

General Disclaimer

One or more of the Following Statements may affect this Document

- This document has been reproduced from the best copy furnished by the organizational source. It is being released in the interest of making available as much information as possible.
- This document may contain data, which exceeds the sheet parameters. It was furnished in this condition by the organizational source and is the best copy available.
- This document may contain tone-on-tone or color graphs, charts and/or pictures, which have been reproduced in black and white.
- This document is paginated as submitted by the original source.
- Portions of this document are not fully legible due to the historical nature of some of the material. However, it is the best reproduction available from the original submission.

$$s \leftarrow \frac{1-z^{-1}}{1+z^{-1}} \quad s \leftarrow$$

$$X_D(\omega) \leftrightarrow x_D(t) \quad X_D$$

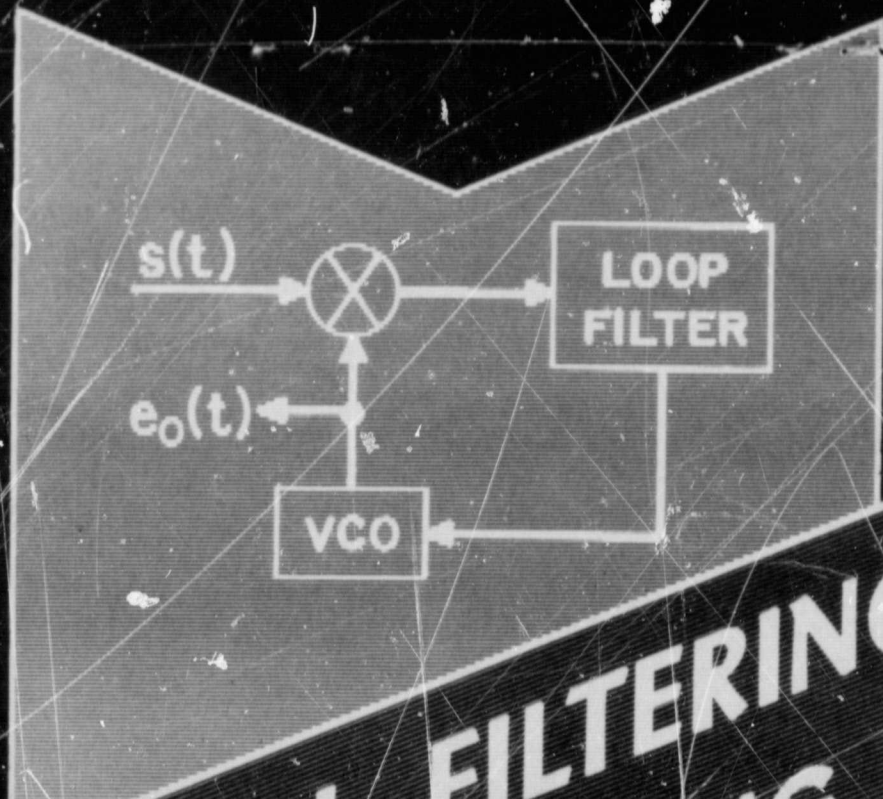
$$x_k \rightarrow H_Z(z) \rightarrow y_k \quad x$$

$$H_Z(\omega) = \sum_{k=-\infty}^{\infty} H(\omega + k\omega_s)$$

$$y_i = -\sum_{n=1}^N b_n y_{i-n} + \sum_{m=0}^M a_m x_{i-m}$$

$$\frac{1}{N} \sum_{n=0}^{N-1} A_n B_n W^{jn} = \sum_{k=0}^{N-1} A_k B_{j-k}$$

$$\sigma_y^2 = \frac{1}{2}$$



DIGITAL FILTERING AND PROCESSING BY TRANSFORM TECHNIQUES VOLUME II

by
EDGAR H. GERMAN JR.
and
KNOWLES LITTLE

N 69-14493	
(ACCESSION NUMBER)	(THRU)
149	1
(PAGES)	(CODE)
CR-98191	08
(NASA CR OR TMX OR AD NUMBER)	(CATEGORY)



DIGITAL FILTERING AND PROCESSING
by
TRANSFORM TECHNIQUES,

it is VOLUME II °

A DIGITAL FILTER GENERATION
and
APPLICATION ROUTINES

FINAL REPORT

30 June 1968

CONTRACT NAS3-20308

NON-LINEAR AND LINEAR DIGITAL FILTERING TECHNIQUES

Prepared by
E. H. GERMAN JR.
K. LITTLE

for

GEORGE C. MARSHALL
SPACE FLIGHT CENTER
HUNTSVILLE, ALABAMA

THE BENDIX CORPORATION
COMMUNICATIONS DIVISION
BALTIMORE, MARYLAND 21204

For Information

contact

**D. H. MacDONALD
Government Marketing Department
Phone 1-823-2200, Ext. 626**

or

**E. H. GERMAN JR.
Advanced Research Department
Phone 1-833-2200 Ext. 715**

**THE BENDIX CORPORATION
COMMUNICATIONS DIVISION
BALTIMORE, MARYLAND 21204**

PREFACE

The general purpose computer routines described in this document were developed to implement digital filtering techniques for the Computation Laboratory, George C. Marshall Space Flight Center under Contract No. NAS8-20308.

The routines included both coefficient generation and applications programs.

The purpose of this Volume (II) is to instruct in the correct use of these routines without giving a detailed explanation of the underlying mathematics in spite of the fact that many of the routines implement fairly sophisticated mathematical operations. It is, therefore, frequently necessary to refer to Volume I of the final report on Digital Filtering and Processing By Transform Techniques to fully understand the operation of the routines.

TABLE OF CONTENTS

	Page
1. GUIDE TO DIGITAL FILTER ROUTINES	1-1
2. RECURSIVE z TRANSFORMATION/DATA RECONSTRUCTION ROUTINES FOR TRANSFER FUNCTION APPROXIMATION	2-1
2.1 z TRANSFORM PROGRAM SEQUENCING	2-3
2.1.1 BLAST	2-6
2.1.2 SEQUN1 and SEQUN2	2-11
2.2 BANDLIMITING THE TRANSFER FUNCTION	2-17
2.2.1 Selection of Reconstruction Technique	2-20
2.2.2 RECONS	2-22
2.2.3 FINDB	2-28
2.2.4 BRECO	2-34
2.3 PARTIAL FRACTION EXPANSION PROGRAM - PARTLF	2-37
2.4 TRANSFORMATION PROGRAM	
2.4.1 CLEANZ	2-50
2.4.2 ZTRAN	2-55
2.4.3 JPOLE	2-61
2.4.4 DZPOLE	2-65
2.4.5 DENOM	2-69
2.4.6 LEVEL	2-72
2.5 POLYNOMIAL MANIPULATION ROUTINES	
2.5.1 COMBIN	2-76
2.5.2 FIX	2-79
2.5.3 POLYAD	2-81
2.5.4 POLYM	2-85
2.5.5 ADJNUM	2-88
2.6 FREQUENCY RESPONSE EVALUATION ROUTINES	
2.6.1 REFORM	2-90
2.6.2 RESP	2-93
2.6.3 RECRSP	2-97
2.6.4 TRANS	2-100
2.6.5 PHASE	2-103
2.6.6 EVPOLY	2-105
3. BILINEAR z TRANSFORM PROGRAM - BIZTRN	3-1
4. NON-RECURSIVE DIGITAL FILTER GENERATION ROUTINES	4-1
4.1 TRANWG	4-4
4.2 WGTCP	4-7
4.3 SMOOTH	4-10

	Page
5. APPLICATIONS OF DIGITAL FILTERS	5-1
5.1 RECF and RECF2	5-3
5.2 SRECF	5-7

1. GUIDE TO DIGITAL FILTER ROUTINES: A GENERAL INTRODUCTION

The following is intended as an introduction for a prospective user of the digital filter generation routines discussed in this document. The routines are designed to run on a computer with a Fortran IV Compiler.

There are two basic types of routines presented here: (1) Synthesis or filter coefficient generation routines, and (2) application or filtering routines. The synthesis routines aid in design of the digital filters while the applications routines use the filter weights computed by the synthesis routines to perform data filtering. The synthesis routines are treated in some detail due to their complexity while the applications routines are relatively simple subroutines. Some fundamental applications routines have been developed which will be useful in many problems. Statistical filtering routines have not been included.

Synthesis routines generally fall into one of two categories depending upon the application. They are: (1) Fitting of an analogue transfer function over a frequency region of interest, and (2) processing data in some general way, e.g. smoothing. Selection of particular synthesis routines depends upon which of these two problems is being solved. In both cases the expression implementing the filter is the same, only the coefficients are changed. Input to the filter is a sequence of numbers which may, for convenience, be thought of as representing samples of a continuous function of time. The digital filter operates on this input number sequence to produce an output number sequence.

When the problem is numerical integration or simulation of an analogue transfer function then the digital filter output number sequence would preferably be the same as the output samples of the analogue transfer function whose input was a continuous function. This is illustrated in Figure 1-1

Exact simulation is achieved when output sequence II is the same as output sequence I. Namely the numerical (digital) solution and the analogue system output samples are identical. Generally this is impossible; a delay is usually necessary.

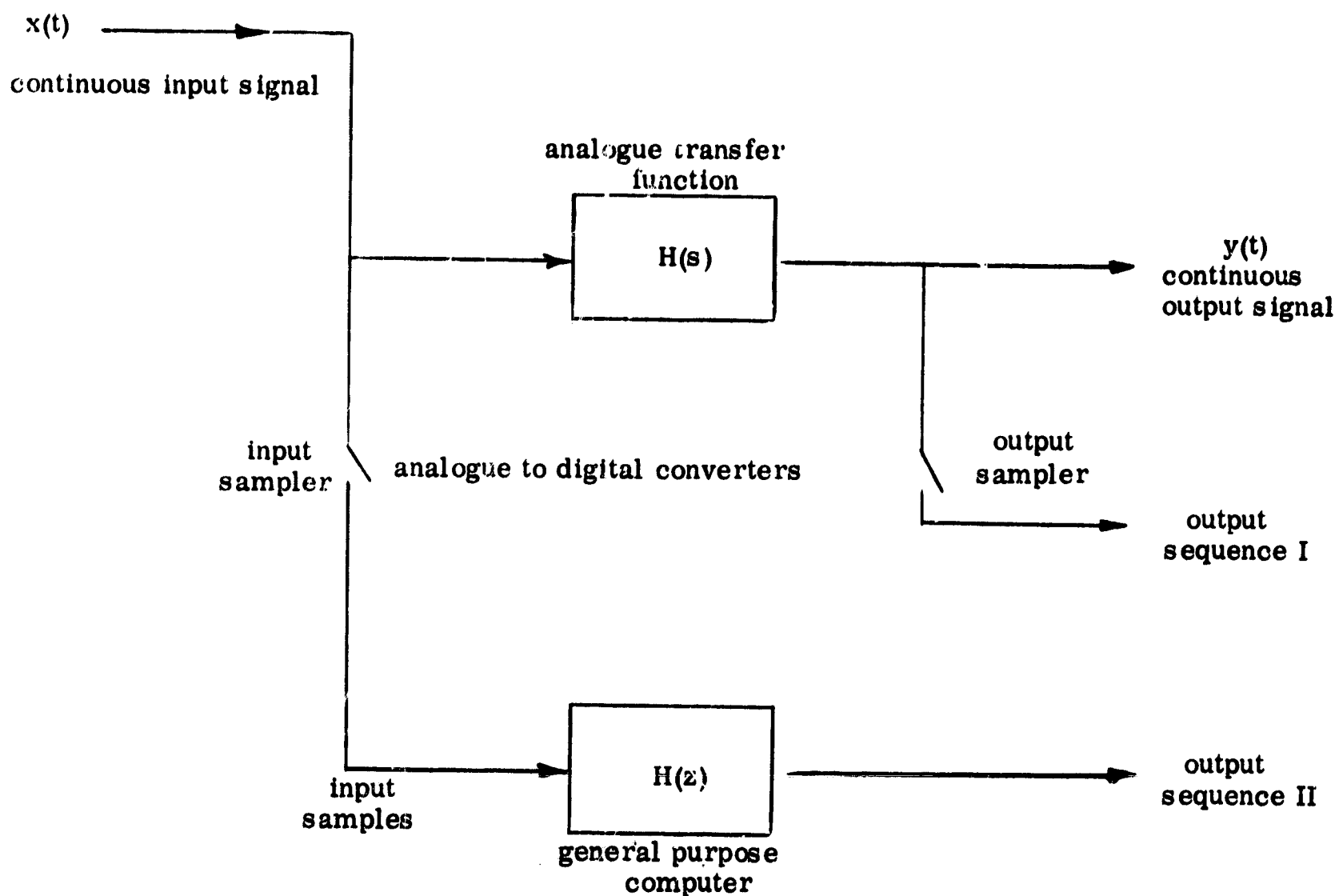


Figure 1-1 Digital Filtering on a General Purpose Computer

On the other hand, if the problem is to process some number sequence in a general way, then strict correspondence to an equivalent analogue transfer function is not required. An important example of this type of processing is the emphasis or removal of a particular band of frequencies in a wide signal spectrum. The routines which develop this kind of filter are much less complicated, are easier to use, and occupy much less storage than do the more general synthesis routines. These are the bilinear z transformation routines.

The transfer function of a practical linear, constant coefficient digital filter may be written as,

$$H_z(z) = \frac{Y(z)}{X(z)} = \frac{\sum_{k=0}^M a_k z^{-k}}{\sum_{k=0}^N b_k z^{-k}} \quad (1-1)$$

where:

$Y(z)$ is the output sequence z transform

$X(z)$ is the input sequence z transform

The "time domain" input-output relation corresponding to the transfer function above is

$$y_k = \frac{1}{b_0} \left[\sum_{m=0}^M a_m x_{k-m} - \sum_{n=1}^N b_n y_{k-n} \right] \quad (1-2)$$

where:

x_n = nth member of the input sequence

y_n = nth member of the output sequence

This is the form of the digital filter used by the applications routines. That is to say (1-2) is the calculation performed to produce the kth output of a digital filter.

The simulation problem requires further discussion at this point because there are two distinct techniques available for simulation synthesis. Use of these routines result in either a recursive or a non-recursive filter. The decision whether to select a recursive or non-recursive digital filter synthesis of an analogue transfer function depends upon the spectrum bandwidth being simulated and the nature of the transfer function itself. The user will always select the method which

realizes the analogue transfer function with sufficient accuracy over the frequency range of interest. A commonly used criteria for determining simulation accuracy are the amplitude and phase versus frequency responses of the digital filter. By comparing the digital filter responses with the responses of the analogue transfer function one can evaluate the success of a particular technique when applied to a specific transfer function. If both the recursive and the non-recursive techniques produce a digital filter which adequately realizes the transfer function over the frequency range of interest then the realization having the fewest number of filter coefficients should be selected in the interest of faster computation time.

The correct choice cannot always be made aprior ; only after both recursive and non-recursive realizations of a particular transfer function have been developed can the decision be confidently made. However, guidance in this selection can be based on past experience. It has been found that for approximately 1 % to 5 % error in the digital filter frequency response it is usually best to use the recursive filter realization if:

- (1) The approximation band is less than $1/3$ the sampling frequency
- or (2) The transfer function itself is "bandlimited" to $1/2$ the sampling frequency.

On the other hand, if the transfer function extends beyond $1/3$ the sampling frequency (it is not sufficiently bandlimited to half the sampling frequency) and it is desired to realize the transfer function to the maximum limit of $1/2$ the sampling frequency then it is usually necessary to use non-recursive realizations.

While certain theoretical considerations lead to initial information as to the expected approximation accuracy, the calculations necessary to finally judge the "aliasing" are generally as much labor as trying the various techniques. Of course, if the analogue transfer function is "obviously" bandlimited below half the sampling frequency, then recursive filtering by the direct z transform is preferable.

There is one further consideration in making the choice between recursive and non-recursive filter realizations. That consideration is the additional time delay introduced by digital filtering. The recursive filter often introduces less time delay than the non-recursive filter if a smaller approximation region is allowable.

Figure 1-2 illustrates the steps which must be taken to develop digital filters for transfer function simulation.

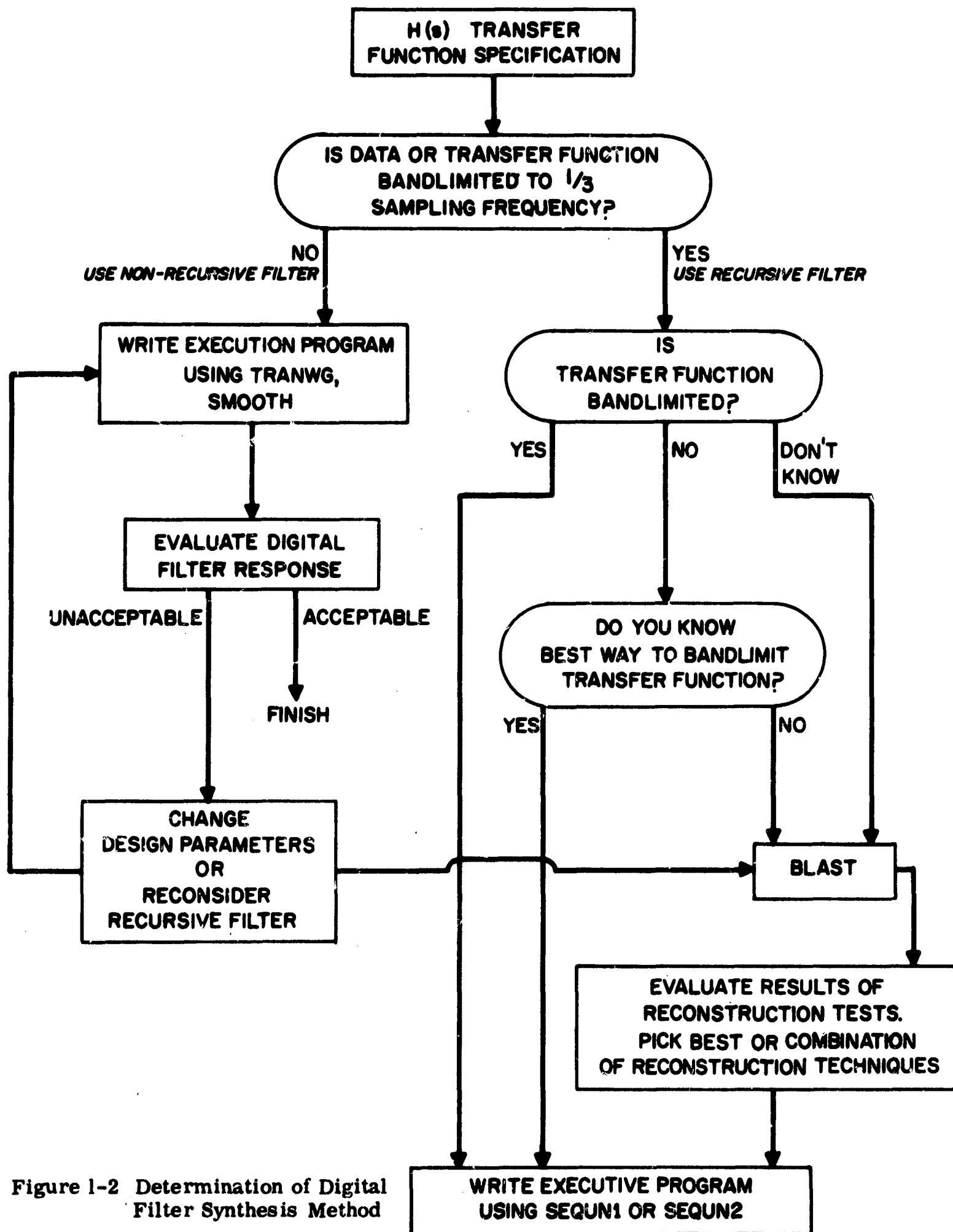


Figure 1-2 Determination of Digital Filter Synthesis Method

2. RECURSIVE z TRANSFORMATION/DATA RECONSTRUCTION ROUTINES FOR TRANSFER FUNCTION APPROXIMATION

Introduction

It is assumed that the transfer function to be simulated is available as the ratio of two polynomials in the Laplace transform variable "s". The denominator polynomial must be factored into its roots which may be either real or complex and may have any multiplicity. That is, the transfer function is of the form,

$$H(s) = \frac{P(s)}{Q(s)} = \frac{p_0 + p_1s + p_2s^2 + \dots + p_ns^n}{(s + R_1)^{m_1} (s + R_2)^{m_2} (s + R_3)^{m_3}} \quad (2-1)$$

where the

p's are the numerator polynomial coefficients (real coefficients)

R's are the denominator roots (complex roots)

m's are the denominator root multiplicities

The convention for polynomials used in all the following routines is that the lowest order coefficient is stored in the first location of its real or complex array. Assume a polynomial of the form

$$P = p_0 + p_1x + p_2x^2 + \dots + p_nx^n \quad (2-2)$$

The coefficients are stored in the linear array P as

$$P(1) = p_0$$

$$P(2) = p_1$$

$$P(3) = p_2$$

.

.

with p_j real or complex.

A second convention used in the following routines is that the length of a polynomial is described by its degree rather than by the number of coefficients. The polynomial above is of degree N and has $N+1$ coefficients, but N (its degree) is used to indicate its length.

The steps required for preparation of the transfer function for use by the z transformation routines are

- (1) define the numerator polynomial of $H(s)$ in a real linear array
- (2) define the roots in a complex linear array
- (3) define the multiplicities in an integer array
- (4) set some integer variable equal to the numerator degree
- (5) set another integer variable equal to the number of distinct roots.

Note that if a root is real it is necessary to specify the imaginary part zero.

2.1 z TRANSFORM PROGRAM SEQUENCING

A list of steps required to z transform a transfer function follows.

The method is fundamentally the same regardless of the reconstruction procedure:

- (1) Modify original transfer function $H(s)$ with desired reconstruction function $R(s)$. The result is a new function $H(s) R(s)$. Program used: RECONS
- (2) Perform the partial fraction expansion on the reconstruction modified transfer function, $H(s) R(s)$. Program used: PARTLF
- (3) z transform the resulting partial fraction expansion. Program used: CLEANZ
- (4) * Adjust the amplitude response at zero frequency of the resulting filter to be the same as the response of the original transfer function. This is particularly useful for those transfer functions which are zero at zero frequencies, e.g. differentiators. Program used: LEVEL
- (5)* Evaluate the amplitude and phase vs. frequency response of the z transformed function and compare its response with the original transfer function. Program name: RESP

Since this procedure does not vary with the reconstruction method there are only two sequencing programs needed. Their names are appropriately SEQUN1 and SEQUN2. They are identical except the SEQUN2 has no print-out and does not evaluate the frequency responses of the original function and the transformed function. Its sole purpose is to find the z transform filter coefficients and return them to the main program.

To use SEQUN1 or SEQUN2 specify the following;

- (1) The transfer function specifications
 - a. Root values - complex
 - b. Number of roots - integer
 - c. Multiplicity of each root-integer
 - d. Numerator polynomial - real coefficients
 - e. Degree of numerator polynomial - integer
 - f. Sampling period in seconds - real

* Optional

(2) Reconstruction Function Specification

- a. Is any reconstruction needed? If the original function is bandlimited to half the sampling frequency no reconstruction filter need be added. In this case set IREC=0, otherwise set IREC=1. However, if a reconstruction function is required to bandlimit the transfer function it is necessary to further specify.
- b. Type of reconstruction function to be used. If Butterworth reconstruction is used then specify:
 1. Approximation bandwidth ω_a
 2. Aliasing level or out-of-band attenuation, α
 3. Whether minimum pole or minimum time delay
Butterworth function is desired

Always consult the document describing SEQUN1 before attempting to use the routine since it has many variables in its CALL statement whose function should be fully understood.

Another program can be used which sequentially tries some of the reconstruction methods. For each reconstruction function applied to the input transfer function this program follows the sequence outlined above. The user can look at the resulting response tabulations and select the method which best fills the requirements of the simulation task.

The name of this program is BLAST. BLAST has no output to the main program. Its only output is printed paper and its only input is the transfer function. Use of this program should give the user an intuitive understanding of the relative merits of the various reconstruction methods. Once the best method is found to solve a given problem SEQUN1 or SEQUN2 should be used to generate the actual filter coefficients.

A flow chart of the various program and their subroutines follows in figure 2.1-1.

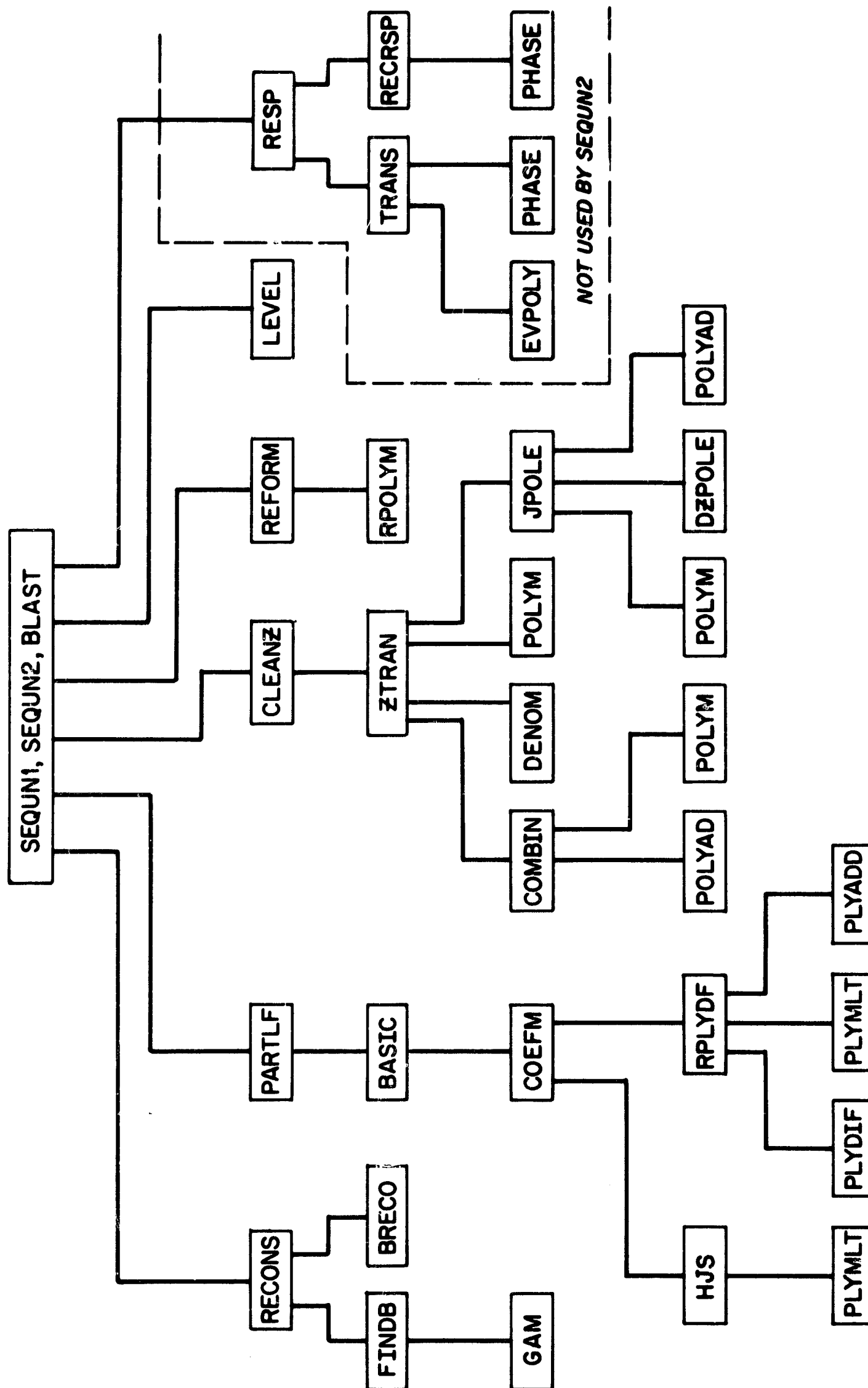


Figure 2.1-1
Recursive Filter Generation Routines

2.1.1 BLAST, A Routine Which Tries Various Reconstruction Techniques With z Transformation

This routine is a sequencing routine which tries all the commonly used reconstruction techniques used in z transforming transfer functions. For a given transfer function there are some reconstruction techniques which work better than others. It may not be obvious to the user which reconstruction filter to use on a given transfer function. The obvious solution is to try all of them and pick the technique which gives best results. For a discussion of the various reconstruction filters used here refer to Section 7, Vol I, Digital Filtering and Processing By Transform Techniques.

