Analysis and synthesis of one and two-dimensional recursive digital filters.

Maher Ahmed. Sid-Ahmed

University of Windsor

Follow this and additional works at: https://scholar.uwindsor.ca/etd

Recommended Citation

https://scholar.uwindsor.ca/etd/1156

This online database contains the full-text of PhD dissertations and Masters' theses of University of Windsor students from 1954 forward. These documents are made available for personal study and research purposes only, in accordance with the Canadian Copyright Act and the Creative Commons license—CC BY-NC-ND (Attribution, Non-Commercial, No Derivative Works). Under this license, works must always be attributed to the copyright holder (original author), cannot be used for any commercial purposes, and may not be altered. Any other use would require the permission of the copyright holder. Students may inquire about withdrawing their dissertation and/or thesis from this database. For additional inquiries, please contact the repository administrator via email (scholarship@uwindsor.ca) or by telephone at 519-253-3000ext. 3208.
ANALYSIS AND SYNTHESIS
OF ONE AND TWO-
DIMENSIONAL RECURSIVE
DIGITAL FILTERS

by

MAHER AHMED-SID-AHMED

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

1974
ABSTRACT

In this dissertation three main areas in one and two-dimensional recursive digital filtering have been investigated:

1) Generation of the impulse response (in closed-form and numerically).
2) Recursive filter design to approximate a given impulse response.
3) Stable recursive filter design to approximate given magnitude-frequency specifications.

In each main area one-dimensional recursive filters have been studied and techniques based on these investigations have been extended to two-dimensional recursive filters.
ACKNOWLEDGEMENT

The advice, guidance, support and constant encouragement of my supervisor, Dr. G. A. Jullien, is acknowledged with gratitude.

The financial support provided by the National Research Council of Canada is gratefully acknowledged.

Thanks and gratitude are also due to my wife, Yossra, for her constant encouragement.
# TABLE OF CONTENTS

<table>
<thead>
<tr>
<th>Section</th>
<th>Page</th>
</tr>
</thead>
<tbody>
<tr>
<td>ABSTRACT</td>
<td>i</td>
</tr>
<tr>
<td>ACKNOWLEDGEMENTS</td>
<td>ii</td>
</tr>
<tr>
<td>LIST OF APPENDICES</td>
<td>v</td>
</tr>
<tr>
<td>CHAPTER I: INTRODUCTION</td>
<td></td>
</tr>
<tr>
<td>1.1 DIGITAL FILTERING</td>
<td>1</td>
</tr>
<tr>
<td>1.2 ONE-DIMENSIONAL DIGITAL FILTERS</td>
<td>2</td>
</tr>
<tr>
<td>1.3 TWO-DIMENSIONAL DIGITAL FILTERS</td>
<td>3</td>
</tr>
<tr>
<td>1.4 Z-TRANSFORM</td>
<td>5</td>
</tr>
<tr>
<td>1.5 STABILITY OF RECURSIVE FILTERS</td>
<td>9</td>
</tr>
<tr>
<td>1.6 MOTIVATION</td>
<td>11</td>
</tr>
<tr>
<td>1.7 LITERATURE SURVEY</td>
<td>12</td>
</tr>
<tr>
<td>1.7a Analysis of Recursive Filters</td>
<td>12</td>
</tr>
<tr>
<td>1.7b Synthesis of Recursive Filters</td>
<td>12</td>
</tr>
<tr>
<td>1.8 THESIS ORGANIZATION</td>
<td>15</td>
</tr>
<tr>
<td>CHAPTER II: GENERATION OF THE IMPULSE RESPONSE</td>
<td></td>
</tr>
<tr>
<td>2.1 ANALYTICAL INVERSION OF THE Z-TRANSFORM</td>
<td>17</td>
</tr>
<tr>
<td>2.1.1 Analysis</td>
<td>17</td>
</tr>
<tr>
<td>2.1.2 The Closed Form Solution</td>
<td>21</td>
</tr>
<tr>
<td>2.1.3 Example</td>
<td>23</td>
</tr>
<tr>
<td>2.1.4 Computation Details</td>
<td>24</td>
</tr>
<tr>
<td>2.1.5 Extension to Laplace Transform</td>
<td>25</td>
</tr>
<tr>
<td>2.1.6 Computer Programme</td>
<td>28</td>
</tr>
<tr>
<td>2.2 GENERATION OF THE IMPULSE RESPONSE OF A TWO-DIMENSIONAL RECURSIVE</td>
<td></td>
</tr>
<tr>
<td>2.2.1 INTRODUCTION</td>
<td>34</td>
</tr>
<tr>
<td>2.2.2 Closed Form Solutions for the Generation of the Horizontal Lines</td>
<td>34</td>
</tr>
<tr>
<td>2.2.3 Closed Form Solutions for the Generation of the Vertical Lines</td>
<td>35</td>
</tr>
<tr>
<td>2.2.4 Necessary Conditions for the Stability of Two-Dimensional</td>
<td>38</td>
</tr>
<tr>
<td>Recursive Filters</td>
<td>39</td>
</tr>
<tr>
<td>2.2.5 Generation of the First (N_b+1) Vertical Lines in Closed Form</td>
<td>40</td>
</tr>
<tr>
<td>2.2.6 Generation of the First (N_b+1) Horizontal Lines in Closed Form</td>
<td>47</td>
</tr>
<tr>
<td>2.2.7 Generating the Full Impulse Response</td>
<td>50</td>
</tr>
<tr>
<td>2.2.8 Examples</td>
<td>51</td>
</tr>
<tr>
<td>2.3 GENERATION OF THE IMPULSE RESPONSE VIA FFT PRUNING</td>
<td>56</td>
</tr>
<tr>
<td>2.4 SUMMARY</td>
<td>59</td>
</tr>
<tr>
<td>Chapter III: Design of Recursive Filters to Approximate a Prescribed Impulse Response</td>
<td></td>
</tr>
<tr>
<td>---------------------------------------------------------------</td>
<td>----------------</td>
</tr>
<tr>
<td>3.1 Introduction</td>
<td>60</td>
</tr>
<tr>
<td>3.2 One-Dimensional Recursive Filters</td>
<td>60</td>
</tr>
<tr>
<td>3.2.1 Formulation of the Problem</td>
<td>61</td>
</tr>
<tr>
<td>3.2.2 Relation Between the Time and Frequency Approach</td>
<td>61</td>
</tr>
<tr>
<td>3.2.3 Analysis</td>
<td>63</td>
</tr>
<tr>
<td>3.2.4 The Iterative Algorithm</td>
<td>64</td>
</tr>
<tr>
<td>3.2.5 Computation Details</td>
<td>68</td>
</tr>
<tr>
<td>3.2.6 Comments</td>
<td>71</td>
</tr>
<tr>
<td>3.2.7 Examples</td>
<td>75</td>
</tr>
<tr>
<td>3.3 Two-Dimensional Recursive Filters</td>
<td>76</td>
</tr>
<tr>
<td>3.3.1 Analysis</td>
<td>93</td>
</tr>
<tr>
<td>3.3.2 Iterative Algorithm</td>
<td>93</td>
</tr>
<tr>
<td>3.3.3 Computational Details</td>
<td>106</td>
</tr>
<tr>
<td>3.3.4 Examples</td>
<td>107</td>
</tr>
<tr>
<td>3.4 Summary</td>
<td>115</td>
</tr>
</tbody>
</table>

| Chapter IV: Stable Recursive Filter Design, Given Arbitrary Magnitude-Frequency Specifications |
|---------------------------------------------------------------|----------------|
| 4.1 Introduction                                             | 126            |
| 4.2 One-Dimensional Recursive Filter Design                  | 126            |
| 4.2.1 Statement of the Problem                               | 127            |
| 4.2.2 Performance Criterion                                 | 127            |
| 4.2.3 Constraint Implementation                              | 128            |
| 4.3 Two-Dimensional Recursive Filter Design                  | 131            |
| 4.3.1 Statement of the Problem                               | 132            |
| 4.3.2 Huang's Special Class of Second-Order Filters          | 133            |
| 4.3.3 Stable Fourth-Order Filter                             | 137            |
| 4.3.4 Performance Criterion                                 | 148            |
| 4.3.5 Minimum Phase                                          | 150            |
| 4.3.6 Examples                                               | 151            |
| 4.3.7 Computational Details                                 | 175            |
| 4.3.8 Computer Programme                                    | 176            |
| 4.4 Recursive Filtering of Spatial Data                     | 177            |
| 4.5 Summary                                                  | 181            |

| Chapter V: Conclusion                                        |
|---------------------------------------------------------------|----------------|
| REFERENCES                                                   | 182            |
| APPENDICES                                                   | 185            |
| VITA AUCTORIS                                                | 190            |
# LIST OF APPENDICES

| APPENDIX A: | Fortran IV Coding of the Algorithm for the Inverse Z-Transform, Given in Chapter II | 189 |
| APPENDIX B: | Time-Domain FFT Pruning | 196 |
| APPENDIX C: | Frequency Domain FFT Pruning | 205 |
| APPENDIX D: | Numerical Generation of Impulse Response Using Time (Or Spatial) Domain Pruned FFT, for the One-Dimensional (or Two-Dimensional) Digital Filters | 213 |
| APPENDIX E: | Design of Recursive Filters to Approximate a Prescribed Impulse Response | 220 |
| APPENDIX E1: | Fortran Listing for the One-Dimensional Filter Design | 221 |
| APPENDIX E2: | Fortran Listing for the Spatial Filter Design | 227 |
| APPENDIX F: | Listings of the Gradients of the Performance Criterion for the Design of Second-Order and Fourth-Order Cascades of Recursive Filters Discussed in Chapter IV | 239 |
| APPENDIX G: | Stable Two-Dimensional Recursive Filter Design to Approximate Magnitude-Frequency Specifications. | 244 |
CHAPTER I

INTRODUCTION

1.1 DIGITAL FILTERING:

Linear filtering is a fundamental signal-processing operation, which may be performed digitally by special- or general-purpose digital computers, as well as continuously by passive networks and active networks (for one-dimensional signal processing), or optically (for picture processing).

Digital filters are, in some cases, more advantageous to use than continuous filters. For example, when many filters are required at the same installation. For the digital approach all the filters may be able to share a common realization, except for the memory required for each. Thus, the cost required would contain a linear factor, the cost of the memory, which is small, and a constant factor, the cost of the realization (essentially an arithmetic element), which is larger. Using analogue techniques, N filters will cost N times as much, and take N times as much space, as a single filter. Another advantage arises with extreme complexity. An analog filter with more than about 20 poles is likely to be almost impossible to tune, even if it is realized actively, with isolation between stages. This is not a problem in digital realization. In situations where adaptive or time varying filters are required, the
digital filter is by far easier to realize.

1.2 ONE-DIMENSIONAL DIGITAL FILTERS:

Linear digital filters can be characterized by two main types: non-recursive filters, which have an impulse response of finite duration, and recursive filters, which have an impulse response of infinite duration.

Much work has been done on both types of one-dimensional filters over the past decade, each filter having its own advantages and disadvantages. In general, the recursive filter is better able to match desired magnitude-frequency responses than the non-recursive filter and at a much reduced number of computations. This advantage lies in the fact that a feedback loop is used to modify the input of the filter with weighted samples of previous outputs, yielding an infinite impulse response with the requirement of only a finite number of computational steps per output point. An inherent instability problem, associated with the feedback loop, is one of the main disadvantages of this filter. Another disadvantage is that it is difficult to match recursive filters to required phase, particularly linear phase. This is not a problem with non-recursive filters.

Non-recursive filters generate their output from weighted samples of previous inputs only, in fact they can be regarded as performing a discrete convolution between the impulse response of the filter and the data. Although this process is inherently more inefficient than the
recursive filter process, certain computational advantages of the Fast Fourier Transform [1] have been used to speed up the process.

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

$$g(m) = \sum_{i=0}^{M_a} a_i f(m-i) - \sum_{i=1}^{M_b} b_i g(m-i)$$  \hspace{1cm} (1.1)

where; $g(m)$ and $f(m)$ are the output and input, respectively, of the filter. \{a_i\} & \{b_i\} are real constants.

The one-dimensional non-recursive filter is characterized by the difference equation

$$g(m) = \sum_{i=0}^{M_c} c_i f(m-i)$$  \hspace{1cm} (1.2)

where; $g(m)$ and $f(m)$ are defined as in (1.1). \{c_i\}_{i=0}^{M_c}$ are real constants.

1.3 TWO-DIMENSIONAL DIGITAL FILTERS:

Filtering of sampled two-dimensional data, such as digitised images, requires the use of two-dimensional digital filters. These, as in the one-dimensional case, are of two types: recursive and non-recursive. There are similar advantages and disadvantages as discussed for one-dimensional filters, except that the stability problem for the two-dimensional filter is much more severe than in the one-dimensional case.
Two-dimensional recursive filters are characterized by the spatial-domain difference equation [3],

\[ \sum_{k=0}^{M_b} \sum_{\ell=0}^{N_b} b_{k\ell} g(m-k,n-\ell) = \sum_{i=0}^{M_a} \sum_{j=0}^{N_a} a_{ij} f(m-i,n-j) \]  

(1.3)

where \( g(m,n) \) and \( f(m,n) \) are the output and input, respectively, of the filter. \( \{b_{k\ell}\} \) and \( \{a_{ij}\} \) are constants.

From equation (1.3), we can express \( g(m,n) \) as

\[ g(m,n) = \frac{1}{b_{ko}} \sum_{i=0}^{M_a} \sum_{j=0}^{N_a} a_{ij} f(m-i,n-j) \]

\[ \sum_{k=0}^{M_b} \sum_{\ell=0}^{N_b} b_{k\ell} g(m-k,n-\ell) ; \]

(1.4)

\( k, \ell \neq 0 \)

thus obtaining a recursive filter that recurses in the \((+m,+n)\) direction. We can also express \( g(m-M_b,n-N_b) \) in terms of the rest, and obtain a recursive filter that recurses in the \((-m,-n)\) direction. Similarly, we can get from (1.3) recursive filters recursing in the \((+m,-n)\) and \((-m,+n)\) directions. Among the four recursive filters associated with (1.3), at most one can be stable [4]. We shall associate (1.3) with the recursive filter that recurses in the \((+m,+n)\) direction, and shall call it causal [4].

To start recursing, we need initial conditions. For the causal filter, we need to know the values of the output
g in the shaded region in Fig. 1.1.

Fig. 1.1. Initial Conditions for a Causal Filter.

Non-recursive two-dimensional digital filters, are characterized by the spatial-domain difference equation

\[ g(m,n) = \sum_{i=0}^{M_C} \sum_{j=0}^{N_C} c_{ij} f(m-i,n-j) \]  

where; \( g(m,n) \) and \( f(m,n) \) are as defined in (1.3). \( \{c_{ij}\} \) are real constants.

1.4 Z-transform:

The z-transform is the mathematical basis of digital signal-processing operations. The one-dimensional z-transform of a sequence \( \{h(n)\} \), \( n=0,1,2,... \) is defined as

\[ H(z) = \sum_{n=0}^{\infty} h(n)z^{-n} \]  

where, \( z \) is a complex variable.

If we consider generating the sequence \( h_n \) by sampling a continuous signal \( h(t) \) at regular intervals of time \( T \), the
Laplace Transform of this sampled sequence is given by

\[ H^*(s) = \int_0^\infty h(t) \sum_{n=0}^{\infty} \delta(t-nT) e^{-st} \, dt \]

\[ = \sum_{n=0}^{\infty} \int_0^\infty h(t) \delta(t-nT) e^{-st} \, dt \]

\[ H^*(s) = \sum_{n=0}^{\infty} h(nT) e^{-STn} \]  \hspace{2cm} (1.7)

Thus, \( H(z) \) can be considered to be equivalent to \( H^*(s) \) via the substitution

\[ z = e^{sT} \]  \hspace{2cm} (1.8)

The \( z \)-transform may be inverted analytically by the integral [1]

\[ h(nT) = \frac{1}{2\pi j} \oint H(z) z^{n-1} \, dz \]  \hspace{2cm} (1.9)

where, the path of integration is a contour in a region of analyticity in the \( z \)-plane enclosing all the poles of \( H(z) \).

Chapters 1, 2, and 4 of Jury [2] contain a development of \( z \)-transform theory and its application to linear difference equations.
Taking the z-transform of equation (1.1), assuming zero initial conditions, and denoting $H(z)$ as the transfer function of the recursive digital filter, we can write

$$H(z) = \frac{\sum_{i=0}^{M_a} a_i z^{-i}}{\sum_{i=0}^{M_b} b_i z^{-i}} \tag{1.11}$$

Similarly, the one-dimensional non-recursive digital filter characterized by equation (1.2) can be written as

$$H(z) = \sum_{i=0}^{M_c} c_i z^{-i} \tag{1.12}$$

The two-dimensional z-transform is defined as $[3]$: 

$$D(z_1, z_2) = \sum_{i=0}^{\infty} \sum_{j=0}^{\infty} d_{ij} z_1^{-i} z_2^{-j} \tag{1.13}$$

where:

$$z_1 = e^{s_1 A} \tag{1.14}$$

$$z_2 = e^{s_2 B}$$

$s_1$ and $s_2$ are, respectively, the horizontal and vertical complex spatial frequency variables, and $A$ and $B$ are constants (sampling periods in the vertical and horizontal directions, respectively).
The two-dimensional $z$-transform may be inverted analytically via the double integral [7]

\[
d_{p,q} = \frac{1}{(2\pi j)^2} \int \int D(z_1, z_2) z_1^{p-1} z_2^{q-1} \, dz_1 \, dz_2
\]

where the paths of integration are within the regions of convergence of the infinite double series $D(z_1, z_2)$.

Taking the two-dimensional $z$-transform of (1.3) with zero-initial conditions, and denoting $H(z_1, z_2)$ as the transfer function of the two-dimensional recursive filter we get

\[
H(z_1, z_2) = \frac{\sum_{i=0}^{N_a} \sum_{j=0}^{N_a} a_{ij} z_1^{-i} z_2^{-j}}{\sum_{k=0}^{M_b} \sum_{\ell=0}^{N_b} b_{k\ell} z_1^{-k} z_2^{-\ell}} \tag{1.15}
\]

Similarly, the transfer function of the two-dimensional non-recursive digital filter characterized by (1.5) can be written as

\[
H(z_1, z_2) = \sum_{i=0}^{M_c} \sum_{j=0}^{N_c} c_{ij} z_1^{-i} z_2^{-j} \tag{1.16}
\]

Important for signal processing applications is the convolution property. In one dimension the property may be stated as
\[ Y(z) = Z(h_n * x_n) = H(z)X(z) \]  

(1.17)

where \( Z \) denotes the z-transform operation and, \( \ast \) denotes linear (as opposed to cyclic) convolution. This property also extends to the two-dimensional z-transform, i.e.

\[ Y(z_1, z_2) = Z_{1,2}(h_{n,m} * x_{n,m}) = H(z_1, z_2)X(z_1, z_2) \]  

(1.18)

1.5 **STABILITY OF RECURSIVE FILTERS:**

A necessary and sufficient condition for the stability of a one-dimensional digital filter is

\[
\sum_{n=0}^{\infty} |h(nT)| < \infty \quad (1.19)
\]

Inequality (1.19) is satisfied if the bounded unit-pulse response \( h(nT) \) is truncated, that is, if \( h(nT) = 0 \) except for \( N_1 < n < N_2 \). It can also be satisfied by unit-pulse responses of unlimited duration, (for recursive filters), if the absolute values of the poles are less than one.

For two-dimensional filters, the stability condition is given as

\[
\sum_{m=0}^{\infty} \sum_{n=0}^{\infty} |h(mA, nB)| < \infty \quad (1.20)
\]
As far as literature surveys reveal [1] - [6], theorems relating to necessary and sufficient conditions for the stability of two-dimensional recursive filters may be related to the following two theorems.

(a) **SHANKS STABILITY THEOREM:**

A causal recursive filter with the z-transform

\[ H(z_1, z_2) = \frac{A(z_1, z_2)}{B(z_1, z_2)}, \]

where A and B are polynomials in \( z_1 \) and \( z_2^* \), is stable if, and only if, there are no values of \( z_1^* \) and \( z_2 \) such that \( B(z_1, z_2) = 0 \) for \( |z_1| < 1 \) and \( |z_2| < 1 \).

(b) **ANSELL STABILITY THEOREM:**

Let:

\[ H(z_1, z_2) = \frac{A(z_1, z_2)}{B(z_1, z_2)} \quad (1.21) \]

where, A and B are polynomials in \( z_1 \) and \( z_2 \). Make the change of variables

\[ p_1 = \frac{1-z_1}{1+z_1} \quad (1.22a) \]

and

\[ p_2 = \frac{1-z_2}{1+z_2} \quad (1.22b) \]

Hence, (1.21) becomes, after gathering terms of equal powers in \( p_1 \) and \( p_2 \).

\[ H(p_1, p_2) = \frac{E(p_1, p_2)}{F(p_1, p_2)} \quad (1.23) \]

where, \( E \) and \( F \) are polynomials in \( p_1 \) and \( p_2 \). Ansell's theorem

\[ \dagger \quad z_1 \text{ and } z_2 \text{ are treated here as delays, i.e., } z_1 = e^{-s_1A_1}, \]

\[ z_2 = e^{-s_2A_2}. \]
can now be stated as follows:

"The causal recursive filter $H(z_1, z_2)$ is stable if and only if: 1) for all real finite $w$, the complex polynomial in $p_2$, $F(jw, p_2)$ has no zeros in $\text{Re } p_2 > 0$; and 2) the real polynomial in $p_1$, $F(p_1, 1)$ has no zeros in $\text{Re } p_1 > 0$.

For 'minimum phase' [3], $A(z_1, z_2)$ should also satisfy either theorems.

1.6 MOTIVATION:

Two-dimensional digital filtering has traditionally been accomplished by convolving data with a finite spatial impulse response of the required filter. The advent of high speed convolution via the FFT algorithm has made this process fairly efficient. It would seem, however, that a two-dimensional recursive filter could offer even more computational advantages than high speed convolution.

Although some work has been reported in the literature on two-dimensional recursive filters (see next section), there is still a great deal of potential work to be done.

It was decided to take a closer look at three main areas in one-dimensional digital filtering, and then extend the ideas to two-dimensional digital filtering. The three main areas to be investigated are:

2) Design to approximate a given impulse-response.

3) Design of stable recursive filters to approximate desired magnitude-frequency specifications.

1.7 LITERATURE SURVEY:

1.7a ANALYSIS OF RECURSIVE FILTERS:

This section covers the generation of the impulse response of recursive digital filters, (or inverse z-transform), which for the one-dimensional case can be approached in three ways [8]:

(i) Long division of denominator into numerator yielding the original infinite series.

(ii) Use of the Heaviside expansion operation, involving partial fractions, to break the single rational form into a summation of simpler forms.

(iii) Evaluation of the inverse integral via the summation of residues.

Based on state space techniques, Chen and Shieh [9], developed a matrix formula for the inverse z-transform. The formula is in a closed form, and therefore is more complete than the results obtained by method (i). Their approach is computer oriented, and therefore is easier to use than methods (ii) or (iii).

1.7b SYNTHESIS OF RECURSIVE FILTERS:

(i) TO APPROXIMATE PRESCRIBED IMPULSE RESPONSE:

For the one-dimensional recursive filters, the time-domain synthesis involves finding the coefficients
\{a_i\}_{i=0}^{M_a}, \{b_i\}_{i=0}^{M_b}\) of the z-transform transfer function given by (1.11), such that its pulse-response best approximates a prescribed pulse response. The least squares solution to this problem, by direct minimization of the approximation error as a function of the filter coefficients, requires the solution of a set of nonlinear equations. Several papers have suggested using nonlinear programming methods such as the least square Taylor method [10] and the Gauss-Newton method [11]. Steiglitz and McBride [12] showed that this problem can be reduced to the repeated solution of a related linear problem. In their technique they minimize a "weighted" squared-error in the time-domain, which results in solving sets of linear simultaneous equations in terms of both the numerator and denominator. They also introduce two iterative linear refinement schemes, which results in a decrease in mean-square error.

Evans and Fishl [13] derived a relationship for the error corresponding to an optimal set of numerator coefficients, for a given set of denominator coefficients. The optimal denominator coefficients are calculated by an algorithm that iteratively solves weighted sets of linear equations in terms of the denominator coefficients only.

The common approach between these methods is that the error minimized is formed in the time-domain. [14] - [17] describe other time-domain approaches to this problem.
For the two-dimensional recursive filter, the spatial-domain synthesis involves finding the coefficients \( \{a_{ij}\}_{i=0}^{M_a}, j=0^{M_a} \) and \( \{b_{ij}\}_{i=0}^{M_b}, j=0^{N_b} \) of the z-transform function given by (1.15), such that its spatial pulse response best approximates a prescribed spatial pulse response. Shanks et al [3] extended to the two-dimensional filter, the synthesis technique in one-dimension given in [14], [15]. This is a space-domain approach.

(ii) TO APPROXIMATE DESIRED MAGNITUDE SPECIFICATIONS IN THE FREQUENCY DOMAIN:

For the one-dimensional digital filter, analytical procedures akin to that of Butterworth and Chebyshev continuous filter design procedures, have been developed. Other techniques include the use of conformal mapping to transform a digital filter design problem to a continuous filter problem. Some of these techniques are listed in [1].

Analytical procedures are confined to the design of certain classes of magnitude-frequency specifications. However, with arbitrary magnitude-frequency specifications, iterative techniques have been developed. An algorithm developed by Steiglitz [18] employs the minimization of a square-error criterion using the unconstrained Fletcher-Powell conjugate-gradient minimization routine [19], [20]. In this algorithm, the poles are examined after a certain number of iterations, and, if necessary, moved inside the unit circle to ensure a stable design. Another approach [21]-[23] which also uses the same minimization routine to
minimize a square-error criterion, constrains the poles, to produce a stable recursive filter, by using a variable transformation method. This algorithm is described briefly in Chapter IV of this thesis.

Shanks et al [3], introduced a technique for designing two-dimensional recursive filters. In their technique they take one-dimensional filters and convert them into two-dimensional recursive filters with arbitrary directivity in a two-dimensional frequency response plane. Their technique, however, exercises little control over the boundary shape, (between "passband" and "stopband"), and does not guarantee the stability of the designed filter.

1.8 THESIS ORGANIZATION:

In Chapter II, a method is developed for evaluating the impulse-response of a one-dimensional linear recursive filter in closed form. The extension of the analysis, used in this method, is applied to the two-dimensional linear recursive filter, and leads to a technique by which the spatial impulse response can be generated by a set of closed form solutions. Necessary conditions for the stability of two-dimensional recursive filters are also derived. A numerical method for calculating the impulse response for either one- or two-dimensional recursive filters, using pruned FFT algorithms, is also presented.

In Chapter III, a technique is developed for identifying the parameters of a linear recursive filter.
from a given impulse response. The technique is first presented for the one-dimensional recursive filter and then extended to linear spatial recursive filters.

In Chapter IV, a technique is presented for designing stable recursive filters. The method described in [21]-[23] for the one-dimensional recursive filter design is first presented. Then, it is extended to the design of two-dimensional recursive filters.
CHAPTER II

GENERATION OF THE IMPULSE RESPONSE

2.1 ANALYTICAL INVERSION OF THE z-TRANSFORM:

A method is developed for obtaining the closed form solution of the inverse z-transform of a rational function of polynomials in z. This method is based on obtaining a general difference equation for the rational form, which can be expressed in a state-variable format. The solution of this equation, with initial-conditions determined from the coefficients of the rational form, is expressed in closed form using the Cayley-Hamilton theorem \[8\]. Results of a digital computer programme, written to carry out the required computations, are shown.

2.1.1 ANALYSIS:

Consider the following rational form in the complex variable z:

\[
Y(z) = \sum_{n=0}^{\infty} Y_n z^{-n} = \sum_{i=0}^{N} \frac{a_i z^{-i}}{\sum_{i=0}^{M} b_i z^{-i}}
\] (2.1)

For the moment we will consider \( N < M \).

Equation (2.1) can be expanded as an infinite series in z:
\[ Y(z) = y_0 + y_1 z^{-1} + y_2 z^{-2} + \ldots + y_M z^{-M} \]
\[ + \ldots + y_{M+i} z^{-M-i} + \ldots \]  \hspace{1cm} (2.2)

Equating (2.1) and (2.2), and cross-multiplying we get

\[ \sum_{i=0}^{N} a_i z^{-i} = \left\{ \sum_{i=0}^{M} b_i z^{-i} \right\} \left\{ \sum_{i=0}^{m} y_i z^i \right\} \]  \hspace{1cm} (2.3)

By equating terms of equal powers in \( z \), we obtain the following relations:

\[ a_0 = b_0 y_0 \]
\[ a_1 = b_1 y_0 + b_0 y_1 \]
\[ a_2 = b_2 y_0 + b_1 y_1 + b_0 y_2 \]
\[ \vdots \]

\[ a_N = b_N y_0 + b_{N-1} y_1 + b_{N-2} y_2 + \ldots + b_0 y_N \]  \hspace{1cm} (2.4)

\[ 0 = b_{N+1} y_0 + b_N y_1 + \ldots + b_0 y_{N+1} \]

\[ \vdots \]

\[ 0 = b_M y_0 + b_{M-1} y_1 + \ldots + b_0 y_M \]

and,

\[ 0 = b_M y_i + b_{M-1} y_{i-1} + \ldots + b_0 y_{M+i} \]  \hspace{1cm} (2.5)

From (2.4) the following relations are obtained
\[ y_0 = \frac{a_0}{b_0} \]

\[ y_i = \frac{a_i - \sum_{j=0}^{i-1} b_{i-j} y_j}{b_0} \quad \text{for } 1 < i < N \]  

\[ y_i = -\frac{\sum_{j=0}^{i-1} b_{i-j} y_j}{b_0} \quad \text{for } N < i < M \]  

(2.6)

Equation (2.5) can be written in the form:

\[ y_{M+i} = -\frac{b_1}{b_0} y_{M+i-1} - \frac{b_2}{b_0} y_{M+i-2} - \cdots - \frac{b_M}{b_0} y_i \]  

(2.7)

and can be represented by the state-variable diagram shown in Fig. 2.1. (This representation was also obtained by Chen and Shieh via a different approach [9])

![State Variable Diagram](image)

Figure 2.1. State Variable Diagram of Equation (2.7).
From Figure 2.1 we write the following relations

\[
\begin{bmatrix}
x_1(i+1) \\
x_2(i+1) \\
\vdots \\
x_M(i+1)
\end{bmatrix} = \begin{bmatrix}
0 & 1 & 0 & \cdots & 0 \\
0 & 0 & 1 & \cdots & 0 \\
\vdots & \ddots & \ddots & \ddots & \vdots \\
-b_M & -b_{M-1} & \cdots & b_1 & b_0 \\
\end{bmatrix} \begin{bmatrix}
x_1(i) \\
x_2(i) \\
\vdots \\
x_M(i)
\end{bmatrix}
\]

or in compact form

\[
X(i+1) = A X(i)
\]

which has a solution

\[
X(i) = A^i X(0)
\]

where:

\[
X(0) = \begin{bmatrix}
y_0 \\
y_1 \\
\vdots \\
y_{M-1}
\end{bmatrix}
\]

the \(y_i\)'s are given by equation (2.6) for \(0 \leq i \leq M\).

Let us now consider the case \(N = M\).

Since \(a_M \neq 0\), equation (2.5) is satisfied for \(i > 0\) and not for \(i = 0\).

This leads to a modification of the solution given by (2.10). Thus:
\[
X(i) = A^{i-1} X(1)
\]

(2.11)

where:
\[
X(1) = \begin{bmatrix}
y_1 \\
y_2 \\
\vdots \\
y_M 
\end{bmatrix}
\]

(2.12)

Note that for \(N < M\), equation (2.10) solves for all values of the output \(y(i)\) \(i = 0, 1, \ldots, \infty\), whereas for \(N = M\), equation (2.11) does not solve for the first value, i.e., equation (2.11) solves for \(y(i)\) \(i = 1, 2, \ldots\). This is of little consequence, since \(y_0\) is easily computed from \(y_0 = a_0/b_0\).

2.1.2 THE CLOSED FORM SOLUTION:

From the Cayley-Hamilton technique for finding the function of a matrix \([8]\), \(A^i\) can be written as:

\[
A^i = a_0 I + a_1 A + a_2 A^2 + \ldots + a_{M-1} A^{M-1}
\]

(2.13)

where the \(\{a_i\}\) \(i = 0, 1, \ldots, M-1\), are obtained from the eigenvalues of \(A(\{\lambda_i\}, i=1, 2, \ldots, M)\), as follows:

(a) Case of Distinct Eigenvalues:

\[
\begin{bmatrix}
\lambda_1^i \\
\lambda_2^i \\
\vdots \\
\lambda_M^i
\end{bmatrix} =
\begin{bmatrix}
1 & \lambda_1 & \lambda_1^2 & \ldots & \lambda_1^{M-1} \\
1 & \lambda_2 & \lambda_2^2 & \lambda_2^{M-1} \\
\vdots & \vdots & \vdots & \ddots & \vdots \\
1 & \lambda_M & \lambda_M^2 & \lambda_M^{M-1}
\end{bmatrix}
\begin{bmatrix}
a_0 \\
a_1 \\
\vdots \\
a_{M-1}
\end{bmatrix}
\]

(2.14)
(b) Case of Multiple Eigenvalues:

We may form the required extra equations by repeated differentiation.

Consider, for example, that \( \lambda_1 \) has an eigenvalue \( \lambda_1 \) of multiplicity 3, and the rest of the eigenvalues are distinct, then:

\[
\begin{bmatrix}
\lambda_1^i \\
\lambda_1^{i-1} \\
i\lambda_1^{i-2} \\
\vdots \\
\lambda_1 \\
\end{bmatrix}
= \begin{bmatrix}
1 & \lambda_1 & \lambda_1^2 & \cdots & \lambda_1^{M-1} \\
0 & 1 & 2\lambda_1 & \cdots & (M-1)\lambda_1^{M-2} \\
0 & 0 & 2 & \cdots & (M-1)(M-2)\lambda_1^{M-3} \\
1 & \lambda_4 & \lambda_4^2 & \cdots & \lambda_4^{M-1} \\
1 & \lambda_5 & \lambda_5^2 & \cdots & \lambda_5^{M-1} \\
\vdots & \vdots & \vdots & \ddots & \vdots \\
1 & \lambda_M & \lambda_M^2 & \cdots & \lambda_M^{M-1}
\end{bmatrix}
\begin{bmatrix}
a_0 \\
a_1 \\
a_2 \\
\vdots \\
a_{M-1}
\end{bmatrix}
\]

(2.15)

Since we are only interested in \( X_1(i) \), we note that:

1st row of \( A \) = \( \{0 \ 1 \ 0 \ 0 \ 0 \ \cdots \ 0\} \)

1st row of \( A^2 \) = \( \{0 \ 0 \ 1 \ 0 \ 0 \ \cdots \ 0\} \)

1st row of \( A^3 \) = \( \{0 \ 0 \ 0 \ 1 \ 0 \ \cdots \ 0\} \)

1st row of \( A^{M-1} \) = \( \{0 \ 0 \ 0 \ 0 \ \cdots \ 1\} \)

Hence, the 1st row of \( A^i \) is

\[
(a_0 \ a_1 \ a_2 \ \cdots \ a_{M-1})
\]

(2.16)
Thus:

\[ x_1(i) = y_1 = a_0y_0 + a_1y_1 + a_2y_2 + \ldots \]

\[ \ldots + a_{M-1}y_{M-1} \]  \hspace{1cm} (2.17)

Equation (2.17) is the closed form solution required.

2.1.3 **EXAMPLE:**

As an illustration consider the following example

\[ Y(z) = \frac{e^{-aT}(\sin bT)z^{-1}}{1-2e^{-aT}(\cos bT)z^{-1} + e^{-2aT}z^{-2}} \]

Initial conditions obtained from equation (3.6) are

\[ y_0 = 0 \]

\[ y_1 = e^{-aT}\sin bT \]

The eigenvalues of \( \lambda \) are given by the poles

\[ \lambda_{1,2} = e^{-aT}(\cos bT \pm j\sin bT) \]

The \( \{a_i\} \) \( i = 0, 1 \) are calculated from

\[
\begin{bmatrix}
a_0 \\ a_1
\end{bmatrix} =
\begin{bmatrix}
1 & \lambda_1 \\
1 & \lambda_2
\end{bmatrix}^{-1}
\begin{bmatrix}
\lambda_1^k \\ \lambda_2^k
\end{bmatrix}
\]
\[ a_0 = \frac{\lambda_2 \lambda_1^k}{\lambda_2 - \lambda_1} - \frac{\lambda_1 \lambda_2^k}{\lambda_2 - \lambda_1} \]

\[ a_1 = -\frac{\lambda_1^k + \lambda_2^k}{\lambda_2 - \lambda_1} \]

The closed form solution is determined from (2.17)

\[ y(iT) = a_0 y(0T) + a_1 y(lT) \]

\[ = \frac{\lambda_1^k - \lambda_2^k}{\lambda_2 - \lambda_1} e^{-aT} \sin bt \]

\[ = \frac{2je^{-kaT} \sin kbT}{2je^{-aT} \sin bt} e^{-aT} \sin bt \]

\[ = e^{-kaT} \sin kbT \]

2.1.4 Computational Details:

The computation of the eigenvalues of the A matrix is very straightforward since \( \dot{A} \) is in the form of a companion matrix \([24]\). We therefore simply find the roots of the polynomial

\[ \lambda^M + \frac{b_1}{b_0} \lambda^{M-1} + \ldots + \frac{b_M}{b_0} = 0 \]  \quad (2.18)
The other computational details are the inversion of the $M \times M$ matrix of equation (2.14) (or 2.15) and the generation of the initial conditions from (2.6). All of the computations can be performed easily and quickly on a digital computer for fairly large orders of systems. The technique is suitable for discrete system analysis and also affords a method of generating the impulse response without recursion and associated quantization noise.

2.1.5 EXTENSION TO LAPLACE TRANSFORM:

Consider the following transfer function

\[ Y(s) = \frac{b_1 s^{M-1} + b_2 s^{M-2} + \ldots + b_M}{s^M + a_1 s^{M-2} + a_2 s^{M-3} + \ldots + a_M} \]  

(2.19).

Chen and Yates [25] developed a homogeneous state equation from (2.19)

\[ \dot{x} = Ax \]  

(2.20)

or

\[ x(t) = e^{At} x(0) \]  

(2.21)

where

\[ A = \begin{bmatrix}
0 & 1 & 0 & 0 & 0 \\
0 & 0 & 1 & 0 & 0 \\
\vdots & \vdots & \vdots & \ddots & \vdots \\
0 & 0 & 0 & \ldots & 1 \\
-a_M & -a_{M-1} & -a_{M-2} & \ldots & -a_1
\end{bmatrix} \]  

(2.22)
and;

\[
x(0) = \begin{bmatrix}
    y(0) \\
    y'(0) \\
    y''(0) \\
    \vdots \\
    y^{(M-1)}(0)
\end{bmatrix}
\]

Equation (2.23) is calculated from

\[
\begin{bmatrix}
b_1 \\
b_2 \\
b_3 \\
\vdots \\
b_M
\end{bmatrix} = \begin{bmatrix}
1 & 0 & 0 & \cdots & 0 \\
0 & 1 & 0 & \cdots & 0 \\
0 & 0 & 1 & \cdots & 0 \\
\vdots & \vdots & \vdots & \ddots & \vdots \\
a_{M-1} & a_{M-2} & a_{M-3} & \cdots & 1
\end{bmatrix} \begin{bmatrix}
y(0) \\
y'(0) \\
y''(0) \\
\vdots \\
y^{(M-1)}(0)
\end{bmatrix}
\]

(2.24)

In their solution [25] a matrix transformation is used on equation (2.20) to produce a Heaviside expansion expression of the given transfer function, from which an analytical solution results.

