GENERAL UTILITY ROUTINES

Part No. 09825-10001

Rev. E

Printed in U.S.A.

Table of Contents

		PAGE
FILE DIRECTORY		3
PART 1	UTILITY ROUTINES	5
SECTION 1	MATHEMATICAL UTILITY ROUTINES. . . .	6
SECTION 2	SORTING AND BUFFERED OUTPUT UTILITY ROUTINES.	225
SECTION 3	PLOTTING AND TEXT-GENERATING ROUTINES FOR THE 9871A PRINTER. .	281
SECTION 4	THE "dot" FUNCTION FOR THE 9866B . .	365
PART 2	TRAINING TAPE	375

HP Computer Museum
www.hpmuseum.net

For research and education purposes only.

FILE DIRECTORY

TRK. NO.	FILE NO.	CONTENTS
0	0	Program Directory
0	1	Driver for File 2
0	2	Subroutine - Complex Number Operations (+,-, *,÷,↑)
0	3	Driver for File 4
0	4	Subroutine - Matrix Operations (+,-,*,scalar*)
0	5	Driver for File 6
0	6	Subroutine - Matrix Inversion
0	7	Driver for File 8
0	8	Subroutine - Matrix Inversion (Using Matrix ROM)
0	9	Driver for File 10
0	10	N Simultaneous Equations in N Unknowns (For Accuracy)
0	11	N Simultaneous Equations in N Unknowns (For Space)
0	12	Driver for File 13
0	13	Numerical Integration of User-Defined Function (Simpson)
0	14	Driver for File 15
0	15	Numerical Integration of User-Defined Function (Romberg)
0	16	Driver for File 17
0	17	Numerical Integration of Equally-Spaced Data (Simpson)
0	18	Driver for File 19
0	19	Numerical Integration of Discrete Data (Greville)
0	20	Driver for File 21
0	21	Numerical Integration, Differetiation, and Interpolation
0	22	Driver for File 23
0	23	Numerical Differentiation of Discrete Data (Greville)
0	24	Driver for File 25
0	25	Fourier Series Coefficients of Equally Spaced Data
0	26	Driver for File 27
0	27	Fourier Series Coefficients of Discrete Data

TRK. NO.	FILE NO.	CONTENTS
0	28	Driver for File 29
0	29	Real Roots of F(X)=0 on an Interval (Bisection)
0	30	Driver for File 31
0	31	Complex Roots of Complex Polynomial
0	32	Driver for Files 33 and 34
0	33	Subroutine - String Sort "QSORT"
0	34	Subroutine - String Sort "SSORT"
0	35	Driver for File 36
0	36	Subroutine - Array Sort on Index
0	37	Driver for File 38
0	38	Subroutine - Buffered I/O for Plotter
0	39	Data for Program in File 37
0	40	Plotting Subroutines for 9871A Printer
0	41	Text-Generating Subroutines for 9871A Printer
0	42	Program Example (Sine Curve) for 9871A Printer
0	43	Program Example (Employee Records) for 9871A Printer
0	44	Program Example (4-Leafed Rose) for 9871A Printer
0	45	Binary Program for dot Function
0	46	Plotting Program Example for 9866B
1	0-11	Training Tape

UTILITY ROUTINES - PART 1

SECTION 1	MATHEMATICAL ROUTINES
SECTION 2	SORTING AND BUFFER ROUTINES
SECTION 3	9871A PLOTTER AND TEXT-GENERATING ROUTINES
SECTION 4	9866B PLOTTING WITH dot FUNCTION

The routines in this part of the UTILITY ROUTINES cartridge have been written to provide the users of the 9825A with commonly requested programs. The routines can also be used to become better acquainted with programming the 9825A.

A conscious effort has been made to divide each of the programs in Sections 1 and 2 into two distinct parts - the driver program and the subroutine. The driver is used to input the information necessary for the subroutine and to output the results computed by the subroutine. The subroutines have been written so they can be included as part of any user program as long as that program provides the necessary input information. This allows the more industrious programmers to write their own drivers to use the subroutines while other programmers can use the whole driver-subroutine package.

A detailed explanation of the programs in Section 3 and Section 4 is included in the introductions to those sections.

Section 1

MATHEMATICAL ROUTINES

PROGRAM	FILES	PAGE
Complex Number Operations	1,2	7
Matrix Operations	3,4	19
Matrix Inversion	5,6	37
Simultaneous Equations Using Matrix ROM	7,8	51
Maximum Accuracy	9,10	63
Maximum Size	11	75
Numerical Integration		
User-Defined Function (Simpson)	12,13	83
User-Defined Function (Romberg)	14,15	95
Equally Spaced Data Points	16,17	109
Unequally Spaced Data Points	18,19	121
Numerical Differentiation, Integration and Interpolation of Unequally Spaced Data Points	20,21	133
Numerical Differentiation of Unequally Spaced Data Points	22,23	149
Fourier Series Coefficients		
Equally Spaced Data Points	24,25	165
Unequally Spaced Data Points	26,27	179
Rootfinder		
Iterative Rootfinder of User-Defined Function	28,29	197
Polynomial Rootfinder with Complex Coefficients	30,31	211

All files are on track 0

Complex Numbers

FILE 1 DRIVER

FILE 2 SUBROUTINE FOR COMPLEX NUMBER OPERATIONS

Complex Number Operations

This program will add, subtract, multiply or divide two complex numbers as well as raising a complex number to an integer power using the subroutine "COMPLEX". The complex numbers are expressed in the form a + bi, where a is the real part and b the imaginary part. The components of the complex numbers (a and b) should be real numbers (e.g. 5, 2.1436, π) or expressions whose values are real numbers (e.g. 1/2, $\sqrt{3}$, sin (π/4)).

SPECIAL CONSIDERATIONS AND DIAGNOSTIC MESSAGES:

1. "DIV BY ZERO" occurs if the operation is division (3) and the divisor is 0 + 0i.
2. "ILLEGAL OPERATOR" occurs if operation is not 0, 1, 2, 3, or 4.

FORMULAE:

1. $(a + bi)\ (c + di) = (ac - bd) + (ad + bc)\ i$

2. $$\frac{a + bi}{c + di} = \frac{ac + bd}{cc + dd} + \frac{bc - ad}{cc + dd}\ i$$

3. $$(a + bi)^n = \underbrace{(a + bi)\ (a + bi)\ \ldots\ (a + bi)}_{n} \quad \text{for } n > 0$$

$$(a + bi)^n = 1 \quad \text{for } n = 0$$

$$(a + bi)^n = \frac{1}{\underbrace{(a + bi)\ (a + bi)\ \ldots\ (a + bi)}_{n}} \quad \text{for } n < 0$$

INSTRUCTIONS:

1. Insert UTILITY ROUTINES cartridge with machine on.

2. Load the file.
   a. Type ldf 1
   b. Press EXECUTE

3. (Optional)
   a. Type fxd N or flt N where N is the number of digits to the right of the decimal point ($0 \leq N \leq 11$).
   b. Press EXECUTE

4. Press RUN

5. "ENTER COMPLEX NO" will appear briefly in the display.
   "FIRST" is printed and "REAL PART =" is printed as well as displayed.
   Either
   Press CONTINUE if you want to stop the program.
   or
   a. Enter the real part of the first number.
   b. Press CONTINUE

6. When "IMAG PART =" is displayed:
   a. Enter the imaginary part of the first number.
   b. Press CONTINUE

7. "OPERATION?" will appear briefly in the display. When "A = 0, S = 1, M = 2, D = 3, P = 4" is displayed:
   a. Enter the desired operation code.
      0 for addition
      1 for subtraction (1^{st} minus 2^{nd})
      2 for multiplication
      3 for division (1^{st} divided by 2^{nd})
      4 for exponentiation (1^{st} raised to the power of 2^{nd})
   b. Press CONTINUE

8. If the operation is not exponentiation (4), go to step 10.

9. When "EXPONENT" is displayed:

   a. Enter integer exponent.

   b. Press CONTINUE

   c. Go to step 12

10. "SECOND" is printed. When "REAL PART =" is displayed:

   a. Enter the real part of the second number.

   b. Press CONTINUE

11. When "IMAG PART =" is displayed:

   a. Enter the imaginary part of the second number.

   b. Press CONTINUE

12. The resultant complex number will be printed in the form.

   REAL PART = _____

   IMAG PART = _____

   If an error has occured the error message will be printed (See description of error messages, page 9).

13. Go to step 5.

EXAMPLES:

(1+2i) + (1+2i) = 2+4i

```
ENTER COMPLEX NO

FIRST

REAL PART=
1

IMAG PART=
2

OPERATION?
A=0,S=1,M=2,D=3,
P=4
0

SECOND

REAL PART=
1

IMAG PART=
2

----------------

SUM IS

REAL PART=   2.00
IMAG PART=   4.00
```

$(1+2i)^2 = -3+4i$

```
FIRST

REAL PART=
1

IMAG PART=
2

OPERATION?
A=0,S=1,M=2,D=3,
P=4
4

EXPONENT?
2

----------------

1ST RAISED TO
POWER OF 2ND

REAL PART= -3.00
IMAG PART=  4.00
```

$\frac{1+2i}{0+0i}$ = undefined

```
FIRST

REAL PART=
1

IMAG PART=
2

OPERATION?
A=0,S=1,M=2,D=3,
P=4
3

SECOND

REAL PART=
0

IMAG PART=
0
DIV BY ZERO
```

"COMPLEX" SUBROUTINE

This subroutine will add, subtract, multiply or divide two complex numbers as well as raising a complex number to an integer power. The complex numbers are expressed in the form a + bi where a is the real part and b is the imaginary part . The components of the complex numbers (a and b) should be real numbers (e.g. 5, 2.1413, π) or expressions whose values are real numbers (e.g. 1/2, $\sqrt{3}$, sin (π/4)). When used as a subroutine, all parameters specified under <INPUT> must be assigned in the user's calling program. The user may then append the subroutine to his program and use gsb "COMPLEX" to take advantage of its capabilities. The results of the calculation and any other output is designated under <OUTPUT>. The user should be aware that the subroutine may alter variables that he is using in his program and should therefore check the variables listed under <DESTROYED>. An example of a typical program which uses this subroutine can be found on page 16 or in file 1.

PARAMETERS AND VARIABLES:

<INPUT>

A	real part of first number
B	imaginary part of first number
C	real part of second number (exponent if raising to power)
D	imaginary part of second number
X	operation to be performed

0 = addition
1 = subtraction
2 = multiplication

3 = division

4 = exponent

<OUTPUT>

A, B, C, D, X unchanged

E	real part of result
F	imaginary part of result
flg 2	set if division by zero occurs
flg 3	set if illegal operation

<DESTROYED>

G	sum of squares of real and imaginary part of division
H	temporary storage for partial product in computing $(A + Bi)^C$
I	absolute value of integer exponent (if any)
flg 1	set to indicate negative integer exponent

INSTRUCTIONS:

1. Load your program into memory. The program should set up all necessary parameters (A, B, C, D, X) to be passed to the subroutine.

2. Insert UTILITY ROUTINES cartridge.

3. (Optional)
   a. Type `fxd N` or `flt N` where N is the number of digits printed to the right of the decimal point ($0 \leq N \leq 11$).
   b. Press EXECUTE

4. Load the file.
   a. Type `ldf 2, N, M` where:
      N = line number where the first line of the subroutine should be loaded into memory.
      M = line number where execution is to begin after subroutine is loaded.
   b. Press EXECUTE

ANNOTATED LISTING:

DRIVER - COMPLEX NUMBERS

```
0: ldf 2,19,1                    Load the subroutine
1: cfg 13;dsp                    First Number
 "ENTER COMPLEX
 NO";wait 500
2: spc 3;prt
 "FIRST";spc ;                   Enter real part
 enp "REAL PART=
 ",A;spc ;if
 flg13;stp
3: enp "IMAG                     Enter imaginary part
 PART=",B;spc ;
 if flg13;cfg
 13;gto +0
4: dsp "OPERATIO
 N?";wait 500                    Enter operation
5: enp "A=0,S=1,
 M=2,D=3,P=4",X;
 spc ;if flg13;
 cfg 13;gto +0
6: if X=4;enp                    Enter exponent if necessary
 "EXPONENT?",C;
 0→D;gto +4;if
 flg13;cfg 13;
 gto +0
7: prt "SECOND";                 Second Number
 spc
8: enp "REAL                     Enter real part
 PART=",C;spc ;
 if flg13;cfg
 13;gto +0
9: enp "IMAG                     Enter imaginary part
 PART=",D;if
 flg13;cfg 13;
 gto +0
10: gsb "COMPLEX                 Branch to subroutine
 "
11: if flg2 or
 flg3;wait 2000;                 Check if error flags are set
 gto -9
12: spc 2;prt "-
 ---------------
 ";spc 2;jmp X+1
13: prt "SUM
 IS";gto +5
14: prt "DIFFERE                 Print label
 NCE IS";gto +4
15: prt "PRODUCT
  IS";gto +3
16: prt "QUOTIEN
 T IS";gto +2
17: prt "1ST
 RAISED TO","POW
 ER OF 2ND"
                                 Print result
```

```
18: spc 2;prt
 "REAL PART=",E,
 "IMAG PART=",F;          Print result
 spc 3;sto -16
*31876
```

ANNOTATED LISTING:

SUBROUTINE - COMPLEX NUMBERS

```
0: "COMPLEX":cfg
  2,3;if X>-1
 and X<5;jmp X+2
1: beep;dsp "ILL
 EGAL OPERATOR";
 sfg 2;ret
```
Check for illegal operation

```
2: A+C→E;B+D→F;
 ret
```
Addition

```
3: A-C→E;B-D→F;
 ret
```
Subtraction

```
4: AC-BD→E;AD+
 BC→F;ret
```
Multiplication

```
5: CC+DD→G;gto +
 7
```
Division

```
6: 1→G;A→E;B→F;
 cfg 1
7: if (C→I)<0;
 sfg 1;abs(C)→I
8: if C=0;1→E;
 0→F;ret
9: A(E→H)-BF→E;
 AF+BH→F;jmp (G+
 1→G)=I
10: if flg1;E/
 (EE+FF→H)→E;-F/
 H→F
11: ret
```
Exponentiation

```
12: if G=0;beep;
 dsp "DIV BY
 ZERO";sfg 3;
 ret
13: (AC+BD)/G→E;
 (BC-AD)/G→F;
 ret
```
Division

```
14: end
*8724
```

Matrix Operations

FILE 3 DRIVER

FILE 4 SUBROUTINE TO PERFORM MATRIX OPERATIONS

Matrix Operations For Real Matrices

This program will compute the sum, difference or product of two matrices as well as compute the product of a matrix and a scalar. The program requests all the necessary input information, calls the subroutine "MATRIX" in file 4 and prints out the results.

The elements of the matrices should be real numbers (e.g. 5, 2.1436, π) or expressions whose values are real numbers (e.g. 1/2, $\sqrt{3}$, sin (π/4)).

SPECIAL CONSIDERATIONS AND DIAGNOSTIC MESSAGES:

1. The maximum sized matrices for which the subroutine will run depends on the size of the calling program. The following limitations assume that the calling program in file 3 is used.
   a. For addition and subtraction: If the dimensions of A and B are N by M then:

for 6,844 bytes	N * M < 188
for 15,036 bytes	N * M < 529
for 23,228 bytes	N * M < 871
for 31,420 bytes	N * M < 1212

   b. For scalar multiplication: If the dimensions of A are N x M then:

for 6,844 bytes	N * M < 283
for 15,036 bytes	N * M < 795
for 23,228 bytes	N * M < 1307
for 31,420 bytes	N * M < 1819

c. For multiplication: If the dimensions of A are D by E and of B are N by M then:

for 6,844 bytes	(D * E + N * M + Y * Z) < 188
for 15,036 bytes	(D * E + N * M + Y * Z) < 529
for 23,228 bytes	(D * E + N * M + Y * Z) < 870
for 31,420 bytes	(D * E + N * M + Y * Z) < 1212

Where Y * Z are the dimensions of teh resulting matrix.

2. "ILLEGAL SIZES" indicates that the operation cannot be performed with the present matrix dimensions.

3. "ILLEGAL OP" indicates that the operation is not 0, 1, 2, or 3.

INSTRUCTIONS:

1. Insert UTILITY ROUTINES cartridge with machine on.

2. Load the file.
   a. Type ldf 3
   b. Press EXECUTE

3. (Optional)
   a. Type fxd N or flt N where N is the number of digits that will be printed to the right of the decimal point ($0 \leq N \leq 11$).
   b. Press EXECUTE

4. Press RUN

5. "MATRIX A" will appear briefly in the display. When "NO. OF ROWS?" is displayed:
   a. Enter the number of rows in the first matrix.
   b. Press CONTINUE

6. When "NO OF COLS?" is displayed:
   a. Enter the number of columns in the first matrix.
   b. Press CONTINUE

7. "A[I, J]" will appear in the display, where "I" will be the row number and "J" column number of the next matrix element to be entered.
   a. Enter the element in the I^{th} row and J^{th} column of the first matrix.
   b. Press CONTINUE
   c. Repeat step 7 until all elements of the first matrix have been entered.

8. When "CHANGES" is displayed:
   Either
   a. Press CONTINUE if you do not want to make changes to entered data.
   b. Go to step 12.
   or
   a. Enter 1 if you need to make changes to entered data.
   b. Press CONTINUE

9. When "ROW?" is displayed:
   a. Enter the row number (I), of the element to be changed.
   b. Press CONTINUE

10. When "COLUMN" is displayed:
   a. Enter the column number (J) of the element to be changed.
   b. Press CONTINUE

11. When "A[I, J]" is displayed:
   a. Enter the correct value for this element.
   b. Press CONTINUE
   c. Go to step 8.

12. "OPERATION" will appear briefly in the display. When "A = 0, S = 1, M = 2, SM = 3" is displayed:
   a. Enter the appropriate code for the desired operation:
      0 for addition
      1 for subtraction
      2 for multiplication
      3 for scalar multiplication
   b. Press CONTINUE

13. If the operation is not scalar multiplication (3), go to step 14. When "SCALAR VALUE" is displayed:
   a. Enter the scalar multiplier.
   b. Press CONTINUE
   c. Go to step 21.

14. If the operation is addition or subtraction, go to step 16. "MATRIX B" will appear briefly in the display. When "NO. OF ROWS?" is displayed:
   a. Enter the number of rows in the second matrix.
   b. Press CONTINUE

15. When "NO. OF COLS?" is displayed:
   a. Enter the number of columns in the second matrix.
   b. Press CONTINUE

16. "B[I, J]" will appear in the display, where "I" and "J" will be equivalent to the row and column number of the next matrix element to be entered.
    a. Enter the element in the I^{th} row and J^{th} column of the second matrix.
    b. Press CONTINUE
    c. Repeat step 16 until all the elements of the second matrix have been entered.

17. When "CHANGES?" is displayed:
    Either
    a. Press CONTINUE if you do not want to make changes.
    b. Go to step 21.
    or
    a. Enter 1 if you need to make changes to entered data.
    b. Press CONTINUE

18. When "ROW?" is displayed:
    a. Enter the row number (I) of the element to be changed.
    b. Press CONTINUE

19. When "COLUMN?" is displayed:
    a. Enter the column number (J) of the element to be changed.
    b. Press CONTINUE

20. When "B[I,J]" is displayed:
    a. Enter the correct value for this element.
    b. Press CONTINUE
    c. Go to step 17.

21. If the operation is addition (0), go to step 22. When "REVERSE ORDER?" is displayed:
    Either
    Press CONTINUE if you want the order of the operation to be A<OP>B, where A is the first matrix and B the second.
    or
    a. Enter 1 if you want the order to be B<OP>A.
    b. Press CONTINUE

22. "A<OP>B" will be printed where <OP> corresponds to the appropriate operation and the resultant matrix will be printed by columns.

EXAMPLES:

```
MATRIX A

NO.OF ROWS?
2
NO. OF COLS?
2

A[1,1]?
1

A[1,2]?
2

A[2,1]?
3

A[2,2]?
4


CHANGES?


A=0,S=1,M=2,SM=3
2
```

$$\begin{bmatrix} 1 & 2 \\ 3 & 4 \end{bmatrix} \begin{bmatrix} 0 & 1 \\ 1 & 0 \end{bmatrix} = \begin{bmatrix} 2 & 1 \\ 4 & 3 \end{bmatrix}$$

```
MATRIX B

NO OF ROWS?
2
NO OF COLUMNS?
2

B[1,1]?
0

B[1,2]?
1

B[2,1]?
1

B[2,2]?
0


CHANGES?
```

```
REVERSE ORDER?

A*B

BY COLUMNS

                2.00
                4.00

                1.00
                3.00
```

```
MATRIX A

NO.OF ROWS?
2
NO. OF COLS?
3

A[1,1]?
1

A[1,2]?
2

A[1,3]?
3

A[2,1]?
4

A[2,2]?
5

A[2,3]?
6


CHANGES?


A=0,S=1,M=2,SM=3
0



MATRIX B


B[1,1]?
6

B[1,2]?
5

B[1,3]?
4

B[2,1]?
3

B[2,2]?
2

B[2,3]?
1


CHANGES?
```

$$\begin{bmatrix} 1 & 2 & 3 \\ 4 & 5 & 6 \end{bmatrix} + \begin{bmatrix} 6 & 5 & 4 \\ 3 & 2 & 1 \end{bmatrix} = \begin{bmatrix} 7 & 7 & 7 \\ 7 & 7 & 7 \end{bmatrix}$$

```
A+B

BY COLUMNS

      7.00
      7.00

      7.00
      7.00

      7.00
      7.00
```

"MATRIX" SUBROUTINE

This subroutine will compute the sum, difference or product of two matrices as well as computing the product of a matrix and a scalar. The elements of the matrix may be real number (e.g. 5, 2.15, π) or expressions whose values are real numbers (e.g. 1/2, $\sqrt{3}$, sin (π/2)).

When used as a subroutine, all parameters specified under <INPUT> must be assigned in the user's calling program. The user may then append the subroutine to his program and use gsb "MATRIX" to take advantage of its capabilities. The results of the calculation and any other output is designated under <OUTPUT>. The user should be aware that the subroutine may alter variables that he is using in his program and should therefore check the variables listed under <DESTROYED>. An example of a typical program which uses this subroutine may be found on page 32 or in file 3.

PARAMETERS AND VARIABLES USED:

<INPUT>

D	# of rows in first matrix A
E	# of columns in first matrix
N	# of rows in second matrix B
M	# of columns in second matrix
A[]	two dimensional array containing elements of first matrix
B[]	two dimensional array containing elements of second matrix
X	operation to be performed 0 = addition 1 = subtraction

2 = multiplication

3 = scalar multiplication

flg 13 set if order is A<OP>B; clear if order is B<OP>A

rØ scalar multiplier if operation is scalar multiplication

<OUTPUT>

D, E, N, M, A [], B [], X unchanged

Y # of rows in resultant array

Z # of columns in resultant array

C[] resultant array computed from A [] and B []

flg 2 set if dimensions are incompatible

flg 3 set if operation is illegal (i.e., X ≠ 0, 1, 2, 3)

<DESTROYED>

flg 1 set if operation is subtraction

I loop counter; usually subscript for rows

J loop counter; usually subscript for columns

K loop counter

G used to compute partial sum for product

INSTRUCTIONS:

1. Load your program into memory. The program should set up all necessary parameters (A [,], B [,], D, E, N, M, X, flg 13) to be passed to the subroutine.

2. Insert UTILITY ROUTINES cartridge.

3. (Optional)
   a. Type fxd N or flt N where N is the number of digits printed to the right of the decimal point ($0 \leq N \leq 11$).
   b. Press EXECUTE

4. Load the file.
   a. Type ldf 4, N, M where:

      N = line number where the first line of the subroutine should be loaded into memory.

      M = line number where execution is to begin after subroutine is loaded.

   b. Press EXECUTE

ANNOTATED LISTING:

DRIVER - MATRIX OPERATIONS

0: ldf 4,40,1	Load the subroutine
1: cfg 13;dsp "MATRIX A";spc 3;prt "MATRIX A";spc ;wait 500	First Matrix
2: enp "NO.OF ROWS?",D;if flg13;cfg 13; gto +0	Enter number of rows
3: enp "NO. OF COLS?",E;spc ; if flg13;cfg 13;gto +0	Enter number of columns
4: 0→I;1→J;dim A[D,E] 5: if (I+1→I)>D; gto +3 6: cfg 13;enp A[I,J];spc ; gto +0;if not flg13;jmp (J+ 1→J)>E 7: 1→J;gto -2	Enter elements of first matrix
8: spc ;enp "CHA NGES?",I;spc ; if flg13;cfg 13;gto +5 9: enp "ROW?",I; if flg13;cfg 13;gto +0 10: enp "COLUMN? ",J;if flg13; cfg 13;gto +0 11: cfg 13;enp A[I,J];spc ; gto +0;if not flg13;gto -3	Make corrections if necessary
12: dsp "OPERATI ON?";wait 500 13: enp "A=0, S=1,M=2,SM=3", X;spc ;if flg13 ;cfg 13;gto +0	Enter operation code
14: if X=3;enp "SCALAR VALUE?" ,r0;gto +14;if flg13;gto +0	Enter scalar multiplier if necessary
15: dsp "MATRIX B";spc 3;prt "MATRIX B";spc	Second Matrix
16: if X=0 or X=1;D→N;E→M; gto +3	If addition or subtraction use same dimensions

ANNOTATED LISTING:

```
17: enp "NO OF
 ROWS?",N;if
 flg13;cfg 13;
 gto +0
```
Number of rows

```
18: enp "NO OF
 COLUMNS?",M;if
 flg13;cfg 13;
 gto +0
```
Number of columns

```
19: spc ;0→I;
 1→J;dim B[N,M]
20: if (I+1→I)>N
 ;gto +3
21: cfg 13;enp
 B[I,J];spc ;
 gto +0;if not
 flg13;jmp (J+
 1→J)>M
22: 1→J;gto -2
```
Enter elements of second matrix

```
23: spc ;enp
 "CHANGES?",I;
 spc ;if flg13;
 gto +4
24: enp "ROW?",
 I;if flg13;cfg
 13;gto +0
25: enp "COLUMN?
 ",J;if flg13;
 cfg 13;gto +0
26: cfg 13;enp
 B[I,J];spc ;
 gto +0;if not
 flg13;gto -3
```
Make corrections if necessary

```
27: cfg 13;if
 X=1 or X=2;enp
 "REVERSE ORDER?
 ",G
```
Reverse order of matrices in operation?
(i.e. B - A instead of A - B)

```
28: gsb "MATRIX"
```
Branch to subroutine

```
29: if flg2 or
 flg3;stp
```
If error flags are set; stop

```
30: if X=0;prt
 "A+B";gto +6
31: if X=1 and
 flg13;prt "A-
 B";gto +5
32: if X=1 and
 not flg13;prt
 "B-A";gto +4
33: if X=2 and
 flg13;prt "A*
 B";gto +3
34: if X=2 and
 not flg13;prt
 "B*A";gto +2
```
Print heading for resultant matrix

ANNOTATED LISTING:

```
35: prt "A*SCALA
 R"
36: 1→I;0→J;spc
 2;prt "BY COLUM
 NS";spc
37: if (J+1→J)>Z
 ;spc 4;end
38: prt C[I,J];
 jmp (I+1→I)>Y
39: spc ;1→I;
 gto -2
*27401
```

Print resultant matrix by columns

ANNOTATED LISTING:

SUBROUTINE - MATRIX OPERATIONS

```
0: "MATRIX":cfg
 2,3;if X=3;0→N
1: cfg 1;0→I→J;
 1→K
2: if flg13;dim
 C[D,M];D→Y;M→Z;
 gto +2
3: dim C[N,E];
 N→Y;E→Z
4: 0→G;if X#1
 and X#0;gto +9
5: if X=1;sfg 1
6: if D=N and
 E=M;gto +2
7: beep;dsp "ILL
 EGAL SIZES";
 sfg 2;ret
8: if (I+1→I)>D;
 ret
9: if (J+1→J)>E;
 0→J;gto -1
10: if flg1 and
 not flg13;-A[I,
 J]→A[I,J];gto +
 2
11: if flg1;-
 B[I,J]→B[I,J]
12: A[I,J]+B[I,
 J]→C[I,J];gto -
 3
13: if X#2;gto +
 12
14: if flg13;
 gto +6
15: if M#D;gto -
 8
16: if (I+1→I)>N
 ;ret
17: if (J+1→J)>E
 ;0→J;gto -1
18: G+B[I,K]A[K,
 J]→G;jmp (K+
 1→K)>D
19: 1→K;G→C[I,
 J];0→G;gto -2
20: if N#E;gto -
 13
21: if (I+1→I)>D
 ;ret
22: if (J+1→J)>M
 ;0→J;gto -1
23: G+A[I,K]B[K,
 J]→G;jmp (K+
 1→K)>E
```

Determine correct dimensions of output matrix

Set flag for subtraction

Check for compatible dimension sizes

Addition or subtraction

flg 1 set for subtraction
flg 13 set for reverse order

Branch if scalar multiplication

Branch to B*A section

Branch if incompatible dimensions

Multiplication A*B

Branch if incompatible dimensions

Reverse order

Multiplication B*A

ANNOTATED LISTING:

```
24: 1→K;G→C[I,
 J];0→G;gto -2
25: 1→J;if X#3;
 dsp "ILLEGAL
 OP";sfg 3;ret
26: if (I+1→I)>D
 ;ret
27: A[I,J]r0→C[I
 ,J];jmp (J+1→J)
 >E
28: 1→J;gto -2
29: end
*25062
```

Check for illegal operation code

Scalar multiplication

Matrix Inversion

FILE 5 DRIVER

FILE 6 MATRIX INVERSION AND DETERMINANT SUBROUTINE

Matrix Inversion And Determinant Function For Real Matrices

This program will invert a square matrix and compute the determinant using the subroutine "INVERT" in file 6. The resultant matrix may be reinverted to check the accuracy. This should always be done when the determinant is relatively small.

The elements of the matrix may be real numbers (e.g. 5, 2.1436, π) or expressions whose values are real numbers (e.g. 1/2, $\sqrt{3}$, sin (π/4)).

$$\begin{bmatrix} a_{11} & a_{12} & a_{13} & \cdots & a_{1n} \\ a_{21} & a_{22} & a_{23} & \cdots & a_{2n} \\ \cdots & \cdots & \cdots & \cdots & \cdots \\ a_{n1} & a_{n2} & a_{n3} & \cdots & a_{nn} \end{bmatrix}$$

SPECIAL CONSIDERATIONS AND DIAGNOSTIC MESSAGES:

1. The largest matrix that can be inverted by the subroutine is dependent on the user's calling program. If file 5 is used in conjunction with the subroutine, you can invert up to a:

   23 by 23 matrix on the 6,844 byte machine
   39 by 39 matrix on the 15,036 byte machine
   50 by 50 matrix on the 23,228 byte machine
   59 by 59 matrix on the 31,420 byte machine

2. "DET = 0" indicates that the matrix cannot be inverted because the determinant is zero.

3. The time to invert a 4 by 4 matrix is approximately two seconds while a 10 x 10 matrix is about 27 seconds.

4. The inv command can be used in lieu of this routine if you have the matrix ROM.

FORMULAE:

a_{ij} is the element in the i^{th} row and j^{th} column, P_k is the pivot for the k^{th} row, and r_k and c_k represent the permuted order of the rows and columns.

1. Normalization:

$$a_{r_k j}/P_k \rightarrow a_{r_k j} \qquad \text{for } j = 1, 2, \ldots, n+1 \quad j \neq c_k$$

$$1/P_k \rightarrow a_{r_k c_k}$$

where r_k, c_k for $k = 1, n$ represent the column and row permutation vectors.

2. Reduction:

$$a_{ij} - a_{ic_k} \quad a_{r_k j} \rightarrow a_{ij} \qquad \text{for } j = 1, n+1 \quad j \neq c_k \qquad \text{for } i = 1, n \quad i \neq r_k$$

3. Unscramble Solution:

$$\left.\begin{array}{ll} a_{r_i j} \rightarrow y_{c_j} & \text{for } i = 1, n \\ y_i \rightarrow a_{ij} & \text{for } i = 1, n \end{array}\right\} \text{for } j = 1, n$$

$$\left.\begin{array}{ll} a_{ic_j} \rightarrow y_{r_j} & \text{for } j = 1, n \\ y_j \rightarrow a_{ij} & \text{for } j = 1, n \end{array}\right\} \text{for } i = 1, n$$

REFERENCES:

1. Carnahan, B., Luther, H.A., and Wilkes, J.O., Applied Numerical Methods (New York: John Wiley & Sons, 1969), pp. 282 - 284.

INSTRUCTIONS:

1. Insert UTILITY ROUTINES cartridge with machine on.

2. Load the file.
   a. Type in ldf 5
   b. Press EXECUTE

3. (Optional)
   a. Type fxd N or flt N where N is the number of digits to be printed to the right of the decimal point $(0 \leq N \leq 11)$.
   b. Press EXECUTE

4. Press RUN

5. When "NO. OF ROWS?" is displayed:
   a. Enter the number of rows in the matrix (this is also the number of columns since matrix is square).
   b. Press CONTINUE

6. "A[I, J]" will appear in the display, where "I" will be the row number and "J" the column number of the next element to be entered.
   a. Enter the element in the I^{th} row and J^{th} column.
   b. Press CONTINUE
   c. Repeat step 6 until all elements of the matrix have been entered.

7. When "CHANGES" is displayed:
   Either
   a. Press CONTINUE if you do not want to make any changes to the entered data.
   b. Go to step 11.
   or
   a. Enter 1, if you need to make some changes.
   b. Press CONTINUE

8. When "ROW?" is displayed:
   a. Enter the row number of the element to be changed.
   b. Press CONTINUE

9. When "COLUMN?" is displayed:
   a. Enter the column number of the element to be changed.
   b. Press CONTINUE

10. When "A[I, J]" is displayed: "I" and "J" will be equivalent to the row and column number of the element to be changed.

    a. Enter the correct value of the element.

    b. Press CONTINUE

    c. Go to step 7.

11. The inverse of the original will be printed by rows. The determinant will also be printed unless it is zero (see Diagnostic Messages).

12. "REINVERT" will be displayed:

    Either

    Press CONTINUE if you do not want to reinvert the matrix

    or

    a. Enter 1 to reinvert the matrix

    b. Press CONTINUE

    c. The reinverted matrix will be printed by rows.

EXAMPLES:

$$\begin{bmatrix} 1 & 1/2 & 1/3 \\ 1/2 & 1/3 & 1/4 \\ 1/3 & 1/4 & 1/5 \end{bmatrix}^{-1} = \begin{bmatrix} 9 & -36 & 30 \\ -36 & 192 & -180 \\ 30 & -180 & 180 \end{bmatrix}$$

```
NO. OF ROWS?
3

A[1,1]?
1

A[1,2]?
1/2

A[1,3]?
1/3

A[2,1]?
1/2

A[2,2]?
1/3

A[2,3]?
1/4

A[3,1]?
1/3

A[3,2]?
1/4

A[3,3]?
1/5

CHANGES?
```

```
----INVERSE----

BY ROWS

        9.000000
      -36.000000
       30.000000

      -36.000000
      192.000000
     -180.000000

       30.000000
     -180.000000
      180.000000

DETERMINANT IS
     0.000463
```

```
REINVERT?
1

----INVERSE----

 BY ROWS

        1.000000
        0.500000
        0.333333

        0.500000
        0.333333
        0.250000

        0.333333
        0.250000
        0.200000

DETERMINANT IS
  2160.000000
```

$$\begin{bmatrix} 6 & 5 \\ 9 & 2 \end{bmatrix}^{-1} = \begin{bmatrix} -0.060606 & 0.151515 \\ 0.272727 & -0.181818 \end{bmatrix}$$

```
NO. OF ROWS?
2

A[1,1]?
6

A[1,2]?
5

A[2,1]?
9

A[2,2]?
2

CHANGES?
```

```
                                  REINVERT?
                                  1
```

```
----INVERSE----

BY ROWS

       -0.060606
        0.151515

        0.272727
       -0.181818
```

```
----INVERSE----

BY ROWS

        6.000000
        5.000000

        9.000000
        2.000000
```

```
DETERMINANT IS
      -33.000000
```

```
DETERMINANT IS
       -0.030303
```

" INVERT" SUBROUTINE

This subroutine will invert a square matrix and compute the determinant of that matrix. A modified Gauss - Jordan elimination technique is used with newly computed elements overlaying the original matrix to save storage space. This matrix destroys the original matrix. A row pivot and column pivot search is used to maximize the accuracy.

A determinant of zero will cause an error message to be printed, but any very small determinant may be an indication of loss of accuracy and should be checked by reinverting the result.

When used as a subroutine, all parameters specified under <INPUT> must be assigned in the user's calling program. The user may then append the subroutine to his program and use gsb "INVERT" to take advantage of its capabilities. The results of the calculation and any other output is designated under <OUTPUT>. The user should be aware that the subroutine may alter variables that he is using in his program and should therefore check the variables listed under <DESTROYED>.

An example of a typical program which uses this subroutine may be found on page 48 or in file 5.

PARAMETERS AND VARIABLES USED:

<INPUT>

N	number of rows (also columns since array is square)
A[,]	matrix to be inverted
flg 1	set for reinversion of the resultant matrix; clear if first inversion of matrix

<OUTPUT>

N	unchanged
A[,]	inverted matrix
D	determinant of matrix
flg 4	set if determinant is zero

<DESTROYED>

B	current largest (in magnitude) available pivot element
C	current pivot column
E	temporary storage
I	loop counter for row
J	loop counter for column
K	loop counter
M	row reduction multiplier
P	pivot element
Q	loop counter
R	current pivot row
C[N]	vector to keep track of column interchanges
R[N]	vector to keep track of row interchanges
Y[N]	used to reorder row vector to determine the sign of the determinant

INSTRUCTIONS:

1. Load your program into memory. Your program should set up all necessary parameters (N, A[,]) to be passed to the subroutine.

2. Insert UTILITY ROUTINES cartridge.

3. (Optional)
   a. Type fxd N or flt N where N is the number of digits printed to the right of the decimal point ($0 \leq N \leq 11$).
   b. Press EXECUTE

4. Load the file.
   a. Type ldf 6, N, M where:

      N = line number where the first line of the subroutine should be loaded into memory.

      M = line number where execution is to begin after subroutine is loaded.

   b. Press EXECUTE

ANNOTATED LISTING:

DRIVER - MATRIX INVERSION

```
0: cfg 1,13;ldf
 6,18,1                    Load subroutine
1: enp "NO. OF
 ROWS?",N;if
 flg13;cfg 13;
 gto +0                    Enter number of rows
2: 0→I;1→J;dim
 A[N,N];spc
3: if (I+1→I)>N;
 gto +3
4: cfg 13;enp
 A[I,J];spc ;
 gto +0;if not
 flg13;jmp (J+
 1→J)>N                    Enter elements of matrix
5: 1→J;gto -2
6: enp "CHANGES?
 ",B;spc ;if
 flg13;cfg 13;
 gto +4
7: enp "ROW?",I;
 spc ;if flg13;
 cfg 13;gto +0
8: enp "COLUMN?"
 ,J;spc ;if flg1
 3;cfg 13;gto +0           Make corrections if necessary
9: cfg 13;enp
 A[I,J];spc ;
 gto +0;if not
 flg13;gto -3
10: gsb "INVERT"           Branch to subroutine
11: if flg4;spc
 4;stp                     Stop if error flag is set
12: 0→I;1→J;spc
 3;prt "----INVE
 RSE----";spc ;
 prt "BY ROWS";            Print headings
 spc 2
13: if (I+1→I)>N
 ;spc 4;prt "DET
 ERMINANT IS",D;           Print determinant
 spc 4;gto +3
14: prt A[I,J];
 jmp (J+1→J)>N
15: 1→J;spc ;              Print inverse
 gto -2
16: cfg 13;enp
 "REINVERT?",I;
 if flg13;spc 4;           Check if reinversion is desired
 end
17: sfg 1;gto -7
*6382
```

ANNOTATED LISTING:

SUBROUTINE MATRIX INVERSION

```
0: "INVERT":cfg
 4;0→K;1→D
1: if not flg1;
 dim R[N],C[N]
2: if (K+1→K)>N;
 gto +27
3: 0→I→J→B
4: if (I+1→I)>N;
 gto +12
5: 0→Q
6: if (Q+1→Q)>K-
 1;gto +3
7: if I=R[Q];
 gto -3
8: gto -2
9: if (J+1→J)>N;
 0→J;gto -5
10: 0→Q
11: if (Q+1→Q)>K
 -1;gto +3
12: if J=C[Q];
 gto -3
13: gto -2
14: if (abs(A[I,
 J])→E)>=B;I→R[K
 ];J→C[K];E→B
15: gto -6
16: if (A[R[K],
 C[K]]→P)=0;dsp
 "DET=0";sfg 4;
 beep;beep;ret
17: DP→D
18: R[K]→R;C[K]→
 C
19: 0→J
20: if (J+1→J)>N
 ;gto +3
21: if J=C;1/
 P→A[R,C];gto -1
22: A[R,J]/P→A[R
 ,J];gto -2
23: 0→I→J
24: if (I+1→I)>N
 ;gto -22
25: A[I,C]→M;if
 I=R;gto -1
26: if (J+1→J)>N
 ;0→J;gto -2
27: if J=C;M/-
 P→A[I,C];gto -1
28: A[I,J]-A[R,
 J]M→A[I,J];gto
 -2
```

Dimension row and column permutation vectors unless reinvert flag is set

Inversion finished - branch to unscramble

Reduce next row

Check that this row has not already been used

Reduce next column

Check that this column has not already been used

B will always contain current largest element
current row saved in row permutation vector R[]
current column saved in column permutation vector C[]

If maximum element is zero then determinant is zero

Update determinant
R and C are temporary simple variables for row and column

Normalize the pivot row

Row reduction for remaining rows

ANNOTATED LISTING:

```
29: 0→I→J;if
 not flg1;dim
 Y[N]
30: if (J+1→J)>N
 ;0→I→J;gto +5
31: if (I+1→I)>N
 ;0→I;gto +2
32: A[R[I],J]→Y[
 C[I]];gto -1
33: if (I+1→I)>N
 ;0→I;gto -3
34: Y[I]→A[I,J];
 gto -1
35: if (I+1→I)>N
 ;gto +5
36: if (J+1→J)>N
 ;0→J;gto +2
37: A[I,C[J]]→Y[
 R[J]];gto -1
38: if (J+1→J)>N
 ;0→J;gto -3
39: Y[J]→A[I,J];
 gto -1
```

Put permuted rows and columns back into original order

```
40: 0→I
41: if (I+1→I)<=
 N;C[I]→Y[R[I]];
 gto +0
42: 0→I
43: if (I+1→I)>N
 ;ret
44: 0→J
45: if (J+1→J)>N
 -1;gto -2
46: if Y[J]<=Y[J
 +1];gto -1
47: Y[J]→E;Y[J+
 1]→Y[J];E→Y[J+
 1];-D→D;gto -2
48: gto -3
*17867
```

Determine correct sign of determinant by number of row interchanges required

Simultaneous Equations Using Matrix ROM

FILE 7 DRIVER

FILE 8 SUBROUTINE "MATS"

Simultaneous Equations Using Matrix ROM

This routine will solve N simultaneous linear equations with real coefficients in N unknowns. The program itself requests all the necessary information, calls the subroutine "MATS" in file 8 on page 58 and prints the solutions.

$$a_{11}x_1 + a_{12}x_2 + \ldots + a_{1n}x_n = c_1$$

$$a_{21}x_1 + a_{22}x_2 + \ldots + a_{2n}x_n = c_n$$

$$\ldots\ldots\ldots\ldots\ldots\ldots\ldots\ldots$$

$$a_{n1}x_1 + a_{n2}x_2 + \ldots + a_{nn}x_n = c_n$$

The advantage of using the matrix ROM is twofold. The subroutine itself is significantly shorter allowing a larger system of equations to be solved and the routine is significantly faster. Approximate size and time estimates are available in the Special Considerations section.