This routine (1) adds a reconstruction function $R(s)$ to the input transfer function $H(s)$, (2) performs the partial fraction expansion and z transformation on the resulting reconstructed function $H(s) \times R(s)$, (3) evaluates the frequency response of functions $[H(s) \times R(s)]$ and the original function $H(s)$, (4) sets the zero frequency level of the z transfer function equal to the original function $H(s)$ and then (5) re-evaluates the frequency response. BLAST goes through the above sequence for each of the following reconstruction front ends

- (1) No reconstruction, that is $R(s) = 1$
- (2) First order delayed sample and hold reconstruction
- (3) Zero order sample and hold reconstruction
- (4) First order extrapolation sample and hold reconstruction
- (5) Butterworth reconstruction with $\alpha = .1$, $\omega_a = 1/4 \omega_s$
- (6) Butterworth reconstruction with $\alpha = .01$, $\omega_a = 1/4 \omega_s$

where α is the attenuation of the aliasing tail in the approximation band ω_a
 ω_s is the sampling frequency in radians/sec.

This routine requires the entire z transformation package of subroutines. BLAST calls the following routines directly:

- (1) RECONS, the reconstruction filter generator
- (2) PARTLF, which finds the partial fraction expansion of $H(s) \cdot R(s)$
- (3) CLEANZ, which generates the modified impulse invariant z transform of $H(s) \cdot R(s)$
- (4) REFORM, which forms the denominator polynomial from its input roots.
- (5) RESP, which evaluates and prints out the frequency response of $[H(s) \cdot R(s)]$ and the frequency response of $H(s)$ side by side.
- (6) LEVEL, which sets the amplitude response of $[H(s) \cdot R(s)]$ equal to the amplitude response of $H(s)$ at zero frequency.

CALL STATEMENT:

CALL BLAST (A, M, B, N, MULT, T)

All of these call variables are inputs. The only output from this routine is printing.

1. A is a real array containing the numerator coefficients of the input transfer function $H(s)$.

$$A(1) = a_0$$

$$A(2) = a_1$$

.

.

.

$$A(m+1) = a_m$$

2. M is the integer degree of A.
3. B is a complex linear array of the Ndenominator roots. B may also be a real array with:

$$B(1) = \text{real part of root \#1}$$

$$B(2) = \text{imaginary part of root \#1}$$

$$B(3) = \text{real part of root \#2}$$

$$B(4) = \text{imaginary part of root \#2}$$

⋮

$$B(2N-1) = \text{real part of the Nth root}$$

$$B(2N) = \text{imaginary part of the Nth root}$$

4. N, the number of denominator roots - integer
5. MULT, an integer array containing the multiplicities of the N denominator roots.
6. T the sampling period, a real variable.

```

      SUBROUTINE BLAST(B,M,A,N,MULT,T)
C   FOR INFORMATION ON THESE ROUTINES CONTACT E.H. GERMAN
C   BENDIX COMMUNICATIONS DIV BALTO, MD 301-VA 3 2200 EX 715
C   IMPLICIT REAL*8(A-H,O-Z)
      DIMENSION A(1),B(1),MULT(1)
      DIMENSION P(50),Q(50),COEF(50),TRA(50),BOUT(50)
C   APPROXIMATION BAND ARBITRARILY SET TO 1/4 OF THE SAMPLING FREQUENCY
      WA=1.570796/T
      DO 20 J=1,6
      NSAV=N
      GO TO (12,13,13,13,14,15),J
12  IREC=0
      NSH=10
      GO TO 18
13  NSH=J-3
      IREC=1
      GO TO 18
14  NSH=10
      ALPHA=.1
      MINPOL=1
      GO TO 18
15  ALPHA=.01
18  CONTINUE
      CALL RECONS(NSH,A,MULT,N,      T,MINPOL,WA,ALPHA,B,M,TAU,WC,BOUT,
1MOUT,MUSAV,IREC)
      CALL PARTLF(BOUT,MOUT,A,MULT,N,COEF,CONST)
      WRITE(6,5) M,N
      NR=0
      DO 30 I=1,N
30  NR=NR+MULT(I)
      NR=2*NR
      ND2=2*N
      WRITE(6,4)(A(I),I=1,ND2)
      WRITE(6,21) CONST
      WRITE(6,6)(COEF(I),I=1,NR)
      CALL CLEANZ(COEF,A,N,MULT,T,MOUT,CONST,P,NP,Q,NQ,NSH)
      IF(NSH.LE.1) MULT(N)=MUSAV
      N=NSAV
      CALL REFORM(A,N,MULT,TRA,NTRA)
      NR1=NTRA+1
      WRITE(6,7)(TRA(I),I=1,NR1)
      WRITE(6,1)
      LAST = NP + 1
      WRITE(6,2)(P(I),I=1,LAST)
      WRITE(6,3)
      LAST = NQ + 1
      WRITE(6,2)(Q(I),I=1,LAST)
      CALL RESP(P,NP,Q,NQ,TRA,NTRA,B,M,T,TAU)
      WRITE(6,11) TAU
      WRITE(6,8)
      WRITE(6,31)
      GO TO (22,23,24,25,26,27),J
22  WRITE(6,32)
      GO TO 28
23  WRITE(6,33)
      GO TO 28

```

```

24 WRITE(6,34)
   GO TO 28
25 WRITE(6,35)
   GO TO 28
26 WRITE(6,36) WA
   GO TO 28
27 WRITE(6,37) WA
28 CONTINUE
   SLEV=B(1)/TRA(1)
   CALL LEVEL(SLEV,P,NP,Q,NQ)
   WRITE(6,10)
   WRITE(6,1)
   LAST = NP + 1
   WRITE(6,2)(P(I),I=1,LAST)
   WRITE(6,3)
   LAST = NQ + 1
   WRITE(6,2)(Q(I),I=1,LAST)
   CALL RESP(P,NP,Q,NQ,TRA,NTRA,B,M,T,TAU)
   WRITE(6,9)
20 CONTINUE
   RETURN
1  FORMAT(/,3X,11H OUTPUT PZ   )
2  FORMAT(/,2X,E15.8)
3  FORMAT(/,3X,11H OUTPUT QZ   )
4  FORMAT(/,3X,8H  ROOT=   ,E15.8,3X,E15.8 )
5  FORMAT(11H14X3H M=I4,5X3H N=I3)
6  FORMAT(/,3X,24H PARTIAL FRACTION COEF=   ,E15.8,3X,E15.8)
7  FORMAT(/,3X,27H REFORMED DENOMINATOR POLY=   ,E15.8)
8  FORMAT(/,34H RESPONSE BEFORE LEVEL CORRECTING   )
9  FORMAT(/,36H RESPONSE AFTER CORRECTION BY LEVEL   )
10 FORMAT(11H1)
11 FORMAT(/,3X,50H ADDITIONAL TIME DELAY DUE TO RECONSTRUCTION TAU=
   1   ,E15.8 )
21 FORMAT(/,10H CONSTANT= ,E15.8)
31 FORMAT(/,2X,29H TO OBTAIN ABOVE RESULTS SET   )
32 FORMAT(9X,48H IREC=0, THE OTHER VARIABLE VALUES DO NOT MATTER   )
33 FORMAT(9X,56H IREC=1, NSH=-1, THE OTHER VARIABLE VALUES DO NOT MAT
   1TER   )
34 FORMAT(9X,56H IREC=1, NSH=0, THE OTHER VARIABLE VALUES DO NOT MATT
   1ER   )
35 FORMAT(9X,56H IREC=1, NSH=1, THE OTHER VARIABLE VALUES DO NOT MATT
   1ER   )
36 FORMAT(9X,32H IREC=1, NSH.GE.2, MINPOL=1, WA= ,E15.8,/,20X,9H ALPH
   1A=.1   )
37 FORMAT(9X,32H IREC=1, NSH.GE.2, MINPOL=1, WA= ,E15.8,/,20X,10HALPH
   1A=.01   )
   END

```

2.1.2 SEQUN1 and SEQUN2, Routines To Develop The z Transform Of A Transfer Function H(s)

This routine is designed to call the various subprograms used when developing the z transform. SEQUN1 writes out various variables as they are developed or modified. While SEQUN2 performs the same computations as SEQUN 1, it does not print anything.

The inputs to both routines are the transfer function H(s) expressed as a numerator polynomial and denominator expressed by its roots and their multiplicities. Sampling period T is an input as are various other variables which serve as instructions to the routine as to which reconstruction techniques to use before attempting to z transform the input transfer function. The original transfer function H(s) is returned to the main program unmodified along with the transfer function

$$H_z(z) = \mathcal{Z}[H(s) \times R(s)] = \frac{P_z(z)}{Q_z(z)} = \frac{p_0 + p_1 z^{-1} + p_2 z^{-2} + \dots + p_m z^{-M}}{q_0 + q_1 z^{-1} + q_2 z^{-2} + \dots + q_N z^{-N}}$$

where:

$\mathcal{Z} [\]$ means "the z transform of", R(s) is the reconstruction function.

The operations performed consist of

1. Modify H(s) with the desired reconstruction function, R(s). At this point we have $H'(s) = H(s) \times R(s)$
2. Develop the partial fraction expansion of H(s)
3. Perform the modified impulse invariant z transformation on H(s)
4. Form the denominator polynomial of H(s) from its roots.
5. Set the response level of $H_z(z)$ equal to the response level of H(s) if requested.
6. In SEQUN1 the frequency response of $H_z(z)$ and H(s) are calculated and printed side by side for comparison. The routines called by

SEQUN1	SEQUN2
1. RECONS	1. RECONS
2. PARTLF	2. PARTLE
3. CLEANZ	3. CLEANZ
4. REFORM	4. REFORM
5. LEVEL	5. LEVEL
6. RESP	

These routines in turn require various subroutines. A maximum of 29 subroutines are required when using SEQUN1

CALL STATEMENTS:

CALL SEQUN1 (A, M, B, N, MULT, T, NSH, WA, ALPHA, MINPOL, LEV, P, NP, Q, NQ,)

or

CALL SEQUN2 (Same Variables)

where

1. A is a real linear array containing the coefficients of the numerator polynomial of $H(s)$. $A(1) = a_0$, $A(2) = a_1$, ..., $A(M+1) = a_m$
2. M is the integer degree of polynomial A
3. B is the real linear array containing the complex roots of the denominator of $H(s)$.
 B (1) = real part of root No. 1
 B (2) = imaginary part of root No. 1
 B (3) = real part of root No. 2
 B (4) = imaginary part of root No. 2

If a root is real be sure to set its imag.part = 0.

4. N is the number of roots of the transfer function.
5. MULT is an integer array containing the multiplicity of each of the N input roots.
6. T is the sampling period in seconds, a real variable.
7. NSH is an integer flag used to indicate which type of reconstruction filter is to be developed.

NSH = -1 will cause first order delayed reconstruction filter to be added to the original transfer function

NSH = 0 will cause a zero order sample and hold to be added to the original transfer function

NSH = 1 will cause a first order extrapolation sample and hold to be added

NSH greater than 1 will cause a Butterworth reconstruction filter to be added to the original transfer function.

The variables numbered 8-10 need be specified only if Butterworth reconstruction is required. For sample and hold the variables 8-10 need not have any particular values.

8. WA is a real variable which specifies the approximation bandwidth in radians per second. WA must have a range of values

$$0 < WA < \left(\frac{9}{10} - \frac{\omega_s}{2} \right)$$

where ω_s is the sampling frequency in rad./sec.

9. ALPHA is a real variable which specifies the minimum attenuation of frequencies in the range $(\omega_s - \omega_a) < \omega < \omega_s$. This determines the amount of aliasing distortion in the approximation band, 0 to ω_a rad./second. For example if ALPHA = .01 then the amplitude of $H(s) \cdot R(s)$ in the frequency range $(\omega_s - \omega_a)$ to ω_s will be 1% of the amplitude of $H(s)$ alone in that region. Use of a sufficiently small value of ALPHA assures that $H(s) \cdot R(s)$ is a bandlimited function. This is necessary for correct response of the transformed function $H_z(z)$
10. MINPOL is an integer flag which indicates whether a minimum time delay or minimum number of pole Butterworth reconstruction filter is to be developed. If MINPOL is set to an integer greater than or equal to 1 a minimum pole reconstruction filter is developed. If MINPOL is set to a value less than or equal to zero, time delay of the reconstruction filter is minimized.
11. LEV is an integer flag which if less than or equal to zero causes the level setting operation performed by subroutine LEVEL to be bypassed.
12. IREC is an integer flag which if less than or equal to zero causes reconstruction filter development to be bypassed entirely. This procedure is generally acceptable when the sampling frequency is significantly higher than the transfer function bandwidth.
13. P is the output numerator polynomial of the z transform function. P is a real linear array.
14. NP is the degree of polynomial P.
15. Q is the real linear array of the z transform denominator polynomial coefficients.
16. NQ is the degree of polynomial Q.

The transform function returned is:

$$H_z(z) = \frac{P_z(z)}{Q_z(z)} = \frac{p_0 + p_1 z^{-1} + p_2 z^{-2} + \dots + p_{NP} z^{-NP}}{q_0 + q_1 z^{-1} + q_2 z^{-2} + \dots + q_{NQ} z^{-NQ}}$$

$$P(1) = p_0$$

$$Q(1) = q_0$$

$$P(2) = p_1$$

$$Q(2) = q_1$$

$$P(3) = p_2$$

$$Q(3) = q_2$$

.

.

.

.

.

.

```

      SUBROUTINE SEQUN1(B,M,A,N,MULT,T,NSH,WA,ALPHA,MINPOL,LEV,IREC,
      1P,NP,Q,NQ)
C   FOR INFORMATION ON THESE ROUTINES CONTACT E.H. GERMAN
C   BENDIX COMMUNICATIONS DIV BALTO, MD 301-VA 3 2200 EX 715
C   IMPLICIT REAL*8 (A-H,O-Z)
      DIMENSION A(1),B(1),MULT(1),P(1),Q(1)
      DIMENSION COEF(50),TRA(50),BOUT(50)
      ONE=1.E0
      NSAV=N
      IF((IREC.LE.0).AND.(NSH.LE.1)) NSH=10
      CALL RECONS(NSH,A,MULT,N,T,MINPOL,WA,ALPHA,B,M,TAU,WC,BOUT,
      1MOUT,MUSAV,IREC)
      CALL PARTLF(BOUT,MOUT,A,MULT,N,COEF,CONST)
      WRITE(6,5) M,N
      NR=0
      DO 10 I=1,N
10    NR=NR+MULT(I)
      NR=2*NR
      ND2=2*N
      WRITE(6,4)(A(I),I=1,ND2)
      WRITE(6,21) CONST
      WRITE(6,6)(COEF(I),I=1,NR)
      CALL CLEANZ(COEF,A,N,MULT,T,MOUT,CONST,P,NP,Q,NQ,NSH)
      IF(NSH.LE.1) MULT(N)=MUSAV
      N=NSAV
      CALL REFORM(A,N,MULT,TRA,NTRA)
      NR1=NTRA+1
      WRITE(6,7)(TRA(I),I=1,NR1)
      IF(LEV.LE.0) GO TO 12
      SLEV=B(1)/TRA(1)
      CALL LEVEL(SLEV,P,NP,Q,NQ)
12   CONTINUE
      WRITE(6,1)
      LAST = NP + 1
      WRITE(6,2)(P(I),I=1,LAST)
      WRITE(6,3)
      LAST = NQ + 1
      WRITE(6,2)(Q(I),I=1,LAST)
      CALL RESP(P,NP,Q,NQ,TRA,NTRA,B,M,T,TAU)
      WRITE(6,8) TAU
      RETURN
1   FORMAT(/,3X,11H OUTPUT PZ   )
2   FORMAT(/,2X,E15.8)
3   FORMAT(/,3X,11H OUTPUT QZ   )
4   FORMAT(/,3X,8H  ROOT=   ,E15.8,3X,E15.8 )
5   FORMAT(1H14X3H M=I4,5X3H N=I3)
6   FORMAT(/,3X,24H PARTIAL FRACTION COEF=   ,E15.8,3X,E15.8)
7   FORMAT(/,3X,27H REFORMED DENOMINATOR POLY=   ,E15.8)
8   FORMAT(/,3X,50H ADDITIONAL TIME DELAY DUE TO RECONSTRUCTION TAU=
1     ,E15.8 )
21  FORMAT(/,10H CONSTANT=   ,E15.8)
      END

```

```

SUBROUTINE SEQUN2(B,M,A,N,MULT,T,NSH,WA,ALPHA,MINPOL,LEV,IREC,
1P,NP,Q,NQ)
C  IMPLICIT REAL*8 (A-H,O-Z)
   DIMENSION A(1),B(1),MULT(1),P(1),Q(1)
   DIMENSION COEF(50),TRA(50),BOUT(50)
   IF((IREC.LE.0).AND.(NSH.LE.1)) NSH=10
   ONE=1.E0
   NSAV=N
   CALL RECONS(NSH,A,MULT,N,      T,MINPOL,WA,ALPHA,B,M,TAU,WC,BOUT,
1MOUT,MUSAV,IREC)
   CALL PARTLF(BOUT,MOUT,A,MULT,N,COEF,CONST)
   CALL CLEANZ(COEF,A,N,MULT,T,MOUT,CONST,P,NP,Q,NQ,NSH)
   IF(NSH.LE.1) MULT(N)=MUSAV
   N=NSAV
   CALL REFORM(A,N,MULT,TRA,NTRA)
   IF(LEV.LE.0) GO TO 12
   SLEV=B(1)/TRA(1)
   CALL LEVEL(SLEV,P,NP,Q,NQ)
12 CONTINUE
   RETURN
   END

```

2.2 BANDLIMITING THE TRANSFER FUNCTION

The transfer function must be bandlimited in order to achieve accurate simulation via the z transformation technique. A bandlimited transfer function can be described as one which has essentially a zero amplitude response above $1/2$ the sampling frequency. Shown in Fig. 2.2-1 is the amplitude response (or spectrum) of a bandlimited transfer function. The transfer function in Figure 2.2-2 is not bandlimited.

Unfortunately, bandlimited functions are not usually found in practice. It is necessary to bandlimit the transfer function before attempting transformation because the frequency response of a transformed non-bandlimited transfer function will not resemble the frequency response of the original transfer function because of "aliasing" distortion. The method used by these routines to bandlimit the function is multiplication of the original transfer function by another transfer function which is approximately bandlimited. This second bandlimited transfer function is called the reconstruction filter, $R(s)$. This technique is graphically illustrated in Figure 2.2-3

The transfer function in Fig. 2.2-3a $H'(s) = H(s) R(s)$ is bandlimited to approximately $1/4$ of the sampling frequency and has an amplitude response which closely resembles the original transfer function $H(s)$ in the approximation band (from zero to $1/4$ the sampling frequency). This new transfer function can be successfully transformed. Over the range from zero to $1/4$ th the sampling frequency the transformed function will simulate the response of the original transfer function accurately except for the addition of an approximately constant time delay associated with the reconstruction filter.

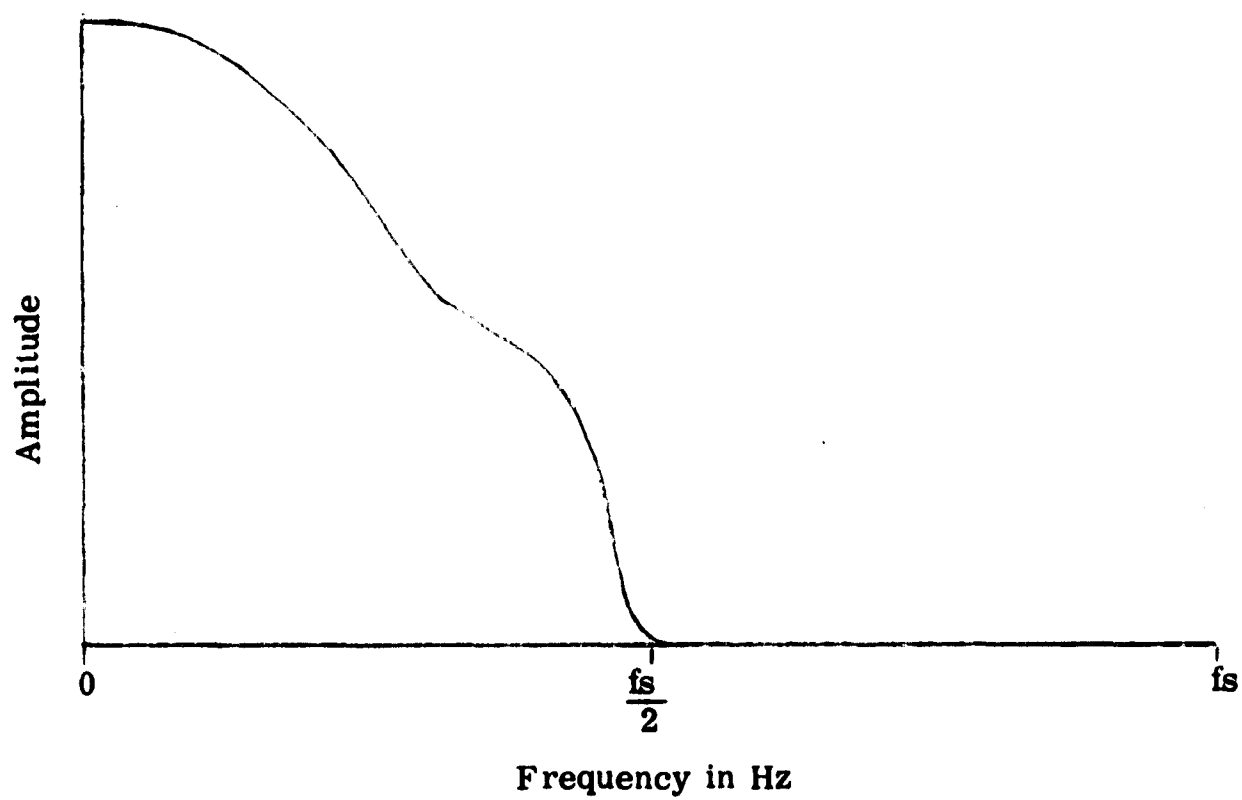


Figure 2.2-1 Amplitude Response of a Bandlimited Transfer Function

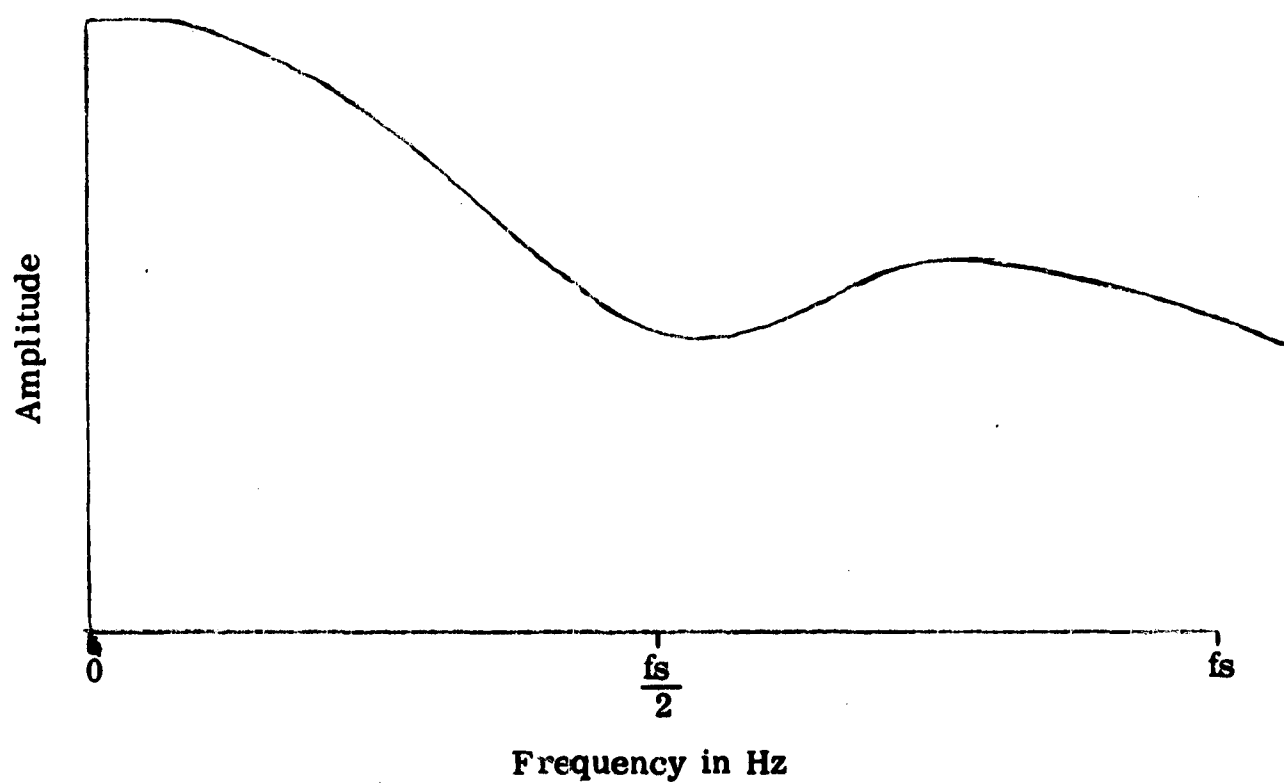


Figure 2.2-2 Amplitude Response of a Non-Bandlimited Transfer Function

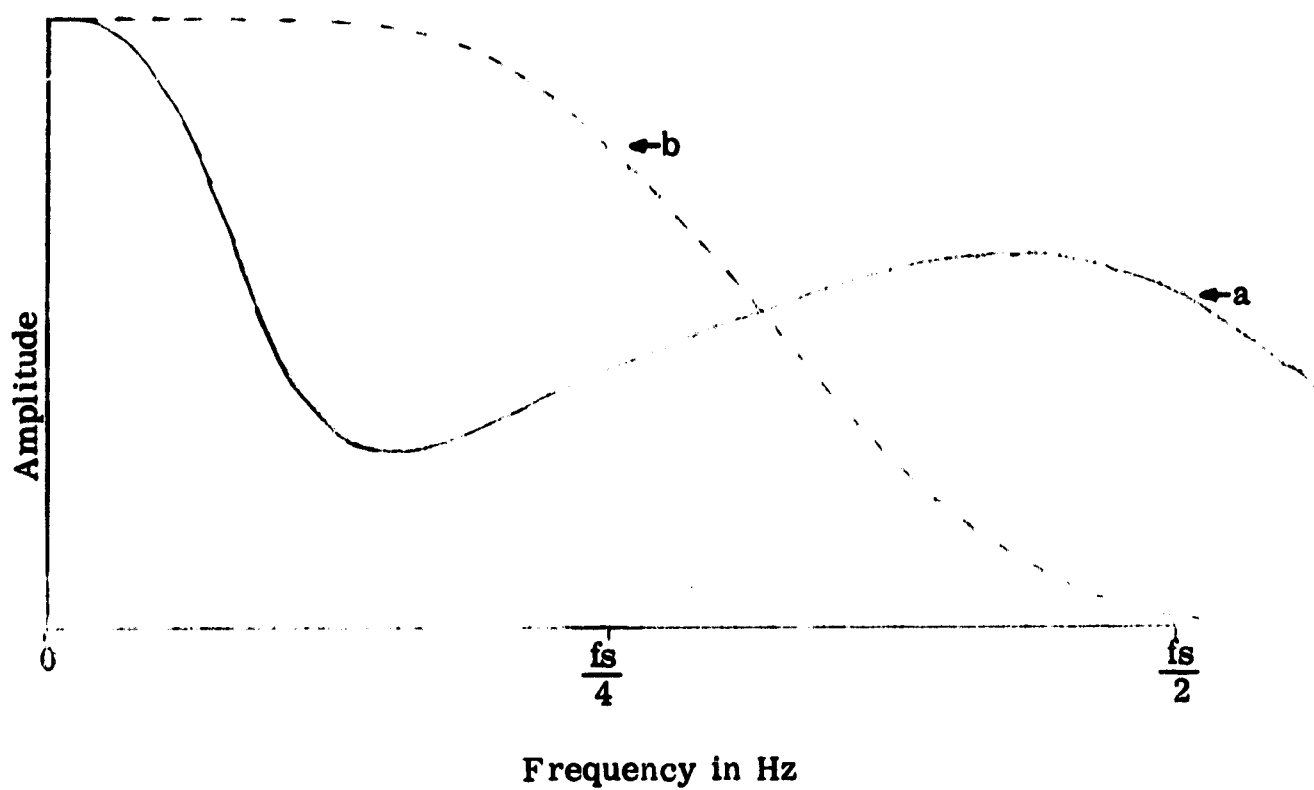


Figure 2.2-3 Transfer Function Amplitude Responses
 a - Non Bandlimited Transfer Function, $H(s)$
 b - Bandlimited Reconstruction Function, $R(s)$

