Translation of a flexible rotor balancing program from FORTRAN to BASIC

Archibald, C. C.; Kim, P. Y.

Laboratory Technical Report (National Research Council Canada. Division of Mechanical Engineering. Engine Laboratory); no. LTR-ENG-104, 1981-08

National Research Council Canada | Conseil national de recherches Canada

DIVISION OF MECHANICAL ENGINEERING

DIVISION DE GÉNIE MÉCANIQUE

PAGES
PAGES

FIG.
DIAG.

TABLES
TABLES

REPORT
RAPPORT

LABORATORY / LABORATOIRE
ENGINE LABORATORY

REPORT
RAPPORT LTR-ENG-104

DATE
DATE AUGUST 1981

LAB. ORDER
COMM. LAB. 21703

FILE
DOSSIER 3641-11

FOR
POUR

REFERENCE
RÉFÉRENCE

LTR-ENG-104

TRANSLATION OF A FLEXIBLE ROTOR BALANCING PROGRAM FROM FORTRAN TO BASIC

SUBMITTED BY
PRÉSENTÉ PAR H. S. Fowler
LABORATORY HEAD
CHEF DE LABORATOIRE

APPROVED
APPROUVÉ E. H. Dudgeon
DIRECTOR
DIRECTEUR

AUTHOR(S)
AUTEUR C.C. Archibald*
P.Y. Kim

Classification: Unclassified
Distribution: Unlimited

*Summer Student, Acadia University
Wolfville, Nova Scotia

TABLE OF CONTENTS

		Page
I	INTRODUCTION	1
II	FORTRAN TO BASIC TRANSLATION	2
1.	BASIC LINE NUMBERS VS FORTRAN STATEMENT NUMBERS	2
2.	RELATIONAL OPERATORS	3
3.	LOGICAL OPERATORS	3
4.	ARITHMETIC OPERATORS	3
5.	TRIGONOMETRIC FUNCTIONS	3
6.	COMMENTS	4
7.	LOOPING	4
8.	ASSIGNMENT	4
9.	BRANCHING	5
A.	Unconditional Branching	5
B.	Conditional Branching	5
10.	SUBROUTINES	6
11.	INPUT/OUTPUT	6
12.	TRIGONOMETRIC FUNCTIONS FOR COMPLEX VARIABLES	7
13.	COMPLEX VARIABLE HANDLING IN BASIC	8
A.	Multiplication	8
B.	Division	9
C.	Addition	9
D.	Subtraction	9
E.	Complex Absolute Value	9

TABLE OF CONTENTS - continued

		Page
	F. Conjugate Complex Variable	10
	G. Complex Array Handling	10
III	**ROTOR BALANCING PROGRAM TRANSLATION**	10
1.	VARIABLES	11
2.	EXPLANATION OF COMPLEX VARIABLES (DAF)	13

LIST OF APPENDICES

Appendix		Page
A	FORTRAN ROTOR BALANCING PROGRAM LISTING	A1
B	BASIC ROTOR BALANCING PROGRAM LISTING	B1

I INTRODUCTION

This report describes the translation of a specific computer program from FORTRAN to BASIC. This program was essential as a sub-problem in the overall project of:

i) Interfacing a desktop computer, namely the TEKTRONIX 4051, with the DIGITAL VECTOR FILTER 2.

ii) Utilization of the flexible rotor balancing program on the TEKTRONIX 4051 to generate an analytical solution for unbalance correction weights and angular location.

The flexible rotor balancing program named WATFIV.TENPLANE has only been written in FORTRAN while the TEKTRONIX 4051 desktop computer uses only the BASIC language. Therefore, the translation of the program to BASIC was essential.

The reader of this report may also be interested in the report LTR-ENG-103, titled "An Interface between the Digital Vector Filter 2 and the TEKTRONIX 4051", which describes the software written to accomplish item (i) mentioned above.

One of the major difficulties in the translation process was the lack of complex variables and complex operations in BASIC. "Simulated" complex variables had to be created to overcome this problem. This is discussed in the "Variables" part of Section III.

The TEKTRONIX 4050 series BASIC does not permit the passing of parameters to subroutines and functions. This can cause repetitive sections of code, but creates no serious translation difficulties provided that the translator is aware of what variables are destroyed in the FORTRAN subroutine or function environment.

The FORTRAN compiler normally does trigonometric functions in radians, while the TEK 4051 defaults to degrees. This can be overcome by translating the FORTRAN program to degrees, or by executing the BASIC statement "SET RADIANS". The latter option was chosen in this case.

The BASIC interpreter accepts a maximum of two-character variable names; one letter, followed by one digit; or one letter alone. This makes the readability very poor for a program of this size and complexity. It is not recommended that the BASIC balancing program be read without a copy of its FORTRAN equivalent.

Keeping track of the dimensions and indices in the multidimensioned variables becomes very confusing. This is a result of two-character variable names, and simulated complex variables. Debugging a translated program should include checking the indices with the utmost care.

The TEK 4051 used for this translation is capable of holding only 16K bytes of memory, so the program had to be reduced to 6 plane, 6 probe and 2 speed balancing. The program was also cut into two pieces with a variable holding interface using the magnetic tape.

II FORTRAN TO BASIC TRANSLATION

The BASIC language elements used to translate this program are discussed here in terms of FORTRAN statements. The reader is expected to be familiar with the FORTRAN language.

1. BASIC LINE NUMBER VS FORTRAN STATEMENT NUMBERS

The BASIC language is line oriented. Every BASIC statement begins with an integer line number between 1 and 60000. Usually line numbers greater than 100 and in multiples of 10 are used. The statements must be in order according to these numbers, but need not be entered in this order, as the BASIC interpreter will sort them. Many BASIC statements use these line numbers in a manner similar to that of FORTRAN statement numbers. In the examples of statements in this report, the numbers preceding BASIC statements are BASIC LINE numbers, and the numbers preceding some FORTRAN statements are FORTRAN STATEMENT numbers.

2. RELATIONAL OPERATORS

	FORTRAN	BASIC
greater than	.GT.	>
less than	.LT.	<
less than or equal	.LT.	=<
greater than or equal	.GE.	=>
not equal	.NE.	<>

3. LOGICAL OPERATORS

.AND.	AND
.OR.	OR
.NOT.	NOT

4. ARITHMETIC OPERATORS

add (positive)	+	+
subtract (negative)	-	-
divide	/	/
multiply	*	*
exponent	**	↑

5. TRIGONOMETRIC FUNCTIONS

Sine	SIN (N)	SIN (N)
Cosine	COS (N)	COS (N)
Tangent	TAN (N)	TAN (N)
Arctangent	ATAN (N)	ATN (N)
	ATAN2 (N1, N2)	-

6. COMMENTS

Placing "REM" after the line number in a BASIC program will cause the BASIC interpreter to ignore everything else on this line.

```
C     FORTRAN COMMENT          100 REM    BASIC COMMENT
```

7. LOOPING

The most common looping structure in BASIC is loosely called the "FOR-NEXT" structure. It functions exactly like the FORTRAN "DO-LOOP" and an example is shown below.

```
      FORTRAN                                  BASIC

   DO  50 I = 1, N                      100 FOR I = 1 TO N
   .  .                                  .  .
   .  .                                  .  .
   .  .                                  .  .
50 CONTINUE                             180 NEXT I
```

8. ASSIGNMENT

Assignment statements are exactly the same in both languages except for the facts that:

i) Variables in BASIC can have up to two characters, that is, the first character must be alphabetic and the second, if used, must be numeral.

ii) In BASIC there is no need to distinguish integers from real numbers.

```
   FORTRAN                                  BASIC

STPHA = 10.6                            100 S1 = 10.6
```

9. BRANCHING

A. Unconditional Branching

The "GOTO" method of branching is the same in both languages, except that in FORTRAN the integer indicates a FORTRAN STATEMENT NUMBER, while in BASIC, it is a BASIC LINE NUMBER.

