AD-A 159 311

LEP-L

AD

AD-E401 390

TECHNICAL REPORT ARLCD-TR-85025

AN ANALYTICAL MODULARIZED TREATMENT OF AUTOPILOTS FOR GUIDED PROJECTILE SIMULATIONS

EUGENE M. FRIEDMAN
MICHAEL J. AMORUSO

AUGUST 1985

US ARMY ARMAMENT MUNITIONS & CHEMICAL COMMAND
ARMAMENT R&D CENTER

US ARMY ARMAMENT RESEARCH AND DEVELOPMENT CENTER

LARGE CALIBER WEAPON SYSTEMS LABORATORY

DOVER, NEW JERSEY

UNCLASSIFIED
SECURITY CLASSIFICATION OF THIS PAGE (*When Data Entered*)

REPORT DOCUMENTATION PAGE		READ INSTRUCTIONS BEFORE COMPLETING FORM
1. REPORT NUMBER Technical Report ARLCD-TR-85025	2. GOVT ACCESSION NO.	3. RECIPIENT'S CATALOG NUMBER
4. TITLE (*and Subtitle*) AN ANALYTICAL MODULARIZED TREATMENT OF AUTOPILOTS FOR GUIDED PROJECTILE SIMULATIONS		5. TYPE OF REPORT & PERIOD COVERED 6. PERFORMING ORG. REPORT NUMBER
7. AUTHOR(*s*) Eugene M. Friedman Michael J. Amoruso		8. CONTRACT OR GRANT NUMBER(*s*)
9. PERFORMING ORGANIZATION NAME AND ADDRESS ARDC, LCWSL Applied Science Div (SMCAR-LCA) Dover, NJ 07801-5001		10. PROGRAM ELEMENT, PROJECT, TASK AREA & WORK UNIT NUMBERS
11. CONTROLLING OFFICE NAME AND ADDRESS ARDC, TSD STINFO Div (SMCAR-TSS) Dover, NJ 07801-5001		12. REPORT DATE August 1985 13. NUMBER OF PAGES 45
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 is 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*)

Simulation
Transfer functions
Autopilot
Guidance and control
Guided munitions
Guided projectiles

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

An innovative technique was developed by which piecewise analytic solutions to guidance and control transfer functions were obtained for use in larger but lower frequency computer simulations of guided munitions. Numerical integration, which is typically used to treat such transfer functions, significantly reduces the integration time step, and increases computer execution time for the overall simulation. The modularized analytical approach accommodates integration time
(cont)

20. ABSTRACT (cont)

steps associated with the lower frequencies of airframe dynamics which results in faster simulation execution and considerable savings in cost and execution time.

CONTENTS

INTRODUCTION

A substantial need has been demonstrated within Armament Research and Development Center (ARDC) for a convenient-to-use and computationally efficient flight simulation for smart munitions. The simulation must be easily adaptable to varying projectile designs and especially to changes in autopilots which undergo rapid evolution. During the design cycle of the projectile, a large number of individual flight simulations must be performed. As an example, during the development of the Copperhead projectile, the ARDC scientific computers were overwhelmed by the workload and a substantial amount of the available time on the BRL CDC 7600 supercomputer was dedicated to the computations. In spite of the continuous development and improvement in computer systems, flight simulations will continue to provide a substantial workload which must be accomplished efficiently.

To meet this need, ARDC is formulating methods and writing modular software for rapidly developing autopilot simulations for guided projectiles and munitions. An innovative technique was developed by which exact analytical solutions to the required transfer functions are applied in a piecewise manner within a larger but lower frequency problem which must be solved numerically.

Since the time constants associated with the autopilot components are often small compared with their driving terms, the integration time step is driven to very small values to obtain stable numerical integrations which results in very long computer run times. The use of piecewise analytical solutions to the transfer functions guarantees valid integration of the autopilot transfer functions regardless of the integration time step, provided that the driving terms vary inappreciably over the time step. This is a less stringent requirement than is needed for stable numerical integration since the driving term rates are commensurate with the airframe time constants, which are typically large compared to that of the autopilot.

Five transfer functions were solved in closed form to provide a fast executing computer code. These particular transfer functions were selected since an autopilot can generally be represented by a concatenation of these functions to rate sensors, switches, limiters, and dead zones.

DISCUSSION

Apart from switches, limiters, and dead zones, autopilot transfer functions are Laplace transform representations of differential equations. Incorporated into a digital simulation, transfer functions are converted into differential equations and are usually solved numerically. Sometimes this causes considerably longer run times than simulation of unguided projectiles.

The time step required to perform the integration is driven by two constraints: (1) The driving term or input must not vary appreciably in the time step. (2) The time step must be sufficiently small to insure a stable integration. If the time constants associated with the autopilot are small compared to those of the airframe (which is typical), the integration time step will be driven to very small values and run times will be very long. Since the driving term is a product of the airframe motion, an inherently slower process, stable integration is a stronger driver to fine integration than the driving term's remaining essentially constant during the integration time step. By analytically solving these transfer functions stepwise with constant driving terms, the second constraint is eliminated. The size of the integration time step is limited only by the first constraint. Therefore, the use of closed form, analytic, stepwise solutions subject only to the first constraint can lead to larger integration time steps and shorter run times.[1]

Only five transfer functions are needed to handle the typical autopilot: the first-order lag, the first-order lag with differentiation, the first-order lead/lag, the first-order lag with integrator, and the second-order lag/oscillator. (Guidance and control systems often contain rate sensors. The treatment of rate sensors will be published in a separate report.) These transfer functions are described and solved analytically below; the computer code implementing the solutions is shown in appendix A.[2]

Throughout this development, the driving term is assumed not to vary appreciably during a time step. Care must be taken to adjust the time step downward as necessary to satisfy this requirement. To take advantage of this technique for reducing run time, the time step should be increased whenever the driving terms are varying slowly.

ANALYSIS

First-Order Lag

A first-order lag is represented by the Laplace operator s as

$$1/(Ts + 1)$$

In differential equation form

$$T\, dy/dt + y = D \tag{1}$$

---

1 Michael J. Amoruso and Dennis D. Ladd, "TELUM, A Comprehensive Digital Flight Simulation of the Copperhead Projectile," Special Publication ARLCD-SP-82003, U.S. Army Armament Research and Development Command, Dover, NJ, June 1982.

2 Analytical methods used in this study were derived from John J. D'Azzo and Constantine H. Houpis, Feedback Control Systems Analysis and Synthesis, McGraw-Hill, New York, 1960; and Murray R. Spiegel, Applied Differential Equations, Prentice-Hall, Englewood Cliffs, NJ, 1958.

where

y = output or dependent variable

t = time or independent variable

T = time constant

D = driving term

The general solution to equation 1 is the sum of a particular solution and a general solution to the homogeneous form. The homogeneous equation was obtained by setting D to zero and can be written as

$$- T\,dy/y = dt$$

which can be verified to yield the solution

$$y = e^{-(t - A)/T} \tag{2}$$

where A is the integration constant determined from the initial conditions $y = y_o$ at $t = t_o$. Using the initial conditions yields the general solution to the homogeneous equation 2.

$$y = y_o e^{-(t - t_o)/T} \tag{3}$$

For a particular solution to equation 1, the expression is verified by substitution

$$y = \left[1 - e^{-(t - t_o)/T}\right] D \tag{4}$$

Adding equations 3 and 4 gives the complete general solution

$$y(t) = y_o e^{-(t - t_o)/T} + \left[1 - e^{-(t - t_o)/T}\right] D \tag{5}$$

Note that equation 5 has the correct limit of D as T goes to zero.

