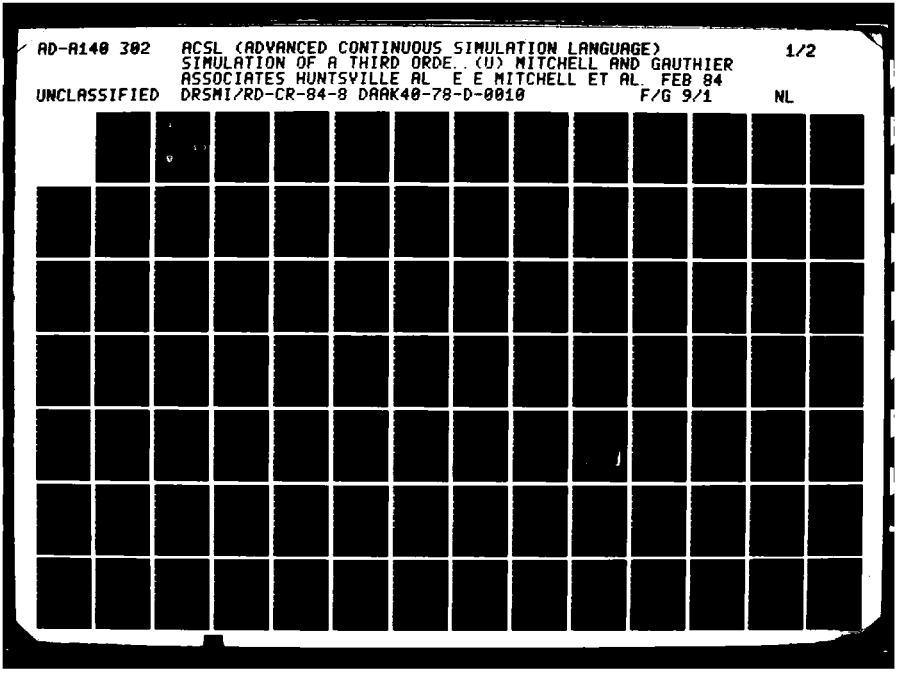
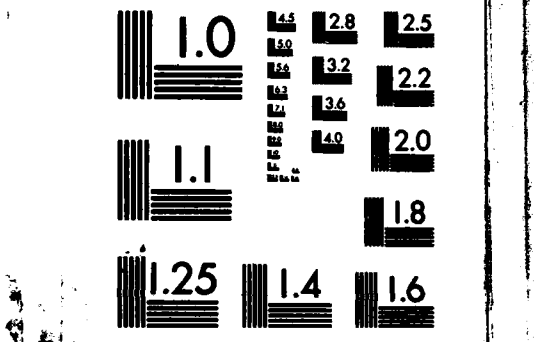
AD-A140 302 ACSL (ADVANCED CONTINUOUS SIMULATION LANGUAGE) SIMULATION OF A THIRD ORDE.. (U) MITCHELL AND GAUTHIER ASSOCIATES HUNTSVILLE AL E E MITCHELL ET AL. FEB 84 1/2

UNCLASSIFIED DRSMI/RD-CR-84-8 DAAK40-78-D-0010 F/G 9/1 NL





(12)

TECHNICAL REPORT RD-CR-84-8

ACSL SIMULATION OF A THIRD ORDER BUTTERWORTH FILTER USING IMPLEMENTATIONS DERIVED FROM BOTH Z AND S DOMAIN REPRESENTATIONS

Edward E. L. Mitchell
Mitchell and Gauthier Associates
Huntsville, Alabama

David B. Merriman
US Army Missile Laboratory

FEBRUARY 1984

Approved for public release; distribution unlimited.

U.S. ARMY MISSILE COMMAND
Redstone Arsenal, Alabama 35898

Prepared for Systems Simulation and Development Directorate
US Army Missile Laboratory

Contract DAAK40-78-D-0010

DISPOSITION INSTRUCTIONS

DESTROY THIS REPORT WHEN IT IS NO LONGER NEEDED. DO NOT RETURN IT TO THE ORIGINATOR.

DISCLAIMER

THE FINDINGS IN THIS REPORT ARE NOT TO BE CONSTRUED AS AN OFFICIAL DEPARTMENT OF THE ARMY POSITION UNLESS SO DESIGNATED BY OTHER AUTHORIZED DOCUMENTS.

TRADE NAMES

USE OF TRADE NAMES OR MANUFACTURERS IN THIS REPORT DOES NOT CONSTITUTE AN OFFICIAL ENDORSEMENT OR APPROVAL OF THE USE OF SUCH COMMERCIAL HARDWARE OR SOFTWARE.

UNCLASSIFIED

SECURITY CLASSIFICATION OF THIS PAGE (When Data Entered)

REPORT DOCUMENTATION PAGE		READ INSTRUCTIONS BEFORE COMPLETING FORM
1. REPORT NUMBER RD-CR-84-8	2. GOVT ACCESSION NO. ADA140302	3. RECIPIENT'S CATALOG NUMBER
4. TITLE (and Subtitle) ACSL SIMULATION OF A THIRD ORDER BUTTERWORTH FILTER USING IMPLEMENTATIONS DERIVED FROM BOTH Z AND S DOMAIN REPRESENTATIONS		5. TYPE OF REPORT & PERIOD COVERED 6. PERFORMING ORG. REPORT NUMBER
7. AUTHOR(s) Edward E. L. Mitchell David B. Merriman		8. CONTRACT OR GRANT NUMBER(s) DAAK40-78-D-0010
9. PERFORMING ORGANIZATION NAME AND ADDRESS Mitchell and Gauthier Associates Huntsville, AL		10. PROGRAM ELEMENT, PROJECT, TASK AREA & WORK UNIT NUMBERS
11. CONTROLLING OFFICE NAME AND ADDRESS Commander, US Army Missile Command ATTN: DRSMI-RPT Redstone Arsenal, AL 35898		12. REPORT DATE February 1984 13. NUMBER OF PAGES 105
14. MONITORING AGENCY NAME & ADDRESS(if different from Controlling Office)		15. SECURITY CLASS. (of this report) Unclassified 15a. DECLASSIFICATION/DOWNGRADING SCHEDULE

16. DISTRIBUTION STATEMENT (of this Report)

Approved for public release; distribution unlimited.

17. DISTRIBUTION STATEMENT (of the abstract entered in Block 20, if different from Report)

18. SUPPLEMENTARY NOTES

19. KEY WORDS (Continue on reverse side if necessary and identify by block number)

ACSL
Butterworth filter
Cyber 74 Intercom
Tektronic

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

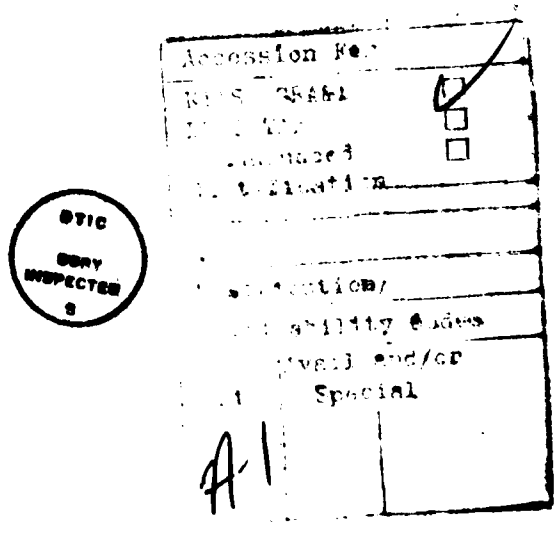
This report describes the implementation of a Third Order Butterworth filter in the ACSL. Observations and comparisons are made with inputs and outputs. Both plots of the S and Z transfer functions were made with an ACSL program. The program and Bode plots are both described in this report.
(author)

DD FORM 1 JAN 73 1473 EDITION OF 1 NOV 65 IS OBSOLETE

UNCLASSIFIED

SECURITY CLASSIFICATION OF THIS PAGE (When Data Entered)

CONTENTS



ILLUSTRATIONS

ILLUSTRATIONS (CONTINUED)

ILLUSTRATIONS (CONTINUED)

ILLUSTRATIONS (CONCLUDED)

1. INTRODUCTION

This report describes the implementation of a Third order Butterworth filter in the Advanced Continuous Simulation Language (ACSL) [1]. Transient responses of the filter in the time domain, where the time domain expressions were derived from both s and z domain representations, are observed and compared with the responses from the analytical time domain expression for the filter. Inputs are step and sine wave functions. There are two z domain implementations used: the bilinear transformation method and the impulse invariant method [2]. These are both based on transforming the continuous representation of the filter in the s domain to the z domain.

An ACSL macro for implementing Z-transfer functions was developed and used in the transient response testing. The macro algorithm is based on the "M-Method" canonical form of the transfer function. Only zero initial conditions are allowed for in the algorithm.

Bode plots of the s and Z transfer functions were made with an ACSL program. The program and Bode plots are both described in this report.

2. MATH MODELS FOR A THIRD ORDER BUTTERWORTH FILTER IN THE Z AND s DOMAINS

A third order Butterworth filter in its normalized s domain form looks as follows

$$\frac{1}{s^3 + 2s^2 + 2s + 1} \tag{1}$$

or,

$$\frac{1}{(s + 1)\ (s^2 + s + 1)}$$

If the cutoff frequency ω_0 is shown, the filter takes the form

$$\frac{1}{\left(\frac{s}{\omega_0} + 1\right)\ \left(\frac{s^2}{\omega_0^2} + \frac{s}{\omega_0} + 1\right)} \tag{2}$$

The s domain representation of the output of the filter for a step input of unity gain is

$$\frac{\omega_o^3}{s\ (s + \omega_o)\ (s^2 + \omega_o s + \omega_o^2)}$$

This can also be written as

$$\frac{1}{s} - \frac{1}{s + \omega_o} - \frac{\omega_o}{s^2 + \omega_o s + \omega_o^2}$$

which is easily transferred to its equivalent in the time domain,

$$u(t) - e^{-\omega_o t} - \frac{2}{\sqrt{3}} e^{\left(-\frac{\omega_o t}{2}\right)} \sin\left(\frac{\sqrt{3}}{2}\omega_o t\right) \tag{3}$$

where u(t) is a step function with unity gain. Similarly for a unity gain sine wave of frequency γ, the time domain output of the filter is

$$A \cos \gamma t + \frac{B}{\gamma} \sin \gamma t + Ce^{-\omega_o t} - D \frac{2}{\sqrt{3}} e^{\frac{-\omega_o}{2} t} \sin \left(\frac{\sqrt{3}}{2}\omega_o t - \frac{\pi}{3}\right) + \frac{E}{\omega_o} e^{\frac{-\omega_o}{2} t} \sin(\omega_o t) \tag{4}$$

where

$$A = \omega_o^3 \gamma \frac{\gamma^2 - 2\omega_o^2}{\gamma^6 + \omega_o^6}$$

$$B = \omega_o^4 \gamma \frac{\omega_o^2 - 2\gamma^2}{\gamma^6 + \omega_o^6}$$

$$C = \frac{\omega_o \gamma}{\omega_o^2 + \gamma^2}$$

$$D = \omega_o \gamma \frac{\omega_o^2 - \gamma^2}{\omega_o^4 - \omega_o^2 \gamma^2 + \gamma^4}$$

$$E = \frac{\omega_o^4 \gamma}{\omega_o^4 - \omega_o^2 \gamma^2 + \gamma^4}$$

These analytical time domain expressions will be used to compare with the s and Z domain results in the transient response simulation.

The s domain filter expression used in the simulation is, from (2),

$$\frac{\omega_o^3}{s^3 + 2\omega_o s^2 + 2\omega_o^2 s + \omega_o^3} \tag{5}$$

In the impulse invariant method of transforming a transfer function from the s domain to the Z domain, the impulse response of the transfer function is mapped directly into the Z domain. Equation (5) can be rewritten as

$$\frac{\omega_o}{s + \omega_o} - \frac{\omega_o s + 0.5\,\omega_o^2}{(s + \frac{\omega_o}{2})^2 + \frac{3}{4}\omega_o^2} + \frac{0.5\,\omega_o^2}{(s + \frac{\omega_o}{2})^2 + \frac{3}{4}\omega_o^2}$$

From tables, the Z-transfer function is

$$\frac{\omega_o}{1 - z^{-1}(u^2+v^2)} - \omega_o \frac{1 - z^{-1}u}{1 - 2z^{-1}u + z^{-2}(u^2+v^2)} + \frac{\omega_o}{\sqrt{3}} * \frac{z^{-1}v}{1 - 2z^{-1}u + z^{-2}(u^2+v^2)}$$

or

$$\frac{\omega_o \left(R_o + R_1 z^{-1} + R_2 z^{-2}\right)}{S_o + S_1 z^{-1} + S_2 z^{-2} + S_3 z^{-3}} \tag{6}$$

where

$$Z = e^{+sT}$$

T is the sampling interval

$$u = e^{-\frac{\omega_o}{2} T} \cos\left(\frac{\sqrt{3}}{2} \omega_o T\right)$$

$$\nu = e^{-\frac{\omega_o}{2} T} \sin\left(\frac{\sqrt{3}}{2} \omega_o T\right)$$

$$u^2 + \nu^2 = e^{-\omega_o T}$$

$$R_o = 0$$

$$R_1 = -u + \frac{\nu}{\sqrt{3}} + u^2 + \nu^2$$

$$R_2 = (u^2 + \nu^2)\left(1 - u - \frac{\nu}{\sqrt{3}}\right)$$

$$S_o = 1$$

$$S_1 = -(2u + u^2 + \nu^2)$$

$$S_2 = (u^2+\nu^2)(1+2u)$$

$$s_3 = -(u^2+v^2)^2$$

Upon implementing (6) one immediately notices the gain is ω_o/T rather than 1. This is due in part to the fact that

$$X(z) = \frac{1}{T} \sum_{k=-\infty}^{\infty} X\left(s + j\frac{2\pi}{T} k\right) \Bigg|_{s = \frac{\ln z}{T}}$$

where X is an arbitrary transfer function [3] and $\Sigma K\ x(s+j\frac{2\pi}{T} K)$ is the repeated spectrum (due to the sampling of X(t)) of the continuous s-transfer function. Since the continuous representation of X in the s domain repeats, one must use care in selecting T. Hopefully the frequency content of the input signal is known to aid in this determination.

In the bilinear transformation method s is directly replaced by z-1/z+1 in Equation (5). This change of variable maps the imaginary axis of the s-plane to the unit circle of the z-plane [2], ω_b is replaced by $\tan \omega_0 T/2$ in (5). The analogy in the frequency domain being $s \rightarrow j\omega$, then

$$s = > \frac{Z-1}{Z+1} = \frac{e^{sT}-1}{e^{sT}+1}$$

or

$$\left| j\omega \right| = > \left| \frac{e^{j\omega T}-1}{e^{j\omega T}+1} \right| = \left| \tan \frac{\omega T}{2} \right|$$

gives

$$\left| \omega_o \right| => \left| \tan \frac{\omega_o T}{2} \right|$$

Thus Equation (5) becomes

$$\frac{a_o + a_1 z^{-1} + a_2 z^{-2} + a_3 z^{-3}}{b_o + b_1 z^{-1} + b_2 z^{-2} + b_3 z^{-3}}$$

$$a_o = \alpha^3/\beta$$

$$a_1 = a_2 = 3a_o$$

$$a_3 = a_o$$

$$b_o = 1$$

$$b_1 = -(3+2\alpha - 2\alpha^2 - 3\alpha^3)/\beta$$

$$b_2 = +(3-2\alpha - 2\alpha^2 + 3\alpha^3)/\beta$$

$$b_3 = -(1-2\alpha + 2\alpha^2 - \alpha^3)/\beta$$

$$\alpha = \tan \frac{\omega_o T}{2}$$

$$\beta = 1 + 2\alpha + 2\alpha^2 + \alpha^3$$

Although this transfer function representation also repeats itself every $\frac{2\pi}{T}$K rad/sec, there is no degradation in the filter roll-off characteristics as will be shown in Section 3.

As in the s domain there are canonical forms for Z-transfer functions. The "M-Method" canonical implementation of the third order Butterworth filter is shown in *Figure 1*. The figure illustrates the representation for Equation (7). Equation (7) becomes

$$M = \frac{1}{b_o}(\text{INPUT} - b_1 Z^{-1} M - b_2 Z^{-2} M - b_3 Z^{-3} M) \quad (8)$$

$$\text{OUTPUT} = (a_o + a_1 Z^{-1} + a_2 Z^{-2} + a_3 Z^{-3})\ M \quad (9)$$

ACSL macro ZXSFRM was designed to implement a generalized form of the M-method that can be used for any order Z-transform whose highest power in the numerator does not exceed that of the denominator. The macro algorithm is described in Appendix A

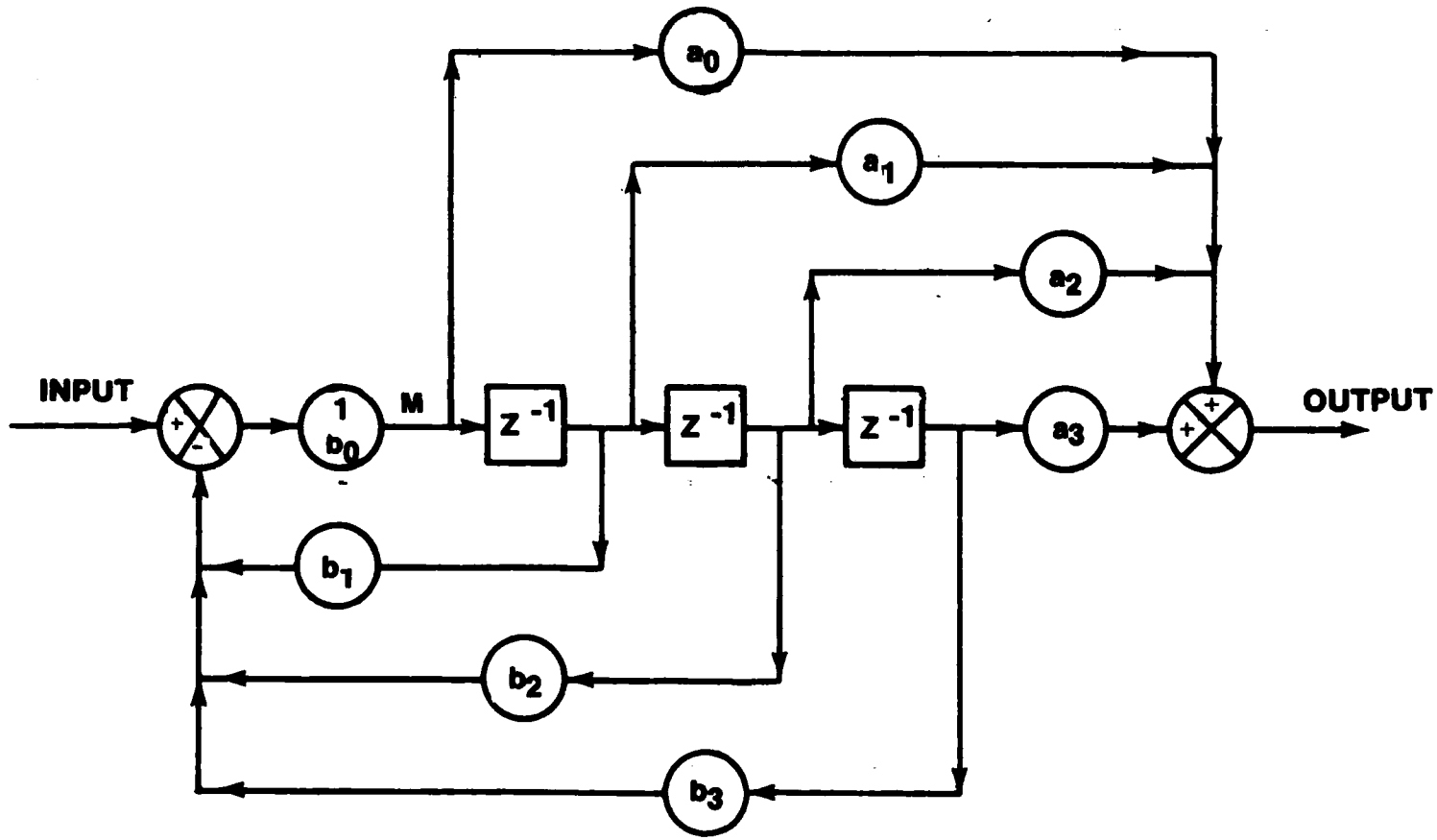


Figure 1. The M-method canonical representation of the Z-transfer function for a third order Butterworth filter.

The continuous s domain filter representation was also implemented in the time domain through the RK2 numerical integration scheme with 1/s interpreted as an integration operation. The ACSL macro TRANS [1] was used to implement the numerical integration of the filter states. TRANS contains an algorithm which generates the state equations based on the M-method canonical form shown in *Figure 2* for Equation (5). It should be noted that non-zero intitial conditions on the states of the filter are not available in TRANS or ZXSFRM.

The direct numerical integration of the s domain filter will be compared with the analytical response and the two Z domain implementations.

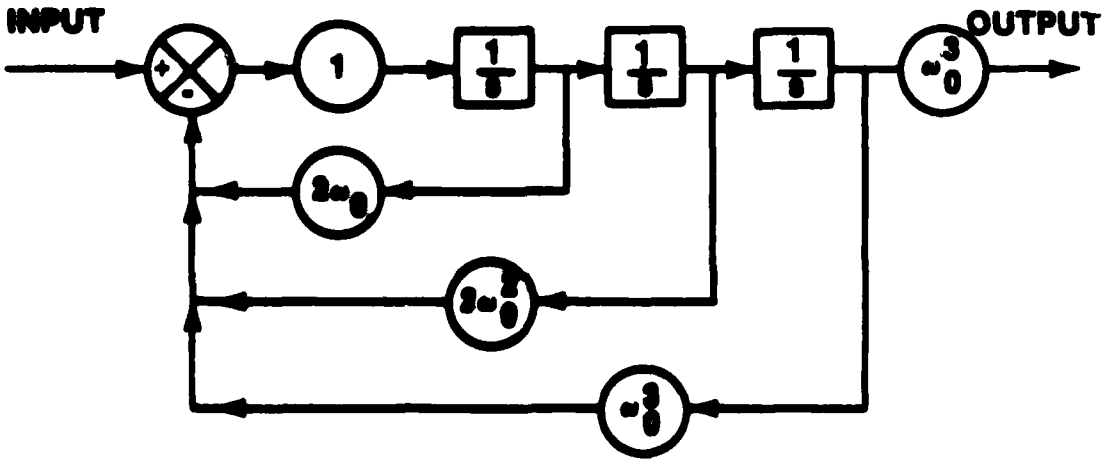


Figure 2. The M-method canonical representation of the s transfer function for a third order Butterworth filter.

3. BODE PLOT PROGRAM AND RESULTS FOR A THIRD ORDER BUTTERWORTH FILTER IN THE Z AND s DOMAINS

In order to check the numerical coefficients of the impulse invariant and bilinear transformation forms of the digital Butterworth filter, an ACSL Bode plot program was modified to produce Z-transform plots. The original algorithm utilizes an embedded representation of the polynomials which is well suited for DO loop implementation. *Figure 3* is a listing of the Bode plot program with three macros added to it for generation of the coefficients of the three filters: the s domain version, CCNTS, and the two Z domain versions, CNVRNT and CBILNR. The macros are used to calculate the coefficients for different break frequencies and sampling intervals in a form that would not clutter up what attempts to be a general purpose Bode plot program. The equations in CNVRNT, CBILNR and CCNTS are

```
      PROGRAM BODE PLOTS FOR TRANSFER FNS IN THE S AND Z DOMAINS
"                                   DAVID B. MERRIMAN                    "
"                                   12 OCT 78                            "
"                                   REDSTONE ARSENAL, AL                 "
"                                   BASED ON A PROGRAM WRITTEN BY        "
"                                   E.E.L. MITCHELL AND AN ALGORITHM GIVEN BY"
"                                   MERV BUDGE.                          "
MACRO CNVRNT(U,V,WB,DT,KG)
MACRO REDEFINE K,E,C,S
PROCEDURAL(U,V=WB,DT,KG)
E=EXP(-WB*DT)
C=SQRT(E)*COS(SQRT(3.)*0.5*WB*DT)
S=SQRT(E)*SIN(SQRT(3.)*0.5*WB*DT)/SQRT(3.)
U(3)=0.0
U(2)=WB*(-C+S+E)
U(1)=WB*E*(1.-C-S)
V(4)=1.0
V(3)=-2.*C-E
V(2)=+E*(1.+2.*C)
V(1)=-E**2
K=(U(1)+U(2))/(V(1)+V(2)+V(3)+V(4))
U(1)=U(1)/K
U(2)=U(2)/K
U(3)=U(3)/K
END
MACRO END
MACRO CRILNR(R,S,WB,DT,KG)
MACRO REDEFINE AL,DEN
PROCEDURAL(R,S=WB,DT,KG)
AL=TAN(0.5*WB*DT)
DEN=1.+2.*AL+2.*AL**2+AL**3
R(4)=AL**3/DEN
R(3)=3.*AL**3/DEN
R(2)=R(3)
R(1)=R(4)
S(4)=1.0
S(3)=(-3.-2.*AL+2.*AL**2+3.*AL**3)/DEN
S(2)=(3.-2.*AL-2.*AL**2+3.*AL**3)/DEN
S(1)=(-1.+2.*AL-2.*AL**2+AL**3)/DEN
END
MACRO END
MACRO CCNTS(A,B,WB)
A(1)=WB**3
B(1)=1.0
B(2)=2.0*WB
B(3)=2.0*WB**2
B(4)=A(1)
MACRO END
```

Figure 2. Listing of the ACSL Bode plot program.

```
            "LIST OF SYMBOLS"

"A-NUMERATOR COEFFICIENT ARRAY IN DECREASING POWERS OF S OR Z**(-1)"
"B-DENOMINATOR COEFFICIENT ARRAY IN DECREASING POWERS OF S OR Z**(-1)"
"CNTNUS-IF .T. XSFER FN IN S, IF .F. XSFER FN IN Z"
"DEN-DENOMINATOR OF G"
"G-TRANSFER FN W/A CONST GAIN (SEE KG)"
"DT-SAMPLING PERIOD FOR Z-TRANSFORM"
"GAIN-GAIN OF THE TRANSFER FUNCTION (DB)"
"KW-GEOMETRIC PROGRESSION MULTIPLYING FACTOR, W(I+1)=W(I)*KW"
"KG-OPTIONAL INPUT FOR THE CONST GAIN OF THE XSFER FN"
"M-ORDER OF THE NUMERATOR POLYNOMIAL + 1"
"N-ORDER OF THE DENOMINATOR POLYNOMIAL + 1"
"NO-TOTAL NUMBER OF FREQUENCY POINTS CALCULATED BETWEEN WMN AND WMX"
"NUM-NUMERATOR OF G"
"PHASE-PHASE OF THE TRANSFER FUNCTION (DEG)"
"LOGW-ALOG10(W)"
"MAG-CALCULATED GAIN OF THE XSFER FN"
"W-PRESENT VALUE OF FREQUENCY (RAD/SEC)"
"WCPS-PRESENT VALUE OF FREQUENCY (CPS)"
"WMN-LOWEST FREQUENCY (RAD/SEC)"
"WMX-HIGHEST FREQUENCY (RAD/SEC)"
"X-REAL PART OF G"
"Y-IMAGINARY PART OF G"

"INPUTS M, N, WMN, WMX, MPTS, A, B, DT, KG"
"OUTPUTS LOGW, W, WCPS, GAIN, PHASE, MAG, X, Y"


"----------------------------GLOBAL CONSTANTS                                "
      LOGICAL         DUMP
      CONSTANT  RMN = 1.0E-30, RMX = 1.0E30 , RADDEG=57.2957795       ...
              , PI=3.14159265, DUMP = .FALSE.
      TWOPI   = 2.0*PI
INITIAL

      REAL            A(50)    , B(50)
      INTEGER         M        , N
      LOGICAL         CNTNUS
      CONSTANT  M = 1          , N = 1         , CNTNUS = .TRUE.      ...
              , A = 50*0.0     , B = 50*0.0
      CONSTANT  DT = 0.010
      CONSTANT  KG = 1.0
"----------------------------FREQUENCY IS SWEPT IN A GEOMETRIC       ...
 PROGRESSION OVER A RANGE FROM *W MN* TO *W MX*, *MPTS* POINTS PER   ...
 PLOT AND KW IS THE MULTIPLIER BETWEEN FREQUENCIES-------------------"
      CONSTANT  WMN = 1.0     , WMX = 350.    , MPTS = 50.0
      KW      = 10.0**(ALOG10(WMX/WMN)/MPTS)
      W       = WMN
```

Figure 3. (Continued)

```
"-----------------------------SET BREAK FREQUENCY (RAD) AND CALCULATE...
                                XSFER FN COEFFICIENTS                    "
      CONSTANT  WBCPS = 7.0
      WB        = WBCPS*TWOPI
      LOGICAL         BILNR    , NVRNT
      CONSTANT  BILNR=.FALSE., NVRNT=.FALSE.
      IF(.NOT.BILNR)GO TO SN999
      CBILNR(A, B=WB, DT, KG)
SN999..IF(.NOT.NVRNT)GO TO SN998
      CNVRNT(A, B=WB, DT, KG)
SN998..CONTINUE
      IF(.NOT.CNTNUS)GO TO SN997
      CCNTS(A, B=WB)
SN997..CONTINUE
END $ "INITIAL"
DYNAMIC
"-----------------------------CALCULATE REAL AND IMAGINARY PARTS OF G  "
      IF(CNTNUS)CALL BODES
      IF(.NOT.CNTNUS)CALL BODEZ
"-----------------------------CALCULATE GAIN AND PHASE OF G            "
      MAG       = SQRT(X**2 + Y**2)
      GAIN      = 20.*ALOG10(MAG)
      PHASE     = RADDEG*ATAN2(Y,X)
"-----------------------------USE LOGARITHM OF FREQUENCY FOR PLOTTING  "
      LOGW      = ALOG10(W)
"-----------------------------MAKE FREQUENCY AVAILABLE IN HERTZ        "
      WCPS      = W/TWOPI
"-----------------------------ADVANCE FREQUENCY GEOMETRICALLY          "
      W         = KW*W
      TERMT(W.GT.WMX)
END $ "DYNAMIC"
TERMINAL
      IF(DUMP)CALL DEBUG
END $ "TERMINAL"
END $ "PROGRAM"
```

direct implementations of equations (6), (7) and (5) of Section 2, respectively. In the CNVRNT macro, instead of modifying the gain of the impulse invariant Z-transform by multiplying by T, the sampling interval (see Section 2), the gain of the transform at ω=0 was used:

$$Z = e^{sT}$$

For s=jω

$$Z = e^{\cdot jwT}$$

and at ω=0

$$Z = 1 \tag{10}$$

Hence for a generalized Z transform

$$\frac{\text{Output}}{\text{Input}} = \frac{a_o + a_1 z^{-1} + a_2 z^{-2} + \ldots}{b_o + b_1 z^{-1} + b_2 z^{-2} + \ldots}$$

the gain at ω=0 is

$$\frac{\text{Output}}{\text{Input}} = \frac{a_o + a_1 + a_2 + \ldots}{b_o + b_1 + b_2 + \ldots}$$

From Equation (6)

$$\frac{\text{Output}}{\text{Input}} = 3.161 \times 10^{3} = \text{gain}$$

when

$$\omega = 0$$

$$\omega_o = 5\text{HZ} = 10\pi \text{ rad/sec}$$

$$T = 0.01 \text{ sec}$$

Note that the gain is not $\frac{1}{T}$, instead the gain is equivalent to $\frac{\omega_0}{T}$ the break frequency (rad) divided by the sampling interval. Perhaps some insight may be gained as to the origin of the ω_0

term by observing that the impulse response of the continuous filter in the time domain also contains a gain factor of ω_0. From Equation (5), the impulse response of the continuous Butterworth filter in the time domain is,

$$\omega_o \left[e^{-\omega_o t} - e^{-\frac{\omega_o}{2}t} \cos \frac{\sqrt{3}}{2} \omega_o t + \frac{1}{\sqrt{3}} e^{-\frac{\omega_o}{2}t} \sin \frac{\sqrt{3}}{2} \omega_o t \right] \tag{11}$$

Next in *Figure 3* is a definition of the standard program variables along with a listing of the input and output variables. In the preINITIAL section, global constants are defined. These are parameters such as computer arithmetic minimum and maximum, RMN and RMX; π; the conversion factor for radians to degrees, RADDEG; and the logical variable DUMP which is used to control the debug printing of the program variables for the final communication interval.

In the INITIAL section the parameters for this particular program are defined. For the Bode plots the frequency ω is swept in a geometric progression. The plotted points will then be equally spaced when the abscissa is $\log_{10}\omega$. The multiplier between frequencies, kW, is calculated based on the minimum and maximum values for ω, WMN and WMX, and the number of plotted points desired, MPTS. ω is initialized to WMN.

The code following the determination of KW and ending with the INITIAL section is specially added for the Butterworth filter plots. The break frequency in Hz, WBCPS, is set by a CONSTANT statement and then converted to radians as represented by the variable WB. BILNR and NVRNT are defined to be logical variables which, when TRUE, cause the calculation of the bilinear Z-transform and the impulse invariant transform coefficients, respectively. If CNTNUS is TRUE the coefficients for the s domain Butterworth filter are calculated. Obviously only one of three logical variables CNTNUS, BILNR and NVRNT should be TRUE for a particular run.

The DYNAMIC section contains the code for calculating the magnitude and phase of the transfer function. The real and imaginary components of the transfer function, x and y, are calculated in FORTRAN subroutine BODES which has an entry BODEZ for Z-transfer functions. For the abscissa the log of ω is used,

```
LOGW = ALOGIO(W) .
```
(12)

ω is made available in Hz for printer output,

```
WCPS=W/TWOPI.
```

ω is incremented and the DYNAMIC section is re-executed until ω>WMX,

```
W=KW*W
TERMT (W.GT.WMX)
```

There are no integrations made so a DERIVATIVE section is not needed.

Figure 4 is a listing of subroutine BODES. A FORTRAN subroutine is used since complex arithmetic can't be handled in ACSL. A dollar sign in column 1 at the beginning of the routine causes the ACSL compiler to add the program LABELED COMMON block to this subroutine. Thus, it is not necessary to specify arguments in a subroutine call.