FORTRAN	BASIC
GOTO 50	500 GOTO 1000
.	.
.	.
.	.
50 CONTINUE	1000 next BASIC statement
where 50 is a FORTRAN statement number	where 1000 is a BASIC line number

B. Conditional Branching

The "IF" statement is more convenient in FORTRAN because almost any executable statement can be controlled by an "IF". In BASIC it is necessary to negate the condition and transfer control to a line number beyond the statements controlled by the "IF".

FORTRAN	BASIC
IF (QUAMP.EQ.5) Executable Statement	IF Q = 5 THEN 1000
where "executable statement" can be any executable FORTRAN statement with the exception of the "DO" statement and another "IF" statement.	where 1000 is a BASIC line number.

The "Arithmetic IF" in FORTRAN has no BASIC equivalent, but the logic can easily be translated into several BASIC "IF" statements. The BASIC translation for the example of a FORTRAN "Arithmetic IF" is indicated below:

FORTRAN	BASIC
IF (K-6) 50, 60, 70	NONE However, it can be transformed into a set of BASIC statements as below. IF K-6 < 0 THEN 50 IF K-6 = 0 THEN 60 IF K-6 > 0 THEN 70 where 50, 60, 70 are BASIC line numbers

10. SUBROUTINES

Calling subroutines in BASIC is of limited use. Parameters, or arguments can not be used as they are in FORTRAN. The only advantage is that a piece of code can be executed and control returned to various points in the main program. Subroutines in BASIC are not named, but are identified by the line number of the first statement.

FORTRAN	BASIC
Call SOLVE (P1, P2,...PN)	100 GOSUB 2000
.	.
.	.
.	.
SUBROUTINE SOLVE (A1, A2,...AN)	2000 REM SUBROUTINE
.	.
.	.
.	.
RETURN	2020 RETURN

11. INPUT/OUTPUT

When using formatted I/O the BASIC "IMAGE" statement corresponds to the FORTRAN "FORMAT" statement. For the many various options in the "IMAGE" statement refer to the I/O section of the BASIC manual. "PRINT" and "INPUT" are the BASIC commands generally used for the magnetic tape drive.

FORTRAN	BASIC
PRINT 90 90 FORMAT (...)	100 PRINT USING 110 110 IMAGE...

12. TRIGONOMETRIC FUNCTIONS FOR COMPLEX VARIABLES

To find the trigonometric value of a complex number in the FORTRAN language requires only a call to a built-in function. There are no such functions in the BASIC language so a short algorithm is required. The purpose of this sequence is to ensure that the resulting angle will be in the correct quadrant.

The "ATAN2" function in FORTRAN, for example, will automatically check the sign of the real and imaginary parts of the quotient and therefore return the angle associated with pair of coordinates in the proper quadrant. For example:

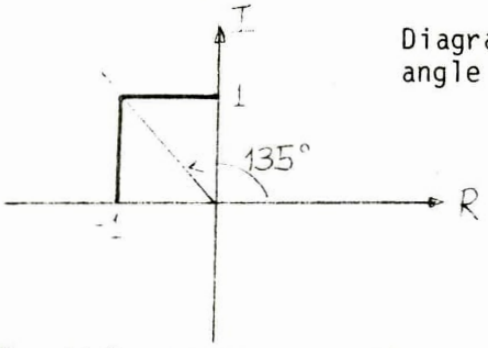
Examining the complex number 1 - 1i

-1/+1 = -1

This Arctan (-1) = -45° which is correct. Next examine the complex number -1 + 1i

+1/-1 = -1

This Arctan (-1) = -45° which is not correct.



Diagramatically we can see that the angle should be 180°-45° = 135°.

For this reason we require a check of the signs of the real and imaginary parts. The following section of BASIC code accomplishes this task. Consider the complex number A1, A2. (A1 is the real part).

```
100     IF A1 <> 0 THEN 400
200     R1 = 0
300     GOTO 1100
400     IF A1 = > 0 THEN 700
500     R1 = ATN(A2/A1) + 180
600     GOTO 1100
700     IF A2 < 0 THEN 1000
800     R1 = ATN(A2/A1)
900     GOTO 1100
1000    R1 = ATN(A2/A1) + 360
1100    REM COMPLETED
```

NOTE: These steps assume that the imaginary part of the complex number is representing degrees. If these are in radians, conversion is necessary. Line 500 becomes:

```
500 R1 = ATN(A2/A1) * 57.2958 + 180.
```

This would result in the answer represented in degrees.

NOTE: Similar steps are required to properly find other trigonometric values of complex numbers.

13. COMPLEX VARIABLE HANDLING IN BASIC

The BASIC language can be used to manipulate complex variables even though the specific handling features are not built into the language. To store a complex number, two numeric variables are used; one for the real part and one for the imaginary part, for example C1 and C2. The functions used in other languages to manipulate this complex variable can be easily be simulated in BASIC. Here are some examples of complex operations.

A. Multiplication

(C1, C2) * (D1, D2) = (A1, A2)

First do the Real part:

A1 = C1 * D1 - C2 * D2

Then, the Imaginary part:

A2 = D1 * C2 + D2 * C1

B. Division

(C1, C2) ÷ (D1, D2) = (A1, A2)

The Real Part:

A1 = (C1 * D1 + C2 * D2)/(D2↑ 2 + D1↑ 2)

The Imaginary Part:

A2 = (D1 * C2 - C1 * D2)/(D2↑ 2 + D1↑ 2)

C. Addition

(C1, C2) + (D1, D2) = (A1, A2)

The Real Part:

A1 = C1 + D1

The Imaginary Part:

A2 = C2 + D2

D. Subtraction

(C1, C2) - (D1, D2) = (A1, A2)

A1 = C1 - D1

A2 = C2 - D2

E. Complex Absolute Value

Compute the complex absolute value of (C1, C2) and store the result in R.

R = ABS (SQR(C1↑ 2 + C2↑ 2))

F. Conjugate Complex Variable

Store the conjugate of (C1, C2) in (A1, A2)

```
A1 = C1

A2 = -C2
```

G. Handling arrays of complex variables can be done in a similar manner. As with the "simple" complex variables the operations can be simulated. One-dimensional or multidimensional arrays can be used. The following is a simple example of multiplying two one-dimensional complex arrays.

```
100 REM DECLARE THE ARRAY VARIABLES

110 DIMENSION A1(10), A2(10), C1(10), C2(10), D1(10),
D2(10)
 .
 .
 . Assign some values to C1, C2, D1, D2 arrays.

300 REM MULTIPLY THE ARRAYS.  STORE RESULTS IN A1, A2.

310 FOR I = 1 to 10

320 A1(I) = C1(I) * D1(I) - C2(I) * D2(I)

330 A2(I) = D1(I) * C2(I) + D2(I) * C1(I)

340 NEXT I
```

III ROTOR BALANCING PROGRAM TRANSLATION

When examining the two programs in the appendix, the reader should realize that the FORTRAN program has not been translated in every detail. The data structures are different, due to lack of complex variable handling in BASIC. The subroutine "SOLVE" is one of the most complicated and important

parts mathematically, and this portion was translated in every detail. By tracing through the FORTRAN program and looking at the BASIC program simultaneously, the reader should have no trouble following the logic of the translation. It is worth noting that there is a difference in the size of the arrays. The FORTRAN program did not need to worry about memory restrictions, but the BASIC program was modified to balance a 6 plane, 6 probe, 2 speed (maximum) rotor because of the limitations in the total memory capacity. An interface between program segments was also necessary. At the end of the first routine some variables were written onto the tape cartridge, and read back at the start of the second routine.

The authors of the BASIC translation do not advise that the reader try to follow the logic in BASIC without following along with the FORTRAN program. The BASIC language was not intended for such complex applications. Although it is possible to translate this complex logic already developed in FORTRAN or some other high level language, it is all but impossible, and certainly not practical to develop software of this complexity in BASIC.

