



HEWLETT-PACKARD CALCULATOR

The program material contained herein is supplied without representation or warranty of any kind. Hewlett-Packard Company therefore assumes no responsibility and shall have no liability, consequential or otherwise, of any kind arising from the use of this program material or any part thereof.



CALCULATOR PRODUCTS DIVISION
P.O. Box 301, Loveland, Colorado 80537, Tel (303) 667-5000
7 Rue du Bois-du-Lan, 1217 Meyrin, Geneva, Tel. (022) 41 54 00

Program Listing

Section I MATRICES AND LINEAR EQUATIONS

1. Real Matrix Arithmetic	7
2. Complex Matrix Arithmetic	13
3. Real Matrix Inversion, $N \leq 9$ or $N \leq 18$	19
4. Complex Matrix Inversion, $N \leq 5$ or $N \leq 12$	25
5. N Simultaneous Linear Equations with Real Coefficients in N Unknowns, $N \leq 9$, $N \leq 18$	31
6. N Simultaneous Linear Equations with Real Coefficients in N Unknowns, $N \leq 17$, $N \leq 36$	37
7. N Simultaneous Linear Equations with Complex Coefficients in N Unknowns, $N \leq 4$, $N \leq 11$	43
8. Characteristic Equation	49
9. Eigenvalues of a Matrix with Real Entries	55

Section II POLYNOMIALS

1. Polynomial Arithmetic	63
2. Polynomial Root Finder	69
3. Polynomial Evaluation, Normalization, or Construction	77
4. Polynomial Integration, Differentiation, or Origin Shift	83

Section III INTEGRATION AND ANALYSIS

1. Simultaneous Ordinary Differential Equations, Runge-Kutta (Gill Method)	91
2. Romberg Quadrature for $\int_a^b f(x) dx$	99
3. Gaussian Quadrature for $\int_a^b f(x) dx$ or $\int_a^\infty f(x) dx$	105
4. Numerical Integration with Equally Spaced Base Points, Newton-Cotes Closed Formulas	113
5. Numerical Integration with Unequally Spaced Base Points, Interpolation, First and Second Derivatives, and Curve Through Points Using Spline Functions	119
6. Curve Fitting by Chebyshev Polynomials	125
7. General Function Plot	131
8. Lagrange Interpolation for Unequally Spaced Data	137
9. Fourier Series Coefficients for Unequally Spaced Data	143
10. Partial Sum of an Infinite Series/Partial Product of an Infinite Product	151
11. Roots of $F(X) = 0$ in an Interval	159

Section IV SPECIAL FUNCTIONS

1. Bessel Functions I	167
2. Bessel Functions II	173

Introduction

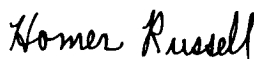
Programs for your 9820A Math Pac have been selected primarily from three sources: requests from you, our customers; programs you have submitted for the 9100 family of calculators; and some useful programs from previous calculator and larger computer systems' math libraries.

A new approach has been used for many of the programs. A few of them such as the Gaussian Quadrature program are multi-purpose. This particular program allows you to integrate a given function by any one of five different formulas enabling you to compare the integration results of several techniques without having to load and run another program or reenter data. Other programs have been designed to be easily used in conjunction with one another. For example, Fourier Series Coefficients for Unequally Spaced Data may be used with the General Function Plot program to produce several plots without having to reenter data. Still others perform tasks not indicated by the title, such as Real Matrix Inversion from which the (absolute value of the) determinant of a matrix may be obtained, and such as the Matrix Arithmetic programs which may be used to add, subtract, or find the dot product of vectors.

With the power of the new 9820A algebraic language and the ease and pleasure with which programs may be written, we believe you will be developing many programs and using techniques previously not possible on a desktop system. We encourage you to submit these to Hewlett-Packard so that we may share them with other users, either through the KEYBOARD, by adding them to the Program Catalog, or by including them in a future Pac. Your contributions, requests, and suggestions are our most important source for supplying you with user-oriented programs in the future. Program submittal forms and details for submitting programs may be found in the back of this Pac. Please use these forms when submitting programs.

We at Hewlett-Packard look forward to serving you.

Sincerely,
HEWLETT-PACKARD COMPANY



Homer Russell
Product Development Engineer
Loveland, Colorado

Comments

About 60 individual programs have been combined to form the 26 multi-purpose programs of your 9820A Math Pac. Selection of a particular option in one of the multi-purpose programs is done by pressing RUN PROGRAM a number of times to bring the different options into the display one at a time. When the desired option is displayed, pressing 1 RUN PROGRAM will cause the option being displayed to be selected. If by accident you pass by the option wanted, pressing RUN PROGRAM several more times will complete the display cycle and cause the display of the options to be repeated from the first.

As mentioned in the introduction, some of the programs perform functions in addition to those indicated in the title. Another example of this is the Eigenvalue program which will print out corresponding eigenvectors for most, but not all, problems. By considering a complex number to be the sole entry of a 1×1 matrix, the Complex Matrix Arithmetic program can be used to add, subtract, or multiply complex numbers. Or the Complex Matrix Inversion program can be used in the same manner to find the reciprocal of a complex number.

When running a program and the calculator is stopped, waiting for data entry or displaying a message and you wish to abort the program, the calculator will probably be in an ENTER statement in which case pressing STOP will first be necessary to get out of the ENTER statement. You may press ERASE instead but remember that this will clear all memory – your program included.

At the top of the User Instructions accompanying each program is the system configuration for which the program was designed. The Internal Registers box that is checked indicates the minimum number of registers required for the program. ROM Blocks should be in the indicated slots, and slots shown blank should contain no blocks. Turn power OFF when inserting or extracting ROM Blocks. This is necessary so that the calculator may internally reconfigure codes assigned to keys.

The Math Pac has been laid out so that the Example page will be opposite the User Instructions for easy reference. To keep the User Instructions brief, the convention of underlining has been adopted to indicate key depression. For instance, "RUN PROGRAM" means the same as "Press RUN PROGRAM", and "GO TO 0 EXECUTE" means "Press GO TO 0 EXECUTE". Normally, each separate keystroke will be underlined, but in an expression like "f(x) \longrightarrow Y" where the number of keystrokes required to define $f(x)$ is only known by you, the user, only one line has been used. Occasionally redundancy is used for clarity by using the words, "key in" or "press" in conjunction with the underlining.

A Memory Allocation page has been included with each program. Most of the variables shown stored in particular registers have not been defined in the program writeup and have meaning only for the program writer. The primary purpose of the page is to show you what registers and flags have been used, indicating which ones are still available.

A pre-recorded set of magnetic cards for the 9820A Math Pac may be purchased under the part number 09829-70000 through your local Hewlett-Packard sales office.



REAL MATRIX ARITHMETIC

With this program, matrices with real entries of any size up to 5×5 (respectively, 16×16 with the expanded internal memory) may be added or subtracted, and matrices up to 4×4 (respectively, up to 12×12) may be multiplied. However, the smaller that one matrix is, the larger the other one may be. For example, a 16×16 matrix may be applied to a 16×1 column matrix. In any case, the resultant matrix is printed but also remains in the calculator and may operate on or be operated on by a third matrix. Thus the program allows such computations as AB , BA , $A+B$, $A-B$, $B-A$, $C(B-A)$, $(B-A)C$, ABC , and $[C(B-A)+D]E$, provided, of course, the operation in each instance is well-defined and matrix size limits are not exceeded. Since the resultant matrix remains in the calculator, data reentry is not required in performing a series of operations.

The actual size limitations are determined from the following relations:

<u>Matrix Size</u>	<u>Operation</u>	<u>Basic 9820A</u>	<u>Expanded 9820A</u>
$A = m \times n, B = m \times n$	$A+B$ or $A-B$	$mn \leq 32$	$mn \leq 288$
$A = m \times n, B = n \times p$	AB	$mn+mp \leq 32$ (if A is entered first) $np+mp \leq 32$ (if B is entered first)	$mn+mp \leq 288$ (if A is entered first) $np+mp \leq 288$ (if B is entered first)

For example, if A is 16×16 and B is 16×1 , AB can be found on the basic 9820A providing B is entered first since:

$$np+mp = 16 \cdot 1 + 16 \cdot 1 = 32 \quad (\leq 32)$$

whereas

$$mn+mp = 16 \cdot 16 + 16 \cdot 1 = 272 \quad (> 32)$$

showing that A cannot be entered first. (Note: The program always calls for matrix A to be entered first; so merely rename the matrices to correspond to this convention when necessary).

As a secondary function of the program, two vectors such as $A = (a_1 \ a_2 \ a_3)$ and $B = (b_1 \ b_2 \ b_3)$ may be added or subtracted by treating A and B as 1×3 matrices. Their dot product $A \cdot B$ may be obtained by treating A as a 1×3 matrix and B as a 3×1 matrix and forming their product:

$$A \cdot B = AB = (a_1 \ a_2 \ a_3) \begin{pmatrix} b_1 \\ b_2 \\ b_3 \end{pmatrix}$$

ROM BLOCKS

1

2

3

INTERNAL REGISTERS

☒ 173 ☐ 429

PERIPHERALS

<input type="checkbox"/> 9860A MARKED CARD READER	<input type="checkbox"/> 9862A PLOTTER	<input type="checkbox"/> 9864A DIGITIZER	<input type="checkbox"/> _____
<input type="checkbox"/> 9861A TYPEWRITER	<input type="checkbox"/> 9863A PAPER TAPE READER	<input type="checkbox"/> _____	<input type="checkbox"/> _____

STEP	DISPLAY	INSTRUCTIONS
1.		<u>ERASE</u> <u>LOAD</u> <u>EXECUTE</u>
2.		Alternately insert magnetic cards and <u>EXECUTE</u> until NOTE 14 no longer appears, indicating completion of loading.
3.		(Optional) <u>FIXED N n EXECUTE</u> , or <u>FLOAT N n EXECUTE</u> .
4.		<u>END</u> <u>RUN PROGRAM</u>
5.	ROWS IN A?	<u>m</u> <u>RUN PROGRAM</u> m = number of rows in A
6.	A BY ROWS	<u>a_{ij}</u> <u>RUN PROGRAM</u> Row by row until A is entered; then <u>RUN PROGRAM</u> and A will be printed.
		Press the A and B keys in one of the following combinations (in Step 7 or 8) to designate the desired operation:
7.	A*B?	<u>A x B</u> <u>RUN PROGRAM</u> and go to Step 11, or <u>A + B</u> <u>RUN PROGRAM</u> and go to Step 12, or <u>A - B</u> <u>RUN PROGRAM</u> and go to Step 12, or <u>RUN PROGRAM</u> to go to Step 8.
8.	B*A?	<u>B x A</u> <u>RUN PROGRAM</u> and go to Step 9, or <u>B + A</u> <u>RUN PROGRAM</u> and go to Step 10, or <u>B - A</u> <u>RUN PROGRAM</u> and go to Step 10, or <u>RUN PROGRAM</u> to go to Step 7.
9.	ROWS IN B?	<u>n</u> <u>RUN PROGRAM</u> n = number of rows in B
10.	B BY COLS.	<u>b_{ij}</u> <u>RUN PROGRAM</u> Column by column until B is entered; then go to Step 13.
11.	COLS. IN B?	<u>p</u> <u>RUN PROGRAM</u> p = number of columns in B
12.	B BY ROWS	<u>b_{ij}</u> <u>RUN PROGRAM</u> Row by row until B is entered.
13.		The result is printed and then renamed A, replacing the old A in memory. Control returns to Step 7 and another operation may be performed with the new matrix A using a new incoming matrix B.

EXAMPLES

1. Let

$$A = \begin{pmatrix} 3 & 2 & 4 \\ 1 & 6 & -1 \\ 1 & 2 & 0 \end{pmatrix} \text{ and } B = \begin{pmatrix} 1 \\ -1 \\ 2 \end{pmatrix}$$

then

$$AB = \begin{pmatrix} 9 \\ -7 \\ -1 \end{pmatrix}$$

$$2. \text{ Show that } (B - A)C = \begin{pmatrix} 0 & -3 \\ -4 & -3 \end{pmatrix}$$

$$\text{for } A = \begin{pmatrix} -5 & -2 \\ 3 & 0 \end{pmatrix}, B = \begin{pmatrix} 4 & 1 \\ 20 & 7 \end{pmatrix}, \text{ and}$$

$$C = \begin{pmatrix} 1 & -1 \\ -3 & 2 \end{pmatrix} \text{ by first forming } B - A \text{ and}$$

then multiplying by C.

A
BY ROWS

3.0000
2.0000
4.0000

1.0000
6.0000
-1.0000

1.0000
2.0000
0.0000

B
BY ROWS

1.0000
-1.0000
2.0000

A*B
BY ROWS

9.0000
-7.0000
-1.0000

A
BY ROWS

-5.0000
-2.0000

3.0000
0.0000

B
BY COLS.

4.0000
20.0000

1.0000
7.0000

B-A
BY ROWS

9.0000
3.0000

17.0000
7.0000

This becomes the new
matrix A, replacing the
old A in the calculator.

B
BY ROWS

1.0000
-1.0000

-3.0000
2.0000

This is the next matrix in
the sequence (= matrix C).

A*B
BY ROWS

0.0000
-3.0000

-4.0000
-3.0000

Final answer

REGISTERS

A 2
B Address
C +, -, x, Indicator
X c
Y Counter, Temporary
Z Temporary

0 Slow Counter
1 Fast Counter
2 n
3 p
4 l
5 m
6

Matrix Elements and
 Working Area



FLAGS

0
1
2 Clear Area Indicator
3
4 Left, Right Flag
5
6
7
8
9
10
11
12
13 Used
14
15

PROGRAM LISTING

```

00:
ENT "ROWS IN A?"
,R(5+X);CFG 13;0
+CH
1:
ENT "A BY ROWS";
R(X+1+X);JMP
FLG 13F
2:
(X-5)/R5+R2F
3:
0+X+R0;CFG 2;R5+
R3;R2+R4;SPC 2F
4:
IF C=0;PRT "
    A"F
5:
IF C=1;PRT "
    A-B"F
6:
IF C=-1;PRT "
    B-A"F
7:
IF CFLG 4=2;PRT
"        A*B";
JMP 4F
8:
IF C=2;PRT "
    B*A"F
9:
IF (C=3)FLG 4;
PRT "        A+B"
;JMP 2F
10:
IF C=3;PRT "
    B+A"F
11:
PRT "        BY ROW
S";SPC 1+YF
12:
PRT R((X+1+X)+5)
;Y(Y+R2)+1+Y;
SPC Y=1;JMP X=R2
R5F
13:
1+X+B;2+AF
14:
SFG 4;CFG 13;
ENT "A*B?";C;IF
FLG 13;CFG 13;
ENT "B*A?";C;
CFG 4;JMP FLG 13
=0F

```

```

15:
IF CFLG 4=2;ENT
"COLS IN B?";R3;
JMP 3F
16:
IF C=2;ENT "ROWS
    IN B?";R4;JMP 2
F
17:
R5+R4;R2+R3F
18:
R0+1+R(0+R1)F
19:
IF R0(R1+1+R1)≠1
;JMP 3F
20:
PRT "        B";
IF FLG 4;PRT "
    BY ROWS";SPC
1;JMP 2F
21:
PRT "        BY COL
S";SPC 1F
22:
IF FLG 4;ENT "B
    BY ROWS";Z;JMP 2
F
23:
ENT "B BY COLS."
,ZF
24:
PRT ;SPC R1=R((
    FLG 4=0)+3)F
25:
IF C=2;JMP 14F
26:
R2R(FLG 4=0)-R2+
RFLG 4+5+BF
27:
IF 2>C;-Z+ZF
28:
Z+RB+ZF
29:
IF (C≠3)(FLG 4=0
);-Z+ZF
30:
Z+RB+
31:
IF R((FLG 4=0)+3
)>R1;JMP -12F
32:
IF R(FLG 4+3)>R0
;JMP -14F

```

```

33:
IF C≠2;GTO 3F
34:
R2R5+Z;0+XF
35:
IF FLG 4=0;R4+R5
;JMP 2F
36:
R3+R2F
37:
IF (X+1+X)>R2R5;
GTO 3F
38:
R(X+5+Z)+R(X+5);
JMP -1F
39:
0+A;IF FLG 2;
JMP 2F
40:
0+R((A+1+A)+R2R5
+5);JMP A=R(FLG
4+4)R(FLG 4+2)F
41:
0+A;SFG 2F
42:
IF (A+1+A)>R(3
    FLG 4+2);JMP -11
F
43:
ZR((R2A-R2+R0)
    FLG 4+(R2R0-R2+A
    )(FLG 4=0)+5)÷BF
44:
(R3A-R3+R1)FLG 4
+(R2R1-R2+A)(
    FLG 4=0)+R2R5+5+
YF
45:
B+RY+RY;JMP -3F
46:
END F

```




COMPLEX MATRIX ARITHMETIC

With this program, matrices with complex entries of any size up to 11×11 may be added or subtracted, and matrices up to 8×8 may be multiplied. However, the smaller one matrix is, the larger the other one may be. For example, a 30×30 matrix may be applied to a 30×1 column matrix. In any case, the resultant matrix is printed but also remains in the calculator and may operate on or be operated on by a third matrix. Thus the program allows such computations as AB , BA , $A+B$, $A-B$, $B-A$, $C(B-A)$, $(B-A)C$, ABC , and $[C(B-A) + D]E$, provided, of course, the operation in each instance is well-defined and matrix size limits are not exceeded. Since the resultant matrix remains in the calculator, data reentry is not required in performing a series of operations.

The actual size limitations are determined from the following relations:

<u>Matrix Size</u>	<u>Operation</u>	<u>Expanded 9820A</u>
$A = m \times n, B = m \times n$	$A \pm B$	$mn \leq 129$
$A = m \times n, B = n \times p$	AB	$mn + mp \leq 129$ (If A is entered first) $np + mp \leq 129$ (If B is entered first)

For example, if A is 11×11 and B is 11×1 , AB can be found provided B is entered first since:

$$np + mp = 11 \cdot 1 + 11 \cdot 1 = 22 (\leq 129)$$

whereas $mn + mp = 11 \cdot 11 + 11 \cdot 1 = 132 (> 129)$

which is too large showing that A cannot be entered first. (Note: The program always calls for matrix A to be entered first; so merely rename the matrices to correspond to this convention when necessary).

As a secondary function of the program, two complex vectors such as $C = (c_1 \ c_2 \ c_3)$ and $C' = (c'_1 \ c'_2 \ c'_3)$ may be added or subtracted by treating C and C' as 1×3 matrices. Their dot product $C \cdot C'$ may be obtained by treating C as a 1×3 matrix and C' as a 3×1 matrix and forming their product:

$$C \cdot C' = CC' = (c_1 \ c_2 \ c_3) \begin{pmatrix} c'_1 \\ c'_2 \\ c'_3 \end{pmatrix}$$

ROM BLOCKS

1

2

3

INTERNAL REGISTERS

☐ 173 ☒ 429

PERIPHERALS

☐ 9860A MARKED CARD READER☐ 9862A PLOTTER☐ 9864A DIGITIZER☐☐ 9861A TYPEWRITER☐ 9863A PAPER TAPE READER☐☐

STEP	DISPLAY	INSTRUCTIONS	
1.		<u>ERASE</u> <u>LOAD</u> <u>EXECUTE</u> , then alternately insert magnetic cards and <u>EXECUTE</u> until completion of loading.	
2.		(Optional) <u>FIXED N n EXECUTE</u> , or <u>FLOAT N n EXECUTE</u> .	
3.		<u>END RUN PROGRAM</u>	
4.	ROWS IN A?	<u>m RUN PROGRAM</u>	m = number of rows in A.
5.	A BY ROWS: RL A BY ROWS: IM	$\left. \begin{array}{l} \text{Rl } a_{ij} \text{ } \underline{\text{RUN PROGRAM}} \\ \text{Im } a_{ij} \text{ } \underline{\text{RUN PROGRAM}} \end{array} \right\}$	Row by row until A is entered; then <u>RUN PROGRAM</u> alone.
Press the A and B keys in one of the following combinations (in Step 6 or 7) to designate the desired operation:			
6.	A*B?	$\begin{array}{l} \underline{A} \times \underline{B} \text{ } \underline{\text{RUN PROGRAM}} \text{ and go to Step 10, or} \\ \underline{A} + \underline{B} \text{ } \underline{\text{RUN PROGRAM}} \text{ and go to Step 11, or} \\ \underline{A} - \underline{B} \text{ } \underline{\text{RUN PROGRAM}} \text{ and go to Step 11, or} \\ \underline{\text{RUN PROGRAM}} \text{ to go to Step 7.} \end{array}$	
7.	B*A?	$\begin{array}{l} \underline{B} \times \underline{A} \text{ } \underline{\text{RUN PROGRAM}} \text{ and go to Step 8, or} \\ \underline{B} + \underline{A} \text{ } \underline{\text{RUN PROGRAM}} \text{ and go to Step 9, or} \\ \underline{B} - \underline{A} \text{ } \underline{\text{RUN PROGRAM}} \text{ and go to Step 9, or} \\ \underline{\text{RUN PROGRAM}} \text{ to go back to Step 6.} \end{array}$	
8.	ROWS IN B?	<u>n RUN PROGRAM</u>	n = number of rows in B.
9.	B BY COLS: RL B BY COLS: IM	$\left. \begin{array}{l} \text{Rl } b_{ij} \text{ } \underline{\text{RUN PROGRAM}} \\ \text{Im } b_{ij} \text{ } \underline{\text{RUN PROGRAM}} \end{array} \right\}$	Column by column until B is entered; then go to Step 12.
10.	COLS. IN B?	<u>p RUN PROGRAM</u>	p = number of columns in B.
11.	B BY ROWS: RL B BY ROWS: IM	$\left. \begin{array}{l} \text{Rl } b_{ij} \text{ } \underline{\text{RUN PROGRAM}} \\ \text{Im } b_{ij} \text{ } \underline{\text{RUN PROGRAM}} \end{array} \right\}$	Row by row until B is entered.
12.		The result is printed and then renamed A, replacing the old A in memory. Control returns to Step 6 and another operation may be performed with the new matrix A using a new incoming matrix B.	

EXAMPLES

Show that $AB + C = \begin{pmatrix} 37 + 26i \\ 39 - 4i \end{pmatrix}$ for

$$A = \begin{pmatrix} 2+0i & 6+5i \\ 4+2i & 3+0i \end{pmatrix}, \quad B = \begin{pmatrix} 2-3i \\ 6-0i \end{pmatrix}, \quad \text{and } C = \begin{pmatrix} -3+2i \\ 7+4i \end{pmatrix}$$

Note: Although not indicated in the User Instructions, if $\text{Im } a_{ij} = 0$ or $\text{Im } b_{ij} = 0$, the program is set up to automatically make a 0 entry if RUN PROGRAM occurs alone when entry of either of these values is called for, thus saving the need to press the 0 key.

A
BY ROWS

2.0000
0.0000
6.0000
5.0000

4.0000
2.0000
3.0000
0.0000

B
BY ROWS

2.0000
-3.0000

6.0000
0.0000

This becomes the new
matrix A, replacing the
old A in the calculator.

A*B
BY ROWS

40.0000
24.0000

32.0000
-8.0000

This is the next matrix in
the sequence (= matrix C).

B
BY ROWS

-3.0000
2.0000

7.0000
4.0000

Final answer

A+B
BY ROWS

37.0000
26.0000

39.0000
-4.0000

REGISTERS

A 2
B Address
C +, -, x, Indicator
X c
Y Counter, Temporary
Z Temporary

0 Slow Counter
1 Fast Counter
2 n
3 p
4 l
5 m
6

Matrix Elements and
 Working Area



FLAGS

0
1
2 Clear Area Indicator
3
4 Left, Right Flag
5
6
7
8
9
10
11
12
13 Used
14
15

PROGRAM LISTING

```

0:
ENT "ROWS IN A?"
,R5:0→X→C←
1:
CFG 13:ENT "A BY
ROWS: RL",R(X+1
+X2+4):IF FLG 13
:JMP 2←
2:
0→A:ENT "A BY RO
WS: IM",A:A+R(2X
+5):JMP -1←
3:
X/R5→R2←
4:
0→X+R0:CFG 2:R5+
R3:R2→R4:SPC 2←
5:
IF C=0:PRT "
A"←
6:
IF C=1:PRT "
A-B"←
7:
IF C=-1:PRT "
B-A"←
8:
IF CFLG 4=2:PRT
"
A*B"←
JMP 4←
9:
IF C=2:PRT "
B*A"←
10:
IF (C=3)FLG 4:
PRT "
A+B"
:JMP 2←
11:
IF C=3:PRT "
B+A"←
12:
PRT "
BY ROW
S":SPC 1+Y←
13:
PRT R((X+1→X)2+4
),R(2X+5):SPC 1:
SPC 2((Y(Y≠R2)+1
→Y)=1):JMP X=R2R
5←
14:
1→X→B:2→A←
15:
SFG 4:CFG 13:
ENT "A*B?",C:IF

```

```

FLG 13:CFG 13:
ENT "B*A?",C:
CFG 4:JMP FLG 13
=0←
16:
IF CFLG 4=2:ENT
"COLS.IN B?",R3:
JMP 3←
17:
IF C=2:ENT "ROWS
IN B?",R4:JMP 2
←
18:
R5→R4:R2→R3←
19:
R0+1→R(0→R1)←
20:
IF R0(R1+1→R1)≠1
:JMP 3←
21:
PRT "
B"
IF FLG 4:PRT "
BY ROWS":SPC
1:JMP 2←
22:
PRT "
BY COL
S":SPC 1←
23:
0→Y:IF FLG 4:
ENT "B BY ROWS:
RL",Z:"B BY ROWS
: IM",Y:JMP 2←
24:
ENT "B BY COLS:
RL",Z:"B BY COLS
: IM",Y←
25:
PRT Z,Y:SPC 1:
SPC 2(R1=R(3+(
FLG 4=0)))←
26:
IF C=2:GTO 40←
27:
2(R2R(FLG 4=0)-R
2+RFLG 4)+4→B←
28:
IF 2>C:-Z→Z:-Y→Y
←
29:
Z+RB→Z:Y+R(B+1)→
Y←
30:
IF (C≠3)(FLG 4=0
): -Z→Z:-Y→Y←

```

```

31:
Z+RB:Y→R(B+1)←
32:
IF R((FLG 4=0)+3
)→R1:GTO 20←
33:
IF R(FLG 4+3)→R0
:GTO 19←
34:
IF C≠2:GTO 4←
35:
2R2R5+Z:0→X←
36:
IF FLG 4=0:R4→R5
:JMP 2←
37:
R3→R2←
38:
IF (X+1→X)→2R2R5
:GTO 4←
39:
R(X+5+Z)→R(X+5):
GTO -1←
40:
0→A:IF FLG 2:
JMP 2←
41:
0→R((A+1→A)+2R2R
5+5):JMP A=2R(
FLG 4+4)R(FLG 4+
2)←
42:
0→A:SFG 2←
43:
IF (A+1→A)→R(3
FLG 4+2):GTO 32←
44:
2(R2A-R2+R0)FLG
4+2(R2R0-R2+A)(
FLG 4=0)+4→X←
45:
2(R3A-R3+R1)FLG
4+2(R2R1-R2+A)(
FLG 4=0)+2R2R5+4
→B←
46:
ZR X-YR(X+1)+RB→R
B:YR X+ZR(X+1)+R(
B+1)→R(B+1):GTO
-3←
47:
END ←

```




REAL MATRIX INVERSION, $N \leq 9$ OR $N \leq 18$

This program will invert up to a 9×9 matrix $A = (a_{ij})$, a_{ij} real, on the basic 9820A and up to an 18×18 matrix with the expanded internal memory¹. A modified Gauss-Jordan elimination technique is used and newly computed elements overlay the original matrix, saving storage space. In addition, a row pivot search is utilized to increase the accuracy.

After the inverse A^{-1} is obtained, a checking option is provided to invert A^{-1} , without keying in its elements, to obtain A again². Comparison with the original A gives an indication of the inversion accuracy, part of the error due to the first inversion and part due to reinversion. This procedure is particularly recommended when the absolute value of the determinant, $|\det A|$, is small; this value may be obtained from register R0 after A^{-1} is printed.

¹ To invert a 7×7 , 8×8 , or 9×9 matrix (respectively, 18×18 for the expanded internal memory), store END in line 45 leaving out the remaining lines. This provides the additional required storage area by eliminating the checking option.

² The option only applies to matrices up to 6×6 in size (respectively, up to 17×17 for the expanded internal memory).

ROM BLOCKS

1

2

3

INTERNAL REGISTERS

☒ 173 ☐ 429

PERIPHERALS

☐ 9860A MARKED CARD READER☐ 9862A PLOTTER☐ 9864A DIGITIZER☐☐ 9861A TYPEWRITER☐ 9863A PAPER TAPE READER☐☐

STEP	DISPLAY	INSTRUCTIONS
1.		<u>ERASE</u> <u>LOAD</u> <u>EXECUTE</u>
2.		Alternately insert magnetic cards and <u>EXECUTE</u> until NOTE 14 no longer appears, indicating completion of loading.
3.		(Optional) <u>FIXED N n EXECUTE</u> , or <u>FLOAT N n EXECUTE</u> as desired.
4.		If the matrix to be inverted is 7x7 or larger (or 18x18 for the expanded internal memory), <u>GO TO 4 5 EXECUTE END STORE</u> . This will annihilate the checking option portion of the program, giving the needed memory for the larger matrix.
5.		<u>END RUN PROGRAM</u>
6. SIZE?	<u>n RUN PROGRAM</u>	n = number of rows
7. A(I,J)?	<u>a_{ij} RUN PROGRAM</u>	(by rows) i,j = 1,2,...,n. After all data is entered, the display blanks and calculations proceed. When completed, the elements of the inverse matrix A^{-1} are printed by rows. det A is in register R0 after printing.
8. CHECK?		Only displayed and applicable for matrices less than 7x7 in size (respectively, 18x18 for the expanded internal memory). <u>RUN PROGRAM</u> to run the checking option and then return to Step 6 for a new problem. Otherwise, or if the matrix is larger than or equal to 7x7 (respectively, 18x18), return to Step 5 for a new problem.
	DET A = 0	Displayed only when the program senses matrix A as being singular. No further processing can be done. Return to Step 5 for a new problem.

EXAMPLES

1. Invert the third order
Hilbert Matrix

$$A = \begin{pmatrix} 1 & \frac{1}{2} & \frac{1}{3} \\ \frac{1}{2} & \frac{1}{3} & \frac{1}{4} \\ \frac{1}{3} & \frac{1}{4} & \frac{1}{5} \end{pmatrix}$$

INVERSE OF

```

1.0000
.5000
.3333

.5000
.3333
.2500

.3333
.2500
.2000

```

IS

```

9.0000
-36.0000
30.0000

-36.0000
192.0000
-180.0000

30.0000
-180.0000
180.0000

```

RE-INVERSION

IS

```

1.0000
.5000
.3333

.5000
.3333
.2500

.3333
.2500
.2000

```

2. Invert

$$A = \begin{pmatrix} 6 & 5 \\ 9 & 2 \end{pmatrix}$$

INVERSE OF

```

6.0000
5.0000

9.0000
2.0000

```

IS

```

-.0606
.1515

.2727
-.1818

```

RE-INVERSION

IS

```

6.0000
5.0000

9.0000
2.0000

```

3. Invert

$$A = \begin{pmatrix} 6 & 8 \\ 3 & 4 \end{pmatrix}$$

INVERSE OF

```

6.0000
8.0000

3.0000
4.0000

```

IS

```

6.666666667E 10
-1.333333333E 11

-5.000000000E 10
1.000000000E 11

```

RE-INVERSION

The matrix A is singular and has determinant $\det A = 0$. However, machine round-off caused calculation of the determinant to be $\det A = 6.000000000 \text{ E-11}$, thus allowing continuation of processing and giving the above erroneous results. This example indicates the worth of either examining the value $|\det A|$ in register R0 to be certain it is not approximately zero, or of reinverting the result. In this case, reinversion catches the error and displays "DET A = 0". This example was specially selected and such occurrences can be expected to be rare. However, the user should be aware of the possibility. Since checking is so easy and desirable anyway, it was decided in writing the program not to abort calculations for all matrices with small determinants. This would in many cases eliminate matrices with perfectly valid inverses, and would unduly burden the user to rescale the data prior to entry.