If we adopt the same procedure given in Sec. 2.1.3, we get

\[
y(t) = a_0 y(0) + a_1 y'(0) + \cdots + a_{M-1} y^{(M-1)}(0)
\]

(2.25)
where \( \{a_i\}_{i=0,1,...,M-1} \) are calculated as in equation (2.14) and (2.15) except now the L.H. column in equation (2.14) changes to

\[
\begin{bmatrix}
e^{\lambda_1 t} \\
e^{\lambda_2 t} \\
\vdots \\
e^{\lambda_M t}
\end{bmatrix}
\quad (2.26)
\]

and in equation (2.15) to

\[
\begin{bmatrix}
e^{\lambda_1 t} \\
t e^{\lambda_1 t} \\
t^2 e^{\lambda_1 t} \\
e^{\lambda_4 t} \\
e^{\lambda_5 t} \\
\vdots \\
e^{\lambda_M t}
\end{bmatrix}
\quad (2.27)
\]

If any of the eigenvalues have multiplicities greater than one, then there are great computational advantages in using the method described here rather than that of Chen and Shieh.
2.1.6 COMPUTER PROGRAMME:

A digital computer programme has been written to perform the calculations outlined for the inverse \( z \)-transform, and is given in Appendix A. The output is produced in a reasonably condensed form, and some examples are given below.

(i)

\[
\begin{align*}
\{ & \frac{1 - e^{-at}(\cos bT)z^{-1}}{1 - 2e^{-at}(\cos bT)z^{-1} + e^{-2at}z^{-2}} \\
& = \{e^{-at}\} \cos(kbT) \\
\end{align*}
\]

Let \( e^{-at} = 0.2 \)

\( \cos bT = 0.5 \)

The solution is \( (0.2)^k \cos(1.05k) \).

The computer printout is given in fig. 2.2.

(ii)

\[
\begin{align*}
\{ & \frac{e^{-at}(\sin bT)z^{-1}}{1 - 2e^{-at}(\cos bT)z^{-1} + e^{-2at}z^{-2}} \\
& = \{e^{-at}\} \sin(kbT) \\
\end{align*}
\]

Let \( e^{-at} = 0.2 \)

\( \cos bT = 0.5 \)

The solution is \( (0.2)^k \sin(1.05k) \).

The computer printout is given in fig. 2.3.
(iii)

\[ z^{-1} \left\{ \frac{0.125z(z+1)}{(z-1)^3} \right\} = 0.125 \ k^2 \]

The computer printout is given in fig. 2.4, and this can be further simplified to the solution

\[ y(k) = 0.125k + 0.125(k(k-1)) = 0.125 \ k^2 \]

(iv)

\[ z^{-1} \left\{ \frac{T^4}{4!} \frac{z(z^3+1) + 11z(z+1)}{(z-5)^5} \right\} = \frac{(kT)^4}{4!} \]

Let \( \frac{T^4}{4!} = 0.417 \times 10^{-4} \)

Hence;

\[ z^{-1} \left\{ 0.417 \times 10^{-4} \frac{z^{-1} + 11z^{-2} + 11z^{-3} + z^{-4}}{1 - 5z^{-1} + 10z^{-2} - 10z^{-3} + 5z^{-4} + z^{-5}} \right\} = 0.417 \times 10^{-4} \times k^4 \]

The computer printout is given in fig. 2.5, and this can be further simplified to the solution:

\[ y(k) = 0.414 \times 10^{-4} \times k^4 \]
THE GIVEN Z-TRANSFORM IS:

\[ Y(z) = (A(1) + A(2) z^{-1} + A(3) z^{-2} + \ldots + A(N+1) z^{-N-1}) / (B(1) + B(2) z^{-1} + B(3) z^{-2} + \ldots + B(M+1) z^{-M}) \]

WHERE:

\[ N = 1 \quad M = 2 \]

\[ A(1) = 0.1000 \quad 01 \]
\[ A(2) = 0 \quad 00 \]
\[ A(3) = -0.1000 \quad 00 \]
\[ A(4) = 0 \quad 00 \]
\[ A(5) = 0 \quad 00 \]
\[ A(6) = 0 \quad 00 \]

THE INVERSE Z-TRANSFORM IS GIVEN BY:

\[ y(kT) =
\begin{align*}
&0.5000 \quad 00 + J \quad 0.4166 \quad -16 \pi \left( 0.1000 \quad 00 + J \quad 0.1730 \quad 00 \right) \ast K \\
&+ 0.5000 \quad 00 + J \quad 0.4166 \quad -16 \pi \left( 0.1200 \quad 00 + J \quad 0.1730 \quad 00 \right) \ast K
\end{align*} \]

WHICH COULD BE COMPRESSED TO THE FORM

\[ 0.1000 \quad 01 + (1 \cdot 0.2730 \quad 00) \ast K \ast \cos(-0.1080 \quad 01 \ast K) \]

Fig. 2.2. Computer Printout of the Inverse z-transform of example (i) given in sec. 2.1.6.
The given Z-transform is:
\[ Y(z) = \frac{A(1) + A(2)/z + A(3)/z^2 + \ldots + A(N+1)/z^{N+1}}{B(1) + B(2)/z + B(3)/z^2 + \ldots + B(N+1)/z^{N+1}} \]

WHERE:

\[ N = 1 \quad M = 2 \]

\[ A(1) = 0.0 \quad B(1) = 0.1000 \quad 01 \]
\[ A(2) = 0.1730 \quad 00 \quad B(2) = 0.2000 \quad 00 \]
\[ B(3) = 0.4000 \quad 01 \]

POLES:
\[ 0.1000 \quad 00 + J \cdot 0.1730 \quad 00 \]
\[ 0.1000 \quad 00 + J \cdot 0.1730 \quad 00 \]

MULTIPLICITIES:
\[ 1 \]

The inverse Z-transform is given by:
\[ y(k) = \begin{cases} 0.0 & + J \cdot 0.4990 \cdot 00 \cdot (0.1000 \quad 00 + J \cdot 0.1730 \quad 00) \cdot k \cdot k \\ 0.0 & + J \cdot -0.4990 \cdot 00 \cdot (0.1000 \quad 00 + J \cdot 0.1730 \quad 00) \cdot k \cdot k \end{cases} \]

which could be condensed to the form
\[ y(k) = -0.9990 \cdot 00 \cdot (0.2000 \quad 00) \cdot k \cdot k \times \sin(-0.1050 \quad 01 \cdot k) \]

Fig. 2.3. Computer Printout of the Inverse z-transform of example (ii) given in sec. 2.1.6.
THE GIVEN Z-TRANSFORM IS:
\[ Y(Z) = \frac{A(1) + A(2)/z + A(3)/z^2 + \ldots + A(N+1)/z^{N+1}}{B(1) + B(2)/z + B(3)/z^2 + \ldots + B(M+1)/z^{M+1}} \]
WHERE:
\[ N = 2 \quad M = 3 \]
\[ A(1) = 0.1 \quad B(1) = 0.2000 \quad B(2) = -0.0000 \quad B(3) = 0.0000 \quad B(4) = -0.2000 \]

THE INVERSE Z-TRANSFORM IS GIVEN BY:
\[ Y(k) = \]
\[ + 0.0 + J 0.0 \quad 0.1000 \quad 0.3280 \quad 0.3280 \quad 0.3280 \]
\[ + 0.1250 + J 0.0 \quad 0.1000 \quad 0.1000 \quad 0.1000 \]
\[ + 0.1250 + J 0.0 \quad 0.1000 \quad 0.1000 \quad 0.1000 \]

WHICH COULD BE EXPRESSED IN THE FORM
\[ +\frac{\text{FACT}(k)}{\text{FACT}(k-1)} \times 0.1250 \quad 0.1000 \quad 0.3280 \quad 0.3280 \quad 0.3280 \]

**Fig. 2.4.** Computer Printout of the Inverse z-transform of example (iii) given in sec. 2.1.6.
THE GIVEN Z-TRANSFORM IS:

\[ Y(z) = \frac{(A(1) + A(2)/z + A(3)/z^2 + \ldots + A(N+1)/z^{N+1})}{B(1) + B(2)/z + B(3)/z^2 + \ldots + B(N+1)/z^{N+1}} \]

WHERE:

\[ N = 4 \]
\[ \alpha = 0.5 \]

A(1) = 0.0
A(2) = 0.4170-14
A(3) = 0.4589-13
A(4) = 0.4589-13
A(5) = 0.4170-14

B(1) = 0.1000 01
B(2) = -0.5000 01
B(3) = 0.1000 02
B(4) = -0.1000 02
B(5) = 0.5000 01
B(6) = -0.1000 01

POLES

0.1000 01 + J 0.0
MULTIPLICITIES

5

THE INVERSE Z-TRANSFORM IS GIVEN BY:

\[ y(k) = \frac{1}{2\pi J} \int_{-\infty}^{\infty} \left[ \sum \frac{A(n)}{\text{FACT}(n)} \right] \cdot \frac{1}{(0.1000 01 + J 0.0)^k} \, dk \]

\[ + \frac{1}{2\pi J} \int_{-\infty}^{\infty} \left[ \sum \frac{B(n)}{\text{FACT}(n)} \right] \cdot \frac{1}{(0.1000 01 + J 0.0)^k} \, dk \]

 WHICH COULD BE CONDENSED TO THE FORM:

\[ \frac{1}{\text{FACT}(k-1)} \cdot 0.4170-14 \cdot \left( \frac{0.1000 01}{(0.1000 01 + J 0.0)^k} \right) \]

\[ + \frac{1}{\text{FACT}(k-2)} \cdot 0.2929-13 \cdot \left( \frac{0.1000 01}{(0.1000 01 + J 0.0)^k} \right) \]

\[ + \frac{1}{\text{FACT}(k-3)} \cdot 0.2500-13 \cdot \left( \frac{0.1000 01}{(0.1000 01 + J 0.0)^k} \right) \]

\[ + \frac{1}{\text{FACT}(k-4)} \cdot 0.4140-14 \cdot \left( \frac{0.1000 01}{(0.1000 01 + J 0.0)^k} \right) \]

Fig. 2.5. Computer-Printout of the Inverse z-transform of example (iv) given in sec. 2.1.6.
2.2 GENERATION OF THE IMPULSE RESPONSE OF A TWO-DIMENSIONAL
RECURSIVE DIGITAL FILTER BY A SET OF CLOSED FORM
SOLUTIONS:

This section presents a method for forming a set of closed form solutions to generate partial and full impulse responses of two-dimensional recursive digital filters. Each closed form solution generates a complete line in the impulse response. The analysis is carried out for the generation of both horizontal and vertical lines. Some stability properties are derived.

2.2.1 INTRODUCTION:

A two-dimensional filter is described as a ratio of two-dimensional polynomials in \((z_1, z_2)\). Consider the filter

\[ H(z_1, z_2) = \frac{\sum_{i=0}^{M_a} \sum_{j=0}^{N_a} a_{ij} z_1^{-i} z_2^{-j}}{\sum_{i=0}^{M_b} \sum_{j=0}^{N_b} b_{ij} z_1^{-i} z_2^{-j}} \tag{2.30} \]

where: \(b_{00} \neq 0\), \(z_1 = e^{S_1 A}\), \(z_2 = e^{S_2 B}\), \(M_a \leq M_b\), and \(N_a \leq N_b\).

The planar pulse response of the filter is given by the recursive equation:
\[ h(m,n) = - \sum_{k=0}^{M_b} \sum_{\ell=0}^{N_b} \frac{b_{k\ell}}{b_{00}} h(m-k, n-\ell) \]
\[ + \sum_{i=0}^{M_a} \sum_{j=0}^{N_a} a_{ij} b_{00} \delta(m-i, n-j) \] (2.31)

where: \[ \delta(m,n) = 1, \quad m = 0, \quad n = 0 \]

\[ \delta(m,n) = 0, \quad \text{otherwise} \]

and zero initial conditions are assumed.

If the transfer function \( H(z_1, z_2) \) is rearranged to yield a polynomial in one complex variable with coefficients as a function of the other complex variable, then it is possible to use an extension of the one-dimensional technique explained previously to obtain closed form solutions.

2.2.2 CLOSED FORM SOLUTIONS FOR THE GENERATION OF THE HORIZONTAL LINES:

Equation (2.30) can be written in the form

\[ H(z_1, z_2) = \frac{f_0(z_2) + z_1^{-1} f_1(z_2) + z_1^{-2} f_2(z_2) + \ldots + z_1^{-M_a} f_{M_a}(z_2)}{r_0(z_2) + z_1^{-1} r_1(z_2) + z_1^{-2} r_2(z_2) + \ldots + z_1^{-M_b} r_{M_b}(z_2)} \] (2.32)

where:

\[ f_i(z_2) = \sum_{j=0}^{N_a} a_{ij} z_2^{-j} \] (2.33)

\[ r_i(z_2) = \sum_{j=0}^{N_b} b_{ij} z_2^{-j} \]
Equation (2.32) can be expressed as an infinite series in $z_1$, thus:

\[ H(z_1, z_2) = d_0(z_2) + z_1^{-1} d_1(z_2) + z_1^{-2} d_2(z_2) + \ldots \]

\[ + z_1^{-M_b} d_{M_b}(z_2) + z_1^{-(M_b+1)} d_{M_b+1}(z_2) + \ldots \quad (2.34) \]

where the $d_i$'s are functions of $z_2$ only, and are given in terms of $h(i,j)$ by the relation

\[ d_i(z_2) = \sum_{k=0}^{\infty} h(i,k) z_2^{-k} \quad (2.35) \]

Equating equation (2.32) to (2.34) and cross-multiplying, we get:

\[ f_0(z_2) + z_1^{-1} f_1(z_2) + \ldots + z_1^{-M} f_M(z_2) \]

\[ = \left\{ r_0(z_2) + z_1^{-1} r_1(z_2) + z_1^{-2} r_2(z_2) + \ldots \right\} \times \left\{ d_0(z_2) + z_1^{-1} d_1(z_2) \right. \]

\[ + z_1^{-M_b} r_{M_b}(z_2) \}

\[ + \left. z_1^{-2} d_2(z_2) + \ldots \right\} \quad (2.36) \]

By equating terms of equal power in $z_1$, we obtain the following relations:
\[ d_0(z_2) = \frac{f_0(z_2)}{r_0(z_2)} \quad (2.37a) \]

For \( 1 \leq i \leq M_a \)

\[ d_i(z_2) = \frac{f_i(z_2)}{r_0(z_2)} - \frac{\sum_{j=0}^{i-1} r_{i-j}(z_2) d_j(z_2)}{r_0(z_2)} \quad (2.37b) \]

For \( M_a < i \leq M_b \)

\[ d_i(z_2) = \frac{f_i(z_2)}{r_0(z_2)} - \sum_{j=0}^{i-1} r_{i-j}(z_2) d_j(z_2) \quad (2.37c) \]

\( i \geq M_b + 1 \)

\[ d_i(z_2) = \frac{f_i(z_2)}{r_0(z_2)} - \sum_{j=0}^{M_b-1} r_{M_b-j}(z_2) d_{i-M_b+j}(z_2) \quad (2.37d) \]

Equations (2.37a) - (2.37d) express the z-transform of the \( i^{th} \) horizontal line as a function in one complex variable \( z_2 \). By examining these equations it is obvious that the order of \( d_i(z_2) \) increases with \( i \). The analytical expression for the inverse of \( d_i(z_2) \) can be obtained by using one-dimensional techniques, such as the inverse integral or the method described in sec. 2.1.
2.2.3 **CLOSED FORM SOLUTIONS FOR THE GENERATION OF THE VERTICAL LINES:**

It is also possible to generate the spatial impulse response by generating the impulse response of all the vertical lines.

Equation (2.30) can also be written in the form:

\[ H(z_1, z_2) = \frac{g_0(z_1) + z_2^{-1} g_1(z_1) + \ldots + z_2^{-Na} g_{Na}(z_1)}{p_0(z_1) + z_2^{-1} p_1(z_1) + \ldots + z_2^{-Nb} p_{Nb}(z_1)} \]  

(2.38)

where:

\[ g_i(z_1) = \sum_{j=0}^{M_a} a_{ji} z_1^{-j} \]

(2.39)

\[ p_i(z_1) = \sum_{j=0}^{M_b} b_{ji} z_1^{-i} \]

Proceeding as before and denoting \( v_i(z_1) \) as the \( z \)-transform of the \( i \)th vertical line, i.e.,

\[ v_i(z_1) = \sum_{k=0}^{\infty} h(k,i) z_1^{-k} \]

we obtain the following relations:

\[ v_0(z_1) = \frac{g_0(z_1)}{p_0(z_1)} \]  

(2.41a)
for: $1 \leq i \leq N_a$

$$v_i(z_1) = \frac{g_i(z_1) - \sum_{j=0}^{i-1} p_{i-j}(z_1) v_j(z_1)}{p_0(z_1)}$$

(2.41b)

for: $N_a < i \leq N_b$

$$v_i(z_1) = \frac{- \sum_{j=0}^{i-1} p_{i-j}(z_1) v_j(z_1)}{p_0(z_1)}$$

(2.41c)

for: $i > N_b + 1$

$$v_i(z_1) = \frac{- \sum_{j=0}^{N_b-1} p_{N_b-j}(z_1) v_{i-N_b+j}}{p_0(z_1)}$$

(2.41d)

Hence: $v_i(k) = z^{-1} \{v_i(z_1)\}$

2.2.4 NECESSARY CONDITIONS FOR THE STABILITY OF TWO-DIMENSIONAL RECURSIVE FILTERS:

From the previous analysis two necessary conditions of stability are derived. These conditions are:

1) The zeros of $r_0(z_2)$ should lie within the region $|z_2| < 1$.

2) The zeros of $p_0(z_1)$ should lie within the region $|z_1| < \Gamma$.

These conditions are not sufficient for the stability of two-dimensional recursive filters, as will be demonstrated in later examples.
2.2.5 GENERATION OF THE FIRST $N_b+1$ VERTICAL LINES

CLOSED FORM:

It is possible to obtain a single matrix expression for generating the first $N_b + 1$ vertical lines.

The equations given by (2.37) are written in the form:

\[
\begin{align*}
    f_0 &= r_0 d_0 \\
    f_1 &= r_1 d_0 + r_0 d_1 \\
    f_2 &= r_2 d_0 + r_1 d_1 + r_0 d_2 \\
    f_{M_a} &= r_{M_a} d_0 + r_{M_a-1} d_1 + r_{M_a-2} d_2 + \ldots + r_0 d_{M_a} \\
    0 &= r_{M_a+1} d_0 + r_{M_a} d_1 + \ldots + r_0 d_{M_a+1} \\
    0 &= r_{M_b} d_0 + r_{M_b-1} d_1 + \ldots + r_0 d_{M_b} \\
    0 &= r_{M_b} d_1 + r_{M_b-1} d_2 + \ldots + r_0 d_{M_b+1} \\
    0 &= r_{M_b} d_i + r_{M_b-1} d_{i+1} + \ldots + r_0 d_{M_b+i} \\
\end{align*}
\]

(2.42)

where; $f_i \equiv f_i(z_2)$, and $r_i \equiv r_i(z_2)$

(2.43)
Consider the 1st equation in (2.42)

\[ f_0 = r_0 d_0 \]

i.e.

\[ \sum_{i=0}^{N} a_0 i z_2^{-i} = \left[ \sum_{i=0}^{N} b_0 i z_2^{-i} \right] \cdots \left[ \sum_{k=0}^{\infty} h(0, k) z_2^{-k} \right] \quad (2.44) \]

By equating equal powers in \( z_2 \) we obtain the following relations:

\[
\begin{bmatrix}
a_{00} \\
a_{01} \\
a_{02} \\
\vdots \\
a_{0N_a} \\
0 \\
0 \\
0
\end{bmatrix}
= 
\begin{bmatrix}
b_{00} & b_{00} \\
b_{01} & b_{01} \\
b_{02} & b_{02} \\
\vdots & \vdots \\
b_{0N_b} & b_{0N_b-1} & \cdots & b_{00}
\end{bmatrix}
\begin{bmatrix}
h(0,0) \\
h(0,1) \\
h(0,2) \\
\vdots \\
h(0,N_b)
\end{bmatrix}
\quad (2.45)
\]

Now, consider the 2nd equation in (2.42)

\[ f_1 = r_1 d_0 + r_0 d_1 \]

or,
\[
\sum_{i=0}^{N_a} a_{1i} z_2^{-i} = \left( \sum_{i=0}^{N_b} b_{1i} z_2^{-i} \right) \times \left( \sum_{k=0}^{\infty} b(0,k) z_2^{-k} \right) 
+ \left( \sum_{i=0}^{N_b} b_{0i} z_2^{-i} \right) \times \left( \sum_{k=0}^{\infty} h(1,k) z_2^{-k} \right)
\] (2.46)

Equating coefficients of equal power in $z_2$ we obtain the following relation:

\[
\begin{bmatrix}
  a_{10} \\
  a_{11} \\
  a_{12} \\
  \vdots \\
  a_{1N_a}
\end{bmatrix} \begin{bmatrix}
  b_{10} \\
  b_{11} \\
  b_{12} \\
  \vdots \\
  b_{1N_b}
\end{bmatrix} = \begin{bmatrix}
  h(0,0) \\
  h(0,1) \\
  h(0,2) \\
  \vdots \\
  h(0,N_b)
\end{bmatrix}
\]

\[
\begin{bmatrix}
  b_{00} \\
  h_{01} \\
  b_{02} \\
  \vdots \\
  b_{0N_b}
\end{bmatrix} \begin{bmatrix}
  b_{00} \\
  b_{01} \\
  b_{02} \\
  \vdots \\
  b_{0N_b}
\end{bmatrix} = \begin{bmatrix}
  h(1,0) \\
  h(1,1) \\
  h(1,2) \\
  \vdots \\
  h(1,N_b)
\end{bmatrix}
\] (2.47)
Let us now define the following matrices:

\[
A_i = \begin{bmatrix}
a_{i0} \\
a_{i1} \\
\vdots \\
a_{iN_a} \\
0 \\
\vdots \\
0
\end{bmatrix}, \quad H_i = \begin{bmatrix}
h(i,0) \\
h(i,1) \\
\vdots \\
h(i,N_b)
\end{bmatrix}
\tag{2.48, 2.49}
\]

\[
B_i = \begin{bmatrix}
b_{i0} \\
b_{i1} & b_{i0} \\
b_{i2} & b_{i1} & b_{i0} \\
\vdots & \vdots & \ddots & \ddots \\
b_{iN_b} & b_{iN_b-1} & \cdots & b_{i0}
\end{bmatrix}
\tag{2.50}
\]

Equations (2.45) and (2.47) can now be written in the condensed forms:

\[
A_0 = B_0 \cdot H_0
\tag{2.51}
\]

and

\[
A_1 - B_1 H_0 = B_0 \cdot H_1
\tag{2.52}
\]

Since, \(b_{00} \neq 0\), \(B_0\) is an invertible matrix, and hence:

\[
H_0 = B_0^{-1} A_0
\tag{2.53a}
\]

\[
H_1 = B_0^{-1} (A_1 - B_1 H_0)
\]
Similarly, from the rest of the equations in (2.42) the following equations are derived:

\[ H_2 = B_0^{-1} (A_2 - B_2 H_0 - B_1 H_1) \]

\[ H_{M_a} = B_0^{-1} (A_{M_a} - B_{M_a} H_0 - B_{M_a-1} H_1 - \cdots - B_1 H_{M_a-1}) \]

\[ H_{M_a+1} = B_0^{-1} (B_{M_a+1} H_0 - B_{M_a} H_1 - \cdots - B_1 H_{M_a}) \]  \hspace{1cm} (2.53b)

\[ H_{M_b} = B_0^{-1} (-B_{M_b} H_0 - B_{M_b-1} H_1 - \cdots - B_1 H_{M_b}) \]

From (2.42) consider the following equation

\[ 0 = r_{M_b-1}^d i + r_{M_b-2}^d i+1 + \cdots + r_0^d M_b+i \]  \hspace{1cm} (2.54)

From this equation the following equation is derived (by inspection)

\[ 0 = B_{M_b} H_i + B_{M_b-1} H_{i+1} + \cdots + B_0 H_{M_b+i} \]

or,

\[ H_{M_b+i} = B_0^{-1} B_{M_b+i-1} - B_0^{-1} B_{M_b+i-2} - \cdots - B_0^{-1} B_{M_b} H_1 \]  \hspace{1cm} (2.55)

Equation (2.55) is a recursive equation and can be represented by the state-variable diagram shown in Fig. (2.6).
Fig. 2.6. State-variable diagram of Eqn. (2.55)

From figure (2.6) the state variables are:

\[ X_1(n+1) = X_2(n) \]
\[ X_2(n+1) = X_3(n) \]
\[ \vdots \]
\[ X_{M_b}(n+1) = -B_0^{-1}B_{M_b}X_1(n) - B_0^{-1}B_{M_b}^{-1}X_2(n) - \]
\[ \vdots - B_0^{-1}B_1X_{M_b}(n) \]

Hence:

\[
\begin{bmatrix}
  X_1(n+1) \\
  X_2(n+1) \\
  \vdots \\
  X_{M_b}(n+1)
\end{bmatrix} =
\begin{bmatrix}
  0 & I & 0 & 0 \\
  0 & 0 & \ddots & I \\
  \vdots & \ddots & \ddots & 0 \\
  -B_0^{-1}B_{M_b} & -B_0^{-1}B_{M_b}^{-1} & -B_0^{-1}B_1
\end{bmatrix}
\begin{bmatrix}
  X_1(n) \\
  X_2(n) \\
  \vdots \\
  X_{M_b}(n)
\end{bmatrix}
\]

(2.57)
or, in condensed form:

\[ X(n+1) = BX(n) \]  \hspace{1cm} (2.58)

which, for the case of \( M_a < M_b \), has the solution

\[ X(n) = B^n X(0) \]  \hspace{1cm} (2.59)

From the state-variable diagram we have

\[ X(0) = \begin{bmatrix} H_0 \\ H_1 \\ H_{M_b-1} \end{bmatrix} \]  \hspace{1cm} (2.60)

Consider now, the case of \( M_a = M_b \). Eqn. (2.54) does not apply for \( i=0 \), since, from equation (2.42) \( f_{M_b} \neq 0 \).

Hence, the solution of equation (2.59) becomes

\[ X(n) = B^{n-1} X(1) \]  \hspace{1cm} (2.61)

where:

\[ X(1) = \begin{bmatrix} H_1 \\ H_2 \\ \vdots \\ H_{M_b} \end{bmatrix} \]  \hspace{1cm} (2.62)

and \( n > 0 \).
Equation (2.59) (or (2.62)) gives a closed form solution for the first \( N_b + 1 \) vertical lines. Hence, the vertical lines are generated in closed form via one of these two equations and equation (2.41d).

2.2.6 GENERATION OF THE FIRST \( M_b + 1 \) HORIZONTAL LINES IN CLOSED FORM

Using results obtained in Sec. 2.2.3 and proceeding as before, we obtain the following result:

\[
H'_{N_b+i} = -(B_0')^{-1}B_1H'_{N_b+i-1} - (B_0')^{-1}H'_{N_b+i-2} - \ldots - (B_0')^{-1}B_{N_b}H_i
\]

(2.63)

where:

\[
H_i' = \begin{bmatrix} h(0,i) \\ h(1,i) \\ \vdots \\ h(M_b,i) \end{bmatrix} \quad (2.64); \quad B_i' = \begin{bmatrix} b_{0i} & b_{0i} \\ b_{1i} & b_{0i} \\ b_{2i} & b_{1i} & b_{0i} \\ \vdots \\ b_{M_b,i} & b_{M_b-1,i} & b_{0i} \end{bmatrix}
\]

(2.65)

Note that, \( H_0' \), \( H_1' \), \ldots, \( H_{N_b}' \) has been calculated in the previous case (Eqn. 2.53).

The linear recursive equation (2.63) is represented in the state-variable diagram shown in figure 2.7.
Fig. 2.7. State-variable diagram of Eqn. (2.63).

As before:

\[
\begin{bmatrix}
X_1'(n+1) \\
X_2'(n+1) \\
\vdots \\
X_{N_b}'(n+1)
\end{bmatrix} =
\begin{bmatrix}
0 & I & 0 \\
0 & 0 & I \\
\vdots & \vdots & \vdots \\
-(B_0'N_{BD})^{-1} & -(B_0'N_{BD})^{-1} & -(B_0'B_1')^{-1}
\end{bmatrix}
\begin{bmatrix}
X_1'(n) \\
X_2'(n) \\
\vdots \\
X_{N_b}'(n)
\end{bmatrix}
\]

This equation is written in the condensed form:

\[
X'(n+1) = B'X'(n)
\]

(2.66)

For \(M_b > N_b\) equation (2.67) has the solution
\[ x'(n) = (B')^n x'(0) \]  

(2.68)

where:
\[
\begin{bmatrix}
  h_0' \\
  h_1' \\
  \vdots \\
  h_{N_b-1}'
\end{bmatrix}
\]

(2.69)

\[ x'(0) = \begin{bmatrix}
  h_0 \\
  h_1 \\
  \vdots \\
  h_{N_b-1}
\end{bmatrix} \]

For \( M_b = N_b \) the solution of (2.67) is
\[ x'(n) = (B')^{n-1} x'(1) \]  

(2.70)

where; \( n > 1 \)

and:
\[
\begin{bmatrix}
  h_1' \\
  h_2' \\
  \vdots \\
  h_{N_b}'
\end{bmatrix}
\]

(2.71)

Hence, a partial impulse-response can be generated horizontally from the set of closed-form solution given by Eqn. (2.68) (or 2.70).
2.2.7 GENERATING THE FULL IMPULSE RESPONSE

Sections 2.2.5 and 2.2.6 provide techniques for generating partial impulse responses (i.e. section 2.2.5 discusses closed form solutions for the first \( M_D + 1 \) columns and section 2.2.6 the first \( M_D + 1 \) rows). Since the z-transform of the complete impulse response is given in a row by row fashion in eqn. (2.37d), it should be possible to obtain each row in a closed form solution by a direct application of the one-dimensional case.

For low orders of filter (say \( M_a, M_b, N_a, N_d \leq 4 \)) the roots of \( r_0(z_2) \) in eqn. (2.37d) or the roots of \( p_0(z_1) \) in eqn. (2.41d) may be determined to great precision.

Since the application of increasing row or column number only affects the multiplicities of their roots, the square matrix of eqn. (2.15) has elements which may also be generated to great precision. Since the inversion of large order matrices of this type are possible, the complete set of closed form solutions should be computationally useful for large frames of data.

The next section illustrates the various techniques by applying them to a few examples.
2.2.8 EXAMPLES:

EX. 1: Consider the following filter

\[
F(z_1, z_2) = \frac{\sum_{i=0}^{1} \sum_{j=0}^{1} a_{ij} z_1^{-i} z_2^{-j}}{\sum_{i=0}^{1} \sum_{j=0}^{1} b_{ij} z_1^{-i} z_2^{-j}} \tag{2.72}
\]

Form the following matrices

\[
B_0 = \begin{bmatrix} b_{00} & 0 \\ b_{01} & b_{00} \end{bmatrix}, \quad B_1 = \begin{bmatrix} b_{10} & 0 \\ b_{11} & b_{10} \end{bmatrix} \tag{2.73}
\]

\[
A_0 = \begin{bmatrix} a_{00} \\ a_{01} \end{bmatrix}, \quad A_1 = \begin{bmatrix} a_{10} \\ a_{11} \end{bmatrix} \tag{2.74}
\]

Therefore, from (2.53)

\[
H_0 = B_0^{-1} A_0, \quad \text{and} \quad H_1 = B_0^{-1} (A_1 - B_1 H_0) \tag{2.75}
\]

where:

\[
H_0 = \begin{bmatrix} h(0,0) \\ h(0,1) \end{bmatrix}, \quad H_1 = \begin{bmatrix} h(1,0) \\ h(1,1) \end{bmatrix} \tag{2.76}
\]

Hence:

\[
\begin{bmatrix} h(0,0) \\ h(0,1) \end{bmatrix} = \begin{bmatrix} a_{00}/b_{00} \\ -b_{01}a_{00}/b_{00} + a_{01}/b_{00} \end{bmatrix} \tag{2.77}
\]
and:

\[
\begin{bmatrix}
  h(1,0) \\
  h(1,1)
\end{bmatrix} = \begin{bmatrix}
  a_{10} & -b_{10} \\
  b_{00} & b_{00}
\end{bmatrix} h(0,0) \\
-\frac{b_{01}}{b_{00}} \{a_{10} - b_{10} h(0,0)\} \\
+ \frac{1}{b_{00}} \{a_{11} - b_{11} h(0,0) - b_{01} h(0,1)\}
\]

(2.78)

The first two vertical lines can be generated in closed-form from

\[
\begin{bmatrix}
  h(i,0) \\
  h(i,1)
\end{bmatrix} = \left(\begin{bmatrix}
  b_{00} & 0 \\
  b_{01} & b_{00}
\end{bmatrix}\right)^{-1} \begin{bmatrix}
  b_{10} & 0 \\
  b_{11} & b_{10}
\end{bmatrix}^i \begin{bmatrix}
  h(1,0) \\
  h(1,1)
\end{bmatrix}
\]

(2.79)

where: \(i > 1\)

Analytic expressions could be obtained for \(h(i,0)\) and \(h(i,1)\) by applying the Cayley-Hamilton theorem for the function of a matrix to (2.79).

For \(k \geq 2\), the \(k^{th}\) vertical line can be generated from

\[
h(i,k) = z^{-1} \left\{ \frac{-(b_{0k} + b_{11} z^{-1}) z(h(i,k-1))}{(b_{00} + b_{10} z^{-1})} \right\}
\]

(2.80)

The \(z\)-transform of the first two vertical lines can be obtained from (2.79) or from equations (2.41a) and (2.41b).
\[ v_0(z_1) = \frac{a_{00} + a_{10}z_1^{-1}}{b_{00} + b_{10}z_1^{-1}} \]  
\[ (2.81) \]

\[ v_1(z_1) = \frac{(a_{01} + a_{11}z_1^{-1})(b_{00} + b_{10}z_1^{-1}) - (b_{01} + b_{11}z_1^{-1})(a_{00} + a_{10}z_1^{-1})}{(b_{00} + b_{10}z_1^{-1})^2} \]  
\[ (2.82) \]

Applying the two conditions for stability given in sec. 2.2.4, we obtain the following two conditions:

\[ \left| \frac{b_{01}}{b_{00}} \right| < 1 \]  
and:

\[ \left| \frac{b_{10}}{b_{00}} \right| < 1 \]  
\[ (2.83) \]

EX. 2: Consider, the two filters given by Shanks [37]:

\[ H(z_1, z_2) = \frac{1}{1 - 0.7z_1^{-1} - 0.5z_2^{-1} + 0.3z_1^{-1}z_2^{-1}} \]  
\[ (2.84) \]

\[ H(z_1, z_2) = \frac{1}{1 - 0.95z_1^{-1} - 0.9z_2^{-1} + 0.5z_1^{-1}z_2^{-1}} \]  
\[ (2.85) \]

The two conditions given by eqn. (2.83) are satisfied for both filters. However, applying the three conditions given in [3] & [4]:
\[
\left| \frac{1 + b_{01}}{b_{10} + b_{11}} \right| > 1
\]
\[
\left| \frac{1 - b_{01}}{b_{10} - b_{11}} \right| > 1
\]
\[
\left| b_{10} \right| < 1
\]

we find that the filter given by (2.84) meets all the conditions, and hence is stable. The filter given by equation (2.85) does not satisfy the first condition in (2.86) and hence is unstable.

This indicates that the conditions derived in this chapter for stability are necessary but not sufficient.

**EX. 3:** Consider the following filter

\[
H(z_1, z_2) = \frac{1}{b_{00} + b_{01}z_2^{-1} + (b_{10} + b_{11}z_2^{-1})z_1^{-1}} \quad (2.87)
\]

The general form of the \( z \)-transform of the \( i \)-th horizontal line could be derived from (2.37) and is given by

\[
d_i(z_2) = (-1)^i \frac{(b_{10} + b_{11}z_2^{-1})^i}{(b_{00} + b_{01}z_2^{-1})^{i+1}} \quad (2.88)
\]

An analytical expression for the inverse of \( d_i(z_2) \) could be derived by evaluating the inverse integral.
\[ h(i,k) = \frac{1}{2\pi j} \oint (-1)^i \frac{(b_{10} + b_{11}z_2^{-1})^i}{(b_{00} + b_{01}z_2^{-1})} z_2^{k-1} \, dz_2 \quad (2.89) \]

From the Residue theorem

\[ h(i,k) = \frac{(-1)^i}{i!} \lim_{z_2 \to -\frac{b_{01}}{b_{00}}} \frac{d^i}{dz_2^i} \left\{ (z_2b_{10} + b_{11})^i z_2^k \right\} \quad (2.90) \]

After some manipulation

\[ h(i,k) = (-1)^i \sum_{j=0}^{i} \frac{k!}{(k-i+j)!} \frac{1}{(i-j)!} \binom{i}{j} \left( \frac{b_{01}}{b_{00}} b_{10} + b_{11} \right)^{i-j} (b_{10})^j \quad (2.91) \]

where:

\[ \binom{i}{j} = \frac{i!}{j!(i-j)!} \]

This is a general formula for calculating the impulse response of (2.87) anywhere in the spatial-domain. Thus, (2.91) is an analytic expression for the inverse of (2.87).

Looking at eqn. (2.91) we can readily see why the partial stability conditions given in Sec. 2.2.4 are not sufficient for total stability. The partial conditions guarantee a finite bound on \( \sum_{k=0}^{\infty} |h(i,k)| \) but do not guarantee a finite
bound on the infinite summation: \[ \sum_{i=0}^{\infty} \left\{ \sum_{k=0}^{\infty} |h(i,k)| \right\} \]

We can arrive at a different expression from the z-transform of the i\textsuperscript{th} vertical line. From (2.41) we can derive

\[ v_i(z_1) = (-1)^i \frac{(b_{01} + b_{11}z_1^{-1})^i}{(b_{00} + b_{10}z_2^{-1})^{i+1}} \]  \( (2.92) \)

The analytical expression can be derived by analogy with the previous expressions. Hence:

\[ h(k,i) = (-1)^i \sum_{j=0}^{i} \frac{k!}{(k-i+j)!} \frac{1}{(i-j)!} \binom{i}{j} \]

\[ \left( \frac{-b_{10}}{b_{00} b_{01} + b_{11}} \right)^{i-j} \times (b_{01})^{i-j} \]  \( (2.93) \)

Expression (2.91), (or (2.93)) shows the complexity of the analytic expression for the inverse of a two-dimensional z-transform. This would suggest the use of numerical rather than analytic techniques for evaluating the impulse response of higher-order two-dimensional recursive filters.

An interesting high speed numerical technique is shown in the next section:

\section*{2.3 \textit{Generation of the Impulse Response Via FFT Pruning}}

The impulse response of a recursive filter can be generated numerically, in a much simpler way, via a time-
pruned FFT algorithm (see Appendices B and D) FFTP, and an inverse FFT (IFFT), as follows:

For the one-dimensional recursive filter:

\[ y(n) = \text{IFFT} \left\{ \frac{\text{FFTP}([a_i])}{\text{FFTP}([b_i])} \right\} \]  
(2.94)

where: \([a_i] = (a_0 \ a_1 \ldots \ a_N \ 0 \ 0 \ldots 0)^T\)

\[ 2^{L_1} \text{ terms} \]

\([b_i] = (b_0 \ b_1 \ldots \ b_M \ 0 \ 0 \ldots 0)^T\)

\[ 2^{L_2} \text{ terms} \]

\(2^{L_1}\), and \(2^{L_2}\) are the nearest higher numbers to \((N+1)\) and \((M+1)\) respectively.