1. VARIABLES

Strictly numeric variables were used in these programs, so only numeric will be discussed here. As previously mentioned, BASIC allows only two-character variable names - one letter, and one digit. In the variable table which follows, the BASIC equivalents of the FORTRAN variable names used are given in approximate order of occurrence. Some variable names in the FORTRAN program were of a suitable format for BASIC so these were not changed, and may not appear on this table. Occasionally more variables were needed in BASIC to facilitate complex operations. These are indicated by the FORTRAN variable name TEMP.

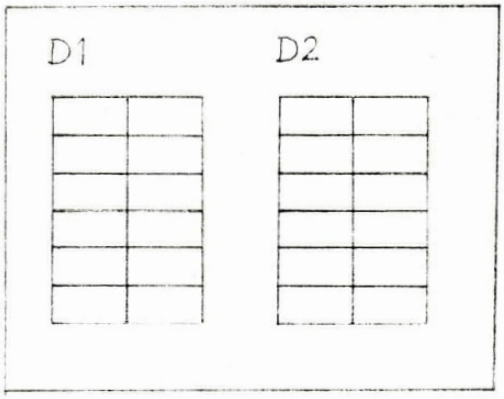
FORTRAN:	BASIC:
AMPR	A
PHASER	P
A1	A1
A2	A2
RUN OUT	R1(6,2)
AMPB	A3
PHASE B	P3
AB1	A4
AB2	A4

FORTRAN:	BASIC:
DI	S1(6,2) S2(6,2)
AMPA	A6
PHASEA	P4
AA1	T1
AA2	T2
DAF	(SEE EXPLANATION - NEXT PAGE)
L	L
M	M
N	N
K	K
UAMP	W
UANG	U
TRUNBA	B(6,2)
IP	X1
JP	X2
ALPHA	C1(12,6) C2(12,6)
TEMP	T4
TEMP	T5
B1	B1
B2	B2
AMPINF	F1(6)
ANGINF	F2(6)
SUM	S(2)
TEMP	T(2)
SYST2	Z1(6,7) Z2(6,7)
CC	C3(6), C4(6)
NORDER	N
C	V1(7), V2(7)
JJJ	J3
NN	N2
MM	M2
K	K9
II	I2
DUMMY	T6, T7
TEMP	T8
TEMP	T9
SQ	Q
FINAL	F8(2,6) F9(2,6)
N	N9
M	M9

2. Explanation of Complex Variables: (DAF)

This complex FORTRAN variable has three dimensions. Since three-dimensional arrays are not allowed in BASIC language, special BASIC variables were created to simulate this in the following manner: the dimensions were with respect to number of speeds, number of planes, and number of probes. In BASIC there must be a real part and an imaginary part for each FORTRAN array element. Several two dimensional arrays were used.

Plane 1



For Plane 1, D1 and D2 are used. D1 is for speed 1 and D2 is for speed 2. The rows represent the probes (max 6) and the columns are the real and imaginary parts respectively. The rest of the array names are:

Plane 2	D3, D4
Plane 3	D5, D6
Plane 4	D7, D8
Plane 5	D9, E1
Plane 6	E2, E3

All complex variables are formatted in a similar manner.

