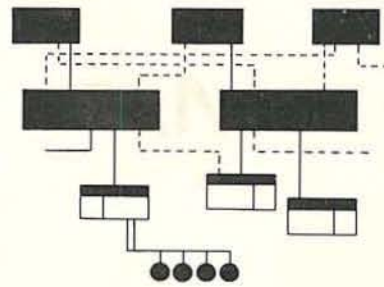
GENERAL ELECTRIC
COMPUTERS

GE-625/635 FORTRAN IV Math Library

SYSTEM
SUPPORT
INFORMATION



ABSTRACT

This manual describes FORTRAN IV Math Routines available for use with all configurations of the GE-625/635.

replaced by
1083B

GENERAL ELECTRIC

HO HO HO!

GENERAL ELECTRIC COMPUTER DEPARTMENT	GE-600 SERIES TECHNICAL INFORMATION BULLETIN	DATE FEB. 1966 NO. 600-82
SUBJECT: Corrections to GE-600 Series FORTRAN IV Math Library SSI		REF. TIB600-66 CPB-1083

This Technical Information Bulletin replaces TIB 600-66 and provides changes that affect four of the FORTRAN IV Math Library programs. Corrections should be made by pen and ink insertions to the existing pages of the manual. These changes will be incorporated in any future revised editions of the subject manual.

Instructions for making corrections:

Program	Page	
	iv	In the 3rd line, 1st word, change XP1 to read: FXP1
	v	In the 7th, 8th, and 9th lines, 1st words, change XP1, XP2, and XP3 to read: FXP1, FXP2, and FXP3, respectively.
CD600D2.001	1	In the program title heading, 1st word, change XP1 to read: FXP1 paragraph III, item 1, change .XP1. to read: .FXP1. item 2, change XP1 to read FXP1
	2	In the circle at the upper left-hand part of the page, change .XP1. to read: .FXP1.
CD600D2.002	1	In the program title heading, 1st word, change XP2 to read: FXP2 paragraph II, item 1, change XP1 to read: FXP1 item 4, 1st line change .XP2. to read: .FXP2. item 4, 2nd line, change .DXP1. to read: .FDXP1. paragraph III, item 1, 1st line change .XP2. to read: .FXP2. item 1, 2nd line change .DXP1. to read: .FDXP1. item 2, change XP2 to read FXP2
	2	Find the two circles at the upper left-hand part of the page. In the circle with .XP2., change to read: .FXP2. In the circle with .DXP1., change to read: .FDXP1.

These corrections require corrections then to the Fortran I/O manual CPB-1137 on pages 35

Program	Page	
CD600D2.003	1	In the program title heading, 1st word, change XP3 to read: FXP3
		paragraph III, item 1, change .XP3. to read: .FXP3. item 2, change XP3 to read: FXP2
	2	In the circle at the upper left-hand part of the page, change .XP3. to read: .FXP3.
CD600D2.004	1	In paragraph III, item 1, change .CXP1. to read: .FCXP1. IN paragraph II, item 1, change XP2 to read FXP2
	2	In the circle at the upper left-hand part of the page, change .CXP1. to read: .FCXP1.
CD600D2.005	1	In paragraph III, item 1, change .DXP2. to read: .FDXP2. In paragraph II, item 1, change XP3 to read FXP3
	2	In the circle at the upper left-hand part of the page, change .DXP2. to read: .FDXP2.
CD600D4.001	1	In paragraph III item 1, line 1, change .CFMP. to read: .FCFMP. item 1, line 2, change .CFDP. to read: .FCFDP.
	3	In the circle at the upper left-hand part of the page, change .CFDP. to read: .FCFDP.
		In the circle at the upper right-hand part of the page, change .CFMP. to read: .FCFMP.

See TIB 600-82! (with noted exceptions) 5/23/66

Technical Publications Group, S&PO
~~Information Systems Equipment Div~~ Computer Equipment Dept.
General Electric Co
PO Box 2961
Phoenix Arizona

No acknowledgement

Sir:

In reference to TIB 600-66 dated January 1966 (received on Sept 23) which references the 600-series Fortran IV Math Library SSI manual, CPB-1083, ~~may~~ I would like to make the following comments.

At first, I thought I had the wrong manual edition because the instructions did not agree with the page or program name indicated for the first and third corrections. When I realized that there had been no revised edition since the September 1964 edition, it became apparent that the instructions given were more than likely made up from the paste-up of the revised edition-to-be. Working from this assumption the following corrections (and additions) should be made to your ~~corrections~~ instructions in TIB 600-66

note: There was a revision dated Nov 65 but the errors still apply

~~Program Page~~

1. In the first instruction, change iii to read iv

2. Change CD600D1.001 to CD600D2.001

3. Under CD600D2.002 add the instruction:
Page 1, paragraph III, item 2, change XP2 to FXP2

4. Under CD600D2.004 add the instruction:
Page 1, paragraph II, item 1, change XP2 to FXP2 — not included

5. Under CD600D2005 add the instruction:
Page 1, paragraph II, item 1, change XP3 to FXP3 — not included

(ZERO DEFECTS program doesn't seem to be in effect!)

note the additions for BJC

GENERAL ELECTRIC COMPUTER DEPARTMENT	GE-600 SERIES TECHNICAL INFORMATION BULLETIN	DATE Jan. 1966
		NO. 600-66
SUBJECT: Corrections to GE-600 Series FORTRAN IV Math Library SSI		REF. CPB-1083

recd 2/23/66

This Technical Information Bulletin provides changes that affect four of the FORTRAN IV Math Library programs. Corrections should be made by pen and ink insertions to the existing pages of the manual. These changes will be incorporated in any future revised editions of the subject manual.

Instructions for making corrections:

Program	Page	
	iii	In the 3rd line, 1st word, change XP1 to read: FXP1
	v	In the 7th, 8th, and 9th lines, 1st words, change XP1, XP2, and XP3 to read: FXP1, FXP2, and FXP3, respectively.
CD600D1.001 (D2)	1	In the program title heading, 1st word, change XP1 to read: FXP1
		paragraph III, item 1, change .XP1. to read: .FXP1. item 2, change XP1 to read FXP1
	2	In the circle at the upper left-hand part of the page, change .XP1. to read: .FXP1.
CD600D2.002	1	In the program title heading, 1st word, change XP2 to read: FXP2
		paragraph II, item 1, change XP1 to read: FXP1 item 4, 1st line change .XP2. to read: .FXP2. item 4, 2nd line, change .DXP1. to read: .FDXP1.
		paragraph III, item 1, 1st line change .XP2. to read: .FXP2. item 1, 2nd line change .DXP1. to read: .FDXP1. item 2, change XP2 to FXP2
	2	Find the two circles at the upper left-hand part of the page. In the circle with .XP2., change to read: .FXP2. In the circle with .DXP1., change to read: .FDXP1.

Program	Page	
CD600D2.003	1	In the program title heading, 1st word, change XP3 to read: FXP3
		paragraph III, item 1, change .XP3. to read: .FXP3.
		item 2, change XP3 to read: FXP2
	2	In the circle at the upper left-hand part of the page, change .XP3. to read: .FXP3.
CD600D2.004	1	Paragraph II, item 1, Change XP2 to FXP2
	1	In paragraph III, item 1, change .CXP1. to read: .FCXP1.
	2	In the circle at the upper left-hand part of the page, change .CXP1. to read: .FCXP1.
CD600D2.005	1	Paragraph II, item 1, Change XP3 to FXP3
	1	In paragraph III, item 1, change .DXP2. to read: .FDXP2.
	2	In the circle at the upper left-hand part of the page, change .DXP2. to read: .FDXP2.
CD600D4.001	1	In paragraph III item 1, line 1, change .CFMP. to read: .FCFMP.
		item 1, line 2, change .CFDP. to read: .FCFDP.
	3	In the circle at the upper left-hand part of the page, change .CFDP. to read: .FCFDP.
		In the circle at the upper right-hand part of the page, change .CFMP. to read: .FCFMP.

GENERAL ELECTRIC COMPUTER DEPARTMENT	GE-600 SERIES TECHNICAL INFORMATION BULLETIN	DATE August 1965 NO. 600-33
SUBJECT: Corrections to GE-600 Series FORTRAN IV Math Library		REF. CPB-1083

Please insert the attached page in your FORTRAN IV Math Library manual.

GE-625/635
FORTRAN IV
MATH LIBRARY

September 1964
Rev. November 1965

no differences noted other than new cover format & title page format!

GENERAL ELECTRIC
COMPUTER DEPARTMENT

PREFACE

The FORTRAN IV Math Routines described in this manual are part of an integrated programming system available for the Compatibles/600. The numbers assigned to the writeups are the same as those assigned to the actual programs which they explain. The numbering system is described on the following page.

As is true of all programs for the GE-600 Series, The FORTRAN IV Math Library Routines are upward compatible. Any program described in this manual can be executed by any central processor in the GE-600 Series of computer systems.

The FORTRAN IV Math Library Manual is distributed in loose leaf form to facilitate the incorporation of additions and changes. As soon as new programs are completed, corresponding writeups will be made available to users. When changes become necessary, change pages will be distributed. Revised pages will be identified by the date at the top of the page, and revisions within pages will be identified by a bar in the margin beside the sentence or sentences changed.

NUMBERING SYSTEM

The FORTRAN IV Math Routines included in this publication are each assigned a number in accordance with a numbering system used for all 600-Series programming routines. For example, XP1--Exponential--Integer Base and Exponent is assigned the number CD600D2.001. This number is described to illustrate the numbering system.

CD600D2.001

The last three digits, which always follow a decimal point, make a sequential listing of the routines in the order they are made available to the Program Library. The sequence is within the classification of the number and letter to the left of the decimal point.

The digit before the decimal point makes a grouping of routine types within the alphabetic classification described in the following paragraph. The Math Routines are classified in eight categories:

1. Programmed Arithmetic
2. Elementary Functions
3. Statistical Routines
4. Operations on Matrices, Vectors and Simultaneous Equations
5. Polynomial and Special Functions
6. Curve Fitting and Other Approximations
7. Operations Research
8. Numerical Integration and Differentiation and Solutions of Differential Equations

The alphabetic letter in the center of the number classifies the routines according to the following list:

A. Diagnostic Routines
B. Service Routines
C. Internal Data Manipulation
D. Math Routines
E. Input/Output Routines
F. Assembly Systems
G. Generators
H. Compilers/Translators
I. Simulators
J. Service Systems
K. Special Systems

The 600 means that the programs are programmed for use on the GE-600 Series Computer Systems.

The CD means that the program was originated by the General Electric Computer Department.

CALEN - GE600 Calendar 0765 A9.001
UTILITY Utility Program 0567 B3.001
BHC - Binary-Hollerith Conversions 0765 C2.001
SORTR - Fortran Internal Sort 070265 C5.001 352

CONTENTS

ALGT base? log
ASIN arcsin/cos
TANG tan/cot
SINH hyperbolic sin/cos

D3.007 is BMD version 11/65

FDMD--DOUBLE-PRECISION MODULUS

I. PURPOSE

To compute A = X (mod Y) for DMOD(X,Y) in an expression.

II. METHOD

1. If Y = 0, then A = X. Otherwise, compute

   Z = the greatest integer $\leq \frac{|X|}{|Y|}$ and give

   Z the same sign as that of $\frac{X}{Y}$. Then A = X - Y * Z.

2. A, X, and Y are double-precision numbers, with values from -2^{127} to $2^{127}-2^{64}$ inclusive.

3. A is accurate to 63 binary positions.

III. USAGE

1. Calling Sequence--CALL DMOD(X,Y)

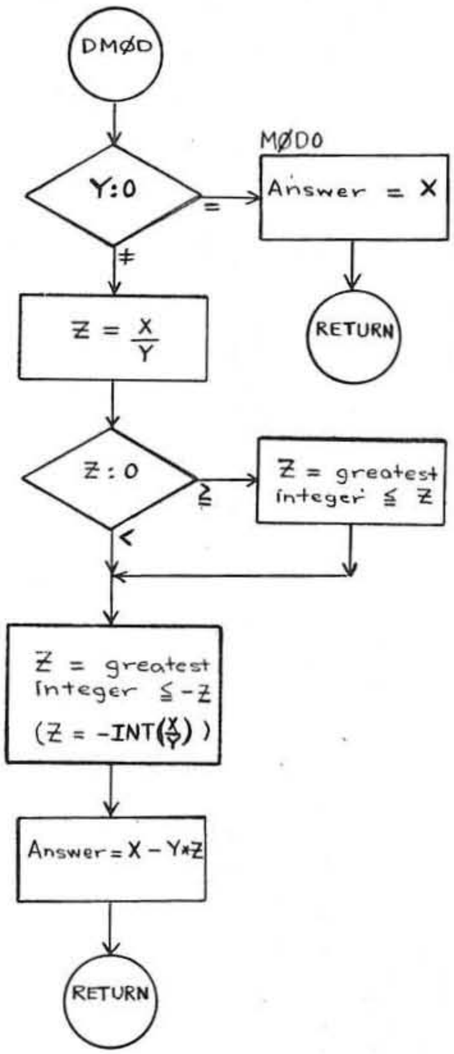
2. FDMD uses 16 words.

3. No error conditions.

IV. RESTRICTIONS

None.

COMPUTE X (MOD Y) FOR DOUBLE PRECISION X AND Y



FDXP--DOUBLE-PRECISION EXPONENTIAL

I. PURPOSE

To compute e^X for EXP(X) in an expression.

II. METHOD

1. Use the same method as in FXPF--Real Natural Exponential, CD600D3.001, except that $2^F = 1 + F \log_e 2 + \frac{(F \log_e 2)^2}{\underline{|2}} + \ldots + \frac{(F \log_e 2)^{13}}{\underline{|13}}$

2. X and e^X are double-precision numbers, with $|X| \leq 88.028$

3. e^X is accurate to 16 decimal positions.

III. USAGE

1. Calling Sequence--CALL DEXP(X)

2. FDXP uses 68 words. 122(8)

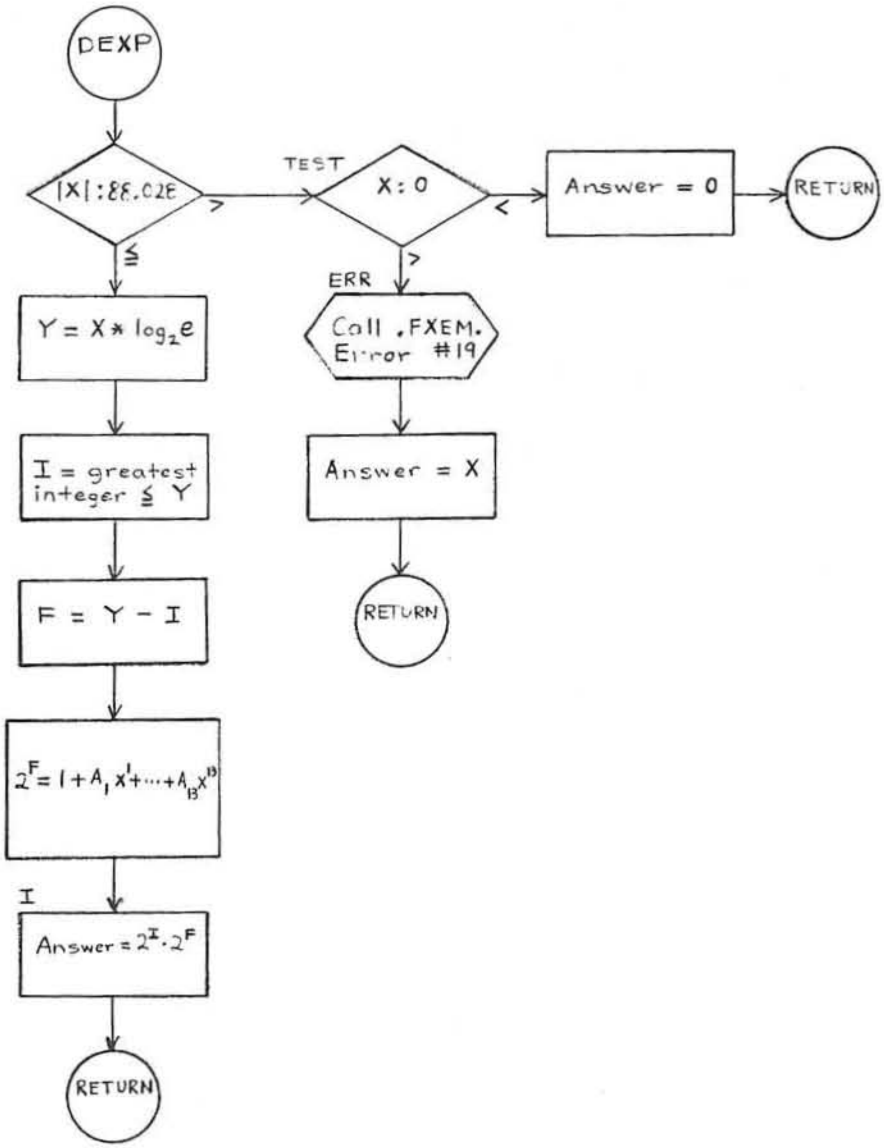
3. The error condition is:

   FXEM Error #19 if $|X| > 88.028$. Then $e^X = X$.

IV. RESTRICTIONS

The subprogram FXEM must be in memory.

COMPUTE e^X FOR DOUBLE PRECISION X



FDSQ--DOUBLE-PRECISION SQUARE ROOT

I. PURPOSE

To compute $\sqrt{X}$ for DSQRT(X) in an expression.

II. METHOD

1. Use the same method as in FSQR--Real Square Root, CD600D3.006 except that $P_3 = \frac{1}{2} * (P_2 + \frac{F}{P_2})$ and $\sqrt{X} = 2^{A-1} * (P_3 + \frac{F}{P_3})$.
2. X and $\sqrt{X}$ are double-precision numbers, with values of X from 0 to $2^{127} - 2^{64}$ inclusive.
3. $\sqrt{X}$ is accurate to 18 decimal positions.

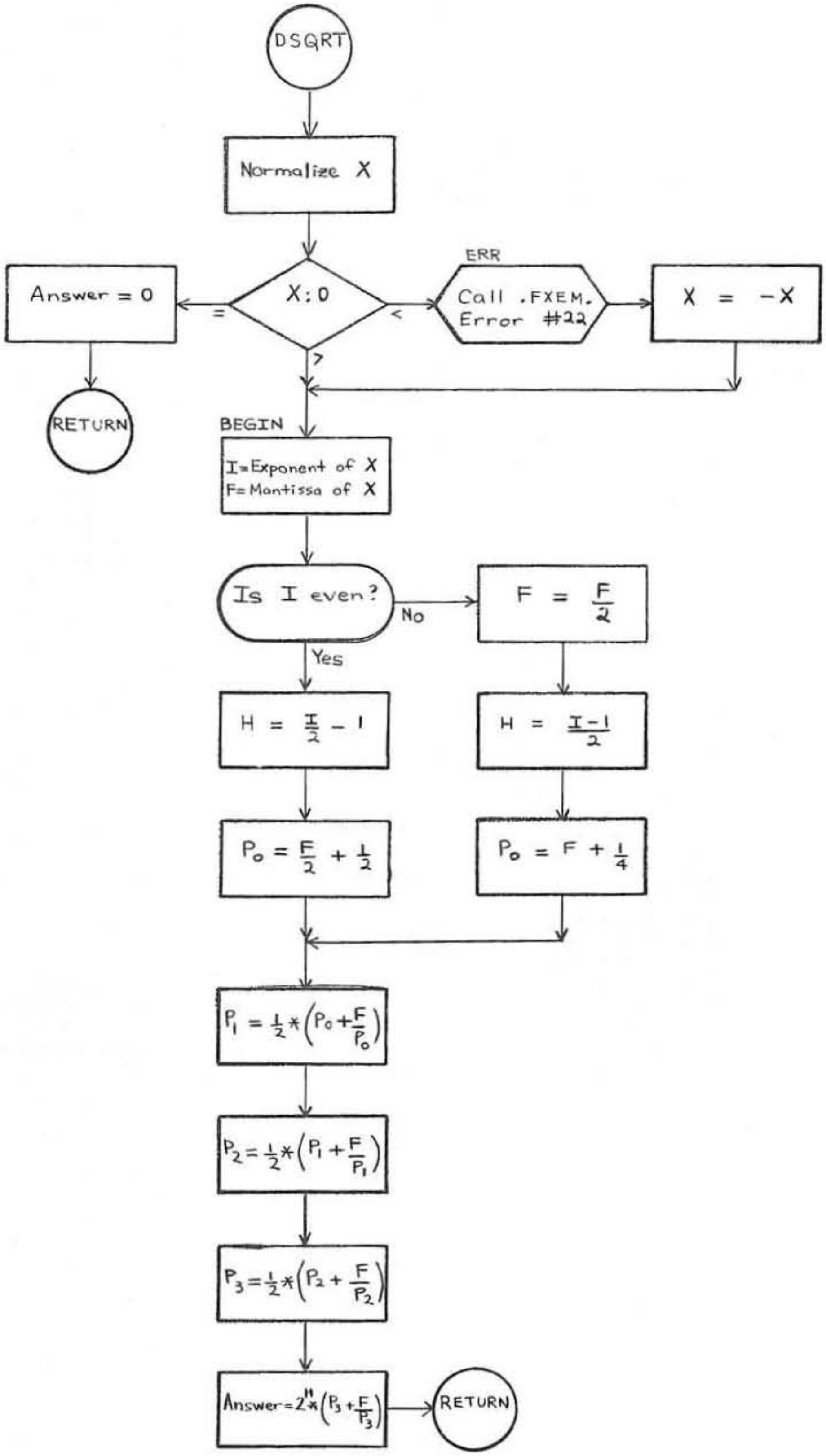
III. USAGE

1. Calling Sequence--CALL DSQRT(X)
2. FDSQ uses 50 words.
3. The error condition is:
   FXEM Error #22 if $X < 0$. Then $\sqrt{X} = \sqrt{|X|}$.

IV. RESTRICTIONS

The subprogram FXEM must be in memory.

COMPUTE $\sqrt{X}$ FOR DOUBLE PRECISION X



FDSN

FDSC--DOUBLE-PRECISION SINE AND COSINE

I. PURPOSE

To compute sin X or cos X for DSIN(X) or DCOS(X) in an expression, where X is in radians.

II. METHOD

1. Use the same method as in FSCN--Real Sine and Cosine, CD600D3.004, with the following exceptions:

   a. Do not make $X < \frac{1}{256}$ a special case. Use $\frac{\pi}{2}$ instead of 0.3 as the breakpoint.

   b. Use a Taylor Series approximation instead of a Continued Fraction:

   $\sin X = X - \frac{X^3}{\lfloor 3} + \frac{X^5}{\lfloor 5} - \dots$ or $\cos X = 1 - \frac{X^2}{\lfloor 2} + \frac{X^4}{\lfloor 4} - \dots$

   Include enough terms in the series until $\frac{X^n}{\lfloor n} < \frac{\text{first term}}{10^{18}}$.

   (When $\frac{\text{first term}}{10^{18}} = 0$, include only the first term in the series.)

2. X, sin X, and cos X are double-precision numbers with $|X| < 2^{54}$.

3. The answer is accurate to 18 decimal positions.

III. USAGE

1. Calling Sequence--CALL DSIN(X) for sin X
   CALL DCOS(X) for cos X

2. DSCN uses ~~98~~ 112₁₀ words. 160(8)

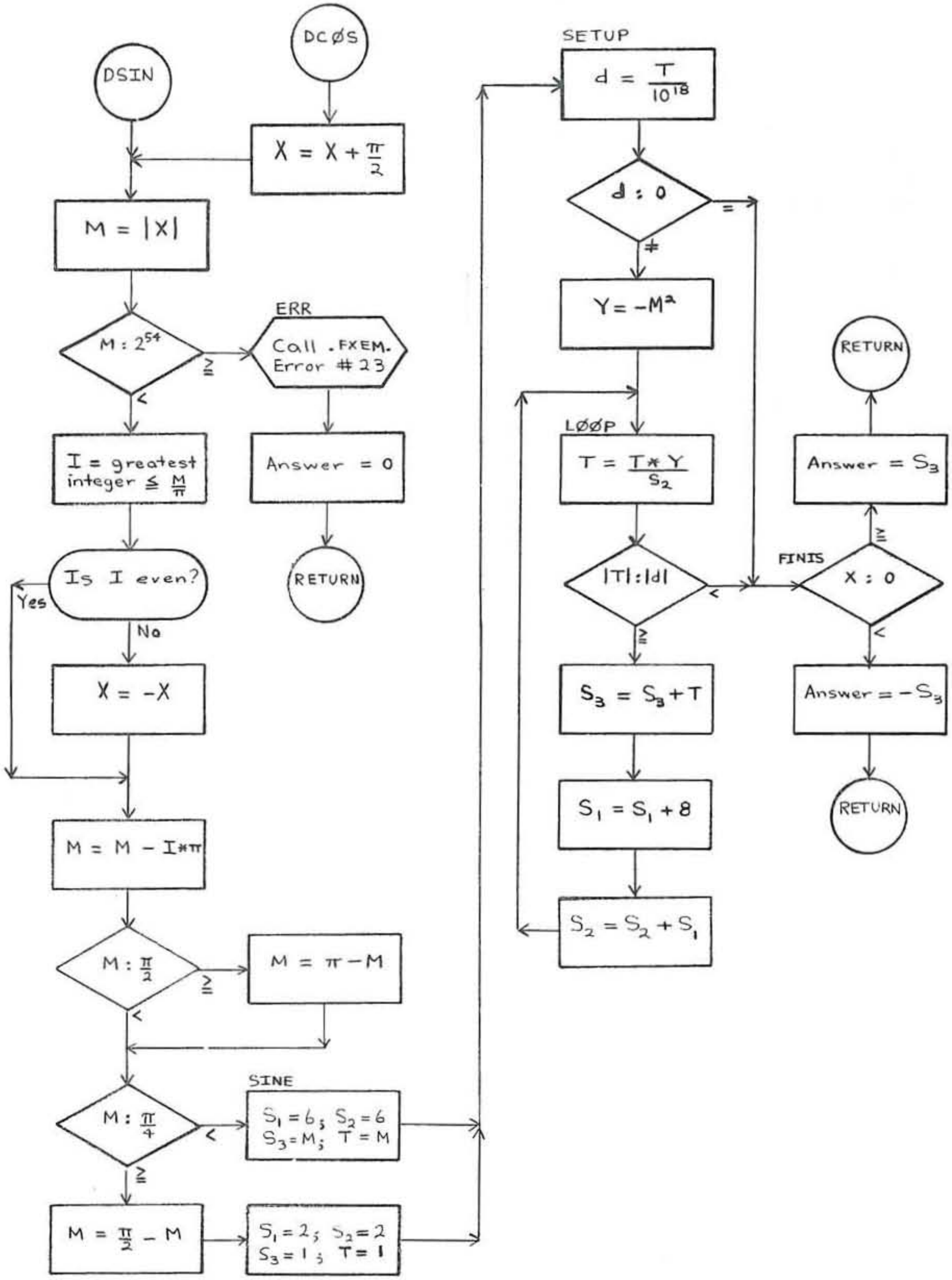
3. The error condition is:

   FXEM Error #23 if $|X| \geq 2^{54}$. Then the answer is 0.