Similarly, for the two-dimensional recursive filter:

\[ y(n_1,n_2) = \text{IFFT} \left\{ \frac{\text{FFTP}([a_{ij}])}{\text{FFTP}([b_{ij}])} \right\} \]  
(2.95)

where; in this case FFTP is a two-dimensional space pruned FFT, and IFFT a two-dimensional inverse FFT.
\[ a_{ij} = \begin{bmatrix} a_{00} & a_{01} & \ldots & a_{0N_a} & 0 & \ldots & 0 \\ a_{10} & a_{11} & \ldots & a_{1N_a} & 0 & \ldots & 0 \\ \vdots & \vdots & \ddots & \vdots & \vdots & \ddots & \vdots \\ a_{M_a0} & a_{M_a1} & \ldots & a_{M_aN_a} & 0 & \ldots & 0 \\ 0 & 0 & \ldots & 0 & 0 & \ldots & 0 \\ 0 & 0 & \ldots & 0 & 0 & \ldots & 0 \end{bmatrix} \]

\[ b_{ij} = \begin{bmatrix} b_{00} & b_{01} & \ldots & b_{0N_b} & 0 & \ldots & 0 \\ b_{10} & b_{11} & \ldots & b_{1N_b} & 0 & \ldots & 0 \\ \vdots & \vdots & \ddots & \vdots & \vdots & \ddots & \vdots \\ b_{M_b0} & b_{M_b1} & \ldots & b_{M_bN_a} & 0 & \ldots & 0 \\ 0 & 0 & \ldots & 0 & 0 & \ldots & 0 \\ 0 & 0 & \ldots & 0 & 0 & \ldots & 0 \end{bmatrix} \]

where, \( 2^{LA1} \), \( 2^{LA2} \), \( 2^{LB1} \), and \( 2^{LB2} \) are the nearest higher numbers to \((N_a + 1)\), \((M_a + 1)\), \((N_b + 1)\), and \((M_b + 1)\) respectively.
2.4 SUMMARY:

A technique has been presented for computing the inverse \( z \)-transform, of a rational function of polynomials in \( z \), as an analytic function. The computational procedure is easy to implement on a digital computer and is suitable for the analysis of large orders of systems. The technique is based on a state-variable formulation of the relationship between numerator and denominator of the rational form in \( z \) to the infinite series expansion. The Cayley-Hamilton theorem is used to produce the solution, in closed form, to the state variable formulation. Examples have been given to illustrate the technique and an extension of the technique to the problem of inverting the Laplace transformation is shown.

The extension of the previous analysis to the two-dimensional recursive filter, leads to the formulation of a set of closed-form solutions for evaluating partial and full spatial impulse responses. Necessary conditions for stability of two-dimensional recursive filters are also derived. The procedure is illustrated by a set of examples. In one example a single expression for the closed form full spatial impulse response of a second-order two-dimensional "all-pole" recursive filter is derived.

A numerical technique involving FFT pruning is also shown for determining a partial impulse response for both one- and two-dimensional digital filters.
CHAPTER III

DESIGN OF RECURSIVE FILTERS TO APPROXIMATE
A PRESCRIBED IMPULSE RESPONSE.

3.1 INTRODUCTION

The literature survey has revealed a great many methods of approaching the identification problem via linear techniques. All of the techniques found so far, however, concentrate the minimization in the time or spatial domain. It would seem useful to examine computational savings by mapping the error function to the frequency domain and making use of the high speed FFT algorithms.

In this chapter, a technique is developed for identifying the parameters of a linear recursive filter from a given impulse response. The method initially minimizes a weighted square-error criterion in the frequency domain, to obtain an initial solution to the parameters. This solution is then improved on, through a linear iterative refinement scheme.

The technique is first presented for the one-dimensional recursive filter and then extended to linear spatial recursive filters.
3.2 ONE-DIMENSIONAL RECURSIVE FILTERS:

3.2.1 FORMULATION OF THE PROBLEM:

Consider the recursive filter transfer function defined by

\[ H(z) = \frac{\sum_{i=0}^{M_a} a_i z^{-i}}{1 + \sum_{i=1}^{M_b} b_i z^{-i}} = \frac{P(z)}{Q(z)} \]  \hspace{1cm} (3.1)

If we let \( z = e^{j\Omega_n} \), \( \Omega_n = \frac{n\pi}{N} \), then \( H(e^{j\Omega_n}) = \frac{P(e^{j\Omega_n})}{Q(e^{j\Omega_n})} \) \hspace{1cm} (3.2)

The problem is now one of producing an optimal fit of the set of values obtained from (3.2), for \( n = 0, 1, \ldots, N \), to a prescribed set of values \( \{G(e^{j\Omega_n})\} \). The set \( \{G(e^{j\Omega_n})\} \) can be obtained via the impulse response of the desired filter, \( g_k (k = 0, 1, \ldots, 2N-1) \), by using the DFT. Normally one would specify an error measure of the form

\[ E = \sum_{n=0}^{N} |e_n|^2 \]  \hspace{1cm} (3.3)

where,

\[ e_n = G(e^{j\Omega_n}) - \frac{P(e^{j\Omega_n})}{Q(e^{j\Omega_n})} \]  \hspace{1cm} (3.4)
However, in order to minimize $E$ with respect to the parameter vectors $A = \{a_0, a_1, \ldots, a_{M_a}\}^T$, and $B = \{b_1, b_2, \ldots, b_{M_b}\}^T$, we require a non-linear minimization procedure and this is something we wish to avoid.

Let us adopt a procedure first described by Levy [26] in which we multiply both sides of (3.4) by $Q(e^{j\Omega n})$.

Thus

$$
e'_n = e_nQ(e^{j\Omega n}) = G(e^{j\Omega n})Q(e^{j\Omega n}) - P(e^{j\Omega n})$$

(3.5)

If we now form a new error measure

$$E = \sum_{n=0}^{N} |e'_n|^2$$

(3.6)

This can be minimized using linear techniques.

The main problem is that we have a solution dependent set of weighting coefficients in (3.6) which may lead to an inferior fit compared to the error measure of (3.3). If we initially identify the parameters of the filter, using (3.6), it is possible to iteratively modify the identification [27] such that we approach the unweighted measure of (3.3). This is discussed in section 3.2.4.

The minimization of (3.6) is now carried out by successive partial differentiation with respect to each parameter and equating to zero.
3.2.2 RELATION BETWEEN THE TIME AND FREQUENCY APPROACH:

In this section we shall prove that the time and frequency domain approaches to this problem yields the same result.

The time-domain error can be written as [3], [12],

\[ e(n) = g(n) + \sum_{i=1}^{M_p} b_i g(n-i) - \sum_{i=0}^{M_a} a_i \delta(n-i) \]  \hspace{1cm} (3.7)

where: \( \delta(n-i) = 1 \), for \( n = i \)
\( = 0 \), otherwise

The total square-error in the time-domain is

\[ E_1 = \sum_{n=0}^{2N-1} e^2(n) \]  \hspace{1cm} (3.8)

Taking the DFT of equation (3.7) we get

\[ E(e^{-j\Omega k}) = G(e^{-j\Omega k}) \left\{ 1 + \sum_{i=1}^{M_p} b_i e^{-j\Omega ki} \right\} - \sum_{i=0}^{M_a} a_i e^{-j\Omega ki} \]
which is the "weighted" error derived in the previous section.

The total absolute square-error in the frequency domain is

$$E_2 = \sum_{k=0}^{2N-1} |E(e^{-j\omega k})|^2$$

(3.9)

But, according to Parseval's theorem we have

$$E = \frac{1}{2N} \sum_{k=0}^{(2N)-1} |E(e^{-j\omega k})|^2 = \sum_{n=0}^{(2N)-1} e^2(n)$$

(3.10)

That is, minimizing (3.8) or (3.9) with respect to the coefficients should yield the same minimum.

It is also reasonable to assume that any of the refinement schemes, to improve the initial solution obtained from eq. (3.8) or (3.9), whether applied in the time- or frequency-domain should yield the same result.

3.2.3 ANALYSIS:

From (3.1) and (3.2) we obtain the following

$$P(e^{j\omega n}) = \sum_{i=0}^{M_a} a_i \cos i\omega_n - j \sum_{i=1}^{M_a} a_i \sin i\omega_n$$

(3.11)

$$Q(e^{j\omega n}) = 1 + \sum_{i=1}^{M_b} b_i \cos i\omega_n - j \sum_{i=1}^{M_b} b_i \sin i\omega_n$$

(3.12)
Let \( G(e^{j\Omega n}) = R_n + jI_n \). The error measure, \( E \), is found from (3.6)

\[
E = \sum_{n=0}^{N} \left( (R_n \sigma_n - I_n \lambda_n - \alpha_n)^2 \right) + \left( (I_n \sigma_n + R_n \lambda_n - \delta_n)^2 \right)
\]

(3.13)

where:

\[
\sigma_n = 1 + \sum_{i=1}^{M_b} b_i \cos i\Omega_n
\]

\[
\lambda_n = -\sum_{i=1}^{M_b} b_i \sin i\Omega_n
\]

(3.14)

\[
\sigma_n = \sum_{i=0}^{M_a} a_i \cos i\Omega_n
\]

\[
\lambda_n = \sum_{i=1}^{M_a} a_i \sin i\Omega_n
\]

If we now partially differentiate (3.13) with respect to \( \{a_i\} \) and \( \{b_i\} \), after some manipulation, the following equation results:
\[
\begin{bmatrix}
C_1 & C_2 \\
C_2^T & C_3
\end{bmatrix}
\begin{bmatrix}
X
\end{bmatrix}
= 
\begin{bmatrix}
F_1 \\
F_2
\end{bmatrix}
\] (3.15)

or \( CX = F \)

where, defining the following terms:

\[
\eta_k = \sum_{n=0}^{N} \cos k\Omega_n
\]

\[
\phi_k = \sum_{n=0}^{N} R_n \cos k\Omega_n
\]

\[
\gamma_k = \sum_{n=0}^{N} I_n \sin k\Omega_n
\]

\[
\beta_k = \sum_{n=0}^{N} |G(e^{j\Omega_n})|^2 \cos k\Omega_n
\] (3.16)

the sub-matrices and vector \( X \) of equation (3.15) can be written as follows:

\[
C_1 =
\begin{bmatrix}
\eta_0 & \eta_1 & \eta_2 & \cdots & \eta_{M_a} \\
\eta_1 & \eta_0 & \eta_2 & \cdots & \eta_{M_a-1} \\
\eta_2 & \eta_1 & \eta_0 & \cdots & \eta_{M_a-2} \\
\cdots & \cdots & \cdots & \cdots & \cdots \\
\eta_{M_a} & \eta_{M_a-1} & \cdots & \cdots & \eta_0
\end{bmatrix}
\] (3.17)
\[
C_2 = \begin{bmatrix}
\phi_1 + \gamma_1 & \phi_2 + \gamma_2 & & \phi_{M_b} + \gamma_{M_b} \\
\phi_0 + \gamma_0 & \phi_1 + \gamma_1 & \phi_2 + \gamma_2 & & \\
\phi_1 - \gamma_1 & \phi_0 + \gamma_0 & \phi_1 + \gamma_1 & & \\
\phi_2 - \gamma_2 & \phi_1 - \gamma_1 & \phi_0 + \gamma_0 & & \\
& & & \ddots & \\
\phi_{M_a - 1} - \gamma_{M_a - 1} & & & & 
\end{bmatrix}
\]

\[
C_3 = \begin{bmatrix}
\beta_0 & \beta_1 & \beta_2 & & \beta_{M_b - 1} \\
\beta_1 & \beta_0 & \beta_1 & \beta_2 & & \\
\beta_2 & \beta_1 & \beta_0 & \beta_1 & & \\
& & & \ddots & \ddots \\
& & & & \beta_{M_b - 1} & 
\end{bmatrix}
\]

\[
x = \begin{bmatrix}
a_0 & a_1 & \cdots & a_{M_a} & b_1 & b_2 & \cdots & b_{M_b}
\end{bmatrix}^T
\]

\[
F_1 = \begin{bmatrix}
\phi_0 - \gamma_0 & \phi_1 - \gamma_1 & & \phi_{M_a} - \gamma_{M_a}
\end{bmatrix}^T
\]

\[
F_2 = -\begin{bmatrix}
\beta_1 & \beta_2 & \beta_3 & \cdots & \beta_{M_b}
\end{bmatrix}^T
\]
Equation (3.15) is now solved for the coefficients \{a_i\} and \{b_i\} of the filter. We may make use of efficient computational strategies in both computing the elements of the sub-matrices and vectors, \(D, F_1, F_2\) in equation (3.15) and also the fact that the \(C\) matrix is symmetrical. These techniques are discussed in section (3.2.5).

3.2.4 THE ITERATIVE ALGORITHM:

Sanathanan and Koerner [27] have discussed a scheme for approaching an unweighted squared-error criterion from an error criterion of the form shown in (3.6). Since their technique was developed for continuous systems we present a discrete analysis below.

The weighted error expression in equation (3.5) can be modified as follows:

\[
\varepsilon_n^L = \frac{e_n^L}{Q_{L-1}(e^{jn})} = \frac{G(e^{jn})Q_L(e^{jn}) - P_L(e^{jn})}{Q_{L-1}(e^{jn})}
\]

(3.23)

where we choose:

\[
Q_0(e^{jn}) = 1\quad \forall\ n
\]

If convergence is obtained, that is, if the coefficients of \(Q_L\) converge as \(L\) becomes large, the error \(\varepsilon_n^L\) is equal to the error \(\varepsilon_n\). While no proof of convergence can be offered, every experiment with this technique conducted so far has resulted in a decrease in mean square-
error. To find the true M.S.E., we have to compute new
gradients and force these to zero.

Multiplying \( \varepsilon_{n_L} \) by its conjugate we obtain

\[
|\varepsilon_{n_L}|^2 = |G(e^{j\Delta n})Q_L(e^{j\Delta n}) - P_L(e^{j\Delta n})|^2
\]

\[
|Q_{L-1}(e^{j\Delta n})|^2
\]

(3.24)

Letting

\[
W_{n_J} = \frac{1}{|Q_J(e^{j\Delta n})|^2}
\]

(3.25)

then

\[
E_L = \sum_{n=0}^{N} |\varepsilon_{n_L}|^2 W_{n_{L-1}}
\]

(3.26)

The iterative algorithm can now be obtained

\[
\begin{bmatrix}
X_L
\end{bmatrix} = \begin{bmatrix}
C_{L-1}
\end{bmatrix}^{-1} \begin{bmatrix}
F_{L-1}
\end{bmatrix}
\]

(3.27)

where the submatrices of \( C_{L-1} \) and subvectors of \( F_{L-1} \) are to be interpreted as before, with an extra subscript attached to the elements.

The new elements are given by

\[
\eta_{k_J} = \sum_{n=0}^{N} W_{n_J} \cos k\omega_n
\]

\[
\phi_{k_J} = \sum_{n=0}^{N} W_{n_J} R_n \cos k\omega_n
\]

(3.28)
\[ \gamma_j = \sum_{n=0}^{N} W_n j n \sin k_n \]
\[ \beta_j = \sum_{n=0}^{N} W_n j n |G(e^{jn})|^2 \cos k_n \]

From equation (3.12) and (3.25)

\[ W_n j n = \frac{1}{|Q_j(e^{jn})|^2} = \frac{1}{1 + \sum_{i=1}^{M_b} b_{ij} e^{-jn}} \]  \hspace{1cm} (3.29)

with \( W_{n0} = 1 \), \( \forall n \)

where the \( \{b_{ij}\} \) are obtained from

\[ X_j = \begin{bmatrix} a_{0j} \\ a_{1j} \\ \vdots \\ a_{Maj} \\ b_{1j} \\ b_{2j} \\ \vdots \\ b_{Mbj} \end{bmatrix} \]  \hspace{1cm} (3.30)
3.2.5 **COMPUTATIONAL DETAILS:**

Equations (3.27) - (3.28) give the basic iterative algorithm for computing the minimum of the unweighted square-error criterion. There are, however, some computational efficiencies that may be employed in the algorithm. These are pointed out below.

**Initial Solution:**

It is important to note that in the preceding analysis we have taken account of the Hermitian symmetry in the DFT output of the original sampled function, by minimizing $E_L$ over the range $0 \leq \Omega_n \leq \pi$. However, equations (3.27) and (3.28) are greatly simplified, for the initial solution, when carrying out the minimization of $E_0$ over the range $0 \leq \Omega_n < 2\pi$.

The computations of the elements in (3.28) are, thus, reformed to:

$$n_{k0} = \Re \left\{ \sum_{n=0}^{2N-1} e^{-jk\Omega_n} \right\} = \frac{1 - e^{-j2\pi k}}{1 - e^{-j\frac{\pi}{N} k}}$$

$$= \begin{cases} 2N & \text{for } k = 0 \\ 0 & \text{for } k > 0 \end{cases}$$

(3.31)
\[ \dot{\gamma}_{k0} + \gamma_{k0} = \sum_{n=0}^{2N-1} R_n \cos(k_n\Omega_n) + \sum_{n=0}^{2N-1} I_n \sin(k_n\Omega_n) \]

\[ \sum_{i=0}^{2N-1} g_i \left\{ \Re \sum_{n=0}^{2N-1} e^{-j\Omega_n(k+i)} \right\} \quad (3.32) \]

Hence:

\[ \dot{\gamma}_{k0} + \gamma_{k0} = 2N\eta_{2N-k} \]

\[ \dot{\gamma}_{k0} - \gamma_{k0} = 2N\eta_k \quad (3.33) \]

\[ \dot{\gamma}_{00} + \gamma_{00} = 2N\eta_0 \]

We may now re-write equation (3.27), for the initial solution as:

\[
\begin{bmatrix}
  a_0 \\
  a_1 \\
  a_2 \\
  \vdots \\
  a_{M_a} \\
  b_1 \\
  b_2 \\
  \vdots \\
  b_{M_b}
\end{bmatrix}
= \begin{bmatrix}
  2NI & -2N\eta_2 & 2N\eta_1 \\
  -2N\eta_2^T & C_3 & F_2
\end{bmatrix}^{-1}
\]

(3.34)

where; I is the identity matrix
\[
D_2 = \begin{bmatrix}
g(2N-1) & g(2N-2) & \cdots & g(2N-M_B) \\
g_0 & g(2N-2) & \cdots & g(2N-1) \\
g_1 & g_0 & \cdots & g(2N-1) \\
g_2 & g_1 & \cdots & g_0 \\
g_{M_a-1}
\end{bmatrix}
\]

(3.35)

and
\[
D_1 = \begin{bmatrix} g_0 & g_1 & g_2 & \cdots & g_{M_a} \end{bmatrix}^T
\]

(3.36)

The direct solution is formed from (3.34):
\[
B = \left[-D_2^T D_2 + \frac{1}{2N} C_3\right]^{-1} \left[D_2^T D_1 + \frac{1}{2N} F_2\right]
\]

(3.37)

\[
A = D_2 B + D_1
\]

(3.38)

So far we have not discussed efficient ways of computing the $\beta_{kJ}$ coefficients.

Rewriting the expression for $\beta_{kJ}$, from (3.28), as follows:
\[
\beta_{kJ} = \Re \left\{ \sum_{n=0}^{2N-1} W_{kJ} \left| G(e^{j\omega n}) \right|^2 e^{-j\omega n} \right\}
\]

(3.39)

Equation (3.39) can be computed efficiently using a frequency-domain pruned FFT algorithm (see Appendix C).
For the iteration $L \geq 1$, equation (3.27) is partitioned into the following two equations:

$$B_L = \left\{ \left[ \hat{R}_C + C_3 \right]^{-1} \left[ F_2 + RF_1 \right] \right\}_{L-1}$$

$$A_L = \left\{ C_1^{-1} \left( F_1 - C_2 B \right) \right\}_{L-1}$$

$$R_{L-1} = \left\{ -C_2^T C_1^{-1} \right\}_{L-1}$$

(3.40)

where $R_{L-1}$ is a symmetric matrix, the bordering method [28] is a very efficient procedure for finding its inverse. Note that $A_L$ need only be computed at the final iteration.

It is more efficient to compute $B_L$ in equation (3.40) by solving the set of simultaneous equations rather than the explicit matrix inversion and multiplication operations as shown. Since $[R_C + C_3]$ is a symmetrical matrix we may use the square-root method [28] to improve the speed of computation.

The weights, $w_{nj}$, given by equation (3.29) may be calculated using a time-domain pruned FFT algorithm. This is shown in Appendix B.

The elements of the $C$ and $F$ matrices, given by equation (3.28), are calculated in the following combinations.
\[ n_{kj} = R_e \left\{ \sum_{n=0}^{N} w_{nj} e^{-jk\omega_n} \right\} \]

\[ k_j = \Re \left\{ \sum_{n=0}^{N} w_{nj} (R_n \text{j} I_n) e^{-jk\omega_n} \right\} \]  \hspace{1cm} (3.41)

\[ g_{kj} = R_e \left\{ \sum_{n=0}^{N} w_{nj} |G(e^{j\omega_n})|^2 e^{-jk\omega_n} \right\} \]

The calculation of the elements in (3.41) may be computed very efficiently using a frequency pruned FFT shown in Appendix C.

A complete Fortran IV, double precision version program of the previously explained algorithm is given in Appendix E1. It incorporates all the computational techniques previously explained.

3.2.6 COMMENTS:

The technique presented so far uses a succession of linear equations whose solutions progressively reduce the mean square error of eqn. (3.6). To find the true minimum we must now force the gradients of eqn. (3.4) to zero. Reports in the literature [12], [13] show that this second phase does not reduce the error by an amount sufficient to warrant the extra computer time required. In the many examples tried with this method, every single one has produced an extremely close fit in one iteration. For high
order systems (~12th. order) more than one iteration produces matrices so ill-conditioned that a solution becomes impossible. To initiate the second phase would seem to be impracticable. With these thoughts in mind it was decided to leave the algorithm as presented. The following examples illustrate the technique.

3.2.7 **EXAMPLES:**

**EXAMPLE 1: DESIGNING TO A PRESCRIBED IMPULSE RESPONSE**

Here we have taken the same example as discussed by Evans and Fischl [13]. The impulse response is given by

\[
g_k = \begin{cases} 
0.1k & 0 \leq k < 5 \\
1 - 0.1k & 5 \leq k < 10 \\
0 & 10 \leq k \leq 31 
\end{cases}
\]

and we require a third order recursive filter; \( M_a = 2; M_b = 3 \). The solution is given in Table 1 and Fig. 3.1.

<table>
<thead>
<tr>
<th>TABLE I</th>
</tr>
</thead>
<tbody>
<tr>
<td><strong>ITERATION NO.</strong></td>
</tr>
<tr>
<td>Coefficients</td>
</tr>
<tr>
<td>( a_0 )</td>
</tr>
<tr>
<td>( a_1 )</td>
</tr>
<tr>
<td>( a_2 )</td>
</tr>
<tr>
<td>( b_1 )</td>
</tr>
<tr>
<td>( b_2 )</td>
</tr>
<tr>
<td>( b_3 )</td>
</tr>
<tr>
<td>( Q = \left[ \sum_{i=0}^{31} (g_k - h_k)^2 \right]^{\frac{1}{2}} )</td>
</tr>
</tbody>
</table>


This problem took 0.29 seconds execution time on an IBM 360/65 computer.

The closed-form solution of the impulse-response, as calculated by the computer programme in Appendix A, for each iteration is:

**Initial Solution:**

\[ y(k \cdot T) = 0.0384 \cdot (-0.1)^k + (0.865)^k \]

\[ \cdot (-0.0384 \cdot \cos(0.315k)) \]

\[ + 0.505 \cdot \sin(0.315k)) \]

**First Iteration:**

\[ y(k \cdot T) = 0.817 \cdot (0.765)^k \]

\[ + (0.801)^k \cdot (-0.825 \cdot \cos(0.470k)) \]

\[ + 0.0956 \cdot \sin(0.470k)) \]

**Second Iteration:**

\[ y(k \cdot T) = 0.916 \cdot (0.756)^k + (0.793)^k \]

\[ \cdot (-0.9 \cdot \cos(0.495k) - 0.028 \cdot \sin(0.495k)) \]

These are used in calculating the impulse-responses shown in Fig. 3.1.
Fig. 3.1a Desired Specifications of Example 1.

Fig. 3.1b. Initial Solution
Fig. 3.1c First iteration

Fig. 3.1d Second iteration
EXAMPLE 2: FITTING SPEECH DATA TO A RECURSIVE FILTER:

The first 256 samples of the word NOON, sampled at 10 KHz, are shown in figures 3.2a-3.5a. Each 64 samples are fitted with a 12th. order recursive filter, and the results are shown in Tables II-V. Figures 3.2b-3.5b and 3.2c-3.5c give plots of the initial solution and first iteration respectively.

The impulse responses are calculated using the FFT method explained in Chapter II, sec. 2.3. The closed-form solutions, however, did not give good results due to the problem of accurately calculating the poles of such high order filters.
### TABLE II

<table>
<thead>
<tr>
<th>ITERATION NO. COEFFICIENTS</th>
<th>INITIAL SOLUTION</th>
<th>(Q) =</th>
<th>Execution time = 0.65 sec.</th>
</tr>
</thead>
<tbody>
<tr>
<td></td>
<td>[A]</td>
<td>[B]</td>
<td>63</td>
</tr>
<tr>
<td>(a_0) b_1</td>
<td>1.282346</td>
<td>-1.975723</td>
<td>(\sum_{k=0}^{n} (g_k - h_k^2)^{1/2}) = 1.9335</td>
</tr>
<tr>
<td>(a_1) b_2</td>
<td>0.986412</td>
<td>1.452383</td>
<td>0.38814</td>
</tr>
<tr>
<td>(a_2) b_3</td>
<td>-0.365076</td>
<td>-1.123727</td>
<td></td>
</tr>
<tr>
<td>(a_3) b_4</td>
<td>0.675046</td>
<td>0.935554</td>
<td></td>
</tr>
<tr>
<td>(a_4) b_5</td>
<td>-0.125023</td>
<td>-0.600193</td>
<td></td>
</tr>
<tr>
<td>(a_5) b_6</td>
<td>0.384410</td>
<td>0.663010</td>
<td></td>
</tr>
<tr>
<td>(a_6) b_7</td>
<td>-0.300757</td>
<td>-0.305349</td>
<td></td>
</tr>
<tr>
<td>(a_7) b_8</td>
<td>-0.126782</td>
<td>-0.109765</td>
<td></td>
</tr>
<tr>
<td>(a_8) b_9</td>
<td>0.095506</td>
<td>-0.159982</td>
<td></td>
</tr>
<tr>
<td>(a_9) b_{10}</td>
<td>0.348176</td>
<td>0.569742</td>
<td></td>
</tr>
<tr>
<td>(a_{10}) b_{11}</td>
<td>-0.352049</td>
<td>-0.303474</td>
<td></td>
</tr>
<tr>
<td>(a_{11}) b_{12}</td>
<td>-0.139164</td>
<td>-0.024678</td>
<td></td>
</tr>
<tr>
<td></td>
<td>[A]</td>
<td>[B]</td>
<td></td>
</tr>
<tr>
<td>(a_{12}) b_{13}</td>
<td>-1.255746</td>
<td>3.676183</td>
<td>(tabulated values)</td>
</tr>
</tbody>
</table>
Fig. 3.2a The first 64 samples of the word NOON

Fig. 3.2b Initial solution
Fig. 3.2c First iteration
### TABLE III

<table>
<thead>
<tr>
<th>ITERATION NO.</th>
<th>INITIAL SOLUTION</th>
<th>1</th>
</tr>
</thead>
<tbody>
<tr>
<td>COEFFICIENTS</td>
<td>[A]</td>
<td>[B]</td>
</tr>
<tr>
<td>a₀</td>
<td>b₁ 0.881395</td>
<td>-1.959438</td>
</tr>
<tr>
<td>a₁</td>
<td>b₂ -1.848984</td>
<td>1.719406</td>
</tr>
<tr>
<td>a₂</td>
<td>b₃ 1.279789</td>
<td>-1.421073</td>
</tr>
<tr>
<td>a₃</td>
<td>b₄ -1.260327</td>
<td>1.083946</td>
</tr>
<tr>
<td>a₄</td>
<td>b₅ 0.733691</td>
<td>-1.218202</td>
</tr>
<tr>
<td>a₅</td>
<td>b₆ -1.528080</td>
<td>1.215044</td>
</tr>
<tr>
<td>a₆</td>
<td>b₇ 0.783508</td>
<td>-0.460840</td>
</tr>
<tr>
<td>a₇</td>
<td>b₈ -0.127767</td>
<td>-0.469811</td>
</tr>
<tr>
<td>a₈</td>
<td>b₉ -0.197838</td>
<td>0.337072</td>
</tr>
<tr>
<td>a₉</td>
<td>b₁₀ 0.419475</td>
<td>0.125159</td>
</tr>
<tr>
<td>a₁₀</td>
<td>b₁₁ 0.663174</td>
<td>-0.655674</td>
</tr>
<tr>
<td>a₁₁</td>
<td>b₁₂ -0.566082</td>
<td>0.309862</td>
</tr>
</tbody>
</table>

\[ Q = \sum_{k=0}^{63} (q_k - h_k)^2 \]

\[ 2.7808 \quad 0.30759 \]

Execution time = 0.65 sec.
Fig. 3.3a The second 64 samples of NOON

Fig. 3.3b Initial solution
Fig. 3.3c First iteration
## TABLE IV

<table>
<thead>
<tr>
<th>ITERATION NO.</th>
<th>INITIAL SOLUTION</th>
<th>1</th>
</tr>
</thead>
<tbody>
<tr>
<td>COEFFICIENTS</td>
<td>[A]</td>
<td>[B]</td>
</tr>
<tr>
<td>$a_0$</td>
<td>$b_1$</td>
<td>-1.067471</td>
</tr>
<tr>
<td>$a_1$</td>
<td>$b_2$</td>
<td>0.196405</td>
</tr>
<tr>
<td>$a_2$</td>
<td>$b_3$</td>
<td>0.224093</td>
</tr>
<tr>
<td>$a_3$</td>
<td>$b_4$</td>
<td>0.236834</td>
</tr>
<tr>
<td>$a_4$</td>
<td>$b_5$</td>
<td>0.304593</td>
</tr>
<tr>
<td>$a_5$</td>
<td>$b_6$</td>
<td>0.323927</td>
</tr>
<tr>
<td>$a_6$</td>
<td>$b_7$</td>
<td>0.099958</td>
</tr>
<tr>
<td>$a_7$</td>
<td>$b_8$</td>
<td>-0.032984</td>
</tr>
<tr>
<td>$a_8$</td>
<td>$b_9$</td>
<td>0.321178</td>
</tr>
<tr>
<td>$a_9$</td>
<td>$b_{10}$</td>
<td>0.193291</td>
</tr>
<tr>
<td>$a_{10}$</td>
<td>$b_{11}$</td>
<td>-0.259919</td>
</tr>
<tr>
<td>$a_{11}$</td>
<td>$b_{12}$</td>
<td>-0.013926</td>
</tr>
</tbody>
</table>

\[ Q = \left\{ \sum_{k=0}^{63} (g_k - h_k)^2 \right\}^{\frac{1}{2}} \]

\[ Q = 3.3481 \quad 1.9256 \]

Execution time = 0.65 sec.
Fig. 3.4a The third 64 samples of NOCN

Fig. 3.4b Initial solution
Fig. 3.4c First iteration
### TABLE V

<table>
<thead>
<tr>
<th>ITERATION NO.</th>
<th>INITIAL SOLUTION</th>
<th>( Q = )</th>
<th>( \left{ \sum_{k=0}^{15} g_k - h_k \right}^2 )</th>
<th>( \sigma )</th>
</tr>
</thead>
<tbody>
<tr>
<td>( a_0 )</td>
<td>( b_1 )</td>
<td>0.459299</td>
<td>-2.453305</td>
<td>0.485842</td>
</tr>
<tr>
<td>( a_1 )</td>
<td>( b_2 )</td>
<td>-1.069125</td>
<td>2.574539</td>
<td>-2.017501</td>
</tr>
<tr>
<td>( a_2 )</td>
<td>( b_3 )</td>
<td>0.985335</td>
<td>-1.700158</td>
<td>3.970436</td>
</tr>
<tr>
<td>( a_3 )</td>
<td>( b_4 )</td>
<td>-0.589482</td>
<td>0.804134</td>
<td>-4.987250</td>
</tr>
<tr>
<td>( a_4 )</td>
<td>( b_5 )</td>
<td>-0.083472</td>
<td>-0.551973</td>
<td>4.366271</td>
</tr>
<tr>
<td>( a_5 )</td>
<td>( b_6 )</td>
<td>-0.141218</td>
<td>0.714746</td>
<td>-3.107855</td>
</tr>
<tr>
<td>( a_6 )</td>
<td>( b_7 )</td>
<td>0.316061</td>
<td>-0.599090</td>
<td>2.457858</td>
</tr>
<tr>
<td>( a_7 )</td>
<td>( b_8 )</td>
<td>-0.162484</td>
<td>0.189659</td>
<td>-2.162352</td>
</tr>
<tr>
<td>( a_8 )</td>
<td>( b_9 )</td>
<td>0.058362</td>
<td>0.252703</td>
<td>1.671379</td>
</tr>
<tr>
<td>( a_9 )</td>
<td>( b_{10} )</td>
<td>0.110235</td>
<td>-0.082504</td>
<td>-0.872042</td>
</tr>
<tr>
<td>( a_{10} )</td>
<td>( b_{11} )</td>
<td>0.016004</td>
<td>-0.270218</td>
<td>0.081960</td>
</tr>
<tr>
<td>( a_{11} )</td>
<td>( b_{12} )</td>
<td>-0.152884</td>
<td>0.145574</td>
<td>0.110935</td>
</tr>
</tbody>
</table>

\( \sigma = 0.72826 \)
Fig. 3.5a The fourth 64 samples of NOON

Fig. 3.5b Initial solution
Fig. 3.5c First iteration
3.3 **TWO-DIMENSIONAL RECURSIVE FILTERS:**

Here we are concerned with fitting the planar pulse response of a two-dimensional recursive digital filter, to given spatial data. The procedure used for the solution of this problem, is a two-dimensional extension of the one given in sec. 3.2.

The discussion given in sec. 3.2.2. still holds, since Parseval's theorem applies also to the two-dimensional case.

3.3.1 **ANALYSIS:**

The transfer function of a two-dimensional recursive digital filter is defined as

\[
H(z_1, z_2) = \frac{\sum_{m=0}^{M_a} \sum_{n=0}^{N_a} a_{mn} z_1^{-m} z_2^{-n}}{\sum_{k=0}^{M_b} \sum_{p=0}^{N_b} b_{kp} z_1^{-k} z_2^{-p}}
\]

\[
= \frac{p(z_1, z_2)}{q(z_1, z_2)} \tag{3.42}
\]

\[
b_{00} = 1.0
\]

\[M_a, N_a, M_b, N_b = \text{arbitrary (but fixed) parameters.}
\]

Define the following set of integers:

\[
I_1 = \{(i, j) : 0 \leq i \leq 2N_1 - 1, 0 \leq j \leq 2N_2 - 1\}
\]
\[ I_2 = \left\{ (i, j) : 0 \leq i \leq N_1, 0 \leq j \leq 2N_2 - 1 \right\} \] (3.43)

In eqn. (3.42) set

\[ z_1 = e^{j\Omega_1 n_1}, \quad z_2 = e^{j\Omega_2 n_2} \] (3.44)

where; \((n_1, n_2) \in I_1\)

\[ \Omega_1 = \frac{\pi}{N_1} \quad \text{and} \quad \Omega_2 = \frac{\pi}{N_2} \]

Hence;

\[ H(e^{j\Omega_1 n_1}, e^{j\Omega_2 n_2}) = \frac{p(e^{j\Omega_1 n_1}, e^{j\Omega_2 n_2})}{Q(e^{j\Omega_1 n_1}, e^{j\Omega_2 n_2})} \]

\[ = \frac{\alpha_{n_1 n_2} + j\delta_{n_1 n_2}}{\sigma_{n_1 n_2} + j\lambda_{n_1 n_2}} \] (3.45)

where;

\[ \alpha_{n_1 n_2} = \sum_{m=0}^{M_a} \sum_{n=0}^{N_a} a_{mn} \cos(\Omega_1 n_1 m + \Omega_2 n_2 n) \]

\[ \delta_{n_1 n_2} = \sum_{m=0}^{M_a} \sum_{n=0}^{N_a} a_{mn} \sin(\Omega_1 n_1 m + \Omega_2 n_2 n) \] (3.46)

\[ \sigma_{n_1 n_2} = \sum_{k=0}^{M_b} \sum_{p=0}^{N_b} b_{kp} \cos(\Omega_1 n_1 k + \Omega_2 n_2 p) \]

\[ \lambda_{n_1 n_2} = \sum_{k=0}^{M_b} \sum_{p=0}^{N_b} b_{kp} \sin(\Omega_1 n_1 k + \Omega_2 n_2 p) \]
Let the required specifications in the spatial frequency-domain be

\[ G(e^{j\Omega_1 n_1}, e^{j\Omega_2 n_2}) = R_{n_1 n_2} + jI_{n_1 n_2} \quad (3.47) \]

where: \((n_1, n_2) \in I_2\)

\[ G(e^{j\Omega_1 n_1}, e^{j\Omega_2 n_2}) \] can be obtained via the spatial impulse response of the desired filter \( g_{k_1, k_2} \{ (k_1, k_2) \in I_1 \} \), by using a two-dimensional DFT.

