MISSISSIPPI STATE UNIVERSITY

State College, Mississippi

Interim Report

Contract No. NAS8-21377

Covering Period September 1, 1970 - January 31, 1971

NATIONAL AERONAUTICS AND SPACE ADMINISTRATION

COMPENSATOR GROWING AND IMPROVING PROGRAM

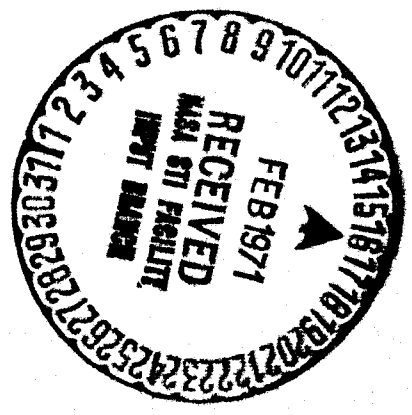
By

Jerrel R. Mitchell

Submitted on behalf of

Willie L. McDaniel, Jr.

Professor of Electrical Engineering



ABSTRACT

Presented in this report is a computer-aided compensator design program. Included with the main program is a set of catalogued comments for clarifying many of the functions of the program. All subprograms are preceded by comments describing the function of the program and defining the input-output (I/O) variables of the program. Also, programs classified as specialized are preceded by theoretical backgrounds.

COMPENSATOR GROWING AND IMPROVING PROGRAM

Introduction

In most control systems it is necessary to compensate them in order that the system possess certain desirable properties. One such property that is desired in almost all cases is a specific degree of relative stability. In the past the design procedures for achieving compensators for accomplishing this purpose have been trial and error techniques. The general ideas behind these procedures are designing a compensator and checking the relative stability; if relative stability is an acceptable level the compensator is accepted; if not, the compensator is rejected and a redesign is necessary. Such a procedure as this can be tedious and time consuming. A better approach is a method called compensator growing.[1] It is the purpose of this report to present a digital computer program for performing compensator growing.

Compensator Growing Program

In the following the various routines for performing compensator growing are given. Along with each program are explanatory comments and a theoretical background, if it has some specialized function not commonly available in computing facility libraries. All the programs are written in Fortran V, and the whole program is self-dependent (does not depend upon the system library).

Program Main

The purpose of the main program is to act as a controlling device. It is used for controlling input, output, and procedural decisions. A general flow chart of this program is shown in Figure 1.

In order to make thorough explanations of the program all comments in the program are numbered, and the explanations are catalogued in the following. When a comment number is followed by the word "Input", this designates that in this comment the variables in an input statement or statements are defined. In these comment statements there may also be explanatory remarks.

Catalogued Comments

Comment 1 - Input

KSTART - The number of the first iteration to be performed.

KQUIT - The number of the last iteration to be performed.

GAIN - Used for connoting whether positive or negative feedback is to be employed. It should be -1 for positive feedback and +1 for negative feedback.

STPMAX - Used to specify the maximum step size to be used on any iteration.

Comment 2 - Input

KINFOR - Denotes whether the open loop transfer function is the input or whether points on the frequency response are the input. KINFOR = 1 means the open loop transfer function is given. KINFOR = 2 means points from the open loop frequency response are given.

KPOINT - The number of points on the open loop frequency response that are given.

Comment 3 - Input

K1 - Maximum order of open loop transfer function.

K2 - Maximum order of the numerator of the compensator.

K3 - Maximum order of the denominator of the compensator.

R - Desired stability margin.

IOP - Designates whether open loop gain is to be held constant. If IOP = 0 open loop gain is allowed to vary. If IOP = +1 open loop gain is held constant.

Comment 4 - Input

COMM - Array of the numerator coefficients of the compensator*.

COMD - Array of the denominator coefficients of the compensator.

The compensator described by these coefficients is known as the variable compensator. This is the compensator that is to be varied so as to improve the relative stability of the system.

Comment 5 - Input

KLOCK - This variable determines whether the directional vector method is to be used for determining how the compensator coefficients are changed. For KLOCK = +1 only the directional vector method is used. The directional vector method is more applicable for compensator improvement.

M1 - Connotes the order of the numerator of the constant compensator.

M2 - Connotes the order of the denominator of the constant compensator.

---

*All input and output polynomial coefficients are assumed to be in ascending order according to the powers of s.

Comment 6 - Input

CNST - Array of the coefficients of the numerator of the constant compensator.

CDST - Array of the coefficients of the denominator of the constant compensator.

The compensator described by CNST and CDST is assumed to be in a cascade configuration with the variable compensator. This compensator is not changed by the program. It is used to make any wanted changes in open loop frequency response information. With the open loop transfer function as the input--rather than frequency response information--this compensator has no meaning. If no constant compensator is to be used then M1 and M2 should be set equal to 0 and two blank cards should be inserted for CNST and CDST.

Comment 7 - Input

POLG - Array of the numerator of the open loop transfer function.

POLH - Array of the denominator of the open loop transfer function.

If open loop frequency response information is used, these arrays are never used. In this case the read statements for these are disregarded by the program.

Comment 8 - Input

OMEGA - Array of chosen frequency points that are to be considered.

GR - An array of the real parts of the open loop frequency response at the chosen OMEGA's.

GI - An array of the corresponding imaginary parts of the open loop frequency response at the chosen OMEGA's.

If the open loop transfer function is read in, the read statement of these variables is ommitted by the program.

Comment 9 - Initializations of some variables used in the program.

Only one of these should be changed; it is STEP. Whatever value is given to this variable at this point will simply determine the initial step size. In most instants it is better for this variable to be at least one-tenth smaller than the smallest coefficient of the variable compensator.

Comment 10 - The following statements determine the minima with respect to -1 + j0 point.

If frequency response information has been read in, the sub-program SRMINS is used. This program finds the approximate minima by a search algorithm. On the other hand, if the open loop transfer function is being used then the sub-program OPENL is used to find the minima with respect to the -1 + j0 point. This program uses an analytical method.

Comment 11

The following statements are used for calculation of the real and imaginary parts of the open loop transfer function at each of the minima with respect to the -1 + j0 point.

Comment 12 - Statements used for calculating the stability margins.

Some of the variables used in these statements are:

NM - Number of minima with respect to the -1 + j0 point

RQ - Array of the required stability margins. The statement RQ(I) = R sets all the margins to the same value. However, by using various IF statements in place of RQ = R the stability margins can be made a function of frequency. For example suppose that it was desirable for all stability margins occuring at frequencies less to 1.0 to be 1.5R and

those greater than or equal to 1.0 to be R. In this case RQ(I) = R should be replaced by

```
IF(X(I).LT.1.0)RQ(I) = 1.5*R
IF(X(I).GE.1.0)RQ(I) = R
```

Of course many other combinations can be achieved.

X - Array of the frequencies at which the minima occur.

STBM - Array of the stability margins.

SML1 - Smallest stability margin for this iteration.

SBC1 - Largest stability margin of all the smallest at all previous iterations.

IBEST - Iteration from which SBC1 came.

Comment 13

The iteration, IBEST, is considered to be the iteration where the best compensator occurred because this is where the system possessed the greatest stability. Thus the variables BCOMN and BCOMD are arrays used to store, respectively, the numerator and the denominator coefficients of the best compensator for later output.

Comment 14

The DO loop terminating at 320 is used to determine if system specifications have been achieved. If they have, the program is terminated after all pertinent information has been printed. If not, the program continues unless the iteration number (LOX) exceeds the stopping iteration number (KQUIT). If LOX > KQUIT, the program is terminated.

Comment 15

The following statements are used to calculate the step size for the present iteration. The theory is that if relative stability

has increased from the past iteration to the present that the step size should be increased. The step size is never allowed to exceed STPMAX. On the other hand, if a decrease in relative stability occurs, the iteration is repeated (although the iteration number increases) with a step size one-half the immediate past step size. This is repeated until a decrease in relative stability occurs or until the step size gets below a predetermined value specified in the IF statement following the statement 360 STEP = STEP/2.0.

Comment 16

The following statements are used to output certain information at some iterations. The variable KHOP designates the number of iterations in which the step size is increased that should be skipped before additional information is printed out. The sub-program OTPT1 is the main output device of the program.

Comment 17

The sub-program PARCLT calculates the vectors (PRX and PRY, numerator and denominator vectors respectively) that determine how the compensator coefficients are to be varied. One of two methods may be employed. The first of these is the weighted gradient sum. The second of these methods is the directional vector method. The second determines a vector which has a component in the direction of the gradients of all the minima where stability margins are less than a certain prescribed level. If KLOCK = 1 the computer uses the directional vector method only.

Comment 18

If the gradient sum method is used, the conjugate gradient method is employed in order to speed up the solution process. The

following statements are used to reset the conjugate gradient directions (PX,PY) to 0 every L1 + L2 + 1 iterations. If the directional vector method is employed resetting occurs every iteration. The result being that the directional vector direction is left unaltered by the conjugate gradient procedures.[2]

Comment 19

Actual calculations of the conjugate directions is accomplished by the following DO loop. It should be obvious that if the directional vector method is used the conjugate directions simply become the negative of the directional vector directions.