IV. RESTRICTIONS

The subprogram FXEM must be in memory.

COMPUTE SIN X OR COS X FOR DOUBLE PRECISION X



FDAT--DOUBLE-PRECISION ARCTANGENT

I. PURPOSE

To compute the principal value of arctan X or arctan $\frac{Y}{Z}$ (in radians) for DATAN(X) or DATAN2 (Y,Z) in an expression.

II. METHOD

1. Use the same method as in FATN--Real Arctangent, CD600D3.003 with the following exceptions:

   a. The intervals are 0° - 7.5°, 7.5° - 22.5°, 22.5° - 37.5°, 37.5° -52.5°, 52.5° - 67.5°, and 67.5° - 82.5°. For 82.5° - 90°, compute $\frac{\pi}{2}$ - arctan $\frac{1}{X}$, where arctan $\frac{1}{X}$ is in the first interval.

   b. For 0° - 7.5°, $T = AL_6 * X$. Otherwise, $T = AL_I - \frac{BETA_I}{G_I + X}$.

   c. $\arctan X = N_I + \frac{C_{12} * T}{C_{12} * C_{14} - C_8}$, where $C_{14} = B + T^2$,

   $C = B_2 + T^2$, $C_2 = B_4 + T^2$, $C_4 = B_6 + T^2$, $C_6 = C_2 * C_4 - A_4$, $C_8 = A * C_6$, $C_{10} = C * C_6$, $C_{12} = C_{10} - A_2 * C_4$.

2. X, Y, and Z are double-precision numbers, with values from $-\left(2^{127}\right)$ to $\left(2^{127} - 2^{64}\right)$ inclusive. The answer is a double-precision number.

3. The answer is accurate to 16 decimal positions.

III. USAGE

1. Calling Sequence--CALL DATAN(X) for arctan X
   CALL DATAN2(Y,Z) for arctan $\frac{Y}{Z}$

2. FDTN uses 204 words.

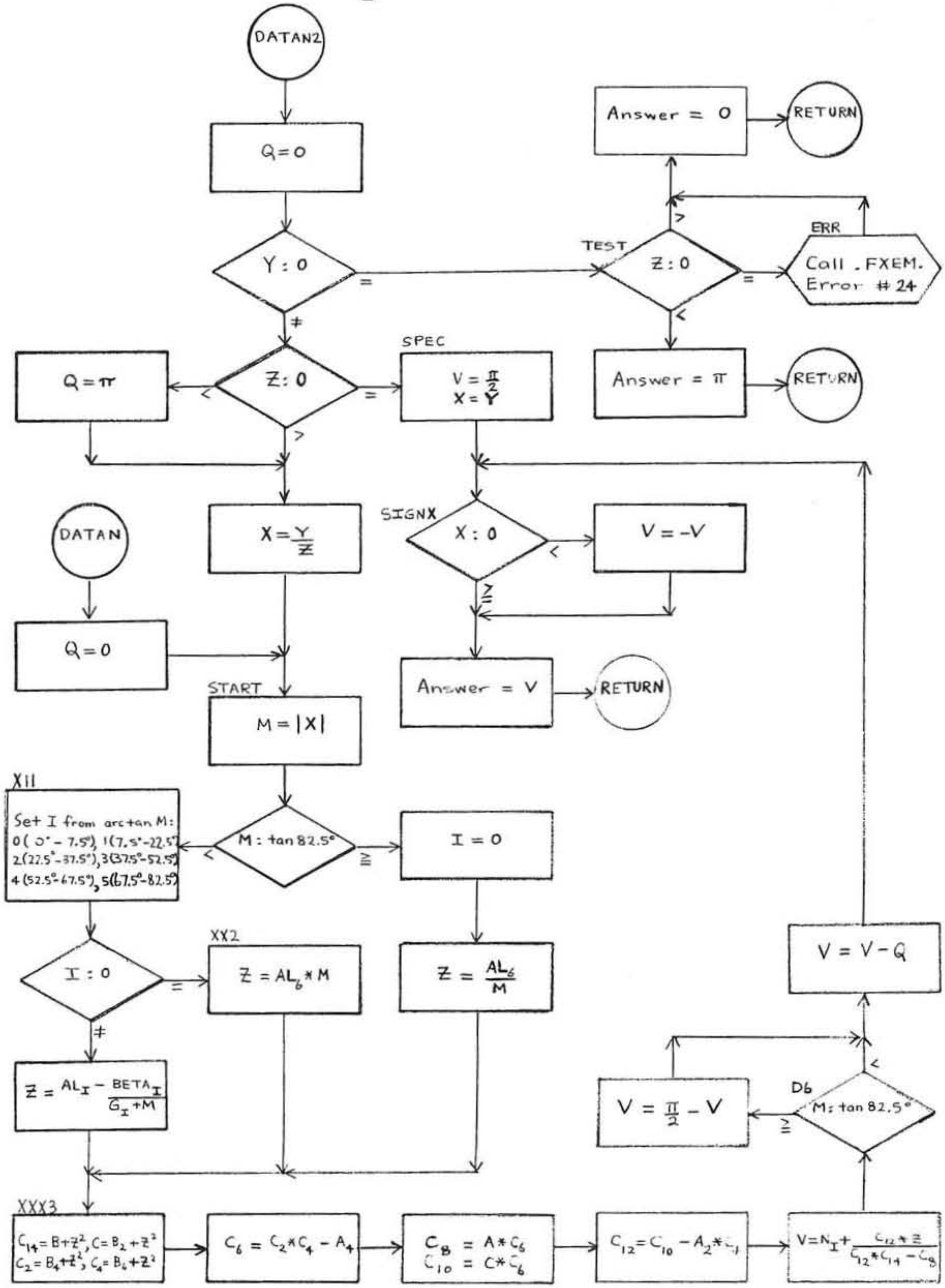
3. The error condition is:

   FXEM Error #24 if Y = 0 and Z = 0. Then arctan $\frac{Y}{Z} = 0$.

IV. RESTRICTIONS

The subprogram FXEM must be in memory.

COMPUTE ARCTAN X OR ARCTAN $\frac{Y}{Z}$ FOR DOUBLE PRECISION X, Y, AND Z



FDLG--DOUBLE-PRECISION LOGARITHM

I. PURPOSE

To compute log X for DLOG(X) or DLOG10(X) in an expression.

II. METHOD

1. $\log_2 X = \log_2 (2^I * F) = I + \log_2 F$, where $X = 2^I * F$.

2. $\log_e X = \log_e 2^{(\log_2 X)} = (\log_2 X) * (\log_e 2)$

   $= I * \log_e 2 + (\log_2 F) * (\log_e 2)$

   $= I * \log_e 2 + \log_e 2^{(\log_2 F)}$

   $= I * \log_e 2 + \log_e F$

3. Let A = most significant 5 bits of F and let $Z = \frac{F - A}{F + A}$

   Then $\log_e F = \log_e A + 2* \left(Z + \frac{Z^3}{3} + \ldots + \frac{Z^{11}}{11}\right)$

4. $\log_{10} X = (\log_e X) * (\log_{10} e)$

5. X and log X are double-precision numbers; values of X range from 2^{-129} to $2^{127}-2^{64}$ inclusive.

6. log X is accurate to 16 places.

III. USAGE

1. Calling Sequence--CALL DLOG(X) for $\log_e X$
   CALL DLOG10(X) for $\log_{10} X$

2. FDLG uses 120 words.

3. The error conditions are:

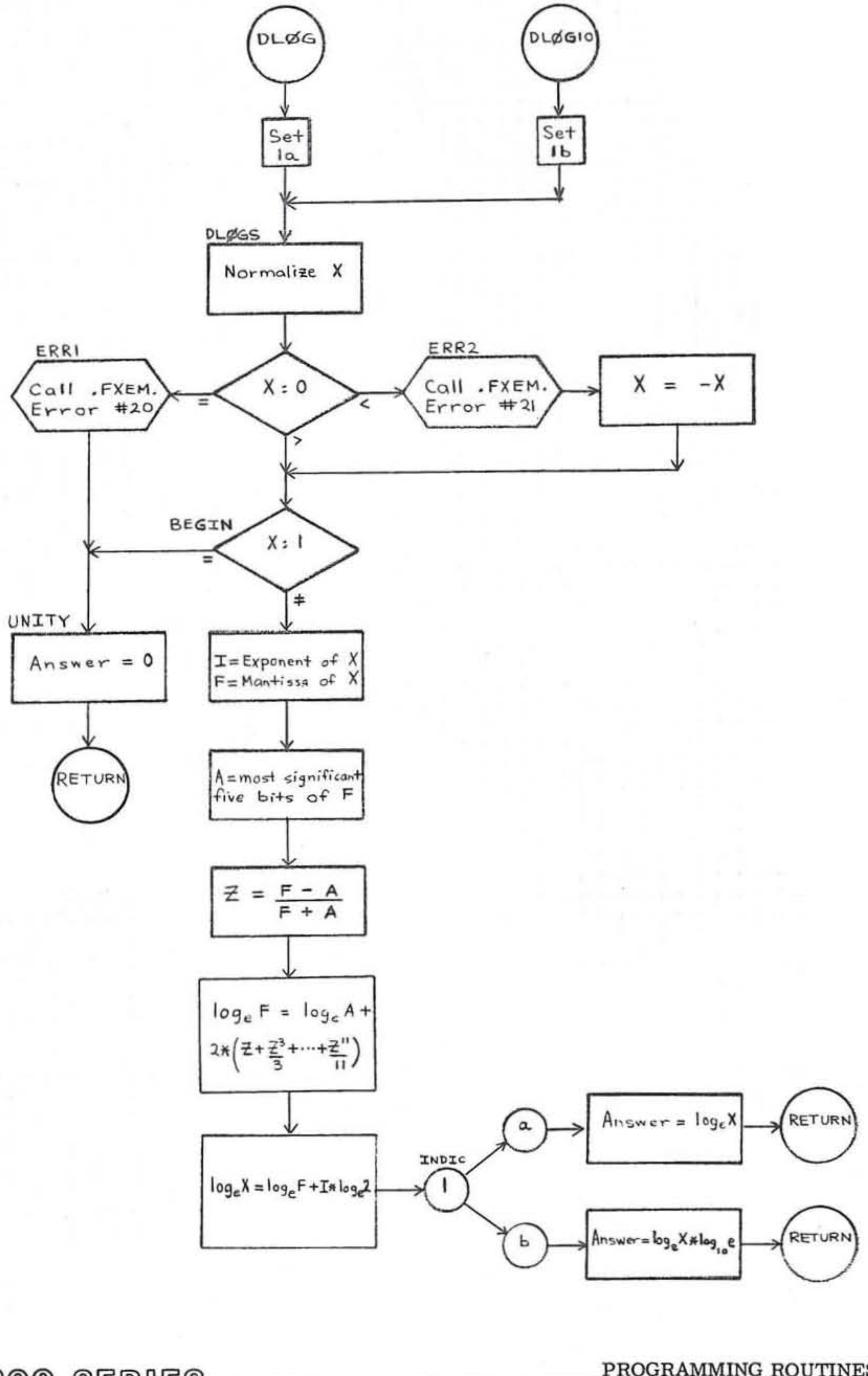
   a. FXEM Error #20 if X = 0. Then log X = 0.

   b. FXEM Error #21 if X < 0. Then log X = log |X| .

IV. RESTRICTIONS

The subprogram FXEM must be in memory.

COMPUTE $\log_e X$ OR $\log_{10} X$ FOR DOUBLE PRECISION X



XP1--EXPONENTIAL--INTEGER BASE AND EXPONENT

I. PURPOSE

To compute I^J for I**J in an expression.

II. METHOD

1. For positive values of J, let $k_m \ldots k_2 k_1 k_0$ be the binary representation of J, where $0 \leqq m \leqq 34$.

   Then $I^J = I^{(k_0 + 2*k_1 + 4*k_2 + \ldots + 2^m*k_m)}$

   $= (I^1)^{k_0}*(I^2)^{k_1}*(i^4)^{k_2}*\ldots*(I^{(2^m)})^{k_m}$

   = the product of those powers of I above for which $k_n = 1$, where $0 \leqq n \leqq m$.

2. For negative values of J, $I^J = 0$ if $|I| \neq 1$. Use the method above with J (mod 2) if $|I| = 1$.

3. I, J, and I^J are integers with values from -2^{35} to $2^{35}-1$ inclusive.

4. The algorithm uses integer multiplication (MPY) exclusively.

5. I^J is accurate to 35 binary positions.

III. USAGE

1. Calling Sequence--CALL .XP1. (I,J)

2. XP1 uses 52 words.

3. The error conditions are:

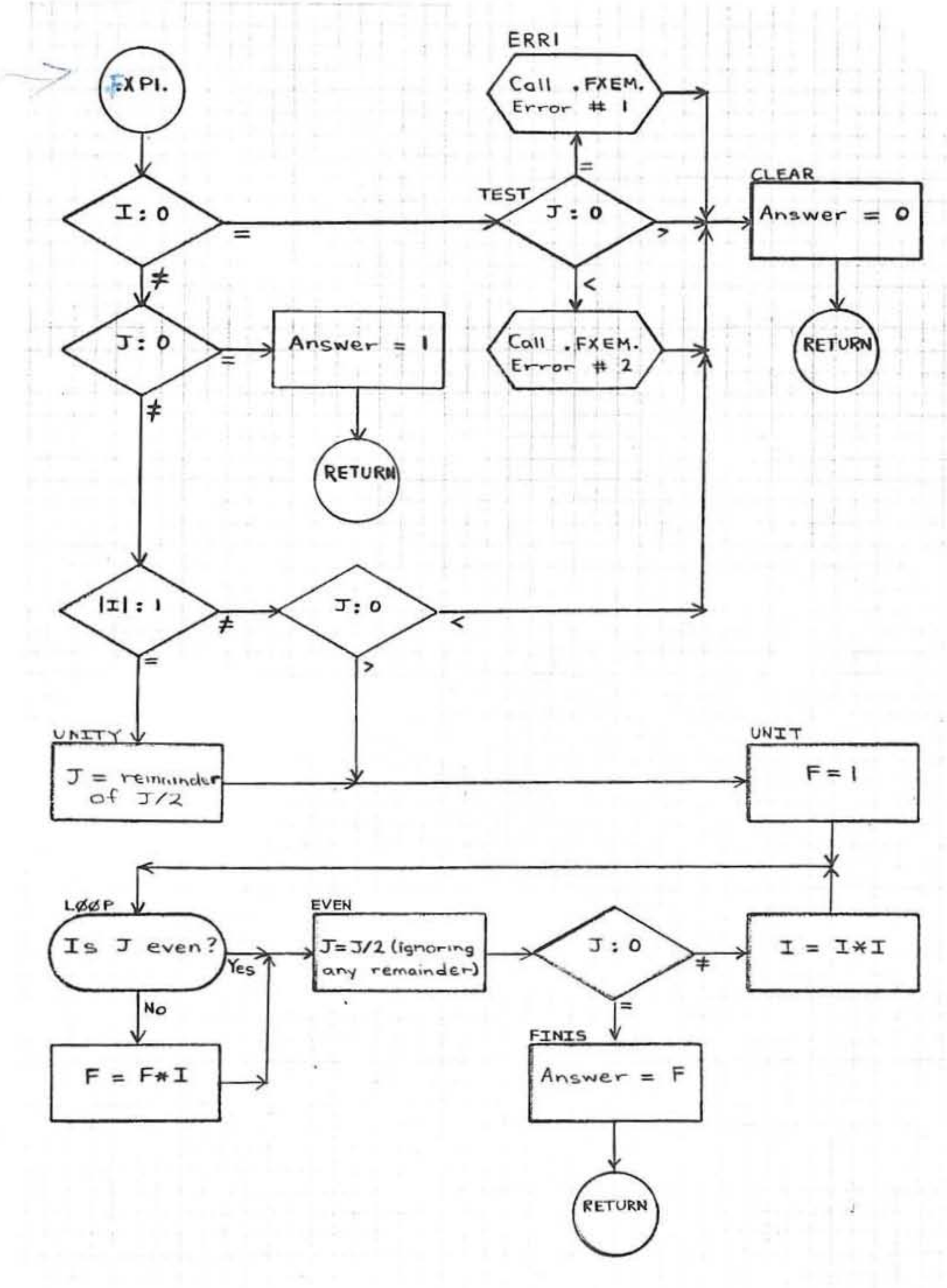
   a. FXEM Error #1 if I = 0 and J = 0. Then $I^J = 0$.

   b. FXEM Error #2 if I = 0 and J < 0. Then $I^J = 0$.

IV. RESTRICTIONS

The subprogram FXEM must be in memory.

COMPUTE I^J FOR INTEGERS I AND J



XP2--EXPONENTIAL--FLOATING-POINT BASE, INTEGER EXPONENT

I. PURPOSE

To compute A^K for A**K in an expression.

II. METHOD

1. For positive values of K, use the same method as in XP1--Exponential--Integer Base and Exponent, CD600D2. 001.

2. For negative values of K, proceed with |K| as above, and then take the reciprocal of the result.

3. K is an integer with values from -2^{35} to $2^{35}-1$ inclusive; A and A^K are floating-point numbers with values from -2^{127} to $2^{127}-2^{64}$ inclusive.

4. A^K is accurate to 8 decimal positions for .XP2.
   or 16 decimal positions for .DXP1.

III. USAGE

1. Calling Sequence--CALL.XP2. (A,K) for Real A
   CALL.DXP1. (A,K) for Double-Precision A

2. XP2 uses 60 words.

3. The error conditions are:

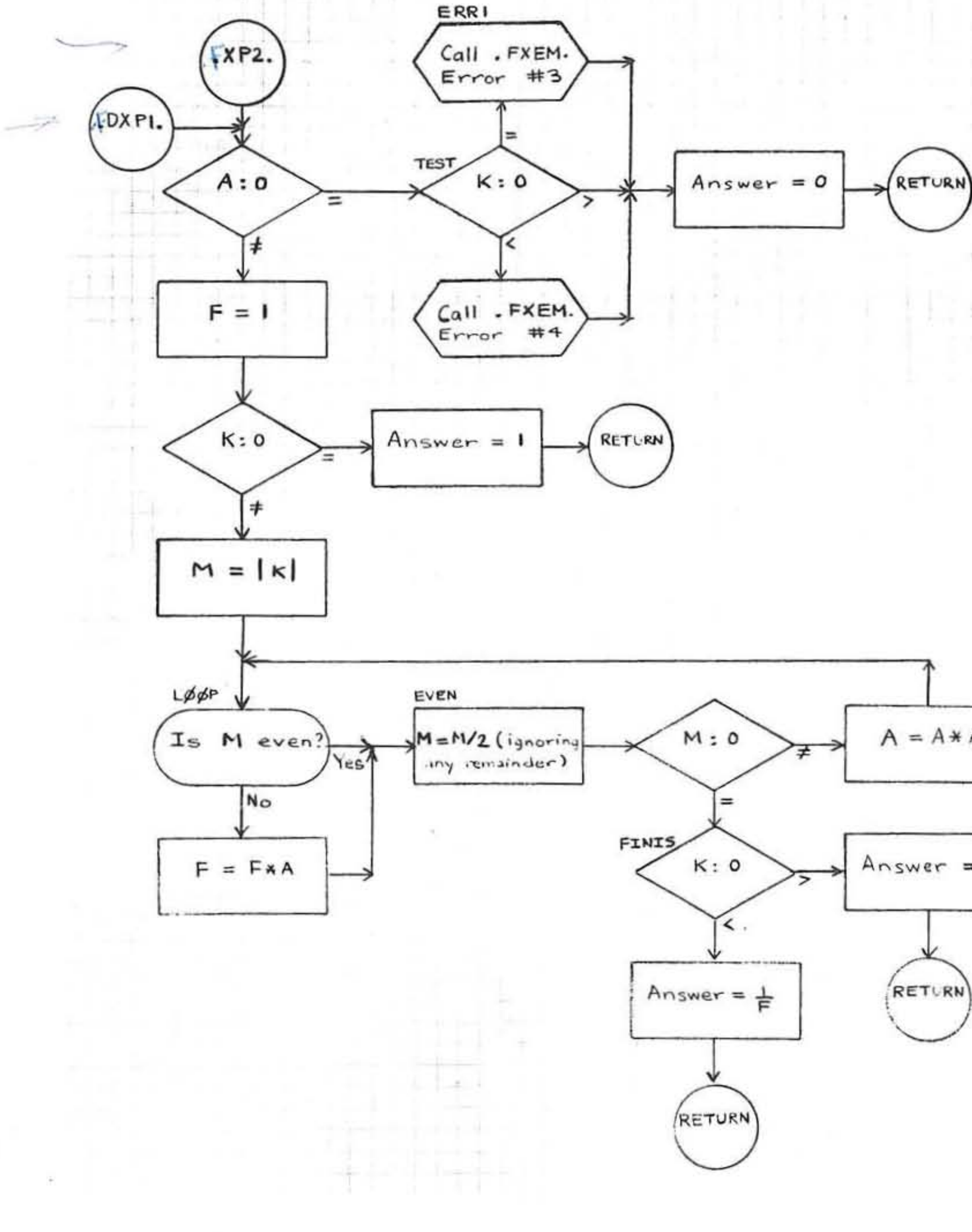
   a. FXEM Error #3 if A = 0 and K = 0. Then $A^K = 0$.

   b. FXEM Error #4 if A = 0 and K < 0. Then $A^K = 0$.

IV. RESTRICTIONS

The subprogram FXEM must be in memory.

COMPUTE A^K FOR FLOATING POINT A AND INTEGER K



F3XP

FXP3--EXPONENTIAL--REAL BASE AND EXPONENT

I. PURPOSE

To compute A^B for A**B in an expression.

II. METHOD

1. $A^B = (e^{\log_e A})^B = e^{(B*\log_e A)}$

2. A, B, and A^B are real numbers with values from -2^{127} to $2^{127}-2^{100}$ inclusive.

3. A^B is accurate to 7 decimal positions.

III. USAGE

1. Calling Sequence--CALL.FXP3. (A,B)

2. FXP3 uses ~~50~~ 80₁₀ words. 120₈

3. The error conditions are:

   a. FXEM Error #5 if $A < 0$ and $B \neq 0$. Then $A^B = |A|^B$.

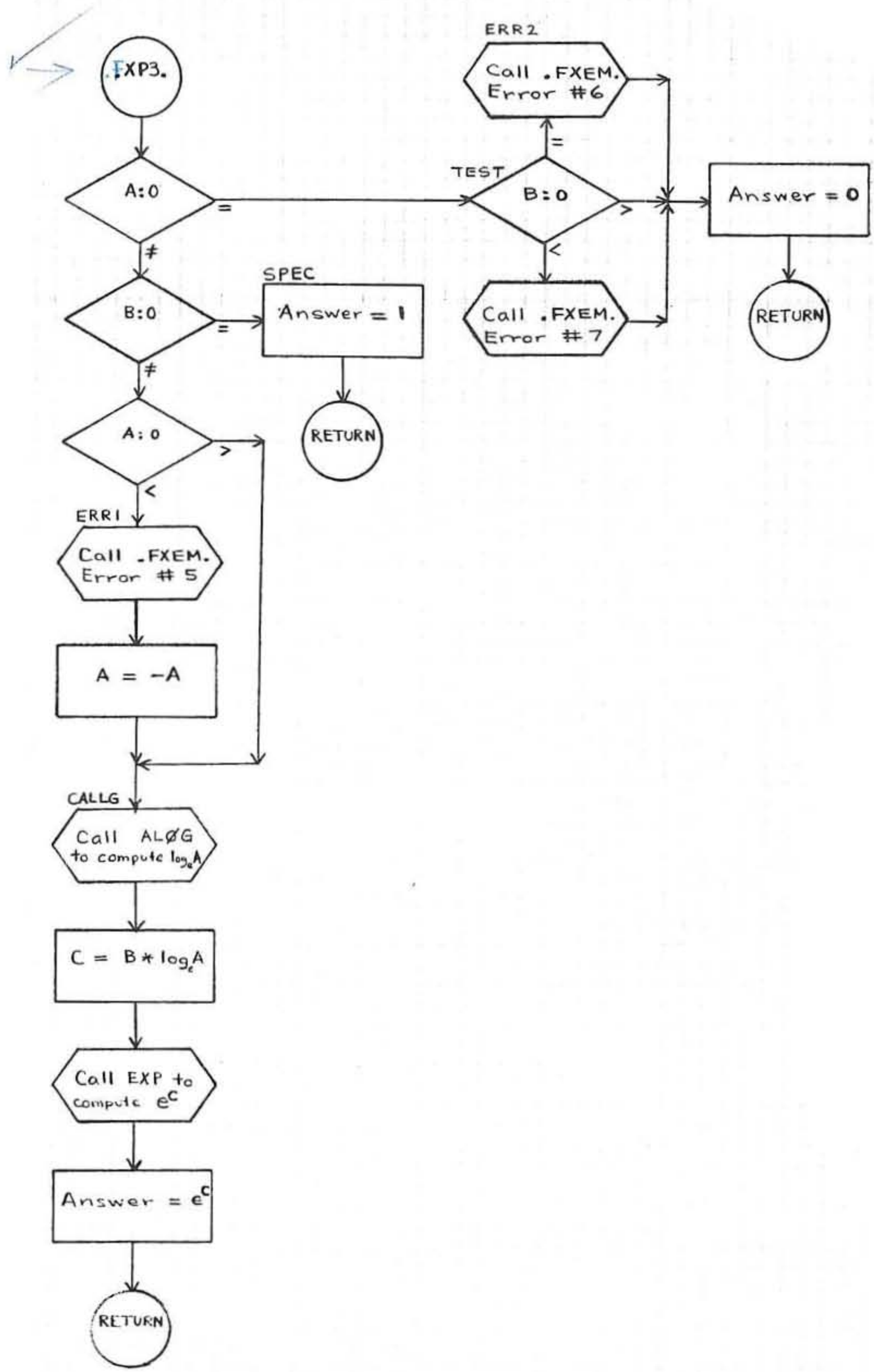
   b. FXEM Error #6 if $A = 0$ and $B = 0$. Then $A^B = 0$.

   c. FXEM Error #7 if $A = 0$ and $B < 0$. Then $A^B = 0$.