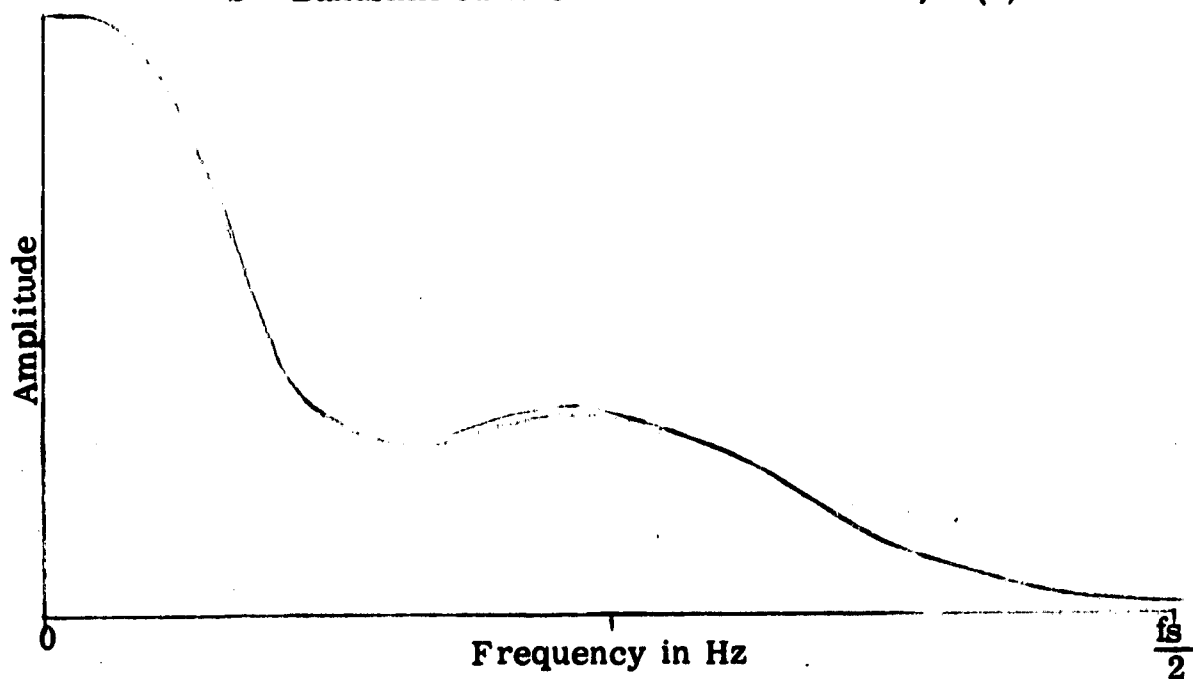


Figure 2.2-3 c- Resultant Product Transfer Function
 $H'(s) = H(s) R(s)$

2.2.1 Selection of Reconstruction Technique

There are two basic reconstruction methods available in the following routines to bandlimit a transfer function. The first method involves the use of various polynomial "sample and hold" systems adapted to digital filters. The second method develops reconstruction functions which are implemented here as Butterworth low-pass filters. The Butterworth filter reconstruction function is preferable to the sample and held methods when the transfer function has a large amplitude at frequencies greater than half the sampling frequency, though more filter weights are necessary. An example of a transfer function which usually has large amplitude at high frequencies is the derivative of a servo control function.

An advantage of the various sample and hold techniques is that they can be employed with only a slight increase in the number of filter coefficients. If it is found that they do an adequate job of reconstruction they should be used. The types of sample and hold systems which have been converted to digital filters within the programs are:

- (1) zero order sample and hold
- (2) first order extrapolation sample and hold
- (3) first order delayed sample and hold

Of the three methods presented for transfer function approximation the zero order, and the first order delayed sample and hold reconstruction are preferable. Their amplitude versus frequency responses are plotted in Figure 2.2.1-1.

If the sample and hold reconstruction techniques can not bandlimit the original transfer function sufficiently to reduce aliasing then use of the Butterworth reconstruction filter is recommended. With this technique it is necessary to specify the approximation band desired, ω_a , and the approximate aliasing level, α , within this approximation band. Notice that with sample and hold reconstruction these parameters may not be varied. With the Butterworth reconstruction filter, the reconstruction function can be developed according to simulation requirements at the cost of additional filter weights. As the reconstruction band and aliasing level approach the ideal ($\alpha = 0$, $\omega_a = \frac{\omega_s}{2}$) the number of weights increase to the point where non-recursive filters should be considered.

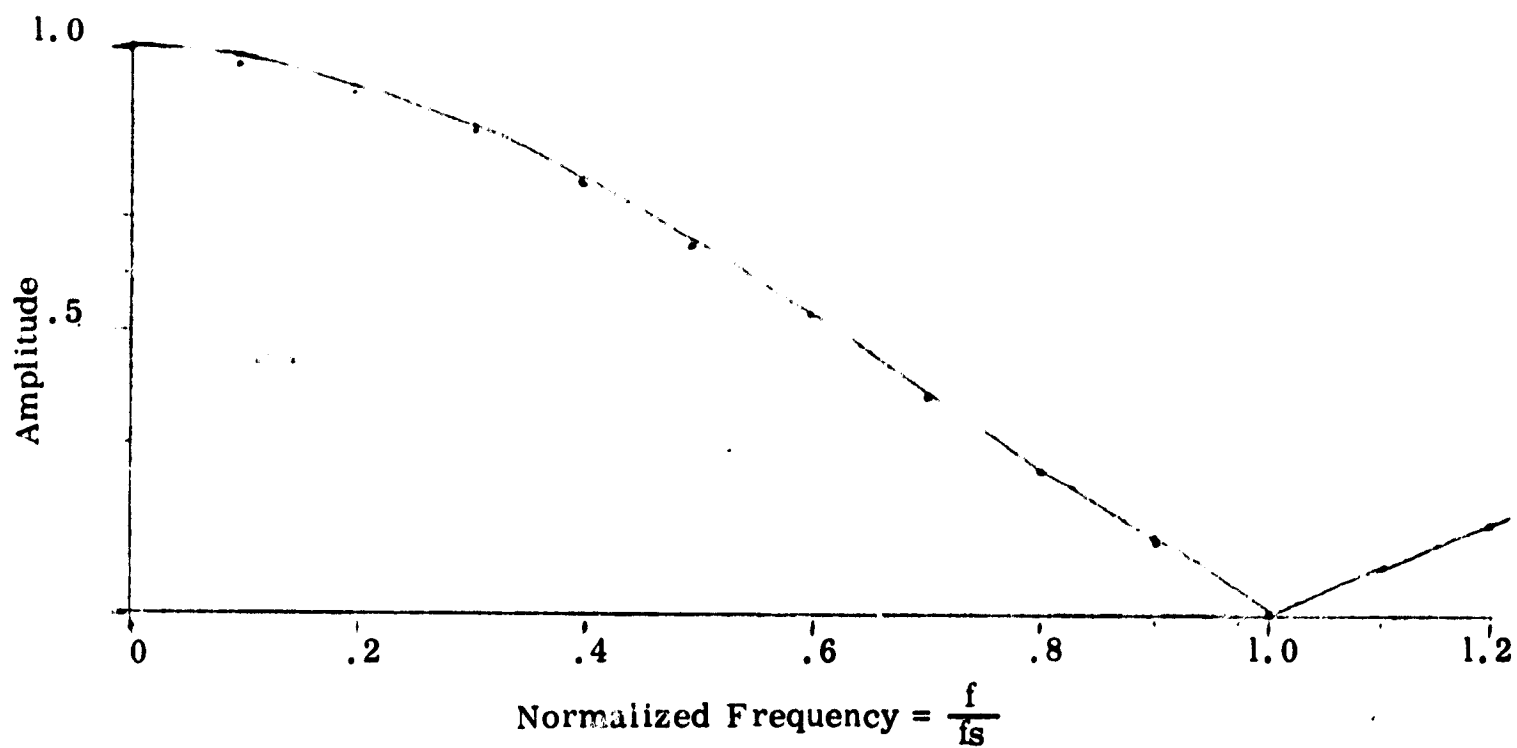


Figure 2.2.1-1a - Amplitude versus Frequency Response of Zero Order Sample and Hold

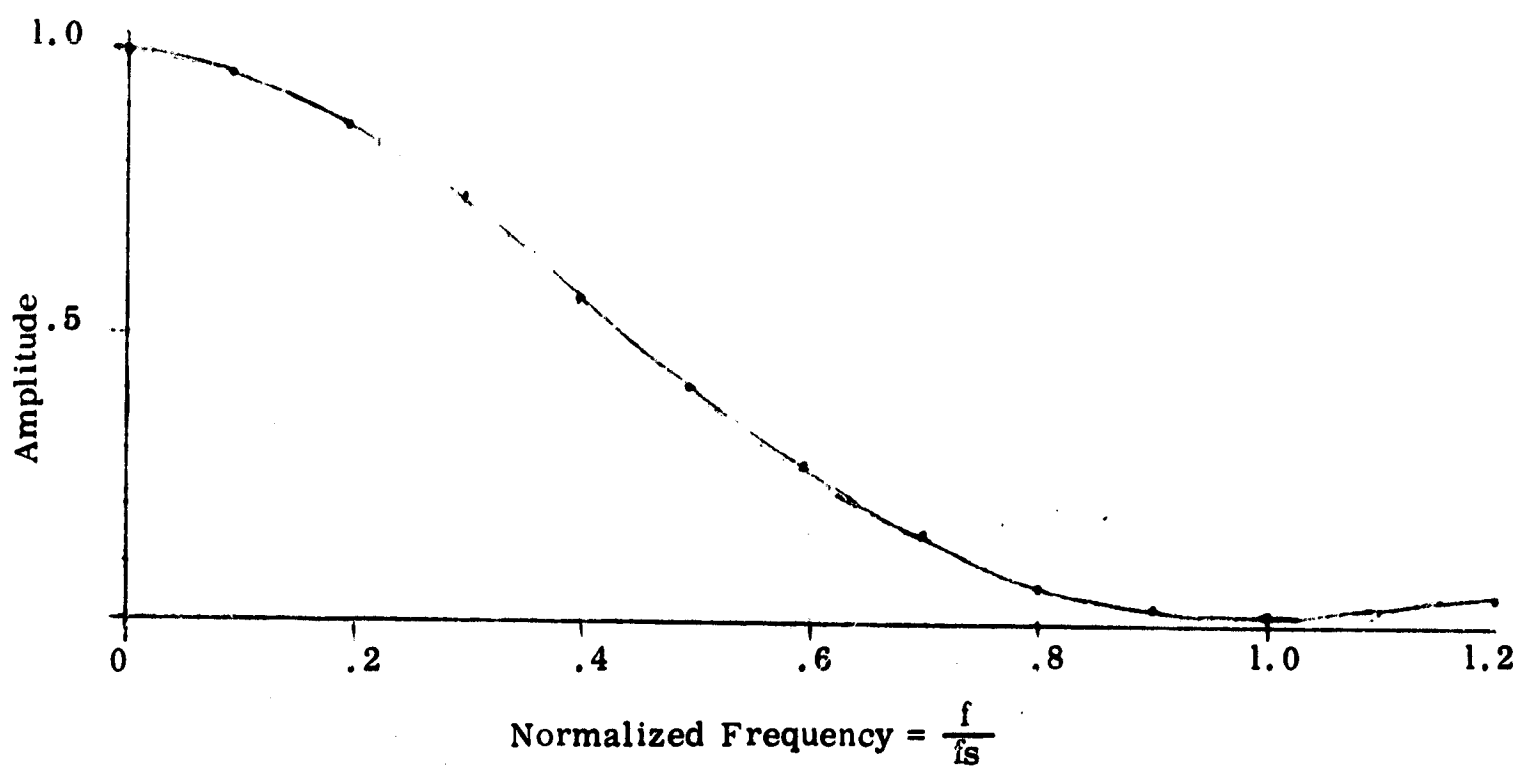


Figure 2.2.1-1b - Amplitude versus Frequency Response of First Order Delayed Sample and Hold

2.2.2 RECONS A Routine to Develop Sample and Hold and Butterworth Reconstruction Front Ends.

This routine executes development of all reconstruction front ends used to assure correct z transformation of a transfer function $H(s)$. Upon execution of this routine the root array and multiplicity array contain additional information. Effectively the input transfer function $H(s)$ is modified to have a new transfer function.

$$H'(s) = H(s) R(s)$$

where $R(s)$ is the reconstruction transfer function

There are four basic types of reconstruction filters which RECONS is able to develop. Control over which type is developed is predetermined by an input flag NSH. The user may want to try all four techniques on a given problem and evaluate the resulting z transformed function $[H(s).R(s)]$ to determine which is most suitable for his needs.

The four types of reconstruction filters available from the routine are

1. Zero order sample and hold. If this type is desired set NSH=0.

The z transform of the input transfer function $H(s)$ with a zero order sample and hold is

$$[H(s)] = H_z(z) = (1 - z^{-1}) \mathcal{Z} \left[\frac{H(s)}{s} \right]$$

where $\mathcal{Z} (\quad)$ means the z transform of the impulse response of the bracketed quantities.

2. First order extrapolation sample and hold. Set NSH=1 to get this type. The z transform of the input function $H(s)$ with first order sample and hold is

$$H_z(z) = (1 - z^{-1})^2 \mathcal{Z} \left[H(s) \frac{s + \frac{1}{T}}{s^2} \right]$$

3. First order delayed sample and hold. Set NSH=-1.

The z transform of the input H(s) is

$$H_z(z) = (1 - z^{-1})^2 \int \left[\frac{H(s)}{s^2} \right]$$

4. Butterworth reconstruction filter. To get Butterworth reconstruction set NSH to a value 2 or greater. This is a low pass filter with a maximally flat amplitude response. The routine picks the optimum order Butterworth filter to assure that aliasing is within specified tolerance (ALPHA) over the approximation band specified (WA). See the amplified discussion of this technique in the documents describing the following routines used by RECONS for development of the Butterworth filters.

- a. FINDB - which finds the optimum Butterworth filter parameters (order and corner frequency).
- b. GAM - a function routine called by FINDB.
- c. BRECO - which generates and loads the Butterworth roots into the root array (A).

CALL STATEMENT:

CALL RECONS (NSH, B, MULT, N, T, MINPOL, WA, ALPHA, A, M, TAU, WC, BS, MS, MULSAV, IREC) where the specification inputs are:

1. NSH an integer which indicates which type of reconstruction filter is desired as discussed above.
2. MINPOL is an integer flag which indicates whether a minimum pole Butterworth reconstruction or a minimum time delay Butterworth reconstruction filter is desired. If minimum pole type is desired then set MINPOL to an integer greater than or equal to one. If Butterworth reconstruction is not to be used MINPOL may have any value.

3. WA is used for Butterworth reconstruction. It is the approximation bandwidth desired in radians per second. Amplitude response of the transformed transfer function will be attenuated at frequencies greater than WA. The amplitude response will be approximately correct at frequencies from zero to WA radians/second. WA is a real variable and need not be specified to any particular value when Butterworth reconstruction is not being used.
4. ALPHA is used to roughly specify the accuracy to which the transformed transfer function matches the original analog transfer function when Butterworth reconstruction is used. It is a real variable. See discussion of ALPHA in document describing the routine FINDB.
5. IREC is an integer flag which indicates whether any form of reconstruction is to be performed. If IREC is less than or equal to zero no reconstruction modification of the transfer function is performed. In this case BOUT is set equal to B, TAU = 0, and control is returned to the calling program.
6. T is the sampling period - a real variable.
7. A is the input real array of the numerator polynomial coefficients. It is not modified by execution of the routine.
8. M is the integer degree of polynomial A

The following are generated or modified by execution of the routine

9. B is the real root array. The reconstruction transfer function's roots are added to the array.
10. MULT is the integer array of the multiplicities of the roots. The reconstruction root multiplicities are added to the array.
11. N is the number of roots of the transfer function. N is increased since reconstruction roots are added. It is an integer.
12. TAU is a real variable whose value is the additional time delay introduced by the addition of the reconstruction filter. TAU is a normalized time delay. That is,

$$TAU = \frac{\tau}{T}$$

where τ is time delay in seconds.

T is the sampling period in seconds

13. WC is a real variable whose value is the corner frequency found by the Butterworth reconstruction when this option is used. WC is the radian frequency at which the amplitude response of $R(s)$ is .707 or "3 db down" from its zero frequency value.
14. BS is the real array of the output numerator. BS is different from the input numerator B when first order sample and hold and Butterworth reconstruction are used.
15. MS is the degree of the output numerator polynomial BS. MS is integer.
16. MULSAV is an integer variable whose value is the multiplicity of the sample and hold root prior to its modification by the routine. The use intended for this variable is to save the original transfer function's multiplicity when that original transfer function contains a denominator root whose value is zero (i.e. when original transfer function contains an ideal integrator).

```

SUBROUTINE RECONS(NSH,A,MULT,N,T,MIP,WA,ALPH,B,M,TAU,WC,BS,MS,
1MULSAV,IREC)
C FOR INFORMATION ON THESE ROUTINES CONTACT E.H. GERMAN
C BENDIX COMMUNICATIONS DIV BALTO, MD 301-VA 3 2200 EX 715
C SAMPLE AND HOLD I-O VARIABLES
C A IS THE ROOT ARRAY, RECONSTRUCTION ROOTS ADDED TO END OF TRANSFER
C FUNCTION ROOTS
C NSH IS FOR SAMPLE AND HOLD RECONSTRUCTION, IF
C NSH = 0 IT A ZERO ORDER SAMPLE AND HOLD
C NSH = 1 IT IS A FIRST ORDER SAMPLE AND HOLD
C NSH = -1 IT IS A MODIFIED FIRST ORDER S AND H A LA ANDRUS
C NSH GREATER THAN 1 A BUTTERWORTH RECONSTRUCTION TYPE IS DEVELOPED
C MULT IS THE MULTIPLICITY ARRAY - INTEGER
C N IS THE NUMBER OF DISTINCT DENOMINATOR ROOTS
C BUTTERWORTH RECONSTRUCTION FRONT END I-O VARIABLES
C MULT IS THE MULTIPLICITY ARRAY, ALL NEW ROOTS IN THIS SECTION HAVE
C A MULTIPLICITY OF 1
C T IS THE SAMPLING PERIOD IN SECONDS-REAL
C A IS THE ROOT ARRAY - REAL
C MIP IS A FLAG TO INDICATE WHETHER A MINIMUM TIME DELAY OR MINIMUM
C NUMBER OF POLES IS DESIRED-INT. IF MIP.GE.1 A MIN.POLE FRONT END
C IS DEVELOPED. OTHERWISE MINIMUM TIME DELAY IS ASSUMED.
C WA IS THE BANDWIDTH IN RADIAN PER SECOND OVER WHICH ACCURATE
C TRANSFORMATION OF THE TRANSFER FUNCTION IS DESIRED - REAL
C ALPHA IS THE DESIRED AMPLITUDE RESPONSE ERROR LEVEL IN PASS BAND WA
C TAU IS THE TIME DELAY INTRODUCED BY RECONSTRUCTION NORMALIZED-REAL
C WC IS THE CORNER FREQUENCY FOR BUTTERWORTH TYPES IN RAD/SEC - REAL
C
C IMPLICIT REAL*8(A-H,O-Z)
C DIMENSION A(1),MULT(1),B(1),BS(1),B2(2)
C PI=3.1415927
C ONE=1.E0
C ZERO=0.E0
C MS=M
C M1=M+1
C IF(IREC.LE.0) GO TO 33
C IF(NSH) 7,5,6
33 TAU=ZERO
C J=0
C DO 32 I=1,M1
32 BS(I)=B(I)
C RETURN
C SAMPLE AND HOLD DESIRED
5 TAU=.5
C DO 30 I=1,M1
30 BS(I)=B(I)
C GO TO 1
6 IF(NSH.GE.2) GO TO 100
C TAU=.361
C B2(1)=ONE/T
C B2(2)=ONE
C CALL RPOLYM(B,M,B2,1,BS,MS)
C GO TO 1
7 TAU=ONE
C DO 31 I=1,M1
31 BS(I)=B(I)

```

```

1 CONTINUE
C CHECK FOR ZERO ROOTS
  DO 2 I=1,N
    IF(A(2*I-1)) 2,3,2
  3 J=I
    IF(A(2*I).EQ.ZERO) GO TO 4
    WRITE(6,20)
  2 CONTINUE
    J=N+1
    MULSAV=MULT(J)
    MULT(J)=NSH+1
    IF(NSH.LE.-1) MULT(J)=2
    A(2*J-1)=ZERO
    A(2*J)=ZERO
    N=J
    RETURN
  4 IF(J.NE.N) GO TO 42
    MULSAV=MULT(J)
    IF(NSH) 40,41,40
  40 MULT(J)=MULT(J)+2
    RETURN
  41 MULT(J)=MULT(J)+1
    RETURN
  42 NSH=10
    WA=.7*PI/T
    ALPH=.01
    WRITE(6,21)
    GO TO 100
100 CONTINUE
    J = 0
C DEVELOP OPTIMUM ORDER RECONSTRUCTION FILTER
    U=WA*T/PI
    IF(U.GE..9) GO TO 101
102 CONTINUE
    CALL FINDB(U,MIP,ALPH,NBUT,WC,TAU)
    WC=WC/T
    MS=M
    M1=M+1
    FACTLO=WC**NBUT
    DO 60 I=1,M1
  60 BS(I)=B(I)*FACTLO
    CALL BRECO(WC,NBUT,A,N,MULT)
    RETURN
101 U=.8E0
    WRITE(6,22)
    GO TO 102
20 FORMAT(///,51H PURE IMAGINARY ROOT DETECTED,IS THIS WHAT YOU WANT)
21 FORMAT(/,4X,63H ZERO ROOT DETECTED WHICH WAS NOT LOADED IN LAST RO
10T LOCATION,      ,/,5X,64H THEREFORE SAMPLE AND HOLD RECONSTRUCTIO
2N WILL NOT BE ATTEMPTED      ,/,10X,48H BUTTERWORTH RECONSTRUCTION WIL
3L BE SUBSTITUTED      )
22 FORMAT(///,60H APPROXIMATION BAND TOO WIDE, IT HAS BEEN REDUCED TO
1.4 FS      )
END

```

2.2.3 FINDB A Routine Which Finds The Optimum Order Butterworth Reconstruction Filter for a Specified Amplitude Response Accuracy

This routine finds the order of Butterworth reconstruction filter to assure that an arbitrary transfer function $H(s)$ can be z transformed to give a correct amplitude response over a specified approximation bandwidth from 0 to ω_A rad./sec. This is accomplished by specifying the maximum level of "aliasing" in the passband (ω_A). The level of aliasing distortion is set by the parameter α . Consider the following figure which is the amplitude response of some arbitrary transfer function

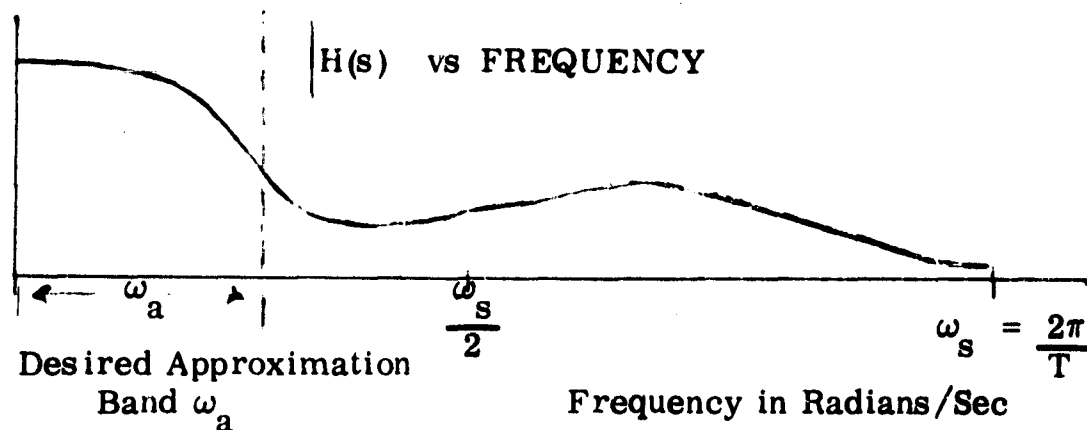


Figure 2.2.3-1 Amplitude versus Frequency Response

where:

ω_a is the upper limit of the desired approximation band in radians/sec

ω_s is the sampling frequency in radians/sec

T is the sampling period in seconds

FINDB generates two parameters which completely specify the appropriate Butterworth (maximumly flat amplitude response) reconstruction transfer function. These two parameters are the order (number of poles) of the reconstruction function N and the cut-off or corner frequency, ω_c such that $\omega_c \geq \omega_a$. An additional output of FINDB is the normalized time delay

$$\text{TAU} = \tau/T$$

where

τ is the time delay in seconds

T is the sampling period in seconds.