Hilbert matrices are often used to test the accuracy of inversion techniques because they are ill-conditioned (a very small change in the coefficients produces a very large change in the solution).

REGISTERS	
A	n
B	i, Counter
C	j, Counter
X	Max. Elt., Counter
Y	Counter
Z	Counter
0	det (a_{ij})
1	<div style="display: flex; align-items: center; justify-content: center;"> <div style="margin-right: 10px;">↑</div> <div>Vector of Pointers</div> <div style="margin-left: 10px;">↓</div> </div>
2	
.	
.	
.	
n	
n+1	a_{11}
n+2	a_{12}
.	
.	
.	
2n	a_{1n}
2n+1	a_{21}
.	
.	
.	
FLAGS	
0	
1	Used
2	
3	
4	
5	
6	
7	
8	
9	
10	
11	
12	
13	
14	
15	

PROGRAM LISTING

```

0:
SPC 8; ENT "SIZE?"
", A; CFG 1+
1:
PRT " INVERSE
OF" ; 0+ B+
2:
B+1+ B; SPC 1+
3:
0+ C+
4:
C+1+ C+
5:
ENT "A(I, J)?" ; R(
AB+C); PRT +
6:
IF A=C; JMP -2+
7:
IF A=B; JMP -5+
8:
0+ B; 1+ R0+
9:
IF (B+1+ B) > A;
GTO 36+
10:
0+ C+ X+ Y+
11:
Y+1+ Y; IF B=1;
JMP 5+
12:
0+ Z+
13:
IF (Z+1+ Z) > B-1;
JMP 3+
14:
IF Y=RZ; JMP 5+
15:
JMP -2+
16:
R(A+Y+ B) + R B; IF 0 >
R B; -R B + R B+
17:
IF R B < X; JMP 2+
18:
R B + X; Y+ C+
19:
IF Y=A; GTO -8+
20:
R(X) R0 + R0; IF C=
0; JMP 2+
21:
DSP "DET A=0";
STP +

```

```

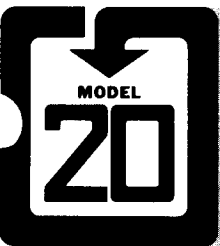
22:
C+ R B+
23:
0+ Z+
24:
IF (Z+1+ Z) = B;
JMP 2+
25:
R(A+ C+ Z) / R(A+ C+ B) +
R(A+ C+ Z) +
26:
IF Z=A; JMP -2+
27:
1/ R(A+ C+ B) + R(A+ C+ B
) +
28:
0+ Y+
29:
IF (Y+1+ Y) > A;
GTO 9+
30:
IF Y=C; JMP -1+
31:
0+ Z+
32:
IF (Z+1+ Z) = B;
JMP 2+
33:
R(A+ Y+ Z) - R(A+ C+ Z) R
(A+ Y+ B) + R(A+ Y+ Z) +
34:
IF Z=A; JMP -2+
35:
-R(A+ Y+ B) R(A+ C+ B) +
R(A+ Y+ B); GTO -6+
36:
0+ C; SPC 1; PRT "
IS" +
37:
C+1+ C; SPC 1+
38:
IF (C > A) (FLG 1=0
); JMP 7+
39:
0+ Z; IF C > A; GTO 0
+
40:
Z+1+ Z+
41:
IF Z > A; JMP -4+
42:
0+ X+
43:
X+1+ X; JMP R X=Z+

```

```

44:
PRT R(A+ C+ X);
GTO -4+
45:
0+ C; DSP "CHECK?"
; SFG 1; STP +
46:
SPC 1; PRT " RE-
INVERSION" +
47:
IF (C+1+ C) > A;
GTO +8+
48:
C+ Y; IF R C < 0; -R C +
R C; JMP -1+
49:
0+ X+
50:
X+1+ X; JMP R X=Y+
51:
-R X + R X; IF X=C;
JMP -3+
52:
X+ Y; 0+ Z+
53:
IF (Z+1+ Z) > A;
JMP -4+
54:
R(A+ C+ Z) + R0; R(A+ X+
Z) + R(A+ C+ Z); R0 + R(
A+ X+ Z); JMP -1+
55:
0+ Z+
56:
IF (Z+1+ Z) > A;
GTO 8+
57:
Z-1+ Y+
58:
Y+1+ Y; JMP R Y=Z+
59:
IF Y=Z; JMP -3+
60:
0+ C+
61:
C+1+ C; R(A+ C+ Z) + R0
; R(A+ C+ Y) + R(A+ C+ Z)
; R0 + R(A+ C+ Y); JMP
C=A+
62:
R Z + R Y; JMP -6+
63:
END +

```

COMPLEX MATRIX INVERSION, $N \leq 5$ OR $N \leq 12$

This program will invert up to a 5×5 matrix $A = (a_{ij})$, a_{ij} complex, on the basic 9820A and up to a 12×12 matrix on the 9820A having the additional 256 internal registers¹. A modified Gauss-Jordan elimination technique is used and newly computed elements overlay the original matrix saving storage space. In addition, a row pivot search is utilized to increase the accuracy.

After the inverse A^{-1} is obtained, a checking option is provided to invert A^{-1} , without keying in its elements, to obtain A again². Comparison with the original A gives an indication of the inversion accuracy, part of the error due to the first inversion and part due to reinversion. This procedure is recommended when possible. (Refer to the Real Matrix Inversion program for a discussion of the effect when the magnitude of the determinant is very small. This program does not include calculation of the determinant).

¹ To invert a 3×3 , 4×4 , or 5×5 matrix (respectively, 12×12 for the expanded 9820A), store END in line 47 of the program, leaving out the remaining lines. This provides the additional required storage area by eliminating the checking option.

² The option only applies to 1×1 and 2×2 matrices (respectively, 1×1 , 2×2 , 3×3 , ..., and 11×11 for the expanded 9820A).

ROM BLOCKS

1

2

3

INTERNAL REGISTERS

☒ 173 ☐ 429

PERIPHERALS

☐ 9860A MARKED CARD READER☐ 9862A PLOTTER☐ 9864A DIGITIZER☐☐ 9861A TYPEWRITER☐ 9863A PAPER TAPE READER☐☐

STEP	DISPLAY	INSTRUCTIONS
1.		<u>ERASE</u> <u>LOAD</u> <u>EXECUTE</u> , then alternately insert magnetic cards and <u>EXECUTE</u> until completion of loading.
2.		If the matrix is 3x3, 4x4, or 5x5 (or 12x12 for the expanded 9820A), <u>GO TO 4 7 EXECUTE END STORE</u> . This will annihilate the checking option portion of the program, giving the needed memory for the larger matrix.
3.		(Optional) <u>FIXED N n EXECUTE</u> , or <u>FLOAT N n EXECUTE</u> as desired.
4.		<u>END RUN PROGRAM</u>
5. SIZE?	<u>n RUN PROGRAM</u>	n = number of rows
6. RL A(I,J)? IM A(I,J)?	$\left. \begin{array}{l} \text{Rl } a_{ij} \\ * \text{Im } a_{ij} \end{array} \right\} \text{ RUN PROGRAM}$	by rows $i, j = 1, 2, \dots, n$ After all data is entered, the display blanks and calculations proceed. When completed, the elements of the inverse matrix A^{-1} are printed by rows (real parts first with imaginary parts below them).
7. CHECK?		Only displayed and applicable for 1x1 or 2x2 matrices (respectively, 1x1, 2x2, ..., 11x11 for the expanded 9820A). <u>RUN PROGRAM</u> to run the checking option and then return to Step 5 for a new problem. Otherwise, or if the matrix is larger than 2x2 (respectively, 11x11), return to Step 4 for a new problem.
	DET A = 0	Displayed only when the program senses matrix A as being singular. No further processing can be done. Return to Step 4 for a new problem.
<p>*NOTE: If $\text{Im } a_{ij} = 0$, <u>RUN PROGRAM</u> alone will automatically make the 0 entry, saving the need to press the 0 key.</p>		

EXAMPLES

1. Invert and check the 2x2 matrix

$$A = \begin{pmatrix} 1+i & 0+2i \\ -1+i & 1-i \end{pmatrix}$$

INVERSE OF

1.0000
1.0000

0.0000
2.0000

-1.0000
1.0000

1.0000
-1.0000

IS

.1000
-.3000

-.2000
-.4000

.1000
-.3000

.3000
.1000

RE-INVERSION

IS

1.0000
1.0000

0.0000
2.0000

-1.0000
1.0000

1.0000
-1.0000

2. Invert the 3x3 Hilbert matrix

$$A = \begin{pmatrix} 1 & 1/2 & 1/3 \\ 1/2 & 1/3 & 1/4 \\ 1/3 & 1/4 & 1/5 \end{pmatrix} \text{ Here, all imaginary parts are 0.}$$

Hilbert matrices are often used to test the accuracy of inversion techniques as they are ill-conditioned (a very small change in the coefficients produces a very large change in the solution).

INVERSE OF

IS

1.0000	9.0000
0.0000	0.0000
.5000	-36.0000
0.0000	0.0000
.3333	30.0000
0.0000	0.0000
.5000	-36.0000
0.0000	0.0000
.3333	192.0000
0.0000	0.0000
.2500	-180.0000
0.0000	0.0000
.3333	30.0000
0.0000	0.0000
.2500	-180.0000
0.0000	0.0000
.2000	180.0000
0.0000	0.0000

11

[illegible]

PROGRAM LISTING

```

0:
SPC 8;ENT "SIZE?
",A;PRT " INVE
RSE OF";CFG 1;
1:
0;B;
2:
B+1;B;SPC 2;
3:
0;C;
4:
C+1;C;
5:
ENT "RL A(I,J)?"
,R(AB+C);PRT ;0;
X;ENT "IM A(I,J)
?",X;X;R(AB+C+AA
);PRT ;SPC 1;
6:
IF A#C;JMP -2;
7:
IF A#B;JMP -5;
8:
0;B;
9:
IF (B+1;B)>A;
GTO 39;
10:
0;C+X+Y;
11:
Y+1;Y;IF B=1;
JMP 5;
12:
0;Z;
13:
IF (Z+1;Z)>B-1;
JMP 3;
14:
IF Y=RZ;JMP 5;
15:
JMP -2;
16:
R(AY+B)R(AY+B)+(
R(AY+B+AA)+R0)R0
+R0;
17:
IF R0<X;JMP 2;
18:
R0;X;Y;C;
19:
IF Y=A;GTO -3;
20:
IF C#0;JMP 2;

```

```

21:
DSP "DET A=0";
STP ;
22:
C;R;B;
23:
(R(AC+B+Y)+Z)Z+(
R(Y+AA)+Z)Z+R(0+
Z);
24:
IF (Z+1;Z)=B;
JMP 3;
25:
(R((AC+AA+Y)+Z)R
(AC+B)-R(Y+B)R(A
C+Z))/R0+X;
26:
(R(AC+Z)R(AC+B)+
R(Y+Z)R(Y+B))/R0
+R(AC+Z);X;R(Y+Z
);
27:
IF Z#A;JMP -3;
28:
R(AC+B)/R0+R(AC+
B);-R(Y+B)/R0+R(
Y+B);
29:
0;Y;
30:
IF (Y+1;Y)>A;
GTO 9;
31:
IF Y=C;JMP -1;
32:
0;Z;
33:
IF (Z+1;Z)=B;
JMP 3;
34:
R(AY+Z)-R(AC+Z)R
(AY+B)+R(AC+Z+AA
)R((AY+AA+X)+B)+
R(AY+Z);
35:
R(X+Z)-R(AC+Z+AA
)R(AY+B)-R(X+B)R
(AC+Z)+R(X+Z);
36:
IF Z#A;JMP -3;
37:
R(X+B)R((AC+B+X)
+AA)-R(AY+B)RX+X
;

```

```

38:
-R(AY+B+AA)R(AC+
B)-R(AC+B+AA)R(A
Y+B)+R(AY+B+AA);
X;R(AY+B);GTO -8
;
39:
0;C;SPC 1;PRT "
IS";
40:
C+1;C;SPC 2;
41:
IF (C>A)(FLG 1=0
);JMP 6;
42:
0;Z;IF C>A;GTO 0
;
43:
IF (Z+1;Z)>A;
JMP -3;
44:
0;X;
45:
JMP R(X+1;X)=Z;
46:
PRT R(ARC+X+B);R
(B+AA);SPC 1;
JMP -3;
47:
0;C;DSP "CHECK?"
;SFG 1;STP ;
48:
SPC 1;PRT " RE-
INVERSION";
49:
IF (C+1;C)>A;
JMP 9;
50:
IF R(C+Y)<0;-RC+
RC;JMP -1;
51:
0;X;
52:
JMP R(X+1;X)=Y;
53:
-RX+RX;IF X=C;
JMP -3;
54:
X;Y;0;Z;
55:
IF (Z+1;Z)>A;
JMP -4;

```

PROGRAM LISTING

```

56:
R(AC+(Z+AA+B))÷R
0:R(AX+B)÷R(AC+B
);R0÷R(AX+B)F
57:
R(AC+Z+B)÷R0;R(A
X+Z)÷RB;R0÷R(AX+
Z);JMP -2F
58:
0÷ZF
59:
IF (Z+1+Z)>A;
GTO 8F
60:
Z-1÷YF
61:
JMP R(Y+1+Y)=ZF
62:
IF Y=Z;JMP -3F
63:
0÷CF
64:
R(A(C+1+C)+Z+B)÷
R0;R(AC+Y)÷RB;R0
÷R(AC+Y)F
65:
R(B+AA)÷R0;R(AC+
Y+AA)÷R(B+AA+B);
R0÷R(B-Z+Y)F
66:
IF C≠A;JMP -2F
67:
RZ÷RY;GTO -8F
68:
END F

```



N SIMULTANEOUS LINEAR EQUATIONS WITH
REAL COEFFICIENTS IN N UNKNOWN, $N \leq 9$, $N \leq 18$

Model 20
MATH PAC
I-5

This program solves a system of up to nine simultaneous linear equations with real coefficients in nine unknowns (up to eighteen with the expanded internal memory):

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

$$a_{21}x_1 + a_{22}x_2 + \dots + a_{2n}x_n = a_{2n+1}$$

.....

$$a_{n1}x_1 + a_{n2}x_2 + \dots + a_{nn}x_n = a_{nn+1}$$

It is assumed that the determinant $\det(a_{ij}) \neq 0$, $i, j = 1, 2, \dots, n$. If this is not the case, the message $\text{DET(AIJ)} = 0$ is printed and displayed and the program is terminated.

The program uses a modified Gauss-Jordan method. A row pivot search is performed to maximize the accuracy of the results.

ROM BLOCKS

1

2

3

INTERNAL REGISTERS

☒ 173 ☐ 429

PERIPHERALS

☐ 9860A MARKED CARD READER☐ 9862A PLOTTER☐ 9864A DIGITIZER☐☐ 9861A TYPEWRITER☐ 9863A PAPER TAPE READER☐☐

STEP	DISPLAY	INSTRUCTIONS
1.		<u>ERASE</u> <u>LOAD</u> <u>EXECUTE</u>
2.		Alternately insert magnetic cards and <u>EXECUTE</u> until NOTE 14 no longer appears, indicating completion of loading.
3.		(Optional) <u>FIXED N n EXECUTE</u> , or <u>FLOAT N n EXECUTE</u> as desired.
4.		<u>END</u> <u>RUN PROGRAM</u>
5.	HOW MANY EQS.?	<u>n</u> <u>RUN PROGRAM</u>
6.	A(I,J)?	a_{ij} <u>RUN PROGRAM</u> (by rows) $\begin{matrix} 1 \leq i \leq n \\ 1 \leq j \leq n+1 \end{matrix}$ After all a_{ij} have been entered, calculations begin automatically and continue until x_1, x_2, \dots, x_n are found and printed out starting with x_1 . Control returns to Step 5 automatically for a new case.
	DET(AIJ) = 0	This message will be displayed and printed when the coefficient matrix on the left-hand side of the equation has zero determinant. The system cannot be processed. For a new case, return to Step 4.

EXAMPLE

$$n = 5$$

$$\begin{aligned} 2.1x_1 - 3.6x_2 + 5.8x_3 - 7.0x_4 - 1.2x_5 &= 12.8 \\ 3.7x_1 + 5.1x_2 - 7.8x_3 - 9.5x_4 - 3.5x_5 &= 9.8 \\ 1.8x_1 + 2.2x_2 + 6.5x_3 + 3.2x_4 - 2.5x_5 &= 15.9 \\ -6.6x_1 - 7.7x_2 - 3.1x_3 + 7.4x_4 + 9.4x_5 &= 11.6 \\ 3.9x_1 - 8.7x_2 + 4.3x_3 - 6.9x_4 + 3.4x_5 &= 17.5 \end{aligned}$$

NO. EQUATIONS

5.00000

COEFFICIENTS

2.10000
-3.60000
5.80000
-7.00000
-1.20000
12.80000

3.70000
5.10000
-7.80000
-9.50000
-3.50000
9.80000

1.80000
2.20000
6.50000
3.20000
-2.50000
15.90000

-6.60000
-7.70000
-3.10000
7.40000
9.40000
11.60000

3.90000
-8.70000
4.30000
-6.90000
3.40000
17.50000

SOLUTION

-.95180
24.10513
11.94184
-10.14219
32.23400

I-5

MEMORY ALLOCATION

REGISTERS

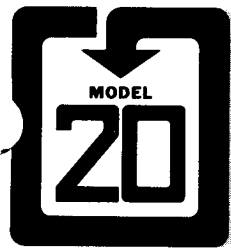
[illegible]

PROGRAM LISTING

```

0:
SPC 8;ENT "HOW M
ANY EQS.?",Z+
1:
PRT " NO. EQUAT
IONS";SPC 1;PRT
Z;SPC 2;PRT " C
DEFFICIENTS";Z+1
+A;0+B+
2:
B+1+B;0+C;SPC 1+
3:
C+1+C;ENT "A(I,J
)?";R(AB-A+C);
PRT ;JMP C=A+
4:
IF A-1=B;GTO 2+
5:
0+B+
6:
B+1+B+C;IF B=A;
GTO 24+
7:
0+X+Y+
8:
R(AC-A+B)+Z;IF 0
>Z;-Z+Z+
9:
IF X<Z;Z+X;C+Y+
10:
C+1+C;IF A>C;
GTO 8+
11:
IF X=0;SPC 1;
PRT " DET(AIJ)
=0 ";SPC 8;STP +
12:
0+C+
13:
C+1+C;IF C>A;
GTO +2+
14:
R(AY-A+C)+Z;R(AB
-A+C)+R(AY-A+C);
Z+R(AB-A+C);GTO
-1+
15:
B+C+
16:
C+1+C;IF C>A;
GTO +2+
17:
R(AB-A+C)/R(AB-A
+B)+R(AB-A+C);
GTO -1+
18:
0+C+
19:
C+1+C;IF C=A;
GTO 6+
20:
IF B=C;GTO -1+
21:
B+Y+
22:
Y+1+Y;IF Y>A;
GTO -3+
23:
R(AC-A+Y)-R(AC-A
+B)R(AB-A+Y)+R(A
C-A+Y);GTO -1+
24:
0+B;SPC 2;PRT "
SOLUTION";
SPC 1+
25:
B+1+B;IF B=A;
GTO 0+
26:
PRT R(AB);GTO -1
+
27:
END +

```

N SIMULTANEOUS LINEAR EQUATIONS WITH
REAL COEFFICIENTS IN N UNKNOWNNS, $N \leq 17$, $N \leq 36$

Model 20
MATH PAC
I-6

This program solves a system of up to seventeen simultaneous linear equations with real coefficients in seventeen unknowns (up to thirty-six with the expanded internal memory):

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

$$a_{21}x_1 + a_{22}x_2 + \dots + a_{2n}x_n = a_{2n+1}$$

$$\dots$$

$$a_{n1}x_1 + a_{n2}x_2 + \dots + a_{nn}x_n = a_{nn+1}$$

The technique used¹ is identical to the 9810A MATH PAC version² and involves processing the elements as they are entered to reduce the coefficient matrix A in the matrix equation $AX = Y$ to the identity matrix.

A system for which the determinant $\det(a_{ij}) = 0$ $i, j = 1, 2, \dots, n$ cannot be solved by the program and will cause "NOTE 10 IN 19" to be displayed, indicating an internal division by zero.

It is assumed that the equations are initially arranged so that $a_{11} \neq 0$ or the same note as above will be displayed. If $\det(a_{ij}) \neq 0$ and $a_{11} \neq 0$, there is still a remote possibility that "NOTE 10 IN 19" may be displayed because of reduction of one of the divisors to zero. In this case, a rearrangement of the original equation (still maintaining $a_{11} \neq 0$) will usually alleviate the problem.

When possible, use of Program I-5 is recommended in lieu of this one to take advantage of the pivot search for increased accuracy and to eliminate the limitation mentioned in the previous paragraph. The primary purpose of this program is to allow solving a large system of equations that could not otherwise be handled by a desktop calculator.

¹ Based on a 9100 program submitted to Hewlett-Packard by Pavo Petricevic of the Food and Agriculture Organization of the United Nations, Rome, Italy, in turn based upon his paper, "A Method for Solving Simultaneous Linear Equations on Electronic Computers with Small Core Storage", presented at the Symposium on Use of Electronic Computers, in Belgrade, December, 1964.

² Implemented by Dr. Jack Walden of the Calculator Labs at Hewlett-Packard.

ROM BLOCKS

☐

1

☐

2

☐

3

INTERNAL REGISTERS

☒ 173 ☐ 429

PERIPHERALS

☐ 9860A MARKED CARD READER☐ 9862A PLOTTER☐ 9864A DIGITIZER☐☐ 9861A TYPEWRITER☐ 9863A PAPER TAPE READER☐☐

STEP	DISPLAY	INSTRUCTIONS
1.		<u>ERASE</u> <u>LOAD</u> <u>EXECUTE</u>
2.		Alternately insert magnetic cards and <u>EXECUTE</u> until NOTE 14 no longer appears, indicating completion of loading.
3.		(Optional) <u>FIXED N n EXECUTE</u> , or <u>FLOAT N n EXECUTE</u> as desired.
4.		<u>END</u> <u>RUN PROGRAM</u>
5.	N	<u>n</u> <u>RUN PROGRAM</u>
6.	DATA	$a_{ij} \quad \text{RUN PROGRAM} \quad (\text{by rows}) \quad \begin{matrix} 1 \leq i \leq n \\ 1 \leq j \leq n+1 \end{matrix}$ <p>Processing occurs as elements are entered. A slightly longer "wait time" is required at the end of each row. After the last entry, final processing occurs and x_1, x_2, \dots, x_n are printed in order and control is returned to Step 5 for a new case.</p>
	NOTE 10 IN 19	Occurs when a division by zero is encountered. The program requires that $a_{11} \neq 0$ and $\det(a_{ij}) \neq 0 \quad i, j = 1, 2, \dots, n$. If the note still occurs, try rearranging the equations (maintaining $a_{11} \neq 0$) and return to Step 4.

EXAMPLE

$$n = 5$$

$$2.1x_1 - 3.6x_2 + 5.8x_3 - 7.0x_4 - 1.2x_5 = 12.8$$

$$3.7x_1 + 5.1x_2 - 7.8x_3 - 9.5x_4 - 3.5x_5 = 9.8$$

$$1.8x_1 + 2.2x_2 + 6.5x_3 + 3.2x_4 - 2.5x_5 = 15.9$$

$$-6.6x_1 - 7.7x_2 - 3.1x_3 + 7.4x_4 + 9.4x_5 = 11.6$$

$$3.9x_1 - 8.7x_2 + 4.3x_3 - 6.9x_4 + 3.4x_5 = 17.5$$

2.10000
-3.60000
5.80000
-7.00000
-1.20000
12.80000

3.70000
5.10000
-7.80000
-9.50000
-3.50000
9.80000

1.80000
2.20000
6.50000
3.20000
-2.50000
15.90000

-6.60000
-7.70000
-3.10000
7.40000
9.40000
11.60000

3.90000
-8.70000
4.30000
-6.90000
3.40000
17.50000

-.95180
24.10513
11.94184
-10.14219
32.23400

The diagram illustrates the Matrix Reduction Area, which is a grid of registers and flags used for matrix operations. The registers are organized into two main sections: a top section for general registers and a bottom section for flags.

Registers:

- Top Section:** Contains registers A through Z, with specific values or labels:
 - A: i
 - B: n
 - C: Elimination Counter
 - X: List Pointer
 - Y: Running Pointer 1
 - Z: Running Pointer 2
 - 0: j
 - 1: a_{ij}
 - 2: Group Counter
 - 3: Working Element
 - 4: Saved Pointer
 - 5: Coeff. Red. Starter
 - 6: (empty)
- Bottom Section:** Contains registers 0 through 15, which are part of the Flags section.

Flags:

- 0
- 1
- 2
- 3
- 4
- 5
- 6
- 7
- 8
- 9
- 10
- 11
- 12
- 13
- 14
- 15

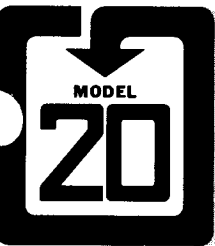
An arrow points from the "Matrix Reduction Area" label to the registers, indicating the area of operation.

PROGRAM LISTING

```

0:
6+X;0+A;ENT "N",
B+
1:
A+1+A;0+R0+
2:
ENT "DATA",R1+
3:
PRT ;SPC (R0+1+R
0)>B+
4:
IF A>R0;R1+RX;1+
X+X;JMP -2+
5:
IF R0>A;GTO 13+
6:
IF A=1;JMP 5+
7:
6+R5;GSB 38+
8:
GSB 39+
9:
GTO 11;IF A-1=C;
GSB 40+
10:
JMP -2+
11:
R1+R3+
12:
R5+1+R5;GTO 2+
13:
IF A=1;JMP 6+
14:
GSB 38+
15:
GSB 39+
16:
GTO 18;IF A-1=C;
GSB 40+
17:
JMP -2+
18:
R5+1+R5+
19:
R1/R3+RX;1+X+X;
IF R0<B;GTO 2+
20:
IF A>1;JMP 2+
21:
X+R4;GTO 1+
22:
1+C+R2;R4+A-1+R4
;6+X+Z+
23:
RZ+R3+
24:
1+Z+Z+
25:
R4+Y+
26:
RZ-R3RY+RX+
27:
1+X+X;1+Y+Y;1+Z+
Z+
28:
IF B+1-A=C;1+C+C
;JMP -2+
29:
1+C;IF R2=A-1;R2
+1+R2;GTO 23+
30:
R4+Y+
31:
RY+RX+
32:
1+X+X;1+Y+Y+
33:
IF B+1-A=C;1+C+C
;JMP -2+
34:
IF B>A;GTO 21+
35:
6+X;SPC 2+
36:
PRT RX;JMP (1+X+
X)=B+6+
37:
SPC 8;GTO 0+
38:
1+C;R5+Y;R4+Z;
RET +
39:
R1-RZRY+R1;RET +
40:
1+Z+Z;1+C+C;B+2-
A+Y+Y;RET +
41:
END +

```

N SIMULTANEOUS LINEAR EQUATIONS WITH
COMPLEX COEFFICIENTS IN N UNKNOWN, $N \leq 4$, $N \leq 11$

Model 20
MATH PAC
I-7

This program solves a system of up to four simultaneous linear equations with complex coefficients in four unknowns (up to eleven with the expanded internal memory):

$$c_{11}Z_1 + c_{12}Z_2 + \dots + c_{1n}Z_n = c_{1\ n+1}$$

$$c_{21}Z_1 + c_{22}Z_2 + \dots + c_{2n}Z_n = c_{2\ n+1}$$

$$\dots$$

$$c_{n1}Z_1 + c_{n2}Z_2 + \dots + c_{nn}Z_n = c_{n\ n+1}$$

It is assumed that the determinant $\det(c_{ij}) \neq 0$, $i, j = 1, 2, \dots, n$. If this is not the case, the message "DET C(IJ)=0" is printed and displayed and the program is terminated.

The program uses a modified Gauss-Jordan method. A row pivot search is performed to maximize the accuracy of the results.

ROM BLOCKS

1

2

3

INTERNAL REGISTERS

☒ 173 ☐ 429

PERIPHERALS

☐ 9860A MARKED CARD READER☐ 9862A PLOTTER☐ 9864A DIGITIZER☐☐ 9861A TYPEWRITER☐ 9863A PAPER TAPE READER☐☐

STEP	DISPLAY	INSTRUCTIONS
1.		<u>ERASE</u> <u>LOAD</u> <u>EXECUTE</u>
2.		Alternately insert magnetic cards and <u>EXECUTE</u> until NOTE 14 no longer appears, indicating completion of loading.
3.		(Optional) <u>FIXED N n EXECUTE</u> , or <u>FLOAT N n EXECUTE</u> as desired.
4.		<u>END</u> <u>RUN PROGRAM</u>
5.	HOW MANY EQS.?	<u>n</u> <u>RUN PROGRAM</u>
6.	RL C(IJ)? IM C(IJ)?	$\left. \begin{array}{l} \text{Rl } c_{ij} \text{ } \underline{\text{RUN PROGRAM}} \\ \text{Im } c_{ij} \text{ } \underline{\text{RUN PROGRAM}} \end{array} \right\} \begin{array}{l} \text{(by rows)} \\ 1 \leq j \leq n+1 \end{array}$ <p>After all data is entered, calculations proceed automatically. The answers are printed in order: z_1, z_2, \dots, z_n (real part of z_i first, imaginary part of z_i below it). Return to Step 5 for a new case.</p>
	DET C(IJ)=0	This message will be displayed and printed when the coefficient matrix on the left-hand side of the equation has zero determinant. The system of equations cannot be processed. For a new case, return to Step 4.

* If $\text{Im } c_{ij} = 0$, RUN PROGRAM alone will cause the program to automatically enter a zero, saving the need to press the "0" key.