APPENDIX A

FORTRAN ROTOR BALANCING PROGRAM LISTING

```
C   THIS VERSION IS IN DOUBLE PRECISION- JULY 29/80
C$JOB          ,XREF
C     PROGRAM LTSQAP(INPUT,OUTPUT)
C
C     PROGRAM FOR PLAIN LEASTS SQUARES PREDICTION OF BALANCE WEIGHTS
C
C     INPUT DESCRIPTION
C
C     CARDS 1 ( 3 OF THESE )
C     HEADING CARDS ON PRINTOUT
C     INCLUDE UNITS OF UNBALANCE AND DISPLACEMENT ON THESE
C
C     CARD 2
C     NPLANE (NUMBER OF UNBALANCE PLANES MAX 10)  I10
C     LPROBE (NUMBER OF PROBES  MAX 10)  I10
C     KSPEED (NUMBER OF SPEEDS  MAX 10)  I10
C
C     CARDS 3 ( LPROBE OF THESE )   RUNOUT AT PROBES
C     AMPR : AMPLITUDE  P-P/2   F10.3
C     PHASER : PHASE LAG FROM VECTOR FILTER IN DEGREES  F10.3
C
C     CARDS 4 (KSPEED TIMES LPROBE OF THESE ) INITIAL PROBE READINGS
C     ENTER PROBE READINGS AT EACH SPEED
C     AMPB : AMPLITUDE  P-P/2   F10.3
C     PHASEB : PHASE LAG IN DEGREES  F10.3
C
C     CARDS 5 ( NPLANE TIMES KSPEED TIMES LPROBE OF THESE ) PROBE DATA
C     AFTER ADDING TRIAL WEIGHTS
C     PROBES ARE THE INNER INDEX TO SPEEDS WHICH IS THE INNER
C     INDEX TO UNBALANCE PLANES
C     AMPA : AMPLITUDE P-P/2   F10.3
C     PHASEA : PHASE LAG IN DEGREES  F10.3
C
C     CARDS 6 ( NPLANE OF THESE )  TRIAL WEIGHT DATA
C     UAMP : MAGNITUDE OF TRIAL UNBALANCE          F10.3
C     UANG : ANGLE IN DEGREES  F10.3
C
C     ALL UNBALANCE ANGLES REFERENCED TO DIRECTION OF KEY PROBE
C     AND IN THE DIRECTION OF ROTATION
C
C     ALL PROBE ANGLES ARE THE PHASE ANGLE LAG OF THE AMPLITUDE
C     SEEN BY THE PROBE BEHIND THE TIMING MARK
C
C
C*****************************************************************
      IMPLICIT REAL*8(A-H,O-Z)
C     THE DIMENSIONS OF THE VARIABLES MUST BE GREATER THAN OR EQUAL TO
C     THE FOLLOWING NUMBERS:
C     AMPINF(NPLANE), ANGINF(NPLANE), C(NPLANE), FINAL(KSPEED,LPROBE)
C     SYST2(NPLANE,NPLANE+1),CC(NPLANE), STORE(NPLANE,NPLANE+1)
C     TRUNBA(NPLANE), ALPHA(LPROBE*KSPEED,NPLANE), RUNOUT(LPROBE)
C     DI(KSPEED,LPROBE), AND DAF(NPLANE,KSPEED,LPROBE)
      DIMENSION ICENT(3,80),AMPINF(10),ANGINF(10)
      COMPLEX*16 C(10),DUMMY,FINAL(10,10),SYST2(15,15),CC(10),
     6 STORE(15,15),TRUNBA(10),ALPHA(100,10),SUM,RUNOUT(10),DCMPLX,
```

```
     6DI(10,10),TEMP,DCONJG,CHECKE,DAF(10,10,10), SUBRUN
C      LIMIT 10 PROBES  10 SPEEDS   10 UNBALANCE PLANES
C      NPLANE IS THE NUMBER OF UNBALANCE PLANES
C      LPROBE IS THE NUMBER OF PROBES
C      K IS THE NUMBER OF SPEEDS
       PRINT 66
   66  FORMAT(1H1,'PROGRAM FOR PLAIN LEAST SQUARES PREDICTION',//,
      6 ' OF CORRECTION UNBALANCES.')
       PRINT 67
   67  FORMAT(1H0,'WRITTEN BY ALAN PALAZZOLO 4/28/77',//,
      6 ' REF. LUND & GOODMAN PAPERS')
       DO 81 J=1,3
       READ(4,83)(ICENT(J,I),I=1,80)
   83  FORMAT(80A1)
   81  CONTINUE
       PRINT 77,((ICENT(J,I),I=1,80),J=1,3)
   77  FORMAT(1H-,3(10X,80A1/1H0))
       READ(4,1)NPLANE,LPROBE,KSPEED
    1  FORMAT(3I10)
       M=LPROBE*KSPEED
       PRINT 5,NPLANE,LPROBE,KSPEED,M
    5  FORMAT(1H-,8X,'NUMBER OF BALANCE PLANES',I3,//,9X,'NUMBER OF PROBE
     6S',8X,I3,//9X,'NUMBER OF SPEEDS',8X,I3,//9X,'NUMBER OF ',
     6' MEASUREMENTS TO MINIMIZE',I3)
C         READ IN RUNOUT AMPLITUDE AND PHASE ( DEGREES) AT EACH PROBE
       PRINT 12
   12  FORMAT(1H-,15X,'RUNOUT,AMPLITUDE...LAG ANGLE IN DEGREES')
       DO 10 I=1,LPROBE
       READ(4,11)AMPR,PHASER
   11  FORMAT(2F10.3)
       PRINT 13,AMPR,PHASER
   13  FORMAT(1H0,18X,F10.3,5X,F10.3)
       PHASER=PHASER/57.2958
       A1=AMPR*DCOS(PHASER)
       A2=-AMPR*DSIN(PHASER)
       RUNOUT(I)=DCMPLX(A1,A2)
   10  CONTINUE
C         ENTER INITIAL PROBE DATA
       PRINT 15
   15  FORMAT(1H-,'INITIAL READINGS,AMPLITUDE..LAG ANGLE    SAME EXCEPT',
      6/,28X,'IN DEGREES   RUNOUT SUBTRACTED')
       SUMSQB=0.0
       DO 20 I=1,KSPEED
C         NUMBER OF SPEEDS INDEX
       DO 22 J=1,LPROBE
C         NUMBER OF PROBES INDEX
       READ(4,24)AMPB,PHASEB
   24  FORMAT(2F10.3)
       STPHB=PHASEB
       PHASEB=PHASEB/57.2958
       AB1=AMPB*DCOS(PHASEB)
       AB2=-AMPB*DSIN(PHASEB)
       DI(I,J)=DCMPLX(AB1,AB2)
C         SUBTRACT RUNOUT FROM INITIAL READINGS FOR PRINTOUT
       SUBRUN=DI(I,J)-RUNOUT(J)
```

```
      D1=DREAL(SUBRUN)
C        PRINT THIS OUT JUST LIKE VECTOR FILTER
      D2=-DIMAG(SUBRUN)
            IF(D1.EQ.0.0.AND.D2.EQ.0.0)  D3=0.0
            IF(D1.EQ.0.0.AND.D2.EQ.0.0) GO TO 42
      D3=DATAN2(D2,D1)*57.2958
   42 CONTINUE
      D4=CDABS(SUBRUN)
      IF(D3.LT.0.0) D3=D3+360.0
      PRINT 16,I,J,AMPB,STPHB,D4,D3
   16 FORMAT(1HO,'SPEED',I3,1X,'PROBE',I3,F10.3,F10.3,
     6F10.3,F10.3)
      SUMSQB=SUMSQB+AMPB*AMPB
   22 CONTINUE
   20 CONTINUE
      PRINT 505,SUMSQB
  505 FORMAT(1H-,'THE SUM OF THE AMPLITUDE SQUARES OF THE INITAL READING
     6S',/,' IS',F12.4)
C        ENTER FINAL PROBE DATA
C        ASUMMING THIS ORDER  ...  PROBES , SPEEDS , UNBALANCE
      PRINT 27
   27 FORMAT(1H-,'READINGS - TRIAL WEIGHT ADDED,--AMPLITUDE--LAG(DEG)')
      PRINT 28
   28 FORMAT(1HO,'                              AMPLITUDE  ANGLE     SAME(MINU
     1S',/,'                                                        RUNOUT)')
      DO 30 I1=1,NPLANE
C        UNBALANCE PLANE INDEX
      DO 40 I2=1,KSPEED
C        SPEED INDEX
      DO 50 I3=1,LPROBE
C        PROBE INDEX
      READ(4,55)AMPA,PHASEA
   55 FORMAT(2F10.3)
      STPHA=PHASEA
      PHASEA=PHASEA/57.2958
      AA1=AMPA*DCOS(PHASEA)
      AA2=-AMPA*DSIN(PHASEA)
      DAF(I1,I2,I3)=DCMPLX(AA1,AA2)
C        SUBTRACT RUNOUT FROM THESE READINGS FOR PRINTOUT
      SUBRUN=DAF(I1,I2,I3)-RUNOUT(I3)
      D1=DREAL(SUBRUN)
C        PRINTOUT RUNOUT CORRECTED READING LIKE VECTOR FILTER
C        SUBRUN IS A REGULAR COMPLEX NUMBER. IN POLAR FORM MAKE ITS
C        ANGLE THE LAG OF THE POSITIVE AMPLITUDE BEHIND THE TIMING MARK
      D2=-DIMAG(SUBRUN)
      IF(D1.EQ.0.0.AND.D2.EQ.0.0)  D3=0.0
      IF(D1.EQ.0.0.AND.D2.EQ.0.0) GO TO 44
      D3=DATAN2(D2,D1)*57.2958
   44 CONTINUE
      D4=CDABS(SUBRUN)
       IF(D3.LT.0.0)  D3=D3+360.0
      PRINT 35,I2,I3,AMPA,STPHA,D4,D3,I1
   35 FORMAT(1HO,'BALANCE  SPEED',I2,' PROBE',I2,F8.3,F8.2,F9.3,
     6F8.2,/,' PLANE',I2)
   50 CONTINUE
```

```
   40 CONTINUE
   30 CONTINUE
C        READ IN TRIAL WEIGHT LOCATIONS
      PRINT 72
   72 FORMAT(1H-,2X,'TRIAL UNBALANCES, MAGNITUDE...ANGLE(DEG.) MEASURED
     6IN',/,2X,'DIRECTION OF ROTATION FROM THE KEY PROBE DIRECTION')
      DO 70 I=1,NPLANE
      READ(4,75)UAMP,UANG
   75 FORMAT(2F10.3)
      PRINT 74,I,UAMP,UANG
   74 FORMAT(1H0,2X,'  BALANCE PLANE',I3,F10.3,2X,F10.3)
      UANG=UANG/57.2958
      A1=UAMP*DCOS(UANG)