First-Order Lag with Differentiation

This transfer function has the Laplace operator representation

$$s/(Ts + 1)$$

or the time domain form

$$T\,dy/dt + y = dD/dt \tag{6}$$

This representation is not desirable since the driving term appears as a time derivative that would generally have to be evaluated numerically. However, equation 6 can be recast as a pair of coupled differential equations that do not contain any derivatives of the driving term D. An auxiliary variable z is introduced and the equation is replaced by the expressions

$$dz/dt = y \tag{7}$$

and

$$y = (D - z)/T \tag{8}$$

Differentiating equation 8 and using equation 7 to eliminate dz/dt verifies that these two equations are equivalent to equation 6. Note that the derivative of D does not appear. By combining equations 7 and 8, the expression for z(t) is obtained,

$$dz/dt + z/T = D/T \tag{9}$$

and, after integration, the results for z(t) substituted into equation 8. Equation 9 can be integrated from equation 5 by noting the identity of equations 1 and 9. Therefore,

$$y(t) = e^{-(t - t_o)/T} (D - z_o)/T \tag{10}$$

where $z_o = z$ at $t = t_o$. The auxiliary variable z_o can be eliminated by using equation 8.

$$y(t) = \left\{ \frac{\left[e^{-(t - t_o)/T}\right]}{T} \right\} (D - D_o) + y_o\, e^{-(t - t_o)/T} \qquad (11)$$

Note that the exponential over T goes to $1/(t - t_o)$ by L'Hospital's rule so that equation 11 has the correct limit dD/dt as T goes to zero.

First-Order Lead/Lag

This transfer function is represented by

$$(T_2 s + 1)/(T_1 s + 1) \qquad T_1 \neq 0$$

or

$$T_1\, dy/dt + y = T_2\, dD/dt + D \qquad (12)$$

The derivative of the driving term is usually not available in analytic form. It is eliminated by introducing an auxiliary variable, z, and a pair of coupled equations

$$dz/dt = (D - y)/T_1 \qquad (13)$$

and

$$y = (T_2/T_1)\, D + z \qquad (14)$$

The equivalence with equation 14 can be verified by differentiating it and then eliminating dz/dt with equation 13. To obtain an expression for y(t), integrate equation 14 as follows:

$$T_1\, dz/dt + z = (1 - T_2/T_1)\, D \qquad (15)$$

This is obtained by substituting equation 14 into equation 13, and then substituting the result for z(t) into equation 14 to obtain y(t). The explicit solution will not be written. Instead, an algorithm will be provided based on the observation that equation 15 is equivalent to equation 1 if the following substitutions are made:

$$T_1 \rightarrow T \qquad z \rightarrow y \qquad \left(1 - T_2/T_1\right)D \rightarrow D \tag{16}$$

Algorithm:

1. Substitute equation 16 into equation 5 to obtain z(t).

2. Save z(t) for use as z_o in the next integration step.

3. Substitute z(t) into equation 14 to obtain y(t).

Note that the lag time constant cannot be zero in equation 16, but a vanishing lead time constant yields the correct limiting case of a simple first-order lag.

First-Order Lag with Integrator

The representation of this transfer function is

$$1/s\ (Ts + 1)$$

or

$$T\ d^2y/dt^2 + dy/dt = D \tag{17}$$

Substitution verifies that the following expression is a particular solution to the nonhomogeneous differential equation 17 since, by assumption, D is constant during the integration time step:

$$y = Dt \tag{18}$$

The homogeneous form of equation 17 can be written as the coupled equations

$$T\ dz/dt + z = 0 \tag{19}$$

and

$$z = dy/dt \tag{20}$$

Equation 19 is similar to equation 1 with the substitution of z for y and zero for D; therefore, the solution can be obtained from equations 2 and 20

$$dy/dt = z(t) = e^{-(t - A)/T}$$

or

$$y(t) = -Te^{-(t - A)/T} + B \tag{21}$$

The complete solution to equation 17 is the sum of the general homogeneous solution (eq 21) and the particular nonhomogeneous solution (eq 18)

$$y(t) = -Te^{-(t - A)/T} + Dt + B \tag{22}$$

$$dy/dt = e^{-(t - A)/T} + D \tag{23}$$

The integration constants A and B can be determined by invoking the initial conditions $y = y_o$ and $dy/dt = y'$ at $t = t_o$

$$y_o = -Te^{-\left(t_o - A\right)/T} + D\, t_o + B \tag{24}$$

$$y' = e^{-\left(t_o - A\right)/T} + D \tag{25}$$

The integration constant A can be eliminated from equation 24 by using equation 25 to eliminate the exponential term and solving for B

$$B = y_o + T(y' - D) - Dt_o \qquad (26)$$

Taking the natural logarithm of equation 25 yields

$$A = T \ln (y' - D) + t_o \qquad (27)$$

The explicit solution is therefore

$$y(t) = y_o + T(y' - D)\left[1 - e^{-(t - t_o)/T}\right] + D(t - t_o) \qquad (28)$$

and

$$dy/dt = (y' - D)\, e^{-(t - t_o)/T} + D \qquad (29)$$

Note that equations 28 and 29 go to the correct limits if T goes to zero.

Second-Order Lag/Oscillator

This transfer function can be represented in the form

$$1/(Is^2 + Ds + K) \qquad K \neq 0,\ I \neq 0$$

or

$$I\, d^2y/dt^2 + dy/dt + Ky = T \qquad (30)$$

The homogeneous solution to equation 30 can be verified to be

$$y(t) = e^{Lt} \qquad (31)$$

by substituting into the homogeneous form of equation 30 to obtain the characteristic equation

$$IL^2 + DL + K = 0 \qquad (32)$$

which has the following roots:

$$M = \left[-D + \left(D^2 - 4\ IK\right)^{1/2} \right] /2I \tag{33}$$

$$N = \left[-D - \left(D^2 - 4\ IK\right)^{1/2} \right] /2I \tag{34}$$

The following three cases are treated separately according to whether the radicand is positive, negative, or zero.