S, G, NUM and DEN are declared complex variables. S is the s domain variable which for steady state sine waves is

```
S=jw
```

or

```
S=CMPLX (0.0,W).
```

G is the complex gain of the transfer function. NUM and DEN are the complex values of the numerator and denominator, respectively. For Z-transfer functions

$$z^{-1} = e^{-sT} = e^{-j\omega T} = \cos \omega T - j\sin \omega T \qquad (13)$$

or

```
S=CMPLX [cos (W*DT),-SIN(W*DT)]
```

G, NUM and DEN are initialized by

```
G=CMPLX (0.,0.)
```

```
NUM=CMPLX [A(1),0.]
```

```
      SUBROUTINE BODES
S
      COMPLEX S, G, NUM, DEN
      S=CMPLX(0.0,W)
      GO TO 30
      ENTRY BODEZ
      S=CMPLX(COS(W*DT),-SIN(W*DT))
   30 G=CMPLX(0.,0.)
      NUM=CMPLX(A(1),0.)
      DEN=CMPLX(B(1),0.)
      IF(M.LE.1)GO TO 40
      DO 10 I=2,M
   10 NUM=NUM*S+A(I)
   40 CONTINUE
      IF(N.LE.1)GO TO 50
      DO 20 I=2,N
   20 DEN=DEN*S+B(I)
   50 CONTINUE
      G=KG*NUM/DEN
      Y=AIMAG(G)
      X=REAL(G)
      RETURN
      END
```

Figure 4. Listing of subroutine bodes.

DEN=CMPLX [B(1), 0.]

here the general s domain transfer function looks like

$$\frac{A(1)\ s^{m} + A(2)s^{m-1} + \ldots + A(m)s^{1} + A(m+1)}{B(1)\ s^{n} + B(2)s^{n-1} + \ldots + B(n)s^{1} + B(n+1)}$$

and the general Z domain transfer function appears as

$$\frac{A(1)Z^{-m} + A(2)Z^{-m+1} + \ldots + A(m)Z^{-1} + A(m+1)}{B(1)Z^{-n} + B(2)Z^{-n+1} + \ldots + A(n)Z^{-1} + A(n+1)}$$

The latter is just the reverse of the way the polynomials of (Z^{-1}) were previously written in Section 2. This was done in order to use the algorithm originally employed for the S polynomials.

Since the numerator polynomial can be written, using Horner's method, as

$$\{\ldots[(A(1)\ s + A(2))\cdot s + A(3)]\ s + A(4) + \ldots + A(n)\}\ s + A(n+1) \qquad (14)$$

then a DO loop can be used,

```
DO 10 I = 2, M

10 NUM = NUM *S + A(I)
```

where M is defined in the INITIAL section and the actual value used is set in the run time commands (since the bilinear transformation and impulse invariant methods give different order numerator polynomials). Similarly for the denominator the algorithm is

```
DO 20 I.= 2,N

20 DEN = DEN *S + B(I)
```

(15)

When the numerator or denominator consists of only a constant term, M=1 or N=1, then its DO loop is skipped.

```
G=KG*NUM/DEN
```

calculates the complex transfer function gain where KG is an optional user input that defaults to 1.0 if not specified. The real and imaginary parts of G, X and Y, are used by the ACSL main program. Since X and Y are named in the ACSL coded portion of the program they are listed in the program LABELED COMMON block.

The simulation was run interactively on a CYBER 74 using ACSL run time commands entered on a Tektronix console. Hard copy plots were generated along with line printer listings. Run time commands that were used every session were put on a temporary mass storage file that could be attached as a user's local file to the Tektronix terminal.

Figure 5 is a list consisting of interactive session setup and ACSL run time commands. After logging onto the CYBER 74 INTERCOM system via the Tektronix terminal the following set up statements are entered:

```
CONNECT, OUTPUT
```

specifies that the system output file, default name OUTPUT, will be displayed on the Tektronix screen.

```
ETL, 170
```

extends the execution time per statement entered to 170 octal seconds.

ATTACH, INPUT, TEK, ID=DDXXXH attaches the highest cycle (2) of TEK to the Tektronix. INPUT is the default file name for data input to the local program executing in the system.

ATTACH, LGOB, TEK, ID=DDXXXH, CY=1 attaches the simulation absolute binary file which was previously compiled in a batch job via the system control cards in *Figure 6*.

After this initial preparation LGOB is executed by entering

```
LGOB.
```

```
CONNECT,OUTPUT
ETL,170
ATTACH,INPUT,TEK,ID=DDXXXH
ATTACH,LGOB,TEK,ID=DDXXXH,CY=1
LGOB
.
.
.
SET TITLE(6)="-7 HZ, DT=10MS"
IT
SET DT=0.020,TITLE(6)="-7 HZ, DT=20MS"$IT
.
.
.
STOP
BATCH,PRINT,PRINT,3D,HDMDD
```

Figure 5. Listing of interactive session setup and ACSL run time commands.

```
HDMDD,CM77000.
ACCT...
ATTACH,MACFIL,DCMACFIL,ID=DCACSLSYS.
ATTACH,ACSL,DCACSL,ID=DCACSLSYS.
ACSL.
FTN(I=COMPILE,R=2)
REQUEST,LGOB,*PF.
MAP,OFF.
ATTACH,ACSLLIB,DCACSLLIB,ID=DCACSLSYS.
ATTACH,GRAPH,TEKTRONIX4014,ID=WTPLOT,CY=2.
LDSET,LIB=GRAPH,SUBST=ZZDRAW-TEKPLT.
LDSET,LIB=ACSLLIB,PRESET=INDEF.
LOAD,LGO.
NOGO,LGOB.
RETURN,ACSL,MACFIL,ACSLLIB.
CATALOG,LGOB,TEK,ID=DDXXXH.
EXIT.
```

Figure 6. CYBER 74 control cards to generate an absolute binary file, LGOB, of the program in Figure 3.

The ACSL Executive reads the ACSL run time commands from logical unit number 5 (by default) which is equivalent to reading the file INPUT, the contents of which are shown in *Figure 7*. Referring to *Figure 7*,

```
SET PRN=9
```

tells the simulation that the line printer data will be placed on file PRINT (logical unit number 9) instead of the default file name OUTPUT (logical unit number 6). After termination of the simulation the PRINT file is batched to a line printer.

```
SET PRNPLT=.FALSE., CALPLT=.TRUE., TTLCPL=.TRUE.
```

replaces the default printer plot routines with the Tektronix interface routines with allowance made for plot titles.

```
SET TITLE (1)= "2 JULY 79"
```

puts the date in the first word of the plot title array.

```
PREPAR T, LOGW, W, WCPS, GAIN, PHASE, MAG, X, Y
```

specifies the variables to be recorded for line printer listing.

```
PROCED ST

SET CNTNUS=.TRUE., FTSPLT=.TRUE., M=1, N=4, BILNR=.FALSE.,
NVRNT=.FALSE.

END
```

defines a PROCED which, when invoked, picks the s domain version of the Butterworth filter for Bode plotting, sets the order of the numerator and denominator polynomials and sets FTSPLT, the logical for suppressing fly back tracing, to TRUE. The intent here is to overlay plot either Z-transform method with the s domain Bode plot. Two runs, one for the s domain and one for the Z, are to be made without rewinding the data file. So when a plot is made the overlay is also automatically generated. The FTSPLT variable set TRUE ensures that a line connecting the end point of the first plot to the first point of the second plot will not appear.

```
00000000000000000000
SET PRN=9
SET PRNPLT=.FALSE., CALPLT=.TRUE., TTLCPL=.TRUE.
SET TITLE(1)="2 JUL 79"
PREPAR T, LOGW, W, WCPS, GAIN, PHASE, MAG, X, Y
PROCED ST
SET CNTNUS=.TRUE., FTSPLT=.TRUE., M=1, N=4, BILNR=.FALSE., NVRNT=.FALSE.
END
PROCED ZN
SET CNTNUS=.FALSE., M=3, N=4, BILNR=.FALSE., NVRNT=.TRUE.
END
PROCED ZB
SET CNTNUS=.FALSE., M=4, N=4, BILNR=.TRUE., NVRNT=.FALSE.
END
PROCED GO
START
PRINT "ALL"
PLOT "XAXIS"=LOGW,"XLO"=0.0,"XHI"=2.5
DISPLY WBCPS, A, B
SET NRWITG=.FALSE.
END
PROCED HO
START
DISPLY WBCPS, A, B
SET NRWITG=.TRUE.
END
PROCED IT
SET TITLE(2)="CONT VS BILNR DISCRETE BUTTERWORTH FILT"
ST
HO
ZB
GO
PLOT GAIN $ SPARE
PLOT PHASE $ SPARE
SET TITLE(2)="CONT VS INV DISCRETE BUTTERWORTH FILT"
ST
HO
ZN
GO
PLOT GAIN $ SPARE
PLOT PHASE $ SPARE
END $ "IT"
SET CMD=DIS
```

Figure 7. The ACSL standard set of run time commands residing on local file INPUT.

```
PROCED ZN

SET CNTNUS=.F., M=3, N=4, BILNR=.F., NVRNT=.T.

END
```

defines a PROCED which is used to select the impulse invariant Z-transform for Bode plotting.

```
PROCED ZB                                                    (16)

SET CNTNUS=.F.,M=4, N=4, BILNR=.T., NVRN=.F.

END
```

defines a PROCED that causes the bilinear Z-transform to be used for Bode plotting.

```
PROCED GO

START

PRINT 'ALL'

PLOT 'XAXIS'=LOGW, 'XLO' =0.0, 'XHI'=2.5

DISPLY WBCPS, A, B

SET NRWITG=.F.

END
```

defines a PROCED that executes the simulation, prints data as described by the PREPAR statement, describes the abscissa for the Bode plots and displays the break frequency and coefficients for the transfer function numerator and denominator polynomials. Setting NRWITG to FALSE causes the file storing the PREPAR list variables not to rewind between runs thereby accumulating all of the data from the runs as a unit to be printed and plotted together.

```
PROCED HO

START

DISPLY WBCPS, A, B

SET NRWITG=.T.

END
```

is a PROCED that functions much the same as 'PROCED GO' but it turns the NRWITG flag to TRUE.

```
PROCED IT
SET TITLE (Z)="CONT VS BILNR DISCRETE BUTTERWORTH FILT"
ST
HO
ZB
GO
PLOT GAIN $ SPARE
PLOT PHASE $ SPARE
SET TITLE (Z)="CONT VS INV DISCRETE BUTTERWORTH FILT"
ST
HO
ZN
GO
PLOT GAIN $ SPARE
PLOT PHASE $ SPARE
END $ "IT"
```

is a PROCED that, when invoked, overlays plots for the bilinear Z-transform and s transfer function and for the impulse invariant Z-transform and s transfer function. The SPARE commands following the PLOT commands automatically generate hard copies of the Bode plots.

The following lists the code for FORTRAN subroutine SPARE,

```
SUBROUTINE SPARE
CALL HDCOPY
```

```
RETURN
END
```

HDCOPY is a special Tektronix plot software routine which automatically causes the hardcopying of whatever is on the Tektronix terminal CRT. The name SPARE in itself is a special name that ACSL recognizes. SPARE can uniquely be used for the execution of any routine named SPARE during run time.

```
SET CMD=DIS
```

indicates to the ACSL Executive that run time commands will now be input from the DISPLY file whose logical unit number is DIS. DIS is by default 6, which is also the number for the OUTPUT file. Thus run time commands are now expected to come from the Tektronix terminal.

Referring to *Figure 5* the following commands are now entered on the Tektronix display,

```
SET TITLE(6)="-7 Hz, DT=10MS"
IT
```

adds to the plot title the break frequency value and the sampling interval. The values shown are the program default values. PROCED IT is then invoked and automatically generates the plots shown in *Figures 8* through *11*. The printer listing (not shown) shows that 7 Hz is approximately the -3 db point for both the bilinear Z-transform and the s transfer function. However the drop in gain afterwards is quite dramatic for the bilinear method as shown in *Figure 8* where GAIN is in hundreds of db. The abscissa range in frequency value from 1 to 50 Hz (LOGW=$\log_{10}\omega$, ω in radians). At 50 Hz the bilinear method has a gain of -150 db and the s transfer function is at -51 db. *Figure 9* illustrates the good phase (in degrees) comparison between the two methods. The maximum deviation is approximately 18 degrees at 50 Hz.

As shown in *Figure 10* the impulse invariant method closely follows the s transfer function. At 50 Hz the gain of the impulse invariant transform is -56 db, only a 5 db difference from the s-transfer function. However in *Figure 11* one can see the phase is quite a bit off at the higher frequencies. At 50 Hz the continuous filter has a phase of 106 degrees (actually -254 degrees, the ATAN2 routine's angle limits are $\pm\pi$ radians) and the digital filter is at 175 degrees. At 40 Hz there is approximately a 10 degree difference in phase.

SET DT=0.020, TITLE (6)="-7 Hz, DT=20MS"$IT sets the sampling interval at 20 ms and generates the Bode plots in *Figures 12* through *15*. The frequency spectrum repetition is

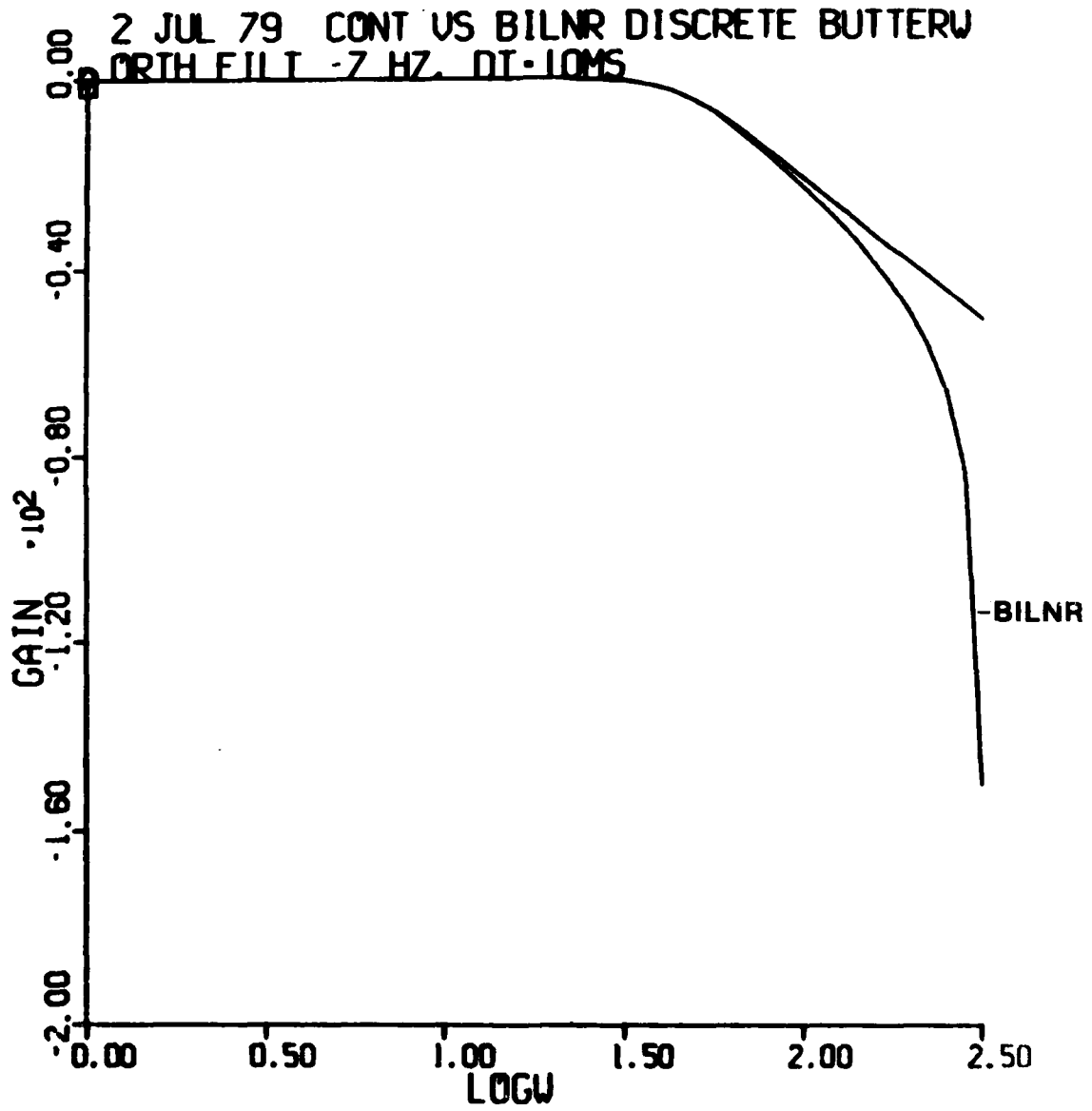


Figure 8. Gain versus frequency comparison of the s domain and bilinear Z domain filter representations at a 10 msec sampling interval.

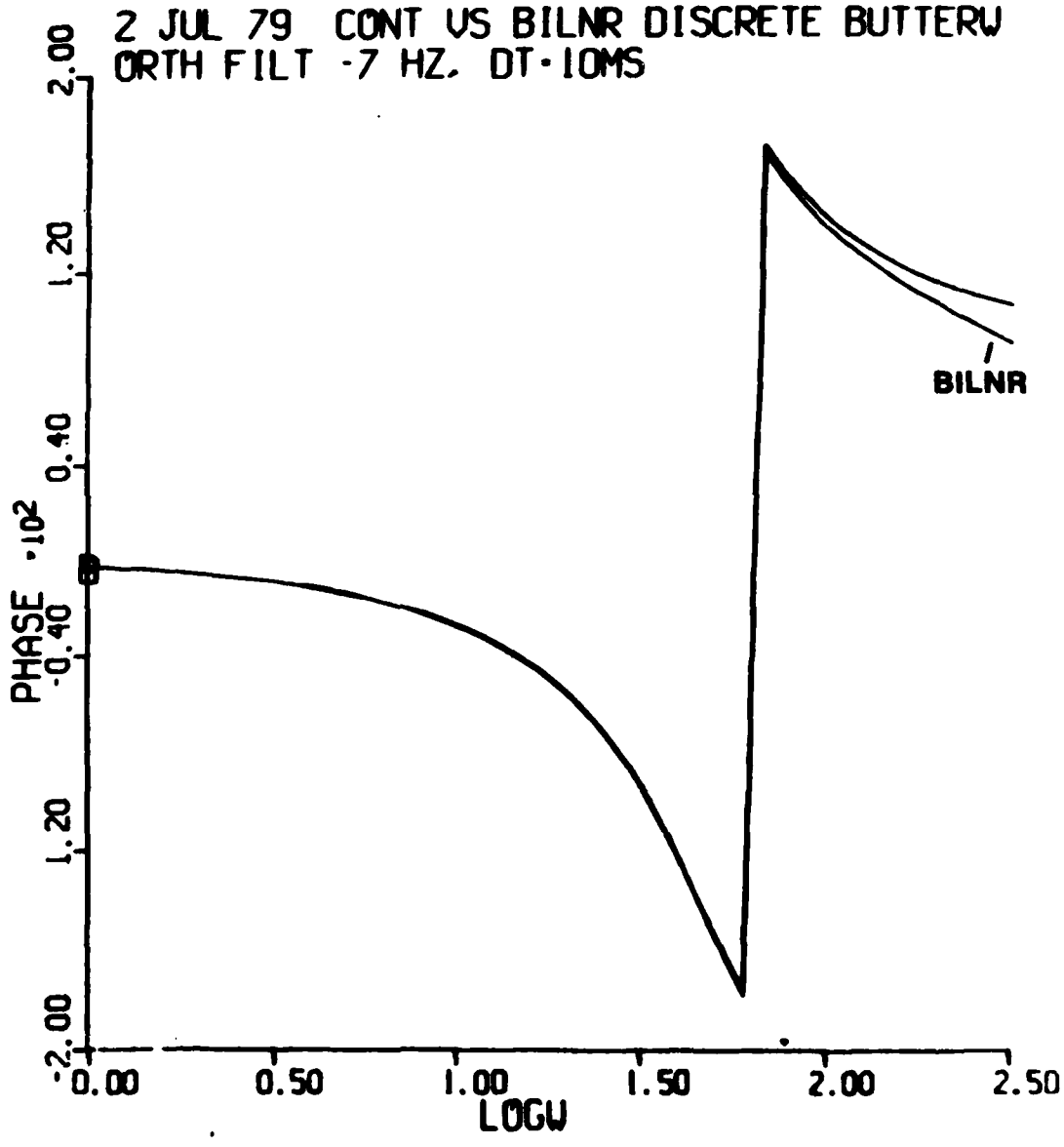


Figure 9. Phase versus frequency comparison of the s domain and bilinear Z domain filter representations at a 10 msec sampling interval.

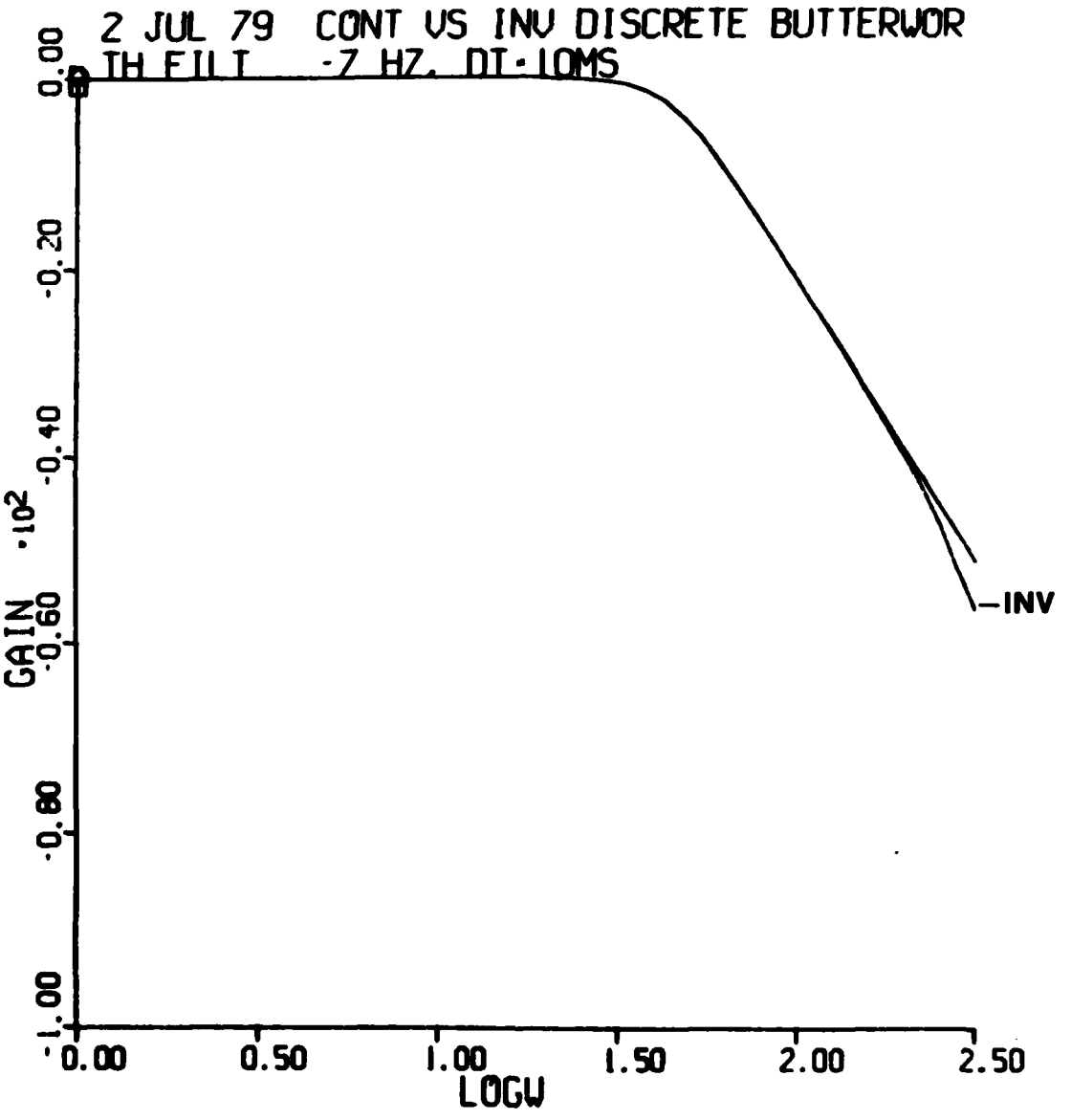


Figure 10. Gain versus frequency comparison of the s domain and impulse invariant Z domain filter representations of a 10 msec sampling interval.

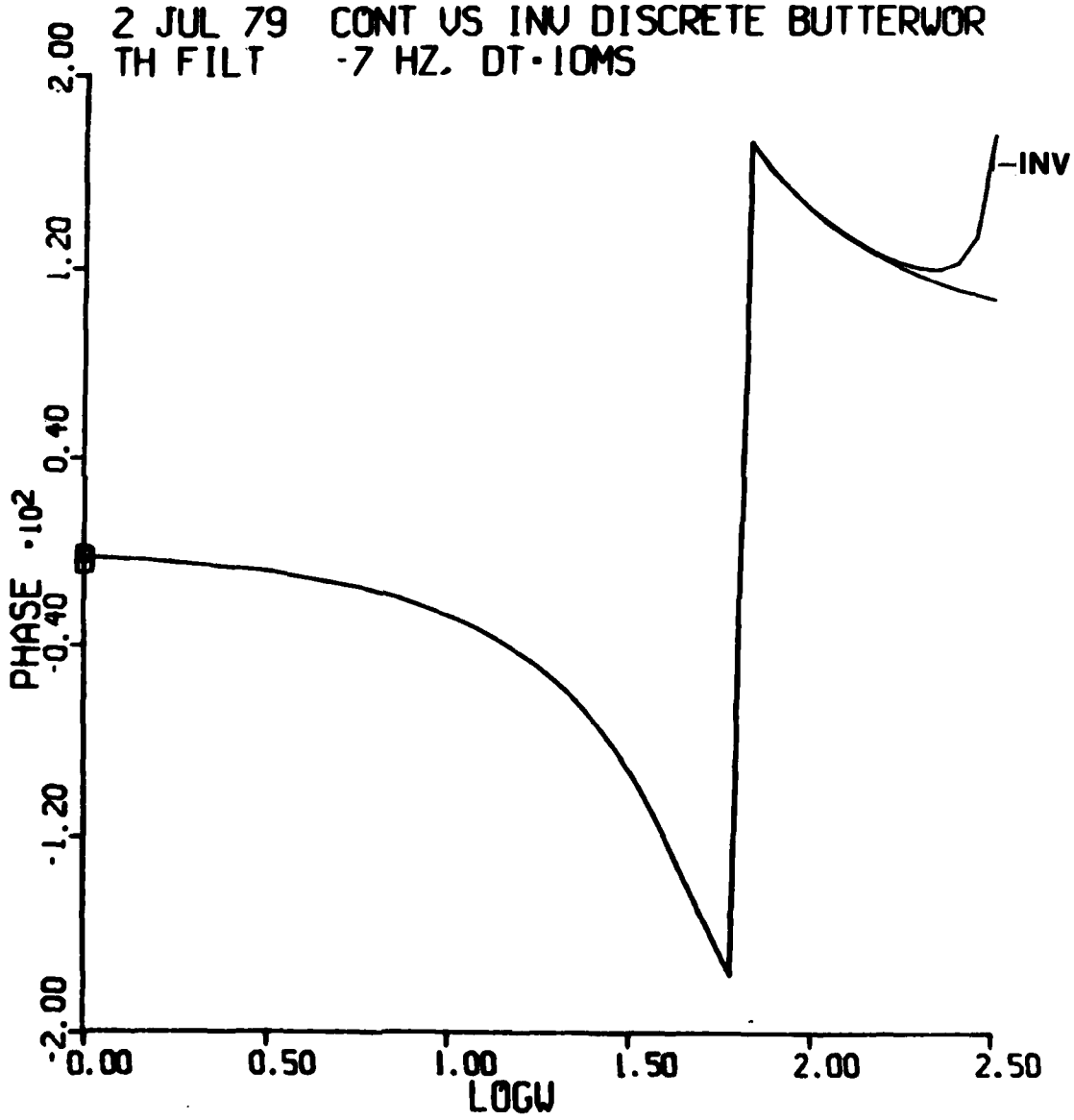


Figure 11. Phase versus frequency comparison of the s domain and impulse invariant Z domain filter representations at a 10 msec sampling interval.

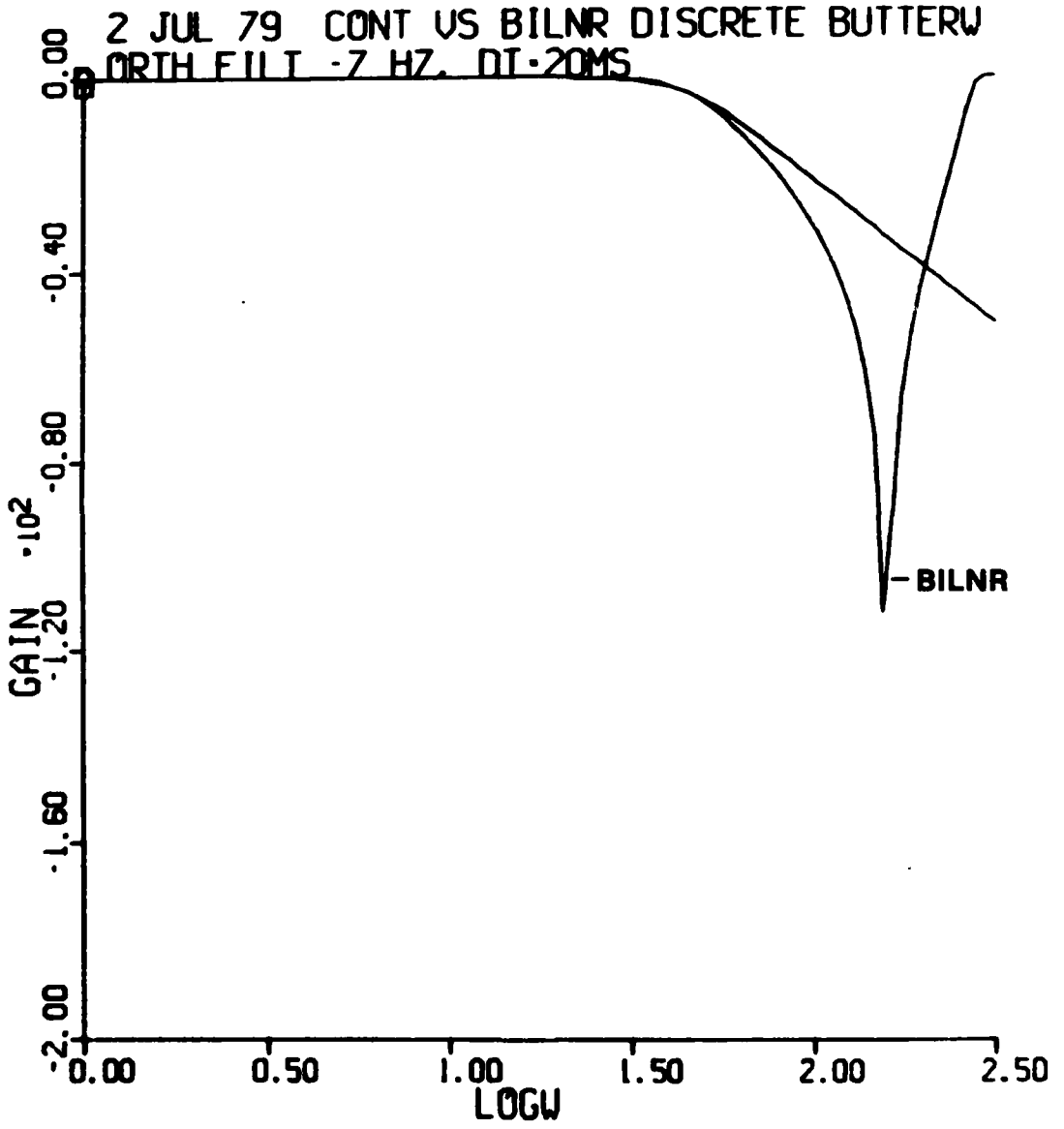


Figure 12. Gain versus frequency comparison of the s domain and bilinear Z domain filter representations at a 20 msec sampling interval.

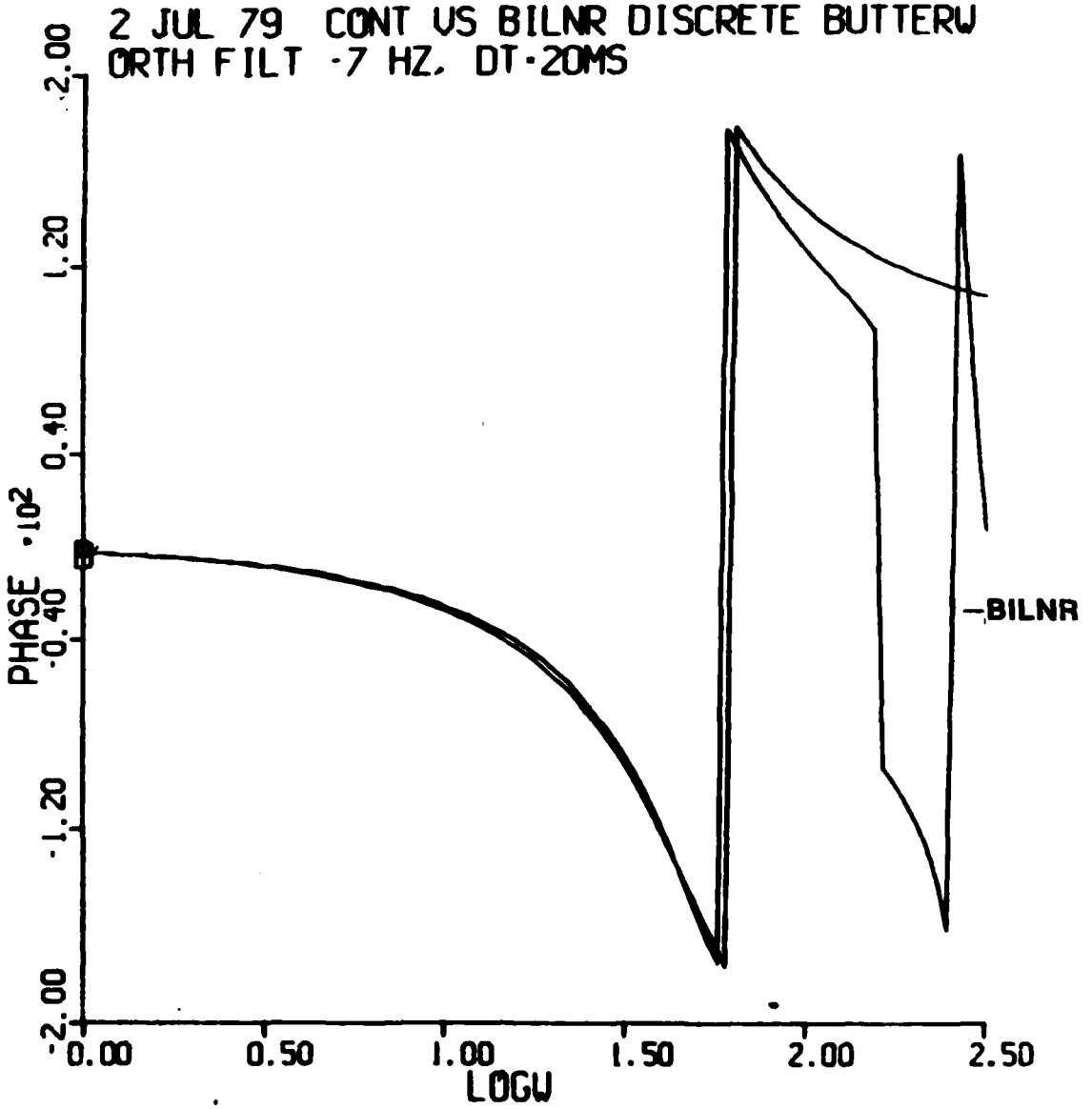


Figure 13. Phase versus frequency comparison of the s domain and bilinear Z domain filter representations at a 20 msec sampling interval.