Two types of reconstruction transfer functions can be developed in FINDB. The types are (1) minimum time delay and (2) minimum number of poles. When the input parameter MINPOL is greater than or equal to one a minimum number of pole reconstruction transfer function is developed. With MINPOL set to zero a minimum time delay reconstruction transfer function is developed. In general, for minimum time delay reconstruction a wider bandwidth ω_c and more poles in the reconstruction function will be found. When a minimum number of poles is desired it is true that $\omega_c \approx \omega_a$. The reconstruction transfer function $R(s)$ is cascaded with the desired input transfer function $H(s)$ in the routine BRECO.

$$H'(s) = R(s) \times H(s)$$

For a complete discussion of this technique refer to section 7 . Digital Filtering and Processing By Transform Techniques, Vol. I

CALL STATEMENT:

CALL FINDB (U, MINPOL, ALPHA, N, WC, TAU)

where

1. U is the ratio of the approximation bandwidth desired to half the sampling frequency. That is

$$U = \frac{\frac{\omega_a}{\frac{\omega_s}{2}}}{2} \quad . \quad U \text{ is a real variable.}$$

2. MINPOL indicates whether a minimum pole or minimum time delay reconstruction front end filter is desired. It can be thought of as an integer flag. If MINPOL is greater than zero a minimum pole front end is found. If MINPOL less than equal or to zero the front end is optimized to give minimum time delay.
3. ALPHA is an accuracy specification on the amplitude response of the z transformed transfer function. It is a real variable. For example, if it is desired that the amplitude response of the transformed $H(s)$ match the amplitude response of $H(s)$ within 1% over the approximation band, then $ALPHA = .01$. ALPHA sets the maximum level of the reconstruction transfer function out of the approximation region. Therefore, the accuracy may be better than the specified accuracy ALPHA. Elimination of aliasing distortion is dependent upon the out-of-band amplitude response of the product function,

$$H'(s) = H(s) R(s)$$

Some transfer functions may have an amplitude response which falls off so quickly out of band that no reconstruction technique need be used.

4. N the order (or number of poles) of the reconstruction transfer function $R(s)$. N is an integer.

5. WC is the corner frequency in radians per second of the reconstruction function. WC is a real variable. Note that $WC \approx WA$. It should also be pointed out that the Butterworth reconstruction filter has an amplitude response of unity at $\omega=0$ and an amplitude response of .707 at $\omega = \omega_c$.
6. TAU is the normalized time delay of the reconstruction front end found. It is a real variable. The actual delay introduced by the reconstruction filter is
- $$\tau = \text{TAU} \times T. \text{ Where } T \text{ is the sampling period.}$$

FINDB requires the function routine GAM°. The routine GAM calculates the maximum normalized corner frequency of a given order Butterworth function.

```

C      SUBROUTINE FINDB(U,MIPOLE,ALPH,N,WC,TAU)
      IMPLICIT REAL*8 (A-H,O-Z)
      PI=3.1415927
      FOUR=4.E0
      N=1
      BETA=1.E0/ALPH
3     X=GAM(U,BETA,N)
      TAU=FLOAT(N)/(FOUR*X)
      IF(X.GE.U) GO TO 4
      N=N+1
      GO TO 3
4     IF(MIPOLE.GE.1)GO TO 1
5     X=GAM(U,BETA,N)
      NP1=N+1
      IF(TDNP1.GE.TAU) GO TO 1
      N=NP1
      GO TO 5
1     WC=X*PI
      TAU=TAU*8.E0/(PI*PI)
      RETURN
      END

```

```

C      DOUBLE PRECISION FUNCTION GAM(UD,BETA,N)
      FUNCTION GAM(UD,BETA,N)
C      IMPLICIT REAL*8 (A-H,O-Z)
      Z=BETA*BETA-1.
      A=(1.E0/(2.E0* FLOAT(N)))*ALOG(Z)
      DENOM= EXP(A)
      GAM=(2.E0-UD)/DENOM
      RETURN
      END

```

2.2.4 BRECO, A Routine To Generate The Denominator Roots of A Butterworth Reconstruction Filter

This routine generates Butterworth denominator roots when the number of roots, NB, and the corner frequency, WC, are specified. The formula for Butterworth roots in rectangular coordinates is,

$$b_i = WC \cos \left(\frac{NB + 2i - 1}{NB} \frac{\pi}{2} \right) + jWC \sin \left(\frac{NB + 2i - 1}{NB} \frac{\pi}{2} \right)$$

$i = 1, \dots, NB; j = \pm 1$

Since this routine is used to generate Butterworth roots which are to be used as a reconstruction filter, R(s) the roots generated are stored after the last root of the transfer function, H(s). If H(s) originally contained N roots it will contain N + NB roots upon execution of this routine.

For a somewhat fuller discussion of the Butterworth reconstruction technique refer to the discussion of routine FINDB. However, a complete discussion of the technique can be found in section 7, Digital Filtering and Processing By Transform Techniques, Volume I.

CALL STATEMENT:

CALL BRECO (WC, NBUT, B, N, MULT)

where Input

1. WC is the corner frequency of the Butterworth reconstruction filter - a real variable.

2. NBUT is the order of the reconstruction filter - integer.

the following variables are modified after execution of BRECO

3. B is the root array, a real linear array. Upon entry B contains only the transfer function roots. After execution B contains the transfer function roots plus the reconstruction function roots located after the transfer function roots.

4. N is the number of roots in B. After execution there are N + NBUT roots or N = N + NBUT.

5. **MULT** is an integer array which contains the multiplicity of each root. After execution **MULT** contains the multiplicities of the reconstruction roots.

```

      SUBROUTINE BRECO(WO,NBUT,A,N,MULT)
C BUTTERWORTH ROOT GENERATOR
C INPUTS
C   WO THE BUTTERWORTH CORNER FREQ IN RADIANS PER SECOND - REAL
C   NBUT THE ORDER OF THE BUTTERWORTH RECONST FILTER-INTEGER
C   A THE ROOT ARRAY CONTAINING TRANSFER FUNCTION ROOTS-REAL
C   N THE NUMBER OF ROOTS CONTAINED IN A-INTEGER
C   MULT THE MULTIPLICITY ARRAY-INTEGER
C OUTPUTS
C   A THE ROOT ARRAY WITH BUTTERWORTH ROOTS ADDED TO TRANSFER ROOTS
C   MULT THE MODIFIED MULTIPLICITY ARRAY
C   IMPLICIT REAL*8(A-H,O-Z)
      DIMENSION A(1),MULT(1)
      PI=3.1415927
      EPS=1.E-3
      TWO=2.E0
      LIM=N+NBUT
      N1=N+1
      DO 3 I=N1,LIM
3 MULT(I)=1
      TI=FLOAT(NBUT)/TWO+EPS
      LIM= INT(TI)
      INDEX=2*(N+1)
      N=N+NBUT
      IF(LIM.EQ.0) GO TO 2
      DO 1 I=1,LIM
      X= FLOAT(NBUT+2*I-1)/ FLOAT(NBUT)*PI/TWO
      A(INDEX)=WO* SIN(X)
      A(INDEX-1)=WO* COS(X)
      A(INDEX+1)=A(INDEX-1)
      A(INDEX+2)=-A(INDEX)
1 INDEX=INDEX+4
      IF(MOD(NBUT,2).EQ.0) RETURN
2 CONTINUE
      INDEX=2*N
      A(INDEX-1)=-WO
      A(INDEX)=0.E0
      RETURN
      END

```

2.3 PARTLF, General Partial Fraction Expansion Subroutine.

Description:

This subroutine expands the ratio of two polynomials into a partial fraction expansion of the form:

$$\frac{P(s)}{Q(s)} = \frac{A_1^{(m_1)}}{(s + b_1)^{m_1}} + \frac{A_1^{(m_1-1)}}{(s + b_1)^{m_1-1}} + \dots + \frac{A_1^{(1)}}{(s + b_1)} \\ + \frac{A_2^{(m_2)}}{(s + b_2)^{m_2}} + \frac{A_2^{(m_2-1)}}{(s + b_2)^{m_2-1}} + \dots + \frac{A_2^{(1)}}{(s + b_2)} + \dots \text{etc}$$

where

$P(S)$ = numerator of input polynomial ratio

$Q(S)$ = denominator of input polynomial ratio

b_j = j^{th} root of $Q(S)$

m_j = multiplicity of j^{th} root

A_r = output expansion coefficient

CONST = output expansion constant

The routine will operate with an input numerator having real coefficients, and a denominator with complex roots. Arbitrary multiplicities are allowed.

Use of the routine is limited only by accuracy (round - off) and machine size (Dimension Statements).

1. Input Data

- a) Numerator of polynomial ratio
- b) Denominator of polynomial in factored form, i.e. roots with multiplicities
- c) Number of roots
- d) Degree of numerator

2. Output Data

- a) Coefficients of partial fraction expansion
- b) Constant factor when degree of nominator is equal to degreed denominator.

3. Call Statement

CALL PARTLF (A, M, B, MULT, N, COEF, CONST)

where:

A = a real array of numerator coefficients such that:

$A(1) = a_0$, real coefficient

$A(2) = a_1$, real coefficient
etc.

M = degree of numerator

B = array of roots of denominator with format:

B(1) = root 1, real part

B(2) = root 1, imag. part

B(3) = root 2, real part

B(4) = root 2, imag. part
etc.

MULT = an integer array of multiplicities of denominator roots with format:

MULT(1) = multiplicity of root 1

MULT(2) = multiplicity of root 2
etc

N = number of roots, an integer

COEF = a real array of output expansion coefficients with format:

$$\text{COEF}(1) = A_1^{(m_1)}; \text{ real}$$

$$\text{COEF}(2) = A_1^{(m_1)}; \text{ imag.}$$

$$\text{COEF}(3) = A_1^{(m_1-1)}; \text{ real}$$

$$\text{COEF}(4) = A_1^{(m_1-1)}; \text{ imag.}$$

.

.

.

$$\text{COEF}(2*\text{MULT}(1)-1) = A_1^{(1)}; \text{ real}$$

$$\text{COEF}(2*\text{MULT}(1)) = A_1^{(1)}; \text{ imag.}$$

$$\text{COEF}(2*\text{MULT}(1)+1) = A_1^{(m_2)}; \text{ real}$$

$$\text{COEF}(2*\text{MULT}(1)+2) = A_1^{(m_2)}; \text{ imag.}$$

etc.

CONST results from dividing input numerator by input denominator when the degree of the denominator is equal to the degree of the numerator. When the denominator degree is greater, CONST = 0 is returned. CONST is a real variable.

The routine uses the following subroutines.

- (1) BASIC
- (2) COEFM
- (3) HJS
- (4) RPLYDF
- (5) PLYADD
- (6) PLYMLT
- (7) PLYDIF

Since the proper location of the input and output variables is important when using this routine a table is shown for specific transfer function.

$$\begin{aligned}
 H(s) = \frac{P(s)}{Q(s)} &= C + \frac{P'(s)}{Q(s)} = C + \frac{P'(s)}{(s+b_1)(s+b_2)^4(s+b_3)^2(s+b_4)} \\
 &= C + \frac{A_1^{(1)}}{s+b_1} + \frac{A_1^{(2)}}{(s+b_2)^4} + \frac{A_2^{(2)}}{(s+b_2)^3} + \frac{A_3^{(2)}}{(s+b_2)^2} + \frac{A_4^{(2)}}{s+b_2} \\
 &\quad + \frac{A_1^{(3)}}{(s+b_3)^2} + \frac{A_2^{(3)}}{s+b_3} + \frac{A_1^{(4)}}{s+b_4}
 \end{aligned}$$

The locations of the input and output variables in their arrays are:

I	P(I)	Q(I)	MULT(I)	A(I)
1	p ₀	-b ₁ real part	1	A ₁ ⁽¹⁾ real
2	p ₁	-b ₁ imag. part	4	A ₁ ⁽¹⁾ imag.
3	p ₂	-b ₂ real part	2	A ₁ ⁽²⁾ real
4	p ₃	-b ₂ imag. part	1	A ₁ ⁽²⁾ imag.
5	p ₄	-b ₃ real part		A ₂ ⁽²⁾ real
6	.	-b ₃ imag. part		A ₂ ⁽²⁾ imag.
7	.	-b ₄ real part		A ₃ ⁽²⁾ real
8		-b ₄ imag. part		A ₃ ⁽²⁾ imag.
.				.
.				.
15				A ₁ ⁽⁴⁾ real
16				A ₁ ⁽⁴⁾ imag.

```

SUBROUTINE PARTLF(B,MSAV,A,MULT,N,COEF,CONST)
C  IMPLICIT REAL*8(A-H,O-Z)
  DIMENSION B(1),A(1),MULT(1),COEF(1)
  M = MSAV
  CONST=0.0
  NUM=0
  DO 1 I=1,N
C  CHECK FOR IMPROPER MULTIPLICITY SPECIFICATION
    MUL=MULT(I)
    IF(MUL.LE.0) GO TO 64
    1 NUM=NUM+MULT(I)
    IF(NUM-M)2,3,4
    2 WRITE(6,5)M,NUM
    5 FORMAT(110H ERROR-PHYSICALLY UNREALIZABLE TRANSFER FUNCTION  DEGR
      1EE OF NUMERATOR IS GREATER THAN DEGREE OF DENOMINATOR      ,I2,2H /,
      2I2)
    GO TO 6
    3 CONST=B(M+1)
    4 CALL BASIC(B,M,A,MULT,N,COEF)
    6 RETURN
  64 WRITE(6,65)
  65 FORMAT(35H1MULTIPLICITIES CAN NOT BE LE ZERO    )
    RETURN
  END

```

```

C      SUBROUTINE BASIC(B,M,A,MULT,N,COEF)
      IMPLICIT REAL*8 (A-H,O-Z)
      DIMENSION B(1),A(1),MULT(1),COEF(1),Q(100)
      IFIX=2*N
      DO 956 IQ=1,IFIX
956    A(IQ)=-A(IQ)
      NUMBER=0
      DO 998 II=1,100
998    Q(II)=0.
      DO 1 J=1,N
      M3=M+1
      DO 16 LUB=1,M3
      LL=2*LUB-1
      NN=LL+1
      Q(LL)=B(LUB)
16    Q(NN)=0.
      CALL COEFM(Q,M,A,MULT,J,COEF,NUMBER,N)
1    CONTINUE
      DO 235 IQ=1,IFIX
C      PULLED
235    A(IQ)=-A(IQ)
      RETURN
      END

```

```

C      SUBROUTINE COEFM(Q,MSAV,A,MULT,J,COEF,NUM,N)
      IMPLICIT REAL*8 (A-H,O-Z)
      REAL NORM
      INTEGER D,D7
      DIMENSION Q(1) ,A(1),MULT(1),P(100),Q1(100),P1(100),COEF(1)
      M=MSAV
      DO 998 II=1,100
      P(II)=0.
      Q1(II)=0.
998 P1(II)=0.
      NORM=1.
      DO 32 MX=1,N
      AMAG= SQRT(A(2*MX-1)**2+A(2*MX)**2)
      IF(AMAG)31,31,32
31 AMAG=1.
32 NORM=NORM*AMAG**MULT(MX)
      CALL HJS(A,MULT,N,P,D,J)
      LIMIT1=2*D+2
      IF(LIMIT1.LT.1) GO TO 343
      DO 33 MQ=1,LIMIT1
33 P(MQ)=P(MQ)/NORM
343 LIMIT2=2*M+2
      IF(LIMIT2.LT.1) GO TO 344
      DO 34 MQ=1,LIMIT2
34 Q(MQ)=Q(MQ)/NORM
344 INDEX=2*J-1
      SR=-A(INDEX)
      SI=-A(INDEX+1)
      FCTRL=1.
      LIM=MULT(J)
      IF(LIM .LT.1) GO TO 345
      DO 10 KALLY=1,LIM
      IF(KALLY-1)9,8,9
9 CALL RPLYDF(Q,P,Q1,P1,M,D,M7,D7)
      M=M7
      D=D7
      FCTRL=FCTRL/DFLOAT(KALLY-1)
      MV=2*M+2
      IF(MV .LT.1) GO TO 346
      DO 1 KU=1,MV
1 Q(KU)=Q1(KU)
346 MV1=2*D+2
      IF(MV1 .LT.1) GO TO 8
      DO 2 KU=1,MV1
2 P(KU)=P1(KU)
8 CONTINUE
      PSUMR=P(1)
      PSUMI=P(2)
      SNR1=1.
      SNI1=0.
      JD=D
      IF(JD .LT.1) GO TO 347
      DO 3 JS=1,JD
      SNR=SNR1*SR-SI*SNI1
      SNI=SR*SNI1+SI*SNR1
      SNR1=SNR

```

```

      SNI1=SNI
      JR=2*JS+1
      JI=JR+1
      PSUMR=PSUMR+SNR*P(JR)-SNI*P(JI)
      3 PSUMI=PSUMI+SNR*P(JI)+SNI*P(JR)
347  SNR1=1.
      SNI1=0.
      QSUMR=Q(1)
      QSUMI=Q(2)
      IF(M .LT.1) GO TO 348
      DO 4 JQ=1,M
      SNR=SNR1*SR-SI*SNI1
      SNI=SR*SNI1+SI*SNR1
      SNR1=SNR
      SNI1=SNI
      JQR=2*JQ+1
      JQI=JQR+1
      QSUMR=QSUMR+SNR*Q(JQR)-SNI*Q(JQI)
      4 QSUMI=QSUMI+SNR*Q(JQI)+SNI*Q(JQR)
348  JX=NUM+2*KALLY-1
      SNU=PSUMR*PSUMR+PSUMI*PSUMI
      COEF(JX)=((QSUMR*PSUMR+QSUMI*PSUMI)/SNU)*FCTRL
      10 COEF(JX+1)=((QSUMI*PSUMR-QSUMR*PSUMI)/SNU)*FCTRL
345  NUM=JX+1
      RETURN
      END

```

```

SUBROUTINE HJS(A,MULT,N,P,D,J)
C  IMPLICIT REAL*8 (A-H,O-Z)
    INTEGER DA,DS,DL,DX,D
    DIMENSION A(1),MULT(1),P(1),AO(4),SCRATCH(100),SCRCH1(100)
    DO 998 II=1,100
        SCRATCH(II)=0.
998  SCRCH1(II)=0.
        LOC=0
        IF(N .LT.1) GO TO 349
        DO 4 I=1,N
            4 LOC=MULT(I)+LOC
349  LOC=LOC*LOC+2
        IF(LOC .LT.1) GO TO 350
        DO 5 I=1,LOC
            5 P(I)=0.
350  P(I)=1.
        D=0
        DO 8 K=1,N
            IF(K-J)6,8,6
            6 JX=2*K-1
            AO(1)=A(JX)
            AO(2)=A(JX+1)
            AO(3)=1.
            AO(4)=0.
            SCRATCH(1)=1.
            SCRATCH(2)=0.
            SCRATCH(3)=0.
            SCRATCH(4)=0.
            DA=1.
            DS=0.
            MULT1=MULT(K)
            IF(MULT1 .LT.1) GO TO 351
            DO 2 NW=1,MULT1
                CALL PLYMLT(AO,SCRATCH,SCRCH1,DA,DS,DL)
                MV=2*DL+2
                IF(MV .LT.1) GO TO 2
                DO 3 L=1,MV
                    3 SCRATCH(L)=SCRCH1(L)
                2 DS=DL
351  CALL PLYMLT(SCRATCH,P,SCRCH1,DS,D,DX)
            MV2=2*DX+2
            D=DX
            IF(MV2 .LT.1) GO TO 8
            DO 9 L=1,MV2
                9 P(L)=SCRCH1(L)
            8 CONTINUE
            RETURN
        END

```

```

C      SUBROUTINE RPLYDF(Q,P,Q1,P1,M,N,L,K)
      IMPLICIT REAL*8 (A-H,O-Z)
      INTEGER R,S
      DIMENSION Q(1),P(1),Q1(1),P1(1),ACCUM(100),Z(100)
      DO 998 II=1,100
      ACCUM(II)=0.
998    Z(II)=0.
      CALL PLYDIF(P,P1,N,N1)
      CALL PLYMLT(Q,P1,Q1,M,N1,R)
      CALL PLYDIF(Q,P1,M,M1)
      CALL PLYMLT(P1,P,ACCUM,M1,N,S)
      FLAG=-1.
      CALL PLYCDE(ACCUM,Q1,Q1,S,R,L,FLAG)
      JX=N
      JY=2*JX+2
      DO 1 MU=1,JY
1      Z(MU)=P(MU)
      CALL PLYMLT(P,Z,P1,N,N.K)
      RETURN
      END

```

C SUBROUTINE PLYADD(A,B,C,M,N,P,FLAG)
IMPLICIT REAL*8 (A-H,O-Z)
INTEGER P
DIMENSION A(1),B(1),C(1)
CALL POLYAD(A,M,B,N,C,P,FLAG)
RETURN
END

DIGITAL FILTERING AND PROCESSING

by

TRANSFORM TECHNIQUES,

it is ⁷⁰ VOLUME II °

A DIGITAL FILTER GENERATION
and
APPLICATION ROUTINES

FINAL REPORT

30 June 1968

CONTRACT NAS8-20308

NON-LINEAR AND LINEAR DIGITAL FILTERING TECHNIQUES

Prepared by
E. H. GERMAN JR.
K. LITTLE

for

GEORGE C. MARSHALL
SPACE FLIGHT CENTER
HUNTSVILLE, ALABAMA

THE BENDIX CORPORATION
COMMUNICATIONS DIVISION
BALTIMORE, MARYLAND 21204

C SUBROUTINE PLYMLT(A,B,C,N,M,J)
 IMPLICIT REAL*8 (A-H,O-Z)
 DIMENSION A(1),B(1),C(1)
 CALL POLYM(A,N,B,M,C,J)
 RETURN
 END

```

C      SUBROUTINE PLYDIF(A,B,N,N1)
      IMPLICIT REAL*8 (A-H,O-Z)
      DIMENSION A(1),B(1)
      N1=N-1
      IF(N)2,3,2
3     B(1)=0.
      B(2)=0.
      N1=0
      GO TO 4
2     CONTINUE
      IF(N .LT.1) GO TO 4
      DO 1 J=1,N
      JU=2*J
      JV=2*(J+1)
      JX=2*J-1
      JY=2*J+1
      B(JX)=A(JY)*FLOAT(J)
1     B(JU)=A(JV)*FLOAT(J)
4     CONTINUE
      RETURN
      END

```

**2.4-1 CLEANZ, A Routine to Develop the Modified Impulse
Invariant z Transformation of a Transfer Function
Expressed as a Partial Fraction Expansion.**

This routine produces the modified impulse invariant z transformation of a transfer function. For the case in which the degree of the numerator of the transfer function is one less than the degree of the denominator, the modification of the z transformation coefficients is,

$$H_z'(z) = \frac{P_z(z)}{Q_z(z)} - \frac{W(0^+)}{2}$$

where $W(0^+)$ is the height of the impulse response discontinuity at $t = 0$.

For the case in which the degree of the numerator is equal to the degree of the denominator (N), the z transformation must be performed on a new function of s ,

$$H(s) = \frac{P(s)}{Q(s)} = \frac{P'(s)}{Q(s)} + \frac{p_N}{q_N} = \frac{P(s)}{Q(s)} + C$$

where p_N is the coefficient of the highest order numerator term and q_N is the coefficient of the highest order denominator term of $H(s)$.

This routine assumes that the partial fraction expansion of a transfer function $H(s)$ generates the constant amplitude factor C when the degree of the numerator is equal to the degree of the denominator. It is desirable to use the routine PARTLF since it was designed to work in conjunction with CLEANZ. C can be evaluated as the quotient of the highest order numerator term over the highest order denominator term.

$$C = \frac{p_N}{q_N}$$

When the degree of the numerator is less than the degree of the denominator of $H(s)$, $C = 0$, and must be specified as zero in the calling program.

This routine requires the subroutine ZTRAN and its various subroutines.

CALL STATEMENT

CALL CLEANZ (CA, CROOT, ND, MULT, T, NN, A, P, NP, Q, NQ, NSH)

where:

INPUTS

1. CA is a complex linear array of the partial fraction expansion coefficients. The coefficients for a given multiple root are stored with the highest order partial fraction expansion factor's coefficient in the first location, the next highest order factor's coefficient in the second location, etc. See the input loading table given in the description of ZTRAN.
2. CROOT is a complex linear array of the ND distinct roots of the denominator polynomial of $H(s)$.
3. ND is an integer whose value is the number of distinct denominator roots of $H(s)$.
4. MULT is an integer array whose values correspond to the multiplicities of each of the ND distinct roots. A multiplicity of one means that a particular denominator root is not repeated; multiplicity two means that the root is repeated once, etc.
5. T is the sampling period - a real variable.
6. NN is the degree of the numerator polynomial in s, an integer variable.
7. NSH is the order of the sample and hold reconstruction filter (if used), an integer.

OUTPUTS

8. P is the real linear array of the z transfer function numerator.
9. NP is the degree of P, an integer.
10. Q is the real linear array of the z transfer function denominator.
11. NQ is the degree of Q, an integer.

All of the actual transformation is performed in routine ZTRAN. For a more complete discussion refer to the description of ZTRAN. The restrictions on CLEANZ are the same as those on ZTRAN with one additional restriction. If the numerator degree is greater than 49 then the DIMENSION statement for AR must be increased beyond 50. The only function performed by CLEANZ is to modify the ZTRAN output function $H_z(z) = \frac{P(z)}{Q(z)}$ as discussed above.