IV. RESTRICTIONS

The subprograms FLOG, FXPF, and FXEM must be in memory.

COMPUTE A^B FOR REAL A AND B



FCXP

FDX1--EXPONENTIAL--COMPLEX BASE, INTEGER EXPONENT

I. PURPOSE

To compute A^K for A**K in an expression.

II. METHOD

F2 XP

1. Use the same method as in FXP2--Exponential--Floating Point Base, Integer Exponent, CD600D2.002.

2. A is a complex number (X, Y), with values of X and Y from -2^{127} to $2^{127} -2^{100}$ inclusive. K is an integer, with values from -2^{35} to $2^{35} - 1$ inclusive.

3. A^K is accurate to 8 decimal positions.

III. USAGE

1. Calling Sequence--CALL.FCXP1. (A,K)

2. FDX1 uses 68 words. 88(10) 130_8

3. The error conditions are:

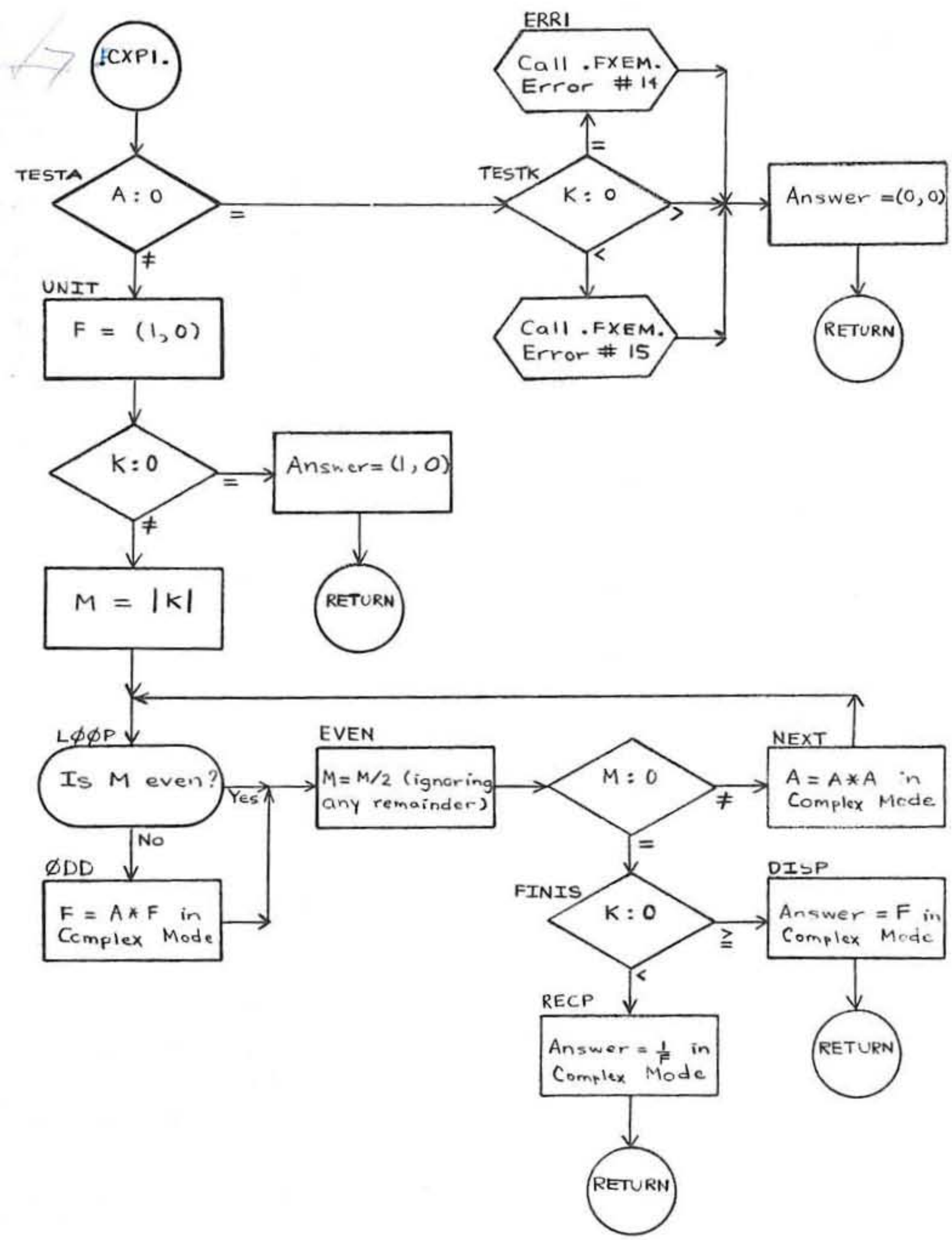
   a. FXEM Error #14 if A = (0,0) and K = 0.
   Then A^K = (0,0).

   b. FXEM Error #15 if A = (0,0) and K < 0.
   Then A^K = (0,0).

IV. RESTRICTIONS

The subprograms FCAS and FXEM must be in memory.

COMPUTE A^K FOR COMPLEX A AND INTEGER K



FDX2--EXPONENTIAL--DOUBLE-PRECISION BASE AND EXPONENT

I. PURPOSE

To compute A^B for A**B in an expression.

II. METHOD

1. Use the same method as in XP3--Exponential--Real Base and Exponent, CD600D2.003.

2. A, B, and A^B are double-precision numbers, with values from -2^{127} to $2^{127}-2^{64}$ inclusive.

3. A^B is accurate to 16 decimal digits.

III. USAGE

1. Calling Sequence--CALL FDXP2. (A, B)

2. FDX2 uses 52 words.

3. The error conditions are:

   a. FXEM Error #16 if $A < 0$ and $B \neq 0$. Then $A^B = |A|^B$.

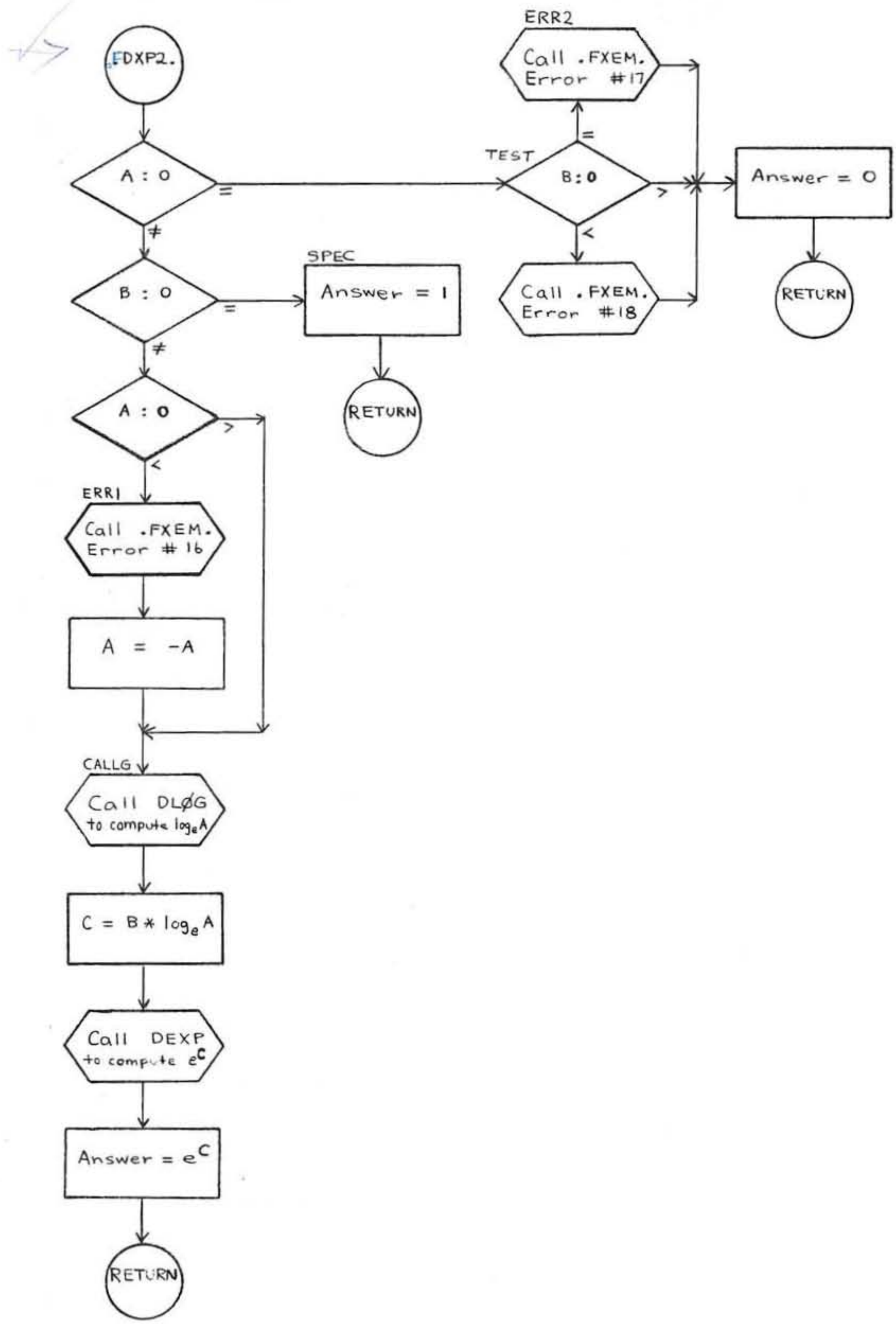
   b. FXEM Error #17 if $A = 0$ and $B = 0$. Then $A^B = 0$.

   c. FXEM Error #18 if $A = 0$ and $B < 0$. Then $A^B = 0$.

IV. RESTRICTIONS

The subprograms FDLG, FDXP, and FXEM must be in memory.

COMPUTE A^B FOR DOUBLE PRECISION A AND B



REAL AND DOUBLE-PRECISION LOGARITHM, BASE 2 (ALGT)

PURPOSE

Real and Double-Precision Logarithm (ALGT) computes $A = \log_2 X$ in a FORTRAN expression.

METHOD

The double-precision $\log_e X$ is divided by $\log_e 2$ and the result returned. Because of the hardware representation, this result is valid for both double and single precision.

USAGE

ALGT is designed to be used as a FORTRAN IV function:

A = ALOG2(X) for single precision;

A = DLOG2(X) for double precision.

RESTRICTIONS

The argument for ALGT must be ≥ 0.

If X = 0, FXEM Error #20 is returned and $\log_2 X = 0$.

If X < 0, FXEM Error #21 is returned and $\log_2 X = \log_2 |X|$.

ARCSINE AND ARCCOSINE (ASIN)

PURPOSE

Arcsine and Arccosine routine (ASIN) computes $\sin^{-1}X$ or $\cos^{-1}X$ in a FORTRAN IV expression.

METHOD

The arcsine or arccosine is calculated by computing the complementary function (sine or cosine), and calling ATAN2 (sin,cos) to get the resulting angle in radians. The computation is done entirely in double precision.

USAGE

ASIN is used as a FORTRAN IV function in the following ways:

- A = ASIN(X) for real arcsine;
- A = ACOS(X) for real arccosine;
- A = DASIN(X) for double-precision arcsine;
- A = DACOS(X) for double-precision arccosine.

TANGENT & COTANGENT (TANG)

PURPOSE

The Tangent and Cotangent routine (TANG) computes tan X or cot X in a FORTRAN IV expression.

METHOD

Using double-precision arithmetic, tan X and cot X are computed from the trigonometric identities:

- Tan X = sin X/cos X
- Cot X = cos X/sin X

If the divisor is zero, the largest possible floating-point number is returned.

USAGE

TANG is used as a FORTRAN IV function in the following ways:

- A = TAN(X) for real tangent;
- A = COT(X) for real cotangent;
- A = DTAN(X) for double-precision tangent;
- A = DCOT(X) for double-precision cotangent.

RESTRICTIONS

TANG produces FXEM Error #23 if $|X| > 2^{54}$.

HYPERBOLIC SINE AND COSINE (SINH)

PURPOSE

The Hyperbolic Sine and Cosine routine (SINH) computes sinh X or cosh X in a FORTRAN IV expression.

METHOD

Sinh X and cosh X are computed, using double-precision arithmetic, from the definitions:

- Sinh X = $0.5(e^{x} - e^{-x})$
- Cosh X = $0.5(e^{x} + e^{-x})$

USAGE

SINH is used as a FORTRAN IV function in the following ways:

- A = SINH(X) for real hyperbolic sine;
- A = COSH(X) for real hyperbolic cosine;
- A = DSINH(X) for double-precision hyperbolic sine;
- A = DCOSH(X) for double-precision hyperbolic cosine.

RESTRICTIONS

SINH produces FXEM Error #19 if $|X| > 88.028$.

MATRIX ADDITION ROUTINE (MADD)

PURPOSE

The Matrix Addition routine (MADD) performs addition of two real matrices.

USAGE

The matrices A and B and the result C are assumed to be stored as i by j matrices in m by n arrays. Associated with each matrix is a dimension vector of four integers (i, j, m, n).

The calling sequence to MADD is:

CALL MADD (A,IA,B,IB,C,IC,IND) to calculate [C] = [A] + [B]

with IA, IB, and IC being the dimension vectors of A, B, and C, respectively.

IND is an error indicator set as follows:

- IND = 0 for correct results;
- IND = 1 if results would have been larger than m_c by n_c;
- IND = 2 if the dimensions are not consistent.

Consistent dimensions are $i_A = i_B = i_C$ and $j_A = j_B = j_C$.

MATRIX SUBTRACTION ROUTINE (MSUB)

PURPOSE

The Matrix Subtraction routine (MSUB) performs subtraction of two real matrices.

USAGE

The matrices A and B and the result C are assumed to be stored as i by j matrices in m by n arrays. Associated with each matrix is a dimension vector of four integers (i, j, m, n).

The calling sequence to MSUB is:

CALL MSUB (A,IA,B,IB,C,IC,IND) to calculate [C] = [A] - [B]

with IA, IB, and IC being the dimension vectors of A, B, and C, respectively.

IND is an error indicator set as follows:

- IND = 0 for correct results;
- IND = 1 if results would have been larger than m_c by n_c;
- IND = 2 if the dimensions are not consistent.

Consistent dimensions are $i_A = i_B = i_c$ and $j_A = j_B = j_C$.

MATRIX MULTIPLY ROUTINE (MMPY)

PURPOSE

The Matrix Multiply routine (MMPY) calculates the product of two real matrices.

USAGE

The matrices A and B and the result C are assumed to be stored as i by j matrices in m by n arrays. Associated with each matrix is a dimension vector of four integers (i, j, m, n).

The calling sequence to MMPY is:

CALL MMPY (A,IA,B,IB,C,IC,IND) to calculate $[C] = [A] \times [B]$

with IA, IB, and IC being the dimension vectors of A, B, and C, respectively.

IND is an error indicator set as follows:

- IND = 0 for correct results;
- IND = 1 if results would have been larger than m_c by n_c;
- IND = 2 if the dimensions are not consistent.

Consistent dimensions are $j_A = i_B$, $i_C = i_A$, and $j_C = j_B$.

MATRIX TRANSPOSE ROUTINE (MTRN)

PURPOSE

The Matrix Transpose routine (MTRN) transposes a matrix.

USAGE

The matrix A and its transpose C are stored as i by j matrices in m by n arrays. Associated with each matrix is a dimension vector of four integers (i, j, m, n).

The calling sequence to MTRN is

CALL MTRANS (A, IA, C, IC, IND) to form $[C] = [A]^T$

with IA and IC being the dimension vectors of A and C, respectively.

IND is an error indicator set as follows:

- IND = 0 for correct results;
- IND = 1 if results would not have fit within an m_c by n_c array.
- IND = 2 if the dimensions are not consistent.

Consistent dimensions are $i_A = j_c$ and $j_c = i_c$.

MATRIX MOVE ROUTINE (MMOV)

PURPOSE

The Matrix Move routine (MMOV) moves a submatrix to another matrix.

USAGE

The sending matrix A and the receiving matrix C are stored as i by j matrices in m by n arrays. Associated with each matrix is a dimension vector of four integers (i, j, m, n).

The calling sequence to MMOV is:

CALL MMOV (A, IA, C, IC, IS, JS, IR, JR, I, J, IND) to move an I by J submatrix whose upper left element is $A_{IS,JS}$ into an area whose upper left element is $C_{IR,JR}$.

IND is an error indicator whose value is 0 for normal execution and 1 if this call would have stored any elements beyond $C_{m,n}$.

BESSEL FUNCTIONS SUBROUTINE (BESSL)†

PURPOSE

The Bessel Functions subroutine (BESSL) calculates the two Bessel Functions of the first kind quickly and accurately. This method is particularly adapted to the problem of calculating more than one order for a given X. For p = NMIN, NMIN+1,...,NMAX, where NMIN and NMAX are zero or positive integers, and for $X > 0$, either Jp(X) or Ip(X) is calculated depending on a parameter ITYPE.

METHOD

The method used by BESSL is discussed in an article by Irene Stegun and Milton Abramowitz.* It consists of three steps for Jp which are:

A. k is chosen the larger of 1.5X and NMAX,
then k = k+10 for sufficient accuracy,

then $\bar{J}_{k+2} = 0$

$J_{k+1} = \alpha$, an arbitrarily small constant which is 0.1000E-10 in this program.

B. The recursion formula

$$\bar{J}_p = \frac{2(p+1)}{X} \bar{J}_{p+1} - \bar{J}_{p+2}$$

is used to generate $\bar{J}_p$ for p = k down to p = 0.

C. The results can be normalized, since

$$J_o + 2 \sum_{m=1}^{\infty} J_{2m} = 1,$$

therefore a constant $c = \bar{J}_o + 2 \sum_{m=1}^{k/2} \bar{J}_{2m}$

is determined and then

$J_p = \bar{J}_p / c$ for p = NMIN, NMIN+1,...,NMAX

The procedure for Ip is essentially the same except that in step B the recursion formula is:

$$\bar{I}_p = \frac{2(p+1)}{X} \bar{I}_{p+1} + \bar{I}_{p+2}$$

†Portions of this routine have been reprinted from C. B. Chandler's "BESSL - Bessel Functions Subroutine," TIS No. 64TIP5, issued by the Telecommunications & Information Processing Department of the General Electric Company at Schenectady, New York.

*Stegun, Irene A., and Abramowitz, Milton, "Generation of Bessel Functions on High Speed Computers," Mathematics and Other Aids to Computation, 1957, 11:255-257.

and in step C the normalization is due to:

$$Io + 2 \sum_{m=1}^{\infty} Im = e^{x}$$

and therefore the constant $c = (\overline{I}o+2 \sum_{m=1}^{k} \overline{I}m)/e^{x}$.

USAGE

The calling sequence for this routine is:

```
CALL BESSL (ITYPE,X,NMIN,NMAX,BESJI)
```

where ITYPE = 1 for Bessel Function Jp(X);
ITYPE = 2 for Modified Bessel Function Ip(X);
X = independent variable > 0;
NMIN, NMAX = 0, 1, 2,... giving range of orders of Jp(X) or Ip(X) desired;
BESJI( ) = a vector where answers are stored in increasing order. The maximum size of this vector is determined by the user.

RESTRICTIONS

The generated $\overline{J}p$ (or $\overline{I}p$) must fall within the limits 10^{-36} and 10^{+36}. If either $\geq 10^{+36}$ then ITYPE is set equal to zero and control is transferred to RETURN. If $\overline{J}p$ (or $\overline{I}p$) $\leq 10^{-36}$ then that term is set equal to zero and the program continues. The user can check on overflow by branching on ITYPE although normally overflow will not occur.

INTERPOLATION ROUTINE (INTP)

PURPOSE

Using a vector of X values in ascending order and a corresponding vector of Y values, the Interpolation routine (INTP) finds, by three-point interpolation, the value of Y for a given value of X.

METHOD

The routine first finds the first X equal to or greater than the given value (X_2). If there is no X meeting this requirement, exit is made with a dummy Y value of plus bits. If X_2 is one of the end points of the X vector, the next value is choosen as X_2. The preceding X is chosen as X_1 and the following as X_3. Assuming a curve of the form

$$Y = a + bx + cx^2$$

to pass through these three points, the unknown constants a, b, and c can be expressed in terms of X_1, Y_1, X_2, Y_2, X_3, Y_3.

Substituting these known values and the given value of X yields a value of Y.

USAGE

INTP is called by a sequence of the form:

CALL INTPL (X,Y,N,XVAL,YVAL)

where X and Y are the vectors required, each of dimension N.

XVAL is the given X value, and the interpolated value of Y will be stored by INTP into YVAL.

RESTRICTIONS

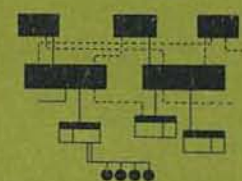
The vector of X values must be stored in ascending order.

GENERAL ELECTRIC
COMPUTERS

EIGENJ

GE-625/635 Math Routines

EIGENJ



GENERAL ELECTRIC

CPB-1166

GE-625/635
MATH ROUTINES
EIGENJ

A FORTRAN SUBROUTINE TO CALCULATE EIGEN SYSTEMS OF SYMMETRIC MATRICES

Program Number
CD600D4.009

October 1965

GENERAL ELECTRIC
COMPUTER DEPARTMENT

CREDITS

The source material used in this manual is taken from a document published by the General Electric Telecommunications and Processing Department, titled EIGENJ - A FORTRAN Subroutine to Calculate EIGEN Systems of Symmetric Matrices by H. W. Moore. Permission to use the original document was given by D. L. Shell, Manager, Computer Applications and Processing of the Telecommunications and Information Processing Department, General Electric Company.

Comments on this publication may be addressed to Technical Publications, Computer Department, General Electric Company, P. O. Box 2961, Phoenix, Arizona, 85002.

GENERAL ELECTRIC
INFORMATION SYSTEMS DIVISION
COMPUTER EQUIPMENT DEPARTMENT

GE-600 SERIES
TECHNICAL INFORMATION BULLETIN

DATE Nov. 1966
NO. 600-117
REF. CPB-1166

SUBJECT: Calling Sequence for the EIGENJ Math Routine, GE-625/635 Math Routines EIGENJ, CPB-1166

INSTRUCTIONS

Replace page 11 in the subject manual with the attached revised page 11.

EXPLANATION OF CHANGES

One of the arguments for the CALL EIGENJ statement was omitted.

TIB DISPOSITION

The revised page will appear in the next edition of GE-625/635 Math Routines EIGENJ, CPB-1166.

3. USAGE

CALL AND DIMENSION STATEMENTS

Normally, the subroutine is called by CALL EIGENJ (A, M, N, V, IV, E) where

A is a one-dimensional array containing the elements of the symmetric matrix to be placed in columns.

N is the order of the matrix A (that is, N = the number of rows)

M is the maximum order of matrix to be solved

V is the NxN matrix of eigenvectors

IV is an indicator; if IV = 0, the vectors will not be included. If IV equals 1, the routine will calculate all of the vectors.

Only the upper triangle of elements is to be stored. The elements are to be stored in the natural order--each row I begins with element X_{11}. For example, the matrix

$$
\begin{matrix}
X_{11} & X_{12} & X_{13} & X_{14} \\
X_{21} & X_{22} & X_{23} & X_{24} \\
X_{31} & X_{32} & X_{33} & X_{34} \\
X_{41} & X_{42} & X_{43} & X_{44}
\end{matrix}
$$

is set up as follows

$$
\begin{matrix}
A_1 & A_2 & A_3 & A_4 & A_5 & A_6 & A_7 & A_8 & A_9 & A_{10} \\
X_{11} & X_{12} & X_{13} & X_{14} & X_{22} & X_{23} & X_{24} & X_{33} & X_{34} & X_{44}
\end{matrix}
$$

Therefore, the length of the array must be N(N + 1)/2.

The vectors are stored in columns in the same order as the values are stored.

The subroutine stores the eigenvalues in the E array. To conserve space, the programmer is free to overlay the matrix A with the eigenvalues (that is, use A instead of E).

The foregoing discussion shows that the dimension statement should be

DIMENSION A(K), V(M, M), E(M)
where K = M(M + 1)/2 and M is the maximum order to be solved.

As explained in Chapter 2, the statement MAX = 6 (which is two statements above statement 31 in the listing) controls the accuracy of the calculation. For double precision compilation, this must be changed to MAX = 7. For less accuracy with a moderate increase in speed, MAX can be set equal to 5. If the value of M = 5 is used, the vectors may not be very good, although this does not affect the eigenvalues as much.

EXTENSIONS AND MODIFICATIONS

The discussion of the Jacobi method for Symmetric Matrices in Chapter 3 included a proof that the matrix T in equation (17), $T=O_1 O_2 \ldots . O_R$, was the matrix of eigenvectors. The Programming Considerations discussion noted that the program computed T simultaneously with the development of eigenvalues, starting with the identity matrix and successively multiplying on the right by the orthogonal matrix O_i, as shown in equation (48).

Obviously,

$$T = O_1 O_2 \ldots\ldots O_K I \qquad (49)$$

yields exactly the same results. However, equation (49) implies that T is calculated by starting with the Identity matrix and multiplying by the O_i's on the left using them in the reverse order; that is, multiplying by the last O_K, then by the next to last, and so forth, until O_1 is used. This latter approach has two significant advantages:

1. This approach develops the matrix T by columns, and the Ith column is the eigenvector corresponding to the Ith eigenvalue. Therefore, if only selected vectors are wanted (like those corresponding to the five largest eigenvalues), they can be developed independently, since there is no need to calculate any of the remaining columns. The work thus saved can be considerable. To do this, the program must save the O_i's and use them in reverse order; it only uses the columns of the Identity matrix corresponding to the desired eigenvalues. For each O_i, only four numbers need to be saved (namely, I_i, J_i, $\text{Cos}O_i$, and $\text{Sin}O_i$). These can be stored on tape or DSU or punched on cards.

2. This approach can allow the solution of a much larger matrix, even though all of the vectors are needed. Since only half of the vectors must be in memory at a given time, they can overlay the area originally occupied by the matrix A.

EIGENP PROGRAM

EIGENJ is also available as a free-standing program package called EIGENP. Input to EIGENP is as follows:

One card containing identification alphanumerics

A second card containing further identification

$SIZE/N= ,IV $

where N and IV are as above

$DATA/I= ,J= ,LL/BUF= , , , , $ (basic input card.)

CONTENTS

1. GENERAL DESCRIPTION

EIGENJ is a FORTRAN subroutine which calculates the eigenvalues and eigenvectors of a real symmetric matrix. It uses a modified Threshold Jacobi Method that does not require a square root and stores only the upper triangle of coefficients. An option to omit the calculation of the vectors is provided.

