THE GRADIENTS DQ .  ALSO IT PRINTS OUT THE
C     FUNCTION EVALUATION NUMBER AND ERROR .
C     THE DIMENSIONS OF THE ARRAYS ARE AS FOLLOWS:-
C     U(KS,5),P(10),V(10),HFD(KF),HSD(KS),HFN(KF)
C     HSN(KS),SUM1(3*KF),SUM2(3*KF),SUM3(8*KS),SUM4(10*KS)
C     THE DIMENSION OF THE REMAINING ARRAYS ARE SAME AS
C     IN THE MAIN PROGRAM.  THE ARRAY YD CORRESPONDS TO
C     THE ARRAY ER IN THE MAIN PROGRAM.
C
      DIMENSION X(N),DQ(N),YF(4,6),YS(4,16),U(4,5),P(10),
     *V(10),TA(41),TB(21),TC(41),TD(21),HFD(4),HSD(4),
     *SUM1(12),SUM2(12),SUM3(32),SUM4(40),HFN(4),HSN(4),
     *TE(41,21),TF(41,21),TG(41,21),RP(41,21),YD(41,21)
      REAL*8 X,DQ,Q,YF,YS,U,P,V,SUM1,SUM2,SUM3,SUM4,SUMA,HFNR,HFNI,
     *HFDR,HFDI,HSNR,HSNI,HSDR,HSDI,DREAL,DIMAG,E1,EA,EH,ET1,ET2,SI1,
     *SI2,SI3,SI4,A01,A02
      LOGICAL*1 ITYF
      COMPLEX*16 DCMPLX,PH,HFN,HFD,HSN,HSD,PTF
      COMMON/B1/YS,YF,YD,RP,TA,TB,TC,TD,TE,TF,TG,A0
      COMMON/B2/KOUNT,M1,M2,KS,KF,LP
      COMMON/B3/ITYF
      KFT=6*KF
      KOUNT=KOUNT+1
      IF(KF) 100,100,101
  101 KFC=-6
      DO 1 I=1,KF
      KFC=KFC+6
      DO 1 J=1,3
      YF(I,J)=X(KFC+J)
    1 YF(I,J+3)=X(KFC+J+3)**2
  100 IF(KS) 102,102,103
  103 KSC=KFT-18
      M=-8
      DO 2 I=1,KS
      M=M+16
      KSC=KSC+18
      DO 3 J=1,8
    3 YS(I,J)=X(KSC+J)
      L=KSC+8
      DO 4 K=1,10
      P(K)=X(L+K)
    4 V(K)=X(L+K)**2
      U(I,1)=P(5)*P(10)-P(6)*P(9)+P(7)*P(8)
      U(I,2)=P(1)*P(8)-P(2)*P(6)+P(3)*P(5)
      U(I,3)=P(1)*P(9)-P(2)*P(7)+P(4)*P(5)
      U(I,5)=P(2)*P(10)-P(3)*P(9)+P(4)*P(8)
      U(I,4)=P(1)*P(10)-P(3)*P(7)+P(4)*P(6)
      YS(I,M+1)=U(I,1)**2
      YS(I,M+2)=U(I,2)**2+U(I,3)**2
      YS(I,M+3)=V(5)
```

```
      YS(I,M+4)=U(I,4)**2+U(I,5)**2
      YS(I,M+5)=V(6)+V(7)+V(8)+V(9)
      YS(I,M+6)=V(1)+V(2)
      YS(I,M+7)=V(10)
    2 YS(I,M+8)=V(3)+V(4)
  102 AO=X(N)
      Q=0.D0
      SUMA=0.D0
      IF(KF) 104,104,105
  105 KK1=-3
      DO 5 K=1,KF
      KK1=KK1+3
      DO 5 L=1,3
      SUM1(KK1+L)=0.D0
    5 SUM2(KK1+L)=0.D0
  104 IF(KS) 106,106,107
  107 KK1=-8
      KK2=-10
      DO 6 K=1,KS
      KK1=KK1+8
      KK2=KK2+10
      DO 7 L=1,8
    7 SUM3(KK1+L)=0.D0
      DO 6 J=1,10
    6 SUM4(KK2+J)=0.D0
  106 DO 8 I=1,M1
      DO 8 J=1,M2
      PH=DCMPLX(1.D0,0.D0)
      IF(KF) 108,108,109
  109 DO 9 K=1,KF
      HFNR=YF(K,1)-TE(I,J)
      HFNI=YF(K,2)*TB(J)+YF(K,3)*TA(I)
      HFN(K)=DCMPLX(HFNR,HFNI)
      HFDR=YF(K,4)-TE(I,J)
      HFDI=YF(K,5)*TB(J)+YF(K,6)*TA(I)
      HFD(K)=DCMPLX(HFDR,HFDI)
    9 PH=PH*(HFN(K)/HFD(K))
  108 IF(KS) 110,110,111
  111 DO 10 K=1,KS
      HSNR=YS(K,1)-YS(K,3)*TD(J)-YS(K,5)*TE(I,J)-YS(K,7)*TC(I)
     *+TE(I,J)**2
      HSNI=YS(K,2)*TB(J)+YS(K,4)*TA(I)-YS(K,6)*TG(I,J)-YS(K,8)*TF(I,J)
      HSDR=YS(K,9)-YS(K,11)*TD(J)-YS(K,13)*TE(I,J)-YS(K,15)*TC(I)
     *+TE(I,J)**2
      HSDI=YS(K,10)*TB(J)+YS(K,12)*TA(I)-YS(K,14)*TG(I,J)-YS(K,16)*TF
     *(I,J)
      HSN(K)=DCMPLX(HSNR,HSNI)
      HSD(K)=DCMPLX(HSDR,HSDI)
   10 PH=PH*(HSN(K)/HSD(K))
  110 IF(.NOT.ITYF) GO TO 125
      RP(I,J)=SNGL(DREAL(PH))
      EA=2.*AO*RP(I,J)-YD(I,J)
      E1=EA**(LP-1)
      EH=E1
```

```
      EHA=2.*E1*RP(I,J)
      PTF=PH
      GO TO 126
  125 RP(I,J)=(SNGL(CDABS(PH)))**2
      EA=AO*RP(I,J)-YD(I,J)
      E1=EA**(LP-1)
      EH=E1*RP(I,J)
      PTF=(1.D0,0.0D0)
      EHA=EH
  126 Q=Q+E1*EA
      ET1=EH*TB(J)
      ET2=EH*TA(I)
      IF(KF) 112,112,113
  113 KK=-3
      DO 11 K=1,KF
      KK=KK+3
      SI1=DIMAG(PTF/HFN(K))
      SI2=DIMAG(PTF/HFD(K))
      SUM1(KK+1)=SUM1(KK+1)-EH*DREAL(PTF/HFN(K))
      SUM1(KK+2)=SUM1(KK+2)+ET1*SI1
      SUM1(KK+3)=SUM1(KK+3)+ET2*SI1
      SUM2(KK+1)=SUM2(KK+1)-EH*DREAL(PTF/HFD(K))
      SUM2(KK+2)=SUM2(KK+2)+ET1*SI2
   11 SUM2(KK+3)=SUM2(KK+3)+ET2*SI2
  112 IF(KS) 8,8,115
  115 KJ=-8
      KL=-10
      L=KFT-10
      DO 12 K=1,KS
      SI1=DREAL(PTF/HSN(K))
      SI2=DIMAG(PTF/HSN(K))
      SI3=DREAL(PTF/HSD(K))
      SI4=DIMAG(PTF/HSD(K))
      KJ=KJ+8
      KL=KL+10
      L=L+18
      SUM3(KJ+1)=SUM3(KJ+1)-EH*SI1
      SUM3(KJ+2)=SUM3(KJ+2)+ET1*SI2
      SUM3(KJ+3)=SUM3(KJ+3)+EH*TD(J)*SI1
      SUM3(KJ+4)=SUM3(KJ+4)+ET2*SI2
      SUM3(KJ+5)=SUM3(KJ+5)+EH*TE(I,J)*SI1
      SUM3(KJ+6)=SUM3(KJ+6)-EH*TG(I,J)*SI2
      SUM3(KJ+7)=SUM3(KJ+7)+EH*TC(I)*SI1
      SUM3(KJ+8)=SUM3(KJ+8)-EH*TF(I,J)*SI2
      SUM4(KL+1)=SUM4(KL+1)+EH*(TB(J)*(U(K,2)*X(L+8)+U(K,3)*X(L+9))
     *+TA(I)*U(K,4)*X(L+10)-TG(I,J)*X(L+1))*SI4
      SUM4(KL+2)=SUM4(KL+2)+EH*(-TB(J)*(U(K,2)*X(L+6)+U(K,3)*X(L+7))
     *+TA(I)*U(K,5)*X(L+10)-TG(I,J)*X(L+2))*SI4
      SUM4(KL+3)=SUM4(KL+3)+EH*(TB(J)*U(K,2)*X(L+5)-TA(I)*(U(K,4)*
     *X(L+7)+U(K,5)*X(L+9))-TF(I,J)*X(L+3))*SI4
      SUM4(KL+4)=SUM4(KL+4)+EH*(TB(J)*U(K,3)*X(L+5)+TA(I)*(U(K,4)*
     *X(L+6)+U(K,5)*X(L+8))-TF(I,J)*X(L+4))*SI4
      SUM4(KL+5)=SUM4(KL+5)-EH*DREAL(DCMPLX((U(K,1)*X(L+10)-TD(J)*
     *X(L+5)),(TB(J)*(U(K,2)*X(L+3)+U(K,3)*X(L+4))))/HSD(K))
```