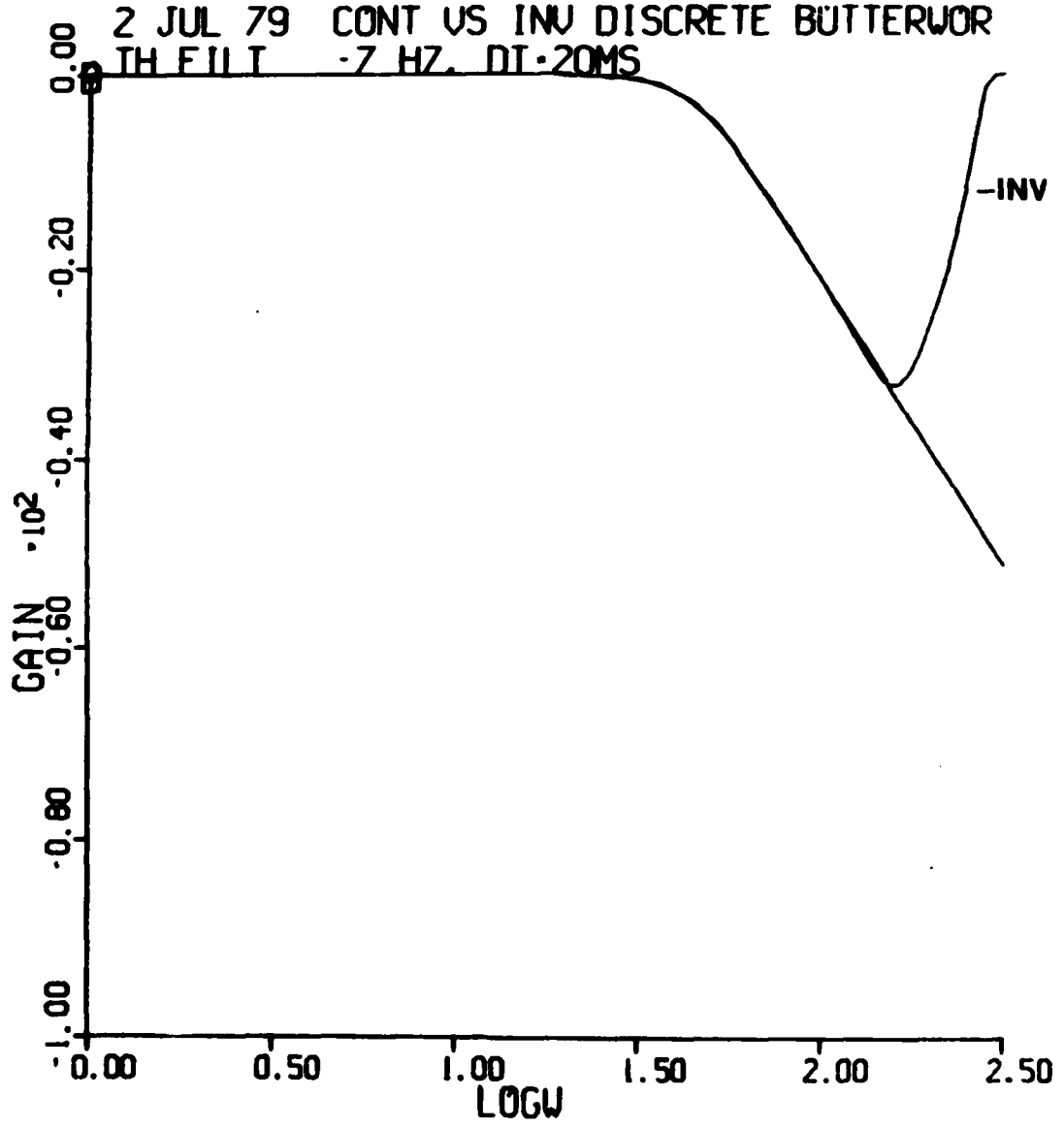


Figure 14. Gain versus frequency comparison of the s domain and impulse invariant Z domain filter representations at a 20 msec sampling interval.

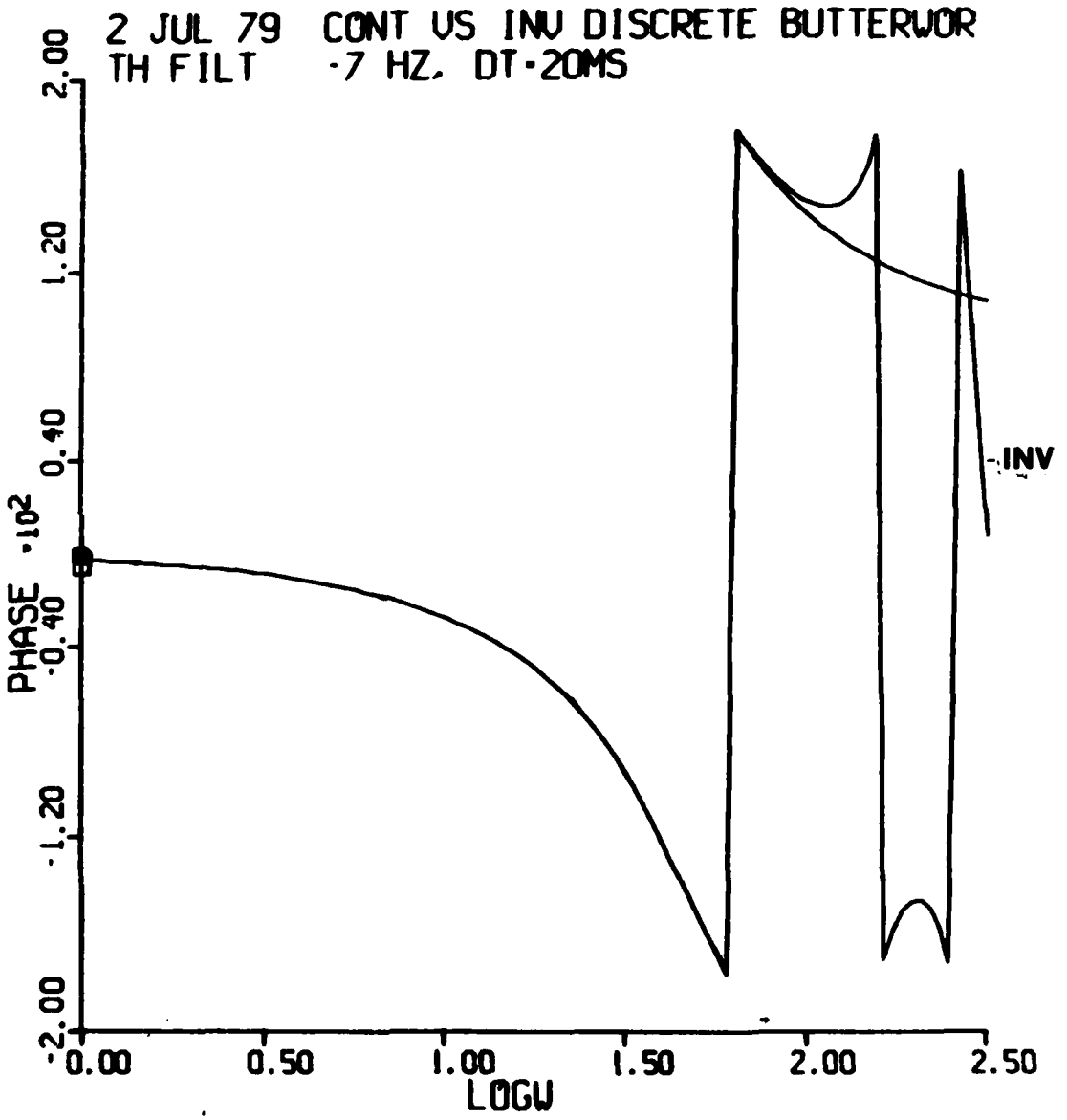


Figure 15. Phase versus frequency comparison of the s domain and impulse invariant Z domain filter representations at a 20 msec sampling interval.

just starting to show up at the higher frequencies. In *Figure 12* the bilinear gain remains below the continuous filter out to 31 Hz. However, in *Figure 13* the phase jumped from positive to negative at 25 Hz-the reciprocal of twice the sampling interval. For the invariant method 25 Hz is the cross over point for both the gain and phase as shown in *Figures 14* and *15*.

DT was set to 50 and 100 msec with the results shown in *Figures 16* through *23*. Note that the bilinear method keeps its steep roll off whereas the invariant method shows the frequency aliasing phenomenon that destroys its roll off characteristics. The discontinuous look of the bilinear method plots is caused by the sparseness of data points at the higher frequencies due to the geometric progression method of generating frequency plot points. *Figures 24* through *27* are plots of the gain and phase versus frequency (rad) of the two digital filters overlayed onto the continuous filter for a sampling interval of 100 msec. Using plain frequency for the abscissa brought out the periodicity of the digital filters.

STOP

terminates the simulation. The PRINT file, now attached to the interactive terminal as a local file, is batched for line printing to interactive terminal 3D,

BATCH, PRINT, PRINT, 3D, HDMDD.

The sample interval value DT is determined by the frequency content of the input signal since input signal frequency fold-over can occur in both digital filters if DT is made too large. However, for the impulse invariant digital filter implementation, the actual shape of the frequency response curve itself becomes "aliased" (as mentioned in Section 2) as $\frac{1 \text{ cycle}}{2*DT \text{ sec}}$ approaches the break frequency.

4. TRANSIENT RESPONSE PROGRAM AND RESULTS FOR A THIRD ORDER BUTTERWORTH FILTER FOR BOTH THE Z AND s DOMAIN REPRESENTATIONS

An ACSL program was written to compare the transient response of the two digital filter implementations with the analytical transient response of the continuous version of the filter. The inputs are sine waves and step functions. Also programmed was a standard digital simulation method that uses the RK2 numerical integration scheme to implement the M-method canonical form of the continuous filter (as discussed in Section 2). The numerical integration scheme will be compared with the digital filter schemes. It would also be desirable

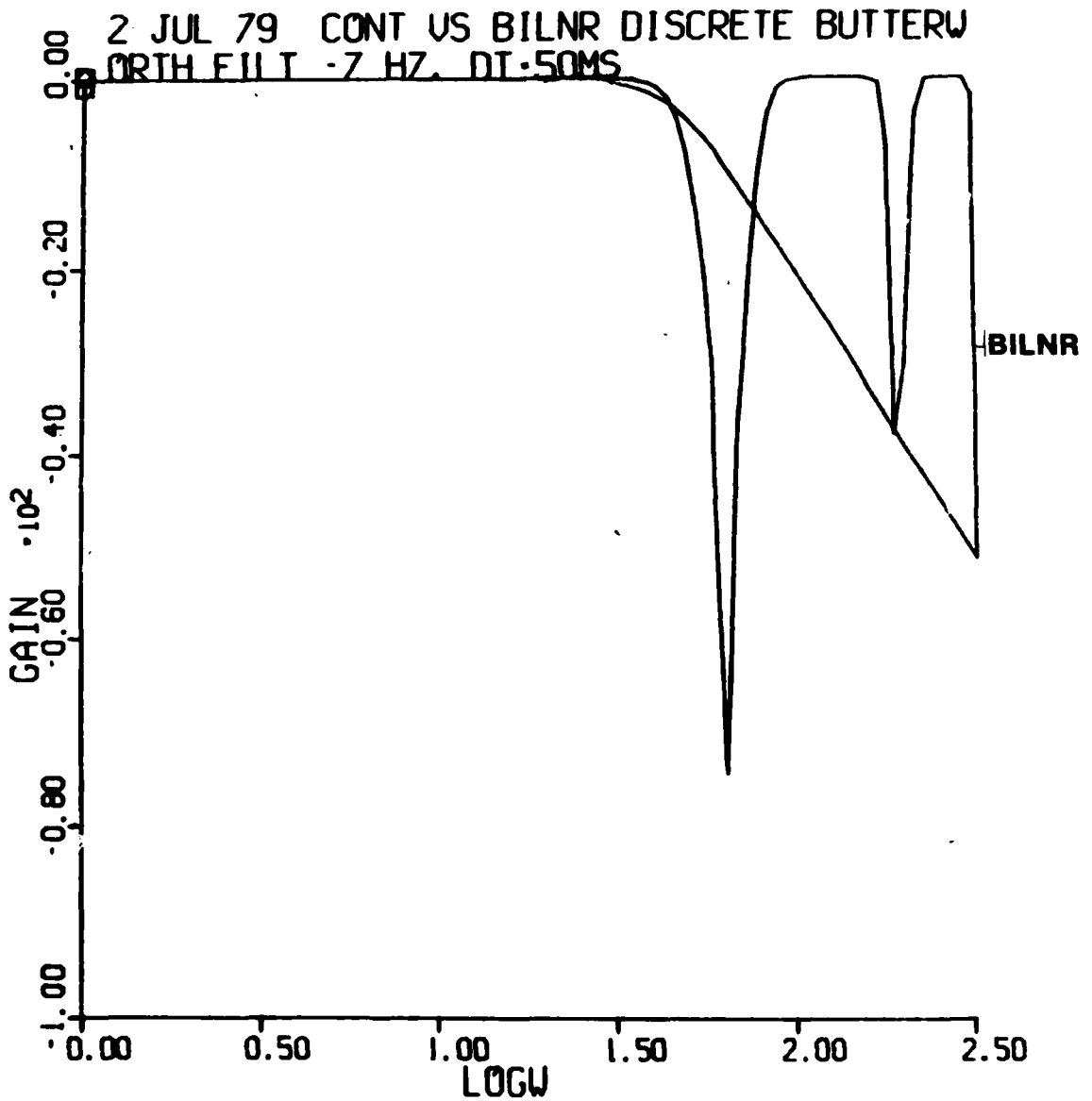


Figure 16. Gain versus frequency comparison of the s domain and bilinear z domain filter representations at a 50 msec sampling interval.

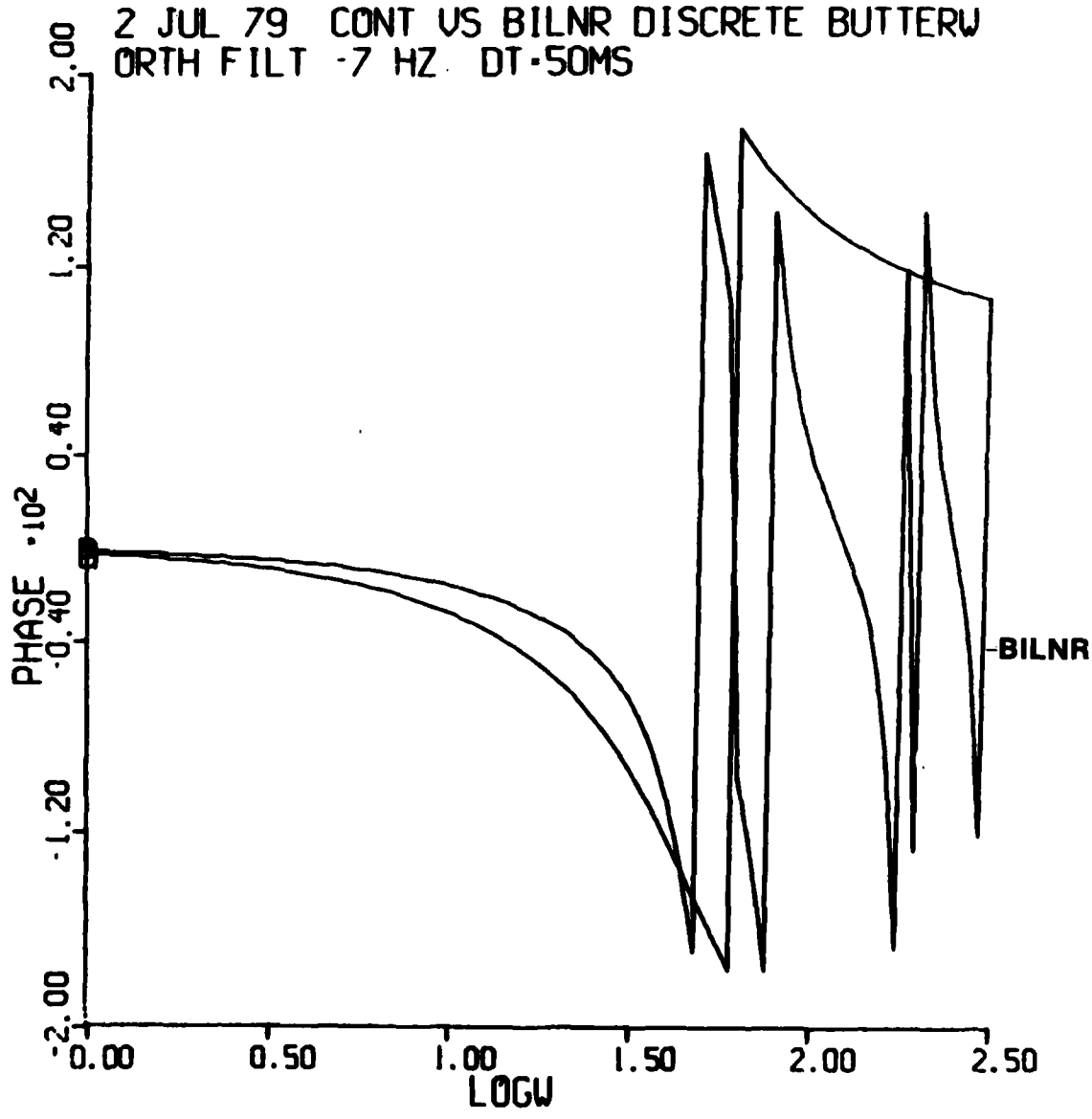


Figure 17. Phase versus frequency comparison of the s domain and bilinear Z domain filter representations at a 50 msec sampling interval.

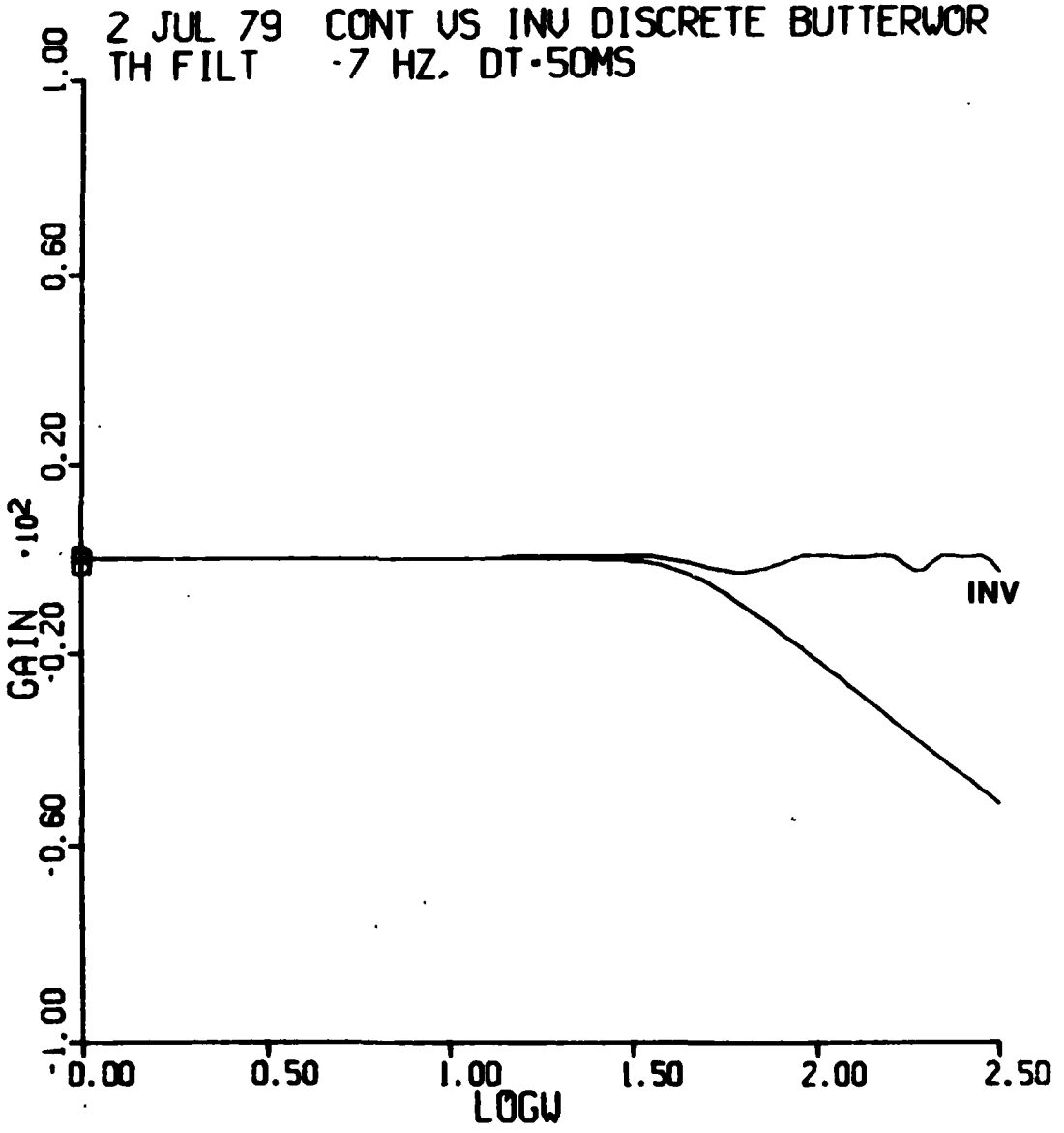


Figure 18. Gain versus frequency comparison of the s domain and impulse invariant Z domain filter representations at a 50 msec sampling interval.

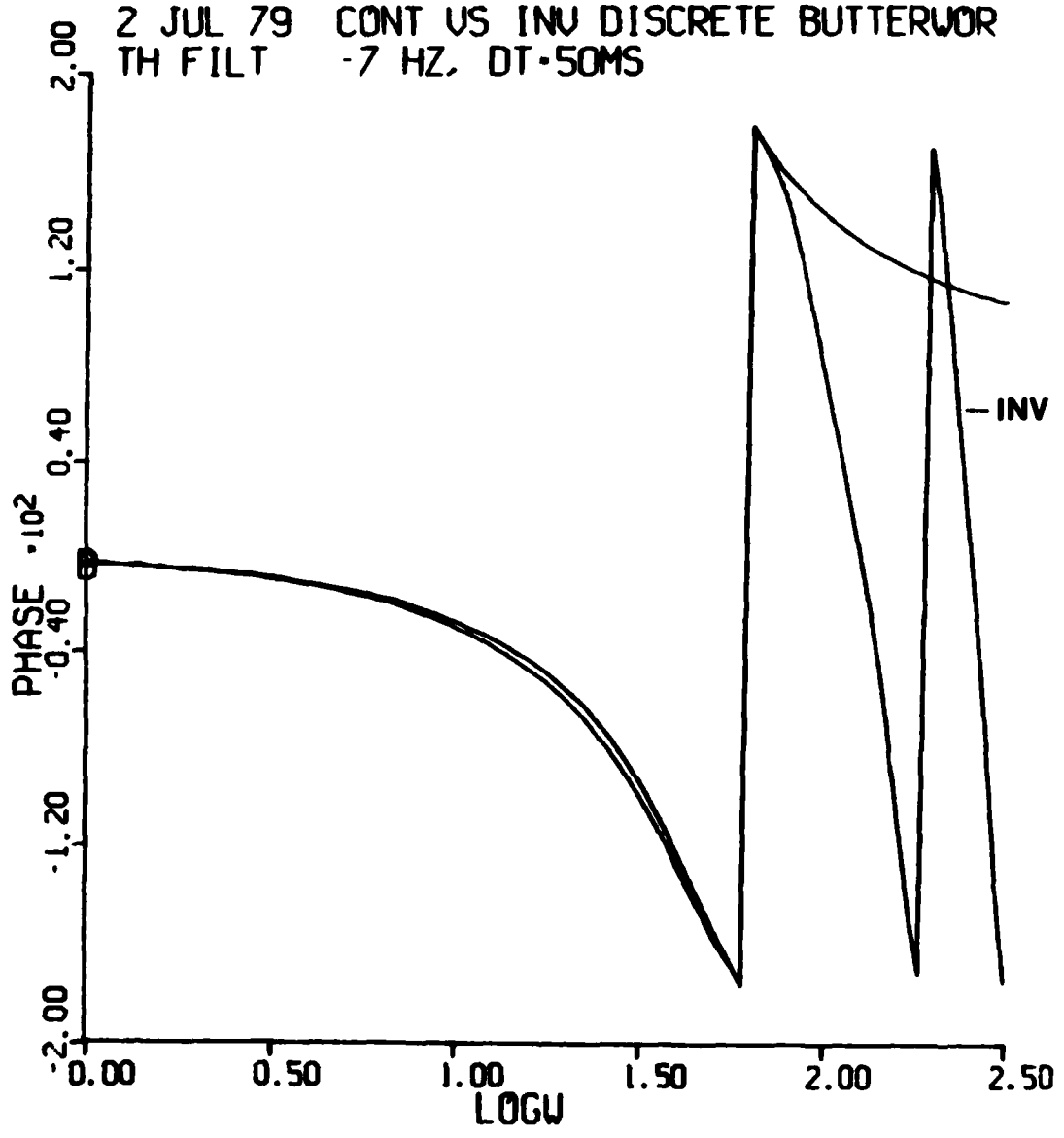


Figure 19. Phase versus frequency comparison of the s domain and impulse invariant Z domain filter representations at a 50 msec sampling interval.

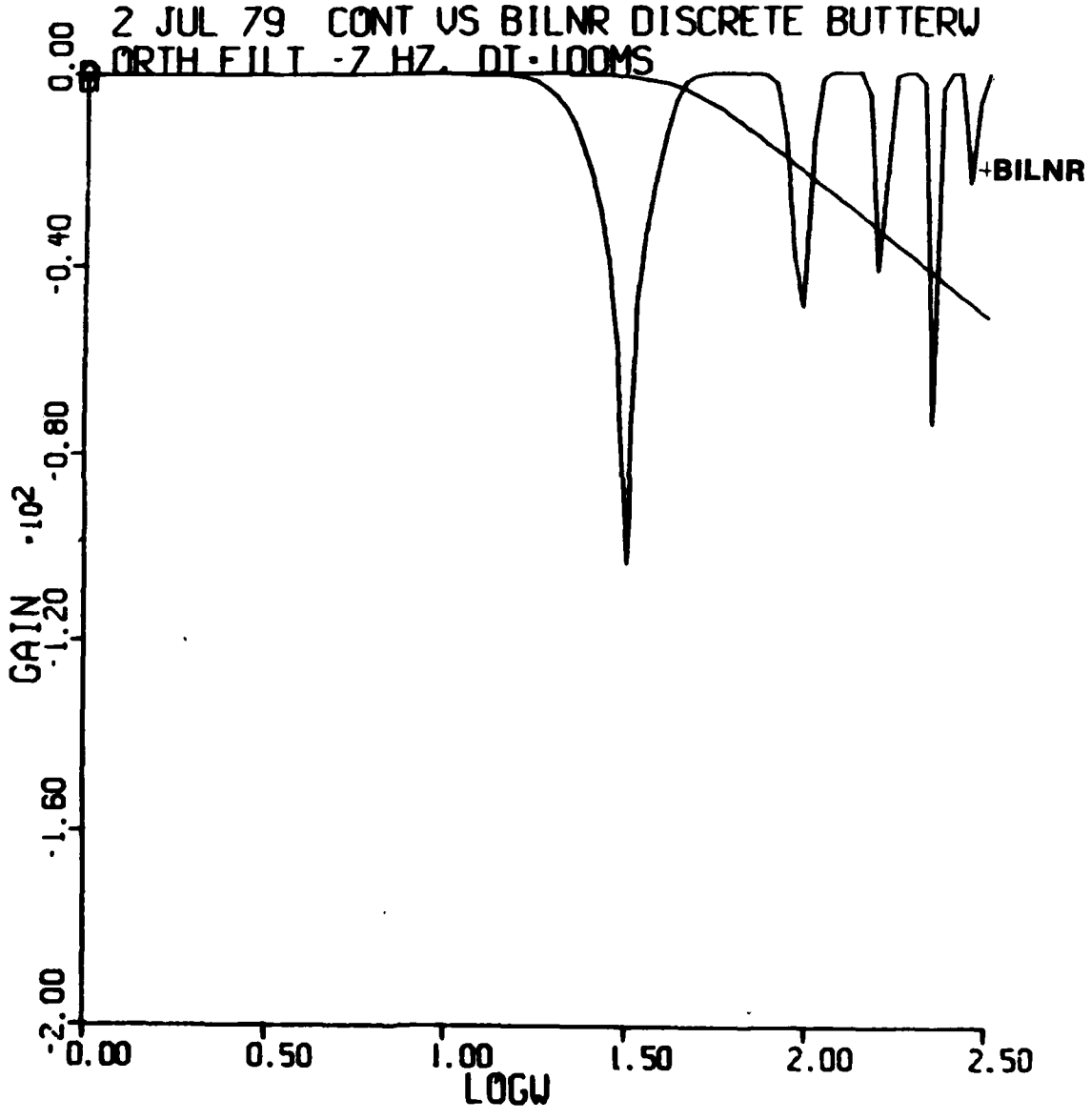


Figure 20. Amplitude versus frequency comparison of the s domain and bilinear Z domain filter representations at a 100 msec sampling interval.

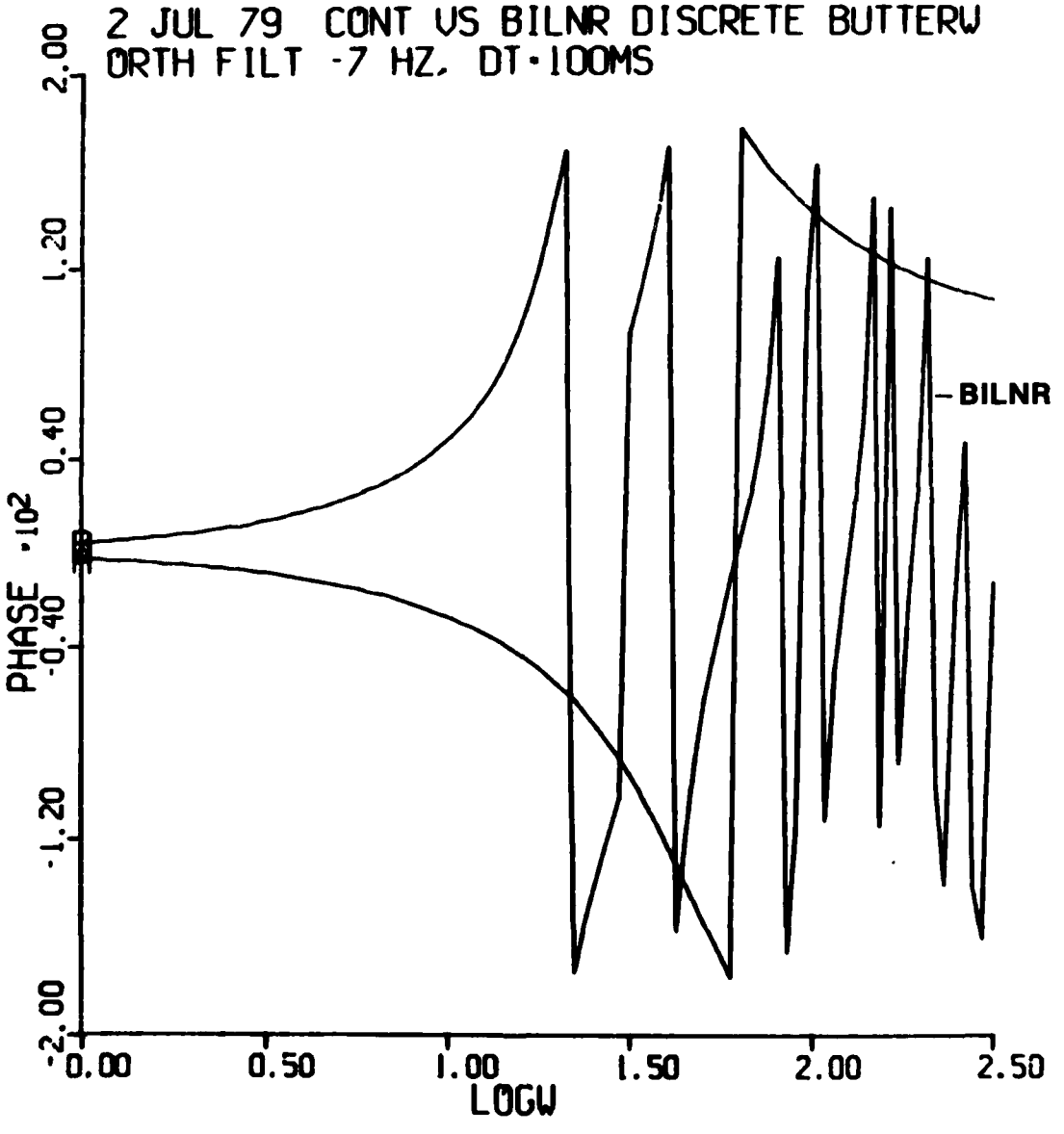


Figure 21. Phase versus frequency comparison of the s domain and bilinear Z domain filter representations at a 100 msec sampling interval.