Chapter 3 on Usage, and the Programming Considerations discussed in Chapter 2 provide information on the modifications of the subroutine to provide more efficient calculation when only selected vectors are needed (such as those corresponding to the 10 largest or 10 smallest eigenvalues).

2. MATHEMATICAL METHOD

GENERAL INFORMATION ON EIGEN SYSTEMS

The latent roots or eigenvalues of a square matrix A are defined as the values of λ for which the set of homogeneous equations

$$A\bar{X} = \lambda\bar{X} \tag{1}$$

has a nonzero solution $\bar{X}$. Equation (1) may be written

$$(A - \lambda I)\ \bar{X} = 0 \tag{2}$$

and in this form it is clear that for a nonzero $\bar{X}$ the determinant must be zero; that is,

$$A - \lambda I = 0 \tag{3}$$

Explicit expansion gives the algebraic equation

$$a + a_1\ \lambda + \ldots\ a_{n-1}\lambda^{n-1} + (-1)^n\lambda^n = 0 \tag{4}$$

which clearly shows that any square matrix A with real or complex coefficients has n eigenvalues. Equation (4) is called the characteristic equation of the matrix A and is always of degree n for a matrix of nth order.

If all the roots of the characteristic equation are distinct, then the following results can be proved:

(a) If the roots are denoted by $\lambda_1 \lambda_2 \ldots \lambda_n$ then for each equation i

$$(A - X_i I)\bar{X}_i = \bar{0} \tag{5}$$

has only one independent solution, determined apart from an arbitrary multiplier.

(b) The vectors $\bar{X}_i$ form an independent set, so that an arbitrary vector may be expressed as a linear combination of the X_i.

(c) A matrix T exists such that

$$T^{-1}\ AT = \begin{bmatrix} \lambda_1 & & & & 0 \\ & \lambda_2 & & & \\ & & \cdot & & \\ & & & \cdot & \\ 0 & & & \cdot & \lambda_n \end{bmatrix} \tag{6}$$

The calculation of the eigenvectors for a characteristic equation with repeated roots is more complicated and is not discussed here. However, these properties can be shown to hold for symmetric matrices. Symmetric matrices recur frequently and their eigen systems have important special properties. The most fundamental property of symmetric matrices is that all their elementary divisors are linear. This means that although the matrix may have coincident roots there is never any deficiency in the number of independent eigenvectors. There is always a matrix T so that (6) applies even when some of the roots are multiple roots. In particular, the matrix T can be chosen so that

$$T^{-1} = T' \text{ (the transpose of T)} \tag{7}$$

which implies $TT' = I = T'T$.
Such a matrix is called an orthogonal matrix. The eigenvectors corresponding to different values of λ are orthogonal.

JACOBI'S METHOD FOR SYMMETRIC MATRICES

If T is any nonsingular matrix, then the matrix

$$B = T^{-1} AT \tag{8}$$

has the same eigenvalues as A. For if

$$A\bar{X} = \lambda\bar{X} \tag{9}$$

$$T^{-1} A\bar{X} = \lambda T^{-1}\bar{X} \tag{10}$$

giving

$$T^{-1} A (TT^{-1}) \bar{X} = \lambda T^{-1}\bar{X} \tag{11}$$

or

$$(T^{-1}AT)\ T^{-1}\bar{X} = \lambda (T^{-1}\bar{X}) \tag{12}$$

λ is, therefore, an eigenvalue of $T^{-1}AT$, and the corresponding eigenvector is $T^{-1}\bar{X}$. If the matrix T is orthogonal,

$$TT' = I \tag{13}$$

Therefore,

$$T' = T^{-1} \tag{14}$$

The product of two orthogonal matrices O_1 and O_2 is itself orthogonal.
For

$$(O_1 O_2)(O_1 O_2)' = O_1 O_2 O_2' O_1'$$

$$= O_1 (O_2 O_2') O_1' = O_1 O_1' = 1 \tag{15}$$

Jacobi's method depends on the choice of a sequence of simple orthogonal matrices O_r such that

$$A_r = O_r' \ldots O_2' O_1' A O_1 O_2 \ldots O_r = \text{diagonal matrix} \tag{16}$$

The roots of each of the A_r are the same as the roots of A, so that, when the process has been continued until A_r is effectively diagonal (that is, until all of the off diagonal elements are zero to machine accuracy), the diagonal elements are the eigenvalues of A.

$$T = O_1 O_2 \ldots O_r \tag{17}$$

$$T'AT = \begin{bmatrix} \lambda_1 & & & & \\ & \lambda_2 & & & \\ & & \cdot & & \\ & & & \cdot & \\ & & & & \lambda_n \end{bmatrix} \tag{18}$$

giving

$$TT'AT = T \begin{bmatrix} \lambda_1 & & & \\ & \lambda_2 & & \\ & & \ddots & \\ & & & \lambda_n \end{bmatrix} \tag{19}$$

or

$$AT = T \begin{bmatrix} \lambda_1 & & & \\ & \lambda_2 & & \\ & & \ddots & \\ & & & \lambda_n \end{bmatrix} \tag{20}$$

Hence, the columns of T are the eigenvectors of A
Each of the orthogonal matrices O's is chosen to be a simple rotation in the (i, j) plane and $i \neq j$.
They take the form

$$O_s = \begin{array}{c} \\ \\ \\ i \\ \\ \\ \\ j \\ \\ \\ \end{array} \begin{bmatrix} 1 & & & i & & & & j & & \\ & 1 & & & & & & & & \\ & & 1 & & & & & & & \\ & & & C & & & & -S & & \\ & & & & 1 & & & & & \\ & & & & & 1 & & & & \\ & & & & & & 1 & & & \\ & & & S & & & & C & & \\ & & & & & & & & 1 & \\ & & & & & & & & & 1 \end{bmatrix} \tag{21}$$

where $C = \text{Cos } \theta$ and
$S = \text{Sin } \theta$
This gives $A_s = O_s' \, A_{s-1} O_s$.

A_s may be constructed from A_{s-1} in two stages: (1) multiplying on the right by O_s and (2) multiplying on the left by O_s'. In the first stage, only the ith and jth row can be altered. The (i, i) (i, j) (j, i) and (j, j) elements are altered by both transformations.

Denoting the (p, q) element of the matrix A_s by $a_{pq}^{(s)}$ gives the relations

$$a_{pi}^{(s)} = a_{pi}^{(s-1)} \cos\theta + a_{pj}^{(s-1)} \sin\theta \quad p \neq i, j \tag{22}$$

$$a_{pj}^{(s)} = -a_{pi}^{(s-1)} \sin\theta + a_{pj}^{(s-1)} \cos\theta \quad p \neq i, j \tag{23}$$

$$a_{ip}^{(s)} = a_{ip}^{(s-1)} \cos\theta + a_{jp}^{(s-1)} \sin\theta \quad p \neq i, j \tag{24}$$

$$a_{jp}^{(s)} = -a_{ip}^{(s-1)} \sin\theta + a_{jp}^{(s-1)} \cos\theta \quad p \neq i, j \tag{25}$$

$$a_{ii}^{(s)} = a_{ii}^{(s-1)} \cos^2\theta + 2a_{ij}^{(s-1)} \cos\theta \sin\theta + a_{jj}^{(s-1)} \sin^2\theta \tag{26}$$

$$a_{ij}^{(s)} = a_{ji}^{(s)} = -a_{ii}^{(s-1)} \sin\theta\cos\theta - a_{ij}^{(s-1)} \sin^2\theta + a_{ij}^{(s-1)} \cos^2\theta + a_{jj}^{(s-1)} \sin\theta\cos\theta \tag{27}$$

$$a_{jj}^{(s)} = a_{ii}^{(s-1)} \sin^2\theta - 2a_{ij}^{(s-1)} \sin\theta\cos\theta + a_{jj}^{(s-1)} \cos^2\theta \tag{28}$$

From (26) and (28)

$$a_{ii}^{(s)} + a_{jj}^{(s)} = a_{ii}^{(s-1)} + a_{jj}^{(s-1)} \tag{29}$$

This relation is to be expected since the transformation leaves the roots, and, therefore, the spur (trace), unchanged. The matrix also remains symmetric since

$$O_s' A_{s-1} O_s = O_s' A_{s-1}' O_s = O_s' A_{s-1} O_s \tag{30}$$

Equation (27) gives

$$a_{ij}^{(s)} = a_{ij}^{(s-1)} (\cos^2\theta - \sin^2\theta) + \sin\theta\cos\theta\,(a_{jj}^{(s-1)} - a_{ii}^{(s-1)}) \tag{31}$$

If θ is chosen so that

$$\tan 2\theta = \frac{2a_{ij}^{(s-1)}}{a_{ii}^{(s-1)} - a_{jj}^{(s-1)}} \tag{32}$$

then $a_{ij}^{(s)} = 0$. Squaring (22) and (23) and adding gives

$$\left[a_{pi}^{(s)}\right]^2 + \left[a_{pj}^{(s)}\right]^2 = \left[a_{pi}^{(s-1)}\right]^2 + \left[a_{pj}^{(s-1)}\right]^2 \quad p \neq i, j \tag{33}$$

Squaring (24) and (25) and adding gives

$$\left[a_{ip}^{(s)}\right]^2 + \left[a_{jp}^{(s)}\right]^2 = \left[a_{ip}^{(s-1)}\right]^2 + \left[a_{jp}^{(s-1)}\right]^2 \quad p \neq i, j \tag{34}$$

The sum of squares of nondiagonal terms excluding the (i, j) term has, therefore, remained constant while the (i, j) term will have become zero if θ is chosen to satisfy (32). Jacobi's method is based on the choice of a sequence of rotations, each of which is chosen to make an off-diagonal element equal to zero. The element of the current matrix which has the largest absolute value should be chosen as the (i, j) term.

SEQUENTIAL AND THRESHOLD MODIFICATIONS

Since a computer takes considerable time to search for the largest current a_{ij}, EIGENJ uses the Sequential Jacobi Method. This method utilizes off diagonal elements in the natural sequence to make a_{ij} equal to zero. The order is (1, 2) (1, 3) . . . (1, n) (2, 3) (2, 4) . . . (2n) . . . (n-1, n); a return to (1, 2); and another sweep. The subroutine also uses a further improvement because of the following condition: Suppose that at an early stage of the iteration an a_{ij} to be made zero is already small. Making this zero has little value, since the sum of the squares of the off diagonal terms will not be decreased appreciably. Therefore, assuming that at least 6 sweeps will be necessary, rotation may be omitted as follows: During the r th sweep, a rotation is omitted if $|a_{ij}| \le E_r$. The set of E_r will be a decreasing set such that $E_r = 0$ (to 9 significant figures) for $r > 6$, so that after the 6th sweep no "nonzero" locations are skipped.

An appropriate choice is

$$
\begin{aligned}
E_1 &= 2^{-2}a \\
E_2 &= 2^{-3}a \\
E_3 &= 2^{-5}a \\
E_4 &= 2^{-9}a \\
E_5 &= 2^{-17}a \\
E_6 &= 2^{-33}a
\end{aligned}
\qquad (35)
$$

where a is the largest off diagonal element of the original matrix. This is known as the Threshold Jacobi Method.

Equation (32) gives

$$\text{Tan } 2\theta = \frac{a}{b} = \frac{a'}{|b|} \qquad (36)$$

where $a' = \pm$ a according to the sign of b. The following is written for convenience:

$a' = p$ and $|b| = q$ so that q is positive.

Then

$$\text{Sec}^2 2\theta = 1 + p^2/q^2 \qquad (37)$$

$$\text{Cos}^2 2\theta = \frac{q^2}{p^2 + q^2} \qquad (38)$$

$$\text{Cos } 2\theta = \frac{q}{\sqrt{p^2 + q^2}} \qquad (39)$$

$$2 \text{ Cos}^2 \theta = 1 + \frac{q}{\sqrt{p^2 + p^2}} \qquad (40)$$

$$\text{Cos } \theta + \sqrt{\frac{1}{2}\left(1 + \frac{q}{\sqrt{p^2 + q^2}}\right)} \qquad (41)$$

and using

$$\text{Sin}^2 2\theta = \frac{p^2}{p^2 + q^2} \qquad (42)$$

$$\text{Sin } 2\theta = \frac{p}{\sqrt{p^2 + q^2}} \qquad (43)$$

$$\text{Sin } \theta = \frac{p}{2 \text{ Cos } \theta} \cdot \frac{1}{\sqrt{p^2 + q^2}} \qquad (44)$$

MODIFICATION TO ELIMINATE SQUARE ROOTS

Any plane rotations may be applied without altering the eigenvalues. There is no need to bring an off diagonal term exactly to zero at each stage. If the sin θ and the cos θ of the rotation are correctly related and the derived matrix is produced accurately, the derived matrix remains equivalent to the original matrix. In order to avoid computation as complex as that in (41) and (44), the angle used is computed from the formula

$$\tan \theta/2 \quad \begin{cases} = \dfrac{a_{ij}}{2\,(a_{ii} - a_{jj})} \\ \text{or} \\ = \text{Sign}\left\{a_{ij}/(a_{ii} - a_{jj})\right\} \tan \pi/8 \end{cases} \qquad (45)$$

whichever has the smallest absolute value. If the second case is chosen, $\theta = \pm \pi/4$, which is clearly the largest useful rotation, is used.

This scheme has the advantage that the functions

$$\text{Sin}\ \theta = \frac{2 \tan \theta/2}{1 + \tan^2 \theta/2} \qquad (46)$$

and

$$\text{Cos}\ \theta = \frac{1 - \tan^2 \theta/2}{1 + \tan^2 \theta/2} \qquad (47)$$

may be computed without extracting square roots.

REFERENCES FOR CONVERGENCE, ERROR ANALYSIS

1. Convergence is proved for this method by D. Pope and C. Tompkins, Maximizing Function of Rotation. J. Assoc. Comput. March 4 (1957) pp 459-466

2. An error analysis is given by J. H. Wilkinson, Error Analysis of Eigenvalue Techniques. J. S. I. A. M. March 1962, pp 162-195

PROGRAMMING CONSIDERATIONS

Characteristics

Successive derived matrices in the EIGENJ program are symmetric. This feature provides for an economical storage procedure in which only the upper triangles of the original matrix are stored and all subsequent matrices are overwritten in the same storage location. Bent lines such as those shown in the following diagram are considered instead of the rows and columns of the original matrix and its derivatives. Therefore, $n(n + 1)/2$ storage locations are sufficient for the eigenvalue calculations.

0	0	0	X	0	X	0	0
	0	0	X	0	X	0	0
		0	X	0	X	0	0
			X	X	X	X	X
				0	X	0	0
					X	X	X
						0	0
							0

The eigenvectors are the columns of the matrix $O_1 O_2 \ldots O_K = T$ where O_K is the last rotation performed (see Equation (20)). If λ_r is the rth diagonal element of A_K (the ultimate matrix resulting from the rotation) then the eigenvector corresponding to λ_r is the rth column of T. The vectors, when desired, are calculated by starting with the unit matrix I of order n and performing the rotation on the right; that is,

$$T = IO_1O_2 \ldots O_K \quad (48)$$

Therefore, the full n x n matrix for the eigenvector must be held in storage and is developed at the same time as the eigenvalue. The elements of the rotation matrices are discarded when used. (See Extensions and Modifications Chapter 3 for an alternate procedure.)

Calculation

The calculation procedure is as follows:

1. Initialize subroutine and set vector matrix equal to Identity. K is the index for the A(I, J) term.

2. Search for the largest off diagonal element and set threshold. Threshold = E_1 from the equation (35).

3. Compute for each successive off diagonal element greater than E_1

   Sin θ from equation (46)
   Cos θ from equation (47)
   a_{ii} from equation (26)
   a_{jj} from equation (28)
   a_{ij} from equation (31), $A(K) = a_{ij}$

4. Reset threshold to E_{i-1} and repeat step 3 until threshold becomes equal to minimum (E_6).

5. Repeat step 3 until every off diagonal is less than E_6.

6. Copy eigenvalues into eigenvalue array.

7. Return

The logic for finding the largest off diagonal element in step 2 and the logic for finding successive elements greater than E_i in step 3 is the same. Therefore, the search is programmed only once and a switch (GO TO (7, 8), IND) on IND is used to proceed to the resetting of AMAX or to the calculation.

The option of bypassing the vector calculation is provided by the indicator IV. The setting of the vector matrix to the Identity in steps 1 and 3 is bypassed if IV = 0.

If the input matrix is already diagonal (if off diagonals all equal zero) statement 29 in the listing provides a quick return with the correct answer.

In the subroutine the variable MAX is set equal to 6. This value controls the size of the minimum threshold, so that it is equal to 2^{-33} (1.16×10^{-11}) times the largest off diagonal element of the original matrix. If less accuracy is needed (say only 4 significant figures in the eigenvalues) and some speedup is wanted, the programmer can set MAX = 5. This will make the minimum threshold equal to 2^{-17} (or 7.6×10^{-6}) times the largest off diagonal elements. CAUTION: The eigenvectors will be considerably less accurate than the eigenvalues since they are much more sensitive. If this routine is to be compiled in the Double Precision Mode the programmer should set MAX = 7 which will make the minimum threshold equal to 2^{-65} (or 2.7×10^{-23}) times the largest off diagonal element.

3. USAGE

CALL AND DIMENSION STATEMENTS

Normally, the subroutine is called by CALL EIGENJ (A, N, V, IV, E) where

A is a one-dimensional array containing the elements of the symmetric matrix to be placed in columns.

N is the order of the matrix A (that is, N = the number of rows)

V is the N xN matrix of eigenvectors

IV is an indicator; if IV = 0, the vectors will not be included. If IV equals 1, the routine will calculate all of the vectors.

Only the upper triangle of elements is to be stored. The elements are to be stored in the natural order--each row I begins with element X_{II}. For example, the matrix

$$\begin{matrix} X_{11} & X_{12} & X_{13} & X_{14} \\ X_{21} & X_{22} & X_{23} & X_{24} \\ X_{31} & X_{32} & X_{33} & X_{34} \\ X_{41} & X_{42} & X_{43} & X_{44} \end{matrix}$$

is set up as follows

$$\begin{matrix} A_1 & A_2 & A_3 & A_4 & A_5 & A_6 & A_7 & A_8 & A_9 & A_{10} \\ X_{11} & X_{12} & X_{13} & X_{14} & X_{22} & X_{23} & X_{24} & X_{33} & X_{34} & X_{44} \end{matrix}$$

Therefore, the length of the array must be N(N + 1)/2.

The vectors are stored in columns in the same order as the values are stored.

The subroutine stores the eigenvalues in the E array. To conserve space, the programmer is free to overlay the matrix A with the eigenvalues (that is, use A instead of E).

The foregoing discussion shows that the dimension statement should be

DIMENSION A(K), V(M, M), E(M)
where K = M(M + 1)/2 and M is the maximum order to be solved.

As explained in Chapter 2, the statement MAX = 6 (which is two statements above statement 31 in the listing) controls the accuracy of the calculation. For double precision compilation, this must be changed to MAX = 7. For less accuracy with a moderate increase in speed, MAX can be set equal to 5. If the value of M = 5 is used, the vectors may not be very good, although this does not affect the eigenvalues as much.

EXTENSIONS AND MODIFICATIONS

The discussion of the Jacobi method for Symmetric Matrices in Chapter 3 included a proof that the matrix T in equation (17), $T=O_1 O_2 \ldots\ldots O_R$, was the matrix of eigenvectors. The Programming Considerations discussion noted that the program computed T simultaneously with the development of eigenvalues, starting with the identity matrix and successively multiplying on the right by the orthogonal matrix O_i, as shown in equation (48).

Obviously,

$$T = O_1 O_2 \ldots\ldots O_K I \qquad (49)$$

yields exactly the same results. However, equation (49) implies that T is calculated by starting with the Identity matrix and multiplying by the O_i's on the left using them in the reverse order; that is, multiplying by the last O_K, then by the next to last, and so forth, until O_1 is used. This latter approach has two significant advantages:

1. This approach develops the matrix T by columns, and the Ith column is the eigenvector corresponding to the Ith eigenvalue. Therefore, if only selected vectors are wanted (like those corresponding to the five largest eigenvalues), they can be developed independently, since there is no need to calculate any of the remaining columns. The work thus saved can be considerable. To do this, the program must save the O_i's and use them in reverse order; it only uses the columns of the Identity matrix corresponding to the desired eigenvalues. For each O_i, only four numbers need to be saved (namely, I_i, J_i, $\mathrm{Cos}O_i$, and $\mathrm{Sin}O_i$). These can be stored on tape or DSU or punched on cards.

2. This approach can allow the solution of a much larger matrix, even though all of the vectors are needed. Since only half of the vectors must be in memory at a given time, they can overlay the area originally occupied by the matrix A.

EIGENP PROGRAM

EIGENJ is also available as a free-standing program package called EIGENP. Input to EIGENP is as follows:

One card containing identification alphanumerics

A second card containing further identification

$SIZE/N= , IV $

where N and IV are as above

$DATA/I= , J= , LL/BUF= , , , , $ (basic input card.)

I and J are row and column numbers, and the remaining blanks are elements of the input matrix. Elements not read in are set to zero by the program. The packing into the triangular form mentioned above is done by the program. (Note that J is never read in less than I; that is, only the upper elements are entered.)

The end of a case is signaled by a $DATA card with I=0.

Further cases may be read in, starting with the two cards of identification. The end of run is signaled by a $SIZE card with N=0.

Several test matrices of various orders through 25 have been run. These have been checked with the results of the other subroutines and with analytical solutions. Those with analytical solutions checked to 8 significant figures.

Input for one case is illustrated on the following pages. Additional input sheets are furnished at the back of this manual. A listing of input cards and an output listing are also included in this section.

Input Coding Form (With Sample Data)

EIGENP INPUT

Col
2 TEST CASE FOR EIGENP

ID CARD 1

ID CARD 2

$SIZE/N= 6, IV= T $

$DATA/I= 1, J= 1, LL/BUF= .51179022, .061142881, .15690355$, ______,

______, ______$

$DATA/I= 2, J= 2, LL/BUF= .43923729, -.11061734, 0.0, -.051441047,

-.0723857223$, ______$

$DATA/I= 3, J= 3, LL/BUF= 1.1485492, 0.0, -.072385722, -.1018582$,

______, ______$

$DATA/I= 4, J= 4, LL/BUF= .51179022, -.061142881, -.15690355$, ______,

______, ______$

$DATA/I= 5, J= 5, LL/BUF= .47923729, -.11061734$, ______, ______,

______, ______$

$DATA/I= 6, J= 6, LL/BUF= 1.1485492$, ______, ______, ______,

______, ______$

$DATA/I= 0$, J=___, LL/BUF=______, ______, ______, ______,

______, ______$

$DATA/I=___, J=___, LL/BUF=______, ______, ______, ______,

______, ______$

$DATA/I=___, J=___, LL/BUF=______, ______, ______, ______,

______, ______$

$DATA/I=___, J=___, LL/BUF=______, ______, ______, ______,

______, ______$

$DATA/I=___, J=___, LL/BUF=______, ______, ______, ______,

______, ______$

Note to Keypunch: Discontinue after written $
First two cards must be included, even if blank.

Col
2 TEST CASE FOR EIGENP (CONT.)
____
ID CARD 1

____
ID CARD 2

$SIZE/N= 01, IV=___$

$DATA/I=___, J=___, LL/BUF=________, ________, ________, ________,
________, ________$

$DATA/I=___, J=___, LL/BUF=________, ________, ________, ________,
________, ________$

$DATA/I=___, J=___, LL/BUF=________, ________, ________, ________,
________, ________$

$DATA/I=___, J=___, LL/BUF=________, ________, ________, ________,
________, ________$

$DATA/I=___, J=___, LL/BUF=________, ________, ________, ________,
________, ________$

$DATA/I=___, J=___, LL/BUF=________, ________, ________, ________,
________, ________$

$DATA/I=___, J=___, LL/BUF=________, ________, ________, ________,
________, ________$

$DATA/I=___, J=___, LL/BUF=________, ________, ________, ________,
________, ________$

$DATA/I=___, J=___, LL/BUF=________, ________, ________, ________,
________, ________$

$DATA/I=___, J=___, LL/BUF=________, ________, ________, ________,
________, ________$

$DATA/I=___, J=___, LL/BUF=________, ________, ________, ________,
________, ________$

Note to Keypunch: Discontinue after written $
First two cards must be included, even if blank.