Following the same procedure as in sec. 3.2, we form the error:

\[ \xi'_{n_1 n_2} = \xi_{n_1 n_2} Q_{n_1 n_2} = (R_{n_1 n_2} + jI_{n_1 n_2})(a_{n_1 n_2} + j\delta_{n_1 n_2}) \]

\[ = (a_{n_1 n_2} + j\delta_{n_1 n_2}) \quad (3.48) \]

where:

\[ \xi_{n_1 n_2} = \frac{p_{n_1 n_2}}{Q_{n_1 n_2}} \quad (3.49) \]

Since, the spatial spectrum of \( h_{n_1 n_2} \) and \( G_{n_1 n_2} \) are both Hermitian symmetrical about the \((0,0)\)-frequency point, we form the following error-measure

\[ E = \sum_{n_1=0}^{N_1} \sum_{n_2=0}^{2N_2-1} \left| \xi'_{n_1 n_2} \right|^2 \]
\[
\sum_{n_1=0}^{N_1} \sum_{n_2=0}^{2N_2-1} \left\{ \sigma_{n_1n_2} R_{n_1n_2} - i_{n_1n_2} \lambda_{n_1n_2} - \alpha_{n_1n_2} \right\}^2 + \left( \lambda_{n_1n_2} R_{n_1n_2} + i_{n_1n_2} i_{n_1n_2} - \delta_{n_1n_2} \right) \right\}^2 \quad (3.50)
\]

Equation (3.50) is now differentiated with respect to each of the unknown coefficients \( \{a_{ij}\} \) and \( \{b_{ij}\} \), and the results set equal to zero. After some manipulation the following equations results:

\[
\begin{bmatrix}
C_1 & C_2 \\
C_2^T & C_3
\end{bmatrix}
\begin{bmatrix}
X
\end{bmatrix}
= 
\begin{bmatrix}
F_1 \\
F_2
\end{bmatrix} \quad (3.51)
\]

or \( \nabla X = F \) \quad (3.52)

where, defining the following relationships

\[
\gamma_{k_1k_2} = \sum_{n_1=0}^{N_1} \sum_{n_2=0}^{2N_2-1} \cos(n_1n_1k_1 + n_2n_2k_2) \quad (3.53)
\]
\[
\beta_{k_1 k_2} = \sum_{n_1=0}^{N_1} \sum_{n_2=0}^{2N_2-1} \left| G(e^{j\Omega_1 n_1}, e^{j\Omega_2 n_2}) \right|^2 
\]

\[
\cos(k_1 \Omega_1 n_1 + k_2 \Omega_2 n_2)
\]

The sub-matrices and vector \( \mathbf{x} \) of equation (3.51) can be written as follows:

\[
C_1 = \begin{bmatrix}
C_{11} & C_{12} & C_{13} & C_{1, M_a + 1} \\
C_{21} & C_{22} & C_{23} & C_{2, M_a + 1} \\
& & & \\
C_{M_a + 1, 1} & C_{M_a + 1, 2} & \cdots & C_{M_a + 1, M_a + 1}
\end{bmatrix}
\]

(3.54)

where:

\[
C_{11} = \begin{bmatrix}
^n0,0 & ^n0,1 & ^n0,2 & \cdots & ^n0,N_a \\
^n0,1 & ^n0,0 & ^n0,1 & ^n0,2 & \cdots & ^n0,N_a \\
^n0,2 & ^n0,1 & ^n0,0 & ^n0,1 & \cdots & ^n0,N_a \\
& & & & & \ddots \\
^n0,N_a & & & & & ^n0,N_a
\end{bmatrix}
\]

(3.55)

\[
C_{ii} \quad 1 \leq i \leq M_a + 1
\]
$$C_{1j} = \begin{bmatrix}
\eta_{j-1,0} & \eta_{j-1,1} & \cdots & \eta_{j-1,N_a} \\
\eta_{j-1,-1} & \eta_{j-1,0} & \eta_{j-1,1} & \cdots \\
\eta_{j-1,-2} & \eta_{j-1,-1} & \eta_{j-1,0} & \cdots \\
\eta_{j-1,-N_a} & & & \\
\end{bmatrix}$$

(3.56)

$C_{1j}$ is defined for $2 \leq j \leq M_a + 1$

$$C_{1j} = C_{2,j+1} = C_{3,j+2} = \cdots \text{ etc.}$$

(3.57)

and:

$$C_{ij} = C_{ji}^T$$

(3.58)

$C_1$ is a symmetrical matrix.

$$C_2 = \begin{bmatrix}
D_{11} & D_{12} & D_{13} & \cdots & D_{1,M_b+1} \\
D_{21} & D_{22} & D_{23} & \cdots & D_{2,M_b+1} \\
D_{31} & D_{32} & D_{33} & \cdots & D_{3,M_b+1} \\
& \ddots & \ddots & \ddots & \ddots \\
D_{M_a+1,1} & D_{M_a+1,2} & \cdots & \cdots & D_{M_a+1,M_b+1} \\
\end{bmatrix}$$

(3.59)

where;
\[ D_{11} = \begin{bmatrix}
\phi_{0,0} - \gamma_{0,0} \\
\phi_{0,1} + \gamma_{0,1} \\
\phi_{0,2} + \gamma_{0,2} \\
\phi_{1,N_a-1} + \gamma_{0,N_a-1}
\end{bmatrix}
\begin{bmatrix}
\phi_{0,0} - \gamma_{0,0} \\
\phi_{0,1} + \gamma_{0,1} \\
\phi_{0,2} + \gamma_{0,2} \\
\phi_{0,N_a} + \gamma_{0,N_a}
\end{bmatrix}
\begin{bmatrix}
\phi_{0,1} - \gamma_{0,1} \\
\phi_{0,2} - \gamma_{0,2} \\
\phi_{0,N_b} - \gamma_{0,N_b}
\end{bmatrix}
\]

(3.60)

\[ D_{ii} = \begin{bmatrix}
\phi_{0,0} - \gamma_{0,0} \\
\phi_{0,1} + \gamma_{0,1} \\
\phi_{0,2} + \gamma_{0,2} \\
\phi_{0,N_a} + \gamma_{0,N_a}
\end{bmatrix}
\]

(3.61)
\[ D_{li} = \begin{bmatrix}
(-\phi_{i-1,0},-\gamma_{i-1,0}) & (-\phi_{i-1,1},-\gamma_{i-1,0}) & \cdots & (-\phi_{i-1,N_{b}},-\gamma_{i-1,N_{b}}) \\
(-\phi_{i-1,-1},-\gamma_{i-1,-1}) & (-\phi_{i-1,0},-\gamma_{i-1,0}) & \cdots & (-\phi_{i-1,1},-\gamma_{i-1,0}) \\
(-\phi_{i-1,-2},-\gamma_{i-1,-2}) & (-\phi_{i-1,-1},-\gamma_{i-1,-1}) & \cdots & (-\phi_{i-1,0},-\gamma_{i-1,0}) \\
\vdots & \vdots & \ddots & \vdots \\
(-\phi_{i-1,-N_{a}},-\gamma_{i-1,-N_{a}}) & \cdots & \cdots & (-\phi_{i-1,0},-\gamma_{i-1,0})
\end{bmatrix}
\]

\[ D_{li} = D_{2,i+1} = D_{3,i+2} = \cdots \text{ etc.} \]
\[ C_3 = \begin{bmatrix} S_{11} & S_{12} & S_{13} & \cdots & S_{1,M_b+1} \\ S_{21} \\ S_{31} \\ S_{M_b+1,1} & S_{M_b+1,2} & \cdots & S_{M_b+1,M_b+1} \end{bmatrix} \] (3.66)

where:

\[ S_{11} = \begin{bmatrix} \beta_{00} & \beta_{01} & \beta_{02} & \cdots & \beta_{0,N_b-1} \\ \beta_{01} & \beta_{00} & \beta_{01} & \beta_{02} & \cdots \\ \beta_{02} & \beta_{01} & \beta_{00} & \beta_{01} & \cdots \\ \cdots & \cdots & \cdots & \cdots & \cdots \\ \beta_{0,N_b-1} \end{bmatrix} \] (3.67)

\[ s_{ii} = \begin{bmatrix} \beta_{0,N_b} \\ S_{11} \end{bmatrix} \] (3.68)
\[ S_{1i} = \begin{bmatrix}
\beta(i-1), -1 & \beta(i-1), 0 & \beta(i-1), 1 & \ldots & \beta(i-1), N_b - 1 \\
\beta(i-1), -2 & \beta(i-1), -1 & \beta(i-1), 0 & \beta(i-1), 1 & \ldots \\
\beta(i-1), -3 & \beta(i-1), -2 & \beta(i-1), -1 & \beta(i-1), 0 & \beta(i-1), 1 \\
\beta(i-1), -N_b & & & & \\
\end{bmatrix} = S_{1i}^T \]  

(3.69)

\[ S_{2i+1} = \begin{bmatrix}
\beta(i-1), 0 & \beta(i-1), 1 & \ldots & \beta(i-1), N_b \\
\ldots & S_{1i} & \ldots \\
\ldots & \ldots & \ldots & \ldots \\
\end{bmatrix} = S_{3i+2} = S_{4i+3} = \ldots \text{ etc.} \]  

(3.70)

\( C_3 \) is a symmetrical matrix.
\[
F_1 = \begin{bmatrix}
\phi_{0,0} - \gamma_{0,0} \\
\phi_{0,1} - \gamma_{0,1} \\
\vdots \\
\phi_{0,N_a} - \gamma_{0,N_a} \\
\phi_{1,0} - \gamma_{1,0} \\
\phi_{1,1} - \gamma_{1,1} \\
\vdots \\
\phi_{1,N_a} - \gamma_{1,N_a} \\
\phi_{M_a, N_a} - \gamma_{M_a, N_a}
\end{bmatrix}
\] (3.71)

\[
F_2 = \begin{bmatrix}
-\beta_{0,1} \\
-\beta_{0,2} \\
\vdots \\
-\beta_{0,N_b} \\
-\beta_{1,0} \\
-\beta_{1,1} \\
\vdots \\
-\beta_{M_b, N_b}
\end{bmatrix}
\] (3.72)
and:

\[ X = \begin{bmatrix} a_{00} \\
a_{01} \\
a_{02} \\
\vdots \\
a_{0N_a} \\
a_{10} \\
a_{11} \\
a_{12} \\
\vdots \\
a_{1N_a} \\
a_{M_a N_a} \\
b_{01} \\
b_{02} \\
b_{0N_b} \\
\vdots \\
b_{M_b N_b} \end{bmatrix} \]
3.3.2 ITERATIVE ALGORITHM:

Following the same procedure as in sec. 3.2.4, we obtain the following iterative algorithm:

$$\left[ X_L \right] = \left[ C_{L-1} \right]^{-1} \left[ F_{L-1} \right]$$

(3.74)

where the submatrices of $\left[ C_{L-1} \right]$ and subvectors of $\left[ F_{L-1} \right]$ are to be interpreted as in sec. 3.3.1, except now the elements are calculated from:

$$\hat{n}_{k_1,k_2,J} = \sum_{n_1=0}^{N_1} \sum_{n_2=0}^{2N_2-1} W_{n_1,n_2} J \cos(k_1^1 n_1 + k_2^0 n_2)$$

$$\hat{\phi}_{k_1,k_2,J} = \sum_{n_1=0}^{N_1} \sum_{n_2=0}^{2N_2-1} W_{n_1,n_2} J R_{n_1,n_2} \cos(k_1^1 n_1 + k_2^0 n_2)$$

$$\hat{\gamma}_{k_1,k_2,J} = \sum_{n_1=0}^{N_1} \sum_{n_2=0}^{2N_2-1} W_{n_1,n_2} J I_{n_1,n_2} \sin(k_1^1 n_1 + k_2^0 n_2)$$

$$\hat{\beta}_{k_1,k_2,J} = \sum_{n_1=0}^{N_1} \sum_{n_2=0}^{2N_2-1} W_{n_1,n_2} J G(e^{j\omega_1 n_1}, e^{j\omega_2 n_2})^2$$

(3.75)

where:

$$W_{n_1,n_2,J} = \frac{1}{\left| Q_J(e^{j\omega_1 n_1}, e^{j\omega_2 n_2}) \right|^2}$$
with:

\[ W_{n_1n_20} = 0.0 \; \varphi(n_1,n_2) \]  \hspace{1cm} (3.77)

3.3.3 COMPUTATIONAL DETAILS:

As in the one-dimensional case certain computational efficiencies may be employed in the algorithm. These are pointed out below.

Initial Solution:

Equations (3.74) and (3.75) are greatly simplified when considering the minimization over the range \((n_1,n_2) \in I_1\). The Eqns. in (3.75) can be written and simplified as follows:

\[
\mathcal{R}_{e}\left\{ \sum_{n_1=0}^{2N_1-1} \sum_{n_2=0}^{2N_2-1} e^{-j(\Omega_1n_1k_1 + \Omega_2n_2k_2)} \right\}
\]

\[
= \mathcal{R}_{e}\left\{ \frac{1 - e^{-j2\pi k_1}}{1 - e^{-j2\pi N_1 k_1}} \cdot \frac{1 - e^{-j2\pi k_2}}{1 - e^{-j2\pi N_2 k_2}} \right\}
\]

\[
= \begin{cases} 
4N_1N_2, & \text{for } k_1=k_2=0 \\
0, & \text{otherwise}
\end{cases} \hspace{1cm} (3.78)
\]
\[ \phi_{k_1k_2} \pm \gamma_{k_1k_2} = \sum_{n_1=0}^{2N_1-1} \sum_{n_2=0}^{2N_2-1} \sum_{n_1=0}^{2N_1-1} \sum_{n_2=0}^{2N_2-1} R_{n_1n_2} \cos(\Omega_1 n_1 k_1 + \Omega_2 n_2 k_2). \]

\[ \pm \sum_{i=0}^{2N_1-1} \sum_{\ell=0}^{2N_2-1} I_{n_1n_2} \sin(\Omega_1 n_1 k_1 + \Omega_2 n_2 k_2). \]

Since: \( R_{n_1n_2} + j I_{n_1n_2} = \sum_{i=0}^{2N_1-1} \sum_{\ell=0}^{2N_2-1} g_{i\ell} e^{-j(\Omega_1 in_1 + \Omega_2 \ell n_2)} \)

Hence; after some manipulation

\[ \phi_{k_1k_2} \pm \gamma_{k_1k_2} = \sum_{i=0}^{2N_1-1} \sum_{\ell=0}^{2N_2-1} g_{i\ell} \left( \frac{-j2\pi (itk_1)}{1 - e^{-j2\pi (\ell k_2)}} \right) \]

Therefore;

\[ \phi_{k_1k_2} - \gamma_{k_1k_2} = 4N_1N_2 g_{k_1k_2} \]

\[ \phi_{k_1k_2} + \gamma_{k_1k_2} = 4N_1N_2 g_{2N_1-k_1, 2N_2-k_2} \]

for \( k_1 > 0 \) and \( k_2 > 0 \)
For $k_2 = 0, k_1 \neq 0$

$$\phi_{0k_2} + \gamma_{0k_2} = 4 N_1 \cdot N_2 g_{0,2N_2-k_2}$$

For $k_1 = 0, k_2 \neq 0$

$$\phi_{k_10} + \gamma_{k_10} = 4 N_1 \cdot N_2 g_{2N_1-k_1,0}$$

$$\phi_{k_1-k_2} + \gamma_{k_1-k_2} = 4 N_1 \cdot N_2 g_{2N_1-k_1,k_2}$$

$$\phi_{-k_1,k_2} + \gamma_{-k_1,k_2} = 4 N_1 \cdot N_2 g_{k_1,2N_2-k_2}$$

Hence, eq. (3.51) is simplified to:

$$x_0 = \begin{bmatrix} 4 N_1 \cdot N_2 & I & -4 N_1 \cdot N_2 R_1 \cdot \xi_1 \\ -4 N_1 \cdot N_2 R_1^T & C_3 & 0 \end{bmatrix}^{-1} \begin{bmatrix} 4 N_1 \cdot N_2 \cdot R_2 \\ F_2 \end{bmatrix}$$

where; $I$ is the identity matrix.

$R_1$ has its sub-matrices defined as:
\[ D_{ll} = \begin{bmatrix}
  g_{0,2N_2-1} & g_{0,2N_2-2} & \cdots & g_{0,2N_2-N_b} \\
  g_{0,0} & g_{0,2N_2-1} & g_{0,2N_2-2} & \cdots \\
  g_{0,1} & g_{0,0} & g_{0,2N_2-1} \\
  g_{0,2} & g_{0,1} & g_{0,0} \\
  g_{0,N_a-1} & & & \\
\end{bmatrix} \] (3.81)

\[ D_{ii} = \begin{bmatrix}
  g_{00} \\
  g_{01} \\
  g_{02} \\
  \vdots \\
  g_{0,N_a} \\
\end{bmatrix} \] (3.82) for \( i \geq 2 \)

\[ D_{li} = \begin{bmatrix}
  g_{2N_1-i+1,0} & g_{2N_1-i+1,2N_2-1} & \cdots & g_{2N_1-i+1,2N_2-N_b} \\
  g_{2N_1-i+1,1} & g_{2N_1-i+1,0} & g_{2N_1-i+1,2N_2-1} & \cdots \\
  g_{2N_1-i+1,2} & g_{2N_1-i+1,1} & g_{2N_1-i+1,0} \\
  \vdots \\
  g_{2N_1-i+1,N_a} & & & \\
\end{bmatrix} \] (3.83)
\[ D_{i+1,2} = D_{i+2,3} = D_{i+3,4} = \cdots \quad \text{etc.} \]
$R_2$ is defined as:

$$
R_2 = \begin{bmatrix}
g_{00} \\
g_{01} \\
g_{02} \\
\vdots \\
g_{0,N_a} \\
g_{1,0} \\
\vdots \\
g_{N_a,N_a}
\end{bmatrix}
$$

(3.88)

$C_3$ and $F_2$ are defined as before except their elements are calculated from

$$
\beta_{k_1,k_2}^{n_1,n_2} = \sum_{n_1=0}^{2N_1-1} \sum_{n_2=0}^{2N_2-1} \left| g(e^{j\omega_1 n_1}, e^{j\omega_2 n_2}) \right|^2 \cos(\omega_1 n_1 k_1 + \omega_2 n_2 k_2)
$$

$$
= \Re \left\{ \sum_{n_1=0}^{2N_1-1} \sum_{n_2=0}^{2N_2-1} \left| G_{n_1 n_2} \right|^2 e^{-j(\omega_1 n_1 k_1 + \omega_2 n_2 k_2)} \right\}
$$

(3.89)
Equation (3.89) could be efficiently computed using the two-dimensional output pruned FFT given in Appendix C.

Defining

\[ X_J = (a_{00 J}, a_{01 J}, \ldots, a_{0N_a J}, \ldots, a_{M N_a J}^T, b_{01 J}, b_{02 J}, \ldots, b_{0N_b J}, b_{10 J}, \ldots, b_{M N_b J}) \]

\[ = (A_J, B_J)^T \] \hfill (3.90)

Equation (3.75) could be partitioned into the two equations:

\[ \begin{bmatrix} R_1^T R_1 - \frac{1}{4N_1 N_2} C_3 \end{bmatrix} B_0 \]

\[ = -R_1^T R_2 - \frac{1}{4N_1 N_2} F_2 \] \hfill (3.91)

and

\[ A_0 = \begin{bmatrix} R_1 B_0 + R_2 \end{bmatrix} \] \hfill (3.92)

The L.H.S. of eqn. (3.91) forms a symmetrical matrix, and hence these set of equations are solved very efficiently using the square-root method.

**L-th Iteration:**

For the iteration \( L > 1 \), equation (3.51) can be partitioned into two equations, similar to equations (3.40). The same computation advantages still apply.
The weights, \( W_{n_1 n_2 J} \), given by equation (3.75) can be calculated using the two dimensional input pruned FFT algorithm given in Appendix B.

The elements of the C and F matrices, given by equation (3.52), are calculated in the following combinations

\[
\begin{align*}
\hat{\eta}_{k_1 k_2 J} & = \Re \left\{ \sum_{n_1=0}^{N_1} \sum_{n_2=0}^{2N_2-1} W_{n_1 n_2 J} e^{-j(k_1 n_1 + k_2 n_2)} \right\} \\
\hat{\phi}_{k_1 k_2 J} & = \Re \left\{ \sum_{n_1=0}^{N_1} \sum_{n_2=0}^{2N_2-1} W_{n_1 n_2 J} (R_{n_1 n_2 J} + I_{n_1 n_2}) e^{-j(k_1 n_1 + k_2 n_2)} \right\} \\
\hat{\gamma}_{k_1 k_2 J} & = \Re \left\{ \sum_{n_1=0}^{N_1} \sum_{n_2=0}^{2N_2-1} W_{n_1 n_2 J} |G_{n_1 n_2}|^2 e^{-j(k_1 n_1 + k_2 n_2)} \right\}
\end{align*}
\]

(3.93)

The calculation of the elements in (3.93) may be computed very efficiently using an output pruned two-dimensional FFT algorithm shown in Appendix C.

A complete Fortran IV, double precision version, program of the previously explained algorithm is given in Appendix E2. It incorporates all the computational
techniques previously explained.

3.3.4 EXAMPLES:

EXAMPLE 1: DESIGNING TO PRESCRIBED SPATIAL IMPULSE RESPONSE

A sixth-order \(N_a = M_a = N_b = M_b = 3\) spatial recursive filter is fitted to the spatial impulse response given by

\[ g_{k_1 k_2} = \begin{cases} 1.0 & 0 \leq \sqrt{k_1^2 + k_2^2} \leq 2 \\ 0 & \text{otherwise} \end{cases} \]

\[
\begin{cases}
(k_1, k_2): 0 \leq k_1 \leq 7, 0 \leq k_2 \leq 7
\end{cases}
\]

The solution is given in Table VI, and Fig. 3.6 (for initial solution, or 1st iteration). The L2 Norm:

\[
Q \triangleq \left( \sum_{i=0}^{7} \sum_{j=0}^{7} (g_{ij} - h_{ij})^2 \right)^{\frac{1}{2}}
\]

is also given in Table VI.

This problem took 1.10 seconds execution time on an IBM 360/65 machine.
<table>
<thead>
<tr>
<th>Initial solution</th>
<th>First iteration</th>
</tr>
</thead>
<tbody>
<tr>
<td>a (1) = 0.123456789</td>
<td>a (1) = 0.123456789</td>
</tr>
<tr>
<td>a (2) = 0.123456789</td>
<td>a (2) = 0.123456789</td>
</tr>
<tr>
<td>a (3) = 0.123456789</td>
<td>a (3) = 0.123456789</td>
</tr>
<tr>
<td>a (4) = 0.123456789</td>
<td>a (4) = 0.123456789</td>
</tr>
<tr>
<td>a (5) = 0.123456789</td>
<td>a (5) = 0.123456789</td>
</tr>
<tr>
<td>a (6) = 0.123456789</td>
<td>a (6) = 0.123456789</td>
</tr>
<tr>
<td>a (7) = 0.123456789</td>
<td>a (7) = 0.123456789</td>
</tr>
<tr>
<td>a (8) = 0.123456789</td>
<td>a (8) = 0.123456789</td>
</tr>
<tr>
<td>a (9) = 0.123456789</td>
<td>a (9) = 0.123456789</td>
</tr>
<tr>
<td>a (10) = 0.123456789</td>
<td>a (10) = 0.123456789</td>
</tr>
</tbody>
</table>

Q = 0.123E-05

Q = 0.935E-04
<table>
<thead>
<tr>
<th>DESIGN MAGNITUDE</th>
</tr>
</thead>
<tbody>
<tr>
<td>1.0 1.0 1.0 0.0 0.0 0.0 0.0 0.0 0.0</td>
</tr>
<tr>
<td>1.0 1.0 1.0 0.0 0.0 0.0 0.0 0.0 0.0</td>
</tr>
<tr>
<td>0.0 0.0 0.0 0.0 0.0 0.0 0.0 0.0 0.0</td>
</tr>
<tr>
<td>0.0 0.0 0.0 0.0 0.0 0.0 0.0 0.0 0.0</td>
</tr>
<tr>
<td>0.0 0.0 0.0 0.0 3.0 0.0 0.0 0.0 0.0</td>
</tr>
</tbody>
</table>

<table>
<thead>
<tr>
<th>DESIGNED MAGNITUDE</th>
</tr>
</thead>
<tbody>
<tr>
<td>1.0 1.0 1.0 -0.0 -0.0 -0.0 -0.0 0.0 0.0</td>
</tr>
<tr>
<td>1.0 1.0 1.0 -0.0 -0.0 -0.0 -0.0 -0.0 0.0</td>
</tr>
<tr>
<td>3.0 -0.0 -0.0 -0.0 -0.0 -0.0 -0.0 -0.0 0.0</td>
</tr>
<tr>
<td>0.0 0.0 0.0 0.0 0.0 0.0 0.0 0.0 0.0</td>
</tr>
<tr>
<td>0.0 0.0 0.0 0.0 3.0 0.0 0.0 0.0 0.0</td>
</tr>
</tbody>
</table>

<table>
<thead>
<tr>
<th>SPEC</th>
</tr>
</thead>
<tbody>
<tr>
<td>-3.0 -3.0 -3.0 3.0 3.0 3.0 3.0 3.0 -3.0</td>
</tr>
<tr>
<td>-3.0 -3.0 -3.0 3.0 3.0 3.0 3.0 3.0 -3.0</td>
</tr>
<tr>
<td>0.0 0.0 0.0 0.0 0.0 0.0 0.0 0.0 0.0</td>
</tr>
<tr>
<td>0.0 0.0 0.0 0.0 0.0 0.0 0.0 0.0 0.0</td>
</tr>
<tr>
<td>0.0 0.0 0.0 0.0 0.0 0.0 0.0 0.0 0.0</td>
</tr>
</tbody>
</table>

Fig. 3.6 Initial solution (or 1st iteration) of Example 1.
EXAMPLE 2: FITTING SPATIAL DATA

Two frames of ERTS data are fitted with sixth order recursive filters \( M_a = N_a = M_b = N_b = 3 \), and the results are given in Tables VII - VIII, and Figs. 3.7 - 3.8.

Execution-time for each problem took 1.10 sec. \( Q \) is defined as in the previous example.
<table>
<thead>
<tr>
<th>Initial solution</th>
<th>First iteration</th>
</tr>
</thead>
<tbody>
<tr>
<td></td>
<td></td>
</tr>
<tr>
<td>( A(1, 1) = -0.1124048148 )</td>
<td>( A(1, 1) = -0.2233503385 )</td>
</tr>
<tr>
<td>( A(1, 2) = -0.8235617846 )</td>
<td>( A(1, 2) = -0.1632366423 )</td>
</tr>
<tr>
<td>( A(1, 3) = 0.2174363643 )</td>
<td>( A(1, 3) = 0.1621599946 )</td>
</tr>
<tr>
<td>( A(1, 4) = -0.1343719950 )</td>
<td>( A(1, 4) = -0.6915812960 )</td>
</tr>
<tr>
<td>( A(2, 1) = -0.1491383653 )</td>
<td>( A(2, 1) = -0.3662839130 )</td>
</tr>
<tr>
<td>( A(2, 2) = -0.2206334937 )</td>
<td>( A(2, 2) = -0.1967871136 )</td>
</tr>
<tr>
<td>( A(2, 3) = 0.1772979118 )</td>
<td>( A(2, 3) = 0.3533230872 )</td>
</tr>
<tr>
<td>( A(2, 4) = -0.1159066745 )</td>
<td>( A(2, 4) = -0.1133355512 )</td>
</tr>
<tr>
<td>( A(3, 1) = -0.3173617033 )</td>
<td>( A(3, 1) = -0.1669269926 )</td>
</tr>
<tr>
<td>( A(3, 2) = -0.1617946903 )</td>
<td>( A(3, 2) = -0.3002419560 )</td>
</tr>
<tr>
<td>( A(3, 3) = -0.5720460357 )</td>
<td>( A(3, 3) = -0.4237666225 )</td>
</tr>
<tr>
<td>( A(3, 4) = 0.2740777961 )</td>
<td>( A(3, 4) = 0.2740777961 )</td>
</tr>
<tr>
<td>( A(4, 1) = -0.2722947680 )</td>
<td>( A(4, 1) = -0.3527465360 )</td>
</tr>
<tr>
<td>( A(4, 2) = -0.2722947680 )</td>
<td>( A(4, 2) = -0.3527465360 )</td>
</tr>
<tr>
<td>( A(4, 3) = 0.1979176760 )</td>
<td>( A(4, 3) = -0.1835023772 )</td>
</tr>
<tr>
<td>( A(4, 4) = -0.2731400520 )</td>
<td>( A(4, 4) = -0.1451501749 )</td>
</tr>
<tr>
<td>( A(1, 1) = -0.1303300009 )</td>
<td>( A(1, 1) = -0.1233666600 )</td>
</tr>
<tr>
<td>( A(1, 2) = -0.4562225189 )</td>
<td>( A(1, 2) = -0.7138161970 )</td>
</tr>
<tr>
<td>( A(1, 3) = -0.1217476397 )</td>
<td>( A(1, 3) = -0.3514018590 )</td>
</tr>
<tr>
<td>( A(1, 4) = -0.1077261385 )</td>
<td>( A(1, 4) = -0.1355167829 )</td>
</tr>
<tr>
<td>( A(2, 1) = -0.5551676138 )</td>
<td>( A(2, 1) = -0.3113095792 )</td>
</tr>
<tr>
<td>( A(2, 2) = -0.2549157799 )</td>
<td>( A(2, 2) = -0.5224827518 )</td>
</tr>
<tr>
<td>( A(2, 3) = -0.1753616125 )</td>
<td>( A(2, 3) = -0.4723513560 )</td>
</tr>
<tr>
<td>( A(2, 4) = -0.1753616125 )</td>
<td>( A(2, 4) = -0.5554713800 )</td>
</tr>
<tr>
<td>( A(3, 1) = -0.2379518205 )</td>
<td>( A(3, 1) = -0.5154849383 )</td>
</tr>
<tr>
<td>( A(3, 2) = -0.2741929975 )</td>
<td>( A(3, 2) = -0.5564547389 )</td>
</tr>
<tr>
<td>( A(3, 3) = -0.1575820360 )</td>
<td>( A(3, 3) = -0.1031663071 )</td>
</tr>
<tr>
<td>( A(3, 4) = -0.4293244300 )</td>
<td>( A(3, 4) = -0.1631352700 )</td>
</tr>
<tr>
<td>( A(4, 1) = -0.3726890039 )</td>
<td>( A(4, 1) = -0.2917741500 )</td>
</tr>
<tr>
<td>( A(4, 2) = -0.1552391795 )</td>
<td>( A(4, 2) = -0.1476105390 )</td>
</tr>
<tr>
<td>( A(4, 3) = 0.1824653319 )</td>
<td>( A(4, 3) = 0.3393250160 )</td>
</tr>
<tr>
<td>( A(4, 4) = -0.8203798700 )</td>
<td>( A(4, 4) = -0.6228832210 )</td>
</tr>
</tbody>
</table>

\[ \eta = 0.484 \times 10^2 \]  
\[ \eta = 0.433 \times 10^1 \]
<table>
<thead>
<tr>
<th>DESIRED MAGNITUDE</th>
<th>DESIGNED MAGNITUDE</th>
<th>ERROR</th>
</tr>
</thead>
<tbody>
<tr>
<td>23.0</td>
<td>22.0</td>
<td>-0.5</td>
</tr>
<tr>
<td>22.0</td>
<td>21.0</td>
<td>-1.0</td>
</tr>
<tr>
<td>21.0</td>
<td>20.0</td>
<td>-1.5</td>
</tr>
<tr>
<td>20.0</td>
<td>19.0</td>
<td>-1.0</td>
</tr>
<tr>
<td>19.0</td>
<td>18.0</td>
<td>-1.0</td>
</tr>
<tr>
<td>18.0</td>
<td>17.0</td>
<td>-1.0</td>
</tr>
<tr>
<td>17.0</td>
<td>16.0</td>
<td>-1.0</td>
</tr>
<tr>
<td>16.0</td>
<td>15.0</td>
<td>-1.0</td>
</tr>
<tr>
<td>15.0</td>
<td>14.0</td>
<td>-1.0</td>
</tr>
<tr>
<td>14.0</td>
<td>13.0</td>
<td>-1.0</td>
</tr>
<tr>
<td>13.0</td>
<td>12.0</td>
<td>-1.0</td>
</tr>
<tr>
<td>12.0</td>
<td>11.0</td>
<td>-1.0</td>
</tr>
<tr>
<td>11.0</td>
<td>10.0</td>
<td>-1.0</td>
</tr>
<tr>
<td>10.0</td>
<td>9.0</td>
<td>-1.0</td>
</tr>
<tr>
<td>9.0</td>
<td>8.0</td>
<td>-1.0</td>
</tr>
<tr>
<td>8.0</td>
<td>7.0</td>
<td>-1.0</td>
</tr>
<tr>
<td>7.0</td>
<td>6.0</td>
<td>-1.0</td>
</tr>
<tr>
<td>6.0</td>
<td>5.0</td>
<td>-1.0</td>
</tr>
<tr>
<td>5.0</td>
<td>4.0</td>
<td>-1.0</td>
</tr>
<tr>
<td>4.0</td>
<td>3.0</td>
<td>-1.0</td>
</tr>
<tr>
<td>3.0</td>
<td>2.0</td>
<td>-1.0</td>
</tr>
<tr>
<td>2.0</td>
<td>1.0</td>
<td>-1.0</td>
</tr>
<tr>
<td>1.0</td>
<td>0.0</td>
<td>-1.0</td>
</tr>
</tbody>
</table>

Fig. 3.7a Initial solution
<table>
<thead>
<tr>
<th>DESIGNED MAGNITUDE</th>
</tr>
</thead>
<tbody>
<tr>
<td>20.0   22.0   22.0   21.0   21.0   21.0   20.0   20.0</td>
</tr>
<tr>
<td>20.0   20.0   19.0   19.0   20.0   20.0   19.0   19.0</td>
</tr>
<tr>
<td>20.0   20.0   19.0   19.0   20.0   20.0   19.0   19.0</td>
</tr>
<tr>
<td>19.0   20.0   22.0   22.0   23.0   22.0   22.0   20.0</td>
</tr>
<tr>
<td>25.0   22.0   21.0   21.0   20.0   20.0   22.0   21.0</td>
</tr>
<tr>
<td>21.0   24.0   22.0   24.0   22.0   22.0   21.0   19.0</td>
</tr>
<tr>
<td>21.0   22.0   22.0   22.0   22.0   21.0   21.0   21.0</td>
</tr>
<tr>
<td>21.0   24.0   21.0   21.0   21.0   20.0   20.0   20.0</td>
</tr>
</tbody>
</table>

<table>
<thead>
<tr>
<th>DESIGNED MAGNITUDE</th>
</tr>
</thead>
<tbody>
<tr>
<td>19.9   21.9   21.5   21.1   20.3   20.7   20.1   19.2</td>
</tr>
<tr>
<td>20.1   19.9   21.0   20.4   19.8   19.5   19.9   19.9</td>
</tr>
<tr>
<td>19.8   20.1   19.3   19.9   20.4   19.9   19.2   18.5</td>
</tr>
<tr>
<td>17.6   20.2   22.1   23.4   22.3   21.4   23.9   19.8</td>
</tr>
<tr>
<td>21.3   22.9   21.9   29.8   20.5   21.5   21.4   20.5</td>
</tr>
<tr>
<td>20.3   22.8   22.4   23.0   22.0   21.7   23.9   23.1</td>
</tr>
<tr>
<td>23.3   22.6   21.3   22.7   21.7   22.0   21.7   20.7</td>
</tr>
<tr>
<td>21.0   23.3   21.3   22.7   21.1   21.0   23.3</td>
</tr>
</tbody>
</table>

<table>
<thead>
<tr>
<th>ERROR</th>
</tr>
</thead>
<tbody>
<tr>
<td>0.2   0.1   0.5   -0.1   0.2   0.3   -0.1   0.3</td>
</tr>
<tr>
<td>-0.1  0.1   -0.3   -0.4   -0.8   -0.2   0.1   0.1</td>
</tr>
<tr>
<td>0.2   -0.1   -0.3   0.1   -0.4   0.1   0.3   0.5</td>
</tr>
<tr>
<td>0.5   -0.2   -0.1   -3.4   0.0   0.6   -0.9   0.2</td>
</tr>
<tr>
<td>-1.3  -0.8   0.2   -0.2   -1.5   0.6   0.5   0.5</td>
</tr>
<tr>
<td>0.7   1.2   -0.4   0.4   0.9   0.3   0.1   1.1</td>
</tr>
<tr>
<td>0.2   -0.6   0.7   -0.1   0.3   0.0   0.3   0.7</td>
</tr>
<tr>
<td>-0.5  0.7   -0.3   0.3   -0.1   -1.0   -0.3</td>
</tr>
</tbody>
</table>

Fig. 3.7b First iteration
<table>
<thead>
<tr>
<th>Initial solution</th>
<th>First iteration</th>
</tr>
</thead>
<tbody>
<tr>
<td>$A(1, 1) = -0.9218629140 \times 10^0$</td>
<td>$A(1, 1) = -0.1257513646 \times 10^2$</td>
</tr>
<tr>
<td>$A(1, 2) = -0.1714230640 \times 10^1$</td>
<td>$A(1, 2) = 0.1489685170 \times 10^1$</td>
</tr>
<tr>
<td>$A(1, 3) = 0.2490727329 \times 10^0$</td>
<td>$A(1, 3) = 0.3143335370 \times 10^1$</td>
</tr>
<tr>
<td>$A(2, 1) = 0.8443237510 \times 10^0$</td>
<td>$A(2, 1) = 0.2482530330 \times 10^0$</td>
</tr>
<tr>
<td>$A(2, 2) = -0.6561548292 \times 10^0$</td>
<td>$A(2, 2) = 0.6514852780 \times 10^0$</td>
</tr>
<tr>
<td>$A(2, 3) = -0.4272374677 \times 10^1$</td>
<td>$A(2, 3) = 0.5353319850 \times 10^1$</td>
</tr>
<tr>
<td>$A(2, 4) = 0.1770323170 \times 10^0$</td>
<td>$A(2, 4) = 0.4131236990 \times 10^0$</td>
</tr>
<tr>
<td>$A(3, 1) = -0.3220668760 \times 10^0$</td>
<td>$A(3, 1) = -0.1757206890 \times 10^1$</td>
</tr>
<tr>
<td>$A(3, 2) = 0.1941128540 \times 10^1$</td>
<td>$A(3, 2) = 0.1921327490 \times 10^1$</td>
</tr>
<tr>
<td>$A(3, 3) = 0.2092589880 \times 10^0$</td>
<td>$A(3, 3) = 0.4293379240 \times 10^0$</td>
</tr>
<tr>
<td>$A(3, 4) = -0.3603051710 \times 10^0$</td>
<td>$A(3, 4) = 0.1943113270 \times 10^0$</td>
</tr>
<tr>
<td>$A(4, 1) = -0.6550616400 \times 10^0$</td>
<td>$A(4, 1) = -0.1714371370 \times 10^0$</td>
</tr>
<tr>
<td>$A(4, 2) = -0.3635070820 \times 10^0$</td>
<td>$A(4, 2) = 0.2332081760 \times 10^1$</td>
</tr>
<tr>
<td>$A(4, 3) = -0.2677585600 \times 10^0$</td>
<td>$A(4, 3) = 0.4253166090 \times 10^0$</td>
</tr>
<tr>
<td>$A(4, 4) = 0.2468513400 \times 10^0$</td>
<td>$A(4, 4) = 0.1413015740 \times 10^0$</td>
</tr>
<tr>
<td>$A(5, 1) = 0.1328359000 \times 10^1$</td>
<td>$A(5, 1) = 0.1022033000 \times 10^1$</td>
</tr>
<tr>
<td>$A(5, 2) = -0.5427775860 \times 10^0$</td>
<td>$A(5, 2) = -0.9206489270 \times 10^0$</td>
</tr>
<tr>
<td>$A(5, 3) = -0.1233462110 \times 10^0$</td>
<td>$A(5, 3) = -0.1923708380 \times 10^0$</td>
</tr>
<tr>
<td>$A(5, 4) = 0.1775757360 \times 10^0$</td>
<td>$A(5, 4) = 0.5693461410 \times 10^0$</td>
</tr>
<tr>
<td>$A(6, 1) = 0.7433676110 \times 10^0$</td>
<td>$A(6, 1) = 0.2339379630 \times 10^0$</td>
</tr>
<tr>
<td>$A(6, 2) = -0.2626186310 \times 10^0$</td>
<td>$A(6, 2) = 0.2949362720 \times 10^0$</td>
</tr>
<tr>
<td>$A(6, 3) = -0.5157613160 \times 10^0$</td>
<td>$A(6, 3) = 0.6868363380 \times 10^0$</td>
</tr>
<tr>
<td>$A(6, 4) = 0.5940266100 \times 10^0$</td>
<td>$A(6, 4) = 0.3863185290 \times 10^0$</td>
</tr>
<tr>
<td>$A(7, 1) = 0.2657827200 \times 10^0$</td>
<td>$A(7, 1) = 0.1325447340 \times 10^0$</td>
</tr>
<tr>
<td>$A(7, 2) = 0.3032331570 \times 10^0$</td>
<td>$A(7, 2) = 0.2111294540 \times 10^0$</td>
</tr>
<tr>
<td>$A(7, 3) = 0.1260621730 \times 10^0$</td>
<td>$A(7, 3) = 0.1519079530 \times 10^0$</td>
</tr>
<tr>
<td>$A(7, 4) = 0.1356713660 \times 10^0$</td>
<td>$A(7, 4) = 0.1366584840 \times 10^0$</td>
</tr>
<tr>
<td>$A(8, 1) = 0.2282625870 \times 10^0$</td>
<td>$A(8, 1) = 0.4456695020 \times 10^0$</td>
</tr>
<tr>
<td>$A(8, 2) = -0.3908646380 \times 10^0$</td>
<td>$A(8, 2) = 0.6487769540 \times 10^0$</td>
</tr>
<tr>
<td>$A(8, 3) = 0.1333565640 \times 10^0$</td>
<td>$A(8, 3) = 0.6156255810 \times 10^0$</td>
</tr>
<tr>
<td>$A(8, 4) = 0.2638247550 \times 10^0$</td>
<td>$A(8, 4) = 0.2312887030 \times 10^0$</td>
</tr>
</tbody>
</table>

\[ q = 0.17 \times 10^0 \]  \[ q = 0.63 \times 10^1 \]
<table>
<thead>
<tr>
<th>DECIMAL MAGNITUDE</th>
<th>DESIGNED MAGNITUDE</th>
<th>ERROR</th>
</tr>
</thead>
<tbody>
<tr>
<td></td>
<td>18.3  17.9  16.9  16.3  21.4  25.3  38.5  29.1</td>
<td>1.7  2.1  1.1  1.7  -1.4  -4.3  -1.5  1.9</td>
</tr>
<tr>
<td></td>
<td>20.2  20.7  15.7  19.8  17.7  18.1  18.9  22.3</td>
<td>0.2  -0.7  1.3  0.2  2.2  3.2  3.9  2.1</td>
</tr>
<tr>
<td></td>
<td>18.1  19.3  20.7  20.3  21.5  22.3  21.7  20.3</td>
<td>1.9  1.7  -6.7  -3.2  0.5  1.1  4.3  2.7</td>
</tr>
<tr>
<td></td>
<td>20.2  20.0  17.6  19.6  19.8  19.5  21.1  23.3</td>
<td>2.2  2.6  -2.2  -3.2  3.2  3.2  3.2  3.2</td>
</tr>
<tr>
<td></td>
<td>20.2  21.6  22.2  21.7  20.4  18.7  18.6  17.3</td>
<td>0.6  -0.9  -3.5  -3.1  0.7  0.2  1.3  2.5</td>
</tr>
<tr>
<td></td>
<td>20.5  19.0  19.6  21.6  22.6  24.0  23.0  22.3</td>
<td>4.3  -3.1  -2.0  -6.2  -9.0  -1.3  -3.3</td>
</tr>
</tbody>
</table>

*Fig. 3.8a Initial solution*
<table>
<thead>
<tr>
<th>DESIGNED MAGNITUDE</th>
</tr>
</thead>
<tbody>
<tr>
<td>20.0</td>
</tr>
<tr>
<td>20.1</td>
</tr>
<tr>
<td>20.0</td>
</tr>
<tr>
<td>21.0</td>
</tr>
<tr>
<td>21.0</td>
</tr>
<tr>
<td>21.0</td>
</tr>
<tr>
<td>24.0</td>
</tr>
</tbody>
</table>

<table>
<thead>
<tr>
<th>DESIGNED MAGNITUDE</th>
</tr>
</thead>
<tbody>
<tr>
<td>20.2</td>
</tr>
<tr>
<td>19.6</td>
</tr>
<tr>
<td>23.5</td>
</tr>
<tr>
<td>21.3</td>
</tr>
<tr>
<td>20.7</td>
</tr>
<tr>
<td>19.6</td>
</tr>
<tr>
<td>21.3</td>
</tr>
<tr>
<td>22.2</td>
</tr>
</tbody>
</table>

| 0.2 | 0.1 | 0.6 | 1.4 | -0.1 | -1.5 | 0.9 | 1.4 |
| 0.4 | 0.4 | 0.1 | -0.9 | -2.2 | -0.7 | 0.7 | -1.0 |
| 0.5 | 0.5 | 0.1 | -0.8 | 0.1 | 1.4 | 0.8 | 0.5 |
| 0.3 | 0.3 | 0.0 | 0.7 | 0.5 | -3.3 | 0.5 | -3.4 |
| 0.1 | 0.1 | 0.7 | 0.3 | 0.7 | 0.3 | -0.7 | 0.3 |
| 1.4 | 0.2 | 1.1 | 0.7 | 1.2 | 0.2 | -1.6 | -0.2 |
| 1.3 | 0.9 | 0.7 | 1.4 | 1.2 | 0.2 | -1.6 | -0.2 |
| 1.3 | 0.2 | 0.3 | -1.9 | -3.4 | 0.3 | -1.7 |

Fig. 3.8b First iteration
3.4 SUMMARY:

A technique has been presented for the identification of parameters of a recursive digital filter from a given impulse response. A weighted square error measure in the frequency domain is minimized, to obtain an initial solution of the parameters. An iterative refinement scheme is also presented. Computational efficiency has been emphasized throughout this chapter and the final algorithm, as presented, is fast.