SPECIAL CONSIDERATIONS AND DIAGNOSTIC MESSAGES:

1. The matrix ROM is necessary for this program.
2. The maximum number of equations that can be solved depends on the size of the calling program. However, if the program in file 7 is used the following limitations hold. If N is the number of equations then:

   N < 26 on a 6,844 byte machine

   N < 40 on a 15,036 byte machine

   N < 51 on a 23,228 byte machine

   N < 60 on a 31,420 byte machine
3. A determinant (available in D) which is very small compared to the average size of the coefficients is an indication of an unstable system of equations and could lead to erroneous solutions.

4. The time to solve 4 equations was instantaneous and to solve 10 equations took approximately 1.5 seconds.

FORMULAE:

If: A[N,N] matrix of coefficients
C[N] constant vector and
X[N] vector of solution then:

A * X = C and the solution is computed with the formula $X = A^{-1}C$

INSTRUCTIONS:

1. Insert UTILITY ROUTINES cartridge with machine on.

2. Load the file.
   a. Type in ldf 7
   b. Press EXECUTE

3. (Optional)
   a. Type fxd N or flt N where N is the number of digits printed to the right of the decimal point ($0 \leq N \leq 11$).
   b. Press EXECUTE

4. Press RUN

5. When "NO. OF EQS?" appears in the display:
   a. Enter the number of equations (also unknowns) to be solved.
   b. Press CONTINUE

6. "A[I, J]" will appear in the display where "I" will be the row number and "J" the column number of the next element to be entered.
   a. Enter the real number or expression for the coefficient in the I^{th} row and J^{th} column.
   b. Press CONTINUE
   c. Repeat step 6, until all coefficients of the matrix have been entered.

7. When "CHANGES?" is displayed:
   Either
   a. Press CONTINUE if you do not want to make any changes to the entered data.
   b. Go to step 11
   or
   a. Enter 1 if you need to make changes.
   b. Press CONTINUE

8. When "ROW?" is displayed:
   a. Enter the row number of the coefficient to be changed.
   b. Press CONTINUE

9. When "COLUMN?" is displayed:
   a. Enter the column number of the coefficient to be changed.
   b. Press CONTINUE

10. When "A[I, J]" is displayed:
   a. Enter the correct coefficient.
   b. Press CONTINUE
   c. Go to step 7

11. "CONSTANT VECTOR?" will be printed.

12. When "C[I]" is displayed, "I" will represent the I^{th} component of the constant vector.
   a. Enter the value of the I^{th} component of the constant vector.
   b. Press CONTINUE
   c. Repeat step 12 until all the components of the constant vector have been entered.

13. When "CHANGES?" is displayed:
   Either
   a. Press CONTINUE , if you do not need to make changes to the constant vector
   b. Go to step 16
   or
   a. Enter 1, if you need to make some changes.
   b. Press CONTINUE

14. When "ROW?" is displayed:
   a. Enter the row number of the component in the constant vector to be changed.
   b. Press CONTINUE

15. When C[I] is displayed:
   a. Enter the correct value.
   b. Press CONTINUE
   c. Go to step 13.

16. The roots will be printed beginning with X_1 and ending with X_n.

EXAMPLE:

$$\begin{aligned} 3X_1 + 2X_2 + X_3 &= 10 \\ 5X_1 - 2X_2 + X_3 &= 4 \\ -6X_1 + 3X_2 &= 0 \end{aligned} \qquad \begin{aligned} X_1 &= 1 \\ X_2 &= 2 \\ X_3 &= 3 \end{aligned}$$

```
NO OF EQS?
3

A[1,1]?
3

A[1,2]?
2

A[1,3]?
1

A[2,1]?
5

A[2,2]?
-2

A[2,3]?
1

A[3,1]?
-6

A[3,2]?
3

A[3,3]?
0

CHANGES?
```

```
CONSTANT VECTOR?

C[1]?
10

C[2]?
4

C[3]?
0

CHANGES?


---ROOTS ARE---

        1.000000
        2.000000
        3.000000
```

"MATS" SUBROUTINE

This subroutine will solve N simultaneous linear equations with real coefficients in N unknowns, where the equations are written in matrix form as follows:

$$\begin{bmatrix} a_{1,1} & a_{1,2} & a_{1,3} & \cdots & a_{1,n} \\ a_{2,1} & a_{2,2} & \cdots & & a_{2,n} \\ a_{3,1} & \cdots & & & \\ \cdots & \cdots & \cdots & \cdots & \cdots \\ a_{n,1} & & \cdots & & a_{n,n} \end{bmatrix} \begin{bmatrix} x_1 \\ x_2 \\ \vdots \\ x_n \end{bmatrix} = \begin{bmatrix} c_1 \\ c_2 \\ \vdots \\ c_n \end{bmatrix}$$

$a_{i,j}$ are the coefficients, x_i the roots and c_i the constants.

The user must be careful to make a copy of the original matrix of equations if he wishes to use it again. An example of a program which uses this subroutine can be found in file 7 or on page 61 .

PARAMETERS AND VARIABLES USED:

<INPUT>

N	Number of equations
A[N,N]	matrix of coefficients
C[N]	vector of constants

<OUTPUT>

N, C[N]	unchanged

A[N,N]	inverse of original matrix
X[N]	vector of solution X_1, X_2, ..., X_n
D	determinant of matrix of coefficients

<DESTROYED>

A[N,N]	contains inverse of original matrix

INSTRUCTIONS:

1. Load you program into memory. The program should set up all necessary parameters (N, A [,], C []) to be passed to the subroutine.

2. Insert UTILITY ROUTINES cartridge.

3. (Optional)
   a. Type `fxd N` or `flt N` where N is the number of digits printed to the right of the decimal point ($0 \leq N \leq 11$).
   b. Press EXECUTE

4. Load the file.
   a. Type `ldf 8, N, M` where:

      N = line number where the first line of the subroutine should be loaded into memory.

      M = line number where execution is to begin after the subroutine is loaded.

   b. Press EXECUTE

ANNOTATED LISTING:

DRIVER - SIMULTANEOUS EQUATIONS

```
0: cfg 13;ldf 8,
 19,1                    Load subroutine
1: enp "NO OF
 EQS?",N;spc ;
 if flg13;cfg            Enter number of equations and dimension
 13;gto +0
2: 0→I;1→J;dim           the matrix of coefficients
 A[N,N],C[N];
 spc
3: if (I+1→I)>N;
 gto +3
4: cfg 13;enp            Enter elements in matrix of coefficients
 A[I,J];spc ;
 gto +0;if not
 flg13;jmp (J+
 1→J)>N
5: 1→J;gto -2
6: enp "CHANGES?
 ",I;spc 2;if
 flg13;cfg 13;
 gto +4
7: enp "ROW?",I;
 spc ;if flg13;          Make necessary corrections to matrix of
 cfg 13;gto +0           coefficients
8: enp "COLUMN?"
 ,J;spc ;if flg1
 3;cfg 13;gto +0
9: cfg 13;enp
 A[I,J];spc ;
 gto +0;if not
 flg13;gto -3
10: spc 2;prt
 "CONSTANT VECTO
 R?";spc ;1→I
11: cfg 13;enp           Enter constant vector
 C[I];spc ;gto +
 0;if not flg13;
 jmp (I+1→I)>N
12: enp "CHANGES
 ?",I;if flg13;
 gto +3
13: enp "ROW?",
 I;spc ;if flg13         Make any necessary changes to constant
 ;cfg 13;gto +0           vector
14: cfg 13;enp
 C[I];spc ;gto +
 0;if not flg13;
 gto -2
15: gsb "MATS"           Branch to subroutine
16: spc 2;prt "-
 --ROOTS ARE---
 ";1→I;spc 2             Print roots
17: prt X[I];
 jmp (I+1→I)>N
18: spc 4;end
*7364
```

SUBROUTINE - SIMULTANEOUS EQUATIONS WITH MATRIX ROM

```
0: "MATS":dim
  X[N]                          Dimension result vector
1: inv A→A,D                    Compute inverse and determinant
2: mat A*C→X                    Compute vector of solutions
3: ret
*22716
```

Simultaneous Equations (Maximum Accuracy)

FILE 9 DRIVER

FILE 10 SUBROUTINE "SOLVE"

Simultaneous Equations (Maximum Accuracy)

This program solves a system of N simultaneous linear equations with real coefficients in N unknowns.

$$a_{1,1}\,x_1 + a_{1,2}\,x_2 + \ldots + a_{1,n}\,x_n = a_{1,\,n+1}$$

$$a_{2,1}\,x_1 + a_{2,2}\,x_2 + \ldots + a_{2,n}\,x_n = a_{2,\,n+1}$$

$$\ldots\ldots\ldots\ldots\ldots\ldots\ldots\ldots$$

$$a_{n,1}\,x_1 + a_{n,2}\,x_2 + \ldots\ a_{n,n}\,x_n = a_{n,n+1}$$

The coefficients may be real numbers (e.g. 5, 2.1416, π) or expressions whose values are real numbers (e.g. 1/2, $\sqrt{3}$, sin (π/4)).

The program requests the necessary inputs and prints the results after calling the subroutine "SOLVE" in file 10 on page 70 .

SPECIAL CONSIDERATIONS AND DIAGNOSTIC MESSAGES:

1. The maximum number of equations that can be solved with this subroutine will depend on the size of the calling program. The following size limitations assume that the calling program in file 9 is used. If N is the number of equations than:

   N < 25 on a 6,844 byte machine

   N < 40 on a 15,036 byte machine

   N < 51 on a 23,228 byte machine

   N < 60 on a 31,420 byte machine

2. "DET = 0" indicates that the system cannot be solved since the determinant is zero.

3. The time to solve four equations was approximately two seconds while ten equations took about four seconds.

FORMULAE:

1. Normalization:

$$\left.\frac{a_{kj}}{a_{kk}} \rightarrow a_{kj}, \text{ for } j = n+m, n+m-1, \ldots, k\right\} k = 1, 2, \ldots, n$$

2. Reduction:

$$\left.\begin{array}{l} a_{ij} - a_{ik}\, a_{kj} \rightarrow a_{ij} \\ j = n+m, n+m-1, \ldots, k \end{array}\right\} \text{for } i = 1, 2, \ldots, n \quad (i \neq k)$$

REFERENCES:

1. Carnahan, B., Luther, H.A. and Wilkes, J.O., Applied Numerical Methods (New York: John Wiley & Sons, 1969), pp. 270 - 273.

INSTRUCTIONS:

1. Insert UTILITY ROUTINES cartridge with machine on.

2. Load the file:
   a. Type in ldf 9
   b. Press EXECUTE

3. (Optional)
   a. Type fxd N or flt N where N is the number of digits printed to the right of the decimal point ($0 \leq N \leq 11$).
   b. Press EXECUTE

4. Press RUN

5. When "NO. OF EQS?" is displayed:
   a. Enter the number of equations to be solved.
   b. Press CONTINUE

6. "A[I, J]" will appear in the display where "I" will be the row number and "J" the column number of the next element to be typed in.
   a. Enter the coefficient in the I^{th} row and J^{th} column.
   b. Press CONTINUE
   c. Repeat step 6 until all coefficients of the matrix have been entered.

7. When "CHANGES?" is displayed:
   Either
   a. Press CONTINUE if you do not want to make changes to the entered data.
   b. Go to step 11.
   or
   a. Enter 1 if you need to make changes.
   b. Press CONTINUE

8. When "ROW?" is displayed:
   a. Enter the row number of the coefficient to be changed.
   b. Press CONTINUE

9. When "COLUMN?" is displayed:

   a. Enter the column number of the coefficient to be changed.

   b. Press CONTINUE

10. When "A[I, J]" is displayed, "I" and "J" will be the row and column numbers of the coefficient to be changed.

   a. Enter the correct coefficient

   b. Press CONTINUE

   c. Go to step 7.

11. The roots will be printed beginning with X_1 and ending with X_n. If the system cannot be solved the message "DET = 0" will be printed.

EXAMPLE:

$x_1 + x_2 = 3$

$2x_1 - 2x_2 = -2$

$x_1 = 1$

$x_2 = 2$

```
NO. OF EQS?
2

A[1,1]?
1

A[1,2]?
1

A[1,3]?
3

A[2,1]?
2

A[2,2]?
-2

A[2,3]?
-2

CHANGES?



---ROOTS ARE----

        1.000000
        2.000000
```

"SOLVE" SUBROUTINE

This subroutine will solve a system of N simultaneous linear equations with real coefficients in N unknowns.

$$a_{1,1}\,x_1 + a_{1,2}\,x_2 + \ldots + a_{1,n}\,x_n = a_{1,\,n+1}$$

$$a_{2,1}\,x_1 + a_{2,2}\,x_2 + \ldots + a_{2,n}\,x_n = a_{2,\,n+1}$$

$$\ldots\ldots\ldots\ldots\ldots\ldots\ldots\ldots$$

$$a_{n,1}\,x_1 + a_{n,2}\,x_2 + \ldots + a_{n,n}\,x_n = a_{n,n+1}$$

A modified Gauss - Jordan method with a row pivot search is used to increase the accuracy. The original matrix is destroyed during the process. When used as a subroutine, all parameters specified under <INPUT> must be assigned in the user's calling program. The user may then append the subroutine to his program and use gsb "SOLVE" to take advantage of its capabilities. The results of the calculation and any other output is designated under <OUTPUT>. The user should be aware that the subroutine may alter variables the he is using in his program and should therefore check the variables listed under <DESTROYED>. An example of a typical program which uses this subroutine may be found on page 73 or in file 9.

PARAMETERS AND VARIABLES USED:

<INPUT>

N	number of equations (unknowns)
A[N, N+1]	matrix of coefficients where the first subscript is the row

<OUTPUT>

N	unchanged
X[N]	vector of solutions X_i for i = 1, N
flg 4	set if the determinant of the matrix of coefficients is zero -indicating that system is unsolvable.

<DESTROYED>

B	current largest (in magnitude) element in search for pivot; also used for pivot.
I	loop counter used as row subscript
J	loop counter used as column subscript
K	loop counter
R	row which contains largest pivot available
S	partial sum used in backsolve process
T	temporary storage
A[N,N+1]	matrix of coefficients

INSTRUCTIONS:

1. Load your program into memory. The program should set up all necessary parameters (N, A[,]) to be passed to the subroutine.

2. Insert UTILITY ROUTINES cartridge.

3. (Optional)
   a. Type **fxd N** or **flt N** where N is the number of digits printed to the right of the decimal point ($0 \leq N \leq 11$).

4. Load the file.
   a. Type **ldf 10, N, M** where:

      N = line number where the first line of the subroutine should be loaded into memory.

      M = line number where execution is to begin after subroutine is loaded.

   b. Press **EXECUTE**

DRIVER - SIMULTANEOUS EQUATIONS


```
0: cfg 13;ldf
 10,15,1
```
Load subroutine

```
1: enp "NO. OF
 EQS?",N;spc ;
 if flg13;cfg
 13;gto +0
2: 0→I;1→J;dim
 A[N,N+1]
```
Enter number of equations and dimension matrix of coefficients

```
3: if (I+1→I)>N;
 gto +3
4: cfg 13;enp
 A[I,J];spc ;
 gto +0;if not
 flg13;jmp (J+
 1→J)>N+1
5: 1→J;gto -2
```
Enter matrix elements

```
6: spc ;enp "CHA
 NGES?",B;spc 2;
 if flg13;gto +4
7: enp "ROW?",I;
 spc ;if flg13;
 cfg 13;gto +0
8: enp "COLUMN?"
 ,J;spc ;if flg1
 3;cfg 13;gto +0
9: cfg 13;enp
 A[I,J];spc ;
 gto +0;if not
 flg13;gto -3
```
Make necessary corrections

```
10: gsb "SOLVE"
```
Branch to subroutine

```
11: spc 4;if
 flg4;stp
12: prt "---ROOT
 S ARE----";spc
 ;1→I
13: prt X[I];
 jmp (I+1→I)>N
14: end
```
Print roots

```
*5084
```

ANNOTATED LISTING:

SUBROUTINE - SIMULTANEOUS EQUATIONS

```
0: "SOLVE":cfg
 4;0→I
1: if (I+1→I)=N;
 gto +15
```

Branch to backsolve if all rows are reduced

```
2: I-1→K;0→B
3: if (K+1→K)>N;
 gto +3
4: if (abs(A[K,
 I])→T)>B;T→B;
 K→R
5: gto -2
```

Find maximum pivot row

```
6: if B=0;sfg 4;
 dsp "DET=0";
 ret
```

If maximum pivot is zero - determinant is zero

```
7: I→J;if I=R;
 gto +2
8: A[I,J]→T;A[R,
 J]→A[I,J];T→A[R
 ,J];jmp (J+1→J)
 >N+1
```

If present row is pivot row - no need to switch rows

Interchange row I with pivot row R

```
9: I+1→J;A[I,
 I]→B
10: A[I,J]/B→A[I
 ,J];jmp (J+1→J)
 >N+1
11: I→K
12: if (K+1→K)>N
 ;gto -11
13: I+1→J
14: A[K,J]-A[K,
 I]A[I,J]→A[K,
 J];jmp (J+1→J)>
 N+1
15: gto -3
```

Normalize pivot row

Reduction of all rows higher than pivot row

```
16: if A[N,N]=0;
 0→B;gto -10
```

Check for zero determinant (last pivot element = 0)

```
17: dim X[N];
 A[N,N+1]/A[N,
 N]→X[N];N→I
18: if (I-1→I)<1
 ;ret
19: 0→S;N→K
20: A[I,K]X[K]+
 S→S;jmp (K-1→K)
 <=I
21: A[I,N+1]-
 S→X[I];gto -3
```

Backsolve process once matrix has been reduced

```
*18983
```

Simultaneous Equations (Maximum Size)

FILE 11 SIMULTANEOUS EQUATION ROUTINE USING MINIMUM AMOUNT OF SPACE

Simultaneous Equations (Maximum Size)

This program will solve N simultaneous linear equations in N unknowns, using the minimum amount of memory possible to allow for the largest possible number of equations.

$$a_{1,1}x_1 + a_{1,2}x_2 + \ldots + a_{1,n}x_n = a_{1,n+1}$$
$$a_{2,1}x_1 + a_{2,2}x_2 + \ldots + a_{2,n}x_n = a_{2,n+1}$$
$$\vdots$$
$$a_{n,1}x_1 + a_{n,2}x_2 + \ldots + a_{n,n}x_n = a_{n,n+1}$$

The method used performs all necessary operations on coefficients as they are input one at a time by rows. Only those coefficients necessary for modification of future coefficients are saved. Two disadvantages accrue with this method. Because no row or column pivoting is performed the accuracy of the solution may be diminished and secondly, it is more likely that a zero diagonal element will be found. In the later case an error message "DET = 0" is printed. The equations may still be solvable if you type in the coefficients again with the rows in a different order. If the system of equations is small enough, these problems can be avoided by using the programs in file 9 and 10 on pp. 63 - 74.

Because of the method used to enter the coefficients, a subroutine has not been written for this program.

SPECIAL CONSIDERATIONS AND DIAGNOSTIC MESSAGES:

1. The maximum number of equations that can be solved with this program will be affected by other programs resident in memory. But if no other user programs are present then the following limitations apply:

If N is the number of equations then:

N < 36 on a 6,844 byte machine

N < 57 on a 15,036 byte machine

N < 73 on a 23,228 byte machine

N < 85 on a 31,420 byte machine

2. "DET = 0" indicates that the system has no solution. Enter the coefficients again with the rows in a different order.

3. The time requirements for this program are negligible since calculations occur after each coefficient is entered.

FORMULAE:

If $j \leq i$

$a_{ij} - a_{ik} * a_{kj} \rightarrow a_{ij}$ for k = 1, j - 1

If $j > i$

$a_{ij} - a_{ik} * a_{kj} \rightarrow a_{ij}$ for k = 1 to i - 1

$a_{ij}/a_{ii} \rightarrow a_{ij}$

For backsolve process:

$a_{n,n+1} \rightarrow x_n$

$$x_i = a_{i,n+1} - \sum_{k=n}^{i+1} x_k a_{ik} \quad \text{for } i = n-1, n-2, \ldots 1$$

For correspondence between U[K] and A[I,J]

$$K = (I - 1)\left[N - \frac{I - 2}{2}\right] + J - I$$

REFERENCES:

1. Hewlett Packard 9820A Math Pac, pp. 37 - 41.

VARIABLES USED:

A	current coefficient being entered
I	current row number
J	current column number - reinitialized to 0 after N + 1 values have been read in (one row)

K	loop counter
N	number of equations
S	partial sum used in backsolve process
A[N]	row elements a_{ij} of current row i where $i \leq j$ A[1] is the first element of row A[I] is the pivot element for i^{th} row The solution vector is also stored here
U[N(N+1)/2]	upper diagonal elements of matrix which must be saved since they modify coefficients which are read in later.

INSTRUCTIONS:

1. Insert UTILITY ROUTINES cartridge with machine on.

2. Load the file.
   a. Type in ldf ll
   b. Press EXECUTE

3. (Optional)
   a. Type fxd N or flt N where N is the number of digits printed to the right of the decimal point (0<N<ll).
   b. Press EXECUTE

4. Press RUN

5. When "NO. OF ROWS?" is displayed:
   a. Enter the number of rows.
   b. Press CONTINUE.

6. When "ROW I" is printed, "I" represents the current row of coefficients to be entered. Go to step 7.
   or
   If all rows have been entered, go to step 8.

7. When "A(I, J)" appears in the display:
   a. Enter the next coefficient in that row
   b. Press CONTINUE
   c. Repeat steps a & b until all elements of row I have been entered.
   d. Go to step 6.

8. The roots X_1, X_2, ... X_n will be printed unless the error "DET = 0" occurs. If this error is printed, you can rearrange the order that the rows are entered and go back to step 4.

EXAMPLE:

$$3X_1 - X_2 + 2X_3 = 0$$

$$X_1 + 5X_2 + 7X_3 = 15$$

$$-5X_1 + 2X_2 + 9X_3 = -11$$

$$X_1 = 2$$

$$X_2 = 4$$

$$X_3 = -1$$

```
NO. OF ROWS?
3

BY ROWS

ROW      1.000000

          3.000000
         -1.000000
          2.000000
          0.000000

ROW      2.000000

          1.000000
          5.000000
          7.000000
         15.000000

ROW      3.000000

         -5.000000
          2.000000
          9.000000
        -11.000000



---ROOTS ARE----

          2.000000
          4.000000
         -1.000000
```

ANNOTATED LISTING:

SIMULTANEOUS EQUATIONS - MINIMUM SPACE

```
0: cfg 13;enp
 "NO. OF ROWS?",
 N;if fl§13;§to
 +0
1: dim A[N],U[.5
 N(N+1)];0→I→J;
 spc ;prt "BY
 ROWS";spc
2: if (I+1→I)>N;
 §to +10
3: spc ;prt "ROW
 ",I;spc
4: if (J+1→J)>N+
 1;spc ;0→J;§to
 -2
5: ent "A[I,J]",
 A;prt A;1→K;if
 I<J;§to +4
6: if J=1;A→A[1]
 ;§to -2
7: A-A[K]U[(K-
 1)(N-.5(K-2))+
 J-K]→A;jmp (K+
 1→K)>J-1
8: A→A[J];§to -4
9: if I#1;A-A[K]
 U[(K-1)(N-.5(K-
 2))+J-K]→A;jmp
 (K+1→K)>I-1
10: if A[I]#0;A/
 A[I]→U[(I-1)(N-
 .5(I-2))+J-I];
 §to -6
11: spc 2;prt
 "DET=0";stp
12: U[.5N(N+1)]→
 A[N];N→I
13: if (I-1→I)<1
 ;§to +4
14: N→J;0→S
15: S+A[J]U[(I-
 1)(N-.5(I-2))+
 J-I]→S;jmp (J-
 1→J)<I+1
16: U[(I-1)(N-
 .5(I-2))+N+1-
 I]-S→A[I];§to -
 3
17: 1→I;spc 3;
 prt "---ROOTS
 ARE----";spc
18: prt A[I];
 jmp (I+1→I)>N
19: spc 4;end
*19871
```

Initialize storage needed

Enter coefficients A[I,J] where

I = row and J = column

If column 1 just save

If column number less than diagonal element perform row reduction

If column number greater than or equal to diagonal element perform row reduction and divide by pivot element

If pivot element is zero - determinant is zero

Backsolve process

$$A_{n,\ n+1} \rightarrow X_n$$

$$A_{i,\ n+1} = \sum_{k=n}^{i+1} X_k A_{ik}$$

Solutions X_1, X_2 ... X_n are stored in array A[1], A[2], ..., A[N]

Numerical Integration Of User-Defined Function (Simpson's One-Third Rule)

FILE 12 DRIVER

FILE 13 SUBROUTINE "SIMPSON"

Numerical Integration Of User-Defined Function (Simpson's One-Third Rule)

This program approximates $\int_a^b f(x)dx$ by calling the subroutine "SIMPSON" in file 13 on page 89 . The user defines the function f(x) which may be algebraic of the form $a_0 + a_1x^{e1} + a_2x^{e2} + \ldots + a_nx^{en}$ with a_i real and e_i rational (e.g., $X^3 + X^{3/2} + \sqrt{X}$) or transcendental (e.g., sin(X) + cos(X)), but must be continuous over the interval [a,b].

SPECIAL CONSIDERATIONS AND DIAGNOSTIC MESSAGES:

1. "BOUNDS REVERSED" indicates the lower intergral bound is larger than upper.
2. "DEFINE FUNCTION" indicates that the user forgot to define the function.

FORMULAE:

Simpson's One-Third Rule:

$$\int_a^b f(x)dx \approx \frac{h}{3}[f(a) + 4f(a + h) + 2f(a + 2h) + 4f(a + 3h) + \ldots + 4f(a + (n - 1)h) + f(a + nh)]$$

where n = number of intervals,

$$h = \frac{(b - a)}{n} = \text{interval size}$$

REFERENCES:

Beckett, Royce and Hurt, James, Numerical Calculations and Algorithms (New York: McGraw-Hill, 1967), pp. 166 - 169.

INSTRUCTIONS:

1. Insert UTILITY ROUTINES cartridge with machine on.
2. Load the file.
   a. Type ldf 12
   b. Press EXECUTE
3. (Optional)
   a. Type fxd N or flt N where N is the number of digits printed to the right of the decimal point ($0 \leq N \leq 11$).
   b. Press EXECUTE
4. Press RUN
5. When "DEFINE FUNCTION" appears in the display:
   a. Type gto "EVAL"
   b. Press EXECUTE
6. Press STEP
7. Press STEP
8. Fetch the line where the function is to be stored.
   a. Observe the number in the display and press FETCH followed by that number.
   b. Press EXECUTE
9. 'beep; beep ; dsp "DEFINE FUNCTION"; stp' will be displayed:
   If the function definition will fit on one line then:
   a. Type the function to be evaluated into the display in the form f(X) → Y (e.g. X↑2 + 3X + 5 → Y).
   b. Press STORE
   c. Go to step 10.
   Otherwise, if the function requires more than one line:
   a. Type the first line of the function definition.
   b. Press STORE
   c. Type the next line of the function definition.
   d. Press INSERT (line INSERT not character INSERT)
   e. Repeat steps c and d until the complete function has been entered.

10. Continue execution

    a. Type cont 2 (you must type the word "cont" character by character rather than pressing the CONTINUE key).

    b. Press EXECUTE

11. "MAX NO OF ITER?" will appear in the display.

    a. Enter the maximum number of times that you would like the interval size halved.

    b. Press CONTINUE

12. When "EPSILON" appears in the display:

    Either

    a. Enter the error tolerance stopping criterion. When the difference between two successive computations of the integral is less than this value, the procedure will stop.

    b. Press CONTINUE

    or

    Press CONTINUE which will automatically assign an epsilon of 10^{-6}.

13. When "LOW BOUND?" is displayed:

    a. Enter the lower bound of integration.

    b. Press CONTINUE

14. When "UPPER BOUND?" is displayed:

    a. Enter the upper bound of integration.

    b. Press CONTINUE

15. When "PRT PARTIAL RES?" is displayed:

    Either

    a. Enter 1 if you want the computed integral printed out for each halving of the interval size.

    b. Press CONTINUE

    or

    Press CONTINUE if you only want the answer printed.

16. The integral will be printed.

EXAMPLES:

```
MAX NO OF ITER?
6

EPSILON?

LOWER BOUND?
-4

UPPER BOUND?
2

PRT PARTIAL RES?
1

# INTV        2.00
AREA       -132.00

# INTV        4.00
AREA       -132.00


INTEGRAL FROM
             -4.00
             TO
              2.00

IS          -132.00
```

$\int_{-4}^{2} (x^3-2x^2+3x-1)dx = -132$

```
MAX NO OF ITER?
6

EPSILON?
.000001

LOWER BOUND?
-2

UPPER BOUND?
2

PRT PARTIAL RES?
1

# INTV  2.000000
AREA    5.333333

# INTV  4.000000
AREA    5.333333


INTEGRAL FROM
        -2.000000
           TO
         2.000000

IS       5.333333
```

$\int_{-2}^{2} x^2dx = 16/3 = 5.333333$

"SIMPSON" SUBROUTINE

This subroutine will approximate $\int_a^b f(x)dx$ for the user-defined function f(x) which must be stored in the form f(X) → Y (e.g., X↑2 + 2X + 3 → Y). Because the function is user defined, it may be algebraic of the form $a_0 + a_1x^{e1} + a_2x^{e2} + \ldots + a_nx^{en}$ with a_i real and e_i rational (e.g., $X^3 + 3X^{3/2} - \sqrt{X}$) or transcendental (e.g. sin(X) + cos(X)), but must be continuous over the interval [a, b].

The method used in Simpson's one-third rule with truncation error $o(h^4)$ where h is the interval size.

The stopping criterion for this method is either a maximum number of interval halvings or successive computations of the integral differing by less than some user-supplied epsilon (error tolerance). When used as a subroutine, all parameters specified under <INPUT> must be assigned in the user's calling program. The user may then append the subroutine to his program and use gsb "SIMPSON" to take advantage of its capabilities. The results of the calculation and any other output is designated under <OUTPUT>. The user should be aware that the subroutine may alter variables that he is using in the program and should therefore check the variables listed under <DESTROYED>. An example of a typical program which uses this subroutine may be found on page 93 or in file 12.

PARAMETERS AND VARIABLES USED:

<INPUT>

M	maximum number of iterations (interval halvings)
A	lower bound of integration

B	upper bound of integration
E	epsilon; error tolerance between successive integral computations that will halt procedure
f(X)→Y	function must be defined by user (see instructions for correct procedure to do this)
flg 3	should be set if you would like intermediate results printed

<OUTPUT>

A, B, M, E	unchanged
S	integral $\int_a^b f(x)\,dx$
flg 2	set if integration bounds are reversed

<DESTROYED>

F	previous value of integral used for stopping criterion
I	loop counter for number of iterations; must be less than M
L	domain argument
N	current number of intervals (2↑I = N)
W	interval size (B - A)/N
X	domain argument for function evaluation
Y	functional value f(X)
flg 6	used to eliminate the epsilon stopping criterion on first computation of integral

INSTRUCTIONS:

1. Load you program into memory. The program should set up all necessary parameters (A, B, M, E, flg 3) to be passed to the subroutine.
2. Insert UTILITY ROUTINES cartridge.
3. Type ldf 13, N where:
   N = line number where first line of subroutine should be loaded into memory.
4. Press EXECUTE
5. (Optional)
   a. Type fxd N or flt N where N is the number of digits printed to the right of the decimal point ($0 \leq N \leq 11$).
   b. Press EXECUTE
6. Find the line where the function is to be stored.
   a. Type gto "EVAL"
   b. Press EXECUTE
7. Press STEP
8. Press STEP
9. Fetch the function line into the display.
   a. Observe the number in the display and press FETCH followed by that number.
   b. Press EXECUTE
10. 'beep; beep; dsp "DEFINE FUNCTION"; stp' will be displayed:
    If the function definition will fit on one line then:
    a. Type the function to be evaluated into the display in the form f(X) → Y (e.g. X↑2 + 3X + 5 → Y).
    b. Press STORE
    c. Go to step 11

    Otherwise, if the function requires more than one line:
    a. Type the first line of the function definition.
    b. Press STORE
    c. Type the next line of the function definition
    d. Press INSERT (line INSERT not character INSERT)

e. Repeat steps c and d until the complete function has been entered

11. Continue execution.

a. Type cont N where N is the location where program execution should begin (do not press CONTINUE key; you must type "cont" one character at a time).

b. Press EXECUTE

12. Your program is ready to run.

ANNOTATED LISTING:

DRIVER - NUMERICAL INTEGRATION - SIMPSON'S METHOD

```
0: ldf 13,10,1
1: dsp "DEFINE
 FUNCTION";stp
2: cfg 13;spc ;
 enp "MAX NO OF
 ITER?",M;spc ;
 if flg13;cfg
 13;gto +0
3: enp "EPSILON?
 ",E;spc ;if
 flg13;cfg 13;
 1e-6→E
4: enp "LOWER
 BOUND?",A;spc ;
 if flg13;cfg
 13;gto +0
5: enp "UPPER
 BOUND?",B;spc ;
 if flg13;cfg
 13;gto +0
6: sfg 3;enp
 "PRT PARTIAL
 RES?",I;spc ;
 if flg13;cfg 3
7: gsb "SIMPSON"
8: if flg2;stp
9: spc 2;prt
 "INTEGRAL FROM"
 ,A,"
 TO",B," ","IS",
 S;spc 4;end
*23520
```

Load subroutine

User must define his function in the subroutine

How many times should interval be halved?

Error tolerance between successive calculations of integral

Lower integration bound

Upper integration bound

Should the partial results be printed?

Branch to subroutine

Stop if error flag is set

Print integral and integration bounds

ANNOTATED LISTING:

SUBROUTINE - NUMERICAL INTEGRAION - SIMPSON'S METHOD

```
0: "SIMPSON":0→I
 →S;cfg 2;sfg 6
1: if A>B;dsp
 "BOUNDS REVERSE
 D";sfg 2;ret
2: A→X;gsb "EVAL
 "
3: S+Y→S;B→X;
 gsb "EVAL"
4: S+Y→S→T
5: if (I+1→I)>M;
 F→S;ret
6: 2↑I→N;((B-A)/
 N→W)+A→X;gsb
 "EVAL"
7: S+4Y→S;A→L
8: if (L+2W→L)<B
 ;gto +5
9: WS/3→S;if
 flg3;prt "#
 INTV",N,"AREA",
 S;spc
10: if flg6;cfg
 6;gto +2
11: if abs(F-
 S)<E;ret
12: S→F;T→S;gto
 -7
13: L→X;gsb "EVA
 L"
14: S+2Y→S;L+
 W→X;gsb "EVAL"
15: S+4Y→S;gto -
 7
16: "EVAL"
17: beep;beep;
 dsp "DEFINE
 FUNCTION";stp
18: ret
*15316
```

Check for lower bound greater than upper bound

Functional values of initial and final values of integral are saved in T

Compute number of intervals (N), interval size (W), and first interval value (Y). Lower bound is assigned to L.

L is incremented by 2 interval sizes up to the upper bound B

Print approximation to integral

Flag 6 cleared after first pass through loop

Check if stopping criterion satisfied

Save former sum for convergence check

Multiply successive function values by 2 or 4 and update the sum

Line 17 will contain the user's function definition

Numerical Integration Of User-Defined Function (Romberg Quadrature)

FILE 14 DRIVER

FILE 15 SUBROUTINE "ROMBERG"

Numerical Integration Of User-Defined Function (Romberg Quadrature)

This program approximates $\int_a^b f(x)dx$ by calling the subroutine "ROMBERG" in file 15 on page 102. The function must be defined by the user in the form f(X) → Y as explained in the instructions and should be continuous over the interval [a, b].

SPECIAL CONSIDERATIONS AND DIAGNOSTIC MESSAGES:

1. Entering an arbitrarily large number of iterations for interval halvings will not necessarily increase the accuracy and will definitely slow the computation down. This method is usually more accurate than Simpson's (with truncation error $o(h^4)$) so the smallest interval size should be chosen accordingly.
2. "BOUNDS REVERSED" indicates that the lower integration bound is larger than the upper.
3. "DEFINE FUNCTION" indicates that the user forgot to define the function to be integrated.

FORMULAE:

Trapezoid Rule:

T_i = integral evaluated for i trapezoid intervals.

$$T_i = \frac{h}{2}(f(a) + f(b)) + h \sum_{j=1}^{i-1} f(a + jh)$$

T_{2i} = integral evaluated for 2i trapezoid intervals.

We can compute a new approximation

$$T_{2i}^{(1)} = \frac{4T_{2i} - T_i}{3}$$

Further extrapolation computes approximations according to the formula:

$$T_{2i}^{(j)} = \frac{4^j T_{2i}^{(j-1)} - T_i^{(j-1)}}{4^j - 1}$$

The resulting matrix:

$$\begin{array}{ccccc} T_1 & T_2^{(1)} & T_4^{(2)} & T_8^{(3)} & T_{16}^{(4)} \\ T_2 & T_4^{(1)} & T_8^{(2)} & T_{16}^{(3)} & . \\ T_4 & T_8^{(1)} & T_{16}^{(2)} & . & \\ T_8 & T_{16}^{(1)} & . & & \\ T_{16} & . & & & \end{array}$$

is called the tableau where an element $T_i^{(j)}$ represent the j^{th} approximation of the integral with i intervals.

REFERENCES:

1. Stark, Peter A., Introduction to Numerical Methods (The MacMillan Company, 1970), pp. 212 - 219.

INSTRUCTIONS:

1. Insert UTILITY ROUTINES cartridge with machine on.
2. Load the file.
   a. Type ldf 14
   b. Press EXECUTE
3. (Optional)
   a. Type fxd N or flt N where N is the number of digits printed to the right of the decimal point ($0 \leq N \leq 11$).
   b. Press EXECUTE
4. Press RUN
5. When "DEFINE FUNCTION" appears in the display:
   a. Type gto "EVAL"
   b. Press EXECUTE
6. Press STEP
7. Press STEP
8. Fetch the line where the function is to be stored.
   a. Observe the number in the display and press FETCH followed by that number.
   b. Press EXECUTE
9. 'beep; beep; dsp "DEFINE FUNCTION"; stp' will be displayed:
   If the function definition will fit on one line then:
   a. Type the function to be evaluated into the display in the form f(X) → Y (e.g. X↑2 + 3X + 5 → Y).
   b. Press STORE
   c. Go to step 10.
   Otherwise, if the function requires more than one line:
   a. Type the first line of the function definition.
   b. Press STORE
   c. Type the next line of the function definition
   d. Press INSERT (line INSERT not character INSERT)
   e. Repeat steps c and d until the complete function has been entered.

10. Continue execution.

    a. Type cont 2 (you must type the word "cont" character by character rather than pressing the CONTINUE key).

    b. Press EXECUTE

11. When "MAX NO OF ITER" is displayed:

    a. Enter the maximum number of interval halvings that should occur during initial computations of the integral using the trapezoidal rule.

    b. Press CONTINUE

12. When "LOW BOUND?" is displayed:

    a. Enter the lower integration bound.

    b. Press CONTINUE

13. When "UPPER BOUND?" is displayed:

    a. Enter the upper integration bound.

    b. Press CONTINUE

14. When "EPSILON" is displayed:

    Either

    a. Enter the error tolerance to determine when the computation of successive approximations should stop.

    b. Press CONTINUE

    or

    Press CONTINUE which will assign a default value of 10^{-6} to epsilon.

15. When "PRINT TABLEAU?" is displayed:

    Either

    a. Enter 1 if you want to print the entire tableau by columns.

    b. Press CONTINUE

    or

    Press CONTINUE , if you simply want the final integral printed.

16. The integral will be printed.

EXAMPLES:

This example is in radian mode.

```
MAX NO OF ITER?
6

LOWER BOUND?
-2

UPPER BOUND?
2

EPSILON
.00001

PRINT TABLEAU?
1
```

```
MAX NO OF ITER?
4

LOWER BOUND?
0

UPPER BOUND?
π

EPSILON


PRINT TABLEAU?
1
```

$\int_{-2}^{2} x^2 dx$

```
        16.000000
         8.000000
         6.000000
         5.500000
         5.375000
         5.343750

         5.333333
         5.333333
         5.333333
         5.333333
         5.333333
```

$\int_{0}^{\pi} \sin(x) dx$

```
         0.000000
         1.570796
         1.896119
         1.974232

         2.094395
         2.004560
         2.000269

         1.998571
         1.999983

         2.000006
```

```
----------------

INTEGRAL FROM
        -2.000000
            TO
         2.000000

IS       5.333333
```

```
----------------

INTEGRAL FROM
         0.000000
            TO
         3.141593

IS       2.000000
```

"ROMBERG" SUBROUTINE

This subroutine will approximate $\int_a^b f(x)dx$ for the user defined function f(x) which must be stored in the form f(X) → Y (e.g. X↑2 + 2X + 3 → Y). Because the function is user defined, it may be algebraic of the form $a_0 + a_1x^{e_1} + a_2x^{e_2} + \ldots + a_nx^{e_n}$ with a_i real and e_i rational (e.g. $X^3 + 3X^{3/2} - \sqrt{X}$) or transcendental (e.g. sin(X) + cos(X)), but should be continuous over the interval [a, b].