```
      SUM4(KL+6)=SUM4(KL+6)-EH*DREAL(DCMPLX((-U(K,1)*X(L+9)-TE(I,J)*
     *X(L+6)),(-TB(J)*U(K,2)*X(L+2)+TA(I)*U(K,4)*X(L+4))))/HSD(K))
      SUM4(KL+7)=SUM4(KL+7)-EH*DREAL(DCMPLX((U(K,1)*X(L+8)-TE(I,J)*
     *X(L+7)),-(TB(J)*U(K,3)*X(L+2)+TA(I)*U(K,4)*X(L+3))))/HSD(K))
      SUM4(KL+8)=SUM4(KL+8)-EH*DREAL(DCMPLX((U(K,1)*X(L+7)-TE(I,J)*
     *X(L+8)),(TB(J)*U(K,2)*X(L+1)+TA(I)*U(K,5)*X(L+4))))/HSD(K))
      SUM4(KL+9)=SUM4(KL+9)-EH*DREAL(DCMPLX((-U(K,1)*X(L+6)-TE(I,J)*
     *X(L+9)),(TB(J)*U(K,3)*X(L+1)-TA(I)*U(K,5)*X(L+3))))/HSD(K))
   12 SUM4(KL+10)=SUM4(KL+10)-EH*DREAL(DCMPLX((U(K,1)*X(L+5)-TC(I)*
     *X(L+10)),TA(I)*(U(K,4)*X(L+1)+U(K,5)*X(L+2))))/HSD(K))
    8 SUMA=SUMA+EHA
  114 AO1=-2*LP*AO
      AO2=-AO1*2.D0
      IF(KF) 116,116,117
  117 KK=-6
      KKL=-3
      DO 14 K=1,KF
      KKL=KKL+3
      KK=KK+6
      KJ=KK+3
      DO 14 LX=1,3
      DQ(KK+LX)=AO1*SUM1(KKL+LX)
   14 DQ(KJ+LX)=AO2*SUM2(KKL+LX)*X(KJ+LX)
  116 IF(KS) 118,118,119
  119 KJ=KFT-18
      KA=-8
      KB=-10
      DO 15 K=1,KS
      KJ=KJ+18
      KA=KA+8
      KB=KB+10
      DO 16 K1=1,8
   16 DQ(KJ+K1)=AO1*SUM3(KA+K1)
      KC=KJ+8
      DO 15 K2=1,10
   15 DQ(KC+K2)=AO2*SUM4(KB+K2)
  118 DQ(N)=LP*SUMA
      WRITE(6,800) KOUNT,Q
  800 FORMAT(' ',2X,'FUNCTION EVALUATION NO. = ',I6,2X,'ERROR= ',D15.8)
      RETURN
      END
```

Program PICSAMP

The program 'PICSAMP' is used to scan the image line by line. One can actually scan up to 4097 x 4097 rectangular array points in an image. Since the display facility can only display images of size 256 x 256 or less, the images are scanned in such a manner so as to fit into an array of 256 x 256.

Consider an image that is scanned as a rectangular array of points as shown in Figure H1. Figure H1(a) shows the rectangular sampling grid. Figure H1(b) shows the extreme integer co-ordinate values of the largest possible sampling grid. The input data to the D/A has to be less than or equal to 12 bits in length, where one bit is reserved for the sign. The co-ordinates of the sampling grid in Figure H1(a) should fall within that of Figure H1(b). The input parameters and subroutines required are as follows:

Parameters:

NSIZE: # of picture elements desired.

NLINES: # of lines to be sampled.

IXSTEP: step size in x direction.

IYSTEP: step size in y direction.

IXSHIFT: starting x co-ordinate.

IYSHIFT: starting y co-ordinate.

The values for all the above parameters should be chosen such that all the sampling grid points lie within the area of Figure H1(b).

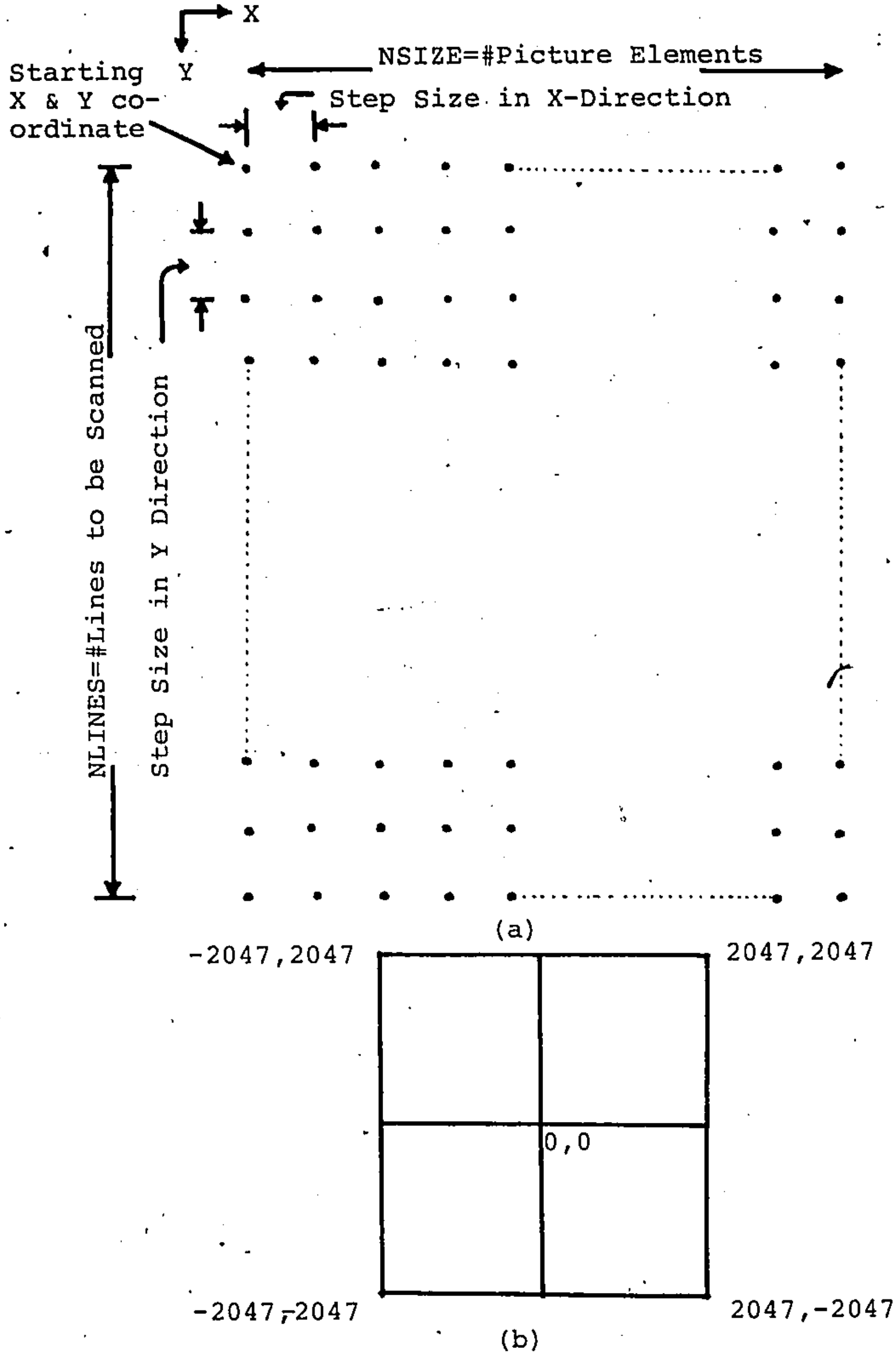


Figure H1 Sampling of Image in a Rectangular Array.

SYINIT: subroutines that initialize the A/D and D/A systems.

PSINIT: subroutine to position the CRT beam to the initial starting point of the sampling grid.

BEGIN: subroutine that waits for operator response to start the image sampling.

DARUNP: subroutine that enables the sampling of image pixel by pixel along each horizontal line and storing it in an array IPIC.

After providing all the parameters, the program sets up the A/D and D/A converter systems and types out the string of characters "***TO START SAMPLING STRIKE ANY KEY***"; then it enters the routine BEGIN and waits for the operator to strike any key on the console. As soon as a key is struck, the program returns from routine BEGIN and executes the rest of the program.

Program PICTEST

The program 'PICTEST' is a test routine which can be used to continuously scan a desired area of the image that is to be sampled. While the image is being continuously scanned, the brightness for the CRT spot is adjusted, such that there is no saturation at the output of the photomultiplier tube or at the output of the integrator. Once this is done, the program 'PICSAMP' can be used to sample that particular area of the image.

In order to stop the program 'PICTEST', it is required to set the NOVA-840 data switches to a value other than -27_{10} (i.e. 100033_8). The description of the input parameters to this

255 minus the intensity values of the original image.

PICROT: Program to rotate the picture through an angle of 90° in a counter clockwise direction.

Image Display Program

CVIDISPI: A program used for displaying the sampled and processed image using the CVI Expander.

Input parameters for this program are:

M2: # of columns of the image to be displayed ($1 \leq M2 \leq 256$).

M1: # of rows ($1 \leq M1 \leq 256$).

IXSHIFT: x co-ordinate of the point from where the display begins.

IYSHIFT: y co-ordinate of the point from where the display begins.

IXSHIFT and IYSHIFT are integer values and they should be subject to the following limits:

$$0 \leq \text{IXSHIFT} \leq 256 \text{ and } \text{IXSHIFT} + M2 \leq 256$$

$$0 \leq \text{IYSHIFT} \leq 256 \text{ and } \text{IYSHIFT} + M1 \leq 256$$

Subroutines required are:

CVSET: sets up the A/D and D/A systems.

CVWRIT: program to transfer data from computer.

BEGIN: same as before.

```
      DIMENSION IRAST(2,512),IPIC(512),NAME(10),INIT(2)
C
C     *********** PICSAMP ***********
C
C     SUBROUTINE FILES REQUIRED:- SYINIT , PSINIT , BEGIN
C     , DARUNP
C
      DATA IZ/'  '/
      DO 10 I=1,10
   10 NAME(I)=IZ
      ACCEPT"# OF PIXELS PER LINE      = ",NSIZE
      ACCEPT"# OF LINES TO BE SAMPLED = ",NLINE
      WRITE(10,500)
  500 FORMAT(' ',1X,'TYPE IN THE FILE NAME FOR IMAGE= ',Z)
      READ(11,600) (NAME(I),I=1,10)
  600 FORMAT(10A2)
      NSIZE1=NSIZE
      NSIZE2=2*NSIZE1
      ACCEPT "STEP SIZE IN X-DIRECTION = ",IXSTEP
      ACCEPT "STEP SIZE IN Y-DIRECTION = ",IYSTEP
      ACCEPT "STARTING X-COORDINATE = ",IXSHIFT
      ACCEPT "STARTING Y-COORDINATE = ",IYSHIFT
      INIT(1)=IXSHIFT
      INIT(2)=IYSHIFT
      IERR=0
      CALL SYINIT(IERR)
      IF(IERR.GT.0) GO TO 50
      NSIZEA=-NSIZE1
      NSIZEB=-NSIZE2
      DO 2 J=2,NSIZE
    2 IRAST(1,J-1)=IXSHIFT+(J-1)*IXSTEP
      IRAST(1,NSIZE)=IXSHIFT
      IYVAL=IYSHIFT
      CALL PSINIT(INIT,IERR)
      IF(IERR.GT.0) GO TO 75
      TYPE" "
      TYPE"*** TO START SAMPLING STRIKE ANY KEY ***"
      TYPE" "
      OPEN 1,NAME,LEN=2*NSIZE,REC=NLINE
      CALL BEGIN
      DO 1 I=1,NLINE
      DO 4 K=1,NSIZE
    4 IRAST(2,K)=IYVAL
      IYVAL=IYVAL-IYSTEP
      IRAST(2,NSIZE)=IYVAL
      CALL DARUNP(IRAST,IPIC,IERR,NSIZEA,NSIZEB,110001K)
      IF(IERR.GT.0) GO TO 200
    1 WRITE(1) (IPIC(NA),NA=1,NSIZE)
      CLOSE 1
      STOP
   50 TYPE"ERROR IN SYINIT: ERROR CODE # = ",IERR
      STOP
   75 TYPE"ERROR IN PSINIT: ERROR CODE # = ",IERR
```

```
      STOP
  200 TYPE"ERROR IN DARUNP: ERROR CODE # = ",IERR
      CLOSE 1
      STOP
      END
```

```
          ;PROGRAM FILE NAME :- SYINIT
          .TITLE SYINIT
          .ENT SYINIT
          .ZREL
SYINIT:   .SYINIT
          .NREL
.SYINIT:        SAVE 0
          LDA 0,ARG0,3
          STA 0,IERR
          SUB 0,0
          STA 0,@IERR
          LDA 0,ADDEV
          .SYSTM
          .DEBL
          JMP .+1
          LDA 0,DADEV
          .SYSTM
          .DEBL
          JMP .+1
          LDA 0,ADDEV
          LDA 1,ADCTAD
          MOVZL 1,1
          MOVOR 1,1          ; SET BIT 0
          LDA 2,BLKA
          .SYSTM
          .IDEF
          JMP ERR1
          LDA 0,DADEV
          LDA 1,ADCTDA
          MOVZL 1,1
          MOVOR 1,1
          LDA 2,BLKB
          .SYSTM
          .IDEF
          JMP ERR1
          RTN
ERR1:     STA 2,@IERR
          RTN
IERR:     0
BLKA:     6
BLKB:     12
ADDEV:    21
DADEV:    23
ADCTAD:   .+2
ADCTDA:   .+4
DCTAD:    .BLK 3
DCTDA:    .BLK 3
          .END
```