Listing of Input Cards

```
TEST CASE FOR EIGEN
     SEE MANUAL
$SIZE/N=6,IV=T$
$DATA/I=1,J=1,LL/BUF=.51179022,.061142881,.15690355$
$DATA/I=2,J=2,LL/BUF=.43923729,-.11061734,0.0,-.051441047,-.072385722$
$DATA/I=3,J=3,LL/BUF=1.1485492,0.0,-.072385722,-.1018582$
$DATA/I=4,J=4,LL/BUF=.51179022,-.061142881,-.15690355$
$DATA/I=5,J=5,LL/BUF=.43923729,-.11061734$
$DATA/I=6,J=6,LL/BUF=1.1485492$
$DATA/I=0$
$SIZE/N=0$
$      ENDJOB
$      EXECUTE
$      INCODE  IBMF
```

Output Listing

```
 EIGENVALUES AND VECTORS
TEST CASE FOR EIGEN
     SEE MANUAL

     I         LAMBDA(I)
     1         5.5254756 E-01
     2         2.91163919E-01
     3         1.28306712E 00
     4         5.29959224E-01
     5         4.17160765E-01
     6         1.12513359E 00
```

I	J	E(I,J)	E(I,J+1)	E(I,J+2)	E(I,J+3)	E(I,J+4)	E(I,J+5)
1	1	4.75609706E-01	-3.14937904E-01	1.39143965E-01	-6.13421415E-01	5.04980020E-01	1.56614834E-01
2	1	5.12694963E-01	5.97235583E-01	-2.26922745E-02	-3.45902652E-01	-4.50905261E-01	-1.53840298E-01
3	1	-7.84265725E-02	2.10070712E-01	6.92909613E-01	6.37590555E-02	-1.17154934E-01	6.72164463E-01
4	1	4.75609356E-01	3.14937778E-01	1.39144052E-01	6.13421604E-01	5.04979976E-01	-1.56614818E-01
5	1	-5.12694924E-01	5.97250546E-01	2.26925132E-02	-3.45902916E-01	4.50905291E-01	-1.53840292E-01
6	1	7.84265231E-02	2.10070709E-01	-6.92909975E-01	6.37599185E-02	1.17154947E-01	6.72164232E-01

APPENDIX A
PROGRAM LISTING

```
$      IDENT    AAV,GE,FORTRAN,EIGENP
$      OPTION   FORTRAN,GO
$      FORTRAN LSTOU,DECK,STAB
$      INCODE   IBMF
*EIGENP                         EIGEN PROGRAM
*                               CD60004.009        DATE 05/05/65
      DIMENSION V(120,120),A(7260),E(120),NAME(12),ID(12),BUF(6)
      LOGICAL IV
      NAMELIST/SIZE/N,IV/DATA/I,J,BUF,LL
    1 WRITE(6,2)
    2 FORMAT(26H1  EIGENVALUES AND VECTORS)
      READ 3,NAME
      READ 3,ID
    3 FORMAT(12A6)
      WRITE(6,3)NAME
      WRITE(6,3)ID
      READ(5,SIZE)
      N23=N+N+3
      IF(N.EQ.0)CALL EXIT
      II=N*(N+1)/2
      DO 13 I=1,II
   13 A(I)=0.0
    5 READ(5,DATA)
      IF(I.EQ.0)GO TO 9
      K=(I*(N23-I))/2+J-N-I
      DO 8 L=1,LL
      A(K)=BUF(L)
    8 K=K+1
      GO TO 5
```

```
    9 CALL EIGENJ(A,120,N,V,IV,E)                                       EIGP0290
      WRITE(6,10)                                                       EIGP0300
   10 FORMAT(23H0       I          LAMBDA(I))                           EIGP0310
      WRITE(6,11)(I,E(I),I=1,N)                                         EIGP0320
   11 FORMAT(I7,1PE20.8)                                                EIGP0330
      IF(.NOT.IV)GO TO 1                                                EIGP0340
      WRITE(6,12)                                                       EIGP0350
   12 FORMAT(98H0 I    J   E(I,J)          E(I,J+1)         E(I,J+2)    EIGP0360
     1  E(I,J+3)          E(I,J+4)          E(I,J+5))                   EIGP0370
      CALL PMAT(V,120,120,N,N)                                          EIGP0380
      GO TO 1                                                           EIGP0390
      END                                                               EIGP0400
$      FORTRAN LSTOU,DECK,STAB                                          EIGJ0000
$      INCODE  IBMF                                                     EIGJ0010
*EIGENJ       EIGENVALUE AND EIGENVECTOR SUBROUTINE                     EIGJ0020
*                               CD600D4.009       DATE 05/05/65         EIGJ0030
      SUBROUTINE EIGENJ(A,MSIZE,NN,V,IV,E)                              EIGJ0040
      DIMENSION A(20100),V(MSIZE,MSIZE),E(MSIZE)                        EIGJ0050
      LOGICAL IV                                                        EIGJ0060
*     EIGENJ FINDS THE EIGENVALUES AND EIGENVECTORS OF                  EIGJ0070
*     A SYMMETRIC MATRIX (A) USING A MODIFIED THRESHOLD JACOBI METHOD   EIGJ0080
*     ONLY THE UPPER TRIANGLE IS STORED                                 EIGJ0090
*     DIMENSION A(K=M(M+1)/2),V(M,M),E(M)  WHERE M=MAXIMUM ORDER        EIGJ0100
*     DIMENSION A(K=M(M+1)/2),V(M,M),E(M)  WHERE M=MAXIMUM ORDER        EIGJ0110
      N=NN                                                              EIGJ0120
      IND=1                                                             EIGJ0130
      NM2=N-2                                                           EIGJ0140
      AMAX=0.0                                                          EIGJ0150
      NM=N-1                                                            EIGJ0160
```

```
      IF(.NOT.IV)GO TO 5
*     SET UP VECTOR MATRIX
    1 DO 3 I=1,N
      DO 2 J=1,N
    2 V(I,J)=0.0
    3 V(I,I)=1.0
    5 K=2
      DO 28 I=1,NM
      IP=I+1
      DO 27 J=IP,N
      Y=A(K)
      X=ABS(Y)
      IF(X-AMAX)27,27,6
    6 GO TO (7,8),IND
    7 AMAX=X
      GO TO 27
*     TRANSFORMATION SETUP FOLLOWS
    8 II=K+I-J
      JJ=(J*(N+N-J+3))/2-N
      ITEST=1
      X=A(II)-A(JJ)
      IF(X)10,11,10
   10 X=Y/X
      Y=.5*X
      IF(ABS(Y)-.414213562)14,11,11
   11 C=.707106781
      IF(Y)12,12,13
   12 S=-C
      GO TO 15
```

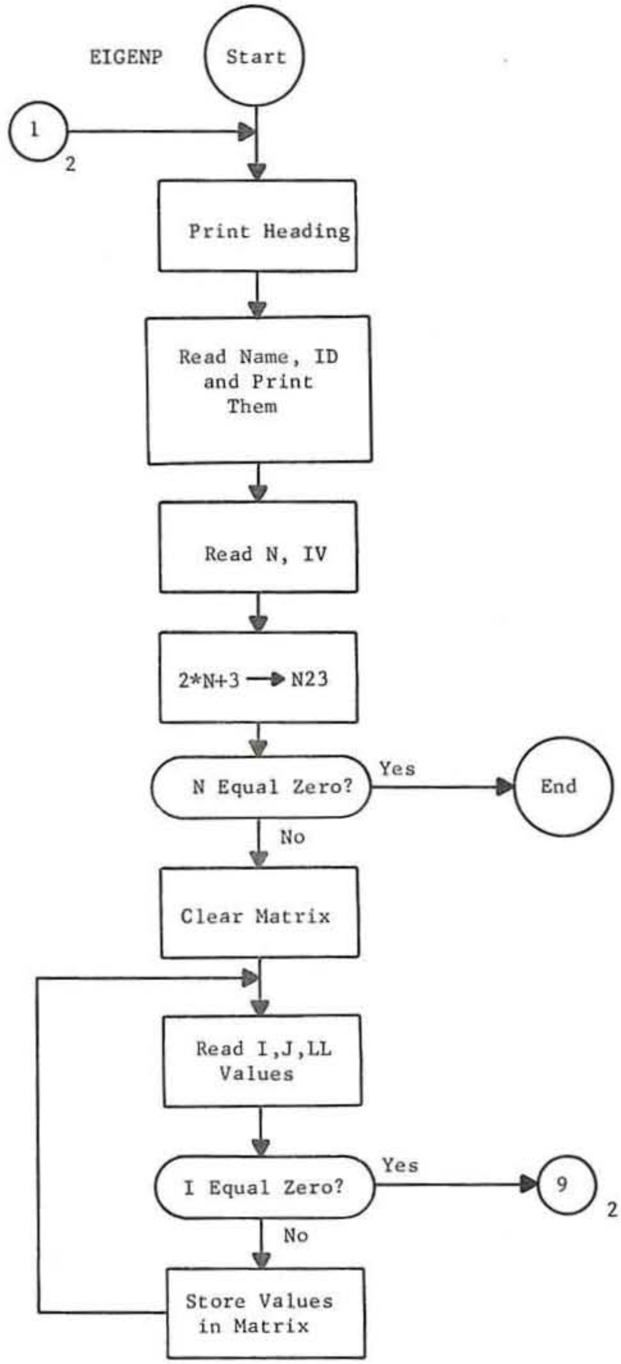
```
   13 S=C                                                EIGJ0460
      GO TO 15                                           EIGJ0470
   14 X=Y*Y                                              EIGJ0480
      C=1.+X                                             EIGJ0490
      S=2.*Y/C                                           EIGJ0500
      C=(1.-X)/C                                         EIGJ0510
   15 X=S*S                                              EIGJ0520
      Y=C*C                                              EIGJ0530
      XY=S*C                                             EIGJ0540
      AXY=2.*A(K)*XY                                     EIGJ0550
      Z=A(II)*Y+AXY+A(JJ)*X                              EIGJ0560
      W=A(II)*X-AXY+A(JJ)*Y                              EIGJ0570
      A(K)=A(K)*(Y-X)+XY*(A(JJ)-A(II))                   EIGJ0580
      A(II)=Z                                            EIGJ0590
      A(JJ)=W                                            EIGJ0600
      IF(NM2)24,24,16                                    EIGJ0610
   16 IT=I                                               EIGJ0620
      JT=J                                               EIGJ0630
*     TRANSFORM A                                        EIGJ0640
      DO 23 M=1,NM2                                      EIGJ0650
      NP=N-M                                             EIGJ0660
      IF(JT-K)20,17,18                                   EIGJ0670
   17 IT=IT+1                                            EIGJ0680
      JT=JT+NP                                           EIGJ0690
      ITEST=2                                            EIGJ0700
   18 IF(IT-K)20,19,19                                   EIGJ0710
   19 IT=IT+1                                            EIGJ0720
      JT=JT+1                                            EIGJ0730
      ITEST=3                                            EIGJ0740
```

```
20 X=A(IT)*C+A(JT)*S                                    EIGJ0750
   A(JT)=A(JT)*C-A(IT)*S                                EIGJ0760
   A(IT)=X                                              EIGJ0770
   GO TO (21,22,23),ITEST                               EIGJ0780
21 IT=IT+NP                                             EIGJ0790
   JT=JT+NP                                             EIGJ0800
   GO TO 23                                             EIGJ0810
22 IT=IT+1                                              EIGJ0820
   JT=JT+NP-1                                           EIGJ0830
23 CONTINUE                                             EIGJ0840
24 IF(.NOT.IV)GO TO 27                                  EIGJ0850
*     TRANSFORM V                                       EIGJ0860
25 DO 26 M=1,N                                          EIGJ0870
   X=V(M,I)*C+V(M,J)*S                                  EIGJ0880
   V(M,J)=V(M,J)*C-V(M,I)*S                             EIGJ0890
26 V(M,I)=X                                             EIGJ0900
27 K=K+1                                                EIGJ0910
28 K=K+1                                                EIGJ0920
*     SEQUENCE TO ADJUST THRESHOLD FOR EACH SWEEP       EIGJ0930
   GO TO (29,31),IND                                    EIGJ0940
29 IF(AMAX)30,35,30                                     EIGJ0950
30 AT=AMAX                                              EIGJ0960
   IND=2                                                EIGJ0970
   F=.25                                                EIGJ0980
   AMAX=F*AT                                            EIGJ0990
   MAX=6                                                EIGJ1000
   GO TO 5                                              EIGJ1010
31 MAX=MAX-1                                            EIGJ1020
   IF(MAX)34,33,32                                      EIGJ1030
```

```
   32 F=2.0*F*F                                                    EIGJ1040
      AMAX=AT*F                                                    EIGJ1050
   33 C=0.0                                                        EIGJ1060
      GO TO 5                                                      EIGJ1070
   34 IF(C)33,35,33                                                EIGJ1080
*     COPY EIGENVALUES INTO E                                      EIGJ1090
   35 MM=1                                                         EIGJ1100
      NM=N+1                                                       EIGJ1110
      DO 36 M=1,N                                                  EIGJ1120
      E(M)=A(MM)                                                   EIGJ1130
   36 MM=MM+NM-M                                                   EIGJ1140
      RETURN                                                       EIGJ1150
      END                                                          EIGJ1160
$       FORTRAN LSTOU,DECK                                         PMAT0000
$       INCODE  IBMF                                               PMAT0010
*PMAT                          SUBROUTINE TO PRINT MATRIX          PMAT0020
*                               CD600D4.009     DATE 05/04/65      PMAT0030
      SUBROUTINE PMAT(A,MM,NN,NR,NC)                               PMAT0040
      DIMENSION A(MM,NN),P(6)                                      PMAT0050
      NROW=NR                                                      PMAT0060
      NCOL=NC                                                      PMAT0070
      I=1                                                          PMAT0080
      J=1                                                          PMAT0090
    1 IP=I                                                         PMAT0100
      JP=J                                                         PMAT0110
      DO 2 K=1,6                                                   PMAT0120
      KK=K                                                         PMAT0130
      P(K)=A(I,J)                                                  PMAT0140
      J=J+1                                                        PMAT0150
```

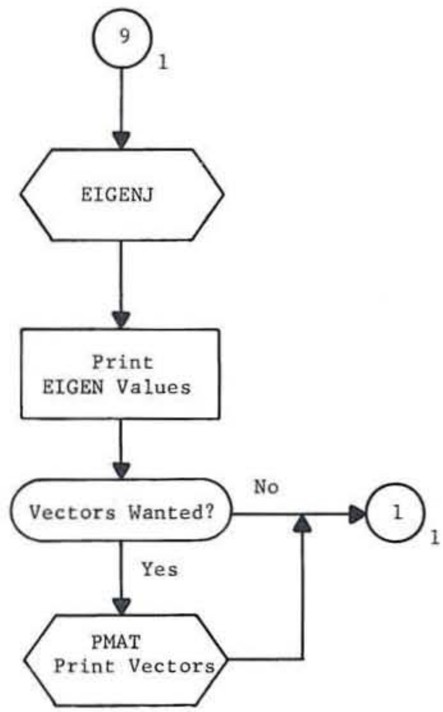
```
      IF(J.GT.NCOL)GO TO 3                                    PMAT0160
    2 CONTINUE                                                PMAT0170
      WRITE(6,4)IP,JP,(P(K),K=1,KK)                           PMAT0180
    4 FORMAT(2I4,6(1PE16.8))                                  PMAT0190
      GO TO 1                                                 PMAT0200
    3 WRITE(6,4)IP,JP,(P(K),K=1,KK)                           PMAT0210
      I=I+1                                                   PMAT0220
      J=1                                                     PMAT0230
      IF(I.LE.NROW)GO TO 1                                    PMAT0240
      RETURN                                                  PMAT0250
      END                                                     PMAT0260
```

APPENDIX B
FLOW CHARTS



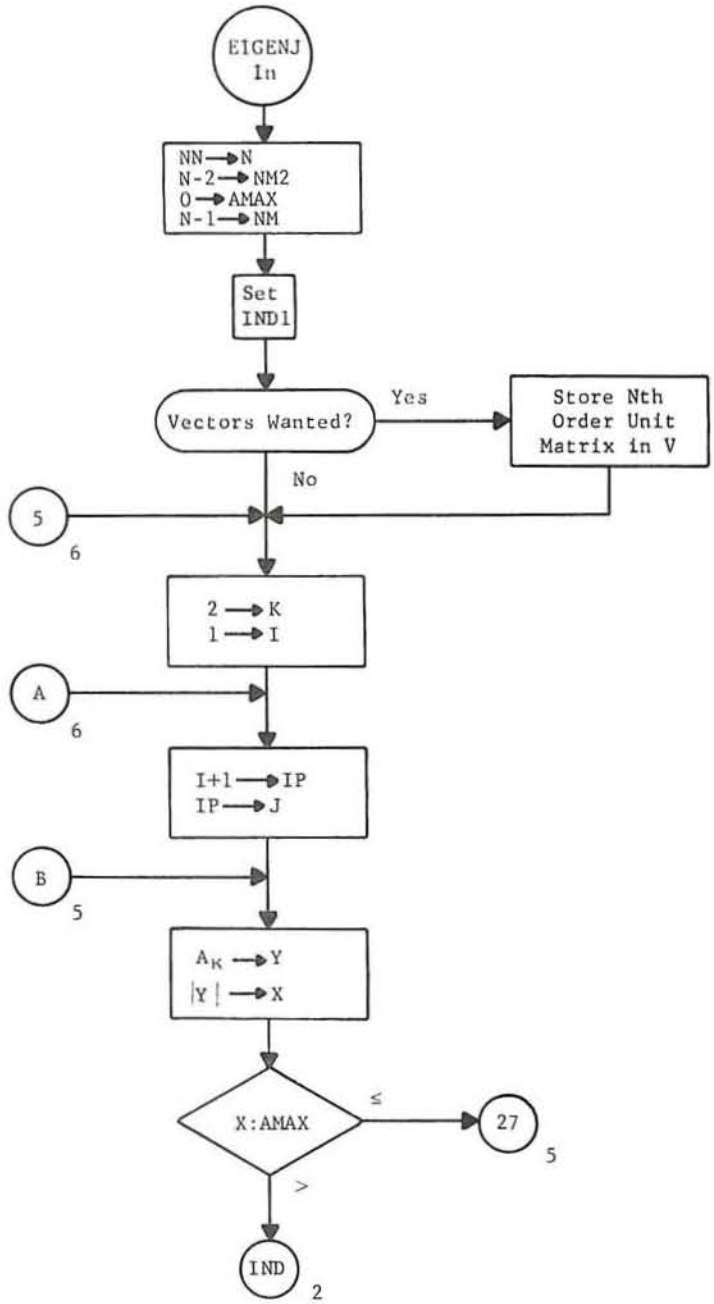
EIGENP

①


9
1
EIGENJ
Print
EIGEN Values
Vectors Wanted?
No
1
1
Yes
PMAT
Print Vectors

EIGENP

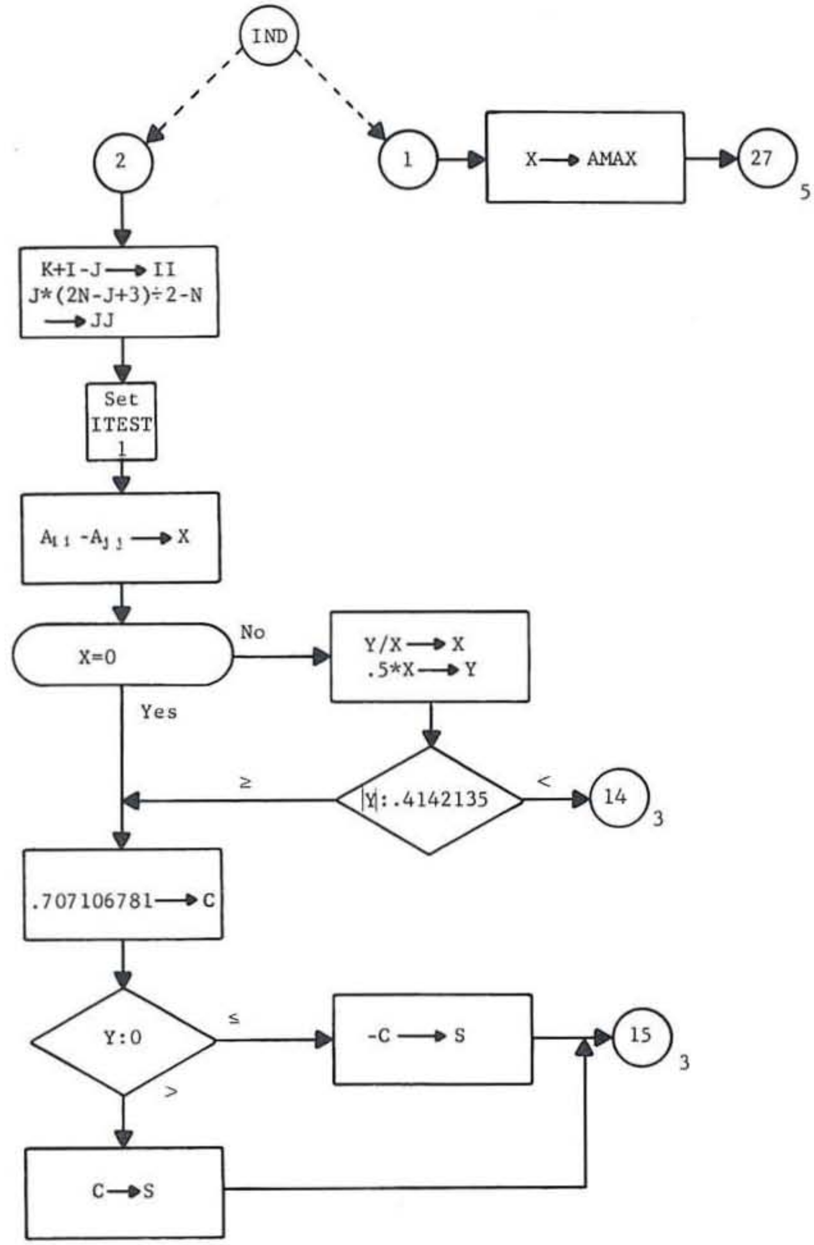
(2)



EIGENJ
In
NN → N
N-2 → NM2
0 → AMAX
N-1 → NM
Set
IND1
Vectors Wanted?
Yes
Store Nth
Order Unit
Matrix in V
No
5
6
2 → K
1 → I
A
6
I+1 → IP
IP → J
B
5
A_K → Y
|Y| → X
X:AMAX
≤
27
5
>
IND
2

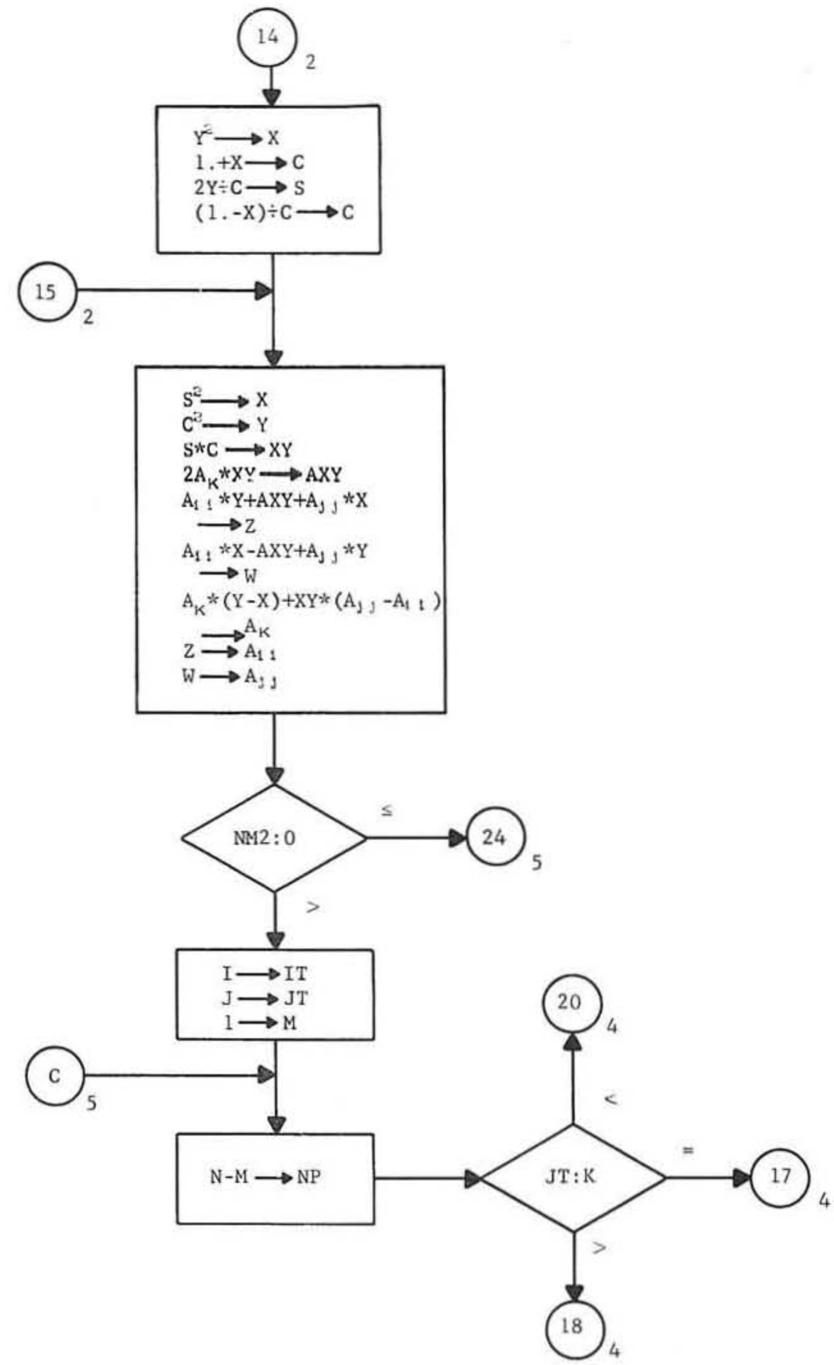
EIGENJ

①


IND
2
1
X → AMAX
27
5
K+I-J → II
J*(2N-J+3)÷2-N
→ JJ
Set
ITEST
1
A_II - A_JJ → X
X=0
No
Yes
Y/X → X
.5*X → Y
≥
|Y|:.4142135
<
14
3
.707106781 → C
Y:0
≤
>
-C → S
15
3
C → S

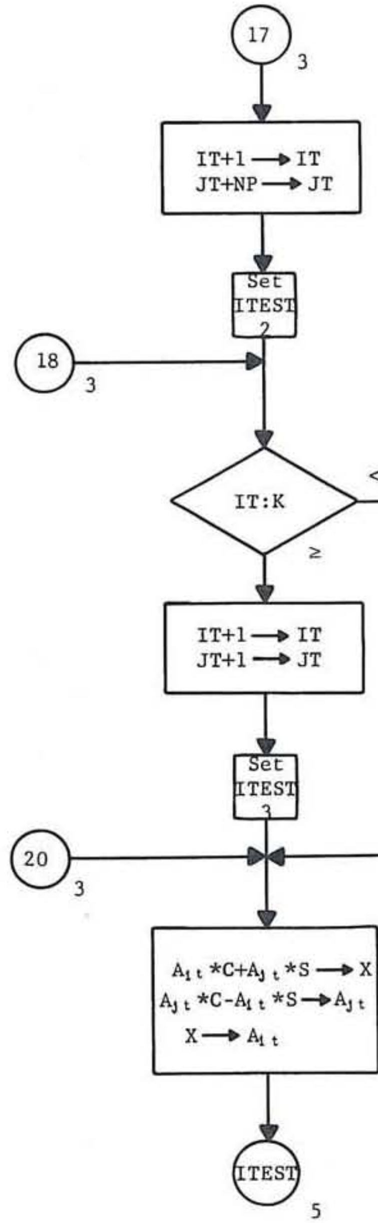
EIGENJ

(2)


14
2
Y² → X
1.+X → C
2Y÷C → S
(1.-X)÷C → C
15
2
S² → X
C² → Y
S*C → XY
2A_K*XY → AXY
A_II*Y+AXY+A_JJ*X → Z
A_II*X-AXY+A_JJ*Y → W
A_K*(Y-X)+XY*(A_JJ-A_II) → A_K
Z → A_II
W → A_JJ
NM2:0
≤
24
5
>
I → IT
J → JT
1 → M
C
5
N-M → NP
JT:K
<
20
4
=
17
4
>
18
4