      A2=UAMP*DSIN(UANG)
      TRUNBA(I)= DCMPLX(A1,A2)
   70 CONTINUE
C        DEFINE INFLUENCE COEFFICIENT MATRIX
      IP=1
      JP=1
      DO 80 I=1,M
      DO 90 J=1,NPLANE
      ALPHA(I,J)= (DAF(J,JP,IP)-DI(JP,IP))/TRUNBA(J)
   90 CONTINUE
C        DEFINE THE PROBE NUMBER IP AND THE SPEED NUMBER
C        JP FROM THE CASE NUMBER  I
      IP=IP+1
      IF(IP.GT.LPROBE) IP=1
      IF(IP.EQ.1)  JP=JP+1
   80 CONTINUE
      PRINT 93
   93 FORMAT(1H-,'COMPLEX INFLUENCE COEFFICIENT MATRIX UNITS ARE UNITS O
     6F',/,' RESPONSE DIVIDED BY UNITS OF UNBALANCE',/10X,'MAGNITUDE...
     6PHASE IN DEGREES')
      DO 97 I=1,M
      DO 99 J=1,NPLANE
      B1=DIMAG(ALPHA(I,J))
      B2=DREAL(ALPHA(I,J))
      AMPINF(J)=CDABS(ALPHA(I,J))
      IF(B1.EQ.0.0.AND.B2.EQ.0.0) ANGINF(J)=0.0
      IF(B1.EQ.0.0.AND.B2.EQ.0.0) GO TO 60
      ANGINF(J)=DATAN2(B1,B2)*57.2958
   60 CONTINUE
      IF(ANGINF(J).LT.0.0) ANGINF(J)=ANGINF(J)+360.0
   99 CONTINUE
      PRINT 98,(AMPINF(J),ANGINF(J),J=1,NPLANE)
   98 FORMAT(1H0,5(8X,F8.3,2X,F8.3))
   97 CONTINUE
C        DEFINE THE CONJUGATE TRANSPOSE INFLUENCE COEFFICIENT MATRIX
C        POSTMULTIPLIED BY THE INFLUENCE COEFFICIENT MATRIX
      DO 100 I=1,NPLANE
      DO 110 J=1,NPLANE
      SUM=(0.0,0.0)
      DO 115 I1=1,M
      TEMP=ALPHA(I1,I)
      TEMP=DCONJG(TEMP)
```

```
      SUM=SUM+ALPHA(I1,J)*TEMP
  115 CONTINUE
      SYST2(I,J)=SUM
  110 CONTINUE
  100 CONTINUE
C        DEFINE THE VECTOR INFLUENCE COEFFICIENT CONJUGATE TRANSPOSE
C        MATRIX TIMES THE RUNOUT SUBTRACTED ORIGINAL VECTOR
      DO 125 I=1,NPLANE
      SUM=(0.0,0.0)
      IP=1
      JP=1
      DO 130 I1=1,M
      TEMP=ALPHA(I1,I)
      TEMP=DCONJG(TEMP)
      SUM=SUM+TEMP*(DI(JP,IP)-RUNOUT(IP))
C        DEFINE THE PROBE NUMBER IP AND THE SPEED NUMBER JP
C        FROM THE CASE NUMBER I1
      IP=IP+1
      IF(IP.GT.LPROBE) IP=1
      IF(IP.EQ.1) JP=JP+1
  130 CONTINUE
      CC(I)=SUM
  125 CONTINUE
C        TRANSFER CC INTO THE LAST COLUMN OF SYST2
      DO 140 I=1,NPLANE
      SYST2(I,NPLANE+1)=CC(I)
  140 CONTINUE
C        SOLVE THE SIMULTANEOUS EQ. AND OBTAIN THE
C        CORRECTION WEIGHTS , BUT FIRST STORE SYST2
      MAT=NPLANE+1
      DO 150 I=1,NPLANE
      DO 160 J=1,MAT
      STORE(I,J)=SYST2(I,J)
  160 CONTINUE
  150 CONTINUE
      CALL SOLVE(SYST2,C,NPLANE)
C        RESTORE SYST2 TO ITS ORIGINAL VALUE ( DESTROYED IN SOLVE)
      DO 170 I=1,NPLANE
      DO 180 J=1,MAT
      SYST2(I,J)=STORE(I,J)
  180 CONTINUE
  170 CONTINUE
      CALL SIMCHE(SYST2,C,NPLANE )
      PRINT 200
  200 FORMAT(1H-,1X,'*************************************************
     6****',/2X,'* CORRECTION UNBALANCES...ANGLES IN DEGREES MEASURED  *
     6',/2X,'* IN DIRECTION  OF ROTATION FROM KEY PROBE DIRECTION  *',
     6/,'  *                                MAGNITUDE        ANGLE',4X,'*')
      DO 205 I=1,NPLANE
      A1=DREAL(C(I))
      A2=DIMAG(C(I))
      IF(A2.EQ.0.0.AND.A1.EQ.0.0) A3=0.0
      IF(A2.EQ.0.0.AND.A1.EQ.0.0) GO TO 62
      A3=DATAN2(A2,A1)*57.2958
   62 CONTINUE
```

```
      A4=CDABS(C(I))
      IF(A3.LT.0.0) A3=A3+360.0
      PRINT 210,I,A4,A3
  210 FORMAT(1HC,1X,'*       UNBALANCE PLANE',I3,F11.3,7X
     6  ,F7.3,4X,'*',/,'  *',53X,'*')
  205 CONTINUE
      IP=1
      JP=1
      DO 300 I=1,M
      SUM=(0.0,0.0)
      DO 400 I1=1,NPLANE
      SUM=SUM+ALPHA(I,I1)*C(I1)
  400 CONTINUE
      SUM=SUM+DI(JP,IP)
      FINAL(JP,IP)=SUM
      IP=IP+1
      IF(IP.GT.LPROBE) IP=1
      IF(IP.EQ.1) JP=JP+1
  300 CONTINUE
      PRINT 450
  450 FORMAT(1H-,' ************************************************
     6**',/,8X,'RESIDUAL DISPLACEMENTS WITH CORRECTION',/9X,
     6'WEIGHTS ADDED INCLUDING RUNOUT')
      SUMSQA=0.0
      DO 475 I=1,KSPEED
      PRINT 480,I
  480 FORMAT(1HO,2X,'SPEED ',I3)
      DO 490 J=1,LPROBE
      A1=DREAL(FINAL(I,J))
      A2=-DIMAG(FINAL(I,J))
      IF(A2.EQ.0.0.AND.A1.EQ.0.0) A3=0.0
      IF(A2.EQ.0.0.AND.A1.EQ.0.0) GO TO 64
      A3=DATAN2(A2,A1)*57.2958
   64 CONTINUE
      A4=CDABS(FINAL(I,J))
      IF(A3.LT.0.0) A3=A3+360.0
      PRINT 500,J,A4,A3
  500 FORMAT(1HO,2X,'PROBE ',I3,2X,'MAGNITUDE ',F7.3,
     63X,'LAG ANGLE(DEG) ',F7.3)
      SUMSQA=SUMSQA+A4*A4
  490 CONTINUE
  475 CONTINUE
      PRINT 510,SUMSQA
  510 FORMAT(1H-,2X,'THE SUM OF THE AMPLITUDE SQUARES OF THE ',
     6'RESIDUAL',/,' DISPLACEMENTS IS ',F12.4)
      STOP
      END
      SUBROUTINE SOLVE(SYST2,C,NORDER)
C        THIS SUBROUTINE SOLVES THE SIMULTANEOUS EQUATIONS CONTAINED
C        IN MATRIX SYST2 BY DIAGONALIZATION AND BACK SUBSTITUTIONS
C        WITHOUT PIVOTING
C        THIS IS ONLY GOOD FOR SYSTEMS OF ORDER NORDER-1
C        SOURCE D.LI  2/77
      IMPLICIT REAL*8(A-H,O-Z)
      COMPLEX*16 DUMMY,DCONJG
```

```
      COMPLEX*16 SYST2(15,15),C(10)
      NN=NORDER
      MM=NORDER+1
      DO 100 I=1,NN
      K=I
   15 IF(CDABS(SYST2(K,I)).GT.0.000001) GO TO 20
      K=K+1
      IF(K-NN) 15,15,200
   20 IF(I-K) 40,60,200
   40 DO 50 M=1,MM
      DUMMY=SYST2(I,M)
      SYST2(I,M)=SYST2(K,M)
   50 SYST2(K,M)=DUMMY
   60 II=I+1
      IF(II.GT.NN) GO TO 100
      DO 70 N=II,NN
      IF(CDABS(SYST2(N,I)).LT.0.000001) GO TO 70
      SQ=(CDABS(SYST2(I,I)))**2
      DUMMY=SYST2(N,I)*DCONJG(SYST2(I,I))/SQ
      DO 80 M=1,MM
   80 SYST2(N,M)=SYST2(N,M)-SYST2(I,M)*DUMMY
   70 CONTINUE
  100 CONTINUE
      GO TO 300
  200 IF(I.EQ.NN)  PRINT 500
      IF(I.LT.NN) PRINT 510
  500 FORMAT(1H ,19HMATRIX HAS ZERO ROW)
  510 FORMAT(1H ,22HMATRIX HAS ZERO COLUMN)
C         THE ROW REMOVED FROM THE SYSTEM IN THE MAIN PROGRAM,
C         THE LAST ROW IN (LAMBDA*I-D)),WAS NOT REDUNDANT
C         THE DETERMINANT OF THE REMAINING SYSTEM IS ZERO
      STOP
C         START BACK SUBSTITUTIONS
  300 I=NN
  320 DUMMY=(0.0,0.0)
      IF(I.EQ.NN)   GO TO 350
      JJJ=I+1
      DO 330 J=JJJ,NN
  330 DUMMY=DUMMY+SYST2(I,J)*C(J)
  350 SQ=(CDABS(SYST2(I,I)))**2
      C(I)=-(DUMMY+SYST2(I,MM))*DCONJG(SYST2(I,I))/SQ
      I=I-1
      IF(I) 400,400,320
  400 RETURN
      END
C         THIS ROUTINE CHECKS SOLVE (COMPLEX SIMULTANEOUS EQ SOLVER)
      SUBROUTINE SIMCHE(SYST2,C,NORDER )
      IMPLICIT REAL*8(A-H,O-Z)
      COMPLEX*16 CHECKE,SYST2(15,15), C(10)
      N=NORDER+1
      NN=N-1
      PRINT 90
   90 FORMAT(1H0,10X,'CHECK ON SIMULTANEOUS EQUATION ROUTINE')
      DO 10 I=1,NN
      CHECKE= (0.0,0.0)
```

```
      DO 20 J=1,NN
      CHECKE = CHECKE+SYST2(I,J)*C(J)
   20 CONTINUE
      CHECKE = CHECKE+SYST2(I,N)
      PRINT 40,CHECKE
   40 FORMAT(1H ,10X,2F15.5)
   10 CONTINUE
      RETURN
      END
C$ENTRY
C$STOP
```