```
        ;PROGRAM FILE NAME :- PSINIT
        .TITLE PSINIT
        .ENT PSINIT
        .ZREL
PSINIT: .PSINIT
        .MACRO SETMAP
        LDA 0,^1
        LDA 1,^2
        .SYSTM
        .STMAP
        JMP ERR1
        DOB 1,^3
%
        .NREL
.PSINIT:         SAVE 0
        LDA 0,ARG0,3
        STA 0,IBUF1
        LDA 0,ARG1,3
        STA 0,IERR
        SUB 0,0
        STA 0,@IERR
        LDA 0,DACHAN
        DOAC 0,23
        SETMAP DADEV IBUF1 23
        LDA 0,ADTRIG
        DOAC 0,21
        LDA 0,DACNT
        DOCP 0,23
        SKPDN 23
        JMP .-1
        LDA 0,DADBL
        DOAC 0,23
        LDA 0,ADDBL
        DOAC 0,21
        LDA 0,ADCHAN
        DOA 0,21
        RTN
ERR1:   STA 2,@IERR
        RTN
IERR:   0
ADCHAN: 140000
DACHAN: 120443
ADTRIG: 127102
DACNT:  -1
IBUF1:  0
ADDEV:  21
DADEV:  23
DADBL:  110443
ADDBL:  103002
        .END
```

```
        .TITLE DARUNP
        .ENT DARUNP
        .ZREL
DARUNP: .DARUNP
        .MACRO SETMAP
        LDA 0,^1
        LDA 1,^2
        .SYSTM
        .STMAP
        JMP ERR1
        DOBC 1,^3
%
        .NREL
.DARUNP:        SAVE 0
        LDA 0,ARG0,3
        STA 0,IBUF1
        LDA 0,ARG1,3
        STA 0,IBUF2
        LDA 0,ARG2,3
        STA 0,IERR
        LDA 0,@ARG3,3
        STA 0,ADCNT
        LDA 0,@ARG4,3
        STA 0,DACNT
        LDA 0,@ARG5,3
        STA 0,ADTRIG
        SUB 0,0
        STA 0,@IERR
        .SYSTM
        .ODIS
        JMP .+1
        SETMAP DADEV IBUF1 23
        SETMAP ADDEV IBUF2 21
        LDA 0,ADTRIG
        DOA 0,21
        LDA 0,DACNT
        DOCP 0,23
        LDA 0,ADCNT
        DOCP 0,21
        SKPDN 21
        JMP .-1
        LDA 0,ADDBL
        DOA 0,21
        .SYSTM
        .OEBL
        JMP .+1
        RTN
ERR1:   STA 2,@IERR
        RTN
IERR:   0
ADTRIG: 0
ADCNT:  0
DACNT:  0
IBUF1:  0
IBUF2:  0
ADDEV:  21
DADEV:  23
ADDBL:  103002
        .END
```

```
        .TITLE BEGIN
        .ENT BEGIN
        .ZREL
BEGIN:  .BEGIN
        .NREL
.BEGIN: SAVE 0
        .SYSTM
        .GCHAR
        JMP .+1
        RTN
        .END
```

```
      DIMENSION IRAST(2,512),IPIC(512),INIT(2)
C
C     *********** PICTEST ***********
C
C     SUBROUTINE FILES REQUIRED:- SYINIT , PSINIT ,
C     BEGIN , DARUNP
C
      ITEST=100033K
      ACCEPT"# OF PIXELS PER LINE      = ",NSIZE
      ACCEPT"# OF LINES TO BE SAMPLED = ",NLINE
      NSIZE1=NSIZE
      NSIZE2=2*NSIZE1
      ACCEPT "STEP SIZE IN X-DIRECTION = ",IXSTEP
      ACCEPT "STEP SIZE IN Y-DIRECTION = ",IYSTEP
      ACCEPT "STARTING X-COORDINATE = ",IXSHIFT
      ACCEPT "STARTING Y-COORDINATE = ",IYSHIFT
      INIT(1)=IXSHIFT
      INIT(2)=IYSHIFT
      IERR=0
      CALL SYINIT(IERR)
      IF(IERR.GT.0) GO TO 50
      NSIZEA=-NSIZE1
      NSIZEB=-NSIZE2
      DO 2 J=2,NSIZE
    2 IRAST(1,J-1)=IXSHIFT+(J-1)*IXSTEP
      IRAST(1,NSIZE)=IXSHIFT
      TYPE" "
      TYPE"*** TO START TESTING STRIKE ANY KEY ***"
      TYPE" "
      CALL BEGIN
    3 IYVAL=IYSHIFT
      CALL PSINIT(INIT,IERR)
      IF(IERR.GT.0) GO TO 75
      DO 1 I=1,NLINE
      DO 4 K=1,NSIZE
    4 IRAST(2,K)=IYVAL
      IYVAL=IYVAL-IYSTEP
      IRAST(2,NSIZE)=IYVAL
      CALL DARUNP(IRAST,IPIC,IERR,NSIZEA,NSIZEB,110001K)
      IF(IERR.GT.0) GO TO 200
      CALL OUT(IVAL)
      IF(IVAL.NE.ITEST) STOP
    1 CONTINUE
      GO TO 3
   50 TYPE"ERROR IN SYINIT: ERROR CODE # = ",IERR
      STOP
   75 TYPE"ERROR IN PSINIT: ERROR CODE # = ",IERR
      STOP
  200 TYPE"ERROR IN DARUNP: ERROR CODE # = ",IERR
      STOP
      END
```

```
        ;PROGRAM FILE :- OUT
        .TITLE OUT
        .ENT OUT
        .ZREL
OUT:    .OUT
        .NREL
.OUT:   SAVE 0
        LDA 0,ARG0,3
        STA 0,IVAL
        SUB 0,0
        STA 0,@IVAL
        .SYSTM
        .RDSW
        JMP .+1
        STA 0,@IVAL
        RTN
IVAL:   0
        .END
```

```
      DIMENSION IWIND(16384),IX(64,256),NAME(5)
C
C     *********** NORMPIC ***********
C
C     IMAGE NORMALIZATION
C
      COMMON/B1/IWIND
      EQUIVALENCE (IWIND,IX)
      CALL VMEM(ICNT,IER)
      IF(IER.EQ.5) GO TO 3000
      TYPE"# OF FREE 1024-WORD BLOCKS= ",ICNT
      CALL MAPDF(ICNT,IWIND,16,IER)
      IF(IER.GE.5) GO TO 3001
      WRITE(10,100)
  100 FORMAT(' ',1X,'FILE NAME OF IMAGE= ',Z)
      READ(11,200) (NAME(I),I=1,5)
  200 FORMAT(5A2)
      ACCEPT "# OF COLOUMNS OF IMAGE = ",M2
      ACCEPT"# OF       ROWS OF IMAGE = ",M1
      ACCEPT"INTENSITY FACTOR = ",FACT
      TYPE"TO START ----- TYPE ANY KEY"
      CALL BEGIN
      OPEN 1,NAME,LEN=2*M2,REC=M1
      M1X=M1/64
      IF(M1X.EQ.0) GO TO 20
      MREM=M1-M1X*64
      IF(MREM.GT.0) GO TO 21
      MREM=64
      GO TO 22
   21 M1X=M1X+1
      GO TO 22
   20 M1X=1
      MREM=M1
   22 KB=-16
```

```
      M1A=64
      DO 23 I=1,M1X
      KB=KB+16
      CALL REMAP(0,KB,16,IER)
      IF(IER.GE.29) GO TO 3002
      IF(I.EQ.M1X) M1A=MREM
      DO 23 J=1,M1A
   23 READ(1) (IX(J,K),K=1,M2)
      REWIND 1
      KB=-16
      M1A=64
      DO 24 I=1,M1X
      KB=KB+16
      CALL REMAP(0,KB,16,IER)
      IF(IER.GE.29) GO TO 3002
      IF(I.EQ.M1X) M1A=MREM
      DO 25 J=1,M1A
      DO 25 K=1,M2
   25 IX(J,K)=((FLOAT(IX(J,K))+8192.0)/64.0)*FACT
      DO 24 L=1,M1A
   24 WRITE(1) (IX(L,N),N=1,M2)
      GO TO 300
 3000 TYPE"ERROR IN VMEM ; ERROR # = ",IER
      GO TO 400
 3001 TYPE"ERROR IN MAPDF ; ERROR # = ",IER
      GO TO 400
 3002 TYPE"ERROR IN REMAP ; ERROR # = ",IER
  300 CLOSE 1
  400 STOP
      END

      DIMENSION IWIND(16384),IX(64,256),NAME(5)
C
C     *********** INVERTPIC ***********
C
C     INVERTING A NORMALIZED IMAGE
C
      COMMON/B1/IWIND
      EQUIVALENCE (IWIND,IX)
      CALL VMEM(ICNT,IER)
      IF(IER.EQ.5) GO TO 3000
      TYPE"# OF FREE 1024-WORD BLOCKS= ",ICNT
      CALL MAPDF(ICNT,IWIND,16,IER)
      IF(IER.GE.5) GO TO 3001
      WRITE(10,100)
  100 FORMAT(' ',1X,'FILE NAME OF IMAGE= ',Z)
      READ(11,200) (NAME(I),I=1,5)
  200 FORMAT(5A2)
      ACCEPT "# OF COLOUMNS OF IMAGE = ",M2
      ACCEPT"# OF       ROWS OF IMAGE = ",M1
      TYPE"TO START -- TYPE ANY KEY "
      CALL BEGIN
      OPEN 1,NAME,LEN=2*M2,REC=M1
```

```
      M1X=M1/64
      IF(M1X.EQ.0) GO TO 20
      MREM=M1-M1X*64
      IF(MREM.GT.0) GO TO 21
      MREM=64
      GO TO 22
   21 M1X=M1X+1
      GO TO 22
   20 M1X=1
      MREM=M1
   22 KB=-16
      M1A=64
      DO 23 I=1,M1X
      KB=KB+16
      CALL REMAP(0,KB,16,IER)
      IF(IER.GE.29) GO TO 3002
      IF(I.EQ.M1X) M1A=MREM
      DO 23 J=1,M1A
   23 READ(1) (IX(J,K),K=1,M2)
      REWIND 1
      KB=-16
      M1A=64
      DO 24 I=1,M1X
      KB=KB+16
      CALL REMAP(0,KB,16,IER)
      IF(IER.GE.29) GO TO 3002
      IF(I.EQ.M1X) M1A=MREM
      DO 25 J=1,M1A
      DO 25 K=1,M2
   25 IX(J,K)=255.0-FLOAT(IX(J,K))
      DO 24 L=1,M1A
   24 WRITE(1) (IX(L,N),N=1,M2)
      GO TO 300
 3000 TYPE"ERROR IN VMEM ; ERROR # = ",IER
      GO TO 400
 3001 TYPE"ERROR IN MAPDF ; ERROR # = ",IER
      GO TO 400
 3002 TYPE"ERROR IN REMAP ; ERROR # = ",IER
  300 CLOSE 1
  400 STOP
      END

      DIMENSION IWIND(16384),IY(256,64),IA(32,256),NAME(5)
C
C     ******************** PICROT ********************
C
C     PROGRAM TO ROTATE THE IMAGE
C
      COMMON/B1/IWIND
      EQUIVALENCE(IWIND,IY)
      CALL VMEM(ICNT,IER)
      IF(IER.EQ.5) GO TO 1000
      TYPE"# OF FREE 1024-WORD BLOCKS= ",ICNT
```