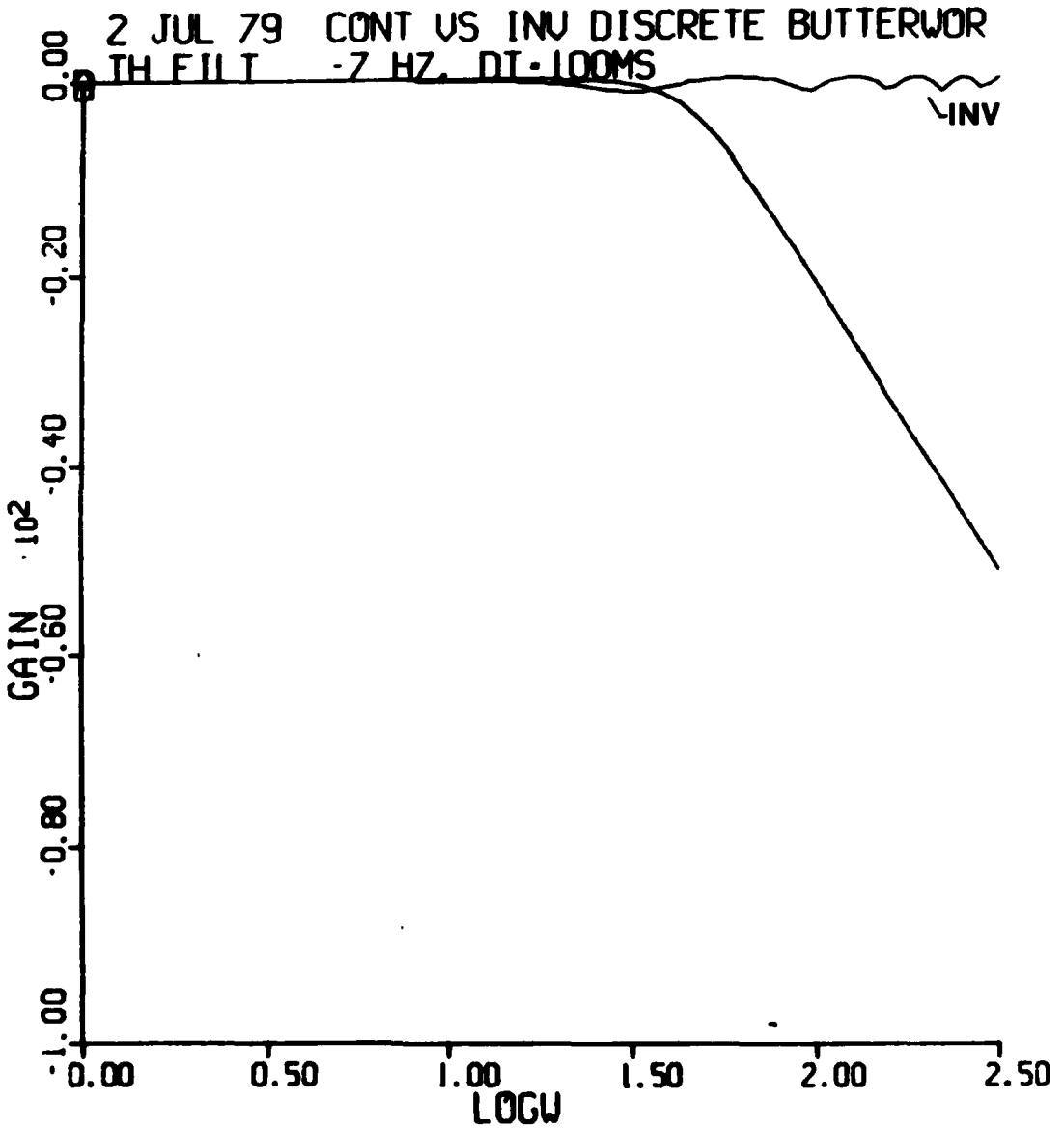


Figure 22. Gain versus frequency comparison of the s domain and impulse invariant Z domain filter representations at a 100 msec sampling interval.

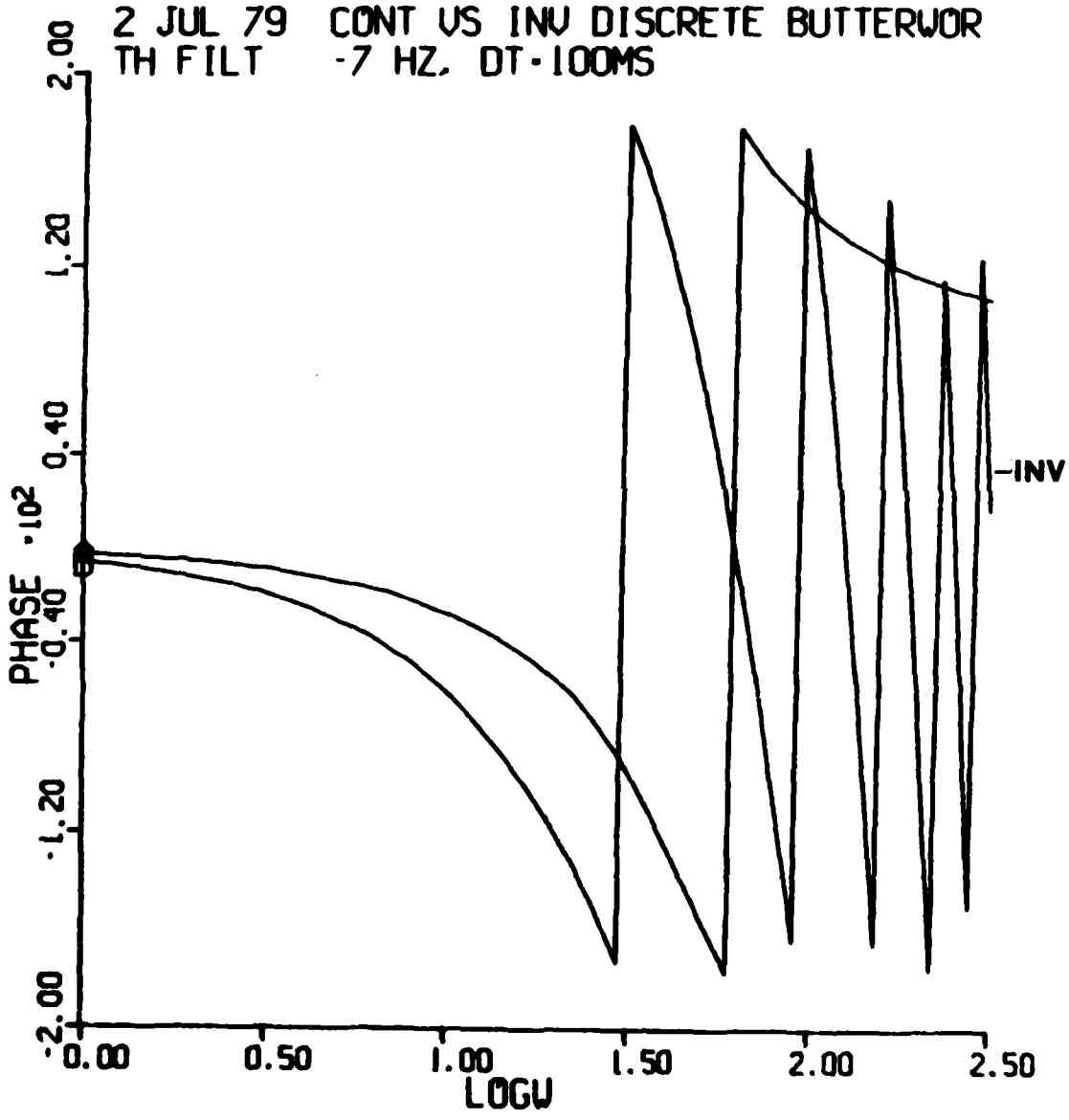


Figure 23. Phase versus frequency comparison of the s domain and impulse invariant Z domain filter representations at a 100 msec sampling interval.

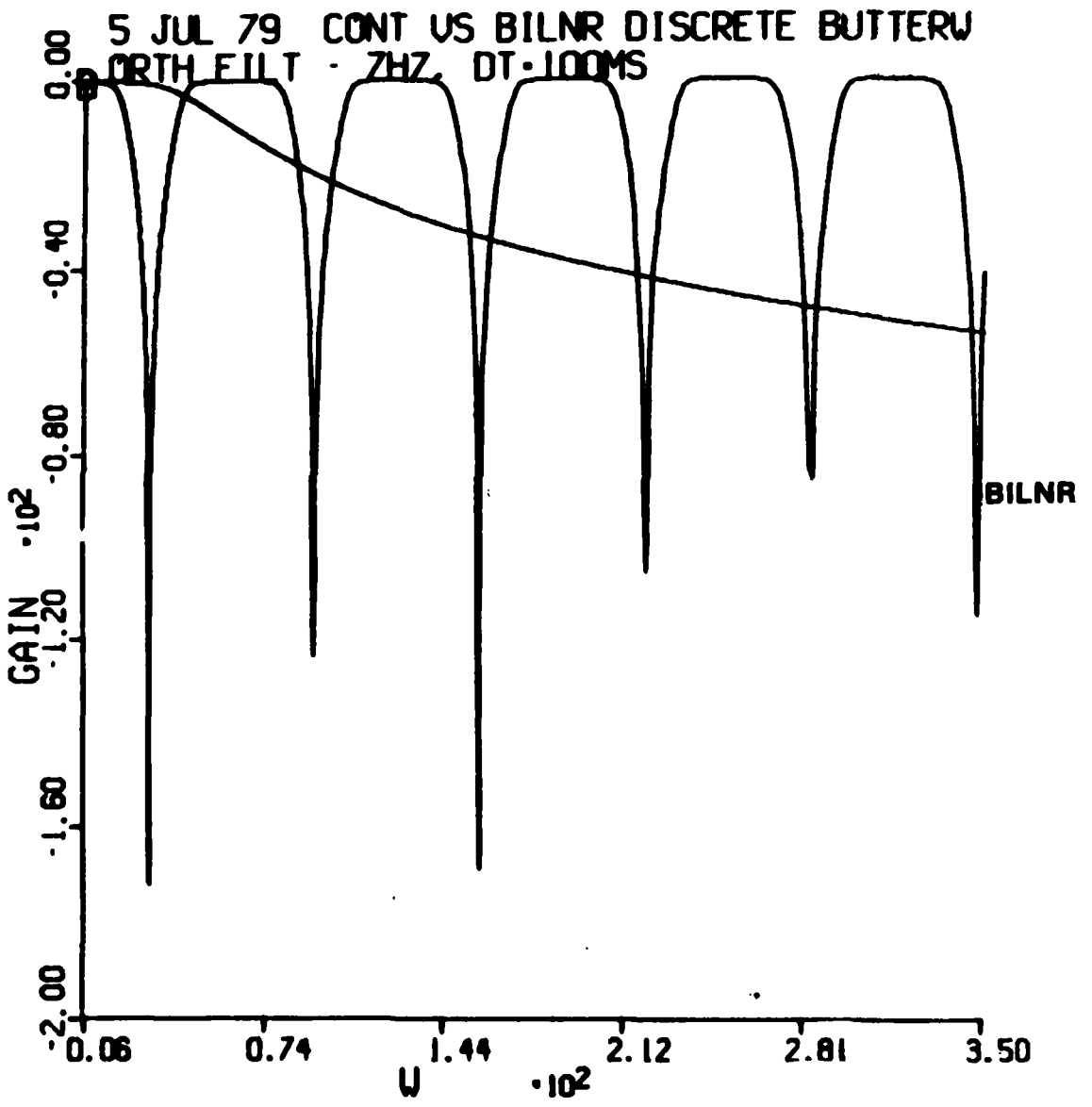


Figure 24. Amplitude versus frequency comparison of the s domain and bilinear Z domain filter representations at a 100 msec sampling interval.

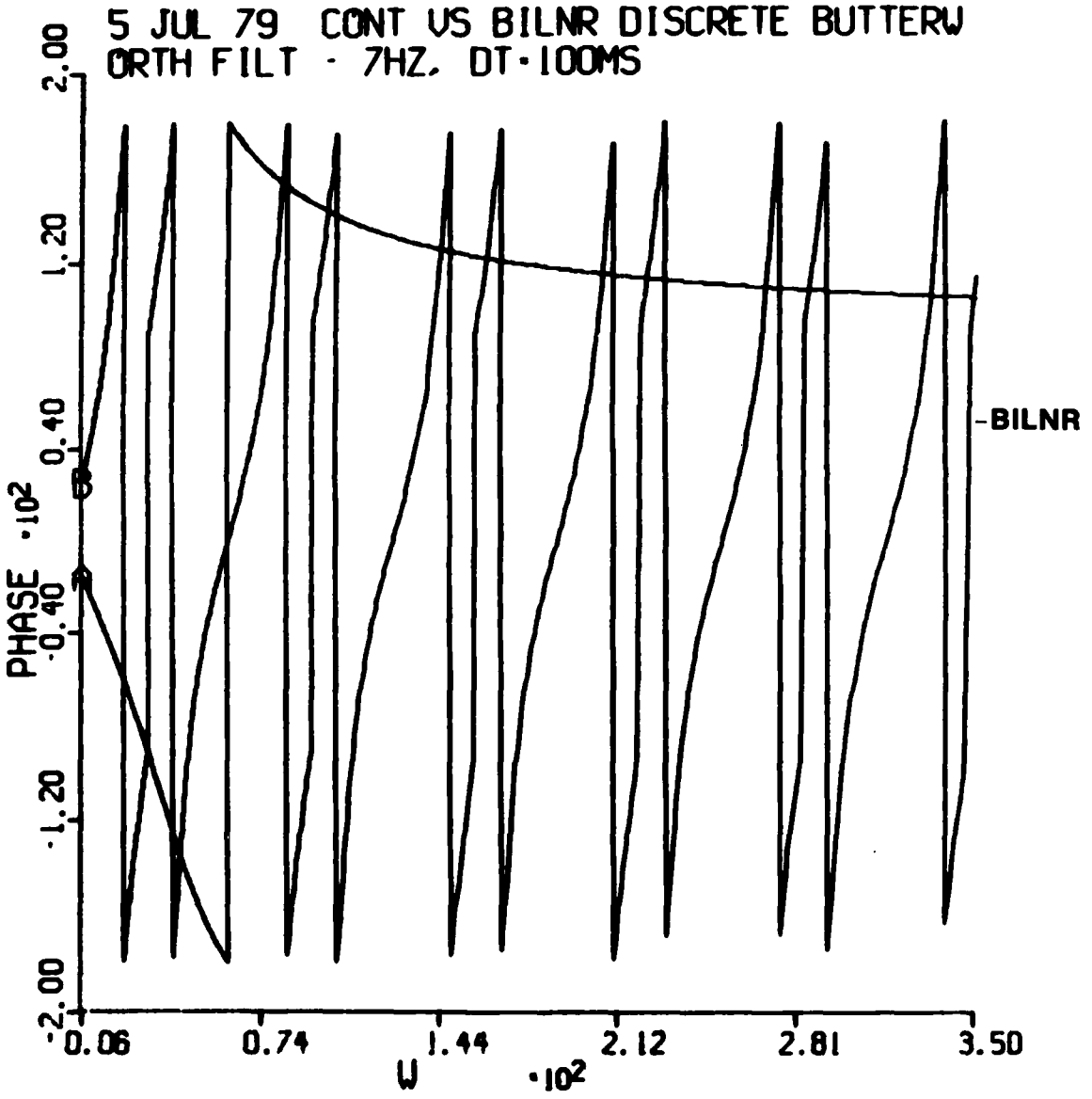


Figure 25. Phase versus frequency comparison of the s domain and bilinear Z domain filter representations at a 100 msec sampling interval.

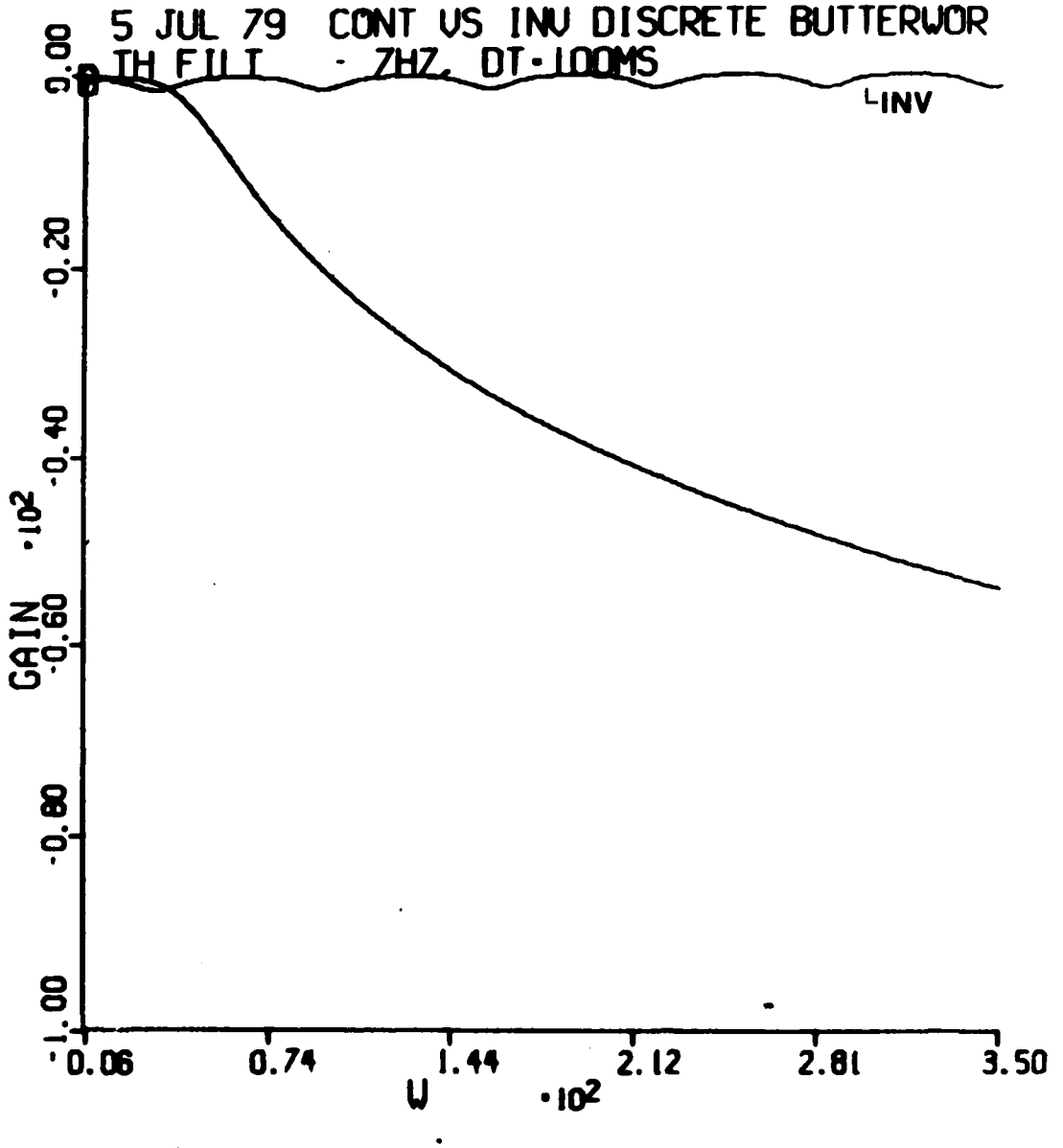


Figure 26. Amplitude versus frequency comparison of the s domain and impulse invariant Z domain filter representations at a 100 msec sampling interv

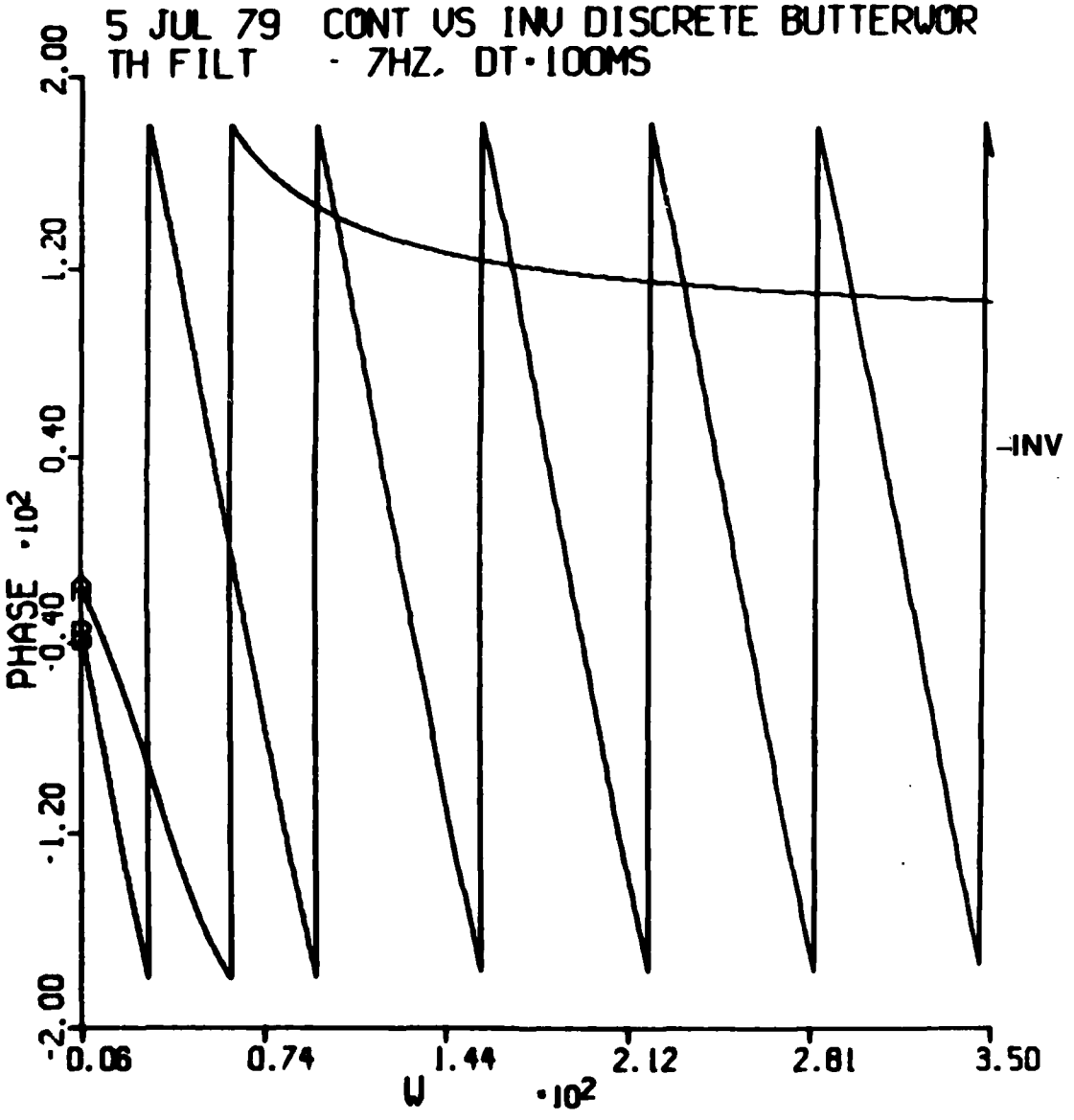


Figure 27. Phase versus frequency comparison of the s domain and impulse invariant Z domain filter representations at a 100 msec sampling interval.

to compare the steadied-out values from the transient response with those predicted by the Bode plot program.

As discussed in Section 2 and Appendix A the digital filters were implemented in macro ZXSFRM which uses the M-method canonical form. The digital filters are also implemented in the program using Bellman's 'Rectangular Method' canonical form for comparison purposes. *Figure 28* shows the general form for the Rectangular method.

The ACSL transient response program listing is in *Figure 29*. Macros BTRWT1 and BTRWT2 are the Rectangular method implementation of the impulse invariant and the bilinear methods for a 7 Hz break frequency and 10 msec sampling interval, respectively. Macros CNVRNT and CBILNR were discussed in Section 3. They generate the polynomial coefficient values for the two digital filters as implemented in the general purpose Z-transform macro ZXSFRM. Parameters are defined next and the program I/O are listed. The I/O will be defined as the program is delineated. In the preINITIAL section global constants are defined. These are parameters such as computer arithmetic minimum and maximum, RMN and RMX; π; the conversion factor for radians to degrees, RADDEG; logical variable DUMP which controls the debug printing of the program variables for the final communication interval; and the stopping time for a simulation run, TSTP.

```
CINTERVAL CINT=0.050
NSTEPS NSTP=1
```

sets the communication interval that is, the interval of time between plot points or line printer outputs.

In the INITIAL section the parameters for this particular program are defined. The input amplitude and frequency for a sine wave and the start time for the step function are given values. Macros CNVRNT and CBILNR define the numerator and denominator coefficients U and V for the impulse invariant method and R and S for the bilinear method. Constant gains for the analytical response of the continuous filter to a unity gain sine wave, Kα, Kβ, Kγ, Kδ and Kϵ, are calculated. These constants correspond to A, B, C, -D $2/\sqrt{3}$ and E/ω_o of Equation (4) in Section 2.

In the Dynamic section the analytical continuous filter response is calculated according to Equations (3) and (4). The output is named YANALF. The TERMT macro is used at the end of the DYNAMIC section to stop the simulation when global time T exceeds the value in variable TSTP.

$$\frac{y}{x} = \frac{\text{OUTPUT}}{\text{INPUT}} = \frac{a_0 \pm a_1 z^{-1} \pm a_2 z^{-2} \pm a_3 z^{-3}}{b_0 \pm b_1 z^{-1} \pm b_2 z^{-2} \pm b_3 z^{-3}}$$

$$y \frac{1}{b_0} [a_0 x + z^{-1}(a_1 x - b_1 y + z^{-1}(a_2 x - b_2 y + z^{-1}(a_3 x - b_3 y)))]$$

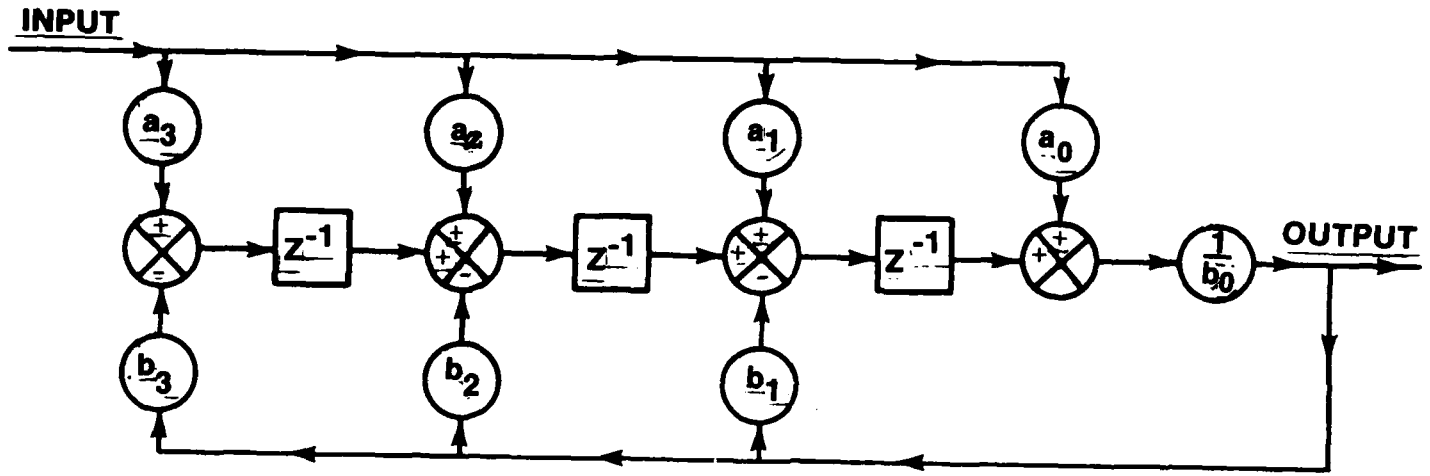


Figure 28. Bellman's rectangular method canonical form.

```
      PROGRAM TRANSIENT RESPONSE FOR CONTINUOUS AND DISCRETE BUTTERWORTH

MACRO BTRWT1(B,A)
MACRO REDEFINE X,Y,XIC,YIC,A1,A2,A3,B1,B2,I
MACRO RELABEL SN001,SN002
ARRAY XIC(3),YIC(3),X(3),Y(3)
CONSTANT YIC=3*0.0,XIC=3*0.0
CONSTANT B1=3.13815386E-2,B2=2.34235779E-2,A1=+2.13428885,A2=-1.60402376
CONSTANT A3=+0.414929794
PROCEDURAL(B=A)
IF(ZZFST(I).LT.0.5)GO TO SN002
DO SN001 I=1,3
CALL ZZICS(YIC(I),Y(I))
SN001..CALL ZZICS(XIC(I),X(I))
B=Y(1)*A1+Y(2)*A2+Y(3)*A3+X(1)*B1+X(2)*B2
Y(3)=Y(2)
Y(2)=Y(1)
Y(1)=B
X(2)=X(1)
X(1)=A
SN002..CONTINUE
END
MACRO END

MACRO BTRWT2(B,A)
MACRO REDEFINE X,Y,XIC,YIC,I,A1,A2,A3,B1,B2,B3,B4
MACRO RELABEL SN001,SN002
ARRAY XIC(3),YIC(3),X(3),Y(3)
CONSTANT YIC=3*0.0,XIC=3*0.0
CONSTANT A1=+2.12664    ...
,A2=-1.59582,A3=0.41184,B1=7.167E-3,B2=2.1503E-2,B3=2.1503E-2,B4=7.16...
7E-3
PROCEDURAL(B=A)
IF(ZZFST(I).LT.0.5)GO TO SN002
DO SN001 I=1,3
CALL ZZICS(YIC(I),Y(I))
SN001..CALL ZZICS(XIC(I),X(I))
B=Y(1)*A1+Y(2)*A2+Y(3)*A3+B1*A+B2*X(1)+B3*X(2)+B4*X(3)
Y(3)=Y(2)
Y(2)=Y(1)
Y(1)=B
X(3)=X(2)
X(2)=X(1)
X(1)=A
SN002..CONTINUE
END
MACRO END
```

Figure 29. Listing of the ACSL transient response program.

```
MACRO CNVRNT(U,V,WB,DT,KG)
MACRO REDEFINE K,E,C,S
ARRAY U(3),V(4)
PROCEDURAL(U,V=WB,DT,KG)
E=EXP(-WB*DT)
C=SQRT(E)*COS(SQRT(3.)*0.5*WB*DT)
S=SQRT(E)*SIN(SQRT(3.)*0.5*WB*DT)/SQRT(3.)
U(1)=0.0
U(2)=WB*(-C+S+E)
U(3)=WB*E*(1.-C-S)
V(1)=1.0
V(2)=-2.*C-E
V(3)=+E*(1.+2.*C)
V(4)=-E**2
K=(U(2)+U(3))/(V(1)+V(2)+V(3)+V(4))
U(1)=U(1)*KG/K
U(2)=U(2)*KG/K
U(3)=U(3)*KG/K
END
MACRO END

MACRO CBILNR(R,S,WB,DT,KG)
MACRO REDEFINE AL,DEN
ARRAY R(4),S(4)
PROCEDURAL(R,S=WB,DT,KG)
AL=TAN(0.5*WB*DT)
DEN=1.+2.*AL+2.*AL**2+AL**3
R(1)=KG*AL**3/DEN
R(2)=KG*3.*AL**3/DEN
R(3)=R(2)
R(4)=R(1)
S(1)=1.0
S(2)=(-3.-2.*AL+2.*AL**2+3.*AL**3)/DEN
S(3)=(3.-2.*AL-2.*AL**2+3.*AL**3)/DEN
S(4)=(-1.+2.*AL-2.*AL**2+AL**3)/DEN
END
MACRO END
```

Figure 29. (Continued)

```
MACRO MACRO ZXSFRM(P,Q)
MACRO REDEFINE Y,I,YIC,J,YP
MACRO RELABEL SN001,SN002,SN003,SN004,SN005
MACRO ASSIGN N
MACRO MULTIPLY 0
MACRO INCREMENT Q(4)
MACRO DIVIDE Q(3)
MACRO IF(N=0)999
MACRO MULTIPLY 0
MACRO INCREMENT Q(4)
ARRAY Y(N),YIC(N)
CONSTANT YIC=N*0.0
PROCEDURAL(P(1)=P(2),P(3),P(4))
IF(ZZFST(I).LT.0.5)GO TO SN005
DO SN001 I=1,N
SN001..CALL ZZICS(YIC(I),Y(I))
YP=P(2)
MACRO DECREMENT 1
DO SN002 I=1,N
MACRO INCREMENT 1
SN002..YP=YP-P(4)(I+1)*Y(N-I)
Y(N)=YP/P(4)(1)
YP=0.0
MACRO INCREMENT 1
DO SN003 I=1,Q(3)
SN003..YP=YP+P(3)(I)*Y(N-I)
P(1)=YP
MACRO DECREMENT 2
DO SN004 I=1,N
SN004..Y(I)=Y(I+1)
SN005..CONTINUE
END
MACRO EXIT
MACRO 999..PRINT NUMERATOR GREATER THAN DENOMINATOR
MACRO END

      "PARAMETERS"
"A-AMPLITUDE OF INPUT SIGNAL TO FILTER"
"MMTHD-LOGICAL, IF .T. INDICATES M-METHOD IMPLEMENTATION IS TO BE USED"
"STPSIN-LOGICAL INDICATING WHETHER INPUT SINE WAVE (.F.) OR STEP (.T.)"
"TZ-TIME AT WHICH STEP TURNS ON - SEC"
"W-INPUT SINE WAVE FREQUENCY - RAD/SEC"
"WBCPS-FILTER BREAK FREQUENCY - HZ"
"WB-FILTER BREAK FREQUENCY - RAD/SEC"
"WCPS-W IN HZ"

      "INPUTS"
"MMTHD, STPSIN, WBCPS, A, WCPS, TSTP, CINT, MAXT1, MAXT2, MAXT3, TZ"

      "OUTPUTS"
"YINTEG, U, V, R, S, YINV, YBILNR, YANALS, YANALF, YANALR, XZFER,    ...
 XINTEG"
```