Case 1. Positive radicand, damped solution

The homogeneous and nonhomogeneous particular solutions can be verified to be

$$y_H(t) = Ae^{Mt} + Be^{Nt} \tag{35}$$

$$y_P(t) = T/K \tag{36}$$

The complete solution is

$$y(t) = Ae^{Mt} + Be^{Nt} + T/K \tag{37}$$

$$dy/dt = MAe^{Mt} + NBe^{Nt} \tag{38}$$

The initial conditions $y_o = y$ at t_o and $y' = dy/dt$ at t_o determine the constants A and B.

$$y_o = Ae^{Mt_o} + Be^{Nt_o} + T/K \tag{39}$$

$$y' = MAe^{Mt_o} + NBe^{Nt_o} \tag{40}$$

Solving simultaneously gives

$$B = \left(y_o - y'/M - T/K\right)e^{-N\,t_o} / \left(1 - N/M\right) \tag{41}$$

$$A = \left(y_o - y'/N - T/K\right)e^{-M\,t_o} / \left(1 - M/N\right) \tag{42}$$

Case 2. Zero radicand, critically damped

The characteristic equation has only one distinct root

$$L = -D/2I \tag{43}$$

The homogeneous general solution and nonhomogeneous particular solution can be verified to be

$$y_H\,(t) = Ae^{Lt} + Bte^{Lt} \tag{44}$$

$$y_H(t) = T/K \tag{45}$$

The complete solution is

$$y(t) = (A + Bt)\,e^{Lt} + T/K \tag{46}$$

$$dy/dt = (LA + B + LBt)e^{Lt} \tag{47}$$

Using the initial conditions $y_o = y$ at t_o and $y' = dy/dt$ at t_o, the integration constant can be determined as follows:

$$y_o = (A + Bt_o)e^{Lt_o} + T/K \tag{48}$$

$$y' = (LA + B + LBt_o)e^{Lt_o} \tag{49}$$

$$B = e^{-Lt_o} [y' + L (T/K - y_o)] \tag{50}$$

$$A = [(y_o - t/K) (1 + L t_o) - t_o y']e^{-Lt_o} \tag{51}$$

Case 3. Negative radicand, oscillation

The roots of the characteristic equation become complex numbers

$$\omega = \left[(K/I) - (D/2I)^2 \right]^{1/2} \tag{52}$$

The homogeneous and nonhomogeneous particular solutions are

$$y_H(t) = Ae^{Lt} \sin (\omega t + \phi) + T/K \tag{53}$$

$$y_H(t) = T/K \tag{54}$$

where A and ϕ are integration constants. The complete solution is

$$y = Ae^{Lt}\sin(\omega t + \phi) + T/K \tag{55}$$

$$dy/dt = LAe^{Lt}\sin(\omega t + \phi) + \omega Ae^{Lt}\cos(\omega t + \phi) \tag{56}$$

Using the initial conditions y_o and y' and solving for A in the expression for y_o gives

$$A = (y_o - T/K)e^{-L\,t_o}/\sin(\omega t_o + \phi) \tag{57}$$

where

$$\sin(\omega t_o + \phi) \neq 0 \tag{58}$$

If equation 58 fails, A can be obtained from the expression for y'

$$A = y'e^{-L\,t_o}/[L\sin(\omega\,t_o + \phi) + \omega\cos(\omega t_o + \phi)] \tag{59}$$

Since ω in equation 52 is always positive definite and the sine and cosine are orthogonal, either equation 57 or 59 is always defined. Note that the expressions still contain the phase angle ϕ. Solving for ϕ yields the same result whether equation 57 is substituted into the initial condition on dy/dt or equation 59 into the initial condition on y.

$$\phi = \tan^{-1}\left[\frac{\omega\,(y_o - T/K)}{[y' - L(y_o - T/K)]}\right] - \omega\,t_o \tag{60}$$

The phase ϕ can be eliminated in equation 57 or 59 by substituting equation 60. Equation 60 shows that equation 58 is equivalent to

$$y_o - T/K = 0 \tag{61}$$

The imposition in equation 30 of the requirement that parameters K and I not vanish was necessary for the validity of the solutions. For example, equations 36, 45, and 54 become singular when K vanishes. Similarly, the roots of the characteristic equation are singular if I is zero (eqs 33, 45, and 52). This is not a limitation since equation 30 reduces to a first-order lag with integrator when K vanishes and to a simple first-order lag when I vanishes.

However, the parameter D in equation 30 may be zero. If D vanishes, the characteristic equation 32 degenerates to

$$IL^2 + K = 0$$

$$M = \pm \left(-K/L\right)^{1/2} = \pm\, i\, \omega$$

Therefore, the roots are imaginary corresponding to a pure oscillation with no damping. Only case 3 should be considered. Nowhere does D appear in the denominator of any expression; therefore, the solution remains valid if D is zero.

RESULTS

The exact saving resulting from this piecewise analytical technique depends on the speed of execution of the competing numerical method. A sample run is included in appendix B so that comparisons can be made with other preferred numerical integration techniques. For comparison, the last case (KTFID=13) was run analytically with time steps of 0.01 and 0.005 second and run numerically using Advanced Continuous Simulation Language (ACSL).[3] Whereas the analytical method gave the same results for time steps 0.001 and 0.005 second, ACSL could not operate with a time step of 0.001 second. ACSL was able to obtain nearly converged results with a time step of 0.00001 second but failed after the third second. These results are compared as follows:

Time (sec)	Analytical DT=0.01 sec	Analytical DT=0.005 sec	Numerical (ACSL) DT=0.00001 sec
0	10.00000	10.00000	10.00000
1	1.54640	1.54640	1.54381
2	-9.52188	-9.52188	-9.52327
3	-4.49206	-4.49206	-4.48494

[3] Advanced Continuous Simulation Language, User Guide/Reference Manual, Mitchell and Gauthier Associates, Inc., Concord, MA, 1981.

In this particular case, the analytical technique resulted in at least two orders of magnitude improvement in integration step size.

CONCLUSIONS

The modular, piecewise analytical technique presented in this report can result is considerable savings in analysis and associated computer time.