```
      ICNT=ICNT-1
      CALL MAPDF(ICNT,IWIND,16,IER)
      IF(IER.GE.5) GO TO 1001
      TYPE"SIZE OF IMAGE TO BE ROTATED  ( > 64X64 ) "
      ACCEPT"# OF COLOUMNS ( POWER OF 2) = ",M2
      ACCEPT"# OF       ROWS ( POWER OF 2) = ",M1
      WRITE(10,100)
  100 FORMAT(' ',1X,'FILE NAME OF IMAGE= ',Z)
      READ(11,200) (NAME(I),I=1,5)
  200 FORMAT(5A2)
      OPEN 1,NAME,LEN=2*M2,REC=M1
      M21=M2+1
      IR1=64
      IR2=32
      IT1=M1/IR1
      IT2=M2/IR2
      IB=16
      KK=-IB
      DO 1 K=1,IT1
      KK=KK+IB
      CALL REMAP(0,KK,IB,IER)
      IF(IER.GE.29) GO TO 1002
      DO 1 I=1,IR1
    1 READ(1) (IY(M21-J,I),J=1,M2)
      REWIND 1
      KK1=-IR2
      DO 2 K=1,IT2
      KK1=KK1+IR2
      KK2=-IR1
      KK=-IB
      DO 3 L=1,IT1
      KK=KK+IB
      KK2=KK2+IR1
      CALL REMAP(0,KK,IB,IER)
      IF(IER.GE.29) GO TO 1002
      DO 3 I=1,IR2
      DO 3 J=1,IR1
    3 IA(I,KK2+J)=IY(KK1+I,J)
      DO 2 IM=1,IR2
    2 WRITE(1) (IA(IM,IK),IK=1,M1)
      CLOSE 1
      STOP
 1000 TYPE"ERROR IN VMEM ;  ERROR # = ",IER
      CLOSE 1
      STOP
 1001 TYPE"ERROR IN MAPDF ;  ERROR # = ",IER
      CLOSE 1
      STOP
 1002 TYPE"ERROR IN REMAP ;  ERROR # = ",IER
      CLOSE 1
      STOP
      END
```

```
      DIMENSION IXYZ(3,4096),IFILE(5)
C
C     *************** CVDISPI ****************
C
C     PROGRAM TO WRITE PICTURE DATA ON CVI(LINE BY LINE)
C
C     SUBROUTINES REQUIRED:- CVSET , CVWRIT , BEGIN
C
      ACCEPT"SAMPLING RATE= ",ISAMP1
      ISAMP=ISAMP1
      IERR=0
      CALL CVSET(0,0,IERR)
      IF(IERR.GT.0) GO TO 150
500   ACCEPT"# OF COLOUMNS TO BE DISPLAYED= ",M2
      ACCEPT"# OF      ROWS TO BE DISPLAYED= ",M1
      IWZ=0
345   ACCEPT"STARTING X -.CO-ORDINATE= ",IXSHIFT
      ACCEPT"STARTING Y - CO-ORDINATE= ",IYSHIFT
      NSL=(M1/16)+1
      NREM=(NSL*16)-M1
      IF(NREM.EQ.16) NSL=NSL-1
      IF(NREM.EQ.16) NREM=0
      IF(IWZ.EQ.1) GO TO 400
      ACCEPT"WISH TO CLEAR(0),OR DISPLAY(1) ON THE
     *SCREEN?->",ID
      IF(ID.EQ.0) GO TO 300
400   WRITE(10,100)
100   FORMAT(' ',"INPUT FILE NAME:--",Z)
      READ(11,200)(IFILE(I),I=1,5)
200   FORMAT(5A2)
      ID=1
      OPEN 1,IFILE,LEN=2*M2,REC=M1
  300 CONTINUE
      NST=16
      TYPE" "
      TYPE"TO START DISPLAYING TYPE ANY CHARACTER"
      TYPE" "
      CALL BEGIN
      DO 30 K1=1,NSL
      IF(K1.EQ.NSL) NST=16-NREM
      DO 25 K2=1,NST
      ICON=(IYSHIFT+(K1-1)*16+(K2-1))*2+(141000K)
      J2=(K2-1)*M2
      IF(ID.EQ.0) GO TO 320
      READ(1) (IXYZ(3,KZ+J2),KZ=1,M2)
      GO TO 330
320   DO 340 KQ=1,M2
340   IXYZ(3,KQ+J2)=0
330   CONTINUE
      DO 25 J=1,M2
      J1=J2+J
      IXYZ(2,J1)=ICON
      IXYZ(1,J1)=IXSHIFT+J-1+(142000K)
```

```
      IXYZ(3,J1)=IXYZ(3,J1)+(103000K)
25    CONTINUE
      MCOUNT=-3*M2*NST
      IERR=0
      CALL CVWRIT(IXYZ,ISAMP,MCOUNT,IERR)
      IF(IERR.GT.0) GO TO 160
30    CONTINUE
      TYPE"WISH TO DISPLAY ANOTHER IMAGE HAVING"
      TYPE"THE SAME NUMBER OF COLOUMNS AND ROWS"
      ACCEPT"FROM THE SAME STARTING POINT? YES(1),NO(0)
     *----->",IWI
      IF(ID.EQ.0) GO TO 310
      CLOSE 1
310   IF(IWI.NE.0) GO TO 400
      ACCEPT"ERASE LAST DISPLAYED IMAGE? YES(1),NO(0)
     *------->",IZR
      IF(IZR.EQ.1) ID=0
      IF(IZR.EQ.1) GO TO 300
      TYPE"WISH TO CHANGE X AND Y STARTING CO-ORDINATES"
      TYPE"AND DISPLAY SAME IMAGE OR ANOTHER IMAGE   OF"
      ACCEPT"SAME # OF COLOUMNS AND ROWS? YES(1),NO(0)
     *------>",IWZ
      IF(IWZ.EQ.1) GO TO 345
      ACCEPT"WISH TO START FROM THE BEGINING? YES(1),
     *NO(0)->",IW2
      IF(IW2.EQ.1) GO TO 500
      STOP
150   TYPE"ERROR IN CVSET: CODE=",IERR
      CLOSE 1
      STOP
160   TYPE"ERROR IN CVWRIT: CODE=",IERR
      CLOSE 1
      STOP
      END
```

```
        .TITL CVWRIT
        .ENT CVWRIT
        .ZREL
CVWRIT: .CVWRIT
        .NREL
.CVWRIT:        SAVE 0
        LDA 0,ARG0,3
        STA 0,IXYZ
        LDA 0,@ARG1,3
        STA 0,ISAMP
        LDA 0,@ARG2,3
        STA 0,MCOUNT
        LDA 0,ARG3,3
        STA 0,IERR
        LDA 0,DADEV
        LDA 1,IXYZ
        .SYSTM
```

```
        .TITL CVSET
        .ENT CVSET
        .ZREL
CVSET:  .CVSET
        .NREL
.CVSET: SAVE 0
        LDA 0,ADCHAN
        LDA 1,@ARG0,3
        ADDZ 1,0
        STA 0,ADCHAN
        LDA 0,DACHAN
        LDA 1,@ARG1,3
        ADDZ 1,0
        STA 0,DACHAN
        LDA 0,ARG2,3
        STA 0,IERR
        SUB 0,0
```

```
         STA 0,@IERR
         LDA 0,ADDEV
         .SYSTM
         .DEBL
         JMP .+1
         LDA 0,DADEV
         .SYSTM
         .DEBL
         JMP .+1
         LDA 0,ADCHAN
         DOAC 0,21
         LDA 0,ADTRIG
         DOA 0,21
         LDA 0,DACHAN
         DOAC 0,23
         LDA 0,ADDEV
         LDA 1,ADCTAD
         MOVZL 1,1
         MOVOR 1,1
         LDA 2,BLK1
         .SYSTM
         .IDEF
         JMP ERR1
         LDA 0,DADEV
         LDA 1,ADCTDA
         MOVZL 1,1
         MOVOR 1,1
         LDA 2,BLKNUM
         .SYSTM
         .IDEF
         STA 2,@IERR
         RTN
ERR1:    STA 2,@IERR
         RTN
ADCHAN:  140000
ADTRIG:  123000
DACHAN:  136001
IERR:    0
BLKNUM:  15
BLK1:    5
ADDEV:   21
DADEV:   23
ADCTAD:  .+2
ADCTDA:  .+4
DCTAD:   .BLK 3
DCTDA:   .BLK 3
         .END
```

```
         .STMAP
         JMP ERR
         DOB 1,23
         LDA 0,ADDEV
         LDA 1,ATEMP
         .SYSTM
         .STMAP
         JMP ERR
         DOB 1,21
         LDA 0,MCOUNT
         DOCC 0,23
         LDA 0,ADCOUNT
         DOCC 0,21
         LDA 0,ISAMP
         DOAP 0,23
         DOAP 0,21
         SKPDN 23
         JMP .-1
         RTN
ERR:     STA 2,@IERR
         RTN
IXYZ:    0
ISAMP:   0
MCOUNT:  0
IERR:    0
ADDEV:   21
DADEV:   23
ATEMP:   .+2
ADCOUNT:          -2
TEMP:    .BLK 2
         .END
```

```
      DIMENSION IWIND(4096),IX(64,64),PS(32,64)
C
C     ***************** PSESTOLW **********************
C
C     ESTIMATION OF THE AVERAGE POWER SPECTRUM OF
C     A GIVEN IMAGE OVER A GIVEN SEGMENT SIZE
C     SEGMENTS CAN BE OVERLAPED AND WINDOWED
C     IF DESIRED . THE SIZE OF THE SEGMENT IS
C     N1= 2**N .  N IS TO BE LESS OR EQUAL TO 6.  THE
C     INPUT IMAGE SIZE IS = M2 X M1 .  M2 AND M1
C     ARE TO BE POWERS OF 2 AND SHOULD BE LESS OR
C     EQUAL TO 256 .  THE ESTIMATED POWER SPECTRUM IS
C     WRITTEN INTO A NEW FILE .
C
C
      DIMENSION X(64,64),IY(64,256),NAME(5),WD(32,64)
      COMPLEX X,CMPLX
      COMMON/B1/IWIND
      COMMON/BLK/X
      EQUIVALENCE (IWIND,IX,PS,WD)
      EQUIVALENCE (X,IY)
      CALL VMEM(ICNT,IER)
      IF(IER.EQ.5) GO TO 1000
      TYPE"# OF FREE 1024-WORD BLOCKS = ",ICNT
      CALL MAPDF(ICNT,IWIND,4,2,IER)
      IF(IER.GE.5) GO TO 1001
      ACCEPT"FFT SIZE    = 2**N ; N= ",N
      N1=2**N
      NA1=N1/2
      N1N1=N1*N1
      TYPE"SIZE OF THE INPUT DATA ( > OR = FFT SIZE ) "
      ACCEPT"# OF COLOUMNS ( POWER OF 2) = ",M2
      ACCEPT"# OF      ROWS ( POWER OF 2) = ",M1
      KINC=4
      M2A=M2/N1
      M1A=M1/N1
C
C     IF THE D.C ( OR MEAN ) VALUE OF THE IMAGE IS TO BE
C     REMOVED THEN SET MDC EQUAL TO 1 AND READ IN THE
C     D.C VALUE
C
      TYPE" "
      ACCEPT"OVERLAP THE SEGMENTS? YES(1),NO(0) ------>",IOL
      ACCEPT"WINDOW THE SEGMENTS? YES(1),NO(0) ------->",IWD
      ACCEPT"ORIGIN OF P.S AT CENTRE? YES(1),NO(0) --->",IMD
      ACCEPT"REMOVE D.C VALUE OF IMAGE? YES(1),NO(0) ->",MDC
      IF(MDC.EQ.1) ACCEPT"D.C COMPONENT VALUE= ",DCV
      IF(IOL.EQ.0) GO TO 201
      KINC=2
      M2A=(M2/N1)*2-1
      M1A=(M1/N1)*2-1
      M2A1=M2/N1
```