EXAMPLE

Solve the following system of three equations in three unknowns:

$$(2 + 8i)z_1 + (-3 + 4i)z_2 + (4 + 6i)z_3 = 0 - 5i$$

$$(7 + i)z_1 + (9 - 8i)z_2 + (3 + 2i)z_3 = 5$$

$$6z_1 + (8 + 7i)z_2 + (1 + i)z_3 = 4 - 5i$$

NO. EQUATIONS

3.00000

COEFFICIENTS

2.00000

8.00000

-3.00000

4.00000

4.00000

6.00000

0.00000

-5.00000

7.00000

1.00000

9.00000

-8.00000

3.00000

2.00000

5.00000

0.00000

6.00000

0.00000

8.00000

7.00000

1.00000

1.00000

4.00000

-5.00000

SOLUTION

1.80375

R1 z₁

-.10366

Im z₁

-.45468

R1 z₂

.22895

Im z₂

-2.30466

R1 z₃

-.72228

Im z₃

MEMORY ALLOCATION

REGISTERS							
A	n + 1						
B	Counter i						
C	Counter j						
X	Max. Row, Temp.						
Y	Counter						
Z	Temporary						
0	Unused						
1	Rl c ₁₁						
2	Rl c ₁₂						
	.						
	.						
	.						
n+1	Rl c _{1 n+1}						
n+2	Rl c ₂₁						
	.						
	.						
	.						
n²+n+1	Im c ₁₁						
n²+n+2	Im c ₁₂						
	.						
	.						
	.						
n²+2n+1	Im c _{1 n+1}						
n²+2n+2	Im c ₂₁						
	.						
	.						
	.						
							FLAGS
						0	
						1	
						2	
						3	
						4	
						5	
						6	
						7	
						8	
						9	
						10	
						11	
						12	
						13	
						14	
						15	

PROGRAM LISTING

```

0:
SPC 8:ENT "HOW M
ANY EQS.?",Z:
1:
PRT " NO. EQUAT
IONS":SPC 1:PRT
Z:SPC 2:PRT " C
OEFFICIENTS":Z+1
+A:0+B:
2:
B+1+B:0+C:SPC 1:
3:
C+1+C:ENT "RL C(
IJ)?":R(AB-A+C):
PRT :
4:
0+X:ENT "IM C(IJ
)?":X:PRT X+R(AB
+C+AA-2A):SPC 1:
IF C=A:GTO -1:
5:
IF A-1=B:SPC 1:
GTO 2:
6:
0+B:
7:
B+1+B+C:IF B=A:
GTO 32:
8:
0+X+Y:
9:
AC-A+B+Z:RZRZ+R(
Z-A+AA+Z)RZ+Z:
10:
IF X<Z:Z+X:C+Y:
11:
IF A>(C+1+C):
JMP -2:
12:
IF X=0:SPC 1:
PRT " DET C(IJ)
=0":SPC 8:STP :
13:
0+C:
14:
IF (C+1+C)>A:
JMP 3:
15:
R(AY+C+AA-2A)+Z:
R(AB+C+AA-2A)+R(
AY+C+AA-2A):Z+R(
AB+C+AA-2A):

```

```

16:
R(AY-A+C)+Z:R(AB
-A+C)+R(AY-A+C):
Z+R(AB-A+C):JMP
-2:
17:
B+C:
18:
AB-A+B+Z:RZRZ+R(
Z-A+AA+Z)RZ+Z:
19:
IF (C+1+C)>A:
JMP 4:
20:
(R(AB+C+AA-2A)R(
AB-A+B)-R(AB+B+A
A-2A)R(AB-A+C))/
Z+X:
21:
(R(AB-A+C)R(AB-A
+B)+R(AB+C+AA-2A
)R(AB+B+AA-2A))/
Z+R(AB-A+C):
22:
X+R(AB+C+AA-2A):
JMP -3:
23:
0+C:
24:
IF (C+1+C)=A:
GTO 7:
25:
IF B=C:JMP -1:
26:
B+Y:
27:
IF (Y+1+Y)>A:
JMP -3:
28:
R(AC+Y+AA-2A)-R(
AC+B+AA-2A)R(AB-
A+Y)-R(AB+Y+AA-2
A)R(AC-A+B)+Z:
29:
Z+R(AC+Y+AA-2A):
30:
R(AC-A+Y)+R(AC+B
+AA-2A)R(AB+Y+AA
-2A)-R(AC-A+B)R(
AB-A+Y)+R(AC-A+Y
):
31:
JMP -4:

```

```

32:
0+B:SPC 2:PRT "
SOLUTION":
SPC 1:
33:
IF (B+1+B)=A:
GTO 0:
34:
PRT R(AB),R(AB-A
+AA):SPC 1:JMP -
1:
35:
END :

```




Model 20
MATH PAC
I-8

CHARACTERISTIC EQUATION

This program uses the method of Danilevsky to determine the coefficients in the characteristic equation $\det (A - \lambda I) = 0$ for a given $n \times n$ matrix A with real entries, $n \leq 9$ (respectively, $n \leq 17$ with a 9820A having the additional 256 internal registers). Since this is a polynomial equation in λ , Program II-2 may then be used to find the roots (λ) which are the eigenvalues of A .

A pivotal condensation scheme is incorporated in the program which increases the accuracy of the results.

Reference:

V. N. Faddeeva, Computational Methods of Linear Algebra (New York: Dover Publications, Inc., 1959), pp 166-177.

ROM BLOCKS

1

2

3

INTERNAL REGISTERS

☒ 173 ☐ 429

PERIPHERALS

☐ 9860A MARKED CARD READER☐ 9862A PLOTTER☐ 9864A DIGITIZER☐☐ 9861A TYPEWRITER☐ 9863A PAPER TAPE READER☐☐

STEP	DISPLAY	INSTRUCTIONS
1.		<u>ERASE</u> <u>LOAD</u> <u>EXECUTE</u>
2.		Alternately insert magnetic cards and <u>EXECUTE</u> until NOTE 14 no longer appears, indicating completion of loading.
3.		(Optional) <u>FIXED N n EXECUTE</u> , or <u>FLOAT N n EXECUTE</u> as desired.
4.		<u>END</u> <u>RUN PROGRAM</u>
5. N?		<u>n</u> <u>RUN PROGRAM</u>
6. DATA		a_{ij} <u>RUN PROGRAM</u> (by rows) $1 \leq i, j \leq n$ After all elements of the matrix A have been entered, calculation of the coefficients of the characteristic polynomial proceeds automatically. Actually, partial or complete factoring of the characteristic polynomial may occur during the calculations. When this happens, the coefficients of each polynomial in the factorization will comprise the output rather than the coefficients of their product (i. e. , the characteristic polynomial). A space will separate the sets of coefficients, and the first coefficient in each set is always 1.

EXAMPLES

Because of the inherent nature of the algorithm used, the characteristic polynomial is sometimes output in partially or fully factored form as shown in the second example.

$$1. \quad A = \begin{bmatrix} 1 & 2 \\ 3 & 2 \end{bmatrix}$$

$$0 = \det(A - \lambda I) = \det \begin{bmatrix} 1-\lambda & 2 \\ 3 & 2-\lambda \end{bmatrix} = \lambda^2 - 3\lambda - 4$$

The values printed out are 1, -3, -4, and no factoring occurs.

MATRIX

1.0000
2.0000

3.0000
2.0000

COEFFICIENTS

1.0000
-3.0000
-4.0000

$$2. \quad A = \begin{bmatrix} 3 & 0 & 0 \\ 0 & 2 & -5 \\ 0 & 1 & -2 \end{bmatrix}$$

$$0 = \det(A - \lambda I) = (\lambda^2 + 0\lambda + 1)(\lambda - 3)$$

In this example, the answer appears in partially factored form.

MATRIX

3.0000
0.0000
0.0000

0.0000
2.0000
-5.0000

0.0000
1.0000
-2.0000

COEFFICIENTS

1.0000
0.0000
1.0000

1.0000
-3.0000

MEMORY ALLOCATION

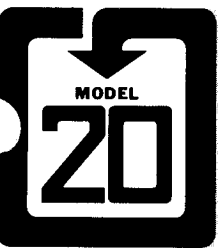
REGISTERS			
A	n		
B	i, Counter		
C	j, Counter		
X	Remaining Rows		
Y	Temporary, Counter		
Z	Max. Elt. , Sum		
0	Column, v		
1	a ₁₁		
2	a ₁₂		
	.		
	.		
	.		
n	a _{1n}		
n+1	a ₂₁		
	.		
	.		
	.		
		FLAGS	
		0	
		1	
		2	
		3	
		4	
		5	
		6	
		7	
		8	
		9	
		10	
		11	
		12	
		13	
		14	
		15	

PROGRAM LISTING

```

0:
ENT "N?",A+
1:
PRT "      MATRIX
"10+B+
2:
SPC 1;B+1+B+
3:
0+C+
4:
ENT "DATA",R(C+1
+C+AB-A);PRT ;
JMP A=C+
5:
IF A=B;JMP -3+
6:
SPC 2;PRT " COE
FFICIENTS";SPC 1
;A+X+
7:
X+B+
8:
0+C+Z+
9:
IF (C+1+C)>B-1;
JMP 3+
10:
IF Z>(R(AB-A+C)+
Y)Y;JMP -1+
11:
YY+Z;C+R0;JMP -2
+
12:
IF Z=0;B+C;JMP 1
+
13:
1+Y;IF R0=B-1;
JMP 4+
14:
R(AY-A+R0)+Z;R(A
Y-A+B-1)+R(AY-A+
R0);Z+R(AY-A+B-1
);JMP (Y+1+Y)>B+
15:
1+Y+
16:
R(AR0-A+Y)+Z;R(A
B-2A+Y)+R(AR0-A+
Y);Z+R(AB-2A+Y);
JMP (Y+1+Y)>X+
17:
0+C+
18:
IF (C+1+C)>X;
JMP 8+
19:
0+Y+Z+
20:
IF (Y+1+Y)>B-1;
JMP 4+
21:
IF C=1;R(AY-A+B-
1)/R((B-1)(A+1))
+R(AY-A+B-1)+
22:
IF C=B-1;R(AY-A+
C)-R(AY-A+B-1)R(
AB-A+C)+R(AY-A+C
)+
23:
Z+R(AY-A+C)R(AB-
A+Y)+Z;JMP -3+
24:
IF C=B-1;Z+R0;
JMP -6+
25:
Z+R(AB-2A+C);
JMP -7+
26:
R0+R(AB-2A+B-1)+
27:
0+Y+
28:
AB-2A+B-1+Y+Z;RZ
+R(AB-A+B+Y)+RZ;
JMP (Y+1+Y)>X-B+
29:
IF (B-1+B)>1;
GTO 8+
30:
1+C+
31:
PRT 1+
32:
C+Y+
33:
PRT -R(AC-A+Y);
JMP (Y+1+Y)>X+
34:
SPC 1;IF C=1;
SPC 8;GTO 0+
35:
GTO 7;IF (B-1+X)
=1;JMP -5+
36:
END +

```

EIGENVALUES OF A MATRIX WITH REAL ENTRIES

For a given $n \times n$ matrix $A = (a_{ij})$, a_{ij} real, $n \leq 8$, this program will find the eigenvalues of A . If during the internal processing no factorization of the characteristic polynomial of A occurs, an eigenvector corresponding to each eigenvalue will also be printed. It is expected that for almost all practical applications, the eigenvectors will be obtained.

The method of Danilevsky¹ is used to obtain the characteristic polynomial of A which is then factored by the Newton-Raphson method² to get the roots of the characteristic equation which are the eigenvalues of A . Three check values are printed after the eigenvalues and give a measure of the accuracy of the results as the check values should all have the same value. If no factorization of the characteristic polynomial occurs, eigenvectors will be printed next, one for each of the preceding eigenvalues. The algorithm is such that the trailing coefficient of each eigenvector is normalized to 1.

References:

¹ V. N. Faddeeva, Computational Methods of Linear Algebra, (New York: Dover Publications, Inc., 1959), pp. 166-176.

² John M. McCormick, Mario G. Salvadori, Numerical Methods in FORTRAN, (New Jersey: Prentice-Hall, Inc., 1964), p. 196.

ROM BLOCKS

1

2

3

INTERNAL REGISTERS

☐ 173 ☒ 429

PERIPHERALS

☐ 9860A MARKED CARD READER☐ 9862A PLOTTER☐ 9864A DIGITIZER☐☐ 9861A TYPEWRITER☐ 9863A PAPER TAPE READER☐☐

STEP	DISPLAY	INSTRUCTIONS
1.		<u>ERASE</u> <u>LOAD</u> <u>EXECUTE</u>
2.		Alternately insert magnetic cards and <u>EXECUTE</u> until NOTE 14 is no longer displayed, indicating completion of loading.
3.		(Optional) <u>FIXED N n EXECUTE</u> , or <u>FLOAT N n EXECUTE</u>
4.		<u>END</u> <u>RUN PROGRAM</u>
5.	MATRIX SIZE?	<u>n RUN PROGRAM</u> for an $n \times n$ matrix
6.	ENTER DATA	<u>a_{ij} RUN PROGRAM</u> row by row
7.		<p>After all matrix elements have been entered, the display blanks while calculations are made. The eigenvalues are printed, real part first with imaginary part under it, followed by the three check values SUM (= sum of the eigenvalues), P1 (= coefficient of λ^{n-1} in the characteristic equation), and TRACE (= trace, i. e., sum of the diagonal elements, of the original matrix), all of which should be approximately equal in value. If no factorization of the characteristic polynomial occurs internally during the calculations, an eigenvector corresponding to each eigenvalue will also be printed, and in the same order (i. e., the first eigenvector printed corresponds to the first eigenvalue printed, etc.). If $Z = (z_1, z_2, \dots, z_n)$ is an eigenvector, z_1 is printed first, then z_2, etc., and z_n will have the value 1. The real part of z_i is printed with the imaginary part of z_i under it. In any case, control returns to Step 5 for a new problem.</p> <p>Occasionally the root finding routine is unable to converge. Changing the convergence tolerance $1E-9$ in line 52 may or may not enable convergence. In some cases, tightening this tolerance will give results with improved accuracy. Whatever value is used, the time to find a root should not exceed about two minutes.</p>

EXAMPLE

Find the eigenvalues (and some corresponding eigenvectors) for the matrix

$$A = \begin{pmatrix} 4 & 2 & 7 \\ 0 & 6 & 5 \\ 1 & 8 & 3 \end{pmatrix}$$

MATRIX

```

4.000000
2.000000
7.000000

0.000000
6.000000
5.000000

1.000000
8.000000
3.000000

```

EIGENVALUES

```

4.08151
.000000

-2.56114
-.000000

11.47963
.000000

```

1st eigenvalue λ_1 2nd eigenvalue λ_2 3rd eigenvalue λ_3

```

SUM
13.000000
.000000

```

```

P1
13.000000

```

Check
numbers

```

TRACE
13.000000

```

EIGENVECTORS

```

21.93122
.000000

-2.60621
-.000000

1.000000
0.000000

```

 z_1 z_2 z_3 an eigenvector for
the 1st eigenvalue

For

$$Z = \begin{pmatrix} z_1 \\ z_2 \\ z_3 \end{pmatrix},$$

the relation

$$AZ = \lambda_i Z \text{ holds,}$$

where λ_i is one of the three eigenvalues and Z is the corresponding eigenvector.

```

-.88886
-.000000

```

 z_1

```

-.58403
-.000000

```

 z_2 an eigenvector for
the 2nd eigenvalue

```

1.000000
0.000000

```

 z_3

```

1.17986
.000000

```

 z_1

```

.91247
.000000

```

 z_2 an eigenvector for
the 3rd eigenvalue

```

1.000000
0.000000

```

 z_3

REGISTERS	
A	N
B	I, M
C	J, J, K
X	R
Y	Temporary, I, L
Z	Maximum, S
0	BR base
1	BI base
2	CR base
3	CI base
4	ADR1, SUM _{real}
5	ADR2, SUM _{imaginary}
6	COL. , V, X
7	Y
8	I
9	N
10	DEN
11	AR base
12	AI base
13	P1
14	Trace
15	$\Sigma\lambda_i$ real
16	$\Sigma\lambda_i$ imaginary
17	Unused
18	Unused
19	NROOT
20	-1
<div>Matrix</div> <div>20+nn</div> <div>Calculation area, roots, and location vector</div>	
FLAGS	
0	
1	Used
2	
3	
4	
5	
6	
7	
8	
9	
10	
11	
12	
13	
14	
15	

PROGRAM LISTING

```

0:
ENT "MATRIX SIZE
?" ; A←
1:
PRT "      MATRIX
"; 0←B←R13←R14←R1
5←R16; CFG 1←
2:
SPC 1; B+1←B←
3:
0←C←
4:
ENT "ENTER DATA"
; R(C+1←C+AB-A+20
)←Z; PRT ; Z(B=C)+
R14←R14; IF A≠C;
JMP 0←
5:
IF A≠B; JMP -3←
6:
A←X; SPC 2; PRT , "
EIGENVALUES"←
7:
X←B←
8:
0←C←Z←
9:
IF (C+1←C) > B-1;
JMP 3←
10:
IF Z > (R(AB-A+C+2
0)←Y) Y; JMP -1←
11:
YY←Z; C←R6; JMP -2
←
12:
R6←R(AA+7A+24+B)
; IF Z=0; SFG 1; B←
C; GTO 35←
13:
1←Y; IF R6=B-1;
JMP 4←
14:
R(AY-A+R6+20←R4)
←Z; R(AY-A+B+19←R
5)←RR4; Z←RR5;
JMP (Y+1←Y) > B←
15:
1←Y←
16:
R(AR6-A+Y+20←R4)
←Z; R(AB-2A+Y+20←
R5)←RR4; Z←RR5;
JMP (Y+1←Y) > X←

```

```

17:
0←C←
18:
IF (C+1←C) > X;
JMP 8←
19:
0←Y←Z←
20:
IF (Y+1←Y) > B-1;
JMP 4←
21:
AY-A+B+19←R4; IF
C=1; RR4←R(AB-A+B
+19)←RR4←
22:
AY-A+C+20←R5; IF
C≠B-1; RR5←RR4R(A
B-A+C+20)←RR5←
23:
Z←RR5R(AB-A+Y+20
)←Z; JMP -3←
24:
IF C=B-1; Z←R6;
JMP -6←
25:
Z←R(AB-2A+C+20);
JMP -7←
26:
R6←R(AB-2A+B+19←
Z)←
27:
0←Y←
28:
RZ←R(Z+A+1)←RZ; Z
+1←Z; JMP (Y+1←Y)
> X←B←
29:
1←Y; AB-A+B+19←C←
30:
IF Y≠B-1; AB-A+Y+
20←Z; -RZ/RC←RZ←
31:
IF (Y+1←Y) ≤ X;
GTO -1←
32:
1/RC←R←
33:
IF (B-1←B) > 1;
GTO 8←
34:
1←C←

```

```

35:
-1←R(AC-A+C+19←R
11); (R(R11+1)←R6
)←R13←R13; 0←R7←
36:
IF ((X-C+1←R9)+1
←Z)=2; GTO 56←
37:
(((AA+21←R12)+Z
←R0)+Z←R1)+Z←R2)
+Z←R3←
38:
0←Z←RR1←RR3←R19;
-1←RR0←RR2←
39:
0←R(Z+R12); JMP (
Z+1←Z) > R9←
40:
0←R6; IF ((1←R7)+
R19←R19) > X-C+1;
GTO 58←
41:
1←Y←
42:
R(R11+Y)+(R((R0+
Y←Z)-1)←R4)R6-(R
((R1+Y←R8)-1)←R5
)R7←RZ←
43:
R(R12+Y)+R4R7+R5
R6←RR8←
44:
IF (Y+1←Y) ≤ R9;
GTO -2←
45:
1←Y←
46:
R(R0+Y)+(R((R2+Y
←Z)-1)←R4)R6-(R(
(R3+Y←R8)-1)←R5)
R7←RZ←
47:
R(R1+Y)+R4R7+R5R
6←RR8←
48:
IF R9 > (Y+1←Y);
GTO -2←
49:
(R(R2+R9-1)←Z)+
(R(R3+R9-1)←Y)Y←
R10←

```

PROGRAM LISTING

```

50:
R6=((R(R0+R9)+R8
)Z+(R(R1+R9)+R4)
Y)/R10+R6+
51:
R7+((R8Y-R4Z)/R10
+R7+
52:
IF R8R8+R4R4>1E-
9;GTO 41+
53:
R9-1+R9+
54:
1+Y+
55:
R(R0+Y)+R(R11+Y)
+R(R1+Y)+R(R12+Y)
;JMP (Y+1+Y)>R9
+
56:
SPC 1;PRT R6,R7+
57:
(R6+R((R3+A+2R19
+Z)-1))+R15+R15;
(R7+RZ)+R16+R16;
GTO 40+
58:
IF C=1;SPC 1;
JMP 3+
59:
IF (B-1+X)=1;
GTO 34+
60:
GTO 7+
61:
PRT "          SUM"
,R15,R16,"",
      P1",R13,"",
      TRACE",R14
;SPC 2+
62:
IF FLG 1(A=1);
GTO 0+
63:
PRT "    EIGENVECT
ORS";SPC 1+
64:
0+0+
65:
IF (C+1+C)>A;
GTO 0+
66:
0+RR0+Y;1+RR1+

```

```

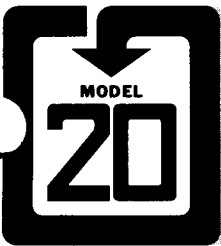
67:
IF (Y+1+Y)=A;
GTO 44+
68:
R((R3+A+2C+Z)-1)
+X+
69:
(R((R1-Y+R4)+1)+
B)X-(R((R0-Y+R5)
+1)+R6)RZ+RR4+
70:
BRZ+R6X+RR5;GTO
-3+
71:
A+Y+
72:
0+R4+R5+X+
73:
IF (X+1+X)>A;
JMP 3+
74:
(R(A(A-Y+1)+X+20
)+Z)R(R1-A+X)+R4
+R4+
75:
ZR(R0-A+X)+R5+R5
;GTO -2+
76:
IF (Y-1+Y)=0;
JMP 5+
77:
R4+R(R1-Y);R5+R(
R0-Y)+
78:
IF (R(AA+8A-Y+25
)-A+Z)=-Y;GTO 72
+
79:
R(R1+Z)+R(R1-Y);
R4+R(R1+Z)+
80:
R(R0+Z)+R(R0-Y);
R5+R(R0+Z);GTO 7
2+
81:
A+Y+
82:
IF 0<(Y-1+Y);
PRT R(R1-Y),R(R0
-Y);SPC 1;JMP 0+
83:
PRT "          -----
";SPC 1;GTO 65+

```

```

84:
END +

```

POLYNOMIAL ARITHMETIC

With this program, two polynomials $P(X)$ and $Q(X)$ with real coefficients may be added, subtracted, multiplied or divided. For the basic 9820A the size constraints are:

Addition: $\deg P(X) + \deg Q(X) + \max(\deg P(X), \deg Q(X)) \leq 55$

Subtraction: $\deg P(X) + \deg Q(X) + \max(\deg P(X), \deg Q(X)) \leq 55$

Multiplication: $\deg P(X) + \deg Q(X) \leq 27$

Division: $\deg P(X) \leq 27$

For example, two polynomials of degree 18 may be added or subtracted and two polynomials of degree 13 may be multiplied. Any polynomial $P(X)$ of degree ≤ 27 may be divided by a non-zero polynomial of the same or lesser degree; attempt to divide by a polynomial of greater degree causes the message "deg Q > deg P" to be displayed.

ROM BLOCKS

1

2

3

INTERNAL REGISTERS

☒ 173 ☐ 429

PERIPHERALS

☐ 9860A MARKED CARD READER☐ 9862A PLOTTER☐ 9864A DIGITIZER☐☐ 9861A TYPEWRITER☐ 9863A PAPER TAPE READER☐☐

STEP	DISPLAY	INSTRUCTIONS
1.		<u>ERASE</u> <u>LOAD</u> <u>EXECUTE</u>
2.		Alternately insert magnetic cards and <u>EXECUTE</u> until NOTE 14 is no longer displayed, indicating completion of loading.
3.		(Optional) <u>FIXED N n EXECUTE</u> , or <u>FLOAT N n EXECUTE</u> as desired.
4.		<u>END</u>
5.		<u>RUN PROGRAM</u>
6.	P(X) COEFF'S?	<u>a_i RUN PROGRAM</u> for $i=0,1,\dots,n$ and $P(X)=a_0+a_1X+\dots+a_nX^n$. <u>RUN PROGRAM</u> after entering all a_i .
7.	Q(X) COEFF'S?	<u>b_i RUN PROGRAM</u> for $i=0,1,\dots,m$ and $Q(X)=b_0+b_1X+\dots+b_mX^m$. <u>RUN PROGRAM</u> after entering all b_i .
8.	P(X)*Q(X)? P(X)/Q(X)? P(X)+Q(X)? P(X)-Q(X)?	<u>RUN PROGRAM</u> to cycle between these four displays; <u>1 RUN PROGRAM</u> to select the option being displayed.
9.		The coefficients of the resultant polynomial $H(X)=h_0+h_1X+\dots+h_kX^k$ and, in the case of division, the remainder polynomial $R(X)=r_0+r_1X+\dots+r_lX^l$, are printed out in order: <div style="text-align: right;"> h_0 h_1 \vdots h_k r_0 r_1 \vdots r_l </div>
For a new case, return to Step 5.		

EXAMPLES

$$P(X) = 4 + 1X + 3X^2$$

$$Q(X) = 2 + 7X$$

INPUT

(Must be reentered for each case below)

P(X)
COEFFICIENTS

4.0000	a_0
1.0000	a_1
3.0000	a_2

Q(X)
COEFFICIENTS

2.0000	b_0
7.0000	b_1

OUTPUT

Addition

$$H(X) = P(X) + Q(X)$$

H(X)
COEFFICIENTS

6.0000	h_0
8.0000	h_1
3.0000	h_2

Subtraction

$$H(X) = P(X) - Q(X)$$

H(X)
COEFFICIENTS

2.0000	h_0
-6.0000	h_1
3.0000	h_2

Multiplication

$$H(X) = P(X) * Q(X)$$

H(X)
COEFFICIENTS

8.0000	h_0
38.0000	h_1
13.0000	h_2
21.0000	h_3

Division

$$P = QH + R$$

H(X)
COEFFICIENTS

.0204	h_0
.4286	h_1

R(X)
COEFFICIENTS

3.9592	r_0
--------	-------

[illegible]

PROGRAM LISTING

```

0:
0+X:CFG 13:PRT "
      P(X)":GSB
19:
1:
ENT "P(X) COEFF"
S?":R(X+1+X):IF
FLG 13=0:PRT ;
JMP 0+
2:
0+Y:CFG 13:SPC 2
:PRT "      Q(X)
":GSB 19+
3:
ENT "Q(X) COEFF"
S?":R(Y+1+Y+X):IF
FLG 13=0:PRT
:JMP 0+
4:
CFG 13:0+A:ENT "
P(X)*Q(X)?":Z:
IF FLG 13=0:SPC
3:PRT " H(X)=P(X)
)*Q(X)":JMP 8+
5:
CFG 13:CFG 1:
ENT "P(X)/Q(X)?"
,Z:IF FLG 13=0:
SFG 1:JMP 15+
6:
CFG 13:ENT "P(X)
+Q(X)?":Z:1+Z:
IF FLG 13=0:SPC
3:PRT " H(X)=P(X)
)+Q(X)":JMP 3+
7:
CFG 13:ENT "P(X)
-Q(X)?":Z:-1+Z:
IF FLG 13:JMP -3
+
8:
SPC 3:PRT " H(X)
=P(X)-Q(X)"
9:
IF Y>(X+C):Y+CH
10:
IF (A+1+A)>C:
JMP 18+
11:
(A<Y)(A<X)((ZR(A
+X)+B)+RA)+(A>Y)
RA+(A>X)B+R(A+X+
Y):GTO 10+

```

```

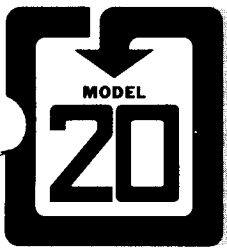
12:
X+Y-1+CH
13:
0+R((A+1+A)+C+1)
:JMP A=CH
14:
0+AH
15:
IF (A+1+A)>X:
JMP 13+
16:
0+B+
17:
IF (B+1+B)>Y:
JMP -2+
18:
A+B+C+Z:RAR(B+X)
+RZ+RZ:GTO 17+
19:
PRT "      COEFFICIE
NTS":SPC 1:RET +
20:
IF (X-Y+1+B)<0:
DSP "DEG Q>DEG P
":JMP 14+
21:
SPC 3:PRT "
P=QH+R"
22:
Y-1+Z+
23:
R(Z+B)/R(X+Y)+R(
X+Y+B):0+AH
24:
A+B+C:IF (A+1+A)
<Z:RC-R(X+Y+B)R(
X+A)+RC:JMP 0+
25:
IF (B-1+B)>0:
JMP -2+
26:
IF (Z>0)(RZ=0):Z
-1+Z:JMP 0+
27:
X-Y+1+CH
28:
1+A:SPC 3:PRT "
      H(X)":GSB 1
9+
29:
PRT R(X+Y+A):
JMP (A+1+A)>CH

```

```

30:
IF FLG 1=0:JMP 4
+
31:
1+A:SPC 2:PRT "
      R(X)":GSB 1
9+
32:
IF Z=0:PRT Z:
JMP 2+
33:
PRT RA:JMP (A+1+
A)>Z+
34:
SPC 8+
35:
END +

```

POLYNOMIAL ROOT FINDER

This program* finds all roots for polynomials of the form $a_0 + ib_0 + (a_1 + ib_1)z + (a_2 + ib_2)z^2 + \dots + (a_n + ib_n)z^n = 0$ for $n \leq 8$ on the basic 9820A and $n \leq 72$ on the expanded 9820A.

Roots are found using the method of steepest descent detailed in the reference. Once a root is found, the polynomial is reduced by synthetic division and another root is extracted. The final root is determined algebraically. The algorithm is very accurate and very stable; it will virtually always find the roots. Multiple roots will be found at some slightly reduced accuracy (see EXAMPLES), and because of the synthetic division, very high order polynomials may show some loss in accuracy as more roots are found; but, in general, the program will find "normally" spaced roots accurate to better than 6 decimal places.

* Written by Larry Choice of Burr-Brown Research Corporation, Tucson, Arizona, with some step-saving modifications by Hewlett-Packard to allow higher order polynomials. The program is based on a FORTRAN program written by Dr. Jack Walden of Hewlett-Packard Calculator Labs, using the algorithm in the following reference:

Reference: J. B. Moore, "A Convergent Algorithm for Solving Polynomial Equations", Journal of the Association for Computing Machinery, Volume 14, Number 2 (April, 1967).

REMARKS

1. Two convergence factors are used in the program (in lines 20 and 24) but experience has shown that these should not be changed; doing so has either reduced the accuracy or increased the computation time. The values used seem to be optimum for a 12 digit floating point machine. If after 50 iterations (line 21 of the program) no root is found, the value 88 is printed and processing halts. The user may press 2 5 \longrightarrow R 4 RUN PROGRAM and obtain 25 more iterations. The other convergence factor is associated with too large a step size in the direction of a root. If 10 successive quarterings of the step size (line 23 of the program) does not adequately reduce the step size, the value 99 is printed and processing halts. This would be a very rare occurrence.

2. Execution time varies with the order of the polynomial. For low order polynomials ($n \leq 5$) about 10 to 15 seconds are required to find each root. A higher order polynomial such as $x^{60} + 1 = 0$ will require about 5 minutes for the first roots, and this time decreases as the order of the polynomial is reduced by the synthetic division. Repeated roots and closely spaced roots take longer to find.

3. The coefficients a_i and b_i , $i = 0, \dots, n$ are stored according to the formulas:

$$\begin{aligned} a_i &\longrightarrow R(8+i) \\ b_i &\longrightarrow R(9 + n+i) \end{aligned} \quad i = 0, 1, \dots, n$$

To correct a coefficient after it has been entered, use one of the above formulas to locate the appropriate "R" register and store the correct value of the coefficient, e.g., a₁ \longrightarrow R 9 EXECUTE and then continue entering the rest of the coefficients in the normal manner following the User Instructions.