APPENDIX A

PROGRAM LISTING AND SAMPLE EXECUTION

PROGRAM LISTING

```
      PROGRAM TFMAIN(INPUT,OUTPUT,TAPE5=INPUT,TAPE6=OUTPUT)
C      MAIN PROGRAM TO DRIVE TRANSFER FUNCTION SUBROUTINE.
C           FOR TEST PURPOSES ONLY.
      COMMON/TRNSF/TAU1(50),TAU2(50),TAU3(50),TFTIME(50),TFAUX(50),
     + TFAUX2(50),KTFTYP(50),GAIN(50)
      OPEN(8)
      WRITE(8,11)
C     READ INPUT DATA AND INITIALIZE.
      CALL TRNF IO
      WRITE(8,11)
C 200 DEFINE INTEGRATION TIME STEP DT.
      DT = .005
      DT2= DT/2.
      WRITE(8,13) DT
      DO 2 J=1,13
      WRITE(8,12)
C 100 DEFINE INITIAL TIME T AND FINAL TIME TMAX.
      T=0.0
      TMAX = 5.0
      KMAX = IFIX(TMAX/DT)+1
      DO 1 K=0,KMAX
      T = DT*FLOAT(K)
C 300 DEFINE DRIVING TERM "DRIVE"
      DRIVE = COS(2.*T) - 1.0
      CALL TRNF (DRIVE,XOUT,T,J)
      IF(ABS(IFIX(T)-T).LT.DT2) WRITE(8,10) J,T,DRIVE,XOUT
    1 CONTINUE
    2 CONTINUE
      STOP
   10 FORMAT(1X,I5,3F20.5)
   11 FORMAT('1')
   12 FORMAT(//,' LTFID',16X,'TIME',14X,'DRIVE',19X,'X')
   13 FORMAT(' THE INTEGRATION TIME STEP DT = ',F9.6,' .',//)
      END


      SUBROUTINE TRNF IO
C     LTFTYP  1 -> GAIN/(TAU1*S+1)
C             2 -> GAIN*S/(TAU1*S+1)
C             3 -> GAIN/(S*(TAU1*S+1))
C             4 -> GAIN*(TAU2*S+1)/(TAU1*S+1)
C             5 -> GAIN/(TAU1*S**2 + TAU2*S +TAU3)
C     LTFID   ID NO. OF CHANNEL (VARIABLE).
C     TU1     FIRST TIME CONSTANT
C     TU2     SECOND TIME CONSTANT (LTFTYP=4 OR 5)
C     TU3     THIRD TIME CONSTANT (LTFTYP=5)
```

```
C      X0       INITIAL CONDITION ON VARIABLE
C      DXO      INITIAL CONDITION ON DERIVATIVE (LTFTYP=5)
C      AA       ALPHAMERIC LABEL FOR PRINTOUT
       COMMON/TRNSF/TAU1(50),TAU2(50),TAU3(50),TFTIME(50),TFAUX(50),
     + TFAUX2(50),KTFTYP(50),GAIN(50)
       CHARACTER AA*30,LABEL(5)*30
       DATA LABEL/'GAIN/(TAU1*S+1)                ',
     +           'GAIN*S/(TAU1*S+1)              ',
     +           'GAIN/(S*(TAU1*S+1))            ',
     +           'GAIN*(TAU2*S+1)/(TAU1*S+1)     ',
     +           'GAIN/(TAU1*S**2+TAU2*S+TAU3)   '/
       WRITE(8,1050)
       DO 1 I=1,50
       TAU1(I)     = 0.
       TAU2(I)     = 0.
       TAU3(I)     = 0.
       TFTIME(I)   = 0.
       TFAUX(I)    = 0.
       TFAUX2(I)   = 0.
       KTFTYP(I)   = 0
       GAIN(I)     = 0.
     1 CONTINUE
       IFLAG = 0
       DO 2 I=1,50
       READ (5,1051,END=4) LTFID,LTFTYP,TU1,TU2,TU3,GA,X0,DX0,AA
       IF (LTFTYP.LT.1 .OR. LTFTYP.GT.5) THEN
        WRITE(8,1052)LTFID,LTFTYP,TU1,TU2,TU3,GA,X0,DX0,AA
        IFLAG = 1
        WRITE(8,1004)
        GO TO 2
       ELSE
        WRITE(8,1052)LTFID,LTFTYP,TU1,TU2,TU3,GA,X0,DX0,AA,LABEL(LTFTYP)
       END IF
       IF (LTFID.LT.1 .OR. LTFID.GT.50) THEN
          IFLAG = 1
          WRITE(8,1001)
          GO TO 2
         END IF
       IF(TAU1(LTFID).NE.0..OR.TAU2(LTFID).NE.0..OR.TAU3(LTFID).NE.0.
     + .OR.KTFTYP(LTFID).NE.0) THEN
          IFLAG =1
          WRITE(8,1002)
          GO TO 2
         END IF
       IF(TU1.GT.0..AND.TU2.EQ.0..AND.TU3.GT.0..AND.LTFTYP.EQ.5)
     + GO TO 3
       IF ((TU1.LE.0..OR.TU2.LE.0..OR.TU3.LE.0.).AND.LTFTYP.EQ.5)THEN
           WRITE(8,1003)
           IF (TU1.EQ.0. .AND. TU3.GT.0. .AND.TU2.GT.0.)  THEN
             WRITE(8,1005)
             LTFTYP   = 1
```

```
            GA = GA/TU3
            TU1=TU2/TU3
            TU2=0.
            TU3=0.
            WRITE(8,1000)
            WRITE(8,1052)LTFID,LTFTYP,TU1,TU2,TU3,GA,X0,DX0,AA
     + ,LABEL(LTFTYP)
            GOTO 3
           END IF
          IF (TU3.EQ.0. .AND. TU1.GT.0. .AND.TU2.GT.0.)  THEN
            WRITE(8,1006)
            LTFTYP  = 3
            GA = GA/TU2
            TU1=TU1/TU2
            TU2=0.
            TU3=0.
            WRITE(8,1000)
            WRITE(8,1052)LTFID,LTFTYP,TU1,TU2,TU3,GA,X0,DX0,AA
     + ,LABEL(LTFTYP)
            GOTO 3
          END IF
            IFLAG = 1
            GOTO 2
        END IF
       IF (TU1.LE.0. .AND. LTFTYP.LE.4) THEN
          IFLAG =1
          WRITE(8,1003)
          GO TO 2
         END IF
    3 CONTINUE
       IF (TU2.GT.0. .AND. LTFTYP.LE.3) THEN
          TU2=0.
          WRITE(8,1003)
       END IF
       IF (LTFTYP.EQ.4 .AND. TU2.EQ.0. .AND. TU1.GT.0.) THEN
          LTFTYP = 1
          WRITE(8,1003)
          WRITE(8,1005)
            WRITE(8,1000)
            WRITE(8,1052)LTFID,LTFTYP,TU1,TU2,TU3,GA,X0,DX0,AA
     + ,LABEL(LTFTYP)
         END IF
       IF (TU1.EQ.TU2 .AND. LTFTYP.EQ.4) THEN
          WRITE(8,1007)
       END IF
       IF (TU3.GT.0. .AND. LTFTYP.LT.5) THEN
          TU3=0.0
          WRITE(8,1003)
         END IF
       IF (GA.LE.0.) WRITE(8,1008)
       TAU1(LTFID)  = TU1
```

```
      TAU2(LTFID)   = TU2
      TAU3(LTFID)   = TU3
      TFAUX(LTFID)  = X0
      TFAUX2(LTFID)= DX0
      KTFTYP(LTFID)   = LTFTYP
      GAIN(LTFID)     = GA
    2 WRITE(8,1053)
    4 WRITE(8,1053)
      IF (IFLAG.NE.0) STOP
      RETURN
 1000 FORMAT(' ***THIS TRANSFER FUNCTION WILL BE TREATED AS:' )
 1001 FORMAT(' ***ERROR IN TRNF IO*** KTFID OUT OF RANGE.' )
 1002 FORMAT(' ***ERROR IN TRNF IO*** THIS TRANSFER FUNCTION '
     + ,'ALREADY DEFINED.')
 1003 FORMAT(' ***ERROR IN TRNF IO*** TAU OUT OF RANGE.' )
 1004 FORMAT(' ***ERROR IN TRNF IO*** KTFTYP OUT OF RANGE.')
 1005 FORMAT('    SUBSTITUTING 1ST ORDER LAG.   KTFTYP -> 1.')
 1006 FORMAT('    SUBSTITUTING 1ST ORDER LAG WITH INTEGRATOR.  ',
     + 'KTFTYP -> 3.')
 1007 FORMAT(' ***ERROR IN TRNF IO*** TAU1 = TAU2. TRANSFER FUNCTION',
     + ' DEGENRATES TO UNITY.' )
 1008 FORMAT(' ***ERROR IN TRNF IO*** GAIN OUT OF RANGE.')
 1050 FORMAT(///,'1TRANSFER FUNCTIONS DEFINITIONS:',//,'  KTFID',
     + ' KTFTYP',6X,'TAU1',6X,'TAU2',6X,'TAU3',6X,'GAIN',9X,'X',
     + 8X,'DX',//)