APPENDIX B

BASIC ROTOR BALANCING PROGRAM LISTING

```
100 REM ************************************************************
110 REM
120 REM SYSTEM ID. DVF 2 - IEEE - TEK 4051 ROTOR BALANCING.
130 REM PROGRAM ID. BALANCING -- MULTI-PLANE, MULTI-PROBE, MULTI-SPEED
140 REM AUTHOR: ALAN PALAZOLLO - TRANSLATED BY COLIN ARCHIBALD.
150 REM DATE: JULY - 1981
160 REM LOCATION: NATIONAL RESEARCH COUNCIL OF CANADA
170 REM           OTTAWA, CANADA
180 REM ABSTRACT: THIS PROGRAM USES THE INFLUENCE COEFICIENT METHOD
190 REM            OF MULTIPLANE BALANCING.  INPUT DATA CAN BE COLLECTE
200 REM           DIRECTLY FROM THE DVF 2 OF INPUT ONTO A TAPE FILE
210 REM           MANUALLY.
220 REM
230 REM ************************************************************
240 SET RADIANS
250 DIM R1(6,2),S1(6,2),S2(6,2)
260 DIM D1(6,2),D2(6,2),D3(6,2),D4(6,2),D5(6,2),D6(6,2),D7(6,2)
270 DIM D8(6,2),D9(6,2),E1(6,2),E2(6,2),E3(6,2),B(6,2),C1(12,6),C2(12,6)
280 PAGE
290 PRINT "... BALANCING ROUTINE"
300 PRINT "...PLEASE INSERT THE DATA TAPE NOW. HIT RETURN WHEN READY."
310 INPUT C$
320 PAGE
330 FIND 1
340 READ @33:N,L,K,L8
350 PRINT "PLANES",N
360 PRINT "PROBES",L
370 PRINT "SPEEDS",K
380 PRINT "EVEN PROBES",L8
390 FIND 2
400 PRINT @37,26:1
410 M=L*K
420 PRINT @40:".....          MULTI-PLANE MULTI-SPEED BALANCING "
430 PRINT @40:"          ================================..."
440 PRINT @40:"NUMBER OF PLANES:",N
450 PRINT @40:"NUMBER OF PROBES:",L
460 PRINT @40:"NUMBER OF SPEEDS:",K
470 PRINT @40:"NUMBER OF MEASUREMENTS:",M
480 PRINT @37,26:1
490 PRINT @40:"..RUNOUT DATA.......AMPLITUDE...DEGREES"
500 PRINT @37,26:0
510 IMAGE 8A,3D,6A,3D.2D,11D
520 FOR I=1 TO L
530 READ @33:P,A
540 PRINT @37,26:1
550 PRINT @40: USING 510:"PROBE ",I,"    ",A,P
560 PRINT @37,26:0
570 P=P/57.2958
580 A1=A*COS(P)
590 A2=-A*SIN(P)
600 R1(I,1)=A1
610 R1(I,2)=A2
620 NEXT I
```

```
630 REM INITIAL PROBE DATA...
640 PRINT @37,26:1
650 PRINT @40:"  INITIAL READINGS...AMPLITUDE....DEGREES"
660 PRINT @37,26:0
670 IMAGE 7A,2D,7A,2D,6D.2D,7D.1D
680 FOR I=1 TO K
690 FOR J=1 TO L
700 READ @33:P3,A3
710 PRINT @37,26:1
720 PRINT @40: USING 670:"SPEED: ",I," PROBE:",J,A3,P3
730 PRINT @37,26:0
740 P3=P3/57.2958
750 A4=A3*COS(P3)
760 A5=-A3*SIN(P3)
770 IF I=2 THEN 810
780 S1(J,1)=A4
790 S1(J,2)=A5
800 GO TO 830
810 S2(J,1)=A4
820 S2(J,2)=A5
830 NEXT J
840 NEXT I
850 REM COLLECT FINAL UNBALANCE DATA......
860 REM ASSUMING THIS ORDER .. PROBES .. SPEEDS .. UNBALANCE..
870 PRINT @37,26:1
880 PRINT @40:"  FINAL UNBALANCE DATA          AMPLITUDE    ANGLE"
890 FOR I1=1 TO N
900 PRINT @40: USING 910:"PLANE: ",I1
910 IMAGE L,8A,3D
920 FOR I2=1 TO K
930 FOR I3=1 TO L
940 PRINT @37,26:0
950 READ @33:P4,A6
960 PRINT @37,26:1
970 PRINT @40: USING 980:"SPEED: ",I2," PROBE:",I3,A6,P4
980 IMAGE 7A,2D,9A,2D,12D.2D,9D
990 P4=P4/57.2958
1000 T1=A6*COS(P4)
1010 T2=-A6*SIN(P4)
1020 REM INTERESTING DATA STRUCTURES.....
1030 REM SEE VARIABLE TABLES IN PROGRAM DOCUMENTATION.
1040 GO TO I1 OF 1050,1130,1210,1290,1370,1450,1510
1050 REM PLANE 1
1060 IF I2=2 THEN 1100
1070 D1(I3,1)=T1
1080 D1(I3,2)=T2
1090 GO TO 1510
1100 D2(I3,1)=T1
1110 D2(I3,2)=T2
1120 GO TO 1510
1130 REM PLANE 2
1140 IF I2=2 THEN 1180
1150 D3(I3,1)=T1
```

```
1160 D3(I3,2)=T2
1170 GO TO 1510
1180 D4(I3,1)=T1
1190 D4(I3,2)=T2
1200 GO TO 1510
1210 REM PLANE 3...
1220 IF I2=2 THEN 1260
1230 D5(I3,1)=T1
1240 D5(I3,2)=T2
1250 GO TO 1510
1260 D6(I3,1)=T1
1270 D6(I3,2)=T2
1280 GO TO 1510
1290 REM PLANE 4....
1300 IF I2=2 THEN 1340
1310 D7(I3,1)=T1
1320 D7(I3,2)=T2
1330 GO TO 1510
1340 D8(I3,1)=T1
1350 D8(I3,2)=T2
1360 GO TO 1510
1370 REM PLANE 5.....
1380 IF I2=2 THEN 1420
1390 D9(I3,1)=T1
1400 D9(I3,2)=T2
1410 GO TO 1510
1420 E1(I3,1)=T1
1430 E1(I3,2)=T2
1440 GO TO 1510
1450 REM PLANE 6......
1460 E2(I3,1)=T1
1470 E2(I3,2)=T2
1480 GO TO 1510
1490 E3(I3,1)=T1
1500 E3(I3,2)=T2
1510 NEXT I3
1520 NEXT I2
1530 NEXT I1
1540 PRINT @37,26:0
1550 PAGE
1560 PRINT "PLEASE ENTER THE REST OF THE DATA AS REQUESTED."
1570 PRINT @40:"...UNBALANCE TEST WEIGHTS AND ANGLES."
1580 FOR I=1 TO N
1590 PRINT "..FOR BALANCE PLANE ",I," ENTER WEIGHT: "