Whenever it is desired to z transform a function whose numerator degree is equal to its denominator degree the user must specify a Butterworth reconstruction front end. Zero and first order sample and hold can be used on transfer functions whose numerator degree is one or more less than its denominator degree. Only the delayed first order S & H can be used on all transfer functions.

```

SUBROUTINE CLEANZ(CA,CROOT,ND,MULT,T,NN,A,P,NP,Q,NQ,NSH)
C FOR INFORMATION ON THESE ROUTINES CONTACT E.H. GERMAN
C BENDIX COMMUNICATIONS DIV BALTO, MD 301-VA 3 2200 EX 715
C IMPLICIT COMPLEX*16 (C), REAL*8 (A,B,D-H,O-Z)
C COMPLEX CA,CROOT
C DIMENSION MULT(1),CA(1),CROOT(1),P(1),Q(1),AR(50)
C THIS ROUTINE GENERATES THE MODIFIED IMPULSE INVARIANT FILTER COEF
C FROM THE ZTRANSFORM OUTPUT COEFFICIENTS
C Z TRAN COEF ARE RESCALED TO GIVE UNITY AMPL RESP
C INPUTS
C CA, THE PARTIAL FRACTION EXPANSION COEFFICIENTS-COMPLEX
C CROOT, AN ARRAY CONTAINING THE ND DISTINCT DENOM ROOTS-COMPLEX
C ND, THE NUMBER OF DISTINCT DENOMINATOR ROOTS-INTEGER
C MULT, AN ARRAY CONTAINING THE ND MULTIPLICITIES OF EACH ROOT-INT
C NN, THE DEGREE OF THE NUMERATOR POLYNOMIAL IN S-INTEGER
C A, CONSTANT RESULT OF POLYNOMIAL DIVISION P(S)/Q(S) WHEN DEGREE OF
C P(S) = DEGREE OF Q(S), OTHERWISE A=0. - REAL
C OUTPUTS
C P, THE NUMERATOR POLYNOMIAL IN Z**(-1) - REAL
C NP, THE DEGREE OF P - INTEGER
C Q, THE DENOMINATOR POLYNOMIAL IN Z**(-1) - REAL
C NQ, THE DEGREE OF Q - INTEGER
C IF SAMPLE AND HOLD RECONSTRUCTION IS USED AND THERE IS AN IDEAL
C INTEGRATOR IN THE INPUT FUNCTION THEN IT MUST BE THE LAST ROOT
C NSH IS THE ORDER OF THE SAMPLE AND HOLD - INT
C EG, NSH EQUAL TO ZERO IS A ZERO ORDER SAMPLE AND HOLD
CALL ZTRAN(CA,CROOT,ND,MULT,T,P,NP,Q,NQ,NSH)
IF(NSH.LE.1) RETURN
ZERO = 0.E0
TWO = 2.E0
NR=0
DO 1 I=1,ND
1 NR=NR+MULT(I)
IF(A)2,3,2
3 IF(NN-NR)5,4,6
4 WRITE(6,50)
RETURN
6 WRITE(6,51)
RETURN
2 IF(NN-NR)7,10,9
7 WRITE(6,52)
RETURN
9 WRITE(6,53)
RETURN
5 FIXIMP=ZERO
A=0, NR>NN
IF(NN-NR+1)20,10,20
10 WO=ZERO
CALL FIX(CA,NR,AR)
LOC=0
DO 8 I=1,ND
LOC=LOC+MULT(I)
8 WO=WO+AR(LOC)
FIXIMP=A-T*WO/TWO
20 NP1=NP+1
DO 15 I=1,NP1

```

```

15 P(I)=Q(I)*FIXIMP+P(I)*T
   IF(NP-NQ)16,17,17
16 NP2=NP+2
   NP = NQ
   NQ1=NQ+1
   DO 25 I=NP2,NQ1
25 P(I)=Q(I)*FIXIMP
17 CONTINUE
   RETURN
50 FORMAT(/,57H INPUT ERROR,A=0., NUMERATOR DEGREE=DENOMINATOR DEGREE
1E
51 FORMAT(/, 55H INPUT ERROR, NUMERATOR DEGREE GT DENOMINATOR DEGREE
1
52 FORMAT(/,60H INPUT ERROR, A SPECIFIED BUT DENOMIN DEGREE GT NUM D
1EGREE
53 FORMAT(/,53H INPUT ERROR, NUMERATOR DEGREE GT DENOMIN DEGREE
END

```

2.4.2 ZTRAN, A Routine to Take the z Transformation of a Transfer Function H(s) Expressed as a Partial Fraction Expansion.

This routine produces the impulse invariant z transformation of the partial fraction expansion of a transfer function.

$$\mathcal{Z}(H(s)) = \mathcal{Z} \left[\sum_{j=1}^{ND} \sum_{i=1}^{N_j} \frac{A_i^{(j)}}{(s+v_j)^i} \right] = \frac{P_z(z)}{Q_z(z)} = \frac{p_0 + p_1 z^{-1} + \dots + p_N z^{-N}}{q_0 + q_1 z^{-1} + \dots + q_M z^{-M}}$$

with $q_0 = 1$

where ND is the number of distinct roots

N_j is the multiplicity of the jth root

$A_i^{(j)}$ is the partial fraction expansion coefficient of the jth root, ith term.

v_j is the jth root

The following subroutines are used by this routine,

1. JPOLE which evaluates the numerator term of a multipole.
2. COMBIN which finds the numerator of the sum of the ratios of two polynomials
3. FIX which removes the real parts of a complex array.
4. DENOM which forms the denominator of a given root term.
5. DZPOLE a subroutine of JPOLE
6. POLYAD a subroutine of JPOLE
7. POLYM a subroutine of COMBINE

CALL STATEMENT

CALL ZTRAN (A, ROOT, ND, MULT, T, P, NP, Q, NQ, NSH)

where:

1. A is a complex linear array of the partial fraction expansion coefficients. The coefficients for a given multiple root must be stored with the highest order partial fraction expansion term coefficient in the first location, the next highest order term coefficient in the second location, etc. See the input loading table which follows. Use of subroutine PARTLF to develop the partial fraction expansion assures correct loading of this array.
2. ROOT is a complex linear array of the ND distinct roots of the denominator polynomial of H(s).
3. ND is an integer whose value is the number of distinct denominator roots of H(s).
4. MULT is an integer array whose values correspond to the multiplicities of each distinct root. A multiplicity of one means that a particular denominator root is not repeated; multiplicity two means that the root is repeated once in the original transfer function; e.g. $(s + v)^2$.
5. T is the sampling period, a real variable.
6. NSH is an integer flag which indicates what order reconstruction function was used.
7. P is the real linear array of the z transfer function numerator.

$$P = p_0 + p_1 z^{-1} + p_2 z^{-2} + \dots + p_{NP} z^{-NP}$$

$$P(1) = p_0, P(2) = p_1, \dots, P(NP+1) = p_{NP}$$

8. NP is the degree of P, an integer
9. Q is the real linear array of the z transfer function denominator.

$$Q = q_0 + q_1 z^{-1} + q_2 z^{-2} + \dots + q_{NQ} z^{-NQ}$$

$$Q(1) = q_0, Q(2) = q_1, \dots, Q(NQ+1) = q_{NQ}$$

10. NQ is the degree of Q , an integer

The output transfer function returned is

$$H_z(z) = \frac{P_z(z)}{Q_z(z)} = \frac{p_0 + p_1 z^{-1} + p_2 z^{-2} + \dots + p_{NP} z^{-NP}}{q_0 + q_1 z^{-1} + q_2 z^{-2} + \dots + q_{NQ} z^{-NQ}}$$

Since the proper location of the various input variables is necessary for the routine to function a table is provided below to show the proper loading of ZTRAN input variables.

Assume a function $H(s)$ has been expressed as a partial fraction expansion series.

$$H(s) = \frac{P(s)}{(s + b_1)(s + b_2)^4(s + b_3)^2(s + b_4)} =$$

$$\frac{A_1^{(1)}}{s + b_1} + \frac{A_1^{(2)}}{(s + b_2)^4} + \frac{A_2^{(2)}}{(s + b_2)^3} + \frac{A_3^{(2)}}{(s + b_2)^2} + \frac{A_4^{(2)}}{s + b_2}$$

$$+ \frac{A_1^{(3)}}{(s + b_3)^2} + \frac{A_2^{(3)}}{(s + b_3)} + \frac{A_1^{(4)}}{s + b_4}$$

The correct loading of the input variables for this example is shown in the table below.

<u>I</u>	<u>A(I)</u>	<u>ROOT (I)</u>	<u>ND =4</u>	<u>MULT (I)</u>
1	$A_1^{(1)}$	$-b_1$		1
2	$A_1^{(2)}$	$-b_2$		4
3	$A_2^{(2)}$	$-b_3$		2
4	$A_3^{(2)}$	$-b_4$		1
5	$A_4^{(2)}$			

<u>I</u>	<u>A(I)</u>	<u>ROOT (I)</u>	<u>ND = 4</u>	<u>MULT (I)</u>
6	$A_1^{(3)}$			
7	$A_2^{(3)}$			
8	$A_1^{(4)}$			

RESTRICTIONS:

ZTRAN requires internal storage and is limited to an output numerator or denominator polynomial in z^{-1} of degree 49. That is to say, a maximum of a 49th degree denominator in $H(s)$ can be handled. If more roots are supplied it is necessary to change the following statements:

1. The DIMENSION statement at the beginning of ZTRAN.
2. The limit on the DO4 I=1, 50 statement which clears CP, CQ storage.
3. The DIMENSION RESULI (50) statement in COMBIN.
4. The DIMENSION CSTORE (50) STATEMENT IN POLYM

Fortran IV with complex arithmetic capability is required.

One final restriction on ZTRAN applies when a pure integrator is present in the input transfer function (a zero value root). If sample and hold reconstruction is used the zero root must be the last (Nth) root of the root array ROOT.

```

SUBROUTINE ZTRAN(A,ROOT,ND,MULT,T,P,NP,Q,NQ,NSH)
C   FOR INFORMATION ON THESE ROUTINES CONTACT E.H. GERMAN
C   BENDIX COMMUNICATIONS DIV BALTO, MD 301-VA 3 2200 EX 715
C   THIS TAKES THE ZTRANSFORMATION OF A F(S) WHICH IS SPECIFIED BY
C   INPUTS
C   A, THE PARTIAL FRACTION EXPANSION COEFFICIENTS-COMPLEX
C   T, THE SAMPLING PERIOD - REAL
C   ROOT, AN ARRAY CONTAINING THE ND DISTINCT DENOM ROOTS-COMPLEX
C   ND, THE NUMBER OF DISTINCT DENOMINATOR ROOTS- INTEGER
C   MULT, AN ARRAY CONTAINING THE ND MULTIPLICITIES OF EACH ROOT-INT
C   JSH IS THE LOCATION OF THE ROOT OF THE SAMPLE AND HOLD - INT
C   JSH MUST BE SET TO ZERO WHEN SAMPLE AND HOLD IS NOT USED
C   NSH IS THE ORDER OF THE SAMPLE AND HOLD - INT
C   EG, NSH EQUAL TO ZERO IS A ZERO ORDER SAMPLE AND HOLD
C   OUTPUT POLYNOMIALS ARE FUNCTIONS OF Z**(-1)
C   CPO, THE COEFFICIENTS OF THE NUMERATOR POLYNOMIAL - COMPLEX
C   NCP, THE DEGREE OF CPO POLY - INTEGER
C   CQO, THE COEFFICIENTS OF THE DENOMINATOR OR POLYNOMIAL - COMPLEX
C   NCQ, THE DEGREE OF CQO POLY - INTEGER
C   P, THE REAL PART OF CPO COEF - REAL
C   Q, THE REAL PART OF CQO COEF - REAL
C
C   IMPLICIT COMPLEX*16 (C), REAL*8 (A,B,D-H,O-Z)
C   COMPLEX CEBT,CPJ,CQJ,CP,CQ,CPC,CQO
C   COMPLEX A(1),BJT,ROOT(1)
C   DIMENSION P(1),Q(1),MULT(1)
C   DIMENSION CPJ(50),CQJ(50),CP(50),CQ(50),CPO(50),CQO(50)
C   ONE=1.E0
C   JSH=0
C   IF(NSH.LE.1) JSH=ND
C   CLEAR EACH TIME
C   DO 4 I=1,50
C   CP(I)=(0.,0.)
C   4 CQ(I)=(1.,0.)
C   CQ(1)=(1.,0.)
C   NCP=0
C   NCQ=0
C
C   KK=0
C   OUTSIDE LOOP OVER THE ND DISTINCT ROOTS
C   DO 1 J=1,ND
C   BJT = ROOT(J)*T
C   CEBT = CEXP(BJT)
C   K=MULT(J)
C   KK=KK+K
C   GENERATE NUMERATOR POLYNOMIAL
C   CALL JPOLE(A,K,KK,CEBT,T,CPJ,NCPJ)
C   GENERATE DENOMINATOR POLYNOMIAL
C   CALL DENOM(CQJ,NCQJ,K,CEBT)
C   COMBINE THE JTH NUMERATOR TERM WITH PRECEEDING NUMERATOR TERMS
C   CALL COMBIN(CP,NCP,CQ,NCQ,CPJ,NCPJ,CQJ,NCQJ,CP,NCP)
C   IF(J.NE.JSH) GO TO 5
C   THIS SECTION OF CODE HANDLES SAMPLE AND HOLD RECONSTRUCTION
C   K=K-NSH-1
C   IF(NSH.EQ.-1) K=K-2
C   CALL DENOM(CQJ,NCQJ,K,CEBT)

```

```

C  END OF RECONSTRUCTION SECTION
C  FORM NEW COMMON DENOMINATOR
    5 CALL POLYM(CQ,NCQ,CQJ,NCQJ,CQ,NCQ)
    1 CONTINUE
C  REVERSE ORDER OF COEFFICIENTS AND SCALE
    LASTP = NCP+1
    TSCAL=ONE
    IF(NSH.EQ.-1) TSCAL=ONE/T
    DO 11 I=1, LASTP
        INDX=LASTP-I+1
11   CPO(I)=CP(INDX)*TSCAL
        LASTQ = NCQ+1
        DO 12 I=1, LASTQ
            INDX=LASTQ-I+1
12   CQO(I)=CQ(INDX)
        CALL FIX(CPO,NCP,P)
        CALL FIX(CQO,NCQ,Q)
        NP = NCP
        NQ = NCQ
        RETURN
    END

```

2.4.3 JPOLE, A Routine to Generate the z Transformation

Numerator Corresponding to a Root of a Transfer Function, H(s).

This routine generates the z transform numerator polynomial corresponding to a transfer function root of any multiplicity. This transformation is performed on the partial fraction expansion representation of a particular (multiple) denominator root of H(s).

$$F(s) = \sum_{\mu=1}^{K_j} \frac{A_{\mu}^{(j)}}{(s + \beta_j)^{\mu}} \quad (2.4.3-1)$$

where

A's are the partial fraction expansion coefficients

β_j is the denominator root

K_j is the multiplicity of the denominator root

The z transformation of the multipole F(s) above is,

$$F_z(z) = \sum_{\mu=1}^{K_j} \left[\frac{A_{\mu}^{(j)}}{(\mu-1)!} \frac{\partial^{\mu-1}}{\partial \beta_j^{\mu-1}} \left(\frac{z}{z - e^{\beta_j T}} \right) \right] \quad (2.4.3-2)$$

where

T is the sampling period

Putting these terms over a common denominator of $(z - e^{\beta_j T})^{K_j}$ we have the expression,

$$F'_z(z) = \sum_{\mu=1}^{K_j} \frac{A_{\mu}^{(j)}}{(\mu-1)!} \left[\frac{\partial^{\mu-1}}{\partial \beta_j^{\mu-1}} \left(\frac{z}{z - e^{\beta_j T}} \right) \right] (z - e^{\beta_j T})^{K_j - \mu} \quad (2.4.3-3)$$

JPOLE develops the numerator of (2.4.3-3). Another routine, DZPOLE, develops the numerator of partial deviative terms in (2.4.3-3).

CALL STATEMENT

CALL JPOLE (A, K, KK, CEBT, T, QTEMP, NQT)

where;

1. A is a complex linear array of the partial fraction expansion coefficients.
The coefficients of a given multipole with multiplicity K must be store in A from highest order term first to zero order term last in location KK. That is,

STORAGE	COEF. STORED	CORRESPONDING P. F. Expansion Term
A (KK-K)	A_K	$\frac{A_{K_j}^{(j)}}{(s + \beta_j)^{K_j}}$
A (KK-K + 1)	A_{K-1}	$\frac{A_{K_j-1}^{(j)}}{(s + \beta_j)^{K_j-1}}$
.	.	.
.	.	.
.	.	.
A (KK)	A_0	$\frac{A_0^{(j)}}{(s + \beta_j)}$

2. K is an integer whose value is the multiplicity of the jth root being transformed by JPOLE.
3. KK is an integer whose value is the location of the zero order partial function coefficient in array A.
4. CEBT is a complex variable whose value is $e^{\beta_j T}$
5. T is a real variable whose value is the sampling period.
6. QTEMP is a complex linear array containing the numerator polynomial output.
7. NQT is an integer whose value is the degree of QTEMP.

Subroutines called by JPOLE:

1. DZPOLE, a routine which generates the partial terms in (2.4.3-3).
2. POLYM, a polynomial multiplication routine for complex arguments.
3. POLYAD, a polynomial addition routine for complex arguments.

RESTRICTIONS:

1. The output polynomial of JPOLE can have an order of 25 or less. That means that the JPOLE is restricted to handling 24 repeated roots. If more than 24 repeated roots are to be handled the dimension of Q must be increased.
2. JPOLE must be run on Fortran IV with complex addition, subtraction and multiplication.

```

      SUBROUTINE JPOLE(A,K,KK,CEBT,T,QTEMP,NQT)
C  JPOLE GENERATES THE NUMERATOR OF A DISTINCT POLE OF ANY MULTIPLICITY
C  INPUTS
C    A, THE ARRAY OF PARTIAL FRACTION COEFFICINTS - COMPLEX
C    K, THE MULTIPLICITY OF THE ROOT BEING TRANSFORMED - INTEGER
C    KK, THE LOCATION OF THE ZERO ORDER PARTIAL FRACTION COEF - INTEGER
C    CEBT, = CEXP(ROOT(J)*T)
C  OUTPUTS
C    QTEMP, THE COEFF OF THE NUMERATOR POLYN - COMPLEX
C    NQT, THE DEGREE OF QTEMP - INTEGER
C    IMPLICIT COMPLEX*16 (C), REAL*8 (A,B,D-H,O-Z)
C    COMPLEX*16 Q,A,QTEMP,P,AA
C    COMPLEX Q,A,QTEMP,P,AA,CEBT
C    DIMENSION Q(50),P(2)
C    DIMENSION A(1),QTEMP(1)
C
      AD = 1.
      ZERO=0.E0
      NZERO=0
      CALL DZPOLE(CEBT,T,NZERO,QTEMP,NQT)
      MU=K-1
      LOC = KK
      AA=A(LOC)
      QTEMP(1)=QTEMP(1)*AA
      QTEMP(2)=QTEMP(2)*AA
      IF(MU.EQ.0) GO TO 2
      P(1)=-CEBT
      P(2)=(1.E0,0.E0)
      NP=1
      FACT = 1.E0
      DO 1 J=1,MU
      CALL DZPOLE(CEBT,T,J,Q,NQ)
      LOC=LOC-1
      FACT = FACT* FLOAT(J)
      AA=A(LOC)/FACT
      CALL POLYM(Q,NQ,AA,NZERO,Q,NQ)
      CALL POLYM(P,NP,QTEMP,NQT,QTEMP,NQT)
      CALL POLYAD(QTEMP,NQT,Q,NQ,QTEMP,NQT,AD)
1  CONTINUE
2  CONTINUE
      RETURN
      END

```

2.4.4 DZPOLE, A Routine to Generate the Numerator of the Partial Derivatives of the z Transformation Corresponding to a Root of a Transfer Function, H(s).

This routine generates the numerator of the partial derivatives required to z transform a partial fraction expansion term of transfer function $H(s) = \frac{P(s)}{Q(s)}$. A single partial fraction expansion term of a multipole is of the following type,

$$\frac{A_u^{(j)}}{(s + \beta_j)^u} \quad (2.4.4-1)$$

where β_j is the multipole root, $A_u^{(j)}$ is the u^{th} partial fraction expansion coefficient of the j^{th} root.

The transform of this typical term is given by,

$$\mathcal{Z} \left[\frac{A_u^{(j)}}{(s + \beta_j)^u} \right] = \frac{A_u^{(j)}}{(u-1)!} \left[\frac{\partial^{u-1}}{\partial \beta_j^{u-1}} \left(\frac{z}{z - e^{\beta_j T}} \right) \right] \quad (2.4.4-2)$$

DZPOLE generates the numerator polynomial of the partial term in (2.4.4-2),

$$\frac{\partial^{u-1}}{\partial \beta_j^{u-1}} \left(\frac{z}{z - e^{\beta_j T}} \right) \quad (2.4.4-3)$$

CALL STATEMENT:

CALL DZPOLE (CEBT, T, K, CZNUM, NCZ)

where:

1. CEBT is a complex variable whose value is $e^{(\sigma + i\omega)}$ where $(\sigma + i\omega)$ is the denominator root of the term being transformed. That is $\beta_j = \sigma + i\omega$. $i = \sqrt{-1}$
2. T is the sampling period, a real variable.
3. K is an integer which indicates which term of the multipole is being transformed.

4. CZNUM is the numerator polynomial in z returned a complex linear array.
5. NCZ is the degree of the polynomial CZNUM, an integer

RESTRICTIONS:

1. The output polynomial CZNUM, has a maximum degree of 24.. This means that DZPOLE is restricted to 25 repeated roots. If more than 25 repeated roots are to be handled the dimension of A and CP must be increased.
2. The order of calling DZPOLE must be from zero or first order term (K=0 or K=1) to the highest order term. Calling DZPOLE from the highest order to the lowest order will result in incorrect values of CZNUM generated.
3. FORTRAN IV with complex arithmetic capability is required.

```

      SUBROUTINE DZPOLE(CEBT,T,J,CZNUM,NDG)
C  DZPCLE GENERATES NUMERATOR DERIVATIVE TERMS NECESSARY FOR MULTIPOLE
C  Z TRANSFORMATIONS
C  INPUTS
C    CEBT= UEXP(ROOT(J)*T) - COMPLEX
C    J, IS WHICH DERIVATIVE IS BEING TAKEN J=MULT-1 - INTEGER
C    T, IS THE SAMPLING PERIOD - REAL
C    ORDER OF EXECUTION PROCEEDS FROM LOWEST ORDER DERIV TO HIGHEST
C    J MAY START WITH EITHER A VALUE OF 0 OR 1 BUT NOT HIGHER THAN 1
C  OUTPUTS
C    CZNUM, IS THE NUMERATOR POLYNOMIAL IN Z
C    NDG, IS THE DEGREE OF CZNUM
C
C    IMPLICIT COMPLEX*16 (C), REAL*8 (A,B,D-H,O-Z)
C    COMPLEX CEBT,CEBTP,CB,CZNUM,CP
C    DIMENSION CZNUM(1),A(50),CP(50)
C    ZERO=0.E0
C    ONE=1.E0
C    CZNUM(1)=ZERO
C    JJ=J+1
C    IF(JJ-3)6,6,7
7  JJ=3
6  GO TO (1,2,3),JJ
C    1 IS FOR FIRST ORDER POLES
C    2 IS FOR SECOND ORDER POLES
C    3 IS FOR HIGHER ORDER POLES
1  CZNUM(2)=ONE
  NDG=1
  RETURN
2  CONTINUE
  CZNUM(2)=T*CEBT
  NDG = 1
  RETURN
3  CONTINUE
  A(1)=ONE
  A(J)=ONE
  JJJ=J-2
  IF(JJJ)4,4,5
4  CONTINUE
  GO TO 18
5  CONTINUE
C  COMPUTE THE A'S
  DO 8 I=1,JJJ
    JMI=J-I
    A(JMI)= FLOAT(JMI)*A(JMI)+A(JMI-1)
8  CONTINUE
18  CONTINUE
  NDG=-1
  CEBTP=ONE
  KFACT = 1
C  GENERATE DERIVATIVE TERMS
  DO 10 K=1,J
    KFACT=KFACT*K
    CEBTP=CEBT*CEBTP
    CB=A(K)*CEBTP*FLOAT(KFACT)
    IF(NDG)12,11,11

```

```

12 CP(1)=CB
   NDG=0
   GO TO 10
11 CP(K)=CP(K-1)
   IF(K.EQ.2) GO TO 15
   LIM = K-2
   DO 16 I=1,LIM
   KMI=K-I
16 CP(KMI)=CP(KMI-1)-CEBT*CP(KMI)
15 CP(1)=-CEBT*CP(1)+CB
10 CONTINUE
   NDG=J
   TPOW=T**J
   DO 20 I=1,J
20 CZNUM(I+1)=CP(I)*TPOW
   RETURN
   END

```

2.4.5 DENOM, A Routine to Generate the z Transformation Denominator Corresponding to a Root of a Transfer Function, H(s)

This routine generates the denominator polynomial of a given multiple denominator root. The z transformation is performed on the partial fraction expansion representation of a transfer function H(s). A particular (jth) multiple denominator root of H(s) is represented in general as

$$F(s) = \sum_{u=1}^{K_j} \frac{A_u^{(j)}}{(s + \beta_j)^u}$$

where the A's are partial fraction expansion coefficients,

β_j the jth denominator root of H(s)

K_j is the multiplicity of this root

It can be shown that the z transformation of the above is (see document on JPOLE)

$$F_z(z) = \sum_{u=1}^{K_j} \frac{A_u^{(j)}}{(u-1)!} \left[\frac{\partial^{u-1}}{\partial \beta_j^{u-1}} \frac{z}{z - e^{\beta_j T}} \right] (z - e^{\beta_j T})^{K_j - u}$$

where T is the sampling period.

From this we see that the common denominator factor is

$$(z - e^{\beta_j T})^{K_j}$$

This is the polynomial generated in DENOM.

CALL STATEMENT:

CALL DENOM (QJ, NQJ, K, CEBT)

where

1. QJ is the output denominator polynomial with complex coefficients. It must be sufficiently dimensioned in the calling program.
2. NQJ is the degree of polynomial QJ, an integer.
3. K is an integer whose value is the multiplicity of the root being transformed.
4. CEBT is a complex variable calculated in the calling program as $CEBT = e^{j\beta} \Gamma$.