 1051 FORMAT(2I2,5X,6F6.0,5X,A30)
 1052 FORMAT(2I7,6F10.5,/,' DESCRIPTION:  ',A30,2X,A30)
 1053 FORMAT(/)
      END
      SUBROUTINE TRNF(XIN,XOUT,T,KTFID)
C     THIS ROUTINE EVALUATES THE FOLLOWING TRANSFER FUNCTIONS
C       1ST ORDER LAG;
C       1ST ORDER LAG WITH DIFFERENTIATION;
C       1ST ORDER LAG WITH INTEGRATION;
C       COMBINED LEAD/LAG;
C       2ND ORDER LAG - HARMONIC OSCILLATOR.
C     XIN     DRIVING TERM.
C     XOUT    OUTPUT OF TRANSFER FUNCTION.
C     T       TIME.
C     KTFTYP  1 -> 1/(TAU1*S+1)
C             2 -> S/(TAU1*S+1)
C             3 -> 1/(S*(TAU1*S+1))
C             4 -> (TAU2*S+1)/(TAU1*S+1)
C             5 -> 1/(TAU1*S**2 + TAU2*S +TAU3)
C     KTFID   ID NO. OF CHANNEL (VARIABLE).
      COMMON/TRNSF/TAU1(50),TAU2(50),TAU3(50),TFTIME(50),TFAUX(50),
     + TFAUX2(50),KTFTYP(50),GAIN(50)
      REAL KK,II
      IF (KTFID.LT.1 .OR.KTFID.GT.50) GOTO 102
      KTF = KTFTYP(KTFID)
      IF (KTF.LT.1 .OR. KTF.GT.5) GOTO 103
```

```
        XOUT          = 0.
        XTMP          = 0.
        XTMP2         = 0.
      IF (TFTIME(KTFID).GT.T) GOTO 101
      TDELTA   = T - TFTIME(KTFID)
      IF (TDELTA.EQ.0.) THEN
         TFEXP  = 1.
         TFEXOT = 1./TAU1(KTFID)
         GOTO 1
      END IF
      TFEXP   = EXP(-TDELTA/TAU1(KTFID))
C     FOLLOWING CODE ASSURES CORRECT LIMITING VALUES AS TAU -> 0.
      IF (ABS(TAU1(KTFID)/TDELTA) .LT. 1.E-4) THEN
         TFEXOT = 1.0/TDELTA
      ELSE
         TFEXOT = TFEXP/TAU1(KTFID)
      END IF
    1 DRIVE    = XIN
      OMEGA    = DRIVE
      KTF  = KTFTYP(KTFID)
      IF (KTF.EQ.4) OMEGA=(1.-TAU2(KTFID)/TAU1(KTFID))*
     + DRIVE
      IF(KTF.EQ.1 .OR. KTF.EQ.4) XTMP =
     +   TFAUX(KTFID)*TFEXP+(1.-TFEXP)*OMEGA
      IF (KTF.EQ.1) XOUT  = XTMP
      IF (KTF.EQ.2) THEN
         XTMP  = TFEXOT*(DRIVE-TFAUX2(KTFID))+TFAUX(KTFID)*TFEXP
         XOUT  = XTMP
         XTMP2= DRIVE
        END IF
      IF(KTF.EQ.3) THEN
         TEMP  = TFAUX2(KTFID)-DRIVE
         XTMP=TFAUX(KTFID)+TAU1(KTFID)*TEMP*(1.-TFEXP)+DRIVE*TDELTA
         XOUT  = XTMP
         XTMP2  = TEMP*TFEXP+DRIVE
      END IF
      IF (KTF.EQ.4) XOUT  = DRIVE*(TAU2(KTFID)/TAU1(KTFID))
     + + XTMP
      IF (KTF.EQ.5) THEN
         KK       = TAU3(KTFID)
         II       = TAU1(KTFID)
         BETA     = TAU2(KTFID)
         FORCE    = DRIVE
         DELZ     = TFAUX(KTFID)
         DELDZ    = TFAUX2(KTFID)
         TLAST    = TFTIME(KTFID)
         CALL SOLAG(II,BETA,KK,DELZ,DELDZ,FORCE,T,TLAST,DELT,DELTD)
         XTMP     = DELT
         XTMP2    = DELTD
         XOUT     = XTMP
      END IF
```

```
  100 TFTIME(KTFID)   = T
      TFAUX(KTFID)    = XTMP
      TFAUX2(KTFID)   = XTMP2
      XOUT = XOUT*GAIN(KTFID)
C ABOVE SAVES XD FOR 2ND ORDER LAG/OSCILLATOR AND DX/DT FOR
C FIRST ORDER LAG WITH DIFFERENTIATOR. FOR USE AS INITIAL
C CONDITIONS ON NEXT INTEGRATION STEP.
      RETURN
  101 WRITE(8,200) T,TFTIME(KTFID),KTFID
      STOP
  102 WRITE(8,201) KTFID
      STOP
  103 WRITE(8,202) KTF
      STOP
  200 FORMAT(' ***ERROR*** IN TRNF. TIME=',F10.5,' IS GREATER THAN ',
     + 'LAST TIME=',F10.5,' FOR CHANNEL',I3,//)
  201 FORMAT(' ***ERROR*** IN TRNF. CHANNEL NO. KTFID=',I4,
     + ' IS OUT OF RANGE.'//)
  202 FORMAT(' ***ERROR*** IN TRNF. TYPE CODE NUMBER KTFID=',I4,
     + ' IS OUT OF RANGE.'//)
      END
      SUBROUTINE SOLAG(II,BETA,KK,DELZ,DELDZ,FORCE,T,TLAST,DELT,
     + DELTD)
C         DIFFERENTIAL EQUATION
C              II*YDDOT + BETA*YDOT + KK*Y = FORCE
C         NOTE: THE CALLING ROUTINE MUST SAVE TLAST,DELZ AND DELDZ
C         FOR THE NEXT CALL TO SOLAG
      REAL NUM ,LAMB1, LAMB2, LAMB, II, KK
      IF (BETA.LT.0. .OR. KK.LE.0.) THEN
         WRITE(8,1000) BETA,KK
         STOP
       END IF
      IF (II.LT.0.) THEN
         WRITE(8,1001) II
         STOP
       END IF
      BOV2I = BETA/(2.*II)
      RDCND = BOV2I**2 -KK/II
      FOVK  = FORCE/KK
      IF(RDCND .GT. 0.)GO TO 100
      IF(RDCND .EQ. 0.)GO TO 120
C         OSCILLATORY SOLUTION
      LAMB  =-BOV2I
      OMEGA = SQRT(-RDCND)
      NUM   = OMEGA*(DELZ-FOVK)
      DENOM = DELDZ-LAMB*(DELZ-FOVK)
      LAMB  = -BOV2I
      IF(NUM .EQ. 0.  .AND. DENOM .EQ. 0.) GO TO 20
      PHI = ATAN2(NUM,DENOM) - OMEGA*TLAST
      GO TO 30
   20 CONTINUE
```

```
      PHI = 0.
   30 CONTINUE
      ETEMP = EXP(-LAMB*TLAST)
      IF(DELZ - FOVK .EQ. 0.)GO TO 40
      A=(DELZ - FOVK)*ETEMP/SIN(OMEGA*TLAST+PHI)
      GO TO 50
   40 CONTINUE
      A=DELDZ*ETEMP/(LAMB*SIN(OMEGA*TLAST + PHI)
     1             +OMEGA*COS(OMEGA*TLAST + PHI)   )
   50 CONTINUE
      FTEMP = EXP(LAMB*T)
      DELT= A*FTEMP*SIN(OMEGA*T + PHI) + FOVK
      DELTD = LAMB*A*FTEMP*SIN(OMEGA*T+PHI) +
     1     OMEGA*A*FTEMP*COS(OMEGA*T+PHI)
      GO TO 150
  100 CONTINUE
C         EXPONENTIAL SOLUTION
      LAMB1 = -BOV2I+SQRT(RDCND)
      LAMB2 = -BOV2I-SQRT(RDCND)
      A = (DELZ - DELDZ/LAMB2 - FOVK)*EXP(-LAMB1*TLAST)/
     1        (1. - LAMB1/LAMB2)
      B = (DELZ - DELDZ/LAMB1 - FOVK)*EXP(-LAMB2*TLAST)/
     1        (1. - LAMB2/LAMB1)
      DELT = A*EXP(LAMB1*T) + B*EXP(LAMB2*T) + FOVK
      DELTD = LAMB1*A*EXP(LAMB1*T)+LAMB2*B*EXP(LAMB2*T)
      GO TO 150
  120 CONTINUE
C     CRITICALLY DAMPED SOLUTION
      LAMB = -BOV2I
      A=((DELZ-FOVK)*(1+TLAST*LAMB)-TLAST*DELDZ)*EXP(-LAMB*TLAST)
      B=(DELDZ+LAMB*(FOVK-DELZ))*EXP(-LAMB*TLAST)
      DELT   = (A+B*T)*EXP(LAMB*T)+FOVK
      DELTD = (LAMB*A+B+LAMB*B*T)*EXP(LAMB*T)
  150 CONTINUE
      TLAST = T
      DELZ  = DELT
      DELDZ = DELTD
      RETURN
 1000 FORMAT(/,' ***-ERR IN SOLAG-*** BETA=',E12.4,',KK=',E12.4,/,
     + ' ONLY POSITIVE DEFINITE VALUES ALLOWED.',/)
 1001 FORMAT(/,' ***-ERR IN SOLAG-*** II=',E12.4,/
     + ' NEGATIVE VALUES NOT ALLOWED.',/)
      END
```