Figure 29. (Continued)

```
"-----------------------------GLOBAL CONSTANTS                                "
      LOGICAL         DUMP
      CONSTANT  RMN = 1.0E-30, RMX = 1.0E30 , RADDEG=57.2957795       ...
              , PI=3.14159265, DUMP=.FALSE. , TSTP = 3.0
      TWOPI   = 2.0*PI
      CINTERVAL CINT = 0.050
      NSTEPS NSTP = 1

INITIAL
"-----------------------------SET CONSTANTS FOR INPUT SIGNAL AND      ...
                              DESIRED BREAK PT FREQUENCY OF FILTERS     "
      LOGICAL         STPSIN
      CONSTANT  STPSIN = .TRUE.
      CONSTANT  WCPS = 50.    , A = 1.0       , TZ = 0.049999999
      W       = WCPS*TWOPI
      CONSTANT  WBCPS = 7.0
      WB      = WBCPS*TWOPI
"-----------------------------INVARIANT FILTER COEFFS FOR 3RD ORDER   ...
                              BUTTERWORTH                               "
      CNVRNT(U, V=WB, MAXT2, 1.0)
"-----------------------------BILINEAR FILTER COEFFS FOR 3RD ORDER    ...
                              BUTTERWORTH                               "
      CBILNR(R, S=WB, MAXT2, 1.0)
"-----------------------------CONSTANTS FOR ANALYTICAL RESPONSE OF    ...
                              FILTER TO A SINE WAVE W/UNITY GAIN        "
      KAL     = WB**3*(W**3 - 2.0*WB**2*W)/(W**6 + WB**6)
      KBE     = WB**4*(-2.0*W**2 + WB**2)/(W**6 + WB**6)
      KGA     = WB*W/(WB**2 + W**2)
      KDL     =-1.1547*WB*(W**3 - WB**2*W)/(W**4 - W**2*WB**2 + WB**4)
      KEP     = WB**3*W/(W**4 - W**2*WB**2 + WB**4)
END $ "INITIAL"
DYNAMIC
"-----------------------------CALCULATE THE ANALYTICAL VALUE FOR THE ...
                         BUTTERWORTH FILTER FOR THE FAST DERIVATIVE SECT"
      YANALF  = RSW(STPSIN, RSW(T.GE.TZ, STEP(TZ) - EXP(-WB*(T-TZ))   ...
                                        -1.1547*EXP(-0.5*WB*(T-TZ))   ...
                                         *SIN(0.866025*WB*(T-TZ))     ...
                                       , 0.0)                         ...
                          , KAL*COS(W*T) + KBE*SIN(W*T) + KGA*EXP     ...
                            (-WB*T) - KDL*EXP(-0.5*WB*T)*SIN          ...
                            (0.866025*WB*T - 1.047198) + KEP*EXP      ...
                            (-0.5*WB*T)*SIN(WB*T)                     ...


DERIVATIVE FAST
      MAXTERVAL MAXT1 = 0.010
      ALGORITHM IALG = 4
      TF      = INTEG(1.0, 0.0)
"-----------------------------CALCULATE INPUT TO XSFER FN               "
      XINTEG  = RSW(STPSIN, A*STEP(TZ), A*SIN(W*TF))
"-----------------------------ENUMERATE XSFER FN COEFFICIENTS           "
      REAL            P(1)    , Q(4)
      CONSTANT  P = 8.50812E4, Q = 1.0      , 87.9646                 ...
              , 3868.89       , 8.50812E4
"-----------------------------NUMERICALLY INTEGRATE FOR FILTER OUTPUT"
      TRAN(YINTEG=0, 3, P, Q, XINTEG)

END $ "DERIVATIVE"
```

Figure 29. (Continued)

```
DERIVATIVE SLOW
"---------------------------MAXT2 REPRESENTS SAMPLE INTERVAL LENGTH  "
      MAXTERVAL MAXT2 = 0.010
      ALGORITHM IALG = 4
      TS      = INTEG(1.0, 0.0)
"---------------------------CALCULATE FILTER OUTPUT USING THE INV  ...
                              AND BILINEAR Z-TRANSFORM METHODS"
"------------------------- -CHOOSE M-METHOD FOR IMPLEMENTING FILTERS "
      LOGICAL         MMTHD
      CONSTANT  MMTHD=.TRUE.
"-----------------------------INVARIANT FILTER                       "
      YINV    = RSW(MMTHD, ZXSFRM(XZFER, U, V), BTRWT1(XZFER))
"-----------------------------BILINEAR FILTER                        "
      YBILNR  = RSW(MMTHD, ZXSFRM(XZFER, R, S), BTRWT2(XZFER))

PROCEDURAL(YANALS,XZFER=)
      IF(ZZFST(TZ).LT.0.5)GO TO SN002
"---------------------------CALCULATE INPUT TO Z TRANSFORM FILTERS   "
      XZFER   = RSW(STPSIN, A*STEP(TZ), A*SIN(W*TS))
"---------------------------CALCULATE THE ANALYTICAL VALUE FOR THE ...
                              BUTTERWORTH FILTER                     "
      YANALS  = RSW(STPSIN, RSW(TS.GE.TZ, STEP(TZ) - EXP(-WB*(TS-TZ))...
                                          -1.1547*EXP(-0.5*WB*(TS-TZ))...
                                          *SIN(0.866025*WB*(TS-TZ))  ...
                                         , 0.0)                      ...
                          , KAL*COS(W*TS) + KBE*SIN(W*TS) + KGA*EXP  ...
                            (-WB*TS) - KDL*EXP(-0.5*WB*TS)*SIN       ...
                            (0.866025*WB*TS - 1.047198) + KEP*EXP    ...
                            (-0.5*WB*TS)*SIN(WB*TS)                  ...
                    )
SN002..CONTINUE
      ENDS"PROCED"

END $ "DERIVATIVE"
DERIVATIVE RECORD
"---------------------------SAMPLE FAST ENOUGH TO SEE THE STAIRCASE...
                              IN YINV AND YBILNR"
      MAXTERVAL MAXT3=0.0020800
"---------------------------UNSYNCHRONIZED EULER"
      ALGORITHM IALG=3
"---------------------------ESTABLISH LOCAL TIME FOR THIS            ...
                              DERIVATIVE SECTION"
      TR      = INTEG(1.0, 0.0)
"---------------------------CALCULATE THE ANALYTICAL VALUE FOR THE ...
                              BUTTERWORTH FILTER                     "
      YANALR  = RSW(STPSIN, RSW(TR.GE.TZ, STEP(TZ) - EXP(-WB*(TR-TZ))...
                                          -1.1547*EXP(-0.5*WB*(TR-TZ))...
                                          *SIN(0.866025*WB*(TR-TZ))  ...
                                         , 0.0)                      ...
                          , KAL*COS(W*TR) + KBE*SIN(W*TR) + KGA*EXP  ...
                            (-WB*TR) - KDL*EXP(-0.5*WB*TR)*SIN       ...
                            (0.866025*WB*TR - 1.047198) + KEP*EXP    ...
                            (-0.5*WB*TR)*SIN(WB*TR)                  ...
                    )
"---------------------------RECORD THE DATA"
      LOGICAL         RECORD
      CONSTANT  RECORD=.FALSE.
      IF(RECORD)CALL ZZLOG
END $ "DERIVATIVE"

      TERMT(T.GT.TSTP)
END $ "DYNAMIC"
```

Figure 29. (Continued)

```
TERMINAL
        IF(DUMP)CALL DEBUG
END $ "TERMINAL"
END $ "PROGRAM"
```

Figure 29. (Concluded)

There are three DERIVATIVE sections: DERIVATIVEs FAST, SLOW and RECORD. DERIVATIVE FAST contains the numerical integration of the continuous filter as implemented in the ACSL TRAN macro. MAXT1 is the maximum integration step size as declared by MAXTERVAL; the RK2 algorithm is selected through the ALGORITHM declarative statement; and TF, the local time for this DERIVATIVE section, is generated by integrating a unit step function. XINTEG, the input to TRAN, is either a step or sine wave depending on whether STPSIN is TRUE or FALSE, respectively. YINTEG, the output of TRAN, is the Butterworth filter output. P and Q are the numerator and denominator coefficients of the continuous filter as configured in Equation (5) of Section 2. TRAN is a standard ACSL system macro [1].

In DERIVATIVE SLOW the digital filter responses are calculated. Instead of specifying integration step size, MAXT2 has a value that is equivalent to the sampling interval for the digital filters. Setting IALG to 4 specifies the RK2 integration algorithm is to be used in this DERIVATIVE section. The RK2 algorithm has no use in this section other than to verify that ZXSFRM is working properly in a DERIVATIVE section employing a multi-step integration algorithm. TS is the local time in this DERIVATIVE section. The logical variable MMTHD is used to select one of two canonical forms, the M-method or the Rectangular method. YINV is the digital filter output for the impulse invariant method and YBILNR is the digital filter output for the bilinear method. XZFER is the digital filter input signal. YANALS, the analytical continuous filter response is calculated according to Equations (3) and (4) of Section 2.

```
IF (ZZFST(TZ).LT.0.5) Go to SN002
:
:
SN002...CONTINUE
```

jumps over the YANALS and XZFER calculations unless the ACSL Executive is executing the first minor step of the multistep integration algorithm. This code is implemented to ensure that YANALS and YINV and YBILNR and XZFER are synchronized in time, in order to compare the phases of the transient response data. It is to be noted that the positions in time of YINV and YBILNR correspond to a zero delay in the time between when the input is received in the filters and when the output is calculated and sent out. This causes no problems inside a digital computer but if these values were D to A'd to a continuous system there is some inherent lag which probably should be represented. Since the digital filter input and output equations are calculated only at the front of a major integration step they will appear in line printer listings and plots to have a one time step delay. However, values for YINTEG, YANALF and XINTEG do match correctly with output time values.

DERIVATIVE RECORD is used specifically for recording data at a faster rate (without changing CINT) in order to show the staircase nature of the digital filters as seen by a continuous system. As mentioned above no time delays due to the time used for the actual calculation of the filters, algorithms are modeled. Time skewing between digital filter inputs and outputs is due solely to the intrinsic features of the filter algorithms themselves. YANALR is the analytical filter output calculated in DERIVATIVE RECORD which is used in plotting as a comparison with the staircase digital filter outputs. MAXT3 is approximately 2 msec which will be the data recording interval. It is made slightly greater than 2 msec to make sure the digital filters get iterated on the first minor step in DERIVATIVE SLOW before a data logging action takes place. IALG is made 3 which corresponds to the synchronized Euler integration (one DERIVATIVE calculation per integration step) for this section. If RECORD is TRUE, a data logging action takes place through the call to ZZLOG

The simulation was run interactively on a CYBER 74 using ACSL run time commands entered on a Tektronix console. Hard copy plots were generated along line printer listings.

The interactive session setup commands are given in Section 3. ACSL run time commands that were used every session were put on a file. These commands are listed in *Figure 30.* Discussion of this run time command file will center on the PROCED's as the other commands are the same as those discussed in Section 3.

ACTION 'VAR'=0.039, 'VAL'=20, 'LOC'=NDBUG

causes a debug printing starting at 0.039 secs or thereafter with NDBUG=20. 'PROCED GO' is much the same as that used in Section 3 with the addition of an ACTION 'CLEAR' statement since only the first run is to have a debug printing.

'PROCED MSTEP' and 'PROCED RSTEP' configure the simulation for running the M-method and Rectangular method implementations of the digital filters, respectively, for a step input. The analytical and numerical integration values of the continuous filter are always calculated. Similarly MSIN05 through MSIN50 run the simulation with 5 Hz through 50 Hz sine waves for the M-method implementation. RSIN40 sets the simulation up with a 40 Hz sine wave input and the rectangular implementation method for the digital filters.

7 Hz is the only filter break frequency used in the following runs. *Figure 31* through *33* are generated by the following run time commands,

```
MSTEP
GO
```

```
SET PRN=9
SET PRNPLT=.FALSE., CALPLT=.TRUE., TTLCPL=.TRUE.
PREPAR T,YANALS,YANALF,YINTEG,YINV,YBILNR,XZFER,XINTEG,TS,TF
PREPAR T,TR,YANALR
SET DUMP=.T.
ACTION "VAR"=0.039,"VAL"=20,"LOC"=NDBUG
SET TITLE(1)="15 AUG 79-3RD ORD BTRWTH-7HZ"
PROCED GO
START
ACTION "CLEAR"
PRINT "ALL"
DISPLY U,V,R,S $ "DISPLAY DIG FILT COEFFS"
PLOT "XHI"=T
END
PROCED MSTEP
SET TITLE(4)="STEP INPUT  M-METHOD                                    "
SET MMTHD=.TRUE.
SET STPSIN=.TRUE., TSTP=0.5, CINT=0.010
END
PROCED RSTEP
SET TITLE(4)="STEP INPUT  RECTANGULAR METHOD                          "
SET MMTHD=.FALSE.
SET STPSIN=.TRUE., TSTP=0.5, CINT=0.010
END
PROCED MSIN05
SET TITLE(4)="SINE INPUT 5HZ, M-METHOD                                "
SET MMTHD=.TRUE.
SET STPSIN=.FALSE., TSTP=1.0, CINT=0.010, WCPS=5.0
END
PROCED MSIN10
SET TITLE(4)="SINE INPUT 10HZ, M-METHOD                               "
SET MMTHD=.TRUE.
SET STPSIN=.FALSE., TSTP=0.5, CINT=0.010, WCPS=10.0
END
PROCED MSIN20
SET TITLE(4)="SINE INPUT 20HZ, M-METHOD                               "
SET MMTHD=.TRUE.
SET STPSIN=.FALSE., TSTP=0.4, CINT=0.010, WCPS=20.0
END
PROCED MSIN40
SET TITLE(4)="SINE INPUT 40HZ, M-METHOD                               "
SET MMTHD=.TRUE.
SET STPSIN=.FALSE., TSTP=0.4, CINT=0.010, WCPS=40.0
END
PROCED RSIN40
SET TITLE(4)="SINE INPUT 40HZ, RECTANGULAR METHOD                     "
SET MMTHD=.FALSE.
SET STPSIN=.FALSE., TSTP=0.4, CINT=0.010, WCPS=40.0
END
PROCED MSIN47
SET TITLE(4)="SINE INPUT 47HZ, M-METHOD                 "
SET MMTHD=.T.
SET STPSIN=.F., TSTP=0.5, CINT=0.010, WCPS=47.
END
PROCED MSIN50
SET TITLE(4)="SINE INPUT 50HZ, M-METHOD                               "
SET MMTHD=.TRUE.
SET STPSIN=.FALSE., TSTP=0.4, CINT=0.010, WCPS=50.0
END
SET CMD=DIS
```

Figure 30. A standard set of ACSL run time commands that were stored on file INPUT.

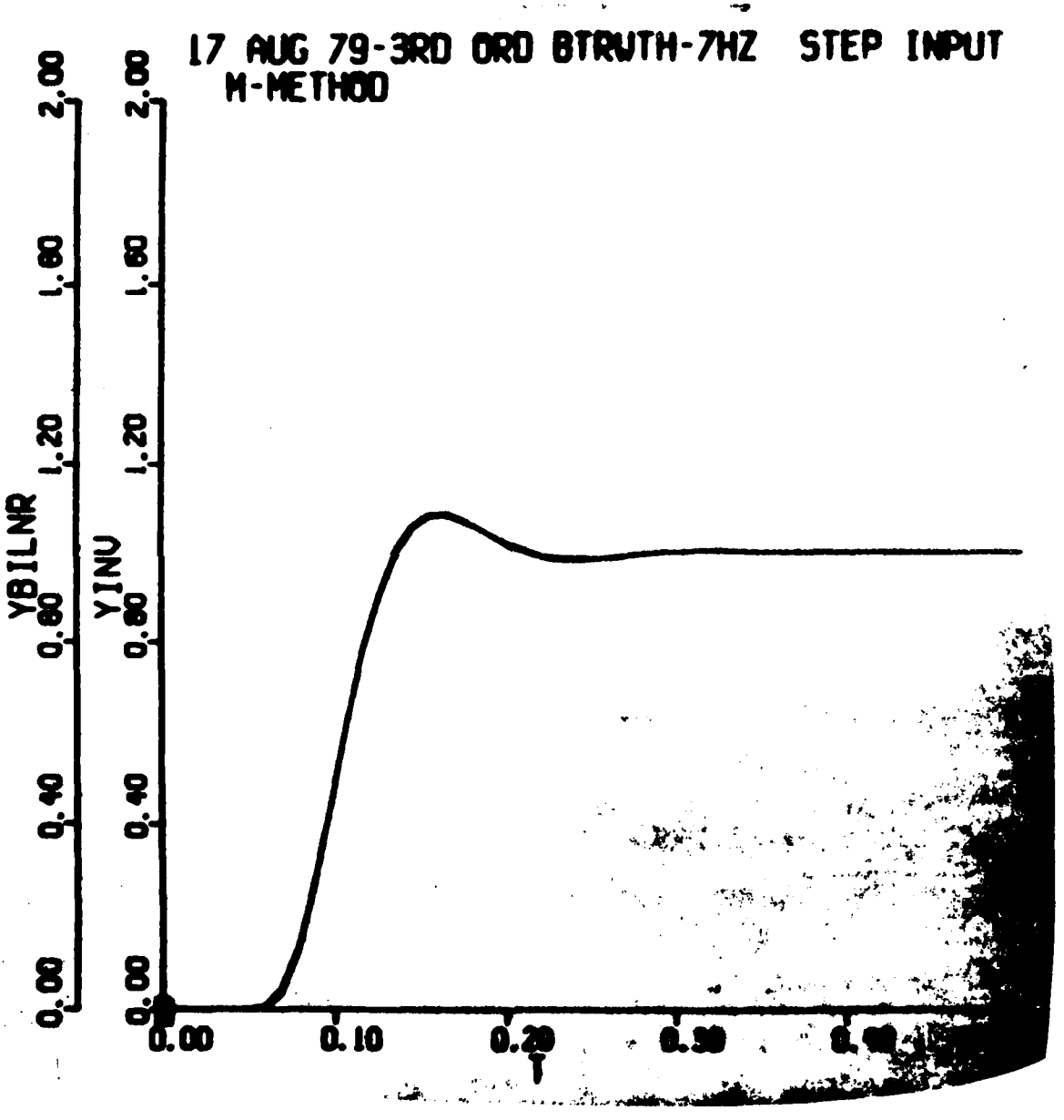


Figure 31. Unity step input response for the bilinear and impulse invariant digital filters with a 10 msec sampling interval.

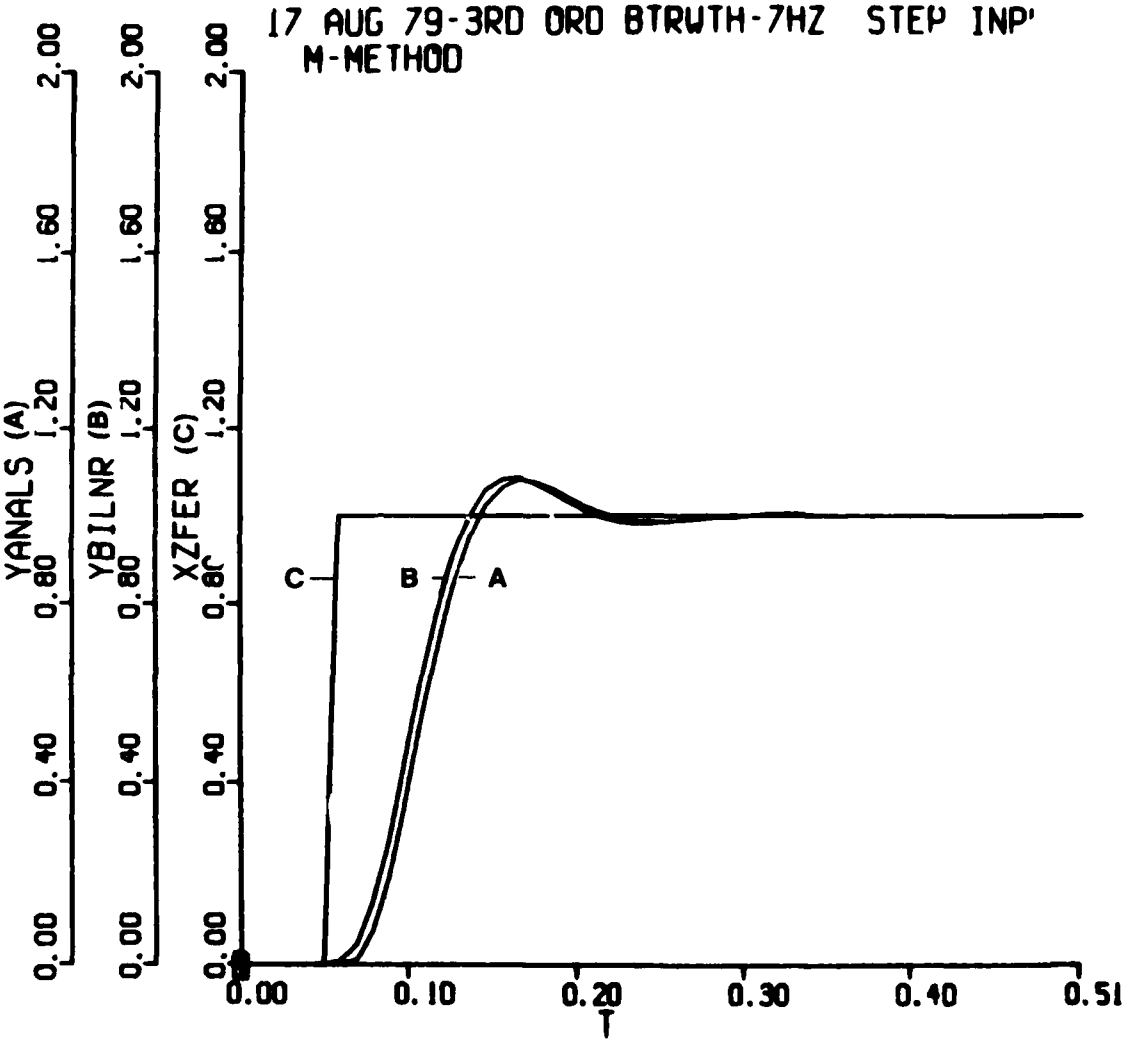


Figure 32. Plot of filter input and resulting bilinear digital and analytical filter response using a 10 msec sampling interval.

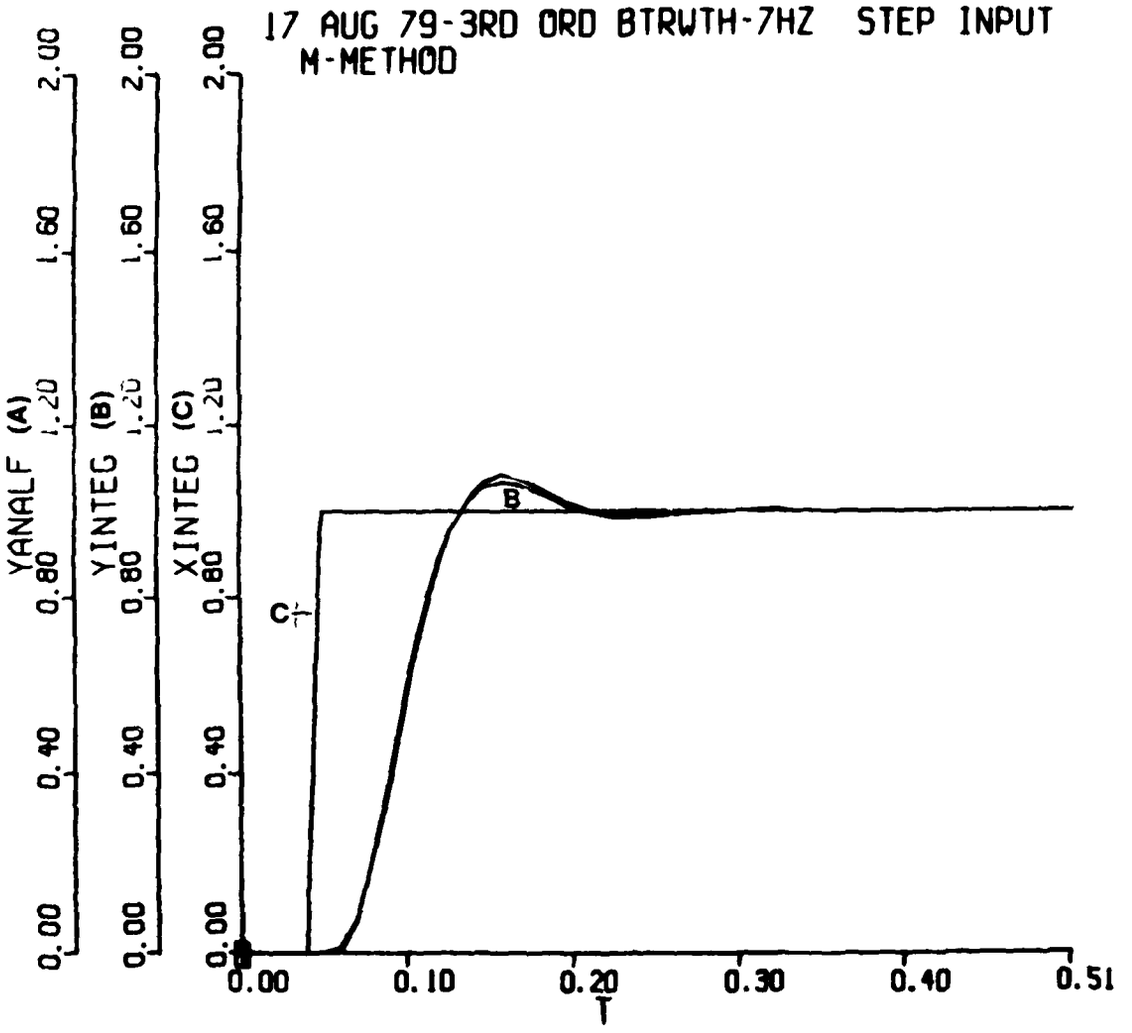


Figure 33. Plot of filter input and resulting numerical integration and analytical filter responses using a 10 msec integration step size.

```
PLOT YBILNR, YINV
PLOT YANALS, YBILNR, XZFER, 'HI'=2.
PLOT YANALF, YINTEG, XINTEG, 'HI'=2.
```

Figure 31 shows that the two z-transform methods give essentially the same transient response to a unit step. *Figure 32* is a comparison plot between the analytical and bilinear filter values. The bilinear filter leads the analytical filter response by about 5 msec. *Figure 33* shows how well the analytical and numerical integration responses compare. YINTEG is a bit low at its peak amplitude but it has excellent phase matching. Special effort was made to synchronize all the filter implementations with their analytical responses. In the program, TZ was made slightly less than 0.05 sec to ensure that the STEP macro went high at 0.05 sec. So observed phase differences should be accurate and not a function of program calculations and I/0 functions.

```
RSTEP $ GO
PLOT YBILNR, YINV
PLOT YANALS, YBILNR, XZFER, 'HI'=2.
```

generates data using the Rectangular method canonical form. Plots are shown in *Figures 34* and *35*. The results are identical to the above where the M-method canonical form was used to implement the digital filters. Thus both canonical forms give, as they should, the same digital filter response.

```
SET RECORD=.T. $ GO $ PLOT YANALR, YBILNR
```

reruns the simulation and causes the data recording interval to drop to 2 msec which is used to bring out the staircase nature of YBILNR as would be seen by a continuous system. YANALR is simply the analytical filter value updated every 2 msec. Although the digital filter actually leads the analytical filter by about one-half of the digital filter sampling interval, *Figure 36* a continuous system would see a staircase function that is practically in phase with the analytical filter response. Of course, the delay in YBILNR, due to the finite time needed to calculate the filter algorithm, has been neglected. To a continuous system a discrete filter signal in phase with the continuous analytical filter signal would appear lagging in phase by one-half of a sampling interval as shown in *Figure 37*.

```
MSINO5 $ SET RECORD =.F.$GO
PLOT YBILNR, YINV
```

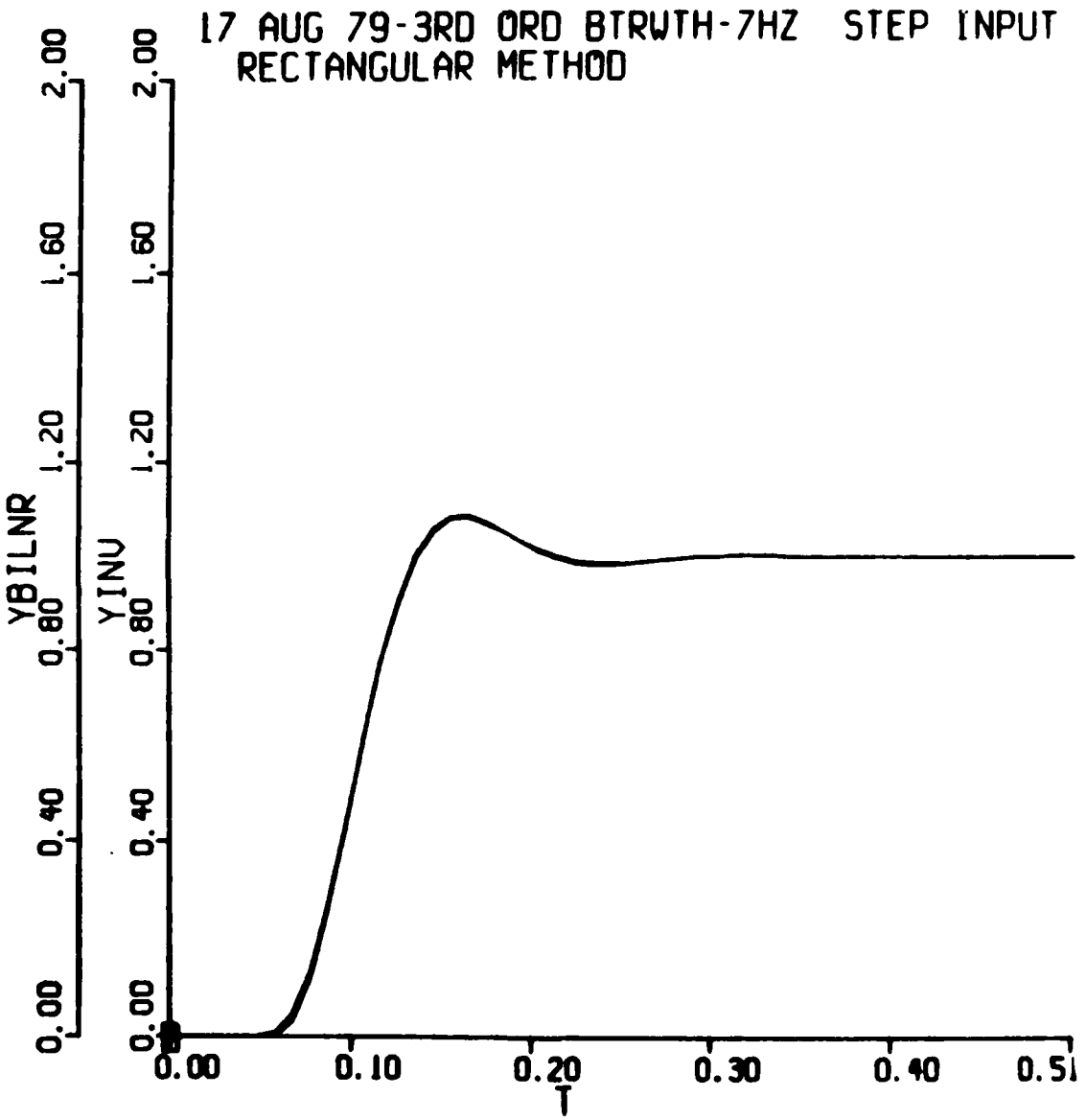


Figure 34. Unity step input response for the bilinear and impulse invariant digital filters implemented in the rectangular canonical form with a 10 msec sampling interval.

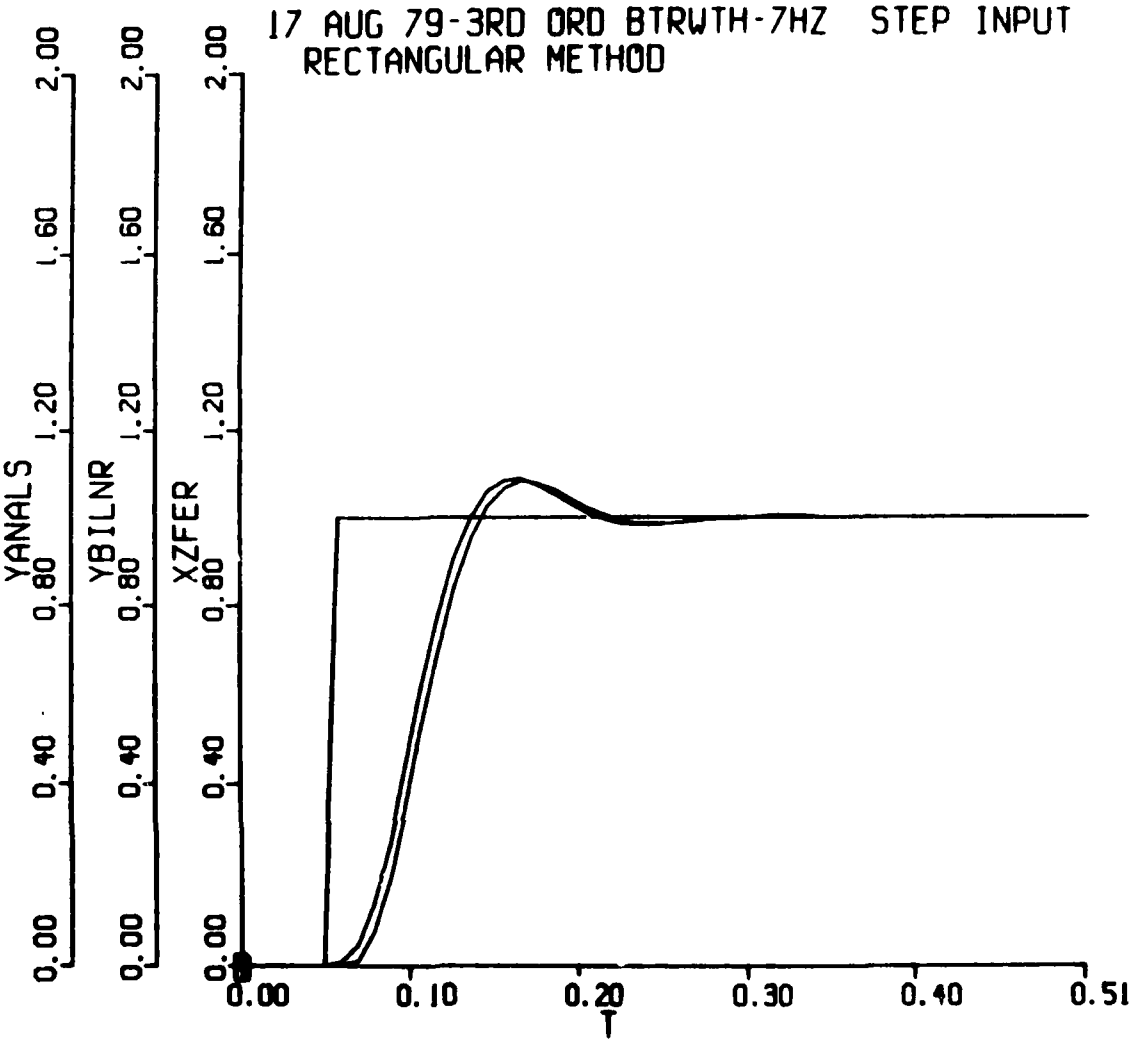


Figure 35. Plot of filter input and corresponding responses from the bilinear digital filter using the rectangular canonical form and the analytical filter with a 10 msec sampling interval.