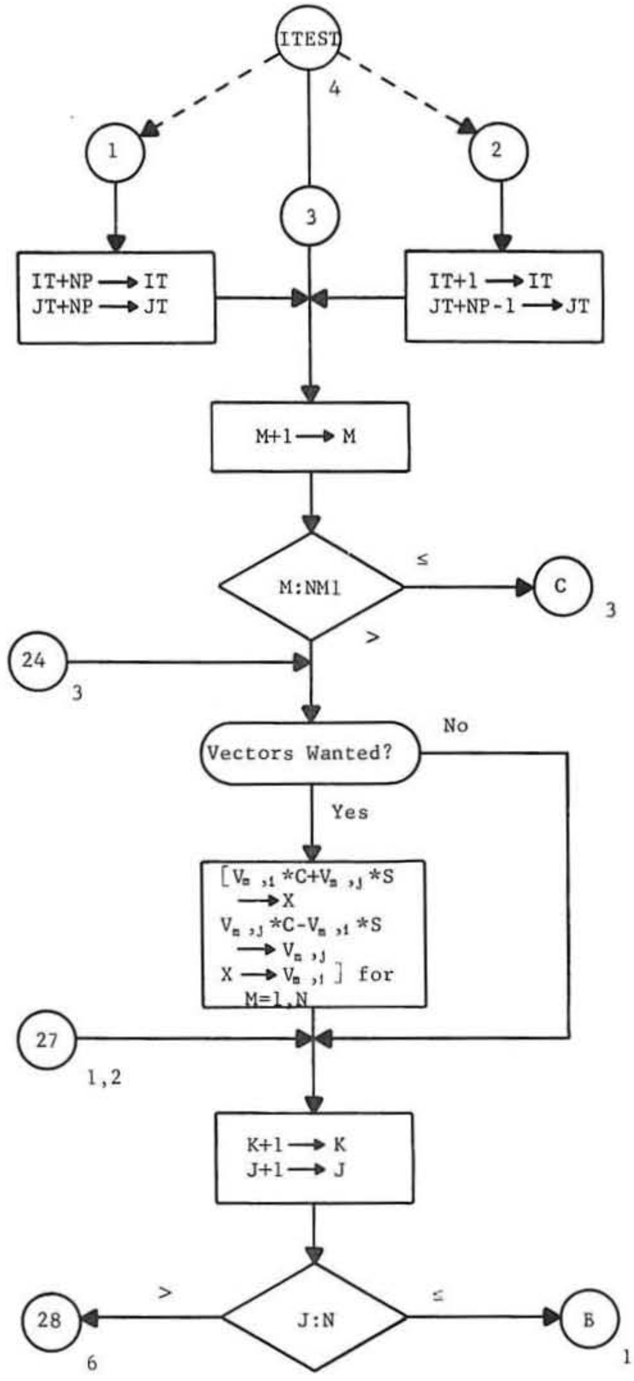
EIGENJ

3


17
3
IT+1 → IT
JT+NP → JT
Set ITEST 2
18
3
IT:K
<
≥
IT+1 → IT
JT+1 → JT
Set ITEST 3
20
3
A_it *C+A_jt *S → X
A_jt *C-A_it *S → A_jt
X → A_it
ITEST
5

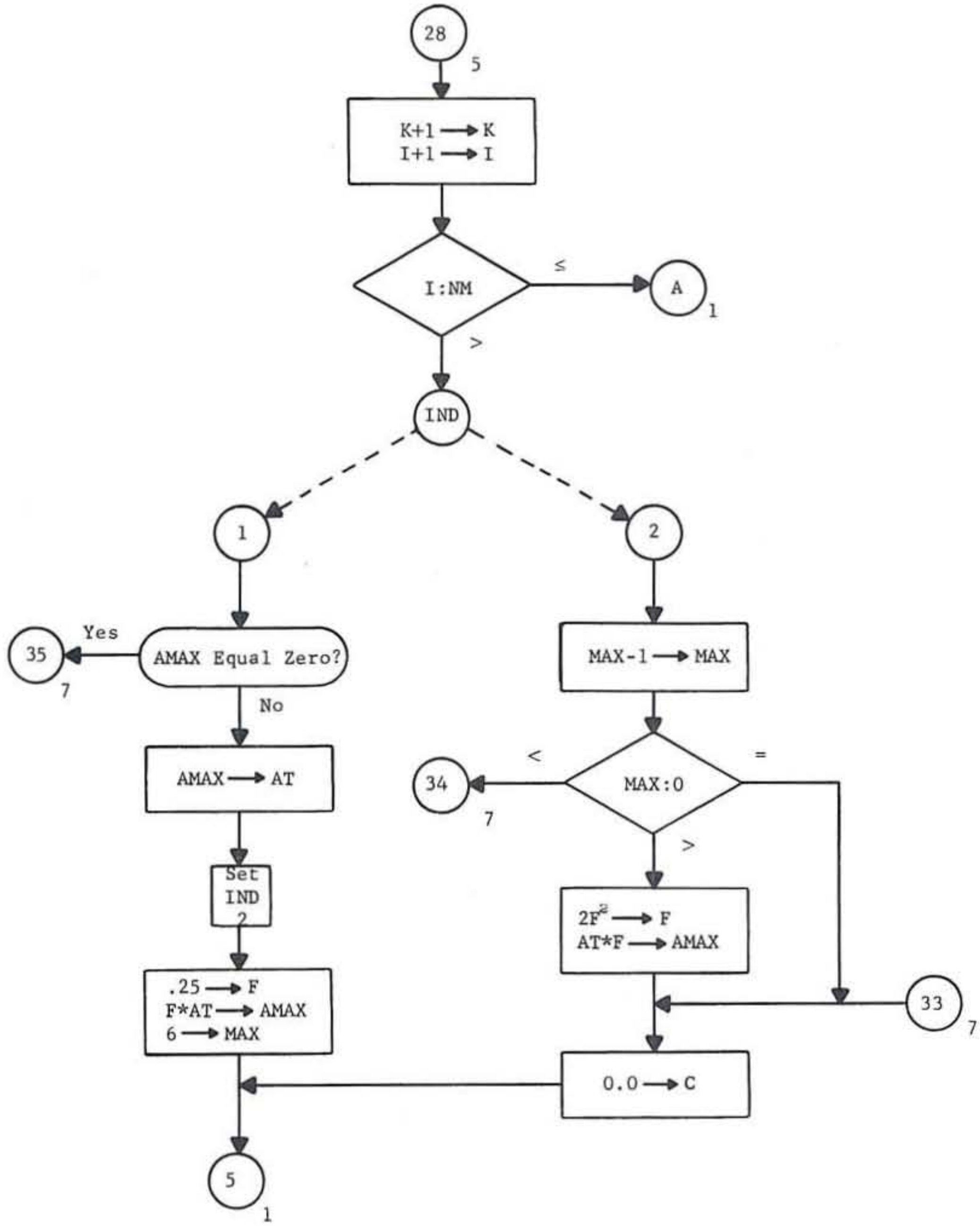
EIGENJ

(4)


ITEST
4
1
2
3
IT+NP → IT
JT+NP → JT
IT+1 → IT
JT+NP-1 → JT
M+1 → M
M:NM1
≤
C
3
>
24
3
Vectors Wanted?
No
Yes
[$V_{m,i}$*C+$V_{m,j}$*S
→ X
$V_{m,j}$*C-$V_{m,i}$*S
→ $V_{m,j}$
X → $V_{m,i}$] for
M=1,N
27
1,2
K+1 → K
J+1 → J
>
28
6
J:N
≤
B
1

EIGENJ

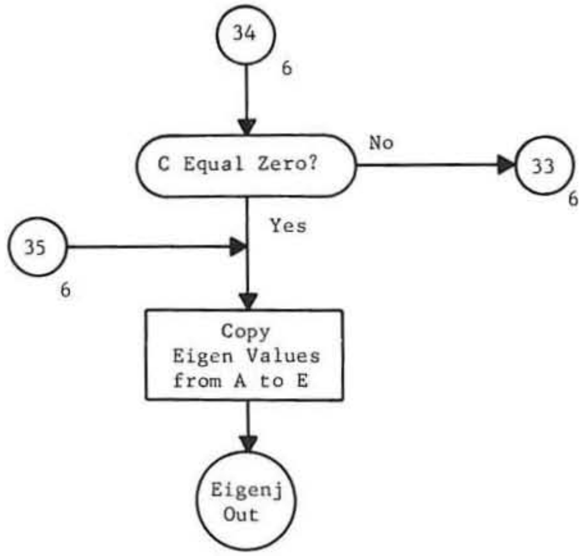
(5)



28
5
K+1 → K
I+1 → I
I:NM
≤
A
1
>
IND
1
2
Yes
35
7
AMAX Equal Zero?
No
AMAX → AT
Set IND 2
.25 → F
F*AT → AMAX
6 → MAX
MAX-1 → MAX
<
=
34
7
MAX:0
>
2F² → F
AT*F → AMAX
33
7
0.0 → C
5
1

EIGENJ

6


34
6
C Equal Zero?
No
33
6
Yes
35
6
Copy
Eigen Values
from A to E
Eigenj
Out

EIGENJ
(7)

EIGENP INPUT

Col
2

____

ID CARD 1

____

ID CARD 2

$SIZE/N=___, IV=___$

$DATA/I=___, J=___, LL/BUF=__________, __________, __________, __________

__________, __________$

$DATA/I=___, J=___, LL/BUF=__________, __________, __________, __________

__________, __________$

$DATA/I=___, J=___, LL/BUF=__________, __________, __________, __________

__________, __________$

$DATA/I=___, J=___, LL/BUF=__________, __________, __________, __________,

__________, __________$

$DATA/I=___, J=___, LL/BUF=__________, __________, __________, __________,

__________, __________$

$DATA/I=___, J=___, LL/BUF=__________, __________, __________, __________,

__________, __________$

$DATA/I=___, J=___, LL/BUF=__________, __________, __________, __________,

__________, __________$

$DATA/I=___, J=___, LL/BUF=__________, __________, __________, __________,

__________, __________$

$DATA/I=___, J=___, LL/BUF=__________, __________, __________, __________,

__________, __________$

$DATA/I=___, J=___, LL/BUF=__________, __________, __________, __________,

__________, __________$

$DATA/I=___, J=___, LL/BUF=__________, __________, __________, __________,

__________, __________$

Note to Keypunch: Discontinue after written $
First two cards must be included, even if blank.

EIGENP INPUT

Col
2

____
ID CARD 1

____
ID CARD 2

$SIZE/N=___, IV=___$

$DATA/I=___, J=___, LL/BUF=__________, __________, __________, __________
__________, __________$

$DATA/I=___, J=___, LL/BUF=__________, __________, __________, __________
__________, __________$

$DATA/I=___, J=___, LL/BUF=__________, __________, __________, __________
__________, __________$

$DATA/I=___, J=___, LL/BUF=__________, __________, __________, __________,
__________, __________$

$DATA/I=___, J=___, LL/BUF=__________, __________, __________, __________,
__________, __________$

$DATA/I=___, J=___, LL/BUF=__________, __________, __________, __________,
__________, __________$

$DATA/I=___, J=___, LL/BUF=__________, __________, __________, __________,
__________, __________$

$DATA/I=___, J=___, LL/BUF=__________, __________, __________, __________,
__________, __________$

$DATA/I=___, J=___, LL/BUF=__________, __________, __________, __________,
__________, __________$

$DATA/I=___, J=___, LL/BUF=__________, __________, __________, __________,
__________, __________$

$DATA/I=___, J=___, LL/BUF=__________, __________, __________, __________,
__________, __________$

Note to Keypunch: Discontinue after written $
First two cards must be included, even if blank.

EIGENP INPUT

Col
2

____

ID CARD 1

____

ID CARD 2

$SIZE/N=\_\_\_, IV=\_\_\_$

$DATA/I=___, J=___, LL/BUF=____________, ____________, ____________, ____________

____________, ____________$

$DATA/I=___, J=___, LL/BUF=____________, ____________, ____________, ____________

____________, ____________$

$DATA/I=___, J=___, LL/BUF=____________, ____________, ____________, ____________

____________, ____________$

$DATA/I=___, J=___, LL/BUF=____________, ____________, ____________, ____________,

____________, ____________$

$DATA/I=___, J=___, LL/BUF=____________, ____________, ____________, ____________,

____________, ____________$

$DATA/I=___, J=___, LL/BUF=____________, ____________, ____________, ____________,

____________, ____________$

$DATA/I=___, J=___, LL/BUF=____________, ____________, ____________, ____________,

____________, ____________$

$DATA/I=___, J=___, LL/BUF=____________, ____________, ____________, ____________,

____________, ____________$

$DATA/I=___, J=___, LL/BUF=____________, ____________, ____________, ____________,

____________, ____________$

$DATA/I=___, J=___, LL/BUF=____________, ____________, ____________, ____________,

____________, ____________$

$DATA/I=___, J=___, LL/BUF=____________, ____________, ____________, ____________,

____________, ____________$

Note to Keypunch: Discontinue after written $
First two cards must be included, even if blank.

EIGENP INPUT

Col
2

____
ID CARD 1

____
ID CARD 2

$SIZE/N=___, IV=___$

$DATA/I=___, J=___, LL/BUF=____________, ____________, ____________, ____________

____________, ____________$

$DATA/I=___, J=___, LL/BUF=____________, ____________, ____________, ____________

____________, ____________$

$DATA/I=___, J=___, LL/BUF=____________, ____________, ____________, ____________

____________, ____________$

$DATA/I=___, J=___, LL/BUF=____________, ____________, ____________, ____________,

____________, ____________$

$DATA/I=___, J=___, LL/BUF=____________, ____________, ____________, ____________,

____________, ____________$

$DATA/I=___, J=___, LL/BUF=____________, ____________, ____________, ____________,

____________, ____________$

$DATA/I=___, J=___, LL/BUF=____________, ____________, ____________, ____________,

____________, ____________$

$DATA/I=___, J=___, LL/BUF=____________, ____________, ____________, ____________,

____________, ____________$

$DATA/I=___, J=___, LL/BUF=____________, ____________, ____________, ____________,

____________, ____________$

$DATA/I=___, J=___, LL/BUF=____________, ____________, ____________, ____________,

____________, ____________$

$DATA/I=___, J=___, LL/BUF=____________, ____________, ____________, ____________,

____________, ____________$

Note to Keypunch: Discontinue after written $
First two cards must be included, even if blank.

EIGENP INPUT

Col
2

____

ID CARD 1

____

ID CARD 2

$SIZE/N=\_\_\_, IV=\_\_\_$

$DATA/I=___, J=___, LL/BUF=__________, __________, __________, __________

__________, __________$

$DATA/I=___, J=___, LL/BUF=__________, __________, __________, __________

__________, __________$

$DATA/I=___, J=___, LL/BUF=__________, __________, __________, __________

__________, __________$

$DATA/I=___, J=___, LL/BUF=__________, __________, __________, __________,

__________, __________$

$DATA/I=___, J=___, LL/BUF=__________, __________, __________, __________,

__________, __________$

$DATA/I=___, J=___, LL/BUF=__________, __________, __________, __________,

__________, __________$

$DATA/I=___, J=___, LL/BUF=__________, __________, __________, __________,

__________, __________$

$DATA/I=___, J=___, LL/BUF=__________, __________, __________, __________,

__________, __________$

$DATA/I=___, J=___, LL/BUF=__________, __________, __________, __________,

__________, __________$

$DATA/I=___, J=___, LL/BUF=__________, __________, __________, __________,

__________, __________$

$DATA/I=___, J=___, LL/BUF=__________, __________, __________, __________,

__________, __________$

Note to Keypunch: Discontinue after written $
First two cards must be included, even if blank.

Progress Is Our Most Important Product

GENERAL ELECTRIC

COMPUTER DEPARTMENT • PHOENIX, ARIZONA

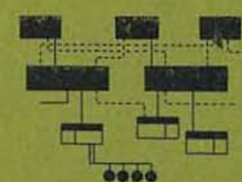
LITHO U.S.A.

GENERAL ELECTRIC
COMPUTERS

GE-625/635 Math Routines

SIMEQ

SIMEQ



GENERAL ELECTRIC

CPB-1167

GE-625/635
MATH ROUTINES
SIMEQ

A SET OF FORTRAN SUBROUTINES FOR SOLVING LINEAR SYSTEMS

Program Number
CD600D4.008

October 1965

GENERAL ELECTRIC
COMPUTER DEPARTMENT

CREDITS

The source material used in this manual is taken from a document published by the General Electric Telecommunications and Information Processing Department, titled SIMEQ: A Set of FORTRAN Subroutines for Solving Linear Systems by H. W. Moore and R. F. Jordan. Permission to use the original document was given by D. L. Shell, Manager, Computer Applications and Processing of the Telecommunications and Information Processing Department, General Electric Company.

Comments on this publication may be addressed to Technical Publications, Computer Department, General Electric Company, P. O. Box 2961, Phoenix, Arizona, 85002.

GENERAL ELECTRIC INFORMATION SYSTEMS DIVISION COMPUTER EQUIPMENT DEPARTMENT	GE-600 SERIES TECHNICAL INFORMATION BULLETIN	DATE June 1967 NO. 600-158
SUBJECT: Correction to CPB-1167		REF. CPB-1167

Update the GE-625/635 Math Routines SIMEQ, CPB-1167, by removing pages 11/12 and replace with the attached revised pages of the same numbers.

It is suggested that this cover sheet be placed in the front of the manual at the time the attached pages are inserted in the manual so it may serve as a quick check to indicate the changes made by this TIB have been incorporated in the manual.

3. PROGRAM USAGE

METHOD

A user employing the SIMEQ routines for the calculation of the determinant or the inverse of a matrix A must head his program with the statement

DIMENSION A(n, n), INTR (n)

where n is an integer not less than the order of the matrix. If the user employs SIMEQ to solve a system of linear equations having the column vector B as a constant term, he must head his program with the statement

DIMENSION A(n, n), INTR (n), B (n, m)

where m is an integer equal to or greater than 1. B is treated as a matrix with double subscripts rather than as a vector or a single-subscripted array, to permit the solution of the system for M sets of constant terms with only one call for the SOLV subroutine. Of course, m must always be equal to or greater than M.

Prior to calling any other SIMEQ subroutines, the user must call DECOM to prepare his matrix for further calculation. This may be done with the statement

CALL DECOM (A, INTR, MSIZE, N)

where A is the name of the matrix, INTR is the name of the auxiliary vector, MSIZE is the dimension of A given in the DIMENSION statement, and N is the order of the matrix. DECOM destroys the original matrix in the process of calculating the decomposed matrix. Therefore, the user must provide additional storage for his matrix if he needs the matrix for later calculation. If the user's matrix has been determined to be singular, a return will be made to the main program with the Nth element of the INTR array set equal to 0. Therefore, the user should test INTR (N) for 0 before proceeding further, since the other SIMEQ subroutines return directly to the calling program without performing any calculation if INTR (N) is 0. If INTR (N) is 0 upon return from DECOM, the user may determine the row in which the matrix was found singular by examining the INTR vector for an element satisfying the relation INTR (K) = K, indicating that the singularity was discovered in the Kth row of the matrix.

Once the user's matrix has been decomposed by DECOM, any of three subroutines may be called, subject only to the restriction that no subroutine can be called after INVRS, which destroys the decomposed matrix in the process of calculating the inverse matrix. (DTMN and INVRS are included in the MINV routine CD600D4.007.)

In order to find the determinant of his matrix, the user calls the DTMN subroutine with the statement

CALL DTMN (A, INTR, MSIZE, N, DET)

where DET is the storage location reserved for the answer. The subroutine returns to the main program with the scaled value of the determinant stored in DET and the integral power of ten by which DET is to be multiplied stored in INTR (N). This form is chosen to avoid overflow or underflow in the calculation of extremely large determinants.

In order to calculate the inverse of this matrix, the user calls the INVRS subroutine with the statement

CALL INVRS (A, INTR, MSIZE, N)

The subroutine returns with the inverse stored in the locations originally occupied by the original matrix. For this reason no other subroutine of SIMEQ, except possibly DECOM, can be called after INVRS has been called.

SIMEQ will probably be used most often to obtain the solutions of systems of simultaneous linear equations. Frequently, several such systems must be solved in one program, all having the same matrix of coefficients and differing only in the constants on their right-hand sides. The SOLV subroutine has been arranged to handle this situation efficiently by allowing the various right-hand sides to be entered as a single matrix each of whose columns represents a right-hand side of the system of equations to be solved. All of the systems are solved simultaneously with a single call of the subroutine.

The user calls SOLV with the statement

CALL SOLV (A, B, INTR, MSIZE, ISIZE, N, M)

where B is the name of the matrix of constant vectors, ISIZE is the maximum number of vectors allowed by the DIMENSION statement, and M is the number of right-hand sides to be solved.

The user stores the right-hand sides in the manner indicated above in the B matrix referred to in the opening DIMENSION statements.

If there is only one right-hand side to be solved for, M is, of course, set equal to 1. Before returning to the main program, the SOLV subroutine stores the solutions of the unknowns in the corresponding positions of the B matrix, destroying in the process the original right-hand sides.

Note that both the right-hand side constants and the unknowns finally obtained must be stored in a double subscripted array even when there is only one right-hand side and one vector of unknown quantities.

CAUTION: Do not solve a system of linear equations by first using DECOM and INVRS to get the inverse of a matrix of coefficients and then multiplying the right-hand side by the inverse to find the solution of the system. This procedure is far more inefficient and time-consuming than the more direct method outlined above, using DECOM and SOLV.

CONTENTS

.1. GENERAL DESCRIPTION

SIMEQ consists of an input/output program and four FORTRAN subroutines called DECOM, SOLV, INVRS, and DTMN. The SIMEQ program uses DECOM and SOLV to solve sets of simultaneous linear equations. The routines may be incorporated in a user's program for equation solutions, matrix inversion, or determinant evaluation.

Storage requirements for the entire system, exclusive of the program instructions, consist of n^2 locations for the user's matrix plus n locations for an auxiliary vector used by all the subroutines. In addition, the SOLV subroutine requires n x m locations for the storage of m vectors, each representing a set of constant terms (a right-hand side) for the system of simultaneous equations to be solved. The required dimensions are passed to the subroutines via the calling sequences, eliminating the necessity of recompiling them for use with different programs.

The basic subroutine of the system is DECOM, and it must, in general, be called before any of the other subroutines may be used. DECOM decomposes the user's matrix into the upper and lower triangular matrices which are used by the other subroutines for further calculation. The user's original matrix is destroyed in the process. This subroutine uses the single library function ABS.

The SOLV subroutine uses the decomposed matrix given by DECOM to solve up to m sets of right-hand sides for the given matrix. The constant terms (right-hand side) of each of the m sets is destroyed in the process.

The DTMN subroutine calculates the determinant of the user's matrix merely by multiplying together the diagonal elements of the decomposed matrix. In order to ensure that this product does not underflow or overflow the arithmetic capabilities of the machine, the determinant is calculated as a fraction and an integral power of ten.

The INVRS subroutine calculates the explicit inverse of the user's matrix from the decomposed matrix, assuming that the original matrix is not singular. The decomposed matrix is destroyed in the process.

2. MATHEMATICAL METHOD

SAMPLE PROBLEM

Although the basic method used by SIMEQ is known as "left-right decomposition," or the decomposition of a matrix into the product of an upper triangular and a lower triangular matrix, it will be advantageous to explain the method initially in terms of an equivalent method created by C. F. Gauss, known as Gaussian elimination.

The following discussion covers the solution of the linear algebraic system

$$a_{11}x_1 + a_{12}x_2 + a_{13}x_3 + a_{14}x_4 = b_1$$

$$a_{21}x_1 + a_{22}x_2 + a_{23}x_3 + a_{24}x_4 = b_2$$

$$a_{31}x_1 + a_{32}x_2 + a_{33}x_3 + a_{34}x_4 = b_3$$

$$a_{41}x_1 + a_{42}x_2 + a_{43}x_3 + a_{44}x_4 = b_4$$

The Gaussian elimination method uses the first equation, which is known as the pivot equation, to eliminate x_1 from each of the other three equations in turn. This is done by dividing the pivot equation by its leading element, known as the pivot element, multiplying it by the leading element of the equation from which x_1 is to be eliminated, and then subtracting it from that equation. In the example below, the first x_1 is to be eliminated from the second equation. The procedure gives

$$(a_{21} - a_{21})x_1 + (a_{22} - \frac{a_{21}}{a_{11}} \cdot a_{12})x_2 + (a_{23} - \frac{a_{21}}{a_{11}} \cdot a_{13})x_3 + (a_{24} - \frac{a_{21}}{a_{11}} \cdot a_{14})x_4 =$$

$$b_2 - \frac{a_{21}}{a_{11}} \cdot b_1$$

That is,

$$0 + a'_{22}x_2 + a'_{23}x_3 + a'_{24}x_4 = b'_2,$$

where

$$a'_{2j} = a_{2j} - \frac{a_{21}}{a_{11}} a_{1j} \text{ and } b'_2 = b_2 - \frac{a_{21}}{a_{11}} b_1.$$

Applied to each of the remaining equations in turn, this procedure results finally in the complete elimination of x_1 from all but the first equation:

$$\begin{array}{llllll} a_{11}x_1 & + a_{12}x_2 & + a_{13}x_3 & + a_{14}x_4 & = & b_1 \\ 0 & + a'_{22}x_2 & + a'_{23}x_3 & + a'_{24}x_4 & = & b'_2 \\ 0 & + a'_{32}x_2 & + a'_{33}x_3 & + a'_{34}x_4 & = & b'_3 \\ 0 & + a'_{42}x_2 & + a'_{43}x_3 & + a'_{44}x_4 & = & b'_4 \end{array}$$

The second equation may be used as a new pivot equation to eliminate x_2 from equations three and four. The procedure used before (that is, dividing the new second equation by its leading element, multiplying it by the leading element of the equation from which x_2 is to be eliminated, and subtracting) gives the new reduced system

$$\begin{array}{llllll} a_{11}x_1 & + a_{12}x_2 & + a_{13}x_3 & + a_{14}x_4 & = & b_1 \\ 0 & + a'_{22}x_2 & + a'_{23}x_3 & + a'_{24}x_4 & = & b'_2 \\ 0 & + 0 & + a''_{33}x_3 & + a''_{34}x_4 & = & b''_3 \\ 0 & + 0 & + a''_{43}x_3 & + a''_{44}x_4 & = & b''_4 \end{array}$$

Finally, the new third equation may be used as the pivot equation and the same procedure may be applied to equation four, giving

$$\begin{array}{llllll} a_{11}x_1 & + a_{12}x_2 & + a_{13}x_3 & + a_{14}x_4 & = & b_1 \\ 0 & + a'_{22}x_2 & + a'_{23}x_3 & + a'_{24}x_4 & = & b'_2 \\ 0 & + 0 & + a''_{33}x_3 & + a''_{34}x_4 & = & b''_3 \\ 0 & + 0 & + 0 & + a'''_{44}x_4 & = & b'''_4 \end{array}$$

At this point, the system of equations is virtually solved. The only unknown in the last equation is x_4, which can, therefore, be determined. When x_4 is known, the only unknown in the third equation is x_3, which can similarly be determined and used to find x_2 from the second equation. Finally, x_1 can be determined from the first equation. The above example shows that Gaussian elimination consists of two basic processes:

(1) A forward course in which the matrix of coefficients is reduced to an upper triangular matrix.