SAMPLE DATA INPUT CARDS.

```
K K      T     T     T     G     X     D          A
T T      A     A     A     A     .     X          A
F F      U     U     U     I     .     .          .
I T      1     2     3     N     .     .          .
D Y      .     .     .     .     .     .          .
. P      .     .     .     .     .     .          .
. .      .     .     .     .     .     .          .

0101     .02   0.    0.    4.0   0.0   0.         PITCH ATTITUDE HOLD
0202     .2    0.    0.    2.0   0.0   0.         PITCH SYNTHETIC DAMPING
0303     .02   0.    0.    2.0   0.0   0.         YAW ATTITUDE HOLD
0404     .02   .01   0.0   1.0   0.0   0.         YAW LOS RATE FILTER
0504     .02   0.    0.0   1.0   0.0   0.         YAW LOS RATE FILTER
0605     1.    3.    .25   1.0   10.0  0.         PURE DAMPING
0705     1.    1.    .25   1.0   10.0  0.         CRITICALLY DAMPED
0805     1.    .5    .25   1.0   10.0  0.         DAMPED OSCILLATION
0905     1.    .0    .25   1.0   10.0  0.         PURE OSCILLATION
1005     0.    .5    .25   1.0   10.0  0.
1105     1.    .5    .0    1.0   10.0  0.
1205     .001  .0002 20.   1.0   10.0  0.
1305     .0002 0.0   5000. 1.0   10.0  0.
```