Comment 20

The following two DO loops change the compensator coefficients according to the calculated direction vectors PX (numerator) and PY (denominator). The variable DEL is the step size divided by the magnitude of the calculated direction vector. The compensator coefficients are constrained to be greater than or equal to 0. The reason for this is to maintain the stability of the compensator.

```
C
C MAIN PROGRAM
C
      DIMENSION PCL(50), PCLL(50), COMR(50), COMI(50), X(30), PRY(30),
     1   PRA(30), STR(30), PX(30), PY(30), RC(30), GR(500), GI(500),
     2   PR(50), PI(50), CRLGA(500), PCLG(50), POLP(50)
      DIMENSION RCOMR(30), RCOMI(30)
      DIMENSION COMR(30), COMI(30), CRST(30), CDST(30)
      COMMON/GAIN/RLOCUS
      COMMON/CGE/COMR,COMI,GAIN
C COMMENT  1
      READ(5,10) RSTAMG,RGLIM,GAIN,STPMAX
   10 FORMAT(15,I5F10.5,5F10.0)
C COMMENT  2
      READ(5,30)RSIMCH,RPOINT
C COMMENT  3
      READ(5,30) N1, N2, N3, NP, IOP
   30 FORMAT(12OUX,I1,3X,I2,8X,F5.3,5X,I1)
      WRITE(6,40) N1, N2, R
   40 FORMAT('1',15X,'MAXIMUM ORDER IN OPEN LOOP TRANSFER FUNCTION =',
     1  I2//13X,'MAXIMUM ORDER IN COMPENSATON TRANSFER FUNCTION =',I2
     2  //23X,'DESIRED STABILITY MARGIN =',F5.3)
      N4= N2
      IF(N3.GT.N2)N4= N3
      L1=N1+1
      L2= N4 + 1
      LNC= N2 + 1
      LDC= N3 + 1
C COMMENT  4
      READ(5,50) (COMR(I),I=1,L2)
      READ(5,50) (COMI(I),I=1,L2)
   50 FORMAT(5E15.5)
   60 FORMAT('0',20X,'INITIAL COMPENSATOR COEFFICIENTS')
C COMMENT  5
      READ(5,10)RLOCH
      READ(5,10) N1, N2
      N1= N1 + 1
      N2= N2 + 1
C COMMENT  6
      READ(5,50) (CRST(I),I=1,N1)
      READ(5,50) (CDST(I),I=1,N2)
      WRITE(6,60)
      WRITE(6,65)
   65 FORMAT('0',10X,'CONSTANT COMPENSATOR')
      WRITE(6,70) (CRST(I),I=1,N1)
      WRITE(6,70) (CDST(I),I=1,N2)
   75 FORMAT('0',10X,'VARIABLE COMPENSATOR')
      WRITE(6,80)
      WRITE(6,75)
      WRITE(6,70) (COMR(I),I=1,L2)
      WRITE(6,70) (COMI(I),I=1,L2)
   70 FORMAT(' ',5X,10F10.5)
```

```
C
C  MAIN PROGRAM
C
      DIMENSION PCLR(50), PCLI(50), COMR(50), COMI(50), X(30), PRY(30),
     1   PRX(30), STEP(30), PX(30), PY(30), RG(30), GR(500), GI(500),
     2   PR(50), PI(50), OMEGA(500), POLG(50), POLP(50)
      DIMENSION ICOMR(30), ICOMI(30)
      DIMENSION COTR(30), COTI(30), CRST(30), CIST(30)
      COMMON/GAIN/KLOOP
      COMMON/COE/COMR,COMI,GAIN
C COMMENT  1
      READ(5,10) KSTART, KWRITE, GAIN, STPMAX
   10 FORMAT(I5,I2,F10.5,F10.0)
C COMMENT  2
      READ(5,30)KAIFOR,KPOINT
C COMMENT  3
      READ(5,30) K1, K2, K3, K, IOP
   30 FORMAT(12X,I4,2X,I2,2X,I2,2X,F5.3,5X,I1)
      WRITE(6,40) K1, K2, K
   40 FORMAT('1',15X,'MAXIMUM ORDER IN OPEN LOOP TRANSFER FUNCTION =',
     1  I2//13X,'MAXIMUM ORDER IN COMPENSATOR TRANSFER FUNCTION =',I2
     2  //23X,'DESIRED STABILITY MARGIN =',F5.3)
      K4= K2
      IF(K3.GT.K2)K4= K3
      L1=K1+1
      L2= K4 + 1
      LRC= K2 + 1
      LLC= K3 + 1
C COMMENT  4
      READ(5,50) (COMR(I),I=1,L2)
      READ(5,50) (COMI(I),I=1,L2)
   50 FORMAT(5E15.5)
   60 FORMAT('0',3X,'INITIAL COMPENSATOR COEFFICIENTS')
C COMMENT  5
      READ(5,10)KLOCK
      READ(5,10) M1, M2
      N1= M1 + 1
      N2= M2 + 1
C COMMENT  6
      READ(5,50) (CRST(I),I=1,N1)
      READ(5,50) (CIST(I),I=1,N2)
      WRITE(6,60)
      WRITE(6,65)
   65 FORMAT('0',15X,'CONSTANT COMPENSATOR')
      WRITE(6,70) (CRST(I),I=1,N1)
      WRITE(6,70) (CIST(I),I=1,N2)
   75 FORMAT('0',15X,'VARIABLE COMPENSATOR')
      WRITE(6,60)
      WRITE(6,75)
      WRITE(6,70) (COMR(I),I=1,L2)
      WRITE(6,70) (COMI(I),I=1,L2)
   70 FORMAT(' ',3X,10F10.5)
```

```
      IF(KINFOR.GT.1)GO TO 130
C COMMENT  7
      READ(5,90) (POLG(I),I=1,L1)
      READ(5,90) (POLH(I),I=1,L1)
   90 FORMAT(5E16.8)
      DO 100 I=1,L1
      KH= L1 + 1 - I
      POLN(KH)= POLG(I)
  100 POLL(KH)= POLH(I)
      WRITE(6,110)
  110 FORMAT('0',10X,'COEFFICIENTS OF OPEN LOOP TRANSFER FUNCTION')
      WRITE(6,120) (POLN(I),I=1,L1)
      WRITE(6,120) (POLL(I),I=1,L1)
  120 FORMAT('0',5X,10F12.3)
      GO TO 150
C COMMENT  8
  130 READ(5,140)  (OMEGA(I),GR(I),GI(I),I=1,KPOINT)
  140 FORMAT(9F8.3)
      IF(M1+M2.EQ.0)GO TO 150
      DO 148 I=1,KPOINT
      XV= OMEGA(I) * 6.2832
      CALL POLFV(CNST,M1,XV,CNR,CNI)
      CALL POLFV(CDST,M2,XV,CDR,CDI)
      G1= SQRT(CNR**2+CNI**2)
      AN1= ATAN2(CNI,CNR)
      G2= SQRT(CDR**2+CDI**2)
      AN2= ATAN2(CDI,CDR)
      G3= SQRT(GR(I)**2+GI(I)**2)
      AN3= ATAN2(GI(I),GR(I))
      GT= (G1*G3)/G2
      ANT= AN1 - AN2 + AN3
      GR(I)= GT * COS(ANT)
  148 GI(I)= GT * SIN(ANT)
  150 CONTINUE
  180 FORMAT('0',25X,'FREQUENCY TO BE SUPPRESSED =',F10.5)
  190 CONTINUE
C COMMENT  9
      STEP= 0.01
      KHOP=0
      LOX= KSTART
      SML2= 0.0
      PSGL= 1.0E20
  200 CONTINUE
      NM=0
      IF(KINFOR-1)220,220,210
C COMMENT 10
  210 CALL SKMINS(COMN,COML,K4,GR,GI,KPOINT,OMEGA,X,NM,NK,HI)
      GO TO 250
  220 CALL OPENL(POLN,POLL,K1,COMN,COMD,K4,X,NG,NM,R,KERROR)
C COMMENT 11
      IF(KERROR)450,230,450
  230 CONTINUE
```

```
      DO 240 I=1,NM
      XV= X(I)
      CALL POLFV(PCLR,K1,XV,PLR,PNI)
      CALL POLFV(PCLD,K1,XV,PLR,PDI)
      PL= PDR**2 + PDI**2
      HR(I)= (PNR*PLR + PNI*PLI)/PL
  240 HI(I)=(-PNR*PLI + PNI*PDR)/PL
  250 CONTINUE
C COMMENT 12
      SBC1= R
      SML1= 100.0
      DO 290 I=1,NM
      RG(I)= R
      XV= X(I)
      CALL POLFV(CONR,K2,XV,CLR,CNI)
      CALL POLFV(COND,K3,XV,CLR,CDI)
      CL= CLR**2 + CDI**2
      CR= (CNR*CLR + CNI*CDI)/CL
      CI=(-CNR*CDI + CNI*CLR)/CL
      STBM(I)= SQRT((1.0+CR*HR(I)-CI*HI(I))**2 + (CR*HI(I)+CI*HR(I))**2)
      IF(STBM(I).GT.SML1)GO TO 288
      SML1= STBM(I)
  288 CONTINUE
      IF(STBM(I).GT.SBC1)GO TO 290
      SBC1= STBM(I)
  290 CONTINUE
      IF(SBC2.GT.SBC1)GO TO 298
      SBC2= SBC1
C COMMENT 13
      IBEST= LOX
      DO 291 I=1,LNC
  291 BCOMN(I)= COMN(I)
      DO 292 I=1,LDC
  292 BCOMD(I)= COMD(I)
  298 CONTINUE
C COMMENT 14
      DO 320 I=1,NM
      PORM= 1.0
  310 IF((STBM(I)-RG(I))*PORM)350,320,320
  320 CONTINUE
      WRITE(6,330)
  330 FORMAT('0',15X,'***** ALL SYSTEM REQUIREMENTS HAVE BEEN MET *****'
     1)
  340 CONTINUE
      CALL OTPT1(STBM,X,NM,COMN,COMD,ICP,L2,0,LOX,PRX,PRY)
      WRITE(6,341) IBEST
  341 FORMAT('0',15X,'***** BEST COMPENSATOR *****'//21X,'OCCURRED ON ST
     /EP',I4)
      WRITE(6,70) (BCOMN(I),I=1,LNC)
      WRITE(6,70) (BCOMD(I),I=1,LDC)
      WRITE(6,345) SBC2
  345 FORMAT('0',21X,'SMALLEST STABILITY MARGIN FOR THE BEST COMPENSATOR
```

```
     / =',F10.8)
       STOP
  350 CONTINUE
C COMMENT 15
       IF(LOX.GE.KQUIT)GO TO 340
       IF(SML1-SML2)360,380,370
  360 STEP= STEP/2.0
       IF(STEP.LT.1.0E-07)GO TO 380
       GO TO 450
  370 STEP= 2.0 * STEP
       SML2= SML1
       IF(STEP.GT.STPMAX)STEP= STPMAX
C COMMENT 16
       IF(LOX.GE.KQUIT-5)KHOP=0
       IF(KHOP.GT.0)GO TO 410
       KHOP=25
       WRITE(6,400) STEP
  400 FORMAT('0', 15X, 'PRESENT STEP SIZE =',F10.7)
       CALL OTPT1(STBM,X,NP,COMM,COMD,IOP,L2,0,LOX,PRX,PRY)
       GO TO 420
  410 KHOP= KHOP - 1
  420 CONTINUE
C COMMENT 17
       CALL PARCLT(COMM,COMD,PR,H1,X,NP,PRX,PRY,K2,K3,STBM,SML1,0,
     /    RQ,R,IOP)
C COMMENT 18
       KRESET=L1+L2+1
       IF(KLOCK.NE.0)KRESET= L1 + L2 + 1
       KRESET= KRESET+1
       IF(KRESET.LT.L1+L2+1)GO TO 280
       KRESET=0
       DO 270 I=1,L2
       PX(I)=0.0
  270 PY(I)=0.0
  280 CONTINUE
       PARSQ=0.0
       DO 430 I=IOP,L2
  430 PARSQ= PARSQ + PRX(I)**2 + PRY(I)**2
       BETA= PARSQ/PSQL
       PSQ= 0.0
C COMMENT 19
       DO 438 I=IOP,LNC
       PX(I)= -PRX(I) + BETA * PX(I)
  438 PSQ= PSQ + PX(I)**2
       DO 440 I=IOP,LDC
       PY(I)= -PRY(I) + BETA * PY(I)
  440 PSQ= PSQ + PY(I)**2
       PMG= SQRT(PSQ)
       PSQL= PSQ
       DEL= STEP/PMG
       DO 462 I=IOP,LNC
  462 COMM(I)= COMM(I)
```

```
      DO 464 I=IOP,LLC
  464 CCTL(I)= COMD(I)
      GO TO 465
  450 DEL= DEL/2.0
      DO 467 I=IOP,LAC
  467 COMN(I)= CCIN(I)
      DO 468 I=IOP,LDC
  468 COML(I)= CCTL(I)
  465 CONTINUE
C COMMENT 20
      DO 470 I=IOP,LAC
      COMN(I)= COMN(I) + DEL * PA(I)
  470 IF(COMN(I)*GAIN.LT.0.0)COMN(I)=0.0
      DO 490 I=IOP,LDC
      COML(I)= COMD(I) + DEL * PY(I)
  490 IF(COMD(I).LT.0.0)COMD(I)=0.0
  500 CONTINUE
      LOX= LOX + 1
      GO TO 200
      END
```