Present advances in hardware realization of FFT algorithms, could be implemented directly in this algorithm further improving computational efficiencies. This advantage, is not directly applicable to time-domain approaches.
CHAPTER IV

STABLE RECURSIVE FILTER DESIGN, GIVEN ARBITRARY MAGNITUDE-FREQUENCY SPECIFICATIONS

4.1 INTRODUCTION:

Recursive digital filters can be designed to approximate arbitrary magnitude-frequency specifications by minimizing an error function. For example, the sum of squared differences between desired magnitude specifications and the magnitude of the filter transfer function, at specified frequency points. This is a non-linear problem and therefore dictates the use of non-linear minimization algorithms. One of the most powerful non-linear minimization algorithms, that has been widely used for problems of this nature, is the Fletcher-Powell conjugate gradient method. This is an unconstrained variable minimization routine that requires the calculation of the gradients of the error function with respect to elements of the parameter vector. In order to employ this algorithm in filter design, constraints should be implemented on the filter variables in order to achieve a stable design. One method [29] is to force these constraints by variable substitution. The error function is minimized with respect to a new set of variables, and the original variables are obtained at the outcome of the minimization algorithm by substitution.
A technique is presented in this chapter for designing two-dimensional recursive filters based on an extension of a one-dimensional variable substitution method [21] - [23].

For clarity, the one-dimensional technique will be briefly reviewed in the next section.

4.2 ONE-DIMENSIONAL RECURSIVE FILTER DESIGN:

4.2.1 STATEMENT OF THE PROBLEM:

Consider the transfer function of a recursive digital filter \( Y(z, X_B) \), where \( X_B \) is a bounded vector of the unknown filter parameters. It is required to find \( X_B \) (given a fixed filter order), such that \( |Y(e^{j\Omega_i}, X_B)| \) best approximates a prescribed magnitude-frequency response \( Y^d_i \), specified at discrete frequency samples \( \Omega_i \).

\[
\Omega_i = \frac{\Omega_i \pi}{T} \, r/s
\]

where \( 0 \leq \Omega_i \leq 1 \).

The bounding on \( X_B \) should be such that the designed filter is stable.

4.2.2 PERFORMANCE CRITERION:

The vector of unknown bounded parameters \( X_B \), can be obtained by minimizing the performance criterion

\[
Q = \sum_{i=1}^{M} (|Y(e^{j\Omega_i}, X_B)| - Y^d_i)^2
\]

with respect to \( X_B \). However, in order to achieve a stable filter, \( X_B \) should be constrained such that the poles of \( Y(z, X_B) \) all lie within the region \( |z| < 1 \).
4.2.3 CONSTRAINT IMPLEMENTATION:

Consider the transfer function to be defined as cascades of second-order filters

\[ Y(z, X_B) = A \prod_{k=1}^{K} \frac{z^2 + \alpha_k z + \beta_k}{z^2 + \gamma_k z + \delta_k} \]  \hspace{1cm} (4.2)

where \( X_B = (\alpha_k, \beta_k, \gamma_k, \delta_k; A)^T \) \hspace{1cm} (4.3)

For a stable filter configuration the poles of \( Y(z, X_B) \) should be constrained to the unit circle \(|z| < 1\).

Suppose we change the boundary conditions on the poles by mapping the z-plane into a p-plane such that the interior of the unit circle in the z-plane is mapped into the left hand side of the p-plane. Our boundary conditions are now realized by restricting the mapped poles to lie in the left half p-plane. A suitable complex mapping function is the bi-linear transform

\[ z = \frac{1 + p}{1 - p} \]  \hspace{1cm} (4.4)

By substituting (4.4) in (4.2) and rearranging terms we may write

\[ G(p) = A \prod_{k=1}^{K} \frac{p^2 + x_{1k}p + x_{2k}}{p^2 + y_{1k}p + y_{2k}} \]  \hspace{1cm} (4.5)

where
\[ E = a \prod_{k=1}^{K} \frac{1 - a_k + \beta_k}{1 - \gamma_k + \delta_k} \]

\[ x_{1k} = \frac{2 - 2\beta_k}{1 - a_k + \beta_k} \]
\[ x_{2k} = \frac{1 + a_k + \beta_k}{1 - a_k + \beta_k} \] (4.6)

\[ y_{1k} = \frac{2 - 2\delta_k}{1 - \gamma_k + \delta_k} \]
\[ y_{2k} = \frac{1 + \gamma_k + \delta_k}{1 - \gamma_k + \delta_k} \]

For the poles of (4.5) to lie in the left half of the \( p \)-plane, the following constraints should be met

\[ y_{1k} > 0, \ y_{2k} > 0 \] (4.7)

Also, for minimum phase

\[ x_{1k} > 0, \ x_{2k} > 0 \] (4.8)

These constraints can be imposed by the following variable substitutions:
\[ y_{1k} = \frac{-2}{c_k}, \quad y_{2k} = \frac{b_k}{c_k}, \quad x_{1k} = \frac{c_k}{d_k}, \quad \text{and} \quad x_{2k} = \frac{d_k}{d_k} \quad (4.9) \]

By substituting (4.4) and (4.9) in (4.5) we obtain

\[ y(z,X_u) = E \prod_{k=1}^{K} \frac{z^2(1+a_k^2+b_k^2) + z(-2+2b_k^2) + (1-a_k^2+b_k^2)}{z^2(1+c_k^2+d_k^2) + z(-2+2d_k^2) + (1-c_k^2+d_k^2)} \quad (4.10) \]

\[ = E \cdot H(z,\psi_u) \quad (4.11) \]

The poles and zeros of (4.10) are now constrained to lie inside the unit circle \(|z| < 1\).

\( x_u \) and \( \psi_u \) are unconstrained parameter vectors defined as:

\[ x_u = (a_k, b_k, c_k, d_k; A)^T \quad (4.12) \]

\[ \psi_u = (a_k, b_k, c_k, d_k)^T \]

\( x_B \) and \( x_u \) are related as follows:

\[ a_k = \frac{(-2+2b_k^2)/(1+a_k^2+b_k^2)}{1+c_k^2+d_k^2} \]

\[ b_k = \frac{(1-a_k^2+b_k^2)/(1+a_k^2+b_k^2)}{1+c_k^2+d_k^2} \quad (4.13) \]

\[ y_k = \frac{(-2+2d_k^2)/(1+c_k^2+d_k^2)}{1+c_k^2+d_k^2} \]

\[ \delta_k = \frac{(1-c_k^2+d_k^2)/(1+c_k^2+d_k^2)}{1+c_k^2+d_k^2} \]

\[ A = E \prod_{k=1}^{K} \frac{1+a_k^2+b_k^2}{1+c_k^2+d_k^2} \]
\( X_u \) can be obtained by minimizing the performance criterion

\[
Q = \sum_{i=1}^{M} |E^H(e^{j\Omega_i}, \psi_u) - y_i^d|^2
\]

(4.14)

with respect to \( \psi_u \) and \( E \), using the unconstrained minimization algorithm of Fletcher and Powell. It is possible to eliminate \( E \) as an unknown parameter \([18]\), by differentiating \( Q \) with respect to \( E \) and equating to zero. Optimum \( E (E^*) \) is given by

\[
E^* = \frac{\sum_{i=1}^{M} |H_i|^2 y_i^d}{\sum_{i=1}^{M} |H_i|^2}
\]

(4.15)

At each function evaluation the value of \( E^* \) is calculated from the current value of \( \psi_u \) and used in place of \( E \) in (4.14).

At the outcome of the algorithm \( X_B \) is calculated from \( \psi_u \) and \( E^* \) by the relations given in (4.13).

4.3 TWO-DIMENSIONAL RECURSIVE FILTER DESIGN:

A method is presented for designing a class of second and fourth order two-dimensional filters which are guaranteed to be stable, and whose magnitude-frequency response approximates a prescribed response in a mean square-error sense.
We shall use Ansell's stability theorem, discussed in Chapter I, to find the constraints required on the parameters of the chosen class of filter for stability. The variable substitution method is then used to enforce these constraints.

4.3.1 STATEMENT OF THE PROBLEM:

Given the transfer function of a two-dimensional recursive digital filter \( Y(z_1, z_2, x_B) \) where \( x_B \) is a bounded vector of the unknown filter parameter. It is required to find \( x_B \) (given a fixed filter order), such that

\[
|Y(e^{-j\Omega_{1i}}, e^{-j\Omega_{2\ell}}, x_B)|
\]

best approximates a prescribed two-dimensional magnitude-frequency response \( Y_{i\ell}^d \), specified on a frequency mesh \( \{w_{1i}\}_{i=1}^M \) and \( \{w_{2\ell}\}_{\ell=1}^N \), where

\[
w_{1i} = \frac{\Omega_{1i}}{A} \quad \text{radians/unit length}
\]

and

\[
w_{2\ell} = \frac{\Omega_{2\ell}}{B} \quad \text{radians/unit length}
\]

and, \(-1 \leq \Omega_{1i} \leq 1\), \(0 \leq \Omega_{2\ell} \leq 1\), \(A\) and \(B\) are sampling intervals in the vertical and horizontal directions of the spatial domain.

\(x_B\) should be bounded such that the designed filter is stable.

Two types of stable filters are presented in the next two sections.
4.3.2  **HUANG'S SPECIAL CLASS OF SECOND ORDER FILTERS** [4]:

Consider the second order polynomial in $p_1$ and $p_2$:

$$F(p_1, p_2) = p_1^2 + ap_2 + bp_1p_2 + c \quad (4.16)$$

Applying Ansell's stability conditions (see page 10).

**First Condition:** $F(jw, p_2) = jw + ap_2 + bjwp_2 + c = 0$.

Therefore:

$$\Re p_2 = -\frac{ca + w^2b}{a^2 + w^2b^2} < 0 \quad (4.17)$$

**Second Condition:** $F(p_1, 1) = p_1 + a + bp_1 + c = 0$.

Therefore:

$$\Re p_1 = -\frac{a + c}{1 + b} < 0 \quad (4.18)$$

Both conditions can be satisfied for any of the following conditions:

1) $a > 0$
   
   $b > 0$
   
   $c > 0 \quad (4.19a)$

2) $a \geq 0$
   
   $b > 0$
   
   $c > 0 \quad (4.19b)$

3) $a > 0$
   
   $b \geq 0$
   
   $c > 0 \quad (4.19c)$
4) \(a > 0\)
\(b > 0\)
\(c \geq 0\) \hspace{1cm} (4.19d)

These constraints can be satisfied by the variable substitution:

\[a = \alpha^2\]
\[b = \beta^2\]
\[c = \gamma^2\] \hspace{1cm} (4.20)

In equation (4.20) at most one of the parameters \(\alpha, \beta, \gamma\) can be allowed to go to zero for (4.16) to satisfy Ansell's stability conditions.

Hence, the second-order polynomial

\[\Phi(p_1, p_2) = p_1 + \alpha^2 p_2 + \beta^2 p_1p_2 + \gamma^2\] \hspace{1cm} (4.21)

with, \(\alpha, \beta, \gamma \neq 0\), or \(\alpha = 0, \beta, \gamma \neq 0\), or \(\beta = 0, \alpha, \gamma \neq 0\), or \(\gamma = 0, \alpha, \beta \neq 0\) \hspace{1cm} (4.22)

satisfies Ansell's stability theorem.

Therefore, the transfer function

\[\chi(z_1, z_2, x_u) = A \prod_{k=1}^{K} \frac{p_1 + a_k^2 p_2 + b_k^2 p_1p_2 + c_k^2}{p_1 + d_k^2 p_2 + e_k^2 p_1p_2 + f_k^2}\] \hspace{1cm} (4.23)

with, \(p_1 = \frac{1 - z_1}{1 + z_1}\); and \(p_2 = \frac{1 - z_2}{1 + z_2}\) \hspace{1cm} (4.24)
is a stable recursive filter with minimum phase, provided that any of the following conditions are met:

\[ d_k, e_k, f_k \neq 0, \text{ or } d_k = 0, e_k, f_k \neq 0, \text{ or} \]

\[ e_k = 0, d_k, f_k \neq 0, \text{ or } f_k = 0, d_k, e_k \neq 0 \]

and similarly for \( a_k, b_k, c_k \).

\( X_u \) is the unconstrained parameter vector given by:

\[ X_u = (a_k, b_k, c_k, d_k, e_k, f_k, A)^T \]

Substituting (4.24) in (4.23) we obtain

\[ Y(z_1, z_2, X_u) = A \prod_{k=1}^{K} \frac{N_k(z_1, z_2)}{D_k(z_1, z_2)} \]

where:

\[ U_k(z_1, z_2) = (1-z_1)(1+z_2) + a_k^2(1-z_2)(1+z_1) \]

\[ + b_k^2(1-z_1)(1-z_2) + c_k^2(1+z_1)(1+z_2) \]

\[ D_k(z_1, z_2) = (1-z_1)(1+z_2) + d_k^2(1-z_2)(1+z_1) \]

\[ + e_k^2(1-z_1)(1-z_2) + f_k^2(1+z_1)(1+z_2) \]

Equation (4.27) can be written in the form
\[ y(z_1, z_2, x_B) = \lambda \prod_{k=1}^{K} \frac{z_2^{(a_k z_1 + \beta_k)} + (\gamma_k z_1 + \delta_k)}{z_2^{(\lambda_k z_1 + \rho_k)} + (\sigma_k z_1 + \tau_k)} \]  

(4.29)

where:  
\[ x_B = (a_k, \beta_k, \gamma_k, \delta_k, \lambda_k, \rho_k, \sigma_k, \tau_k, \lambda) \]  

(4.30)

is the bounded parameter vector of the filter coefficients.  

\( x_B \) can be calculated from \( x_u \) from the relations:

\[ a_k = -1 + a_k^2 + b_k^2 + c_k^2 \]

\[ \beta_k = 1 - a_k^2 - b_k^2 + c_k^2 \]

\[ \gamma_k = -1 + a_k^2 - b_k^2 + c_k^2 \]

\[ \delta_k = 1 + a_k^2 + b_k^2 + c_k^2 \]

\[ \lambda_k = -1 - d_k^2 + e_k^2 + f_k^2 \]

\[ \rho_k = 1 - d_k^2 - e_k^2 + f_k^2 \]

\[ \sigma_k = -1 + d_k^2 - e_k^2 + f_k^2 \]

\[ \tau_k = 1 + d_k^2 + e_k^2 + f_k^2 \]

(4.31)
4.3.3 **STABLE FOURTH ORDER FILTERS:**

The class of second-order filter, discussed in the previous section, can be cascaded to form any even order of filter. There are, obviously, limitations to the type of magnitude response that may be designed using this type of filter. If we set either of the complex variables, \( p_1 \) or \( p_2 \), to zero, the roots of the denominator polynomial will be real. Thus, along either axis in the \( \Omega_1 - \Omega_2 \) plane, we are effectively fitting an arbitrary one-dimensional filter response with a filter which can have only real poles. This will definitely limit the type of response near the two axes. To avoid this limitation we introduce a fourth-order filter which is cascaded from two conjugate second-order sections, i.e. each second-order section has, in general, complex coefficients.

Let us consider a general form of a fourth-order polynomial in \( p_1 \) and \( p_2 \):

\[
D(p_1, p_2) = (p_1 + ap_2 + bp_1p_2 + c)(p_1 + a^*p_2 + b^*p_1p_2 + c^*)
\]

(4.32)

where; \( a, b, c \) are complex and \( * \) represents complex conjugate.

Let \( a = a_r + ja_i \), \( b = b_r + jb_i \), \( c = c_r + jc_i \)

(4.33)
Applying the two conditions of Ansell's stability theorem to each of the two factors in (4.32) we obtain the following:

From condition 1)

\[
\frac{c_r a_r + w^2 b_r + c_i a_i + w(b_c c_r - c_i b_r - a_i)}{(a_r + w b_i)^2 + (w b_r + a_i)^2} > 0
\]

and from condition 2)

\[
\frac{(a_r + c_r)(1 + b_r) + b_i(a_i + c_i)}{(1 + b_r)^2 + b_i^2} > 0
\]

(4.34)

(4.35)

For Huang's special class of second-order filter \((a_i = b_i = c_i = 0)\), the two inequalities (4.34) and (4.35) are reduced to the inequalities (4.17) and (4.18) respectively.

Let us examine the following conditions:

\[
a_i = b_i = b_r = 0
\]

(4.36)

For these conditions inequalities (4.34) and (4.35) are reduced to:

\[
\frac{c_r}{a_r} > 0
\]

(4.37)

\[
a_r + c_r > 0
\]

(4.38)
Inequalities (4.37) and (4.38) can only be satisfied simultaneously, by the constraints:

\[ c_r > 0 \]  \hspace{1cm} (4.38)

\[ a_r > 0 \]

and these constraints can easily be implemented by the variable substitution:

\[ c_r = x_1^2 \]  \hspace{1cm} (4.39)

\[ a_r = x_2^2 \]

provided \( x_1, x_2 \neq 0 \).

Thus the recursive filter having the transfer function:

\[
Y(z_1, z_2, x_u) = A \prod_{k=1}^{K} \frac{(p_1 + y_{1k}^2 p_2 + y_{2k}^2 + j y_{3k}^2)(p_1 + y_{1k}^2 p_2 + y_{2k}^2 - j y_{3k}^2)}{(p_1 + x_{1k}^2 p_2 + x_{2k}^2 + j x_{3k}^2)(p_1 + x_{1k}^2 p_2 + x_{2k}^2 - j x_{3k}^2)}
\]

where \( p_1 \) and \( p_2 \) are as defined in (4.24), is guaranteed to be stable and have minimum phase.

\( X_u \) is the unconstrained parameter vector defined as:

\[
X_u = (y_{1k}, y_{2k}, y_{3k}, x_{1k}, x_{2k}, x_{3k}; A)^T \]  \hspace{1cm} (4.41)
Although this form of filter does not have an interactive term, \( p_1 p_2 \), in each smallest factor of its denominator and numerator, it does have the property of yielding complex poles and zeros for \( p_1 = 0 \) or \( p_2 = 0 \). This form of filter should produce superior fits, in the region of the \( \Omega_1-\Omega_2 \) plane axes, than Huang's special class of filter.

The filter has the added advantage that very good starting points may be obtained from Shank's 'rotated' filter design \([3]\). If \((a_r, a_i)\) and \((b_r, b_i)\) are the complex zeros and poles of a second-order one-variable filter (in \( p_2 \)), and \( \delta \) is the angle by which the frequency response of this one-variable filter is rotated in the \( \Omega_1-\Omega_2 \) plane, then the starting point \( x_{u0} \) of the filter given by equation (4.40) can be calculated from:

\[
\begin{align*}
y_1 &= \sqrt{\frac{-\cos \delta}{\sin \delta}}, \quad y_2 = \sqrt{\frac{-a_r}{\sin \delta}}, \quad y_3 = \frac{-a_i}{\sin \delta} \\
x_1 &= \sqrt{\frac{-\cos \delta}{\sin \delta}}, \quad x_2 = \sqrt{\frac{-b_r}{\sin \delta}}, \quad x_3 = \frac{-b_i}{\sin \delta}
\end{align*}
\]

(4.42)

for any cascade.

The parameters \((a_r, a_i)\), \((b_r, b_i)\) can be obtained from a one-dimensional filter design, as will be demonstrated in later examples.
The number of iterations required to minimize the error-criterion, for the two-dimensional filter, should be greatly reduced by using the starting point for $X_u$ calculated from (4.42).

As in the previous section, equation (4.40) can be written as:

$$y(z_1, z_2, X_B) = A \prod_{k=1}^{K} \sum_{i=0}^{2} \sum_{j=0}^{2} a_{ijk} z_1^{-i} z_2^{-j}$$

$$2 \sum_{i=0}^{2} \sum_{j=0}^{2} b_{ijk} z_1^{-i} z_2^{-j}$$

(4.43)

where, $X_B$ is the bounded parameter vector defined as

$$X_B = \begin{bmatrix} \{a_{ijk}\}, \{b_{ijk}\}; & A \end{bmatrix}^T$$

(4.44)

$X_B$ is related to $X_u$ by the bilinear transformations defined in (4.24).

It is possible to derive other forms of stable filter from inequalities (4.34) and (4.35).

Inequality (4.34) can be written as

$$\frac{(a_c r + a_i c_i) + w^2 b_r}{(a_r + w b_i)^2 + (w b_r + a_i)^2} > \frac{|w(b_i c_r - c_i b_r - a_i)|}{(a_r + w b_i)^2 + (w b_r + a_i)^2}$$

(4.45)
Examining this inequality, we note that the numerator of its L.H.S. is a second-order equation in \( w \):

\[
y = (a_r c_r + a_i c_i) + w^2 b_r
\]  

(4.46)

and the numerator of its R.H.S. represents the equations of two straight lines in \( w \):

\[
y = \pm w(b_i c_r - c_i b_r - a_i)
\]  

(4.47)

Now let us assume that

\[
b_r > 0
\]  

(4.48)

\[
(a_r c_r + a_i c_i) > 0
\]  

(4.49)

Inequality (4.45) is satisfied if the straight line:

\[
y = w|b_i c_r - c_i b_r - a_i|
\]  

(4.50)

has its slope less than the slope of the tangent to equation (4.46), passing through the origin and lying in the \((+y,+w)\) quadrant, (see Fig. 4.1).
Fig. 4.1 Condition for inequality (4.45) to be satisfied.

The tangents to (4.46) passing through the origin are given by:

\[ \frac{\partial y}{\partial w} = \pm 2 \sqrt{b_r (a_r c_r + a_i c_i)} \]  \hspace{1cm} (4.51)

Hence the inequality given by (4.45) is satisfied if (4.48), (4.49) and

\[ 2 \sqrt{b_r (a_r c_r + a_i c_i)} > |b_i c_r - c_i b_r - a_i| \]  \hspace{1cm} (4.52)

are satisfied.
If \( b_r = 0 \), then inequality (4.45) becomes

\[
\frac{a_r c_r + a_i c_i}{(a_r + wb_i)^2 + a_i^2} < \frac{|w(b_i c_r - a_i)|}{(a_r + wb_i)^2 + a_i^2} \tag{4.52}
\]

and this is satisfied if, and only if:

\[
b_i c_r = a_i \tag{4.53}
\]

and

\[a_r c_r + a_i c_i > 0 \tag{4.54}\]

The inequalities (4.48), (4.49) and (4.52), or (4.53) and (4.54), together with the inequality (4.35) form conditions, independent of \( w \), satisfying Ansell's theorem for the fourth-order polynomial given by equation (4.32). It does not seem that variable substitution methods can be used to implement general constraints on the variables, such as to satisfy all the inequalities given for \( h_r > 0 \), for all possible classes. However, we can derive further classes of filters where variable substitution can be used. For example, consider the following two classes:

1) \( a_i = b_i = 0 \):

Applying these conditions, the inequalities (4.35), (4.48), (4.49) and (4.52) are simplified to:
\[
\frac{a_r + c_r}{1 + b_r} > 0 \quad (4.55)
\]
\[
b_r > 0 \quad (4.56)
\]
\[
a_r c_r > 0 \quad (4.57)
\]
\[
2 \sqrt{b_r a_r c_r} > |c_1 b_r| \quad (4.58)
\]

From (4.55) and (4.56) we can write
\[
a_r + c_r > 0 \quad (4.59)
\]

and from (4.58) we have
\[
a_r c_r > \frac{c_1^2 b_r}{4} \quad (4.60)
\]

The constraint on \(b_r\) can be satisfied by the variable substitution
\[
b_r = x_1^2 \quad (4.61)
\]

\[(x_1 \neq 0)\]

Inequality (4.60) can be satisfied by the substitution
\[
a_r c_r = x_2^2 + \frac{c_1^2 x_1^2}{4} \quad (4.62)
\]

\[(x_2 \neq 0)\]
From inequality (4.57) and (4.59) we conclude that:

\[ a_r > 0 \]  
\[ c_r > 0 \]  
\[ (4.63) \]
\[ (4.64) \]

If we let:

\[ b_r = x_3 \]  
\[ (4.65) \]

then, from (4.62) we have

\[ c_r = \frac{x_2^2 + c_i^2 x_1^2 / 4}{x_3} \]  
\[ (4.66) \]

Therefore, from (4.61), (4.65) and (4.66) we can write the polynomial in \( p_1 \) and \( p_2 \) that satisfies Ansell's theorem. This is given by the multiplication of the two polynomials inherent in the following:

\[ p_1 + x_3^2 p_2 + x_1^2 p_1 p_2 + \frac{x_2^2 + x_4^2 x_1^2 / 4}{x_3} \pm j x_4 \]  
\[ (4.67) \]

2) \[ b_r = 0 \]

For this class of filter inequality (4.35) is simplified to:

\[ \frac{(a_r + c_r)^2 + b_i (a_i + c_i)}{1 + b_i^2} > 0 \]  
\[ (4.68) \]
From (4.53) and (4.54) we have

\[ c_r = \frac{a_i}{b_i} \]  \hspace{1cm} (4.69)

and

\[ a_r c_r > -a_i c_i \]  \hspace{1cm} (4.70)

Inequality (4.68) can be written as:

\[ (a_r + c_r) > -b_i(a_i + c_i) \]  \hspace{1cm} (4.71)

Substituting (4.69) in (4.70) and (4.71) we obtain the following inequalities:

\[ a_r > -b_i c_i \]  \hspace{1cm} (4.72)

\[ a_r > -b_i(a_i + c_i) - \frac{a_i}{b_i} \]  \hspace{1cm} (4.73)

From (4.72) and (4.73) we can write:

\[ a_r > \frac{1}{b_i} \left\{ b_1 a_i + \frac{a_i}{b_i} - b_i c_i - b_i(a_i + c_i) - \frac{a_i}{b_i} \right\} \]  \hspace{1cm} (4.74)

which can be satisfied by the substitution:

\[ a_r = x_1^2 + \frac{1}{b_i} \left\{ b_1 a_i + \frac{a_i}{b_i} - b_i c_i - b_i(a_i + c_i) - \frac{a_i}{b_i} \right\} \]  \hspace{1cm} (4.75)

\[ (x_1 \neq 0) \]
Therefore, the filter having its denominator formed by multiplying the two polynomials inherent in the following:

\[ P_1 + (a_r \pm ja_i)P_2 \pm jb_iP_1P_2^* + \frac{a_i}{b_i} \pm jc_i \]  \hspace{1cm} (4.76)

where, \( a_r \), is given by (4.75), is guaranteed to be stable.

4.3.4 PERFORMANCE CRITERION:

We shall choose a squared-error criterion of the form:

\[ Q = \sum_{\ell=1}^{M} \sum_{i=1}^{N} (A_i \|H(z_{1i}, z_{2\ell}, \psi_u)\| - y_{il})^2 \]  \hspace{1cm} (4.77)

where:

\[ z_{1i} = e^{-j\Omega_{1i}} \quad z_{2\ell} = e^{-j\Omega_{2\ell}} \]  \hspace{1cm} (4.78)

\( \Omega_{1i} \) and \( \Omega_{2\ell} \) are the spatial frequencies in terms of their Nyquist rates, defined as in sec. 4.3.1.

\( \psi_u \) is the vector of unbounded parameters.

\( y_{il} \) is the desired magnitude specifications in the spatial frequency domain.

\( M \times N \) – Number of prescribed samples of \( y_{il}^d \).

\( A \), as in the one-dimensional case, can be expressed in terms of the other parameters, by finding its optimum value \( A^* \) for a given \( \psi_u \). \( A^* \) is given by:
\[
A^* = \frac{\sum_{\ell=1}^{M} \sum_{i=1}^{N} y_{i\ell}^d |H_{i\ell}|}{\sum_{\ell=1}^{M} \sum_{i=1}^{N} |H_{i\ell}|^2}\]

(4.79)

The Fletcher-Powell algorithm requires the calculation of the gradient of \( Q \) with respect to \( \psi_u \):

\[
\frac{\partial Q}{\partial \psi_u} = 2A^* \sum_{\ell=1}^{M} \sum_{i=1}^{N} (A^* |H_{i\ell}| - y_{i\ell}^d) \frac{\partial |H_{i\ell}|}{\partial \psi_u}
\]

(4.80)

Since, \( |H_{i\ell}| = \left[H_{i\ell} \cdot \overline{H_{i\ell}}\right]^{\frac{1}{2}} \)

Therefore,

\[
\frac{\partial |H_{i\ell}|}{\partial \psi_u} = \frac{i}{|H_{i\ell}|} \cdot \Re \{ \overline{H_{i\ell}} \frac{\partial H_{i\ell}}{\partial \psi_u} \}
\]

(4.81)

The gradients with respect to each parameter in \( \psi_u \), for two classes of filters, are listed in Appendix F.

\( Q \) is now minimized with respect to \( \psi_u \).

In order to start the minimization algorithm an initial guess of \( \psi_u \) should be given. For the special class of fourth-order filters given by equation (4.40), the starting point can be calculated from the design of a one-dimensional filter. In other cases we choose an arbitrary starting point, with the proviso that constraint conditions, such as those given in section 4.3.2 (equation (4.25)) for Huang's special class of second-order filters, are not violated.
4.3.5 MINIMUM PHASE

Of particular interest are minimum phase filters [3]. These are filters where the numerator satisfies the same constraints as the denominator; i.e. if \( N(z_1, z_2) \) is the numerator, then it does not vanish for \( |z_1| < 1 \) and \( |z_2| < 1 \). If we design a filter whose frequency response is always non-zero, then cascading this filter with its inverse will result in an exact cancellation of the original filter's frequency response. Such filters are useful in transmitting and recovering spatial data over noisy channels. Suppose a picture is to be transmitted over a channel where high-frequency noise predominates. In order to minimize the effect of the noise, the received data is passed through a low-pass filter (with non-zero response over the complete frequency range). In order to compensate for the frequency selectivity of the channel, the data is first passed through a high-pass filter before being transmitted. The filters at the transmitting and receiving ends are called pre-emphasis and de-emphasis filters, respectively. For the received data to be identical to the transmitted data, for the case of a non-noisy channel, the pre-emphasis and de-emphasis filters should be the reciprocal of each other. If the de-emphasis filter has minimum phase, then the same design can be used for both filters, and a true reciprocal obtained, that is guaranteed to be stable.
4.3.6 **EXAMPLES:**

**EX. 1: CIRCULARLY SYMMETRIC LOW-PASS FILTER:**

Consider the specifications:

\[ w_{1i} = 0.0, 0.12, (0.01)^+ \quad i = 0, 1, \ldots, 12 \]

\[ w_{2i} = 0.2, 1.0, (0.1) \quad i = 13, 14, \ldots, 19 \]

\[ w_{219+1} = w_{1i} \]

\[ \gamma_{ij}^d = 1.0 \quad \text{for} \quad \sqrt{i^2 + (19-j)^2} \leq 10, \]

\[ \gamma_{ij}^d = 0.5 \quad \text{for} \quad \sqrt{i^2 + (19-j)^2} = 11, \]

\[ \gamma_{ij}^d = 0.0 \quad \text{otherwise}. \]

We shall first design this filter with one cascade \((K=1)\) of Huang's special class of second-order filter.

The initial value of the unconstrained parameter vector, \( \psi_{0} \), was chosen as

\[ \psi_{0} = (1.0, 1.0, 1.0, 1.0, 1.0, 1.0, 1.0, 1.0)^T \]

**LIMIT**, in the DFMP subroutine \([20]\), was set to 50, and \( \text{EPS} = 10^{-5} \).

After 252 function evaluations, execution terminated and the following results were obtained:

+ the notation \( z = x, y, (w) \) means \( z \) takes on the consecutive values \( x, x+w, x+2w, \ldots, y \).
\[
\psi_u = (32.372, -160.54, 102.05, 1.0101, 4.1081, 0.32927)^T
\]

\[
A^* = 0.13135 \times 10^{-4}
\]

\[
H(z_1, z_2) = 10^3 \frac{z_2^{-1}(z_1^{-1} * 0.35139 - 0.16406 - z_1 * 0.14312 + 0.37236)}{z_2 (z_1 * 0.14965 - 0.16789 - z_1 * 0.16748 + 0.19005)}
\]

The perspective plot is shown in Fig. 4.2.

A bad fit is obvious on the axes \( w_1 = 0 \) and \( w_2 = 0 \). This is because the filter has real zeros and poles for \( w_1 = 0 \) or \( w_2 = 0 \), as was indicated previously.

Consider, for the same specifications, the stable fourth-order recursive filter, with minimum phase, defined by the transfer function:

\[
Y(z_1, z_2, \psi_u) = A \frac{(p_1 + a_2 p_2 + b_2^2 + jb_1)(p_1 + a_1 p_2 + b_2^2 - jb_1)}{(p_1 + c_2 p_2 + d_2^2 + j d_1)(p_1 + c_2 p_2 + d_2^2 - j d_1)} \tag{4.82}
\]

where:

\[
\psi_u = (a, b_r, b_i, c, d_r, d_i)^T \tag{4.83}
\]

and,

\[
p_1 \text{ and } p_2 \text{ are as previously defined.}
\]

Let:

\[
\psi_{u_0} = (0.1, 0.1, 0.1, 0.1, 0.1, 0.1)^T
\]

and set

\[
\text{LIMIT} = 30
\]

\[
\text{EPS} = 10^{-5}
\]
Fig. 4.2. Low-Pass Filter Designed from One Cascade of Huang's Special Class of Second Order Filter.
After 83 function evaluations execution terminated and the following results were obtained

\[ Q = 59.608 \]

\[ A^* = 0.014743 \]

\[ \psi_{uf} = (-0.60169, 0.027796, 0.96930, 0.93243, -0.25597, 0.12781)^T \]

\[ H(z_1, z_2) = \frac{N(z_1, z_2)}{D(z_1, z_2)} \]

where:

\[ N(z_1, z_2) = z_1^2 (z_2^2*2.7926 + z_2*0.14012 + 1.3475) + z_1(z_2^2*3.6139 - z_2*0.76607 + 3.6201) + (z_2^2*1.3456 + z_2*0.14236 + 2.7968) \]

\[ D(z_1, z_2) = z_1^2(z_2^2*3.2704 - z_2*0.67479 + 0.054789) + z_1(z_2^2*0.26736 - z_2*6.9411 + 0.79153) + (z_2^2*0.020568 - z_2*0.21906 + 3.7604) \]

The "intensity" plot of this filter is given in Fig. 4.3. The filter looks like a one-dimensional filter, viewed as a two-dimensional filter, and rotated at an angle. This is similar to the rotated low-pass filter given by Shanks [3], and which is shown in Fig. 4.4.
Fig. 4.3. 'Rotated' Low-Pass Filter Designed from the Fourth-Order Filter Given in Equation (4.54).
Fig. 4.4a. One-Dimensional Low-Pass Filter Viewed as a Two-Dimensional Filter.