The method used is Romberg Quadrature which combines trapezoidal integration and extrapolation to obtain a result with more accuracy than the same number of intervals using Simpson's method. The resultant Romberg tableau may optionally be printed.

The stopping criterion for this method is either a maximum number of interval halvings or successive computations of the interval differing by less than some user supplied epsilon (error tolerance).

When used as a subroutine, all parameters specified under <INPUT> must be assigned in the user's calling program. The user may then append the subroutine to his program and use gsb "ROMBERG" to take advantage of its capabilities. The results of the calculation and any other output is designated under <OUTPUT>. The user should be aware that the subroutine may alter variables that he is using in his program and should therefore check the variables listed under <DESTROYED>. An example of a typical program which uses this subroutine may be found on page 107 or in file 14.

PARAMETERS AND VARIABLES USED:

<INPUT>

M	maximum number of iterations where 2^M is the maximum number of intervals. Thus there will be at most M computations of the integral using the trapezoidal rule.
A	lower integration bound
B	upper integration bound
E	error tolerance
flg 3	should be set if tableau is to be printed
f(x)→Y	functional equation should be stored in "EVAL" subroutines

<OUTPUT>

M, A, B, E	unchanged
T[0]	approximation of $\int_a^b f(x)dx$
flg 2	set if integration bounds were reversed

<DESTROYED>

F	previous computed integral value used to check if error tolerance is satisfied
G	temporary storage
I	loop counter
N	used to determine the number of intervals in initial computation of trapezoid approximations (1<N<M), and as subscript for column entries in tableau
P	2^N-1, used for extrapolation
Q	2^N, the number of intervals
R	2^{N-1}, used for extrapolation

S	used for partial sums in computation of integral
X	domain argument for functional evaluation
Y	functional value f(X)
T[ ]	contains a single column of the tableau
flg 6	used to eliminate error tolerance check on initial computation

INSTRUCTIONS:

1. Load your program into memory. The program should set up all necessary parameters (A, B, M, E, flg 3) to be passed to the subroutine.
2. Insert UTILITY ROUTINES cartridge.
3. Type ldf 15, N where:
   N = line number where first line of subroutine should be loaded into memory.
4. Press EXECUTE
5. (Optional)
   a. Type fxd N or flt N where N is the number of digits printed to the right of the decimal point ($0 \leq N \leq 11$).
   b. Press EXECUTE
6. Find the line number where the function is to be stored.
   a. Type gto "EVAL"
   b. Press EXECUTE
7. Press STEP
8. Press STEP
9. Fetch the line where the function is to be stored.
   a. Observe the number in the display and press FETCH followed by that number.
   b. Press EXECUTE
10. 'beep; beep; dsp "DEFINE FUNCTION"; stp' will be displayed:
    If the function definition will fit on one line then:
    a. Type the function to be evaluated into the display in the form f(X) → Y (e.g. X↑2 + 3X + 5 → Y).
    b. Press STORE
    c. Go to step 11.
    Otherwise, if the function requires more than one line:
    a. Type the first line of the function definition.
    b. Press STORE
    c. Type the next line of the function definition
    d. Press INSERT (line INSERT not character INSERT)

e. Repeat steps c and d until the complete function has been entered.

11. Contine execution

a. Type cont N , where N designates the location where program execution should begin (do not press the CONTINUE key. You must type "cont" character by character).

b. Press EXEUCTE

12. Your program is ready to run.

ANNOTATED LISTING:

DRIVER - NUMERICAL INTEGRATION - ROMBERG

```
0: ldf 15,11,1
1: dsp "DEFINE
 FUNCTION";stp
2: cfg 13;spc ;
 enp "MAX NO OF
 ITER?",M;spc ;
 if flg13;cfg
 13;gto +0
3: enp "LOWER
 BOUND?",A;spc ;
 if flg13;cfg
 13;gto +0
4: enp "UPPER
 BOUND?",B;spc ;
 if flg13;cfg
 13;gto +0
5: enp "EPSILON"
 ,E;spc ;if flg1
 3;cfg 13;1e-6→E
6: sfg 3;enp
 "PRINT TABLEAU?
 ",R;spc 2;if
 flg13;cfg 3
7: gsb "ROMBERG"
8: if flg2;stp
9: spc 2;prt "--
 ----------
 ";spc 2
10: prt "INTEGRA
 L FROM",A,"
        TO",B,"
 ","IS",T[0];
 spc 4;end
*11074
```

Load the subroutine

User must define his function in the sub-routine

Maximum number of times interval is halved in computation of initial trazoidal integrals

Lower integration bound

Upper integration bound

Error tolerance between successive computa-tions of the integral

Should tableau of integral approximations be printed out?

Branch to subroutine

Stop if the error flag is set

Print the integral

ANNOTATED LISTING:

NUMERICAL INTEGRAION - ROMBERG

```
0: "ROMBERG":dim
  T[0:M];cfg 2;
 if A>B;sfg 2;
 dsp "BOUNDS
 REVERSED";ret
```

Error check to see if integral bounds are in correct order

```
1: 0→S;A→X;gsb
 "EVAL"
2: Y+S→S;B→X;
 gsb "EVAL"
3: Y+S→S;((B-A)/
 2)S→T[0];0→N
4: if (N+1→N)>M;
 spc 2;gto +6
5: (2(2↑(N-1)→R)
 →Q)-1→P
6: -1→I;0→S
7: if (I+2→I)>P;
 (((B-A)/R)S+
 T[N-1])/2→T[N];
 gto -3
8: A+((B-A)/Q)I→
 X;gsb "EVAL"
9: S+Y→S;gto -2
```

COMPUTE FIRST COLUMN OF TABLEAU USING TRAPEZOIDAL RULE

$$T_N = T_{N-1} + \frac{(B-A)}{2^{N-1}} \sum_{\substack{I=1 \\ \Delta I=2}}^{2^N-1} f\left(A + \frac{(B-A)}{2^N} I\right)$$

The trapezoidal approximations are computed for 2^1 up to 2^m intervals

Branch to subroutine to compute functional value (Y) of X

```
10: 1→J
11: if (J+1→J)>M
 +1;ret
12: -1→N;sfg 6;
 if flg3;gsb
 "PRINT"
13: if (N+1→N)>M
 +1-J;gto -2
14: T[N]→F;((4↑(
 J-1)→G)T[N+1]-
 T[N])/(G-1)→T[N
 ];if flg6;cfg
 6;gto -1
```

COMPUTE THE REMAINING COLUMNS USING EXTRAPOLATION FORMULA

If flag 3 is set go to print routine

$$T_{N,J} = \frac{4^{J-1} T_{N+1,J-1} - T_{N,J-1}}{4^{J-1} - 1}$$

```
15: if abs(F-
 T[N])<E;T[N]→T[
 0];ret
16: gto -3
```

Check for error tolerance being satisfied

```
17: "EVAL"
18: beep;beep;
 dsp "DEFINE
 FUNCTION";stp
19: ret
```

Line 18 will contain the user defined function

```
20: "PRINT":0→I
21: prt T[I];
 jmp (I+1→I)>M+
 1-J
22: spc 1;ret
```

Subroutine for optional printing of the intermediate results

```
*3172
```

Numerical Integration Of Equally-Spaced Data Points (Simpson's One-Third Rule)

FILE 16 DRIVER

FILE 17 SUBROUTINE "SIMPEQ"

Numerical Integration Of Equally-Spaced Data Points (Simpson's One-Third Rule)

This program approximates $\int_a^b f(x)dx$ where f(x) is represented by discrete function values $f(x_i)$ at equally spaced points x_i. The program requests the necessary information, calls the subroutine "SIMPEQ" in file 17 on page 115 and then prints the results computed by the subroutine.

SPECIAL CONSIDERATIONS AND DIAGNOSTIC MESSAGES:

1. The maximum number of function values that can be entered with this subroutine will depend on the size of the calling program. The following limitations assume that the calling program in file 15 is used. If N is the number of function values, then:

   N < 744 on the 6,844 byte machine
   N < 1768 on the 15,036 byte machine
   N < 2792 on the 23,228 byte machine
   N < 3816 on the 31,420 byte machine

2. "MORE POINTS" indicates that less than three points were entered.

3. "ODD NO. OF PTS" indicates that you need an odd number of functional values.

FORMULAE:

$$\int_a^b f(x)dx \approx \frac{h}{3}\{f(a) + 4f(a+h) + 2f(a+2h) + 4f(a+3h) + \ldots + 4f(a+(n-1)h) + f(a+nh)\}$$

where: n = number of intervals (number of data points minus one)

$$h = \frac{b - a}{n}$$

REFERENCES:

1. Beckett, Royce and Hunt, James Numerical Calculations and Algorithms (New York: McGraw-Hill, 1967), pp. 166 - 169.

INSTRUCTIONS:

1. Insert UTILITY ROUTINES cartridge with machine turned on.
2. Load the file.
   a. Type ldf 16
   b. Press EXECUTE
3. (Optional)
   a. Type fxd N or flt N where N is the number of digits printed to the right of the decimal point ($0 \leq N \leq 11$).
   b. Press EXECUTE
4. Press RUN
5. When "INCREMENT?" is displayed:
   a. Enter the increment between successive functional values.
   b. Press CONTINUE
6. When "NO. OF POINTS?" is displayed:
   a. Enter the number of data points to be entered.

   NOTE: This number must be odd.
   b. Press CONTINUE
7. "FUNCTION VALUES" will be displayed briefly.
8. "F[I]" will be displayed where "I" should indicate the I^{th} functional value.
   a. Enter the functional value of the I^{th} data point.
   b. Press CONTINUE
   c. Repeat step 8 until all function values have been entered.
9. When "CHANGES?" is displayed:

   Either
   a. Press CONTINUE if no changes are desired.
   b. Go to step 12

   or
   a. Enter 1 if you desire to correct one or more functional values.
   b. Press CONTINUE
10. When "DATA PT. NO?" is displayed:
    a. Enter the number of the data point.
    b. Press CONTINUE

11. When "F[I]?" is displayed:

    a. Enter the corrected functional value for the I^{th} data point.

    b. Press CONTINUE

    c. Go to step 9.

12. The integral will be printed.

x	f(x)
0	0
.1	.1
.2	.2
.3	.3
.4	.4
.5	.5
.6	.6
.7	.7
.8	.8
.9	.9
1	1

```
INCREMENT?
.1

NO. OF POINTS?
11

F[1]?
0

F[2]?
.1

F[3]?
.2

F[4]?
.3

F[5]?
.4

F[6]?
.5

F[7]?
.6

F[8]?
.7

F[9]?
.8

F[10]?
.9

F[11]?
1

CHANGES?


INTEGRAL=
        0.500000
```

x	f(x)
0	0
.25	2.8
.50	3.8
.75	5.2
1.00	7.0
1.25	9.2
1.50	12.1
1.75	15.6
2.00	20

```
INCREMENT?
.25

NO. OF POINTS?
9

F[1]?
0

F[2]?
2.8

F[3]?
3.8

F[4]?
5.2

F[5]?
7.0

F[6]?
9.2

F[7]?
12.1

F[8]?
15.6

F[9]?
20

CHANGES?


INTEGRAL=
        16.416667
```

"SIMPEQ" SUBROUTINE

This subroutine approximates $\int_a^b f(x)\,dx$ where discrete values of f(x) are known at equally spaced base points over the interval [a, b]. Simpson's one-third rule is used to approximate the integral.

The user must supply the increment between intervals, the number of data points, which must be odd, and the functional value at each of the data points. Since the truncation error using this method is $o(h^4)$, the approximate size of the error can be estimated from the interval size.

When used as a subroutine, all parameters specified under <INPUT> must be assigned in the user's calling program. The user may then append the subroutine to his program and use gsb "SIMPEQ" to take advantage of its capabilities. The results of the calculation and any other output is designated under <OUTPUT>. The user should be aware that the subroutine may alter variables that he is using in his program and should therefore check the variables listed under <DESTROYED>.

An example of a typical program which uses this subroutine may be found on page 118 or in file 16.

PARAMETERS AND VARIABLES USED:

<INPUT>

N	the number of data points (must be odd).
D	the increment, ΔX, between equally spaced data points.
F[N]	should contain the N discrete functional values in increasing order of domain value.

<OUTPUT>

N, D, F[N]	unchanged
S	integral $\int_a^b f(x)\,dx$
flg 2	set if less than three function points
flg 3	set if number of points is even

<DESTROYED>

I	loop counter used as subscript for functional values

INSTRUCTIONS:

1. Load your program into memory. The program should set up all necessary parameters (N, D, F[ ]) to be passed to the subroutine.

2. Insert UTILITY ROUTINES cartridge.

3. (Optional)
   a. Type fxd N or flt N where N is the number of digits printed to the right of the decimal point ($0 \leq N \leq 11$).
   b. Press EXECUTE

4. Load the file.
   a. Type ldf 17, N, M where:
      N = line number where the first line of the subroutine should be loaded into memory.
      M = line number where execution is to begin after subroutine is loaded.
   b. Press EXECUTE

ANNOTATED LISTING:

DRIVER - NUMERICAL INTEGRATION - EQUALLY SPACED POINTS - SIMPSON

```
0: cfg 13;ldf
 17,11,1
```
Load the subroutine after your program

```
1: enp "INCREMEN
 T?",D;spc ;if
 flg13;cfg 13;
 gto +0
```
Enter increment size between successive X values

```
2: enp "NO. OF
 POINTS?",N;spc
 ;if flg13;cfg
 13;gto +0
```
Enter number of data points

```
3: dim F[N];1→I;
 dsp "FUNCTION
 VALUES?";spc ;
 wait 1000
4: cfg 13;enp
 F[I];spc ;gto +
 0;if not flg13;
 jmp (I+1→I)>N
```
Input functional values for data points

```
5: spc ;enp "CHA
 NGES?",I;spc ;
 if flg13;gto +3
6: enp "DATA
 PT. NO.?",I;
 spc ;if flg13;
 cfg 13;gto +0
7: cfg 13;enp
 F[I];spc ;gto +
 0;if not flg13;
 gto -2
```
Make necessary corrections, if any

```
8: gsb "SIMPEQ"
```
Branch to subroutine

```
9: if flg2 or
 flg3;stp
```
Stop if error flags are set

```
10: spc 2;prt
 "INTEGRAL=",S;
 spc 4;end
```
Print integral

```
*26305
```

ANNOTATED LISTING:

SUBROUTINE - NUMERICAL INTEGRAION OF EQUALLY SPACED POINTS - SIMPSON


```
0: "SIMPEQ":cfg
 2,3;if N<3;sfg
 2;dsp "MORE
 POINTS";ret
```
Check if not enough points

```
1: if int(N/2)=N
 /2;dsp "ODD
 NO. OF PTS";
 sfg 3;ret
```
Check if number of points is odd

```
2: F[1]+4F[2]+
 F[N]→S
```
Functional values of first two points and last point initially stored in S

```
3: 1→I
4: if (I+2→I)>N-
 1;SD/3→S;ret
5: S+2F[I]+4F[I+
 1]→S;gto -1
```
Add successive pairs of functional values with a multiplier of 2 or 4

```
*5262
```

Numerical Integration Of Discrete, Unequally-Spaced Data Points

FILE 18 DRIVER

FILE 19 SUBROUTINE "INT"

Numerical Integration Of Discrete, Unequally-Spaced Data Points

This program approximates $\int_a^b f(x)\,dx$ where f(x) is represented by discrete functional values for unequally spaced domain values x over the interval [a, b]. The program requests all necessary information, calls the subroutine "INT" in file 19 on page 127 and prints the results computed by the subroutine. The user must input the data points $(x_i y_i)$, i = 1,n with the restriction that the x_i are discrete and $x_i < x_{i+1}$ for i = 1, n-1.

SPECIAL CONSIDERATIONS:

The maximum number of data points that can be entered with this subroutine will depend on the size of the calling program. The following limitations assume that the calling program in file 18 is used.

If N is the number of data points $(x_i y_i)$, then N < 141 on the 6844 byte machine, N < 1165 on the 15,036 byte machine, N < 2189 on the 23,228 byte machine and N < 3213 on the 31,420 byte machine.

FORMULA:

$$\int_{x_i}^{x_{i+1}} s(x)\,dx \approx \sum_{i=1}^{n-1} \left\{\frac{1}{2}(x_{i+1}-x_i)\,(y_i+y_{i+1}) - \frac{1}{24}(x_{i+1})^3\,[S''(x_i) + S''(x_{i+1})]\right\}$$

REFERENCES:

1. Ralston and Wilf, Mathematical Methods for Digital Computers, Vol. II New York: John Wiley and Sons, 1967) pp. 156 - 158.
2. Greville, T.N.E., Editor, "Proceedings of An Advanced Seminar Conducted by the Mathematics Research Center", U.S. Army, University of Wisconsin, Madison. October 7 - 9, 1968. Theory and Application of Spline Functions(New York, London: Academic Press, 1969), pp. 156-167.

INSTRUCTIONS:

1. Insert UTILITY ROUTINES cartridge with machine turned on.

2. Load the file.
   a. Type ldf 18
   b. Press EXECUTE

3. (Optional)
   a. Type fxd N or flt N where N is the number of digits printed to the right of the decimal point ($0 \leq N \leq 11$).
   b. Press EXECUTE

4. Press RUN

5. When "NO. OF DATA PTS?" is displayed:
   a. Enter the number of data points in the data set.
   b. Press CONTINUE

6. When "X[I]" is displayed, "I" is the number of the next data point to be entered.
   a. Enter the domain value of the I^{th} data point.
   b. Press CONTINUE

7. When "Y[I]" is displayed:
   a. Enter the functional value of the I^{th} data point.
   b. Press CONTINUE
   c. If you have not entered all the data, go to step 6.

8. When "CHANGES?" is displayed:
   Either
   a. Press CONTINUE if you do not need to make any changes to the entered data.
   b. Go to step 12.
   or
   a. Enter 1 if you need to make changes.
   b. Press CONTINUE

9. When "DATA PT NO.?" is displayed:
   a. Enter the number of the data point that is to be changed.
   b. Press CONTINUE

10. When "X[I]" is displayed:
    a. Enter the new domain value for the I^{th} data point
    b. Press CONTINUE

11. When "Y[I]" is displayed:
    a. Enter the new functional value for the I^{th} data point.
    b. Press CONTINUE
    c. Go to step 8.

12. When "EPSILON?" is displayed:
    Either
    Press CONTINUE and default value of 10^{-6} will be assigned to epsilon.
    or
    a. Enter the error tolerance desired for the iterative solution of the second derivatives.
    b. Press CONTINUE

13. The integral will be printed.

EXAMPLE:

X	Y
0	0
1	1
2	2
3	3
4	4

```
NO. OF DATA PTS?
5

X[1]?
0
Y[1]?
0

X[2]?
1
Y[2]?
1

X[3]?
2
Y[3]?
2

X[4]?
3
Y[4]?
3

X[5]?
4
Y[5]?
4

CHANGES?

EPSILON?
.00001

INTEGRAL FROM
          0.000000
             TO
          4.000000

IS        8.000000
```

X	Y
0	0
.2	.04
.6	.36
1	1
1.1	1.21
1.5	2.25
1.6	2.56
2	4

```
NO. OF DATA PTS?
8

X[1]?
0
Y[1]?
0

X[2]?
.2
Y[2]?
.04

X[3]?
.6
Y[3]?
.36

X[4]?
1
Y[4]?
1

X[5]?
1.1
Y[5]?
1.21

X[6]?
1.5
Y[6]?
2.25

X[7]?
1.6
Y[7]?
2.56

X[8]?
2
Y[8]?
4

CHANGES?

EPSILON?
.0000001

INTEGRAL FROM
          0.000000
             TO
          2.000000

IS        2.669976
```

"INT" SUBROUTINE

This subroutine approximates $\int_a^b f(x)dx$ where f(x) is represented by discrete functional values for unequally spaced domain values x over the interval [a, b].

The method implemented involves fitting a curve through the data points and integrating that curve. The curve used is the cubic natural spline function which derives its name from a draftsman's mechanical spline. If the spline is considered as a function represented by s(x), the second derivative s"(x) approximates the curvature. For the curve through data points (x_1,y_1), (x_2,y_2), ..., (x_n,y_n) we want $\int_{x_1}^{x_n} (s"(x))^2 dx$ to be minimized in order to achieve the "smoothest" curve.

The spline function with minimum curvature has cubic polynomials between adjacent data points. Adjacent polynomials are joined continuously with continuous first and second derivatives as well as $s"(x_1) = s"(x_n) = 0$.

The procedure to determine s(x) involves the iterative solution of a set of simultaneous linear equations by the method of successive over-relaxation. The accuracy to which these equations are solved is specified by the user. For a detailed discussion of the algorithm, see Reference 2.

When used as a subroutine, all parameters specified under <INPUT> must be assigned in the user's calling program. The user may then append the subroutine to his program and use gsb "INT" to take advantage of its capabilities. The results of the calculation and any other output is designated under <OUTPUT>. The user should be aware that the subroutine may alter variables that he is using in his program and should therefore check the variables listed under <DESTROYED>.

The data points (x_i, y_i), i = 1,n should be entered in increasing order of the x_i, where the x_i are discrete and $x_i < x_{i+1}$ for i = 1, n-1. An example of a typical program which uses this subroutine may be found on page 131 or in file 18.

PARAMETERS AND VARIABLES USED:

<INPUT>

N	number of data points
X[N]	domain values of data points
Y[N]	range values of data points
E	error tolerance for solution of simultaneous equations.

<OUTPUT>

N, X[N], Y[N], E	unchanged
A	integral

<DESTROYED>

H	temporary storage for $x_{i+1}-x_{i-1}$; abs(T); $x_{i+1}-x_i$
I	loop counter used as subscript for vectors
T	$\dfrac{\dfrac{y_{i+1}-y_i}{x_{i+1}-x_i} - \dfrac{y_i-y_{i-1}}{x_i-x_{i-1}}}{H}$
U	(μ) current maximum value of $\lvert T \rvert$ used to test for error tolerance.
W	$8 - 4\sqrt{3}$, relaxation factor for iterative solution.
X	temporary storage for $x_i - x_{i-1}$
B[i]	$\frac{1}{2}\left(\dfrac{x_i - x_{i-1}}{x_{i+1} - x_{i-1}}\right)$

G[i]	3T
S[i]	2T or $S''(x_i)$ vector of second derivatives

INSTRUCTIONS:

1. Load your program into memory. Your program should set up all necessary parameters (N, X[N], Y[N], E) to be passed to the subroutine.

2. Insert UTILITY ROUTINES cartridge.

3. (Optional)
   a. Type fxd N or flt N where N is the number of digits printed to the right of the decimal point ($0 \leq N \leq 11$).
   b. Press EXECUTE

4. Load the file.
   a. Type ldf 19, N, M where:
      N = line number where the first line of the subroutine should be loaded into memory.
      M = line number where execution is to begin after subroutine is loaded.
   b. Press EXECUTE

```
0: cfg 13;ldf
 19,13,1
```
Load the subroutine after the program

```
1: enp "NO. OF
 DATA PTS?",N;
 spc ;if flg13;
 cfg 13;gto +0
2: dim X[N],Y[N]
 ;1→I
```
Enter the number of points to be computed and dimension data arrays

```
3: enp X[I];if
 flg13;cfg 13;
 gto +0
4: enp Y[I];spc
 ;if flg13;cfg
 13;gto +0
5: if (I+1→I)<=N
 ;gto -2
```
Enter the (X, Y) coordinates into the X and Y arrays

```
6: spc ;enp "CHA
 NGES?",I;spc ;
 if flg13;cfg
 13;gto +4
7: enp "DATA PT
 NO.?",I;spc ;
 if flg13;cfg
 13;gto +0
8: enp X[I];if
 flg13;cfg 13;
 gto +0
9: cfg 13;enp
 Y[I];spc ;gto +
 0;if not flg13;
 gto -3
```
Make any necessary corrections to the data points

```
10: enp "EPSILON
 ?",E;if flg13;
 1e-6→E
```
Enter error tolerance

```
11: gsb "INT"
```
Branch to subroutine

```
12: spc 2;prt
 "INTEGRAL FROM"
 ,X[1],"
    TO",X[N],"
 ","IS",A;spc 4;
 end
```
Print integral

```
*30698
```

ANNOTATED LISTING:

NUMBER INTEGRATION OF UNEQUALLY SPACED POINTS

```
0: "INT":dim
 B[2:N-1],G[2:N-
 1],S[N];1→I
1: if (I+1→I)>N-
 1;sto +3
2: .5(X[I]-X[I-
 1]→X)/(X[I+1]-
 X[I-1]→H)→B[I]
```

$$\frac{X_i - X_{i-1}}{X_{i+1} - X_{i-1}} \rightarrow X \qquad \frac{1}{2}\,\frac{(X_i - X_{i-1})}{(X_{i+1} - X_{i-1})} \rightarrow B[]$$

$$\frac{X_i - X_{i-1}}{X_{i+1} - X_{i-1}} \rightarrow H$$

```
3: 2(((Y[I+1]-
 Y[I])/(X[I+1]-
 X[I])-(Y[I]-
 Y[I-1])/X)/H→T)
 →S[I];3T→G[I];
 sto -2
```

$$\frac{\frac{Y_{i+1} - Y_i}{X_{i+1} - X_i} - \frac{Y_i - Y_{i-1}}{X_i - X_{i-1}}}{H} \rightarrow T \qquad 2T \rightarrow S[] \qquad 3T \rightarrow G[]$$

```
4: 0→S[1]→S[N];
 8-4r3→W
```

W is relaxation factor

```
5: 0→U;2→I
6: W(-S[I]-B[I]S
 [I-1]-(.5-B[I])
 S[I+1]+G[I])→T
```

U is largest correction factor

Compute new correction factor T

```
7: if (abs(T)→H)
 >U;H→U
```

Larger correction factor stored in U

```
8: S[I]+T→S[I]
9: if I#N-1;I+
 1→I;sto -3
```

Add correction factor to S[] = 2nd derivative approximation

```
10: if U>=E;sto
 -5
```

If correction factor is larger than error tolerance, recompute

```
11: 0→I→A
12: if (I+1→I)>N
 -1;ret
13: A+(.5(X[I+
 1]-X[I]→H)(Y[I]
 +Y[I+1])-(1/
 24)H↑3(S[I]+
 S[I+1]))→A;sto
 -1
```

INTEGRAL COMPUTED

$$\sum_{i=1}^{n-1} \left\{\frac{1}{2} H(Y_i + Y_{i+1}) - \frac{1}{24} H^3 (S[I] + S[I+1])\right\} \rightarrow A$$

where $S''(X_i) = S(I)$

```
*2700
```

Numerical Interpolation, Differentiation & Integration Of Discrete, Unequally-Spaced Data

FILE 20 DRIVER

FILE 21 SUBROUTINE "DERINT"

Numerical Interpolation, Differentiation & Integration Of Discrete, Unequally-Spaced Data

This program computes a third-degree polynomial which passes through n data points (x_i, y_i), i = 1,n supplied by the user. From this curve s(x) the user can obtain the integral over the interval (x_1, x_n) and the derivative or functional value at any point on the interval. The program requests all necessary information and prints out the results after calling the subroutine "DERINT" in file 21 on page 141. The only restriction on the data is that the x_i must be discrete and $x_i < x_{i+1}$ for i=1, n-1.

SPECIAL CONSIDERATIONS AND DIAGNOSTIC MESSAGES:

1. The maximum number of data points and output arguments that can be accepted by the subroutine depends on the size of the calling program. The following limitations assume that the calling program in file 20 is used.

   If N is the number of data points and M is the number of output arguments then:

   4N + 2M + maximum (N,M) < 609 on the 6844 byte machine
   4N + 2M + maximum (N,M) < 1634 on the 15,036 byte machine
   4N + 2M + maximum (N,M) < 2658 on the 23,228 byte machine
   4N + 2M + maximum (N,M) < 3682 on the 31,420 byte machine

2. "ARG OUT OF BOUND" indicates that a derivative or functional value was requested for a point outside the data interval.

FORMULAE:

$$\int_{x_i}^{x_{i+1}} s(x)\,dx \approx \{\frac{1}{2}(x_{i+1} - x_i)(y_i + y_{i+1}) - \frac{1}{24}(x_{i+1} - x_i)^3 [s''(x_i) + s''(x_{i+1})]\}$$

$$s(t_j) = y_i + (t_j - x_i)\left(\frac{y_{i+1} - y_i}{x_{i+1} - x_i}\right) + (t_j - x_i)(t_j - x_{i+1})\frac{1}{6}[s''(x_i) + s''(x_{i+1}) + s''(t_j)]$$

$$s'(t_j) = \frac{y_{i+1} - y_i}{x_{i+1} - x_i} + \frac{1}{6}[(t_j - x_i) + (t_j - x_{i+1})](s''(x_i) + s''(t_j)) + \frac{1}{6}(t_j - x_i)(t_j - x_{i+1})\left(\frac{s''(x_{i+1}) - s''(x_i)}{x_{i+1} - x_i}\right)$$

REFERENCES:

1. Ralston and Wilf, Mathematical Methods for Digital Computers, Vol. II (New York: John Wiley and Sons, 1967) pp. 156 - 158.

2. Greville, T.N.E., Editor, "Proceedings of An Advanced Seminar Conducted by the Mathematics Research Center", U.S. Army, University of Wisconsin, Madison. October 7 - 9, 1968. Theory and Applications of Spline Functions (New York, London: Academic Press, 1969), pp. 156 - 167.

INSTRUCTIONS:

1. Insert UTILITY ROUTINES cartridge with machine turned on.

2. Load the file.
   a. Type ldf 20
   b. Press EXECUTE

3. (Optional)
   a. Type fxd N or flt N where N is the number of digits printed to the right of the decimal point ($0 \leq N \leq 11$).
   b. Press EXECUTE

4. Press RUN

5. When "NO. OF DATA PTS?" is displayed:
   a. Enter the number of data points in the data set.
   b. Press CONTINUE

6. When "X[I]" is displayed, "I" is the number of the data point to be entered.
   a. Enter the domain value of the I^{th} data point.
   b. Press CONTINUE

7. When "Y[I]" is displayed:
   a. Enter the functional value of the I^{th} data point.
   b. Press CONTINUE
   c. If you have not entered all the data, go to step 6.

8. When "CHANGES?" is displayed:
   Either
   a. Press CONTINUE if you do not need to make any changes to the entered data.
   b. Go to step 12.
   or
   a. Enter 1 if you need to make changes.
   b. Press CONTINUE

9. When "DATA PT NO?" is displayed:
   a. Enter the number of the data point that is to be changed.
   b. Press CONTINUE

10. When "X[I]" is displayed:
    a. Enter the new domain value for the I^{th} data point.
    b. Press CONTINUE

11. When "Y[I]" is displayed:
    a. Enter the new functional value for the I^{th} data point.
    b. Press CONTINUE
    c. Go to step 8.

12. When "NO. OUTPUT ARGS?" is displayed:
    a. Enter the number of arguments for which you want the derivative or functional value computed.
    b. Press CONTINUE

13. When "T[J]" is displayed, "J" is the number of the next output argument.
    a. Enter the next output argument that you want to be evaluated.
    b. Press CONTINUE
    c. If you have not entered all the desired output arguments, repeat step 13.

14. When "CHANGES?" is displayed:
    Either
    a. Press CONTINUE if you do not need to make any changes in the output arguments.
    b. Go to step 17.
    or
    a. Enter 1 if you need to make changes.
    b. Press CONTINUE

15. When "OUTPUT ARG NO?" is displayed:
    a. Enter the number of the output argument to be changed.
    b. Press CONTINUE

16. When "T[J]" is displayed, "J" is the number of the output argument to be changed.
    a. Enter the corrected value of the output argument.
    b. Press CONTINUE
    c. Go to step 8.

17. When "EPSILON?" is displayed:

Either

Press CONTINUE , if you want the default value of 10^{-6} assigned to the error tolerance.

or

a. Enter the error tolerance to which the simultaneous equations will be solved.

b. Press CONTINUE

18. The functional values and derivatives of all output arguments should be printed as well as the integral over the interval (x_1, x_n).

EXAMPLE:

X	Y
0	1
.2	.72
.5	.75
1	2
1.2	2.92
1.6	5.48
1.8	7.12
2	9

```
NO OF DATA PTS?
8

X[1]?
0
Y[1]?
1

X[2]?
.2
Y[2]?
.72

X[3]?
.5
Y[3]?
.75

X[4]?
1
Y[4]?
2

X[5]?
1.2
Y[5]?
2.92

X[6]?
1.6
Y[6]?
5.48

X[7]?
1.8
Y[7]?
7.12

X[8]?
2
Y[8]?
9

CHANGES?
```

```
NO. OUTPUT ARGS?
2

T[1]?
.81

T[2]?
1.1

CHANGES?


EPSILON?
.00001

INTEGRAL FROM
          0.000000
            TO
          2.000000

IS        6.002930


X=        0.810000
S[X]=     1.350580
S'[X]=    2.848500

X=        1.100000
S[X]=     2.429585
S'[X]=    4.600004
```

OUTPUT ARGUMENTS

X	Y	Y'
.81	1.35058	2.8485
1.1	2.429585	4.600004

"DERINT" SUBROUTINE

This subroutine computes a curve s(x) that passes through the n data points (x_i, y_i) supplied by the user and computes certain information at any point t_j on the curve as long as t_j is in the interval $[x_1, x_n]$. The information that can be computed is the integral over the interval and the derivative or functional value at any point on the interval.

The method implemented fits a curve through the points and integrates, differentiates and interpolates that curve. The curve used is the cubic natural spline which derives its name from a draftsman's mechanical spline. If the spline is considered as a function represented by s(x), then the second derivative s"(x) approximates the curvature. For the curve through data points (x_1, y_1), (x_2, y_2), ... , (x_n, y_n) we want $\int_{x_1}^{x_n} (s"(x))^2 \, dx$ to be minimized in order to achieve the "smoothest" curve.

The spline function with minimum curvature has cubic polynomials between adjacent data points. Adjacent polynomials are joined continuously with continuous first and second derivatives as well as $s"(x_1) = s"(x_n) = 0$.

The procedure to determine s(x) involves the iterative solution of a set of simultaneous linear equations by the method of successive over-relaxation. The accuracy to which these equations are solved is specified by the user. For a detailed discussion of the algorithm, see Reference 2.

When used as a subroutine, all parameters specified under <INPUT> must be assigned in the user's calling program. The user may then append the subroutine to his program and use gsb "DERINT" to take advantage of its capabilities. The results of the calculation and any other output is designated under <OUTPUT>. The user should be aware that the subroutine

may alter variables that he is using in his program and should therefore check the variables listed under <DESTROYED>.

The data points (x_i, y_i), i = 1,n should be entered in increasing order of the x_i, where the x_i are discrete and $x_i < x_{i+1}$ for i = 1,n-1. The output arguments t_j, j = 1,m may be entered in any order.

The error factor of the solutions is approximately equal to h^4 for the integral, h^3 for the functional values and h^2 for the derivative, where h is the average interval size.

An example of a typical program which uses this subroutine may be found on page 145 or in file 20.

PARAMETERS AND VARIABLES USED:

<INPUT>

N	number of data points
X[N]	domain values of data points
Y[N]	range values of data points
M	number of output arguments desired
T[M]	domain values of output arguments
E	error tolerance in iterative solution of simultaneous equations.

<OUTPUT>

N, X[N], Y[N], M, T[M], E	unchanged
A	integral $\int_{x_1}^{x_n} s(x)\,dx$
B[M]	function values for output arguments in T[M]
D[M]	derivative values for output arguments in T[M]
flg 3	if set, indicates an output argument out of bounds.

<DESTROYED>

H temporary storage for $x_{i+1} - x_{i-1}$, abs(T) , $x_{i+1} - x_i$, or $t_j - x_i$

I loop counter

J loop counter

X $s''(t_j) + 3(t_j - x_i)T$

T $\dfrac{\dfrac{y_{i+1} - y_i}{x_{i+1} - x_i} - \dfrac{y_i - y_{i-1}}{x_i - x_{i-1}}}{H}$; $t_j - x_{i+1}$

U (μ) current maximum value of $|T|$ used to test error tolerance; $\frac{1}{6}(s''(x_i) + s''(x_{i+1})) + s''(x_i) + 3T(t_j - x_i)$

W $8-4\sqrt{3}$, relaxation factor for iterative solution; $\dfrac{y_{i+1} - y_i}{x_{i+1} - x_i}$

X temporary storage for $x_i - x_{i-1}$; $(t_j - x_i)(t_j - x_{i+1})$

Z 1/6

B[i] $1/2\left[\dfrac{x_i - x_{i-1}}{x_{i+1} - x_{i-1}}\right]$; function values $s(t_j)$

G[i] 3T

S[i] 2T or $s''(t_j)$ vector of second derivatives.

INSTRUCTIONS:

1. Load your program into memory. The program should set up all necessary parameters (N, X[N], Y[N], E, M, T[M]) to be passed to the subroutine.

2. Insert UTILITY ROUTINES cartridge.

3. (Optional)
   a. Type `fxd N` or `flt N` where N is the number of digits printed to the right of the decimal point ($0 \leq N \leq 11$).
   b. Press EXECUTE

4. Load the file.
   a. Type `ldf 21, N, M` where:

      N = line number where the first line of the subroutine should be loaded into memory.

      M = line number where execution is to begin after subroutine is loaded.

   b. Press EXECUTE

ANNOTATED LISTING:

NUMERICAL INTERPOLATION, DIFFERENTIATION, AND INTEGRATION OF UNEQUALLY SPACED POINTS

```
0: cfg 13;ldf
 21,22,1
```
Load subroutine after the program

```
1: enp "NO OF
 DATA PTS?",N;
 spc ;if flg13;
 cfg 13;gto +0
2: dim X[N],Y[N]
 ;1→I
```
Enter number of data points and dimensions the X and Y arrays

```
3: enp X[I];if
 flg13;cfg 13;
 gto +0
4: enp Y[I];if
 flg13;cfg 13;
 gto +0
5: if (I+1→I)<=N
 ;spc ;gto -2
```
Enter the (X, Y) coordinates into the X and Y arrays

```
6: spc ;enp "CHA
 NGES?",I;spc 2;
 if flg13;cfg
 13;gto +4
7: enp "DATA
 PT. NO.?",I;
 spc ;if flg13;
 cfg 13;gto +0
8: enp X[I];if
 flg13;cfg 13;
 gto +0
9: cfg 13;enp
 Y[I];gto +0;if
 not flg13;spc ;
 gto -3
```
Make any necessary corrections to the X and Y arrays

```
10: enp "NO.
 OUTPUT ARGS?",
 M;spc ;if flg13
 ;cfg 13;gto +0
11: 1→I;if M#0;
 dim T[M]
12: cfg 13;enp
 T[I];gto +0;if
 not flg13;spc ;
 jmp (I+1→I)>M
```
Enter the X values for which the information is to be computed

```
13: spc ;enp
 "CHANGES?",I;
 spc 2;if flg13;
 cfg 13;gto +3
14: enp "OUTPUT
 ARG NO?",I;if
 flg13;cfg 13;
 gto +0
```
Make any necessary corrections to the output arguments

ANNOTATED LISTING:

```
15: cfg 13;enp
 T[I];sto +0;if
 not flg13;spc ;
 sto -2
16: enp "EPSILON
 ?",E;if flg13;
 1e-6→E
17: gsb "DERINT"
18: if flg3;stp
19: spc 2;prt
 "INTEGRAL FROM"
 ,X[1],"
   TO",X[N]," ",
 "IS",A;1→J;spc
 2
20: prt "X=",
 T[J],"S[X]=",
 B[J],"S'[X]=",
 D[J];spc ;jmp
 (J+1→J)>M
21: spc 4;end
*19437
```

Enter error tolerance

Branch to subroutine

Stop if error flag is set

Print integral, derivatives, and funtional values

T[J] = value of domain arguments
B[J] = functional values
D[J] = derivatives

ANNOTATED LISTING:

NUMERICAL INTERPOLATION, DIFFERENTIATION, AND INTEGRATION OF UNEQUALLY SPACED POINTS

```
0: "DERINT":cfg
 3:if M>N;dim
 B[M];gto +2
1: dim B[N]
2: dim S[N],G[N-
 1];1→I
```

Dimension functional value array to the larger of M and N

```
3: if (I+1→I)>N-
 1;gto +3
4: .5(X[I]-X[I-
 1]→X)/(X[I+1]-
 X[I-1]→H)→B[I]
```

$X_i - X_{i-1} \rightarrow X$

$X_{i+1} - X_{i-1} \rightarrow H \quad \frac{1}{2} \quad \frac{(X_i - X_{i-1})}{(X_{i+1} - X_{i-1})} \rightarrow B[I]$

```
5: 2(((Y[I+1]-
 Y[I])/(X[I+1]-
 X[I])-(Y[I]-
 Y[I-1])/X)/H→T)
 →S[I];3T→G[I];
 gto -2
```

$$\frac{\frac{Y_{i+1}-Y_i}{X_{i+1}-X_i} - \frac{Y_i-Y_{i-1}}{X_i-X_{i-1}}}{H} \rightarrow T \qquad 2T \rightarrow S[I] \qquad 3T \rightarrow G[I]$$

```
6: 0→S[1]→S[N];
 8-4r3→W
```

W is relaxation factor

```
7: 0→U;2→I
8: W(-S[I]-B[I]S
 [I-1]-(.5-B[I])
 S[I+1]+G[I])→T
```

U is largest correction factor
Compute new correction factor T

```
9: if (abs(T)→H)
 >U;H→U
```

Larger correction factor stored in U

```
10: S[I]+T→S[I]
11: if I≠N-1;I+
 1→I;gto -3
```

Add correction factor to S[I] = 2nd derivative

```
12: if U>=E;gto
 -5
13: 0→I
14: if (I+1→I)>N
 -1;gto +2
15: (S[I+1]-S[I]
 )/(X[I+1]-X[I])
 →G[I];gto -1
```

If correction factor is larger than error tolerance, recompute

Compute $s'''(i) = \frac{s''(X_{i+1}) - s''(X_i)}{X_{i+1} - X_i}$

where s'''(i)=G[I] and s''(i)=S[I]

```
16: if M=0;gto +
 13
```

If no output arguments skip this section

```
17: 0→J;dim D[M]
18: if (J+1→J)>M
 ;gto +11
19: 1→I;if (T[J]
 →T)>=X[1];gto +
 2
```

Check that argument T[J] is in bounds

$$X[0] \leq T[J] \leq X[N]$$

ANNOTATED LISTING:

```
20: sfg 3;dsp
 "ARG OUT OF
 BOUND";ret
21: if (I+1→I)>N
 ;gto -1                    Compute I such that X[I] ≤ T[J] ≤ X[I+1]
22: if T>X[I];
 gto -1
-------------------------------------------------------------------
23: I-1→I
24: T[J]-X[I]→H;            Temporary variables used in computing
 T[J]-X[I+1]→T;             functional and derivative values
 HT→X
-------------------------------------------------------------------
25: S[I]+HG[I]→S
26: (1/6→Z)(S[I]
 +S[I+1]+S)→U
27: ((Y[I+1]-               B[J] is the approximation to the functional
 Y[I])/(X[I+1]-             value at T[J]
 X[I])→W)H+Y[I]+
 XU→B[J]
-------------------------------------------------------------------
28: W+(H+T)U+
 ZXG[I]→D[J];               D[J] is the approximation to the derivative
 gto -10                    at T[J]
-------------------------------------------------------------------
29: 0→I→A
30: if (I+1→I)>N
 -1;ret                     A is the integral
31: A+(.5(X[I+
 1]-X[I]→H)(Y[I]
 +Y[I+1])-(1/
 24)H↑3(S[I]+
 S[I+1]))→A;gto
 -1
-------------------------------------------------------------------
*16954
```

Numerical Differentiation Of Discrete, Unequally-Spaced Data Points

FILE 22 DRIVER

FILE 23 SUBROUTINE "DER"

Numerical Differentiation Of Discrete, Unequally-Spaced Data Points

This program computes the derivative at any point x over an interval of discrete data points (x_i, y_i), i = 1,n. The user supplies the data points with the restriction that the x_i must be discrete and $x_i < x_{i+1}$ for i = 1,n-1. The program itself requests all necessary information and prints out the results after calling the subroutine "DER" in file 23 on page 157.