(2) A return course, or "back substitution," in which the unknowns are found recursively, starting with the last equation, which contains only a single unknown quantity.

The pivot element, or leading coefficient of the pivot equation might be 0 or a very small number and, consequently, unsuitable for use in dividing the pivot equation. Thus, one slight modification of the procedure is desirable. As shown above, the first step of the procedure is to eliminate x_i from all equations after the ith. When the pivot element is 0, it is only necessary to choose an equation after the ith with a nonzero coefficient for x_i as a substitute pivot equation and proceed as before. It can be shown that if at the ith step all the coefficients of x_i are 0, the original matrix of coefficients is singular, and, consequently, the system of equations does not have a unique solution. Hence, assuming the system has a unique solution, a nonzero pivot element can always be found. In practice, partly to avoid numerical problems, the steps listed below are used in choosing the ith pivot equation:

(1) Examine the ith column for the largest coefficient of x_i.

(2) Move the equation containing this coefficient to the ith row as the new pivot equation.

(3) Move the old ith equation down to the row originally occupied by the new pivot equation.

This procedure is known as "pivoting on the largest element."

With this modification, Gaussian elimination becomes a highly efficient method of solving linear algebraic systems and is extremely well adapted to digital computers. In fact, it is vastly superior to the more familiar "Cramer's Rule," even for hand calculation, since the number of operations required by Cramer's Rule to evaluate n + 1 determinants is far greater than the number of operations required by Gaussian elimination.

In addition, Gaussian elimination enables any number of sets of equations having the same matrix of coefficients to be solved simultaneously; this problem arises frequently in practice. Suppose, for example that the system of equations given as the original example was to be solved not only with the right-hand side,

$$\begin{matrix} b_1 \\ b_2 \\ b_3 \\ b_4 \end{matrix}$$

but also for the right-hand sides

$$\begin{matrix} c_1 \\ c_2 \\ c_3 \\ c_4 \end{matrix} \quad \text{and} \quad \begin{matrix} d_1 \\ d_2 \\ d_3 \\ d_4 \end{matrix} \quad \text{and} \quad \begin{matrix} e_1 \\ e_2 \\ e_3 \\ e_4 \end{matrix}$$

It would only be necessary to write the system of equations as

$$\begin{aligned}
a_{11}x_1 + a_{12}x_2 + a_{13}x_3 + a_{14}x_4 &= b_1,\ c_1,\ d_1,\ e_1 \\
a_{21}x_1 + a_{22}x_2 + a_{23}x_3 + a_{24}x_4 &= b_2,\ c_2,\ d_2,\ e_2 \\
a_{31}x_1 + a_{32}x_2 + a_{33}x_3 + a_{34}x_4 &= b_3,\ c_3,\ d_3,\ e_3 \\
a_{41}x_1 + a_{42}x_2 + a_{43}x_3 + a_{44}x_4 &= b_4,\ c_4,\ d_4,\ e_4
\end{aligned}$$

and then perform the ith pivot operation, previously performed on b_i, simultaneously on b_i, c_i, d_i and e_i. The back substitution process would consist of simultaneously obtaining the values of each of the four unknowns corresponding to each distinct right-hand side. A comparison of this simple procedure with the steps which would be required by Cramer's Rule to handle this problem demonstrates the power of Gauss' method.

A discussion of one further extension of Gaussian elimination leads almost directly to the process known as left-right decomposition. The elimination process can be performed on the matrix of coefficients before the right-hand sides for the equations are known, and, if certain necessary information is retained, the back substitution process can later be performed when the right-hand sides have been specified. To see what information must be retained from the forward course, consider again the problem of eliminating x_1 from the ith equation, assuming that the first equation is being used as a pivot:

$$a_{11}x_1 + a_{12}x_2 + a_{13}x_3 + a_{14}x_4 = b_1$$

- -

$$a_{i1}x_1 + a_{i2}x_2 + a_{i3}x_3 + a_{i4}x_4 = b_i$$

Dividing the first equation by a_{11}, multiplying it by a_{i1}, and performing subtraction gives for the ith equation

$$0 + (a_{i2} - a_{i1} \cdot a_{12} / a_{11})x_2 + (a_{i3} - a_{i1} . a_{13} / a_{11})x_3 + (a_{i4} - a_{i1} . a_{14} / a_{11})x_4 = b_i - a_{i1} . b_1 / a_{11}$$

Now if $m_{i1} = - a_{i1}/a_{11}$ is defined, then the ith equation may be written

$$0 + (a_{i2} + a_{12}\ m_{i1})x_2 + (a_{i3} + a_{13}\ m_{i1})x_3 + (a_{i4} + a_{14}\ m_{i1})x_4 = b_i + b_1\ m_{i1}$$

and, in this case, it is only necessary to know the multiplier m_{i1} to be able to perform the necessary reduction on the right-hand side of the ith equation. In general, assuming that no interchanges have been made, the new right-hand side can be calculated from the multipliers m_{ij} saved from the jth pivot operation, as j goes from 1 to N-1. That is, when j = 1, the following is calculated

$$b_i' = b_i + m_{i1} . b_1 \text{ for } i = 2, N$$

and then

$$b_i'' = b_i' + m_{i2} b_2' \text{ for } i = 3, N$$

etc.

Except for the necessity of storing all the multipliers m_{ij}, this delayed calculation of the new right-hand side is just as efficient as the concurrent calculation of the right-hand side and the upper triangular matrix, since it requires no additional operations. Note also that only one set of stored multipliers is required no matter how many different right-hand sides are to be solved for, since the transformation is completely determined by the matrix of coefficients.

Finally, if the rows of the matrix are to be interchanged at any point in the reduction to an upper triangular matrix, it is only necessary to interchange right-hand side elements at the corresponding point in the calculation. For example, suppose that on the ith pivot operation on the matrix of coefficients, the (i + j)th row was chosen as the new pivot row, causing the ith and (i + j)th rows to be interchanged. If a record of this fact is preserved when the corresponding point in the calculation of the right-hand side is reached, b_i and $b_i + j$ are merely interchanged and the calculation proceeds as before. Establishing an interchange vector consisting of N elements, one for each row, facilitates preserving the record of the interchanges. If an interchange of the ith and (i + j)th rows takes place, the ith element of the interchange vector is set equal to i + j; otherwise, it is 0. In calculating the right-hand side, before performing the ith pivot operation, the ith element of the interchange vector is examined, and, if it is not 0, the indicated interchange is made before proceeding.

The multipliers m_{ij} may also be stored conveniently by using the positions in the matrix of coefficients where 0's have been introduced. For example, when the first pivot operation is performed on the second row, the leading coefficient of that row becomes 0 and will not be used again. This permits the multiplier $m_{21} = -a_{21}/a_{11}$ to be stored in the leading coefficient's position; similar action may be taken for remaining rows. Hence, after the first pivot operation is concluded, the matrix of coefficients for the fourth degree system used above as an example takes the form

$$\begin{matrix} a_{11} & a_{12} & a_{13} & a_{14} \\ m_{21} & a'_{22} & a'_{23} & a'_{24} \\ m_{31} & a'_{32} & a'_{33} & a'_{34} \\ m_{41} & a'_{42} & a'_{43} & a'_{44} \end{matrix}$$

This may be repeated for the second pivot operation until finally, at the conclusion of the third pivot operation, when the matrix has been completely reduced to upper triangular form, the multipliers are all stored in the lower half as follows:

$$\begin{pmatrix} a_{11} & a_{12} & a_{13} & a_{14} \\ m_{21} & a'_{22} & a'_{23} & a'_{24} \\ m_{31} & m_{32} & a''_{33} & a''_{34} \\ m_{41} & m_{42} & m_{43} & a'''_{44} \end{pmatrix}$$

If an interchange vector has also been made, the above matrix may be used to solve the system of equations represented for any right-hand side or group of right-hand sides.

The DECOM and SOLV subroutines of the SIMEQ system use the above method for solving systems of linear equations as follows:

(1) DECOM initially reduces the matrix of coefficients to the above form recording row interchanges in an interchange vector.

(2) SOLV applies the multipliers m_{ij} and the interchange vector in the manner indicated above to perform the corresponding operations on the right-hand side, and solves the system completely by recursively calculating x_n, x_{n-1}, ----- x_1 by the above mentioned process of "back substitution."

The process known as left-right decomposition must be examined before the methods that SIMEQ uses in securing the determinant and the inverse of a matrix can be clarified.

The Gaussian elimination process which has been discussed above can also be represented by a series of matrix multiplications according to the following steps:

(1) Starting with the fourth order matrix of coefficients used as an example above, define a multiplying matrix M_1 as

$$M_1 = \begin{pmatrix} 1 & 0 & 0 & 0 \\ m_{21} & 1 & 0 & 0 \\ m_{31} & 0 & 1 & 0 \\ m_{41} & 0 & 0 & 1 \end{pmatrix}$$

where the elements m_{i1} are as previously defined.
If the original matrix is represented as

$$A = \begin{pmatrix} a_{11} & a_{12} & a_{13} & a_{14} \\ a_{21} & a_{22} & a_{23} & a_{24} \\ a_{31} & a_{32} & a_{33} & a_{34} \\ a_{41} & a_{42} & a_{43} & a_{44} \end{pmatrix}$$

It is easily verified that the matrix product $M_1 A$ is given by

$$M_1 A = \begin{pmatrix} a_{11} & a_{12} & a_{13} & a_{14} \\ 0 & a'_{22} & a'_{23} & a'_{24} \\ 0 & a'_{32} & a'_{33} & a'_{34} \\ 0 & a'_{42} & a'_{43} & a'_{44} \end{pmatrix} = A_2$$

where the elements a_{ij} are exactly the same as those secured from Gaussian elimination.

(2) Define a new multiplying matrix M_2

$$M_2 = \begin{pmatrix} 1 & 0 & 0 & 0 \\ 0 & 1 & 0 & 0 \\ 0 & m_{32} & 1 & 0 \\ 0 & m_{42} & 0 & 1 \end{pmatrix}$$

where $m_{3j} = -a'_{j2}/a'_{22}$ as before, and form the product

$$M_2 A_2 = \begin{pmatrix} a_{11} & a_{12} & a_{13} & a_{14} \\ 0 & a'_{22} & a'_{23} & a'_{24} \\ 0 & 0 & a''_{33} & a''_{34} \\ 0 & 0 & a''_{43} & a''_{44} \end{pmatrix} = A_3$$

Again the elements a''_{ij} are exactly the same as those secured at this step by Gaussian elimination.

(3) Proceed in the same fashion for the final step. This gives

$$M_3 A_3 = \begin{pmatrix} a_{11} & a_{12} & a_{13} & a_{14} \\ 0 & a'_{22} & a'_{23} & a'_{24} \\ 0 & 0 & a''_{33} & a''_{34} \\ 0 & 0 & 0 & a'''_{44} \end{pmatrix} = A_4$$

where $A_4 = M_3 M_2 M_1 A$

Now, the following equation may be set up to secure the representation of A as a product of an upper triangular and a lower triangular matrix

$$M = M_3 \cdot M_2 \cdot M_1 .$$

Since each matrix M_1 is lower triangular, with 1's on the diagonal, it is easy to show that the product matrix M is also lower triangular with 1's on the diagonal. Therefore,

$$A_4 = M \cdot A$$

and

$$A = M^{-1} A_4$$

where the inverse matrix M^{-1} is also lower triangular with 1's on the diagonal. Thus, A is represented as the product of an upper and a lower triangular matrix. Further, although the matrix M and its inverse are not calculated explicitly in the Gaussian process, it is obviously simple to do so from the multipliers m_{ij} used to reduce the matrix A to upper triangular form.

Having represented the matrix A as the product of an upper and a lower triangular matrix, the solution of a set of linear equations is simple, for, if the equations are represented as

$$A \cdot X = B,$$

substituting M^{-1} A_4 for A gives

$$M^{-1} \cdot A_4 \cdot X = B$$

so that

$$A_4 \cdot X = M \cdot B.$$

Having calculated A_4 and M, the matrix product $M \cdot B$ forms the right hand side and A_4 the left hand side ready for the same process of back substitution used in the Gaussian elimination.

Finding the inverse of the matrix A once the decomposition has taken place is also quite simple, for since

$$A \cdot A^{-1} = I ,$$

making the substitution

$$A = M^{-1} \cdot A_4$$

gives

$$M^{-1} \cdot A_4 \cdot A^{-1} = I ,$$

so that

$$A_4 \cdot A^{-1} = M.$$

Since A_4 and M are known and triangular, the elements of A^{-1} could easily be solved for recursively, starting with the last row and working up just as in back substitution. However, it is also possible to proceed by operating further on A_4 and M in the above equation until A_4 becomes the identity matrix; at this point, M must equal A^{-1}. As a first step in accomplishing this, the elements of A_4 above the diagonal must be replaced by 0's; this may be done by the method of elimination used to introduce 0's below the diagonal in obtaining A_4. After these steps have been performed on A_4 and M, division of A_4 and M by the corresponding diagonal element completes the reduction of A_4 to the identity matrix and leaves the inverse matrix stored in M.

The INVRS subroutine of the SIMEQ system uses this method to calculate the inverse of a matrix, once the DECOM subroutine has reduced the matrix to triangular form. DECOM does not, however, calculate explicitly the matrix M shown in the equation on the preceding page, but rather merely stores the multiplier m_{ij} in columns below the diagonal of the reduced matrix. Hence, before further reductions can take place, the INVRS subroutine must first multiply these columns together in the manner required to produce the matrix M which is the product of the multiplier matrices M_1, M_2, and M_3. The complete reduction of A_4 then proceeds in a manner which allows the newly calculated elements to be stored over the original matrix, making provision of additional storage locations for the inverse unnecessary. If any row interchanges were recorded in the interchange vector in the process of reducing A to triangular form, the corresponding column interchange must be made in the inverse matrix.

Finally, the determinant of the matrix A may be found almost immediately from the decomposed matrix, for

$$\det(A_4) = \det(M \cdot A) = \det(M) \cdot \det(A)$$

But

$$\det(M) = 1$$

since M is lower triangular, and its determinant is consequently the product of its diagonal elements, which are all 1's. Hence

$$\det(A) = \det(A_4).$$

But A_4 is upper triangular, so that its determinant also is given by the product of its diagonal elements; therefore

$$\det(A) = a_{11} \cdot a'_{22} \cdot a''_{33} \cdot a''_{44}$$

If row interchanges took place in the reduction of A because of pivot operations, the sign of the calculated determinant must be changed only once for each such interchange, as is well known from the theory of determinants. The DTMN subroutine of the SIMEQ system uses this method to calculate the determinant of a given matrix once DECOM has reduced it to triangular form.

REFERENCES

Further descriptions of Gaussian elimination and left-right decomposition can be found in almost any standard book on numerical analysis. An excellent discussion is contained in "Computational Methods of Linear Algebra" by V. N. Faddeeva, Dover Publications, 1959.

3. PROGRAM USAGE

METHOD

A user employing the SIMEQ routines for the calculation of the determinant or the inverse of a matrix A must head his program with the statement

DIMENSION A(n, n), INTR (n)

where n is an integer not less than the order of the matrix. If the user employs SIMEQ to solve a system of linear equations having the column vector B as a constant term, he must head his program with the statement

DIMENSION A(n, n), INTR (n), B (n, m)

where m is an integer equal to or greater than 1. B is treated as a matrix with double subscripts rather than as a vector or a single-subscripted array, to permit the solution of the system for M sets of constant terms with only one call for the SOLV subroutine. Of course, m must always be equal to or greater than M.

Prior to calling any other SIMEQ subroutines, the user must call DECOM to prepare his matrix for further calculation. This may be done with the statement

CALL DECOM (A, INTR, MSIZE, N)

where A is the name of the matrix, INTR is the name of the auxiliary vector, MSIZE is the dimension of A given in the DIMENSION statement, and N is the order of the matrix. DECOM destroys the original matrix in the process of calculating the decomposed matrix. Therefore, the user must provide additional storage for his matrix if he needs the matrix for later calculation. If the user's matrix has been determined to be singular, a return will be made to the main program with the Nth element of the INTR array set equal to 0. Therefore, the user should test INTR (N) for 0 before proceeding further, since the other SIMEQ subroutines return directly to the calling program without performing any calculation if INTR (N) is 0. If INTR (N) is 0 upon return from DECOM, the user may determine the row in which the matrix was found singular by examining the INTR vector for an element satisfying the relation INTR (K) = K, indicating that the singularity was discovered in the Kth row of the matrix.

Once the user's matrix has been decomposed by DECOM, any or all of the other three SIMEQ subroutines may be called, subject only to the restriction that no subroutine can be called after INVRS, which destroys the decomposed matrix in the process of calculating the inverse matrix.

In order to find the determinant of his matrix, the user calls the DTMN subroutine with the statement

CALL DTMN (A, INTR, MSIZE, N, DET)

where DET is the storage location reserved for the answer. The subroutine returns to the main program with the scaled value of the determinant stored in DET and the integral power of ten by which DET is to be multiplied stored in INTR (N). This form is chosen to avoid overflow or underflow in the calculation of extremely large determinants.

In order to calculate the inverse of this matrix, the user calls the INVRS subroutine with the statement

CALL INVRS (A, INTR, MSIZE, N)

The subroutine returns with the inverse stored in the locations originally occupied by the original matrix. For this reason no other subroutine of SIMEQ, except possibly DECOM, can be called after INVRS has been called.

SIMEQ will probably be used most often to obtain the solutions of systems of simultaneous linear equations. Frequently, several such systems must be solved in one program, all having the same matrix of coefficients and differing only in the constants on their right-hand sides. The SOLV subroutine has been arranged to handle this situation efficiently by allowing the various right-hand sides to be entered as a single matrix each of whose columns represents a right-hand side of the system of equations to be solved. All of the systems are solved simultaneously with a single call of the subroutine.

The user calls SOLV with the statement

CALL SOLV (A, B, INTR, MSIZE, ISIZE, N, M)

where B is the name of the matrix of constant vectors, ISIZE is the maximum number of vectors allowed by the DIMENSION statement, and M is the number of right-hand sides to be solved.

The user stores the right-hand sides in the manner indicated above in the B matrix referred to in the opening DIMENSION statements.

If there is only one right-hand side to be solved for, M is, of course, set equal to 1. Before returning to the main program, the SOLV subroutine stores the solutions of the unknowns in the corresponding positions of the B matrix, destroying in the process the original right-hand sides.

Note that both the right-hand side constants and the unknowns finally obtained must be stored in a double subscripted array even when there is only one right-hand side and one vector of unknown quantities.

CAUTION: Do not solve a system of linear equations by first using DECOM and INVRS to get the inverse of a matrix of coefficients and then multiplying the right-hand side by the inverse to find the solution of the system. This procedure is far more inefficient and time-consuming than the more direct method outlined above, using DECOM and SOLV.

INPUT/OUTPUT

SIMEQ is also available as a free-standing package program, with input as follows:

```
$P/NAME=60H         Alphanumeric identification
.                             (Note comma on separate card)
CASE=60H            Second card of identification
,                             (Note comma on separate card)
TITLE=60H           Third card of identification
,                             (Note comma on separate card)

N=          , M=        , NEW=      $
```

where

N is the number of equations
M is the number of constant vectors (up to 6)
If this is a new case, NEW is T
If this is only a new set of constant vectors, NEW is F

```
$X/L  /C= ,  ,  ,  ,  ,  ,  $ (basic input card)
```

The blank after L is filled with A for as many cards as needed to read in the coefficients, then with B for the constant vectors, and, finally, with E to mark the end of the case. The first two blanks after the = sign are for row number and column number. Succeeding blanks are filled with elements, by rows. Any elements not entered are set to zero by the program.

Further cases may be read in by repeating the above, starting with the $P card.

Input for a sample case is illustrated on the following page. Additional input sheets are furnished at the back of this manual for the user's convenience. A listing of input cards and an output listing are also included in this section.

Input Coding Form

SIMEQ
INPUT CODING FORM

Col
2

$P/NAME=60H *MATRIX SOLUTION*
,
CASE=60H______
,
TITLE=60H______
,
N= *6* , M= *1* , NEW= *T* $

$X/L *A* /C= *1* , *1* , *-475.69* , *-22.854* , *20.493* , *-18.404* , *16.755* ,
*-15.3$* , ______, ______, ______, ______, ______$
$X/L *A* /C= *2* , *1* , *11.427* , *487.47* , *12.222* , *-11.628* , *10.99* ,
*-10.32$* , ______, ______, ______, ______, ______$
$X/L *A* /C= *3* , *1* , *6.83* , *-8.148* , *-491.593* , *-8.292* , *8.055* ,
*-7.61$* , ______, ______, ______, ______, ______$
$X/L *A* /C= *4* , *1* , *4.601* , *-5.814* , *6.219* , *493.68* , *6.26* ,
*-6.06$* , ______, ______, ______, ______, ______$
$X/L *A* /C= *5* , *1* , *3.351* , *-4.38* , *4.833* , *-5.008* , *-494.9* ,
*-4.872$* , ______, ______, ______, ______, ______$
$X/L *A* /C= *6* , *1* , *2.55* , *-3.44* , *3.84* , *-4.04* , *4.075* ,
*499.96$* , ______, ______, ______, ______, ______$
$X/L *B* /C= *1* , *1* , *39.368$* , ______, ______, ______, ______,
______, ______, ______, ______, ______, ______$
$X/L *B* /C= *2* , *1* , *12.555$* , ______, ______, ______, ______,
______, ______, ______, ______, ______, ______$
$X/L *B* /C= *3* , *1* , *6.3006$* , ______, ______, ______, ______,
______, ______, ______, ______, ______, ______$
$X/L *B* /C= *4* , *1* , *3.8327$* , ______, ______, ______, ______,
______, ______, ______, ______, ______, ______$
$X/L *B* /C= *5* , *1* , *2.5966$* , ______, ______, ______, ______,
______, ______, ______, ______, ______, ______$
$X/L *B* /C= *6* , *1* , *1.8844$* , ______, ______, ______, ______,
______, ______, ______, ______, ______, ______$
$X/L *E* /C= *$* , ___, ______, ______, ______, ______, ______,
______, ______, ______, ______, ______, ______$
$X/L___/C=___, ___, ______, ______, ______, ______, ______,
______, ______, ______, ______, ______, ______$
$X/L___/C=___, ___, ______, ______, ______, ______, ______,
______, ______, ______, ______, ______, ______$
$X/L___/C=___, ___, ______, ______, ______, ______, ______,
______, ______, ______, ______, ______, ______$

Notes:
1. Values not entered are set to zero
2. Punch all data starting in column 2
3. Discontinue punching after handwritten $

SIMEQ
INPUT CODING FORM

Col
2

$P/NAME=60H MATRIX SOLUTION

CASE=60H____

TITLE=60H____

N= 6 ,M= 1 ,NEW= T $

$X/L A /C= 1 , 1 , -275.69 , -22.854 , 20.493 , -18.404 , 16.755 ,
-15.3$, ____ , ____ , ____ , ____ , ____ $

$X/L A /C= 2 , 1 , 11.427 , 287.47 , 12.222 , -11.628 , 10.99 ,
-10.32 , ____ , ____ , ____ , ____ , ____ $

$X/L A /C= 3 , 1 , 6.83 , -8.148 , -291.593 , -8.292 , 8.055 ,
-7.68$, ____ , ____ , ____ , ____ , ____ $

$X/L A /C= 4 , 1 , 4.601 , -5.814 , 6.219 , 293.68 , 6.26 ,
-6.06$, ____ , ____ , ____ , ____ , ____ $

$X/L A /C= 5 , 1 , 3.351 , -4.38 , 4.833 , -5.008 , -294.9 ,
-4.872$, ____ , ____ , ____ , ____ , ____ $

$X/L A /C= 6 , 1 , 2.55 , -3.44 , 3.84 , -4.04 , 4.075 ,
299.96$, ____ , ____ , ____ , ____ , ____ $

$X/L B /C= 1 , 1 , 39.308$, ____ , ____ , ____ , ____ ,
____ , ____ , ____ , ____ , ____ , ____ $

$X/L B /C= 2 , 1 , 12.555$, ____ , ____ , ____ , ____ ,
____ , ____ , ____ , ____ , ____ , ____ $

$X/L B /C= 3 , 1 , 6.3006$, ____ , ____ , ____ , ____ ,
____ , ____ , ____ , ____ , ____ , ____ $

$X/L B /C= 4 , 1 , 3.8327$, ____ , ____ , ____ , ____ ,
____ , ____ , ____ , ____ , ____ , ____ $

$X/L B /C= 5 , 1 , 2.5966$, ____ , ____ , ____ , ____ ,
____ , ____ , ____ , ____ , ____ , ____ $

$X/L B /C= 6 , 1 , 1.8844$, ____ , ____ , ____ , ____ ,
____ , ____ , ____ , ____ , ____ , ____ $

$X/L E /C= $, ____ , ____ , ____ , ____ , ____ , ____ ,
____ , ____ , ____ , ____ , ____ , ____ $

$X/L____/C=____, ____ , ____ , ____ , ____ , ____ , ____ ,
____ , ____ , ____ , ____ , ____ , ____ $

$X/L____/C=____, ____ , ____ , ____ , ____ , ____ , ____ ,
____ , ____ , ____ , ____ , ____ , ____ $

$X/L____/C=____, ____ , ____ , ____ , ____ , ____ , ____ ,
____ , ____ , ____ , ____ , ____ , ____ $

Notes
1. Values not entered are set to zero
2. Punch all data starting in column 2
3. Discontinue punching after handwritten $