Program PARCLT

The purpose of this program is to calculate the first partial derivatives of the stability margins with respect to the compensator coefficients. The theory supporting the program is given in the following discussion. In [1] a stability margin is defined as a $\min[|1 + G_c(j\omega)G_o(j\omega)|^2]$ in which $G_c(j\omega)$ and $G_o(j\omega)$ represent the compensator and open loop transfer functions, respectively, evaluated at the radian frequency ω. Assume that a stability margin occurs at $\omega = \omega_1$. Then

$$G_o(j\omega_1) = a + jb \tag{1}$$

If the numerator and the denominators of the compensator are of order K and L respectively, then they can be written as

$$N(j\omega_1) = \sum_{i=0}^{K} X_i \cdot (j\omega_1)^i \tag{2}$$

and

$$D(j\omega_1) = \sum_{i=0}^{L} Y_i \cdot (j\omega_1)^i \tag{3}$$

where N means numerator, D means denominator, X_i is the i^{th} numerator coefficient, and Y_i is the i^{th} denominator coefficient. Next, dividing (2) and (3) into their real and imaginary parts the results are

$$N_R(j\omega_1) = \sum_{i=0}^{N} X_{2i}(-1)^i(\omega_1)^{2i} \tag{4}$$

$$N_I(j\omega_1) = \sum_{i=0}^{P} X_{2i+1}(-1)^i(\omega_1)^{2i+1} \tag{5}$$

$$D_R(j\omega_1) = \sum_{i=0}^{Q} Y_{2i}(-1)^i(\omega_1)^{2i} \tag{6}$$

$$D_I(j\omega_1) = \sum_{i=0}^{R} Y_{2i+1}(-1)^i(\omega_1)^{2i+1} \tag{7}$$

where the number of odd terms in N(s) is N+1; the number of even terms in N(s) is P+1; the number of odd terms in D(s) is Q+1; the number of even terms in D(s) is R+1. After some simplification the stability margin (SM) at ω_1 becomes

$$SM = \frac{(D_R + aN_R - bN_I)^2 + (D_I + bN_R + aN_I)^2}{D_R^2 + D_I^2} \tag{8}$$

where the functional notation has been dropped for convenience. Rather than maximize (8) the compensator growing method chosen was the minimization of the reciprocal of (8).[1] The reciprocal of (8) becomes

$$H = \frac{D_R^2 + D_I^2}{(D_R + aN_R + bN_I)^2 + (D_I + bN_R + aN_I)^2} \tag{9}$$

The desired partial derivatives become:

$$\frac{\partial H}{\partial X_{2i}} = \frac{-2H[(D_R + aN_R + bN_I)a + (D_I + bN_R + aN_I)b](-1)^i(\omega_1)^{2i}}{(D_R + aN_R + bN_I)^2 + (D_I + bN_R + aN_I)^2} \tag{10}$$

$$\frac{\partial H}{\partial X_{2i+1}} = \frac{-2H[(D_R + aN_R + bN_I)b + (D_I + bN_R + aN_I)a](-1)^i(\omega_1)^{2i+1}}{(D_R + aN_R + bN_I)^2 + (D_I + bN_R + aN_I)^2} \tag{11}$$

$$\frac{\partial H}{\partial Y_{2j}} = [H/D_R^2 + D_I^2)]2D_R(-1)^j(\omega_1)^{2j}$$

$$\frac{2(D_R^2 + D_I^2)[(D_R + aN_R + bN_I)(-1)^j(\omega_1)^{2j}]}{[(D_R^2 + D_I^2)/H]^2} \tag{12}$$

$$\frac{\partial H}{\partial Y_{2j+1}} = [H/(D_R^2 + D_I^2)]2D_R(-1)^j(\omega_1)^{2j+1}$$

$$\frac{2(D_R^2 + D_I^2)[(D_I + bN_R + aN_I)(-1)^j(\omega_1)^{2j+1}}{[(D_R^2 + D_I^2)/H]^2} \qquad (13)$$

$i = 0, 1, 2, \ldots, K$

$j = 0, 1, 2, \ldots, L^*$

From the preceding equations the partials of a stability margin can be calculated. Equations (4), (5), (6), (7), (9), (10), (11), (12), and (13) are programed in PARCLT.

This program calls one sub-program, VECTOR. VECTOR does not affect the inter workings of PARCLT. VECTOR takes the partial vectors from several stability margins and combines them to form two vectors. These two vectors, one for the numerator and one for the denominator, are used to change the compensator coefficients so as to increase relative stability.