This routine is restricted to use on machines with FORTRAN IV which has complex arithmetic capability.

```

SUBROUTINE DENOM(CQJ,NCQJ,K,CEBT)
COMPLEX CQJ,CEBT
C  IMPLICIT COMPLEX*16 (C), REAL*8 (A,B,D-H,O-Z)
DIMENSION CQJ(1)
ONE=1.E0
IF(K.GT.0) GO TO 1
CQJ(1)=ONE
NCQJ=0
RETURN
1 CQJ(1)=-CEBT
CQJ(2)=ONE
IF(K.LE.1) GO TO 3
KM1=K-1
DO 2 IT=1,KM1
LIM=IT+1
CQJ(LIM+1)=(0.E0,0.E0)
DO 7 JJ=1,LIM
LOC=LIM-JJ+2
CQJ(LOC)=CQJ(LOC-1)-CEBT*CQJ(LOC)
7 CONTINUE
CQJ(1)=-CEBT*CQJ(1)
2 CONTINUE
3 NCQJ=K
RETURN
END

```

2.4.6 LEVEL, A Routine To Adjust The Response Of A Transfer Function in z to a Specified Level at Zero Frequency

This routine adjusts a transfer function in z to give a specified response at zero frequency. It is designed to make small corrections in response and should not be used for gross level setting. Gross level setting must be done before z transforming the transfer function $H(s)$. Hence this routine should be called after the modified impulse invariant polynomials have been developed (see routine CLEANZ).

A transfer function in z is represented as a ratio of two polynomials in z,

$$H_z(z) = \frac{P_z(z)}{Q_z(z)} = \frac{p_0 + p_1 z^{-1} + p_2 z^{-2} + \dots + p_N z^{-N}}{q_0 + q_1 z^{-1} + q_2 z^{-2} + \dots + q_M z^{-M}}$$

The response of $H_z(z)$ at some frequency ω can be evaluated by making the substitution

$$z = e^{i\omega T}$$

where:

ω is the radian frequency at which the transfer function is to be evaluated

T is the sampling period in seconds.

Now at $\omega = 0$, $z = 1$ hence the response at $\omega = 0$ of a transfer function in z is simply the sum of the numerator coefficients over the sum of the denominator coefficients,

$$\left| H_z(z) \right|_{z=1} = \frac{\sum_{j=1}^N p_j}{\sum_{j=1}^M q_j} = \text{ZLEV}$$

The desired response is the response of the original transfer function at zero frequency.

$$\left| H(s) \right|_{s=0} = \frac{b_0 + b_1 s + b_2 s^2 + \dots + b_N s^N}{a_0 + a_1 s + b_2 s^2 + \dots + b_M s^M} = \frac{b_0}{a_0} = \text{SLEV}$$

The z transformed filter response error is the difference

$$\text{DIFFER} = \text{SLEV} - \text{ZLEV}$$

Small corrections in the transfer function zero frequency response can be made by adding DIFFER to the original transfer function.

$$\begin{aligned} H'_z(z) &= \frac{P_z(z)}{Q_z(z)} + \text{DIFFER} \\ &= \frac{P_z(z) + \text{DIFFER} \cdot Q_z(z)}{Q_z(z)} \end{aligned}$$

$$H'_z(z) = \frac{P'_z(z)}{Q_z(z)} \quad \text{which is the new transfer function returned to the calling program.}$$

No other subroutines are used by this routine. This routine has only two dimensioned arrays, P and Q. These variables derive their dimension from the calling program.

CALL STATEMENT

CALL LEVEL (SLEV, P, NP, Q, NQ)

where:

1. SLEV is the response of the transfer function H(s) at zero frequency, a real variable. This is the zero frequency response which H_z(z) will have after execution of this routine.
2. P is a real linear array containing the numerator coefficients of the z transfer function. It serves as an input and an output. The corrected coefficients are returned to the input variable.

3. NP is the degree of polynomial P. It serves as both an input and output. Note that when the degree of the input numerator is less than the degree of the denominator the resulting numerator polynomial will have a degree equal to the denominator degree. This variable is an integer.
4. Q is a real linear array containing the denominator polynomial coefficients of the z transfer function. It serves as an input but is always returned unchanged.
5. NQ is the degree of the denominator polynomial. This variable is an integer.

```

      SUBROUTINE LEVEL(SLEV,P,NP,Q,NQ)
C      IMPLICIT REAL*8 (A-H,O-Z)
      DIMENSION P(1),Q(1)
C      F = 0 LEVEL SET ROUTINE
      ZERO=0.E0
      EPS=1.E-5
      NP1=NP+1
      NQ1=NQ+1
      SUMP=ZERO
      SUMQ=ZERO
      DO 1 I=1,NP1
1      SUMP=SUMP+P(I)
      DO 2 I=1,NQ1
2      SUMQ=SUMQ+Q(I)
      ZLEV=SUMP/SUMQ
      DIFFER=SLEV-ZLEV
      IF( ABS(DIFFER).LT.EPS) RETURN
      NP2=NP+2
      IF(NP.GT.NQ) NP1=NQ1
      DO 3 I=1,NP1
3      P(I)=P(I)+Q(I)*DIFFER
      IF(NP.GE.NQ) RETURN
      DO 4 I=NP2,NQ1
4      P(I)=Q(I)*DIFFER
      NP=NQ
      RETURN
      END

```

2.5.1 COMBIN, A Routine For Adding Two Ratios Of Two Polynomials

This routine finds the numerator resulting from the addition of two ratios of two polynomials. All polynomials are represented by complex linear arrays. The function performed is:

$$\frac{P_1}{Q_1} + \frac{P_2}{Q_2} = \frac{P_1 Q_2 + P_2 Q_1}{Q_1 Q_2}, \quad P_3 = P_1 Q_2 + P_2 Q_1$$

The routine requires two other routines for execution (1) POLYM, a complex polynomial multiplying routine, and (2) POLYAD a complex polynomial adding routine.

The listing shown is for complex polynomials. The routine can be modified to handle other types of variables by changing the type statements in COMBIN and by changing POLYM and POLYAD to handle the desired type of variables.

CALL STATEMENT

CALL COMBIN (P1, NP1, Q1, NQ1, P2, NP2, Q2, NQ2, P3, NP3)

where:

1. P1 is the numerator polynomial of the first ratio of polynomials, a complex linear array.
2. NP1 is the degree of P1, an integer.
3. Q1 is the denominator polynomial of the first ratio of polynomials, a complex linear array.
4. NQ1 is the degree of Q1, an integer
5. P2 is the numerator polynomial of the second ratio of two polynomials, a complex linear array.
6. NP2 is the degree of P2, integer.
7. Q2 is the denominator of the second ratio of two polynomials, complex linear array.
8. NQ2 is the degree of Q2, integer.
9. P3 is the numerator of the returned ratio of two polynomials, a linear array.

10. NP3 is the degree of P3, integer.

Note that one input ratio of two polynomials may be used as the output ratio of two polynomials. For example Call COMBIN (P1, NP1, Q1, NQ1, P2, NP2, Q2, NQ2, P1, NP1) is legal. The resulting output numerator is returned to the first input numerator location in the calling program.

RESTRICTIONS:

1. Combine is restricted to polynomials whose inputs $(NP + NQ2) \leq 49$.
If this condition is to be exceeded the dimension of RESULT must be increased beyond the dimension of 50 given.
2. Combine is further restricted in that $(NQ1 + NP2) \leq 49$. This restriction can be removed by increasing the dimension of the array CSTORE in the subroutine POLYM.

```

C      SUBROUTINE COMBIN(P,NP,Q,NQ,PTEMP,NPT,QTEMP,NQT,POUT,NPO)
C      NP,NQ,NPT,ETC ARE THE DEGREES OF THIR RESPECTIVE POLYNOMIALS
C      COMBIN FORMS THE NUMERATOR POLYNOMIAL OF THE SUM OF THE RATIO
C      OF TWO POLYNOMIALS, THAT IS
C      POUT = (P) X (QTEMP) + (PTEMP) X (Q)
C      NUMERATOR IS FORMED IN POUT POLYNOMIAL
C      IMPLICIT COMPLEX*16 (C), REAL*8 (A,B,D-H,O-Z)
COMPLEX    Q(1),QTEMP(1),P(1),PTEMP(1),POUT(1)
COMPLEX    RESUL1
DIMENSION RESUL1(50)
CALL POLYM(P,NP,QTEMP,NQT,RESUL1,NRES)
CALL POLYN(Q,NQ,PTEMP,NPT,POUT,NPO)
AD = 1.
CALL POLYAD(POUT,NPO,RESUL1,NRES,POUT,NPO,AD)
RETURN
END

```

2.5.2 **FIX, A Routine For Obtaining the Real Part Of A Complex Array.**

This routine obtains the real parts of a complex linear array which represents a complex polynomial and returns them to a real polynomial. The main program supplies the dimensions of the two variables. There is a severe restriction on the routine that the compiler must store the real and imaginary parts of a complex variable in sequential address locations.

CALL STATEMENT

CALL FIX (CPLX, NDG, OREAL)

where:

1. CPLX is a complex linear array
2. NDG is the degree of the polynomial CPLX hence it is also the degree of the output polynomial OREAL. The dimension of both arrays is then NDG + 1
3. OREAL is the output real linear array containing the real parts of the complex input.

or more graphically,

	Input	Output	
<u>I</u>	<u>CPLX (I)</u>	<u>OREAL (I)</u>	<u>NDG = 3</u>
1	$\alpha_1, j\beta_1$	α_1	
2	$\alpha_2, j\beta_2$	α_2	
3	$\alpha_3, j\beta_3$	α_3	
4	$\alpha_4, j\beta_4$	α_4	

```

C      SUBROUTINE FIXICPLX,NC,OREAL)
      IMPLICIT REAL*8(A-H,O-Z)
      DIMENSION CPLX(1),OREAL(1)
      LAST = NC+1
      DO 1 I=1, LAST
1 OREAL(I) = CPLX(2*I-1)
      RETURN
      END

```

2.5.3 POLYAD and RPOLYA , Routines For Adding Or Subtracting Two Polynomials .

This routine adds polynomial A to polynomial B and returns the result in polynomial C. When subtraction is required the second polynomial in the call statement is subtracted from the first polynomial in the call statement. Subtraction is performed when a flag is less than or equal to zero. All polynomials are stored in a linear array, from lowest to highest order coefficients. If polynomial $A = a_0 + a_1 x + a_2 x^2 + \dots + a_n x^n$, then it must be stored as a linear array.

$$\begin{aligned} A(1) &= a_0 \\ A(2) &= a_1 \\ A(3) &= a_2 \\ &\vdots \\ &\vdots \\ &\vdots \\ A(N+1) &= a_n \end{aligned}$$

The listing shown is for complex polynomials. The routine can be modified to handle other types of variables by changing the explicit type statement, and by changing the definition statement for the variable CONST, the first operating statement of the routine.

CALL STATEMENT

CALL POLYAD (A, NA, B, NB, C, NC, FLAG)

where

1. A is the complex linear array used as one input.
2. NA is the degree of A, an integer variable.
3. B is the complex linear array used as the second input.
4. NB is the degree of B, an integer variable.
5. C is the output linear array also a complex array.
6. NC is the degree of C, an integer variable.

7. FLAG is a real variable which if greater than zero causes addition to occur and if less than or equal to zero causes subtraction to occur $[C(I) = A(I) - B(I)]$.

Note that in the call statement one of the input variables can be the output variable. For example we can say. CALL POLYAD (X, NX, Y, NY, X, NX, FLAG)
The resulting summed polynomial and its degree replace X and NX in the calling program in this case.

Another version of this routine works the same but has all real arguments. Its CALL statement is

CALL RPOLYA (A, NA, B, NB, C, NC, FLAG)

```

SUBROUTINE POLYAD(A,NA,B,NB,C,NC,FLAG)
C   SUBROUTINE POLYAD ADDS POLYNOMIAL A TO POLY B, THE RESULT IS C
C   NA AND NB ARE THE DEGREE OF POLY A AND B RESPECTIVELY. NC IS
C   THE DEGREE OF POLY C. FLAG INDICATES WHETHER ADDITION OR SUBT
C   IS TO BE PERFORMED. IF FLAG>0, ADDITION IS PERFORMED.
C   LOADING OF POLYNOMIALS MUST BE FROM LOWEST TO HIGHEST ORDER TERM
C   EITHER A OR B MAY BE THE SAME VARIABLE AS C IN THE CALL STMT, EG
C   CALL POLYAD(X,NX,Y,NY,Y,NY) IS LEGAL
C   IMPLICIT COMPLEX*16 (C), REAL*8 (A,B,D-H,O-Z)
COMPLEX    A(1),B(1),C(1)
CONST=1.0
IF(FLAG.LE.0.E0) CONST=-1.E0
IF((A-NB)/1,2,2
C   B IS THE LARGER POLYNOMIAL
1  NC=NB
   NCP1=NC+1
   LIM=NA+2
   DO 5 I=LIM,NCP1
5  C(I)=CONST*B(I)
   GO TO 3
C   A IS THE LARGER POLYNOMIAL
2  NC=NA
   NCP1=NC+1
   LIM=NB+2
   IF(LIM.GT.NCP1) GO TO 3
   DO 6 I=LIM,NCP1
6  C(I)=A(I)
3  CONTINUE
   LIM=LIM-1
   DO 4 I=1,LIM
4  C(I)=A(I)+CONST*B(I)
   RETURN
END

```

```

SUBROUTINE RPOLYA(A,NA,B,NB,C,NC,FLAG)
DIMENSION A(1),B(1),C(1)
CONST=1.0
IF(FLAG.LE.0.E0) CONST=-1.E0
IF(NA-NB)1,2,2
C B IS THE LARGER POLYNOMIAL
1 NC=NB
  NCP1=NC+1
  LIM=NA+2
  DO 5 I=LIM,NCP1
5 C(I)=CONST*B(I)
  GO TO 3
C A IS THE LARGER POLYNOMIAL
2 NC=NA
  NCP1=NC+1
  LIM=NB+2
  IF(LIM.GT.NCP1) GO TO 3
  DO 6 I=LIM,NCP1
6 C(I)=A(I)
3 CONTINUE
  LIM=LIM-1
  DO 4 I=1,LIM
4 C(I)=A(I)+CONST*B(I)
  RETURN
  END

```

2.5.4 POLYM and RPOLYM, Polynomial Multiplication Routines

This routine multiplies polynomial A by polynomial B and stores the result in polynomial C. All polynomials are stored in linear arrays. The listing shown is for complex polynomials (POLYM). The routine can be modified to handle other types of variables by changing the explicit type statement and the defining statement for the variable zero. RPOLYM is the version which multiplies two real polynomials.

CALL STATEMENT

CALL POLYM (A, NA, B, NB, C, NC)

1. A is a complex linear array used as one input.
2. NA is the degree of polynomial A, integer.
3. B is a complex linear array used as the second input.
4. NB is the degree of polynomial B, integer.
5. C is the complex linear array product polynomial returned.
6. NC is the degree of polynomial C, integer.

Note that one of the input variables can be used as an output variable. For example we can say, Call POLYM (A, NA, B, NB, B, NB). The resulting output polynomial and its degree replace the input array and its degree in the calling program.

RESTRICTION:

POLYM is restricted to multiplying two polynomials, the sum of whose degrees is less than 49. That is $(NA + NB) \leq 49$. If higher order results are required it is necessary to increase the dimension of the variable CSTORE.

```

SUBROUTINE POLYM(A,NA,B,NB,C,NC)
C   THIS SUBROUTINE MULTIPLIES POLYNOMIALS A AND B AND STORES
C   THE RESULT IN C.NA,NB,NC ARE THE DEGREES OF THEIR POLYNOMIALS
C   EITHER A OR B MAY BE THE SAME VARIABLE AS C IN THE CALL STMT, EG
C   CALL POLYM(X,NX,Y,NY,Y,NY) IS LEGAL
C   IMPLICIT COMPLEX*16 (C), REAL*8 (A,B,D-H,O-Z)
COMPLEX    CSTORE(50),A(1),B(1),C(1)
NSTO=NA+NB
NCP1=NSTO +1
ZERO = 0.E0
DO 1 I=1,NCP1
1 CSTORE(I)=ZERO
LIM1=NA+1
LIM2=NB+1
DO 2 I=1,LIM1
DO 2 J=1,LIM2
IJM1=I+J-1
2 CSTORE(IJM1)=CSTORE(IJM1)+A(I)*B(J)
DO 3 I=1,NCP1
3 C(I)=CSTORE(I)
NC= NSTO
RETURN
END

```

```

SUBROUTINE RPOLYM(A,NA,B,NB,C,NC)
C  IMPLICIT REAL*8 (A-H,O-Z)
C  THIS SUBROUTINE MULTIPLIES POLYNOMIALS A AND B AND STORES
C  THE RESULT IN C.NA,NB,NC ARE THE DEGREES OF THEIR POLYNOMIALS
DIMENSION CSTORE(50),A(1),B(1),C(1)
NSTO=NA+NB
NCP1=NSTO +1
DO 1 I=1,NCP1
1 CSTORE(I)= 0.
LIM1=NA+1
LIM2=NB+1
DO 2 I=1,LIM1
DO 2 J=1,LIM2
IJM1=I+J-1
2 CSTORE(IJM1)=CSTORE(IJM1)+A(I)*B(J)
DO 3 I=1,NCP1
3 C(I)=CSTORE(I)
NC= NSTO
RETURN
END

```

2.5.5 ADJNUM: A Routine To Scale A Polynomial

This routine scales a polynomial P

$$P = p_0 + p_1x + p_2x^2 + \dots + p_nx^n$$

by a scale factor A such that the polynomial becomes

$$P = \frac{p_0}{A} + \frac{p_1}{A}x + \frac{p_2}{A}x^2 + \dots + \frac{p_n}{A}x^n$$

where N is the degree of the polynomial

CALL STATEMENT:

CALL ADJNUM(P, N, A)

where:

1. P is a real linear array containing the coefficients of the polynomial to be scaled.
2. N is the integer degree of polynomial P.
3. A is the real variable whose value is the scale factor.

C SUBROUTINE ADJNUM(B,M,HODC)
 IMPLICIT REAL*8 (A-H,O-Z)
 DIMENSION B(1)
 MP1=M+1
 DO 7 I=1,MP1
7 B(I)=B(I)/HODC
 RETURN
 END

2.6.1 REFORM, A Routine to Form a Polynomial from the Roots of the Polynomial

This routine (re) forms a polynomial from its roots which can be of any multiplicity. In general, the roots are complex roots. It is always true that complex roots appear in conjugate pairs. A pair of complex conjugate roots form a quadratic polynomial,

$$P_j(s) = (s + \alpha_j + i\beta_j)(s + \alpha_j - i\beta_j) = s^2 + 2\alpha_j s + (\alpha_j^2 + \beta_j^2)$$

where α is the real part of the complex root pair

β is the imaginary part of the complex root pair

Note that α is always negative in value for physically reliable transfer functions.

For real roots a first order polynomial results,

$$P_j(s) = s + \alpha_j$$

The polynomial of the j th root of multiplicity M_j is

$$[P_j(s)]^{M_j}$$

The polynomial found by REFORM is then the product of the polynomials of each root,

$$P(s) = \prod_{j=1}^N [P_j(s)]^{M_j}$$

where N is the number of distinct roots of the polynomial.

CALL STATEMENT

CALL REFORM (ROOT, N, MULT, POLY, NPOLY)

where

1. ROOT is a real array containing each of the N roots of the polynomial to be found.

ROOT (1) = real part of root #1

ROOT (2) = imaginary part of root #2

.

.

ROOT (2N-1) = real part of the Nth root

ROOT (2N) = imaginary part of the Nth root

If a root is complex its conjugate must immediately follow it in the array.

If a root is real; its imaginary part must be exactly zero. REFORM has no other way of knowing whether a root is real or imaginary.

- (2) N is an integer whose value is the number of roots in array ROOT.

- (3) MULT is an integer array containing the multiplicities of each root. If the jth root is repeated once then MULT (J) = 2 and so on.

- (4) POLY is a real array which contains the (re) formed polynomial.

POLY (1) = p_0

POLY (2) = $p_1 s$

POLY (3) = $p_2 s^2$

.

.

- (5) NPOLY is the degree of the polynomial returned. There are (NPOLY + 1) coefficients returned.

```

C      SUBROUTINE REFORM(ROOT,ND,MULT,TRA,NTRA)
      IMPLICIT REAL*8 (A-H,O-Z)
      DIMENSION ROOT(1),MULT(1),TRA(1),TEM(3)
      TRA(1)=1.E0
      NTRA=0
      I=0
2     I=I+1
      IR=2*I-1
      II=2*I
      ROII=ROOT(II)
      IF(ROII.EQ.0.E0) GO TO 4
      TEM(1)=ROOT(II)*ROOT(II)+ROOT(IR)*ROOT(IR)
      TEM(2)=2.E0*(-ROOT(IR))
      TEM(3)=1.E0
      NTEM=2
      I=I+1
      GO TO 5
4     TEM(1)=-ROOT(IR)
      TEM(2)=1.E0
      NTEM=1
5     K=MULT(I)
      DO 6 J=1,K
      CALL RPOLYM(TRA,NTRA,TEM,NTEM,TRA,NTRA)
6     CONTINUE
      IF((I-ND).GE.0) GO TO 1
      GO TO 2
1     CONTINUE
      RETURN
      END

```

2.6.2 RESP, A Routine to Evaluate the Frequency Response of Two Transfer Functions

This routine evaluates the frequency response of two transfer functions at 101 equally spaced frequencies from zero to half the sampling frequency. The purpose of this routine is to print out the response of the z transform function beside the response of the original transfer function. The z transform function is of the form

$$H_z(z) = \frac{P_z(z)}{Q_z(z)} = \frac{p_0 + p_1 z^{-1} + p_2 z^{-2} + \dots + p_M z^{-M}}{q_0 + q_1 z^{-1} + q_2 z^{-2} + \dots + q_N z^{-N}}$$

Its response is evaluated by substituting

$$z = e^{i\omega T}$$

where:

ω is the variable frequency $[0 \leq \omega \leq \frac{\pi}{T}]$ - in rad./sec.

The original transfer function $H(s)$ must be inputted to the routine in the form,

$$H(s) = \frac{P(s)}{Q(s)} = \frac{p_0 + p_1 s + p_2 s^2 + \dots + s^M}{q_0 + q_1 s + q_2 s^2 + \dots + s^N}$$

This transfer function is evaluated by making the substitution

$$s = i\omega$$

This routine requires the following subroutines

1. RECRSP which evaluates the transform transfer function response.
2. TRANS which evaluates the original transfer function response.

Additional storage is required inside the routine, hence more equally spaced points can be evaluated by changing the variable NPTS and the Dimension statements for FREQ, AMPL1, AMPL2, PHAZ1, PHAZ2. For example, if it were desired to evaluate the two transfer functions at 201 equally spaced points between zero and half the sampling frequency then change the definition to NPTS = 201 and increase the above dimensioned variables to 201.

CALL STATEMENT

CALL RESP (P, NP, Q, NQ, DENOM, NDD, NUM, NDN, T, TAU)

where:

1. P is the z transform numerator - real array

$$\begin{aligned} P(1) &= p_0 \\ P(2) &= p_1 \\ &\text{etc.} \end{aligned}$$

2. NP is the degree of polynomial P - integer.

3. Q is the z transform denominator - real array

$$\begin{aligned} Q(1) &= q_0 \\ Q(2) &= q_1 \\ &\text{etc.} \end{aligned}$$

4. NQ is the degree of polynomial Q - integer.

5. DENOM is the denominator polynomial of H(s) - real array

$$\begin{aligned} \text{DENOM}(1) &= q_0 \\ \text{DENOM}(2) &= q_1 \\ &\text{etc.} \end{aligned}$$

6. NDD is the degree of polynomial DENOM - integer

7. NUM is the numerator polynomial of H(s) - real array

8. NDN is the degree of polynomial NUM - integer

9. T is the sampling period in seconds - real

10. TAU is the additional time delay introduced by the reconstruction function if

used. If the user makes the z transformation directly without calling one of the reconstruction routines he should specify TAU=0 before calling RESP. Note that TAU is a normalized time delay

$$\text{TAU} = \frac{\tau}{T}$$

where τ is the time delay in seconds
 T is the sampling period in seconds.

```

C      SUBROUTINE RESP(P,NP,Q,NQ,TRA,NTRA,TRANU,NTRANU,T,TAU)
      IMPLICIT REAL*8 (A-H,O-Z)
      DIMENSION P(1),Q(1),TRA(1),TRANU(1)
      DIMENSION TIX1(12),TIY1(12),TIY2(12),TIY3(12),TIY4(12)
      DATA TIX1/72HFREQUENCY IN CYCLES
1      DATA TIY1/72HANALOGUE FILTER GAIN
1      DATA TIY2/72HDIGITAL FILTER GAIN
1      DATA TIY3/72HANALOGUE FILTER PHASE
1      DATA TIY4/72HDIGITAL FILTER PHASE
1      DIMENSION FREQ(101),AMPL1(101),AMPL2(101),PHAZ1(101),PHAZ2(101)
      PI=3.1415927
      OMEGA=0.
      NPTS=101
      DELFR=PI/FLOAT(NPTS-1)
      WRITE(6,11)
      WRITE(6,12)
      DO 1 I=1,NPTS
      CALL RECRSP(P,NP,Q,NQ,OMEGA,AMPL1,PHI1,TAU)
      FREQ(I)=OMEGA/(T*2.*PI)
      OMEG2 = FREQ(I)*2.*PI
      CALL TRANS(OMEG2,TRANU,NTRANU,TRA,NTRA,AMP2,PHI2)
      FRE=FREQ(I)
      WRITE(6,10) FRE ,AMPL1,PHI1,AMP2,PHI2
      AMPL1(I)=AMP1
      AMPL2(I)=AMP2
      PHAZ1(I)=PHI1
      PHAZ2(I)=PHI2
      OMEGA=OMEGA+DELFR
      KKKK = -NPTS
      AMPL1(NPTS)=0.
      AMPL2(NPTS)=0.
      XMIN=FREQ(1)
      XMAX=FREQ(NPTS)
      CALL PLOT(FREQ,AMPL1,1,101,TIX1)
C      CALL QUIK3V(-2,TIX1,TIY2,KKKK,FREQ,AMPL1)
C      CALL QUIK3V(2,TIX1,TIY1,KKKK,FREQ,AMPL2)
C      CALL QUIK3L(-2,XMIN,XMAX,-180.,180.,44,TIX1,TIY4,KKKK,FREQ,PHAZ1)
C      CALL QUIK3L(2,XMIN,XMAX,-180.,180.,44,TIX1,TIY3,KKKK,FREQ,PHAZ2)
10  FORMAT(1X ,F7.2,3X,E15.8,4X,F7.2,7X,E15.8,3X,F7.2)
11  FORMAT(1H1,16X,37H FREQUENCY RESPONSE OF Z-TRAN FILTER      )
12  FORMAT(2X,4HFREQ ,4X,15HAMPLITUDE ZTRAN ,2X,11HPHASE ZTRAN ,5X,
13HREF AMPLITUDE ,4X,9HREF PHASE )
      RETURN
      END

```

2.6.3 RECRSP A Routine To Evaluate The Frequency Response Of A z Transfer Function.

This routine evaluates the amplitude and phase response of a transfer function in z at a single frequency.

$$H_z(z) = \frac{P_z(z)}{Q_z(z)} = \frac{p_0 + p_1 z^{-1} + p_2 z^{-2} + p_3 z^{-3} + \dots + p_m z^{-m}}{q_0 + q_1 z^{-1} + q_2 z^{-2} + q_3 z^{-3} + \dots + q_n z^{-n}}$$

This is accomplished by setting

$$z^m = e^{i\omega m T}$$

$$H_z(z) = \frac{P_z(z)}{Q_z(z)} = \frac{p_0 + p_1 e^{-i\omega T} + p_2 e^{-i\omega 2T} + \dots}{q_0 + q_1 e^{-i\omega T} + q_2 e^{-i\omega 2T} + \dots} = A / \phi$$

where:

$$i = \sqrt{-1}$$

ω is the radian frequency at which the transfer function is to be evaluated.