ROM BLOCKS

1	2	3

INTERNAL REGISTERS

☒ 173 ☐ 429

PERIPHERALS

<input type="checkbox"/> 9860A MARKED CARD READER	<input type="checkbox"/> 9862A PLOTTER	<input type="checkbox"/> 9864A DIGITIZER	<input type="checkbox"/> _____
<input type="checkbox"/> 9861A TYPEWRITER	<input type="checkbox"/> 9863A PAPER TAPE READER	<input type="checkbox"/> _____	<input type="checkbox"/> _____

STEP	DISPLAY	INSTRUCTIONS
1.		<u>ERASE</u> <u>LOAD</u> <u>EXECUTE</u>
2.		Alternately insert magnetic cards and <u>EXECUTE</u> until NOTE 14 no longer appears, indicating completion of loading.
3.		(Optional) <u>FIXED N n EXECUTE</u> , or <u>FLOAT N n EXECUTE</u> as desired.
4.		<u>END</u> <u>RUN PROGRAM</u>
5.	N	<u>n</u> <u>RUN PROGRAM</u>
6.	A	$\left. \begin{array}{l} \underline{a_k} \text{ RUN PROGRAM} \\ \underline{b_k} \text{ RUN PROGRAM} \end{array} \right\} \begin{array}{l} \text{for } k = 0, 1, \dots, n \text{ and} \\ P(z) = a_0 + ib_0 + (a_1 + ib_1)z + \dots + (a_n + ib_n)z^n \end{array}$
7.	B	
		After all coefficients have been entered, the calculator stops with the value $n + 1$ in the display. The purpose of this stop is to allow the input data to be checked and manually corrected by overstoreing the correct value in the proper register prior to going to Step 9. Refer to the REMARKS section for locating registers.
9.		<u>RUN PROGRAM</u>
		The roots are printed, real part first, imaginary part under it, and numbered using a counter that goes from n to 1. For a new case, return to Step 4.
<p>*NOTE: If $b_k = 0$, <u>RUN PROGRAM</u> alone will cause a 0 to be entered automatically for b_k, saving an extra key stroke.</p>		

EXAMPLES

1. Find the roots of

$$P(z) = (0 + 2i) + (2 - 3i)z + (-3 + 1i)z^2 + (1 + 2i)z^3 + (2 - 3i)z^4 + (-3 + 1i)z^5 + (1 + 0i)z^6$$

0.000000
0.000000
2.000000

1.000000
2.000000
-3.000000

2.000000
-3.000000
1.000000

3.000000
1.000000
2.000000

4.000000
2.000000
-3.000000

5.000000
-3.000000
1.000000

6.000000
1.000000
0.000000

Roots:

6.000000
.500000
.86603

Real Part
Imaginary Part

5.000000
.500000
-.86603

4.000000
1.000000
.000000

3.000000
.000000
-1.000000

2.000000
-1.000000
-.000000

1.000000
2.000000
.000000

2. Find the roots of

$$P(z) = 105 + 105z + 45z^2 + 10z^3 + z^4$$

0.000000
105.000000
0.000000

1.000000
105.000000
0.000000

2.000000
45.000000
0.000000

3.000000
10.000000
0.000000

4.000000
1.000000
0.000000

Roots:

Real Part
Imaginary Part

4.000000
-2.89621
.86723
:
3.000000
-2.89621
-.86723
:
2.000000
-2.10379
2.65742
:
1.000000
-2.10379
-2.65742

REGISTERS

A	du				
B	dv				
C	n				
X	$i, du^2 + dv^2$				
Y	u, F_1				
Z	$v, F_2, \text{Temporary}$				
0	dx				
1	dy				
2	F				
3	F_S				
4	L				
5	M				
6	X_S				
7	Y_S				
8	a_0				
9	a_1				
	.				
	.				
	.				
8+n	a_n				
9+n	b_0				
10+n	b_1				
	.				
	.				
	.				
9+2n	b_n				
10+2n	\hat{x}_0				
11+2n	\hat{x}_1				
	.				
	.				
	.				
10+3n	\hat{x}_n				
11+3n	\hat{y}_0			0	
12+3n	\hat{y}_1			1	
	.			2	
	.			3	
	.			4	
11+4n	\hat{y}_n			5	
				6	
				7	
				8	
				9	
				10	
				11	
				12	
				13	
				14	
				15	

FLAGS

PROGRAM LISTING

```

0:
ENT "N",C;0+X;
1:
0+B;ENT A,B;PRT
X,A+R(8+X),B+R(9
+C+X);SPC 1
2:
IF (X+1+X)≤C;
JMP -1;
3:
STP 1
4:
1+R(10+2C)+R7+R(
12+3C);.1+R6+R(1
1+2C);
5:
0+R(11+3C)+R4;
6:
GSB 26;
7:
R2+R3;
8:
(1+X)+R4+R4;0+R5
+A+B;
9:
(XR((9+C+X+R0)+C
)+R1)R(8+X)-RR0(
XR(10+3C+X)+R2)+
A+A;
10:
B+R(8+X)R2+RR0R1
+B;
11:
IF (X+1+X)≤C;
GTO 9;
12:
AA+BB+X;
13:
-(YA+ZB)/X+R0;
14:
(YB-ZA)/X+R1;
15:
R5+1+R5;
16:
R0+R6+R(11+2C);
17:
R1+R7+R(12+3C);
18:
GSB 26;
19:
IF R3≤R2;GTO 23;

```

```

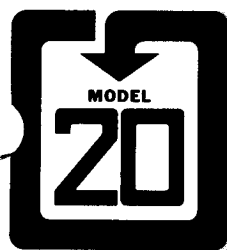
20:
IF R0R0≤1E-10;
IF R1R1≤1E-10;
JMP 15;
21:
IF R4>50;PRT 88;
STP 1
22:
R(11+2C)+R6;R(12
+3C)+R7;GTO 7;
23:
IF R5≤10;R0/4+R0
;R1/4+R1;GTO 15;
24:
IF YY≤1E-8;IF ZZ
≤1E-8;JMP 11;
25:
PRT 99;STP 1
26:
(2R((2+X)C+11)+Y
)Y/4+(R(12+3C)+Z
)Z+Z;
27:
YR((8+2C+X+A)+1)
-ZRA+R(A+2);
28:
YR((A+C+1+A)+1)-
ZRA+R(A+2);
29:
IF (X+1+X)≤C;
JMP -2;
30:
0+X+Y+Z;
31:
(-R(((9+C+X+B)+C
+1+A)+C+1)+R2)R
+R(8+X)RA+Y+Y;
32:
Z-R(8+X)R2+RARB+
Z;
33:
IF (X+1+X)≤C;
JMP -2;
34:
YY+ZZ+R2;RET 1
35:
SPC ;PRT C,R(11+
2C),R(12+3C);
36:
C-1+X;

```

```

37:
(R(11+2C)+A)R(9+
X)-R(12+3C+Z)(R(
(9+C+X+Y)+1)+B)+
R(8+X)+R(8+X);
38:
RZR(9+X)+AB+RY+R
Y;
39:
IF 0≤(X-1+X);
JMP -2;
40:
R((8+X)+C)+Z;
41:
R(X+1)+RX;R((X+C
+A)+2)+RA;
42:
IF (X+1+X)-7≤C;
JMP -1;
43:
Z+R(7+C);IF (C-1
+0)≠1;GTO 4;
44:
(R11+X)X+R9R9+Z;
45:
SPC ;PRT 1,-((R1
0+C)X+R8R9)/Z,(R
8X-R9C)/Z;SPC 9;
46:
END 1

```

POLYNOMIAL EVALUATION,
NORMALIZATION, OR CONSTRUCTION

Model 20
MATH PAC
II-3

Three options are available in this program:

1. Evaluate a polynomial $P(z) = c_0 + c_1z + \dots + c_nz^n$, $\deg P(z) \leq 29$, where the c_k and z may be real or complex. The notation $c_k = \text{Rl}(c_k) + i\text{Im}(c_k)$ and $z = \text{Rl}(z) + i\text{Im}(z)$ is used where $\text{Im}(c_k)$ and $\text{Im}(z)$ are taken as 0 in the real case.
2. Normalize the coefficients c_i of $P(z) = c_0 + c_1z + \dots + c_nz^n$, $\deg P(z) \leq 29$, by dividing each c_i by c_n . Both real and complex values are allowed for the c_i .
3. Construct the coefficients c_0, c_1, \dots, c_n of $P(z)$ given the roots r_1, r_2, \dots, r_n of $P(z)$, $n \leq 29$. In general, the root r_k is given by $r_k = \text{Rl}(r_k) + i\text{Im}(r_k)$ with $\text{Im}(r_k)$ specified as 0 when r_k is real.

ROM BLOCKS

1

2

3

INTERNAL REGISTERS

☒ 173 ☐ 429

PERIPHERALS

☐ 9860A MARKED CARD READER

☐ 9862A PLOTTER

☐ 9864A DIGITIZER

☐
☐ 9861A TYPEWRITER

☐ 9863A PAPER TAPE READER

☐
☐

STEP	DISPLAY	INSTRUCTIONS
1.		<u>ERASE</u> <u>LOAD</u> <u>EXECUTE</u>
2.		Alternately insert magnetic cards and <u>EXECUTE</u> until NOTE 14 is no longer displayed, indicating completion of loading.
3.		(Optional) <u>FIXED N n EXECUTE</u> , or <u>FLOAT N n EXECUTE</u> as desired.
4.		<u>END</u>
5.		<u>RUN PROGRAM</u>
6.	EVALUATE P(Z)? NORMALIZE P(Z)? CONSTRUCT P(Z)?	<u>RUN PROGRAM</u> to cycle between these three displays; 1 <u>RUN PROGRAM</u> to select the option being displayed. Go to Step 11 if constructing P(z).
7.	RL(C)	$\left. \begin{array}{l} \text{Rl}(c_i) \text{ } \underline{\text{RUN PROGRAM}} \\ * \text{Im}(c_i) \text{ } \underline{\text{RUN PROGRAM}} \end{array} \right\} \text{ for } i=0, 1, \dots, n \text{ where } P(z) = c_0 + c_1 z + \dots + c_n z^n$
8.	IM(C)	
		After all c_i have been entered, <u>RUN PROGRAM</u> . If normalizing, the normalized coefficients are printed and control is returned to Step 6. If evaluating P(z), proceed with Step 9.
9.	RL(Z)	<u>Rl(z)</u> <u>RUN PROGRAM</u>
10.	IM(Z)	* <u>Im(z)</u> <u>RUN PROGRAM</u>
		P(z) is printed for the given z and control is returned to Step 9. Another z may then be processed or control returned to Step 6 by <u>RUN PROGRAM</u> .
11.	RL(R)	$\left. \begin{array}{l} \text{Rl}(r_i) \text{ } \underline{\text{RUN PROGRAM}} \\ * \text{Im}(r_i) \text{ } \underline{\text{RUN PROGRAM}} \end{array} \right\} \text{ for } i=1, 2, \dots, n \text{ where } r_1, r_2, \dots, r_n \text{ are the roots of } P(z).$
12.	IM(R)	
		After all r_i have been entered, <u>RUN PROGRAM</u> . The coefficients c_0, c_1, \dots, c_n of P(z) are printed in order (real part first with imaginary part underneath) and control is returned to Step 6.
		* NOTE: If Im(c_i), Im(z), or Im(r_i) is 0, no entry need be keyed in; <u>RUN PROGRAM</u> only and 0 will be assumed for the imaginary entry.

EXAMPLES

Evaluate

$$P(z) = 5 - 3z + 6z^2 \text{ at } z = 8$$

P(Z)
COEFFICIENTS

5.0000
0.0000

-3.0000
0.0000

6.0000
0.0000

AT Z =

8.0000
0.0000

P(Z) IS

365.0000
0.0000

Normalize

$$P(z) = 5 - 3z + 6z^2$$

P(Z)
COEFFICIENTS

5.0000
0.0000

-3.0000
0.0000

6.0000
0.0000

NORMALIZED
P(Z)
COEFFICIENTS

.8333
0.0000

-.5000
0.0000

1.0000
0.0000

Construct

$$P(z) \text{ from } r_1 = 2, r_2 = 3, r_3 = -7$$

GIVEN ROOTS
OF P(Z) ARE

2.0000
0.0000

3.0000
0.0000

-7.0000
0.0000

CONSTRUCTED
P(Z)
COEFFICIENTS

42.0000
0.0000

-29.0000
0.0000

2.0000
0.0000

1.0000
0.0000

Evaluate

$$P(z) = (6 - 5i) + (4 + 2i)z + (-7 + i)z^2 \text{ at } z = 6 + 3i$$

P(Z)
COEFFICIENTS

6.0000
-5.0000

4.0000
2.0000

-7.0000
1.0000

AT Z =

6.0000
3.0000

P(Z) IS

-201.0000
-206.0000

Normalize

$$P(z) = (6 - 5i) + (4 + 2i)z + (-7 + i)z^2$$

P(Z)
COEFFICIENTS

6.0000
-5.0000

4.0000
2.0000

-7.0000
1.0000

NORMALIZED
P(Z)
COEFFICIENTS

-.9400
.5000

-.5200
-.3600

1.0000
0.0000

Construct

$$P(z) \text{ from } r_1 = 3 + 6i, r_2 = 7 + 4i$$

GIVEN ROOTS
OF P(Z) ARE

3.0000
6.0000

7.0000
4.0000

CONSTRUCTED
P(Z)
COEFFICIENTS

-3.0000
54.0000

-10.0000
-10.0000

1.0000
0.0000

REGISTERS									
A	Counter, Temp.								
B	Counter, Temp.								
C	Counter								
X	Real								
Y	Imaginary								
Z	Temporary								
0	0, Temporary								
1	0								
2									
.									
.									
.									
Coefficients									
.									
.									
.									
61									
								FLAGS	
								0	
								1	Used
								2	
								3	
								4	
								5	
								6	
								7	
								8	
								9	
								10	
								11	
								12	
								13	Used
								14	
								15	

PROGRAM LISTING

```

0:
0+A>C+
1:
CFG 13;CFG 1;
ENT "EVALUATE P(
Z)?",Z;IF FLG 13
=0;SPC 1;JMP 12+
2:
CFG 13;ENT "NORM
ALIZE P(Z)?",Z;
IF FLG 13=0;JMP
11+
3:
CFG 13;ENT "CONS
TRUCT P(Z)?",Z;
IF FLG 13;JMP -2
+
4:
PRT "      GIVEN RO
OTS","      OF P(Z)
ARE"+
5:
0+R0+R1+
6:
(C+B)+1+C;1+R(2C
);0+R(2C+1)+
7:
CFG 13;ENT "RL(R
)",X;IF FLG 13;
SPC 3;PRT "      CO
NSTRUCTED";JMP 2
2+
8:
0+Y;ENT "IM(R)",
Y;SPC 1;PRT X,Y+
9:
R(2B+2)+A;R(2B+3
)+Z+
10:
R(2B)-XA+YZ+R(2B
+2)+
11:
R(2B+1)-YA-XZ+R(
2B+3)+
12:
GTO 6;IF 0<(B-1+
B);GTO 9+
13:
GSB 16+
14:
CFG 13;ENT "RL(C
)",X+R(2C+2);IF
FLG 13;JMP 3+

```

```

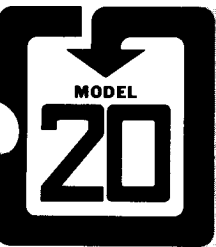
15:
0+Y;ENT "IM(C)",
Y;Y+R(2C+3);SPC
1;PRT X,Y;C+1+C;
JMP -1+
16:
PRT "      P(Z)"
,"      COEFFICIENTS
";RET +
17:
IF FLG 1=0;JMP 7
+
18:
CFG 13;ENT "RL(Z
)",X;IF FLG 13;
JMP 14+
19:
0+Y;ENT "IM(Z)",
Y;SPC 3;PRT "
      AT Z =" ;SPC 1;
PRT X,Y+
20:
SPC 2;PRT "
P(Z) IS";SPC 1+
21:
0+A+B;C+Z+
22:
R(2Z)+AX-BY+R0;R
(2Z+1)+BX+AY+B;R
0+A;JMP (Z-1+Z)=
0+
23:
PRT A,B;JMP -5+
24:
XX+YY+Z+
25:
IF (A+1+A)>C;
JMP 3+
26:
R(2A)+B;R(2A+1)+
R0+
27:
(XE+YR0)/Z+R(2A)
;(XR0-YB)/Z+R(2A
+1);GTO 25+
28:
SPC 3;PRT "      NO
RMALIZED"+
29:
GSB 16+
30:
1+R+

```

```

31:
SPC 1;PRT R(2A),
R(2A+1);A+1+A;
JMP A>C+
32:
SPC 8;GTO 0+
33:
END +

```

POLYNOMIAL INTEGRATION,
DIFFERENTIATION, OR ORIGIN SHIFT

Model 20
MATH PAC
II-4

For a given polynomial $P(X)$ with real coefficients, this program can be used to find the coefficients of the polynomial $\int P(X)dX$, $\frac{d}{dX} P(X)$, or $P(X-a)$ where a is a real number specified by the user.

With the basic 9820 memory size of 173 R-registers, the size limitations are:

degree $P(X) \leq 49$	for integration
degree $P(X) \leq 50$	for differentiation
degree $P(X) \leq 100$	for origin shift

Polynomials $P(X)$ with complex coefficients can be handled by writing $P(X)$ in the form $P(X) = Q(X) + iH(X)$ where $Q(X)$ and $H(X)$ are polynomials with real coefficients, and then applying the program to each of $Q(X)$ and $H(X)$ separately since $\int P(X) = \int Q(X) + i \int H(X)$, $\frac{d}{dX} P(X) = \frac{d}{dX} Q(X) + i \frac{d}{dX} H(X)$, and $P(X-a) = Q(X-a) + iH(X-a)$.

ROM BLOCKS

1

2

3

INTERNAL REGISTERS

☒ 173 ☐ 429

PERIPHERALS

☐ 9860A MARKED CARD READER☐ 9862A PLOTTER☐ 9864A DIGITIZER☐☐ 9861A TYPEWRITER☐ 9863A PAPER TAPE READER☐☐

STEP	DISPLAY	INSTRUCTIONS
1.		<u>ERASE</u> <u>LOAD</u> <u>EXECUTE</u>
2.		Alternately insert magnetic cards and <u>EXECUTE</u> until NOTE 14 is no longer displayed, indicating completion of loading.
3.		<u>END</u>
4.		<u>RUN PROGRAM</u>
5.	P(X) COEFF'S?	<u>a_i</u> <u>RUN PROGRAM</u> i = 0, 1, ..., n where $P(X) = a_0 + a_1X + \dots + a_nX^n$. <u>RUN PROGRAM</u> after all <u>a_i</u> have been entered.
6.	INTEGRATE P(X)? DIFF. P(X)? SHIFT P(X)?	<u>RUN PROGRAM</u> to cycle between these three displays; <u>1</u> <u>RUN PROGRAM</u> to select the option being displayed. Bypass the next step when integrating or differentiating.
7.	P(X-A) FOR A = ?	<u>a</u> <u>RUN PROGRAM</u> where <u>a</u> is the shift value.
8.		The coefficients of the resultant polynomial $b_0 + b_1X + \dots + b_kX^k$ are now printed in the order b_0, b_1, \dots, b_k . To run a new problem, return to Step 4.

EXAMPLES

1. Integrate $P(X) = 5 + 3X + 9X^2 + 8X^3$

P(X)
COEFFICIENTS
5.0000
3.0000
9.0000
8.0000

Answer: $5X + 1.5X^2 + 3X^3 + 2X^4$

INTEGRAL OF
P(X)
COEFFICIENTS
0.0000
5.0000
1.5000
3.0000
2.0000

2. Differentiate $P(X) = 2 + 3X + X^2 + 6X^3$

P(X)
COEFFICIENTS
2.0000
3.0000
1.0000
6.0000

Answer: $3 + 2X + 18X^2$

DERIVATIVE OF
P(X)
COEFFICIENTS
3.0000
2.0000
18.0000

3. Find $P(X-4)$ if $P(X) = 5 + 8X$

P(X)
COEFFICIENTS
5.0000
8.0000

A=

4.0000

P(X-A)
COEFFICIENTS

Answer: $-27 + 8X$

-27.0000
8.0000

REGISTERS

A Shift Value
B Counter
C Shift Counter
X Dimension of P(X)
Y Dimension of Result
Z Counter

0 Unused
1

**P(x) coefficients
 overlaid by
 resultant polynomial
 coefficients**

FLAGS

0
1
2
3
4
5
6
7
8
9
10
11
12
13 Used
14
15

PROGRAM LISTING

```

0:
0+X;1+B-
1:
CFG 13;GSB 12-
2:
ENT "P(X) COEFF"
S?",R(X+1+X);IF
FLG 13=0;PRT ;
JMP 0-
3:
CFG 13;ENT "INTE
GRATE P(X)?",Z;
IF FLG 13=0;JMP
12-
4:
CFG 13;ENT "DIFF
. P(X)?",Z;IF
FLG 13=0;JMP 15-
5:
CFG 13;ENT "SHIF
T P(X)?",Z;IF
FLG 13;JMP -2-
6:
ENT "P(X-A) FOR
A=?",A-
7:
SPC 3;PRT "
A=";SPC 1;PRT
A;SPC 2;PRT "
P(X-A)";GSB 14
-
8:
X+C+Y-
9:
IF C>B;R(C-1+C)-
AR(C+1)+RC;JMP 0
-
10:
IF X>(B+1+B);
JMP -2-
11:
0+X;JMP 11-
12:
PRT " P(X)"
;GSB 14-
13:
RET -
14:
PRT " COEFFICIE
NTS";SPC 1;RET -

```

```

15:
SPC 3;PRT " IN
TEGRAL OF";GSB 1
2-
16:
(1+Z)+X+Y;0+RY-
17:
IF (B+1+B)>Y;
JMP 5-
18:
R(B-1)/Z+R(X+B);
Z+1+Z;GTO -1-
19:
SPC 3;PRT " DER
IVATIVE OF";GSB
12-
20:
X-1+Y;0+Z-
21:
Z+1+Z;R(B+1)Z+R(
X+B);JMP (B+1+B)
>Y-
22:
IF (0+B)=Y;PRT Y
;JMP 2-
23:
PRT R((B+1+B)+X)
;JMP B=Y-
24:
SPC 8-
25:
END -

```




SIMULTANEOUS ORDINARY DIFFERENTIAL EQUATIONS
Runge-Kutta (Gill Method)

Model 20
MATH PAC
III-1

This program solves the following system of simultaneous first-order ordinary differential equations:

$$\frac{dy_1}{dx} = f_1(x, y_1, y_2, \dots, y_n)$$

$$\frac{dy_2}{dx} = f_2(x, y_1, y_2, \dots, y_n)$$

.

.

.

$$\frac{dy_n}{dx} = f_n(x, y_1, y_2, \dots, y_n)$$

with initial conditions $x_0, y_1(x_0), y_2(x_0), \dots, y_n(x_0)$ specified by the user.

The program may also be used to solve an equation of the form:

$$\frac{d^m y}{dx^m} = f\left(x, y, \frac{dy}{dx}, \frac{d^2 y}{dx^2}, \dots, \frac{d^{m-1} y}{dx^{m-1}}\right)$$

with given initial conditions $y(x_0), \frac{dy}{dx}(x_0), \dots, \frac{d^{m-1} y}{dx^{m-1}}(x_0)$ by rewriting the equation as a system of first-order equations as above.

References:

Ralston and Wilf, Mathematical Methods for Digital Computers, Vol. 1 (New York: John Wiley and Sons, Inc., 1960), p. 115.

Carnahan, Luther, Wilkes, Applied Numerical Methods, (New York: John Wiley and Sons, Inc., 1969), pp. 363-366.

REMARKS

1. The number of equations, n , that can be solved simultaneously depends on the amount of programming required for each equation since there is a trade-off between program length and data storage. Of the 87 registers available, $3n + 8$ are needed for data, leaving $87 - (3n + 8)$ registers for programming the n equations at about 8 keystrokes per register. For example, with the basic calculator memory, the maximum number of equations is 20 which allows about 8 keystrokes to define each equation. A more reasonable number is 10 equations which allow an average of about 39 keystrokes for each equation.
2. To define the equations, the user must first formulate expressions for $dy_1/dx, dy_2/dx, \dots, dy_n/dx$ obtaining as needed the values x, y_1, \dots, y_n from register X, R8, R9, \dots , $R(7+n)$ respectively. Then, beginning in program line 28 and proceeding for as many lines as are needed, the user keys in and stores as program lines the expressions: $dy_1/dx \rightarrow R(8+n), dy_2/dx \rightarrow R(9+n), \dots, dy_n/dx \rightarrow R(7+2n)$ where n refers to the number of equations. Registers R0 and Z are available as temporary storage areas.

ROM BLOCKS

1

2

3

INTERNAL REGISTERS

☒ 173 ☐ 429

PERIPHERALS

☐ 9860A MARKED CARD READER☐ 9862A PLOTTER☐ 9864A DIGITIZER☐☐ 9861A TYPEWRITER☐ 9863A PAPER TAPE READER☐☐

STEP	DISPLAY	INSTRUCTIONS
1.		<u>ERASE</u> <u>LOAD</u> <u>EXECUTE</u>
2.		Alternately insert magnetic cards and <u>EXECUTE</u> until NOTE 14 is no longer displayed, indicating completion of loading.
3.		(Optional) <u>FIXED N n EXECUTE</u> , or <u>FLOAT N n EXECUTE</u> as desired.
4.		<u>GO TO 2 8 EXECUTE</u> and then using as needed register X for x, R8 for y_1 , R9 for y_2, \dots , and $R(7+n)$ for y_n , key in: $\frac{dy_1}{dx} \longrightarrow R(8+n)$ <u>STORE</u> $\frac{dy_2}{dx} \longrightarrow R(9+n)$ <u>STORE</u> . . . $\frac{dy_n}{dx} \longrightarrow R(7+2n)$ <u>STORE</u> <u>RETURN</u> <u>STORE</u> <u>END</u> <u>STORE</u> .
5.		<u>END</u> <u>RUN PROGRAM</u>
6.	HOW MANY EQS?	<u>n</u> <u>RUN PROGRAM</u>
7.	START AT X = ?	<u>x_0</u> <u>RUN PROGRAM</u>
8.	END AT X = ?	<u>x_{\max}</u> <u>RUN PROGRAM</u>
9.	DELTA X = ?	<u>Δx</u> <u>RUN PROGRAM</u>
10.	PRINT INTERVAL?	<u>p</u> <u>RUN PROGRAM</u> for printing every p^{th} increment in x.
11.	INITIAL VALUES?	<u>$y_i(x_0)$</u> <u>RUN PROGRAM</u> $i=1,2,\dots,n$. Calculations proceed automatically after the last initial value is entered. After the results are printed, proceed to Step 12.
12.	NEW ENDING X?	<u>x_{\max}</u> <u>RUN PROGRAM</u> if results are desired for a new x_{\max} larger than the previous one. After these additional results are obtained, control returns to Step 12.

EXAMPLES

Solve $d^2y/dx^2 = x^3 + 7x^2 - 14x + 40 - y + dy/dx$ for $0 \leq x \leq .5$, $\Delta x = .01$ and printing every fifth one, subject to the initial conditions

$$y(0) = 20$$

$$\frac{dy}{dx}(0) = 0$$

The solution may be obtained by first reducing the equation to a set of simultaneous equations using the following substitutions: $y_1 = y$, $y_2 = dy_1/dx = dy/dx$.

This gives $dy_1/dx = y_2$, $dy_2/dx = x^3 + 7x^2 - 14x + 40 - y_1 + y_2$ with initial conditions $y_1(0) = 20$, $y_2(0) = 0$. A solution to this set of two simultaneous equations will yield the solution to the original equation: $y = y_1$ is the relation giving the desired result, and the printout of the y_2 values may be ignored.

The Memory Allocation page shows that y_1 is in R8, y_2 is in R9, dy_1/dx is to be stored in R10, and dy_2/dx is to be stored in R11.

Key in:

Printout:

```

27:
" F " F
28:
R9+R10 F
29:
XXX+7XX-14X+40-R
8+R9+R11 F
30:
RET F
31:
END F

```

INPUT DATA

2.0000	Number of Equations
0.0000	Starting x
.5000	Ending x
.0100	Delta x
5.0000	Print Interval
20.0000	$y_1(0)$
0.0000	$y_2(0)$

RESULTS

.0500	x
20.0251	$y_1(x)$
1.0075	$y_2(x)$
.1000	x
20.1010	$y_1(x)$
2.0300	$y_2(x)$
.1500	x
20.2284	$y_1(x)$
3.0675	$y_2(x)$
.	.
.	.
.	.
.5000	x
22.6250	$y_1(x)$
10.7500	$y_2(x)$

REGISTERS

A	Counter				
B	Number of Equations				
C	y Pointer				
X	x				
Y	Q Pointer				
Z	Available to User				
0	Available to User				
1	Delta x				
2	Maximum x				
3	1 or 2				
4	Print Interval				
5	Print Counter				
6	Temporary				
7	dy/dx Pointer				
8	y ₁				
9	y ₂				
.	.				
.	.				
.	.				
8+n	dy ₁ /dx				
9+n	dy ₂ /dx				
.	.				
.	.				
.	.				
8+2n	Q ₁				
9+2n	Q ₂				
.	.				
.	.				
.	.				
7+3n	Q _n				
					FLAGS
					0
					1
					2 Used
					3
					4
					5
					6
					7
					8
					9
					10
					11
					12
					13
					14
					15

PROGRAM LISTING

```

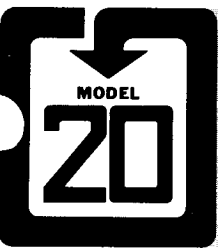
0:
ENT "HOW MANY EQ
S.?",B,"START AT
X= ?";X,"END AT
X= ?";R2F
1:
ENT "DELTA X= ?"
,R1,"PRINT INTER
VAL?";R4F
2:
(7+C)+2B+Y;PRT "
INPUT DATA";
SPC 1;PRT B,X,R2
,R1,R4;SPC 2F
3:
0+AF
4:
A+1+A;0+R(Y+A);
ENT "INITIAL VAL
UES?";R(C+A);
PRT ;JMP A=BF
5:
1+R5;SPC 2;PRT "
RESULTS";
SPC 1F
6:
SFG 2;.5+R6;2+R3
;GSB 19F
7:
R1/2+X+X;1-r.5+R
6;1+R3;GSB 19F
8:
1+r.5+R6;GSB 19F
9:
R1/2+X+X;CFG 2;1
/6+R6;2+R3;GSB 1
9F
10:
IF R5#R4;R5+1+R5
;GTO 15F
11:
PRT X;SPC 1;8+C;
0+AF
12:
PRT R(C+A)F
13:
A+1+A;IF B>A;
GTO 12F
14:
SPC 2;1+R5F
15:
IF R2>X;GTO 6F

```

```

16:
DSP "NEW ENDING
X?";SPC 6F
17:
STP F
18:
Z+R2;GTO 6F
19:
GSB "F"F
20:
((8+C)+B+A+R7)+B
+YF
21:
RAR1+RA;A+1+A;
IF 2B>A-8;GTO +0
F
22:
1+AF
23:
RC+R6RR7-R6R3RY+
RCF
24:
RY-3R6R3RY+2R6RR
7FLG 2+RYF
25:
(((A+1+A)+7+C)+B
+R7)+B+Y;IF A≤B;
GTO 23F
26:
RET F
27:
"F"F
28:
END F

```

ROMBERG QUADRATURE FOR $\int_a^b f(x) dx$

The Romberg method for obtaining successive approximations of a definite integral $\int_a^b f(x) dx$ is considered to be a very elegant and efficient technique. Wilf² states that the method virtually eclipses the Newton-Cotes formulas on every score, and that only the Gaussian quadrature formulas can be considered to be on a comparable level. Further comparisons of the different methods together with a description of the Romberg method may be found in references 1 and 2.