---

*If the numerator or denominator polynomial is even ordered, then the number of odd term partials should only be K-1 or L-2 respectively.

```
      SUBROUTINE PARCLI(X,Y,GR,GI,OMEGA,NFREQ,PFX,PFY,N,M,STBM,SMIN,
     /   NTOTAL,IER)
C
C  PROGRAM FOR CALCULATING THE CHANGE OF A FREQUENCY RESPONSE WITH
C  RESPECT TO A COMPENSATION COEFFICIENTS
C
C  DEFINITIONS OF I/O VARIABLES
C
C   X      -VECTOR OF THE NUMERATOR COEFFICIENTS OF A COMPENSATOR
C   Y      -VECTOR OF THE DENOMINATOR COEFFICIENTS OF A COMPENSATOR
C   GR     -REAL PART OF OPEN LOOP TRANSFER FUNCTION EVALUATED AT ONE OF
C               THE FREQUENCIES OF INTEREST
C   GI     -CORRESPONDING VECTOR OF IMAGINARY PARTS OF OPEN LOOP TRANFER
C               FUNCTION
C   OMEGA  -FREQUENCIES OF INTEREST
C   NFREQ  -NUMBER OF FREQUENCIES UNDER CONSIDERATION
C   PFX    -SUM OF PARTIALS AT FREQUENCIES CONSIDERED - NUMERATOR
C   PFY    -SUM OF PARTIALS AT FREQUENCIES CONSIDERED - DENOMINATOR
C   N      -ORDER OF NUMERATOR COEFFICIENTS
C   M      -ORDER OF DENOMINATOR COEFFICIENTS
C
      DIMENSION C(10), D(10), E(10), GR(30), GI(30), OMEGA(30), Y(10),
     1   X(10), PFY1(10), PFX1(10), PFY(1), PFX(1), STBM(1), RG(1)
      DIMENSION G(30,30)
      M1= M+1
      DO 10 I=1,M1
      PFY(I)=0.0
10    PFX(I)=0.0
      N1=N+1
      DO 140 J=1,NFREQ
      FREQ=1.0
      KSKIP=1
      DO 40 I=1,N1
      FCL1=(-1.0)**((I+1)/2)
      FCL2=(-1.0)**((I+2)/2)
      IF(KSKIP-1)20,20,30
20    C(I)=-GR(J)*FREQ*FCL2
      D(I)=-GI(J)*FREQ*FCL1
      KSKIP=2
      GO TO 40
30    C(I)=-GI(J)*FREQ*FCL2
      D(I)=-GR(J)*FREQ*FCL1
      KSKIP=1
40    FREQ= FREQ*OMEGA(J)
      FREQ= 1.0
      DO 50 I=1,M1
      EMUL= (-1.0)**((I+1)/2)
      E(I)= -FREQ * EMUL
50    FREQ= FREQ * OMEGA(J)
      FLA1=0.0
      FLA2=0.0
      DO 60 I=1,N1
```

```
      FDA1=FDA1+C(I)*X(I)
   60 FDA2=FDA2+C(I)*X(I)
      FN2=0.0
      K1= 2 * ((N+1)/2)
      DO 70 I=2,K1,2
   70 FN2=FN2+E(I)*Y(I)
      FN1=0.0
      KL= 2 * ((N+2)/2) - 1
      DO 80 I=1,KL,2
   80 FN1=FN1+E(I)*Y(I)
      FD1=FN1+FDA1
      FD2=FN2+FDA2
      FD=FD1**2+FD2**2
      FN=FN1**2+FN2**2
      FYE=(FD*FN1-FN*FD1)/(FD**2)
      FYO=(FD*FN2-FN*FD2)/(FD**2)
      FX1=-FN*FD1/(FD**2)
      FX2=-FN*FD2/(FD**2)
      G(1,1)= 0.0
      DO 90 I=1,KL,2
      PFY1(I)=FYE*E(I)
      G(1,I+N1)= PFY1(I)
   90 CONTINUE
      DO 100 I=2,K1,2
      PFY1(I)=FYO*E(I)
      G(1,I+N1)= PFY1(I)
  100 CONTINUE
      IF(IOP.EQ.2)G(1,N1+1)=0.0
      DO 110 I=2,M1
      PFX1(I)=FX1*C(I)+FX2*D(I)
      G(1,I)= PFX1(I)
  110 CONTINUE
  140 CONTINUE
      CALL VECTOR(G,NF,EG,N1,M1,PFX,PFY,STM,RG)
      RETURN
      END
```

Program VECTOR

The purpose of VECTOR is to take the gradient vectors (the partials of the numerator and the denominator are placed in a single vector with the numerator partials coming first) of several stability margins and combine the more pertinent ones by the weighted gradient sum method. The weighting factor is chosen so that the gradients corresponding to the points of smallest stability margins are weighted more heavily. Also, any gradient whose weight is less than a certain level is disregarded in the sum.

One of the major ideas in the whole program is to increase the smallest stability margin on every iteration. It has been assumed that in doing this the relative stability of the system is being increased. It was reasoned that the probability of increasing the smallest stability margin is very low if the calculated vector for changing the compensator coefficient does not possess a component in the direction of the gradient of the smallest stability margin. Thus the dot product of the gradient sum vector and the gradient corresponding to the smallest stability margin are checked. If this dot product falls below a certain prescribed level the gradient sum vector is rejected and an alternate approach is sought to calculate the coefficient change vectors. (This is accomplished by the program, DIRVEC.) Once this happens the gradient sum method is disregarded on all higher iterations and the alternate approach becomes the sole means of calculating the change vector.

On the other hand, the programmer may specify that the gradient sum method never be used. This is accomplished by specifying the variable KLOCK as 1 in the input of the main program. This is very advantageous

where the program is being used for compensator improvement, rather than for compensator growing. The reason for this is that the alternate method calculates a vector for increasing all stability margins below the desired level. This cannot be said in regard to the gradient sum. However, the gradient sum possesses the advantage of easily being calculated.

```
      SUBROUTINE VECTOR(G,IM,M1,M1,PFX,PFY,STEM,RG)
C  GRADIENT SUM PROGRAM
C
C  DEFINITIONS OF I/O VARIABLES
C
C  G      -MATRIX WHOSE ROWS CONTAIN NECESSARY GRADIENT VECTORS
C  IM     -NUMBER OF STABILITY MARGINS TO BE INCREASED (NUMBER OF ROWS
C           IN G)
C  M1     -NUMBER OF NUMERATOR COEFFICIENTS OF THE COMPENSATOR
C  M1     -NUMBER OF DENOMINATOR COEFFICIENTS OF THE COMPENSATOR
C  PFX    -NUMERATOR CHANGE VECTOR
C  PFY    -DENOMINATOR CHANGE VECTOR
C  STEM   -VECTOR OF STABILITY MARGINS
C  RG     -VECTOR OF REQUIRED STABILITY MARGINS
C
      DIMENSION G(30,30), GI(30,30), DV(30), STEM(1), RG(1), PFX(1),
     /   PFY(1), WEIGHT(30)
      DIMENSION DOTSP(30)
      DIMENSION G(30)
      COMMON/JIM/KLOCK
      KPARC= M1 + M1
      WMAX=0.0
      DO 100 I=1,IM
      WEIGHT(I)= 1.0/STEM(I) - 1.0/RG(I)
      G(I)= WEIGHT(I)
      IF(WEIGHT(I).GT.0.0)WEIGHT(I)=WEIGHT(I)
      IF(WEIGHT(I).LT.WMAX)GO TO 100
      WMAX= WEIGHT(I)
      IT=I
100   CONTINUE
      K=0
      DO 200 J=1,IM
      IF(WEIGHT(J).LT.0.10*WMAX)GO TO 200
      K=K+1
      WEIGHT(K)= WEIGHT(J)
      G(K)=1.0
      DO 150 I=1,KPARC
150   GI(K,I)= G(J,I)
200   CONTINUE
      DO 300 J=1,KPARC
      SUM=0.0
      DO 250 I=1,K
250   SUM= SUM + GI(I,J) * WEIGHT(I)
300   DV(J)= SUM
      DO 470 J=1,IM
      SUM=0.0
      DO 465 I=1,KPARC
465   SUM= SUM + G(J,I) * DV(I)
470   DOTSP(J)= SUM
      IF(KLOCK.EQ.1)GO TO 485
      IF(DOTSP(IT).GT.0.0)GO TO 490
      KLOCK=1
```

```
465 CALL LINVEC(GI,K,I,J,1,IFA,IFY,G)
    GO TO 700
490 CONTINUE
    DO 500 I=1,N1
500 IFX(I)= DV(I)
    DO 600 I=1,N1
600 IFY(I)= DV(I+N1)
700 RETURN
    END
```

Program DIRVEC

The function of this program is to find a vector $\vec{D}$ which is calculated as

$$\vec{D} = a_1 \nabla G_1 + a_2 \nabla G_2 + \ldots + a_m \nabla G_m \qquad (14)$$

in which ∇G_i is the i^{th} gradient vector of the i^{th} stability margin chosen for improvement and m is the number of stability margins to be improved. The vector $(a_1, a_2, \ldots, a_m)$ is a vector of weights that are to be determined in order that

$$D \cdot \nabla G_i = q_i \qquad i = 1, 2, \ldots, m \qquad (15)$$

where the q_i's are selected positive constants.* Using (14) and (15) the following matrix equation is obtained

$$\begin{bmatrix} \nabla G_1 \cdot \nabla G_1 & \nabla G_1 \cdot \nabla G_2 & \ldots & \nabla G_1 \cdot \nabla G_m \\ \nabla G_2 \cdot \nabla G_1 & \nabla G_2 \cdot \nabla G_2 & \ldots & \nabla G_2 \cdot \nabla G_m \\ \vdots & & & \\ \nabla G_m \cdot \nabla G_1 & \nabla G_m \cdot \nabla G_2 & \ldots & \nabla G_m \cdot \nabla G_m \end{bmatrix} \begin{bmatrix} a_1 \\ a_1 \\ \vdots \\ a_m \end{bmatrix} = \begin{bmatrix} q_1 \\ q_1 \\ \vdots \\ q_m \end{bmatrix} \qquad (16)$$

If the gradients form a basis for m-space, then the matrix formed by the dot products of the gradients, $[\nabla G \cdot \nabla G]$, is non-singular. Hence the weighting factors are calculated as

$$[a_1, a_2, \ldots, a_m]^T = [\nabla G \cdot \nabla G]^{-1} [q_1, q_2, \ldots, q_m] \quad . \qquad (17)$$

Thus it is seen that it is impossible to calculate a change vector (or change vectors if the numerator and denominator are considered separately)

*In this program they were selected arbitrarily as 1.

for the compensator coefficients which can result in improvement of all the chosen stability margins. It should be obvious that this program is very applicable in compensator improvement.