A is the amplitude response

ϕ is the phase response

T is the sampling period

This routine calls another routine, PHASE which evaluates the phase response of the transfer function

CALL STATEMENT

CALL RECRSP (P, M, Q, N, OMEGA, AMP, PHI, TAU)

where:

INPUTS

1. P is a real linear array of the z transfer function numerator coefficients.
2. M is the degree of numerator P, an integer.
3. Q is a real linear array of the z transfer function denominator coefficients.

4. N is the degree of denominator Q an integer.
5. OMEGA is the normalized radian frequency at which the transfer function is to be evaluated. That is $OMEGA = 2\pi f \frac{1}{f_s} = \omega T$, where f is the frequency in Hz at which the transfer function is to be evaluated. OMEGA can have a range of values $[0 \leq OMEGA \leq \pi]$
6. TAU is a real variable which is the time delay of the digital filter output with respect to the corresponding analogue filter output. It is the time delay which it is desired to have removed from the digital filter response. See explanation which follows.

OUTPUTS

7. AMP is a real variable which is the amplitude response of the transfer function at ωT .
8. PHI is a real variable which is the phase response of the transfer function at ωT .

An Explanation of the Time Delay Factor TAU. TAU is the time delay produced when synthesizing a digital filter, via various techniques, from an analogue prototype. When comparing the analogue transfer function with the digital transfer function it is often useful to remove the additional phase shift. Specification of TAU will cause the additional phase shift to be removed. Setting TAU to zero gives the exact digital transfer function.

For non-recursive filters TAU should be set equal to KT, where KT is the time delay from end point filtering, when removing the phase shift. K is, then, the number of sampling intervals of delay from the most recent data point being filtered and T is the sampling interval in seconds.

```

C      SUBROUTINE RECRSP(A,M,B,N,OMEGA,AMP,PHI,TAU)
      IMPLICIT REAL*8(A-H,O-Z)
      DIMENSION A(1),B(1),SING(50),COSG(50)
      CO=1.
      SO=0.
      DL=TAU*OMEGA
      C1= COS(OMEGA)
      S1= SIN(OMEGA)
      C2=2.*C1
      IF(M-N)4,4,5
4     LIM=N
      GO TO 6
5     LIM=M
6     DO 1 K=1,LIM
      COSG(K)=C1
      SING(K)=S1
      CCO=CO
      SS0=SO
      CO=C1
      SO=S1
      C1=C2*C1-CCO
      S1=C2*S1-SS0
1     CONTINUE
      RNUM=A(1)
      XNUM=0.
      DO 2 K=1,M
      AA=A(K+1)
      RNUM=RNUM+AA*COSG(K)
2     XNUM=XNUM+AA*SING(K)
      RDENOM=B(1)
      XDENOM=0.
      DO 3 K=1,N
      BB=B(K+1)
      RDENOM=RDENOM+BB*COSG(K)
3     XDENOM=XDENOM+BB*SING(K)
      COEF=RDENOM*RDENOM+XDENOM*XDENOM
      REAL=RNUM*RDENOM+XNUM*XDENOM
      XMAG=XDENOM*RNUM-XNUM*RDENOM
      REAL1=REAL* COS(DL)-XMAG* SIN(DL)
      XMAG1=REAL* SIN(DL)+XMAG* COS(DL)
      REAL=REAL1/COEF
      XMAG=XMAG1/COEF
      AMP= SQRT(REAL*REAL+XMAG*XMAG)
      PHI=PHASE(REAL,XMAG)-360.
      IF (PHI.GE.-360.0.AND.PHI.LT.-180.0)PHI=PHI+360.0
      RETURN
      END

```

2.6.4 TRANS, FORTRAN-IV Subroutine For The Calculation Of The Transfer Function Of Physically Realizable Passive Networks.

The transfer function of a network composed of lumped-parameter R, L, and C components is given by the ratio of two polynomials in S:

$$H(\omega) = \frac{\sum_{m=0}^M A_{m+1} S^m}{\sum_{n=0}^N B_{n+1} S^n}$$

where A_m and B_n are real coefficients and $S = j\omega$. The result is, in general complex such that:

$$H(\omega) = R(\omega) + j X(\omega)$$

Transfer functions are usually specified in terms of amplitude and phase:

$$|H(\omega)| = \sqrt{R^2(\omega) + X^2(\omega)}; \phi(\omega) = \tan^{-1} \frac{X(\omega)}{R(\omega)}$$

These values will be calculated for a given argument ω by using the statement:

CALL TRANS (OMEGA, A, M, B, N, AMPL, PHI)

where:

OMEGA = ω , the radian frequency

A = the array of A_m 's, the coefficients in the numerator polynomial

M = the order of the numerator polynomial

B = the array of B_n 's, the coefficients in the denominator polynomial

N = the order of the denominator polynomial

AMPL = $|H(\omega)|$

PHI = $\phi(\omega)$

It is important to note that this subroutine also calls upon two other subroutines:

SUBROUTINE EVPOLY (OMEGA, A, N, POLYR, POLYI)

and:

FUNCTION PHASE (FREAL, FIMAG)

```

C      SUBROUTINE TRANS(OMEGA,A,M,B,N,AMPL,PHI)
      IMPLICIT REAL*8(A-H,O-Z)
      DIMENSION A(1),B(1)
      CALL EVPOLY(OMEGA,A,M,PR,PP)
      CALL EVPOLY(OMEGA,B,N,QR,QI)
      REAL=PR*QR+QI*PP
      FIMAG=PP*QR-PR*QI
      FNORM=QR*QR+QI*QI
      REAL=REAL/FNORM
      FIMAG=FIMAG/FNORM
      AMPL=REAL*REAL+FIMAG*FIMAG
      AMPL= SQRT(AMPL)
      PHI=PHASE(REAL,FIMAG) - 360.0
      IF (PHI.GE.-360.0.AND.PHI.LT.-180.0)PHI=PHI+360.0
      RETURN
      END

```

2.6.5 PHASE FORTRAN IV Subroutine For The Evaluation Of Phase.

The phase of the vector representation of a complex number, $R + jX$, is given by:

$$\phi = \tan^{-1} \left(\frac{X}{R} \right)$$

If the arctangent of the entire quotient is taken, ambiguity with respect to the proper quadrant will result.

This ambiguity may be resolved by using the following FORTRAN IV function:

FUNCTION PHASE (FREAL: FIMAG)

where:

FREAL = R , the real part of the complex number

FIMAG = X , the imaginary part of the complex number

The single-valued output will be the phase in degrees, $0^\circ \leq \phi \leq 360^\circ$, where the positive phase corresponds to counter-clockwise rotation of the vector.

This function is written in single-precision FORTRAN IV.

```

C  FUNCTION PHASE(FREAL, FIMAG)
    IMPLICIT REAL*8 (A-H,O-Z)
    SMALL=1.E-12
    PI=3.1415927
    CONV=PI/180.
    IF(FIMAG)1,2,2
2  CONTINUE
    IF(FIMAG.LE.SMALL.AND.FREAL.LE.SMALL) GO TO 3
    PHASE= ATAN2(FIMAG,FREAL)/CONV
    RETURN
1  PHASE=360.+ATAN2(FIMAG,FREAL)/CONV
    RETURN
3  WRITE(6,4)
    WRITE(6,5)
    PHASE=0.
4  FORMAT(/,78H *BOTH REAL AND IMAGINARY PARTS OF TRANSFER FUNCTION W
    HERE LESS THAN 1.E-12*
5  FORMAT(20X,43H ****PHASE WAS ASSIGNED A VALUE OF ZERO****
    RETURN
    END

```

2.6.6 EVPOLY, FORTRAN IV Subroutine For The Evaluation Of A Polynomial In $S = j\omega$ With Real Coefficients.

The evaluation of a polynomial in $S = j\omega$ with real coefficients is most economically carried out by performing the series of multiplications.

$$\sum_{n=0}^N A_{n+1} S^n = S (\dots S (S (S A_{n+1} + A_n) + A_{n-2}) \dots + A_2) + A_1$$

The result of this calculation will, in general, be complex.

The polynomial will be evaluated for a given argument by using the statement:

CALL EVPOLY (OMEGA, A, N, POLYR, POLYI)

where:

OMEGA = ω , the radian frequency

A = the array of a_n 's, the real coefficients of the polynomial.

N = the order of the polynomial

POLYR = the real part of the result

POLYI = the imaginary part of the result

```

C      SUBROUTINE EVPOLY(OMEGA,A,N,POLYR,POLYI)
      IMPLICIT REAL*8(A-H,O-Z)
      DIMENSION A(1)
      IF(N)2,1,2
2     M=N+1
      POLYR=A(M)
      POLYI=0.
      DO 3 J=1,N
      X=POLYR
      K=M-J
      POLYR=A(K) -POLYI*OMEGA
      POLYI=X*OMEGA
3     CONTINUE
      GO TO 4
1     POLYR=A(1)
      POLYI=0.
4     CONTINUE
      RETURN
      END

```

3. BIZTRN, A Routine to Convert a Normalized Low-Pass Analogue Filter to Four Varieties of Digital Filters

Purpose of Routine

This routine converts an analogue prototype low-pass filter to a digital filter which is one of four types:

- 1) low pass
- 2) high pass
- 3) band pass
- 4) band reject

The analogue low pass prototype must be normalized such that its cut-off frequency is at $\omega_c = 1$ radian per second or $f_c = \omega_c / 2\pi = 1/2\pi$ Hertz as in the following figure



Digital filters produced by this routine take on the same basic characteristics as the analogue filters that can be produced by well known frequency transformations from a normalized analogue low pass.

Sharp cut-off and narrow - band digital filters require an analogue prototype with similar characteristics.

Mathematics of the Procedure

A stable digital band pass filter can be created from a stable analogue prototype $H(s)$ via the bilinear z transformation

$$s = (1 - z^{-1}) / (1 + z^{-1}) \quad (3-1)$$

where the analogue filter frequency, ω , is related to the digital filter frequency, Ω , by the relation

$$\omega = \tan(\Omega T / 2) \quad (3-2)$$

where T is the data sampling period in seconds.

The transformations that produce other analogue designs from a normalized low pass filter are

$$\begin{aligned}
 s &\leftarrow \frac{s}{\omega_u}, & \text{low-pass to low-pass} \\
 s &\leftarrow \frac{s^2 + \omega_u \omega_l}{s(\omega_u - \omega_l)}, & \text{low-pass to band-pass} \\
 s &\leftarrow \frac{s(\omega_u - \omega_l)}{s^2 + \omega_u \omega_l}, & \text{low-pass to band-reject} \\
 s &\leftarrow \frac{\omega_l}{s}, & \text{low-pass to high-pass}
 \end{aligned} \tag{3-3}$$

Upon replacing s in the transformations by the bi-linear z transformation the direct analogue low-pass to digital filter transformation

$$\begin{aligned}
 s &\leftarrow \frac{1}{\omega_u} \frac{1-z^{-1}}{1+z^{-1}}; & \text{low-pass to low-pass} \\
 s &\leftarrow \frac{(1+\omega_u \omega_l)z^{-2} - (1-\omega_u \omega_l)}{(\omega_u - \omega_l)(1-z^{-2})}; & \text{low-pass to band-pass} \\
 s &\leftarrow \frac{(\omega_u - \omega_l)(1-z^{-2})}{(1+\omega_u \omega_l)z^{-2} - (1-\omega_u \omega_l)}; & \text{low-pass to band-reject} \\
 s &\leftarrow \omega_l \frac{1-z^{-1}}{1-z^{-1}}; & \text{low-pass to high-pass}
 \end{aligned} \tag{3-4}$$

are obtained, where

$$\begin{aligned}
 \omega_u &= \tan(\omega_u T/2) \\
 \omega_l &= \tan(\omega_l T/2)
 \end{aligned} \tag{3-5}$$

and

Ω_u = upper digital cut-off frequency in radians per second,

Ω_l = lower digital cut-off frequency in radians per second,

Upon replacing s by the appropriate z transformation in $H(s)$ a digital filter

$$H_z(z) = H(s) = \frac{\sum_{m=0}^{NA} a_m z^{-m}}{\sum_{n=0}^{NB} b_n z^{-n}}, \quad (3-6)$$

is obtained. Normalization of $H_z(z)$ is usually desired so that $b_0=1$, which can be obtained by dividing the numerator and denominator of $H_z(z)$ by b_0 .

Programming Procedure

All four analogue-to-digital transformations are of the form

$$X(z) = \frac{c_0 + c_1 z^{-1} + c_2 z^{-2}}{r_0 + r_1 z^{-1} + r_2 z^{-2}} = \frac{C(z)}{R(z)} \quad (3-7)$$

Insertion of $s=X(z)$, for the desired design, into a normalized analogue low-pass filter

$$H(s) = \frac{\sum_{j=0}^{NP} p_j s^j}{\sum_{k=0}^{NQ} q_k s^k}, \quad NP < NQ, \quad (3-8)$$

produces

$$H_z(z) = \sum_{j=0}^{NP} p_j \left[\frac{C(z)}{R(z)} \right]^j / \sum_{k=0}^{NQ} q_k s^k. \quad (3-9)$$

Clearing Fractions:

$$H_z(z) = \sum_{j=0}^{NP} p_j C(z)^j R(z)^{NQ-j} / \sum_{k=0}^{NQ} q_k C(z)^k R(z)^{NQ-k} \quad (3-10)$$

The programming procedure is now evident.

1. A subroutine TYPEF is used to generate C(z) and R(z) depending on the digital band-pass filter desired specified by the integer variable ITYPE:

ITYPE = 1 = low-pass
 ITYPE = 2 = high-pass
 ITYPE = 3 = band-pass
 ITYPE = 4 = band-reject

2. A subroutine CNORD is used, utilizing polynomial manipulation, to compute the numerator or denominator of $H_z(z)$ from (3-10) above. The same subroutine is used for both numerator and denominator of the digital transfer function, but the numerator (pk), or denominator (qk), of H(s) is used as input, respectively.
3. The numerator and denominator obtained by step 2 are divided by b_0 of step 2 to obtain appropriate normalization of the digital filter.

CALL STATEMENT

To utilize the routine BIZTRN the following call statement is used:

CALL BIZTRN (P, NP, Q, NQ, OMEGAL, OMEGAU, T, ITYPE, A, NA, B, NB) where the input variables are:

P = a one dimensional real array of the numerator coefficients of H(s),

$$P(s) = \sum_{j=0}^{NP} p_j s^j$$

$$P(1) = p_0$$

$$P(2) = p_1$$

.

.

.

etc.

NP = the degree of P(s), an integer variable

Q = a one dimensional, real array of the denominator coefficients of H(s)

$$Q(s) = \sum_{k=0}^{NQ} q_k s^k$$

$$Q(1) = q_0$$

$$Q(2) = q_1$$

.

.

.

NQ = the degree of Q(s), an integer variable (NQ NP)

OMEGAL = the desired lower cut-off frequency in radians per second of the digital filter, * a real variable

OMEGAU = the desired upper cut-off frequency in radians per second of the digital filter, * a real variable

* If the filter is only low-pass (OMEGAU) or high-pass (OMEGAL), the other frequency should be set to zero.

T = the data sampling interval in seconds

ITYPE = type of filter desired, an integer variable,

1 = low-pass

2 = high-pass

3 = band-pass

4 = band-reject

and the output variables are:

with
$$H_z(z) = \frac{\sum_{m=0}^{NA} a_m z^{-m}}{\sum_{n=0}^{NB} b_n z^{-n}}$$

A = a one dimensional, real array of the numerator coefficients of $H_z(z)$

$$A(1) = a_0$$

$$A(2) = a_1$$

.

.

.

etc.

NA = the degree of A, an integer variable

B = a one dimensional, real array of the denominator coefficients of $H_z(z)$

$$B(1) = b_0 = 1$$

$$B(2) = b_1$$

.

.

.

NB = the degree of B, an integer variable

Subroutines called by BIZTRN

Subroutines called directly by BIZTRN are:

TYPEF

CNORD

Subroutines called indirectly by BIZTRN through CNORD are more general purpose real polynomial multiplication and addition routines:

RPOLYM

RPOLYA

All four subroutines must be used along with BIZTRN.

Limitations of Program

Other than arithmetic precision, the only limitation upon BIZTRN occurs in the dimensioning of auxiliary storage in.

CNORD: SA (50)

RPOLYM: CSTORE (50)

If desired SA and CSTORE may have their dimensioning changed.

```

C      SUBROUTINE BIZTRN(P,NP,Q,NQ,OMEGAL,OMEGAU,T,ITYPE,A,NA,B,NB)
C      IMPLICIT REAL*8 (A-H,O-Z)
C      DIMENSION P(1),Q(1),A(1),B(1)
C      DIMENSION C(3),R(3)
C      THIS ROUTINE TAKES A LOW PASS ANALOGUE
C      FILTER, NORMALIZED SUCH THAT ITS CUT-OFF
C      FREQUENCY OCCURS AT  $\Omega = 2\pi \text{FREQ} = 1$ 
C      RADIAN, AND GENERATES FOUR KINDS OF
C      DIGITAL FILTERS FROM THE ANALOGUE PROTOTYPE
C      **** INPUT DATA ****
C      ITYPE=1 = LOW PASS
C      ITYPE=2 = HIGH PASS
C      ITYPE=3 = BAND PASS
C      ITYPE=4 = BAND REJECT
C      OMEGAL IS THE DESIRED LOWER CUT-OFF FREQUENCY OF
C      ITYPE=2,3, OR 4 IN RADIANS/SEC
C      OMEGAU IS THE DESIRED UPPER CUT-OFF FREQUENCY OF
C      ITYPE=1,3, OR 4 IN RADIANS/SEC
C      T IS THE DATA SAMPLING INTERVAL IN SECONDS
C      P IS THE NUMERATOR COEFFICIENT ARRAY OF
C      THE ANALOGUE FILTER H(S)
C      NP IS ITS DEGREE
C      Q IS THE DENOMINATOR COEFFICIENT ARRAY OF
C      THE ANALOGUE FILTER H(S)
C      NQ IS ITS DEGREE
C      **** OUTPUT DATA ****
C      A IS THE DIGITAL FILTER NUMERATOR ARRAY
C      NA IS ITS DEGREE
C      B IS THE DIGITAL FILTER DENOMINATOR ARRAY
C      NB IS ITS DEGREE
C
C      **** BEGIN ROUTINE ****
C
C      GENERATE DIGITAL FILTER PARAMETERS PER ITYPE
C      CALL TYPEF(OMEGAL,OMEGAU,T,C,NC,R,NR,ITYPE)
C      COMPUTE DIGITAL FILTER NUMERATOR (A)
C      CALL CNORD(P,NP,R,NR,C,NC,NQ,A,NA)
C      COMPUTE DIGITAL FILTER DENOMINATOR (B)
C      CALL CNORD(Q,NQ,R,NR,C,NC,NQ,B,NB)
C      NORMALIZE DIGITAL FILTER SUCH THAT
C      B0=B(1)=1
C      BNORM=B(1)
C      LASTA=NA+1
C      DO 1 J=1,LASTA
1  A(J)=A(J)/BNORM
C      DO 2 K=1,NB
C      KK=K+1
2  B(KK)=B(KK)/BNORM
C      B(1)=1.
C      RETURN
C      END

```

```

SUBROUTINE TYPEF(OMEGAL,OMEGAU,T,C,NC,R,NR,ITYPE)
C  IMPLICIT REAL*8 (A-H,O-Z)
    DIMENSION C(3),R(3)
    OU = TAN(OMEGAU*T/2.)
    OL = TAN(OMEGAL*T/2.)
C  ITYPE=1 GENERATES PARAMETERS FOR LOW PASS
C  ITYPE=2 GENERATES PARAMETERS FOR HIGH PASS
C  ITYPE=3 GENERATES PARAMETERS FOR BAND PASS
C  ITYPE=4 GENERATES PARAMETEPS FOR BAND REJECT
    IF(ITYPE.EQ.1) GO TO 1
    IF(ITYPE.EQ.2) GO TO 2
C  HANDLE CASES 3 AND 4
    A=OU-OL
    B=1.-OU*OL
    CC=1.+OU*OL
    IF(ITYPE.EQ.4) GO TO 4
C  BAND PASS PARAMETERS (ITYPE=3)
    C(1) = CC/A
    C(2) = -2.*B/A
    C(3) = CC/A
    NC=2
    R(1) = 1.
    R(2) = 0.
    R(3) = -1.
    NR=2
    RETURN
4  CONTINUE
C  BAND-REJECT CASE
    C(1)=1.
    C(2)=0.
    C(3)=-1.
    NC=2
    R(1)=CC/A
    R(2)=-2.*B/A
    R(3)=CC/A
    NR=2
    RETURN
1  CONTINUE
C  LOW PASS PARAMETERS
    C(1)=1./OU
    C(2)=-1./OU
    NC=1
    R(1)=1.0
    R(2)=1.0
    NR=1
    RETURN
2  CONTINUE
C  2 HIGH PASS PARAMETERS
    C(1)=OL
    C(2)=OL
    NC=1
    R(1)=1.
    R(2)=-1.
    NR=1
    RETURN
END

```

```

SUBROUTINE CNORD(P,NP,R,NR,C,NC,N,A,NA)
C  IMPLICIT REAL*8 (A-H,O-Z)
C  DIMENSION SA(50),P(1),R(1),C(1),A(1)
C  N IS THE DEGREE OF THE DENOM OF H(S)
C  MN IS THE DEGREE OF NUMERATOR OR DENOMINATOR
C  OF H(S), ACCORDING TO WHICH IS BEING CALCULATED
C  FLAG=1.
C  FIRST TERM COMPUTATION
C  SA(1)=1.
C  NSA=0
C  DO 1 J=1,N
C  CALL RPOLYM(SA,NSA,R,NR,SA,NSA)
1 CONTINUE
C  DIMENSION SS(1)
C  SS(1)=P(1)
C  NSS=0
C  CALL RPOLYM(SA,NSA,SS,NSS,A,NA)
C  IF(NP.EQ.0) GO TO 6
C  DO 5 L=1,NP
C  SA(1)=1.0
C  NSA=0
C  POWERS OF C PER TERM
C  DO 2 K=1,L
C  CALL RPOLYM(SA,NSA,C,NC,SA,NSA)
2 CONTINUE
C  POWERS OF R PER TERM
C  IF (L.EQ.N) GO TO 3
C  LIM=N-L
C  DO 4 KL=1,LIM
C  CALL RPOLYM(SA,NSA,R,NR,SA,NSA)
4 CONTINUE
3 CONTINUE
C  SS(1)=P(L+1)
C  MULTIPLICATION BY TRANSFER FUNCTION COEF
C  CALL RPOLYM(SA,NSA,SS,NSS,SA,NSA)
C  ACCUMULATION OF TERMS
C  CALL RPOLYA(SA,NSA,A,NA,A,NA,FLAG)
5 CONTINUE
6 CONTINUE
C  RETURN
C  END

```

4. NON-RECURSIVE DIGITAL FILTER GENERATION ROUTINES

As mentioned in Section 1, filter coefficient generation for transfer function simulation can be done with the recursive z transformation plus reconstruction routines of Section 2 or with the non-recursive routines presented here. A guideline given in Section 1 states that if the input data spectrum extends beyond $1/3$ the sampling frequency, and it is desired to accurately simulate the transfer function to half the sampling frequency, then probably the best method of filtering is non-recursive. The advantage of non-recursive filtering is, simply, accurate simulation to half the sampling frequency.

The cost of accurately simulating a transfer function to the maximum limit (half the sampling frequency) by non-recursive filtering is often high. The disadvantages of non-recursive filtering are:

1. More filter coefficients or "weights" are often required, resulting in longer computation times to perform filtering of data.
2. Unsuitability of the equi-spaced phase and amplitude response sampling technique used by these routines when applied to transfer functions with numerator or denominator roots at very low frequencies.
3. Greater time delay of the output often results from non-recursive filtering.

If these disadvantages pose a significant difficulty it would be advisable to reconsider the need to simulate the transfer function to the maximum limit; that is, reconsider the z transformation methods plus reconstruction.

Non-recursive filter weights are generated from phase and amplitude characteristics of a transfer function rather than by an analytical transformation of a transfer function. From phase and amplitude samples a least squares approximation is computed.

A routine is available which can compute the samples from an analytical form but the transfer function must be the ratio of two polynomials.

$$\begin{aligned} H(s) &= \frac{P(s)}{Q(s)} \\ &= \frac{p_0 + p_1 s + p_2 s^2 + \dots + p_m s^m}{q_0 + q_1 s + q_2 s^2 + \dots + q_n s^n} \end{aligned}$$

If the transfer function is available only in its factored form (as was required for the z transformation method) it is necessary to form the denominator polynomial from denominator roots and multiplicities. The routine for this purpose is described in Section 2.6.1.

The routine, TRANWG, evaluates a transfer function at a specified number of points and then computes the filter weights. The weights are computed in routine, WGTCP, which may be called directly if the phase and amplitude samples are available.

It is always necessary to specify the following:

1. The number of weights desired ($2N+1$).
2. The number of phase and amplitude samples to be taken of the transfer function ($Q+1$).
3. The delay factor (ITAU) which ranges from 0 to N .

Determining these parameters, as in determining the correct reconstruction function for recursive synthesis, is often a trial and error process. A good set of rules to follow is:

1. Pick Q large enough to include all the detail of the transfer function.
2. Pick $N \approx Q/2$ or $Q/3$. In any case $Q \geq N$.
3. Initially set $ITAU = N$. It may be possible to decrease $ITAU$ to a minimum value of zero. $ITAU = 0$ is often called "end point filtering" and $ITAU = N$ is often called "central filtering."

These considerations are discussed in Vol. I, Section 5.

Since it is necessary to evaluate the frequency response of the transfer function $Q+1$ points, some thought must be given to establishing this parameter. If the spectrum is sampled at wide intervals (few samples) it is possible to miss some of the detail and the resulting digital filter will not simulate the original transfer function accurately. Consider the transfer function whose amplitude response is shown in Figure 4-1 on a semi-log plot.