Basically, the technique consists of generating a table of values called the Romberg tableau, the columns of which (and, eventually the rows) converge to the value of the integral assuming no roundoff error.

The tableau looks like:

$$\begin{array}{cccc} T_0^0 & & & \\ T_0^1 & T_1^0 & & \\ T_0^2 & T_1^1 & T_2^0 & \\ T_0^3 & T_1^2 & T_2^1 & T_3^0 \\ . & . & . & . \end{array}$$

where the elements are generated from the recursion relation $T_m^k = \frac{4^m T_{m-1}^{k+1} - T_{m-1}^k}{4^m - 1}$ and the recursion

is initiated using the trapezoid sums (that is, successive approximations to the integral using repeated interval halving and the Trapezoid Rule) given in the first column.

The program is based on a FORTRAN version¹ and includes a convergence check on the trapezoidal sums since for some functions these will converge faster than other columns in the Romberg table. In addition, the improvements suggested by McCalla have been incorporated in this program: a recursion formula for successive trapezoidal sums has been included to reduce computations⁴, and a modification has been made to make the basic Romberg algorithm less sensitive to the accumulation of rounding errors³.

The user has the option to print the complete tableau or to designate a tolerance and have only the final estimate of the integral printed (when two successive elements of a column are within the designated tolerance). In either case, the column elements are flashed in the display as they are calculated so the user may view the convergence.

References:

- ¹ Thomas Richard McCalla, Introduction to Numerical Methods and FORTRAN Programming, (New York: John Wiley and Sons, Inc., 1967), pp. 287-297.
- ² Ralston and Wilf, Mathematical Methods for Digital Computers, Volume II, (New York: John Wiley and Sons, Inc., 1967), pp. 133-143.
- ³ A. M. Krasun and W. Prager, "Remark on Romberg Quadrature", Communications of the ACM, Vol. 8, No. 4 (April, 1965), pp. 236-237.
- ⁴ Peter Henrici, Elements of Numerical Analysis, (New York: John Wiley and Sons, Inc., 1964), pp. 259-262.

ROM BLOCKS

1	2	3

INTERNAL REGISTERS

☒ 173 ☐ 429

PERIPHERALS

<input type="checkbox"/> 9860A MARKED CARD READER	<input type="checkbox"/> 9862A PLOTTER	<input type="checkbox"/> 9864A DIGITIZER	<input type="checkbox"/> _____
<input type="checkbox"/> 9861A TYPEWRITER	<input type="checkbox"/> 9863A PAPER TAPE READER	<input type="checkbox"/> _____	<input type="checkbox"/> _____

STEP	DISPLAY	INSTRUCTIONS
1.		<u>ERASE</u> <u>LOAD</u> <u>EXECUTE</u>
2.		Alternately insert magnetic cards and <u>EXECUTE</u> until NOTE 14 no longer appears, indicating completion of loading.
3.		(Optional) <u>FIXED N n EXECUTE</u> , or <u>FLOAT N n EXECUTE</u> and set degrees or radians, if desired.
4.		<u>GO TO 3 5 EXECUTE</u>
5.		Using register X to represent the variable x in the expression for f(x), key in the following: <u>f(x)</u> \longrightarrow <u>Y STORE RETURN STORE END STORE</u> .
6.		<u>END RUN PROGRAM</u>
7.	A	<u>a RUN PROGRAM</u> a = lower limit
	B	<u>b RUN PROGRAM</u> b = upper limit
8.	PRINT TABLEAU?	<u>RUN PROGRAM</u> to omit tableau printout and go to Step 9, or <u>1 RUN PROGRAM</u> to get tableau printout and go to Step 10.
9.	EPSILON	<u>ϵ RUN PROGRAM</u> ϵ = accuracy tolerance for the value of the integral. <u>RUN PROGRAM</u> alone in this step will automatically enter a default value of $\epsilon = 10^{-11}$.
10.		Calculated tableau elements are flashed in the display. If the tableau print option has been selected, the tableau is printed, row by row, until <u>STOP</u> is pressed. Otherwise, the tableau is not printed and calculations proceed until the estimate of the integral is within the designated tolerance and printed. Control returns to Step 7 in this case.

EXAMPLES

1. Using $\epsilon = 10^{-6}$, verify that $\int_1^2 x^{-2} dx = .5$

Program Steps

```

34:
  "F" F
35:
  1/XX+YF
36:
  RET F
37:
  END F

```

Results

```

      INTERVAL
1.0000000000E 00
2.0000000000E 00

      INTEGRAL
      OF F(X) IS
5.0000000014E-01

```

2. Print the Romberg tableau for $\int_1^2 x^{-2} dx$

Romberg Tableau

```

.62500 00000
.53472 22222 → .50462 96296
.50899 37642 → .50041 76115 → .50013 68103
.50227 08503 → .50002 98790 → .50000 40302 → .50000 19226
.50056 91701 → .50000 19434 → .50000 00810 → .50000 00183 → .50000 00109
.50014 23846 → .50000 01227 → .50000 00014 → .50000 00001 → .50000 00000 → .50000 00000
.50003 56019 → . . . . .
. . . → . . . . .

```

The arrows indicate the actual order of printout.

```

      INTERVAL
1.0000000000E 00
2.0000000000E 00

```

```

ROMBERG TABLEAU
6.2500000000E-01

```

```

5.347222222E-01

```

```

5.089937642E-01
5.046296296E-01

```

```

5.022708503E-01
5.004176115E-01
5.001368103E-01

```

```

5.005691701E-01
5.000298790E-01
5.000040302E-01
5.000019226E-01

```

```

5.001423846E-01
5.000019434E-01
5.000000810E-01
5.000000183E-01
5.000000109E-01

```

```

5.000356019E-01
5.000001227E-01
5.000000014E-01
5.000000001E-01
5.000000000E-01
5.000000000E-01

```

REGISTERS

A	a
B	b, R_0^{n-1}, K
C	n
X	x
Y	f(x), Temporary
Z	Available to User

0	Available to User
1	h
2	δ
3	Lim
4	j, n max
5	ϵ
6	T_1
7	R_1
8	T_2
9	R_2

•
•
•

FLAGS

0	
1	Branching Indicator
2	Print Indicator
3	
4	
5	
6	
7	
8	
9	
10	
11	
12	
13	Option Indicator
14	
15	

PROGRAM LISTING

```

0:
SPC 8;CFG 1;CFG
2;CFG 13;ENT A;B
F
1:
PRT "      INTERVA
L";A;B;SPC 2F
2:
ENT "PRINT TABLE
AU?";Z;IF FLG 13
JMP 2F
3:
SFG 2;PRT " ROMB
ERG TABLEAU";
JMP 3F
4:
1E-11;R5;ENT "EP
SILON";R5F
5:
PRT "      INTEGRA
L";"      OF F(X) I
S"F
6:
((B+X)-A+R1)/4;R
2;GSB "F"F
7:
Y+B;A+X;GSB "F"F
8:
(Y+B)R1/2;R6F
9:
1+C;R3F
10:
R1/2+R1;0+B;1;R4
F
11:
A+(2R4-1)R1;X;
GSB "F"F
12:
Y+B+B;IF (R4+1;R
4);R3;GTO -1F
13:
2R1B;R(2C+5);ZF
14:
((Z+(R(2C+4)+B))
/2+R(2C+6)+Z)-B;
BF
15:
DSP R(2C+4);IF
FLG 2;PRT R(2C+4
);GTO +3F
16:
IF 0>B;-B+BF
17:
IF R5>B;PRT Z;
GTO 0F
18:
IF R1<R2;C+R4;
GTO +3F
19:
C+1;C;IF FLG 2;
SPC 1F
20:
2R3+R3;GTO -10F
21:
1+(C-2)FLG 1;C;
SFG 1F
22:
0+Z;C+BF
23:
4Z+3+ZF
24:
(R(2B+7)+Y)+(Y-R
(2B+5))/Z+YF
25:
IF B>1;GTO +3F
26:
R7+(Z-1)(R6-R7)/
2Z+R6;Y+R7F
27:
GTO -8;IF R4>(C+
1+C);GTO -5F
28:
Y+R(2B+5)F
29:
(((R(2B+3)+Y)+R(
2B+2))/2+R(2B+4)
+X)-Y+YF
30:
DSP R(2B+2);IF
FLG 2;PRT R(2B+2
);GTO +3F
31:
IF 0>Y;-Y+YF
32:
IF R5>Y;PRT X;
GTO 0F
33:
B-1+B;GTO -10F
34:
"F"F
35:
END F

```




GAUSSIAN QUADRATURE FOR $\int_a^b f(x) dx$ OR $\int_a^\infty f(x) dx$

Model 20
MATH PAC
III-3

For a given function $f(x)$, this program will approximate $\int_a^b f(x) dx$ or $\int_a^\infty f(x) dx$ by the 8, 12, 16, 20, or 24-point Gauss-Legendre quadrature formula.

Derivation of the formulas may be found in either reference, and an extensive table of constants is given in reference 2 for orders other than 8, 12, 16, 20, or 24

References:

¹ Brice Carnahan, H. A. Luther, James O. Wilkes, Applied Numerical Methods (New York: John Wiley and Sons, Inc.), pp. 101-112.

² A. H. Stroud, Don Secrest, Gaussian Quadrature Formulas, (Prentice-Hall, Inc. , 1966).

REMARKS

1. Since a large number of constants must be keyed in and stored, it is recommended that they then be recorded on magnetic cards and retained for later use by RECORD " D A " , R 8 4 where 84 refers to the highest data register containing one of the constants.
2. If the user wishes to use only one or two of the formulas, only those related constants need be keyed in and stored in the designated registers. However, comparing the integral estimates using several of the formulas has the advantage of giving the user some indication of convergence and accuracy of the individual estimates.

Constants Required

8-Point Formula

9.602898565E-01
 7.966664774E-01
 5.255324099E-01
 1.834346425E-01
 1.012285363E-01
 2.223810345E-01
 3.137066459E-01
 3.626837834E-01

Store in registers
 R5 through R12

12-Point Formula

9.815606342E-01
 9.041172564E-01
 7.699026742E-01
 5.873179543E-01
 3.678314990E-01
 1.252334085E-01
 4.717533639E-02
 1.069393260E-01
 1.600783285E-01
 2.031674267E-01
 2.334925365E-01
 2.491470458E-01

Store in registers
 R13 through R24

16-Point Formula

9.894009350E-01
 9.445750231E-01
 8.656312024E-01
 7.554044084E-01
 6.178762444E-01
 4.580167777E-01
 2.816035508E-01
 9.501250984E-02
 2.715245941E-02
 6.225352394E-02
 9.515851168E-02
 1.246289713E-01
 1.495959888E-01
 1.691565194E-01
 1.826034150E-01
 1.894506105E-01

Store in registers
 R25 through R40

20-Point Formula

9.931285992E-01
 9.639719273E-01
 9.122344283E-01
 8.391169718E-01
 7.463319065E-01
 6.360536807E-01
 5.108670020E-01
 3.737060887E-01
 2.277858511E-01
 7.652652113E-02
 1.761400714E-02
 4.060142980E-02
 6.267204833E-02
 8.327674158E-02
 1.019301198E-01
 1.181945320E-01
 1.316886384E-01
 1.420961093E-01
 1.491729865E-01
 1.527533871E-01

Store in registers
 R41 through R60

24-Point Formula

9.951872200E-01
 9.747285560E-01
 9.382745520E-01
 8.864155270E-01
 8.200019860E-01
 7.401241916E-01
 6.480936519E-01
 5.454214714E-01
 4.337935076E-01
 3.150426797E-01
 1.911188675E-01
 6.405689286E-02
 1.234122980E-02
 2.853138863E-02
 4.427743882E-02
 5.929858492E-02
 7.334648141E-02
 9.619016153E-02
 9.761865210E-02
 1.074442701E-01
 1.155056681E-01
 1.216704729E-01
 1.258374563E-01
 1.279381953E-01

Store in registers
 R61 through R84

ROM BLOCKS

1

2

3

INTERNAL REGISTERS

☒ 173 ☐ 429

PERIPHERALS

☐ 9860A MARKED CARD READER☐ 9862A PLOTTER☐ 9864A DIGITIZER☐☐ 9861A TYPEWRITER☐ 9863A PAPER TAPE READER☐☐

STEP	DISPLAY	INSTRUCTIONS
1.		<u>ERASE</u> <u>LOAD</u> <u>EXECUTE</u>
2.		Alternately insert the program magnetic cards and <u>EXECUTE</u> until NOTE 14 is no longer displayed, indicating completion of loading.
3.		<u>LOAD</u> <u>" D A "</u> <u>EXECUTE</u>
4.		Alternately insert the data magnetic cards and <u>EXECUTE</u> until NOTE 14 is no longer displayed, indicating completion of loading.
5.		(Optional) <u>FIXED N n EXECUTE</u> , or <u>FLOAT N n EXECUTE</u> as desired.
6.		<u>GO TO 1 7 EXECUTE</u>
7.		Using register X to represent the variable x in the expression for f(x), key in the following: <u>f(x)</u> <u>→</u> <u>Y STORE RETURN STORE END STORE</u>
8.		<u>END RUN PROGRAM</u>
9. A		<u>a RUN PROGRAM</u>
10. B		<u>b RUN PROGRAM</u> For $\int_a^{\infty} f(x) dx$, use b = 1E 99.
11. 8, 12, 16, 20, or 24?		<u>n RUN PROGRAM</u> where n is one of the displayed values and refers to the n-point Gauss-Legendre integration formula.
12.		The estimate of the integral is printed for the chosen n and control returns to Step 11. Another n may then be selected or control may be transferred to Step 9 for selection of a new interval by <u>RUN PROGRAM</u> alone.

EXAMPLES

1. Evaluate $\ln t = \int_1^t \frac{1}{x} dx$ for $t = 5$ using the

Output

8-point formula and then the 12-point formula.

Here $f(x) = \frac{1}{x}$ and the following program lines are required.

```
16:
" F" F
17:
1/X → Y F
18:
RET F
19:
END F
```

```
INTERVAL
1.0000000000E 00
5.0000000000E 00
```

8-Point Formula

```
INTEGRAL
OF F(X) IS
1.609437439E 00
```

```
INTEGRAL
OF F(X) IS
1.609437912E 00
```

12-Point Formula

This last value corresponds exactly with $\ln 5$ for the number of digits shown.

2. (This example requires the ~~Trigonometric~~ ^{Mathematical} Block). Evaluate the gamma function

$$\Gamma(\alpha) = \int_0^{\infty} e^{-x} x^{\alpha-1} dx \quad \text{for } \alpha = 1.8 \text{ using}$$

each of the formulas.

Program Lines

```
16:
" F" F
17:
EXP (-X) X ↑ .8 → Y F
18:
RET F
19:
END F
```

Output

```
INTERVAL
0.0000000000E 00
1.0000000000E 99
```

```
INTEGRAL
OF F(X) IS
9.318350473E-01
```

8-Point Formula

```
INTEGRAL
OF F(X) IS
9.310731317E-01
```

12-Point Formula

```
INTEGRAL
OF F(X) IS
9.313784467E-01
```

16-Point Formula

```
INTEGRAL
OF F(X) IS
9.313880955E-01
```

20-Point Formula

```
INTEGRAL
OF F(X) IS
9.313838718E-01
```

24-Point Formula

The values are seen to be converging to the true value $\Gamma(1.8) = 9.313837710 \text{ E-01}$.

MEMORY ALLOCATION

REGISTERS

A Lower Limit
B Upper Limit
C Pointer 1
X x
Y f(x)
Z Available to User

0 Sum
1 $(b+a)/2$
2 $(b-a)/2$
3 Pointer 2
4 n

FLAGS

0
1 Used
2
3
4
5
6
7
8
9
10
11
12
13 Used
14 Used
15

PROGRAM LISTING

```

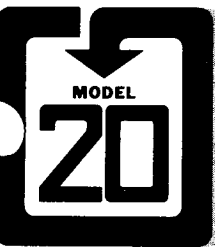
0:
CFG 1;SFG 14;
ENT A,B;
1:
PRT "      INTERVA
L",A,B;SPC 2;IF
B=1E99;A+1+B;
SFG 1;
2:
(B+A)/2+R1;(B-A)
/2+R2;
3:
CFG 13;ENT "8,12
,16,20,OR 24";C/
2+R4;IF FLG 13;
GTO 0;
4:
IF (8(C=8)+16(C=
12)+32(C=16)+50(
C=20)+72(C=24)+R
3)=0;GTO -1;
5:
R3-R4+CH
6:
0+R0;
7:
IF (C+1+C)>R3;
PRT "      INTEGRA
L","      OF F(X) I
S",R0R2;SPC 8;
GTO -4;
8:
R2RC+R1+X;IF
FLG 1;2/(1+RC)-1
+A+X;
9:
GSB "F";
10:
IF FLG 1;4Y/(1+R
C)(1+RC)+Y;
11:
R(C+R4)Y+R0+R0;
12:
-R2RC+R1+X;IF
FLG 1;2/(1-RC)-1
+A+X;
13:
GSB "F";
14:
IF FLG 1;4Y/(1-R
C)(1-RC)+Y;
15:
R(C+R4)Y+R0+R0;
GTO -8;

```

```

16:
"F";
17:
END ;

```


NUMERICAL INTEGRATION WITH
EQUALLY SPACED BASE POINTS
NEWTON-COTES CLOSED FORMULAS

Let x_0, x_1, \dots, x_n be equally spaced points each h units apart at which corresponding values $f(x_0), f(x_1), \dots, f(x_n)$ of a function f are known. Using only this information, with no explicit expression for f itself,

$$\int_{x_0}^{x_n} f(x) dx$$

may be approximated using one of the Newton-Cotes closed integration formulas. This program allows selection of any one of the first six of these formulas:

$$1. \int_{x_0}^{x_1} f(x) dx \approx \frac{h}{2} [f(x_0) + f(x_1)] \quad (\text{Trapezoidal Rule})$$

$$2. \int_{x_0}^{x_2} f(x) dx \approx \frac{h}{3} [f(x_0) + 4f(x_1) + f(x_2)] \quad (\text{Simpson's Rule})$$

$$3. \int_{x_0}^{x_3} f(x) dx \approx \frac{3h}{8} [f(x_0) + 3f(x_1) + 3f(x_2) + f(x_3)] \quad (\text{Simpson's Second Rule})$$

$$4. \int_{x_0}^{x_4} f(x) dx \approx \frac{2h}{45} [7f(x_0) + 32f(x_1) + 12f(x_2) + 32f(x_3) + 7f(x_4)]$$

$$5. \int_{x_0}^{x_5} f(x) dx \approx \frac{5h}{288} [19f(x_0) + 75f(x_1) + 50f(x_2) + 50f(x_3) + 75f(x_4) + 19f(x_5)]$$

$$6. \int_{x_0}^{x_6} f(x) dx \approx \frac{h}{140} [41f(x_0) + 216f(x_1) + 27f(x_2) + 272f(x_3) + 27f(x_4) + 216f(x_5) + 41f(x_6)]$$

The program is designed so that formula k may be used m times to successively integrate over m adjacent intervals provided that $mk + 1$ pairs $(x_i, f(x_i))$ are known. A running sum is kept of the integral over these intervals and, at the user's option, may be output after the last point of any interval has been processed.

Several formulas have been included to allow the user to match one of the formulas to the number of available data points. Also, more than one formula may apply in which case the user has the option of running the program for each formula to compare results.

Reference:

Brice Carnahan, H. A. Luther, James O. Wilkes, Applied Numerical Methods (New York: John Wiley and Sons, Inc., 1969), pp. 70-75.

ROM BLOCKS

1

2

3

INTERNAL REGISTERS

☒ 173 ☐ 429

PERIPHERALS

☐ 9860A MARKED CARD READER☐ 9862A PLOTTER☐ 9864A DIGITIZER☐☐ 9861A TYPEWRITER☐ 9863A PAPER TAPE READER☐☐

STEP	DISPLAY	INSTRUCTIONS
1.		<u>ERASE</u> <u>LOAD</u> <u>EXECUTE</u>
2.		Alternately insert magnetic cards and <u>EXECUTE</u> until NOTE 14 is no longer displayed, indicating completion of loading.
3.		(Optional) <u>FIXED N n EXECUTE</u> , or <u>FLOAT N n EXECUTE</u> as desired.
4.		<u>END</u> <u>RUN PROGRAM</u>
5.	TRAPEZOID RULE? SIMPSON'S RULE? SIMPSON RULE 2? NEWTON-COTES 4? NEWTON-COTES 5? NEWTON-COTES 6?	<u>RUN PROGRAM</u> to cycle between these 6 displays; <u>1 RUN PROGRAM</u> to select the option being displayed.
6.	X0?	<u>x₀</u> <u>RUN PROGRAM</u>
7.	Y0?	<u>y₀</u> <u>RUN PROGRAM</u>
8.	X INCREMENT?	<u>h</u> <u>RUN PROGRAM</u>
9.	NEXT Y?	<u>y_i</u> <u>RUN PROGRAM</u> or <u>RUN PROGRAM</u> to print the running sum of the integral. A request to print (i.e., <u>RUN PROGRAM</u> alone) at other than the end of an interval will be ignored, and more values of y must be entered. However, it is easy to recognize when the end of an interval is reached: a series of dashes will be printed. To run a new case, <u>STOP</u> and go to Step 4.
NOTE: In Step 9, when the end of an interval has been reached, the x-increment size h may be manually changed by <u>h</u> \rightarrow <u>C</u> <u>EXECUTE</u> prior to entering the next y.		

EXAMPLES

Increment Constant	
X	Y
(x ₀) 0	(y 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

$$h = .25$$

$$\int_0^2 f(x) dx = 16.58$$

Increment Change	
X	Y
(x ₀) 0	(y ₀) 2
.25	2.8
.50	3.8
.75	5.2
1.00	7.0
1.50	12.1
2.00	20

$$h = .25$$

$$h = .5$$

$$\int_0^2 f(x) dx = 16.62$$

SIMPSON'S RULE

X0
Y0
0.0000
2.0000

INCREMENT
.2500

.2500 X₁
2.8000 Y₁
.5000 X₂
3.8000 Y₂

.7500 X₃
5.2000 Y₃
1.0000 X₄
7.0000 Y₄

1.2500
9.2000

1.5000
12.1000

1.7500
15.6000

2.0000
20.0000

INTEGRAL
X0 TO LAST X
16.5833

SIMPSON'S RULE

X0
Y0
0.0000
2.0000

INCREMENT
.2500

.2500 X₁
2.8000 Y₁
.5000 X₂
3.8000 Y₂

.7500 X₃
5.2000 Y₃
1.0000 X₄
7.0000 Y₄

1.5000 X₅
12.1000 Y₅

2.0000 X₆
20.0000 Y₆

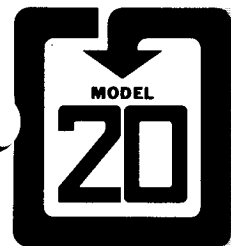
INTEGRAL
X0 TO LAST X
16.6167

PROGRAM LISTING

```

0:
CFG 13:ENT " TRA
PEZOID RULE?",Z;
IF FLG 13=0;1+A;
PRT " TRAPEZOID
RULE";GTO 7F
1:
CFG 13:ENT "SIMP
SON'S RULE?",Z;
IF FLG 13=0;2+A;
PRT " SIMPSON'S
RULE";GTO 7F
2:
CFG 13:ENT " SIM
PSON RULE 2?",Z;
IF FLG 13=0;3+A;
PRT " SIMPSON RU
LE 2";GTO 7F
3:
CFG 13:ENT "NEWT
ON-COTES 4?",Z;
IF FLG 13=0;4+A;
PRT " NEWTON-COT
ES 4";GTO 7F
4:
CFG 13:ENT " NEW
TON-COTES 5?",Z;
IF FLG 13=0;5+A;
PRT " NEWTON-COT
ES 5";GTO 7F
5:
CFG 13:ENT "NEWT
ON-COTES 6?",Z;6
+A;IF FLG 13;
GTO 0F
6:
PRT " NEWTON-COT
ES 6"F
7:
SPC 1;ENT "X0?",
X;ENT "Y0?",R0+R
A;0+YF
8:
PRT "      X0",
"      Y0",X,R0
;SPC 1F
9:
ENT "X INCREMENT
?",CF
10:
PRT "      INCREME
NT";C;SPC 1F
11:
SFG 1+B;RA+R0;
PRT " -----
-----";SPC 1F
12:
CFG 13:ENT "NEXT
Y?",RB;GTO +1;
IF FLG 13;GTO +0
;IF FLG 13;GTO 26
F
13:
CFG 1;PRT X+C+X,
RB;SPC 1;JMP 2A(
(B(B+A)+1+B)=1)-
1F
14:
C(R0+R1)/2+Y+YF
15:
GTO 11F
16:
C(R0+4R1+R2)/3+Y
+YF
17:
GTO 11F
18:
3C(R0+3(R1+R2)+R
3)/8+Y+YF
19:
GTO 11F
20:
2C(7(R0+R4)+32(R
1+R3)+12R2)/45+Y
+YF
21:
GTO 11F
22:
5C(19(R0+R5)+75(
R1+R4)+50(R2+R3)
)/288+Y+YF
23:
GTO 11F
24:
C(41(R0+R6)+216(
R1+R5)+27(R2+R4)
+272R3)/140+Y+YF
25:
GTO 11F
26:
PRT "      INTEGRA
L";"      X0 TO LAST
X";Y;SPC 1;GTO
11F
27:
END F

```

NUMERICAL INTEGRATION WITH UNEQUALLY
SPACED BASE POINTS, INTERPOLATION,
FIRST AND SECOND DERIVATIVES, AND
CURVE THROUGH POINTS USING SPLINE FUNCTIONS

Model 20
MATH PAC
III-5

This program will determine a curve $s(x)$ passing through a given set of data points $(x_1, y_1), (x_2, y_2), \dots, (x_n, y_n)$, where the x_i are distinct but need not be equally spaced, and then print the value of the integral,

$\int_{x_1}^{x_n} s(x)dx$, of this curve. The user then has the following options:

1. Plot the points and the curve $s(x)$ through them (with automatic data scaling), and a set of labeled axes.
2. Interpolate. For a given x , $x_1 \leq x \leq x_n$, print all of the following (in this order):

x	value of x
$s''(x)$	value of the second derivative of the curve at x
$s'(x)$	value of the first derivative of the curve at x
$s(x)$	value of the curve, $s(x)$, at x

Up to 17 points are allowed on the basic 9820A and up to 59 on the 9820A having the additional 256 internal registers.

The curve used to fit the given points is a third-degree natural spline function and derives its name from the fact that such a curve approximates the behavior of a mechanical spline used by draftsmen to draw a smooth curve through a set of points. The curve has the properties of having continuous first and second derivatives, and of being the "smoothest" curve through the given points in the sense that

$\sigma_k = \int_{x_1}^{x_n} [s^k(x)]^2 dx$ is made as small as possible for $k = 1, 2$. In fact, $s(x)$ is the unique curve with these

properties and is a piecewise function given by a polynomial of degree at most 3 in each of the intervals $[x_i, x_{i+1}]$, in general by a different polynomial in each such interval. These polynomial segments making up the curve $s(x)$ join smoothly at the juncture points x_i in that the two polynomials which meet at a given x_i have the same ordinate and the same first and second derivatives.

The process to determine $s(x)$ involves the solution of a set of simultaneous linear equations which is iteratively carried out by the method of successive overrelaxation. The accuracy tolerance, ϵ , to within which the parameters $s''(x_i)$ are to be found in this iteration is specified by the user.

For a derivation of the equations and the flow chart upon which this program is based, see Greville's article in reference 1. Reference 2 goes more into the theory of spline functions in general with applications in various other areas.

For users not having a plotter but wishing to make use of the other options in the program, an easy editing of Pass 2 of the program makes this possible: simply eliminate lines 0 through 12 of Pass 2, re-number the remaining lines starting with 0, and ignore Steps 7 and 8 of the User Instructions.

References: ¹ Ralston and Wilf, Mathematical Methods for Digital Computers, Volume II
(New York: John Wiley and Sons, Inc., 1967), pp. 156-168.

² T. N. E. Greville, Theory and Applications of Spline Functions
(New York: Academic Press, 1969).

ROM BLOCKS

1

2

Peripheral Control I*

3

INTERNAL REGISTERS

☒ 173 ☐ 429

PERIPHERALS

<input type="checkbox"/> 9860A MARKED CARD READER	<input type="checkbox"/> 9862A PLOTTER	<input type="checkbox"/> 9864A DIGITIZER	<input type="checkbox"/> _____
<input type="checkbox"/> 9861A TYPEWRITER	<input type="checkbox"/> 9863A PAPER TAPE READER	<input type="checkbox"/> _____	<input type="checkbox"/> _____

STEP	DISPLAY	INSTRUCTIONS
1.		<u>ERASE</u> <u>LOAD</u> <u>EXECUTE</u> , then alternately insert magnetic cards for Pass 1 and <u>EXECUTE</u> until completion of loading.
2.		<u>END</u> <u>RUN PROGRAM</u>
3. E?	ϵ	<u>RUN PROGRAM</u> ϵ = error tolerance (e. g. , 10^{-3} or 10^{-6}).
4. X	$\underline{x_i}$	<u>RUN PROGRAM</u> } data point entry (where $x_1 < x_2 < \dots < x_n$). <u>RUN PROGRAM</u> } <u>RUN PROGRAM</u> alone after all data is entered.
Y	$\underline{y_i}$	
		After calculations are completed and the integral is printed, go to Step 5.
5. PASS 2		<u>LOAD</u> <u>EXECUTE</u> , then alternately insert magnetic cards for Pass 2 and <u>EXECUTE</u> until completion of loading.
6.		<u>END</u> <u>RUN PROGRAM</u>
7. XMIN?		If no plot is to be made, <u>RUN PROGRAM</u> and go to Step 9.
8. XMIN?	$\underline{x_{min}}$	<u>RUN PROGRAM</u> } plot parameters, where $\underline{x_{min}} \leq x_i < \underline{x_{max}} \quad i = 1, \dots, n$ $\underline{y_{min}} \leq y_i < \underline{y_{max}} \quad i = 1, \dots, n$ 200 points are plotted (as specified in line 2, Pass 2).
XMAX?	$\underline{x_{max}}$	
YMIN?	$\underline{y_{min}}$	
YMAX?	$\underline{y_{max}}$	
9. EVALUATE AT X=?	\underline{x}	<u>RUN PROGRAM</u> . The following are printed in order: <div style="margin-left: 400px;"> x value of x $s''(x)$ second derivative at x $s'(x)$ first derivative at x $s(x)$ ordinate at x </div>
		Control returns to Step 9 for another x .
		*Not required if modifications noted at bottom of the previous page are made.