SPECIAL CONSIDERATIONS AND DIAGNOSTIC MESSAGES:

1. The maximum number of data points and output arguments that can be entered into this subroutine will depend on the size of the calling program. The following limitations assume that the calling program in file 22 is used.

   If N is the number of data points and M is the number of derivatives then:

   4N + 2M + maximum (N,M) < 631 on the 6,844 byte machine

   4N + 2M + maximum (N,M) < 1655 on the 15,036 byte machine

   4N + 2M + maximum (N,M) < 2679 on the 23,228 byte machine

   4N + 2M + maximum (N,M) < 3703 on the 31,420 byte machine

FORMULAE:

$$s'(t_j) = \frac{y_{i+1} - y_i}{x_{i+1} - x_i} + \frac{1}{6}[(t_j - x_i) + (t_j - x_{i+1})]\ \ s''(x_i) +$$

$$s''(x_{i+1}) + \ s''(t_j)) + \frac{1}{6}\ (t_j - x_i)\ (t_j - x_{i+1})$$

$$\frac{s''(x_{i+1}) - \ s''(x_i)}{x_{i+1} - x_i}$$

REFERENCES:

1. Ralston and Wilf, Mathematical Methods for Digital Computers, Vol. II (New York: John Wiley and Sons, 1967) pp. 156 - 158.

2. Greville, T.N.E., Editor, "Proceedings of An Advanced Seminar Conducted by the Mathematics Research Center", U.S. Army, University of Wisconsin, Madison. October 7 - 9, 1968. Theory and Applications of Spline Functions (New York, London: Academic Press, 1969), pp. 156 - 167.

INSTRUCTIONS:

1. Insert UTILITY ROUTINES cartridge with machine turned on.

2. Load the file.
   a. Type ldf 22
   b. Press EXECUTE

3. (Optional)
   a. Type fxd N or flt N where N is the number of digits printed to the right of the decimal point ($0 \leq N \leq 11$).
   b. Press EXECUTE

4. Press RUN

5. When "NO. OF DATA PTS?" is displayed:
   a. Enter the number of data points in the data set.
   b. Press CONTINUE

6. When "X[I]" is displayed, "I" is the number of the data point to be entered.
   a. Enter the domain value of the I^{th} data point.
   b. Press CONTINUE

7. When "Y[I]" is displayed:
   a. Enter the functional value of the I^{th} data point.
   b. Press CONTINUE
   c. If you have not entered all the data, go to step 6.

8. When "CHANGES?" is displayed:
   Either
   a. Press CONTINUE if you do not need to make any changes to the entered data.
   b. Go to step 12.
   or
   a. Enter 1 if you need to make changes.
   b. Press CONTINUE

9. When "DATA PT NO.?" is displayed:
   a. Enter the number of the data point that is to be changed.
   b. Press CONTINUE

10. When "X[I]" is displayed:
    a. Enter the new domain value for the I^{th} data point.
    b. Press CONTINUE

11. When "Y[I]" is displayed:
    a. Enter the new functional value for the I^{th} data point.
    b. Press CONTINUE
    c. Go to step 8.

12. When "NO. OF DERIVS?" is displayed:
    a. Enter the number of arguments for which you want the derivative computed.
    b. Press CONTINUE

13. When "T[J]" is displayed, "J" is the number of the next output argument.
    a. Enter the next output argument that you want to be differentiated.
    b. Press CONTINUE
    c. If you have not entered all the desired output arguments, repeat step 13.

14. When "CHANGES?" is displayed:
    Either
    a. Press CONTINUE if you do not need to make any changes to the output arguments.
    b. Go to step 17.
    or
    a. Enter 1 if you need to make changes.
    b. Press CONTINUE

15. When "OUTPUT ARG NO?" is displayed:
    a. Enter the number of the output argument to be changed.
    b. Press CONTINUE

16. When "T[J]" is displayed, "J" is the number of the output argument to be changed.
    a. Enter the correct value of the output argument.
    b. Press CONTINUE
    c. Go to step 14.

17. When "EPSILON?" is displayed:

    Either

    Press CONTINUE , if you want the default value of 10^{-6} assigned to the error tolerance.

    or

    a. Enter the error tolerance to be used in the iterative solution of the simultaneous equations.

    b. Press CONTINUE

18. The derivatives of all output arguments will be printed.

EXAMPLE:

X	Y
0	1
.2	.72
.5	.75
1	2
1.1	2.43
1.6	5.48
1.8	7.12
2	9

```
NO. OF DATA PTS?
8

X[1]?
0
Y[1]?
1

X[2]?
.2
Y[2]?
.72

X[3]?
.5
Y[3]?
.75

X[4]?
1
Y[4]?
2

X[5]?
1.1
Y[5]?
2.43

X[6]?
1.6
Y[6]?
5.48

X[7]?
1.8
Y[7]?
7.12

X[8]?
2
Y[8]?
9

CHANGES?
```

```
NO. OF DERIVS.?
2

T[1]?
.81

T[2]?
1.2

CHANGES?

EPSILON?
.0000001

--DERIVATIVES--

X=        0.810000
S'[X]=    2.848435

X=        1.200000
S'[X]=    5.211391
```

DERIVATIVES

X	Y'
.81	2.848435
1.2	5.211391

"DER" SUBROUTINE

This subroutine computes the derivative of points on a curve s(x) which passes through n discrete, unequally spaced data points supplied by the user.

The method implemented fits a curve through the n data points and then differentiates the function which describes that curve. The curve used is the cubic natural spline which derives its name from a draftsman's mechanical spline. If the spline is considered as a function represented by s(x), then the second derivative s"(x) approximates the curvature. For the curve passing through data points (x_1, y_1), (x_2, y_2), ... , (x_n, y_n) we want $\int_{x_1}^{x_n} (s"(x))^2 \, dx$ to be minimized in order to achieve the "smoothest" curve.

The spline function with minimum curvature has cubic polynomials between adjacent data points. Adjacent polynomials are joined continuously with continuous first and second derivatives as well as $s"(x_1) = s"(x_n) = 0$.

The procedure to determine s(x) involves the iterative solution of a set of simultaneous linear equations by the method of successive over-relaxation. The accuracy to which these equations are solved is specified by the user. For a detailed discussion of the algorithm, see Reference 2.

When used as a subroutine, all parameters specified under <INPUT> must be assigned in the user's calling program. The user may then append the subroutine to his program and use gsb "DER" to take advantage of its capabilities. The results of the calculation and any other output is designated under <OUTPUT>. The user should be aware that the subroutine

may alter variables that he is using in his program and should therefore check the variables listed under <DESTROYED>.

The data points (x_i, y_i), i = 1,n should be entered in increasing order of the x_i, where the x_i are discrete and $x_i < x_{i+1}$ for i = 1,n-1. The arguments to be differentiated may be entered in any order but should be in the interval $[x_1, x_n]$.

The user should recognize that numerical differentiation is an inherently unstable process and should be circumspect when using the results. An example of a typical program which uses this subroutine may be found on page 161 or in file 22.

PARAMETERS AND VARIABLES USED:

<INPUT>

N	number of data points
X[N]	domain values of data points
Y[N]	range values of data points
M	number of output arguments desired
T[M]	domain values of derivative arguments
E	error tolerance in iterative solution of simultaneous equations

<OUTPUT>

N, X[N], Y[N], M, T[M], E	unchanged
D[M]	derivative values for output arguments in T[M]
flg 3	if set, indicates an output argument out of bounds

<DESTROYED>

H	temporary storage for $x_{i+1}-x_{i-1}$, abs(T), $x_{i+1}-x_i$, t_j-x_j

I loop counter

J loop counter

S $s''(t_j) + 3(t_j - x_i)T$

T $\dfrac{\dfrac{y_{i+1} - y_i}{x_{i+1} - x_i} - \dfrac{y_i - y_{i-1}}{x_i - x_{i-1}}}{H}$; $t_j - x_{i+1}$; t_j

U (μ) current maximum value of $|T|$ used to test error tolerance; $\frac{1}{6}\ (s''(x_i) + s''(x_{i+1})) + s''(x_i) + 3T\ (t_j - x_i)$

W $8-4\sqrt{3}$, relaxation factor for iterative solution; $\dfrac{y_{i+1} - y_i}{x_{i+1} - x_i}$

X temporary storage for $x_i - x_{i-1}$; $(t_j - x_i)\ (t_j - x_{i+1})$

Z 1/6

B[i] $\frac{1}{2}\ [\dfrac{x_i - x_{i-1}}{x_{i+1} - x_{i-1}}]$

G[i] 3T

S[i] 2T or $s''(t_j)$ vector of second derivatives

INSTRUCTIONS:

1. Load your program into memory. Your program should set up all necessary parameters (N, X[N], Y[N], E, M, T[M]) to be passed to the subroutine.

2. Insert UTILITY ROUTINES cartridge.

3. (Optional)
   a. Type `fxd N` or `flt N` where N is the number of digits printed to the right of the decimal point ($0 \leq N \leq 11$).
   b. Press EXECUTE

4. Load the file.
   a. Type `ldf 23, N, M` where:

      N = line number where the first line of the subroutine should be loaded into memory.

      M = line number where execution is to begin after subroutine is loaded.

   b. Press EXECUTE

ANNOTATED LISTING:

NUMERICAL DIFFERENTIATION OF UNEQUALLY SPACED POINTS

```
0: cfg 13;ldf
 23,22,1
```

Load the subroutine after the program

```
1: enp "NO. OF
 DATA PTS?",N;
 spc ;if flg13;
 cfg 13;gto +0
2: dim X[N],Y[N]
 ;1→I
```

Enter number of data points and dimension the X and Y arrays

```
3: enp X[I];if
 flg13;cfg 13;
 gto +0
4: enp Y[I];if
 flg13;cfg 13;
 gto +0
5: if (I+1→I)<=N
 ;spc ;gto -2
```

Enter the (X, Y) coordinates into the X and Y arrays

```
6: spc ;enp "CHA
 NGES?",I;spc 2;
 if flg13;cfg
 13;gto +4
7: enp "DATA
 PT. NO.?",I;
 spc ;if flg13;
 cfg 13;gto +0
8: enp X[I];if
 flg13;cfg 13;
 gto +0
9: cfg 13;enp
 Y[I];gto +0;if
 not flg13;gto -
 3
```

Make any necessary corrections to the X and Y arrays

```
10: enp "NO. OF
 DERIVS.?",M;
 spc ;if flg13;
 cfg 13;gto +0
11: 1→I;if M#0;
 dim T[M]
12: cfg 13;enp
 T[I];gto +0;if
 not flg13;spc ;
 jmp (I+1→I)>M
```

Enter the X values for which the derivative is to be computed

```
13: spc ;enp
 "CHANGES?",I;
 spc 2;if flg13;
 cfg 13;gto +3
14: enp "OUTPUT
 ARG NO?",I;if
 flg13;cfg 13;
 gto +0
15: cfg 13;enp
 T[I];gto +0;if
 not flg13;spc ;
 gto -2
```

Make any necessary corrections to the output arguments

ANNOTATED LISTING:

```
16: enp "EPSILON
 ?",E;if flg13;
 1e-6→E
17: gsb "DER"
18: if flg3;stp
19: spc 2;prt "-
 -DERIVATIVES--
 ";spc ;1→J
20: prt "X=",
 T[J],"S'[X]=",
 D[J];spc ;jmp
 (J+1→J)>M
21: spc 4;end
*1944
```

Enter error tolerance

Branch to subroutine

Stop if error flag is set

Print derivatives

ANNOTATED LISTING:

NUMERICAL DIFFERENTIATION OF UNEQUALLY SPACED POINTS

```
0: "DER":cfg 3;
 if M>N;dim B[M]
 ;gto +2
1: dim B[N]
2: dim S[N],G[N-
 1];1→I
```

Dimension arrays used to compute derivative

```
3: if (I+1→I)>N-
 1;gto +3
4: .5(X[I]-X[I-
 1]→X)/(X[I+1]-
 X[I-1]→H)→B[I]
```

$$\frac{X_i - X_{i-1} \rightarrow X}{X_{i+1} - X_{i-1} \rightarrow H} \qquad \frac{1}{2}\frac{(X_i - X_{i-1})}{(X_{i+1} - X_{i-1})} \rightarrow B[I]$$

```
5: 2(((Y[I+1]-
 Y[I])/(X[I+1]-
 X[I])-(Y[I]-
 Y[I-1])/X)/H→T)
 →S[I];3T→G[I];
 gto -2
```

$$\frac{\frac{Y_{i+1} - Y_i}{X_{i+1} - X_i} - \frac{Y_i - Y_{i-1}}{X_i - X_{i-1}}}{H} \rightarrow T \qquad \begin{array}{l} 2T \rightarrow S[I] \\ 3T \rightarrow G[I] \end{array}$$

```
6: 0→S[1]→S[N];
 8-4r3→W
```

W is relaxation factor

```
7: 0→U;2→I
8: W(-S[I]-B[I]S
 [I-1]-(.5-B[I])
 S[I+1]+G[I])→T
```

U is largest correction factor
Compute new correction factor T

```
9: if (abs(T)→H)
 >U;H→U
```

Larger correction factor stored in U

```
10: S[I]+T→S[I]
11: if I#N-1;I+
 1→I;gto -3
```

Add correction factor to S[I] = 2^{nd} derivative

```
12: if U>=E;gto
 -5
```

If correction factor is larger than error tolerance, recompute

```
13: 0→I
14: if (I+1→I)>N
 -1;gto +2
15: (S[I+1]-S[I]
 )/(X[I+1]-X[I])
 →G[I];gto -1
```

Compute $S'''(i) = \frac{S[I+1]-S[I]}{X[I+1]-X[I]}$

where S"'(i)=G[I] and S"(i)=S[I+1]

```
16: 0→J;dim D[M]
17: if (J+1→J)>M
 ;ret
18: 1→I;if (T[J]
 →T)>=X[1];gto +
 2
19: sfg 3;dsp
 "ARG OUT OF
 BOUND";ret
```

Store T[J] in simple variable T

Check if argument T[J] is out of bounds
$X[0] \leq T[J] \leq X[N]$

```
20: if (I+1→I)>N
 ;gto -1
21: if T>X[I];
 gto -1
22: I-1→I
```

Compute I such that
$X[I] \leq T[J] \leq X[I+1]$

```
23: T[J]-X[I]→H;
 T[J]-X[I+1]→T;
 HI→X
```

H, T, X are temporary variables for intermediate results

ANNOTATED LISTING:

```
24: S[I]+HG[I]→S
25: (1/6→Z)(S[I]
 +S[I+1]+S)→U
26: (Y[I+1]-Y[I]
 )/(X[I+1]-X[I])
 →W
27: W+(H+T)U+
 ZXG[I]→D[J];
 sto -10
*18523
```

Derivatives computed and stored in D[J]

Fourier Series Coefficients For Equally-Spaced Data Points

FILE 24 DRIVER

FILE 25 SUBROUTINE "FOURE"

Fourier Series Coefficients For Equally-Spaced Data Points

This program calculates the Fourier Series Coefficients a_i and b_i of the Fourier Series corresponding to a function f(x) which is specified by n discrete equally spaced data points (x_i, y_i), i = 1,n. The program requests only the functional values (y_i) since equally spaced data is assumed. It then calls the subroutine "FOURE" in file 25 on page 172 to perform the calculations and prints the results.

SPECIAL CONSIDERATIONS AND DIAGNOSTIC MESSAGES:

1. The maximum number of coefficients and function values that can be entered into the subroutine will depend on the size of the calling program. The following limitations assume that the calling program in file 24 is used.

   If N is the number of the highest coefficient and K is the number of function values then:

   2N + K < 683 on the 6,844 byte machine
   2N + K < 1707 on the 15,036 byte machine
   2N + K < 2731 on the 23,228 byte machine
   2N + K < 3755 on the 31,420 byte machine

2. For valid results, the user should provide at least N data points to compute N coefficients.

3. "ODD NO OF POINTS" indicates that the user must supply an odd number of data points.

4. The time to run the program is proportional to the number of coefficients and data points. With 17 points and 5 coefficients it took 2.5 seconds and with 33 points and 7 coefficients it took 4.5 seconds.

FORMULAE:

If: $g(x) = f(x) \cos\left(\frac{2\pi i x}{T}\right) dx$

and $h(x) = f(x) \sin\left(\frac{2\pi i x}{T}\right) dx$ then:

$$a_i \approx \frac{2\Delta x}{3T} \{g(x_1) + 4g(x_2) + 2g(x_3) + 4g(x_4) + \ldots + 4g(x_{n-1}) + g(x_n)\}$$

$$b_i \approx \frac{2\Delta x}{3T} \{h(x_1) + 4h(x_2) + 2h(x_3) + 4h(x_4) + \ldots + 4h(x_{n-1}) + h(x_n)\}$$

Sine and cosine functional values are computed recursively with the following formula:

$$\sin\left(\frac{2\pi x_i}{\Delta x}(J + 1)\right) = \sin\left(\frac{2\pi x_i}{\Delta x}\right) \cos\left(\frac{2\pi x_i}{\Delta x} J\right) + \cos\left(\frac{2\pi x_i}{\Delta x}\right) \sin\left(\frac{2\pi x_i}{\Delta x} J\right)$$

$$\cos\left(\frac{2\pi x_i}{\Delta x}(J + 1)\right) = \cos\left(\frac{2\pi x_i}{\Delta x}\right) \cos\left(\frac{2\pi x_i}{\Delta x} J\right) - \sin\left(\frac{2\pi x_i}{\Delta x}\right) \sin\left(\frac{2\pi x_i}{\Delta x} J\right)$$

REFERENCES:

1. Hamming, R.W., Numerical Methods for Scientists and Engineers (McGraw-Hill, 1962), pp. 67 - 80.

2. Acton, Forman S., Numerical Methods that Work (Harper and Row, 1970), pp. 221 - 257.

INSTRUCTIONS:

1. Insert UTILITY ROUTINES cartridge with machine turned on.

2. Load the file.
   a. Type ldf 24
   b. Press EXECUTE

3. (Optional)
   a. Type fxd N or flt N where N is the number of digits printed to the right of the decimal point ($0 \leq N \leq 11$).
   b. Press EXECUTE

4. Press RUN

5. When "HIGHEST COEF?" is displayed:
   a. Enter the highest numbered coefficient a_n, b_n that you want computed.
   b. Press CONTINUE

6. When "INIT DOMAIN VAL?" is displayed:
   a. Enter the left bound of the period over which the series is to be evaluated.
   b. Press CONTINUE

7. When "INCREMENT?" is displayed:
   a. Enter the increment between successive first coordinates $(x_{i+1} - x_i)$.
   b. Press CONTINUE

8. When "NO OF PTS?" is displayed:
   a. Enter the number of data points that you are going to evaluate (this number must be odd or an error message will be printed).
   b. Press CONTINUE

9. "RANGE VALUES?" will be printed.

10. When "Y[I]" is displayed, "I" designates the I^{th} data point to be entered.
    a. Enter the functional value (y_i) of the I^{th} data point (x_i, y_i).
    b. Press CONTINUE
    c. If all data values have not been entered, repeat step 10.

11. When "CHANGES?" is displayed:

    Either

    a. Press CONTINUE , if you do not need to make changes to the entered data.

    b. Go to step 14.

    or

    a. Enter 1 if you need to make some changes.

    b. Press CONTINUE

12. When "DATA PT NO?" is displayed:

    a. Enter the data point number which is to be changed.

    b. Press CONTINUE

13. When "Y[I]" is displayed:

    a. Enter the correct value for the functional value of the I^{th} data point.

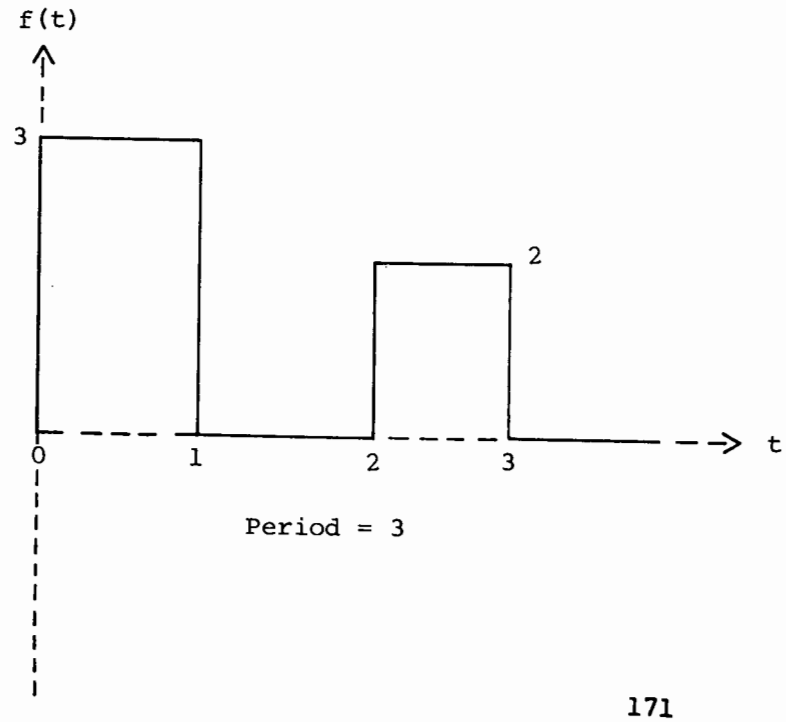
    b. Press CONTINUE

    c. Go to step 11.

14. The coefficients will be printed.

EXAMPLE:

```
HIGHEST COEF?
5
INIT DOMAIN VAL?
0
INCREMENT?
.1
NO. OF PTS.?
31

RANGE VALUES?

Y[1]?
1.5

Y[2]?
3

Y[3]?
3

Y[4]?
3

Y[5]?
3

Y[6]?
3

Y[7]?
3

Y[8]?
3

Y[9]?
3

Y[10]?
3

Y[11]?
1.5

Y[12]?
0

Y[13]?
0

Y[14]?
0

Y[15]?
0

Y[16]?
0

Y[17]?
0

Y[18]?
0

Y[19]?
0

Y[20]?
0

Y[21]?
1

Y[22]?
2

Y[23]?
2

Y[24]?
2

Y[25]?
2

Y[26]?
2

Y[27]?
2

Y[28]?
2

Y[29]?
2

Y[30]?
2

Y[31]?
1

CHANGES?

--COEFFICIENTS--

N=          0.000000
A(N)=       1.638889

N=          1.000000
A(N)=       1.322781
B(N)=       0.477470

N=          2.000000
A(N)=      -0.744837
B(N)=       0.238774

N=          3.000000
A(N)=      -0.055556
B(N)=      -0.000000

N=          4.000000
A(N)=       0.290053
B(N)=       0.119722

N=          5.000000
A(N)=      -0.333333
B(N)=       0.096225
```



"FOURE" SUBROUTINE

This subroutine calculates the Fourier Series Coefficients a_i and b_i for the Fourier Series corresponding to a function f(x) which is specified by n discrete equally spaced data points (x_i, y_i), i = 1,n.

The finite Fourier series for a point x is given by:

$$\frac{a_0}{2} + \sum_{i=1}^{n} \left(a_i \cos\left(\frac{i\pi x}{T}\right) + b_i \sin\left(\frac{i\pi x}{T}\right)\right)$$

where the Fourier coefficients a_i and b_i are:

$$\left.\begin{aligned} a_i &= \frac{2}{T} \int_{x_1}^{x_1+T} f(x) \cos\left(\frac{2\pi i x}{T}\right) dx \\ b_i &= \frac{2}{T} \int_{x_1}^{x_1+T} f(x) \sin\left(\frac{2\pi i x}{T}\right) dx \end{aligned}\right\} \quad i = 0,n$$

T specifies the period equivalent to $(x_n - x_1)$ and n indicates the number of coefficients desired. The coefficients are evaluated by numerical integration using Simpson's Rule. The number of data points must be odd and the user need only specify the initial domain value x_1, the increment $\Delta x = x_{i+1} - x_i$ and the range values for each x_i. Execution time depends on the number of coefficients that are computed.

When used as a subroutine, all parameters specified under <INPUT> must be assigned in the user's calling program. The user may then append the subroutine to his program and use gsb "FOURE" to take advantage of its capabilities. The results of the calculation and any other output is designated under <OUTPUT>. The user should be aware that the subroutine may alter variables he is using in his program and should therefore

check the variables listed under <DESTROYED>.

An example of a typical program which uses this subroutine may be found on page 176 or in file 24.

PARAMETERS AND VARIABLES USED:

<INPUT>

N	highest numbered coefficient
A	initial domain value x_1
W	increment between domain values $\Delta x = x_{i+1} - x_i$
K	number of data points
Y[K]	range values for data points

<OUTPUT>

N, A, W, K, Y[K] unchanged

A[0:N], B[N] } Fourier Series coefficients

flg 2 set if number of points is not odd

<DESTROYED>

D	(K - 1)W = period
E	$\frac{2\pi x_i}{\Delta x}$ argument for sine and cosine
F	$\cos(\frac{2\pi x_i}{\Delta x} J)$ } used in evaluation of integral
G	$\sin(\frac{2\pi x_i}{\Delta x} J)$ } used in evaluation of integral
I	loop counter for data points
J	loop counter for coefficients

M	multiplier = 1, 2, or 4, used in Simpson's Rule
S	$\cos(\frac{2\pi x_i}{\Delta x})$ invariant used in recursive computation of sine and cosine; partial sum for A[0]
T	temporary storage for new F
X	current domain value x_i for integral computation
Y	current functional value y_i for integral computation
Z	$\sin(\frac{2\pi x_i}{\Delta x})$ loop invariant used in recursive computation of sine and cosine

INSTRUCTIONS:

1. Load your program into memory. The program should set up all necessary parameters (N, A, W, K, Y[K]) to be passed to the subroutine.

2. Insert UTILITY ROUTINES cartridge.

3. (Optional)
   a. Type fxd N or flt N where N is the number of digits printed to the right of the decimal point ($0 \leq N \leq 11$).
   b. Press EXECUTE

4. Load the file.
   a. Type ldf 25, N, M where:
      N = line number where the first line of the subroutine should be loaded into memory.
      M = line number where execution is to begin after subroutine is loaded.
   b. Press EXECUTE

ANNOTATED LISTING:

FOURIER SERIES COEFFICIENTS FOR EQUALLY SPACED POINTS

```
0: cfg 13;ldf
 25,16,1
```
Load subroutine after program

```
1: enp "HIGHEST
 COEF?",N;if
 flg13;cfg 13;
 gto +0
```
Enter the highest numbered coefficient desired

```
2: enp "INIT
 DOMAIN VAL?",A;
 if flg13;cfg
 13;gto +0
```
Enter the initial domain value

```
3: enp "INCREMEN
 T?",W;if flg13;
 cfg 13;gto +0
```
Enter the increment between consecutive points

```
4: enp "NO. OF
 PTS.?",K;if
 flg13;cfg 13;
 gto +0
```
Enter the number of points

```
5: spc ;prt "RAN
 GE VALUES?";
 spc ;1→I;dim
 Y[K]
6: cfg 13;enp
 Y[I];gto +0;if
 not flg13;spc ;
 jmp (I+1→I)>K
```
Enter the functional values Y[I] for each equally spaced point

```
7: enp "CHANGES?
 ",I;spc ;if
 flg13;gto +3
8: enp "DATA PT
 NO?",I;spc ;if
 flg13;cfg 13;
 gto +0
9: cfg 13;enp
 Y[I];gto +0;if
 not flg13;gto -
 2
```
Make any necessary changes

```
10: gsb "FOURE"
```
Branch to subroutine

```
11: if flg2;stp
```
Stop if error flag is set

```
12: spc 3;prt "-
 -COEFFICIENTS--
 ";spc ;1→I
13: prt "N=",0,
 "A(N)=",A[0];
 spc
14: prt "N=",I,
 "A(N)=",A[I],
 "B(N)=",B[I];
 spc ;jmp (I+
 1→I)>N
15: spc 4;end
```
Print the Fourier coefficients

```
*27477
```

ANNOTATED LISTING:

FOURIER SERIES COEFFICIENTS FOR EQUALLY SPACED POINTS

```
0: "FOURE":rad;
 cfg 2;dim A[0:N
 ],B[N];1→J;0→I;
 (K-1)W→D
```
Dimension A and B arrays to hold the coefficients

```
1: if K/2=int(K/
 2);sfg 2;dsp
 "ODD NO OF PTS"
 ;ret
```
Check that an odd number of points have been entered

```
2: if (I+1→I)>K;
 gto +8
3: A+(I-1)W→X;
 Y[I]→Y;if I=1
```
X and Y are coordinates of the current data point

```
 or I=K;1→M;gto
 +3
```
M is multiplier - used in Simpson's Rule

```
4: if M=4;2→M;
 gto +2
5: 4→M
```
Multiplier alternates between 2 and 4

```
6: cos(2πX/D→E)→
 S→F;sin(E)→Z→G
```
Initial computation for sine and cosine used in recursion

```
7: MYF+A[J]→A[J]
 ;MYG+B[J]→B[J]
```
A[I] and B[J] are the J^{th} coefficients

```
8: if (J+1→J)<=N
 ;SF-ZG→T;ZF+
 SG→G;T→F;gto -1
```
Recursive computation of sine and cosine

```
9: 1→J;gto -7
10: Y[1]+4Y[2]+
 Y[K]→S;3→I
11: S+2Y[I]+4Y[I
 +1]→S;jmp (I+
 2→I)>=K
12: SW/3D→A[0];
 1→J
```
Computation for A[0] coefficient without use of sine and cosine

```
13: 2A[J]W/3D→A[
 J];2B[J]W/3D→B[
 J];jmp (J+1→J)>
 N
14: ret
*31639
```
Final modification with Simpson's Rule to compute final coefficients

Fourier Series Coefficients For Unequally-Spaced Data Points

FILE 26 DRIVER

FILE 27 SUBROUTINE "FOURIER"

Fourier Series Coefficients For Unequally-Spaced Data Points

This program calculates the Fourier Series Coefficients for a function defined by discrete data points (x_i, y_i), i = 1,n. The data pairs must be entered such that the x_i are discrete, but not necessarily equally spaced, and $x_i < x_{i+1}$ for i = 1,n-1.

The program requests all the necessary information, calls the subroutine "FOURIER" in file 27 on page 189 , and prints out the computed coefficients.

SPECIAL CONSIDERATIONS:

1. The maximum number of data points and coefficients that can be used by this subroutine will depend on the size of the calling program. The following limitations assume that the calling program in file 26 is used.

   If N is the highest numbered coeficient and K the number of data points then:

   2K + 2N < 584 on the 6,844 byte machine
   2K + 2N < 1608 on the 15,036 byte machine
   2K + 2N < 2632 on the 23,228 byte machine
   2K + 2N < 3656 on the 31.420 byte machine

2. For valid results the user should provide at least N data points to compute N coefficients.

3. For 17 data points and 5 coefficients the program took 7.8 seconds while for 33 data points and 7 coefficients it took 14.5 seconds.

FORMULAE:

$$A_j = \sum_{i=2}^{k-1} S_i \qquad \text{for } j = 1, n$$

$$B_j = \sum_{i=2}^{k-1} T_i \qquad \text{for } j = 1, n$$

Where:

$$Q_i = \frac{(x_{i+1}-x_{i+2})\, y_i - (x_i-x_{i+2})\, y_{i+1} + (x_i-x_{i+1})\, y_{i+2}}{(x_{i+1}-x_{i+2})\,(x_i-x_{i+2})\,(x_i-x_{i+1})}$$

$$A = \frac{1}{2}\,(Q_i+Q_{i-1})$$

$$R_i = \frac{-(x_{i+1}^2 - x_{i+2}^2)\, y_i - (x_i^2-x_{i+2}^2)\, y_{i+1} + (x_i^2-x_{i+1}^2)\, y_{i+2}}{(x_{i+1}-x_{i+2})\,(x_i-x_{i+2})\,(x_i-x_{i+1})}$$

$$B = \frac{1}{2}\,(R_i+R_{i-1})$$

$$S_i = \frac{(2Ax_{i+1}+B)\cos\left[2\pi\left(\frac{x_{i+1}J}{x_k-x_1}\right)\right] - (2Ax_i+B)\cos\left[2\pi\left(\frac{x_iJ}{x_k-x_1}\right)\right]}{\left(\frac{J}{x_k-x_1}\right)^2} +$$

$$\frac{y_{i+1}\left(\frac{J}{x_k-x_1}\right)^2 - 2A}{\left(\frac{2\pi J}{x_k-x_1}\right)^3} \sin\left[2\pi\left(\frac{x_{i+1}J}{x_k-x_1}\right)\right] -$$

$$\frac{y_i\left(\frac{J}{x_k-x_1}\right)^2 - 2A}{\left(\frac{2\pi J}{x_k-x_1}\right)^3} \sin\left[2\pi\left(\frac{x_iJ}{x_k-x_1}\right)\right]$$

$$T_i = \frac{(2Ax_{i+1}+B)\ \sin[2\pi frc(\frac{x_{i+1}J}{x_k-x_1})] - (2Ax_i+B)\ \sin[2\pi frc(\frac{x_iJ}{x_k-x_1})]}{(\frac{J}{x_k-x_1})^2} -$$

$$\frac{y_{i+1}(\frac{J}{x_k-x_1})^2 - 2A}{(\frac{2\pi J}{x_k-x_1})^3}\ \cos[2\pi(\frac{x_{i+1}J}{x_k-x_1})] +$$

$$\frac{y_i(\frac{J}{x_k-x_1})^2 - 2A}{(\frac{2\pi J}{x_k-x_1})^3}\ \cos\ [2\pi(\frac{x_iJ}{x_k-x_1})]$$

Where J is the number of coefficient being computed and frc ( ) indicates fractional part.

$$A_0 = \sum_{i=2}^{k=1} U_i$$

$$U_i = \frac{A(x_{i+1}^3-x_i^3)}{3} + \frac{B(x_{i+1}^2-x_i^2)}{2} + C(x_{i+1}-x_i)$$

Where A, B are defined above and:

$$C = \frac{1}{2}(P_i+P_{i-1})$$

$$P_i = \frac{(x_{i+1}-x_{i+2})x_{i+1}x_{i+2}y_i-(x_i-x_{i+2})x_ix_{i+2}y_{i+1} + (x_i-x_{i+1})x_ix_{i+1}y_{i+2}}{(x_{i+1}-x_{i+2})\ (x_i-x_{i+2})\ (x_i-x_{i+1})}$$

Sine and cosine functional values are computed recursively with the

following formulae:

$$\sin\left(\frac{2\pi x_i}{x_k - x_1}(J+1)\right) = \sin\left(\frac{2\pi x_i}{x_k - x_1}\right)\cos\left(\frac{2\pi x_i}{x_k - x_1}J\right) + \cos\left(\frac{2\pi x_i}{x_k - x_1}\right)\sin\left(\frac{2\pi x_i}{x_k - x_i}J\right)$$

$$\cos\left(\frac{2\pi x_i}{x_k - x_1}(J+1)\right) = \cos\left(\frac{2\pi x_i}{x_k - x_1}\right)\cos\left(\frac{2\pi x_i}{x_k - x_1}J\right) - \sin\left(\frac{2\pi x_i}{x_k - x_1}\right)\sin\left(\frac{2\pi x_i}{x_k - x_1}J\right)$$

Where i is the data point number and J is the coefficient number

REFERENCES:

1. Hewlett Packard 9820A Math Pac, pp. 43 - 50.

2. Hamming, R.W. Numerical Methods for Scientists and Engineers (M^{C}Graw-Hill, 1962), pp. 67 - 80.

3. Acton, Forman S., Numerical Methods that Work (Harper and Row, 1970), pp. 221 - 257.

INSTRUCTIONS:

1. Insert UTILITY ROUTINES cartridge with machine turned on.

2. Load the file.
   a. Type ldf 26
   b. Press EXECUTE

3. (Optional)
   a. Type fxd N or flt N where N is the number of digits printed to the right of the decimal point ($0 \leq N \leq 11$).
   b. Press EXECUTE

4. Press RUN

5. When "HIGHEST COEF?" is displayed:
   a. Enter the highest numbered coefficient a_n, b_n that you want computed.
   b. Press CONTINUE

6. When "NO. OF DATA PTS?" is displayed:
   a. Enter the number of data points (x_i, y_i) which you are going to evaluate.
   b. Press CONTINUE

7. When "X[I]" is displayed, "I" is the number of the next data point to be entered.
   a. Enter the first coordinate of the I^{th} data point.
   b. Press CONTINUE

8. When "Y[I]" is displayed:
   a. Enter the second coordinate of the I^{th} data point.
   b. Press CONTINUE
   c. If there are more data points to be entered, go to step 7.

9. When "CHANGES?" is displayed:
   Either
   a. Press CONTINUE , if you do not need to change any of the entered data.
   b. Go to step 13.
   or
   a. Enter 1 if you wish to make changes to the data.

b. Press CONTINUE

10. When "DATA PT NO.?" is displayed:

a. Enter the number of the data point to be changed.

b. Press CONTINUE

11. When "X[I]" is displayed, "I" is the data point number to be changed.

a. Enter the new first coordinate

b. Press CONTINUE

12. When "Y[I]" is displayed:

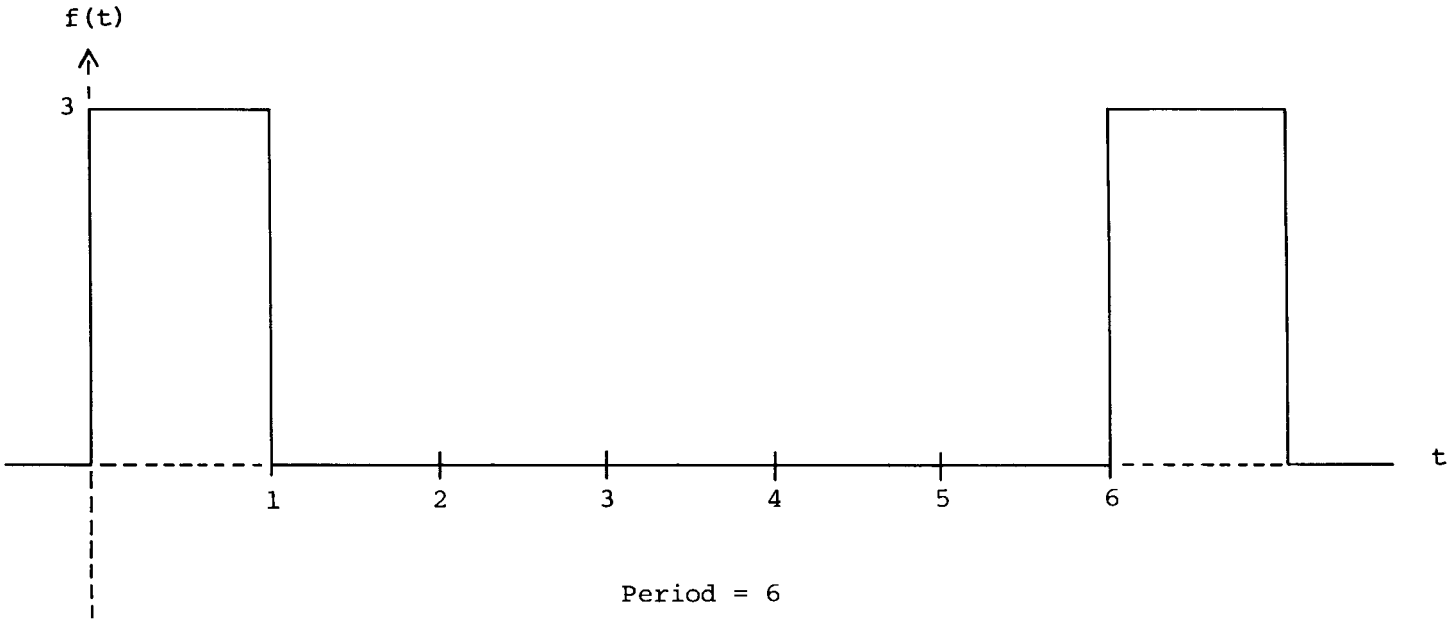
a. Enter the new second coordinate.

b. Press CONTINUE

c. Go to step 9.