It should be pointed out that if the number of chosen stability margins (those chosen for improvement) becomes greater than the number of compensator coefficients that are being varied, this program will not be applicable. The reason is that in this case $[\nabla G \cdot \nabla G]^{-1}$ will not exist. When such a condition occurs the complete program is automatically terminated.

```
      SUBROUTINE DIRVEC(G,NM,N1,M1,PFX,PFY,WEIGHT)
C  DIRECTIONAL  VECTOR PROGRAM
C
C  DEFINITIONS OF I/O VARIABLES
C
C  G      -MATRIX WHOSE ROWS CONTAIN THE GRADIENT VECTORS OF THOSE
C           STABILITY MARGINS ONLY CONSIDERED PERTINENT
C  NM     -NUMBER OF STABILITY MARGINS CONSIDERED PERTINENT
C  N1     -NUMBER OF NUMERATOR COEFFICIENTS OF THE COMPENSATOR
C  M1     -NUMBER OF DENOMINATOR COEFFICIENTS OF THE COMPENSATOR
C  PFX    -NUMERATOR CHANGE VECTOR
C  PFY    -DENOMINATOR CHANGE VECTOR
C  WEIGHT-WEIGHTING FACTOR VECTOR
C
      DIMENSION G(30,30), A(30,30), PFX(1), PFY(1), WEIGHT(1),
     /    AI(30,30), X(30), DV(30)
      DIMENSION DUISP(30)
      DIMENSION COMM(50), COML(50)
      COMMON/COL/COMM,COML,GAIN
      KPARC= N1 + M1
      KRE=0
  100 CONTINUE
      DO 200 K=1,NM
      DO 200 J=K,NM
      SUM= 0.0
      DO 150 I=1,KPARC
  150 SUM= SUM + G(J,I) * G(K,I)
      A(J,K)= SUM
      A(K,J)= SUM
  200 CONTINUE
      IF(NM.GT.1)GO TO 230
      AI(1,1)= 1.0/A(1,1)
      X(1)= WEIGHT(1) * AI(1,1)
      GO TO 310
  230 CONTINUE
      CALL MATINV(A,NM,AI,IER)
      IF(IER.EQ.0)GO TO 300
      WRITE(6,250)
  250 FORMAT('0',15X,'THE PARTIALS ARE NOT LINEARLY INDEPENDENT.  THUS T
     /HE PROGRAM IS TERMINATED.')
      STOP
  300 CALL MATMUL(NM,AI,NM,WEIGHT,1,X)
  310 CONTINUE
      DO 450 I=1,KPARC
      SUM= 0.0
      DO 400 J=1,NM
  400 SUM= SUM + G(J,I) * X(J)
  450 DV(I)= SUM
      DO 470 J=1,NM
      SUM=0.0
      DO 465 I=1,KPARC
  465 SUM= SUM + G(J,I) * DV(I)
```

```
470 DCTSP(J)= SUM
    DO 500 I=1,N1
500 PFX(I)= DV(I)
    DO 600 I=1,N1
600 PFY(I)= DV(I+N1)
    IF(KRE.EQ.1)GO TO 650
    IF(GAIN*COND(N1)-1.0E-06.GT.0.0)GO TO 660
    IF(PFX(N1)*GAIN.LT.0.0)GO TO 660
    KRE=1
    DO 650 J=1,NM
650 G(J,N1)=0.0
660 IF(COND(N1)-1.0E-06.GT.0.0)GO TO 680
    IF(PFY(N1).LT.0.0)GO TO 680
    KRE=1
    DO 670 J=1,NM
670 G(J,N1+N1)=0.0
680 IF(KRE.GT.0)GO TO 100
690 RETURN
    END
```

Program SRMINS

The goal of this program is to find the approximate minima of the open loop compensated frequency response of a control system with respect to the -1 + j0 point. It is assumed that the open loop frequency response is accurately described by N descrete points. The compensator transfer function is evaluated at the same N points. From this the compensated open loop frequency response is calculated by multiplying the corresponding points. From this information the program uses a search algorithm to determine the approximate minima of

$$[|1 + G_o(j\omega)G_c(j\omega)|^2] \quad .$$

```
      SUBROUTINE SFMINS(CONN,COND,K2,GR,GI,KPOINT,OMEGA,X,NM,HR,HI)
      DIMENSION CONN(1), COND(1), GR(1), GI(1), OMEGA(1), HR(1), HI(1),
     1    X(1)
C
C   SUBPROGRAM FOR DETERMINING THE MINIMUMS OF THE OPEN LOOP FREQUENCY
C      RESPONSE WITH RESPECT TO THE -1 POINT WHEN GIVEN POINTS ON THE
C      OPEN LOOP FREQUENCY RESPONSE
C
C   DESCRIPTION OF I/O VARIABLES
C      CONN   - VECTOR OF THE NUMERATOR COEFFICIENTS OF THE COMPENSATOR
C      COND   - VECTOR OF THE DENOMINATOR COEFFICIENTS OF THE COMPENSATOR
C      K2     - MAXIMUM COMPENSATOR ORDER
C      GR     - VECTOR OF REAL PARTS OF OPEN LOOP FREQUENCY RESPONSE
C      GI     - VECTOR OF IMAGINARY PARTS OF OPEN LOOP FREQ. RESPONSE
C      KPOINT - NUMBER POINTS OF THE OPEN LOOP FREQ. RESPONSE GIVEN
C      OMEGA  - CORRESPONDING FREQUENCIES OF CHOSEN POINTS
C      X      - VECTOR OF MINIMUMS
C      NM     - NUMBER FREQUENCIES OF INTEREST
C
      ASN1=0.0
      S1=0.0
      DO 50 I=1,KPOINT
      XV= OMEGA(I) * 6.283185
      CALL POLFV(CONN,K2,XV,CNR,CNI)
      CALL POLFV(COND,K2,XV,CDR,CDI)
      CD=CDR**2+CDI**2
      CR=(CNR*CDR+CNI*CDI)/CD
      CI=(-CNR*CDI+CNI*CDR)/CD
      S2=(1.0+CR*GR(I)-CI*GI(I))**2+(CR*GI(I)+CI*GR(I))**2
      ASN2=S2-S1
      IF(ASN2)10,50,10
10    IF(ASN1*ASN2)20,40,40
20    IF(ASN1)30,40,40
30    NM=NM+1
      I1= I - 1
      HR(NM)= GR(I1)
      HI(NM)= GI(I1)
      X(NM)= OMEGA(I1) * 6.283185
40    S1=S2
      ASN1= ASN2
50    CONTINUE
      RETURN
      END
```