1600 INPUT W
1610 PRINT "SAME PLANE, ANGLE (MEASURED IN DIRECTION OF ROTATION"
1620 PRINT "                    FROM THE KEY PROBE MARKER):"
1630 INPUT U
1640 PRINT @37,26:1
1650 PRINT @40: USING 1660:"PLANE: ",I,"  WEIGHT: ",W,"  ANGLE: ",U
1660 IMAGE   7A,2D,10A, 3D.3D,12A,4D.2D
1670 PRINT @37,26:0
1680 U=U/57.2958
```

```
1690 A1=W*COS(U)
1700 A2=W*SIN(U)
1710 B(I,1)=A1
1720 B(I,2)=A2
1730 PAGE
1740 NEXT I
1750 X1=1
1760 X2=1
1770 REM DEFINE INFLUENCE COEFICIENT MATRIX....
1780 FOR I=1 TO M
1790 FOR J=1 TO N
1800 IF X2=2 THEN 2060
1810 GO TO J OF 1820,1860,1900,1940,1980,2020
1820 T4=D1(X1,1)-S1(X1,1)
1830 T5=D1(X1,2)-S1(X1,2)
1840 GOSUB 2400
1850 GO TO 2310
1860 T4=D3(X1,1)-S1(X1,1)
1870 T5=D3(X1,2)-S1(X1,2)
1880 GOSUB 2400
1890 GO TO 2310
1900 T4=D5(X1,1)-S1(X1,1)
1910 T5=D5(X1,2)-S1(X1,2)
1920 GOSUB 2400
1930 GO TO 2310
1940 T4=D7(X1,1)-S1(X1,1)
1950 T5=D7(X1,2)-S1(X1,2)
1960 GOSUB 2400
1970 GO TO 2310
1980 T4=D9(X1,1)-S1(X1,1)
1990 T5=D9(X1,2)-S1(X1,2)
2000 GOSUB 2400
2010 GO TO 2310
2020 T4=E2(X1,1)-S1(X1,1)
2030 T5=E2(X1,2)-S1(X1,2)
2040 GOSUB 2400
2050 GO TO 2310
2060 REM CONTINUE....SPEED 2
2070 GO TO J OF 2080,2120,2160,2200,2240,2280
2080 T4=D2(X1,1)-S2(X1,1)
2090 T5=D2(X1,2)-S2(X2,2)
2100 GOSUB 2400
2110 GO TO 2310
2120 T4=D4(X1,1)-S2(X1,1)
2130 T5=D4(X1,2)-S2(X1,2)
2140 GOSUB 2400
2150 GO TO 2310
2160 T4=D6(X1,1)-S2(X1,1)
2170 T5=D6(X1,2)-S2(X1,2)
2180 GOSUB 2400
2190 GO TO 2310
2200 T4=D8(X1,1)-S2(X1,1)
2210 T5=D8(X1,2)-S2(X1,2)
```

```
2220 GOSUB 2400
2230 GO TO 2310
2240 T4=E1(X1,1)-S2(X1,1)
2250 T5=E1(X1,2)-S2(X1,2)
2260 GOSUB 2400
2270 GO TO 2310
2280 T4=E3(X1,1)-S2(X1,1)
2290 T5=E3(X1,2)-S2(X1,2)
2300 GOSUB 2400
2310 REM CONTINUE
2320 NEXT J
2330 X1=X1+1
2340 IF X1<=L THEN 2360
2350 X1=1
2360 IF X1<>1 THEN 2380
2370 X2=X2+1
2380 NEXT I
2390 GO TO 2430
2400 C1(I,J)=(T4*B(J,1)+T5*B(J,2))/(B(J,1)^2+B(J,2)^2)
2410 C2(I,J)=(B(J,1)*T5-T4*B(J,2))/(B(J,1)^2+B(J,2)^2)
2420 RETURN
2430 PRINT "REPLACE THE SOFTWARE TAPE NOW. HIT RETURN   ."
2440 INPUT C$
2450 PAGE
2460 PRINT "THIS IS THE BALANCING ROUTINE. WAIT FOR RESULTS."
2470 REM INTERFACE BETWEEN PART 1 AND PART 2
2480 FIND 7
2490 WRITE @33:N,L,M,K
2500 FOR I=1 TO L
2510 WRITE @33:R1(I,1),R1(I,2)
2520 NEXT I
2530 FOR I=1 TO M
2540 FOR J=1 TO N
2550 WRITE @33:C1(I,J),C2(I,J)
2560 NEXT J
2570 NEXT I
2580 FOR I=1 TO K
2590 FOR J=1 TO L
2600 IF I=2 THEN 2630
2610 WRITE @33:S1(J,1),S1(J,2)
2620 GO TO 2640
2630 WRITE @33:S2(J,1),S2(J,2)
2640 NEXT J
2650 NEXT I
2660 FIND 6
2670 OLD
2680 END
```

```
50 REM*************************************************************************
55 REM
60 REM SYSTEM ID. DVF 2 - IEEE - TEK 4051 ROTOR BALANCING
65 REM PROGRAM ID. BALANCING ROUTINE PART II.
70 REM
100 REM************************************************************************
110 REM INTERFACE WITH PART 1
120 DIM R1(6,2),C1(12,6),C2(12,6)
130 DIM F1(6),F2(6),Z1(6,7),Z2(6,7),C3(6),C4(6)
140 DIM E1(6,7),E2(6,7),S(2),T(2),S1(6,2),S2(6,2)
150 DIM V1(7),V2(7),F8(2,6),F9(2,6)
160 FIND 7
170 READ @33:N,L,M,K
180 FOR I=1 TO L
190 READ @33:R1(I,1),R1(I,2)
200 NEXT I
210 FOR I=1 TO M
220 FOR J=1 TO N
230 READ @33:C1(I,J),C2(I,J)
240 NEXT J
250 NEXT I
260 FOR I=1 TO K
270 FOR J=1 TO L
280 IF I=2 THEN 310
290 READ @33:S1(J,1),S1(J,2)
300 GO TO 320
310 READ @33:S2(J,1),S2(J,2)
320 NEXT J
330 NEXT I
340 REM CONJUGATE THE TRANSPOSE ***************************************
350 FOR I=1 TO N
360 FOR J=1 TO N
370 S(1)=0
380 S(2)=0
390 FOR I1=1 TO M
400 T(1)=C1(I1,I)
410 T(2)=-C2(I1,I)
420 S(1)=S(1)+(C1(I1,J)*T(1)-C2(I1,J)*T(2))
430 S(2)=S(2)+(T(1)*C2(I1,J)+T(2)*C1(I1,J))
440 NEXT I1
450 Z1(I,J)=S(1)
460 Z2(I,J)=S(2)
470 NEXT J
480 NEXT I
490 FOR I=1 TO N
500 S(1)=0
510 S(2)=0
520 X1=1
530 X2=1
540 FOR I1=1 TO M
550 T(1)=C1(I1,I)
560 T(2)=-C2(I1,I)
570 IF X2=2 THEN 630
```

```
580 T4=S1(X1,1)-R1(X1,1)
590 T5=S1(X1,2)-R1(X1,2)
600 S(1)=S(1)+(T(1)*T4-T(2)*T5)
610 S(2)=S(2)+(T4*T(2)+T5*T(1))
620 GO TO 670
630 T4=S2(X1,1)-R1(X1,1)
640 T5=S2(X1,2)-R1(X1,2)
650 S(1)=S(1)+(T(1)*T4-T(2)*T5)
660 S(2)=S(2)+(T4*T(2)+T5*T(1))