13. The coefficients will be printed.

EXAMPLE:




```
HIGHEST COEF?
10
NO. OF DATA PTS?
37

X[1]?                X[6]?                X[11]?
0                    .3                   .8
Y[1]?                Y[6]?                Y[11]?
1.5                  3                    3

X[2]?                X[7]?                X[12]?
.1                   .4                   .9
Y[2]?                Y[7]?                Y[12]?
3                    3                    3

X[3]?                X[8]?                X[13]?
.15                  .5                   1
Y[3]?                Y[8]?                Y[13]?
3                    3                    1.5

X[4]?                X[9]?                X[14]?
.2                   .6                   1.1
Y[4]?                Y[9]?                Y[14]?
3                    3                    0

X[5]?                X[10]?               X[15]?
.25                  .7                   1.2
Y[5]?                Y[10]?               Y[15]?
3                    3                    0
```

EXAMPLE:

```
X[16]?
1.3
Y[16]?
0

X[17]?
1.5
Y[17]?
0

X[18]?
1.7
Y[18]?
0

X[19]?
1.9
Y[19]?
0

X[20]?
2
Y[20]?
0

X[21]?
2.5
Y[21]?
0

X[22]?
2.75
Y[22]?
0

X[23]?
3
Y[23]?
0

X[24]?
3.5
Y[24]?
0

X[25]?
3.7
Y[25]?
0

X[26]?
3.9
Y[26]?
0

X[27]?
4
Y[27]?
0

X[28]?
4.25
Y[28]?
0

X[29]?
4.5
Y[29]?
0

X[30]?
4.8
Y[30]?
0

X[31]?
5
Y[31]?
0

X[32]?
5.2
Y[32]?
0

X[33]?
5.4
Y[33]?
0

X[34]?
5.8
Y[34]?
0

X[35]?
5.85
Y[35]?
0

X[36]?
5.9
Y[36]?
0

X[37]?
6
Y[37]?
1.5

CHANGES?
```

```
--COEFFICIENTS--

N=        0.000000
A(N)=     0.500000

N=        1.000000
A(N)=     0.826236
B(N)=     0.476830

N=        2.000000
A(N)=     0.411971
B(N)=     0.713173

N=        3.000000
A(N)=     0.000000
B(N)=     0.630736

N=        4.000000
A(N)=    -0.203614
B(N)=     0.352015

N=        5.000000
A(N)=    -0.161406
B(N)=     0.092469

N=        6.000000
A(N)=     0.000000
B(N)=    -0.000723

N=        7.000000
A(N)=     0.112290
B(N)=     0.064172

N=        8.000000
A(N)=     0.096514
B(N)=     0.166647

N=        9.000000
A(N)=    -0.000000
B(N)=     0.193603

N=       10.000000
A(N)=    -0.073658
B(N)=     0.127569
```

"FOURIER" SUBROUTINE

The subroutine calculates the Fourier Series Coefficients a_i and b_i of the Fourier series corresponding to a function f(x) which is specified by n discrete data points (x_i, y_i), i = 1,n.

The finite Fourier series is given by the formula:

$$\frac{a_0}{2} + \sum_{i=1}^{n} \left(a_i \cos\left(\frac{i\pi x}{T}\right) + b_i \sin\left(\frac{i\pi x}{T}\right)\right)$$

where the Fourier coefficients a_i and b_i are:

$$\left.\begin{aligned} a_i &= \frac{2}{T} \int_{x_1}^{x_1+T} f(x)\cos\left(\frac{2\pi i x}{T}\right) dx \\ b_i &= \frac{2}{T} \int_{x_1}^{x_1+T} f(x)\sin\left(\frac{2\pi i x}{T}\right) dx \end{aligned}\right\} \quad i = 0, n$$

T specifies the period equivalent to $(x_n - x_1)$ and n indicates the number of coefficients desired. The coefficients are evaluated by numerically integrating a parabola passing through three successive points. Execution time depends on the number of coefficients calculated.

When used as a subroutine, all parameters specified under <INPUT> must be assigned in the user's calling program. The user may then append the subroutine to his program and use gsb "FOURIER" to take advantage of its capabilities. The results of the calculation and any other output is designated under <OUTPUT>. The user should be aware that the subroutine may alter variables that he is using in his program and should therefore check the variables listed under <DESTROYED>.

The data points (x_i, y_i), i = 1,n should be entered such that the x_i are discrete and $x_i < x_{i+1}$ for i = 1,n - 1. The points need not be equally spaced. An example of a typical program which uses this subroutine may be found on page 194 or in file 26.

PARAMETERS AND VARIABLES USED:

<INPUT>

K	number of data points
N	highest numbered coefficient desired
X[K], Y[K]	data points (x_i, y_i), i = 1, K

<OUTPUT>

K, N, X[K], Y[K] unchanged

A[0:N], B[N]	contain the Fourier Series Coefficients

<DESTROYED>

A	r1 if first set of data, r4 if last, otherwise the average of r1 and r4
B	r2 if first set of data, r5 if last, otherwise the average of r2 and r5
C	r3 if first set of data, r6 if last, otherwise the average of r3 and r5
D	x_i; $x_n - x_1$
E	x_{i+1}
F	x_{i+2}

G $\sin\left(\frac{2\pi x_i J}{x_k - x_1}\right)$

H $\cos\left(\frac{2\pi x_i J}{x_k - x_1}\right)$

I counter for data points

J counter for coefficients

L $\sin\left(\frac{2\pi x_{i+1} J}{x_k - x_1}\right)$

M $\cos\left(\frac{2\pi x_{i+1} J}{x_k - x_1}\right)$

P $2Ax_{i+1} + B$

Q $2Ax_i + B$

S partial sum of integral for a_j, the j^{th} coefficient

T $\frac{2\pi J}{x_k - x_1}$ argument for function evaluation; temporary storage for integral computation

U y_i

V y_{i+1}

W y_{i+2}

Z denominator $(x_{i+1}-x_{i+2})\ (x_i-x_{i+2})\ (x_i-x_{i+1})$; temporary storage

r1 $\{(x_{i+1}-x_{i+2})\ y_i-(x_i-x_{i+2})\ y_{i+1}+(x_i-x_{i+1})\ y_{i+2}\}/Z$

r2 $\{(x_{i+1}^2-x_{i+2}^2)\ y_i-(x_i^2-x_{i+2}^2)\ y_{i+1}+(x_i^2-x_{i+1}^2)\ y_{i+2}\}/Z$

r3 $\{(x_{i+1}-x_{i+2})\ x_{i+1}x_{i+2}y_i-(x_i-x_{i+2})\ x_ix_{i+2}y_{i+1}+(x_i-x_{i+1})^*$

$x_ix_{i+1}y_{i+2}\}/Z$

r4 former value of r1

r5 former value of r2

r6 former value of r3

r7 $\frac{J}{x_k - x_1}$ where J is current coefficient number being computed; $(\frac{J}{x_k - x_1})^2$

r8 $\frac{2\pi x_i}{x_k - x_1}$ first argument for functional evaluation

r9 $\frac{2\pi x_{i+1}}{x_k - x_1}$ second argument for functional evaluation

r10 $(\frac{2\pi J}{x_k - x_1})^3$

r11 $\sin(\frac{2\pi x_i J}{x_k - x_1})$

r12 $\cos(\frac{2\pi x_i J}{x_k - x_1})$

r13 $\sin(\frac{2\pi x_{i+1} J}{x_k - x_1})$

r14 $\cos(\frac{2\pi x_{i+1} J}{x_k - x_1})$

r15 partial sum of integral for b_j, the j^{th} coefficient

INSTRUCTIONS:

1. Load your program into memory. The program should set up all necessary parameters (K, N, X[K], Y[K]) to be passed to the subroutine.

2. Insert UTILITY ROUTINES cartridge.

3. (Optional)
   a. Type fxd N or flt N where N is the number of digits printed to the right of the decimal point ($0 \leq N \leq 11$).
   b. Press EXECUTE

4. Load the file.
   a. Type ldf 27, N, M where:
      N = line number where the first line of the subroutine should be loaded into memory.
      M = line number where execution is to begin after subroutine is loaded.
   b. Press EXECUTE

ANNOTATED LISTING:

DRIVER - FOURIER SERIES COEFFICIENTS FOR UNEQUALLY SPACED POINTS

```
0: ldf 27,17,1
```
Load the subroutine after the program

```
1: cfg 13;rad;
 spc 2
2: enp "HIGHEST
 COEF?",N;if
 flg13;cfg 13;
 gto +0
```
Enter the number of coefficients to be computed

```
3: enp "NO. OF
 DATA PTS?",K;
 if flg13;cfg
 13;gto +0
4: 1→J;dim X[K],
 Y[K];spc
```
Enter the number of data points

```
5: enp X[J];if
 flg13;cfg 13;
 gto +0
6: enp Y[J];if
 flg13;cfg 13;
 gto +0
7: if (J+1→J)<=K
 ;spc ;gto -2
```
Enter the data points (X_i, Y_i) into the X and Y arrays

```
8: spc ;enp "CHA
 NGES?",A;spc ;
 if flg13;gto +4
9: enp "DATA PT
 NO.?",J;if flg1
 3;cfg 13;gto +0
10: enp X[J];if
 flg13;cfg 13;
 gto +0
11: cfg 13;enp
 Y[J];gto +0;if
 not flg13;spc ;
 gto -3
```
Make any necessary changes

```
12: gsb "FOURIER
 "
```
Branch to subroutine

```
13: spc 2;prt "-
 -COEFFICIENTS--
 ";spc ;1→I
14: prt "N=",0,
 "A(N)=",A[0];
 spc
15: prt "N=",I,
 "A(N)=",A[I],
 "B(N)=",B[I];
 spc ;jmp (I+
 1→I)>N
16: spc 4;end
```
Print the Fourier coefficients

```
*12523
```

ANNOTATED LISTING:

SUBROUTINE - FOURIER SERIES COEFFICIENTS FOR UNEQUALLY SPACED POINTS


```
0: "FOURIER":dim
  A[0:N],B[N];
 1→I;1→J;0→A[0]
1: 0→A[J]→B[J];
 jmp (J+1→J)>N
```

Initialize coefficient arrays to zero

```
2: X[I]→A→D;X[I+
 1]→B→E;Y[I]→U;
 Y[I+1]→V
3: if I=K-1;sto
 +8
4: X[I+2]→C→F;
 Y[I+2]→W
```

Three consecutive data points (X_i,Y_i), (X_{i+1},Y_{i+1}), (X_{i+2},Y_{i+2}) are stored in simple variables for later use (except last two data points) X_i=D Y_i=U X_{i+1}=E Y_{i+1}=V X_{i+2}=F Y_{i+2}=W

```
5: (E-F)(D-F)(D-
 E)→Z
```

$(X_{i+1}-X_{i+2})(X_i-X_{i+2})(X_i-X_{i+1})\rightarrow Z$

```
6: ((E-F)U-(D-
 F)V+(D-E)W)/
 Z→r1
```

$\{(X_{i+1}-X_{i+2})Y_i-(X_i-X_{i+2})Y_{i+1}+(X_i-X_{i+1})Y_{i+2}\}$ $/Z \rightarrow r1 = A'$

```
7: -((EE-FF)U-
 (DD-FF)V+(DD-
 EE)W)/Z→r2
```

$\{(X_{i+1}^2-X_{i+2}^2)Y_i-(X_i^2-X_{i+2}^2)Y_{i+1}+(X_i^2-X_{i+1}^2)$ $Y_{i+2}\}/Z \rightarrow r2 = B'$

```
8: ((E-F)EFU-(D-
 F)DFV+(D-E)DEW)
 /Z→r3
```

$\{(X_{i+1}-X_{i+2})X_{i+1}X_{i+2}Y_i-(X_i-X_{i+2})X_iX_{i+2}Y_{i+1}+$ $(X_i-X_{i+1})X_iX_{i+1}Y_{i+2}\}/Z \rightarrow r3 = C'$

```
9: if I=1;r1→A;
 r2→B;r3→C;sto +
 3
```

If first set of points A'→A,B'→B,C'→C

```
10: .5(r1+r4)→A;
 .5(r2+r5)→B;
 .5(r3+r6)→C;
 sto +2
```

Average of computed values goes into A,B,C

$\frac{(A'+AP)}{2} \rightarrow A \quad \frac{(B'+BP)}{2} \rightarrow B \quad \frac{(C'+CP)}{2} \rightarrow C$

```
11: r4→A;r5→B;
 r6→C
```

If last set of points AP→A, BP→B, CP→C

```
12: 0→J
```

J is subscript of coefficient being computed

```
13: 2AX[I+1]+
 B→P;2AX[I]+B→Q
```

P and Q are used in computation of arguments for sine and cosine

---

```
14: r1→r4;r2→r5;
 r3→r6
```

Old value of A',B',C' saved in AP, BP, and CP

```
15: A(EEE-DDD)/
 3+B(EE-DD)/2+
 C(E-D)+A[0]→A[0
 ]
```

A[0] coefficient computed

```
16: sin(2πX[I]/
 (X[K]-X[1]))→r8)
 →G
17: cos(r8)→H
18: sin(2πX[I+
 1]/(X[K]-X[1]))→
 r9)→L
19: cos(r9)→M
20: 1→J;G→r11;
 H→r12;L→r13;
 M→r14;sto +5
```

Invariants for recursive computation of sine and cosine values are stored in simple variables

---

ANNOTATED LISTING:

```
21: Gr12+Hr11→r7
22: Hr12-Gr11→r1
 2;r7→r11
23: Lr14+Mr13→r7
24: Mr14-Lr13→r1
 4;r7→r13
```
Recursive computation of sine and cosine

```
25: J/(X[K]-X[1]
 )→r7
```
Coefficient number divided by period

```
26: ((2πr7→T)T→r
 7)T→r10
27: (Y[I+1]r7-
 2A)/r10→Z;(Y[I]
 r7-2A)/r10→T
```
Temporary variables for integration

```
28: (Pr14-Qr12)/
 r7+Zr13-Tr11→S
29: (Pr13-Qr11)/
 r7-Zr14+Tr12→r1
 5
```
S and r15 are partial sums of numerical integration approximation for A[J] and B[J]

```
30: S+A[J]→A[J]
31: r15+B[J]→B[J
 ]
```
Integral approximations updated for A[J] and B[J]

```
32: if J#N;J+
 1→J;gto -11
```
Repeat for the next coefficients

```
33: if (I+1→I)#K
 ;gto -31
```
Repeat computation for next set of 3 data points

```
34: 0→J;X[K]-
 X[1]→D
```
D is period

```
35: if (J+1→J)>N
 ;A[0]/D→A[0];
 ret
36: 2A[J]/D→A[J]
 ;2B[J]/D→B[J];
 gto -1
```
Finishing touches put on Fourier coefficients

```
*24671
```

Iterative Rootfinder For User-Defined Function

FILE 28 DRIVER

FILE 29 SUBROUTINE "ROOTS"

Iterative Rootfinder For User-Defined Function

This program will search for solutions of f(x) = 0 over an interval [a, b] where the user defines the continuous function f(x). The function may be algebraic of the form $(a_0 + a_1 x^{e1} + a_2 x^{e2} + \ldots + a_n x^{en})$ with a_i real and e_i rational (e.g. $X^3 + 3X^{3/2} + \sqrt{X}$) or transcendental (e.g. sin(X) + cos(X)).

The user specifies the initial domain value, search increment, error tolerance, and maximum interval-halvings to be used. The program then calls the subroutine "ROOTS" in file 29 on page 203 to calculate the roots.

SPECIAL CONSIDERATIONS AND DIAGNOSTIC MESSAGES:

1. "ILLEGAL BOUNDS" indicates that the lower interval bound was larger than the upper.
2. "DEFINE FUNCTION" indicates that you forgot to define the function.

REFERENCES:

1. Stark, Peter A., Introduction to Numerical Methods (London: MacMillan Company, Collier - MacMillan Limited, 1970), pp. 95 - 96.

INSTRUCTIONS:

1. Insert UTILITY ROUTINES cartridge with machine turned on.
2. Load the file.
   a. Type ldf 28
   b. Press EXECUTE
3. (Optional)
   a. Type fxd N or flt N where N is the number of digits printed to the right of the decimal point ($0 \leq N \leq 11$).
   b. Press EXECUTE
4. Press RUN
5. When "DEFINE FUNCTION" appears in the display:
   a. Type gto "EVAL"
   b. Press EXECUTE
6. Press STEP
7. Press STEP
8. Fetch the line where the function is stored.
   a. Observe the number in the display and press FETCH followed by that number.
   b. Press EXECUTE
9. 'beep; beep; dsp "DEFINE FUNCTION"; stp' will be displayed:
   If the function definition will fit on one line then:
   a. Type the function to be evaluated into the display in the form f(X) → Y (e.g. X↑2 + 3X + 5 → Y).
   b. Press STORE
   c. Go to step 10.
   Otherwise, if the function requires more than one line:
   a. Type the first line of the function definition.
   b. Press STORE
   c. Type the next line of the function definition
   d. Press INSERT (line INSERT not character INSERT)
   e. Repeat steps c and d until the complete function has been entered.

10. Continue execution.
    a. Type cont 2 (you must type the word "cont" character by character rather than pressing the CONTINUE key).
    b. Press EXECUTE

11. When "LOW BOUND?" is displayed:
    a. Enter the lower bound of the interval to be searched for roots.
    b. Press CONTINUE

12. When "UPPER BOUND?" is displayed:
    a. Enter the upper bound of the interval to be searched for roots.
    b. Press CONTINUE

13. When "SEARCH INC?" is displayed:
    a. Enter the size of each subinterval that will be searched for roots.
    b. Press CONTINUE

14. When "EPSILON?" is displayed:
    Either
    Press CONTINUE , if you want the default value of 10^{-6} for the f(x) error tolerance.
    or
    a. Enter the error tolerance for f(x)
    b. Press CONTINUE

15. When "MAX BISECTIONS?" is displayed:
    a. Enter the maximum number of interval bisections that can occur.
    b. Press CONTINUE

16. The roots x, their functional values f(x), and the accuracy to which the root is found will be printed for all roots found over the interval.

EXAMPLE:

Roots of sin(2X) = 0

on interval [0, 6]

are 0, π/2, π, 3π/2

```
LOW BOUND?
0
UPPER BOUND?
6
SEARCH INC?
.5
MAX BISECTIONS?
12

EPSILON?
.000001

---ROOTS ARE---

X=       0.000000
F(X)=    0.000000

ACCURATE TO
         0.000000


X BETWEEN
         1.570313
         1.570801
F(X)=    0.000479

ACCURATE TO
         0.000488


X BETWEEN
         3.141113
         3.141602
F(X)=   -0.000470

ACCURATE TO
         0.000488


X BETWEEN
         4.711914
         4.712402
F(X)=    0.000462

ACCURATE TO
         0.000488
```

"ROOTS" SUBROUTINE

This subroutine will search for solutions of f(x) = 0 over an interval [a, b] where the user defines the continuous function f(x). The function may be algebraic of the form $a_0 + a_1x^{e1} + a_2x^{e2} + \ldots + a_nx^{en}$ with a_i real and e_i rational (e.g. $X^3 + 3X^{3/2} + \sqrt{X}$) or transcendental (e.g. sin(X) + cos(X)).

The user specifies the search increment Δx and the error tolerance for f(x). The program then begins at the left of the interval and compares functional values at the ends of the subinterval [a, a+Δx]. If the functional values are of opposite sign then the method of interval halving (bisection) is used to locate the root. Each subinterval (A + iΔx, A+(i+1)Δx) is examined for a possible root. At most one root per interval will be located and if there are multiple roots per interval, none may be located. The user must also specify a maximum number of interval-halvings (M) so that an error tolerance which is not satisfied will result in the root localized to an interval of size $2^{-M}(B - A)$. The subroutine will examine $N = \text{int}(\frac{B - A}{\Delta x})$ intervals.

Because the number of roots located cannot be computed ahead of time, the program is only a quasi-subroutine whose output occurs in the subroutine itself and is not saved. The user may modify this method of output by changing lines 8 and 16 in the "ROOTS" program.

One of the advantages of this method is that whenever a root is found the accuracy of the root is known precisely. This value is printed out for each root.

When used as a subroutine, all parameters specified under <INPUT> must be assigned in the user's calling program. The user may then append

the subroutine to his program and use gsb "ROOTS" to take advantage of its capabilities. The results of the calculation and any other output is designated under <OUTPUT>. The user should be aware that the subroutine may alter variables that he is using in his program and should therefore check the variables listed under <DESTROYED>.

An example of a program which uses this subroutine can be found on page 208 or in file 28.

PARAMETERS AND VARIABLES USED:

<INPUT>

A	lower bound for search interval
B	upper bound for search interval
D	search increment (Δx)
E	error tolerance
M	maximum # of bisections for each subinterval
f(x)→Y	must be defined in "EVAL" subroutine

<OUTPUT>

B, D, E, M unchanged

Roots are printed in subroutine, but not saved.

S	interval size bordering roots when found
flg 2	set if bounds are reversed

<DESTROYED>

C	counter for number of interval halvings
L	left (lower) bound
R	right (upper) bound

X	domain argument for function evaluation in "EVAL"
Y	functional value f(x)
Z	product of f(A+iΔx) and f(A+(i+1)Δx). If negative or zero, a root is contained in the interval.

INSTRUCTIONS:

1. Load your program into memory. The program should set up all necessary parameters (K, N, X[K], Y[K]) to be passed to the subroutine.
2. Insert UTILITY ROUTINES cartridge.
3. Type ldf 29, N, M where:
   N = line number where first line of subroutine should be loaded into memory.
   M = line number where execution is to begin after subroutine is loaded.
4. Press EXECUTE
5. (Optional)
   a. Type fxd N or flt N where N is the number of digits printed to the right of the decimal point ($0 \leq N \leq 11$).
   b. Press EXECUTE
6. Find the line where the function is stored.
   a. Type gto "EVAL"
   b. Press EXECUTE
7. Press STEP
8. Press STEP
9. Fetch the line where the function is stored.
   a. Observe the number in the display and press FETCH followed by that number.
   b. Press EXECUTE
10. 'beep; beep; dsp "DEFINE FUNCTION"; stp' will be displayed:
    If the function definition will fit on one line then:
    a. Type the function to be evaluated into the display in the form f(X) → Y (e.g. X↑2 + 3X + 5 → Y).
    b. Press STORE
    c. Go to step 11.
    Otherwise, if the function requires more than one line:
    a. Type the first line of the function definition
    b. Press STORE
    c. Type the next line of the function definition

d. Press INSERT (line INSERT not character INSERT)

e. Repeat steps c and d until the complete fucntion has been entered.

11. Continue execution.

a. Type cont N , where N is the location where the program execution should begin. (Do not press the CONTINUE key; you must type in "cont" one character at a time).

b. Press EXECUTE

ANNOTATED LISTING:

DRIVER - ITERATIVE ROOTFINDER FOR USER-DEFINED FUNCTION

```
                                  Load the subroutine after the program
0: ldf 29,10,1
1: cfg 13;dsp                     User must define function in subroutine
 "DEFINE FUNCTIO
 N";stp
2: enp "LOW BOUN                  Lower bound of search interval
 D?",A;if flg13;
 cfg 13;gto +0
3: enp "UPPER                     Upper bound of search interval
 BOUND?",B;if
 flg13;cfg 13;
 gto +0
4: enp "SEARCH                    Size of search interval
 INC?",D;if flg1
 3;cfg 13;gto +0
5: enp "MAX BISE                  Maximum number of bisections during search
 CTIONS?",M;if
 flg13;cfg 13;
 gto +0
6: spc ;enp "EPS                  Error tolerance
 ILON?",E;spc ;
 if flg13;1e-6→E
7: prt "---ROOTS
  ARE---";spc 2
8: gsb "ROOTS"                    Branch to subroutine - Roots are printed
9: spc 4;end                      in subroutine
*27805
```

SUBROUTINE - ITERATIVE ROOTFINDER FOR USER-DEFINED FUNCTION

```
0: "ROOTS":cfg
 2;if A>B;sfg 2;
 dsp "ILLEGAL
 BOUNDS";ret
```
Check for illegal order of search bounds

```
1: A→X;gsb "EVAL
 "
```
Functional value fo left bound

```
2: Y→F
```
Left bound saved in F

```
3: if (A+D→A)>B;
 ret
4: A→X;gsb "EVAL
 "
```
Right bound saved in L

```
5: if (FY→Z)>0;
 gto -3
6: if Z<0;gto +3
```
If product is positive, search next interval
If product is negative, look for root

```
7: if F=0;A-D→X;
 F→Y
8: prt "X=",X;
 "F(X)=",Y;spc ;
 A+D→A;1e-12→S;
 gto +9
```
ROOT has been found

```
9: A-D→L;A→R;0→C
10: (L+R)/2→X;
 gsb "EVAL"
```
Compute midpoint and its functional value

```
11: if (C+1→C)>M
 ;gto +5
```
Check for maximum number of iterations

```
12: if abs(Y)<E;
 0→Z;gto +2
```
Check if error tolerance is satisfied

```
13: if (FY→Z)>0;
 X→L;gto -3
```
If product is positive look on right interval

```
14: if Z=0;prt
 "X=",X,"F(X)=",
 Y;spc ;R-L→S;
 gto +3
```
Check if root has been found

```
15: X→R;gto -5
```
Search on left interval

```
16: prt "X BETWE
 EN",L,R,"F(X)="
 ,Y;spc ;R-L→S
17: spc ;prt
 "ACCURATE TO",
 S;spc 3;gto -16
```
Print roots and accuracy

```
18: "EVAL":
19: beep;beep;
 dsp "DEFINE
 FUNCTION";stp
20: ret
*25662
```
Function will be defined in line 19

Polynomial Rootfinder

FILE 30 DRIVER

FILE 31 SUBROUTINE "ROOT"

Polynomial Rootfinder

This program will find all roots, Z, of polynomials of the form

$$a_0 + ib_0 + (a_1 + ib_1)\ Z + (a_2 + ib_2)\ Z^2 + \ldots (a_n + ib_n)\ Z^n = 0.$$

After the coefficients are entered, the program calls the subroutine "ROOT" in file 31, listed on page 218 , and then prints the roots.

SPECIAL CONSIDERATIONS AND DIAGNOSTIC MESSAGES:

1. The maximum degree polynomial this subroutine can handle will depend on the calling program. The following limitations assume that the calling program in file 30 is used. If N is the degree of the polynomial then:

   N < 103 on the 6,844 byte machine
   N < 1127 on the 15,036 byte machine
   N < 2151 on the 23,228 byte machine
   N < 3175 on the 31,420 byte machine

2. "NO CONVERGENCE" indicates that a root cannot be found after 50 iterations or 10 successive quarterings of the search increment. These limits may be modified by changing line 13 or 15 respectively in the "ROOT" subroutine.

FORMULAE

$$f(z) = \sum_{k=0}^{n} (a_k + ib_k)\ z^k = 0$$

Siljak functions X_k and Y_k are defined by

$z^k = X_k + iY_k$ and may be calculated recursively

$$X_0 = 1,\ X_1 = .1,\ Y_0 = 0,\ Y_1 = 1$$

$$\left.\begin{aligned} X_{k+2} &= 2x\, X_{k+1} - (x^2 + y^2)\, X_k \\ Y_{k+2} &= 2x\, Y_{k+1} - (x^2 + y^2)\, Y_k \end{aligned}\right\}$$ where x + iy are the root approximations

$$u = \sum_{k=0}^{n} (a_k X_k - b_k Y_k)$$

$$v = \sum_{k=0}^{n} (a_k Y_k + b_k X_k)$$

$$\frac{\partial u}{\partial x} = \sum_{k=1}^{n} k\, (a_k X_{k-1} - b_k Y_{k-1})$$

$$\frac{\partial v}{\partial x} = \sum_{k=1}^{n} k\, (a_k Y_{k-1} + b_k X_{k-1})$$

REFERENCES:

1. Moore, J.B., "A Convergent Algorithm for Solving Polynomial Equations", Journal of the Association for Computing Machinery. vol. 14, No. 2 (April, 1967), pp. 311 - 315.

INSTRUCTIONS:

1. Insert UTILITY ROUTINES cartridge with machine turned on.

2. Load the file.
   a. Type ldf 30
   b. Press EXECUTE

3. (Optional)
   a. Type fxd N or flt N where N is the number of digits printed to the right of the decimal point ($0 \leq N \leq 11$).
   b. Press EXECUTE

4. Press RUN

5. When "DEG OF POLY?" is displayed:
   a. Enter the degree of the polynomial that you are going to evaluate.
   b. Press CONTINUE

6. When "A[I]" is displayed, "I" designates the exponent of the term which is to be entered next:
   a. Enter the real part of the coefficient of the term with exponent I.
   b. Press CONTINUE

7. When "B[I]" is displayed:
   a. Enter the imaginary part of the coefficient for the term with exponent I.
   b. Press CONTINUE
   c. If more coefficients need to be entered, go to step 6.

8. When "CHANGES?" is displayed:
   Either
   a. Press CONTINUE if no changes need to be made.
   b. Go to step 12.
   or
   a. Enter 1 if you want to make changes.
   b. Press CONTINUE

9. When "COEF EXP?" is displayed:
   a. Enter the exponent of the coefficient to be changed.

b. Press CONTINUE

10. When "A[I]" is displayed:

a. Enter the real part of the correct coefficient.

b. Press CONTINUE

11. When "B[I]" is displayed:

a. Enter the imaginary part of the correct coefficient.

b. Press CONTINUE

c. Go to step 8.

12. The roots will be printed.

EXAMPLES:

$(1+0i)z^2+(-2+i)z+3-i = 0$

$z_1 = 1-2i$

$z_2 = 1+i$

```
DEG OF POLY?
2

A[2]?
1
B[2]?
0

A[1]?
-2
B[1]?
1

A[0]?
3
B[0]?
-1

CHANGES?



----------------


ROOT NO.
          2.000000
A=        1.000000
B=        1.000000

ROOT NO.
          1.000000
A=        1.000000
B=       -2.000000
```

```
DEG OF POLY?
2

A[2]?
2
B[2]?
0

A[1]?
-7
B[1]?
0

A[0]?
-15
B[0]?
0

CHANGES?



----------------


ROOT NO.
          2.000000
A=       -1.500000
B=        0.000000

ROOT NO.
          1.000000
A=        5.000000
B=        0.000000
```

$2z^2-7z-15 = 0$

$z_1 = 5$

$z_2 = -1.5$

"ROOT" SUBROUTINE

This subroutine will find all the roots, Z, of polynomials of the form $a_0 + ib_0 + (a_1 + b_1)\ Z + (a_2 + ib_2)\ Z^2 + \ldots + (a_n + ib_n)\ Z^n = 0$.

Roots are found by expressing the polynomials in terms of Siljak functions and using the method of steepest descent to determine the zeros. Once a root is found, the polynomial is reduced by synthetic division and the process is repeated. The last root is computed algebraically. The algorithm is very accuraté and stable; it will virtually always find the roots and you do not need to provide an initial value. Multiple roots are found at some slightly reduced accuracy, and higher order polynomials may show some loss of accuracy as more roots are found. In general, the program will find "normally" spaced roots accurate to better than 6 decimal places. Newton's method could find the roots faster but convergence is not guaranteed and no a priori information is necessary.

When used as a subroutine, all parameters specified under <INPUT> must be assigned in the user's calling program. The user may then append the subroutine to his program and use gsb "ROOT" to take advantage of its capabilities. The results of the calculation and any other output is designated under <OUTPUT>. The user should be aware that the subroutine may alter variables that he is using in his program and should therefore check the variables listed under <DESTROYED>.

An example of a program which uses this subroutine can be found in file 30 listed on page 222.

PARAMETERS AND VARIABLES USED:

<INPUT>

N	degree of polynomial
A[ ] B[ ]	$A_i + iB_i$ represents the coefficients of the polynomial where the subscript corresponds to the exponent of the variable.

<OUTPUT>

C[J] D[J]	$C_j + iD_j$ represent the j^{th} root of the polynomial
flg 2	set if the algorithm does not converge to a root

<DESTROYED>

A	temporary storage for A[K], K = 0,N-1
B	temporary storage for B[K], K = 0,N-1
C	temporary storage for B[K], K = 1,N
D	ΔX; temporary storage for A[K], K = 1,N
E	ΔY
F	$u^2 + v^2$, function value for Siljak functions
G	former value of F used to test for convergence
I	loop counter
K	loop counter
L	counter to determine if after 50 iterations convergence has not occurred
M	counter to determine if 10 successive quarterings of the search increment have not reduced the function value
P	$\frac{\partial u}{\partial x}$

Q	$\frac{\partial v}{\partial x}$
T	temporary storage
U	real part of F; temporary storage for A[K + 1], K = 0,N-1
V	imaginary part of F; temporary storage for B[K + 1], K = 0,N-1
X, Y	X + iY is the root approximation
Z	$X^2 + Y^2$ used in computation of Siljak Functions: $(\frac{\partial u}{\partial x})^2 + (\frac{\partial v}{\partial x})^2$
r0	temporary new value of X
r1	temporary new value of Y
X[ ], Y[ ]	Siljak function where $Z^k = X_k + iY_k$

INSTRUCTIONS:

1. Load your program into memory. The program should set up all necessary parameters (K, N, X[K], Y[K]) to be passed to the subroutine.

2. Insert UTILITY ROUTINES cartridge.

3. (Optional)
   a. Type fxd N or flt N where N is the number of digits printed to the right of the decimal point ($0 \leq N \leq 11$).
   b. Press EXECUTE

4. Load the file.
   a. Type ldf 31, N, M where:
      N = line number where the first line of the subroutine should be loaded into memory.
      M = line number where execution is to begin after the subroutine is loaded.
   b. Press EXECUTE

ANNOTATED LISTING:

POLYNOMIAL ROOTFINDER

```
0: cfg 13;ldf
 31,15,1
```
Load subroutine after the program

```
1: enp "DEG OF
 POLY?",N;N→0;
 if flg13;cfg
 13;gto +0
2: N→I;dim A[0:N
 ],B[0:N];spc
3: enp A[I];if
 flg13;cfg 13;
 gto +0
4: enp B[I];if
 flg13;cfg 13;
 gto +0
5: if (I-1→I)>=0
 ;spc ;gto -2
```
Enter the coefficients of the polynomial into the A and B arrays

```
6: spc ;enp "CHA
 NGES?",I;spc ;
 if flg13;gto +4
7: enp "EXP NO.?
 ",I;if flg13;
 cfg 13;gto +0
8: enp A[I];if
 flg13;cfg 13;
 gto +0
9: cfg 13;enp
 B[I];gto +0;if
 not flg13;spc ;
 gto -3
```
Make necessary changes

```
10: gsb "ROOT"
```
Branch to subroutine

```
11: if flg2;stp
```
Stop if error flag set

```
12: spc 2;prt "-
 ---------------
 ";spc 2;0→J
13: spc ;prt
 "ROOT NO.",J,
 "A=",C[J],"B=",
 D[J];jmp (J-
 1→J)<1
14: spc 4;end
```
Print roots

```
*17273
```

ANNOTATED LISTING:

SUBROUTINE - POLYNOMIAL ROOTFINDER

```
0: "ROOT":cfg 2;
 dim X[0:N],Y[0:
 N],C[N],D[N];
 1→X[0]→Y[1]→Y;
 .1→X[1]→X;0→L→Y
 [0]
```

X[] and Y[] hold Siljak coefficients

A[] and B[] will hold root approximations

```
1: 1→X[0]→Y[1]→Y
 ;.1→X[1]→X;0→L→
 Y[0]
```

Initialize

```
2: gsb "SILJAK"
```

Branch to computation of Siljak coefficients

```
3: F→G;0→K→P→Q→M
 ;L+1→L
```

G gets former value of F, initialize and increment iteration counter

```
4: if (K+1→K)>N;
 PP+QQ→Z;gto +3
```

$\frac{\partial u^2}{\partial x} + \frac{\partial v^2}{\partial x} \rightarrow Z$

```
5: K(A[K]X[K-1]-
 B[K]Y[K-1])+P→P
6: K(A[K]Y[K-1]+
 B[K]X[K-1])+
 Q→Q;gto -2
```

$P=\frac{\partial u}{\partial x} = \sum_{K=1}^{N} K(A[K]X[K-1]-B[K]Y[K-1])$

$Q=\frac{\partial u}{\partial x} = \sum_{K=1}^{N} K(A[K]Y[K-1]+B[K]X[K-1])$

```
7: -(UP+VQ)/Z→D;
 (UQ-VP)/Z→E
```

D=ΔX change in X, E=ΔY change in Y

```
8: M+1→M
```

Increment successive quartering counter

```
9: X+D→X[1];Y+
 E→Y[1]
```

New root approximation X,Y loaded into X[1] and Y[1]

```
10: gsb "SILJAK"
```

Recompute Siljak coefficients

```
11: if F>=G;gto
 +4
```

If new error estimate greater than old - quarter the size of ΔX and ΔY

```
12: if abs(D)<1e
 -8 and abs(E)<1
 e-8;gto +13
```

Are increments small enough to satisfy the stopping conditions?

```
13: if L>50;gto
 +4
```

If more than 50 iterations, stop.

```
14: X[1]→X;Y[1]→
 Y;gto -11
```

Iterate again

```
15: if M<=10;D/
 4→D;E/4→E;gto -
 7
```

Quarter the interval size and try again

```
16: if abs(U)<=1
 e-7 and abs(V)<
 =1e-7;gto +9
```

Check if stopping conditions satisfied

```
17: sfg 2;dsp
 "NO CONVERGENCE
 ";ret
```

Error message

```
18: "SILJAK":X[1
 ]X[1]+Y[1]Y[1]→
 Z;-1→K;2X[1]→T
19: if (K+1→K)>N
 -2;0→U→V;-1→K;
 gto +3
20: TX[K+1]-ZX[K
 ]→X[K+2]
21: TY[K+1]-ZY[K
 ]→Y[K+2];gto -2
```

SUBROUTINE TO COMPUTE SILJAK COEFFICIENTS

X[0]=1, X[1]=X, Y[0]=0, Y[1]=Y

recursively

$X[K+2] = 2X*X[K+1]-(X^2+Y^2)X[K]$

$Y[K+2] = 2X*Y[K+1]-(X^2+Y^2)Y[K]$

ANNOTATED LISTING:

```
22: if (K+1→K)>N
 ;UU+VV→F;ret
23: A[K]X[K]-
 B[K]Y[K]+U→U
24: A[K]Y[K]+
 B[K]X[K]+V→V;
 gto -2
```

$$U = \sum_{K=0}^{N} (A[K]X[K]-B[K]Y[K])$$

$$V = \sum_{K=0}^{N} (A[K]Y[K-1]+B[K]X[K-1])$$

```
25: X[1]→C[N];
 Y[1]→D[N]
```

Store computed root in C and D array element

```
26: A[N]→A;B[N]→
 B;0→A[N]→B[N];
 N→K;X[1]→X;Y[1]
 →Y
```

Initialize variables for synthetic division

```
27: if (K-1→K)<0
 ;gto +4
28: A[K]→C;B[K]→
 D
29: A+X(A[K+1]→U
 )-Y(B[K+1]→V)→A
 [K]
30: B+XV+YU→B[K]
 ;C→A;D→B;gto -3
```

Synthetic division to calculate new coefficients A[] and B[]

```
31: if (N-1→N)#1
 ;gto -30
```

Reduce number of coefficients and begin again

```
32: -((A[0]→A)(A
 [1]→U)+(B[0]→B)
 (B[1]→V))/(UU+
 VV→T)→C[1]
33: (AV-UB)/T→D[
 1];ret
```

Since degree of resultant polynomial is one, compute final root algebraically

```
*18857
```

String Sort For Strings Or Substrings Within A String Array

FILE 32 DRIVER

FILE 33 "QSORT" SUBROUTINE

FILE 34 "SSORT" SUBROUTINE

String Array Sort On A Substring Key

This program sorts a string array A$[] into ascending (or descending) order such that A$[K]<(or>)A$[K+1] for K = b, b+1, ..., t-1, t where the user can specify the sorting bounds b and t. The sort can use the whole string or any substring as a key to allow maximum flexibility.

The program requests all necessary information, calls the subroutine "QSORT" on file 33 or the subroutine "SSORT" on file 34 and prints the sorted array.

The "QSORT" subroutine should be used if a fast sorting algorithm is desired for a large number of strings (>15). The speed of the sort is achieved by sacrificing additional memory for the larger program and more intermediate storage.

The "SSORT" subroutine should be used if you have a small number of strings to be sorted (≤15) or if memory usage is to be kept to a minimum. This routine will be slightly slower for a large number of strings but the memory used will be kept to a minimum. Specific speed and sizes are discussed in the Special Considerations section of each routine.

To run either of these string sorting routines you must have the Strings ROM in the machine.

"QSORT" SUBROUTINE

SPECIAL CONSIDERATIONS:

1. The maximum number and size of strings that can be handled depends on the size of the calling program. If the calling program in file 32 on page 242 is used, the following limitations apply:

   If N is the number of strings in the array and L is the length of the strings then:

   If N is the number of strings in the array and L is the length of the strings then:

   $2\log_2 N + L + N + \frac{1}{2} L * N < 621$ on the 6,844 byte machine

   $2\log_2 N + L + N + \frac{1}{2} L * N < 1645$ on the 15,036 byte machine

   $2\log_2 N + L + N + \frac{1}{2} L * N < 2669$ on the 23,228 byte machine

   The 31,420 option is not compatible with the Strings-AP ROM.

2. The speed of the sort is proportional to $N\log_2 N$ where N is the number of strings in the array. For a small number of array elements the speed difference between the QSORT subroutine and the routine in file 34 on page 239 is negligible. With 100 strings of length 20 this program was approximately two seconds faster (5 seconds vs. 7 seconds) and with 50 strings about one second faster (3 seconds vs. 4.3).

   Thus for smaller arrays you can save considerable space by using the subroutine in file 34 on page 239.

REFERENCES:

1. Hibbard, Thomas N. "An Empirical Study of Minimal Storage Sorting", Communications of the ACM 6 (May 1963), pp. 206 - 213.

2. Shell, D.L. "A High Speed Sorting Procedure", Communications of the ACM 2 (July 1959), pp. 30 - 32.

3. Singleton, Richard C. "Algorithm 347 - An Efficient Algorithm for Sorting With Minimal Storage", Communications of the ACM (March 1969), pp. 185 - 186.