```
  201 TYPE "<15>"
      TYPE"# OF SEGMENTS IN HORZ DIRECTION= ",M2A
      TYPE"# OF SEGMENTS IN VERT DIRECTION= ",M1A
      TYPE "<15>"
      KB1=12
      DO 9 L1=1,2
      KB1=KB1+4
      CALL REMAP(0,KB1,4,IER)
      IF(IER.GE.29) GO TO 1002
      DO 9 K=1,NA1
      DO 9 L=1,N1
    9 PS(K,L)=0.0
      WRITE(10,100)
  100 FORMAT(' ',3X,'INPUT DATA FILE NAME= ',Z)
      READ(11,200) (NAME(I),I=1,5)
  200 FORMAT(5A2)
      OPEN 1,NAME,LEN=2*M2,REC=M1
      IF(IWD.EQ.0) GO TO 202
      WRITE(10,104)
  104 FORMAT(' ',3X,'FILE  NAME OF WINDOW= ',Z)
      READ(11,200) (NAME(I),I=1,5)
      OPEN 3,NAME,LEN=4*N1,REC=N1
      KB2=20
C
C     READING IN 2D-WINDOW VALUES IF IWD=1
C
      DO 11 L2=1,2
      KB2=KB2+4
      CALL REMAP(0,KB2,4,IER)
      IF(IER.GE.29) GO TO 1002
      DO 11 KA=1,NA1
   11 READ(3) (WD(KA,KZ),KZ=1,N1)
      CLOSE 3
  202 WRITE(10,101)
  101 FORMAT(' ',3X,'FILE NAME FOR OUTPUT= ',Z)
C
C     READ IN FILE NAME TO STORE THE ESTIMATED POWER
C     SPECTRUM VALUES .
C
      READ(11,200) (NAME(I),I=1,5)
      SUM=0.0
      IF(MDC.EQ.1) SUM=DCV
      CALL MAPDF(1,IER)
      IF(IER.GE.5) GO TO 1001
      NXT=N1
      NIT=0
      DO 1 I=1,M1A
      DO 2 J=1,NXT
C
C     READING IMAGE DATA OF BLOCK SIZE N1 X M2
C
    2 READ(1) (IY(NIT+J,K),K=1,M2)
      IF(I.EQ.1.OR.IOL.EQ.0) GO TO 222
      KB=-4
```

```
      KS=-N1
      DO 14 J3=1,M2A1
      KB=KB+4
      KS=KS+N1
      CALL REMAP(0,KB,4,IER)
      IF(IER.GE.29) GO TO 1002
      DO 14 K=1,NA1
      DO 14 L=1,N1
   14 IY(K,KS+L)=IX(NA1+K,L)
  223 KB=-4
      KS=-N1
C
C     STORE PART OF INPUT DATA FOR OVERLAPPING THE
C     DATA FOR NEXT ESTIMATION.
C
      DO 3 J1=1,M2A1
      KB=KB+4
      KS=KS+N1
      CALL REMAP(0,KB,4,IER)
      IF(IER.GE.29) GO TO 1002
      DO 3 K=1,N1
      DO 3 L=1,N1
    3 IX(K,L)=IY(K,KS+L)
      KB=0
      DO 8 J2=1,M2A
      WRITE(10,102) I,J2
  102 FORMAT(/,' ',4X,'COMPUTING POWER SPECTRUM OF
     * SEGMENT = (',I2,',',I2,')')
      CALL REMAP(0,KB,4,IER)
      IF(IER.GE.29) GO TO 1002
      IF(IMD.EQ.1) GO TO 222
C
C     SUBTRACTING THE D.C VALUE FROM THE DATA
C
      DO 5 K=1,N1
      DO 5 L=1,N1
      XXA=IX(K,L)-SUM
    5 X(K,L)=CMPLX(XXA,0.0)
      GO TO 333
  222 ICA=-1
C
C     SHIFTING THE ORGIN OF THE POWER SPECTRUM
C     TO THE CENTRE BY MULTIPLYING THE INPUT
C     DATA BY -1.
C
      DO 50 K=1,N1
      ICA=ICA*(-1)
      ICB=-1
      DO 50 L=1,N1
      ICB=ICB*(-1)
      ICC=ICB*ICA
      XXA=(IX(K,L)-SUM)*ICC
   50 X(K,L)=CMPLX(XXA,0.0)
  333 CALL MAPDF(2,IER)
```

```
      IF(IER.GE.5) GO TO 1001
      IF(IND.EQ.0), GO TO 204
      KS2=-NA1
      KB2=20
C
C     MULTIPLYING THE DATA BY THE WINDOW VALUES
C
      DO 12 L3=1,2
      KS2=KS2+NA1
      KB2=KB2+4
      CALL REMAP(0,KB2,4,IER)
      IF(IER.GE.29) GO TO 1002
      DO 12 KE=1,NA1
      DO 12 KU=1,N1
   12 X(KS2+KE,KU)=X(KS2+KE,KU)*WD(KE,KU)
C
C     COMPUTING THE TWO - DIMENSIONAL FFT
C
  204 CALL FFTP2(N,N,N,N,-1.,N1,N1)
      KB1=12
      KS1=-NA1
C
C     SUMMING THE POWER SPECTRUM OF SEGMENTS
C
      DO 4 L1=1,2
      KS1=KS1+NA1
      KB1=KB1+4
      CALL REMAP(0,KB1,4,IER)
      IF(IER.GE.29) GO TO 1002
      DO 4 K=1,NA1
      DO 4 L=1,N1
      XX=CABS(X(KS1+K,L))
    4 PS(K,L)=PS(K,L)+(XX*XX)/N1N1
      CALL MAPDF(1,IER)
      IF(IER.GE.5) GO TO 1001
    8 KB=KB+KINC
      IF(IOL.EQ.0) GO TO 1
      NXT=NA1
      NIT=NA1
    1 CONTINUE
      MC=M2A*M1A
      CALL MAPDF(2,IER)
      IF(IER.GE.5) GO TO 1001
      OPEN 2,NAME,LEN=4*N1,REC=N1
      KB1=12
      WRITE(10,103)
  103 FORMAT(/,' ',4X,'COMPUTING AVERAGE POWER
     * SPECTRUM',/)
      DO 6 L1=1,2
      KB1=KB1+4
      CALL REMAP(0,KB1,4,IER)
      IF(IER.GE.29) GO TO 1002
C
C     COMPUTING THE AVERAGE POWER SPECTRUM
```

```
C
      DO 7 K=1,NA1
      DO 7 L=1,N1
    7 PS(K,L)=PS(K,L)/MC
      DO 6 K1=1,NA1
    6 WRITE(2) (PS(K1,K2),K2=1,N1)
      CLOSE 2
      CLOSE 1
      STOP
 1000 TYPE"ERROR IN VMEM ; ERROR # = ",IER
      GO TO 500
 1001 TYPE"ERROR IN MAPDF ; ERROR # = ",IER
      GO TO 500
 1002 TYPE"ERROR IN REMAP ; ERROR # = ",IER
      CLOSE 2
  500 CLOSE 1
      STOP
      END
```

```
      DIMENSION IWINDOW(2048),IT(32,64),IY(32,64),X1(64,8)
C
C     ****************** OSFFT ******************
C
C     FILTERING DATA USING THE OVER-LAP SAVE METHOD OF FFT
C
C     SUBROUTINE REQUIRED:- TDFFT
C
C     THE INPUT IMAGE SIZE = 2**M X 2**M .  AND M IS TO BE
C     LESS OR EQUAL TO 8.  THE FFT ARRAY SIZE IS EQUAL TO
C     2**N X 2**N ., AND N IS TO BE LESS OR EQUAL TO 6 .
C     LET M1=2**N .  THEN THE BLOCK SIZE OF IMAGE THAT IS
C     FILTERED AT A TIME IS = M1 X 2*M1 . THE BLOCKS ARE
C     OVERLAPPED .  THE FILTERED IMAGE IS WRITTEN BACK
C     TO THE ORIGINAL  IMAGE FILE
C
      DIMENSION NAME(5),X(64,64),ID(32,256),IE(32,64)
      DIMENSION IY1(32,32),IT1(32,32)
      COMPLEX X1,X
      COMMON/WINDOW/IWINDOW
      COMMON/BLK/X
      EQUIVALENCE(IWINDOW,IT,IY,X1)
      EQUIVALENCE(X,ID,IE)
      CALL VMEM(ICNT,IER)
      IF(IER.EQ.5) GO TO 1000
      TYPE"# OF FREE BLOCKS OF 1024-WORDS= ",ICNT
      CALL MAPDF(ICNT,IWINDOW,2,IER)
      IER1=1
      IF(IER.GT.5) GO TO 1001
      WRITE(10,660)
  660 FORMAT(' ',1X,'FILE NAME OF BLUR/DEBLUR T.F. = ',Z)
      READ(11,200) (NAME(I),I=1,5)
  200 FORMAT(5A2)
      ACCEPT"SIZE OF T.F=SIZE OF FFT=2**N N= ",N
      M1=2**N
      OPEN 1,NAME,LEN=8*M1,REC=M1
      DO 20 I=1,M1
   20 READ(1) (X(I,J),J=1,M1)
      CLOSE 1
      CALL MAPDF(4,IER)
      IER1=2
      IF(IER.GE.5) GO TO 1001
      KS=-8
      IER2=8
      DO 25 KA=1,8
      KBK=(KA-1)*2
      CALL REMAP(0,KBK,2,IER)
      IF(IER.GE.29) GO TO 1003
      KS=KS+8
      DO 25 KB=1,8
      KSA=KS+KB
      DO 25 KC=1,M1
```

VERY POOR COPY

```
   25 X1(KC,KB)=X(KSA,KC)
      WRITE(10,101)
  101 FORMAT(' ',1X,'FILE NAME OF IMAGE=',Z)
      READ(11,200) (NAME(I),I=1,5)
      TYPE"SIZE OF IMAGE"
      ACCEPT"# OF COLOUMNS=ROWS=2**M M= ",M
      NC=2**M
      OPEN 2,NAME,LEN=2*NC,REC=NC
      NC1=(2*NC)/M1
      M2=M1/2
      MTN=M2*NC
      NT=NC1/2
      DO 1 I=1,NC1
      IREC=(I-1)*M2+1
      CALL FSEEK(2,IREC)
C
C     READING OUT AN IMAGE BLOCK OF SIZE NC X M2
C
      DO 7 I1=1,M2
    7 READ(2) (ID(I1,I2),I2=1,NC)
      CALL MAPDF(1,IER)
      IER1=5
      IF(IER.GE.5) GO TO 1001
      MIN=-M1
      KBL=14
      DO 400 IU=1,NT
      KBL=KBL+2
      MIN=MIN+M1
      CALL REMAP(0,KBL,2,IER)
      IER2=0
      IF(IER.GE.29) GO TO 1003
      DO 400 IV=1,M2
      DO 400 IW=1,M1
  400 IT(IV,IW)=ID(IV,MIN+IW)
      IJ=16
      IK=24
C
C     FILTERING OVERLAPPED IMAGE BLOCKS OF
C     SIZE EQUAL TO M1 X 2*M1
C
      DO 100 J=1,NT
      IBL1=(J-1)*2+1
      IBL2=IBL1+1
      WRITE(10,555) I,IBL1,I,IBL2
  555 FORMAT(' ',1X,'PROCESSING BLOCKS (',I2,',',I2,')',
     *' & ','(',I2,',',I2,')',/)
      CALL REMAP(0,IJ,2,IER)
      IER2=1
      IF(IER.GE.29) GO TO 1003
      IF(I.EQ.1) GO TO 300
      IF(J.EQ.1) GO TO 310
      GO TO 320
  300 DO 2 K1=1,M2
      DO 2 K2=1,M1
```