Fig. 4.4b. Shank's Rotated Low-Pass Filter.
Let us examine the relationship between Shank's 'rotated' two-dimensional filter and the filter given by (4.82).

Consider the following transfer function:

\[ F_1(p_1, p_2) = \frac{p_2 + a}{p_2 + b} \tag{4.84} \]

The magnitude-frequency response of \( F(p_1, p_2) \) can be rotated at an angle \( \beta \) by the transformation [3]

\[ p_1 = p_1' \cos \beta + p_2' \sin \beta \]
\[ p_2 = p_2' \cos \beta - p_1' \sin \beta \tag{4.85} \]

Substituting (4.85) in (4.84) we obtain

\[ F_1(p_1', p_2') = \frac{p_2' \cos \beta - p_1' \sin \beta + a}{p_2' \cos \beta - p_1' \sin \beta + b} \tag{4.86} \]

Assuming \( a \) and \( b \) are complex, we form the following filter

\[ H(p_1, p_2) = F_1(p_1, p_2) \cdot F_2(p_1, p_2) \tag{4.87} \]

where:

\[ F_2(p_1, p_2) = \frac{p_2 \cos \beta - p_1 \sin \beta + a^*}{p_2 \cos \beta - p_1 \sin \beta + b^*} \tag{4.88} \]

"*" represents complex conjugate.
Applying the conditions of stability, from equation (4.38), we obtain:

\[ \frac{-b_r}{\sin \beta} > 0 \]  

(4.89)

\[ \frac{-\cos \beta}{\sin \beta} > 0 \]

These can only be satisfied if \( b_r > 0 \), and \( -\frac{\pi}{2} < \beta < 0 \).

From the previous analysis, we can reasonably assume that the type of filter we can obtain from the transfer function given by equation (4.82) is a one-dimensional 'rotated' filter.

To design the low-pass filter specified in this example, we need at least 2-cascades of the 4th order section. Each is a one-dimensional low-pass filter, with a cut-off close to that of the desired response on either axis, and viewed as a two-dimensional filter. One filter will be considered at a rotated angle of \( 0^0 \) and the other at an angle of \( -90^0 \). We shall use Shank's technique, previously described, to obtain starting points for a design.

First, the one-dimensional low-pass filter is designed using the technique described in sec. 4.2. The specifications for the one-dimensional low-pass filter are given as:
\[ y_i^d = 1.0, \ w_i = 0.0, 0.09, (0.01) \]
\[ y_i^d = 0.5, \ w_i = 0.1 \]
\[ y_i^d = 0.0, \ w_i = 0.11, 0.2, (0.01) \]
\[ y_i^d = 0.0, \ w_i = 0.21, 1.0, (0.1) \]

Setting: \( K = 1 \), \( \text{LIMIT} = 30 \), \( \text{EPS} = 10^{-5} \) and
\[ \psi_{u_0} = (1.0, 0.5, 1.0, 0.5^T) \]

After 91 function evaluations the following results were obtained.

\[ Q = 0.58047 \]
\[ A^* = 0.10953 \]

Poles: \( 0.89796 \pm j 0.18983 \)

Zeros: \( 0.80541 \pm j 0.57863 \)

and:
\[ \psi_{u_f} = (0.095821, 0.32200, 0.29437, -0.11299^T) \]

The designed one-dimensional filter is shown in Fig. 4.5b.

Viewing this as a rotated two-dimensional filter, with \( \beta \approx 0^\circ \), \( (\beta > 0^\circ) \), the unconstrained parameters are calculated as follows:
Fig. 4.5a. Desired Specifications.

Fig. 4.5b. Designed Low-Pass Filter.
\[ a_1 = 10^{-6} \]
\[ b_{r_1} = \frac{0.095821}{\sqrt{2}} = 0.067756 \]
\[ b_{i_1} = \frac{1}{2} \sqrt{4 \times (0.322)^2 - (0.095821)^4} = 0.32197 \]
\[ c_1 = 10^{-6} \]
\[ d_{r_1} = \frac{0.29437}{\sqrt{2}} = 0.20815 \]
\[ d_{i_1} = \frac{1}{2} \sqrt{4 \times (0.11299)^2 - (0.29437)^4} = 0.10435 \]

Note that \( a_1 \) was set to \( 10^{-6} \) and not to 0.0. Since \( a_1 = 0.0 \) will result in grad. \( (a_1) = 0.0 \) (see Appendix F.b), and since \( \Omega \), the error-criterion, is minimized using a gradient method, a local minimum will be reached with \( a_1 = 0.0 \).

Setting \( \beta \approx 90^\circ \), \( (\beta < 90^\circ) \), the previous parameters are calculated as follows:

\[ a_2 = 10^6 \]
\[ b_{r_2} = b_{r_1} \times 10^6 \]
\[ b_{i_2} = b_{i_1} \times 10^{12} \]
\[ c_2 = 10^6 \]
Fig. 4.6. 2-D Eighth-Order Low-Pass Filter, Designed from Two, 1-D 'rotated' Second-Order Filters.
\[ d_{r_2} = d_{r_1} \times 10^6 \]

\[ d_{i_2} = d_{i_1} \times 10^{12} \]

These values effectively set the coefficient of \( b_2 \) in (4.82) to \( 10^{-12} \).

Cascading the two (rotated) two-dimensional filters, we obtain the filter whose response is shown in Fig. 4.6. \( Q \) is calculated to be 23.834.

Using the parameters of this filter as a starting point for an initial design, and setting LIMIT = 25. After 81 function evaluations, execution terminated, and the results obtained were:

\[ Q = 22.307 \]

\[ a_1 = -0.304463075 \times 10^{-3} \]

\[ b_{r_1} = 0.505224510 \times 10^{-2} \]

\[ b_{i_1} = 0.448424734 \]

\[ c_1 = 0.229233872 \times 10^{-2} \]

\[ d_{r_1} = 0.237467984 \]

\[ d_{i_1} = 0.0973223128 \]
Fig. 4.8. Eighth-Order Low-Pass Filter.
a_2 = 10^6

b_{r2} = 0.677556789 \times 10^5

b_{i2} = 0.321967191 \times 10^{12}

c_2 = 10^6

d_{r2} = 0.208151031 \times 10^6

d_{i2} = 0.104352883 \times 10^{12}

A^* = 0.671633949 \times 10^{-2}

The frequency response of the designed filter is shown in Fig. 4.7 and Fig. 4.8.

Consider, for the same specification, the fourth-order filter whose numerator and denominator are of the form given in equation (4.67). Since the specifications have equal responses along either axes, it is reasonable to assume that the coefficient of \( p_1 \) is equal to the coefficient of \( p_2 \).

Thus the filter formed by multiplying the two cascades inherent in the following equation:

\[
\frac{p_1 + p_2 + x_1^2 p_1 p_2 + (x_2^2 + \frac{x_3 x_1^2}{4}) + j x_3}{(A)^{\frac{1}{2}}}
\]

\[p_1 + p_2 + x_4^2 p_1 p_2 + (x_5^2 + \frac{x_6^2 x_4^2}{4}) + j x_6\]  

(4.90)
has the same response on the \( p_1 = 0 \) axis as on the \( p_2 = 0 \) axis, and is guaranteed to be stable and have minimum phase.

Substituting equation (4.24) in (4.90) and multiplying the two equations thus obtained, we get:

\[
\begin{align*}
H(z_1, z_2, \psi_u) &= A \cdot H_1(z_1, z_2, \psi_u) \cdot H_2(z_1, z_2, \psi_u) \\
&= \frac{N_1(z_1, z_2)}{D_1(z_1, z_2)} \\
&= \frac{(1-z_1)(1+z_2) + (1-z_2)(1+z_1) + x_1^2 (1-z_1)(1-z_2)}{(1-z_1)(1+z_2) + (1-z_2)(1+z_1) + x_4^2 (1-z_1)(1-z_2)} \\
&\quad + (x_2 + \frac{x_2^2 x_3^2}{4} \pm jx_3) \ast (1+z_1)(1+z_2) \\
&\quad + (x_5 + \frac{x_5^2 x_6^2}{4} \pm jx_6) \ast (1+z_1)(1+z_2)
\end{align*}
\]  

(4.91)

where:

\[
H_1(z_1, z_2, \psi_u) = \frac{N_1(z_1, z_2)}{D_1(z_1, z_2)}
\]

\[
H_2(z_1, z_2, \psi_u) = \frac{N_2(z_1, z_2)}{D_1(z_1, z_2) - 2 j x_6 (1+z_1)(1+z_2)}
\]

(4.92)

and:

\[
\psi_u = (x_1, x_2, x_3, x_4, x_5, x_6)^T
\]

(4.93)

\[
\psi_{u_0} = (1, 1, 1, 1, 1)^T
\]

(4.94)

The starting value for \( \psi_u \) was chosen to be:
Fig. 4.9. Fourth-Order Low-Pass Filter.
and

\[
\text{LIMIT} = 20, \quad \text{EPS} = 10^{-5}
\]

Execution terminated after 47 function evaluations, and the results obtained were:

\[
Q = 25.976
\]

\[
\psi_u = (1.01435, 2.44860, 2.61682, 2.47065, 0.269818, 0.109803)^T
\]

The designed filter is shown in Fig. 4.9. It is obvious that this form gives a better fit on the axes then Huang's special class, and a flatter pass-band then the 'rotated' filter previously designed.

**EX. 2: A FILTER FOR IMAGE ENHANCEMENT: "SHARPENING"**

The filter we shall consider is the Laplacian \[33\]. It is particularly useful for image enhancement (sharpening).

Given specifications:

\[
w_{1i} = 0.0, 1.0, (0.05) \quad i = 0.1, \ldots, 20
\]

\[
w_2(20+1) = \frac{1}{2} w_i
\]

\[
\gamma_{ij}^d = \sqrt{i^2 + (20-j)^2} / 800 \quad \text{where;}
\]
Fig. 4.10. Specifications for Laplacian Filter Design.
\[
\begin{align*}
\mathbf{a}(i,j) : & \quad 0 \leq i \leq 20, \quad 0 \leq j \leq 20 \\
(\text{See Fig. 4.10}).
\end{align*}
\]

Huang's special class of second-order filter was tried for \( K_f = 1 \), \( \text{LIMIT} = 30 \), and

\[
\mathbf{\psi}_0 = (1.0, 1.0, 1.0, 0, 1.0, 1.0, 1.0, 1.0)^T
\]

Convergence was obtained after 42 function evaluations.

\[
Q = 1.8496
\]

\[
A^* = 0.41622
\]

\[
\mathbf{\psi}_f = (0.99928, 0.98453, 0.72691, 0.0199739, 0.65210, 1.7137)^T
\]

\[
H(z_1, z_2) = \frac{z_2(-0.5008z_1 - 0.43945) - z_1^*0.44234 + 3.4962}{z_2(1.3671z_1 + 2.5167) + z_1^*2.5062 + 5.3567}
\]

The magnitude-frequency response of this filter is given in figures 4.11 and 4.12.

The filter given by eq. (4.82) was tried with

\[
\mathbf{\psi}_0 = (1.0, 1.0, 1.0, 0, 1.0, 1.0, 1.0)^T
\]

and

\[
\text{LIMIT} = 25.
\]

After 74 function evaluations convergence was obtained, \((\text{EPS} = 10^{-5})\). The results are
Fig. 4.11. Second-Order Design of a Laplacian Filter.
Fig. 4.12. Second-Order Design of a Laplacian Filter.
\[ Q = 12.518 \]

\[ \Psi_f = (1.1279, 1.2600, -0.9455 \times 10^{-8}, 1.1163, 1.8627, 0.28755 \times 10^{-7})^T \]

The spatial magnitude-frequency response of the designed filter is given in fig. 4.13. The response looks like that of a 'rotated' high-pass filter. This result agrees with the statement given in the previous example, that filters designed with this form appear as 'rotated' filters.

4.3.7 **COMPUTATIONAL DETAILS**

The execution time required for the design of two-dimensional filters, using non-linear minimization algorithms, is usually high. A step towards decreasing execution time is to eliminate points from the specifications that might not have much effect on the final design. Since more emphasis should usually be set on the pass-band and cut-off rate, the specification points can be set closer in the pass band and transition region than in the stop-band. However, in some regions the stop-band will contain more points than in others. For example, consider the low-pass filter specifications given in the previous section. These are described over the mesh in the \( \Omega_1 - \Omega_2 \) plane, shown in fig. 4.14. Region I contains the pass-band. It is obvious that regions II, V and VI,
Fig. 4.14. Specification Mesh in the $n_1-n_2$ Plane.

Fig. 4.15. Elimination Mesh in the $n_1-n_2$ Plane. Shaded Regions Contain 'True', Unshaded Regions Contain 'False'. $T = 'True'$. 
which are in the stop-band, are more closely spaced than
regions III and IV. To force the spacing in the stop-
band to be nearly uniform, we set another mesh, of the
same size as the mesh on which the specifications are
chosen, which takes on values of "true" or "false". If
a point on this mesh is false, then the specification
at the equivalent location in the $\Omega_1 - \Omega_2$ plane is
effectively removed and vice-versa for true. The élimina-
tion mesh for this example would then be as shown in Fig.
4.15.

4.3.8 COMPUTER PROGRAMME:

Listing of the computer programme for the design of
Huang's special class of filter is given in Appendix G.

For other classes of filters, the programmes are
very similar in steps to the one given in Appendix G, and
therefore are not listed.

4.4 RECURSIVE FILTERING OF SPATIAL DATA:

Given the spatial data, $f(m,n)$, as in input to the
recursive filter

$$Y(z_1, z_2) = \frac{\sum_{i=0}^{N_a} \sum_{j=0}^{N_a} a_{ij} z_1^i z_2^j}{1 + \sum_{k=0}^{N_b} \sum_{\ell=0}^{N_b} b_{k\ell} z_1^k z_2^\ell}$$

(4.95)

where; $z_1 = e^{-S_1 A}$, $z_2 = e^{-S_2 B}$
The output data, \(g(m,n)\), can be computed from the recursive algorithm

\[
g(m,n) = \sum_{i=0}^{M_a} \sum_{j=0}^{N_a} a_{ij} f(m-i,n-j) - \sum_{k=0}^{M_a} \sum_{\ell=0}^{N_a} b_{k\ell} g(m-k,n-\ell)
\]

Here, we assume either all the output values \(g(m-k,n-\ell)\) have been computed previously (for \(m-k, n-\ell \geq 0\)), or are equal to zero (for \(m-k, n-\ell < 0\)).

A convenient implementation of this algorithm would be to compute the output array one row (or column) at a time. If we compute the output data row-wise, then the input data is scanned row-wise, and only the previous \(M_a+1\) rows of the input data, and \(M_b+1\) rows of the calculated output need to be stored at a time.

For cascaded sections, the output data of one section can be considered as input data to the next section, and so on.

A filtering example is shown in Figures 4.16 and 4.17. Figure 4.16 is the given input to the second-order Laplacian designed in example 2. Figure 4.17 is the filtered output.
Fig. 4.16. An Annulus Normalized for Maximum Contrast.
Fig. 4.17. Output data from Laplacian, Normalized
for Maximum Contrast, Given the
Annulus in Fig. 4.16 as the Input..
4.5 SUMMARY:

A technique has been presented for designing stable forms of two-dimensional recursive filters. The technique is based on minimizing a mean squared error criterion using the unconstrained minimization algorithm of Fletcher and Powell.

Constraints required on the parameters of second and fourth-order filters, for stability and minimum phase, have been derived using Ansell's stability theorem. A variable substitution method has been applied, for selected classes of filters, to impose these constraints. This results in filter forms with parameter, constrained for stability and minimum phase.

The designed forms are in cascades of second or fourth order filters.
CHAPTER 5

CONCLUSION

The major results, conclusions and contributions of the research described in this dissertation may be summarized as follows:

(i) A computer oriented technique for obtaining the analytic expression of the impulse response of one-dimensional recursive filters has been derived. The technique requires the calculation of the poles of the filter, and a single matrix inversion, both of which can be accomplished by a digital computer for fairly large orders of filter.

(ii) A procedure for obtaining closed-form solutions of the horizontal lines (or vertical lines) of the spatial impulse response of the two-dimensional recursive filter, has been presented. The total closed form solution for the spatial impulse response can be obtained for simple cases, via this method. This procedure avoids the calculation of the continuum of poles of the two dimensional recursive filter. Effectively it transforms the double inverse integral, for the analytical inversion, to a single inverse integral. This can be solved analytically by the one-dimensional algorithm.
(iii) Two necessary conditions for the stability of two-dimensional recursive filters have been derived. These conditions guarantee the stability of the horizontal and vertical lines of the recursive filter, but do not guarantee total stability.

(iv) Algorithms have been developed for designing one and two-dimensional recursive filters, to approximate a given impulse-response. A 'weighted' square-error in the frequency domain is minimized, and leads to the solution of a set of simultaneous equations for the calculation of the filter coefficients. An iterative refinement scheme is used for approaching the 'unweighted' square-error. In this technique, FFT pruning algorithms are used for the calculation of the coefficients of the simultaneous equations, and of the weights in the refinement scheme. Efficient computational means have been used throughout the algorithm.

(v) An algorithm for the design of stable two-dimensional recursive filters, to approximate magnitude-frequency specifications, has been presented. In the algorithm a square-error criterion between desired specifications and magnitude of cascades of second or fourth order two-dimensional recursive filter is minimized. The method employs the unconstrained conjugate-gradient minimization algorithm of Fletcher and Powell. The parameters are constrained for
stability via an extension of a one-dimensional variable substitution method.

The analysis is based on several classes of stable two-dimensional recursive filters. One of these classes corresponds to the 'rotated' filters described by previous workers.
REFERENCES


OTHER REFERENCES


APPENDIX A

FORTRAN IV CODING OF THE ALGORITHM,
FOR THE INVERSE Z-TRANSFORM,
GIVEN IN CHAPTER II
SUBROUTINE INVZ(A,B,N,M,Y,NPTS)
THIS SUBROUTINE IS FOR FINDING THE INVERSE Z-TRANSFORM IN CLOSED
FORM SOLUTION, OF THE RATIONAL FUNCTION:
G(z) = A(1) + A(2)/z + ⋯ + A(N+1)/z^N

ARGUMENTS OF SUBROUTINE:
Y IS A VECTOR CONTAINING CALCULATED POINTS (=NPTS) OF THE INVERSE
Z-TRANSFORM.
DIMENSIONS:
A(N+1),B(M+1),XCOF(M+1),COF(M+1),ROOTI(M),POOTI(M),
M(M),D(M,M+1),POW(M),Y(NPTS)
SUBROUTINE REQUIRED: POLRT, AN IBM SUBROUTINE.
REAL*8 A,B,XCOF,COF,DABS,DATAN,X1,X2,X3,COABS,ROOTR,ROOTI,THETA
COMPLEX*16 COMPLX,DMPLX,DCMPLX,DCONJG,XC
DIMENSION A(17),B(17),XCOF(17),COF(17),ROOTR(16),ROOTI(16),MU(16)
*,D(16,17),POW(16),Y(130)
KOUNT=0
N1=N+1
M1=M+1.
GO TO 1
1 XCOF(1)=B(M1-I+1)
CALL POLRT(XCOF,COF,M,POOTR,ROOTI,IER)
IF (IER-1) 4,5,6
5 WRITE(6,7)
7 FORMAT(5X,DEROR- THE ORDER OF THE DENOMINATOR IS LESS THAN ONE)
GO TO 100
6 IF (IER-3) 8,9,10
8 WRITE(6,11)
11 FORMAT(5X,DEROR- THE ORDER OF THE DENOMINATOR IS GREATER THAN 36)
GO TO 100
9 WRITE(6,12)
12 FORMAT(5X,DEROR- THE HIGH ORDER COEFFICIENT OF THE DENOMINATOR IS
* ZERO)
GO TO 100
10 WRITE(6,13)
13 FORMAT(5X,DEROR- UNABLE TO DETERMINE THE POLES OF GIVEN TRANSFER
* FUNCTION)
GO TO 100

DETERMINING THE MULTIPLICITIES OF THE POLES
4 MC=M
D3 81 I=1,M
81 M(I)=I.
14 K1=I+1
K=K1
17 IF (DABS(ROOTR(I)-ROOTR(K)).GT.1.D-2) GO TO 15
IF (DABS(POOTI(I)-ROOTI(K)).GT.1.D-2) GO TO 15
L=K
MC=MC-1
M(I)=MU(I)+1
IF (L.GT.MC) GO TO 15
D3 16 J=L,MC
ROOTR(J)=ROOTR(J+1)
16 ROOTI(J)=ROOTI(J+1)
GO TO 17
15 K=K+1
A

IF(K .LE. MC) GO TO 17

I = I + 1

IF(I .LT. MC) GO TO 14

WRITE(6, 300)

300 FORMAT(1H1, 5X, 'THE GIVEN Z-TRANSFORM IS: \(Y(Z) = (A(1) + A(2)/Z + A(3)/Z^2 + \ldots + A(N+1)/Z^{N+1})/(1 + B(1)/Z + B(2)/Z^2 + \ldots + B(M+1)/Z^{M+1})\)')

WRITE(6, 301) N

301 FORMAT(5X, 'N=', I4, 2X, 'M=', I4)

WRITE(6, 302) (A(I), I = 1, NI)

302 FORMAT(5X, 'A(', I3, ')=', F10.3, 5X, 'B(', I3, ')=', F10.3)

IF(M .NE. N) GO TO 303

303 CONTINUE

WRITE(6, 202)

202 FORMAT(//, 1X, 'POLES', 13X, 'MULTIPLICITIES')

WRITE(6, 201) (POLE(I), ROOT(I), M(I), I = 1, MC)

201 FORMAT(4X, D10.3, 1X, '+J*', D10.3, 10X, I3)

WRITE(6, 203)

203 FORMAT(//)

C FROMING THE MATRIX REQUIRED IN THE EVALUATION OF ALPHA(0), ALPHA(1),

C ..., ALPHA(N-1) AS GIVEN BY THE CAYLEY-HAMILTON'S THEOREM.

J = 1

18 IF(J .LE. M) GO TO 20

DO 19 I = 2, M

19 XC = CMPLX(ROOT(I), 0)

14 DO 21 J = 1, M

21 IF(J .NE. I) GO TO 23

K = 0

22 J = J + 1

23 K = K + 1

DO 24 L = 1, K

24 PW(L+1) = PW(L) - 1

DO 25 L = 1, K

25 PP = PP * PW(L)

26 IF(K1) 26, 27, 28

27 D(J, I) = CMPLX(0, 0)

28 CONTINUE

29 IF((K+1) .LT. N) GO TO 22

J = J + 1

32 JL = JL + 1

GO TO 18
20. CONTINUE

C INVERSION OF THE FORMED MATRIX

       KK = 0
       JJ = 0

       DJ 30 K = 1, M
       DJ 31 J = 1, M

31 D(J,JM) = DCMLX(0.00, 0.00)
       D(K, KM) = DCMLX(1.00, 0.00)

       JJ = KK + 1

       LL = JJ

       KK = KK + 1

32 IF(CDAR5(D(JJ,KK)) - 1.0D-4) 33, 34

33 JJ = JJ + 1

       GO TO 32

34 IF(LL - JJ) 35, 36, 35

35 DO 37 M = 1, M1
       DMP = D(LL, MM)
       D(JJ, MM) = D(LL, MM)

37 D(MPI) = D(MPI) / DMP

36 DIV = D(K, K1)
       DJ 38 LJ = 1, M1

J = M1 + 1 - LJ

38 D(K, J) = D(K, J) / DIV

DO 40 I = 1, M

       FAC = D(I, K)

60 DO 40 LJ = 1, M1

       J = M1 + 1 - LJ

61 IF(I - K) 41, 40, 41

41 D(I, J) = D(I, J) - FAC * D(K, J)

40 CONTINUE

62 DJ 42 J = 1, M

42 D(I, K) = D(J, M1)

30 CONTINUE

C CALCULATING THE INITIAL CONDITIONS

XCOF(I) = A(I) / B(1)

IF(M > 0, 1) GO TO 80

IF(N) 44, 43, 44

44 DJ 48 I = 2, N1

XCOF(I) = 0.00

I = I - 1

DO 207 J = 1, N1

207 XCOF(I) = XCOF(I) + B(I - J + 1) * XCOF(J)

48 XCOF(I) = (A(I) - XCOF(I)) / B(1)

IF(N - M) 43, 80, 43

43 N2 = M + 1

N3 = N + 2

DO 208 I = N3, N2

XCOF(I) = 0.00

I = I - 1

DO 209 J = 1, N1

209 XCOF(I) = XCOF(I) + B(I - J + 1) * XCOF(J)

208 XCOF(I) = -XCOF(I) / B(1)

C DETERMINING THE RESIDUES

80 IND = 0

IF(N EQ M) IND = 1
SOLUTION IN THE N*T-DOMAIN

WRITE(6,57)
57 FORMAT(5X,'THE INVERSE Z-TRANSFORM IS GIVEN BY ',/,'5X','Y(K*T)=')
   I=0
   X=0
52 I=I+1
   X=I*X+1
   IF(I.GT.MC) GO TO 56
   WRITE(6,51) '(1,IX),ROOTRI(I),ROOTII(I)
51 FORMAT(I1X,+',D10.3,'+',J',D10.3,'+',K',D10.3,'+',L',D10.3,'+',M')
   IF(MU(I).EQ.1) GO TO 52
   L=MU(I)
   DO 53 J=2,L
   IX=I*X+1
   IDX(I,J)=IX+1
   IF(I.GT.MC) GO TO 701
   AVOIDING UNDERFLOWS
   DTEMP=DCMPLX(ROOTRI(I),ROOTII(I)
   XL=CDABS(DTEMP)
   IF(XL.D0.XL) 550,552,551
      551 AA=SNGL(IX)
      IF(AA.LT.0.01) GO TO 550
      JJ=FIX(-60.0/ALOG10(AA))
      IF(JJ.LE.IJJ) GO TO 552
      FAC=FAC*D(1,IX)*DCMPLX(ROOTRI(I),ROOTII(I))**(J-1)
      DO 550 I=1,L
552 IF(MU(I).EQ.1) GO TO 402
   L=MU(I)
   DO 204 JJ=2,L
   IX=I*X+1
   IF(JJ.LE.IJJ) GO TO 204
   K=JJ-1
   IP=1
   LL=J-1
   DO 205 II=1,K
   IP=IP*LL
   205 LL=LL-1
   FAC=FAC+D(1,IX)*IP*DCMPLX(ROOTRI(I),ROOTII(I))**(J-JJ))
204 CONTINUE
   IF(I.LT.NC) GO TO 402
701 XL=IDAC+DCONJS(DAC))2.0
200 YJ=IND)=NSL(XL)

C TRANSFORMING THE SOLUTION TO A MORE CONDENSED FORM.
901 WRITE(6,120)
120 FORMAT('///;5X,\'WHICH COULD BE CONDENSED TO THE FORM ',/11X,\'1\')
   LL=0
   J=0
60 J=J+1
   LL=LL+1
804 IF(LL=M) 163,163,700
163 IF(DABS(RNTJ(J))=(L.D-5)*DABS(RNTP(J))) 112,112,96
112 XI=(D1,LL)+DCONJG(D1,LL))/2.0
   IF(XI) 70,58,70
.70 WRITE(6,111) XI, RNTT(J)
58 IF(MU(J)>0.1) GO TO 60
L=MU(J)
   THEN 900 JJ=2,L
13 K=JL-1
   LL=LL+1
LXL=(D1,LL)+DCONJG(D1,LL))/2.0
   IF(XL) 60,70,60
20 WRITE(6,902) K, XL, RNTT(J), K
900 CONTINUE
   GO TO 60
96 IF(DABS(RNTT(J))-(1.D-05)*DABS(RNTT(J))) 101,101,69
101 THEJA =1.57079632679489
   GO TO 97
69 THEJA=DATAM(RNTT(J)/RNTT(J)
97 XC=DMLX(RNTT(J), RNTT(J)
   X3=DABS(XC)
 X1=(D1,LL)+DCONJG(D1,LL))
   X2=(2.D0*(1,LL)-X1)/DMLX(0.D0,1.D0)
33 IF(DABS(X1)-(1.D-05)*DABS(X2)) 98,98,164
98 IF(X2) 74,98,74
74 WRITE(6,104) X2, X3, THEJA
   IF(MU(J)-1)) 1000,72,1000
72 LL=LL+2
   J=J+2
39 GO TO 804
164 IF(DABS(X2)-(1.D-051)*DABS(X1)) 95,95,67
95 IF(X1) 75,1003,75
75 WRITE(6,162) X1, X3, THEJA
1003 IF(MU(J)-1)) 1000,72,1000
67 IF(X2) 801,802,802
801 WRITE(6,803) X3, X1, X3, X2, THEJA
   GO TO 107
802 WRITE(6,102) X3, X1, THEJA, X2, THEJA
107 IF(MU(J)-1)) 1000,72,1000
99 L=MU(J)
1000 JJ=2,L
   THEN 1001, JJ=2,L
 K=JJ-1
   LL=LL+2
   X1=D1,LL)+DCONJG(D1,LL))
   X2=(2.D0*(1,LL)-X1)/DMLX(0.D0,1.D0)
   IF(DABS(X1)-(1.D-05)*DABS(X2)) 903,903,1007
903 IF(X2) 1005,1001,1005.
1005 WRITE(6,77), K, X2, X3, K, THETA, K
  GO TO 1001
1007 IF(DABS(X2)-(1.D-05)*DABS(X1)) 1111, 1111, 1110
1111 IF(X1) 1009, 1001, 1009
1009 WRITE(6,99), K, X1, X3, K, THETA, K
  GO TO 1001
1110 IF(CMABS(D(1,L))) 1008, 1001, 1008
1008 WRITE(6,82), K, X3, K, X1, THETA, K, X2, THETA, K
1001 CONTINUE
  GO TO 60
111 FORMAT(5X,**,D10.3,**K)
902 FORMAT(5X,**,(FACT(0)/FACT(K-1)),13,1)**, D10.3,**((1,D10.3,**K)**(11,13,**K))**K)
104 FORMAT(5X,**,D10.3,**((1,D10.3,**K)*SIN((1,D10.3,**K)**(11,13,**K))**K)
162 FORMAT(5X,**,D10.3,**((1,D10.3,**K)*COS((1,D10.3,**K)**(11,13,**K))**K)
1803 FORMAT(5X,**,(1,D10.3,**K)**(*,(1,D10.3,**K)**(11,13,**K))**K)
77 FORMAT(5X,**(FACT(0)/FACT(K-1)),13,1)**, D10.3,**((1,D10.3,**K)**(11,13,**K))**K)
99 FORMAT(5X,**(FACT(0)/FACT(K-1),13,1)**, D10.3,**((1,D10.3,**K)**(11,13,**K))**K)
82 FORMAT(5X,**(FACT(0)/FACT(K-1),13,1)**(1,D10.3,**K)**(11,13,**K))**K)
700 IF(N.EQ.0) WRITE(6,500) XCNF(1)
500 FORMAT(5X,**WHERE** K=1,2,**ETC,**/5X,**AND,** Y(0)=-1,D12.5).
100 RETURN
END
APPENDIX B

TIME DOMAIN FFT PRUNING
APPENDIX B

TIME DOMAIN FFT PRUNING

The problem can be stated as follows. Given $2^M$ data points in the time domain, how do we efficiently compute $2^L$ output points in the frequency domain, with $M < L$? Here we use a decimation in frequency algorithm with pruning discussed by Markel [30]. In his computer programme, Markel requires that some of the $2^{M-L}$ zero samples be used in the computation of the required "butterflies", (see Figures B1 & B2), i.e., whenever a "butterfly" is computed it is done in its entirety. We may write a more efficient programme by only computing partial "butterflies" when an entire computation is not required. The flow graph and the programme are shown in Figures B3 and B4 respectively.

The modified routine discussed was compared with the programme by Markel for $L=3$, $M=7$, The execution time for the modified programme was 27% faster.

The extension of this algorithm for two-dimensional FFT pruning, is done in two stages. Since:

$$
H(k_1,k_2) = \sum_{n_1=0}^{N_1-1} \sum_{n_2=0}^{N_2-1} h(n_1,n_2) e^{-j2\pi \left( \frac{k_1 n_1}{N_1} + \frac{k_2 n_2}{N_2} \right)} \tag{B.1}
$$
by defining;

\[ g(n_1, k_2) = \sum_{n_2=0}^{N_2-1} h(n_1, n_2) e^{-j2\pi \frac{k_2 n_2}{N_2}} \]  
(B.2)

Let \( W_1 = e^{-j \frac{2\pi}{N_1}} \) \( W_2 = e^{-j \frac{2\pi}{N_2}} \),

we can write

\[ H(k_1, k_2) = \sum_{n_1=0}^{N_1-1} g(n_1, k_2) W_1^{n_1} W_2^{k_2} \]  
(B.3)

Equations (B.2) and (B.3) each represent a one-dimensional FFT algorithm. Hence, the previous algorithm could first be applied to eq. (B.2). The result is then used in equation (B.3) to calculate the two-dimensional FFT.

The flow chart and computer programme for the two-dimensional FFT space domain pruned algorithm are given in Figures B5 and B6.
Fig. B3. Time Domain Pruned FFT.
SUBROUTINE FFTP(X,H,M,L)

A TIME PRUNED FFT, USING THE DECIMATION IN FREQUENCY ALGORITHM.
NUMBER OF INPUT SAMPLES = 2**L.
NUMBER OF OUTPUT SAMPLES = 2**M.
WHERE M IS GREATER THAN OR EQUAL TO L.
REAL*8 SCL,ARG,DCOS,DSIN
COMPLEX*16 DCPLX,H,X(I2R),T
K=M-L
N=2**M
L2=2**L
DO 1 LO=1,M
L4X=2**((M-LO))
LIX=2*L4X
SCL=6.28318530717958/LIX
IF(LO<K12,2,3).
2 DO 4 LM=1,L2
ARG=(LM-1)*SCL
W=DCPLX(DCOS(ARG),-DSIN(ARG))
4 DO 4 LI=LI+L4X
J1=LI-LIX+LM
J2=J1+LMX
4 X(J2)=W*X(J1).
GO TO 1
3 DO 5 LM=1,LMX
ARG=(LM-1)*SCL
W=DCPLX(DCOS(ARG),-DSIN(ARG))
5 DO 5 LI=LI+L4X
J1=LI-LIX+LM
J2=J1+LMX
T=X(J1)-X(J2)
X(J1)=X(J1)+X(J2)
5 X(J2)=W*T
CONTINUE
1 CONTINUE
NV2=N/2
NM1=N-1
J=1
DO 7 I=1,NM1
IF(I.GE.J) GO TO 6
7 T=X(J)
X(J)=X(I)
X(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
RETURN
END

Fig. B.4
Fig. B5. Flow Chart for the Two-Dimensional Space Domain Pruned FFT.
SUBROUTINE FFT2(X,M1,M2,L1,L2)
      A TWO-DIMENSIONAL TIME PRUNED FFT, USING THE DECIMATION IN FRE.
      ALGORITHM.
      INPUT DATA OF DIMENSION 2**L1 BY 2**L2, ARE STORED IN THE MATRIX X.
      OUTPUT DATA ARE OF DIMENSION 2**M1 BY 2**M2, ARE STORED IN X, AND
      THE INPUT IS DESTROYED. (M1.GE.L1), AND (M2.GE.L2).
      REAL*8 SCL,ARG,DCOS,DSIN,
      COMPLEX*16 PCRMLX,W,X( 8, 81, T
      M1=2**M1
      M2=2**M2
      LP1=2**L1
      LP2=2**L2
      DJ 9 LL=1,LP1
      DJ 1 LO=1,M2
      LIX=2**(M2-LO)
      LIX=2*LPX
      SCL=6.28318530717958/LIX
      IF(LO-M2+L2) 2,2,3
      2 DJ 4 LP=1,LPZ
      ARG=(L-1)*SCL
      W=DC*PCRMLX(ARG),DSIN(ARG)
      DJ 4 LL=LI*LX,N2+LIX
      J1=LI-LIX+LM
      J2=J1+LNX
      4 X(LL,J2)=W*X(LL,J1)
      GO TO 1
      3 DJ 5 LM=1,LMX
      ARG=(LM-1)*SCL
      W=DC*PCRMLX(ARG),DSIN(ARG)
      DJ 5 LL=LI*LX*N2+LIX
      J1=LI-LIX+LM
      J2=J1+LNX
      T=X(LL,J1)+X(LL,J1)-X(LL,J2)
      X(LL,J1)=X(LL,J1)+X(LL,J2)
      5 X(LL,J2)=W*T
      1 CONTINUE
      NW2=M2/2
      NM1=M2-1
      J=1
      DO 7 I=1,NM1
      IF(I.GE.J) GO TO 6
      T=X(LL,J)
      X(LL,J)=X(LL,J)
      X(LL,J)=T
      6 K=NW2
      8 IF(K.GE.J) GO TO 7
      J=J-K
      K=K/2
      GO TO 8
      7 J=J+K
      9 CONTINUE
      DJ 10 LL=1,N2
      DJ 11 LO=1,M1
      LIX=2**(M1-LO)
      LIX=2*LMX
      SCL=6.28318530717958/LIX
      IF(LO-M1+L1) 12,12,13
      12 DJ 14 LM=1,LP1
Fig. B6
APPENDIX C

FREQUENCY DOMAIN FFT PRUNING
APPENDIX C
FREQUENCY DOMAIN FFT PRUNING

The problem is the reverse of that shown in the previous Appendix. i.e., \( M > L \). We may solve the problem by an efficient modification of the technique discussed by Cooley et al [30]. Their technique uses a decimation in time algorithm with an efficient inner-loop nesting procedure discussed by Sande [32]. All that it is necessary to do is to efficiently prune the algorithm so that only \( 2^L \) output points are calculated. The flow graph is shown in Figure C2 and the computer programme in Figure C3.

The extension of the solution of this problem to the two-dimensional FFT is given by the flow-chart in Figure C4 and the computer programme in Figure C5.
Fig. Cl. Decimation in Time FFT.
Fig. C2. Frequency Domain Pruned FFT.
SUBROUTINE FFTP(A,M,L)
  A FREQUENCY PRUNED FFT, USING THE DECIMATION IN TIME ALGORITHM.
  NUMBER OF INPUT SAMPLES=2**M, NUMBER OF OUTPUT SAMPLES=2**L,
  WHERE, M IS GREATER THAN OR EQUAL TO L.
  COMPLEX*16 A(128),U,W,T,DCMPLX
  REAL*8 PI,DCOS,DSIN
  DATA PI/3.14159265358979/
  N=2**M
  NI=2**L
  NV2=N/2
  NV1=N-1
  J=1
  DO 7 I=1,NV1
  7 IF(I.GE.J) GO TO 5
     T=A(J)
     A(J)=A(I)
     A(I)=T,
  5 K=NV2
  6 IF(K.GE.J) GO TO 7
     J=J-K
     K=K/2
     GO TO 6
  7 J=J+K
  8 DO 40 LO=1,M
     LE=2**LO
     LE1=LE/2
     U=DCMPLX(1.00,0.00)
     W=DCMPLX(DCOS(PI/LE1),-DSIN(PI/LE1))
     IF(LO-L) 20,20,30.
  20 DO 11 J=1,LE1
  11 DO 9 I=J,NV1,LE
     IP=I+LE1
     T=A(IP)*U
     A(IP)=A(I)-T
     A(I)=A(I)+T
  11 U=U*W
     GO TO 40
  30 DO 12 J=1,NI
  12 DO 10 I=J,NV1,LE
     IP=I+LE1
     A(I)=A(I)+A(IP)*U
  10 U=U*W
     CONTINUE
     RETURN
     END

Fig. C3
Fig. C4. Flow Chart for the Two-Dimensional Frequency Pruned FFT.
SUBROUTINE FFTP2(A,M1,M2,L1,L2)
A TWO-DIMENSIONAL FREQUENCY PRUNED FFT, USING THE DECIMATION IN
TIME ALGORITHM.