4. Scowen, R.S. "Algorithm 271 - Quickersort", Communications of the ACM (Octcber 1965), pp. 669 - 670.

SPECIAL CONSIDERATIONS:

1. The maximum number and size of strings that can be handled depends on the size of the calling program. If the calling program in file 32 on page 242 is used, the following limitations apply:

   If N is the number of strings in the array and L is the length of the strings then:

   If N is the number of strings in the array and L is the length of the strings then:

   $\frac{1}{2} L(1 + N) + N < 676$ on the 6,844 byte machine

   $\frac{1}{2} L(1 + N) + N < 1700$ on the 15,036 byte machine

   $\frac{1}{2} L(1 + N) + N < 2724$ on the 23,228 byte machine

   The 31,420 byte option is not compatible with the Strings-AP ROM.

2. The speed of the sort is approximately proportional to $N^{1.5}$ where N is the number of strings in the array. There is no noticeable difference in speeds between this sort and the sort in file 33 on page 237 when N is small. However as N becomes larger the difference in timing becomes apparent. With 100 strings of length 20 this sort was approximately two seconds slower (7 seconds vs. 5 seconds) and with 50 strings about one second slower (4.3 seconds vs. 3 seconds).

REFERENCES:

1. Knuth, Donald E., Searching and Sorting (Addison Wesley, 1973), pp. 84 - 87.

2. Shell, D.L., "A High Speed Sorting Procedure", Communciations of the AMC 2 (July 1959), pp. 30 - 32.

INSTRUCTIONS:

1. Insert UTILITY ROUTINES cartridge with machine on. (Check that the Strings ROM has been inserted).

2. Load the file.
   a. Type ldf 32
   b. Press EXECUTE

3. Press RUN

4. When "ENTER 1 FOR QSORT" and "ENTER 2 FOR SSORT" is printed:
   a. Either
      Enter 1 to sort by the faster "QSORT" subroutine.
      or
      Enter 2 to sort by the space saving "SSORT" subroutine.
   b. Press CONTINUE

5. When "NO. OF STRINGS?" is displayed:
   a. Enter the number of strings in the array.
   b. Press CONTINUE

6. When "LENGTH OF STRING" is displayed:
   a. Enter the length of the strings in the array. (All strings must be dimensioned to the same length although different actual lengths are allowed. Therefore, you should dimension the string array to the longest string expected.)
   b. Press CONTINUE

7. When A$[I] is displayed, "I" will be the number of the I^{th} string in the array.
   a. Type the I^{th} string of the array.
   b. Press CONTINUE
   c. Repeat step 7 until all strings in the array have been entered.

8. When "CHANGES?" is displayed:
   Either
   a. Press CONTINUE if you do not need to make any changes to the entered data.
   b. Go to step 11.

or

a. Enter 1 if you need to make changes.

b. Press CONTINUE

9. When "STRING NUMBER?" is displayed:

a. Enter the number of the array element to be changed.

b. Press CONTINUE

10. When "A$[I]" is displayed, "I" will be the number of the string to be changed.

a. Type the correct I^{th} string

b. Press CONTINUE

c. Go to step 8.

11. "ENTER SUBSET OF
ARRAY TO BE will be printed.
SORTED "
When "LOW BOUND" is displayed:

a. Enter the number of the array element which is the lower bound for the sort.

b. Press CONTINUE

12. When "UPPER BOUND" is displayed:

a. Enter the number of the array element which is the upper bound for the sort.

b. Press CONTINUE

13. "ENTER SUBSTRING
USED AS KEY " will be printed.
When "LOW BOUND?" is displayed:

a. Enter the number of the character in the string which is the lower bound for the sort key.

b. Press CONTINUE

14. When "UPPER BOUND?" is displayed:

a. Enter the number of the character in the string which is the upper bound for the sort key.

b. Press CONTINUE

15. When "DECREASING ORDER?" is displayed:

    Either

    Press CONTINUE if you want to sort the elements into increasing order.

    or

    a. Enter 1 if you want to sort the elements into decreasing order.

    b. Press CONTINUE

16. The sorted string array will be printed.

EXAMPLE:

```
ENTER 1
FOR QSORT

ENTER 2
FOR SSORT
1

NO OF STRINGS
10

LENGTH OF STRING
16


A$[1]?
SNURD    MORTIMER

A$[2]?
DON BOB BILLY JO

A$[3]?
BERASOL ARCHIBAL

A$[4]?
TILLA    RHODA

A$[5]?
BREATH   BUFFALO

A$[6]?
FRED     SOBER

A$[7]?
DANIELS JACK

A$[8]?
DOME     CHROME

A$[9]?
PAUL     TALL

A$[10]?
COYOTE   WILEY

CHANGES?
```

```
ENTER SUBSET OF
ARRAY TO BE
SORTED


LOW BOUND?
1
UPPER BOUND?
10

ENTER SUBSTRING
USED AS KEY

LOW BOUND?
1
UPPER BOUND?
8


DECREASING ORDER
?
1


--SORTED ARRAY--

TILLA    RHODA
SNURD    MORTIMER
PAUL     TALL
FRED     SOBER
DON BOB BILLY JO
DOME     CHROME
DANIELS JACK
COYOTE   WILEY
BREATH   BUFFALO
BERASOL ARCHIBAL
```

"QSORT" SUBROUTINE

This subroutine sorts a string array A$ into ascending (or descending) order such that A$[K]<(or>)A$[K + 1] for K = b, b + 1, ... t - 1,t where the user can specify the sorting bounds b and t. The sort can use the whole string or any substring as a key to allow maximum flexibility.

The method implemented is similar to QUICKERSORT by R.S. Scowen (see references) and has been shown to be among the fastest sorting algorithms available.

The algorithm partitions the array into two smaller arrays with all the elements in one array less than the elements in the other. The larger (in length) of the two arrays is then partitioned by the same method recursively while saving the smaller segment to be sorted later.

When used as a subroutine, all parameters specified under <INPUT> must be assigned in the user's calling program. The user may then append the subroutine to his program and use gsb "QSORT" to take advantage of its capabilities. The results of the sort and any other output is designated under <OUTPUT>. The user should be aware that the subroutine may alter variables that he is using in his program and should therefore check the variables listed under <DESTROYED>.

An example of a typical program which uses this subroutine may be found in file 32 on page 242.

PARAMETERS AND VARIABLES USED:

<INPUT>

N	number of elements in the string array
L	length of the strings (longest string) in array
A$[N,L]	string array
B	number of array element where sort should start
T	number of array element where sort should stop
Y	number of first character of substring used as key
Z	number of last character of substring used as key
flg 3	set if you want increasing order; clear if you want decreasing order

<OUTPUT>

N, L, Y, Z, B, T, flg 13	unchanged
A$[N,L]	sorted array

<DESTROYED>

I	lower segment bound
J	upper segment bound
L	temporary segment lower bound; length of strings
M	pointer to median of segment int((I + J)/2)
L[ ]	stack for lower bound of unsorted segments
U[ ]	stack for upper bound of unsorted segments
T$	temporary storage for exchanging element; median value used for sorting upper and lower division elements in a segment.
V$	temporary storage for exchanging upper and lower division elements in a segment.

"SSORT" SUBROUTINE

This subroutine sorts a string array A$ into ascending (or descending) order such that A$[K]<(or>)A$[K + 1] for k = b, b + 1, ..., t - 1, t where the user can specify the sorting bounds b and t. The sort can use the whole string or any substring as a key to allow maximum flexibility.

The method implemented is a shellsort (see references) which uses a minimal amount of intermediate storage. The algorithm chooses a comparison interval of int(N/2) where N is the number of elements in the array. All elements separated by the interval distance are put into ascending (or descending) order. The interval is then halved and the process repeated until the interval size is one.

When used as a subroutine, all parameters specified under <INPUT> must be assigned in the user's calling program. The user may then append the subroutine to his program and use gsb "SSORT" to take advantage of its capabilities. The results of the sort and any other output is designated under <OUTPUT>. The user should be aware that the subroutine may alter variables that he is using in his program and should therefore check the variables listed under <DESTROYED>.

An example of a typical program which uses this subroutine may be found in file 32 on page 242.

PARAMETERS AND VARIABLES USED:

<INPUT>

N number of elements in the string array

L	length of the strings (longest string) in array
A$[N,L]	string array
B	number of array element where sort should start
T	number of array element where sort should stop
Y	number of first character of substring used as key
Z	number of last character of substring used as key
flg 13	set - if you want increasing order clear - if you want decreasing order

<OUTPUT>

N, L, B, T, Y, Z unchanged

A$[N,L]	sorted array

<DESTROYED>

I	secondary loop counter which modifies array such that intervals of M are in order.
J	loop counter incremented by one to T - 1
K	upper bound for sort comparison loop counter (I)
M	diminishing increment
T$	temporary storage for exchanging strings

INSTRUCTIONS:

1. Load your program into memory. The program should set up all necessary parameters (N, L, A$[N,L], B, T, Y, Z, flg 13) to be passed to the subroutine.

2. Insert UTILITY ROUTINES cartridge.

3. Load the file.

   a. Type ldf 33, N, M (for "QSORT")
   or type ldf 34, N, M (for "SSORT") where:

      N = line number where the first line of the subroutine should be loaded into memory.

      M = line number where execution is to begin after subroutine is loaded.

   b. Press EXECUTE

ANNOTATED LISTING:

STRING ARRAY SORT

```
0: cfg 13;prt
 "ENTER 1","FOR
 QSORT";spc
1: prt "ENTER
 2";enp "FOR
 SSORT",J
2: spc ;if J=2;
 ldf 34,25,4
3: ldf 33,25,4
4: enp "NO OF
 STRINGS",N;spc
 ;if flg13;cfg
 13;gto +0
5: enp "LENGTH
 OF STRING",L;
 spc ;if flg13;
 cfg 13;gto +0
6: dim A$[N,L];
 1→I;spc 2
7: cfg 13;enp
 A$[I];gto +0;
 if not flg13;
 spc ;jmp (I+
 1→I)>N
8: enp "CHANGES?
 ",I;spc ;if
 flg13;cfg 13;
 gto +3
9: enp "STRING
 NUMBER?",I;if
 flg13;cfg 13;
 gto +0
10: cfg 13;enp
 A$[I];gto +0;
 if not flg13;
 spc ;gto -2
11: spc ;prt
 "ENTER SUBSET
 OF","ARRAY TO
 BE","SORTED";
 spc 2
12: enp "LOW
 BOUND?",B;if
 flg13;cfg 13;
 gto +0
13: enp "UPPER
 BOUND?",T;if
 flg13;cfg 13;
 gto +0
14: spc ;prt
 "ENTER SUBSTRIN
 G","USED AS
 KEY";spc
```

Enter 1 to use the QSORT subroutine

Enter 2 to use the SSORT subroutine

Load the appropriate file

Enter the number of strings

Enter the longest string that will be in the array

Enter the strings into the string array

Make any necessary changes

Enter the lower bound for the sort

Enter the upper bound for the sort

ANNOTATED LISTING:

```
15: enp "LOW
 BOUND?",Y;if
 flg13;cfg 13;
 gto +0
```
Enter the lowest character number that is used as sort key

```
16: enp "UPPER
 BOUND?",Z;if
 flg13;cfg 13;
 gto +0
```
Enter the highest character number that is used as sort key

```
17: cfg 13;spc
 2;enp "DECREASI
 NG ORDER?",I
```
Enter 1 if you want to sort in decreasing order

```
18: if J=2;gto +
 3
19: gsb "QSORT"
20: gto +2
21: gsb "SSORT"
```
Branch to appropriate subroutine

```
22: 1→I;spc 2;
 prt "--SORTED
 ARRAY--";spc
23: prt A$[I];
 jmp (I+1→I)>N
24: spc 4;end
*29668
```
Print sorted array

ANNOTATED LISTING:

QUICK SORT

```
0: "QSORT":2→I;
 1→J;dim T$[L],
 V$[L]
```
T$ and V$ are used for temporary storage

```
1: if I>N;dim
 U[J],L[J];sto +
 2
```
Compute maximum amount of storage needed to hold the unsorted segment stack

```
2: 2I→I;J+1→J;
 sto -1
```

```
3: 1→S;B→I;T→J
```
Initialize stack pointer and segment bounds

```
4: if I>=J;sto +
 15
```
Find median value

```
5: I→L;int(.5(I+
 J))→M;A$[M]→T$
```

```
6: if A$[I,Y,
 Z]<=T$[Y,Z];
 sto +2
```
Compare low bound to median

```
7: A$[I]→A$[M];
 T$→A$[I];A$[M]→
 T$
```
Switch if low bound greater than median

```
8: J→R;if A$[J,
 Y,Z]>=T$[Y,Z];
 sto +5
```
Compare high bound to median

```
9: A$[J]→A$[M];
 T$→A$[J];A$[M]→
 T$
```
Switch if high bound smaller than median

```
10: if A$[I,Y,
 Z]<=T$[Y,Z];
 sto +3
```
Compare new median to lower bound

```
11: A$[I]→A$[M];
 T$→A$[I];A$[M]→
 T$;sto +2
```
Switch if new median smaller than lower bound

```
12: A$[L]→A$[R];
 V$→A$[L]
```
Complete upper and lower segment switch

```
13: R-1→R;if
 A$[R,Y,Z]>T$[Y,
 Z];sto +0
```
Is this upper segment element larger than median?

```
14: A$[R]→V$
```
Begin switch

```
15: L+1→L;if
 A$[L,Y,Z]<T$[Y,
 Z];sto +0
16: if L<=R;sto
 -4
```
Find element in lower segment to switch with

```
17: if R-I<=J-L;
 1→L[S];R→U[S];
 L→I;S+1→S;sto +
 4
18: L→L[S];J→U[S
 ];R→J;S+1→S;
 sto +3
```
Put smaller of upper and lower segments on the stack to be sorted later

```
19: S-1→S;if
 S=0;sto +12
20: L[S]→I;U[S]→
 J
```
Pop segment from stack to be sorted

ANNOTATED LISTING:

```
21: if J-I>11;                    If more than 11 elements in segment repeat
 gto -16                          sorting process on that segment
22: if I=B;gto -                              SINKING SORT
 18                               If 11 elements or less are in the segment
23: gto +2                        use a sinking sort method
24: I+1→I
25: if I=J;gto -                  If sort is finished pop another segment off
 6                                the stack
26: A$[I+1]→T$;
 if A$[I,Y,Z]<=T                  Compare (I+1)st and Ith element in segment
 $[Y,Z];gto -2
27: I→L
28: A$[L]→A$[L+                   Shift element up
 1];L-1→L
29: if T$[Y,Z]<A
 $[L,Y,Z];gto -1                  Complete switch
30: T$→A$[L+1];
 gto -6
31: if flg13;
 ret                              If flg 13 is set, switch the order of
32: B→I;T→J                       all elements in the sorted array
33: if I>=J;ret
34: A$[I]→T$;
 A$[J]→A$[I];
 T$→A$[J];I+1→I;
 J-1→J;gto -1
*1232
```

ANNOTATED LISTING:

SHELL SORT

```
0: "SSORT":N→M;
 dim T$[L]
```
T$ used as temporary storage for switch

```
1: if (int(M/
 2)→M)=0;sto +8
```
Halve comparison interval; if zero branch to finish

```
2: T-M→K;B→J
```
Set bounds for sorting

```
3: J→I
4: if A$[I,Y,
 Z]<=A$[I+M,Y,
 Z];sto +3
```
Compare elements A$[I] and A$[I+M]

```
5: A$[I]→T$;A$[I
 +M]→A$[I];T$→A$
 [I+M];I-M→I
```
Switch elements

```
6: if I>=B;sto -
 2
```
Check previous interval if possible

```
7: if (J+1→J)>K;
 sto -6
8: sto -5
```
Check if end of interval for this pass

```
9: if flg13;ret
10: B→I;T→J
11: if I>=J;ret
12: A$[I]→T$;
 A$[J]→A$[I];
 T$→A$[J];I+1→I;
 J-1→J;sto -1
```
If flag 13 is set reverse the order of the complete sorted array

```
*23234
```

Array Sort On Row Or Column Index

FILE 35 DRIVER

FILE 36 SUBROUTINE "ISORT"

Array Sort On Row Or Column Index

This program will sort a user specified row (or column) of an array and permute the elements of the other rows (or columns) to agree with the sorted row (or column). The user can also specify whether the sort should be increasing or decreasing.

The program requests all necessary information, calls the subroutine "ISORT" in file 36 on page 256, and prints the sorted array.

SPECIAL CONSIDERATIONS:

1. The maximum number and size of arrays that can be handled depends on the size of the calling program. If the calling program in file 35 on page 260 is used, the following size limitations apply:

   If N is the number of rows and M the number of columns then:

   N*M + 2T < 631 on the 6,844 byte machine
   N*M + 2T < 1655 on the 15,036 byte machine
   N*M + 2T < 2679 on the 23,228 byte machine
   N*M + 2T < 3703 on the 31,420 byte machine

   Where T = M if sort by row and T = N if sort by column.

2. The speed of the sort is approximately proportional to $N^{1.5}$ where N is the number of elements in the array. An example with 60 elements was sorted in approximately one second.

REFERENCES:

1. Knuth, Donald E., Searching and Sorting (Addison Wesley, 1973), pp. 84 - 87.

2. Shell, D.L., "A High Speed Sorting Procedure", Communications of the ACM 2 (July 1959), pp. 30 - 32.

INSTRUCTIONS:

1. Insert UTILITY ROUTINES cartridge with machine turned on.
2. Load the file.
   a. Type ldf 35
   b. Press EXECUTE
3. Press RUN
4. When "DECREASING SORT?" is displayed:
   Either
   a. Press CONTINUE if you want to sort into increasing order.
   b. Go to step 9.
   or
   a. Type gto "L" if you want to sort into decreasing order.
   b. Press EXECUTE
5. Press STEP
6. Fetch the line.
   a. Observe the number in the display and press FETCH followed by that number.
   b. Press EXECUTE
7. ' "L": if A[rW, rX] < = A[rY, rZ]; gto + 4' will be displayed:
   a. Move the cursor to the "<" character.
   b. Enter ">" (the ">" symbol should replace the "<" symbol in the display).
   c. Press STORE
8. Continue execution
   a. Type cont 2 (you must type the word "cont" character by character rather than pressing the CONTINUE key).
9. "MATRIX A" will be printed.
   When "NO. OF ROWS?" is displayed:
   a. Enter the number of rows in the matrix.
   b. Press CONTINUE
10. When "NO. OF COLS?" is displayed:
    a. Enter the number of columns in the matrix.

b. Press CONTINUE

11. When "A[I,J]" is displayed, "I" and "J" will denote the row and column number of the next element to be entered.
    a. Enter the value of the element in the I^{th} row and J^{th} column.
    b. Press CONTINUE
    c. Repeat step 11 until all matrix elements have been entered.

12. When "CHANGES?" is displayed:
    Either
    a. Press CONTINUE , if you do not want to make changes to the entered data.
    b. Go to step 16.
    or
    a. Enter 1 if you need to make changes.
    b. Press CONTINUE

13. When "ROW?" is displayed:
    a. Enter the row number of the element to be changed.
    b. Press CONTINUE

14. When "COLUMN?" is displayed:
    a. Enter the column number of the element to be changed.
    b. Press CONTINUE

15. When "A[I,J]" is displayed:
    a. Enter the correct element for the I^{th} row and J^{th} column.
    b. Press CONTINUE
    c. Go to step 12.

16. When "ROW OR COL INDEX" is displayed:
    a. Enter the number of the row or column which is to be the index for the sort.
    b. Press CONTINUE

17. When "SORT BY ROW?" is displayed:
    Either
    Press CONTINUE , if the index to be sorted is a column
    or
    a. Enter 1 if the index to be sorted is a row.
    b. Press CONTINUE

18. The sorted matrix will be printed.

DATA EXAMPLE

STUDENT NUMBER	AGE	YEAR	HEIGHT (INCHES)	GRADE
034206864	19	3	61	90
127635124	20	4	58	95
329876410	18	3	54	83
921764444	20	4	63	72
642971302	19	2	60	65
544987713	18	2	57	81
224100618	21	4	60	91
421110677	21	4	59	60
597799011	19	3	66	75
123498210	18	2	60	89

The data can be entered into an array A[10, 5] and sorted by any column desired.

EXAMPLE:

```
MATRIX A

NO. OF ROWS?
10
NO. OF COLS?
5

A[1,1]?
034206864

A[1,2]?
19

A[1,3]?
3

A[1,4]?
61

A[1,5]?
90


A[2,1]?
127635124

A[2,2]?
20

A[2,3]?
4

A[2,4]?
58

A[2,5]?
95


A[3,1]?
329876410

A[3,2]?
18

A[3,3]?
3

A[3,4]?
54

A[3,5]?
83


A[4,1]?
921764444

A[4,2]?
20

A[4,3]?
4

A[4,4]?
63

A[4,5]?
72


A[5,1]?
642971302

A[5,2]?
19

A[5,3]?
2

A[5,4]?
60

A[5,5]?
65


A[6,1]?
544987713

A[6,2]?
18

A[6,3]?
2

A[6,4]?
57

A[6,5]?
81


A[7,1]?
224100618

A[7,2]?
21

A[7,3]?
4

A[7,4]?
60

A[7,5]?
91


A[8,1]?
421110677

A[8,2]?
21

A[8,3]?
4

A[8,4]?
59

A[8,5]?
60


A[9,1]?
597799011

A[9,2]?
19

A[9,3]?
3

A[9,4]?
66

A[9,5]?
75


A[10,1]?
123498210

A[10,2]?
18
```

SORTED MATRIX PRINTED BY COLUMNS

EXAMPLE:

```
A[10,3]?
2

A[10,4]?
60

A[10,5]?
89

CHANGES?

ROW OR COL INDEX
5

SORT BY ROW?
```

```
421110677
642971302
921764444
597799011
544987713
329876410
1234982210
 34206864
224100618
127635124
```

Student Number

```
21
19
20
19
18
18
18
19
21
20
```

Age

```
4
2
4
3
2
3
2
3
4
4
```

Class

```
59
60
63
66
57
54
60
61
60
58
```

Height in Inches

```
60
65
72
75
81
83
89
90
91
95
```

Sorted Grades

"ISORT" SUBROUTINE

This subroutine will sort a user specified row (or column) of an array and permute the elements of the other rows (or columns) to agree with the sorted row (or column). The user can also specify whether the sort should be increasing or decreasing.

The method implemented is the shellsort (see references) which uses a minimal amount of intermediate storage. The algorithm chooses a comparison interval of int(N/2), where N is the number of elements in the array. All elements separated by the interval distance are put into ascending or descending order. The interval is then halved and the process repeated until the interval size is one.

When used as a subroutine, all parameters specified under <INPUT> must be assigned in the user's calling program. The user may then append the subroutine to his program and use gsb "ISORT" to take advantage of its capabilities. The results of the sort and any other output is designated under <OUTPUT>. The user should be aware that the subroutine may alter variables that he is using in his program and should therefore check the variables listed under <DESTROYED>.

An example of a typical program which uses this subroutine may be found in file 35 on page 260.

PARAMETERS AND VARIABLES USED:

<INPUT>

N number of rows

M	number of columns
A[N,M]	matrix to be sorted
r0	number of column or row to be used as index
flg 13	set - if sorting a column index clear - if sorting a row index

<OUTPUT>

N, M, r0, flg 13	unchanged
A[N,M]	matrix sorted on index r0

<DESTROYED>

I	loop index, register index of 2 if sorting by columns or 1 if sorting by rows.
J	counter from one to K; register index of 2 if sorting by rows or 1 if sorting by columns.
K	number of elements in a row (or column) minus the interval size. Used as a terminator for comparison loop.
Q	N if sorted by columns (number of rows) M if sorted by row (number of columns)
R	M if sorted by row (number of columns) N if sorted by columns (number of rows)
S	size of interval of comparison
T	temporary storage for exchanging elements
W	0 if sorted by rows 1 if sorted by columns
X	0 if sorted by columns 2 if sorted by rows
Y	0 if sorted by rows 2 if sorted by columns

Z	0 if sorted by columns 1 if sorted by rows
P[ ]	permutation vector which indicates the rearranged order of index vector
Q[ ]	temporary storage for row (or column), used during the application of permutation vector to array.
r1	loop counter for row (if sorting by columns) or columns (if sorting by rows). Used in sorting section and permutation section.
r2	r1 plus interval size in comparison section. Loop counter for columns (if sorting by columns) or rows (if sorting by rows).
r3	row subscript
r4	column subscript

INSTRUCTIONS:

1. Load your program into memory. The program should set up all necessary parameters (N, M, A[N,M], flg 13, r0) to be passed to the subroutine.

2. Insert UTILITY ROUTINES cartridge.

3. Load the file.

   a. Type ldf 36, N, M where:

      N = line number where the first line of the subroutine should be loaded into memory.

      M = line number where execution is to begin after subroutine is loaded.

   b. Press EXECUTE

ANNOTATED LISTING:

DRIVER - ARRAY SORT

```
0: ldf 36,21,1
1: dsp "DECREASI
 NG SORT?";stp
2: cfg 13;dsp
 "MATRIX A";spc
 3;prt "MATRIX
 A";spc ;wait
 500
3: enp "NO. OF
 ROWS?",N;if
 flg13;cfg 13;
 gto +0
4: cfg 13;enp
 "NO. OF COLS?",
 M;if flg13;cfg
 13;gto +0
5: spc ;0→I;1→J;
 dim A[N,M]
6: if (I+1→I)>N;
 gto +3
7: cfg 13;enp
 A[I,J];gto +0;
 if not flg13;
 spc ;jmp (J+
 1→J)>M
8: 1→J;spc ;gto
 -2
9: spc ;enp "CHA
 NGES?",X;spc ;
 if flg13;gto +4
10: enp "ROW?",
 I;if flg13;cfg
 13;gto +0
11: enp "COLUMN?
 ",J;if flg13;
 cfg 13;gto +0
12: cfg 13;enp
 A[I,J];gto +0;
 if not flg13;
 spc ;gto -3
13: enp "ROW OR
 COL INDEX",r0;
 if flg13;cfg
 13;gto +0
14: cfg 13;spc ;
 enp "SORT BY
 ROW?",I;spc
15: gsb "ISORT"
16: spc 2;prt "
 SORTED MATRIX",
 "   PRINTED
 BY","    COLUMN
 S";spc 2
```

Load the subroutine

If decreasing sort, change line in subroutine

ENTER MATRIX

Enter number of rows

Enter number of columns

Enter the matrix elements

Make any necessary changes

Enter the column or row number which is to be used as the index

Enter 1 if the rows are to be sorted or press Continue if columns are to be sorted

Branch to subroutine

Print heading for sorted matrix

ANNOTATED LISTING:

```
17: 1→I;0→J
18: if (J+1→J)>M
 ;spc 4;end
19: prt A[I,J];                    Print sorted matrix
 jmp (I+1→I)>N
20: spc ;1→I;
 gto -2
*4083
```

SUBROUTINE ARRAY SORT

```
0: "ISORT":if
 fl≠13;0→X→Z;
 N→Q→S;M→R;≠to +
 2
```
Sort by columns - rX and rZ become r0 which is the column index

```
1: 0→W→Y;M→Q→S;
 N→R
```
Sort by row - rW and rY become r0 which is the row index

```
2: dim P[Q];1→I
3: I→P[I];jmp
 (I+1→I)>Q
```
P[] is the permutation vector initialized such that P[I]=I

```
4: if (int(S/
 2)→S)=0;≠to +11
```
S is interval size. If size is zero, branch to permute section to sort the rest of the matrix

```
5: Q-S→K;1→J
```
K is upper bound of comparison intervals

```
6: J→r1
7: r1+S→r2;if
 fl≠13;1→W;2→Y;
 ≠to +2
```
r1 and r2 index the row or column elements to be compared

```
8: 1→X;2→Z
9: "L":if A[rW,
 rX]<=A[rY,rZ];
 ≠to +4
```
Compare two elements

```
10: P[r1+S]→T;
 P[r1]→P[r1+S];
 T→P[r1]
```
Update permutation vector

```
11: A[rW,rX]→T;
 A[rY,rZ]→A[rW,
 rX];T→A[rY,rZ];
 r1-S→r1
```
Switch elements

Decrement interval index

```
12: if r1>=1;
 ≠to -5
```
If possible compare switched element with previous one

```
13: if (J+1→J)>K
 ;≠to -9
14: ≠to -8
```
Continue until interval size is 1

```
15: 0→r1→r2;dim
 Q[Q]
```
PERMUTE REMAINING ELEMENTS

```
16: if fl≠13;
 2→I;1→J;≠to +2
```
Q[] used for temporary storage in switch

```
17: 1→I;2→J
18: if (r1+1→r1)
 >R;ret
```
Row or column index (row if sorting by rows)

```
19: if r1=r0;
 ≠to -1
```
If row (or column) = index, don't permute

```
20: if (r2+1→r2)
 >Q;0→r2;≠to -2
```
Row or column index (column if sorting by rows)

```
21: if P[r2]=r2;
 ≠to -1
```
If permuted value same as orginal don't permute

```
22: if P[r2]<r2;
 Q[P[r2]]→A[rI,
 rJ];≠to -2
```
If permuted value less than index copy temporary storage into array in correct place

```
23: A[rI,rJ]→Q[r
 2];if fl≠13;
 P[rI]→r3;rJ→r4;
 ≠to +2
```
Copy array element into temporary storage for later use

ANNOTATED LISTING:

```
24: P[rJ]→r4;                    Copy permuted value into correct position
  rI→r3
25: A[r3,r4]→A[r
  I,rJ];goto -5
*11432
```

Output Buffer For Plotter

FILE 37 DRIVER

FILE 38 SUBROUTINES "BUFFER", "SAVE"

FILE 39 DATA FOR DRIVER IN FILE 37

Output Buffer For Plotter

This routine will create an output buffer for the plotter to allow overlap in computation of the points to be plotted and the plotting of the points. The routine initializes the buffer by calling the subroutine "BUFFER" in file 38, computes the points to be plotted and calls the subroutine "SAVE" to enter the points into the buffer.

Although the points can be computed in various ways, this example simply inputs the triplets (X, Y, P) and stores the resultant data on a cartridge file. The user could already have a data file filled with points or more likely, he could compute the points during program execution. The data file is then loaded back into the machine memory and the individual points are massaged before being entered into the buffer.

After all the data points have been entered into the buffer the program will not end until the plotter has emptied the buffer. The program could be performing other computations while the buffer is being emptied.

The user determines the size of the buffer used and can adjust the size if he finds that the buffer is usually full or empty. The full or empty buffer state can be determined by observing how often the message "BUFFER FULL" appears in the display while the program in running.

SPECIAL CONSIDERATIONS AND DIAGNOSTIC MESSAGES:

1. The maximum buffer size and number of data points plotted depends on the size of the calling program. If the program in file 37 is used then the following limitations hold:

   If K is the number of triplets (x_i, y_i, p_i), i = 1,K and N is the size of the buffer then:

K + N < 232 on the 6,844 byte machine

K + N < 1256 on the 15,036 byte machine

K + N < 2280 on the 23,228 byte machine

The 31,420 byte option is not compatible with the Extended I/O ROM.

2. The Plotter, General I/O and Extended I/O ROMs must be in the machine and all necessary initialization of the plotter itself should be completed before the program is run (e.g., lower left, upper right, axes, scale, etc.).

3. "BUFFER FULL" indicates that the program has filled the output buffer and the plotter is busy. If this message appears often, the buffer size should be increased.

4. "WAITING FOR PLOTTER" indicates that the program has put all the data into the buffer and is waiting for the plotter to empty the buffer.

INSTRUCTIONS:

1. Perform all necessary plotter control operations (see Plotter Control Manual).
2. Insert UTILITY ROUTINES cartridge with machine on.
3. Load the file.
   a. Type ldf 37
   b. Press EXECUTE
4. Press RUN
5. When "NO. OF POINTS?" is displayed:
   a. Enter the number of data triplets (X, Y, P) to be plotted.
   b. Press CONTINUE
6. When "X[I]" is displayed, "I" will correspond to the I^{th} data point.
   a. Enter the X coordinate of the I^{th} data point.
   b. Press CONTINUE
7. When "Y[I]" is displayed:
   a. Enter the Y coordinate of the I^{th} data point.
   b. Press CONTINUE
8. When "P[I]" is displayed:
   a. Enter the pen control for the I^{th} data point.
   b. Press CONTINUE
   c. If more data points are to be entered, go to step 6.
9. When "CHANGES?" is displayed:
   Either
   a. Press CONTINUE if you do not want to make any changes.
   b. Go to step 14.
   or
   a. Enter 1 if you need to make changes.
   b. Press CONTINUE
10. When "DATA TRIPLET NO?" is displayed:
   a. Enter the number of the data point to be changed.
   b. Press CONTINUE

11. When "X[I]" is displayed, "I" will be the data point number to be changed.
    a. Enter the correct X coordinate.
    b. Press CONTINUE

12. When "Y[I]" is displayed:
    a. Enter the correct Y coordinate.
    b. Press CONTINUE

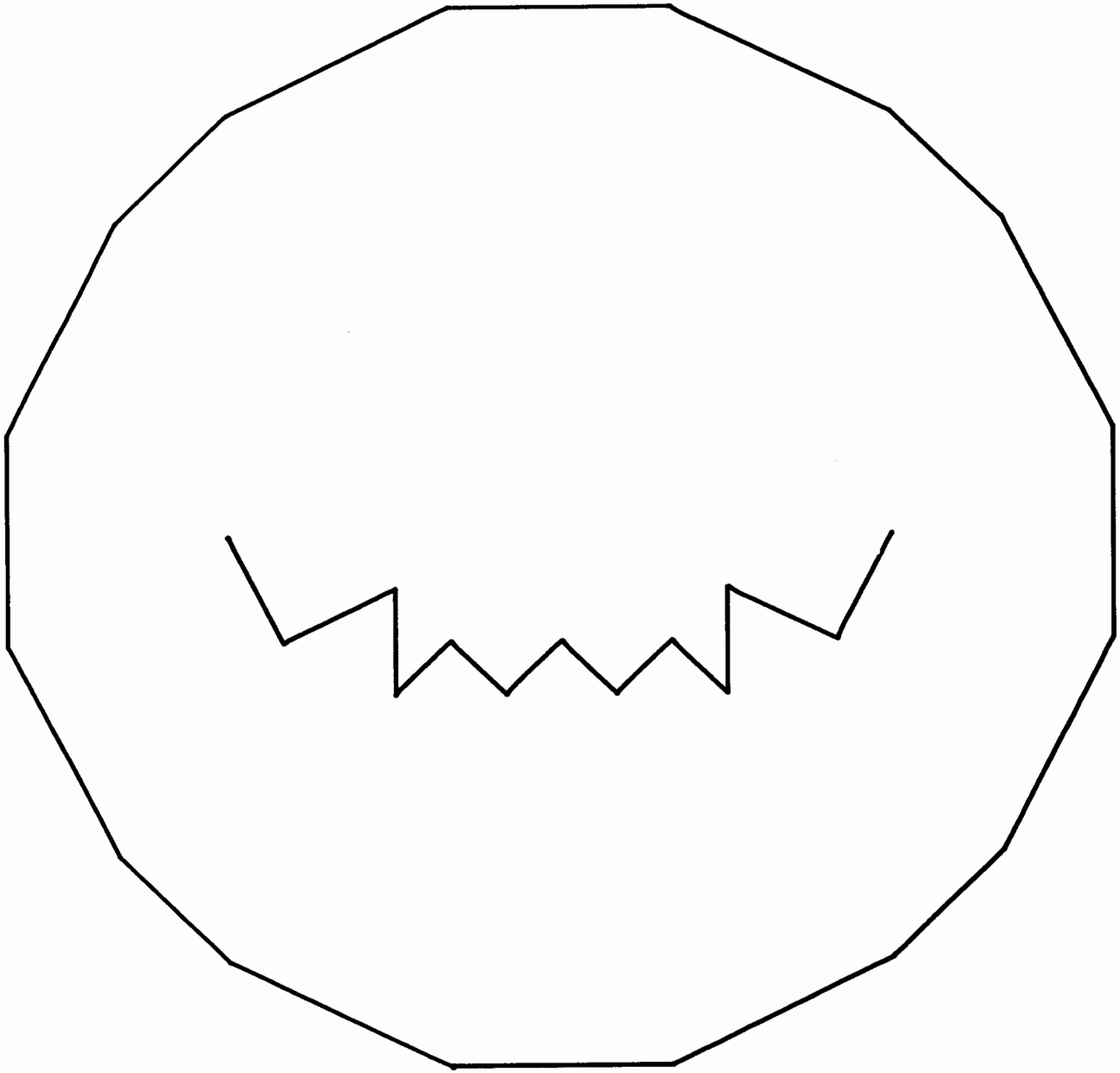
13. When "P[I]" is displayed:
    a. Enter the correct pen control parameter.
    b. Press CONTINUE
    c. Go to step 9.

14. When "SELECT CODE FOR PLOTTER" is displayed:
    a. Enter the select code for the plotter.
    b. Press EXECUTE

15. The data points will be plotted.

EXAMPLE:

```
NO. OF POINTS?
30

X[1]?
4
Y[1]?
0
P[1]?
-2

X[2]?
2
Y[2]?
1
P[2]?
0

X[3]?
1
Y[3]?
2
P[3]?
0

X[4]?
0
Y[4]?
4
P[4]?
0

X[5]?
0
Y[5]?
6
P[5]?
0

X[6]?
1
Y[6]?
8
P[6]?
0

X[7]?
2
Y[7]?
9
P[7]?
0

X[8]?
4
Y[8]?
10
P[8]?
0

X[9]?
6
Y[9]?
10
P[9]?
0

X[10]?
8
Y[10]?
9
P[10]?
0

X[11]?
9
Y[11]?
8
P[11]?
0

X[12]?
10
Y[12]?
6
P[12]?
0

X[13]?
10
Y[13]?
4
P[13]?
0

X[14]?
9
Y[14]?
2
P[14]?
0

X[15]?
8
Y[15]?
1
P[15]?
0

X[16]?
6
Y[16]?
0
P[16]?
0

X[17]?
4
Y[17]?
0
P[17]?
-1

X[18]?
2
Y[18]?
5
P[18]?
-2

X[19]?
2.5
Y[19]?
4
P[19]?
0

X[20]?
3.5
Y[20]?
4.5
P[20]?
0

X[21]?
3.5
Y[21]?
3.5
P[21]?
0

X[22]?
4
Y[22]?
4
P[22]?
0

X[23]?
4.5
Y[23]?
3.5
P[23]?
0

X[24]?
5
Y[24]?
4
P[24]?
0

X[25]?
5.5
Y[25]?
3.5
P[25]?
0

X[26]?
6
Y[26]?
4
P[26]?
0

X[27]?
6.5
Y[27]?
3.5
P[27]?
0

X[28]?
6.5
Y[28]?
4.5
P[28]?
0

X[29]?
7.5
Y[29]?
4
P[29]?
0

X[30]?
8
Y[30]?
5
P[30]?
-1

CHANGES?

SELECT CODE FOR
PLOTTER
5
BUFFER SIZE?
5
```



VICIOUS CIRCLE

SUBROUTINES FOR BUFFERED OUTPUT TO PLOTTER

This set of subroutines creates an output buffer for the plotter to allow overlap in the computation of the points to be plotted and the plotting of the points.

The "BUFFER" subroutine creates the buffer, initializes the pointers to indicate that the buffer is empty and sets up a call on the "PLOT" subroutine whenever the plotter is ready to plot a point.

The "SAVE" subroutine will insert a data point into the buffer if the buffer is not full, or it will wait until the plotter removes a point if the buffer is full.

The "PLOT" subroutine plots a data point from the buffer whenever an interrupt from the plotter occurs.

When using these subroutines, all parameters specified under <INPUT> must be assigned in the user's calling program. The user may then append the subroutines to his program and use gsb "BUFFER" and gsb "SAVE" to take advantage of their capabilities. The user should be aware that the subroutines may alter variables that he is using in his program and should therefore check the variables listed under <DESTROYED>.

An example of a typical program which uses these subroutines may be found in file 37 on page 277.

PARAMETERS AND VARIABLES USED:

"BUFFER" SUBROUTINE

<INPUT>

N	size of buffer (maximum number of points that can be stored in the buffer)
r0	select code for plotter

<OUTPUT>

F	pointer to location where next data point will be stored.
R	pointer to location with next data to be plotted (when F=R the buffer is empty and the interrupt from plotter is disabled).

<DESTROYED>

B[N,3]	the buffer for N data points.

"SAVE" SUBROUTINE

<INPUT>

N	size of buffer
F	pointer to location where next element is to be inserted into buffer
R	pointer to location containing next element to be plotted
B[N,3]	output buffer where N is the size of the buffer
X	x-coordinate of data point
Y	y-coordinate of data point
P	pen coordinate for data point

r0 select code for plotter

<OUTPUT>

B[F,1], B[F,2], B[F,3] contain X, Y, P respectively

F incremented by 1 modulo N and
interrupt from plotter enabled.

<DESTROYED>

B[F,1], B[F,2], B[F,3] the original contents are replaced by X, Y, P respectively

INSTRUCTIONS:

1. Load your program into memory. The program must set up all necessary parameters (N, X, Y, P) to be passed to the subroutines.

2. Insert UTILITY ROUTINES cartridge.

3. Load the file.

   a. Type ldf 38, N, M where:

      N = line number where the first line of the subroutine should be loaded.