EXAMPLES

1. Using a tolerance $\epsilon = 10^{-3}$, use the program to fit a curve $s(x)$ through the following points and print the

value of the integral $\int_1^{14} s(x)dx$:

x	y
1	2
2.5	12
4	6
7	10
9	10.5
10	8.3
14	1

```
INTEGRAL
94.70860
```

If a plotter is available, plot the curve using:

$$\begin{aligned} x_{\min} &= 1 & y_{\min} &= 0 \\ x_{\max} &= 15 & y_{\max} &= 12 \end{aligned}$$

2. The program may be of aid in lofting associated with boat designing. Using a tolerance $\epsilon = 10^{-3}$, plot half of the top view of a boat hull through the points:

x	y
0	2
3	2.5
7	2
10	0

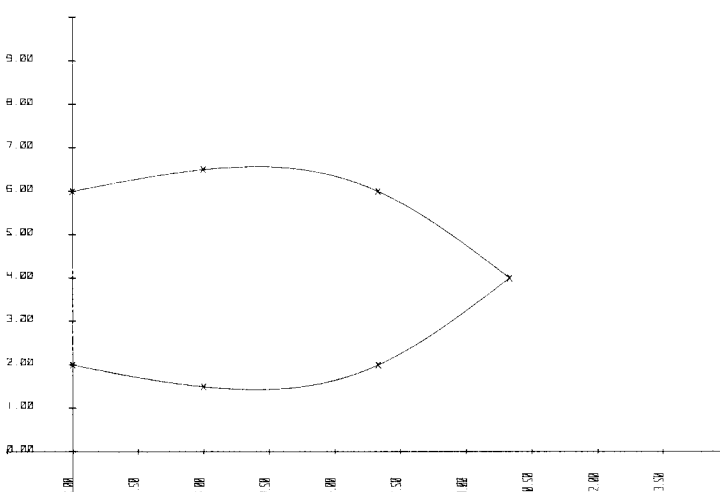
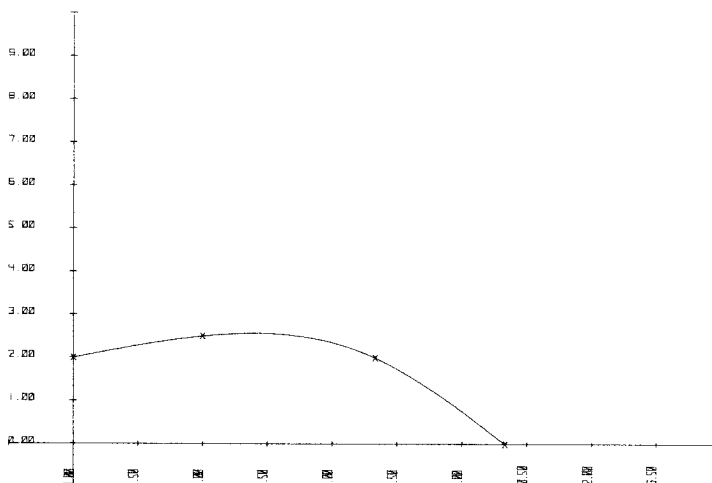
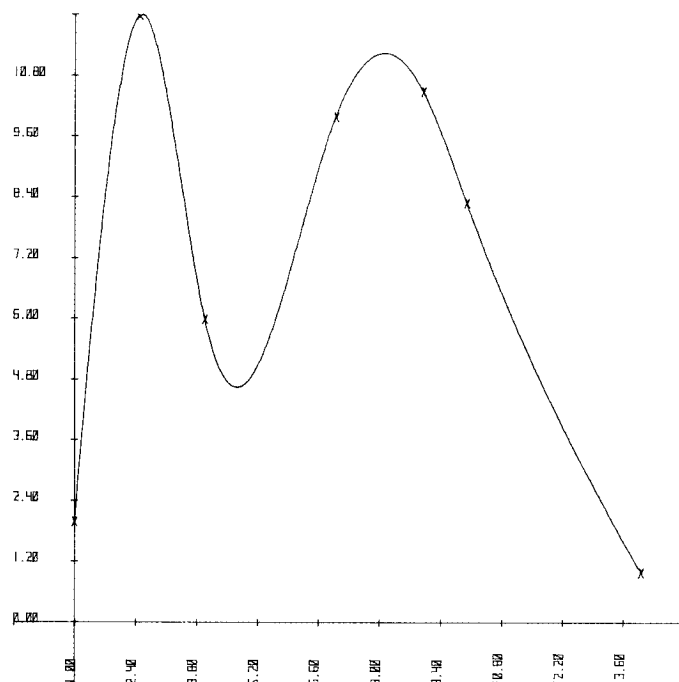
$$\begin{aligned} \text{Use } x_{\min} &= 0 & y_{\min} &= 0 \\ x_{\max} &= 15 & y_{\max} &= 10 \end{aligned}$$

Then find the ordinates, y , corresponding to $x=5$ and $x=8$.

```
5.000000  X
-1.136866 S''(x)
-1.100000 S'(x)
2.527733 S(x)

8.000000  X
-1.142538 S''(x)
-1.631022 S'(x)
1.452151 S(x)
```

Note: The full top view may be obtained as shown to the right by running the program twice: first using the above x values with the y values shifted up 4 units, then running the program using the same x values and $4-y$ with y as in the tables.



REGISTERS			
A	n		
B	i		
C	ω , Integral		
X	η , x		
Y	Temporary		
Z	Temporary		
0	ε , Increment		
1	x_1		
2	y_1		
3	x_2		
4	y_2		
	.		
	.		
	.		
	h_i, s_i''		
	.		
	.		
	.		
	δy_i		
	.		
	.		
	.		
	$s''(x_i)$		
	.		
	.		
	.		
	g_i		
	.		
	.		
	.		
			FLAGS
		0	
		1	Used
		2	
		3	
		4	
		5	
		6	
		7	
		8	
		9	
		10	
		11	
		12	
		13	Used
		14	
		15	

PROGRAM LISTING

Pass 1

```

0:
ENT "E?",R(0+A);
CFG 13+
1:
ENT X;IF FLG 13;
JMP 2+
2:
ENT Y;PRT X+R(2(
A+1+A)-1),Y+R(2A
);SPC 1;JMP -1+
3:
0+R(4A+1)+R(5A)+
B;4(2-+3)+C+
4:
(R(2(B+1+B)+2)-R
(2B))/R(2B+1)-R
(2B-1)+R((3A+B+Z
)-A))+RZ;JMP B>1
+
5:
(2(RZ-R(Z-1)))/R
(Z-1-A)+R(Z-A))+
R(Z+A))3/2+R(Z+2
A)+
6:
IF A-1>B;GTO -2+
7:
0+X;2+B+
8:
C(((R((4A+B+Z)+
1)+Y)-R(Z-1))/1
+R(Z-2A)/R(Z-2A-
1))-Y)/2-RZ+R(Z+
A))+Y+
9:
IF r(YY)>X;r(YY)
+X+
10:
RZ+Y+RZ;IF (B+1+
B)≠A;GTO -2+
11:
IF R0≤X;GTO -4+
12:
0+C;1+B+
13:
(R((4A+B+Z)-2A)+
Y)(R(2B)+R(2B+2)
)/2-YYY(RZ+R(Z+1
))/24+C+C;JMP A=
(B+1+B)+
14:
1+B+

```

```

15:
(R((4A+B+Z)+1)-R
Z)/R(Z-2A+Z)+RZ;
JMP (B+1+B)=A+
16:
PRT "      INTEGRA
L",C;SPC 2+
17:
DSP "PASS 2"+
18:
END +

```

Pass 2

```

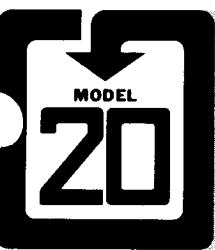
0:
CFG 1;CFG 13;
ENT "XMIN?",X;
IF FLG 13;JMP 13
+
1:
FLT 2;FXD 2;ENT
"XMAX?",C,"YMIN?
",Y+B,"YMAX?",Z+
2:
(C-X)/200+R0+
3:
SCL 1.1X-.1C,C,1
.1Y-.1Z,Z;AXE X,
Y,.1(C-X),.1(Z-Y
)+
4:
LTR 1.1X-.1C,B,2
11;PLT B;JMP (B+
.1(Z-Y)+B)>.95(Z
-Y)+Y+
5:
X+B+
6:
LTR B,1.1Y-.1Z,2
12;PLT B;JMP (B+
.1(C-X)+B)>.95(C
-X)+X+
7:
1+B;FLT 9;.00352
(C-X)+X;.00704(Z
-Y)+Y+
8:
LTR R(2B-1)-X,R(
2B)-Y,211;PLT "X
";JMP (B+1+B)>A+
9:
R(1+B)+X+
10:
GSB +5+

```

```

11:
PLT X,Y;IF (X+R0
+X)≤R(2A-1);JMP
-1+
12:
PEN +
13:
SFG 1+B;ENT "EVA
LUATE AT X=?",X;
GSB +2+
14:
JMP -1+
15:
IF B>A;RET +
16:
GTO +1;IF A=B;
GTO +6;IF (X-R(2
A-1)+Z)=0;JMP 2+
17:
IF (0>(X-R(2B-1)
+Y))+0≤(X-R(2B+
1)+Z));B+1+B;
JMP -2+
18:
R(4A+B)+YR(2A+B)
+C;IF FLG 1;PRT
X,C+
19:
(R(4A+B)+R(4A+B+
1)+C)/6+C+
20:
R(2B)+YR(3A+B)+Y
ZC+Y+
21:
IF FLG 1;PRT R(3
A+B)+(X-R(2B-1)+
Z)C+Z(X-R(2B-1))
R(2A+B)/6,Y;SPC
1+
22:
RET +
23:
END +

```

CURVE FITTING BY CHEBYSHEV POLYNOMIALS

This program¹ fits a least-squares curve to a set of given data points $(x_1, y_1), (x_2, y_2), \dots, (x_n, y_n)$ where the x_i lie in an interval (a, b) and are equally spaced. The user specifies a degree m and the program outputs the coefficients a_0, a_1, \dots, a_m of a polynomial $P(x) = a_0 + a_1x + \dots + a_mx^m$ passing near or through each input point.

The program determines the a_i by considering $P(x)$ as a linear combination of Chebyshev polynomials $T_i(x)$, $P(x) = c_0T_0(x) + c_1T_1(x) + \dots + c_mT_m(x)$ and applying the least-squares criterion to the expression

$$S = \sum_{i=1}^n \left(y_i - P(x) \right)^2 \text{ to give a system of simultaneous equations } \frac{\partial S}{\partial c_j} = 0, j=0, 1, \dots, m \text{ from which the } c_i$$

can be determined. Calculation of the c_i is facilitated by using the orthogonality properties of the Chebyshev polynomials and evaluating $P(x)$ at special points

$$\bar{x}_i = \frac{\cos(2i-1)\pi}{2(m+1)} \text{ within the interval } (-1, 1) \text{ to force off-diagonal elements to be zero in this system of equations.}$$

Corresponding values \bar{y}_i are needed in the system of equations in order to be able to solve for the c_i . The program obtains these by applying a linear transformation to the x_i to bring the x_i within the interval $(-1, 1)$ and then using these transformed values and applying the Lagrange interpolation formula to the \bar{x}_i to obtain the \bar{y}_i . The system of equations is then easily solved for the c_i . The program then applies a linear transformation to \bar{x} to change the expression $P(\bar{x}) = c_0T_0(\bar{x}) + c_1T_1(\bar{x}) + \dots + c_mT_m(\bar{x})$ to the form $P(x) = a_0 + a_1x + \dots + a_mx^m$ over the original interval (a, b) .

If a plotter is available, provisions are made to use the program in conjunction with the General Function Plot, program III-7, to plot $P(x)$ without any data reentry as shown in the Example.

Currently the program is set up for a 9820A with 429 total registers. However, the program may still be used on a 9820A with 173 registers by segmenting the program into three parts: part 1 consisting of lines 0 through 19, part 2 consisting of lines 20 through 35, and part 3 consisting of lines 36 through 56. Re-number the lines starting from 0 for each part, add an END statement as the terminal line, and run each as a separate program. First run part 1, and when calculations are completed (indicated by a non-blank display) load part 2 and run it by LOAD EXECUTE END RUN PROGRAM and then run part 3 the same way. DO NOT press ERASE prior to loading part 2 or part 3.

If n denotes the number of data pairs, m the polynomial degree specified, and R the number of available R -registers, then problem size is determined by the relation $2n + 3m \leq R - 3$. With 173 registers, this gives $2n + 3m \leq 78$ and with 429 registers, this gives $2n + 3m \leq 217$. Reduce these numbers by 6 if the Peripheral Control I block is used.

¹ Written by Frank Yockey of Hewlett-Packard Calculator Labs, based on the FORTRAN program given in the following reference:

Shan S. Kuo, Numerical Methods and Computers, (Palo Alto: Addison-Wesley, 1965), pp. 228-238.

ROM BLOCKS

☐ 1

Mathematics

☒ 2 Trigonometric

(If plotting)

☐ 3 Peripheral Control I

INTERNAL REGISTERS

☐ 173 ☒ 429 ¹

PERIPHERALS

<input type="checkbox"/> 9860A MARKED CARD READER	<input type="checkbox"/> 9862A PLOTTER	<input type="checkbox"/> 9864A DIGITIZER	<input type="checkbox"/> _____
<input type="checkbox"/> 9861A TYPEWRITER	<input type="checkbox"/> 9863A PAPER TAPE READER	<input type="checkbox"/> _____	<input type="checkbox"/> _____

STEP	DISPLAY	INSTRUCTIONS
1.		<u>ERASE</u> <u>LOAD</u> <u>EXECUTE</u>
2.		Alternately insert magnetic cards and <u>EXECUTE</u> until NOTE 14 no longer appears, indicating completion of loading.
3.		(Optional) <u>FIXED N n EXECUTE</u> , or <u>FLOAT N n EXECUTE</u>
4.		² <u>END</u> <u>RUN PROGRAM</u>
5. DEGREE?	<u>m</u>	<u>RUN PROGRAM</u> m = polynomial degree desired
6. X1?	<u>x₁</u>	<u>RUN PROGRAM</u> x ₁ = first x value
7. DELTA X?	<u>Δx</u>	<u>RUN PROGRAM</u> Δx = x increment
8. Y?	<u>y_i</u>	<u>RUN PROGRAM</u> i = 1, 2, ..., n
<p>When all y_i have been entered, press <u>RUN PROGRAM</u>. The display blanks while calculations are made and the coefficients a₀, a₁, a₂, ..., a_m in the expression $P(x) = a_0 + a_1x + \dots + a_mx^m$ are then printed in order. Return to Step 4 for a new problem. ³</p>		
<p>¹ Only 173 registers are required if the program is segmented as described at bottom of previous page.</p>		
<p>² If plotting of the data points is desired, follow procedure I shown in the example. Do this prior to Step 4 of the instructions above.</p>		
<p>³ If plotting of the polynomial is desired, follow procedure II shown in the example.</p>		

EXAMPLE

Use the program to fit a parabola through the points

x	y	x	y
1	2	6	11
2	5	7	11
3	7	8	10
4	8	9	9
5	10	10	8

Input: degree = 2
 $x_1 = 1$
 $\Delta x = 1$

Output

COEFFICIENTS

0.0000
 -1.5601

1.0000
 3.8159

2.0000
 -1.2910

I. DATA PLOTTING

For plotting the input data, load the program then press:

x_{\min} → A EXECUTE

x_{\max} → B EXECUTE

y_{\min} → C EXECUTE

y_{\max} → Y EXECUTE

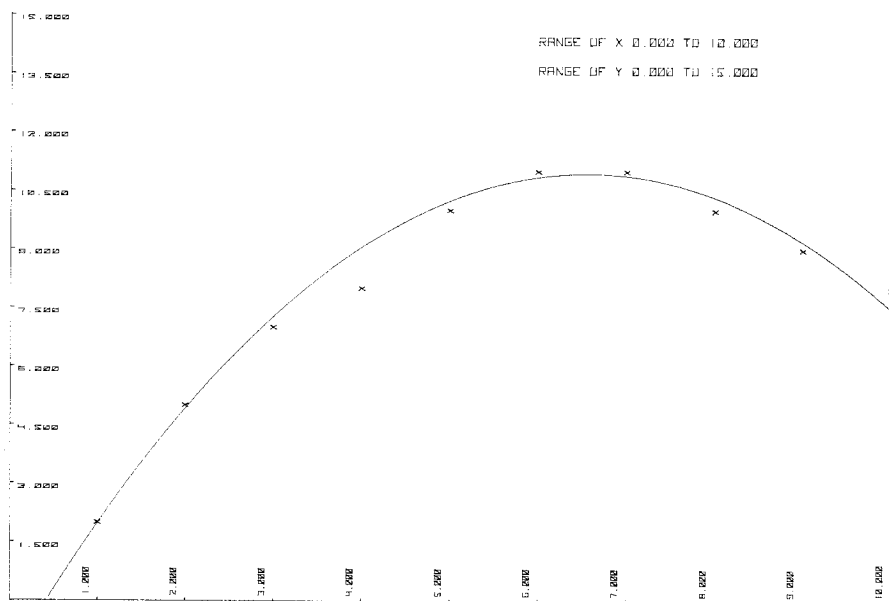
For this example, use: $x_{\min} = 0$, $x_{\max} = 10$,
 $y_{\min} = 0$, $y_{\max} = 15$.

Modify line 2 as shown below and insert the new line 3 (key it in and press INSERT STORE).

```
2:
  ENT "Y?",RZ;GTO
  +2;IF FLG 13=0;
  PRT X,RZ;SPC 1;X
  +R9+X;Z+1+Z+R1;
  JMP 1F
3:
  SCL A,B,C,Y;LTR
  X-R9-.0032(B-A),
  R(Z-1)-.0032(Y-C
  ),111;PLT "X";
  JMP -1F
4:
  SPC 1;PRT "  COE
  FFICIENTS";SPC 1
  F
```


II. POLYNOMIAL PLOT

To plot the polynomial, END EXECUTE and load the General Function Plot, program III-7, (DO NOT PRESS ERASE). Key in the below function and then run the program using the same x_{\min} , x_{\max} , y_{\min} , and y_{\max} as above and no. of points = 100.



```
25:
  "F" F
26:
  R(R7+(R6-1+R5))÷
  YF
27:
  YX+R(R7+(R5-1+R5
  ))+Y;JMP R5=0F
28:
  RET F
29:
  END F
```

REGISTERS

A	Counter
B	Counter
C	Counter
X	x, Working Register
Y	Working Register
Z	Temporary
0	Data Pointer
1	Norm. Data Pointer
2	Root Pointer
3	Coeff. Pointer
4	Temp. Area Pointer
5	Product
6	m + 1
7	n
8	x1
9	Δx
10	SUM 1
11	SUM 2
	
Original Data	
Normalized Data	
Roots	
Chebyshev Coefficients	
Intermediate Storage	
	0
	1
	2
	3
	4
	5
	6
	7
	8
	9
	10
	11
	12
	13 Used
	14
	15

PROGRAM LISTING

```

0:
ENT "DEGREE?", Z+
1→R6, "X1?", R8→X,
"DELTA X?", R9;
CFG 13; TBL 2;
1:
PRT "      DEGREE
"; SPC 1; PRT Z;
SPC 2; PRT "
  X", "      Y";
SPC 1; 12→R0→Z;
2:
ENT "Y?", RZ; IF
FLG 13=0; PRT X, R
Z; SPC 1; X+R9→X; Z
+1→Z→R1; JMP 0;
3:
SPC 1; PRT "      COE
FFICIENTS"; SPC 1
;
4:
(((R1-12→R7)+R(1
→A)+R2)+R6→R3)+R
6→R4;
5:
COS ((π/2)(2A-1)
/R6)+R(R6-A+R2);
JMP (A+1→A)→R6;
6:
R7-1→A; 2/A→B; -1→
RR1; R1→C;
7:
RC+B→R(C+1); JMP
(C+1→C)=R2-1;
8:
0→A→B;
9:
IF R(R2+A)→R(R1+
B); GTO "150";
10:
"151"; (R(R2+A)-R
(R1+B-1→C))/(R(C
+1)-RC)→Y;
11:
IF B>1; GTO +2;
12:
Y(R(R0+B+C)-R(C-
1))+R(C-1)→R(R4+
A); GTO "157";
13:
IF R7≤B+1; GTO -1
;

```

```

14:
-Y(Y-1)(Y-2)R(R0
+B-2→C)/6+(YY-1)
(Y-2)R(C+1)/2→Z;
15:
Z-(Y+1)(Y-2)YR(C
+2)/2+Y(YY-1)R(C
+3)/6→R(R4+A);
16:
"157"; A+1→A;
17:
IF R6≤A; GTO +3;
18:
IF R(R2+A)≤R(R1+
B); GTO "151";
19:
"150"; IF (B+1→B)
≤R7-1; GTO -10;
20:
0→A;
21:
0→Y; 0→B; IF A=1;
GTO "265";
22:
IF A>1; GTO "270";
23:
Y+R(R4+B)→Y; JMP
(B+1→B)→R6-1;
24:
GTO "275";
25:
"265"; Y+R(R2+B)R
(R4+B)→Y; JMP (B+
1→B)→R6+1;
26:
GTO "275";
27:
"270"; 1→RR1;
28:
R(R2+B)→R(R1+1);
2→C;
29:
2R(R2+B)R(R1+C-1
)-R(R1+C-2)→R(R1
+C); JMP (C+1→C)→
A;
30:
Y+R(R4+B)R(R1+A)
→Y; IF (B+1→B)≤R6
-1; GTO -2;

```

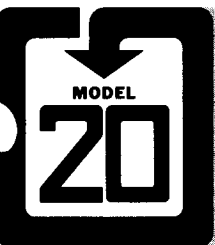
```

31:
"275"; 2Y/R6→R(R3
+A); IF (A+1→A)≤R
6-1; GTO -10;
32:
RR3/2→RR3;
33:
(R8→C)+R9(R7-1)→
B;
34:
(B+C)/2→R10; (B-C
)/2→R11;
35:
RR3→RR4; R(R3+1)→
R(R4+1); 0→C;
36:
IF R6≤2; GTO "597";
37:
0→R(R1+C)→R(R0+C
)→R(R4+C+2); JMP
(C+1→C)→R6-1;
38:
1→R(R1+1); 2→C;
39:
COS (πC/2)→RR0; 1
→B;
40:
2R(R1+B-1)-R(R0+
B)→R(R0+B); JMP (
B+1→B)→C;
41:
0→B;
42:
R(R4+B)+R(R3+C)R
(R0+B)→R(R4+B); R
(R1+B)→A; R(R0+B)
→R(R1+B); A→R(R0+
B);
43:
IF (B+1→B)≤C;
GTO -1;
44:
IF (C+1→C)≤R6-1;
GTO -5;
45:
"597";
46:
RR4→RR0; 1→C;
47:
0→R(R0+C); JMP (C
+1→C)=R6;
48:
1→C;

```

PROGRAM LISTING

```
49:
"580":R(R0+C)+R(
R4+C)/R11+C+R(R0
+C)11+A+R5+B:C+1
+Y+
50:
B+X:XR5+R5:R5R11
+C+Z:A(Y-X)+A+
51:
R(R0+C-B)+A*R10+
B*(-1)+E*R(R4+C)
/Z+R(R0+C-B):IF
(E+1+B)<C:GTO -1
+
52:
IF (C+1+C)=R6:
GTO "580"+
53:
0+A+
54:
PRT A,R(R0+A):
SPC #JMP (A+1+A)
=R6+
55:
R0+R7:SPC 8+
56:
END +
```



GENERAL FUNCTION PLOT

This program¹ plots a given function over a specified range and then, at the user's option, draws a set of axes and labels them. An automatic scaling feature is included which automatically calculates the range of y for the designated range of x, the plot then being made over this range. Or, if the user chooses, the range of y may be designated in the input. This would usually be done when the user wants the labeled values along the y-axis to come out in even increments.

¹ Written by Frank Yockey of Hewlett-Packard Calculator Labs.

ROM BLOCKS

1

2

Peripheral Control I

3

INTERNAL REGISTERS

☒ 173 ☐ 429

PERIPHERALS

☐ 9860A MARKED CARD READER☒ 9862A PLOTTER☐ 9864A DIGITIZER☐☐ 9861A TYPEWRITER☐ 9863A PAPER TAPE READER☐☐

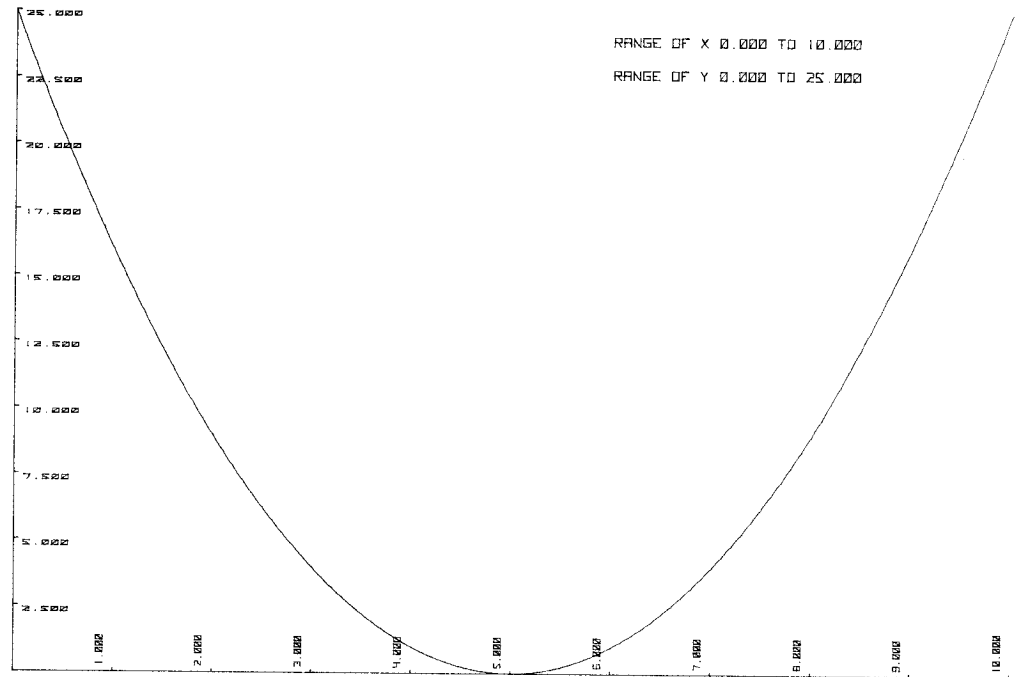
STEP	DISPLAY	INSTRUCTIONS
1.		<u>ERASE</u> <u>LOAD</u> <u>EXECUTE</u>
2.		Alternately insert magnetic cards and <u>EXECUTE</u> until NOTE 14 no longer appears, indicating completion of loading.
3.		Set radians, if desired.
4.		<u>GO TO</u> <u>2</u> <u>6</u> <u>EXECUTE</u>
5.		Using register X to represent the variable x in the expression for f(x), key in: <u>f(x)</u> <u>→</u> <u>Y</u> <u>STORE</u> <u>RETURN</u> <u>STORE</u> <u>END</u> <u>STORE</u>
6.		<u>END</u> <u>RUN PROGRAM</u>
7. X MIN?		<u>x_{min}</u> <u>RUN PROGRAM</u>
8. X MAX?		<u>x_{max}</u> <u>RUN PROGRAM</u>
9. NO. OF POINTS?		<u>n</u> <u>RUN PROGRAM</u>
10. Y MIN?		<u>y_{min}</u> <u>RUN PROGRAM</u> (or <u>RUN PROGRAM</u> alone for auto-scale).
11. Y MAX?		<u>y_{max}</u> <u>RUN PROGRAM</u>
The function is now plotted. If auto-scale has been selected, there will be a slight delay prior to the plot while the minimum and maximum y values are calculated by the program.		
12. AXES?		<u>RUN PROGRAM</u> to get axes or <u>1</u> <u>RUN PROGRAM</u> to return to Step 7 to plot the same function over a new range. If axes have been designated, they will be plotted and labeled after which control returns to Step 7.

EXAMPLES

1. Plot $y = (x-5)^2$ with $x_{\min} = 0$, $x_{\max} = 10$, number of points = 100, and letting the program calculate y_{\min} and y_{\max} .

Program Lines:

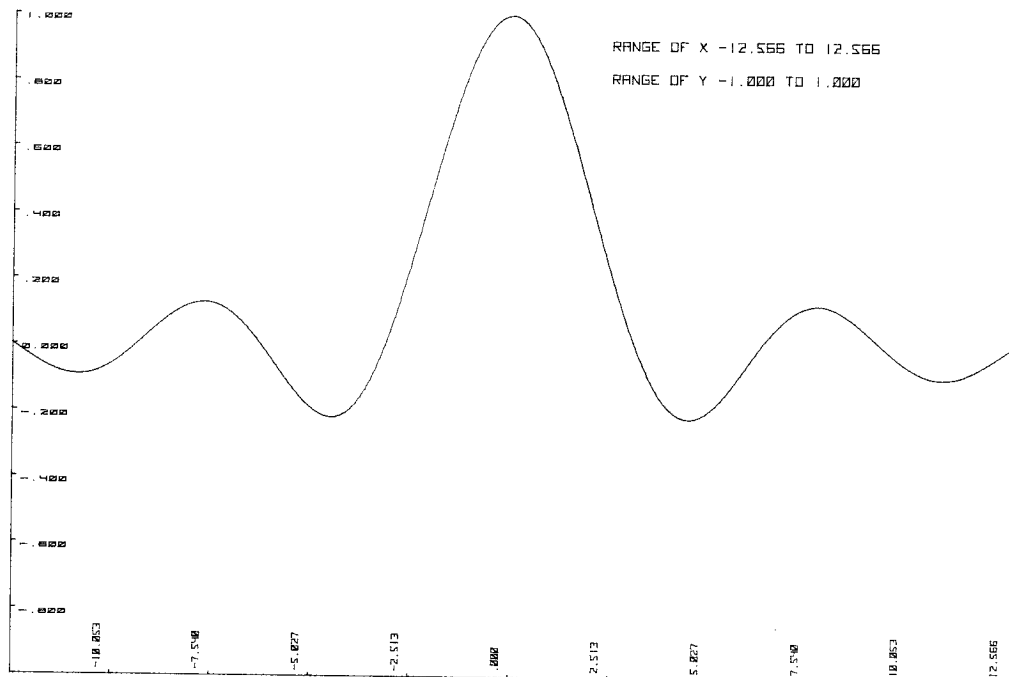
```
25:
  "F"↑
26:
  (X-5)(X-5)+Y↑
27:
  RET ↑
28:
  END ↑
```



2. Plot $y = \sin x/x$ with $x_{\min} = -4\pi$, $x_{\max} = 4\pi$, $y_{\min} = -1$, $y_{\max} = 1$, number of points = 200, and with the calculator set in radians. (This example requires the ~~Trigonometric~~ ROM block).