670 REM CONTINUE
680 X1=X1+1
690 IF X1<=L THEN 710
700 X1=1
710 IF X1<>1 THEN 730
720 X2=X2+1
730 NEXT I1
740 C3(I)=S(1)
750 C4(I)=S(2)
760 NEXT I
770 FOR I=1 TO N
780 Z1(I,N+1)=C3(I)
790 Z2(I,N+1)=C4(I)
800 NEXT I
810 REM SOLVE ROUTINE....
820 N2=N
830 M2=N+1
840 N9=N
850 FOR I=1 TO N2
860 K9=I
870 T9=ABS(SQR(Z1(K9,I)^2+Z2(K9,I)^2))
880 IF T9>1.0E-6 THEN 920
890 K9=K9+1
900 IF K9-N2<=0 THEN 870
910 IF K9-N2>0 THEN 1230
920 IF I-K9<0 THEN 950
930 IF I-K9=0 THEN 1030
940 IF I-K9>0 THEN 1230
950 FOR M9=1 TO M2
960 T6=Z1(I,M9)
970 T7=Z2(I,M9)
980 Z1(I,M9)=Z1(K9,M9)
990 Z2(I,M9)=Z2(K9,M9)
1000 Z1(K9,M9)=T6
1010 Z2(K9,M9)=T7
1020 NEXT M9
1030 I2=I+1
1040 IF I2>N2 THEN 1200
1050 FOR N9=I2 TO N2
1060 T9=ABS(SQR(Z1(N9,I)^2+Z2(N9,I)^2))
1070 IF T9<1.0E-6 THEN 1190
1080 Q=ABS(SQR(Z1(I,I)^2+Z2(I,I)^2))^2
1090 T4=Z1(N9,I)*Z1(I,I)-Z2(N9,I)*-Z2(I,I)
1100 T5=Z1(I,I)*Z2(N9,I)+-Z2(I,I)*Z1(N9,I)
```

```
1110 T6=T4/Q
1120 T7=T5/Q
1130 FOR M9=1 TO M2
1140 T4=Z1(I,M9)*T6-Z2(I,M9)*T7
1150 T5=T6*Z2(I,M9)+T7*Z1(I,M9)
1160 Z1(N9,M9)=Z1(N9,M9)-T4
1170 Z2(N9,M9)=Z2(N9,M9)-T5
1180 NEXT M9
1190 NEXT N9
1200 NEXT I
1210 REM CONTINUE
1220 GO TO 1280
1230 IF I<>N2 THEN 1250
1240 PRINT @40:"....MATRIX HAS ZERO ROW - - DATA ERROR."
1250 IF I=>N2 THEN 1270
1260 PRINT @40:"....MATRIX HAS ZERO COLUMN - - DATA ERROR."
1270 STOP
1280 REM START BACK SUBSTITUTIONS LABEL 300
1290 I=N2
1300 T6=0
1310 T7=0
1320 IF I=N2 THEN 1400
1330 J3=I+1
1340 FOR J=J3 TO N2
1350 T4=Z1(I,J)*V1(J)-Z2(I,J)*V2(J)
1360 T5=V1(J)*Z2(I,J)+V2(J)*Z1(I,J)
1370 T6=T6+T4
1380 T7=T7+T5
1390 NEXT J
1400 Q=ABS(SQR(Z1(I,I)^2+Z2(I,I)^2))^2
1410 T4=T6+Z1(I,M2)
1420 T5=T7+Z2(I,M2)
1430 V1(I)=(-T4*Z1(I,I)-T5*Z2(I,I))/Q
1440 V2(I)=(Z1(I,I)*-T5+Z2(I,I)*T4)/Q
1450 I=I-1
1460 IF I<=0 THEN 1480
1470 IF I>0 THEN 1300
1480 REM END OF SOLVE SUBROUTINE.....
1490 REM THE LAST PART
1500 PRINT @37,26:1
1510 PRINT @40:"..CORRECTION UNBALANCES...MAGNITUDE....ANGLE IN DEGREES'
1520 PRINT @40:"      (MEASURED IN DIRECTION OF ROTATION)"
1530 FOR I=1 TO N
1540 IF V1(I)<>0 THEN 1600
1550 IF V2(I)<0 THEN 1580
1560 A3=90
1570 GO TO 1710
1580 A3=270
1590 GO TO 1710
1600 IF V1(I)<>0 AND V2(I)<>0 THEN 1630
1610 A3=0
1620 GO TO 1710
1630 IF V1(I)=>0 THEN 1660
```

```
1640 A3=ATN(V2(I)/V1(I))*57.2958+180
1650 GO TO 1700
1660 IF V2(I)<0 THEN 1690
1670 A3=ATN(V2(I)/V1(I))*57.2958
1680 GO TO 1700
1690 A3=ATN(V2(I)/V1(I))*57.2958+360
1700 REM CONTINUE
1710 REM CONTINUE
1720 A4=ABS(SQR(V1(I)^2+V2(I)^2))
1730 IF A3=>0 THEN 1760
1740 A3=A3+360
1750 IMAGE 7A,2D,12A,3D.3D,8A,8D.2D
1760 PRINT @40: USING 1750:"PLANE:",I,"  MAGNITUDE:",A4,"  ANGLE:",A3
1770 NEXT I
1780 X1=1
1790 X2=1
1800 FOR I=1 TO M
1810 S(1)=0
1820 S(2)=0
1830 FOR I1=1 TO N
1840 S(1)=S(1)+(C1(I,I1)*V1(I1)-C2(I,I1)*V2(I1))
1850 S(2)=S(2)+(V1(I1)*C2(I,I1)+V2(I1)*C1(I,I1))
1860 NEXT I1
1870 IF X2=2 THEN 1910
1880 S(1)=S(1)+S1(X1,1)
1890 S(2)=S(2)+S1(X1,2)
1900 GO TO 1930
1910 S(1)=S(1)+S2(X1,1)
1920 S(2)=S(2)+S2(X1,2)
1930 REM CONTINUE
1940 F8(X2,X1)=S(1)
1950 F9(X2,X1)=S(2)
1960 X1=X1+1
1970 IF X1<=L THEN 1990
1980 X1=1
1990 IF X1<>1 THEN 2010
2000 X2=X2+1
2010 NEXT I
2020 PRI @40:"     RESIDUAL DISPLACEMENTS WITH CORRECTION WEIGHTS ADDED'
2030 PRINT @40:"                INCLUDING RUNOUT."
2040 REM LABEL 450
2050 Q1=0
2060 FOR I=1 TO K
2070 IMAGE L,6A,3D
2080 PRINT @40: USING 2070:"SPEED ",I
2090 FOR J=1 TO L
2100 A1=F8(I,J)
2110 A2=-F9(I,J)
2120 IF A1<>0 THEN 2150
2130 A3=0
2140 GO TO 2220
2150 IF A1=>0 THEN 2180
2160 A3=ATN(A2/A1)*57.2958+180
```

```
2170 GO TO 2220
2180 IF A2<0 THEN 2210
2190 A3=ATN(A2/A1)*57.2958
2200 GO TO 2220
2210 A3=ATN(A2/A1)*57.2958+360
2220 REM CONTINUE
2230 REM CONTINUE
2240 A4=ABS(SQR(F8(I,J)^2+F9(I,J)^2))
2250 IF A3=>0 THEN 2280
2260 A3=A3+360
2270 IMAGE 6A,2D,11A,3D.3D,17A,4D.2D
2280 PRI @40: USI 2270:"PROBE ",J,"MAGNITUDE ",A4,"LAG ANGLE(DEG) ",A3
2290 Q1=Q1+A4^2
2300 NEXT J
2310 NEXT I
2320 PRINT @40:"..... THE SUM OF THE AMPLITUDE SQUARES OF THE RESIDUAL'
2330 IMAGE 24A,3E
2340 PRINT @40: USING 2330:"      DISPLACEMENTS IS:",Q1
2350 PRINT "....RUN COMPLETE"
2360 PRINT @37,26:0
2370 END
```