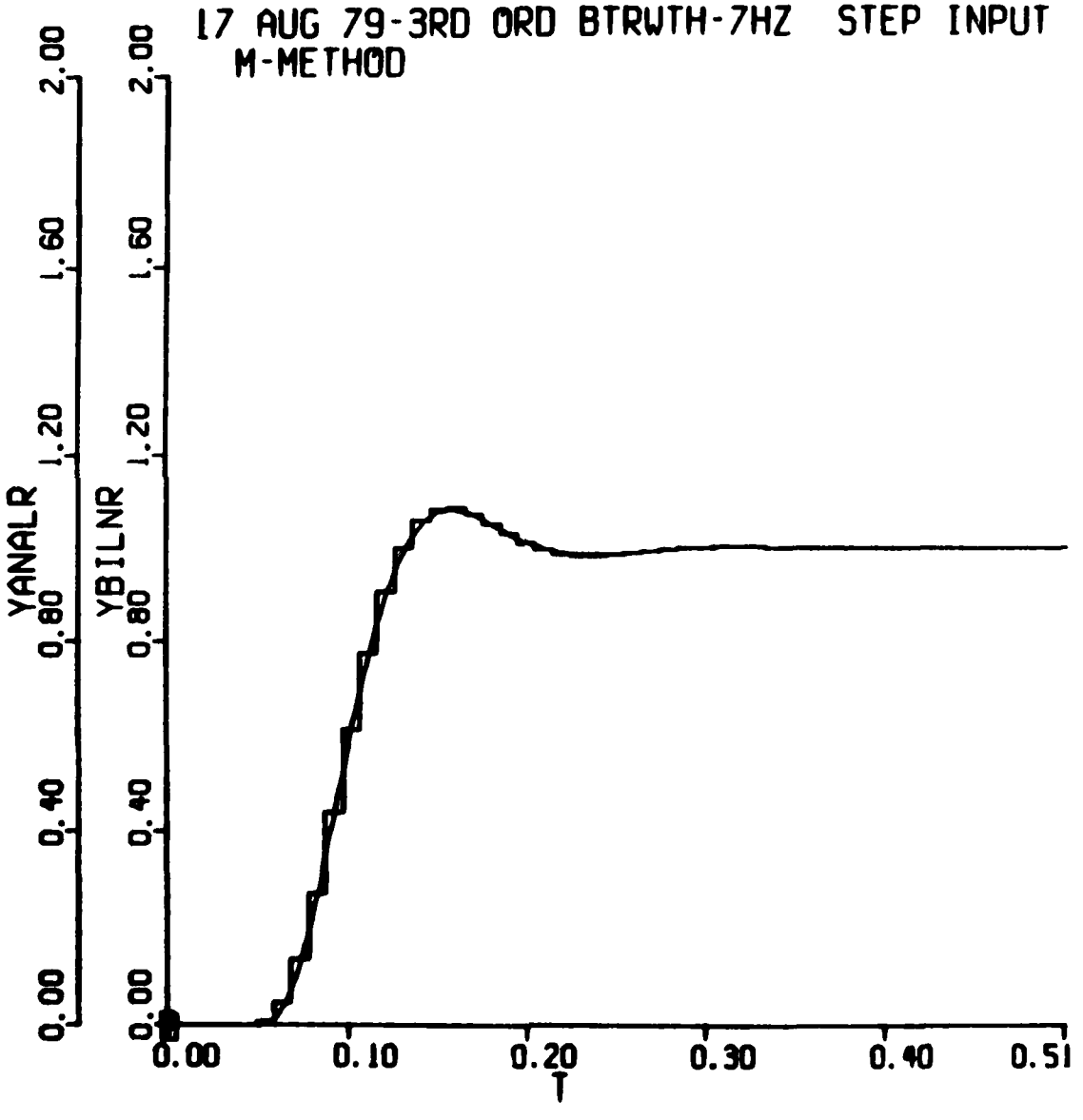


Figure 36. Comparison of the discrete representation of the bilinear filter with the continuous analytical filter response-10 msec sampling interval.

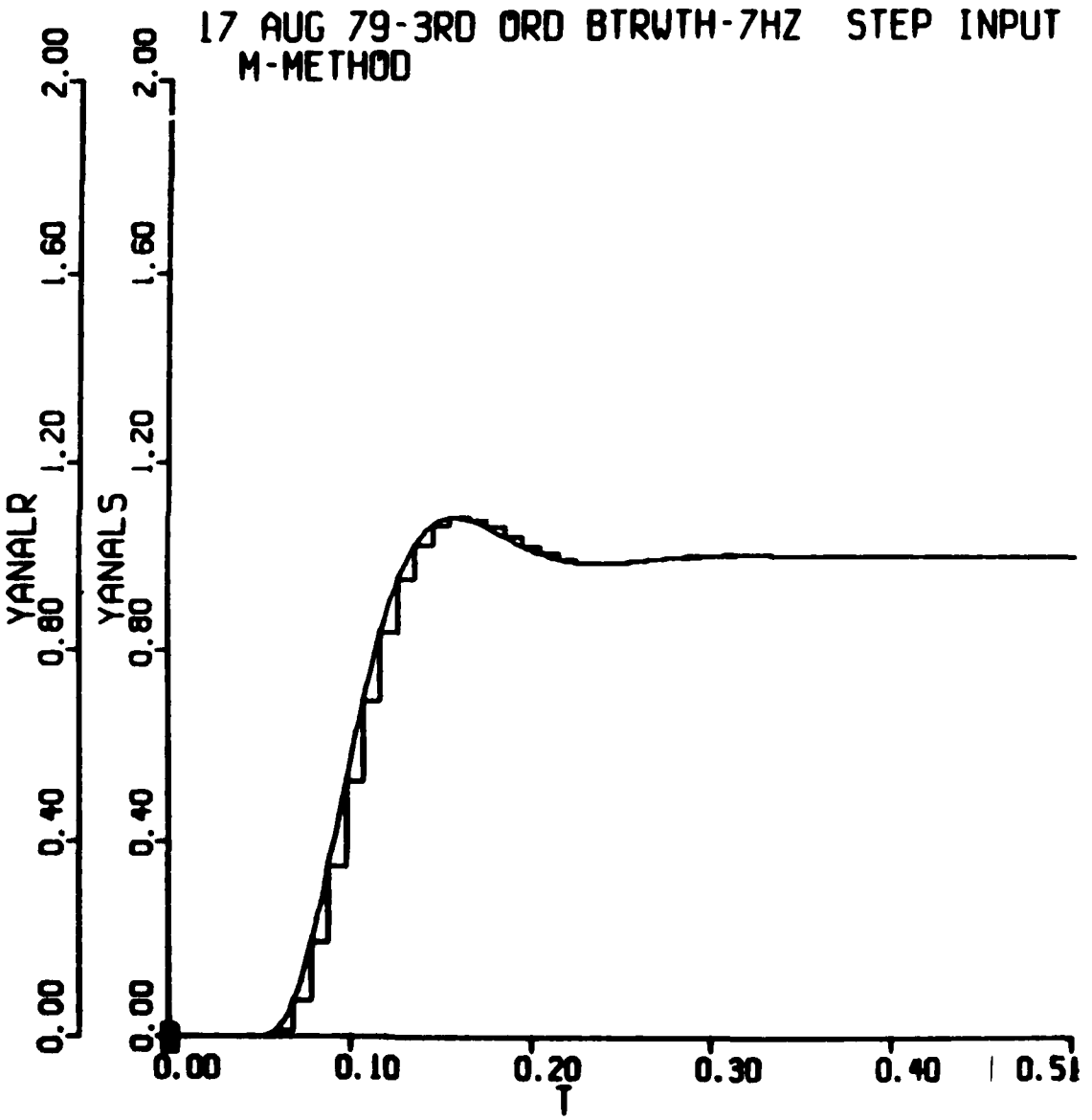


Figure 37. Comparison of a discrete signal with a continuous signal where the two signals are in phase –10 msec sampling interval.

PLOT YANALS, YBILNR, XZFER
PLOT YANALF, YINTEG, XINTEG

runs the simulation with the 5 Hz sine wave input and with the M-method canonical form for the digital filters. *Figures 38* through *40* are generated from these commands. *Figure 38* shows that the bilinear and impulse invariant digital filters compare quite well. *Figure 39* shows that, except for an initial lead transient of about 5 or 6 msec in the bilinear filter, the analytical and bilinear responses are similar in amplitude and phase. *Table 1* lists the amplitudes and phases for all four filter representations. The analytical response is calculated in two ways:

- From the analytical expression as $t \rightarrow \infty$ and

- From calculations based on the transient response data. The digital integration method's amplitude and phase are also calculated based on transient response data. Bode plot data for phase and amplitude are listed in the table. All phase and amplitude calculations are based on expressions developed in Appendix B.

All three sources of data for the analytical filter's amplitude and phase give essentially the same results. The closeness of the two digital filters' amplitudes and phases, as calculated from transient response data, to both their Bode plot data and the analytical filter response can be seen in the table.

Similarly the numerical integration form of filter implementation has an output whose amplitude and phase match those of the analytical filter quite well as shown in *Figure 40* and *Table 1*. RECORD is set to TRUE in order to generate *Figure 41* where the discrete bilinear output is compared to the analytical filter as calculated every 2 msec.

Next a 10 Hz sine wave was input to the filters. *Figure 42* indicates that the two digital filters have similar phases but the amplitudes are now starting to differ because of the faster roll-off of the bilinear filter. *Figure 43* compares the invariant digital filter with the analytical response. The closeness with which the digital filters match their Bode amplitude and phase as well as those of the analytical response, can be seen in *Table 1*.

Figure 44 compares the numerical integration scheme with the continuous analytical filter The phase match is quite good but the amplitudes do not correspond as well. *Table 1* shows the phase of the numerical integration scheme has about 4 degrees of lead over the analytical filter and has approximately 16 percent more amplitude.

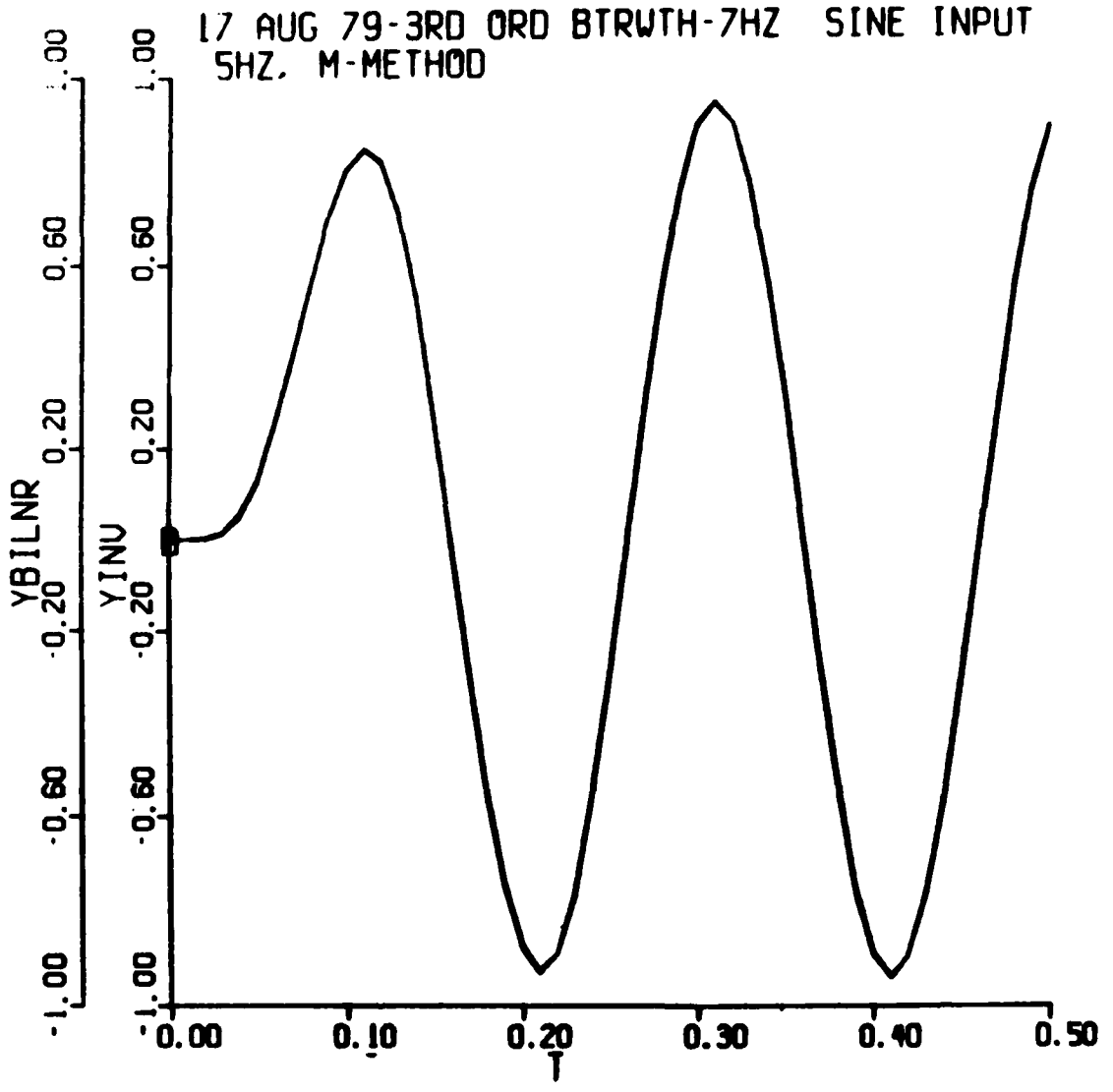


Figure 38. 5Hz sine wave input responses from the bilinear and impulse invariant digital filters with a 10 msec sampling interval.

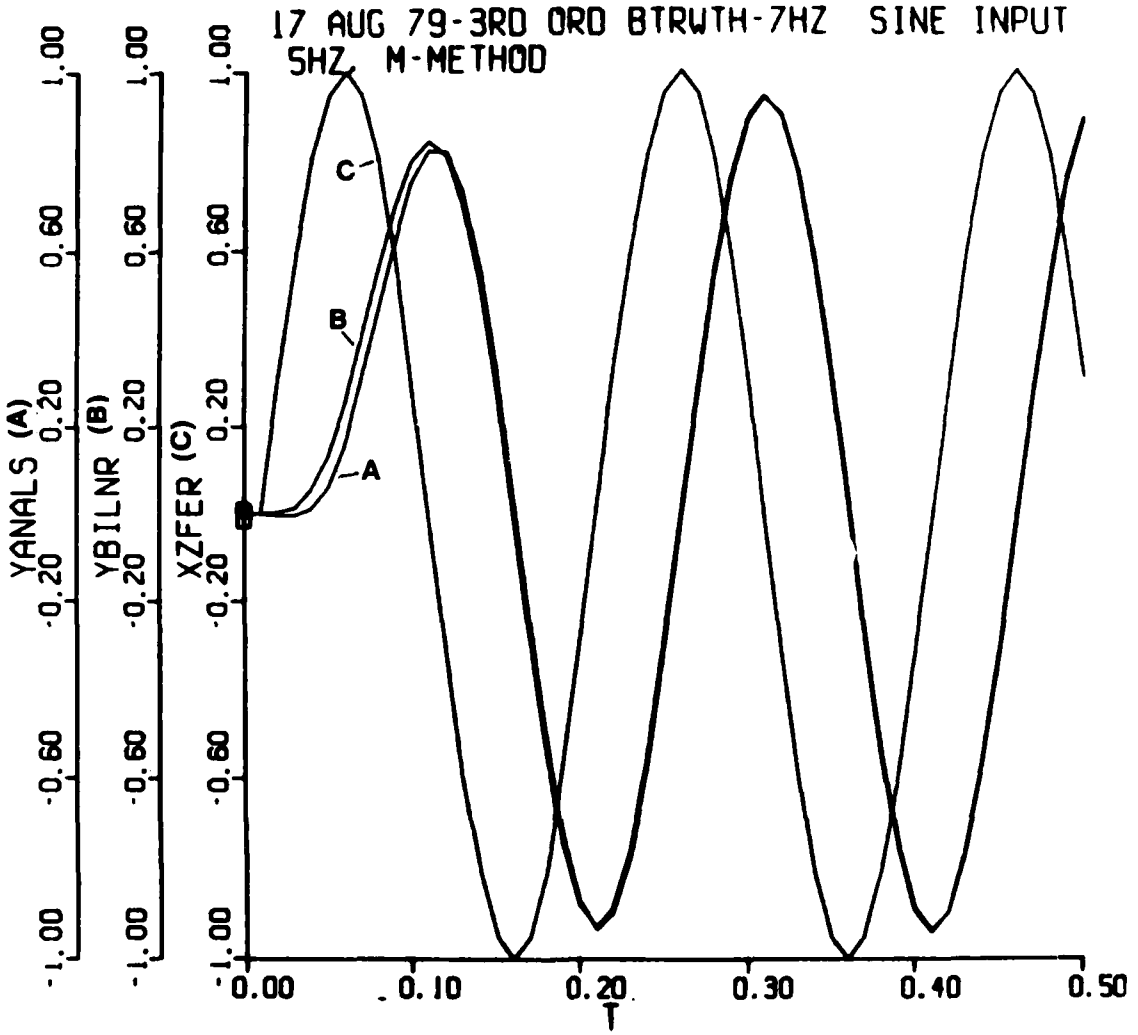


Figure 39. Plot of filter input and filter outputs for the bilinear digital and analytical representations with a 10 msec sampling interval.

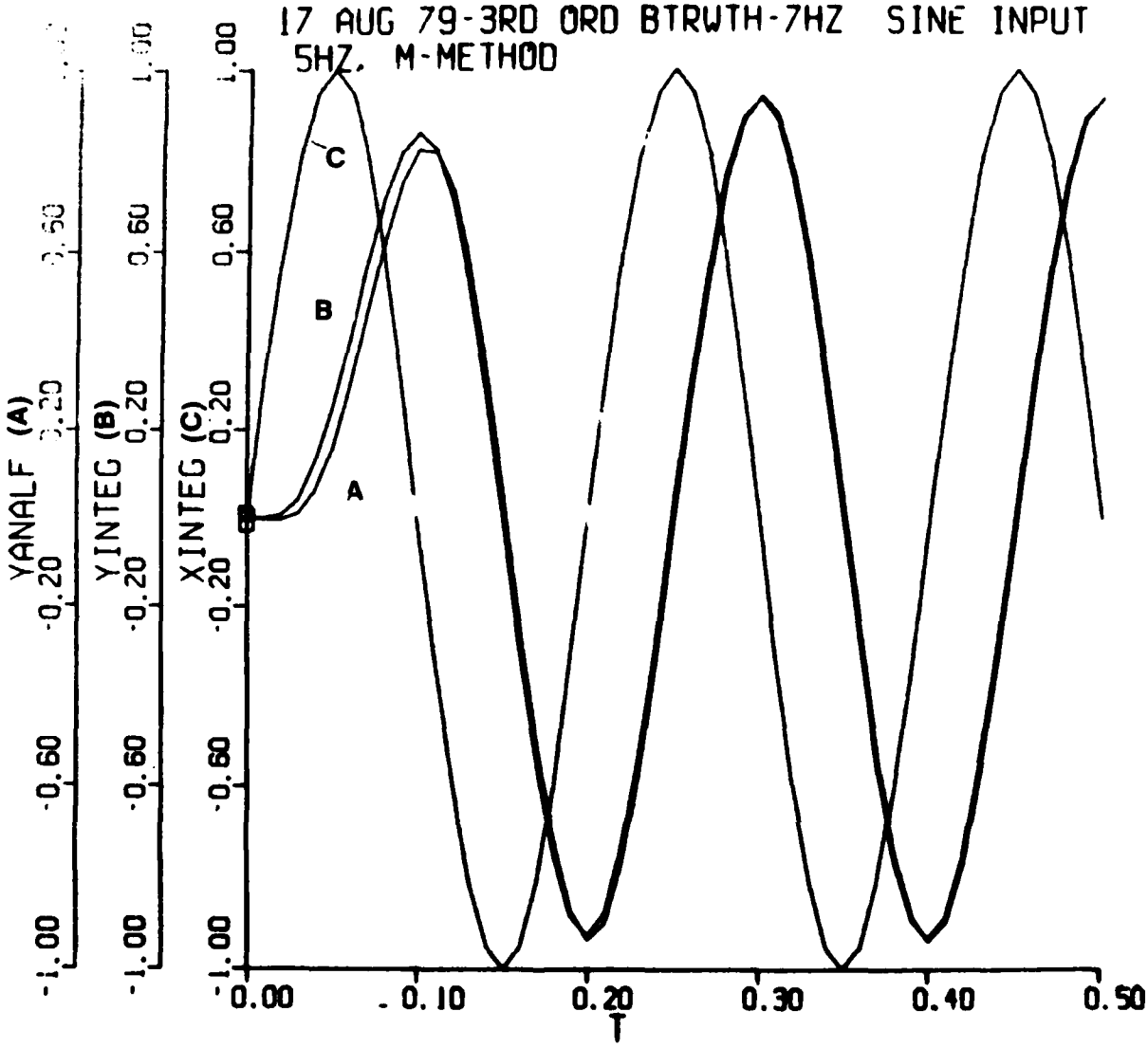


Figure 40. Plot of filter input and filter outputs for the integration method and the analytical expression with a 10 msec integration step size.

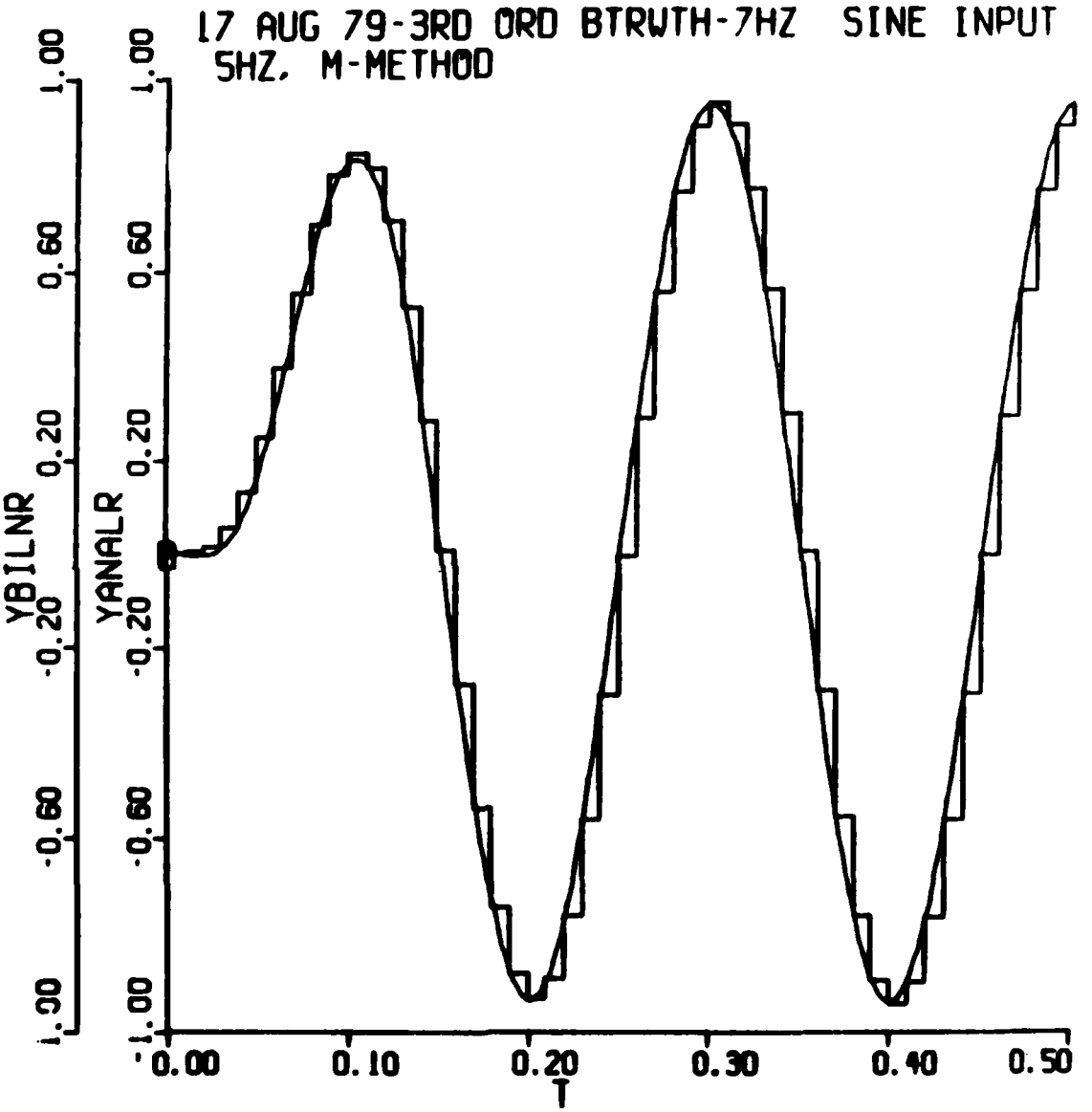


Figure 41. Comparison of the discrete representation of the bilinear filter with the continuous analytical filter response –10 msec sampling interval.

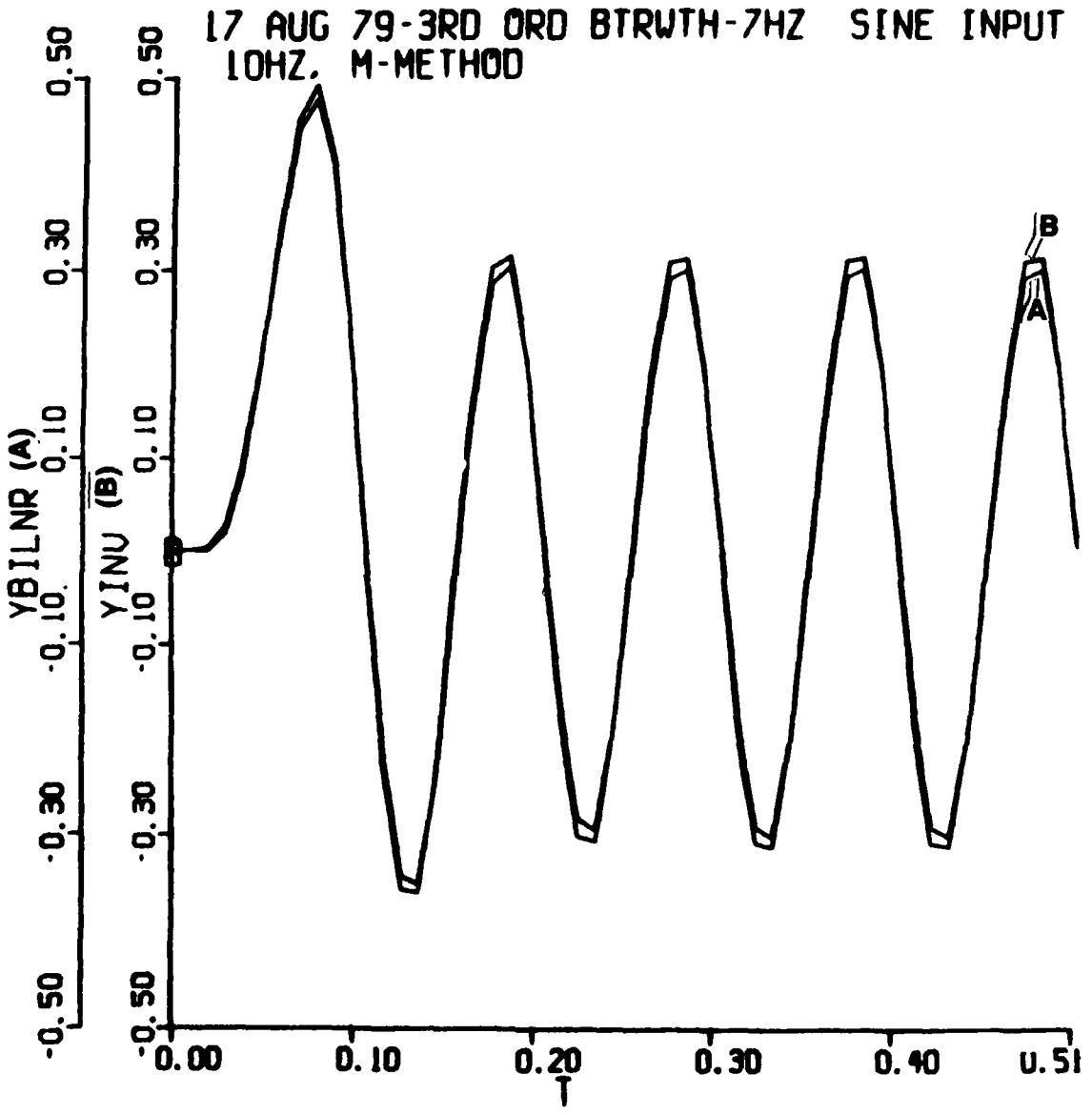


Figure 42. Plots of the impulse invariant and bilinear digital filters with a 10 msec sampling interval.

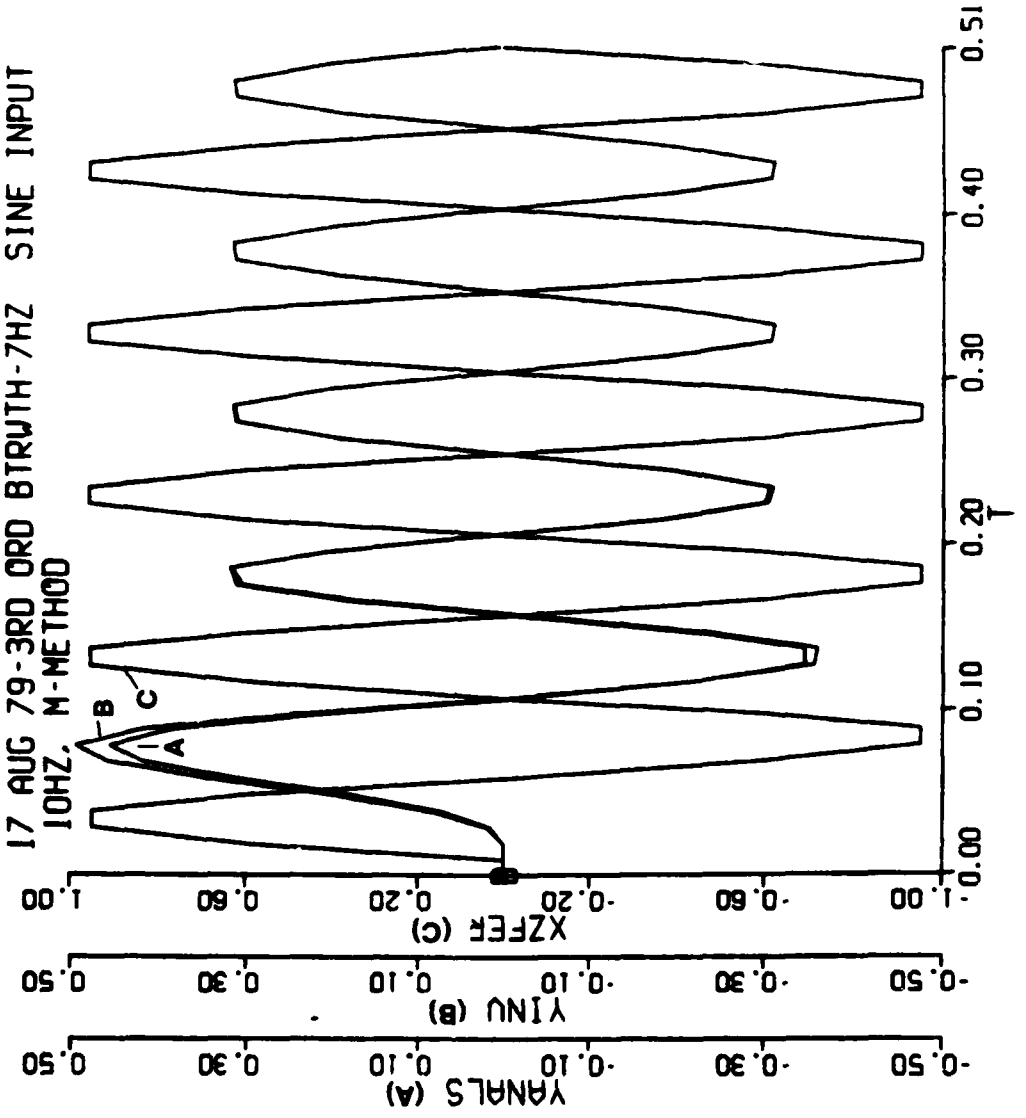


Figure 43. Plot of filter input and filter outputs for the impulse invariant method and analytical expression using a 10 msec sampling interval.

TABLE 1. AMPLITUDE AND PHASE COMPARISONS OF BODE PLOT RESULTS AND ESTIMATED STEADY STATE VALUES FROM THE TRANSIENT ANALYSIS PROGRAM FOR A UNITY GAIN, THREE POLE BUTTERWORTH FILTER WITH A 7 Hz BREAK FREQUENCY.

[ϕ]=deg
A is dimensionless

	5Hz		10Hz		20Hz		40Hz		47Hz	
	A	ϕ	A	ϕ	A	ϕ	A	ϕ	A	ϕ
YANAL 1)	.939	-91.1	.324	-181.	.0428	-229.	.00536	-250.	.00331	-253.
YANAL 2)	.940	-91.1	.324	-181.	.0430	-229.	.00535	-249.	00332	-252.
Bode Plot	.940	-91.1	.324	-181.	.0428	-229.	.00536	-250.	.00330	-253.
YEDS (YBILNR)	.942	-90.2	.310	-183.	.0292	-234.	3.78×10^{-4}	-264.	1.15×10^{-5}	-240.
Bode Plot	.942	-90.2	.310	-183.	.0291	-234.	3.83×10^{-4}	-262.	9.43×10^{-6}	-268.
YMINE YINV	.940	-91.1	.324	-181.	.0427	-229.	.00418.	-237.	00185	-214.
Bode Plot	.940	-91.1	.325	-181.	.0426	-228.	.00418	-238.	.00186	-211.
10 msec YINTEG	.937	-90.8	.375	-177.	.0680	-236.	.0151	-300. or -312.	.0108	-327. or -342.