```
    2 X(K1,K2)=(0.,0.)
      IF(J.EQ.1) GO TO 310
  320 DO 4 K1=1,M2
      DO 4 K2=1,M2
      X(M2+K1,K2)=CMPLX(FLOAT(IT1(K1,K2)),FLOAT(IT(K1,
     *K2)))
    4 X(M2+K1,M2+K2)=CMPLX(FLOAT(IT(K1,K2)),FLOAT(IT(
     *K1,M2+K2)))
      IF(I.NE.1) GO TO 330
      GO TO 10
  310 DO 3 K1=1,M2
      DO 3 K2=1,M2
      X(M2+K1,K2)=CMPLX(0.,FLOAT(IT(K1,K2)))
    3 X(M2+K1,M2+K2)=CMPLX(FLOAT(IT(K1,K2)),FLOAT(IT(
     *K1,M2+K2)))
      IF(I.EQ.1) GO TO 10
      CALL REMAP(0,IK,2,IER)
      IER2=2
      IF(IER.GE.29) GO TO 1003
      DO 5 K1=1,M2
      DO 5 K2=1,M2
      X(K1,K2)=CMPLX(0.,FLOAT(IY(K1,K2)))
    5 X(K1,M2+K2)=CMPLX(FLOAT(IY(K1,K2)),FLOAT(IY(K1,
     *M2+K2)))
      GO TO 10
  330 CALL REMAP(0,IK,2,IER)
      IER2=3
      IF(IER.GE.29) GO TO 1003
      DO 6 K1=1,M2
      DO 6 K2=1,M2
      X(K1,K2)=CMPLX(FLOAT(IY1(K1,K2)),FLOAT(IY(K1,K2)))
    6 X(K1,M2+K2)=CMPLX(FLOAT(IY(K1,K2)),FLOAT(IY(K1,
     *M2+K2)))
C
C     COMPUTE FFT OF OVERLAPPED IMAGE BLOCKS
C
   10 CALL FFTP2(6,6,6,6,-1.,M1,M1)
      CALL MAPDF(4,IER)
      IER1=3
      IF(IER.GE.5) GO TO 1001
      KS=-8
C
C     FREQUENCY DOMAIN MULTIPLICATION OF THE FFT OF
C     IMAGE WITH THE TRANSFER FUNCTION OF FILTER
C
      DO 11 KA=1,8
      KBK=(KA-1)*2
      CALL REMAP(0,KBK,2,IER)
      IER2=4
      IF(IER.GE.29) GO TO 1003
      KS=KS+8
      DO 11 KB=1,8
      KSA=KS+KB
      DO 11 KC=1,M1
```

```
   11 X(KSA,KC)=(X(KSA,KC)*X1(KC,KB))/4096.0
C
C     COMPUTE THE INVERSE FFT OF THE PRODUCT OF
C     FFT OF IMAGE BLOCK AND TRANSFER FUNCTION
C
      CALL FFTP2(6,6,6,6,1.,M1,M1)
      CALL MAPDF(1,IER)
      IER1=4
      IF(IER.GE.5) GO TO 1001
      CALL REMAP(0,IJ,2,IER)
      IER2=5
      IF(IER.GE.29) GO TO 1003
      DO 12 KA=1,M2
      DO 12 KB=1,M1
      IF(KB.LE.M2) GO TO 12
      IT1(KA,KB-M2)=IT(KA,KB)
   12 IE(KA,KB)=IT(KA,KB)
      CALL REMAP(0,IK,2,IER)
      IER2=6
      IF(IER.GE.29) GO TO 1003
      DO 14 KA=1,M2
      DO 14 KB=1,M2
   14 IY1(KA,KB)=IY(KA,M2+KB)
      DO 402 KA=1,M2
      DO 402 KB=1,M1
  402 IY(KA,KB)=IE(KA,KB)
      CALL REMAP(0,IJ,2,IER)
      IER2=7
      IF(IER.GE.29) GO TO 1003
      DO 15 KA=1,M2
      DO 15 KB=1,M2
      IT(KA,KB)=REAL(X(M2+KA,M2+KB))
   15 IT(KA,M2+KB)=AIMAG(X(M2+KA,M2+KB))
      IJ=IJ+2
  100 IK=IK+2
      MIN=-M1
      KBL=14
      DO 401 IU=1,NT
      KBL=KBL+2
      MIN=MIN+M1
      CALL REMAP(0,KBL,2,IER)
      IER2=9
      IF(IER.GE.29) GO TO 1003
      DO 401 IV=1,M2
      DO 401 IW=1,M1
  401 ID(IV,MIN+IW)=IT(IV,IW)
      CALL FSEEK(2,IREC)
C
C     WRITING BACK THE FILTERED IMAGE BLOCK INTO
C     THE SAME IMAGE FILE
C
      DO 16 I1=1,M2
   16 WRITE(2) (ID(I1,I2),I2=1,NC)
    1 CONTINUE
```

```
      STOP
1000  TYPE"ERROR IN VMEM --->ERROR # = ",IER
      STOP
1001  TYPE"ERROR IN MAPDP--->ERROR AT= ",IER1,"  ERROR # = ",
      STOP
1003  TYPE"ERROR IN REMAP--->ERROR AT= ",IER2,"  ERROR # = ",
      STOP
      END

      SUBROUTINE FFTP2(M1,M2,L1,L2,SI,MX,MY)
C
C     ************** TDFFT **************
C
C     SUBROUTINE FFTP2 COMPUTES THE TWO-DIMENSIONAL FFT
C     OF A GIVEN TWO-DIMENSIONAL ARRAY OF SIZE MX X MY .
C
      DIMENSION X(64,64)
      DOUBLE PRECISION AA,BB,CC,SCL,ARG,DCOS,DSIN.
      COMPLEX CMPLX,W,X,T
      COMMON/BLK/X
      DATA CC/6.28318530717958/
      N1=2**M1
      N2=2**M2
      LP1=2**L1
      LP2=2**L2
      DO 9 LL=1,LP1
      DO 1 LO=1,M2
      LMX=2**(M2-LO)
      LIX=2*LMX
      SCL=CC/LIX
      IF(LO-M2+L2) 2,2,3
    2 DO 4 LM=1,LP2
      ARG=(LM-1)*SCL
      AA=DCOS(ARG)
      BB=SI*DSIN(ARG)
      AA1=SNGL(AA)
      BB1=SNGL(BB)
      W=CMPLX(AA1,BB1)
      DO 4 LI=LIX,N2,LIX
      J1=LI-LIX+LM
      J2=J1+LMX
    4 X(LL,J2)=W*X(LL,J1)
      GO TO 1
    3 DO 5 LM=1,LMX
      ARG=(LM-1)*SCL
      AA=DCOS(ARG)
      BB=SI*DSIN(ARG)
      AA1=SNGL(AA)
      BB1=SNGL(BB)
      W=CMPLX(AA1,BB1)
      DO 5 LI=LIX,N2,LIX
      J1=LI-LIX+LM
      J2=J1+LMX
```

```
      T=X(LL,J1)-X(LL,J2)
      X(LL,J1)=X(LL,J1)+X(LL,J2)
    5 X(LL,J2)=W*T
    1 CONTINUE
      NV2=N2/2
      NM1=N2-1
      J=1
      DO 7 I=1,NM1
      IF(I.GE.J) GO TO 6
      T=X(LL,J)
      X(LL,J)=X(LL,I)
      X(LL,I)=T
    6 K=NV2
    8 IF(K.GE.J) GO TO 7
      J=J-K
      K=K/2
      GO TO 8
    7 J=J+K
    9 CONTINUE
      DO 10 LL=1,N2
      DO 11 LO=1,M1
      LMX=2**(M1-LO)
      LIX=2*LMX
      SCL=CC/LIX
      IF(LO-M1+L1) 12,12,13
   12 DO 14 LM=1,LP1
      ARG=(LM-1)*SCL
      AA=DCOS(ARG)
      BB=SI*DSIN(ARG)
      AA1=SNGL(AA)
      BB1=SNGL(BB)
      W=CMPLX(AA1,BB1)
      DO 14 LI=LIX,N1,LIX
      J1=LI-LIX+LM
      J2=J1+LMX
   14 X(J2,LL)=W*X(J1,LL)
      GO TO 11
   13 DO 15 LM=1,LMX
      ARG=(LM-1)*SCL
      AA=DCOS(ARG)
      BB=SI*DSIN(ARG)
      AA1=SNGL(AA)
      BB1=SNGL(BB)
      W=CMPLX(AA1,BB1)
      DO 15 LI=LIX,N1,LIX
      J1=LI-LIX+LM
      J2=J1+LMX
      T=X(J1,LL)-X(J2,LL)
      X(J1,LL)=X(J1,LL)+X(J2,LL)
   15 X(J2,LL)=W*T
   11 CONTINUE
      NV2=N1/2
      NM1=N1-1
      J=1
```

```
      DO 17 I=1,NM1
      IF(I.GE.J) GO TO 16
      T=X(J,LL)
      X(J,LL)=X(I,LL)
      X(I,LL)=T
   16 K=NV2
   18 IF(K.GE.J) GO TO 17
      J=J-K
      K=K/2
      GO TO 18
   17 J=J+K
   10 CONTINUE
      RETURN
      END
```

```
      DIMENSION A(10,10),B(10,10),NAME(5),IWIND(16384)
C
C     ****************** TDRDF ******************
C
C     TWO DIMENSIONAL RECURSIVE FORWARD FILTERING
C     BY A RECURSIVE FILTER WHICH IS REALIZED IN
C     THE DIRECT FORM. THE SIZE OF THE INPUT DATA
C     IS M1 X M2 . M1 AND M2 ARE TO BE POWERS OF
C     2 . AND SHOULD BE G.E. 64 AND L.E. 256 .
C
      DIMENSION WN(8,264),X0(64,256)
      INTEGER X0
      COMMON/B1/IWIND
      EQUIVALENCE (IWIND,X0)
      CALL VMEM(ICNT,IER)
      IF(IER.EQ.5) GO TO 1000
      TYPE"# OF FREE 1024-WORD BLOCKS= ",ICNT
      CALL MAPDF(ICNT,IWIND,16,IER)
      IF(IER.GE.5) GO TO 1001
      TYPE "SIZE OF INPUT IMAGE"
      ACCEPT"# OF COLOUMNS = ",M2
      ACCEPT"# OF      ROWS = ",M1
      WRITE(10,100)
  100 FORMAT(' ',1X,'FILE NAME OF IMAGE= ',Z)
      READ(11,200) (NAME(I),I=1,4)
C
C     READ IN A MULTIPLICATION FACTOR AFC
C     IF THE FILTERED OUTPUT IS TO AMPLIFIED
C     BY A CERTAIN FACTOR . IF NOT READ IN
C     AFC = 1.
C
      ACCEPT"MULTIPLICATION FACTOR= ",AFC
C
      OPEN 2,NAME,LEN=2*M2,REC=M1
  200 FORMAT(5A2)
C
C     READ COEFFICIENTS OF THE FILTER . NP1 &
C     NQ1 ARE THE HIGHEST POWERS OF Z1 & Z2
C     RESPECTIVELY IN THE NUMERATOR OF THE
C     2D TRANSFER FUNCTION . SIMILERLY MP1 & MQ1
C     ARE FOR THE DENOMINATOR . AO IS THE GAIN
C     FACTOR OF THE TRANSFER FUNCTION .
C
C
      WRITE(10,102)
      READ(11,200) (NAME(I),I=1,4)
  102 FORMAT(' ',1X,'FILE NAME OF COEFS= ',Z)
      OPEN 4,NAME
      READ(4) NP1,NQ1,MP1,MQ1
      READ(4) ((A(I,J),J=1,NQ1),I=1,NP1)
      READ(4) ((B(I,J),J=1,MQ1),I=1,MP1)
      READ(4) AO
      CLOSE 4
```