Program Lines:

```
25:
  "F"↑
26:
  SIN X/X+Y↑
27:
  RET ↑
28:
  END ↑
```



REGISTERS

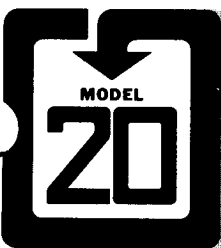
[illegible]

PROGRAM LISTING

```

0:
ENT "XMIN?",R0,"
XMAX?",R1,"NO. 0
F POINTS?",R2+
1:
CFG 13:ENT "YMIN
?",B:GTO +1:IF
FLG 13=0:ENT "YM
AX?",A:JMP 7+
2:
R0+X:(R1-R0)/R2+
Z:GSB "F"+
3:
Y+A+B:GTO +4+
4:
GSB "F"+
5:
IF Y>A:Y+AF
6:
IF Y<B:Y+BF
7:
IF (X+Z+X)<R1:
GTO -3+
8:
SCL R0,R1,B,A:
PRT "      XMIN"
,"      XMAX",R0
,R1:SPC +
9:
PRT "      YMIN"
,"      YMAX",B:
A:SPC 8+
10:
R0+X:(R1-R0)/R2+
Z+
11:
GSB "F"+
12:
PLT X,Y:IF (X+Z+
X)<R1:GTO -1+
13:
PEN :CFG 13:ENT
"AXES?",Z:IF
FLG 13=0:GTO 0+
14:
SCL 0,10,0,10:
AXE 0,0,1,1+
15:
FLT 2+
16:
B+C:-.1+X:IF ((A
-B)/10+Z)<1E2:
FXD 0:IF Z<9:
FXD 3:IF Z<1E-2:
FLT +
17:
IF (1+X+X)<10:
LTR .1,X,111:
PLT C+Z+C:GTO +0
+
18:
LTR 6,9,211+
19:
PLT "RANGE OF Y
":PLT B:PLT " TO
":PLT A+
20:
FLT :-.1+X:R0+C:
IF ((R1-R0)/10+Z
)<1E2:FXD 0:IF Z
<9:FXD 3:IF Z<1E
-2:FLT +
21:
IF (1+X+X)<10:
LTR X,.1,112:
PLT C+Z+C:GTO +0
+
22:
LTR 6,9.5,211+
23:
PLT "RANGE OF X
":PLT R0:PLT " T
O ":PLT R1+
24:
GTO 0+
25:
"F"+
26:
END +

```

LAGRANGE INTERPOLATION FOR UNEQUALLY SPACED DATA

Model 20
MATH PAC
III-8

If $n+1$ data points $(x_0, y_0), (x_1, y_1), \dots, (x_n, y_n)$ are given, this program will evaluate at any x_j the unique polynomial $P(x) = a_0 + a_1x + \dots + a_nx^n$ of degree n which passes through the given $n+1$ points, $n \leq 60$.

As shown in the reference, $P(x)$ may be expressed in the form

$$P(x) = \sum_{k=0}^n \left[\prod_{\substack{i=0 \\ i \neq k}}^n \left(\frac{x - x_i}{x_k - x_i} \right) \right] y_k$$

known as the Lagrange interpolation formula, and this is the form used in the program to find $P(x_j)$ for a given x_j .

In the Example section is shown an example of extrapolating intermediate values of $\ln x$ using several values obtained from a table covering the interval of interest.

Reference:

Shan S. Kuo, Numerical Methods and Computers (Reading, Massachusetts: Addison Wesley Publishing Company, 1965).

ROM BLOCKS

1	2	3

INTERNAL REGISTERS

☒ 173 ☐ 429

PERIPHERALS

<input type="checkbox"/> 9860A MARKED CARD READER	<input type="checkbox"/> 9862A PLOTTER	<input type="checkbox"/> 9864A DIGITIZER	<input type="checkbox"/> _____
<input type="checkbox"/> 9861A TYPEWRITER	<input type="checkbox"/> 9863A PAPER TAPE READER	<input type="checkbox"/> _____	<input type="checkbox"/> _____

STEP	DISPLAY	INSTRUCTIONS
1.		<u>ERASE</u> <u>LOAD</u> <u>EXECUTE</u>
2.		Alternately insert magnetic cards and <u>EXECUTE</u> until NOTE 14 no longer appears, indicating completion of loading.
3.		(Optional) <u>FIXED N n EXECUTE</u> , or <u>FLOAT N n EXECUTE</u> as desired.
4.		<u>END</u> <u>RUN PROGRAM</u>
5. GIVEN X VALUE?	x_i	<u>RUN PROGRAM</u>
6. GIVEN Y VALUE?	y_i	<u>RUN PROGRAM</u>
		for $i = 0, 1, \dots, n$
		After all of the given points have been entered, <u>RUN PROGRAM</u> and go to Step 7.
7. EVALUATE AT X= ?	x	<u>RUN PROGRAM</u>
		The value of x entered and the corresponding interpolated value $y = P(x)$ will be printed and control returned to Step 7 for another x . To run a new problem, <u>STOP</u> and return to Step 4.

EXAMPLE

Using the following values of $\ln x$ obtained from a table, find the value of $\ln x$ at the points, $x = 1.981$, 1.9825 , 1.985 , and 1.987 .

Given Values	
x	$\ln x$
1.980	.6830968447
1.982	.6841064359
1.983	.6846108495
1.986	.6861225656

The answers agree with the actual values of $\ln x$ to the full 10 digits printed.

GIVEN
VALUES ARE

X
Y
1.980000000E 00
6.830968447E-01

X
Y
1.982000000E 00
6.841064359E-01

X
Y
1.983000000E 00
6.846108495E-01

X
Y
1.986000000E 00
6.861225656E-01

INTERPOLATED
VALUES ARE

X
Y
1.981000000E 00
6.836017677E-01

X
Y
1.982500000E 00
6.843586745E-01

X
Y
1.985000000E 00
6.856189141E-01

X
Y
1.987000000E 00
6.866259636E-01

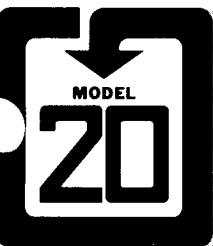
[illegible]

PROGRAM LISTING

```

0:
Q←A:CFG 13:SPC 4
:PRT "      GIVEN
"      VALUES ARE
"
1:
SPC 1:A+1→A:ENT
"GIVEN X VALUE?"
,R(2A-1):IF FLG
13:GTO +2F
2:
ENT "GIVEN Y VAL
UE?":R(2A):PRT "
      X":
      Y":R(2A-1):R(2
A):GTO 1F
3:
SPC 3:PRT "      INT
ERPOLATED":      V
ALUES ARE"F
4:
SPC 1:ENT "EVALU
ATE AT X= ?":XF
5:
Q←C→B:F
6:
IF (B+1→B)=A:
PRT "      X":
      Y":X,C:
GTO -2F
7:
Q←Z:1→Y:F
8:
IF (Z+1→Z)=A:C+Y
R(2B)→C:GTO -2F
9:
IF Z=B:GTO -1F
10:
Y(X-R(2Z-1))/(R(
2B-1)-R(2Z-1))+Y
:GTO -2F
11:
END F

```

FOURIER SERIES COEFFICIENTS FOR UNEQUALLY SPACED DATA POINTS

Model 20
MATH PAC
III-9

This program¹ calculates the Fourier Series coefficients a_n and b_n that represent a periodic time function $f(t)$ represented by discrete x, y pairs of data points over a period T . The x coordinate values have to be entered in ascending order but need not be equally spaced. The Fourier Series representation of $f(t)$ is:

$$f(t) = a_0 + \sum_{n=1}^{\infty} a_n \cos \frac{2\pi nt}{T} + b_n \sin \frac{2\pi nt}{T}$$

where

$$a_0 = \frac{1}{T} \int_{t_0}^{t_0+T} f(t) dt$$

$$a_n = \frac{2}{T} \int_{t_0}^{t_0+T} f(t) \cos \frac{2\pi nt}{T} dt \quad n = 1, 2, 3, \dots$$

$$b_n = \frac{2}{T} \int_{t_0}^{t_0+T} f(t) \sin \frac{2\pi nt}{T} dt \quad n = 1, 2, 3, \dots$$

The program evaluates the coefficients by numerically integrating using a parabolic fit through each grouping of three consecutive points. Execution time is dependent upon the number of pairs of data points and the number of coefficients desired. Also, if n denotes the highest coefficients a_n, b_n desired and k the number of data pairs, problem size is subject to the constraint $n + k \leq 116$ (or ≤ 113 if the Peripheral Control I block is in the calculator). For example, 100 data pairs could be entered and 16 values of a_n and b_n obtained.

If a plotter is available, the General Function Plot program, III-7, may be used in conjunction with this program to produce an approximating Fourier Series plot, an $f(t)$ frequency spectrum plot, and a phase spectrum of $f(t)$ plot. The necessary programming is included in the Example section.

¹ Based on a FORTRAN program written by Ivar Larson, Hewlett-Packard Calculator Marketing, New Products Development.

ROM BLOCKS

(If plotting)

INTERNAL REGISTERS

☐☒ Trigonometric☐ Peripheral Control I☐ 173 ☒ 429

1

2

3

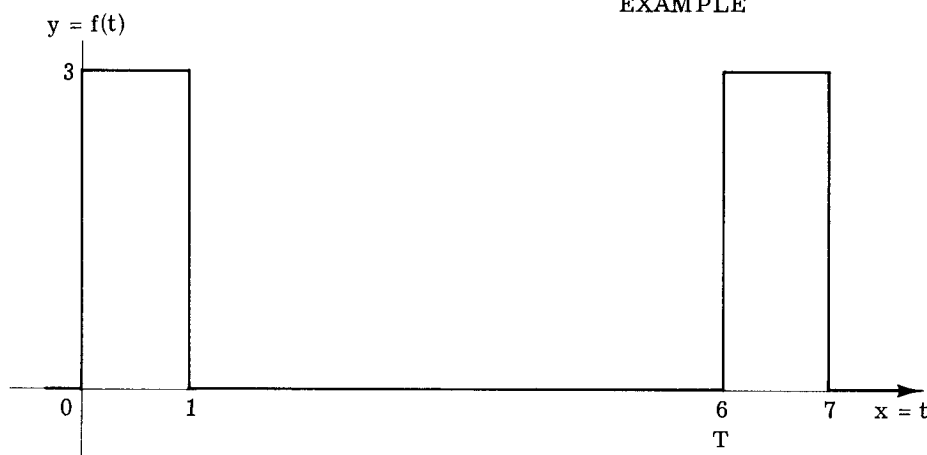
PERIPHERALS

☐ 9860A MARKED CARD READER
 ☐ 9862A PLOTTER
 ☐ 9864A DIGITIZER
 ☐ _____

☐ 9861A TYPEWRITER
 ☐ 9863A PAPER TAPE READER
 ☐ _____
 ☐ _____

STEP	DISPLAY	INSTRUCTIONS
1.		<u>ERASE</u> <u>LOAD</u> <u>EXECUTE</u>
2.		Alternately insert magnetic cards and <u>EXECUTE</u> until NOTE 14 no longer appears, indicating completion of loading.
3.		(Optional) <u>FIXED N</u> <u>n</u> <u>EXECUTE</u> , or <u>FLOAT N</u> <u>n</u> <u>EXECUTE</u> as desired.
4.		<u>END</u> <u>RUN PROGRAM</u>
5.	FINAL N?	<u>n</u> <u>RUN PROGRAM</u> indicating the highest coefficients a_n , b_n desired.
6.	X?	<u>x_i</u> <u>RUN PROGRAM</u> Data pair entry with $x_1 < x_2 < x_3 < \dots$
	Y?	<u>y_i</u> <u>RUN PROGRAM</u>
		After all data pairs have been entered, press <u>RUN PROGRAM</u> . The a_n and b_n will be calculated and printed. Execution time depends on n and the number of points entered. For another case, return to Step 4.

EXAMPLE



Using the x, y pairs below selected from the above rectangular wave form, find the first 10 values of a_n and b_n of the Fourier Series representing this wave form.

(Note: If plotting is to be done, be sure to place the Peripheral Control I block in place prior to Step 1 in the User Instructions).

Input Data

Final $n = 10$

x	y	x	y	x	y	x	y
0	1.5	.8	3	2.5	0	5	0
.1	3	.9	3	2.75	0	5.2	0
.15	3	1	1.5	3	0	5.4	0
.2	3	1.1	0	3.5	0	5.8	0
.25	3	1.2	0	3.7	0	5.85	0
.3	3	1.3	0	3.9	0	5.9	0
.4	3	1.5	0	4	0	6	1.5
.5	3	1.7	0	4.25	0		
.6	3	1.9	0	4.5	0		
.7	3	2	0	4.8	0		

COEFFICIENTS

N	
AN	
BN	
0.00000000	
.50000000	a_0
1.00000000	
.82623580	a_1
.47682981	b_1
2.00000000	
.41197091	a_2
.71317318	b_2
3.00000000	
.00000000	a_3
.63073598	b_3
4.00000000	.
-.20361381	.
.35201543	.
5.00000000	
-.16140575	
.09246861	
6.00000000	
.00000000	
-.00072307	
7.00000000	
.11229009	
.06417231	
8.00000000	
.09651357	
.16664684	
9.00000000	
.00000000	
.19360269	
10.00000000	
-.07365751	a_{10}
.12756916	b_{10}

The data pairs must be entered such that each succeeding x increases in value. About 5 minutes of calculation time is required for this example. Since the data points and coefficients a_n, b_n are stored, the General Function Plot program, III-7, may be used to produce the following plots:

Approximating Fourier Series Plot

Frequency Spectrum of $f(t)$ Plot

Phase Spectrum of $f(t)$ Plot

The function routines "F" used in the General Function Plot program are shown on the next page.

EXAMPLE (Continued)

CAUTION: DO NOT press ERASE in Step 1 of the User Instructions of the General Function Plot program. Instead, press END EXECUTE and then proceed with the program loading and remaining User Instructions.

1. Approximating Fourier Series Plot

Load the General Function Plot program and key in this function:

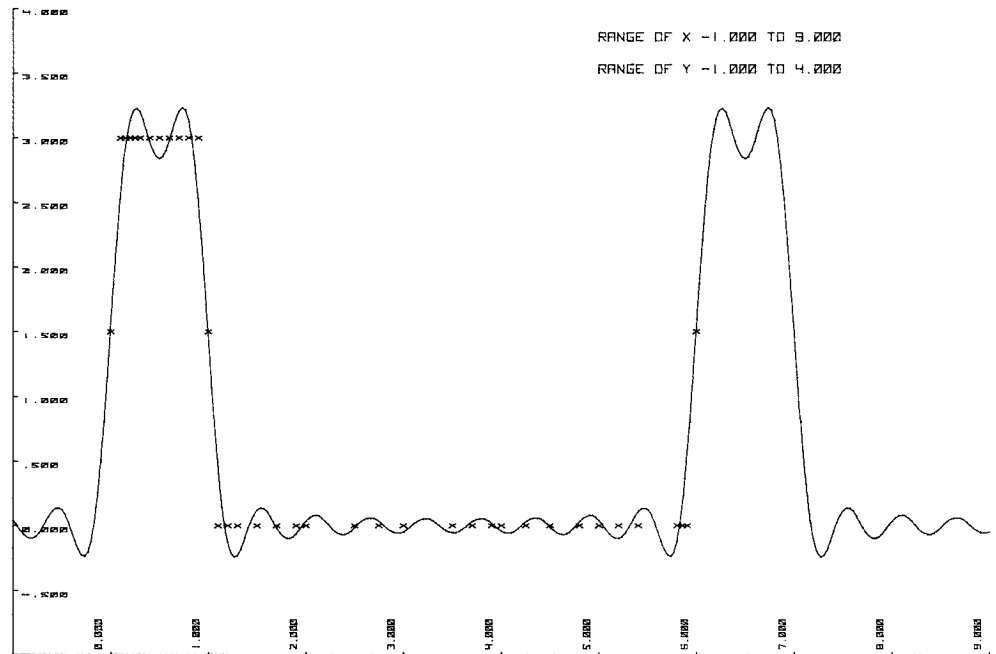
```

25:
"F"↓
26:
IF FLG 6=0:SFG 6
:GSB +5↓
27:
R6+R5:0+R10↓
28:
IF (R5+1+R5)>R7:
R10+R10+Y:RET ↑
29:
R(2R9+29+R5-R6)
COS (2R5πX/R8)+R
10+R10↓
30:
R(2R9+R7+30+R5-2
R6)SIN (2R5πX/R8
)+R10+R10:GTO -2
↑
31:
0+R10↓
32:
IF (R10+1+R10)>R
9:RET ↑
33:
LTR R(28+2R10)-.
0032(R1-R0),R(29
+2R10)-.0032(A-B
),111↓
34:
PLT "X":GTO -2↑

```

Input Data

$x_{\min} = -1$
 $x_{\max} = 9$
 no. of points = 300
 $y_{\min} = -1$
 $y_{\max} = 4$



For the next two plots, load the General Function Plot program. Key in the function and the following changes for proper x axis labeling.

Changes

```

14:
SCL 0,R2,0,10:
AXE 0,0,1,1↑

```

```

18:
LTR .6R2,9,211↑

```

```

20:
FLT :-.1+X:R0+C:
IF ((R1-R0)/R2+2
)<1E2:F&D 0:IF 2
<9:F&D 3:IF 2<1E
-2:FLT ↑
21:
IF (1+X+X)<R2:
LTR X,.1,112:
PLT C+Z+C:GTO +0
↑
22:
LTR .6R2,9.5,211
↑

```

EXAMPLE (Continued)

2. Frequency Spectrum of $f(t)$ Plot

```

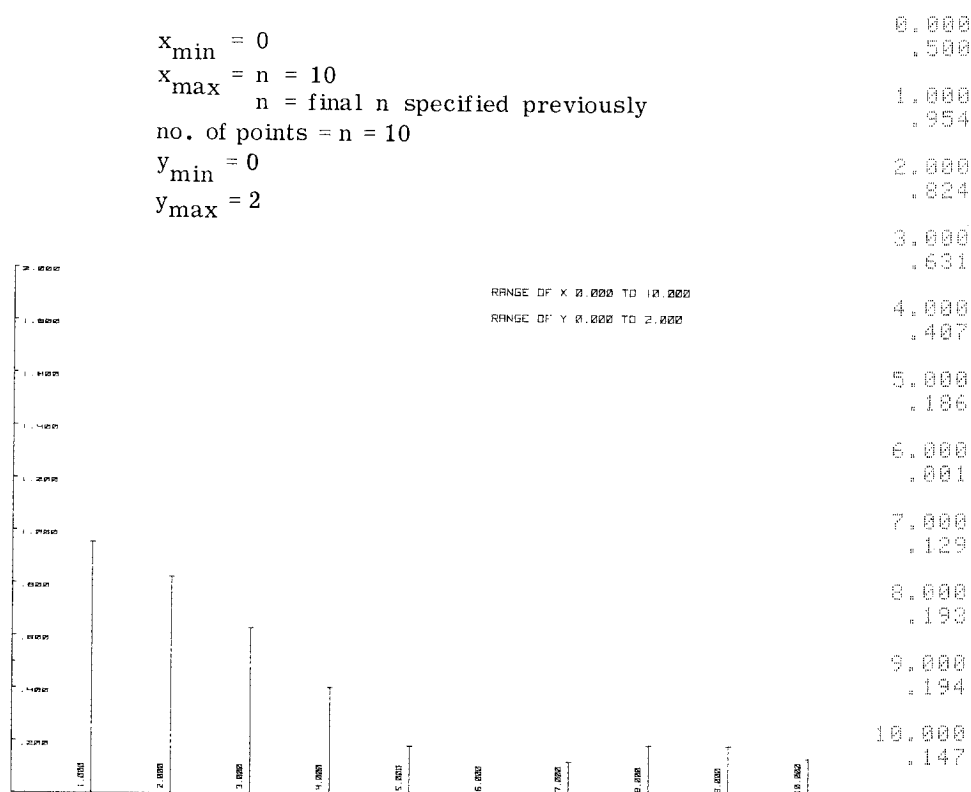
25:
"F"
26:
PEN ;IF FLG 5=0;
0+R10;SFG 5;R10+
Y;PRT " f(AN+2+
BN+2)" ;SPC 1;
JMP 2F
27:
R10+1+R10;f(R(2R
9+R10+29)12+R(2R
9+R10+30+R7-R6)1
2)+Y;
28:
PLT X+.03,Y;PLT
X,Y;PLT X-.03,Y;
PLT X,Y;PRT X,Y;
SPC 1;0+Y;RET F
29:
END F

```

Input Data

$x_{\min} = 0$
 $x_{\max} = n = 10$
 $n = \text{final } n \text{ specified previously}$
 $\text{no. of points} = n = 10$
 $y_{\min} = 0$
 $y_{\max} = 2$

f (AN+2+BN+2)

3. Phase Spectrum of $f(t)$ PlotInput Data

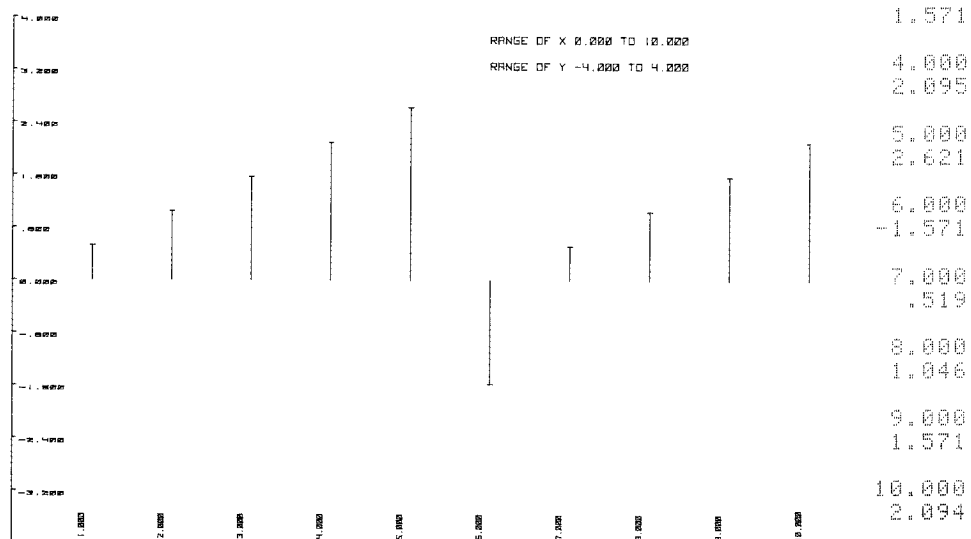
$x_{\min} = 0$
 $x_{\max} = n = 10$
 $\text{no. of points} = n = 10$
 $y_{\min} = -4$
 $y_{\max} = 4$

ARCTAN BN/AN

```

25:
"F"
26:
PEN ;IF FLG 5=0;
0+R10+Y;SFG 5;
SFG 14;PRT " AR
CTAN BN/AN";SPC
1;RET F
27:
R10+1+R10;R(2R9+
R10+30+R7-R6)+R1
1;R(2R9+R10+29)+
R12;
28:
ATN (R11/R12)+((
R11>0)-(0>R11))
(0>R12)+Y;
29:
PLT X+.03,Y;PLT
X,Y;PLT X-.03,Y;
PLT X,Y;PRT X,Y;
SPC 1;0+Y;RET F
30:
END F

```



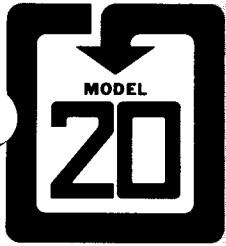
REGISTERS					
A	k	.			
B	I	.			
C	x_1	.			
X	x_2	$29+2k+n$	a_n		
Y	x_3		b_1		
Z	DEN		b_2		
			b_3		
0	0	.			
1	n	.			
2	R	.			
3	y_1	$29+2k+2n$	b_n		
4	y_2				
5	y_3				
6	A				
7	B				
8	C				
9	A'				
10	B'				
11	C'				
12	AP				
13	BP				
14	CP				
15	J				
16	QUAN				
17	QUAN 1				
18	a_0				
19	QQ, QSQ				
20	ARGU				
21	ARGU 1				
22	QCB				
23	SUM				
24	SUM 1				
25	Temporary				FLAGS
26	SN			0	
27	CS			1	Used
28	SN 1			2	
29	CS 1			3	
30	x_1			4	
31	y_1			5	
32	x_2			6	
33	y_2			7	
.				8	
.				9	
.				10	
28+2k	x_k			11	
29+2k	y_k			12	
	a_1			13	Used
	a_2			14	
	a_3			15	

PROGRAM LISTING

```

0:
TBL 2;TBL 4;TBL
5;TBL 6+
1:
ENT "FINAL N?";R
1;PRT "      DAT
A","      X","
      Y";SPC 1
+
2:
ENT "X?";R(2A+30
);IF FLG 13;R(2A
+28)-R30+R2;SPC
2;JMP 2+
3:
PRT ;ENT "Y?";R(
2A+31);PRT ;SPC
1;A+1+R;JMP -1+
4:
B+1+R+
5:
R(2B+28)+C;R(2B+
30)+X;R(2B+32)+Y
+
6:
R(2B+29)+R3;R(2B
+31)+R4;R(2B+33)
+R5+
7:
(X-Y)(C-Y)(C-X)+
Z+
8:
((X-Y)R3-(C-Y)R4
+(C-X)R5)/Z+R9+
9:
-((XX-YY)R3-(CC-
YY)R4+(CC-XX)R5)
/Z+R10+
10:
((X-Y)XYR3-(C-Y)
CYR4+(C-X)CXR5)/
Z+R11+
11:
IF B>1;GTO +2+
12:
R9+R6;R10+R7;R11
+R8;JMP 6+
13:
IF B=A-1;JMP 4+
14:
(R9+R12)/2+R6+
15:
(R10+R13)/2+R7+
16:
(R11+R14)/2+R8;
GTO +2+
17:
R12+R6;R13+R7;R1
4+R8+
18:
0+R15+
19:
2R6X+R7+R16;2R6C
+R7+R17+
20:
R9+R12;R10+R13;R
11+R14+
21:
IF R15+R0#0;GTO
+2+
22:
R6(XXX-CCC)/3+R7
(XX-CC)/2+R8(X-C
)+R18+R18;GTO +1
3+
23:
IF R0>0;SPC 1+
24:
((R15+R0)/R2+R19
)C+R20+
25:
((R15+R0)/R2+R19
)C+R20+
26:
2π(R20-INT R20)+
R20+
27:
2π(XR19-INT (XR1
9))→R21+
28:
((2πR19+R22)R22+
R19)R22+R22+
29:
(R4R19-2R6)/R22+
Z;(R3R19-2R6)/R2
2+R25;SIN R20+R2
6+
30:
R16(COS R21+R29)
/R19+Z(SIN R21+R
28)-R17(COS R20+
R27)/R19-R25R26+
R23+
31:
R16R28/R19-ZR29-
R17R26/R19+R25R2
7+R24+
32:
FLG 1+R15+2A+29+
Z+
33:
R23+RZ+RZ+
34:
R24+R(Z+R1-R0+1)
+R(Z+R1-R0+1)+
35:
IF R15+R0#R1;R15
+1+R15;GTO -14+
36:
IF B#A-1;GTO -32
+
37:
SPC 2;PRT "      COE
FFICIENTS";PRT "
      N","      BN
      AN","      BN
";SPC 1+
38:
PRT 0+Z,R18/(R2+
R8)+R18;SPC 1;(R
1+R7)-(R0+R6)+1+
B;A+R9+
39:
IF (Z+1+Z)>B-1;
SPC 8;JMP 3+
40:
PRT Z+R0,2R(2A+Z
+29+C)/R2+RC,2R(
B+C)/R2+R(B+C)+
41:
SPC 1;JMP -2+
42:
END +

```

PARTIAL SUM OF AN INFINITE SERIES/
PARTIAL PRODUCT OF AN INFINITE PRODUCT

This program will evaluate sums or products of the form

$$\sum_{n=k}^N f(n, x) \quad \text{or} \quad \prod_{n=k}^N f(n, x)$$

and, in particular, sums or products of the form

$$\sum_{n=k}^N g(n) \quad \text{or} \quad \prod_{n=k}^N g(n)$$

REMARKS

1. After loading the program, the expression $f(n, x) \rightarrow Y$ (or $g(n) \rightarrow Y$) is keyed in and stored beginning in program line 30 and continuing for as many lines as required to define the function. When keying in the expression, substitute C for n and X for x, the storage locations of n and x respectively. Register R1 contains $(-1)^n$ for $n=k, \dots, N$ so that in the expression for $f(n, x)$ or $g(n)$,

$(-1)^n$ may be replaced by R1

$(-1)^{n \pm 1}$ may be replaced by -R1

$(-1)^{2n}$ may be replaced by R1R1

$(-1)^{2n \pm 1}$ may be replaced by -R1R1

If storage registers or flags are needed, refer to the MEMORY ALLOCATION page for availability. Flags 2, 3, and 4 are available to the user and are cleared by the program each time a new sum or product is formed starting at the lower limit.

2. An option is provided in the User Instructions to enter an epsilon. If entered, summation of a series will continue until $|f(n, x)| \leq \epsilon$ or $n = N$, the upper limit, whichever occurs first. When a product is being formed, the test is for $|f(n, x) - 1| \leq \epsilon$ and $n = N$. Similarly for $g(n)$. The test can essentially be made to depend only upon ϵ by specifying a very large N. The test can be made to depend only upon N by not specifying an epsilon, in which case calculations will continue until $n = N$.

3. When "UPPER LIMIT?" is displayed, the user may set flag 1 prior to keying in N, RUN PROGRAM for displaying the sum or product as it is being formed, giving a visual check for convergence or divergence.

4. After an upper limit has been selected, calculations proceed until the end results are printed. Calculations may be interrupted by pressing the SFG* key, causing the value of x (if there is one) and the current value of n to be printed together with the sum or product to that point. Calculations then resume and this process may be repeated any number of times independent of whether or not intermediate sums or products are being displayed.

5. After the end results are printed and if no ϵ has been specified, "UPPER LIMIT?" is again displayed and a new upper limit N' may be specified. If $N < N'$, calculations will continue from $n = N + 1$ saving recalculation time up to $n = N$.

* SFG refers to the SET/CLEAR FLAG N key.

ROM BLOCKS

1

2

3

INTERNAL REGISTERS

☒ 173 ☐ 429

PERIPHERALS

☐ 9860A MARKED CARD READER☐ 9862A PLOTTER☐ 9864A DIGITIZER☐☐ 9861A TYPEWRITER☐ 9863A PAPER TAPE READER☐☐

STEP	DISPLAY	INSTRUCTIONS
1.		<u>ERASE</u> <u>LOAD</u> <u>EXECUTE</u>
2.		Alternately insert magnetic cards and <u>EXECUTE</u> until NOTE 14 is no longer displayed, indicating completion of loading.
3.		<u>GO TO</u> <u>3 0</u> <u>EXECUTE</u> and key in and store the function being considered: <u>f(n, x) → Y</u> (or <u>g(n) → Y</u>) <u>STORE</u> <u>RETURN</u> <u>STORE</u> <u>END</u> <u>STORE</u> using register X to represent x and register C to represent n, as needed.
4.		<u>END</u> <u>RUN PROGRAM</u>
5.	SUM? PRODUCT?	<u>RUN PROGRAM</u> to cycle between these two displays; <u>1</u> <u>RUN PROGRAM</u> to select the option being displayed.
6.	EPSILON?	<u>ε</u> <u>RUN PROGRAM</u> or <u>RUN PROGRAM</u> if there is no <u>ε</u> .
7.	VALUE OF X?	<u>x</u> <u>RUN PROGRAM</u> or <u>RUN PROGRAM</u> if there is no x (g(n) case).
8.	LOWER LIMIT?	<u>k</u> <u>RUN PROGRAM</u>
9.	UPPER LIMIT?	* <u>SFG</u> <u>1</u> <u>EXECUTE</u> (optional) for display of intermediate results and then <u>N</u> <u>RUN PROGRAM</u> or <u>RUN PROGRAM</u> to return to Step 6 for a new case.
10.		* <u>SFG</u> while the program is running to print x (if there is one), the current value of n, and the sum or product up to that n. Calculations continue automatically. After final answers are printed, control returns to Step 9 unless an epsilon has been specified in which case control returns to Step 6.
* SFG refers to the SET/CLEAR FLAG N key.		