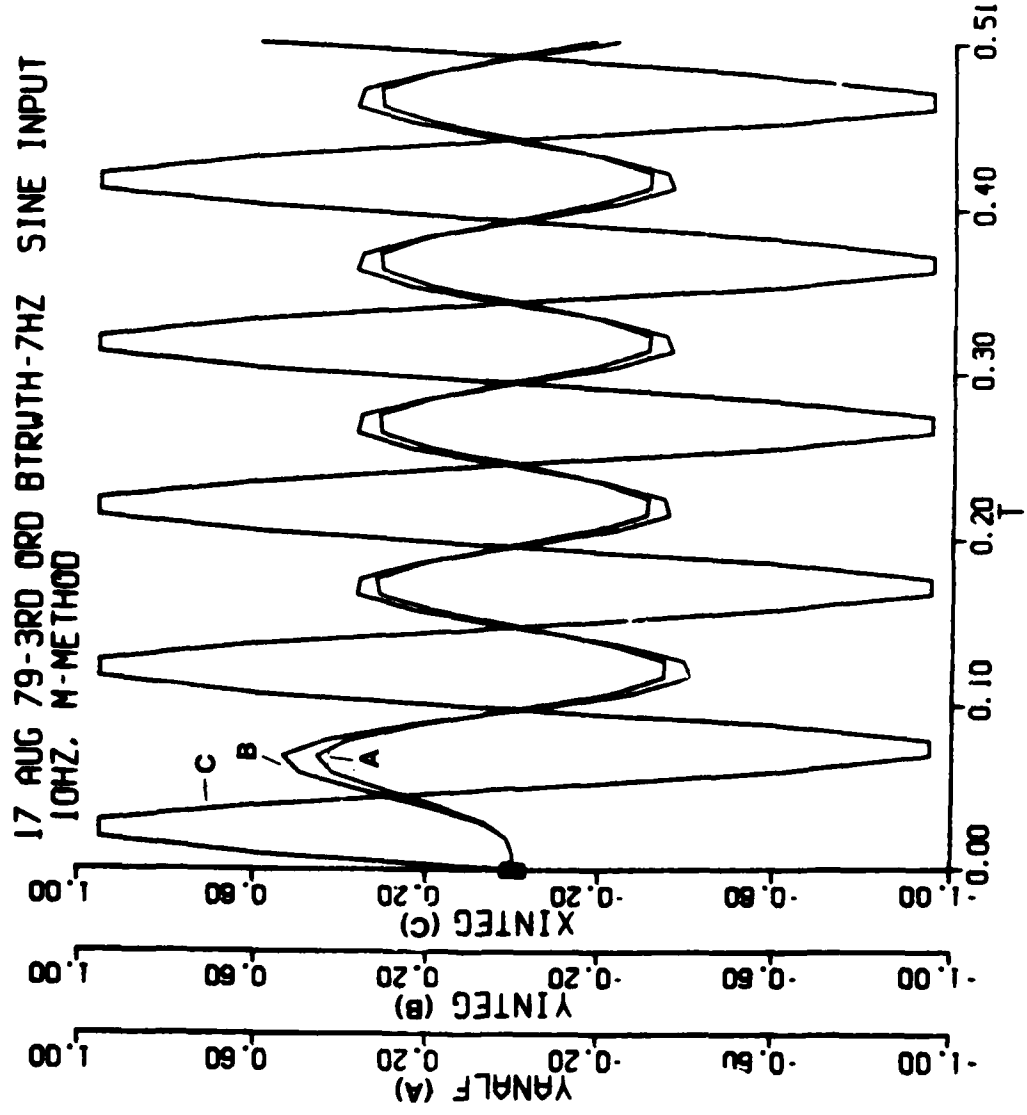


Figure 44. Plot of filter input and filter outputs for the numerical method and analytical expression with a 10 msec integration step size.

A 20 Hz sine wave was used to drive the filters. *Figure 45* is a comparison plot of the two digital filters. Again their phases match and their amplitudes compare with their respective Bode data as shown in *Table 1*. In *Figure 46* the invariant digital filter is compared with the analytical response. They are essentially the same after the first transient oscillation. *Figure 47* and *Table 1* show that although the integration method has 58 percent more amplitude than the analytical solution it lags the analytical phase by only 7 degrees.

A 40 Hz sine wave input resulted in digital filter outputs as shown in *Figure 48*. *Figure 49* compares the impulse invariant and analytical responses. Numerical values for amplitudes and phases are shown in *Table 1*. Notice how the sampled input sine wave bears little resemblance to its continuous counterpart. The sampled input essentially has two peaks that alternate in time and the filter outputs have the same look. The filters' amplitudes and phases were calculated from the larger of the two peaks (for example, points (3) and (4) in *Figure 48* since using low peak data (for example, points (1) and (2) in *Figure 48*) gave lower amplitudes.

Using points (3) and (4), Equations (B-6) and (B-7) in Appendix B give the amplitude as 0.00418 and the phase as -237 degrees which closely agrees with the Bode plot information (*Table 1*). However points (1) and (2) should work as well but their values for amplitude and phase are 0.00157 and -211 degrees respectively, considerably different, given how well the other data set matches the Bode plots. If the amplitude and phase derived from points (3) and (4) are used to calculate points (1) and (2) we obtain approximately the correct values. Obviously the amplitude and phase derived from points (1) and (2) will not generate points (3) and (4).

Figure 50 is a plot of the analytical response and integration method output. The integrated output is about three times larger than the analytical response and lags it by 50 or 62 degrees. The reason for the ambiguity in phase is discussed in Appendix B. The RK2 integration method does not work well at the 40 Hz input frequency because of the 10 msec integration step size. For a 40 Hz input an integration step of 4 msec would be more appropriate. However the digital filters are still performing well at the 10 msec sampling interval.

Figure 51 illustrates the large spacing of the sampling intervals for the 40 Hz input frequency where the plotted variables, YANALS and YANALR, are the analytical filter response calculated at a 10 msec interval and a 2 msec interval, respectively. In *Figures 49* and *50* YANALS were plotted by drawing straight lines between data points rather than holding a data point value until a new value was calculated.

For a 47 Hz sine wave the digital filters compare as shown in *Figure 52*. The sampled 47 Hz sine wave looks like a modulated signal as shown in *Figure 53* where the invariant and

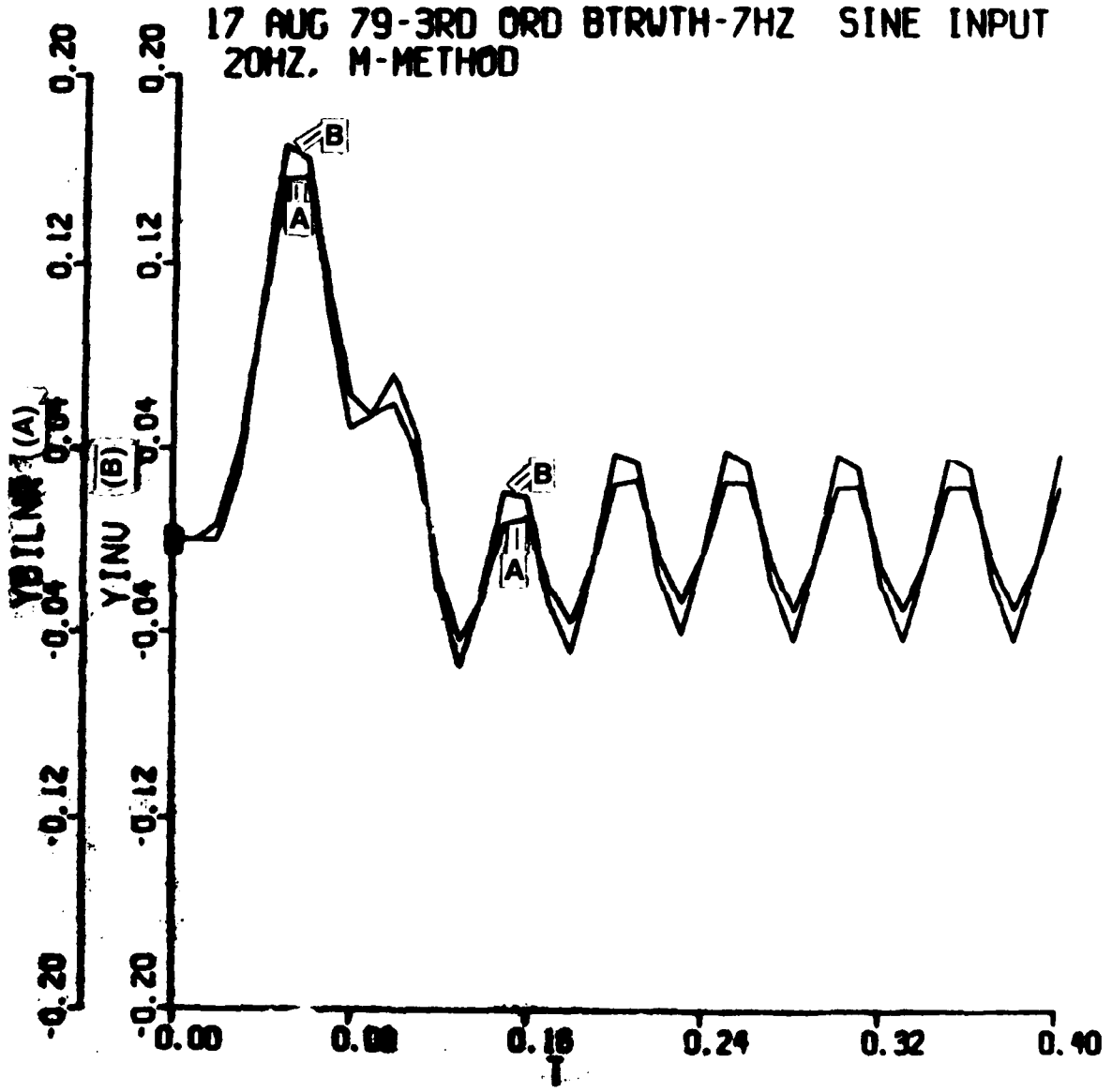


Figure 45. Plot of the impulse invariant and bilinear digital filters with a 10 msec sampling interval.

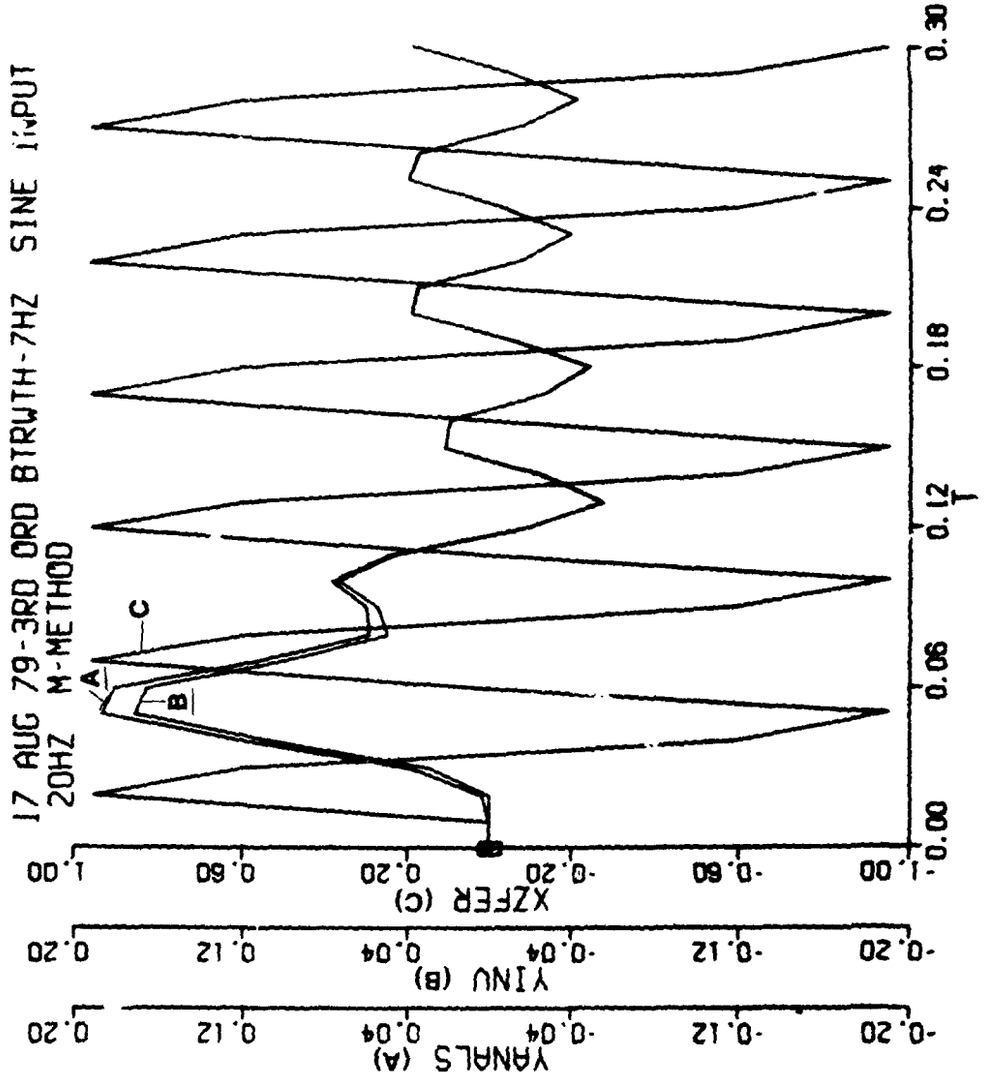


Figure 46. Plot of filter input and filter outputs for the impulse invariant method and the analytical expression with a 10 msec sampling interval.

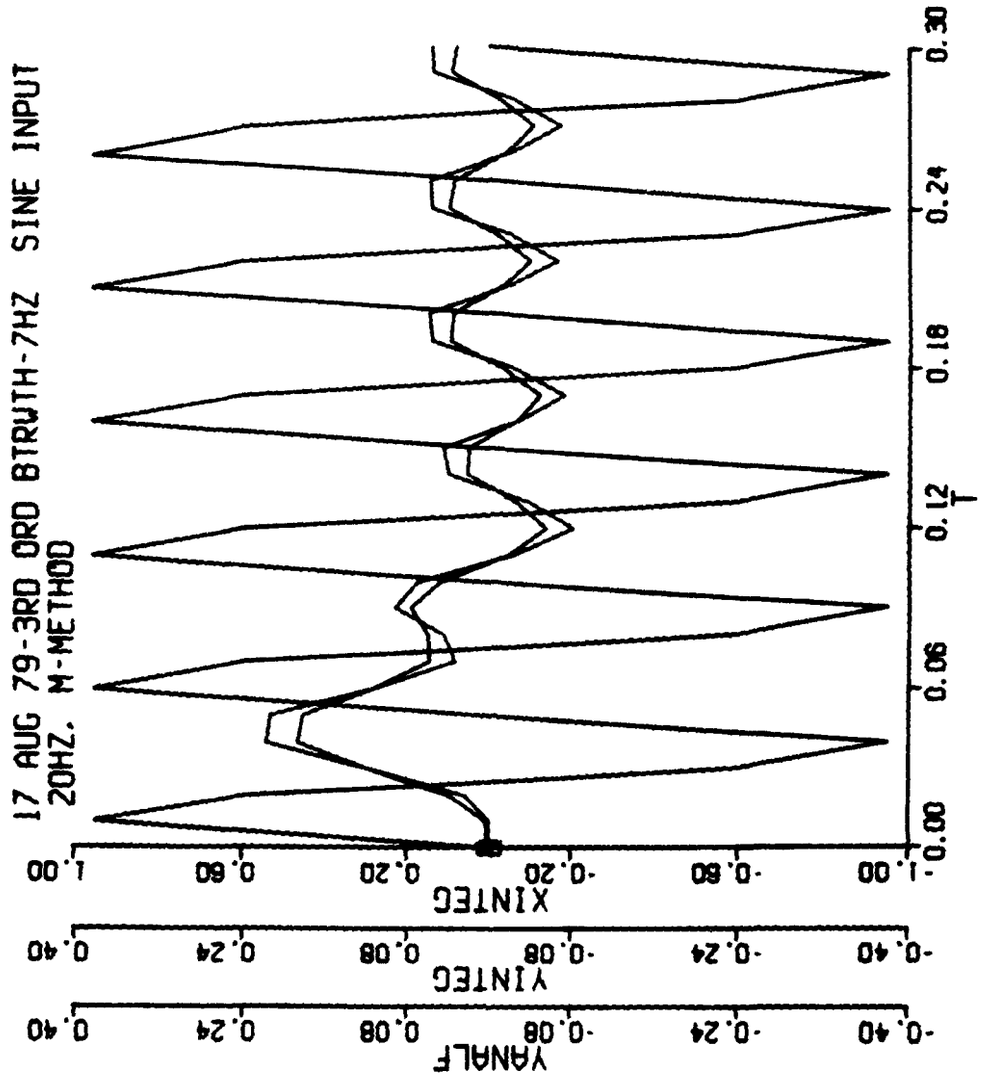


Figure 47. Plot of filter input and filter outputs for the numerical integration method and the analytical expression with a 10 msec integration step size.

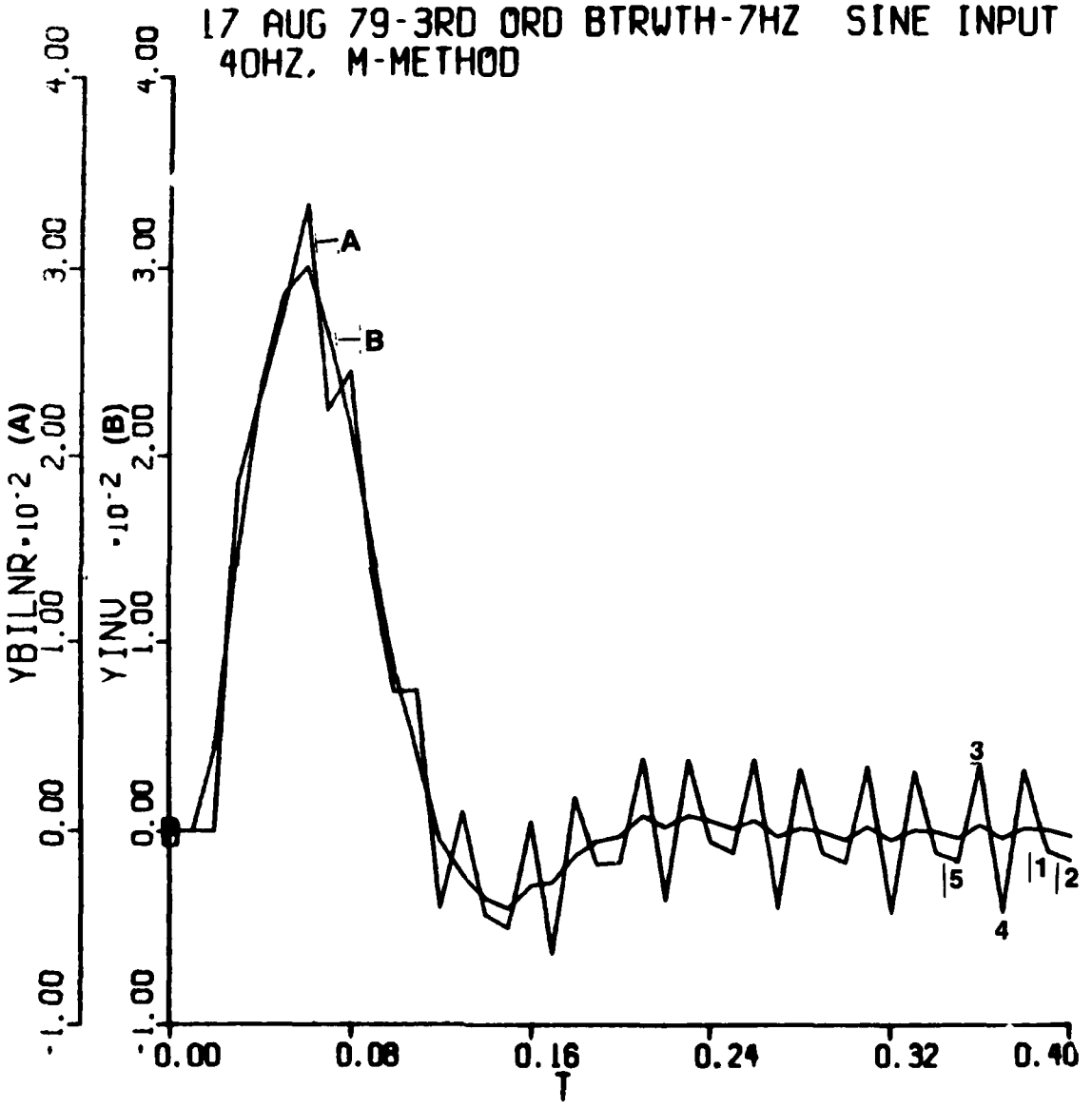


Figure 48. Plot of the impulse invariant and bilinear digital filters with a 10 msec sampling interval.

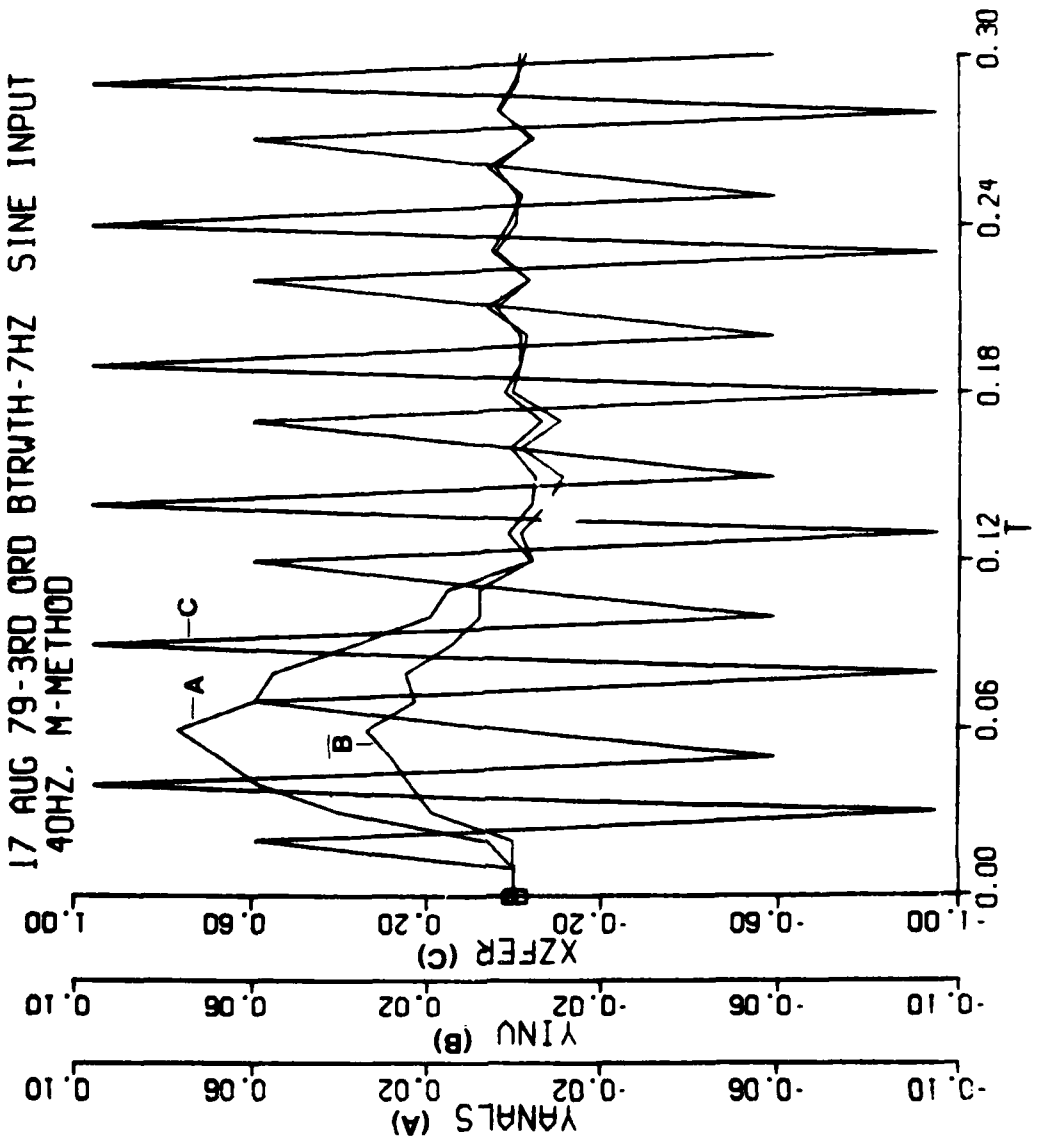


Figure 49. Plot of filter input and filter outputs for the impulse invariant method and the analytical expression with a 10 msec sampling interval.

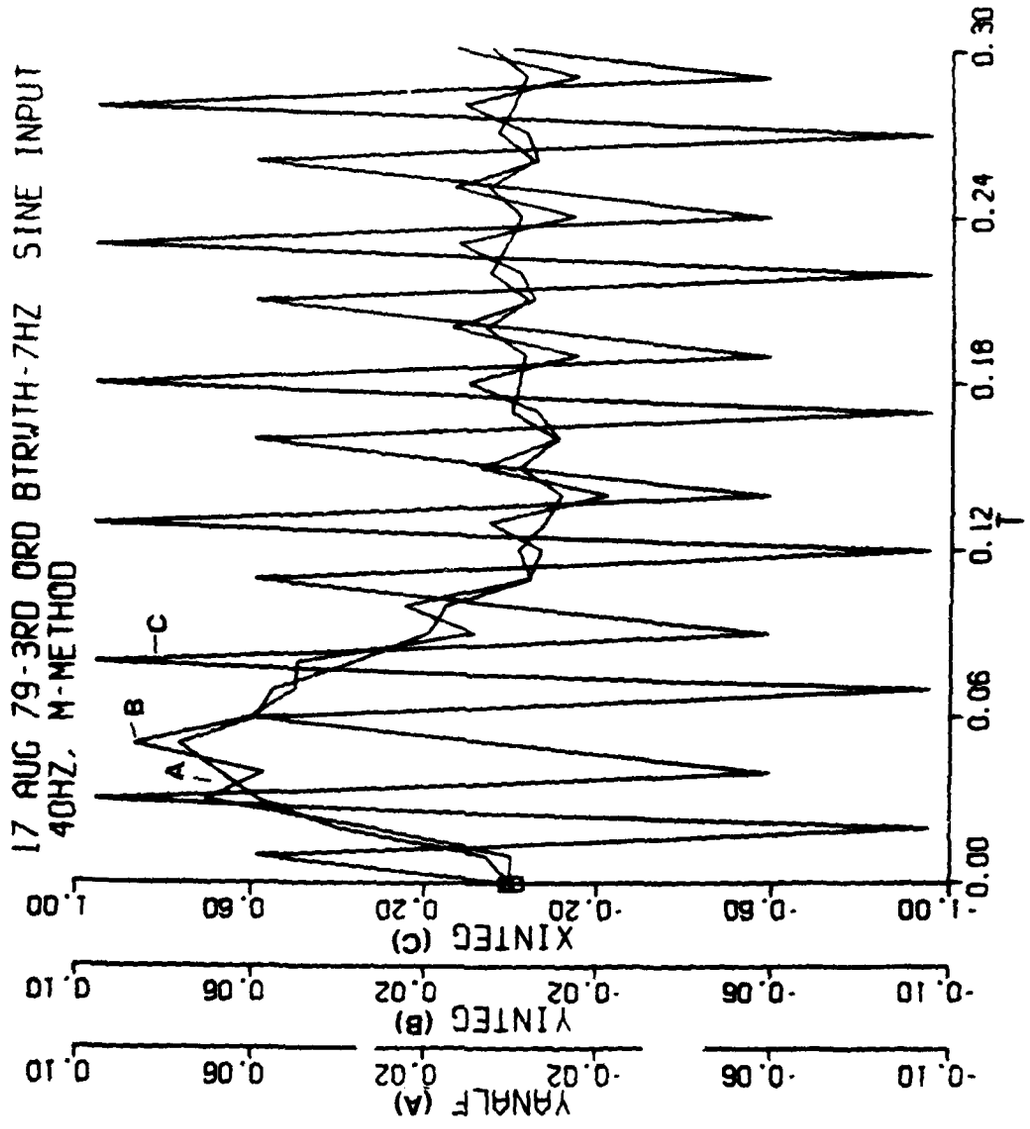


Figure 50. Plot of filter input and filter outputs for the integration method and the analytical expression .10 msec integration step size.

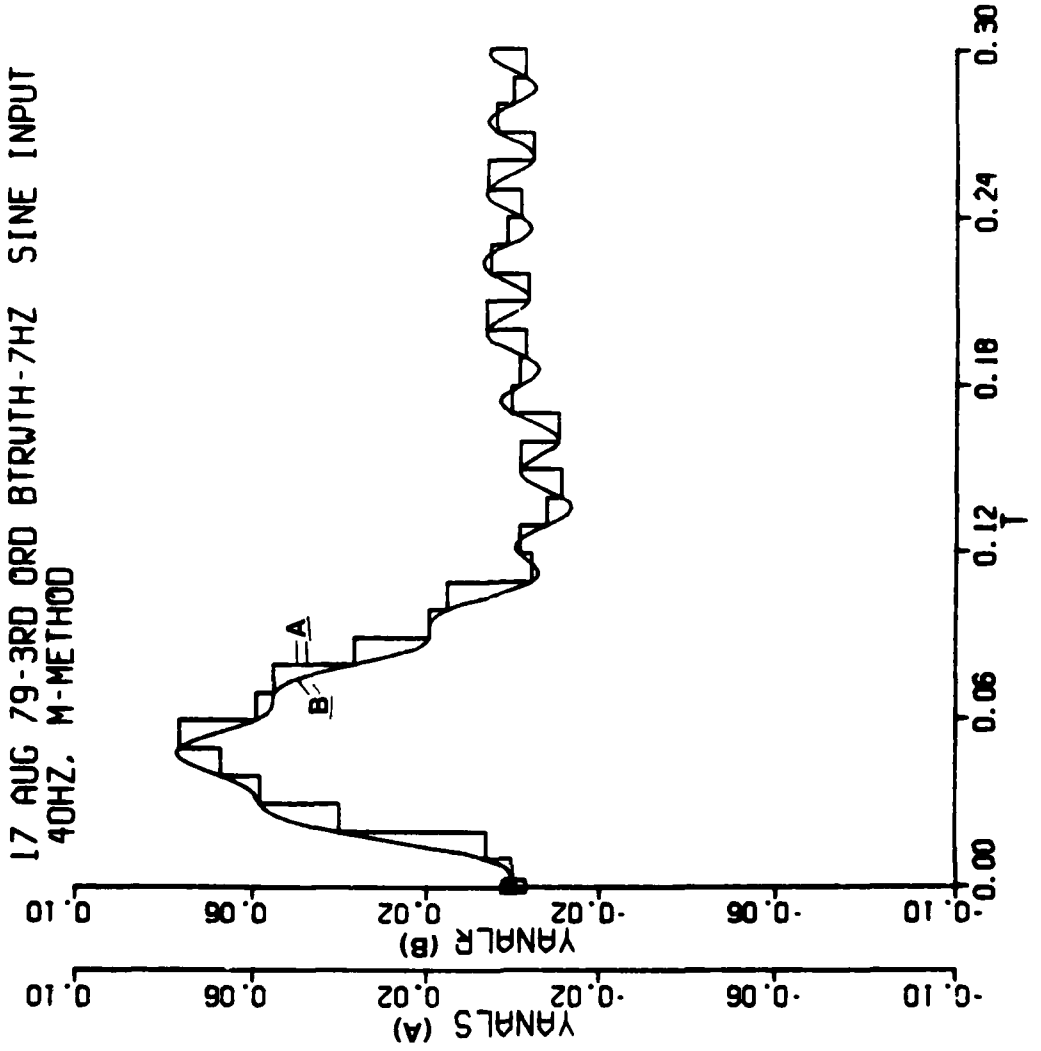


Figure 51. Illustration of how large the digital sampling interval is with respect to the frequency content of the filter output where the analytical filter output was used for plotting.

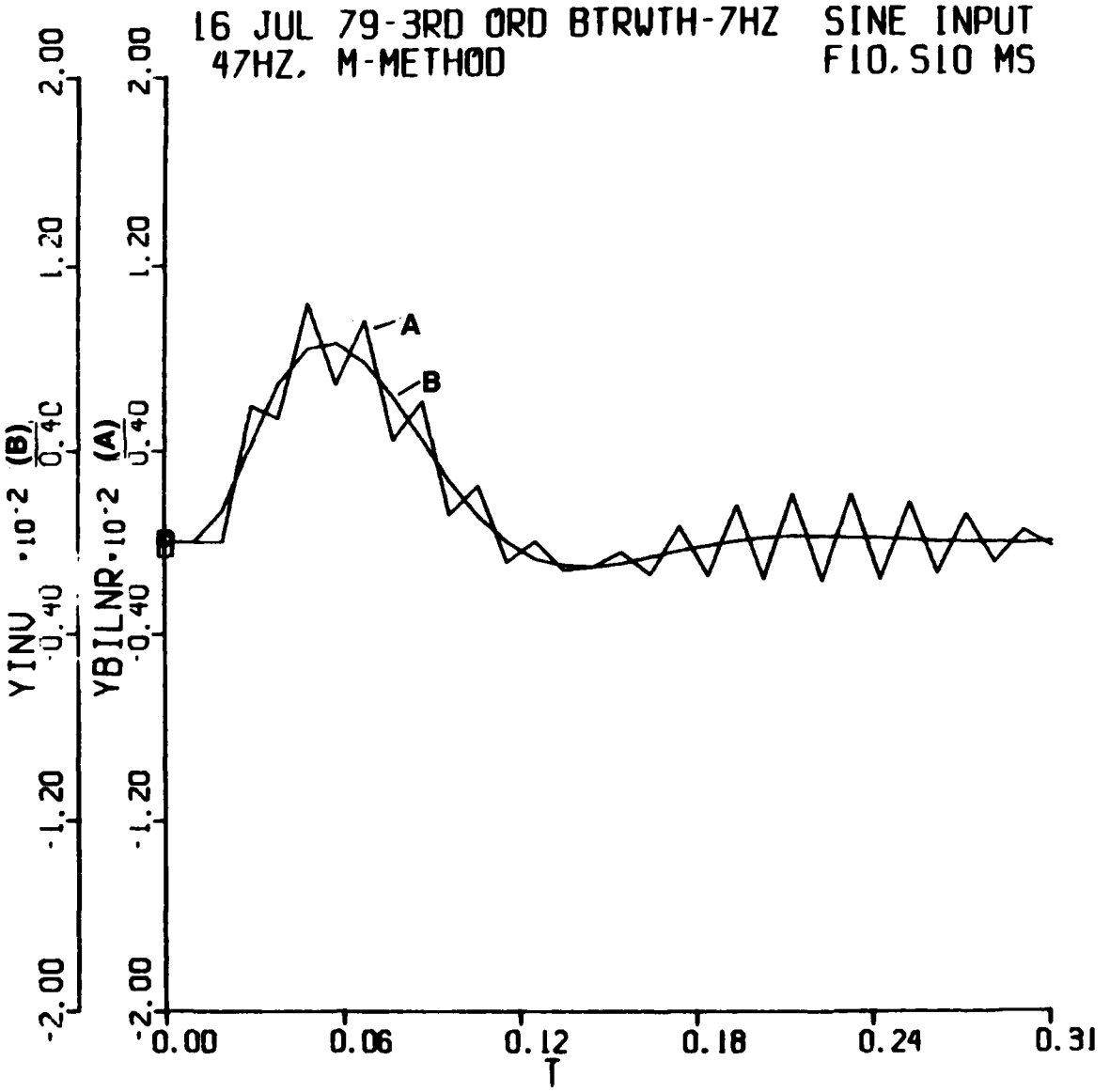


Figure 52. Plot of the impulse invariant and bilinear digital filters -10 msec sampling interval.