```
C
C     NORMALIZE THE COEFFICIENTS OF THE FILTER
C     SUCH THAT A(1,1) AND B(1,1) ARE EQUAL TO
C     1 . THE GAIN FACTOR AO IS MODIFIED
C     ACCORDINGLY.
C
      IF(A(1,1).EQ.1.0) GO TO 500
      AO=AO*A(1,1)
      XCOF=A(1,1)
      DO 1 I=1,NP1
      DO 1 J=1,NQ1
      YCOF=A(I,J)
    1 A(I,J)=YCOF/XCOF
  500 IF(B(1,1).EQ.1.0) GO TO 501
      XCOF=B(1,1)
      AO=AO/XCOF
      DO 2 I=1,MP1
      DO 2 J=1,MQ1
      YCOF=B(I,J)
    2 B(I,J)=YCOF/XCOF
  501 MP2=MP1+1
      MP=MP1-1
      MQ=MQ1-1
C
C     SET THE INITIAL VALUES OF THE INTERMEDIATE
C     ARRAY WN EQUAL TO ZERO .
C
      DO 888 L=1,8
      DO 888 M=1,264
  888 WN(L,M)=0.0
      MT=M1/64
      KB=-16
C
C     READ THE INPUT DATA AND PERFORM THE
C     FILTERING OPERATION
C
      DO 800 NT=1,MT
      KB=KB+16
      CALL REMAP(0,KB,16,IER)
      IF(IER.GE.29) GO TO 1002
      DO 800 I=1,64
  800 READ(2) (X0(I,J)J=1,M2)
      REWIND 2
      KB=-16
      DO 3 NT=1,MT
      KB=KB+16
      CALL REMAP(0,KB,16,IER)
      IF(IER.GE.29) GO TO 1002
      DO 3 I=1,64
      DO 4 J0=1,M2
      XX=X0(I,J0)*AO
      SUM=0.
      JI=J0+MQ1
      JA=JI-1
```

VERY POOR COPY

```
      LS=2
      DO 33 K=1,MP1
      MPX=MP2-K
      DO 44 L=LS,MQ1
   44 SUM=SUM+B(K,L)*WN(MPX,JI-L)
   33 LS=1
      WN(MP1,JA)=XX-SUM
      SUM=0.
      LS=2
      DO 34 K=1,NP1
      MPX=MP2-K
      DO 45 L=LS,NQ1
   45 SUM=SUM+A(K,L)*WN(MPX,JI-L)
   34 LS=1
      XX=SUM+WN(MP1,JA)
    4 X0(I,J0)=XX*AFC
      DO 177 K=1,M2
      JI=K+M0
      DO 177 L=1,MP
  177 WN(L,JI)=WN(L+1,JI)
    3 CONTINUE
C
C     WRITE THE FILTERED DATA BACK TO THE
C     SAME FILE FROM WHICH THE INPUT DATA
C     WAS OBTAINED .
C
      KB=-16
      DO 16 NT=1,MT
      KB=KB+16
      CALL REMAP([illegible],KB,16,IER)
      IF(IER.GE.[illegible]) GO TO 1002
      DO 16 I=1,64
   16 WRITE(2) (X0(I,J),J=1,M2)
      CLOSE 2
      STOP
 1000 TYPE"ERROR IN VMEM ; ERROR # = ",IER
      STOP
 1001 TYPE"ERROR IN MAPDF ; ERROR # = ",IER
      STOP
 1002 TYPE"ERROR IN REMAP ; ERROR # = ",IER
      STOP
      END
```

```
      DIMENSION YS(3,18),YF(2,8),NAME(5)
C
C     ************* 2DRECMB *********************
C
C     REMOVAL OF MOTION BLUR BY FILTERING THE IMAGE BY
C     THE REAL PART OF TWO-D RECURSIVE DIGITAL FILTER.
C     IMAGE SIZE SHOULD BE EQUAL TO ( M1 X M2 ) WHERE
C     M1 & M2 ARE POWERS OF 2 AND GREATER THAN 63
C     THE DIMENSIONS OF YS,YF,WN,WM,WN1,WM1 SHOULD
C     BE AS FOLLOWS :- YS(KS,18),YF(KF,8),WN(KS,3,258),
C     WM(KS,3,258),WN1(KF,2,258),WM1(KF,2,258)
C     THE DIMENSIONS OF OTHER ARRAYS ARE FIXED AND THEY
C     CAN HANDLE IMAGES OF SIZE UPTO 256 X 256
C     THE PROGRAM WRITES BACK THE DEBLURRED IMAGE
C     INTO THE ORIGINAL BLURRED IMAGE FILE .
C     KS AND KF ARE THE # OF 2ND AND 1ST ORDER FILTER
C     SECTIONS RESPECTIVELY OF THE 2D FILTER TRANSFER
C     FUNCTION .
C
      DIMENSION WN(3,3,258),WM(3,3,258),WN1(2,2,258),
     *WM1(2,2,258)
      DIMENSION IWIND(4096),IX(16,256),IY(16,256)
      DOUBLE PRECISION YS,YF,AO,BO
      COMMON/B1/IWIND
      EQUIVALENCE(IWIND,IY)
      CALL VMEM(ICNT,IER)
      IF(IER.EQ.5) GO TO 3000
      TYPE"# OF FREE 1024-WORD BLOCKS= ",ICNT
      IWS=4
      CALL MAPDF(ICNT,IWIND,IWS,IER)
      IF(IER.GE.5) GO TO 3001
      TYPE"SIZE OF INPUT IMAGE"
C
C     ACCEPT THE IMAGE SIZE AND READ OUT THE IMAGE
C     DATA INTO THE EXTENDED MEMORY
C
      ACCEPT"# OF COLOUMNS= ",M2
      ACCEPT"# OF     ROWS= ",M1
      WRITE(10,100)
  100 FORMAT(' ',1X,'FILE NAME OF NOISY BLURRED IMAGE= ',Z)
      READ(11,200) (NAME(I),I=1,5)
      OPEN 2,NAME,LEN=2*M2,REC=M1
      NR=16
      NRA=NR+1
      M1X=M1/NR
      M2X1=M2+1
      M2X2=M2+2
      KBX=IWS*M1X
      KB=-IWS
      DO 60 I=1,M1X
      KB=KB+IWS
      CALL REMAP(0,KB,IWS,IER)
      IF(IER.GE.29) GO TO 3002
```

```
      DO 60 K=1,NR
   60 READ(2) (IY(K,J),J=1,M2)
      REWIND 2
C
C     READ IN THE COEFFICIENT FILE NAME AND STORE
C     THE COEFFICIENTS IN THE COEFFICIENT ARRAYS
C     YS AND YF
C
      WRITE(10,102)
      READ(11,200) (NAME(I),I=1,5)
  102 FORMAT(' ',1X,'FILE NAME OF COEFS  = ',Z)
      OPEN 4,NAME
      READ(4) KS,KF
      NT1=KS*18
      NT2=KF*8
      IF(KS.GE.1) READ(4) ((YS(K,L),L=1,18),K=1,KS)
      IF(KF.GE.1) READ(4) ((YF(K,L),L=1,8),K=1,KF)
      READ(4) AO
      CLOSE 4
C
C     SET THE INITIAL VALUES OF THE INTERMEDIATE
C     ARRAYS WN,WM,WN1,WM1 TO ZERO
C
      DO 234 NZ1=1,3
      DO 234 NZ2=1,3
      DO 234 NZ3=1,M2X2
      WN(NZ1,NZ2,NZ3)=0.D0
  234 WM(NZ1,NZ2,NZ3)=0.D0
      DO 456 NZ1=1,2
      DO 456 NZ2=1,2
      DO 456 NZ3=1,M2X1
      WN1(NZ1,NZ2,NZ3)=0.D0
  456 WM1(NZ1,NZ2,NZ3)=0.D0
      KX1=-NR+1
      KB1=KBX
C
C     FILTER THE IMAGE BY THE REAL PART OF THE 2D FILTER
C
      DO 3 I=1,M1X
      KX1=KX1+NR
      CALL FSEEK(2,KX1)
      DO 43 IM=1,NR
   43 READ(2) (IX(IM,IN),IN=1,M2)
      KB1=KB1-IWS
      CALL REMAP(0,KB1,IWS,IER)
      IF(IER.GE.29) GO TO 3002
      DO 555 IO=1,NR
      DO 4 J=1,M2
      XX=AO*FLOAT(IX(IO,J))
      YY=AO*FLOAT(IY(NRA-IO,M2X1-J))
      IF(KS.EQ.0) GO TO 222
      JZ=J+2
      DO 50 K=1,KS
C
```

```
      WN(K,3,JZ)=XX-YS(K,11)*WN(K,3,JZ-1)-YS(K,12)*WN(K,3,JZ-2)
     *-YS(K,13)*WN(K,2,JZ)-YS(K,14)*WN(K,2,JZ-1)-YS(K,15)*WN(K,
     *2,JZ-2)-YS(K,16)*WN(K,1,JZ)-YS(K,17)*WN(K,1,JZ-1)-YS(K,18
     *)*WN(K,1,JZ-2)
      WM(K,3,JZ)=YY-YS(K,11)*WM(K,3,JZ-1)-YS(K,12)*WM(K,3,JZ-2)
     *-YS(K,13)*WM(K,2,JZ)-YS(K,14)*WM(K,2,JZ-1)-YS(K,15)*WM(K
     *,2,JZ-2)-YS(K,16)*WM(K,1,JZ)-YS(K,17)*WM(K,1,JZ-1)-YS(K,
     *18)*WM(K,1,JZ-2)
      XX=WN(K,3,JZ)+YS(K,2)*WN(K,3,JZ-1)+YS(K,3)*WN(K,3,JZ-2)+Y
     *S(K,4)*WN(K,2,JZ)+YS(K,5)*WN(K,2,JZ-1)+YS(K,6)*WN(K,2,JZ-
     *2)+YS(K,7)*WN(K,1,JZ)+YS(K,8)*WN(K,1,JZ-1)+YS(K,9)*WN(K,1
     *,JZ-2)
   50 YY=WM(K,3,JZ)+YS(K,2)*WM(K,3,JZ-1)+YS(K,3)*WM(K,3,JZ-2)+Y
     *S(K,4)*WM(K,2,JZ)+YS(K,5)*WM(K,2,JZ-1)+YS(K,6)*WM(K,2,JZ
     *-2)+YS(K,7)*WM(K,1,JZ)+YS(K,8)*WM(K,1,JZ-1)+YS(K,9)*WM(K
     *,1,JZ-2)
C
  222 IF(KF.EQ.0) GO TO 44
      JD=J+1
      DO 52 K=1,KF
C
      WN1(K,2,JD)=XX-YF(K,6)*WN1(K,2,JD-1)-YF(K,7)*WN1(K,1,JD)-
     *YF(K,8)*WN1(K,1,JD-1)
      WM1(K,2,JD)=YY-YF(K,6)*WM1(K,2,JD-1)-YF(K,7)*WM1(K,1,JD)-
     *YF(K,8)*WM1(K,1,JD-1)
      XX=WN1(K,2,JD)+YF(K,2)*WN1(K,2,JD-1)+YF(K,3)*WN1(K,1,JD)+
     *YF(K,4)*WN1(K,1,JD-1)
   52 YY=WM1(K,2,JD)+YF(K,2)*WM1(K,2,JD-1)+YF(K,3)*WM1(K,1,JD)+
     *YF(K,4)*WM1(K,1,JD-1)
C
   44 IX(IO,J)=XX
      IY(NRA-IO,M2X1-J)=YY
    4 CONTINUE
      IF(KS.EQ.0) GO TO 444
      DO 177 K=1,KS
      DO 177 L=1,M2
      LZ=L+2
      WN(K,1,LZ)=WN(K,2,LZ)
      WN(K,2,LZ)=WN(K,3,LZ)
      WM(K,1,LZ)=WM(K,2,LZ)
  177 WM(K,2,LZ)=WM(K,3,LZ)
  444 IF(KF.EQ.0) GO TO 555
      DO 28 K=1,KF
      DO 28 L=1,M2
      LZ=L+1
      WN1(K,1,LZ)=WN1(K,2,LZ)
   28 WM1(K,1,LZ)=WM1(K,2,LZ)
  555 CONTINUE
      CALL FSEEK(2,KX1)
      DO 3 IM=1,NR
    3 WRITE(2) (IX(IM,IN),IN=1,M2)
      REWIND 2
      KB=-IWS
      DO 55 I=1,M1X
```