A INPUT DATA OF DIMENSION 2**(M1) BY 2**(M2), ARE STORED IN A.
OUTPUT DATA OF DIMENSION 2**(L1) BY 2**(L2) FOR STORED IN A, AND
THE INPUT DESTROYED. (L1,L2,M1), AND (L2,L1,M2).
NOTE OUTPUT A(W1,M2) ARE STORED IN A(I,J), WHERE, I=1,2,...2**L1,
AND J=1,2,...,2**L2.
A(-W1,M2), ARE STORED IN A(I,J), I=1,2,...2**L1, AND J=(2**L2)+1,
(2**L2)+2,...,2**(L2+1).

COMPLEX*16 A, W, U, H, T, DCMPLX, UL, W1, DCONJG
REAL*8 PI, PI/3, 14159265358979/

N1=2**M1
N2=2**M2
NP1=2**L1
NP2=2**L2
NV2=N2/2
NM1=N2-1
DJ 4 LL=1,N1
J=1

DJ 7 I=1,NM1
T=DCMPLX(1.0D,0.0D)
T=DCMPLX(DCOS(PI/LE1),DSIN(PI/LE1))
W=DCMPLX(W1,0.0D)
IF(LO=L2,2,2,3
2 DJ 8 J=1,LE1
2 DJ 9 I=J,NM7,LE
IP=I+LE1
T=DCMPLX(I,IP)*T
4 A(LL,IP)=A(LL,I)+T
9 A(LL,I)=A(LL,I)+T
8 U=U*W
3 DJ 4
3 DJ 11 J=1,NP2
2 DJ 10 I=J,NP2,LE
4 A(LL,IP)=A(LL,I)+A(LL,IP)*U
11 U=U*W
4 CONTINUE
4 CONTINUE
NV2=N1/2
NM1=N1-1
DJ 12 LL=1,NP2
J=1
DJ 17 I=1,NM1
Fig. C5
IF(I.SEQ.J) GO TO 15
   T=A(J,LL)
   A(J,LL)=A(I,LL)
   A(I,LL)=T
15  K=NV2
16  IF(K.SEQ.J) GO TO 17
    J=J-K
    K=K/2
17  GO TO 16
18  J=J+K
    KK=LL+NP2
19  DO 40 I=1,N1
20   A(I,KK)=A(I,LL)
21   DJ 14,20,N1
22   LE=2*IL/B
    LEI=LE/2
23   J=DCMPLX(1.00,0.00)
24   UL=DCMPLX(1.00,0.00)
25   W=DCMPLX(DCCS(PI/LEI),-CSIN(PI/LLI))
26   W1=CONJ(W)
27   IF(II/LL) 20,20,30
28  DO 18 J=1,LL/2
29   DJ 19,1,NN,LL
30   IP=I+LE1
31   T=A(IP,LL)*U
32   A(IP,LL)=A(I,LL)-T
33   A(I,LL)=A(I,LL)+T
34   T=I(IP,KK)*U1
35   A(IP,KK)=A(I,KK)-T
36   A(I,KK)=A(I,KK)+T
37   UL=UL*W1
38  18  UL=UL*W1
39  20  GO TO 14
40  21 J=1,NP1
41  22 DO 21 J=1,N1,LL
42   IP=I+LE1
43   A(I,LL)=A(I,LL)+A(IP,LL)*U
44  22  A(I,KK)=A(I,KK)+A(IP,KK)*U1
45   UL=UL*W1
46  21  CONTINUE
47  14 CONTINUE
48  12 CONTINUE
49  10 RETURN
50  END

Fig. 5.
APPENDIX D

NUMERICAL GENERATION OF IMPULSE RESPONSE USING TIME (OR SPATIAL) DOMAIN PRUNED FET, FOR THE ONE-DIMENSIONAL (OR TWO-DIMENSIONAL) DIGITAL FILTERS
APPENDIX D

NUMERICAL GENERATION OF IMPULSE RESPONSE USING TIME (OR SPATIAL) DOMAIN PRUNED FFT, FOR THE ONE-DIMENSIONAL (OR TWO-DIMENSIONAL) DIGITAL FILTERS

Figures D1 and D3 are listings of the Fortran programmes for both the one-dimensional and spatial digital filters respectively.

Figures D2 and D4 are listings of the inverse FFT FORTRAN programmes, which are to be used with the listings in Figures D1 and D3 respectively.
SUBROUTINE TIME(X,Y,A,B,NA,NB,LA,LB,M)

    THIS SUBROUTINE CALCULATES THE IMPULSE RESPONSE OF A RECURSIVE
    DIGITAL FILTER USING A TIME-PRUNED FFT ALGORITHM.

    Y(Z) = (A(1) + A(2) * Z + A(3) * Z**2 + ... + A(NA+1) * Z**NA) /
    [B(1) + B(2) * Z + ... + B(NB+1) * Z**NB]

    2**LA AND 2**LB ARE THE NEAREST HIGHER NUMBERS TO NA+1 AND NB+1

    Y = CALCULATED IMPULSE RESPONSE, DIMENSIONED Y(2**M)

REAL*8 D,Y(128),A(17),P(17)

NA = NA+1
NB = NB+1
NN = 2**LA
N = 2**L

DO 1 I = 1, NA
  1 XL(I) = CMPLX(A(I), 0.0D0)

IF (NN-NA1) 3, 3, 2

2 4K = NA1+1

DO 4 I = NK, NN

4 XL(I) = CMPLX(0.0D0, 0.0D0)

3 CALL FFTP(XL,M,LA)

DO 5 I = 1, NR1

5 XL(I) = CMPLX(B(I), 0.0D0)

NN = 2**LB

7 IF (NN-NR1) 7, 7, 6

8 4K = NB1+1

DO 8 I = NK, NN

6 XL(I) = CMPLX(0.0D0, 0.0D0)

CALL FFTP(XL,M, LB)

DO 9 I = 1, N

9 XI(I) = XL(I) / X(L)

10 9 XI(I) = XI(I)

CALL IFFT(XL,M)

DO 10 I = 1, N

10 Y(I) = DREAL(X(I)) / N

RETURN

END

Fig. D1.
SUBROUTINE IFFT(A,N)
SUBROUTINE FOR FINDING THE INVERSE FFT.
DECIMATION IN TIME ALGORITHM.
COMPLEX*16 A(128),U,W,T
REAL*8 PI,DCOS,DSIN
DATA PI/3.14159265358979/
N=2**4
NV2=N/2
NM1=N-1
J=1
DO 7 I=1,NM1
7 IF(I.GT.J)GO TO 5
T=A(J)
A(J)=A(I)
A(I)=T
K=NV2
IF(K.GT.J)GO TO 7
J=J-K
K=K/2
GO TO 6
DO 20 L=1,N
20 LE=2**L
LE=LE/2
U=DCMPLX(1.0,0.0)
W=DCMPLX(DCOS(PI/LE1),DSIN(PI/LE1))
DO 26 J=1,LE1
26 DO 10 I=J,N,LE
9 IP=I+LE1
10 T=A(IP)*U
A(IP)=A(IP)+T
U=U*W
RETURN
END

Fig. D2.
SUBROUTINE TIME2(X,G,M1,M2,A,B,NA,MA,NB,MB,LAI,LA2,LBI,LB2)
    THIS SUBROUTINE CALCULATES THE FREQUENCY RESPONSE G, OF A TWO-DIMENSIONAL FILTER USING A TWO-
    DIMENSIONAL TIME PRUNED FFT.
    FOR EXPLANATION OF PARAMETERS SEE SUBROUTINE IDENT2.

    SUBROUTINE REQUIRED :
    FFTP2 : A TWO-DIMENSIONAL TIME PRUNED FFT.

C    REAL*8 G,A,B,REAL
C    DIMENSION X(8,8),X1(8,8),*(4,4),R(4,4)
    *A1=NI+1
    *A1=NA1+1
    *NI=2**LAI
    *V2=2**LAI2
    DC 1 I=1,MA1
    DC 1 J=1,NA1
    X(I,J)=DCMPLX(A(I,J),0.00)
    IF(N=1-N**2) 2,3,3
    2  IK=NA1+1
        DD 4 I=1,MA1
        DD 4 J=IK,NN2
    4  X(I,J)=(0.00,0.00)
    3  IF(N=1-N**1) 5,6,6
    5  IK=2 MA1+1
    DC 7 I=1,NN1
    DC 7 J=1,NN2
    7  X(I,J)=(0.00,0.00)
    6  CALL FFTP2(X,M1,M2,LA1,LA2)
        M=NI+1
        V1=NI+1
        *NI=2**L11
        M=M=M+1
        DC 8 I=1,MA1
        DC 8 J=1,NA1
    8  X(I,J)=DCMPLX(B(I,J),0.00)
    M=M=M=M+1
    DC 10 I=1,NN1
    DC 10 J=1,NN2
    10  IF(N=1-N**1) 12,13,13
    12  IK=M+1
        DC 14 I=1,NN1
        DC 14 J=1,NN2
    14  X(I,J)=(0.00,0.00)
    13  CALL FFTP2(X1,M1,M2,LA1,LA2)
        N1=2**N1
        N2=2**N2
        NT=N1*N2
        DC 15 I=1,N1
        DC 15 J=1,N2
        X1(I,J)=X1(I,J)/X1(I,J)
    15  X1(I,J)=X1(I,J)
    CALL IFFT2(X1,M1,M2)
    DC 16 I=1,N1
    DC 16 J=1,N2

16 5(I,J)=REAL(X1(I,J))/NT
RETURN
END
SUBROUTINE IFFT2(A,N1,M2)
  C
  THIS IS A SUBROUTINE FOR FINDING THE INVERSE FFT OF SPATIAL DATA, 
  USING A DECIMATION IN TIME ALGORITHM.
  COMPLEX*16 A(I, J), U, W, T, DCMPLEX
  REAL*8 PI, DCOS, DSIN
  DATA PI/3.14159265358979/
  N1=2**N1
  N2=2**N2
  NV2=N2/2
  NM1=N2-1
  DO 1 LO=1,N1
  1 J=1
  DO 2 I=1,NM1
  IF(I.GE.J) GO TO 3
  T=A(LO,J)
  A(LO,J)=A(LO,I)
  A(LO,I)=T
  3 K=NV2
  4 IF(K.GE.J) GO TO 2
  J=J+K
  K=K/2
  GO TO 4
  2 J=J+K
  DO 5 L=1,M2
  LE=2**L
  LE1=LE/2
  U=DCMPLEX(1.0D0,0.0D0)
  W=DCMPLEX(DCOS(PI/LE1), DSIN(PI/LE1))
  DO 6 J=1,LE1
  6 IP=1+LE1
  T=A(LO,IP)*U
  A(LO,IP)=A(LO,IP)+T
  A(LO,IP)=A(LO,IP)-T
  5 CONTINUE
  NV2=N1/2
  NM1=N1-1
  DO 7 LO=1,N2
  J=1
  DO 8 I=1,NM1
  IF(I.GE.J) GO TO 9
  T=A(I,LO)
  A(I,LO)=A(I,LO)-T
  A(I,LO)=A(I,LO)+T
  8 J=J+K
  9 K=NV2
  IF(K.GE.J) GO TO 8
  J=J-K
  K=K/2
  GO TO 10
  10 8 J=J+K
  DO 11 L=1,M1
  LE=2**L
  LE1=LE/2
  U=DCMPLEX(1.0D0,0.0D0)
  W=DCMPLEX(DCOS(PI/LE1), DSIN(PI/LE1))

Fig. D4
DO 12 I=J,M+1,LF
  IP=I+L-1
  T=A(IP,LO)*U,
  A(IP,LO)=A(I,LO)-T
  12 A(I,LO)=A(I,LO)+T
  11 U=U+W
  7 CONTINUE
  RETURN
END

FIG. D 4
APPENDIX E

DESIGN OF RECURSIVE FILTERS
TO APPROXIMATE A PRESCRIBED IMPULSE RESPONSE
APPENDIX E1

FORTRAN LISTING FOR THE
ONE-DIMENSIONAL FILTER DESIGN
SUBROUTINE IDENT(YD,G,M,A,R,NL,NA,NL,NB,LT,IT)

THIS SUBROUTINE IS FOP IDENTIFYING THE PARAMETERS OF A RECURSIVE
DIGITAL FILTER GIVEN THE DFT YD OF ITS IMPULSE RESPONSE.

EXPLANATION OF PARAMETERS:
1) YD = THE DFT OF THE IMPULSE RESPONSE
2) G = THE IMPULSE RESPONSE
3) 2**M = NUMBER OF SAMPLES
4) A AND B ARE THE COEFFICIENTS TO BE IDENTIFIED OF THE DISCRETE
FILTER
5) NA=ORDER OF NUMERATOR. NB=ORDER OF DENOMINATOR.
6) 2**LA AND 2**LB ARE THE NEAREST HIGHER NUMBERS TO NA+1 AND
   NB+1 RESPECTIVELY.
7) IT= REQUIRED NUMBER OF ITERATIONS

DIMENSIONS: X(2**M), YD(2**M), S(NB, NB), Y(NB), C(NA+1, NR), A(NA+1)
   B(NB+1), D(NA+1, NA+1), R1(NA+1), P2(NA+1), CI(NB, NB), C2(NA+1, NB),
   CW(NB, NB), RI(NA+1), WT(2**M-1), G(2**M), F(NB)

IF(NA+1, ST, NB) THEN C IS DIMENSIONED AS C(NA+1, NA+1)

SUBROUTINES REQUIRED:
1) FFTF, TIME-DOMAIN PRUNED FFT.
2) FFTF, FREQUENCY-DOMAIN PRUNED FFT.

SUBROUTINE IDENT(YD, G, M, A, R, NL, NA, NL, NB, LT, IT)

REAL*8 PI, DEP, DBS, CI(16, 16), A(17), B(17), D(16, 16), F1(16), F2(16),
   CI(16, 16), C2T(16, 16), R1(16), WT(128), S(128), F(16), ALPHN

N=2**M
M=2**N
N1=NA+1
N2=NB+1
N3=NA+2
KOUNT=0
KK=2**NB
NK=N2-KK
IND=0

IF(NB, EQ, 0) GO TO 300
DJ 400, I=1, NB
IF(G(N+1-I), NF, 0.0) GO TO 401
CONTINUE

SO TO 13
DJ 12 I=1, NA
D(N1-I, NA) = G(I) * G(I)

D(NA, N1-I) = G(N1-I, NA)
DJ 3 I=2, NA
DJ 3 J=1, NA
D(N1-I, NL-J) = D(N1-I, NL-J) + G(J) * G(I)

D(N1-J, N1-I) = D(N1-J, N1-I)
13 DJ 9 I=1, NB
DJ 9 J=1, NA
9dj(j, j), = 0.0
IF(NA, EQ, 0) GO TO 87
DJ 81 I=1, NA
DJ 81 J=1, NA
81 D(I, J), = 0.0
DJ 81 J=1, NA

GO TO 87
401 IND=1

DJ 402 J=1, NA
402 \( (1,J) = G(M4+1-J) \)

\[
\text{IF}(\text{NA}, \text{EQ}, 0) \text{ GO TO 403}
\]

\[
\text{D} \text{J} 404 \text{ I} = 2, \text{NL}
\]

404 \( C(I, I) = G(I-1) \)

\[
\text{IF}(\text{NB}, \text{LT}, 2) \text{ GO TO 403}
\]

\[
\text{D} \text{J} 405 \text{ I} = 2, \text{NL}
\]

\[
\text{D} \text{J} 405 \text{ J} = 2, \text{NB}
\]

405 \( C(I, J) = C(I-1, J-1) \)

403 \( \text{D} \text{J} 450 \text{ I} = 1, \text{NB} \)

\[
\text{D} \text{J} 450 \text{ J} = 1, \text{NB}
\]

\[
\text{D} \text{D} 455 \text{ I} = 0, \text{DO}
\]

\[
\text{D} \text{J} 450 \text{ K} = 1, \text{NL}
\]

450 \( D(I, J) = D(I, J) + C(K, I) \times C(K, J) \)

87 \( \text{D} \text{J} 10 \text{ I} = 1, \text{M4} \)

10 \( X(I) = \text{DCMPLX}((\text{CDARS}(YD(I)) \times 2), 0, 00) \)

\[
\text{CALL FFTFP}(X, M, LD)
\]

\[
\text{D} \text{J} 11 \text{ J} = 1, \text{NR}
\]

\( R2(J) = \text{DREAL}(X(J)) / \text{M4} \)

28 \( \text{D} \text{J} 28 \text{ I} = 1, \text{NB} \)

\[
\text{L} = 1
\]

\[
\text{D} \text{J} 28 \text{ J} = 1, \text{NR}
\]

\[
\text{C}(I, J) = 2(L) - D(I, J)
\]

28 \( \text{L} = \text{L} + 1 \)

\[
\text{IF}(\text{IND}, \text{EQ}, 1) \text{ GO TO 407}
\]

\[
\text{IF}(\text{NA}, \text{EQ}, 0) \text{ GO TO 19}
\]

\[
\text{D} \text{J} 29 \text{ I} = 1, \text{NA}
\]

\( R2(I) = 0, \text{DO} \)

\[
K = \text{NA} - 1 - I
\]

\[
\text{D} \text{J} 30 \text{ J} = 1, K
\]

30 \( R2(J) = R2(I) + G(J, I) \times G(J+I) \)

29 \( R2(I) = R2(M) - \text{DREAL}(X(I+I)) / \text{M4} \)

\[
\text{IF}(\text{NA}, \text{EQ}, \text{NR}) \text{ GO TO 900}
\]

19 \( \text{D} \text{J} 35 \text{ I} = 1, \text{NR} \)

35 \( R2(I) = -\text{DREAL}(X(I+I)) / \text{M4} \)

\( \text{GO TO 900} \)

407 \( \text{D} \text{J} 409 \text{ I} = 1, \text{NB} \)

\( R2(I) = 0, \text{DO} \)

\[
\text{D} \text{J} 408 \text{ J} = 1, \text{NI}
\]

408 \( R2(J) = R2(I) + C(J, I) \times G(J) \)

409 \( R2(I) = R2(I) - \text{DREAL}(X(I+I)) / \text{M4} \)

\( \text{GO TO 900} \)

7 \( \text{D} \text{J} 8 \text{ I} = 1, \text{M2} \)

8 \( X(I) = \text{DCMPLX}(\text{WT}(I), 0, 00) \)

302 \( \text{D} \text{J} 302 \text{ I} = M3, \text{M4} \)

\[
\text{CALL FFTFP}(X, M, LA)
\]

\[
\text{D} \text{J} 4 \text{ I} = 1, \text{NI}
\]

\[
\text{C}(I, I) = \text{DREAL}(X(I))
\]

4 \( \text{C}(I, I) = C(I, I) \)

\[
\text{D} \text{J} 5 \text{ I} = 2, \text{NI}
\]

\[
\text{D} \text{J} 5 \text{ J} = 1, \text{NI}
\]

\[
\text{C}(I, J) = C(I-1, J-1)
\]

5 \( \text{C}(J, I) = C(I, J) \)

\( \text{ORDERING METHOD FOR SYMMETRICAL MATRIX INVERSION} \)

\[
D(I, I) = 1.00 / C(I, I)
\]

\[
\text{IF}(\text{NA}, \text{EQ}, 0) \text{ GO TO 45}
\]

\[
\text{D} \text{J} 801 \text{ I} = 1, \text{NA}
\]

\( \text{D} \text{J} \text{ } \text{ } \text{ } \text{ } \text{ } \text{ } \)
DJ 802 K=1,I
F(K)=0.0
DJ 802 J=1,I
802 F(K)=F(K)+2(K,J)*C(I+1,J)
D(LPHN)=C(I+1,J+1)
DJ 803 J=1,I
803 D(LPHN)=D(LPHN)-C(I+1,J)*F(J)
DJ 804 J=1,I
804 D(K,J)=D(J,K)
D(J,J)=D(J,J)+F(J)*F(J)/D(LPHN)
805 D(J,J)=D(J,J)/D(LPHN)
806 D(I+1,J)=F0(I+1,J)
807 D(I+1,I+1)=1.00/D(LPHN)

45 DJ 202 I=1,M2
202 X(I)=YD(I)*WT(I)
DJ 301 I=M3,M4
301 YD(I)=C1MPLX(0.00,0.00)
203 CALL FFTFP(X,M,LR)
DJ 14 J=1,MR
14 C2(I,J)=C1DFAL(X(J+1))
IF(NA.EQ.0) GO TO 70
DJ 61 I=1,M2
61 X(I)=C1NH5(YD(I))*WT(I)
DJ 303 I=M3,M4
303 YD(I)=C1MPLX(0.00,0.00)
203 CALL FFTFP(X,M,LA)
DJ 18 I=2,NI
18 C2(I+1)=C1DFAL(X(I-1))
IF(NB.LT.2) GO TO 70
DJ 17 I=2,NI
17 J=2,MR
17 J=2,MR
17 C2(I,J)=C2(I-1,J-1)
70 DJ 46 I=1,N1
46 R1(I)=C1DFAL(X(I))
DJ 100 I=1,NB
100 J=1,N1
100 C1(I,J)=0.00
100 K=1,N1
100 C1(I,J)=C1(I,J)+C2(K,I)*D(K,J)
DJ 101 I=1,NB
101 J=1,NB
101 C1(I,J)=0.00
101 K=1,N1
101 C1(I,J)=C1(I,J)+C2(K,I)*C2(K,J)
DJ 102 I=1,NB
R(I)=0.00
DJ 102 J=1,N1
102 S(I)=B(I)+C1(I,J)*R1(J)
DJ 213 I=1,N2
213 X(I)=C1MPLX(C1ABS(YD(I))**2)*WT(I),0.00
304 I=M3,M4
304 YD(I)=C1MPLX(0.00,0.00)
203 CALL FFTFP(X,M,LB)
DJ 27 J=1,NI
27 C2(I+1,J)=DFEAL(X(J))
SQUARE-ROOT METHOD FOR SOLUTION OF SIMULTANEOUS LINEAR EQUATIONS.

```
900 P=CCMPLX(C1(I,1),0.D0)
   S(1,1)=CMQRT(P)
   DO 701 J=2,NB
    701 S(1,J)=C1(I,J)/S(1,1)
   DO 702 I=2,NB
    702 S(I,1)=CMPLX(-C1(I,1),0.D0)
   I=I-1
   I2=I+1
   DO 703 L=1,I1
   703 S(I,1)=S(I,1)+(S(L,1)*S(L,1))
   S(I,1)=CMQRT(-S(I,1))
   IF(I.EQ.NB) GO TO 709
   DO 704 J=2,NB
    704 S(I,J)=CMPLX(-C1(I,J),0.D0)
   DO 705 L=1,I1
   705 X(L)=S(L,1)/S(1,1)
   DO 706 I=2,NB
    706 X(I)=CMPLX(-P2(I),0.D0)
   I=I-1
   DO 707 L=1,I1
   707 Y(L)=X(L)*S(L,1)
   Y(NN1)=X(NN1)/S(NR,NB)
   NN1=NB-1
   DO 708 L=1,NN1
   708 Y(NNN)=Y(NNN)+S(NNN,K)*Y(K)
   Y(NNN)=-Y(NNN)/S(NNN,NNN)
   IF(I.T.EQ.0) GO TO 300
   X(I)=CMPLX(1.D0,0.D0)
   DC 104 I=2,N2
   104 X(I)=Y(I-1)
   IF(NK) 105*106,106
   105 JJ=N2+1
   DC 107 I=JJ,KK
   107 X(I)=CMPLX(0.D0,0.D0)
   CALL FFTPR(X,N,M,LB)
   DC 108 I=1,N2
   108 W(I)=1.D0/CDABS(X(I)**2)
```

KOUNT=KOUNT+1
IF(KOUNT.LE.IT) GO TO 7

300 B(I)=1.D0
IF(NB.EQ.0) GO TO 500
DC 36 I=2,N2
36 B(I)=3PE4L(Y(I-1))
IF(IT) 906,901,906

600 DC 601 I=1,N1
601 A(I)=G(I)
GO TO 902
901 IF(IND.EQ.0) GO TO 410
DC 411 I=1,N1
A(I)=G(I)
DC 411 J=1,NB
411 A(I)=A(I)+C(I,J)*B(J+1)
GO TO 902

410 DC '903 I=1,N1
A(I)=0.D0
L=1
DC '903 J=1,1
A(I)=A(I)+C(I,J)*B(J)
903 L=L+1
GO TO 902
906 DC 33 I=1,N1
DC 33 J=1,NB
33 R(I)=R(I)-C2(I,J)*B(J+1)
DC 110 I=1,N1
A(I)=0.D0
DC 110 J=1,N1
110 A(I)=A(I)+R(I,J)*R(I,J)

902 WRITE(6,32)
32 FORMAT(5X,'TRANSFER FUNCTION IS GIVEN BY 7X,'G(Z)=(A(1)+A(2)*Z+A(13)*Z**2+...+A(NA+1)*Z**NA)/(1.0+R(12)*Z+R(3)*Z**2+...+R(NR+1)*Z**NB)
'7X, 'WHERE Z=EXP(-J*HT), W=FREQ. IN RADIANS/SEC., T=SYMLING TIME')
907 WRITE(6,907)(I,A(I),I=1,N1)
907 FORMAT(20X,'A(1','13','=')=','N25.18)
WRITE(6,80)
80 FORMAT(//)
WRITE(6,37) (I,B(I),I=1,N2)
37 FORMAT(20X,'B(1','13','=')=','N25.18)
RETURN

---END---

REAL FUNCTION DREAL*8(X)
COMPLEX*16 X,DCOMJS
DREAL=IN*+DCONJS(X)/2
RETURN
END
APPENDIX E2

FORTRAN LISTING FOR THE
SPATIAL FILTER DESIGN
NKK2=NA1-ML+2
NP1=(N1/2)+1

C......0- ITERATION

IF(M1.EQ.0) GO TO 82
IF(NA1.EQ.0) GO TO 6
DJ 2 I=2,NA1
2 D(I,1)=G(I,1-I)
6 IF(NB.EQ.0) GO TO 51
DJ 200 J=1,MA
200 C(J1,J1)=G(I,J1-I-J-1)
C(J1,J1)=G(I,J1-I-J-1)
51 IF(MA.EQ.0) GO TO 26
DJ 4 I=1,MA

4 D(NA1+I,NA1)=G(I,1)
6 IF(NB.EQ.0) GO TO 25
DJ 5 I=1,NA1
8 DO 5 J=1,MA
9 DO 5 I=1,MA

25 IF(MA.EQ.0) GO TO 26
DJ 7 II=2,MA
K=II+NP1-1
L=II*MA

26 IF(M1.EQ.0) OR K.EQ.MM1 GO TO 26
DJ 7 I=1,MA
7 DO 7 J=1,MA

7 D(I+K+J)=C(NA1+I+NA1+J)
26 IF(M1.EQ.0) OR K.EQ.MM1 GO TO 26
7 DO 10 J=2,NA1
10 DO 10 J=2,NA1

101 D(I+KK)=G(N1+2-11,J)
1001 D(I+KK)=G(N1+2-11,J)
10 IF(MA.EQ.0) GO TO 12
DJ 202 I=1,MA
8 DO 202 J=1,MA
9 DO 202 J=1,MA
202 D(I+11)=G(N1+2-11,N2+1-I)
202 D(I+11)=G(N1+2-11,N2+1-I)
8 IF(MA.EQ.0) GO TO 12
DJ 203 I=2,MA
203 D(I+11)=G(N1+2-11,N2+1-I)
32 DO 203 J=2,NA1
31 L=KK+J-1
203 D(I,L)=C(I-1,L-1)
32 DO 13 I=1,MA
34 LL=II+1
34 LL=II+1
35 D(I+11)=G(N1+2-11,N2+1-I)
13 D(I,1)=LL*MA1
37 KU=II+NP1-1
13 D(I,1)=LL*MA1
38 LU=LU+NA1
9 KU=II*MA1
39 IF(LU.EQ.MM OR KU.GT.MM1) GO TO 9
9 L=LU-NA1
71 K=LU-MB1
72 DJ 13 I=1,MA1
73 DJ 13 J=1,NA1
13 L=L+I*MK1
74 CONTINUE
75 IF(MA.EQ.0) GO TO 20
27
DO 16 II=2,NA1
    KK=(II-1)*NA1+1
    IF(NB.EQ.0) GO TO 19
    IF(NA.EQ.0) GO TO 90
    DO 17 I=1,NA
    K=KK+I
17    D(K,I)=D(II,I)
    DO 204 I=1,NB
204    D(KK,I)=S(II,I,N2+1-I)
    IF(NB.LT.2.OR.NA1.LT.2) GO TO 19
    DO 20 J=2,NB
    DO 18 J=2,NA1
    L=KK+J-1
18    DO (L,I)=D(L-1,I-1)
    DO 19 KU=(II+1)*NA1
    LU=NB1+N0
    IF(KU.GT.MM1.OR.LU.GT.MM1) GO TO 16
    DO 20 I=1,NA1
    K=II*NA1+1
    DO (K,M)=D(K-NA1,L-NA1)
    LL=II+2
    I=(NA1.LT.LL) GO TO 16
    DO 23 J=LL,NA1
    K=(J-11)*NA1
    L=L+NA1
    IF(K.EQ.MM1.OR.L.EQ.MM1) GO TO 16
    DO 24 II=1,NA1
    K1=K+I
    DO 24 J=1,NP1
    L1=L+J1
24    DO (K1,L1)=D(K1-NA1,L1-NA1)
23    CONTINUE
22    16 CONTINUE
21    15 CONTINUE
19    DO 31 I=1,MM1
18    DO 31 J=I,MM1
17    C(I,J)=0.00
16    DO 31 L=I,MM
15    DO 31 I=1,MM
14    DO 31 J=1,MM
13    C(I,J)=C(I,J)+D(L,I)*D(L,J)
12    DO 32 K=1
11    DO 32 J=1,NA1
10    DO 32 I=1,NA1
9     E1(K)=G(I,J)
8    DO 32 K=K+1
7    IF(MM1.EQ.0) GO TO 81
6    DO 83 I=1,MM1
5    E2(I)=0.00
4    DO 83 J=1,MM1
3    E2(I)=E2(I)+D(J,I)*E1(J)
2    DO 83 I=1,N1
1    DO 34 J=1,N2
0    DO 34 L=1,MM
-1   X(I,J)=COMPLEX(CDABS(YD(I,J1))**2,0.00)
CALL FFTP2(X, M1, M2, LBI, LB2)
IF(NB. NE. 0) GO TO 1003
DO 1004 I=1, MB
1004 E(I, I) = DRFAL(X(I, I)) / NT
DO 1005 I=2, MB
1005 E(I, J) = E(I-1, J)
CC TO 1008
1003 DO 35 I=1, MB
35 E(I, I) = DRFAL(X(I, I)) / NT
IF(NB. LT. 2) GO TO 30
DO 36 I=2, MB
36 E(I, J) = E(I-1, J)
IF(NR. LE. 0) GO TO 86
30 K=1
37 NN=K*NB1
IF(NN. LT. MM1) GO TO 40
MK=NN+MB-1
DO 38 I=NN, NK
38 E(I, J) = E(I-NB, J-NB)
E(NN, NK+1) = DRFAL(X(I, NP1)) / NT
DO 39 I=1, NP1
39 E(NN+I, NK+1) = E(NN+I-1, NK)
K=K+1
GO TO 37
40 DO 47 II=1, MB
47 E(I, II * NB1+1) = DRFAL(X(II+1, I)) / NT
42 E(I, II * NB1) = DRFAL(X(II+1, MK+1)) / NT
KK=II * NB1
IF(NB. LT. 2) GO TO 88
DO 43 I=2, NP1
43 K=KK+I-1
DO 44 J=2, NR
44 E(J, K) = E(J-1, K)
88 LJ=II * NB1-1
LL=1
46 NN=LL * NP1
LU=LU+NB1
IF(NN, NT, MM1-OPLUEC, MM1) GO TO 47
DO 45 I=1, NR1
45 E(IN, LU+I) = DRFAL(X(I, I)) / NT
DO 46 I=1, NR
46 E(I, LU+J) = E(I, KK+J-1)
LL=LL+1
GO TO 46
47 CONTINUE
86 DO 41 I=2, NB1
41 E(I-1, I) = DRFAL(X(I, I)) / NT
IF(NR. LE. 0) GO TO 1006
K=II+1
DO 48 I=2, MB1
48 K=K+1
DD TO 1006
1008 DD 1007 I=1, MB
1007 DD(I)=DREAL(X(I(I+1,I)))/N
1006 DD 49 J=1, MB
1006 DD 49 J=1, MB
49 CI(I,J)=CI(I,J)+F(I,J)
1006 DD 50 I=1, MB
1006 DD 50 I=1, MB
50 E2(I)=E2(I)+E3(I)
1006 DD TO 55