      M = line number where execution is to begin after subroutine is loaded.

   b. Press EXECUTE

ANNOTATED LISTING:

BUFFERED OUTPUT TO PLOTTER

```
0: deg;cfg 13;
 ldf 38,22,1
```
Degree measure used for angles; load the subroutine

```
1: enp "NO. OF
 POINTS?",K;if
 flg13;cfg 13;
 gto +0
2: dim X[K],Y[K]
 ,P[K];1→I;spc
```
Enter the number of data points to be plotted and dimension the X,Y and P arrays

```
3: enp X[I];if
 flg13;cfg 13;
 gto +0
4: enp Y[I];if
 flg13;cfg 13;
 gto +0
5: enp P[I];if
 flg13;cfg 13;
 gto +0
6: if (I+1→I)<=K
 ;spc ;gto -3
```
Enter the data triplets (X[I], Y[I], P[I]) where (X,Y) are the coordinates and P is the pen control

```
7: spc ;cfg 13;
 enp "CHANGES?",
 I;spc ;if flg13
 ;gto +5
8: enp "DATA
 TRIPLET NO?",I;
 spc ;if flg13;
 cfg 13;gto +0
9: enp X[I];if
 flg13;cfg 13;
 gto +0
10: enp Y[I];if
 flg13;cfg 13;
 gto +0
11: cfg 13;enp
 P[I];gto +0;if
 not flg13;spc ;
 gto -4
```
Make any necessary changes

```
12: enp "SELECT
 CODE FOR PLOTTE
 R",r0;if flg13;
 cfg 13;gto +0
```
Enter the select code for the plotter

```
13: rcf 39,K,X[*
 ],Y[*],P[*]
```
Record the entered data on a file

```
14: enp "BUFFER
 SIZE?",N;if
 flg13;cfg 13;
 gto +0
```
Enter the size of the output buffer

```
15: gsb "BUFFER"
```
Branch to the buffer set up subroutine

```
16: ldf 39,K,X[*
 ],Y[*],P[*];1→I
```
Load the data

```
17: X[I]→X;Y[I]→
 Y;P[I]→P
```

ANNOTATED LISTING:

```
18: gsb "SAVE"
19: if (I+1→I)<=
 K;sto -2
```

Put one data triplet into buffer
Get next data triplet

```
20: if F#R;dsp
 "WAITING FOR
 PLOTTER";sto +0
21: dsp ;pen;
 plt 3,9,-1,0;
 lbl "VICIOUS
 CIRCLE";pen;end
*8459
```

If F#R the buffer is not empty yet

ANNOTATED LISTING:

SUBROUTINE - BUFFERED OUTPUT TO PLOTTER

	BUFFER SET UP
0: "BUFFER":dim B[0:N-1,3];1→F→R;eir r0,0	F and R are the input and output pointers. Disable interrupt from plotter initially. Set up interrupt branch to plot subroutine.
1: oni r0,"PLOT" ;ret	
2: "SAVE":if (1+F)modN=R;dsp "BUFFER FULL";	FILL BUFFER Check if buffer is full. Wait until a point is plotted.
gto +0	
3: dsp ;X→B[F,1];Y→B[F,2]; P→B[F,3];(1+F)modN→F	Put data triplet into buffer. No interrupt can occur until whole line is executed.
4: eir r0;ret	Enable the plotter interrupt.
5: "PLOT":plt B[R,1],B[R,2], B[R,3];(1+R)mod N→R	Plot the next available point in the buffer
6: if F#R;eir r0	Enable interrupt unless the buffer is empty.
7: iret	
*1305	

Section 3

PLOTTING AND TEXT-GENERATING
ROUTINES FOR THE 9871A

PROGRAM	FILES	PAGE
Plotter Subroutines for 9871A	40	294
Text-Generating Subroutines for 9871A	41	299
Plotting Program Example 1	42	348
Text-Generating Program Example 2	43	350
Plotting Program Example 3	44	354

All files are on track 0

INDEX

	PAGE
INTRODUCTION	284
PROGRAMMING HINTS	285
READING THE MANUAL	286
LOADING INSTRUCTIONS	288
LISTING OF ALL PROGRAMS BY FILES	291
SUBROUTINE DOCUMENTATION AND EXPLANATION	293
PROGRAMMING EXAMPLES	348
APPENDIX A - ADDITIONAL COMMANDS	357
APPENDIX B - CHARACTER CODES	359
APPENDIX C - POWER-UP STATE AND PLOT AREA	361

INTRODUCTION

This set of subroutines allows the 9825A desk top calculator to drive the 9871A printer. With this implementation, plotting,tabbing, axis drawing, etc., are accomplished via subroutine calls such as cll 'plt', cll'tab', cll 'xaxis', etc.

The subroutines are divided into a file for plotting routines and a file for text-generating routines. The purpose of this dichotomy is to save memory space. The routines that can be used in either plotting or text printing are included in both files. The files can be combined into one all-purpose file or can be edited by deleting routines which may not be used by your program. The loading and editing procedures are elucidated in the LOADING INSTRUCTIONS section.

PROGRAMMING HINTS

1. To achieve faster execution of your program, you can load the most frequently called subroutines into the beginning of memory with your program loaded after these. Then if you insert a "gto" statement which branches around the subroutines, you can RUN your programs as if they were located at the beginning of memory. The program will execute faster because labels are searched for sequentially from the first line of the program.

2. If these subroutines are modified or if other subroutines are added to the files, be sure that the formal parameters (p1, p2, ...) are not modified. Because variables are passed by reference, a modification of the formal parameter may alter the value of the actual parameter used in the calling program.

3. There are certain global variables that are used by several subroutines in the plotting file which must not be altered by the user's program. These variables (X, Y, U, V, H, W) are used to convert user units to plotter units and must remain unchanged throughout your program. The subroutines which require these global variables can be determined by looking at the 'Variables Used' section of each subroutine explanation.

4. Before using any of the subroutines, the user's program must set r0 to the select code of the 9871A.

5. Because of the nature of incremental plotting devices, successive use of iplt, fiplt and imove commands may cause noticeable round-off errors.

Although most of the conventions used in writing this manual are self-explanatory, some clarification may be helpful. A short description of each of the major subdivisions of the subroutine explanations follows:

CALLING SYNTAX:

1. cll is an actual 9825A command which calls the subroutine indicated by the name in single quotes.

2. Parameters passed to the subroutine are enclosed in parentheses and separated by commas. Optional parameters are enclosed in square braces [ ].

EXAMPLE:

The example indicates an actual 9825A instruction that could be used in a program.

VARIABLES USED:

1. This section is divided into three parts:

   a. <INPUT>

      All variables listed under input are those that are required by the subroutine. These variables can be local (p1, p2, p3, etc.) or global (any legal variable). Local variables are assigned values from the actual parameters in the order p1 = first actual parameter, p2 = second actual parameter, etc. Global variables will retain their value throughout the program and may be modified

by any statement in the program.

b. <DESTROYED>

The only variables listed here will be global variables whose values are changed by the subroutine. Any variable listed here should not be used in the main program.

c. <OUTPUT>

Variables listed under <OUTPUT> have been computed by the subroutine to be used by other subroutines. All these variables will be global.

SPECIAL CONSIDERATIONS:

Any limitations, side effects, or helpful hints are listed here.

LOADING INSTRUCTIONS

These instructions are divided into several sections depending on how you want the subroutines loaded. All subroutines require the General I/O ROM and the AP ROM, and are recorded on track 0 of the tape.

A. Instructions to load either the plotting subroutines (File 40) or the text subroutines (File 41) after your program.

1. Load your program, which uses the subroutines, into memory.
2. To load the appropriate file (40 or 41) into memory immediately after your program:
   a. Determine the line number that would come immediately after your program.
   b. Type ldf A, N where:
      A = the file number to be loaded (40 or 41) and
      N = the line number determined in step a.
   c. Press EXECUTE
3. Your program is now ready to run.

B. Instructions to load both the plotting and text subroutines (File 40 and 41) after your program.

1. Load your program, which uses the subroutines, into memory.
2. To load the plotting file (File 40) into memory immediately after your program:
   a. Determine the line number that would come immediately after your program.

b. Type ldf 40, N where:

N = the line number determined in step a.

c. Press EXECUTE

3. Load text file (File 41) into memory, overlaying the duplicate subroutines as follows:

a. Determine the line number of the "space" label. This can be computed in either of two ways. You can fetch various lines into the display until you find the label or you can add 64 to the last line number of your program.

b. Type ldf 41, M where:

M = the line number determined in step a.

c. Press EXECUTE

C. Instructions to load your program after either the plotting subroutines (File 40) or the text subroutines (File 41) with a gto branch around the subroutines. (See Note)

1. Record your program in a file to be loaded later.

a. Type rcf F where:

F = the file number to which you are recording your program

b. Press EXECUTE

2. Store a gto + N to branch around the subroutines where N is 109 if file 40 is to be loaded, or N is 83 if file 41 is to be loaded.

a. Press FETCH

b. Press EXECUTE

c. Type gto + N where N is 83 or 109

d. Press STORE

3. Load File 40 or File 41.

a. Type ldf A, 1 where:

A = 40 for plotter subroutines file or 41 for text subroutines file.

b. Press EXECUTE

4. To load your program after the subroutines:

   a. Type ldf A, N where:

      A = the file number where you stored your program in step 1.

      N = 110 if file 40 was loaded in step 3 or 84 if file 41 was loaded.

   b. Press EXECUTE

5. Your program should be ready to run.

Note: If the loading instructions described in C are used, then the user program must not contain any statements referencing absolute line numbers.

D. Instructions for deleting subroutines that are not needed.

1. Load the program and appropriate file or files of subroutines into memory. You can use the instructions in parts A, B, or C to do this.

2. Determine the first and last line numbers of the subroutines to be deleted (the first line number will always be the label and the last is usually a ret instruction). The line numbers can be determined by listing the program or by fetching memory locations into the display.

3. Beginning with the last (highest line numbered) subroutine, do the following (if you don't start with the highest line number, all successive line numbers will be altered).

   a. Type del A, B where:

      A = the first line to be deleted and

      B = the last line to be deleted

   b. Press EXECUTE

   c. Repeat step 3 until all extraneous subroutines are deleted.

INDEX OF SUBROUTINES

FILE NUMBER	SUBROUTINE NAME	PAGE
40	char	301
41	chtab	303
40,41	cspc	305
41	cvtab	307
40,41	form	309
40	fiplt	311
40	fplt	313
40	imove	315
40	iplt	317
40	move	319
40	plt	321
40	psiz	323
40,41	ptyp	325
40	scl	327
41	setlm	329
41	shtab	331
40,41	skip	333
40,41	space	335
41	svtab	337
41	tab	339
40,41	view	341
40	xaxis	343
40	yaxis	345

SUBROUTINE

DOCUMENTATION AND EXPLANATION

FILE 40 (Plot Routines)

```
0: "move":                                  move
1: wtb r0;27;65;
 int((p1-X)U/
 64);int((p1-                    p1 is horizontal coordinate in user units
 X)U);int((p2-                   p2 is vertical coordinate in user units
 Y)V/64);int((p2
 -Y)V)                           Convert from user units to plotter units
                                 with scaling factor and move to coordinates
2: ret
3: "imove":                                 imove
4: wtb r0;27;82;                 p1 is horizontal distance in user units
 int(p1U/64);                    p2 is vertical distance in user units
 int(p1U);int(p2
 V/64);int(p2V)                  Convert from user units to plotter units and
5: ret                           move distance specified
6: "plt":                                   plt
7: wtb r0;27;65;
 int((p1-X)U/                    p1 is horizontal coordinate in user units
 64);int((p1-                    p2 is vertical coordinate in user units
 X)U);int((p2-
 Y)V/64);int((p2                 Convert from user units to plotter units and
 -Y)V)                           move to coordinates
8: if p3=0;46→p3                 Default value for character is decimal point = 46
9: if p3=46;wtb
 r0;27;82;0;0;0;                 If decimal point, shift to center the character
 6
10: wtb r0,p3;                   Print specified character
 wtb r0;8
11: if p3=46;                    If decimal point, shift back
 wtb r0;27;82;0;
 0;63;-6
12: ret
13: "fplt":                                 fplt
14: wtb r0;27;
 97;int((p1-X)U/                 p1 is horizontal coordinate in user units
 64);int((p1-                    p2 is vertical coordinate in user units
 X)U);int((p2-
 Y)V/64);int((p2                 Convert from user units to plotter units
 -Y)V)                           and move to coordinates printing fill character
15: if p3=0;46→p                 Default value for character is decimal point = 46
 3
16: if p3=46;                    If decimal point, shift to center the character
 wtb r0;27;82;0;
 0;0;6
17: wtb r0,p3;                   Print the specified character
 wtb r0;8
18: if p3=46;
 wtb r0;27;82;0;                 If decimal point, shift back
 0;63;-6
19: ret
```

ANNOTATED LISTING:

```
20: "iplt":                          iplt
21: wtb r0,27,
 82,int(p1U/64),
 int(p1U),int(p2
 V/64),int(p2V)                      pl is hoizontal distance moved in user units
                                     p2 is vertical distance moved in user units
                                     Convert units and move the specified distance
22: if p3=0;46→p                     Default value for character is decimal point = 46
 3
23: if p3=46;                        If decimal point, shift to center the character
 wtb r0,27,82,0,
 0,0,6
24: wtb r0,p3;                       Print the specified character
 wtb r0,8
25: if p3=46;                        If decimal point, shift back
 wtb r0,27,82,0,
 0,63,-6
26: ret
```

```
27: "fiplt":                         fplt
28: wtb r0,27,
 114,int(p1U/
 64),int(p1U),                       pl is horizontal distance moved in user units
 int(p2V/64),                        p2 is vertical distance moved in user units
 int(p2V)                            Convert from user to plotter units and move
                                     the specified distance
29: if p3=0;46→p                     Default value for character is decimal point = 46
 3
30: if p3=46;                        If decimal point, shift to center the character
 wtb r0,27,82,0,
 0,0,6
31: wtb r0,p3;                       Print the specified character
 wtb r0,8
32: if p3=46;                        If decimal point, shift back
 wtb r0,27,82,0,
 0,63,-6
33: ret
```

```
34: "char":                          char
35: if p2=0;5→p2                     Default values for spacing and vertical offset
 ;0→p3                               are 5 and 0 respectively
36: wtb r0,27,
 46,p1,int(p2/
 64),p2,p3                           Specify the fill character
37: ret
```

```
38: "psiz":                          psiz
39: p1→H;p2→W                        Height (H) and width (W) are saved for later
40: wtb r0,27,                       use
 79,int(p4*120/                      Convert from user units to plotter units
 64),p4*120,int(                     and specify the plot size
 p3*96/64),p3*96                     p3 and p4 are lower left margin offsets
41: ret
```

```
42: "scl":                           scl
43: 120W/(p2-                        U is the horizontal scale factor
 p1)→U
44: 96H/(p4-p3)→                     V is the vertical scale factor
 V
45: p1→X;p3→Y                        Save the minimum X and Y values for later
46: ret                              use with plotting commands
```

ANNOTATED LISTING:

```
47: "xaxis":                    xaxis
48: wtb r0,27,                  "-" becomes the fill character
 46,95,0,5,9
49: if p3=0 and                 Default values for bounds are minimum and
 p4=0;X→p3;X+                   maximum values on xaxis
 120W/U→p4
50: if p2=0;p4-                 Default value for tic interval is (max - min)
 p3→p2
51: wtb r0,27,
 65,int((p3-X)U/                Move to minimum value on xaxis
 64),int((p3-
 X)U),int((p1-
 Y)V/64),int((p1
 -Y)V)
52: p3→p5;wtb                   p5 is tic interval counter
 r0,43;wtb r0,8                 Print a "+" and backspace
53: wtb r0,27,
 114,int(p2U/                   Fill plot to the next tic
 64),p2U,0,0;
 wtb r0,43;wtb                  Print a "+" and backspace
 r0,8;jmp (p5+                  Increment tic counter and test for end of
 p2→p5)>=p4                     xaxis
54: ret
```

```
55: "yaxis":                    yaxis
56: wtb r0,27,
 46,124,0,3,0                   "|" becomes the fill character
57: if p3=0 and                 Default values for bounds are the minimum
 p4=0;Y→p3;Y+                   and maximum values on the yaxis
 96H/V→p4
58: if p2=0;p4-                 Default value for tic interval is (max - min)
 p3→p2
59: wtb r0,27,
 65,int((p1-X)U/
 64),int((p1-                   Move to minimum value on yaxis
 X)U),int((p3-
 Y)V/64),int((p3
 -Y)V)
60: p3→p5;wtb                   p5 is tic interval counter
 r0,43;wtb r0,8                 Print a "+" and backspace
61: wtb r0,27,
 114,0,0,int(p2V                Fill plot to the next tic
 /64),p2V;wtb
 r0,43;wtb r0,8;                Print a "+" and backspace
 jmp (p5+p2→p5)>                Increment tic counter and test for end of
 =p4                            yaxis
62: ret
```

ANNOTATED LISTING:

```
63: "space":                          space
64: if p1<0;gto +2                    If negative branch to backspace
65: wtb r0,32;                        Print a blank
 jmp 2((p1-1→p1)=0)                   Decrement number of spaces and test for
                                      stopping condition
66: wtb r0,8;                         Backspace
 jmp (p1+1→p1)=0                      Increment number of spaces and test for
67: ret                               stopping conditions
68: "skip":                           skip
69: if p1<0;gto +2                    If negative branch to reverse line feed
70: wtb r0,10;                        Line feed
 jmp 2((p1-1→p1)=0)                   Decrement number of line feeds and test for
                                      stopping conditions
71: wtb r0,27,10;jmp (p1+1→p1)=0      Reverse line feed
                                      Increment number of reverse line feeds and
                                      test for stopping conditions
72: ret
73: "cspc":                           cspc
74: if p2=0;6→p2                      Default value for lines per inch is 6
75: wtb r0,27,86,int(96/p2/64),96/p2  Convert to plotter units and specify lines per
                                      inch
76: if p1<0;wtb r0,27,80;ret          If horizontal spacing is negative, implement
                                      proportional spacing
77: wtb r0,27,72,int(120/p1/64),120/p1 Convert to plotter units and specify number of
                                      characters printed per inch
78: ret
79: "form":                           form
80: wtb r0,27,77                      Set left margin
81: wtb r0,27,84                      Set top of form
82: if p1=0;13.2→p1;11→p2→p3          Default values are text width = 13.2 and
                                      text length and form length = 11
83: wtb r0,27,87,int(120*p1/64),120*p1 Convert to user units and specify text width
84: wtb r0,27,76,int(96*p2/64),96*p2  Convert to user units and specify text length
85: wtb r0,27,70,int(96*p3/64),96*p3  Convert to user units and specify form length
86: ret
```

ANNOTATED LISTING:

```
87: "ptyp";                          ptyp
88: rdb(0)→I;if                      Read in the character code form keyboard
 I=1;ret                             Check if STOP has been pressed
89: if I<124;                        If code less than 124 branch to line 101
 gto +12
90: if I>224
 and I<251;I-
 160→I;gto +15
91: if I>176
 and I<182;I-
 144→I;gto +14
92: if I=182;
 38→I;gto +13
93: if I=183;                        These lines convert 9825A decimal code to
 64→I;gto +12                        the 9871A plotting code
94: if I=184;
 91→I;gto +11
95: if I=185;
 93→I;gto +10
96: if I=222;
 92→I;gto +9
97: if I=251;
 124→I;gto +8
98: if I=191;
 58→I;gto +7
99: if I=176;
 39→I;gto +6
100: if I=174
 or I=172;I-112→
 I;gto +5
101: if I>77
 and I<88;I-30→I
 ;gto +4
102: if I=88;
 46→I;gto +3
103: if I=94;
 wtb r0,14
104: if I=89;
 44→I
105: wtb r0,I;                       Print the character and go back for next
 gto -17
106: "view";
107: wtb r0,27,                      p1 is number of milliseconds before the
 68,int(p1/64),                      platen is advanced for easy viewing
 p1
108: ret
*1083
```

FILE 41 (Text Routines)

Lines 0 to 45 are the same subroutines as lines 63 to 108 of FILE 40.

```
46: "tab":
47: if p2=0;gto                          tab
 +4                                      If vertical tab omitted skip vertical tab
48: if p2<0;gto                          section
 +2                                      If negative vertical tab go to line 50
49: wtb r0;11;
 jmp 2((p2-1→p2)                         Vertical tab down
 =0)                                     Decrement tab counter and check stopping
50: wtb r0;27;                           conditions
 56;jmp (p2+1→p2                         Vertical tab up
 )=0                                     Increment tab counter and check stopping
51: if p1=0;1→p1                         conditions
52: if p1<0;gto                          Default for horizontal tab is one
 +2                                      If negative horizontal tab go to line 54
53: wtb r0;9;
 jmp 2((p1-1→p1)                         Horizontal tab right
 =0)                                     Decrement tab counter and check stopping
54: wtb r0;27;                           conditions
 52;jmp (p1+1→p1                         Horizontal tab left
 )=0                                     Increment counter and check stopping
55: ret                                  conditions
```

```
56: "shtab":
57: 0→I;wtb r0;                          shtab
 13                                      I is counter for number of tabs set
58: if p(I+1→I)=                         Execute carrier return
 0;ret                                   Increment the parameter number and test for
59: wtb r0;27;                           the last parameter
 82;int(120pI/
 64);120pI;0;0                           Move to the position specified for the Ith
60: wtb r0;27;49                         parameter
61: wtb r0;13;                           Set the horizontal tab
 wtb r0;10;gto -
 3                                       Carrier return and branch back to get next
                                         parameter
```

ANNOTATED LISTING:

```
62: "chtab":
63: 0→I;if p1<0;
 wtb r0,27,51;
 ret
64: wtb r0,13
65: if p(I+1→I)=
 0;ret
66: wtb r0,27,
 82,int(120pI/
 64),120pI,0,0
67: wtb r0,27,50
68: wtb r0,13;
 wtb r0,10;gto -
 3
```

chtab

I is counter for number of tabs cleared
If first parameter is negative, clear all tabs

Carrier return
Increment the parameter number and test for the last parameter

Move to the position specified by the I^{th} parameter
Clear the horizontal tab
Carrier return and branch back to get next tab

```
69: "svtab":
70: 0→I;1→J;wtb
 r0,12
71: if p(I+1→I)=
 0;ret
72: wtb r0,10;
 jmp (J+1→J)>pI
73: wtb r0,27,53
74: wtb r0,27,
 10;jmp -3((J-
 1→J)=1)
```

svtab

I is counter for number of tabs set
Form feed
Increment the parameter number and test for the last parameter
Line feed
Continue line feeds until parameter size exceeded
Set vertical tab
Reverse line feed
Continue reverse line feed until you get back to top of form

```
75: "cvtab":
76: 0→I;1→J;wtb
 r0,12
77: if p1<0;wtb
 r0,27,55;ret
78: if p(I+1→I)=
 0;ret
79: wtb r0,10;
 jmp (J+1→J)>pI
80: wtb r0,27,54
81: wtb r0,27,
 10;jmp -3((J-
 1→J)=1)
```

cvtab

I is counter for number of tabs cleared
Form feed
If first parameter negative, clear all vertical tabs
Increment parameter number and test for the last parameter
Line feed
Continue line feeds until parameter size exceeded
Clear horizontal parameter
Reverse line feed
Continue reverse line feeds until you get back to top of form

```
82: "setlm":
83: wtb r0,13;
 wtb r0,27,82,
 int(120p1/64),
 int(120p1),0,0
84: wtb r0,27,
 77;wtb r0,13
85: ret
*3651
```

setlm

Carrier return and incremental move to correct position

Set left margin
Carrier return

char

CALLING SYNTAX:

cll 'char' (character[, interval, offset])

character	Decimal code for character to be used as fill in fplt and fiplt
interval	Number of 1/120" increments between characters (optional)
offset	Number of 1/96" vertical increments that character is offset (optional)

EXAMPLE: cll 'char' (46, 3, 6)

Execution of this subroutine call sets up the conditions for plotting with character fill. The first parameter (character) specifies the character that will be printed, interval specifies the number of 1/120" increments between characters, and offset specifies the number of 1/96" increments that the character is offset vertically. The interval and offset parameters are optional and will default to 5 and 0 respectively.

VARIABLES USED:

<INPUT>

Local:

p1	Decimal code for character
p2	Interval size in 1/120" increments (default = 5)

p3 Vertical offset in 1/96" increments (default = 0)

Global:

r0 Select code for the 9871A

SPECIAL CONSIDERATIONS:

1. If the decimal point (46) is used as a fill character, an offset of 6 units seems to be most accurate.

2. This subroutine should be executed after an 'xaxis' or 'yaxis' subroutine call since these routines will destroy the current character fill.

chtab

CALLING SYNTAX:

cll 'chtab' ($htab_1$, $htab_2$, ..., $htab_n$)

$htab_i$	The distance in inches from the left margin to the position of the horizontal tab to be cleared (if $htab_1 < 0$ all tabs are cleared)
n	The number of tabs to be cleared

EXAMPLE: cll 'chtab' (2, 6, 8)

Execution of this subroutine will clear horizontal tabs at all positions specified by the parameters $htab_i$ ($1 \leq i \leq n$). The parameters indicate the distance in inches from the left margin to the tab. If the first parameter is negative, then all tabs are cleared (cll 'chtab' (-1)).

Additional tabs can be set by executing the 'shtab' subroutine.

VARIABLES USED:

<INPUT>

Local:

p1	First horizontal tab to be cleared - distance in inches from left margin. If p1 < 0 then all horizontal tabs are cleared
p2	Second horizontal tab to be cleared
.....	.....
pn	n^{th} horizontal tab to be cleared

Global:

r0 Select code for the 9871A

<DESTROYED>

Global:

I Used as index for parameter number

SPECIAL CONSIDERATIONS:

1. Tab parameters may be listed in any order in the parameter list.

2. Power up default conditions will clear all horizontal tabs.

cspc

CALLING SYNTAX:

cll 'cspc' (CPI[, LPI])

CPI	Character per inch (CPI < 0 implements proportional spacing)
LPI	Lines per inch

EXAMPLE: cll 'cspc' (6, 10)

Execution of this subroutine will specify the number of characters printed per inch (CPI) and the number of lines printed per inch (LPI). If the first parameter (CPI) is negative, then characters will be spaced in proportion to their size. The second parameter (LPI) is optional and has default value 6.

VARIABLES USED:

<INPUT>

Local:

p1	Number of characters printed per inch. If negative, then proportional spacing is used
p2	Number of lines per inch (default = 6)

Global:

r0	Select code for the 9871A

SPECIAL CONSIDERATION:

The power up default conditions will print 10 characters per inch and 6 lines per inch.

cvtab

CALLING SYNTAX:

cll 'cvtab' ($vtab_1$, $vtab_2$, ..., $vtab_n$)

$vtab_i$	The distance in lines from the top of form to the vertical tab to be cleared (if $vtab_1 < 0$ all tabs are cleared)
n	The number of tabs to be cleared

EXAMPLE: cll 'cvtab' (2, 6, 8)

Execution of this subroutine will clear vertical tabs at all positions specified by the parameters $vtab_i$ ($1 \leq i \leq n$). The parameters indicate the number of lines from the top of form to the tab. If the first parameter is negative, then all tabs are cleared (cll 'cvtab' (-1)).

Additional tabs can be set by executing the 'svtab' subroutine.

VARIABLES USED:

<INPUT>

Local:

p1	First vertical tab to be cleared; number of lines from top of form. If p1 < 0 then <u>all</u> vertical tabs are cleared
p2	Second vertical tab to be cleared
.....	
pn	n^{th} vertical tab to be cleared

Global:

r0	Select code for the 9871A

<DESTROYED>

Global:

I	Used as index for parameter number
J	Used to count line feeds

SPECIAL CONSIDERATIONS:

1. Tab parameters may be listed in any order in the parameter list.

2. Power up default conditions will clear all vertical tabs.

form

CALLING SYNTAX:

```
cll 'form' [(W, L, F)]
```

W	Text width
L	Text length
F	Form length

EXAMPLE: `cll 'form' (10, 10, 11)`

Execution of this subroutine call sets the top of form and the left margin at the present printer head position. The user may specify the text width (W), text length (L), and form length (F) in inches. The parameters (W, L, F) are optional with default values of W = 13.2 inches, L = 11 inches, and F = 11 inches.

VARIABLES USED:

<INPUT>

Local:

p1	Text width in inches
p2	Text length in inches
p3	Form length in inches

Global:

r0	Select code for the 9871A

SPECIAL CONSIDERATIONS:

1. The maximum text width is 13.2 inches.

2. The text length should be less than or equal to the form length.

3. If this subroutine call is not executed, default conditions will prevail. These values are:

   text width = 13.2 inches
   text length = 11 inches
   form length = 11 inches.

fiplt

CALLING SYNTAX:

cll 'fiplt' (± X-distance, ± Y-distance[, character])

X-distance	An expression for the horizontal distance to be moved
Y-distance	An expression for the vertical distance to be moved
character	Decimal code for character to be printed (optional) Character codes are found in Appendix B

EXAMPLE cll 'fiplt' (0, -1)

Execution of this subroutine will move the printer to a relative position, X horizontal and Y vertical units from the present position and plot the user specified character.

The difference between this command and the 'iplt' routine is that the fill character specified in the 'char' routine will be printed along a straight line from the current (X,Y) position to the point to be plotted.

The X and Y distances are expressed in user units and can be constants, variables, or expressions. The position moved to should fall inside the range of the plot scale.

Specifying the character to be printed is optional and has the decimal point,".",as a default value.

VARIABLES USED:

<INPUT>

Local:

p1	Horizontal distance moved in user units
p2	Vertical distance moved in user units
p3	Decimal code for character to be plotted (default = 46)

Global:

U	Scale factor for horizontal units
V	Scale factor for vertical units
r0	Select code for the 9871A

SPECIAL CONSIDERATIONS:

1. Special care should be taken to assure that the plotting program does not destroy the global variables U and V.

2. The decimal codes for characters that can be plotted are found in Appendix B of this manual. Only those characters listed in the Appendix can be printed.

3. The fill character, if not specified in 'char' will default to the decimal point (46), with horizontal spacing of 1/40" (3 horizontal plot units) and vertical offset of 1/16" (6 vertical plot units). When using the decimal point as a fill character, a vertical offset of 1/16" seems to achieve the best results.

fplt

CALLING SYNTAX:

cll 'fplt' (X-coordinate, Y-coordinate[, character])

X-coordinate	An expression for the X coordinate of the point to be plotted
Y-coordinate	An expression for the Y coordinate of the point to be plotted
character	Decimal code of character to be plotted (optional) Character codes are found in Appendix B

EXAMPLE: cll 'fplt' (2.5, 1)

Execution of this subroutine call will plot the user specified character at position (X,Y) where X is the horizontal and Y the vertical distance from the origin (0,0).

The difference between this command and the 'plt' routine is that the fill character specified in the 'char' routine will be printed along a straight line from the current (X,Y) position to the point being plotted.

The X and Y coordinates are expressed in user units and may be constants, variables, or expressions, but should always fall in the range of the plot scale.

Specifying the character to be printed is optional and has the decimal point, ".", as a default value.

VARIABLES USED:

<INPUT>

Local:

p1	X coordinate in user units
p2	Y coordinate in user units
p3	Decimal code for character to be plotted (default = 46)

Global:

X	Absolute minimum value on X axis as specified in the 'scl' routine
Y	Absolute minimum value on Y axis as specified in the 'scl' routine
U	Scale factor for horizontal units
V	Scale factor for vertical units
r0	Select code for the 9871A

SPECIAL CONSIDERATIONS:

1. Special care should be taken to assure that the plotting program does not destroy the global variables X, Y, U, and Z.

2. The decimal codes for characters that can be plotted are in Appendix B of this manual. Only those characters listed in the Appendix can be printed.

3. The fill character, if not specified in 'char', will default to the decimal point (46), with horizontal spacing of 1/40" (3 horizontal plot units) and vertical offset of 1/16" (6 vertical plot units). When using the decimal point as a fill character, a vertical offset of 1/16" seems to achieve the best results.

imove

CALLING SYNTAX:

cll 'imove' (± X-distance, ± Y-distance)

X-distance	An expression for the horizontal distance to be moved
Y-distance	An expression for the vertical distance to be moved

EXAMPLE: cll 'imove' (0, -1)

Execution of this subroutine will move the printer to a relative position which is X horizontal and Y vertical units from the present position of the printer. The X and Y distances are expressed in user units and can be constants, variables or expressions. The position moved to should fall inside the range of the plot scale.

VARIABLES USED:

<INPUT>

Local:

p1	Horizontal distance moved in user units
p2	Vertical distance moved in user units

Global:

U	Scale factor for horizontal units
V	Scale factor for vertical units
r0	Select code for the 9871A

SPECIAL CONSIDERATION:

Special care should be taken to assure that the plotting program does not destroy the global variables U, V.

iplt

CALLING SYNTAX:

cll 'iplt' (± X-distance, ± Y-distance[, character])

X-distance	An expression for the horizontal distance to be moved
Y-distance	An expression of the vertical distance to be moved
character	Decimal code for character to be printed (optional) Character codes can be found in Appendix B

EXAMPLE: cll 'iplt' (0, -1, 43)

Execution of this subroutine will move the printer to a relative position, X horizontal and Y vertical units from the present position and plot the user specified character. The X and Y distances are expressed in user units and can be constants, variables, or expressions. The position moved to should fall inside the range of the plot scale.

Specifying the character to be printed is optional and has a decimal point, ".", as the default value.

VARIABLES USED:

<INPUT>

Local:

p1	Horizontal distance moved in user units
p2	Vertical distance moved in user units

p3 Decimal code for character to be plotted (default = 46)

Global:

U Scale factor for horizontal units

V Scale factor for vertical units

r0 Select code for the 9871A

SPECIAL CONSIDERATIONS:

1. Special care should be taken to assure that the plotting program does not destroy the global variables U and V.

2. The decimal codes for characters that can be plotted can be found in Appendix B of this manual. Only those characters listed in the Appendix can be printed.

move

CALLING SYNTAX:

```
call 'move' (X-coordinate, Y-coordinate)
```

X-coordinate	An expression for the X-coordinate of the point to be moved to
Y-coordinate	An expression for the Y-coordinate of the point to be moved to

EXAMPLE:

```
call 'move' (1, 1)
```

Execution of this subroutine will move the printer to position (X, Y) where X is the horizontal and Y the vertical distance from the origin (0, 0). The X- and Y-coordinates are expressed in user units and can be constants, variables, or expressions but should always fall in the range of the plot scale.

VARIABLES USED:

<INPUT>

Local:

p1	X-coordinate in user units
p2	Y-coordinate in user units

Global:

X	Absolute minimum value on X axis as specified in the 'scl' routine

Y	Absolute minimum value on Y axis as specified in the 'scl' routine
U	Scale factor for horizontal units (from 'scl')
V	Scale factor for vertical units (from 'scl')
r0	Select code for the 9871A

SPECIAL CONSIDERATION:

Special care should be taken to assure that the plotting program does not destroy the global variables X, Y, U, V.

plt

CALLING SYNTAX:

cll 'plt' (X-coordinate, Y-coordinate[, character])

X-coordinate	An expression for the X coordinate of the point to be plotted
Y-coordinate	An expression for the Y coordinate of the point to be plotted
Character	Decimal code of character to be plotted (optional) (Characters codes are listed in Appendix B).

EXAMPLE: cll 'plt' (Z, sin(Z), 42)

Execution of this subroutine call will plot the user specified character at position (X, Y) where X is the horizontal and Y the vertical distance from the origin (0, 0). The X and Y coordinates are expressed in user units and may be constants, variables or expressions, but should always fall in the range of the plot scale.

Specifying the character to be printed is optional. The default character is the decimal point,"." .

VARIABLES USED:

<INPUT>

Local:

pl X coordinate in user units

p2 Y coordinate in user units

p3 Decimal code for character to be plotted (default = 46)

Global:

X Absolute minimum value on X axis as specified in the 'scl' routine

Y Absolute minimum value on Y axis as specified in the 'scl' routine

U Scale factor for horizontal units (from 'scl')

V Scale factor for vertical units (from 'scl')

r0 Select code for the 9871A printer

SPECIAL CONSIDERATIONS:

1. The decimal codes for characters that can be plotted are found in Appendix B of this manual. Only those characters listed in the Appendix can be printed on the 9871A.

2. Special care should be taken to assure that the plotting program does not destroy the global variables X, Y, U, V.

psiz

CALLING SYNTAX:

cll 'psiz' (Vs, Hs, Vo, Ho)

Vs	Vertical size of plot (height) in inches
Hs	Horizontal size of plot (width) in inches
Vo	Vertical offset from bottom margin in inches
Ho	Horizontal offset from left margin in inches

EXAMPLE: cll 'psiz' (10, 10, 1, 1)

Execution of this subroutine call sets the plot size by specifying the height and width in inches and determines the location of the plot on the paper by specifying a horizontal and vertical offset in inches from the lower left corner of the margins.

The 'scl' subroutine call must be executed immediately after this subroutine call to assure proper computation of the scale factor for user units.

VARIABLES USED:

<INPUT>

Local:

p1	Height of plot in inches
p2	Width of plot in inches

p3 Horizontal offset from left margin in inches

p4 Vertical offset from bottom margin in inches

Global:

r0 Select code for the 9871A

<OUTPUT>

Global:

H Height of plot in inches (same as p1)

W Width of plot in inches (same as p2)

SPECIAL CONSIDERATIONS:

1. The 'form' subroutine should be executed before executing the 'psiz' subroutine.

2. The 'scl' subroutine must be executed immediately after the 'psiz' subroutine to avoid destroying the value of the global variables H and W which are required by the 'scl' routine.

ptyp

CALLING SYNTAX:

cll 'ptyp'

EXAMPLE:

cll 'ptyp'

Execution of this subroutine places the calculator in a mode which causes each character typed to be printed immediately on the 9871A. To exit the mode you simply press the STOP key.

VARIABLES USED:

<INPUT>

Global:

r0 Select code for the 9871A

<DESTROYED>

I Used as temporary storage for decimal code

SPECIAL CONSIDERATION:

Only the upper and lower case alphanumeric keys and the numeric keys will cause the appropriate characters to be printed. Special keys and command keys will not be printed.

scl

CALLING SYNTAX:

cll 'scl' (X_{min}, X_{max}, Y_{min}, Y_{max})

X_{min}	Minimum X value (user units)
X_{max}	Maximum X value (user units)
Y_{min}	Minimum Y value (user units)
Y_{max}	Maximum Y value (user units)

EXAMPLE: cll 'scl' (-1, 10, -5, 5)

Execution of this subroutine call establishes the scale values, in user units, of the X and Y axes. X_{min} to X_{max} and Y_{min} to Y_{max} correspond exactly to the respective limits of the horizontal and vertical edges of the plotting area. This also establishes where the plot origin (coordinates (0,0)) is located, whether on or off the plotting area.

This subroutine call must be executed before any plotting subroutine calls can be executed.

VARIABLES USED:

<INPUT>

Local:

p1	Minimum X value in user units
p2	Maximum X value in user units

p3 Minimum Y value in user units

p4 Maximum Y value in user units

Global:

H Height (in inches) of plot

W Width (in inches) of plot

r0 Select code for the 9871A

<OUTPUT>

Global:

X Minimum X value

Y Minimum Y value

U Horizontal scale factor to convert user units to plot units

V Vertical scale factor to convert user units to plot units

SPECIAL CONSIDERATIONS:

1. This subroutine call must be executed before any plotting is done.

2. A 'psiz' subroutine execution should always immediately precede execution of this subroutine.

setlm

CALLING SYNTAX:

cll 'setlm' (L)

L Width of left margin in inches

EXAMPLE: cll 'setlm' (1)

Execution of this subroutine will set the size of the left margin in inches. The default value for the left margin will be the leftmost side of the page.

VARIABLES USED:

<INPUT>

Local:

pl Width of left margin in inches

Global:

rØ Select code for 9871A

SPECIAL CONSIDERATION:

The psize command which determines the size of the plotting area will also set the left margin.

shtab

CALLING SYNTAX:

cll 'shtab' ($htab_1$, $htab_2$, $htab_3$, ..., $htab_n$)

$htab_i$	The distance in inches from the left margin to the position of the horizontal tab
n	The number of tabs being set

EXAMPLE: cll 'shtab' (2, 4, 6, 8, 10)

Execution of this subroutine will set horizontal tabs at all positions specified by the parameters $htab_i$ ($1 \leq i \leq n$). The parameters designate the distance in inches from left margin to the tab.

Subsequent changes in the left margin or character spacing will not alter the position(s) of the tab(s).

Horizontal tabs can be cleared by executing the 'chtab' subroutine.

VARIABLES USED:

<INPUT>

Local:

p1	First horizontal tab - distance in inches from left margin
p2	Second horizontal tab
.....	
pn	n^{th} horizontal tab

Global:

r0 Select code for the 9871A

<DESTROYED>

Global:

I Used as index for parameter number

SPECIAL CONSIDERATIONS:

1. Tab parameters may be listed in any order in the parameter list.

2. Power up default conditions clear all horizontal tabs.

3. The number of tabs is restricted only by the maximum line length of the 9825A calculator.

skip

CALLING SYNTAX:

cll 'skip' (N)

N Number of lines skipped by either normal line feeds or reverse line feeds

EXAMPLE: cll 'skip' (-2)

Execution of this subroutine will cause the printer to skip N lines if N is positive or reverse line feed N times if N is negative.

VARIABLES USED:

<INPUT>

Local:

p1 The number of lines skipped. Positive p1 will cause normal line feeds and negative p1 will cause reverse line feeds

p2 Counter for number of lines

Global:

r0 Select code for the 9871A

SPECIAL CONSIDERATION:

Modification of the number of lines printed per inch by executing the 'cpsc' routine will also affect the distance moved with this command.

space

CALLING SYNTAX:

cll 'space' (N)

N Number of spaces moved either forward or backward

EXAMPLE: cll 'space' (3)

Execution of this subroutine will cause the printer to move N spaces to the right if N is positive or backspace N times if N is negative. Skipping spaces and printing blanks move the printer the same distance.

VARIABLES USED:

<INPUT>

Local:

p1 The number of spaces moved. Positive p1 will move to the right and negative p1 will move to the left.

p2 Counter for number of spaces

Global:

r0 Select code for the 9871A

SPECIAL CONSIDERATION:

Modification of character spacing by executing the 'cspc' routine will also affect the distance moved with this command.

svtab

CALLING SYNTAX:

cll 'svtab' ($vtab_1$, $vtab_2$, ..., $vtab_n$)

$vtab_i$	The number of lines from the top of form to the position of the tab
n	The number of tabs being set

EXAMPLE: cll 'svtab' (6, 12, 18)

Execution of this subroutine will set vertical tabs at all positions specified by $vtab_i$ ($1 \leq i \leq n$). The parameters designate the number of lines from the top of form to the tab. Subsequent changes in the location of top of form will not alter the relative positions of the tabs.

Vertical tabs can be cleared by executing the 'cvtab' subroutine.

VARIABLES USED:

<INPUT>

Local:

p1	First vertical tab; number of lines from top of form
p2	Second vertical tab
.....	.....
pn	n^{th} vertical tab

Global:

r0	Select code for the 9871A

<DESTROYED>

Global:

I	Used as an index for parameter number
J	Used as counter for number of line feeds

SPECIAL CONSIDERATIONS:

1. Tab parameters may be listed in any order in the parameter list.

2. Any number of tabs can be set up to a maximum of one per line.

3. Power up default conditions clear all vertical tabs.

tab

CALLING SYNTAX:

cll 'tab' [(± htab[, ± vtab])]

htab — Tab horizontally htab times (default = 1)

vtab — Tab vertically vtab times (optional)

EXAMPLE: cll 'tab' (+2, -1)

Execution of this subroutine will tab horizontally and/or vertically the indicated number of times. Tabs can be specified in a positive or negative direction. If a tab is executed beyond the last tab that was set, the printer head will move to the appropriate margin.