Non-Specialized Program

```
      SUBROUTINE OUTPT1(STEP,X,NM,CONN,COND,ICP,L2,KTCT,LOX,PRX,PRY)
      DIMENSION STBM(1), CONN(1), COND(1), PRX(1), PRY(1), X(1)
C
C   SUBPROGRAM FOR THE OUTPUT OF INFORMATION CALCULATED
C
      WRITE(6,10) LOX
   10 FORMAT('0',25X,'ITERATION NO. ',I4)
      DO 110 I=1,NM
      FREQ= X(I)
      IF(I-KTCT)50,50,70
   50 WRITE(6,60)
   60 FORMAT('0',25X,'ATTENUATED FREQUENCY INFORMATION')
      GO TO 90
   70 WRITE(6,80)
   80 FORMAT('0',25X,'RELATIVE STABILITY INFORMATION')
   90 CONTINUE
      WRITE(6,100) I, STBM(1), FREQ
  100 FORMAT('0',15X,'STABILITY RADIUS NO.',I2,'=',E10.3,5X,'FREQUENCY='
     1F10.5)
  110 CONTINUE
      WRITE(6,130) (PRX(I),I=ICP,L2)
      WRITE(6,120) (PRY(I),I=ICP,L2)
  120 FORMAT('0',,'PARTIALS WITH RESPECT TO Y',5E10.3)
  130 FORMAT('0',,'PARTIALS WITH RESPECT TO X',5E10.3)
      WRITE(6,140)
  140 FORMAT('0',15X,'COMPENSATOR COEFFICIENTS')
      WRITE(6,150) (CONN(I),I=1,L2)
      WRITE(6,150) (COND(I),I=1,L2)
  150 FORMAT('0',5X,9F10.5)
      RETURN
      END


      SUBROUTINE MATINV(X,N,Y,IER)
      DIMENSION X(30,30), Y(30,30)
C
C     SUBROUTINE FOR FINDING AN INVERSE OF A MATRIX
C
C     X = MATRIX FOR WHICH INVERSE IS TO BE TAKEN
C     N = DIMENSION OF SQUARE MATRIX X
C     Y = INVERSE OF X
C
C     IER - ERROR CODE
C        IER= 0 - NO ERROR EXISTS
C        IER= 2 - MATRIX DOES NOT POSSESS AN INVERSE
C
      ILR= 0
      N2=N + N
```

```
      N1= N + 1
      DO 30 I=1,N
      DO 30 K=N1,N2
      L= K - N
      IF(L.NE.I)X(I,K)=0.0
      IF(L.EQ.I)X(I,K)=1.0
30    CONTINUE
      IF(ABS(X(I,1)).LT.1.0E-25)GO TO 42
      DO 40 I=1,N
      SCHM=X(I,1)
      DO 35 K=1,N2
35    X(I,K)=X(I,K)/SCHM
      DO 40 L=1,N
      IF(L.EQ.I)GO TO 40
      SLOP= X(L,I)
      DO 39 K=1,N2
      X(L,K)= X(L,K) - SLOP    * X(I,K)
39    CONTINUE
40    CONTINUE
      DO 41 I=1,N
      DO 41 K=N1,N2
      L=K-N
41    Y(I,L)= X(I,K)
      GO TO 43
42    IER=2
43    RETURN
      END


      SUBROUTINE MATMUL(NR,A,N,B,NC,X)
      DIMENSION A(30,30),B(1), X(1)
C
C     MATRIX MULTIPLICATION
C
C     MULTIPLIES (A) * (B)
C     A IS AN NR X N
C     B IS AN N X NC
C     X IS AN NR X NC
C
      DO 4 I=1,NR
    4 X(I) = 0.0
      DO 5 I=1,NR
      DO 5 K=1,NC
      DO 5 J=1,N
    5 X(I)= X(I) + A(I,J) * B(J)
      RETURN
      END


      SUBROUTINE POLFV(FW,K,X,FREAL,FIMAG)
C  PROGRAM FOR EVALUATING A POLYNOMIAL AT AN IMAGINARY FREQUENCY
C
```

```
C  DEFINITIONS OF I/O VARIABLES
C
C   FN      -VECTOR POLYNOMIAL COEFFICIENTS
C   K       -ORDER OF POLYNOMIAL
C   X       -FREQUENCY TO BE EVALUATED AT
C   FREAL -REAL PART OF F(X)
C   FIMAG -IMAGINARY OF F(X)
      DIMENSION FN(1)
      KEVEN=(K+2)/2
      KODD=(K+1)/2
      Y=1.0
      FREAL=0.0
      DO 10 I=1,KEVEN
      L=2*I-1
      FREAL= FREAL + FN(L)*Y
   10 Y=-Y*X*X
      Y=X
      FIMAG=0.0
      DO 20 I=1,KODD
      L=2*I
      FIMAG= FIMAG + FN(L)*Y
   20 Y=-Y*X*X
      RETURN
      END
```

```
      SUBROUTINE OPENL(POLN,POLD,K1,COMN,COMD,K2,X,RG,NM,R,KERROR)
C   PROGRAM FOR DETERMINING MINIMUMS WITH RESPECT TO THE -1 + J0.0 IF
C   THE OPEN LOOP TRANSFER FUNCTION IS GIVEN
C
C   DEFINITIONS OF I/O VARIABLES
C
C   POLN   -OPEN LOOP TRANSFER FUNCTION NUMERATOR COEFFICIENTS
C   POLD   -OPEN LOOP TRANSFER FUNCTION DENOMINATOR COEFFICIENTS
C   K1     -MAXIMUM ORDER IN OPEN LOOP TRANSFER FUNCTION
C   COMN   -COMPENSATOR NUMERATOR COEFFICIENTS
C   COMD   -COMPENSATOR DENOMINATOR COEFFICIENTS
C   K2     -MAX. ORDER IN THE COMPENSATOR
C   X      -VECTOR OF MINIMUMS WITH RESPECT TO THE -1 + J0.0 POINT
C   RG     -REQUIRED STABILITY MARGINS
C   NM     -NUMBER OF MINIMUMS
C   R      -REQUIRED STABILITY RADIUS
C   KERROR-ERROR FOR TERMINATING PROGRAM UNDER CERTAIN CONDITIONS
C
      DIMENSION POLN(1), POLD(1), COMN(1), COMD(1),GN1(50), GD1(50),
     1    GDR(50),GNSUM(50),X(1), RG(1), XR(30), FDPD(50), FDDX(50),
     2    FNX(50), FDX(50), DPOL(50), GDSUM(50), GNR(50)
      CALL CALFX(POLN,POLD,K1,COMN,COMD,K2,K3,GN1,GNR,GD1,GDR,GNSUM,
     1    GDSUM,KERROR)
      IF(KERROR)1,0,10,100
   10 CONTINUE
      KO= K3
```

```
      NC= KS
      KS=KS+1
      DO 20 I=1,KS
      LS=I*2-1
      FNX(I)= GNSUM(LS)
20    FDX(I)= GDSUM(LS)
      CALL FCLDRA(FDX,KC,FNX,KC,FDX,KDC,DMCL,IDC)
      KDX= KDC
      KRUN= KDC + 1
      DO 40 I=1,KDX
      IVAR= KRUN + 1 - I
      IF(ABS(FDX(IVAR)/CMAX(FDX,KDC))-1.0E-20)30,30,50
30    KDC= KDC - 1
40    CONTINUE
      GO TO 180
50    CONTINUE
      ISF= KDC + 1
      ASNE= FDX(ISF)/ABS(FDX(ISF))
      DO 60 I=1,ISF
60    FDX(I)= FDX(I)/FDX(ISF)
      CALL REALRT(FDX,KDC,XB,NP,NPRR)
      IF(NPRR)180,60,70
70    IF(NPRR-NR)80,140,100
80    WRITE(6,90)
90    FORMAT('0',25X,  'NO REAL ROOTS EXIST')
      GO TO 180
100   WRITE(6,110)
110   FORMAT('0',25X,'ALL REAL ROOTS WERE NOT FOUND')
      IF(NR)180,180,120
120   WRITE(6,130) (XB(I),I=1,NP)
130   FORMAT('0',5X,5F12.5)
      GO TO 180
140   CONTINUE
      DO 150 I=1,KDC
150   FDX(I)= I*FDX(I+1)*ASNE
      DO 170 I=1,NR
      XV=XB(I)
      CALL POLY(FDX,KDC-1,XV,VAL2)
      VALUE= VAL2
      IF(VALUE)165,170,170
165   NP=NP+1
      X(NP)= SQRT(XB(I))
      RG(NP)= R
170   CONTINUE
      GO TO 190
180   KERROR=100
190   RETURN
      END


      SUBROUTINE CALFX(FDX,FDX,N,DMCL,DMCL,IDC,KDC,GNSUM,GDSUM,GDI,GDX,
     1  GNSUM,GDSUM,KERROR)
```

```
C
C ROUTINE FOR DETERMINING THE FUNCTION F(X).
C
C
C
C DEFINITION OF VARIABLES
C     POLN - OPEN LOOP TRANSFER FUNCTION NUMERATOR POLYNOMIAL
C     POLD - OPEN LOOP TRANSFER FUNCTION DENOMINATOR POLYNOMIAL
C     K1   - ORDER OF POLN AND POLD
C     CONN - COMPENSATOR NUMERATOR POLYNOMIAL
C     COND - COMPENSATOR DENOMINATOR POLYNOMIAL
C     K2   - ORDER OF COMPENSATOR
C     FX   - RESULTANT POLYNOMIAL
C     K3   - ORDER OF RESULTANT POLYNOMIAL
C
      DIMENSION POLN(1), POLD(1), CONN(1), COND(1), GN1(50), GD(50),
     1   GN(50), GDR(1), GDI(1), GNR(1), GNI(1), GDRS(50), GDIS(50),
     2   GNRS(50), GNIS(50), GNSUM(1), GDSUM(1)
      DIMENSION RARRAY(50,50)
      CALL POLMUL(POLN,CONN,K1,K2,GN1)
      CALL POLMUL(POLD,COND,K1,K2,GD)
      L1=K1+K2
      CALL POLADD(GD,GN1,L1,1.0,GN)
      CALL ROUTH(GN,L1,RARRAY,NC,KERROR,NR)
      IF(KERROR)12,30,20
12 WRITE(6,15)