C IT REAT IVE ALGORITHM
440 DD 300 I=1, MB
440 DD 300 J=1, MB
300 X(I,J)=COMPLEX(R(I,J),0.0)
300 IF (NKK2) 301, 302, 303
301 IK=NB1+1
301 DD 303 I=1, MB
301 DD 303 J=IK,NLR2
301 DD 303 J=IK,NLR2
303 X(I,J)=0.0,0.0
302 DD (NKK1) 304, 305, 305
304 IK=NL1+1
304 DD 306 I=IK,NLR1
304 DD 306 J=1, NLR2
306 DD 306 J=1, NLR2
305 CALL FFTP2(X*,M1,M2,L1,L2)
305 DD 313 I=1, NP1
305 DD 313 J=1, N2
313 WT(I,J)=L.D0/(CDABS(X(I,J)*+2)
313 DD 307 I=1, NP1
313 DD 307 J=1, N2
307 X(I,J)=COMPLEX(KT(I,J),0.0)
307 LL=NP1+1
307 DD 308 I=LL, NL
307 DD 309 J=1, N2
308 DD 308 J=1, N2
308 X(I,J)=0.0,0.0
308 CALL FFTP2(X*,M1,M2,L1,L2)
308 DD 310 I=1, NA1
310 CI(I,I)=DREAL(X(I,I))
310 DD (MA, EO) 311 TO 1100
310 DD 311 I=2, NA1
310 DD 311 J=1, NA1
311 CI(L,J)=C(I-1,J-1)
1100 DD (MA, EO) 312 TO 601
1100 DD 312 I=1, MA
1100 DD 312 I=1, MA
1100 DD 312 J=1, MA
1100 DD 312 J=1, MA
312 CI(K+I,K+J)=C(K+I-NA1,K+J-NA1)
312 DD 315 I=1, MA
312 DD 315 I=1, MA
312 DD 315 J=1, MA
312 DD 315 J=1, MA
316 CI(I,K+I)=DREAL(X(I+I,I))
316 DD (MA, EO) 318 TO 1101
316 DD 318 I=2, NA1
318 CI(I,K+I)=DPFAL(X(I+1,MBK+1-1))
318 DD 319 J=2, NA1
319 CI(I,L)=C(I-1,L-1)
1101 I1=MA-II
  IF(I1.EQ.0) GO TO 601
  CC 320 I=1,II
  L=(II+1)*NA1
  K1=I*NA1
  CC 320 I2=1,NA1
  K2=K1+12
  CC 320 J=1,NA1
  L1=L+J
  320 C(K2,L1)=C(K2-NA1,L1-NA1)
  315 CONTINUE
  601 CC 6QO I=1,MM
  CC 600 J=1,MM
  CC 600 (J+I)=C(I,J)
  C THE BORDERING METHOD FOR SYMMETRICAL MATRIX INVERSION
  321 F(I,J)=1.0D0/C(I,J)
    MM=M=M+1.
    IF(MM.EQ.1) GO TO 77
    CC 322 I=1,MM
    CC 323 K=1,1
    F(K)=0.0D0
    CC 323 J=1,1
    323 F(K)=F(K)+F(K,J)*C(I+1,J)
    ALPHN=C(I+1,I+1)
    CC 324 J=1,1
    CC 325 J=1,1
    CC 325 K=J,1
    CC 325 E(J,K)=E(J,K)+F(J)*C(I+1,J)
    CC 325 E(K,J)=E(K,J)+F(J)*ALPHN
    CC 326 J=1,1
    CC 326 J=1,1
    CC 326 E(I+1,J)=E(J,K)
    CC 326 E(I+1,J)=E(I+1,J)
    CC 326 E(I+1,J)=E(I+1,J)
    77 CC 327 I=1,N1
    CC 327 J=1,N2
    327 XI(I,J)=YD(I,J)*WT(I,J)
    CC 328 I=II,N1
    CC 328 J=I,N2
    328 XI(I,J)=YD(I,J)*WT(I,J)
    CALL FFTP2(X,M1,M2,LB1,LB2)
    IF(N3.EQ.0) GO TO 78
    CC 330 I=1,NB
    CC 330 I=1,NB
    330 D(I,I)=DREAL(X(I+1,I+1))
    78 IF(NB.EQ.2) GO TO 1103
    CC 332 II=2,MA1
    KK=(II-1)*NP1-1
    CC 332 I=1,NB1
    332 D(KK+I)=DREAL(X(I+1,I+1))
    1103 IF(MA1.LT.2 .OR. NB.EQ.0) GO TO 370
    CC 338 I=2,MA1
    KK=(II-1)*NA1+1
    CC 338 I=1,NA1
    CC 338 I=1,NA1
    338 D(KK+I)=DREAL(X(I+1,MK+I))
    370 CC 333 I=1,NP1
    CC 333 J=1,N2
    333 D(I,J)=DCONJG(YD(I,J))*WT(I,J)
    CC 334 I=II,N1
DO 334 J=1,N2
334 XI,JJ=(0.00,0.00)
CALL FFTFP2(X,J1,M1,M2,LA1,LA2)
IF(NA,EQ.0) GO TO 348
CC 341 I=2,NA1
341 D(I+1)=DREAL(X(I+I))
IF(NB.LT.2) GO TO 34B
CC 342 J=2,NA1
CC 342 J=J+1
342 D(I,J)=D(I,J-1)
348 IF(MA.EQ.0) GO TO 347
CC 343 I=I+1,NA1
343 D(NA1+I,NE1)=DREAL(X(I+I))
IF(NB.EQ.0) GO TO 345
CC 344 J=J+1,NA1
CC 344 J=J+1
344 D(NE1+J,NE1+J)=D(I,J)
345 IF(MA.LT.2) GO TO 347
CC 346 II=II+1,MA
K=II*NB1-1
L=II*NA1
IF(L.EQ.MM OR K.EQ.MM1) GO TO 347
CC 346 I=I+1,NA1
CC 346 J=J+1,NA1
346 D(L+J,K+J)=D(NA1+I,NA1+J)
347 IF(MA.LT.2 OR NA1.LT.2) GO TO 371
CC 349 II=II+1,MB1
KK=(II-1)*NB1
CC 350 J=J+1,NA1
350 D(J,KK)=DREAL(X(II+MKK+J))
IF(NB.LT.2) GO TO 340
DO 351 J=2,NB1
L=KK+J-1
DO 351 I=2,NA1
351 D(I,L)=D(I-1,L-1)
340 LU=NA1
LL=II+1
IF(MB1.LT.L.GetObject+MA1.LT.2) GO TO 349
CC 352 I=I+1,MB1
KU=I*KM-1
L=LU+NA1
IF(L>GT.MM OR KU>GT.MM1) GO TO 349
L=LU-NA1
K=KU-NB1
CC 352 I=I+1,NA1
CC 352 J=J+1,NB1
352 D(I+1,K+J)=D(L+I-NA1,K+J-NB1)
349 CONTINUE
371 IF(MA.EQ.0) GO TO 372
DO 353 I=1,MA1
KK=(II-1)*NA1+1
IF(NB.EQ.0) GO TO 35B
IF(NA.EQ.0) GO TO 359
DO 354 I=1,NA
K=KK+1
CC 351 D(K+1)=DREAL(X(II+I))
CC 356 IF(NB.LT.2 OR MA1.LT.2) GO TO 356
DO 366 J=2,NA1
356 CONTINUE
70 366 J=J+1,NA1
L = KK + J - 1
DO 366 I = 2, NB
366 D(L, I) = D(L - 1, I - 1)
356 KU = (II + 1) * NA1
LJ = II * NA1 + NB
IF(KU .GE. MM, GO TO 353)
K = II * NA1 + I
DO 357 I = 1, NA1
357 D(K, NB1) = DREAL(X(I, J))
IF(NB1 .EQ. 0) GO TO 359
K = II * NA1 + I
DO 358 J = 1, NB
358 D(K, L) = D(K - NA1, L - NB1)
359 L = NR
LL = II + 2
IF(MA1 .LT. LL) GO TO 353
DO 360 J = LL, MA1
360 K = (J - 1) * NA1
L = L + NB1
IF(K .EQ. MM, GO TO 353)
DO 361 JI = 1, NA1
361 KI = K + II
DO 362 J1 = 1, NB1
362 L = L + J1
360 CONTINUE
353 CONTINUE
372 L = 1
DO 362 I = 1, MA1
DO 362 J = 1, NA1
E1(L) = DREAL(X(I, J))
362 L = L + 1
DO 400 I = 1, NP1
DO 400 J = 1, N2
400 XI, J = DCMPLX((CDARS(YD(I, J)) + X(I, J)) * 2) * WT(I, J), 0.00
401 XI, J = DREAL(X(I, J))
401 XI, J = (0.0, 0.0, 0.0)
CALL FFTP2(X, M1, M2, LA1, L2)
407 IF(NB1 .NE. 0) GO TO 1104
408 DO 1105 I = 1, MB
1105 CI, I = DREAL(X(I, J))
1108 C1 = (1 - J, - 1)
DO 1106 J = 1, MB
1106 CI, J = C(I - 1, J - 1)
403 CI, J = DREAL(X(I, J))
403 CI, J = (0.0, 0.0, 0.0)
407 IF(NB1 .LT. 2) GO TO 405
408 DO 404 I = 2, NB
404 DO 404 J = 1, NB
404 XI, J = C(I - 1, J - 1)
405 XI, J = C1
405 XI, J = (1 - 1, 1 - 1)
405 K = 1
406 NN = K * MB1
IF(NN.GT.MM1) GO TO 409
NN=NN+NR-1
CJ 407 I=MM1,NK
CJ 407 J=1,NK
CJ 407 C(I,J)=C(I-NR,J-NB)
C(JN,NK+1)=DREAL(X(I1,NB1))
CJ 408 I=1,NN
CJ 408 C(NN+I,NK+1)=C(NN+I-1,NK)
CJ K=K+1
CJ GO TO 406
CJ 409 CJ 410 I=1,MB.
CJ 411 I=1,NN
CJ C(I,II+NN1+1)=DREAL(X(II+1,II))
CJ C(I,II+NN1)=DREAL(X(II+1,II+NN1))
CJ KK=II+NN1
CJ IF(NB.LT.2) GO TO 415
CJ 412 I=2,NN1
CJ K=KK-I-1
CJ 412 J=2,NN
CJ C(J,K)=C(J-1,K-1)
CJ 415 LU=LL+NN1-1
CJ LL=1
CJ 416 NN=LL*NN1
CJ LU=LU+NN1
CJ IF(NN*ST.MM1,NN,LU*FO.MM1) GO TO 410
CJ 417 I=1,NN1
CJ C(NN,LU+I)=DREAL(X(II+1,II))
CJ 418 I=1,NN
CJ 418 J=1,NN1
CJ C(NN+I,LU+J)=C(I,KK+J-1)
CJ LL=LL+1
CJ GO TO 416
CJ 410 CONTINUE
CJ 419 CJ 420 I=2,NN1
CJ 420 EZ(I-1)=-DREAL(X(II,II))
CJ IF(MB.EQ.0) GO TO 1107
CJ K=NN+1
CJ 421 I=2,MB1
CJ 421 J=1,NN1
CJ EZ(K)=-DREAL(X(Z,J))
CJ 421 K=K+1
CJ GO TO 1107
CJ 1108 CJ 1109 I=1,MB
CJ 1109 EZ(I)=-DREAL(X(II+1,II))
CJ 1107 CJ 430 I=1,MM1
CJ 430 J=1,MM
CJ R(I,J)=0.DO.
CJ 430 K=1,MM
CJ R(I,J)=R(I,J)+D(K,I)*F(K,J)
CJ 431 I=1,MM1
CJ 431 J=1,MM1
CJ 431 K=1,MM
CJ 431 C(I,J)=C(I,J)-F(I,K)*D(K,J)
CJ 432 I=I+1,MM1
CJ 432 J=J+1,MM
CJ 432 EZ(I)=EZ(I)-R(I,J)*EL(J)
CJ SQUARE-ROOT METHOD FOR THE SOLUTION OF A SET OF SIMULTANEOUS
CJ LINEAR EQUATIONS WITH A SYMMETRICAL MATRIX.
55 P = N × M + L × (C(I, J) × 0.00)  
56 S(I, J) = C SORT(P)  
57 IF (M1 × L < 2) GO TO 50  
58 S(I, J) = C(I, J) / S(I, J)  
59 A = 50  
60 J = 2 × M1  
61 S(I, J) = D M PLEX(−C(I, J), 0.00)  
62 I = I − 1  
63 J = J + 1  
64 GO TO 56  
65 S(I, J) = S(I, J) + S(L, I) × S(L, J)  
66 S(I, J) = S(I, J) / S(I, J)  
67 XX(I) = C(I, J) / S(I, J)  
68 IF (M1 × L < 2) GO TO 68  
69 A = 60  
70 I = 2 × M1  
71 XX(I) = D M PLEX(−I, J, 0.00)  
72 I = I − 1  
73 GO TO 60  
74 XX(I) = XX(I) + S(L, J) × XX(L)  
75 XX(I) = XX(I) / S(I, J)  
76 Y(M1) = XX(M1) / S(M1, M1)  
77 M1 = M1 − 1  
78 A = 64  
79 I = 1 × N  
80 NNN = M1 − 1  
81 Y(NNN) = Y(NNN) − XX(NNN)  
82 A = 65  
83 L = 1, I  
84 K = M1 − L + 1  
85 Y(NNN) = Y(NNN) + C(NNN, N) × Y(K)  
86 Y(NNN) = Y(NNN) / S(NNN, NNN)  
87 A = 70  
88 B(I, J) = I × 0.00  
89 IF (N1 × L < 2) GO TO 74  
90 A = 66  
91 I = 1 × N1  
92 K = N1  
93 A = 67  
94 I = 2 × M1  
95 A = 68  
96 J = 1 × N1  
97 3(I, J) = DREAL(Y(K))  
98 K = K + 1  
99 KOUNT = KOUNT + 1  
100 IF (KOUNT × L × IT) GO TO 440  
101 IF (IT) = 442, 441, 442  
102 A = 441  
103 I = 1 × M1  
104 A = 70  
105 J = 1 × M1  
106 E(I, J) = E(I, K)  
107 A = 71  
108 K = K + 1  
109 A = 73  
110 GO TO 500  
111 A = 452  
112 A = 443  
113 I = 1 × M1  
114 A = 443  
115 J = 1 × M1
443 E1(I) = E1(I) - D(I,J) * DREAL(Y(J))
   DO 444 I = 1, MM
   444 E2(I) = 0.0
   DO 444 J = 1, MM
   444 E2(I) = E2(I) + E(I,J) * F1(J)
   K = 1
   DO 445 I = 1, NA1
   445 K = K + 1
   DO 446 J = 1, NA1
   WRITE(6, 72) I, J, A(I, J)
   DO 447 I = 1, NBI
   DO 447 J = 1, NBI
   WRITE(6, 69) I, J, B(I, J)
   69 FORMAT(10X, 'P(', ',I3', ',', ',I3', ')=', ',D16.9)
   72 FORMAT(10X, 'A(', ',I3', ',', ',I3', ')=', ',D16.9)
   RETURN
   END
APPENDIX F

LISTINGS OF THE GRADIENTS OF THE PERFORMANCE CRITERION FOR THE DESIGN OF SECOND-ORDER, AND FOURTH ORDER CASCADES OF RECURSIVE FILTERS DISCUSSED IN CHAPTER IV.
APPENDIX F

LISTINGS OF THE GRADIENTS OF THE PERFORMANCE CRITERION FOR THE DESIGN OF SECOND-ORDER, AND FOURTH ORDER CASCADES OF RECURSIVE FILTERS DISCUSSED IN CHAPTER IV.

(a) SECOND-ORDER CASCADES

\[ \frac{\partial \psi}{\partial u} = 2 \sum_{\ell=1}^{M_1} \sum_{i=1}^{M_2} (A^* |H_{\ell_i}| - y_{\ell_i}) \frac{\partial |H_{\ell_i}|}{\partial \psi u} \]  

\[ (F.1) \]

where; \( \psi_u = (a_k, b_k, c_k, d_k, e_k, f_k)^T \)

\[ \frac{\partial |H_{\ell_i}|}{\partial a_k} = |H_{\ell_i}| \cdot \Re \left\{ \frac{2a_k (1-z_{2i}) (1+z_{1\ell})}{N_k} \right\} \]

\[ \frac{\partial |H_{\ell_i}|}{\partial b_k} = |H_{\ell_i}| \cdot \Re \left\{ \frac{2b_k (1-z_{1\ell}) (1-z_{2i})}{N_k} \right\} \]

\[ \frac{\partial |H_{\ell_i}|}{\partial c_k} = |H_{\ell_i}| \cdot \Re \left\{ \frac{2c_k (1+z_{1\ell}) (1+z_{2i})}{D_k} \right\} \]

\[ \frac{\partial |H_{\ell_i}|}{\partial d_k} = -|H_{\ell_i}| \cdot \Re \left\{ \frac{2d_k (1-z_{2i}) (1+z_{1\ell})}{D_k} \right\} \]

\[ \frac{\partial |H_{\ell_i}|}{\partial e_k} = -|H_{\ell_i}| \cdot \Re \left\{ \frac{2e_k (1-z_{1\ell}) (1-z_{2i})}{D_k} \right\} \]
\[
\frac{\partial |H|}{\partial f_k} = -|H_{z_1}| \cdot \mathfrak{Re} \left\{ \frac{2f_k (1+z_1)(1+z_2)}{D_k} \right\}
\]

where;
\[
H(z_1, z_2) = \prod_{k=1}^{K} \frac{(1-z_1)(1+z_2) + a_k^2 (1-z_2)(1+z_1) + b_k^2 (1-z_1)(1-z_2) + c_k^2 (1+z_1)(1+z_2)}{(1-z_1)(1+z_2) + d_k^2 (1-z_2)(1+z_1) + e_k^2 (1-z_1)(1-z_2) + f_k^2 (1+z_1)(1+z_2)}
\]

and;
\[
z_{1,\ell} = e^{-j\Omega_1 \ell \pi}, \quad z_{2,\ell} = e^{-j\Omega_2 \ell \pi}
\]

\[
|\Omega_1| \leq 1.0
\]

\[
|\Omega_2| \leq 1.0
\]

(b) **FOURTH-ORDER CASCADES**:

As an example, consider the form given in sec. 4.3.3 eqn. (4.40):

Let:
\[
H = \prod_{k=1}^{K} H_{1_k} H_{2_k}
\]
where:

\[
H_k = \frac{p_2 + a_k^2 a_1 + b_k^2 + j c_k}{p_2 + d_k^2 + e_k^2 + j f_k} = \frac{N_{1k}(p_1, p_2)}{D_{1k}(p_1, p_2)}
\]

(F.5)

Hence:

\[
H_k = \frac{N_{1k}(p_1, p_2) - 2 j c_k}{D_{1k}(p_1, p_2) - 2 j f_k} = \frac{N_{2k}(p_1, p_2)}{D_{2k}(p_1, p_2)}
\]

(F.6)

Substituting \( p_2 = \frac{1 - z_1}{1 + z_2} \), \( p_1 = \frac{1 - z_1}{1 + z_1} \) in equations (F.5) and (F.6) we get

\[
H_k(z_1, z_2) = \frac{(1 - z_2)(1 + z_1) + d_k^2(1 - z_1)(1 + z_2) + (b_k^2 + j c_k)^*}{(1 + z_2)(1 + z_1) - 2 j f_k(1 - z_1)(1 + z_2) + (e_k^2 + j f_k)^*}
\]

\[
H_k(z_1, z_2) = \frac{N_{1k}(z_1, z_2)}{D_{1k}(z_1, z_2)}
\]

The gradients are now given by equations (F.1) and

\[
\frac{\partial H}{\partial a_k} = |H| \Re \left\{ 2 a_k (1 - z_1) (1 + z_2) \right\} \left\{ \frac{1}{N_{1k}^2} + \frac{1}{N_{2k}^2} \right\}
\]
\[ \frac{\partial |H|}{\partial b_k} = |H| \cdot \Re \left\{ \frac{2b_k (1+z_{1i}) (1+z_{2i})}{N_{k1\ell_i}} \left\{ \frac{1}{N_{k1\ell_i}} + \frac{1}{N_{k2\ell_i}} \right\} \right\} \]

\[ \frac{\partial |H|}{\partial c_k} = |H| \cdot \Re \left\{ \frac{j (1+z_{1i}) (1+z_{2i})}{N_{k1\ell_i}} \left\{ \frac{1}{N_{k1\ell_i}} - \frac{1}{N_{k2\ell_i}} \right\} \right\} \]

\[ \frac{\partial |H|}{\partial e_k} = -|H| \cdot \Re \left\{ \sqrt{2} d_k (1-z_{1i}) (1+z_{2i}) \left\{ \frac{1}{D_{k1\ell_i}} + \frac{1}{D_{k2\ell_i}} \right\} \right\} \]

\[ \frac{\partial |H|}{\partial f_k} = -|H| \cdot \Re \left\{ \frac{2e_k (1+z_{1i}) (1+z_{2i})}{D_{k1\ell_i}} \left\{ \frac{1}{D_{k1\ell_i}} - \frac{1}{D_{k2\ell_i}} \right\} \right\} \]

where; \( z_{1i} \), \( z_{2i} \) and \( \psi_0 \) are defined as before.
APPENDIX G

STABLE TWO-DIMENSIONAL RECURSIVE FILTER
DESIGN TO APPROXIMATE MAGNITUDE-
FREQUENCY SPECIFICATIONS
APPENDIX G

STABLE TWO-DIMENSIONAL RECURSIVE FILTER DESIGN TO APPROXIMATE MAGNITUDE-FREQUENCY SPECIFICATIONS

Listing of a Fortran programme for the design of Huang's special class of second-order filter.

The programme is written in Fortran IV, double-precision complex arithmetic.

NOTE: PLOT8 and PICOUT are subroutines for plotting the desired and designed magnitude-frequency specifications in a picture-form (top-view), with 20 levels of intensity on the line-printer.
MAIN PROGRAM

DESIGNING TWO-DIMENSIONAL DIGITAL FILTERS WITH REAL POLES AND ZEROS
ON THE \( w_1 = 0 \) & \( w_2 = 0 \) AXES.
EXAMPLE: A LOW-PASS FILTER.
REAL=9 X,Y, M, H, C, A, P
REAL=8 WIN, W1, W2
DIMENSION Y0(31, 61), M(31), N2(41), X(12), H(39), A(3), B(3)
COMMON/Y0/6/6
COMMON/W1, W2
LOGICAL=M(31, 61)
COMMON/W1
LIMIT=30
K=1
N=6
M1(1)=0.00
DO 1 I=1, 12
   1 M1(I+1)=M1(I)+0.0100
   M1(14)=0.200
   DO 2 I=14, 18
      2 M1(I+1)=M1(I)+0.100
         M1(19)=0.800
         M1(20)=1.000
         M2(1)=-1.00
         M2(2)=-0.800
         M2(3)=-0.600
         DO 3 I=2, 6
            3 M2(I+1)=M2(I)+0.100
               M2(8)=-0.1200
               DO 4 I=8, 32
                  4 M2(I+1)=M2(I)+0.0100
                     M2(33)=0.200
                     DO 5 I=33, 36
                        5 M2(I+1)=M2(I)+0.100
                           M2(38)=0.800
                           M2(39)=1.00
                           M1=20
                           M2=39
                           DO 6 I=1, M1
                              6 M(I,J)=.TRUE.
                           DO 7 I=1, 12
                              7 DO 8 J=9, 31
                                 8  M(I,J)=I-1
                                 M=20-J
                                 IR=IRx(SORT(AI+2*J+2))
                                 IF(IR.LE.10) Y0(I,J)=1.00
                                 IF(IR.EQ.11) Y0(I,J)=0.500
                                 CONTINUE
                                 NC=(N*(N+7))2
                                 KK=-6
                                 DO 30 I=1, K
                                    30  KK=KK+6
                                    X(KK+1)=1.00
                                    X(KK+2)=1.00
                                    X(KK+3)=1.00
                                    X(KK+4)=1.00
                                    X(KK+5)=1.00
                                  X(KK+6)=1.00
CALL TRC2(K,W1,M1,42,X,LIWI,W1,2,4C,HG,A3)

Calculating response on a uniform mesh for plotting.

M1=21
M2=41
W1(1)=0.00
W2(1)=-1.00
EM1=1-1
EM2=M2-1
WINCP1=1.00/E**M1
WINCP2=2.00/E**M2
N=M1-2
DO 27 I=1,N
27 W1(I+1)=W1(I)+WINCP1
N=M2-2
DC 28 I=1,N
28 W2(I+1)=W2(I)+WINCP2
W1(M1)=1.00
W2(M2)=1.00
DO 12 I=1,M1
DO 12 J=1,M2
M(I,J)=.TRUE.
12 Y0(I,J)=0.00
DO 26 I=1,M1
DO 26 J=1,M2
AI=I-1
AJ=21-J
IF(AI**2+AJ**2)<2)
IF(IF*LE.2) YR(I,J)=1.00
26 CONTINUE
N=6
CALL PLOT8(W1,42,X,YR,1,W2,K)
STOP
END
SUBROUTINE TRDF2(K, M1, M2, X, LIMIT, W1, W2, HC, NC, A, B)
THIS IS A SUBROUTINE FOR DESIGNING STABLE TWO-DIMENSIONAL
RECURSIVE FILTERS, FORMED OF SECOND ORDER CASCADES.

EXPLANATION OF PARAMETERS:
K IS THE NUMBER OF SECOND-ORDER CASCADES.
X IS A VECTOR CONTAINING AN INITIAL GUESS OF THE PARAMETERS,
DEFINED IN THE THESIS, IT IS OF DIMENSION N=6*K. THE PARAMETERS ARE
STORED IN X, SUCH THAT, THE FIRST 6 FOR THE FIRST CASCADE, THE
SECOND 6 FOR THE SECOND CASCADE, AND SO ON. ON RETURN IT CONTAINS
THE OPTIMIZED PARAMETERS.
LIMIT IS THE MAXIMUM NUMBER OF ITERATIONS AS DEFINED IN THE
FLETCHER-POWELL ALGORITHM.
W1 IS THE VERTICAL FREQUENCY AXIS, OF DIMENSION M1, DEFINED FROM
ZERO TO THE NYQUIST RATE.
W2 IS THE HORIZONTAL FREQUENCY AXIS, OF DIMENSION M2, DEFINED
FROM -1 TO +1 THE NYQUIST RATE.
HC IS A DOUBLE PRECISION VECTOR REQUIRED BY THE DFMFP SUBROUTINE,
AND IS OF DIMENSION NC=6*(M1+M2)/2.
A AND B ARE VECTORS CONTAINING THE OPTIMIZED PARAMETERS
OF THE RECURSIVE FILTER, ON RETURN, EACH IS OF DIMENSION 3*K.
DIMENSION OF OTHER VARIABLES:
H0, Y(M1), C(M1, M2), Z1(M1), Z2(M2), YD(M1, M2).

COMMON/AR6/YD
WHERE YD IS THE REQUIRED SPATIAL MAGNITUDE-FREQUENCY RESPONSE,
AND IS TO BE GIVEN OVER THE LOWER HALF SPATIAL FREQUENCY DOMAIN.
COMMON/DT/M
M IS A LOGICAL MESH. IF ASSIGNED TRUE, AT A LOCATION, THEN
THIS LOCATION IS INCLUDED IN THE OPTIMIZATION, AND VICE-VERSA.
M IS OF DIMENSION M(M1, M2).

SUBROUTINES REQUIRED:
DFMFP_IBM VERSION SUBROUTINE, FOR FLETCHER-POWELL ALGORITHM.
SUBROUTINE TRDF2 REQUIRED BY DFMFP, FOR CALCULATING THE
GRADIENTS.

1  SUBROUTINE TRDF2(K, M1, M2, X, LIMIT, W1, W2, HC, NC, A, B)
2  COMPLEX*16 Z1, Z2, CP, CDXP, DCMPLX
3  REAL*8 X, A(0:Q, YD), C(0, PY), W1, W2, DREAL, HA, B, S(1, S2, s3
4  LOGICAL*1 M(31), 611)
5  DIMENSION YD(M1, M2), C(M1, M2), Z1(M1), Z2(M2), X(12),
6  W1(31), W2(61), A(8), B(8), NQ(12)
7  EXTERNAL FUNCT
8  COMMON/R1/Z1, Z2
9  COMMON/R2/C, AD
10  COMMON/R6/YD
11  COMMON/B3/KOUNT, K1, M3, M4
12  COMMON/B3/IND
13  COMMON/B7/M
14  DATA PY/3.14159265358970/  
16  K=1
17  M1=1
18  M2=1
19  N=6*K
20  CP=DCMPLX(0.00, W1(1)*PY)
21  Z1(1)=CDXP(-CP)
22  CP=DCMPLX(0.00, W2(1)*PY)
23  Z1(1)=CDXP(-CP)
24  CP=DCMPLX(0.00, W2(1)*PY)
3  Z(1) = CD*EXP(-CP)
   KOUNT = 0
   EPS = 0.0
   CALL DFMFP(FUNCT, R, X, Q, DO, FST, EPS, LIMIT, IER, HC)
   WRITE(6,4) KOUNT
4  FORMAT(1H1, 5X, 'NUMBER OF FUNCTION EVALUATION=', I16)
   WRITE(6,30) IER
30 FORMAT(5X, 'IER=', I13)
   IF (IER) 5, 6, 7
5  WRITE(6,5)
   FORMAT(5X, 'CONVERGENCE NOT ACHIEVED, LIMIT OF ITERATIONS')
   GO TO 7
6  WRITE(6,9)
   FORMAT(5X, 'CONVERGENCE NOT ACHIEVED, LIMIT OF ITERATIONS')
    WRITE(6,10) Q
7  FORMAT(5X, 'SUM OF ERRORS-SQUARED BETWEEN DESIGNED AND DESIRED MAGNITUDE SPECIFICATIONS=', D12.5)
   WRITE(6,11)
11 FORMAT(2X, 'FINAL VALUE OF PARAMETERS')
   WRITE(6,12) (I, XI(I), I=1, N)
12 FORMAT(5X, 'X(', I4, ',') = ', D12.5)
   WRITE(6,18) AN
18 FORMAT(5X, 'K0 = ', D12.5)
   WRITE(6,19)
19 FORMAT(5X, 'THE DESIGNED TWO-DIMENSIONAL RECURSIVE DIGITAL FILTER IS GIVEN BY', //5X, 'H(Z1, Z2) = ')
   KL = 1
   KK = 4
   KK1 = 1
   DO 14 I = 1, K
3  KK = KK + 4
   KK1 = KK1 + 6
   S1 = X(KK1 + 1) ** 2
   S2 = X(KK1 + 2) ** 2
   S3 = X(KK1 + 3) ** 2
   A(KK + 1) = -1.0D0 - S1 + S2 + S3
   A(KK + 2) = 1.00 - S1 - S2 + S3
   A(KK + 3) = -1.0D0 - S1 - S2 + S3
   A(KK + 4) = 1.0D0 + S1 + S2 + S3
   S1 = X(KK1 + 4) ** 2
   S2 = X(KK1 + 5) ** 2
   S3 = X(KK1 + 6) ** 2
   B(KK + 1) = -1.0D0 - S1 + S2 + S3
   B(KK + 2) = 1.00 - S1 - S2 - S3
   B(KK + 3) = -1.0D0 - S1 - S2 - S3
   B(KK + 4) = 1.00 + S1 + S2 - S3
   WRITE(6,15) (A(KK + L), L = 1, 4)
15 FORMAT(5X, 'Z2*(Z1**2, D12.5, **, D12.5) + Z1**2, D12.5, **, D12.5)
   WRITE(6,16)
16 FORMAT(5X, '---------')
   WRITE(6,15) (B(KK + L), L = 1, 4)
   IF (K.EQ.1) GO TO 14
   IF (K.KL.EQ.0) GO TO 14
   KL = KL - 1
   WRITE(6,17)
17 FORMAT(//, 5X, 'TIMES')
14 CONTINUE
15 WRITE(6,71)
71 FORMAT(111,60X,'MESH')
17 DO 72 J=1,M2
72 WRITE(6,73) (M(I,J),I=1,M1)
73 FORMAT(2X,31(L4))
18 WRITE(6,22)
22 FORMAT(141,53X,'DESIGNED MAGNITUDE SPECIFICATIONS')
20 DO 70 J=1,M2
21 WRITE(6,23) (YD(L,J),L=1,M1)
23 FORMAT(141,53X,'DESIGNED FILTER MAGNITUDE RESPONSE')
25 DO 75 I=1,M1
26 DO 75 J=1,M2
27 IF(.NOT.M(I,J)) GO TO 77
28 CONTINUE
29 GO TO 76
70 M(I,J)=HIT'M(I,J)
30 IND=0
31 CALL FUNCT(N,X,Q,DO)
33 DO 76 J=1,M2
34 DO 74 L=1,M1
35 Y(L)=SNGL(C(L,J)*AN)
37 WRITE(6,21) (Y(L),L=1,M1)
39 RETURN
41 END
SUBROUTINE FUNCT(N,X,Q,DO)
THIS SUBROUTINE CALCULATES THE SQUARE-ERROR CRITERION, Q, AND ITS
GRADIENTS WITH RESPECT TO THE PARAMETERS, FOR SUBROUTINE TRDF2.
IF THIS SUBROUTINE IS USED WITH DFMFP, THEN THE COMMON BLOCK
COMMON/B3/IND
SHOULD TRANSFER IND=1. IF USED ONLY TO CALCULATE THE TRANSFER
FUNCTION THEN SET IND=0.
SUBROUTINE FUNCT(N,X,Q,DO)
COMPLEX*16 Z1,Z2,P1,P2,P3,DCMPLX,PN,PD
REAL*8 AN,YD,Y,X,Q,DO,SUM1,SUM2,F,CNARS,F,S1,S2,S3,S4,S5,S6,PL,
VALUES
DIMENSION Z1(31),Z2(31),PN(2,31,61),RD(2,31,61),X(12),Y(16),
XE(31,61),DO(12),F(31,61),YN(31,61)
LOGICAL*K=1 M(31,61)
COMMON/P1/Z1,Z2
COMMON/P2/AN,YD
COMMON/P3/RD/KOUNT,K,M1,M2
COMMON/P4/IND
COMMON/P5/M
KK1=-8
KK=-6
CC 8 I=1,K
KK=KK+6
KK1=KK1+8
S1=X(KK+1)*2
S2=X(KK+2)*2
S3=X(KK+3)*2
Y(KK1+1)=L.DO-S1+S2+S3
Y(KK1+2)=L.DO-S1-S2+S3
Y(KK1+3)=L.DO+S1-S2+S3
Y(KK1+4)=L.DO+S1+S2+S3
S1=X(KK+4)*2
S2=X(KK+5)*2
S3=X(KK+6)*2
Y(KK1+5)=L.DO-S1+S2+S3
Y(KK1+6)=L.DO-S1-S2+S3
Y(KK1+7)=L.DO+S1-S2+S3
Y(KK1+8)=L.DO+S1+S2+S3
Q=0.70
SUM1=Q,DO
SUM2=Q,DO
DO 2 L=1,M2
DO 2 J=1,M1
IF(J,NOT.M(J,L)) GO TO 2
P=DCMPLX(L.DO,0.DO)
KK1=-8
DO 1 I=1,K
KK1=KK1+8
1 P=SN*M1(J,L)/P*2*(J,L)
C(J,L)=CNARS(P)
IF(IND.EQ.0) GO TO 2
SUM1=SUM1+C(J,L)*YN(J,L)
SUM2=SUM2+C(J,L)*2
2 CONTINUE
IF(IND.EQ.0) RETURN
050  AO=SUM1/SUM2.
051  CC 3 L=1,M2
052  CC 3 J=1,M1
053  IF(.NOT.M(J,L)) GO TO 3
054  E(J,L)=C(J)+D(J,L)
055  Q=0+E(J,L)**2
056  CONTINUE
057    KK=-6
058    CC 4 I=1,K
059    KK=KK+6
060    K1=KK+1
061    K2=KK+2
062    K3=KK+3
063    K4=KK+4
064    K5=KK+5
065    K6=KK+6
066    S1=0.000
067    S2=0.000
068    S3=0.000
069    S4=0.000
070    S5=0.000
071    S6=0.000
072    CC 5 L=1,M2
073    CC 5 J=1,M1
074    IF(.NOT.M(J,L)) GO TO 5
075    PL=E(J,L)*C(J,L)
076    P1=(1.00-Z2(L,J))*(1.00+Z1(J))
077    P2=(1.00-Z1(J))*(1.00-Z2(L,J))
078    P3=(1.00+Z1(J))*(1.00+Z2(L,J))
079    IF(.NOT.DABS(RH1,J,L)*LF+1.0-50) GO TO 9
080    P=1.00/RN(J,L)
081    S1=S1+PL*DREAL(P1*P)
082    S2=S2+PL*DREAL(P2*P)
083    S3=S3+PL*DREAL(P3*P)
084    9 P=1.00/RD(J,L)
085    S4=S4+PL*DREAL(P1*P)
086    S5=S5+PL*DREAL(P2*P)
087    S6=S6+PL*DREAL(P3*P)
088    CONTINUE
089    P=4.00*AD
090    DQ(K1)=S1*PL*X(K1)
091    DQ(K2)=S2*PL*X(K2)
092    DQ(K3)=S3*PL*X(K3)
093    DQ(K4)=S4*PL*X(K4)
094    DQ(K5)=S5*PL*X(K5)
095    4 DQ(K6)=S6*PL*X(K6)
096    KCOUNT=KCUNT+1
097    RETURN
098    END
SUBROUTINE PLOT8(K1,42,X,Y,v1,v2,K)

THIS IS A SUBROUTINE FOR PLOTTING, IN A PICTURE FORM, THE DESIRED
AXIS-DESIRED SPATIAL MAGNETIZATION-FREQUENCY RESONANCE ON A UNIFORM MESH
SUBROUTINES REQUIRED :

1) NORMI : TO NORMALIZE THE DATA FOR PLOTTING.
2) PICALC : FOR THE PLOTTING PART.

COMPLEX*16 2L(31),2L(61),CP,COMPLX,CCXP
REAL*4 Y(I(31),I(61),I(31),I(61),I(12),I(12),I(31,61,31,61,31,61))
LOGICAL I(31,61)
DATA 2Y/3.1415265358979707/,
COMMON/R1/1,22,
COMMON/R2/C,10,
COMMON/R5/A(44,41),A(21,21),
COMMON/R4/IND,
COMMON/R3/OLCNT,K1,N3,M4,
COMMON/R7/M

M3=M1
M4=M2
K1=K
N=M
CC 2 I=1,M1
CP=COMPL(0.0D0,-1(I)*PY)
2 ZL(I)=CCXP(-CP)
DO 3 J=1,M2
CP=COMPL(0.0D0,-2(J)*PY)
3 ZZ(J)=CCXP(-CP)
IND=CC
DJ=0 J=1,M2
AA(1+1,I)=0.
40 AA(1+2,I)=0.
DO 1 I=1,M1
DO 1 J=1,M2
1 B3(I,J)=SNGL(Y(I,J))
CALL NORMI(Y1,Y2)
DO 35 I=1,M1
DO 35 J=1,M2
35 AA(I,J)=B3(I,J)
CALL FUNCT(N,X,*,*,*,*)
WRITE(6,45) J
45 FOPRT(10X,'=',1,912.5)
DO 25 I=1,M1
DO 25 J=1,M2
25 B3(I,J)=SNGL(AO*C(I,J))
CALL NORMI(Y1,Y2)
DO 36 I=1,M1
DO 36 J=1,M2
36 AA(I+1+2,J)=B3(I,J)
CALL PICOUT(2*PY+2,Y2,2.5)
RETURN
END
SUBROUTINE NORMIZ(I1, M2)
COMMON/ES/AA(44,41), BI(21*M1).
!NORMALISING DATA.
C
      YMIX=BI(1,1)
      YMIN=BI(1,1)
      DO 251 I=1,M1
      DC 251 J=1,M2
      YMIX=MAMAXI(YMIX, BI(I,J))
      251 YMIN=MAMINI(YMIN, BI(I,J))
      RANGE=YMAX-YMIN
      IF(RANGE.EQ.0.0) RETURN
      DO 253 I=1,M1
      DO 253 J=1,M2
      253 BI(I,J)=(BI(I,J)-YMIN)/RANGE
      RETURN
      END
SUBROUTINE PICOUT(M1,M2,RATIO)
REAL*8 PLINE(16,8),BLANKS/
DIMENSION ALINE(128)
DIMENSION LINE(128,8)
LOGICAL LINE,IA,IR,IC,IH,IM,IO,IX,IJ,IZ,IBLK,IMNS,IEOL,
13K1,N1ON,IPLS,IOIE,ISTP
EQUIVALENCE (LINE,LINE)
DATA IA,IR,IC,IH,IM,IO,IX,IZ,IBLK,IMNS,IEOL,IPKS,INVE,
13PLS,IOIE,ISTP,JX
2/*",",",",","",",","",",","",","",","",","",","",",","",","",","",",","",",","",",","",",","",",","",",","",",","",",","",",","",",","",",","",",","",",","",",","",",","",",","",",","",",","",",","",",",",",",",",",",",",",",",",",",",",",",",",",",",",",",",",",",",",",",",",",",",",",",",",",",",",",",",",",",",",",",",",",",",",",",",",",",",",",",",",",",",",",",",",",",",",",",",",",",",",",",",",",",",",",",",",",",",",",",",",",",",",",",",",",",",",",",",",",",",",",",",",",",",",",",",",",",",",",",",",",",",",",",",",",",",",",",",",",",",",",",",",",",",",",",",",",",",",",",",",",",",",",",",",",",",",",",",",",",",",",",",",",",",",",",",",",",",",",",",",",",",",",",",",",",",",",",",",",",",",",",",",",",",",",",",",",",",",",",",",",",",",",",",",",",",",",",",",",",",",",",",",",",",",",",",",",",",",",",",",",",",",",",",",",",",",",",",",",",",",",",",",",",",",",",",",",",",",",",",",",",",",",",",",",",",",",",",",",",",",",",",",",",",",",",",",",",",",",",",",",",",",",",",",",",",",",",",",",",",",",",",",",",",",",",",",",",",",",",",",",",",",",",",",",",",",",",",",",",",",",",",",",",",",",",",",",",",",",",",",",",",",",",",",",",",",",",",",",",",",",",",",",",",",",",",",",",",",",",",",",",",",",",",",",",",",",",",",",",",",",",",",",",",",",",",",",",",",",",",",",",",",",",",",",",",",",",",",",",",",",",",",",",",",",",",",",",",",",",",",",",",",",",",",",",",",",",",",",",",",",",",",",",",",",",",",",",",",",",",",",",",",",",",",",",",",",",",",",",",",",",","
IF (DVAL1.LT.0.546) GO TO 81
IF (DVAL1.LT.0.582) GO TO 82
IF (DVAL1.LT.0.619) GO TO 83
IF (LMAX.LT.3) LMAX=3
IF (DVAL1.LT.0.655) GO TO 84
IF (DVAL1.LT.0.729) GO TO 85
IF (LMAX.LT.5) LMAX=5
IF (DVAL1.LT.0.820) GO TO 87
IF (DVAL1.LT.0.874) GO TO 86
IF (LMAX.LT.6) LMAX=6
IF (DVAL1.LT.0.910) GO TO 88
IF (DVAL1.LT.0.946) GO TO 89
IF (LMAX.LT.7) LMAX=7
IF (DVAL1.LT.0.982) GO TO 90
IF (LMAX.LT.8) LMAX=8
GJ TO 91
72 LINE(J,1)=IMNS
GJ TO 100
73 LINE(J,1)=IFOL
GJ TO 100
74 LINE(J,1)=IPLS
GJ TO 100
75 LINE(J,1)=IBKT
GJ TO 100
76 LINE(J,1)=IONE
GJ TO 100
77 LINE(J,1)=IZ
GJ TO 100
78 LINE(J,1)=IX
GJ TO 100
79 LINE(J,1)=IA
GJ TO 100
80 LINE(J,1)=IM
GJ TO 100
81 LINE(J,2)=IMNS
GJ TO 96
82 LINE(J,2)=IFOL
GJ TO 96
83 LINE(J,5)=IFOL
GJ TO 96
84 LINE(J,4)=ISTP
85 LINE(J,3)=IOKE
86 LINE(J,2)=IPLS
GJ TO 96
87 LINE(J,5)=IMNS
GJ TO 96
88 LINE(J,6)=PC
GJ TO 92
89 LINE(J,8)=IA
90 LINE(J,7)=IV
91 LINE(J,6)=IB
92 LINE(J,5)=IH
93 LINE(J,4)=ISTP
94 LINE(J,3)=IOKE
95 LINE(J,2)=IX
96 LINE(J,1)=IO
100 CONTINUE
WRITE(6,101) (PLINE(J,1),J=1,N)MB
IF (LMAX.LT.2) GO TO 200
DJ 110 K=2, LMAX
WRITE(6, 102) (RLINE(J, K), J=1,LMAX)
110 CONTINUE
200 CONTINUE
101 FORMAT('* ',16A8)
102 FORMAT('*+',16A8)
RETURN
END
VITA AUCTORIS

1945  Born on the 24th of March in Alexandria, Egypt.

1963  Completed high school at Victoria College, Alexandria, Egypt.

1968  Graduated from the University of Alexandria with a B.Sc. (Honours) degree in Electrical Engineering.

1970  (Sept.) Registered as a candidate for M.A.Sc. degree in Electrical Engineering, at the University of Windsor, Windsor, Ontario, Canada.

1972  Graduated from the University of Windsor, with the degree of M.A.Sc. in Electrical Engineering.

1974  Candidate for the degree of Ph.D., Electrical Engineering, University of Windsor, Windsor, Ontario, Canada.