SAMPLE OUTPUT

```
TRANSFER FUNCTIONS DEFINITIONS:

 KTFID KTFTYP       TAU1       TAU2       TAU3       GAIN         X          DX

     1       1     .02000     .00000     .00000    4.00000     .00000     .00000
DESCRIPTION:   PITCH ATTITUDE HOLD                 GAIN/(TAU1*S+1)

     2       2     .20000     .00000     .00000    2.00000     .00000     .00000
DESCRIPTION:   PITCH SYNTHETIC DAMPING             GAIN*S/(TAU1*S+1)

     3       3     .02000     .00000     .00000    2.00000     .00000     .00000
DESCRIPTION:   YAW ATTITUDE HOLD                   GAIN/(S*(TAU1*S+1))

     4       4     .02000     .01000     .00000    1.00000     .00000     .00000
DESCRIPTION:   YAW LOS RATE FILTER                 GAIN*(TAU2*S+1)/(TAU1*S+1)

     5       4     .02000     .00000     .00000    1.00000     .00000     .00000
DESCRIPTION:   YAW LOS RATE FILTER                 GAIN*(TAU2*S+1)/(TAU1*S+1)
***ERROR IN TRNF IO*** TAU OUT OF RANGE.
   SUBSTITUTING 1ST ORDER LAG.   KTFTYP -> 1.
***THIS TRANSFER FUNCTION WILL BE TREATED AS:
     5       1     .02000     .00000     .00000    1.00000     .00000     .00000
DESCRIPTION:   YAW LOS RATE FILTER                 GAIN/(TAU1*S+1)

     6       5    1.00000    3.00000     .25000    1.00000   10.00000     .00000
DESCRIPTION:   PURE DAMPING                        GAIN/(TAU1*S**2+TAU2*S+TAU3)

     7       5    1.00000    1.00000     .25000    1.00000   10.00000     .00000
DESCRIPTION:   CRITICALLY DAMPED                   GAIN/(TAU1*S**2+TAU2*S+TAU3)

     8       5    1.00000     .50000     .25000    1.00000   10.00000     .00000
DESCRIPTION:   DAMPED OSCILLATION                  GAIN/(TAU1*S**2+TAU2*S+TAU3)

     9       5    1.00000     .00000     .25000    1.00000   10.00000     .00000
DESCRIPTION:   PURE OSCILLATION                    GAIN/(TAU1*S**2+TAU2*S+TAU3)
```

```
    10       5    .00000    .50000    .25000   1.00000  10.00000    .00000
DESCRIPTION:                                   GAIN/(TAU1*S**2+TAU2*S+TAU3)
***ERROR IN TRNF IO*** TAU OUT OF RANGE.
   SUBSTITUTING 1ST ORDER LAG.   KTFTYP -> 1.
***THIS TRANSFER FUNCTION WILL BE TREATED AS:
    10       1   2.00000    .00000    .00000   4.00000  10.00000    .00000
DESCRIPTION:                                   GAIN/(TAU1*S+1)


    11       5   1.00000    .50000    .00000   1.00000  10.00000    .00000
DESCRIPTION:                                   GAIN/(TAU1*S**2+TAU2*S+TAU3)
***ERROR IN TRNF IO*** TAU OUT OF RANGE.
   SUBSTITUTING 1ST ORDER LAG WITH INTEGRATOR.  KTFTYP -> 3.
***THIS TRANSFER FUNCTION WILL BE TREATED AS:
    11       3   2.00000    .00000    .00000   2.00000  10.00000    .00000
DESCRIPTION:                                   GAIN/(S*(TAU1*S+1))


    12       5    .00100    .00020  20.00000   1.00000  10.00000    .00000
DESCRIPTION:                                   GAIN/(TAU1*S**2+TAU2*S+TAU3)


    13       5    .00020    .000005000.00000   1.00000  10.00000    .00000
DESCRIPTION:                                   GAIN/(TAU1*S**2+TAU2*S+TAU3)
```

```
THE INTEGRATION TIME STEP DT =    .005000 .


LTFID                  TIME               DRIVE                   X
    1                .00000              .00000              .00000
    1               1.00000            -1.41615            -5.53438
    1               2.00000            -1.65364            -6.71729
    1               3.00000             -.03983             -.20403
    1               4.00000            -1.14550            -4.44207
    1               5.00000            -1.83907            -7.42804

LTFID                  TIME               DRIVE                   X
    2                .00000              .00000              .00000
    2               1.00000            -1.41615            -3.67251
    2               2.00000            -1.65364             1.68669
    2               3.00000             -.03983             2.25946
    2               4.00000            -1.14550            -3.56729
    2               5.00000            -1.83907              .70957


LTFID                  TIME               DRIVE                   X
    3                .00000              .00000              .00000
    3               1.00000            -1.41615            -1.04245
    3               2.00000            -1.65364            -4.69789
    3               3.00000             -.03983            -6.27757
    3               4.00000            -1.14550            -6.97196
    3               5.00000            -1.83907           -10.47893

LTFID                  TIME               DRIVE                   X
    4                .00000              .00000              .00000
    4               1.00000            -1.41615            -1.39987
    4               2.00000            -1.65364            -1.66648
    4               3.00000             -.03983             -.04542
    4               4.00000            -1.14550            -1.12801
    4               5.00000            -1.83907            -1.84804

LTFID                  TIME               DRIVE                   X
    5                .00000              .00000              .00000
    5               1.00000            -1.41615            -1.38360
    5               2.00000            -1.65364            -1.67932
    5               3.00000             -.03983             -.05101
    5               4.00000            -1.14550            -1.11052
    5               5.00000            -1.83907            -1.85701
```

```
LTFID           TIME          DRIVE              X
    6         .00000         .00000       10.00000
    6        1.00000       -1.41615        9.35272
    6        2.00000       -1.65364        8.09988
    6        3.00000        -.03983        7.04001
    6        4.00000       -1.14550        6.35853
    6        5.00000       -1.83907        5.38481

LTFID           TIME          DRIVE              X
    7         .00000         .00000       10.00000
    7        1.00000       -1.41615        8.97774
    7        2.00000       -1.65364        6.33926
    7        3.00000        -.03983        3.54539
    7        4.00000       -1.14550        1.78784
    7        5.00000       -1.83907         .10284

LTFID           TIME          DRIVE              X
    8         .00000         .00000       10.00000
    8        1.00000       -1.41615        8.82387
    8        2.00000       -1.65364        5.36682
    8        3.00000        -.03983        1.15018
    8        4.00000       -1.14550       -1.86486
    8        5.00000       -1.83907       -4.23518

LTFID           TIME          DRIVE              X
    9         .00000         .00000       10.00000
    9        1.00000       -1.41615        8.62980
    9        2.00000       -1.65364        3.87712
    9        3.00000        -.03983       -3.25255
    9        4.00000       -1.14550       -9.90176
    9        5.00000       -1.83907      -15.20981

LTFID           TIME          DRIVE              X
   10         .00000         .00000       40.00000
   10        1.00000       -1.41615       23.29664
   10        2.00000       -1.65364       11.23015
   10        3.00000        -.03983        5.73197
   10        4.00000       -1.14550        2.81734
   10        5.00000       -1.83907       -1.12071

LTFID           TIME          DRIVE              X
   11         .00000         .00000       20.00000
   11        1.00000       -1.41615       19.86679
   11        2.00000       -1.65364       18.71997
   11        3.00000        -.03983       16.91363
   11        4.00000       -1.14550       15.57969
```

```
      11              5.00000            -1.83907           13.85090

LTFID                   TIME              DRIVE                   X
   12                 .00000             .00000            10.00000
   12                1.00000           -1.41615            -9.10857
   12                2.00000           -1.65364             8.06504
   12                3.00000            -.03983            -7.32883
   12                4.00000           -1.14550             6.51484
   12                5.00000           -1.83907            -5.97223

LTFID                   TIME              DRIVE                   X
   13                 .00000             .00000            10.00000
   13                1.00000           -1.41615             1.54640
   13                2.00000           -1.65364            -9.52188
   13                3.00000            -.03983            -4.49206
   13                4.00000           -1.14550             8.13177
   13                5.00000           -1.83907             7.00721
```

APPENDIX B

PROGRAM USAGE

The program consists of three subroutines (TRNF IO, TRNF, and SOLAG) and a main program, TFMAIN.* TRNF IO is the initialization routine. The data records that define the transfer functions are read by this routine. (See the explanation for input data format given below.) TRNF performs the analytical transfer function simulations. It accesses subroutine SOLAG for treatment of the second order lag/harmonic oscillator. The driving main program TFMAIN is supplied for demonstration and testing only. Use it as a model for interfacing TRNF IO, TRNF, and SOLAG with your computer program. The initial time is at line 100, and the integration time step is at line 200. The driving term is defined in the lines following 300.

The call to TRNF contains four arguments:

DRIVE	Driving function
XOUT	Output of the transfer function channel
T	Time
J	Channel I.D. number LTFID

One input data record is required for each transfer function variable to be integrated. These records have the following form:

Column	Format	Variable	Definition
1	I2	LTFID	I.D number of channel; values are from 1 to 50
3	I2	LTFTYP	Code for type of transfer function; values are 1 to 5
10	F9.0	TAU1	First constant, for all LTFTYP
19	F9.0	TAU2	Second constant, for LTFTYP = 4 or 5
28	F9.0	TAU3	Third constant, for LTFTYP=5
37	F9.0	X0	Initial condition for variable
46	F9.0	DX0	Initial condition for derivative of variable
55	A26	AA	Alphanumeric label

LTFID is an integer variable from 1 to 50 that is associated with each transfer function channel (output variable) to be simulated. For example, a lag

* A listing of a program implementing these techniques and a sample run are in appendix A.

might be applied to both a fin yaw deflection command and to a fin pitch deflection command. Each of these two channels is defined as a separate data input record and is assigned an identifying integer LTFID. If the user chooses an integer that was previously assigned on another input data record, execution terminates with an explanatory message.