15 FORMAT('0',15X,'AN ELEMENT IN THE FIRST COLUMN IS ZERO')
   GO TO 30
20 WRITE(6,25)
25 FORMAT('0',15X,'A WHOLE ROW IS ZERO')
30 CONTINUE
   LA=NC
   DO 40 I=1,NR
   IF(RARRAY(I,1))37,40,40
37 KERROR=-1
   WRITE(6,38)
38 FORMAT('0',15X,'***** SYSTEM IS UNSTABLE *****')
40 LA=NC-I/2
   IF(KERROR)50,45,50
45 CONTINUE
   CALL PEVODD(GD,L1,GDR,GDI)
   CALL PEVODD(GN,L1,GNR,GNI)
   CALL POLMUL(GDR,GDR,L1,L1,GDRS)
   CALL POLMUL(GDI,GDI,L1,L1,GDIS)
   CALL POLMUL(GNR,GNR,L1,L1,GNRS)
   CALL POLMUL(GNI,GNI,L1,L1,GNIS)
   L2=L1+L1
   CALL POLADD(GDRS,GDIS,L2,1.0,GDSUM)
   CALL POLADD(GNRS,GNIS,L2,1.0,GNSUM)
   K3= L1
50 RETURN
   END
```

```
      FUNCTION CMAX(B,K)
      DIMENSION B(1)
C
C FUNCTION SUB-PROGRAM FOR FINDING THE VALUE OF AN ARRAY WHOSE
C MAGNITUDE IS THE LARGEST.
C
C     I/O VARIABLES
C        B      -ARRAY
C        K      -NUMBER OF ELEMENTS IN B
C        CMAX   - LARGEST ELEMENT IN B ACCORDING TO MAGNITUDE
C
      CMAX= ABS(B(1))
      DO 10 I=1,K
      IF(ABS(B(I))-CMAX)10,10,5
    5 CMAX= ABS(B(I))
   10 CONTINUE
      RETURN
      END


      SUBROUTINE POLDRV(PCLN,J,PCLD,K,DPOL,L,DPOLD,M)
C
C FOR FINDING THE DERIVATIVE OF A RATIONAL POLYNOMIAL
C
C I/O VARIABLES DEFINITION
C     PCLN - VECTOR OF COEFFICIENTS OF POLYNOMIAL IN NUMERATOR
C     J    - ORDER OF PCLN
C     PCLD - VECTOR OF COEFFICIENTS OF POLYNOMIAL IN DENOMINATOR
C     K    - ORDER OF PCLD
C     DPCLN- NUMERATOR OF DERIVATIVE
C     L    - ORDER OF DPCLN
C     DPCLD- DENOMINATOR OF DERIVATIVE
C     M    - ORDER OF DPCLD
C
C ASSUMPTION : INPUT POLYNOMIAL COEFFICIENTS ARE IN DESCENDING ORDER
C
C SUBROUTINES NEED :  PCLMUL - POLYNOMIAL MULTIPLICATION
C                     PCLADD - POLYNOMIAL ADDITION
C
      DIMENSION PCLN(1), PCLD(1), DPCLN(1), DPCLD(1), EPCL(50),FPCL(50)
     1,GPCL(50),HPCL(50)
      DO 10 I=1,J
   10 EPCL(I)= I * PCLN(I+1)
      DO 20 I=1,K
   20 FPCL(I)= I * PCLD(I+1)
      CALL PCLMUL(PCLD,EPCL,K,J-1,GPCL)
      CALL PCLMUL(PCLN,FPCL,J,K-1,HPCL)
      L=K+J-1
      CALL PCLADD(GPCL,PCL,L,-1.0,DPCLN)
      CALL PCLMUL(PCLD,PCLD,K,K,DPCLD)
```

```
      N=K+K
      RETURN
      END


      SUBROUTINE ROUTH(FS,KORDER,RARRAY,K,KERROR,NR)
C
C  SUBROUTINE FOR APPLYING THE ROUTH HURWITZ CRITERION TO A POLYNOMIAL
C
C  I/O VARIABLES
C     FS   - POLYNOMIAL COEFFICIENTS IN DESCENDING ORDER
C     KORDER-ORDER OF THE POLYNOMIAL
C     RARRAY- THE ROUTH HURWITZ ARRAY
C     K      - NUMBER OF COLUMNS IN THE ARRAY
C     NR     - NUMBER ROWS IN THE ARRAY
C
C     KERROR-ERROR CODE
C             +1 DENOTES A WHOLE ROW OF THE ARRAY AS BEING ZERO
C             -1 DENOTES THE FIRST ELEMENT OF A ROW AS BEING ZERO
C
C
      DIMENSION FS(50),RARRAY(50,50), FB(1)
      KT= KORDER + 1
      DO 5 I=1,KT
      L= KT + 1 -I
    5 FS(I)= FB(L)
      K=(KORDER+2)/2
      DO 10 I=1,K1
      DO 10 J=1,K
   10 RARRAY(I,J)=0.0
      NUMO=(KORDER+1)/2
      NUME=(KORDER+2)/2
      IF (NUME-NUMO)12,12,14
   12 I1=2
      I2=1
      GO TO 16
   14 I1=1
      I2=2
   16 CONTINUE
      DO 20 J=1,NUME
      JX=2*J+1-I2
   20 RARRAY(I1,J)= FS(JX)
      DO 30 J=1,NUMO
      JX= 2*J+1-I1
   30 RARRAY(I2,J)= FS(JX)
      L2=(KORDER+2)/2
      LG=KORDER-1
      KERROR=0
      DO 70 I=1,LG
      DO 60 J=2,L2
      M=J-1
   60 RARRAY(I+2,M)= (RARRAY(I+1,1)*RARRAY(I,J)-RARRAY(I,1)*RARRAY(I+1,J
```

```
   1))/RARRAY(I+1,1)
    IF(RARRAY(I+2,1))70,62,70
 62 KERROR=-1
    DO 65 J=2,L2
    IF(RARRAY(I+2,J))80,65,80
 65 CONTINUE
    KERROR=+1
    GO TO 80
 70 L2= K - I/2
 80 NR=I+2
    RETURN
    END


    SUBROUTINE POLADD(POL1,POL2,K,R,POLA)

C THIS PROGRAM WILL FIND THE WEIGHTED SUM OF TWO POLYNOMIALS.

    DIMENSION POL1(1), POL2(1), POLA(1)
    N=K+1
    DO 10 I=1,N
 10 POLA(I)= POL1(I) + R * POL2(I)
    RETURN
    END


    SUBROUTINE PEVOD(POL,N,POLEV,POLOD)
    DIMENSION POL(1), POLEV(1), POLOD(1)

C THIS PROGRAM DIVIDES A POLYNOMIAL INTO ITS ODD AND EVEN PARTS. IN THE
C ODD AND EVEN PARTS EVERY OTHER NON ZERO ELEMENTS IS MULTIPLIED BY -1.

    N=N+1
    DO 10 I=1,N
    J=(I+3)/2
    POLEV(I)= 0.5 * (POL(I) +(-1.0)**(I+1) * POL(I)) * (-1.0)**J
 10 POLOD(I)= 0.5 * (POL(I) +(-1.0)**(I+2) * POL(I)) * (-1.0)**J
    RETURN
    END


    SUBROUTINE REALRT(FX,KORDER,ROOTS,NROOT,NPD)

C PROGRAM FOR FINDING THE POSITIVE REAL ROOTS OF A POLYNOMIAL

C THEORY USED IN PROGRAM - STURM METHOD OF FINDING REAL ROOTS

C DEFINITION OF I/O VARIABLES
C     FX      -VECTOR OF COEFFICIENTS IN ASCENDING ORDER OF THE
C                    POLYNOMIAL FOR WHICH THE ROOTS ARE TO BE FOUND
C     KORDER  -ORDER OF FX
C     ROOTS   -POSITIVE REAL ROOTS OF FX
```

```
C      NRCCT   -NUMBER OF POSITIVE REAL ROOTS FOUND
C
       DIMENSION FX(1), FX(50), ROOTS(1), STFT(50,50), X(5), NSC(5)
       DOUBLE PRECISION STFT, FX, BMAX, ANORM
       NRCCT=0
       NPC=0
C
C GENERATE STURM FUNCTIONS
C
       CALL STURMF(FX,KORDER,STFT,KFX)
       KL= KFX
       DO 5 I=1,KFX
       DO 3 J=1,KL
     3 FX(J)= STFT(1,J)
       ANORM= BMAX(FX,KL)
       DO 4 J=1,KL
     4 STFT(I,J) = STFT(I,J)/ANORM
     5 KL= KL-1
C
C DETERMINE AN UPPER BOUND ON P.R. ROOTS
C
       X(4)= 1.0E02
       IF(X(4))7,7,28
     7 CONTINUE
       MPR=0
       G=0.0
       NC=KORDER+1
       DO 8 I=1,NC
     8 FX(I)= FX(I)/FX(NC)
       DO 25 I=1,KORDER
       M=KORDER+1-I
       IF(FX(M))19,21,21
    19 IF(MPR)20,20,21
    20 MPR=I
    21 IF(FX(M)-G)22,25,25
    22 G=FX(M)
    25 CONTINUE
       IF(G)26,80,26
    26 X(4)= 1.0 + (ABS(C/FX(KORDER+1)))**(1.0/MPR)
C
C DETERMINE ROOTS
C
    28 CONTINUE
       X(1)= 0.0
       X(3)= X(4)
       X(2)=X(1)
       KSKIP=0
       KSTOP=0
    29 CONTINUE
       KSTOP=KSTOP+1
       IF(KSTOP-100)30,80,80
    30 CONTINUE
```

```
      NSC(1)=0
      NSC(2)=0
      DO 40 J=1,2
      XV=X(J+1)
      KL=KFX
      VAL2=0.0
      DO 40 L=1,KFX
      DO 32 I=1,KL
   32 FX(I)= STFT(L,I)
      KL=KL-1
      CALL POLV1(FX,KL,XV,VAL1)
      VAL1= VAL1/ABS(VAL1)
      IF(VAL1)34,40,34
   34 IF(VAL2*VAL1)36,38,38
   36 NSC(J)= NSC(J)+1
   38 VAL2= VAL1
   40 CONTINUE
      IF(NPD)80,42,50
   42 NPD=IABS(NSC(1)-NSC(2))
      IF(NPD)80,80,50
   50 IF(IABS(NSC(1)-NSC(2))-1)65,52,56
   52 CONTINUE
      X1= X(2)
      X2= X(3)
      X(4)= X(2)
      X(3)= X3(X1,X2,FX,KORDER)
      GO TO 75
   56 X(1)=X(2)