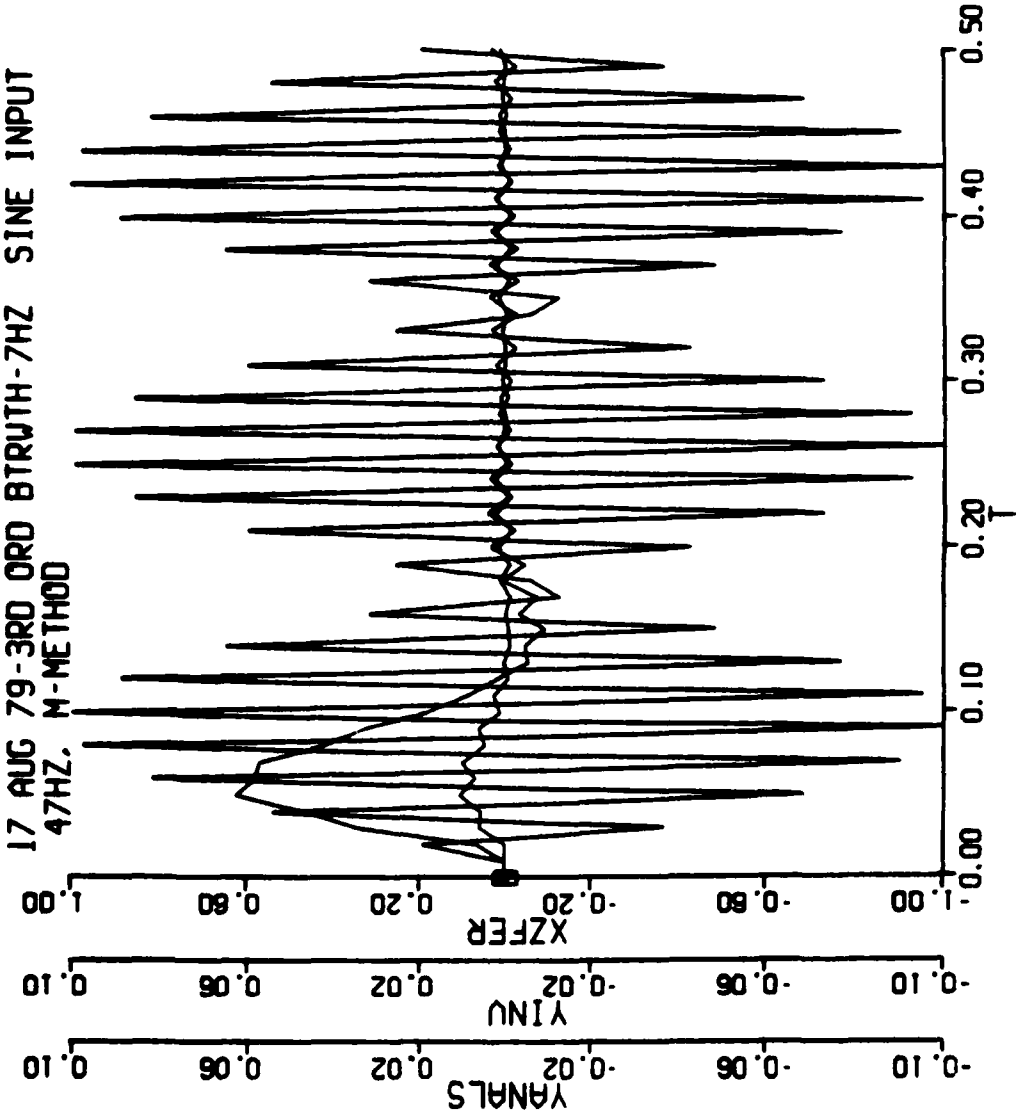


Figure 53. Plot of filter input and filter outputs for the impulse invariant method and the analytical expression -10 msec sampling interval.

analyatical responses are also plotted. *Figure 54* also contains plots of the invariant and analytical responses but a bias is added to YINV to aid in comparing the two signals. In *Figure 54* YINV appears to be lagging YANALS. However from *Table 1* the opposite is seen to be true. Thus the relative shifting of modulation peaks between signals is not an indicator of which signal is leading or lagging the other. In general the digital filters are doing a good job even though they are being updated at essentially the Shannon limit.

As would be expected the RK2 method of implementing the filter does not perform well at the 10 msec integration step size, as can be seen in *Figure 55* and *Table 1*. *Figure 56* illustrates the sampling coarseness with respect to the 47 Hz frequency.

As previously discussed the values for amplitude and phase in *Table 1*, as calculated from the transient response data at 47 Hz, were based on peak values.

```
STOP
BATCH, PRINT, PRINT, 3D, HDMDD
```

terminates the simulation and provides a line printer listing of the recorded data for all of the simulation runs.

The Butterworth digital filters' transient response matched their Bode plot amplitude and phase data very well. The impulse invariant response closely follows the continuous Butterworth response over most of the frequency range of interest. In the region where the exponentially decaying terms have sizeable values the transient responses of the digital filters do not overlay the analytical filter plots but are indicative of the analytical response. Amazingly enough the digital filters were relatively accurate right up to the Shannon limit.

The RK2 numerical integration implementation of the continuous filter showed an excellent phase match and an adequate amplitude match with the analytical filter until, of course, the input sine wave frequency became too large for the 10 msec integration step size to handle. The RK2 algorithm actually outperformed the digital filters for a delayed step input-in the important sense that it did not differ in phase with the analytical response.

A general purpose ACSL macro for implementing Z-transform filters (with zero I.C.'s) was coded and tested. Details of the algorithm are discussed in Appendix A.

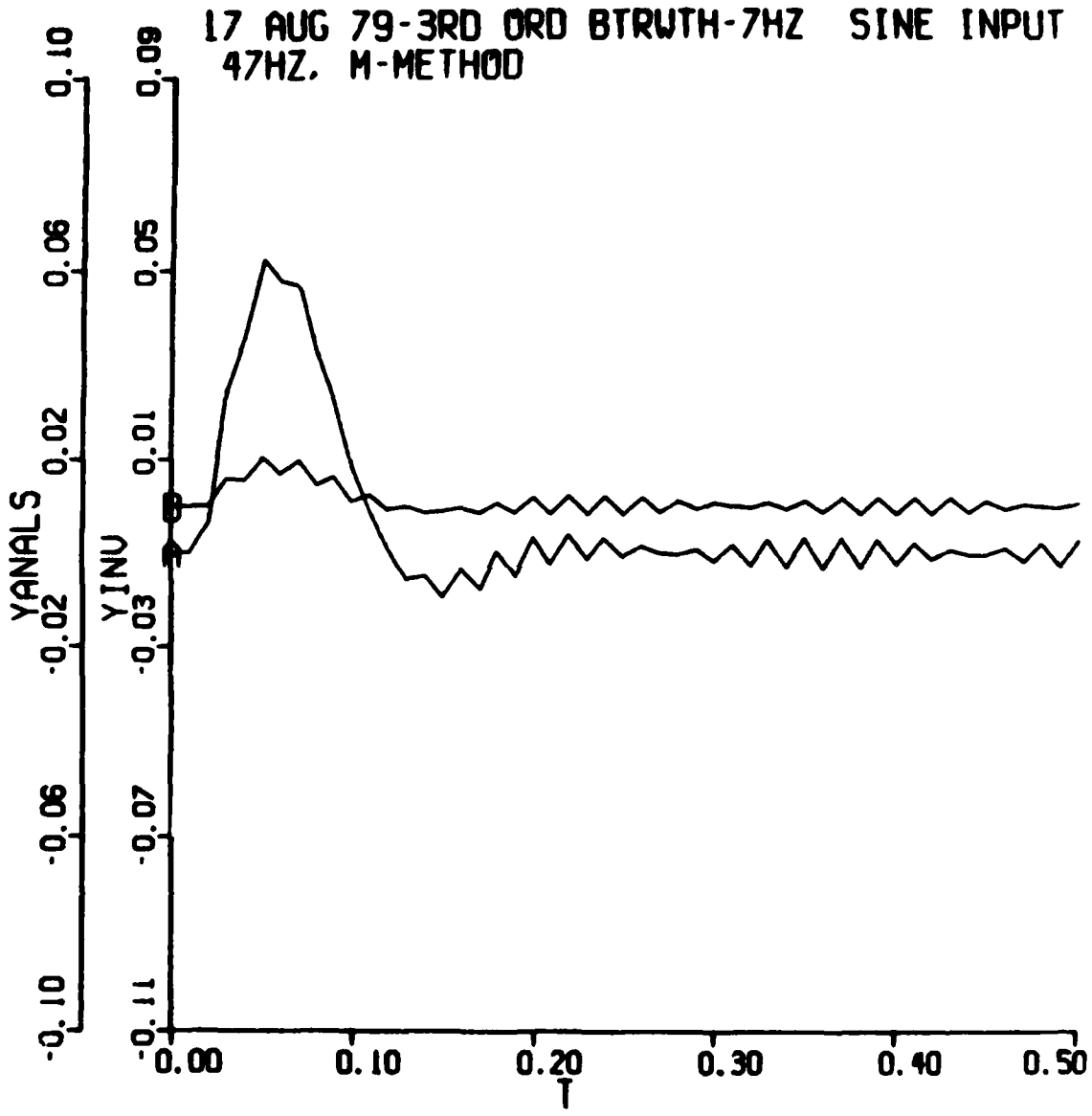


Figure 54. Comparison of the impulse invariant method and the analytical response – 10 msec sampling interval.

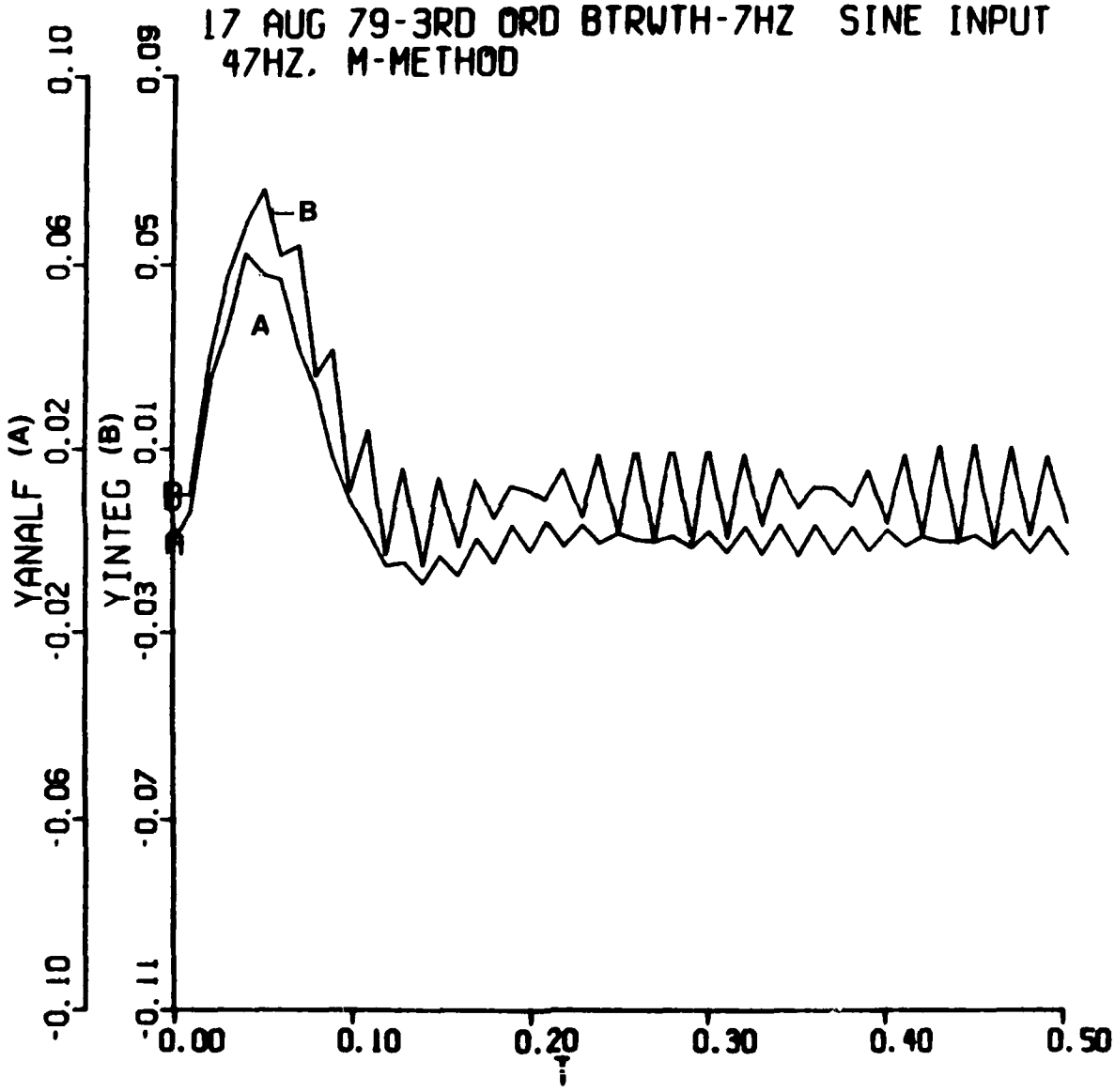


Figure 55. Comparison of the numerical integration method and the analytical response - 10 msec integration step size.

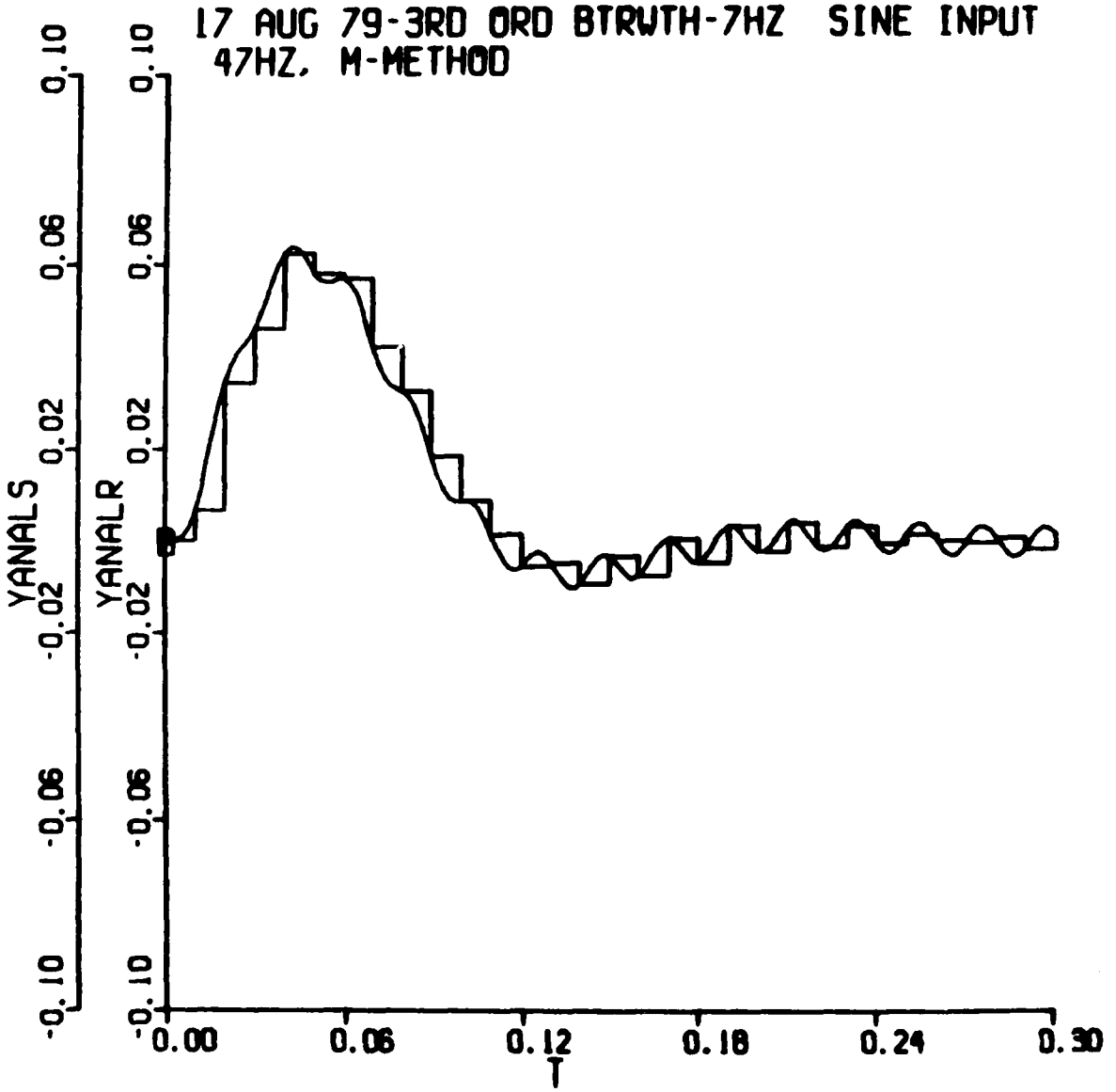


Figure 56. Illustration of how large the digital sampling interval is with respect to the frequency content of the filter output where the analytical filter output was used for plotting.

REFERENCES

1. E.E.L. Mitchell and Joseph S. Gauthier, *Advanced Continuous Simulation Language (ACSL) User Guide/ Reference Manual*, 1975, Mitchell and Gauthier, Assoc.

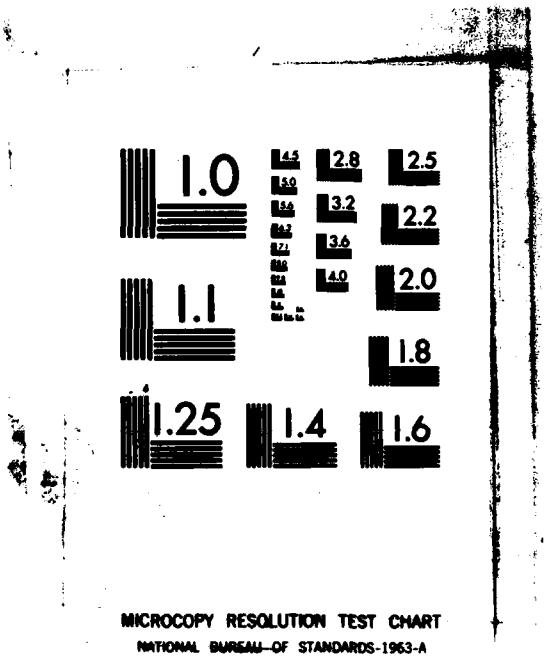
2. Charles M. Radar and Bernard Gold, "Digital Filter Design Techniques in the Frequency Domain," Proc. IEEE, Vol. 55, February 1967, pp. 149-171.

3. Katsuhiko Ogata, *Modern Control Engineering*, Englewood Cliffs, N.J.: Prentice-Hall, Inc., 1970.

APPENDIX A

MACRO CODE FOR ZXSFRM

AD-A140 302 ACSL (ADVANCED CONTINUOUS SIMULATION LANGUAGE) 2/2
SIMULATION OF A THIRD ORDE..(U) MITCHELL AND GAUTHIER
ASSOCIATES HUNTSVILLE AL E E MITCHELL ET AL. FEB 84
UNCLASSIFIED DRSMI/RD-CR-84-8 DAAK40-78-D-0010 F/G 9/1 NL


1.0
1.1
1.25
1.4
1.6
1.8
2.0
2.2
2.5
2.8
3.2
3.6
4.0
4.5
5.0
5.6
6.3
7.1
MICROCOPY RESOLUTION TEST CHART
NATIONAL BUREAU OF STANDARDS-1963-A

```
MACRO MACRO ZXSFRM(P,C)
MACRO REDEFINE Y,I,YIC,J,YP
MACRO RELABEL SN001,SN002,SN003,SN004,SN005
MACRO ASSIGN N
MACRO MULTIPLY 0
MACRO INCREMENT Q(4)
MACRO DIVIDE Q(3)
MACRO IF(N=0)999
MACRO MULTIPLY 0
MACRO INCREMENT Q(4)
ARRAY Y(N),YIC(N)
CONSTANT YIC=N*0.0
PROCEDURAL(P(1)=P(2),P(3),P(4))
IF(ZZFST(I).LT.0.5)GO TO SN005
DO SN001 I=1,N
SN001..CALL ZZICS(YIC(I),Y(I))
YP=P(2)
MACRO DECREMENT 1
DO SN002 I=1,N
MACRO INCREMENT 1
SN002..YP=YP-P(4)(I+1)*Y(N-I)
Y(N)=YP/P(4)(1)
YP=0.0
MACRO INCREMENT 1
DO SN003 I=1,Q(3)
SN003..YP=YP+P(3)(I)*Y(N-I)
P(1)=YP
MACRO DECREMENT 2
DO SN004 I=1,N
SN004..Y(I)=Y(I+1)
SN005..CONTINUE
END
MACRO EXIT
MACRO 999..PRINT NUMERATOR GREATER THAN DENOMINATOR
MACRO END
```

Figure A. Listing of the ACSL code for macro ZXSFRM.

Figure A is a listing of the ZXSFRM macro. Its intended use is the implementation of any order Z-transfer function. The macro algorithm is based on the M-method canonical form of the transfer function.

A standard call to ZXSFRM would appear in ACSL code as,

```
ZXSFRM (Y=X, R, S)
```

where y is the output of the filter; X is the input; R is the numerator coefficient array; and S is the denominator coefficient array. The numerator and denominator coefficient arrays are in ascending powers of (Z^{-1}).

Referring to *Figure A*, the statement

```
MACRO MACRO ZXSFRM(P,Q)
```

obviously does not correlate with the argument description discussed above. So the following explanation is in order. P is an array containing all of the argument list variables in the call to the ZXSFRM macro. Q is an array containing the first dimensional argument for each variable in the argument list. For instance, if R is an array of 2 and S is an array of 3, P and Q would be defined as follows:

```
P(1)=Y
P(2)=X
P(3)(1)=R(1)
P(3)(2)=R(2)
P(4)(1)=S(1)
P(4)(2)=S(2)
P(4)(3)=S(3)
Q(1)=1
Q(2)=1
Q(3)=2
Q(4)=3
```

The 'MACRO REDEFINE..' and 'MACRO RELABEL...' statements define local variable and statement label names. 'MACRO ASSIGN N' assigns the actual number of arguments from the macro call to the integer variable N. N can be made to have any value desired through

use of appropriate macro math operators. N aids in the generation of specific macro code in ACSL when an ACSL program is being compiled.

```
MACRO MULTIPLY 0
MACRO INCREMENT Q(4)
MACRO DIVIDE Q(3)
MACRO IF(N=0) 999
:
MACRO 999..PRINT NUMERATOR GREATER THAN DENOMINATOR
```

is the code used to prevent the attempted implementation of Z polynomials where the numerator order exceeds the denominator order.

```
MACRO MULTIPLY 0
```

multiplies N by zero.

```
MACRO INCREMENT Q(4)
```

adds the value of Q(4) to N. Thus N is now equal to the dimension of the denominator coefficient array of the Z-transfer function.

```
ARRAY Y(N), YIC(N)
```

defines the intermediate state variable array Y (as for the M-method formulation) and the intermediate state variable initial conditions, YIC. A PROCEDURAL block is used as there is branching in the code,

```
PROCEDURAL (P(1)=P(2),P(3),P(4)).
IF (ZZFST(I).LT.0.5)GO TO SN005
:
:
:
SN005..CONTINUE
```

causes the filter calculation to be skipped unless this is the first minor step of a multi-step integration algorithm. ZZICS is called to initialize the intermediate state variable array and to also reset the array to its last values when the REINIT run time command is used,

```
DO SN001 I=1,N
SN001..CALL ZZICS(YIC(I),Y(I))
```

The intermediate state variable array (the $(Z^{-1})^i$ M states) is solved as shown in Equation (8) of Section 2,

```
YP=P(Z)
MACRO DECREMENT I
DO SNOO2 I=1,N
MACRO INCREMENT I
SN002..YP=YP-P(4)(I+1)*Y(N-I)
Y(N)=YP/P(4)(1)
```

where

```
Y(N)=M
Y(N-i)=(Z^-1)^iM
```

The output of the transfer function is then solved by multiplying the intermediate states by the numerator coefficients as shown in Equation (9) of Section 2.

```
YP=0.0
MACRO INCREMENT I
DO SN003 I=1, Q(3)
SN003..YP=YP+P(3)(I)*Y(N-I)
P(1)=YP
```

The intermediate state array, Y, is updated in preparation for the next calculation interval,

```
MACRO DECREMENT 2
DO SN004 I=1, N
SN004.. Y(I)=Y(I+1)
SN005..CONTINUE
END
MACRO EXIT
MACRO 999...PRINT NUMERATOR GREATER THAN DENOMINATOR
```

APPENDIX B

CALCULATIONS FOR OBTAINING THE TRANSIENT RESPONSE AMPLITUDE AND PHASE OF THE FOUR FILTER IMPLEMENTATIONS

From Equation (4) of Section 2, the steady state value of the analytical response is

$$A \cos \gamma t + \frac{B}{\gamma} \sin \gamma t \qquad \text{(B1)}$$

or

$$\Gamma (\sin \phi \cos \gamma t + \cos \phi \sin \gamma t)$$

$$= \Gamma \sin (\gamma t + \phi)$$

where

Γ=the steady state amplitude of the filter output

ϕ=phase shift of the filter output with respect to the input

$\phi<0$ is phase lag

$\phi>0$ is phase lead

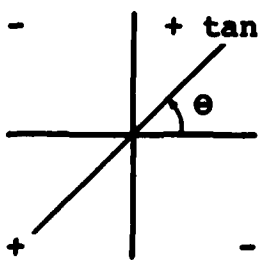
So

$$\phi = \tan^{-1}\left(\frac{A\gamma}{B}\right)$$

where A and B are defined in Equation (4) of Section 2 and

$$\Gamma = A/\sin \phi \text{ or } B/(\gamma \cos \phi) \qquad \text{(B2)}$$

Due to the arbitrary quadrant location of the arctangent function as shown below,



$$\tan (\theta \pm n\pi) = \text{same value}, \quad n = 0, 1, 2, \ldots$$

the phase lag from the Bode plots and a corresponding rough measure of phase lag in the transient response plots are used as justification for the choice of ϕ made. There are also certain restrictions on ϕ. Substituting in (1) for A and B the actual terms,

$$\phi = \tan^{-1}\left(\frac{\gamma}{\omega_o}\,\frac{(\frac{\gamma}{\omega_o})^2 - 2}{1 - 2(\frac{\gamma}{\omega_o})^2}\right) \qquad \text{(B-3)}$$

which except for the 5 Hz=γ case (γ=10, 20, 40,...Hz) is always going to be less than zero. Secondly the filter will have phase lag and therefore $\phi<0$. Combining these two facts with the knowledge that $|\phi|<360$ degrees leaves only two choices for ϕ: $\tan^{-1}(...)$ or $\tan^{-1}(...)-\pi$.

So the true analytical values for amplitude and phase can be determined. The Bode plot for three of the four filters may be obtained from a modified version of the program in Section 3. The amplitude and phase values for the two digital filters and the integration method can be obtained from the transient response program data.

For a linear transfer function the steady response to a sine wave is another sine wave shifted in phase and changed in amplitude,

$$\sin \omega t \Rightarrow A \sin(\omega t + \phi) \qquad \text{(B-3)}$$

where, as noted above, $\phi<0$ is phase lag and $\phi>0$ is phase lead. For two arbitrary points X_1, and X_2, in the transient response output we have

$$A \sin(\omega t_1 + \phi) = X_1 \qquad \text{(B-4)}$$

$$A \sin(\omega t_2 + \phi) = X_2 \qquad \text{(B-5)}$$

where from (4)

$$\phi = \sin^{-1}(\frac{X_1}{A}) - \omega t_1 \qquad \text{(B-6)}$$

Substituting (6) into (5) gives

$$A \sin\left(\omega t_2 + \sin^{-1}(\frac{X_1}{A}) - \omega t_1\right) = X_2$$

Letting $\gamma=\omega t_2-\omega t_1$,

$$A\left[\sin\gamma\cos\left(\sin^{-1}\frac{X_1}{A}\right)+\cos\gamma\left(\frac{X_1}{A}\right)\right]=X_2$$

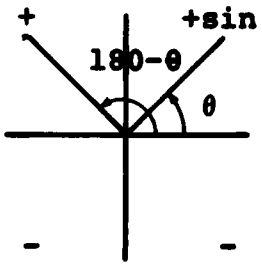
or

$$A\left[\sin\gamma\,\frac{\sqrt{A^2-X_1^2}}{A}+\cos\gamma\left(\frac{X_1}{A}\right)\right]=X_2.$$

Squaring both sides and solving for A,

$$A=\pm\left[\left(\frac{X_2-X_1\cos\gamma}{\sin\gamma}\right)^2+X_1^2\right]^{\frac{1}{2}} \qquad \text{(B-7)}$$

where the + sign is chosen. As mentioned above there is a multitude of angles which satisfy (6). Due to the arbitrary quadrant location of the arc sine function as shown below,



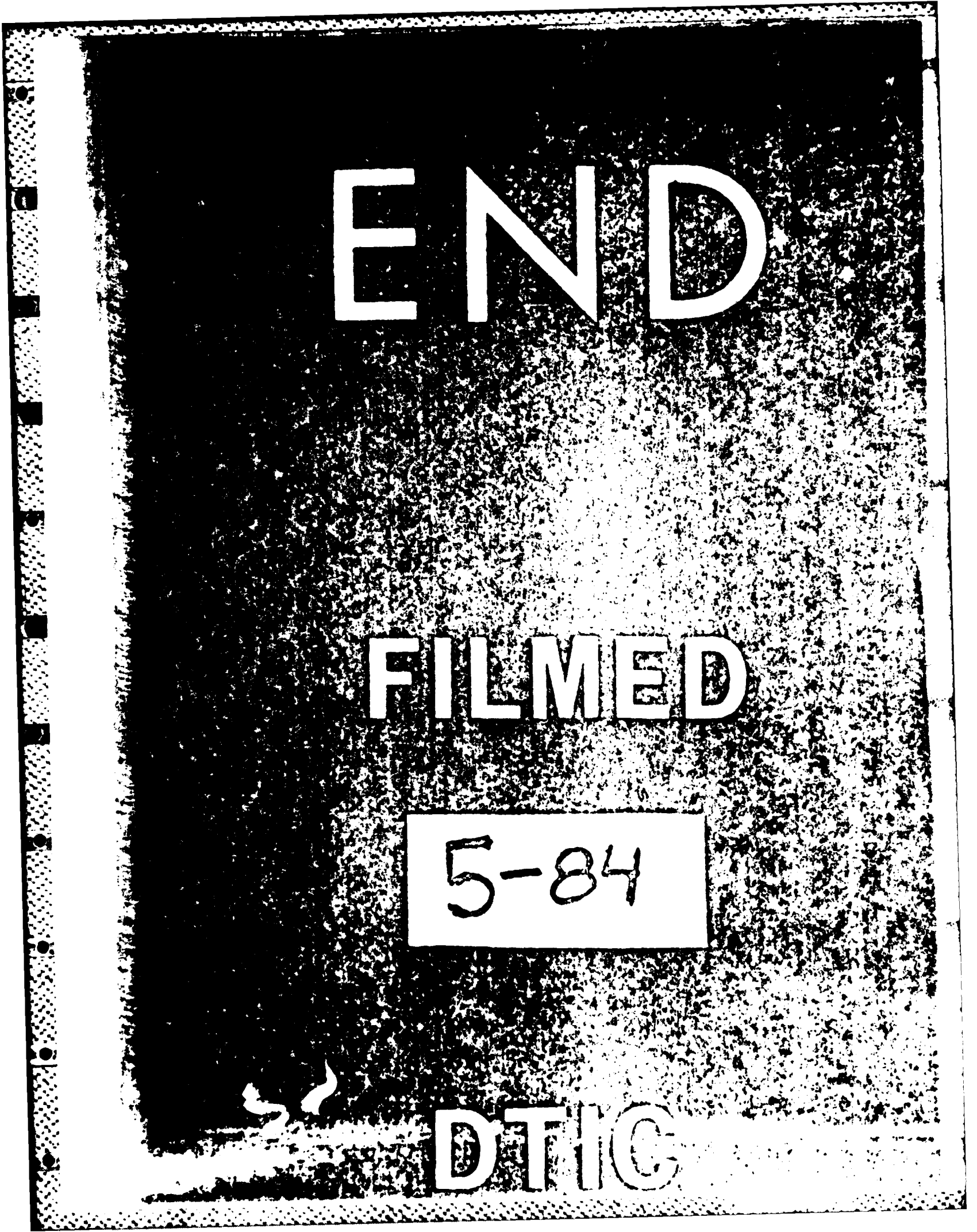
$$\sin^{-1}(\text{value})=\begin{cases}\theta+2n\pi\\ \text{SIGN}\ (180,\ \theta)+2n\pi-\theta\\ n=0,\ \pm 1,\ \pm 2,\ \ldots\end{cases}$$

the phase lag from the Bode plots and a corresponding rough measure of phase lag in the transient response are used to aid in determining ϕ. Again $-360<\phi\leq 0$ so if ϕ is calculated positive -2π is added to it and if $\phi<0$ it can only have two values: ϕ or $-180-|\phi$.

Equation (B-7) is undefined when γ, or $\omega t_2-\omega t_1$, is a multiple of π. Other quirks in (7) were found and are mentioned for the appropriate cases in Section 4. But evidently the problem stems from the fact that $\omega t_2-\omega t_1$, can also be interpreted as another $\omega^* t_2^*-\omega^* t_1^*$ as long as the value of the subtraction remains the same. However the amplitude expression holds quite well at the lower frequencies and for the purposes of this report were considered adequate. t_1, t_2, X_1 and X_2 are judiciously picked near the end of the plotted filter responses to obtain approximately steady state values.

DISTRIBUTION

	No. of Copies
US Army Materiel Systems Analysis Activity ATTN: DRXSY-MP Aberdeen Proving Ground, MD 21005	1
IIT Research Institute ATTN: GACIAC 10 West 35th Street Chicago, IL 60616	1
DRSMI-R, Dr. McCorkle	1
Dr. Rhoades	1
-RD	5
-RPR	15
-RPT (Record Set)	1
-LP	1


END
FILMED
5-84
DTIC