```
      KB=KB+IWS
      CALL REMAP(0,KB,IWS,IER)
      IF(IER.GE.29) GO TO 3002
      DO 56 K=1,NR
   56 READ(2) (IX(K,L),L=1,M2)
      DO 55 K=1,NR
      DO 55 L=1,M2
   55 IY(K,L)=IY(K,L)+IX(K,L)
      REWIND 2
      KB=-IWS
C
C     WRITE BACK THE FILTERED IMAGE BACK
C     TO THE ORIGINAL BLURRED IMAGE FILE
C
      DO 57 I=1,M1X
      KB=KB+IWS
      CALL REMAP(0,KB,IWS,IER)
      IF(IER.GE.29) GO TO 3002
      DO 57 K=1,NR
   57 WRITE(2) (IY(K,L),L=1,M2)
      CLOSE 2
  200 FORMAT(5A2)
      STOP
 3000 TYPE"ERROR IN VMEM ; ERROR # = ",IER
      CLOSE 2
      STOP
 3001 TYPE"ERROR IN MAPDF ; ERROR # = ",IER
      CLOSE 2
      STOP
 3002 TYPE"ERROR IN REMAP ; ERROR # = ",IER
      CLOSE 2
      STOP
      END
```

```
      DIMENSION YS(3,18),YF(2,8),NAME(5)
C
C     ************* 2DRECAB ********************
C
C     REMOVAL OF ATMOSPHERIC BLUR BY ZERO PHASE
C     TWO-DIMENSIONAL RECURSIVE DIGITAL FILTERING.
C     IMAGE SIZE SHOULD BE EQUAL TO ( M1 X M2 )
C     WHERE M1 & M2 ARE POWERS OF 2 AND GREATER
C     THAN 63 BUT LESS OR EQUAL TO 256 . THE
C     DIMENSIONS OF YS,YF,WN,WN1 ARE AS FOLLOWS
C     YS(KS,18),YF(KF,8),WN(KS,3,258),WN1(KF,2,258)
C     KS,KF ARE THE # OF 2ND AND 1ST ORDER FILTER
C     SECTIONS RESPECTIVELY OF THE 2D FILTER. THE
C     DIMENSIONS OF THE OTHER ARRAYS ARE FIXED AND
C     THEY CAN HANDLE IMAGES OF SIZE UPTO 256 X 256
C     THE PROGRAM WRITES BACK THE RESTORED IMAGE TO
C     THE ORIGINAL BLURRED IMAGE FILE
C
C
      DIMENSION WN(3,3,258),WN1(2,2,258)
      DIMENSION IWIND(16384),IY(64,256)
      DOUBLE PRECISION YS,YS1,YF,YF1,A0,B0
      COMMON/B1/IWIND
      EQUIVALENCE(IWIND,IY)
      CALL VMEM(ICNT,IER)
      IF(IER.EQ.5) GO TO 3000
      TYPE"# OF FREE 1024-WORD BLOCKS= ",ICNT
      IWS=16
      CALL MAPDF(ICNT,IWIND,IWS,IER)
      IF(IER.GE.5) GO TO 3001
      TYPE"SIZE OF INPUT IMAGE"
C
C     READ IN THE SIZE OF IMAGE AND READ OUT THE
C     IMAGE DATA INTO THE EXTENDED MEMORY
C
      ACCEPT"# OF COLOUMNS= ",M2
      ACCEPT"# OF      ROWS= ",M1
      WRITE(10,100)
  100 FORMAT(' ',1X,'FILE NAME OF NOISY BLURRED IMAGE= ',Z)
      READ(11,200) (NAME(I),I=1,5)
      OPEN 2,NAME,LEN=2*M2,REC=M1
      NR=64
      NRA=NR+1
      M1X=M1/NR
      M2X1=M2+1
      M2X2=M2+2
      KBX=IWS*M1X
      KB=-IWS
      DO 60 I=1,M1X
      KB=KB+IWS
      CALL REMAP(0,KB,IWS,IER)
      IF(IER.GE.29) GO TO 3002
      DO 60 K=1,NR
   60 READ(2) (IY(K,J),J=1,M2)
```

```
C
C     READ IN THE COEFFICIENT FILE NAME AND STORE
C     THE COEFFICINTS IN THE COEFFICIENT ARRAYS
C     YS AND YF
C
      WRITE(10,102)
      READ(11,200) (NAME(I),I=1,5)
  102 FORMAT(' ',1X,'FILE NAME OF COEFS  = ',Z)
      OPEN 4,NAME
      READ(4) KS,KF
      NT1=KS*18
      NT2=KF*8
      IF(KS.GE.1) READ(4) ((YS(K,L),L=1,18),K=1,KS)
      IF(KF.GE.1) READ(4) ((YF(K,L),L=1,8),K=1,KF)
      READ(4) A0
      CLOSE 4
C
C     SET THE INITIAL CONDITIONS OF THE INTERMEDIATE
C     ARRAYS WN AND WN1 TO ZERO AND FILTER THE IMAGE
C     BY THE MAGNITUDE SQUARED 2D FILTER TRANSFER
C     FUNCTION .
C
      DO 3 IJ=1,2
      DO 234 NZ1=1,3
      DO 234 NZ2=1,3
      DO 234 NZ3=1,M2X2
  234 WN(NZ1,NZ2,NZ3)=0.D0
      DO 456 NZ1=1,2
      DO 456 NZ2=1,2
      DO 456 NZ3=1,M2X1
  456 WN1(NZ1,NZ2,NZ3)=0.D0
      KB=KBX
      IF(IJ.EQ.1) KB=-IWS
      DO 3 I=1,M1X
      IF(IJ.EQ.2) GO TO 778
      KB=KB+IWS
      CALL REMAP(0,KB,IWS,IER)
      IF(IER.GE.29) GO TO 3002
      GO TO 779
  778 KB=KB-IWS
      CALL REMAP(0,KB,IWS,IER)
      IF(IER.GE.29) GO TO 3002
  779 DO 3 IO=1,NR
      DO 4 J=1,M2
      IF(IJ.EQ.2) GO TO 780
      XX=IY(IO,J)*A0
      GO TO 781
  780 XX=IY(NRA-IO,M2X1-J)*A0
  781 IF(KS.EQ.0) GO TO 222
      JZ=J+2
      DO 50 K=1,KS
C
      WN(K,3,JZ)=XX-YS(K,11)*WN(K,3,JZ-1)-YS(K,12)*WN(K,3,JZ-2)
     *-YS(K,13)*WN(K,2,JZ)-YS(K,14)*WN(K,2,JZ-1)-YS(K,15)*WN(K,
```

```
     *2,JZ-2)-YS(K,16)*WN(K,1,JZ)-YS(K,17)*WN(K,1,JZ-1)-YS(K,18
     *)*WN(K,1,JZ-2)
   50 XX=WN(K,3,JZ)+YS(K,2)*WN(K,3,JZ-1)+YS(K,3)*WN(K,3,JZ-2)+Y
     *S(K,4)*WN(K,2,JZ)+YS(K,5)*WN(K,2,JZ-1)+YS(K,6)*WN(K,2,JZ-
     *2)+YS(K,7)*WN(K,1,JZ)+YS(K,8)*WN(K,1,JZ-1)+YS(K,9)*WN(K,1
     *,JZ-2)
C
  222 IF(KF.EQ.0) GO TO 44
      JD=J+1
      DO 52 K=1,KF
C
      WN1(K,2,JD)=XX-YF(K,6)*WN1(K,2,JD-1)-YF(K,7)*WN1(K,1,JD)-
     *YF(K,8)*WN1(K,1,JD-1)
   52 XX=WN1(K,2,JD)+YF(K,2)*WN1(K,2,JD-1)+YF(K,3)*WN1(K,1,JD)+
     *YF(K,4)*WN1(K,1,JD-1)
C
   44 IF(IJ.EQ.2) GO TO 556
      IY(IO,J)=XX
      GO TO 4
  556 IY(NRA-IO,M2X1-J)=XX
    4 CONTINUE
      IF(KS.EQ.0) GO TO 444
      DO 177 K=1,KS
      DO 177 L=1,M2
      LZ=L+2
      WN(K,1,LZ)=WN(K,2,LZ)
  177 WN(K,2,LZ)=WN(K,3,LZ)
  444 IF(KF.EQ.0) GO TO 3
      DO 28 K=1,KF
      DO 28 L=1,M2
      LZ=L+1
   28 WN1(K,1,LZ)=WN1(K,2,LZ)
    3 CONTINUE
      REWIND 2
      KB=-IWS
C
C     WRITE THE DEBLURRED IMAGE BACK INTO THE
C     ORIGINAL BLURRED IMAGE FILE
C
      DO 57 I=1,M1X
      KB=KB+IWS
      CALL REMAP(0,KB,IWS,IER)
      IF(IER.GE.29) GO TO 3002
      DO 57 K=1,NR
   57 WRITE(2) (IY(K,L),L=1,M2)
      CLOSE 2
  200 FORMAT(5A2)
      STOP
 3000 TYPE"ERROR IN VMEM ; ERROR # = ",IER
      CLOSE 2
      STOP
 3001 TYPE"ERROR IN MAPDF ; ERROR # = ",IER
      CLOSE 2
      STOP
 3002 TYPE"ERROR IN REMAP ; ERROR # = ",IER
      CLOSE 2
      STOP
      END
```

VITA AUCTORIS

1951	Born on 7th of December 1951, Coorg District, Mysore State, INDIA.
1966	Completed high school at St. Anne's Higher Secondary School, Virajpet, Coorg, Mysore State, INDIA.
1967	Completed Pre-University Course, University of Madras, Madras, INDIA.
1972	Graduated from University of Madras, with a B.E. (Bachelor of Engineering) degree in Electronics and Communications Engineering.
1974	Graduated from University of Windsor, with the degree of M.A.Sc. in Electrical Engineering.
1979	Candidate for the degree of Doctor of Philosophy in Electrical Engineering. University of Windsor, Windsor, Ontario, Canada.