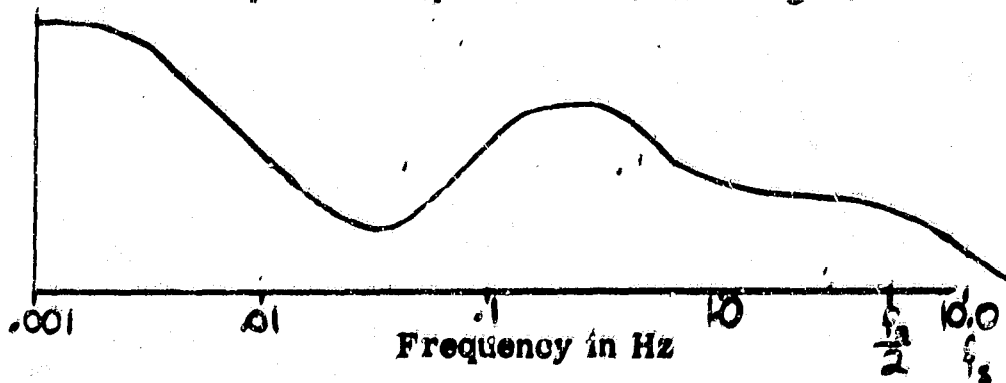


Figure 4-1 Transfer Function Amplitude vs Frequency

By sampling this spectrum at 50 equally spaced points from zero to half the sampling frequency much of the low frequency behavior would be obscured. Accurate simulation of this transfer function could be achieved by sampling in .01 Hz steps; that is 500 samples. This response is not atypical of control system transfer functions. The z transform routine should be used on this transfer function, if possible.

The last routine of this section is designed to remove the oscillations in the digital filter response which occur when the specification transfer function has fast roll-off or rise characteristics. This is a result of convergence difficulty of the method (the "Gibbs phenomena"). These oscillations are most apparent between the transfer function sample points, therefore, it is always best to examine the digital filter frequency response at more points than the $Q+1$ samples of the analogue transfer function. If these oscillations are present they can be smoothed out by the routine SMOOTH. As one might expect, there is a price to pay in using SMOOTH; that is some loss of steepness of roll-off which caused the oscillations in the first place. These ideas are discussed in Vol. 1, Section 5. The routine SMOOTH can be adjusted to apply one of three methods:

1. Lanczos weights
2. Cosine-squared weights
3. Linear or Fejer weights
4. No smoothing

Usually Lanczos or cosine-squared weights will operate most effectively to remove "ripples."

4.1 TRANWG, A Routine to Generate Non-Recursive Filter Weights from a Transfer Function

This routine computes non-recursive filter coefficients or "weights" from a transfer function expressed as a ratio of two polynomials.

$$H(s) = \frac{A(s)}{B(s)} = \frac{a_0 + a_1s + a_2s^2 + \dots + a_ms^m}{b_0 + b_1s + b_2s^2 + \dots + b_ns^n}$$

The transfer function is evaluated at $Q+1$ equally spaced points from zero to half the sampling frequency. The weights are computed from those samples in routine WGTCPY.

In addition to the input transfer function it is necessary to specify the number of samples of the transfer function to be taken ($JQ+1$), the number of weights desired ($2NL+1$), the normalized time delay of the resulting digital filter (ITAU), and the sampling period, T . The time delay factor, ITAU, can have a range of values from 0 to NL . The meaning of ITAU=0 is a minimum delay filter producing an "end point filter." ITAU= N produces what is called a "central filter". See Vol. 1, Section 5 for details.

CALL STATEMENT

CALL TRANWG (A, M, B, N, T, JQ, W, ITAU, NL, HMAG, HPHASE)

where

1. A is a real linear array containing the numerator coefficients of the transfer function.
2. M is the integer degree of the numerator polynomial.
3. B is a real linear array containing the denominator coefficients of transfer function.
4. N is the integer degree of the denominator polynomial.
5. T is a real variable whose value is the sampling period.
6. JQ is an integer with ($JQ+1$) equal to the number of samples to be taken of the transfer function.
7. W is real linear array which will contain the $2N+1$ weights.
8. ITAU is an integer whose value is the normalized time delay of the resulting non-recursive filter. It can have values from zero to NL .

9. NL is an integer value and is related to the number of weights desired. The number of weights computed is $2NL+1$.
10. HMAG is a real linear array containing the magnitude samples of the transfer function.
11. HPHASE is a real linear array containing the phase samples of the transfer function.

All arrays are dimensioned outside the subroutine in the calling program. It is necessary that the dimension for A be greater than N; B greater than M; W be greater than $2N$. HMAG, HPHASE must be dimensioned greater than JQ.

TRANWG requires the following routines:

- (1) WGTCPY
- (2) TRANS
- (3) EVPOLY
- (4) PHASE

WGTCPY is the only routine of this group which has internal storage, which may have to have its DIMENSION statement changed. The arrays CP and SP are dimensioned at 501. If JQ is greater than 500, this dimension statement would have to be increased.

```

SUBROUTINE TRANHGA(A,M,B,N,T,JQ,W,ITAU,NL,HHAG,HPHASE)
DIMENSION A(1),B(1),HHAG(1),HPHASE(1),W(1)
DATA PI/3.14159265/
OMEGA=0.0E 00
CELFR=PI/IT* FLOAT(JQ)
CONV=PI/180.0E00
NMAX=JQ+1
DO 1 I=1,NMAX
CALL TRANS(OMEGA,A,M,B,N,AMPL,PHI)
HHAG(I)=AMPL
HPHASE(I)=PHI*CONV
1 OMEGA=OMEGA+CELFR
CALL NGTCPT(NL,JQ,ITAU,HHAG,HPHASE,W)
RETURN
END

```

4.2 WGTCP, A Routine to Compute the Weights of a Non-Recursive Filter

This routine uses samples of the amplitude and phase response of a transfer function as inputs to generate non-recursive filter weights. There must be $(Q+1)$ equally spaced response samples covering the range from zero to $1/2$ the sampling frequency to produce $(2N+1)$ weights. It is necessary that $Q \geq N$ for the routine to function. In fact, it is desirable that $Q \approx 3N$ for a compromise between theoretical accuracy and actual numerical accuracy problems. If Q is very large the routine loses numerical accuracy. The calculation performed is a least squares curve fitting to the complex samples (expressed as amplitude and phase). A given weight is calculated by the formula:

$$W_r = \frac{A(0)}{2Q} \cos[\phi(0)] + \frac{1}{Q} \sum_{q=1}^{Q-1} A(w_q) \cos[w_q (ITAU-r) T - \phi(w_q)] \\ + \frac{A(w_Q)}{2Q} \cos[w_Q (ITAU-r) T - \phi(w_Q)]$$

where $A(\)$ represents the amplitude samples

$\phi(\)$ represents the phase samples

$w_q = \frac{\pi q}{QT}$ is the normalized frequency; $0 \leq w_q \leq \pi$

ITAU is a time delay factor; $0 \leq ITAU \leq N$

T is the sampling period in second.

The development and interpretation of this formula is presented in Vol. 1, Section 5.

CALL STATEMENT

CALL WGTCP (N, JQ, ITAU, HMAG, HPHASE, W)

where

- (1) N indicates the number of weights to be generated. There are $2N+1$ weights generated (integer).
- (2) JQ corresponds to Q in the explanation above. $JQ+1$ is the number of amplitude and phase samples presented (integer).

(3) ITAU is the time delay factor. ITAU=0 implies that the resulting filter has no additional output delay. This is often called "end point filtering." ITAU=N produces a filter with NT seconds of additional delay. This condition is called "central filtering." ITAU may have any integer value from 0 to N.

(4) HMAG is a real linear array containing the JQ+1 magnitude samples. It corresponds to A() function above. The first value of HMAG array, HMAG (1), contains the magnitude of the transfer function at zero frequency and so on.

$$\begin{aligned} \text{HMAG (1)} &= \left| H(\omega) \right|_{\omega=0} \\ \text{HMAG (2)} &= \left| H(\omega) \right|_{\omega=\frac{\omega_s}{2JQ}} ; \quad \omega_s = \text{sampling frequency} \\ &\vdots \\ \text{HMAG (JQ+1)} &= \left| H(\omega) \right|_{\omega=\frac{\omega_s}{2}} \end{aligned}$$

(5) HPHASE is a real linear array containing the phase samples. It corresponds to $\phi()$ in the expression above. HPHASE is ordered the same as HMAG.

(6) W is a real linear array which contains the 2N+1 values generated in the routine.

The arrays appearing in the CALL statement obtain their dimension from the calling program. Two other arrays are dimensioned in this routine;

(1) CP, (2) SP. They are dimensioned at 501 which is sufficient for JQ≤500. It is not recommended that JQ=500, however, if this is attempted CP and SP should be dimensioned to a value greater than or equal to JQ+1.

This routine does not require any other routine to function.

```

SUBROUTINE NGTCPTIN,JQ,ITAU,MMAG,MPHASE,W)
DIMENSION MMAG(1),MPHASE(1),W(1)
DIMENSION CP(501),SP(501)
DATA PI/3.14159265/
K=N-ITAU
AMPDC=MMAG(1)*COS(MPHASE(1))
DO 3 MU=1,JQ
MMU=MMU+1
PP=MPHASE(MMU)
CP(MU)=COS(PP)
3 SP(MU)=SIN(PP)
XQ=JQ
CONS=PI/XQ
LIM=2*N+1
LIMQ=JQ-1
DO 2 J=1,LIM
J1=J-1
6 JJ=LIM-J1          IS FOR HUNTSVILLE REVERSED WEIGHTS
JJ=J
V=(N-(K+J1))
ANG=CONS*V
X=COS(ANG)
X2=X*2.
CM1=X
CM0=1.
SM1=-SIN(ANG)
SM0=0.
WM=0.
DO 1 LQ=1,LIMQ
C=X2*CM0-CM1
S=X2*SM0-SM1
LLQ=LQ+1
WM=WM+MMAG(LLQ)*(CP(LLQ)*C+SP(LLQ)*S)
CM1=CM0
SM1=SM0
CM0=C
1 SM0=S
C=X2*CM0-CM1
S=X2*SM0-SM1
W(JJ)=WM/XQ+(AMPDC+MMAG(MMU)*(CP(MU)*C+SP(MU)*S))/(2.*XQ)
2 CONTINUE
IF(JQ-N) 4,5,4
5 W(1)=W(1)/2.
W(LIM)=W(LIM)/2.
4 CONTINUE
RETURN
END

```

4.3. SMOOTH, A Routine to Remove Ripples in the Response of Non-Recursive Filters

This routine smooths ripples in the amplitude and phase response of non-recursive filters. The ripples, described as the Gibbs phenomena, are the result of convergence difficulties characteristic of Fourier series representation of transfer functions having a rapid change in amplitude and phase. Sharp cutoff smoothing filters are one example of the class of transfer functions which produce these ripples. The purpose of these routine is to eliminate these ripples or reduce them to a tolerable level.

Three methods of modifying the weights to eliminate ripples are available:

1. Lanczos Weighting
2. Cosine Square Weighting
3. Linear or Fejer Weighting

In each case the original weights are multiplied by a constant factor. The value of the constant is a function of the type of smoothing, the weight being smoothed, and the time delay of the filter. Generally the Lanczos and cosine squared modified filters are the most desirable. Lanczos weighting is usually recommended as it does not attenuate the high frequency components as radically as the Cosine Squared method and also does not reduce the low frequency components as much as Fejer Weighting. It gives a happy compromise between the rise time of a band-pass filter and its ripple. These smoothing techniques are discussed in Vol. 1, Section 5.

CALL STATEMENT

CALL SMOOTH (N, ITAU, ITYPE, W)

where

- (1) N is an integer used to indicate that there are $2N+1$ weights.
- (2) ITAU is the additional normalized time delay of the filter (integer)
- (3) ITYPE is an integer flag used to indicate which smoothing technique the routine is to use. 1 is for Lanczos, 2 is for cosine squared, 3 is for Fejer, and 0 indicates that the smoothing procedure is to be bypassed. That is, if ITYPE = 0 the routine returns to the calling program.

(4) *W* is a real linear array containing $2 \times N+1$ weights. *W* contains modified weights upon return to the calling program, the old values are replaced by new values.

This routine does not call any other routine and contains only one array, *W* which obtains its dimension from the calling program.

```

SUBROUTINE SMOOTH(N, ITAU, ITYPE, W)
  DIMENSION W(1)
  IF (ITYPE) 26, 25, 24
26  E= 1.E-02
    K=N-ITAU
    RK=K
    X= ABS(RK)
    Y=N
    Y=Y+X
    LIM=2*N+1
    PI=3.1415927
    DO 21 J=1, LIM
      VJ=J-1
      ARG= ABS(Y-VJ)
      IF (ARG-E) 21, 21, 23
23  GO TO (1, 2, 3), ITYPE
24  CONTINUE
      JJ=LIM+1-J
      W(JJ)=W(JJ)*Z
      W(J)=W(J)*Z
                                IS FOR HUNTSVILLE REVERSED HEIGHTS
21  CONTINUE
25  CONTINUE
    RETURN
    LANCZOS WEIGHTING
    1 CONTINUE
      WW=(PI*ARG)/(Y+1)
      Z= SIN(WW)/WW
      GO TO 24
    COSINE SQUARED WEIGHTING
    2 CONTINUE
      WW=(PI*ARG)/(2*(Y+1))
      Z= COS(WW)**2
      GO TO 24
    LINEAR OR FEJER WEIGHTING
    3 CONTINUE
      Z=(Y+1-ARG)/(Y+1)
      GO TO 24
    END

```

5. APPLICATIONS OF DIGITAL FILTERS

The preceding sections of this report covered various methods of synthesizing digital filter coefficients. This section covers their application to actual digital filtering and processing.

It is assumed here that the digital filter coefficients have been found and have been verified by steady-state amplitude and phase response versus frequency testing. The programs which use the digital filter coefficients or "weights" to "process" sampled data are outlined.

Having established that linear, constant coefficient, digital filters are of one of two types; recursive digital filters and non-recursive digital filters it is possible to use the same numerical integration programs for both types. The non-recursive filter realization formula for both types relates the present filter output, y_k , to the past inputs and outputs.

$$y_k = \frac{1}{b_0} \left[\sum_{m=0}^M a_m x_{k-m} - \sum_{n=1}^N b_n y_{k-n} \right]$$

where a_m is the m th filter numerator coefficient
 b_n is the n th filter denominator coefficient

M and N are the degrees of the numerator and denominator polynomials, respectively, of the z transform transfer function

The distinction between recursive and non-recursive filters is that the non-recursive filter does not use past outputs while the recursive filter uses both past outputs and past inputs. Hence the correct form for implementing non-recursive filters with the programs which follow is to set $b_0 = 1.0$ and set the degree of the denominator polynomial, N , equal to zero. The weights are the a_m 's above and the degree of the weight polynomial, M , is the number of weights minus one. ($M = 2NL$)

This fundamental computation is made by the filter routines RECF and RECF2. RECF is useful for implementing single filters while RECF2 is designed to handle a large number of filters whose coefficients are stored in two large arrays; one for the numerator coefficients and the other for the denominator coefficients. A use for RECF2 might be implementing a bank of bandpass filters for spectrum analysis. Rather than storing each filter numerator and denominator polynomial in a separate array (with its own distinct name) it is easier to use two arrays, referring to a specific set of filter coefficients by an index.

A special type of filter covered here is the moving filter. It performs the same function as a bandpass filter whose center frequency can be moved as a function of time. It can be moved incrementally, in discrete steps in time or it can remain stationary. A polynomial is used to control the movement of the center frequency. The program name for the moving filter is SRECF. Applications for the routine might include sweeping through a signal spectrum periodically as a crude spectrum analyzer and tracking a single sinusoidal "tone" whose frequency as a function of time is known.

PRECEDING PAGE BLANK NOT FILMED.

5.1 RECF AND RECF2 Function Routines to Perform Digital Filtering

These two routines are the fundamental digital filtering routines. They perform the computations required to produce a filter output when given; 1. an input, 2. necessary past input and output history and, 3. the filter coefficients. Since this information is presented to the routine via call variables from the main program all filters used in a program can share a single routine.

The computation performed to find the present filter output is

$$y_j = \frac{1}{q_0} \left[\sum_{n=1}^n q_n y_{j-n} + \sum_{m=0}^m p_m x_{j-m} \right]$$

where x_{j-m} 's are filter inputs

y_{j-n} 's are filter outputs

CALL STATEMENT

FILOUT = RECF (P, NP, Q, NQ, X, Y, Z)

- (1) P is a real linear array containing the filter numerator coefficients.
- (2) NP is the degree of the numerator polynomial, integer.
- (3) Q is a real linear array containing the filter denominator coefficients.
- (4) NQ is the degree of the denominator polynomial, integer.
- (5) X is a real array containing the NP+1 past filter inputs.
- (6) Y is a real array containing the NQ+1 past filter outputs. Both X and Y arrays should either be initialized to zero at the start of a record or if initial conditions are known they should be assigned in X and Y.
- (7) Z is the present input.

For function RECF2 all of the above variables are the same except that the arrays contain the coefficients and past histories of many filters. Reference to a specific filter is made by an index, I. The call statement is

FILOUT = RECF2 (P, NPI, Q, NQI, X, Y, Z, I, J)

(8) I is the filter coefficient reference index, an integer variable which has values $I = 1, 2, 3, \dots, NF$. NF is the number of filters.

(9) J is the filter storage reference index, an integer which has values $1, 2, 3, \dots, NF$.

Since I and J are reference numbers rather than storage location indices, the linear arrays P, Q, X, Y are composed of NF equal size subgroups. That is, each filter in these arrays occupies the same length of storage regardless of the actual degree of a particular filter in the group. Therefore, the length of storage in P, Q, X, Y for each filter is the length required for the largest filter in the group. Consider three filters whose transfer functions are

$$H_1(z) = \frac{p_{01} + p_{11}z^{-1}}{q_{01} + q_{11}z^{-1}}$$

$$H_2(z) = \frac{p_{02} + p_{12}z^{-1} + p_{22}z^{-2} + p_{32}z^{-3} + p_{42}z^{-4}}{q_{02} + q_{12}z^{-1} + q_{22}z^{-2} + q_{22}z^{-3}}$$

$$H_3(z) = \frac{p_{03} + p_{13}z^{-1}}{q_{03} + q_{13}z^{-1} + q_{23}z^{-2}}$$

The second filter transfer function has a fourth degree numerator which is the largest of the six polynomials.

The numerator coefficients are stored in array P as

$$P(1) = p_{01}, P(2) = p_{11}, P(3) = 0, P(4) = 0, P(5) = 0.,$$

$$P(6) = p_{02}, P(7) = p_{12}, P(8) = p_{22}, P(9) = p_{32}, P(10) = p_{42},$$

etc.

The denominator coefficients are stored in array Q as

$$Q(1) = q_{01}, Q(2) = q_{11}, Q(3) = 0, Q(4) = 0.,$$

$$Q(5) = q_{02}, Q(6) = q_{12}, Q(7) = q_{22}, Q(8) = q_{23},$$

etc.

The degree of the largest numerator polynomial is 4. Hence $NP=4$. The degree of the largest denominator polynomial is, 3, hence $NQ=3$. Also it is absolutely essential that unused coefficient locations such as $Q(4)$ and $Q(5)$ in this example be specified as zero, otherwise inaccurate filtering will result.

```

FUNCTION RECF(A,M,S,N,X,Y,Z)
DIMENSION A(1),B(1),X(1),Y(1)
Z1=A(1)*Z
S=X(1)
IF(M.LE.0) GO TO 25
MP1=M+1
DO 24 K=2,MP1
U=X(K)
Z1=Z1+A(K)*S
X(K) = S
S=U
24 CONTINUE
25 CONTINUE
T=Y(1)
IF(N.LE.0) GO TO 27
NP1=N+1
DO 26 J=2,NP1
V=Y(J)
Z1=Z1-B(J)*T
Y(J)=T
T=V
26 CONTINUE
27 CONTINUE
X(1)=Z
Y(1)=Z1
RECF=Z1
RETURN
END

```

```

FUNCTION RECF2(P,MP,Q,NQ,XD,YD,X,I,J)
C TWO DIMENSIONAL ARRAY IS USED HOWEVER INDEXES ARE COMPUTED
C DIMENSION P(1),Q(1),XD(1),YD(1)
C FRONT INDEX OF ARRAYS CORRESPONDS TO PART OF INDIVIDUAL
C FILTERING OPERATION
C IF A SINGLE LINEAR ARRAY IN THE @CALL@IT CORRESPONDS TO
C THE FRONT INDEX ,P(K)=P(K,1)
C I=FILTER INDEX (BACK INDEX)
C J=DATA STORAGE INDEX (BACK INDEX)
NP1=NP+1
NQ1=NQ+1
IDXF=NP1*(I-1)+1
IDXS=NP1*(J-1)+1
Z1=P(IDXF)*X
S1=XD(IDXS)
IF(NP1.LE.1) GO TO 2
DO 1 K=1,MP
IX=K+IDXS
U=XD(IX)
Z1=Z1+P(K+IDXF)*S1
XO(IX)=S1
1 S1=U
2 XO(IDXS)=X
IDXF=NQ1*(I-1)+1
IDXS=NQ1*(J-1)+1
TT=YD(IDXS)
IF(NQ1.LE.1) GO TO 4
DO 3 K=1,NQ
IX=K+IDXS
V=YD(IX)
Z1=Z1-Q(K+IDXF)*TT
YD(IX)=TT
3 TT=V
4 YD(IDXS)=Z1
RECF2=Z1
RETURN
END

```

5.2 SRECF, A Function Routine which Produces a Moving Bandpass Filter

This function routine produces a bandpass filter whose center frequency can be moved as a function of time. Center frequency movement is specified polynomial function.

$$\text{SPFUN} = w_0 + w_1 t + w_2 t^2 + \dots + w_n t^n$$

where w_0 is the initial or starting center frequency in radians/second

$w_1 \dots w_n$ are coefficients for higher order frequency

increment terms in radians/second

t is time in seconds

N is the degree of the specification polynomial

The characteristic response shape of the bandpass filter is that of an $N=3$ Butterworth bandpass function. It has a fairly flat bandpass characteristic and good attenuation of out-of-band frequencies. The bandwidth of the filter is set by a specification parameter when the function routine is first called and can not be modified in future "Calls". It may be incorporated in the routine with some minor modifications including the addition of another flag variable in the call statement.

The center frequency function specification polynomial, however, can be changed at any time. If the function is changed it is necessary to set a flag (ICGE) to 1 to indicate that a change was made in the frequency specification function. This flag is reset to zero by the routine after the changes have been made. A second flag, (CLER) clears out all past filter inputs and outputs when set to 1. This action is tantamount to starting the bandpass filter at time zero, hence sufficient time should be allowed for the filter output to build up again after clearing. After the clearing operation this flag is also reset to zero. Clearing of the filter should be done whenever the center frequency is moved a great distance in one step or when a new input record is presented to the moving filter. The time required for the filter output to build up to 90% of its final value after clearing is

$$t \approx \frac{6}{\text{BW}}, \text{ seconds}$$

where:

BW is the filter bandwidth in radians/second

A discussion of the theoretical basis for the moving bandpass filter can be found in Vol. 1, Section 15.

CALL STATEMENT

Y = SRECF (SPFUN, NFUN, ICGE, FILBW, XIN, ICLER, T)

where

- (1) **SPFUN** is a real linear array of the specification polynomial.
- (2) **NFUN** is the degree of the specification polynomial, an integer.
- (3) **ICGE** is an integer flag, usually set to zero but which when set to 1 or more indicates that the specification function has been changed.
- (4) **FILBW** is the bandwidth of the moving bandpass filter in radians/second, a real variable.
- (5) **XIN** is the input data point being filtered, a real variable.
- (6) **ICLER** is an integer flag used to clear the bandpass filter. Clearing occurs if the flag is set to one or more, otherwise clearing is not performed.
- (7) **T** is the sampling period in seconds, a real variable.

BIZTRN and its subroutines are required to perform the design of the moving bandpass filter. The filter design is performed the first time the routine is called, hence the bandpass filter can not be changed thereafter. The $N=3$ Butterworth design used can be changed by putting different coefficients in the **DATA** statement for array **QS** which is the denominator of the normalized transfer function. Other Butterworth denominators can be found in *Introduction to Modern Network Synthesis* by M. E. Valkenburg, John Wiley & Sons, Inc., Publisher.

Suppose it is desired to linearly move the center frequency of the bandpass filter from 3 radian/second to 5 radians per second in 100 seconds. Since the center frequency is moved linearly with time the polynomial is first order.

Then,

$$NFUN = 1$$

The initial frequency is 3 radians/second, hence

$$SPFUN(1) = 3.0$$

The rate of change of frequency with time is

$$\frac{\Delta \omega}{\Delta t} = \frac{5-3}{100} = .02 \frac{\text{radians}}{(\text{sec})^2}$$

or

$$SPFUN(2) = .02$$

There are not to be any changes in the function during the 100 second run hence,

$$ICGE = 0$$

$$ICLER = 0$$

Note that there is a maximum rate at which the filter can be moved. If this rate is exceeded the filter output will be mainly transients and will not represent the spectral composition the input signal. The maximum rate is,

$$\frac{\Delta\omega}{\Delta t} \ll \frac{BW}{2T}$$

where

BW = moving filter bandwidth in radians/second

T = filter response time in seconds for example see "Transient Responses of conventional Filters," Henderson and Kavtz pp 334-337, IRE Transactions on Circuit Theory, Vol. CT-5, Dec. '58

```

FUNCTION SRECF(SPFUN,NFUN,ICGE,FILBW,XIN,ICLER,T)
DIMENSION SPFUN(1),P(10),Q(10),X01(10),Y01(10),X02(10),Y02(10)
DIMENSION QS(10),PS(10)
DATA IFIRST/1/,QS/1.0,2.0,2.0,1.0,0.0/,NQS/3/
DATA PS/1.0,9*0.0/,NPS/0/,ZERO/0.0/,P12/6.283185/,ITYPE/1/
IF(IFIRST)1,1,2
2 IFIRST=0
ICLER=1
ICGE=1
OMU=P12*FILBW/2.0
CALL B1ZTMIPS,MPS,QS,NQS,ZERO,OMU,T,ITYPE,P,NP,U,MU)
1 IF(ICLER)3,3,4
4 DO 5 I=1,10
X01(I)=ZERO
Y01(I)=ZERO
X02(I)=ZERO
5 Y02(I)=ZERO
ICLER=0
3 IF(ICGE)6,6,7
7 ICGE=0
TIME=ZERO
6 OMT=SPFUN(1)
IF(NFUN.LE.0) GO TO 9
NF1=NFUN+1
TIM=TIME
DO 8 I=2,NF1
OMT=OMT+SPFUN(I)*TIM
8 TIM=TIME+TIM
9 OMT=OMT+TIME
TIME=TIME+T
C OPERATIONAL PORTION OF MOVING BANDPASS FILTER
SK= SIN(OMT)
CK= COS(OMT)
XM1 = XIN*SK
XM2 = XIN*CK
XL1=RECF(P,NP,Q,MQ,X01,Y01,XM1)
XL2=RECF(P,NP,U,MU,X02,Y02,XM2)
SRECF=SK*XL1 + CK*XL2
RETURN
END

```