LTFTYP is an integer variable in the range 1 to 5 that defines the type of transfer function:

LTFTYP	Description and FORTRAN code
1	First-order lag 1/(TAU1*S + 1)
2	First-order lag with differentiator S/(TAU1*S + 1)
3	First-order lag with integrator 1/[S*(TAU1*S + 1)]
4	Combined lead/lag (TAU2*S + 1)/(TAU1*S + 1)
5	Second-order lag/harmonic oscillator 1/(TAU1*S^2 + TAU2*S + TAU3)

TAU1 is the first constant (see LTFTYP above). All five transfer function types (LTFTYP = 1 to 5) require this quantity to be positive.

TAU2 is the second constant. Lead/lag and second-order lag/oscillator transfer functions (LTFTYP = 4,5) require a positive value.

TAU3 is the third constant. The second-order lag/oscillator (LTFTYP = 5) requires a positive value.

XO is the initial condition on the output variable, i.e., $y_o = y$ at $t = t_o$.

DXO is the initial condition on the derivative of the output variable, $y' = dy/dt$ at $t = t_o$.

AA is an alphanumeric label of 26 characters that identifies the nature of the transfer function when an echo of the input data is printed out by TRNF IO. Examples might be "PITCH SYNTHETIC DAMPING" or "YAW ATTITUDE HOLD."

If an input variable is out of range, the program prints an error message and reformulates the transfer function into an equivalent form. For example, if an attempt is made to run an oscillator (LTFTYP = 5) with TAU1 = 0, subroutine TRNF IO will substitute the equivalent simple lag (LTFTYP = 1), as can be seen in the sample output for KTFID = 10. Two other examples appear in the sample output in the sample execution, i.e., for KTFID = 5 and KTFID = 11.

SYMBOLS

Algebraic Expressions

A	Constant of integration
B	Constant of integration
D	Coefficient of the Laplace operator s in the transfer function of the second-order lag/oscillator
I	Coefficient of the square of the Laplace operator s in the transfer function of the second-order lag/oscillator
K	Constant term in the transfer function of the second-order lag/oscillator
L	Damping coefficient -D/2I for the second-order lag/oscillator
s	Laplace transform operator
t	Time, independent variable
T	Time constant
D	Driving term
y	Output of transfer function, dependent variable
y_o	Initial condition on y at $t = t_o$
y'	Initial condition on dy/dt at $t = t_o$
z	Auxiliary dependent variable for first-order lag with differentiation or for first-order lag with integration
ω	Frequency of oscillation for second-order lag/oscillator
ϕ	Phase angle of oscillation for second-order lag/oscillator

FORTRAN Variable Names

AA	Alphanumeric label for printout (80 characters)
DX	Initial condition on derivative when KTFTYP=5
KTFID	I.D. number of channel or variable (range is 1 to 50)
KTFTYP	Integer variable used to indicate type of transfer function (range 1 to 50) KTFTYP = 1 is first-order lag Transfer function: GAIN/(TAU1*S+1) KTFTYP = 2 is first-order lag with differentiation Transfer function: GAIN*S/(TAU1*S+1) KTFTYP = 3 is first-order lag with integration Transfer function: GAIN/[S*(TAU1*S+1)] KTFTYP = 4 is combined lead/lag Transfer function: GAIN*(TAU2*S+1/(TAU1*S+1) KTFTYP = 5 is second-order lag/harmonic oscillator Transfer function: GAIN/(TAU1*S**2 + TAU2*S +TAU3)
T	Time
TAU1	First time constant
TAU2	Second time constant (T_1 for KTFTYP=4 or D for KTFTYP)
TAU3	Third time constant (K for KTFTYP=5)
X	Initial condition on variable
XIN	Driving term
XOUT	Output of transfer function

DISTRIBUTION LIST

Department of the Army
ATTN: DAMA-CSM
      DAMO-RQS
Washington, DC 20314

Commander
U.S. Army Materiel Command
ATTN: AMCDE-DM
5001 Eisenhower Avenue
Alexandria, VA 22304

Commander
Armament Research and Development Center
U.S. Army Armament, Munitions and Chemical Command
ATTN: SMCAR-TSS (5)
      SMCAR-LC, COL A. T. Crumpton
                T. Davidson
      SMCAR-LCS, J. Gregorits
      SMCAR-LCA, L. Ambrosini
                 L. Marino
                 F. Scerbo
      SMCAR-LCA-F, A. Loeb
                   D. Mertz (10)
                   R. Kline (3)
                   E. Friedman (3)
                   M. Amoruso (3)
      SMCAR-LCU, B. Bushey
Dover, NJ 07801-5001

Commander
U.S. Army Armament, Munitions and Chemical Command
ATTN: AMSMC-GCL(D)
Dover, NJ 07801-5001

Administrator
Defense Technical Information Center
ATTN: Accessions Division (12)
Cameron Station
Alexandria, VA 22304-6145

Director
U.S. Army Materiel Systems Analysis Activity
ATTN: AMXSY-MP
Aberdeen Proving Ground, MD 21005-5066

Commander
Chemical Research and Development Center
U.S. Army Armament, Munitions and Chemical Command
ATTN: SMCCR-SPS-IL
Aberdeen Proving Ground, MD 21010-5423

Commander
Chemical Research and Development Center
U.S. Army Armament, Munitions and Chemical Command
ATTN: SMCCR-RSP-A
Aberdeen Proving Ground, MD 21010-5423

Director
Ballistic Research Laboratory
ATTN: AMXBR-OD-ST (2)
AMXBR-BLL, R. Lieske
W. Mermagen
C. Murphy
V. Oskay
Aberdeen Proving Ground, MD 21005-5066

Chief
Benet Weapons Laboratory, LCWSL
Armament Research and Development Center
U.S. Army Armament, Munitions and Chemical Command
ATTN: SMCAR-LCB-TL
Watervliet, NY 12189-5000

Commander
U.S. Army Armament, Munitions and Chemical Command
ATTN: AMSMC-LEP-L
AMSMC-PDM
AMSMC-ASI
Rock Island, IL 61299-6000

Director
U.S. Army TRADOC Systems Analysis Activity
ATTN: ATAA-SL
White Sands Missile Range, NM 88002

Headquarters
Air Force Weapons Laboratory (WLX)
Kirtland Air Force Base, NM 87117

Headquarters
U.S. Army Training and Doctrine Command
ATTN: ATCD-MC
ATAA-SL
Ft. Monroe, VA 23651

Director
Marine Corps Development and Engineering Command
ATTN: Code D092
Quantico, VA 22134

Headquarters
Eglin Air Force Base
ATTN: Technical Library
Eglin Air Force Base, FL 32542

Commander
U.S. Army Test and Evaluation Command
ATTN: DRSTE-CT-T
Aberdeen Proving Ground, MD 21005

Commander
U.S. Naval Weapons Center
ATTN: Technical Library (Code 5557)
China Lake, CA 93555

Commander
U.S. Naval Surface Weapons Center
White Oak Laboratory
ATTN: Research Library
Silver Spring, MD 20910

Commander
U.S. Naval Surface Weapons Center
Dahlgren Laboratory
ATTN: Technical Library
Dahlgren, VA 22448

Commander/Director
Naval Ship Research and Development Center
ATTN: Technical Library
Washington, DC 20007

Commanding General
U.S. Army Missile Command
ATTN: Technical Library, R. Deep
Redstone Arsenal, AL 35809