VARIABLES USED:

<INPUT>

Local:

p1 — Number of horizontal tabs. May be positive or negative. If omitted the default is one tab to the right.

p2 — Number of vertical tabs. May be positive or negative. If omitted the default is zero.

p3 — Counter for horizontal tabs

p4 — Counter for vertical tabs

Global:

r0 Select code for the 9871A

SPECIAL CONSIDERATIONS:

1. When using the wrt command, an automatic carriage return/line feed will be generated unless a z format specification is encountered. This should be taken into account when tabbing.

2. Vertical tabbing with no horizontal tabs can be implemented by specifying 0 horizontal tabs.

view

CALLING SYNTAX:

cll 'view' (time)

time — Number of milliseconds before paper is moved up to view what was printed

EXAMPLE: cll 'view' (500)

Execution of this subroutine specifies the number of milliseconds after the last character is printed until the platen advances to allow easy viewing of that character. The platen automatically retracts to its original position before printing the next character.

If the optional parameter is omitted, then the timer will be set to a default of 0 seconds.

VARIABLES USED:

<INPUT>

Local:

pl — Number of milliseconds before platen advances (default = 0)

Global:

r0 — Select code for the 9871A

SPECIAL CONSIDERATION:

If this subroutine is not executed the power up default condition is -200 msec.

xaxis

CALLING SYNTAX:

cll 'xaxis' (Y_{off} [,+tic [,X_{min},X_{max}]])

Y_{off}	Point on Y axis where X axis intersects
tic	Positive increment between tic marks
X_{min}	Position where X axis printing begins
X_{max}	Position where X axis printing ends

EXAMPLE: cll 'xaxis' (0, 1, -10, 10)

Execution of this subroutine call will print an X axis from X_{min} to X_{max} with tic marks printed at the user-defined intervals (tic). The X axis will intersect the Y axis at the point, Y_{off}, which is also specified by the user.

If the optoinal parameters X_{min} and X_{max} are omitted, then the default values are the minimum and maximum X values specified in the 'scl' subroutine call. If the optional parameter tic is also omitted, then the default value will only print tic marks at the beginning and end of the axis.

VARIABLES USED:

<INPUT>

Local:

pl	Y axis point at which X axis will intersect

p2	Interval spacing between tic marks
p3	Beginning point for printing of X axis
p4	End point where printing of X axis stops
p5	Used as increment in loop to print axis tic marks

Global:

W	Width of plot (from 'psiz')
H	Heigth of plot (from 'psiz')
U	Scale factor for horizontal units (from 'scl')
V	Scale factor for vertical units (from 'scl')
X	Minimum value on X axis scale (from 'scl')
Y	Minimum value on Y axis scale (from 'scl')
r0	Select code for the 9871A

SPECIAL CONSIDERATIONS:

1. The X axis must always be printed in the positive direction which means that $X_{min} < X_{max}$ and tic > 0.

2. Execution of this subroutine will cause any fill character specified in a 'char' subroutine call to be destroyed.

yaxis

CALLING SYNTAX:

cll 'yaxis' (X_{off} [, + tic [, Y_{min}, Y_{max}]])

X_{off}	Point on X axis where Y axis will intersect
tic	Positive increment between tic marks
Y_{min}	Position where Y axis printing begins
Y_{max}	Position where Y axis printing ends

EXAMPLE: cll 'yaxis' (0, .5, -5, 5)

Execution of this subroutine call will print a Y axis from Y_{min} to Y_{max} with tic marks printed at the user specified intervals (tic). The Y axis will intersect the X axis at the point, X_{off}.

If the optional parameters Y_{min} and Y_{max} are omitted, then the default values are the minimum and maximum Y values as specified in the 'scl' subroutine call. If the optional parameter tic is also omitted, then the default value will only print tic marks at the beginning and end points of the axis.

VARIABLES USED:

<INPUT>

Local:

pl	X axis offset through which Y axis will intersect

p2	Interval spacing between tic marks
p3	Beginning point for printing of Y axis
p4	End point where printing of Y axis stops
p5	Used as increment in loop to print axis tic marks

Global:

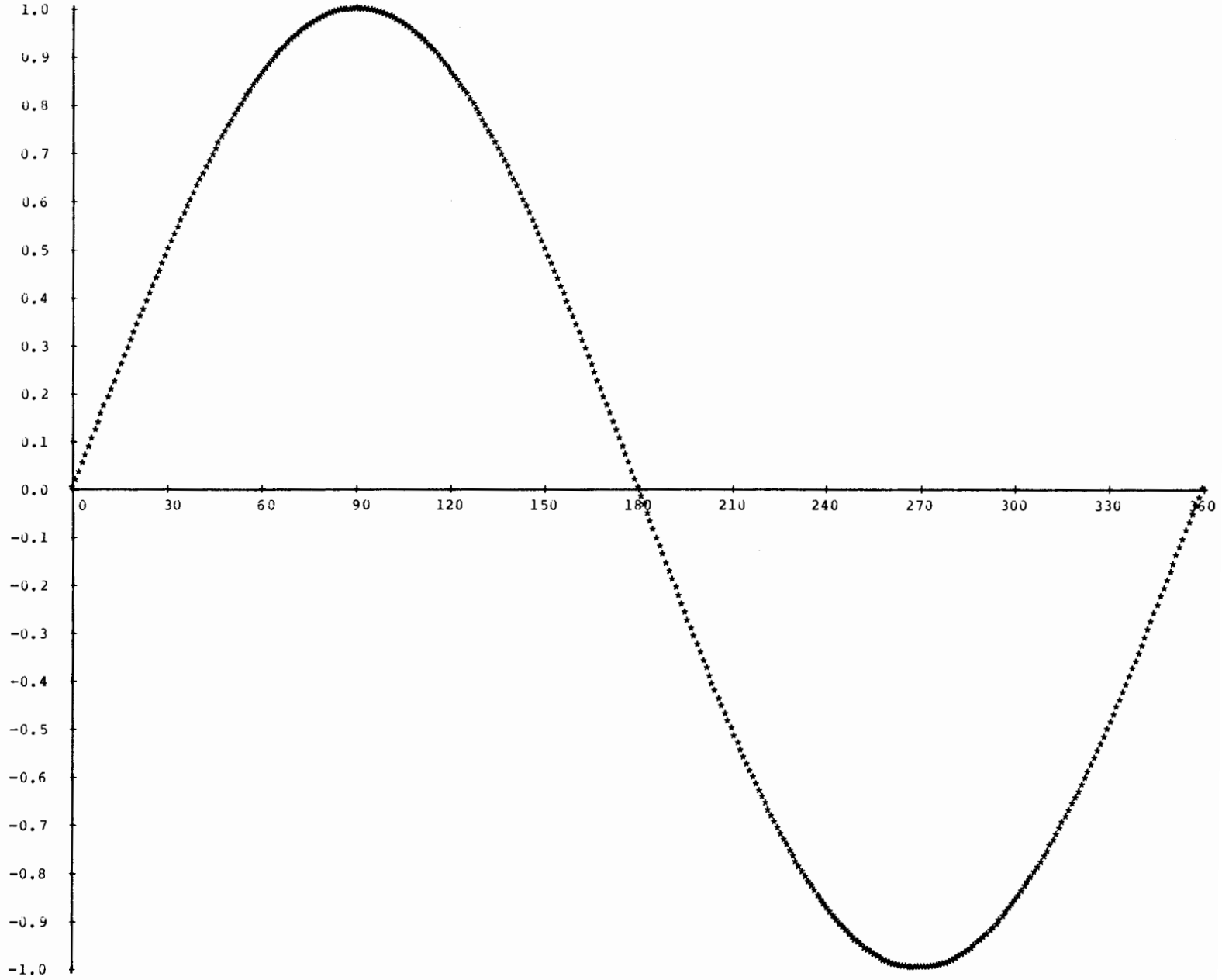
W	Width of plot (from 'psiz')
H	Height of plot (from 'psiz')
U	Scale factor for horizontal units (from 'scl')
V	Scale factor for vertical units (from 'scl')
X	Absolute minimum value on X axis scale (from 'scl')
Y	Absolute minimum value on Y axis scale (from 'scl')
r0	Select code for the 9871A

SPECIAL CONSIDERATIONS:

1. The Y axis must always be printed in the positive direction which means that $Y_{min} < Y_{max}$ and tic > 0.

2. Execution of this subroutine will cause any fill character specified in a 'char' subroutine call to be destroyed.

File 42



Reduced 35%

ANNOTATED LISTING:

PROGRAM EXAMPLE FOR 9871A PRINTER (FILE 42 SINE FUNCTION)

```
0: 6→r0;deg                    Initialize select code for 9871A;Degree measure
1: cll 'form'(13               Set up size of page
 ,12,12)
2: cll 'psiz'(12               Initialize plot size
 ,12,.5,.5)
3: cll 'scl'(-
 10,360,-1.2,                  Initialize plot scale
 1.2)
4: cll 'xaxis'(0
 ,30,0,360)                    Print x and y axis with no tic marks
5: cll 'yaxis'(0
 ,.1,-1,1)
6: 0→I;cll 'move               Move to origin and initialize domain I
 '(0,0)
7: cll 'plt'(I,
 sin(I),42)                    Plot function coordinates (I, sin(I))
8: 'if (I+1→I)<=3              over interval [0, 360]
 60;gto -1
9: 0→I;fmt f5.0,               Initialize format to label x-axis
 z
10: cll 'move'(I               Move to appropriate position on x-axis
 ,0)
11: cll 'space'(               Backspace 3 times
 -3)
12: cll 'skip'(1               Skip one line
 )
13: wrt r0,I;if
 (I+30→I)<=360;                Print the label
 gto -3
14: -1→I;fmt
 f4.1,z                        Initialize format to label y-axis
15: cll 'move'(0
 ,I)                           Move to appropriate position on y-axis
16: cll 'space'(
 -6)                           Backspace 6 times
17: wrt r0,I;if
 (I+.1→I)<=1;                  Print label
 gto -2
18: cll 'move'(1
 80,-1.05)
19: cll 'space'(               Move to bottom of plot and
 -3)
20: wrt 6,"SIN(X               label the plot
 )"
21: end
*4846
```

File 43

EMPLOYEE RECORD

EMPLOYEE	EMP #	JOB CODE	SALARY	YRS EMP	VAC DAYS
Captain Hook	654321	206	11500	32	112
Cookie Monster	222222	401	10250	6	32
Frank N Stein	987890	531	15600	43	175
Mad Hatter	100000	912	23000	31	150
Rin Tin Tin	123619	632	2500	21	211
Roadrunner	437692	592	40000	27	101
The Shadow	999999	999	41000	41	209
Tinker Bell	513964	100	17500	37	151
Winnie the Pooh	101010	236	20000	339	423
Zorro	111111	111	25000	25	234

ANNOTATED LISTING:

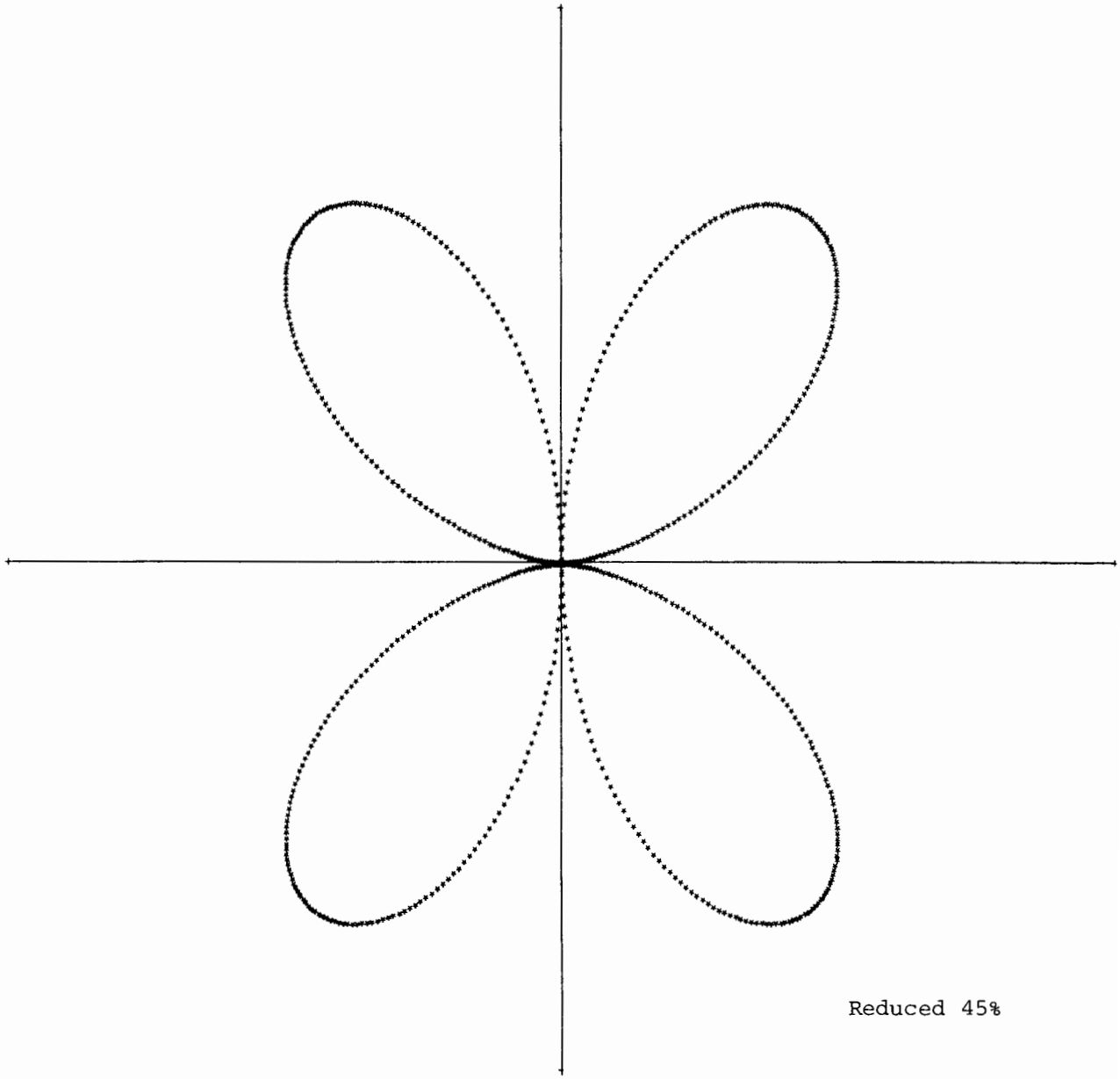
PROGRAM EXAMPLE FOR 9871A PRINTER (FILE 43 EMPLOYEE RECORDS)

```
0: enp "NO. OF                    Enter number of records in data bank
 EMPLOYEES?",N
1: dim A$[N,20],                  Initialize data bank
 A[N,6]
2: 1→I;6→K
3: ent "NAME?",                   Loop to enter elements into data bank
 A$[I,1,20];1→J
4: ent "EMPLOYEE                  Employee number
  NO.?",A[I,1]
5: ent "TITLE                     Title code for job
 CODE?",A[I,2]
6: ent "SALARY?"                  Salary in dollars
 ,A[I,3]
7: ent "NO. OF                    Number of years employed
 YEARS?",A[I,4]
8: ent "VACATION                  Number of vacation days earned
  DAYS?",A[I,5]
9: if (I+1→I)<=N
 ;gto -6
10: 6→r0;cll                      Initialize select code for 9871A
 'skip'(6)                        Skip 6 lines
11: cll 'space'(
 40);wrt r0,"                     Skip 40 spaces and print heading for data base
    EMPLOYEE
 RECORD"
12: cll 'skip'(3
 );fmt 1,f6.0,z;                  Skip 3 lines and set up format specifications
 fmt 2,c20,z
13: cll 'shtab'(
 .5,3,4.5,6,7.5,                  Set horizontal tabs at 1/2",3",4.5",6",7.5",
 9,10.5);fmt c14,z                9",10.5"
14: 1→I;cll 'tab                  Tab right and
 ';wrt 6.2,"EMPL                  Print heading for Employee column
 OYEE    ";cll                    Tab right and
 'tab';wrt 6,                     Print heading for Employee Number column
 "EMP #          "
15: cll 'tab';                    Tab right and
 wrt 6,"JOB CODE                  Print heading for Job Code Number column
       ";cll 'tab                 Tab right and
 ';wrt 6,"SALARY                  Print heading for Salary
        "
16: cll 'tab';                    Tab right and
 wrt 6,"YRS EMP                   Print heading for Number of Years Employed
     ";cll 'tab'                  Tab right and
 ;wrt 6,"VAC                      Print heading for Number of Vacation Days
 DAYS       "
17: wtb 6,13;
 cll 'skip'(2)                    Carrier return and skip two lines
18: cll 'tab';
 wrt 6.2,A$[I];                   Tab right and print name
 1→J                              Initialize column counter
```

ANNOTATED LISTING:

19: cll 'tab'; wrt 6.1,A[I,J]; jmp (J+1→J)>5	Tab right and print record entry
20: cll 'skip'(1);if (I+1→I)>N; gto +4	Skip one line and test for end of records
21: cll 'tab'(- 1);wrt 6.1,A[I, J-1→J]	Tab left and print rightmost entry of the record
22: cll 'tab'(- 2);wrt 6.1,A[I, J-1→J];jmp J<2	Tab left twice and print next record entry
23: cll 'tab'(- 2);wrt 6.2,A$[I];cll 'tab'(-2)	Tab left and print name
24: if (I+1→I)<= N;cll 'skip'(1) ;gto -6	Check if last record. If not branch to print next record from left to right
25: end	
*31551	

File 44 Four-Leafed Rose



Reduced 45%

ANNOTATED LISTING:

PROGRAM EXAMPLE FOR 9871A PRINTER (FILE 44 FOUR-LEAFED ROSE)

```
0: 6→r0;rad                    Initialize select code for 9871A;Radian measure
1: cll 'form'(13               Set up size of page
 ,12,12)
2: cll 'psiz'(12               Initialize size of plot
 ,12,.5,.5)
3: cll 'scl'(-                 Initialize plot scale
 .5,.5,-.5,.5)
4: cll 'xaxis'(0
 )                             Print x and y axis with no tic marks
5: cll 'yaxis'(0
 )
6: 0→A;1→C                     A is angle; C is constant multiplier
7: Csin(A)cos(A)               Function value D = (CsinAcosA)^2 computed
 ↑2→D;cll 'plt'(
 Dsin(A),Dcos(A)               Plot the cartesian coordinates (x,y) associated
 ,42)                          with the polar coordinates A,C
8: if (A+.01→A)<               Increment angle and compute next point
 =π;sto -1
9: Csin(A)cos(A)
 ↑2→D;cll 'plt'(
 -Dsin(A),Dcos(A               Same as above over interval π, 2π
 ),42)
10: if (A+.01→A)
 <=2π;sto -1
11: end
*17882
```

APPENDIX A

The following are additional commands that may be useful when using the 9871A and are easily executed without resorting to subroutines. <sc#> should be replaced by the select code number of the 9871A.

1. RESET — Sets the printer to its power-up state.
   syntax: wtb <sc#>, 27, 69

2. SELF TEST — Causes the printer to perform internal checks on its ROM and READ/WRITE memory. Then it prints a test pattern and returns to the power-up state (see Appendix C).
   syntax: wtb <sc#>, 27, 122

3. BELL — Causes the printer to make an audible beep.
   syntax: wtb <sc#>, 7

4. FORM FEED — Causes the printer to move to the top of the next form.
   syntax: wtb <sc#>, 12

5. CARRIER RETURN — Moves the carrier to the current left margin of the same line.
   syntax: wtb <sc#>, 14

6. SHIFT OUT — Causes all subsequent circumflex and acute accent characters to be replaced by exponentiation and apostrophe characters respectively. The "shift out" remains in effect until a "shift in" is recognized or until the printer is returned to its power up state.
   syntax: wtb <sc#>, 14

7. SHIFT IN — Cancels the "shift out" condition.
   syntax: wtb <sc#>, 15

8. CHARACTER REPLACEMENT

This command allows any ASCII code (see Appendix B) to be redefined to any sequence of other codes. The first parameter (char) is the ASCII code of the character to be replaced. The second parameter (N), is the number of characters in the replacement list which follows. A character replacement list is generated in the printer's memory in the area normally occupied by buffer. Buffer length is reduced by one character plus one character for every two characters in the replacement list. If the "character replacement" exceeds the memory available, the command will be ignored and a "beep" will sound. To restore a character to its original definition, a "character replacement" command must be given with the character to be restored and a list of length zero. This will delete it from the replacement list in memory, and expand the buffer accordingly.

syntax: wtb <sc#>, 27, 67, char, N, list

APPENDIX B

The following table gives a list of the characters that can be printed by the 9871A and their ASCII code (decimal code).

9871A ASCII CHARACTER CODE

DECIMAL CODE	CHARACTER	DECIMAL CODE	CHARACTER
32	Blank	58	:
33	!	59	;
34	"	60	<
35	#	61	=
36	$	62	>
37	%	63	?
38	&	64	@
39	´	65	A
39 (SO)	'	66	B
40	(	67	C
41	)	68	D
42	*	69	E
43	+	70	F
44	,	71	G
45	-	72	H
46	.	73	I
47	/	74	J
48	0	75	K
49	1	76	L
50	2	77	M
51	3	78	N
52	4	79	O
53	5	80	P
54	6	81	Q
55	7	82	R
56	8	83	S
57	9	84	T

DECIMAL CODE	CHARACTER
85	U
86	V
87	W
88	X
89	Y
90	Z
91	[
92	√
93	]
94	^
94 (SO)	↑
97	a
98	b
99	c
100	d
101	e
102	f
103	g
104	h
105	i
106	j
107	k
108	l
109	m
110	n
111	o
112	p
113	q
114	r
115	s
116	t
117	u
118	v
119	w

DECIMAL CODE	CHARACTER
120	x
121	y
122	z
123	π
124	|
125	→

APPENDIX C

POWER UP STATE

When power is first applied to the printer, or during a "test" or "reset" command, control goes through a "power up state". In this state, the following functions are performed:

1. An internal self test is made on ROM and read/write memory. A beep is sounded if a fault occurs.

2. The carrier is moved to the extreme left and rotated into a mechanical stop. This synchronizes the internal logic with mechanical position. The carrier is then moved to Column 1.

3. All horizontal and vertical tabs are cleared.

4. Horizontal spacing is set to 10 characters per inch. Variable horizontal spacing is disabled.

5. Vertical spacing is set to 6 lines per inch.

6. The view-advance function is disabled.

7. Top of form is assumed at the current platen position.

8. Left margin is assumed at column 1.

9. Text width is assumed to be 13.2 inches.

10. Form length is assumed to be 11 inches.

11. Text length is assumed to be 11 inches.

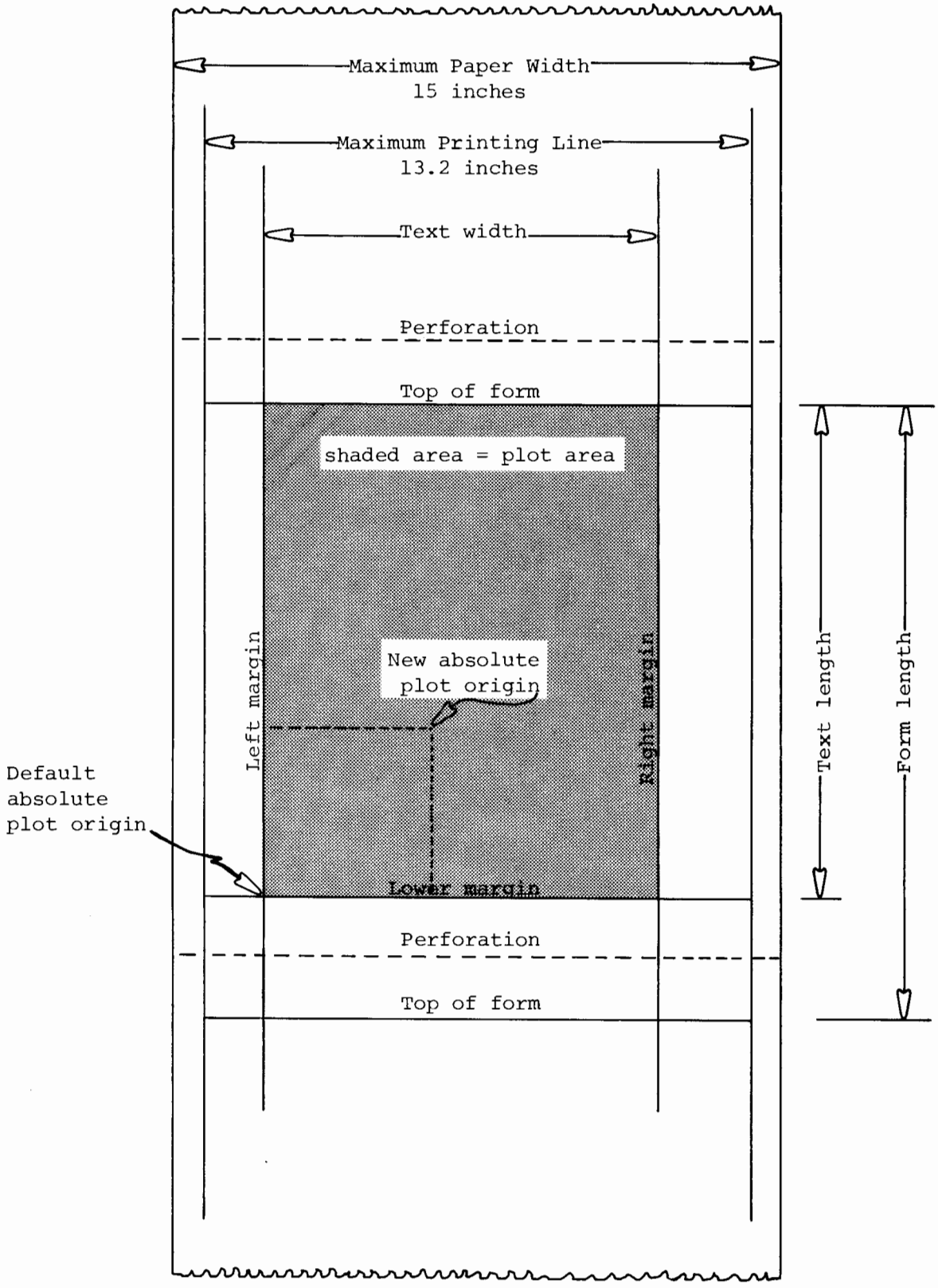
12. The origin for absolute plotting is set at the left margin and lower

margin.

13. Plot fill parameters are set to: decimal point, spacing = 3, vertical offset = +5.

14. The standard "shift-in" character set is assumed.

15. All character replacements are cleared.



Section 4

PLOTTING WITH THE dot STATEMENT
ON THE 9866B

PROGRAM	FILES	PAGE
Binary Program for dot Statement	45	367
Plotting Program Example	46	372

"dot" STATEMENT

When in the plot mode*, the 9866B printer will interpret each character it receives as a representation of a five dot pattern instead of an ASCII character. The 9866B print field consists of 80 columns which contain five printing dots and two non-printing dots, for a total of 560 dots.

Producing a plot on the 9866B printer can be a tedious procedure. The desired pattern to be printed must be determined by choosing appropriate sequences of characters. The dot statement has been created to perform the translation from a dot position in the plot row to the proper character in the column.

An 80-character string variable must be dimensioned to use the dot statement. Before plotting another row on the printer, the string variable must be initialized to blanks (" " → A$ [1,80]) so that all dots are in the non-printing state initially. Then the dot statement is used to "turn on" the desired dots for this plot row. Use of the dot statement is an additive process that only turns dots on, so it may be used several times before finally sending the string representing one row of the plot to the printer. When all the dots needed for the current row of the plot have been "activated", the plot code and the string are written to the printer (fmt c 80, b; wrt 6, 17, A$).

The syntax for the dot statement is:

dot <string variable>,<dot position>[,<dot position>]

<string variable> Name of string variable used to plot a line

<dot position> Expression yielding an integer between 0 and 559. If only the first dot position is specified then only that dot will be turned on. If both dot

positions are specified then all dots beginning at the first dot position and ending at the second dot position will be turned on.

The dot positions are numbered from 0 to 559 beginning with the leftmost printing dot of the 9866B. In general, it will be necessary to convert your functional values to the range 0 - 559 for use with the dot statement. If the largest and smallest values to be plotted across the width of the 9866B are known (call them MIN and MAX) and the point to be plotted is called P, then the following expression will convert your values to the range 0 - 559:

```
dot position = 559 * (P - MIN)/(MAX - MIN)
```

SPECIAL CONSIDERATIONS:

1. Error BØ will be caused by any of the following:
   a. Incorrect number of parameters for the dot statement (two or three are expected).
   b. The first parameter is not a string variable.
   c. The current length of the string variable is less than 80.
   d. The first or second character position is not in the range -32768 to 32767.
2. Each string should be reinitialized to blanks after each line of the plot is sent to the 9866B so that all dots are turned off for the next line (e.g.," " → A$[1,80]).
3. The plot mode must be set each time a new line is sent to the printer (e.g., fmt c80, b; wrt 6, A$, 17).
4. The dot statement is available only after loading the binary program in file 45 (ldb 45).

---

* Refer to the "98032A Option 66 Operating Note" for further details about the plot mode of the 9866B.

9866B PLOTTING CAPABILITIES

This program has been written to highlight the plotting capabilities of the 9866B printer. The implementation of these capabilities is facilitated by the dot statement.

The program itself will plot a user-defined function over any desired interval. To take full advantage of the dot statement, the x-axis has been printed down the page and the y-axis across the page. Each line that the 9866B prints in plotter mode is assigned an x-(domain) value over the interval [x min, x max]. For each domain value, the function value is computed and scaled to an integer between 0 and 559. That integer is stored in the appropriate position of the string to be printed by the dot statement.

The program requires that the user specify the following:

1. The function to be plotted.
2. The plot bounds - (x min, x max) and (y min, y max).
3. The location of the x- and y-axis.
4. The plot height and width is in inches.
   a. Height will be the distance in inches between y min and y max. The maximum height is 8".
   b. Width is the distance in inches between x min and x max.

VARIABLES USED:

Local:

p1	user point (Y) to be plotted
p2	if p2 has a value, then all points between p1 and p2 will be plotted

Global:

A	yaxis offset
B	xaxis offset
C	minimum X value
D	maximum X value
E	minimum Y value
F	maximum Y value
G	user interval size between successive X values
H	height of plot in inches
M	temporary storage for first parameter of dot statement
N	index for loop which computes functional values to be printed
W	width of plot in inches
X	domain values over the interval [Xmin, Xmax]
Y	function values
r0	select code for 9866B
A$	string in which the dot statement stores the points to be plotted
flg 13	set if user does not enter a requested input value

SPECIAL CONSIDERATIONS:

1. The binary program in file 45 should be loaded before any storage for variables is allocated.

2. The entire string that is used by the dot statement must be initialized to blanks before storing values in the string.

3. The decimal code, 17, must be sent to the 9866B before writing the string, to signal that the string is to be printed in plot mode, e.g., fmt b, c80; wrt <sc#>, A$, 17.

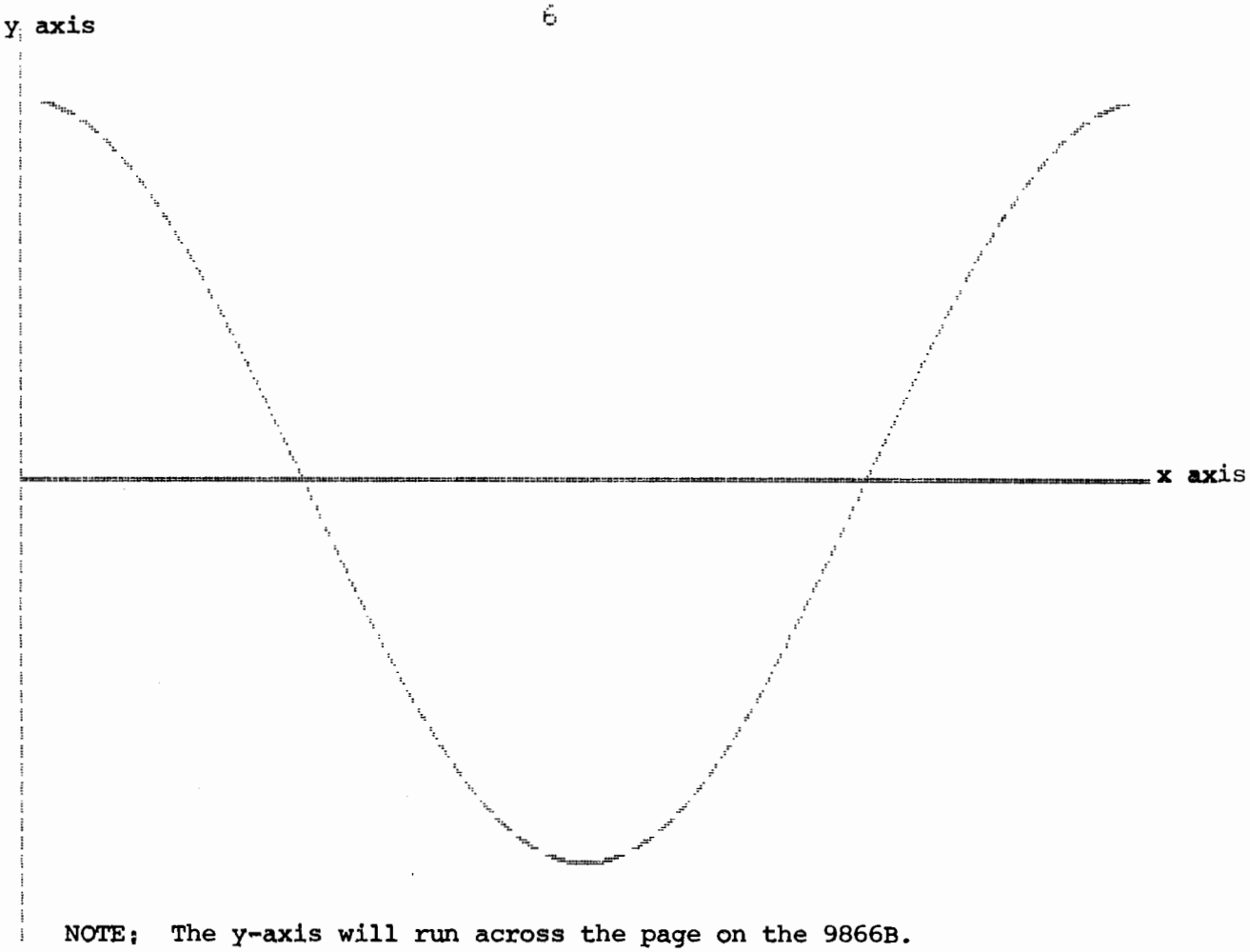
4. To use this plotting program the following ROMs are required: STRINGS ROM, AP ROM, GENERAL I/O ROM.

5. Only 5 out of 7 consecutive points in a horizontal line can be printed on the 9866B. Therefore, it is possible that points or entire lines which are parallel to the X axis may not appear on the plot.

EXAMPLE:

```
cos(X)→Y

0≤X≤2π
```

```
PLOT WIDTH
7
PLOT HEIGTH
5

XMIN
0
XMAX
2π

YMIN
-1.2
YMAX
1.2

FOR XAXIS

YAXIS OFFSET
0


FOR YAXIS

XAXIS OFFSET
0

SELECT CODE?
6
```



NOTE; The y-axis will run across the page on the 9866B.

INSTRUCTIONS:

1. Insert UTILITY ROUTINES cartridge with machine turned on.

2. Load the file.
   a. Type ldf 46
   b. Press EXECUTE

3. Press RUN

4. When "ENTER FUNCTION" is displayed:
   a. Press FETCH
   b. Enter 5
   c. Press EXECUTE

5. "cos(X)→Y" will appear in the display.
   a. Enter the function to be plotted in the form f(X)→Y (e.g. X↑3 + 2X↑2 + 1→Y)
   b. Press STORE

6. Continue execution
   a. Type cont 1 (you must type the word "cont" character by character rather than pressing the CONTINUE key)
   b. Press EXECUTE

7. When "PLOT WIDTH" is displayed:
   a. Enter the desired plot width in inches (length of X axis)
   b. Press CONTINUE

8. When "PLOT HEIGHT" is displayed:
   a. Enter the desired plot height in inches (length of Y axis)
   b. Press EXECUTE

9. When "XMIN" is displayed:
   a. Enter the minimum value of X (in user units) to be plotted
   b. Press CONTINUE

10. When "XMAX" is displayed:
   a. Enter the maximum value of X (in user units) to be plotted
   b. Press CONTINUE

11. When "YMIN" is displayed:
   a. Enter the minimum value of Y that can be plotted

b. Press CONTINUE

12. When "YMAX" is displayed:
    a. Enter the maximum value of Y that can be plotted:
    b. Press CONTINUE

13. "FOR XAXIS" will be printed. When "YAXIS OFFSET" is displayed:
    a. Enter the position on the x-axis through which the y-axis intersects
    b. Press CONTINUE

14. "FOR YAXIS" will be printed. When "XAXIS OFFSET" is displayed:
    a. Enter the position on the y-axis through which the x-axis intersects
    b. Press CONTINUE

15. "SELECT CODE?" will be displayed:
    a. Enter the select code for the 9866B Printer
    b. Press CONTINUE

16. The function will be plotted on the 9866B.

ANNOTATED LISTING:

```
0: ldb 45;dsp                    Load the binary file containing dot statement
 "ENTER FUNCTION                 Stop program for user to enter desired function
 ";rad;stp
1: dim A$[80];                   Dimension string used in dot statement
 fmt b,c80                       Set up format
2: gsb "SETUP"                   Branch to routine to set up the plot area
3: for X=C to D                          LOOP TO PLOT FUNCTION
 by G                            C is initial X value, D is final X value, G is
4: " "→A$[1,80]                  increment.  Initialize entire string to blanks
5: cos(X)→Y                      User-defined function stored here
6: if abs((X-A)/                 Check if yaxis should be printed
 G)<1;cll 'plot'
 (E,F)
7: cll 'plot'(Y)                 Put function value in A$
8: cll 'plot'(B)                 Put xaxis value in A$
9: wrt r0,17,A$                  Write A$ on 9866B in plot mode
10: next X
11: end
12: "plot":if                            dot STATEMENT
 p0=1;dotA$,int(                 If only one point, fill the appropriate scaled
 (p1-E)V)→M,M+1;                 position in A$ and the one next to it.
 ret
13: dotA$,int((p                 If two points are specified, fill all positions
 1-E)V),int((p2-                 between the two points
 E)V);ret
14: "SETUP":cfg                          SETUP PLOT AREA
 13
15: enp "PLOT
 WIDTH",W;if
 flg13;cfg 13;                   Enter the width of the plot in inches
 gto +0
16: enp "PLOT
 HEIGTH",H;if
 flg13;cfg 13;                   Enter the height of the plot in inches
 gto +0
17: spc ;enp
 "XMIN",C;if                     Enter the minimum X value
 flg13;cfg 13;
 gto +0
18: enp "XMAX",
 D;if flg13;cfg                  Enter the maximum X value
 13;gto +0
19: spc ;enp
 "YMIN",E;if                     Enter the minimum Y value
 flg13;cfg 13;
 gto +0
20: enp "YMAX",
 F;if flg13;cfg                  Enter the maximum Y value
 13;gto +0
21: (D-C)/54W→G                  G is the increment between X values plotted
22: 70H/(F-E)→V                  V is the scale factor for the yaxis
```

```
23: spc ;prt
 "FOR XAXIS";
 spc                          Enter the position on the xaxis
24: enp "YAXIS                where the yaxis intersects
 OFFSET",A;spc ;
 if fls13;cfs
 13;sto +0
25: spc ;prt
 "FOR YAXIS";
 spc                          Enter the position on the yaxis
26: enp "XAXIS                where the xaxis intersects
 OFFSET",B;spc ;
 if fls13;cfs
 13;sto +0
27: enp "SELECT
 CODE?",r0;spc ;
 if fls13;cfs                 Enter the select code of the 9866B
 13;sto +0
28: ret
*4483
```

TRAINING TAPE

SECTION

UTILITY ROUTINES - PART 2

TRAINING TAPE

PROGRAM	FILES	PAGE
Training Tape	0	379

TRAINING TAPE

STARTING INSTRUCTIONS

1. Relax

2. Turn the calculator ON

3. Insert the UTILITY ROUTINES cartridge

4. When"⊢"appears in the display,
   Type trk 1; ldf 0
   Press EXECUTE

5. When"⊢"appears in the display,
   Press RUN

6. When"READY?" appears in the display, the calculator is stopped.
   To go on,
   Press CONTINUE

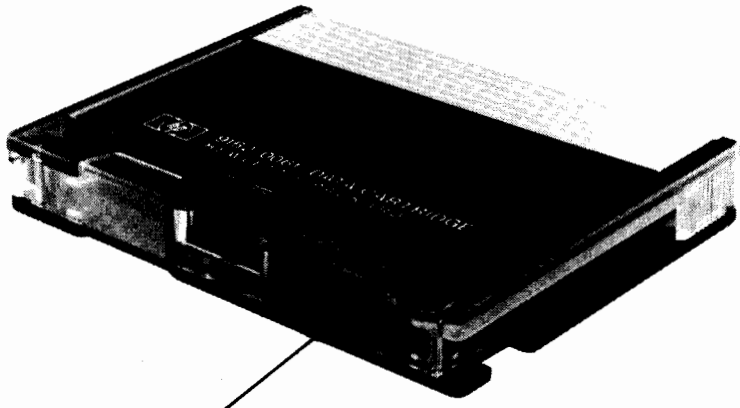
This training program introduces you to the world of the programmable calculator. Though it won't teach you to be an expert programmer, it will show you some features of the 9825A. It is meant to be fun to use.

You cannot damage either the calculator or the training cartridge if you take the following precautions:

- If someone else has been using the calculator, make sure that they are finished before you begin. Otherwise, you might erase a valuable program or data.

- Check the training cartridge to make sure that the "RECORD" slide is to the left in the protect position (see Diagram). This insures that the training cartridge will not be accidentally written on.

For the training program, you will need a blank cartridge of your own, a basic 9825A calculator and the training cartridge.



The HP Tape Cartridge