EXAMPLES

1. Evaluate the sum $\sum_{n=1}^{10} x^n$ for $x = .5$ and no epsilon. If the ^{mathematics} Trigonometric block is in the calculator, $X \uparrow C \rightarrow Y$ may be keyed in for the function. If not, choose a register not used by the program, R3 for instance. Manually store the value 1 in R3 after loading the program. Then x^n may be formed as follows:

Program Lines	Results
29: "F"␣	X N
30: R3X→R3+Y␣	.500000
31: RET ␣	10.000000
32: END ␣	SUM .99902

2. Evaluate the sum $\sum_{n=1}^N \frac{(-1)^n}{n}$ and terminate when $\left| \frac{(-1)^n}{n} \right| \leq 10^{-2}$. When the display requests the upper limit, use $N = 10^9$, for example.

Program Lines	Results
29: "F"␣	N
30: R1/C→Y␣	100.000000
31: RET ␣	SUM
32: END ␣	-.68817

3. Evaluate the product $10! = \prod_{n=1}^{10} n$

Program Lines	Results
29: "F"␣	N
30: C→Y␣	10.0000
31: RET ␣	PRODUCT
32: END ␣	3628800.000

[illegible]

PROGRAM LISTING

```

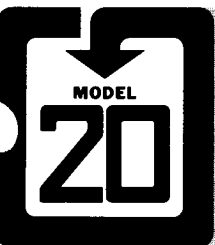
0:
CFG 13;CFG 5;
ENT "SUM?";Z;IF
FLG 13=0;SFG 5;
GTO 2F
1:
CFG 13;ENT "PROD
UCT?";Z;IF FLG 1
3;GTO 0F
2:
-1+R0F
3:
ENT "EPSILON?";R
0F
4:
CFG 13;CFG 6;
ENT "VALUE OF X?
";X;IF FLG 13=0;
SFG 6F
5:
ENT "LOWER LIMIT
?";R2;GSB 24F
6:
CFG 13;CFG 0;
CFG 1;ENT "UPPER
LIMIT?";A;IF
FLG 13;GTO 2F
7:
IF C>A;GSB 24F
8:
GSB "F"F
9:
IF FLG 5;B+Y+B;
GTO +2F
10:
BY+BF
11:
IF FLG 1;DSP BF
12:
IF FLG 0;CFG 0;
GSB 17F
13:
-R1+R1;(FLG 5=0)
-Y+Z;IF 0>Z;-Z+Z
F
14:
GTO +2;IF (C=A)+
(Z<R0);GSB 17F
15:
C+1+C;GTO 6;IF 0
<R0;GTO 2F
16:
C+1+C;GTO 8F

```

```

17:
IF FLG 6;PRT "
X"F
18:
PRT "          N"F
19:
IF FLG 6;PRT XF
20:
PRT C;SPC 1F
21:
IF FLG 5;PRT "
SUM";JMP 2F
22:
PRT "          PRODUC
T"F
23:
PRT B;SPC 1;PRT
"-----
-";SPC 1;RET F
24:
CFG 2;CFG 3;CFG
4F
25:
(1+R1)-FLG 5+B;R
2+C+Z;IF 0>Z;-Z+
ZF
26:
IF (Z=0)+(Z=2);
RET F
27:
IF Z/2#Z/2-.5+1E
11-1E11;-1+R1F
28:
RET F
29:
"F"F
30:
END F

```

ROOTS OF $F(X) = 0$ IN AN INTERVAL

This program uses the principle of interval-halving to find real roots of an equation $f(x) = 0$ in a closed interval $[a, b]$ where the equation may be algebraic (e.g., $5x^4 - 3x + 1 = 0$), rational (e.g., $x^{3/2} + \sqrt{x} - 2 = 0$), or transcendental (e.g., $3 \cos x - 4x = 0$).

The user specifies the continuous, real function f , an interval $[a, b]$, an accuracy tolerance ϵ , and a search increment Δx ($=\Delta x$). The program then begins at the left of the interval and compares the functional values at the ends of the interval $[a, a + \Delta x]$. If $f(a)$ and $f(a + \Delta x)$ are of opposite sign, this interval will be searched for a root. Otherwise, or even after a root is found, the program proceeds in the same manner with the interval $[a + \Delta x, a + 2\Delta x]$, etc. At most one root will be found by the program for each of these small intervals and the program will examine $N = \text{INT } \frac{(b-a)}{\Delta x}$ of these intervals.

Reference:

Royce Beckett and James Hurt, Numerical Calculations and Algorithms (New York: McGraw-Hill Book Company, 1967), pp. 46-52.

ROM BLOCKS

1

2

3

INTERNAL REGISTERS

☒ 173 ☐ 429

PERIPHERALS

☐ 9860A MARKED CARD READER☐ 9862A PLOTTER☐ 9864A DIGITIZER☐☐ 9861A TYPEWRITER☐ 9863A PAPER TAPE READER☐☐

STEP	DISPLAY	INSTRUCTIONS
1.		<u>ERASE</u> <u>LOAD</u> <u>EXECUTE</u>
2.		Alternately insert magnetic cards and <u>EXECUTE</u> until NOTE 14 no longer appears, indicating completion of loading.
3.		(Optional) <u>FIXED N n EXECUTE</u> , or <u>FLOAT N n EXECUTE</u> as desired.
4.		<u>GO TO 2 1 EXECUTE</u> and key in and store the function being considered using register X to represent the variable x and storing the resulting function value in Y: <u>f(x) → Y STORE RETURN STORE END STORE</u> .
5.		<u>END RUN PROGRAM</u>
6. START AT X = ?	<u>a</u>	<u>RUN PROGRAM</u>
7. END AT X = ?	<u>b</u>	<u>RUN PROGRAM</u>
8. DELTA X = ?	<u>Δx</u>	<u>RUN PROGRAM</u>
9. EPSILON = ?	<u>ε</u>	<u>RUN PROGRAM</u> or <u>RUN PROGRAM</u> alone and $\epsilon = 10^{-10}$ will be assumed.
<p>The search begins automatically and roots are printed as they are found together with a reevaluation of the function at the roots. Control returns to Step 6 for designation of a new interval and the same function. For a new function, <u>STOP</u> and go to Step 4.</p>		

EXAMPLES

1. Find the roots of $x^3 - 8x^2 + 5x + 14 = 0$ in the interval $[-10, 10]$ using $\Delta x = 1$ and $\epsilon = 10^{-6}$.

Key in these
program lines:

Results:

```
20:
"F"␣
21:
XXX-8XX+5X+14+Y␣
22:
RET ␣
23:
END ␣
```

```
ROOTS OF F(X)
ON THE INTERVAL
-10.00000000
10.00000000

ARE
X
F(X)
-1.00000000
0.00000000

X
F(X)
2.00000000
0.00000000

X
F(X)
7.00000000
0.00000000

END OF SEARCH
```

2. Find the roots of $x^{5/2} - 2\sqrt{x} = 0$ in the interval $[0, 20]$ using $\Delta x = 1$ and $\epsilon = 10^{-6}$.

Key in these
program lines:

Results:

```
20:
"F"␣
21:
F(XXXXX)-2FX+Y␣
22:
RET ␣
23:
END ␣
```

```
ROOTS OF F(X)
ON THE INTERVAL
0.00000000
20.00000000

ARE
X
F(X)
0.00000000
0.00000000

X
F(X)
1.41421366
.00000032

END OF SEARCH
```

REGISTERS

[illegible]

PROGRAM LISTING

```

01:
1E-10+R0;ENT "ST
ART AT X= ?";A+
1:
ENT "END AT X= ?
";R2;"DELTA X= ?
";R1;"EPSILON= ?
";R0+
2:
PRT "  ROOTS OF
F(X)", " ON THE I
NTERVAL";SPC 1;
PRT A,R2;SPC 2;
PRT "      ARE"
;SPC 1+
3:
A+X;GSB "F"+
4:
Y+C+
5:
A+R1+B+
6:
IF C=0;GTO 17+
7:
IF B>R2;GTO 19+
8:
B+X+
9:
GSB "F"+
10:
IF 0<(Y+R3)C;R3+
C;B+A;GTO 5+
11:
B+R4+
12:
(A+R4)/2+X+
13:
GSB "F"+
14:
IF (Y<R0)(Y>-R0)
;PRT "      X"
+ "      F(X)";X
+Y;SPC 1;R3+C;B+
A;GTO 5+
15:
IF 0>CY;X+R4;
GTO -3+
16:
X+A;Y+C;GTO -4+
17:
PRT "      X",
"      F(X)";X;
Y;SPC 1;B+A+
18:
IF R2>B;GTO 3+
19:
SPC 1;PRT "  END
OF SEARCH";SPC
8;GTO 0+
20:
"F"+
21:
END +

```




BESSEL FUNCTIONS I

Most methods to compute Bessel functions such as $J_n(x)$ and $Y_n(x)$ have certain restrictions. For instance, the power series representation for $J_n(x)$ converges slowly and loses accuracy due to roundoff for values of x greater than 10. Asymptotic expansions of $J_n(x)$ and $Y_n(x)$ are applicable when x is large relative to n . Neither of these two standard techniques are useful when both n and x are large. A method developed by J. C. P. Miller¹ overcomes these restrictions and provides a simple algorithm to calculate $J_n(x)$, $Y_n(x)$, and many other Bessel functions over the entire range $x > 0$, $n \geq 0$, n an integer. This program² uses the technique developed by Miller to compute any of the following Bessel functions of integer order:

Function	Name	Symbol Used in Program
$J_n(x)$	Bessel function of the first kind	JN(X)
$Y_n(x)$	Bessel function of the second kind	YN(X)
$I_n(x)$	Modified Bessel function of the first kind	IN(X)
$j_n(x)$	Spherical Bessel function of the first kind	JSN(X)
$y_n(x)$	Spherical Bessel function of the second kind	YSN(X)
$i_n(x)$	Modified Spherical Bessel function of the first kind	ISN(X)
$k_n(x)$	Modified Spherical Bessel function of the third kind	KSN(X)

As an example of the method, the modified Bessel function of the first kind, $I_n(x)$, is obtained from the descending recurrence relation

$$I_{p-1} = (2p/x) I_p + I_{p+1}$$

with normalization by

$$I_0 + 2 \sum_{m=1}^{\infty} I_m = e^x$$

¹ Generation of Bessel Functions on High Speed Computers, Irene Stegun and Milton Abramowitz, Mathematical Tables and Other Aids to Computation, Volume 11, 1957, pp. 255-257.

² Based on a very complete, accurate, and efficiently programmed set of 9100 Bessel function programs submitted to Hewlett-Packard by J. Robert Cooke, Associate Professor, New York State College of Agriculture, Cornell University.

ROM BLOCKS

☐

1

☒ *Mathematics*
Trigonometric

2

☐

3

INTERNAL REGISTERS

☒ 173 ☐ 429

PERIPHERALS

☐ 9860A MARKED CARD READER☐ 9862A PLOTTER☐ 9864A DIGITIZER☐☐ 9861A TYPEWRITER☐ 9863A PAPER TAPE READER☐☐

STEP	DISPLAY	INSTRUCTIONS
1.		<u>ERASE</u> <u>LOAD</u> <u>EXECUTE</u>
2.		Alternately insert magnetic cards and <u>EXECUTE</u> until NOTE 14 no longer appears, indicating completion of loading.
3.		(Optional) <u>FIXED N</u> <u>n</u> <u>EXECUTE</u> , or <u>FLOAT N</u> <u>n</u> <u>EXECUTE</u> as desired.
4.		<u>END</u> <u>RUN PROGRAM</u>
5.	N?	<u>n</u> <u>RUN PROGRAM</u> n = order
6.	X?	<u>x</u> <u>RUN PROGRAM</u> x = argument
7.	JN(X)?	<u>RUN PROGRAM</u> to cycle between these 7 displays;
	YN(X)?	<u>1</u> <u>RUN PROGRAM</u> to select the function displayed.
	IN(X)?	
	JSN(X)?	
	YSN(X)?	
	ISN(X)?	
	KSN(X)?	
<p>After the selected function is evaluated and the results printed, control returns to Step 5 for a new case.</p>		

EXAMPLES

Verify the following values selected from Bessel function tables:

	<u>Symbol in Program</u>
$J_4(2) = 3.399571981 \text{ E} - 2$	JN(X)
$I_5(2) = 9.825679323 \text{ E} - 3$	IN(X)
$j_3(.1) = 9.5185 \text{ E} - 6$	JSN(X)
$i_1(3.7) = 3.99283865$	ISN(X)
$k_2(1.6) = .80213693$	KSN(X)
$y_2(4.5) = .10491554$	YSN(X)
$Y_0(10) = 5.567116730 \text{ E} - 2$	YN(X)
$Y_2(100) = 7.683686678 \text{ E} - 2$	YN(X)

In the last example above, x is large relative to n and for such cases, Program IV-2, Bessel Function II, using an asymptotic expansion will find the answer much faster. The time required to find a result using this program should not exceed about 75 seconds. If no result is obtained in that time, the program is unable to converge, as in the case of $i_{50}(1)$.

```

      N
      4.0000000000E 00
      X
      2.0000000000E 00
      JN(X)
      3.399571981E-02

```

```

      N
      5.0000000000E 00
      X
      2.0000000000E 00
      IN(X)
      9.825679323E-03

```

```

      N
      3.0000000000E 00
      X
      1.0000000000E-01
      JSN(X)
      9.518519721E-06

```

```

      N
      1.0000000000E 00
      X
      3.7000000000E 00
      ISN(X)
      3.992838654E 00

```

```

      N
      2.0000000000E 00
      X
      1.6000000000E 00
      KSN(X)
      8.021369335E-01

```

```

      N
      2.0000000000E 00
      X
      4.5000000000E 00
      YSN(X)
      1.049155451E-01

```

```

      N
      0.0000000000E 00
      X
      1.0000000000E 01
      YN(X)
      5.567116728E-02

```

```

      N
      2.0000000000E 00
      X
      1.0000000000E 02
      YN(X)
      7.683686712E-02

```

IV-1

MEMORY ALLOCATION

REGISTERS

[illegible]

PROGRAM LISTING

```

0:
TBL 2:TBL 6:SFG
14:ENT "N?";C+R4
,"X?";XF
1:
PRT "      N",C,"
      X";XF
2:
CFG 13:ENT "JN(X
)?",Z;IF FLG 13=
0:SFG 1;JMP 8F
3:
CFG 13:ENT "YN(X
)?",Z;IF FLG 13=
0:SFG 2;1+C;SFG
1;JMP 7F
4:
CFG 13:ENT "IN(X
)?",Z;IF FLG 13=
0:SFG 3;JMP 6F
5:
CFG 13:ENT "JSN(
X)?",Z;IF FLG 13
=0:SFG 4;JMP 5F
6:
CFG 13:ENT "YSN(
X)?",Z;IF FLG 13
=0:SFG 5;(-COS X
/X+A)TAN X+A/X+B
;GTO 37F
7:
CFG 13:ENT "ISN(
X)?",Z;IF FLG 13
=0;JMP 3F
8:
CFG 13:ENT "KSN(
X)?",Z;IF FLG 13
;JMP -6F
9:
SFG 6;-π/2XEXP X
+A;-A(X+1)/X+B;
GTO 37F
10:
30+Y;0+ZF
11:
Z+Y+YF
12:
2INT ((C+X)/2)+Y
+BF
13:
0+A+R0+R1+R2+R5F
14:
1+Z+R3;COS ((B-2
)π/2)+R6F

```

```

15:
IF B=C;R3+R1;IF
1E13>(Z-2(0>Z)Z+
Z);10+10(C<1)
FLG 1+Z;GTO 11F
16:
IF B=0;GTO 21F
17:
IF (R0=0+R0)+
FLG 3;R3+A+A;IF
FLG 2;R3R6/B+R5+
R5;-R6+R6F
18:
IF (FLG 1=0)(
FLG 3=0);.5+B+BF
19:
2B(R3+(2(FLG 1=0
)(FLG 4=0)-1)R2X
/2B)/X+Z;R3+R2;Z
+R3F
20:
B-1-.5(FLG 1=0)(
FLG 3=0)+B;GTO 1
5F
21:
IF (FLG 1=0)(
FLG 3=0);GTO 25F
22:
2A+R3+ZF
23:
IF FLG 3;Z/EXP X
+ZF
24:
GTO 28F
25:
IF FLG 4=0;EXP X
+Z;(Z-1/Z)/2+Z;
GTO 27F
26:
SIN X+ZF
27:
R3X/Z+ZF
28:
R1/Z+A;R3/Z+B;R5
/Z+R5;IF 0>R2;-R
2+R2F
29:
IF 0>Z;-Z+ZF
30:
IF (Z>1E98)+(R2>
1E98);-10-10(C<1
)FLG 1+Z;GTO 11F
31:
IF FLG 2;GTO 45F

```

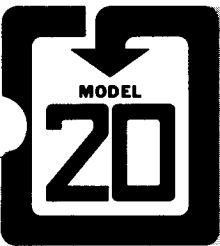
```

32:
IF FLG 1;PRT "JN
(X)"F
33:
IF FLG 3;PRT "IN
(X)"F
34:
IF FLG 4;PRT "JS
N(X)"F
35:
IF (FLG 1=0)(
FLG 3=0)(FLG 4=0
);PRT "ISN(X)"F
36:
GTO 53F
37:
0+YF
38:
Y+1+YF
39:
IF Y>C;0+Z;GTO +
3F
40:
IF Y=C;B+A;π+Z;
GTO +2F
41:
(2Y+1)B/X-A+Z;B+
A;Z-2FLG 6Z+B;
GTO 38F
42:
IF FLG 6;COS (πY
+Z)A+BF
43:
IF FLG 5;PRT "YS
N(X)";GTO 53F
44:
PRT "KSN(X)";
GTO 53F
45:
X/2+XF
46:
2((LN X+.5772156
649)B+4R5)/π+ZF
47:
(AZ-1/πX)/B+A;Z+
BF
48:
IF R4=0;B+BF
49:
IF R4<1;GTO 52F
50:
1+YF

```

PROGRAM LISTING

```
E1:
AY/X-B+Z:A+B:Z+A
:JMP (Y+1+Y)=R4F
E2:
ART "YN(X)"F
E3:
ART A:SPC 1:GTO
E4:
END F
```



BESSEL FUNCTIONS II

This program¹ uses the method of asymptotic expansion² to find, for a given x and real number r with $x \gg r \geq 0$ (x very much larger than r), any one of the following Bessel functions of fractional order:

Function	Name	Symbol Used in Program
$J_r(x)$	Bessel function of the first kind	JR (X)
$Y_r(x)$	Bessel function of the second kind	YR (X)
$I_r(x)$	Modified Bessel function of the first kind	IR (X)
$K_r(x)$	Modified Bessel function of the second kind	KR (X)

An accuracy check is built into the program so that if the result obtained is not accurate to within one unit in the tenth significant figure, as is usually the case for small arguments, an error bound, ϵ , will also be printed. The actual answer will be between the answer A that is printed and $A - \epsilon$. For small values of the argument, say $x \leq 10$, the asymptotic expansion is truncated when the terms are not monotonically decreasing.

¹ Based on a select set of 9100 Bessel function programs submitted to Hewlett-Packard by J. Robert Cooke, Associate Professor, New York State College of Agriculture, Cornell University.

² Handbook of Mathematical Functions, Milton Abramowitz and Irene Stegun, National Bureau of Standards, 7th printing, 1968, Chapter 9.

ROM BLOCKS

1

☒ ~~Mathematics~~
Trigonometric

2

3

INTERNAL REGISTERS

☒ 173 ☐ 429

PERIPHERALS

☐ 9860A MARKED CARD READER☐ 9862A PLOTTER☐ 9864A DIGITIZER☐☐ 9861A TYPEWRITER☐ 9863A PAPER TAPE READER☐☐

STEP	DISPLAY	INSTRUCTIONS
1.		<u>ERASE</u> <u>LOAD</u> <u>EXECUTE</u>
2.		Alternately insert magnetic cards and <u>EXECUTE</u> until NOTE 14 no longer appears, indicating completion of loading.
3.		(Optional) <u>FIXED N n EXECUTE</u> , or <u>FLOAT N n EXECUTE</u> as desired.
4.		<u>END</u> <u>RUN PROGRAM</u>
5. R?	<u>r</u>	<u>RUN PROGRAM</u> r = order
6. X?	<u>x</u>	<u>RUN PROGRAM</u> x = argument
7. JR(X)?		<u>RUN PROGRAM</u> to cycle between the 4 displays;
YR(X)?	<u>1</u>	<u>RUN PROGRAM</u> to select the function displayed.
IR(X)?		
KR(X)?		
After the selected function is evaluated and the result printed, control returns to Step 5 for a new case.		

EXAMPLES

Verify the following values selected from Bessel function tables:

$$J_{2.5}(4) = .4409$$

$$Y_0(10) = .0556711673$$

$$Y_2(100) = .076836867$$

$$I_{\frac{1}{3}}(9.8) = 2315.5$$

$$I_0(225) = 1.384583 \text{ E}96$$

$$K_1(13) = 8.0786 \text{ E} - 7$$

```

R
2.500000000E 00
X
4.000000000E 00
JR(X)
4.408849746E-01

```

```

R
0.000000000E 00
X
1.000000000E 01
ERROR IS
-3.649010818E-10
YR(X)
5.567116726E-02

```

```

R
2.000000000E 00
X
1.000000000E 02
YR(X)
7.683686714E-02

```

```

R
3.333333333E-01
X
9.000000000E 00
ERROR IS
-2.803464640E-10
IR(X)
2.315459289E 03

```

```

R
0.000000000E 00
X
2.250000000E 02
IR(X)
1.384583169E 96

```

```

R
1.000000000E 00
X
1.300000000E 01
KR(X)
8.078588413E-07

```

IV-2

MEMORY ALLOCATION

REGISTERS

A	Q_k or P_k
B	Counter
C	$2r$
X	x
Y	$\Sigma Q, f(x)$
Z	Temporary
0	0 or 1 Indicator
1	ΣP
2	Q or P, Error
3	
4	
5	
6	
7	
8	
9	
10	
11	
12	
13	
14	
15	

0	
1	$J_r(x)$ Indicator
2	$Y_r(x)$ Indicator
3	$I_r(x)$ Indicator
4	
5	
6	
7	
8	
9	
10	
11	
12	
13	Used
14	Used
15	

PROGRAM LISTING

```

0:
TBL 2:TBL 6:SFG
14:ENT "R?",Z*2+
C,"X?",X+
1:
PRT "      R",Z,"
      X",X+
2:
CFG 13:ENT "JR(X
)?",Z;IF FLG 13=
0:SFG 1:GTO 6+
3:
CFG 13:ENT "YR(X
)?",Z;IF FLG 13=
0:SFG 2:GTO 6+
4:
CFG 13:ENT "IR(X
)?",Z;IF FLG 13=
0:SFG 3:GTO 6+
5:
CFG 13:ENT "KR(X
)?",Z;IF FLG 13:
GTO 2+
6:
0+Y+R0;1+A+B+Z+R
1+
7:
(Z/B+Z)*(C-(2B-
1)(2B-1))/8X+Z+R
2+
8:
IF 0>Z:-Z+Z+
9:
IF Z>A;Z+A;PRT "
ERROR IS",R2;
GTO 16+
10:
Z+A+
11:
COS ((FLG 1+FLG
2)πINT (B/2))R2+
Z+
12:
IF R0=0+R0;Z+Y+Y
;GTO +2+
13:
Z+R1+R1+
14:
B+1+B+
15:
IF A>1E-10;R2+Z;
GTO 7+
16:
r(2/πX)+Z+
17:
X-(C+1)π/4+C+
18:
ZSIN C+R0+
19:
ZCOS C+C+
20:
CR1-YR0+A+
21:
YC+R1R0+B+
22:
EXP X+C;r(π/2X)+
Z+
23:
C(R1-Y)/r(2πX)+X
+
24:
Z(R1+Y)/C+Y+
25:
IF FLG 1:PRT "JR
(X)",A;GTO 29+
26:
IF FLG 2:PRT "YR
(X)",B;GTO 29+
27:
IF FLG 3:PRT "IR
(X)",X;GTO 29+
28:
PRT "KR(X)",Y+
29:
SPC 1;GTO 0+
30:
END +

```


Program Submittal Form

Hewlett-Packard Company
Calculator Products Division

1. Initial Submission ☐ Revision ☐
2. Equipment required: _____
3. Program Title: _____
4. Program Description and Application: _____

5. Contributor's Name: _____
Organization: _____ Title: _____
Address: _____

Telephone: _____
6. Do you want your name to appear in the catalog?..... yes ☐ no ☐
7. Do you want your organization's name to appear in the catalog?... yes ☐ no ☐
8. May an HP customer contact you directly?.....yes ☐ no ☐
9. Check (✓) to be sure each item below is included.
☐ Program introduction, including equations solved and text reference.
☐ User instructions.
☐ Numeric input/output example, including sample plot or printer tape if applicable.
☐ Program listing (including program steps and step codes if applicable).
☐ Recorded magnetic cards.
Shall we return original recorded cards ☐ or blank cards ☐ ?
10. Acknowledgement and Agreement

To the best of my knowledge, this contributed program is free of any proprietary information belonging to any person or organization. I am making this program available to Hewlett-Packard, and I agree that HP may reproduce, publish, and use it, and authorize others to do so without obligation or liability of any kind.

(Signature)

(Date)

Program Submittal Form

Hewlett-Packard Company
Calculator Products Division

1. Initial Submission ☐ Revision ☐

2. Equipment required: _____

3. Program Title: _____

4. Program Description and Application: _____

5. Contributor's Name: _____

Organization: _____ Title: _____

Address: _____

Telephone: _____

6. Do you want your name to appear in the catalog?..... yes ☐ no ☐

7. Do you want your organization's name to appear in the catalog?.. yes ☐ no ☐

8. May an HP customer contact you directly?.....yes ☐ no ☐

9. Check (✓) to be sure each item below is included.

- ☐ Program introduction, including equations solved and text reference.
- ☐ User instructions.
- ☐ Numeric input/output example, including sample plot or printer tape if applicable.
- ☐ Program listing (including program steps and step codes if applicable).
- ☐ Recorded magnetic cards.

Shall we return original recorded cards ☐ or blank cards ☐ ?

10. Acknowledgement and Agreement

To the best of my knowledge, this contributed program is free of any proprietary information belonging to any person or organization. I am making this program available to Hewlett-Packard, and I agree that HP may reproduce, publish, and use it, and authorize others to do so without obligation or liability of any kind.

(Signature)

(Date)

Program Submittal Form

Hewlett-Packard Company
Calculator Products Division

1. Initial Submission ☐ Revision ☐
2. Equipment required: _____
3. Program Title: _____
4. Program Description and Application: _____

5. Contributor's Name: _____
Organization: _____ Title: _____
Address: _____

Telephone: _____
6. Do you want your name to appear in the catalog?..... yes ☐ no ☐
7. Do you want your organization's name to appear in the catalog?... yes ☐ no ☐
8. May an HP customer contact you directly?.....yes ☐ no ☐
9. Check (✓) to be sure each item below is included.
☐ Program introduction, including equations solved and text reference.
☐ User instructions.
☐ Numeric input/output example, including sample plot or printer tape if applicable.
☐ Program listing (including program steps and step codes if applicable).
☐ Recorded magnetic cards.
Shall we return original recorded cards ☐ or blank cards ☐ ?
10. Acknowledgement and Agreement

To the best of my knowledge, this contributed program is free of any proprietary information belonging to any person or organization. I am making this program available to Hewlett-Packard, and I agree that HP may reproduce, publish, and use it, and authorize others to do so without obligation or liability of any kind.

(Signature)

(Date)

Program Submittal Form

Hewlett-Packard Company
Calculator Products Division

1. Initial Submission ☐ Revision ☐
2. Equipment required: _____
3. Program Title: _____
4. Program Description and Application: _____

5. Contributor's Name: _____
Organization: _____ Title: _____
Address: _____

Telephone: _____
6. Do you want your name to appear in the catalog?..... yes ☐ no ☐
7. Do you want your organization's name to appear in the catalog?... yes ☐ no ☐
8. May an HP customer contact you directly?.....yes ☐ no ☐
9. Check (✓) to be sure each item below is included.
☐ Program introduction, including equations solved and text reference.
☐ User instructions.
☐ Numeric input/output example, including sample plot or printer tape if applicable.
☐ Program listing (including program steps and step codes if applicable).
☐ Recorded magnetic cards.
Shall we return original recorded cards ☐ or blank cards ☐ ?
10. Acknowledgement and Agreement

To the best of my knowledge, this contributed program is free of any proprietary information belonging to any person or organization. I am making this program available to Hewlett-Packard, and I agree that HP may reproduce, publish, and use it, and authorize others to do so without obligation or liability of any kind.

(Signature)

(Date)

Program Submittal Form

Hewlett-Packard Company
Calculator Products Division

1. Initial Submission ☐ Revision ☐
2. Equipment required: _____
3. Program Title: _____
4. Program Description and Application: _____

5. Contributor's Name: _____
Organization: _____ Title: _____
Address: _____

Telephone: _____
6. Do you want your name to appear in the catalog?..... yes ☐ no ☐
7. Do you want your organization's name to appear in the catalog?...yes ☐ no ☐
8. May an HP customer contact you directly?.....yes ☐ no ☐
9. Check (✓) to be sure each item below is included.
 - ☐ Program introduction, including equations solved and text reference.
 - ☐ User instructions.
 - ☐ Numeric input/output example, including sample plot or printer tape if applicable.
 - ☐ Program listing (including program steps and step codes if applicable).
 - ☐ Recorded magnetic cards.Shall we return original recorded cards ☐ or blank cards ☐ ?
10. Acknowledgement and Agreement

To the best of my knowledge, this contributed program is free of any proprietary information belonging to any person or organization. I am making this program available to Hewlett-Packard, and I agree that HP may reproduce, publish, and use it, and authorize others to do so without obligation or liability of any kind.

(Signature)

(Date)

ROM BLOCKS

1

2

3

INTERNAL REGISTERS

173

429

PERIPHERALS

9860A MARKED CARD READER

9862A PLOTTER

9864A DIGITIZER

9861A TYPEWRITER

9863A PAPER TAPE READER

STEP	DISPLAY	INSTRUCTIONS

ROM BLOCKS

1

2

3

INTERNAL REGISTERS

173

429

PERIPHERALS

9860A MARKED CARD READER

9862A PLOTTER

9864A DIGITIZER

9861A TYPEWRITER

9863A PAPER TAPE READER

STEP	DISPLAY	INSTRUCTIONS

ROM BLOCKS

1

2

3

INTERNAL REGISTERS

173

429

PERIPHERALS

9860A MARKED CARD READER

9862A PLOTTER

9864A DIGITIZER

9861A TYPEWRITER

9863A PAPER TAPE READER

STEP	DISPLAY	INSTRUCTIONS

ROM BLOCKS

1

2

3

INTERNAL REGISTERS

173

429

PERIPHERALS

9860A MARKED CARD READER

9862A PLOTTER

9864A DIGITIZER

9861A TYPEWRITER

9863A PAPER TAPE READER

STEP	DISPLAY	INSTRUCTIONS

ROM BLOCKS

1

2

3

INTERNAL REGISTERS

173429

PERIPHERALS

9860A MARKED CARD READER

9862A PLOTTER

9864A DIGITIZER

9861A TYPEWRITER

9863A PAPER TAPE READER

STEP	DISPLAY	INSTRUCTIONS