      X(2)= (X(3)+X(2))/2.0
      GO TO 70
   65 X(3)=X(2)
      X(2)= (X(2)+X(1))/2.0
      GO TO 29
   70 IF(ABS(X(3)-X(2))-1.0E-02)75,29,29
   75 NROOT=NROOT+1
      ROOTS(NROOT)= X(3)
      IF(NROOT-NPD)28,80,80
   80 RETURN
      END


      FUNCTION X3(X1,X2,FX,K)
      DIMENSION FX(1)
      XM= X2
      Y1= X1
      Y2= X2
      CALL POLV(FX,K,Y1,G1)
      CALL POLV(FX,K,Y2,G2)
      IF(G1)20,10,10
   10 F1= G1
      F2= G2
      GO TO 30
```

```
   20 X1= Y2
      X2= Y1
      F2= G1
      F1= G2
   30 CONTINUE
      X3= (X2+X1)/2.0
      IF(ABS(X3-X4)-1.E-04)60,60,35
   35 CALL POLV(PX,K,X3,F3)
      IF(F3)50,60,40
   40 X1= X3
      F1= F3
      X4= X1
      GO TO 30
   50 X2= X3
      F2= F3
      X4= X2
      GO TO 30
   60 RETURN
      END


      DOUBLE PRECISION FUNCTION BMAX(B,K)
      DIMENSION B(1)
      DOUBLE PRECISION B
C
C FUNCTION SUB-PROGRAM FOR FINDING THE VALUE OF AND ARRAY WHOSE
C MAGNITUDE IS THE LARGEST.
C
C      I/O VARIABLES
C         B      -ARRAY
C         K      -NUMBER OF ELEMENTS IN B
C         BMAX   - LARGEST ELEMENT IN B ACCORDING TO MAGNITUDE
C
C   ASSUMPTION- B AND BMAX ARE DOUBLE PRECISION QUANTITIES
C
      BMAX= DABS(B(1))
      DO 10 I=1,K
      IF(DABS(B(I))-BMAX)10,10,5
    5 BMAX= DABS(B(I))
   10 CONTINUE
      RETURN
      END


      SUBROUTINE POLV1(PX,K,X,VAL1)
      DIMENSION PX(1), FX(50)
      DOUBLE PRECISION PX, VAL, FX
      KC=K+1
      VAL=0.0
      DO 10 I=1,KC
      FX(I)= PX(I)
   10 VAL= VAL*X + FX(I)
```

```
      VAL1= VAL
      RETURN
      END


      SUBROUTINE STURMF(FX,K,STFT,KFX)
      DIMENSION FX(1), STFT(50,50), PF1(50), PF2(50), RPF(50)
      DOUBLE PRECISION STFT, PF1, PF2, RPF
C
C         PURPOSE OF PROGRAM - GENERATE STURM FUNCTIONS OF A POLYNOMIAL
C
C   I/O VARIABLES DEFINITION
C     FX          - VECTOR OF COEFFICIENTS OF POLYNOMIAL FROM WHICH THE
C                   STURM FUNCTIONS ARE TO BE GENERATED.
C     K           - ORDER OF FX
C     STFT        - ARRAY OF COEFFICIENTS OF STURM FUNCTIONS
C     KFX         - NUMBER OF STURM FUNCTIONS
C
      KFX=K+1
      DO 10 I=1,KFX
      NFX= KFX+1-I
      PF1(I)= FX(NFX)
   10 STFT(1,I)= PF1(I)
      DO 20 I=1,K
      PF2(I)= (K+1-I)*PF1(I)
   20 STFT(2,I)= PF2(I)
      L=K-1
      DO 80 J=3,KFX
      CALL POLDIV(PF2,PF1,L,L+1,RPF)
      DO 40 I=1,L
   40 STFT(J,I)= -PF1(I)
      KLAP=L+1
      DO 50 I=1,KLAP
   50 PF1(I)= PF2(I)
      DO 60 I=1,L
   60 PF2(I)= STFT(J,I)
      IF(STFT(J,1))80,90,80
   80 L=L-1
   90 KFX=J
      RETURN
      END


      SUBROUTINE  POLDIV(DX,AX,N,M,RPC)
      DIMENSION AX(1), DX(1), RPC(1), SX(50)
C
C     FOR DOUBLE PRECISION REMOVE C FROM FIRST COLUMN OF NEXT CARD.
      DOUBLE PRECISION AX, SX, RPC, DX
C
C                     POLYNOMIAL DIVISION
C
C     NTH ORDER POLYNOMIAL WHOSE VECTOR OF COEFFICIENTS IS AX IS DIVIDED
```

```
C      BY AN NTH ORDER POLYNOMIAL MOST VECTOR OF COEFFICIENTS IS EX.
C
C      ASSUMPTION-M IS GREATER THAN N.
C
C
C   DEFINITIONS OF VARIABLES
C     DX        - DIVISOR
C     EX        - DIVIDEND
C     N         - ORDER OF DX
C     M         - ORDER OF AX
C     RPC       - QUOTIENT OF AX/DX
C   PROGRAM EFFECT ON VARIABLES OF INPUT
C     DX        - UNCHANGED
C     AX        - DESTROYED AND REPLACED BY REMAINDER OF ORDER N-1
C
      NX= N + 1
      MX= M + 1
C
C     ALGMENT BX
C
      LM=N + 2
      DO 3 I=LM,MX
    3 BX(I)= 0.0
      CX=M-N + 1
      DO 5 I=1,CX
      RPC(I)= AX(1)/BX(1)
      DO 4 C=1,MX
    4 CX(C)= AX(C) - BX(C) * RPC(I)
      MX= MX-1
      DO 5 L=1,MX
      LK= L+1
    5 AX(L)=  DX(LK)
      RETURN
      END
      SUBROUTINE POLYLL(CON,COM,N,M,XCOF)
      DIMENSION CON(1), COM(1), XCOF(1), CONA(50), COMRA(50)
C
C     FOR DOUBLE PRECISION REMOVE C FROM FIRST COLUMN OF NEXT CARD.
C     DOUBLE PRECISION CON, COM, XCOF, CONA, COMRA
C
C     THE VECTOR CON IS A VECTOR OF THE COEFFICIENT OF A POLYNOMIAL
C     OF ORDER N.
C     THE VECTOR COM IS A VECTOR OF THE COEFFICIENTS OF A POLYNOMIAL OF
C     ORDER M.
C     THE VECTOR XCOF IS A VECTOR OF THE COEFFICIENTS OF THE PRODUCT OF
C     A POLYNOMIAL OF ORDER N AND A POLYNOMIAL OF ORDER M.  THE
C     POLYNOMIAL OF WHICH THE COEFFICIENTS ARE THE VECTOR XCOF HAS AN
C     ORDER OF M + N.
C
      DO 1 I=1,M
    1 CONA(I)=0.0
      NX=N+1
```

```
      DO 2  I=1,KA
      LX=M+1
    2 CONA(LX)=CON(I)
      MX=M+1
      DO 3  I=1,MX
      MY=M+2-I
    3 COMA(I)=COM(MY)
      DO 4  I=1,N
      NX=M+1+I
    4 COMA(NX)=0.0
      KY=M+N+1
      KX=KY
      DO 7  K=1,KY
      XCOM(K)=0.0
      DO 5 L=1,KX
    5 XCOM(K)= CONA(L) * COMA(L)+XCOM(K)
      KX=KX-1
      DO 6 L=1,KX
    6 CONA(L)=CONA(L+1)
    7 CONTINUE
      RETURN
      END


      SUBROUTINE POLY(M,X,VALUE)
C
C THIS PROGRAM EVALUATES A POLYNOMIAL OF ORDER M IN X WHOSE VECTOR OF
C COEFFICIENTS IN ASCENDING ORDER IS A AT A PARTICULAR VALUE, X.
C
      DIMENSION A(50)
      Y=X
      VALUE= A(1)
      DO 10 I=1,M
      VALUE= VALUE + A(I+1) * Y
   10 Y= Y * X
      RETURN
      END
```

Additional Comments

In the discussion of the MAIN Program, it was pointed out that two types of input data could be used in the program--that is frequency response information or open loop transfer function information. Past experience has shown that for high order systems the frequency response information is more practical. Thus a designer whose only interest was with high order systems might not need the flexibility of giving the program one of two types information. Thus, he might want to change the program to only accept frequency response information. This can be easily done as described below:

1. In MAIN eliminate the variables KINFOR, K1, POLN, POLD, POLG, and POLH. These variables are mentioned on cards 1, 3, 12, 14, 16, 50, 52, 53, 57, 58, 61, 62, 94, 104, and 105. Cards which can be wholely removed without resulting in a program error are 50-64, 94, and 97-109. Other cards containing these variables will have to be modified.
2. After making the preceding mentioned changes in the MAIN Program the following sub-programs must accompany MAIN: SRMINS, POLFV, OTPT1, PARCLT, VECTOR, DIRVEC, MATINV, MATMUL.

If the program is divided as described above, the bulk content of the program will be reduced by more than one half.