Listing of Input Cards

```
$P/NAME=60HMATRIX SOLUTION
,
CASE=60H
,
TITLE=60H
,
N=6,M=1,NEW=T$
$X/LA/C=1,1,-475.69,-22.854,20.493,-18.404,16.755,-15.3$
$X/LA/C=2,1,11.427,487.47,12.222,-11.628,10.99,-10.32$
$X/LA/C=3;1,6.83,-8.148,-491.593,-8.292,8.055,-7.68$
$X/LA/C=4,1,4.601,-5.814,6.219,493.68,6.26,-6.06$
$X/LA/C=5,1,3.351,-4.38,4.833,-5.008,-494.9,-4.872$
$X/LA/C=6,1,2.55,-3.44,3.84,-4.04,4.075,499.96$
$X/LB/C=1;1,39.308$
$X/LB/C=2,1,12.555$
$X/LB/C=3;1,6.3006$
$X/LB/C=4,1,3.8327$
$X/LB/C=5,1,2.5966$
$X/LB/C=6;1,1.8844$
$X/LE/C$
$P/NAME=60HMATRIX SOLUTION


,
CASE=60H
,
TITLE=60H
,
N=6,M=1,NEW=T$
$X/LA/C=1,1,-275.69,-22.854,20.493,-18.404,16.755,-15.3$
$X/LA/C=2,1,11.427,287.47,12.222,-11.628,10.99,-10.32$
$X/LA/C=3,1,6.83,-8.148,-291.593,-8.292,8.055,-7.68$
$X/LA/C=4,1,4.601,-5.814,6.219,293.68,6.26,-6.06$
$X/LA/C=5,1,3.351,-4.38,4.833,-5.008,-294.9,-4.872$
$X/LA/C=6,1,2.55,-3.44,3.84,-4.04,4.705,299.96$
$X/LB/C=1,1,39.308$
$X/LB/C=2,1,12.555$
$X/LB/C=3;1,6.3006$
$X/LB/C=4,1,3.8327$
$X/LB/C=5;1,2.5966$
$X/LB/C=6,1,1.8844$
$X/LE/C$
```

```
          MATRIX SOLUTION


  INPUT MATRIX
    1    1 -4.75689999E 02 -2.28540001E 01  2.04930000E 01 -1.84040000E 01  1.67550001E 01 -1.53000000E 01
    2    1  1.14270000E 01  4.87470001E 02  1.22220000E 01 -1.16280000E 01  1.09900000E 01 -1.03200001E 01
    3    1  6.82999998E 00 -8.14800000E 00 -4.91592999E 02 -8.29200006E 00  8.05499995E 00 -7.68000001E 00
    4    1  4.60100001E 00 -5.81400001E 00  6.21899998E 00  4.93680000E 02  6.25999999E 00 -6.06000000E 00
    5    1  3.35100001E 00 -4.38000000E 00  4.83300000E 00 -5.00800002E 00 -4.94900002E 02 -4.87199998E 00
    6    1  2.55000001E 00 -3.44000000E 00  3.84000000E 00 -4.04000002E 00  4.07499999E 00  4.99959999E 02


  CONSTANT VECTORS
    1    1  3.93080001E 01
    2    1  1.25549999E 01
    3    1  6.30059999E 00
    4    1  3.83270001E 00
    5    1  2.59660000E 00
    6    1  1.88440000E 00


  SOLUTION
    1    1 -8.53752512E-02
    2    1  2.85896577E-02
    3    1 -1.48090016E-02
    4    1  9.22009815E-03
    5    1 -6.36143453E-03
    6    1  4.64135903E-03
```

```
      MATRIX SOLUTION

INPUT MATRIX
   1    1 -2.75689999E 02 -2.28540001E 01  2.04930000E 01 -1.84040000E 01  1.67550001E 01 -1.53000000E 01
   2    1  1.14270000E 01  2.87470001E 02  1.22220000E 01 -1.16280000E 01  1.09900000E 01 -1.03200001E 01
   3    1  6.82999998E 00 -8.14800000E 00 -2.91592999E 02 -8.29200006E 00  8.05499995E 00 -7.68000001E 00
   4    1  4.60100001E 00 -5.81400001E 00  6.21899998E 00  2.93680000E 02  6.25999999E 00 -6.06000000E 00
   5    1  3.35100001E 00 -4.38000000E 00  4.83300000E 00 -5.00800002E 00 -2.94900002E 02 -4.87199998E 00
   6    1  2.55000001E 00 -3.44000000E 00  3.84000000E 00 -4.04000002E 00  4.70499998E 00  2.99959999E 02

CONSTANT VECTORS
   1    1  3.93080001E 01
   2    1  1.25549999E 01
   3    1  6.30059999E 00
   4    1  3.83270001E 00
   5    1  2.59660000E 00
   6    1  1.88440000E 00

SOLUTION
   1    1 -1.51385216E-01
   2    1  5.23641114E-02
   3    1 -2.76868069E-02
   4    1  1.74900496E-02
   5    1 -1.22016157E-02
   6    1  8.95102869E-03
```

APPENDIX A
PROGRAM LISTING

```
$       IDENT   AAV,GE,FORTRAN,SIMEQ
$       OPTION  FORTRAN,GO
$       FORTRAN LSTOU,DECK,STAB
$       INCODE  IBMF
*SIMEQ          SOLUTION OF SIMULTANEOUS EQUATIONS
*                                CD600D4.008       DATE 05/04/65
      LOGICAL NEW,LAST
      COMMON A(150,150),B(150,6),INTR(150)
      DIMENSION C(33)
      DIMENSION NAME(10),CASE(10),TITLE(10)
      NAMELIST/P/NAME,CASE,TITLE,N,M,NEW
      NAMELIST/X/A,B,C,LA,LB,LE
      CALL FLGEOF(5,LAST)
    1 READ(5,P)
      IF(LAST)CALL EXIT
      WRITE(6,2)NAME,CASE,TITLE
    2 FORMAT(1H15X,10A6/(1H05X,10A6))
      IF(.NOT.NEW)GO TO 4
*     CLEAR MATRIX
      DO 3 J=1,N
      DO 3 I=1,N
    3 A(I,J)=0.0
*     CLEAR VECTORS
    4 DO 5 J=1,M
      DO 5 I=1,N
    5 B(I,J)=0.0
*     READ DATA
    6 LA=0
      READ(5,X)
      IF(LA.EQ.0)GO TO 8
      I=C(1)
```

```
      J=C(2)
      DO 7 K=3,LA
      A(I,J)=C(K)
    7 J=J+1
      GO TO 6
*     ONE CARD FROM B READ WHILE READING A
    8 I=C(1)
      J=C(2)
      DO 17 K=3,LB
      B(I,J)=C(K)
   17 J=J+1
      LB=0
      READ(5,X)
      IF(LB.NE.0) GO TO 8
      IF(.NOT.NEW)GO TO 10
*     PRINT MATRIX AND CALL DECOM
      WRITE(6,9)
    9 FORMAT(///13H INPUT MATRIX)
      CALL PMAT(A,150,150,N,N)
      CALL DECOM(A,INTR,150,N)
*     PRINT VECTORS AND CALL SOLV
   10 WRITE(6,11)
   11 FORMAT(///17H CONSTANT VECTORS)
      CALL PMAT(B,150,6,N,M)
      CALL SOLV(A,B,INTR,150,6,N,M)
      IF(INTR(N))14,12,14
   12 WRITE(6,13)
   13 FORMAT(//9H SINGULAR)
      GO TO 1
   14 WRITE(6,15)
   15 FORMAT(///9H SOLUTION)
      CALL PMAT(B,150,6,N,M)
      GO TO 1
```

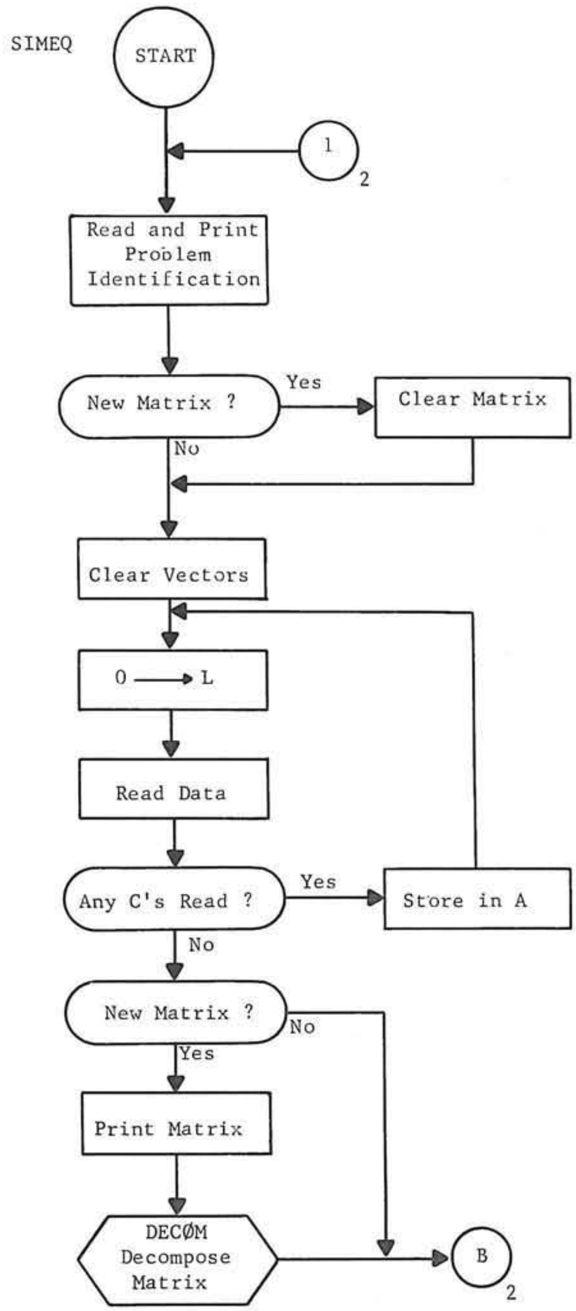
```
      END                                                               SIMEQ640
$     FORTRAN LSTOU,DECK,STAB                                           DECOM000
$     INCODE  IBMF                                                      DECOM010
*DECOM      SUBROUTINE TO DECOMPOSE MATRIX FOR SIMULTANEOUS EQUATIONS   DECOM020
*                           CD600D4.008      DATE 05/04/65              DECOM030
      SUBROUTINE DECOM(A,INTR,MSIZE,NN)                                 DECOM040
      DIMENSION A(MSIZE,MSIZE),INTR(MSIZE)                              DECOM050
*     MATRIX DECOMPOSITION USED WITH SOLV SUBROUTINE FOR SOLUTION       DECOM060
*     OF LINEAR SYSTEMS                                                 DECOM070
*     IF MATRIX A IS SINGULAR INTR(N) WILL BE SET TO ZERO               DECOM080
      N=NN                                                              DECOM090
      NTR=1                                                             DECOM100
      NM=N-1                                                            DECOM110
      DO 10 J=1,NM                                                      DECOM120
      AMAX=ABS(A(J,J))                                                  DECOM130
      JP=J+1                                                            DECOM140
      IN=0                                                              DECOM150
      DO 2 I=JP,N                                                       DECOM160
      AT=ABS(A(I,J))                                                    DECOM170
      IF(AMAX-AT)1,2,2                                                  DECOM180
    1 AMAX=AT                                                           DECOM190
      IN=I                                                              DECOM200
    2 CONTINUE                                                          DECOM210
      IF(AMAX)4,3,4                                                     DECOM220
    3 INTR(J)=J                                                         DECOM230
      GO TO 11                                                          DECOM240
    4 IF(IN)5,7,5                                                       DECOM250
    5 NTR=-NTR                                                          DECOM260
      DO 6 I=J,N                                                        DECOM270
      AT=A(J,I)                                                         DECOM280
      A(J,I)=A(IN,I)                                                    DECOM290
    6 A(IN,I)=AT                                                        DECOM300
    7 INTR(J)=IN                                                        DECOM310
```

```
      AMAX=-1.0/A(J,J)                                                  DECOM320
      DO 10 I=JP,N                                                      DECOM330
      IF(A(I,J))8,10,8                                                  DECOM340
    8 AT=A(I,J)*AMAX                                                    DECOM350
      A(I,J)=AT                                                         DECOM360
      DO 9 K=JP,N                                                       DECOM370
    9 A(I,K)=A(J,K)*AT+A(I,K)                                           DECOM380
   10 CONTINUE                                                          DECOM390
      IF(A(N,N))12,11,12                                                DECOM400
   11 NTR=0                                                             DECOM410
   12 INTR(N)=NTR                                                       DECOM420
      RETURN                                                            DECOM430
      END                                                               DECOM440
$      FORTRAN LSTOU,DECK,STAB                                          SOLV0000
$      INCODE  IBMF                                                     SOLV0010
*SOLV         SUBROUTINE TO SOLVE LINEAR SYSTEMS, CALL DECOM FIRST      SOLV0020
*                                 CD600D4.008      DATE 05/04/65        SOLV0030
      SUBROUTINE SOLV(A,B,INTR,MSIZE,ISIZE,NN,MM)                       SOLV0040
      DIMENSION A(MSIZE,MSIZE),B(MSIZE,ISIZE),INTR(MSIZE)               SOLV0050
*     DECOM SUBROUTINE MUST BE CALLED BEFORE SOLV TO GET SOLUTIONS      SOLV0060
*     OF M SETS OF N EQUATIONS IN N UNKNOWNS                            SOLV0070
      N=NN                                                              SOLV0080
      M=MM                                                              SOLV0090
      IF(INTR(N))1,15,1                                                 SOLV0100
    1 NM=N-1                                                            SOLV0110
      DO 8 K=1,NM                                                       SOLV0120
      L=INTR(K)                                                         SOLV0130
      IF(L)2,4,2                                                        SOLV0140
    2 DO 3 I=1,M                                                        SOLV0150
      X=B(K,I)                                                          SOLV0160
      B(K,I)=B(L,I)                                                     SOLV0170
    3 B(L,I)=X                                                          SOLV0180
    4 KP=K+1                                                            SOLV0190
```

```
      DO 7 I=KP,N                                                       SOLV0200
      X=A(I,K)                                                          SOLV0210
      IF(X)5,7,5                                                        SOLV0220
    5 DO 6 J=1,M                                                        SOLV0230
    6 B(I,J)=B(K,J)*X+B(I,J)                                            SOLV0240
    7 CONTINUE                                                          SOLV0250
    8 CONTINUE                                                          SOLV0260
*     BACK SUBSTITUTION                                                 SOLV0270
      DO 14 K=1,N                                                       SOLV0280
      L=N-K                                                             SOLV0290
      KP=L+1                                                            SOLV0300
      X=1.0/A(KP,KP)                                                    SOLV0310
      DO 9 J=1,M                                                        SOLV0320
    9 B(KP,J)=B(KP,J)*X                                                 SOLV0330
      IF(L)10,14,10                                                     SOLV0340
   10 DO 13 I=KP,N                                                      SOLV0350
      X=A(L,I)                                                          SOLV0360
      IF(X)11,13,11                                                     SOLV0370
   11 DO 12 J=1,M                                                       SOLV0380
   12 B(L,J)=B(L,J)-B(I,J)*X                                            SOLV0390
   13 CONTINUE                                                          SOLV0400
   14 CONTINUE                                                          SOLV0410
   15 RETURN                                                            SOLV0420
      END                                                               SOLV0430
$      FORTRAN LSTOU,DECK                                               PMAT0000
$      INCODE  IBMF                                                     PMAT0010
*PMAT         SUBROUTINE TO PRINT MATRIX                                PMAT0020
*                                  CD600D4.008      DATE 05/04/65       PMAT0030
      SUBROUTINE PMAT(A,MM,NN,NR,NC)                                    PMAT0040
      DIMENSION A(MM,NN),P(6)                                           PMAT0050
      NROW=NR                                                           PMAT0060
      NCOL=NC                                                           PMAT0070
      I=1                                                               PMAT0080
```

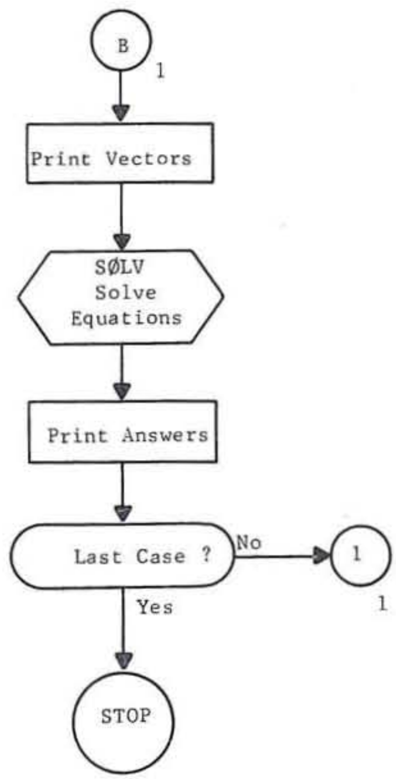
```
      J=1                                                               PMAT0090
    1 IP=I                                                              PMAT0100
      JP=J                                                              PMAT0110
      DO 2 K=1,6                                                        PMAT0120
      KK=K                                                              PMAT0130
      P(K)=A(I,J)                                                       PMAT0140
      J=J+1                                                             PMAT0150
      IF(J.GT.NCOL)GO TO 3                                              PMAT0160
    2 CONTINUE                                                          PMAT0170
      WRITE(6,4)IP,JP,(P(K),K=1,KK)                                     PMAT0180
    4 FORMAT(2I4,6(1PE16.8))                                            PMAT0190
      GO TO 1                                                           PMAT0200
    3 WRITE(6,4)IP,JP,(P(K),K=1,KK)                                     PMAT0210
      I=I+1                                                             PMAT0220
      J=1                                                               PMAT0230
      IF(I.LE.NROW)GO TO 1                                              PMAT0240
      RETURN                                                            PMAT0250
      END                                                               PMAT0260
```

APPENDIX B
FLOW CHARTS



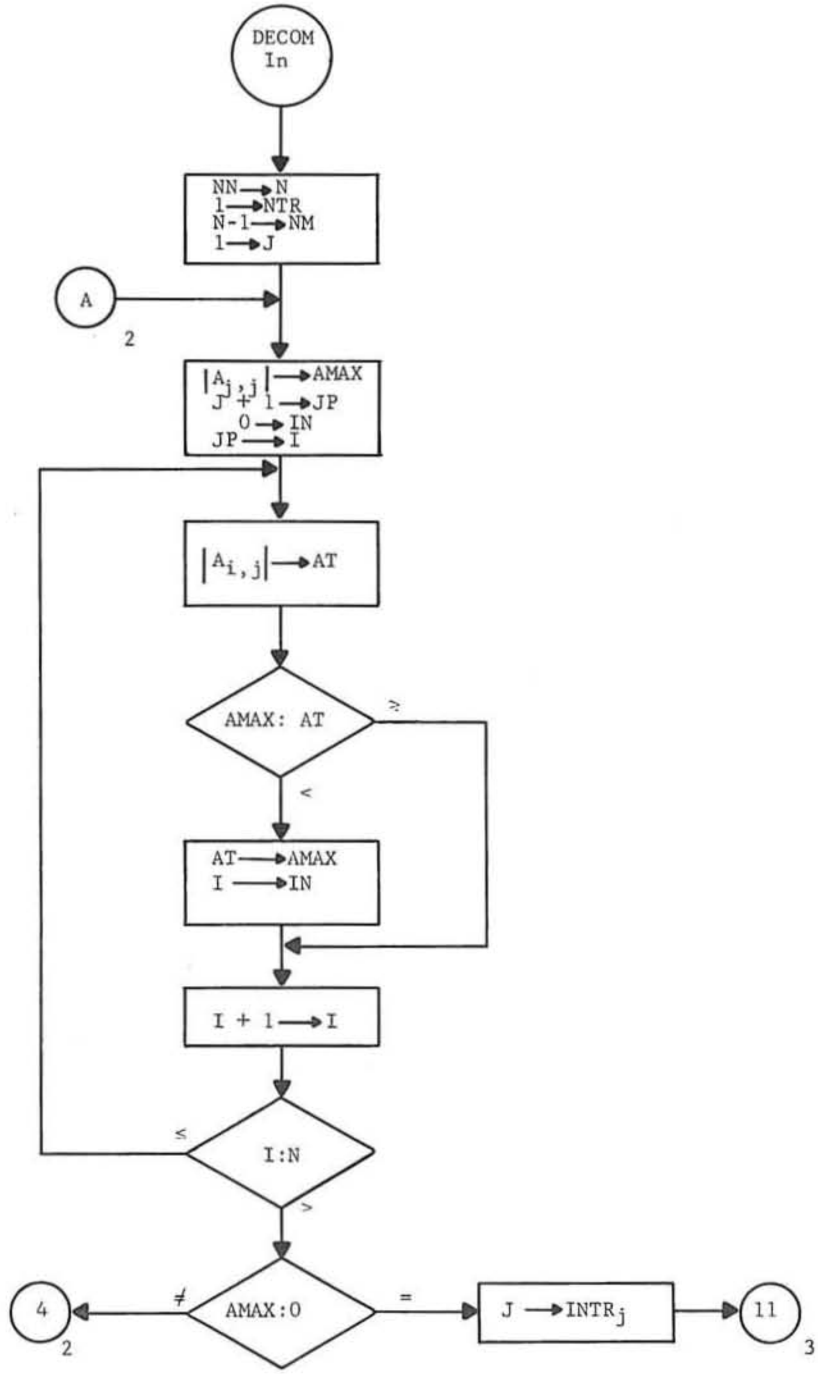
SIMEQ

①


B
1
Print Vectors
SØLV
Solve
Equations
Print Answers
Last Case ?
No
1
1
Yes
STOP

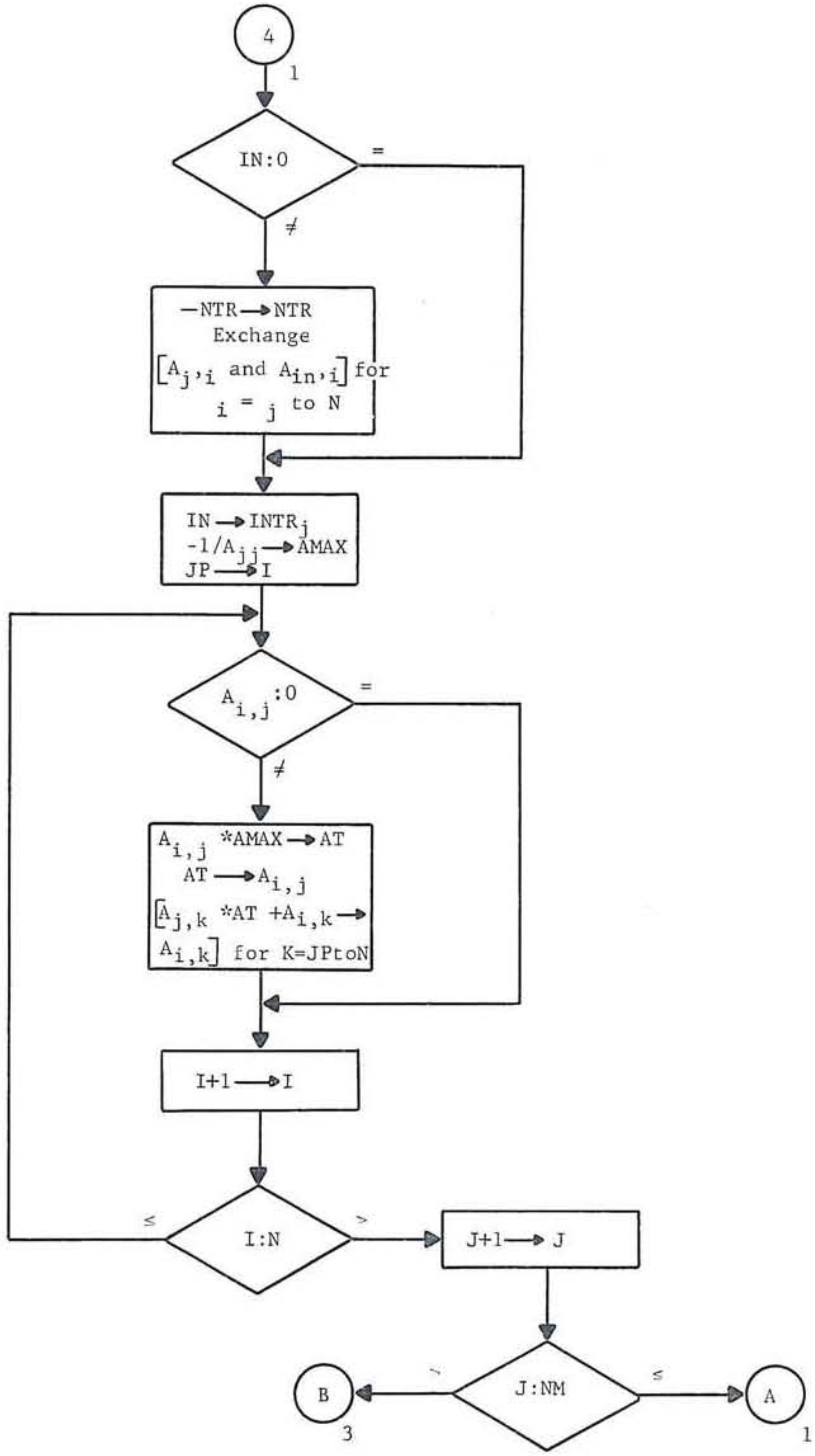
SIMEQ

(2)


DECOM In
NN → N
1 → NTR
N-1 → NM
1 → J
A
2
|A_{j,j}| → AMAX
J + 1 → JP
0 → IN
JP → I
|A_{i,j}| → AT
AMAX: AT
≥
<
AT → AMAX
I → IN
I + 1 → I
I:N
≤
>
AMAX:0
≠
=
4
2
J → INTR_j
11
3

DECØM

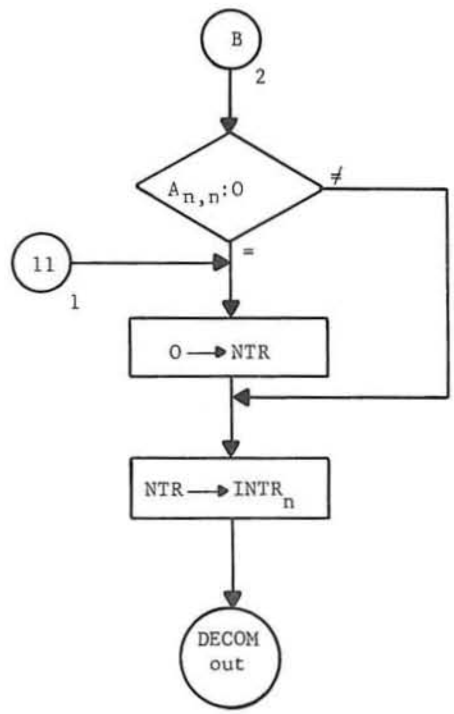
①



4
1
IN:0
=
≠
—NTR → NTR
Exchange
[A j,i and A in,i] for
i = j to N
IN → INTR j
-1/A jj → AMAX
JP → I
A i,j :0
=
≠
A i,j *AMAX → AT
AT → A i,j
[A j,k *AT +A i,k →
A i,k] for K=JPtoN
I+1 → I
≤
I:N
>
J+1 → J
>
J:NM
≤
B
3
A
1

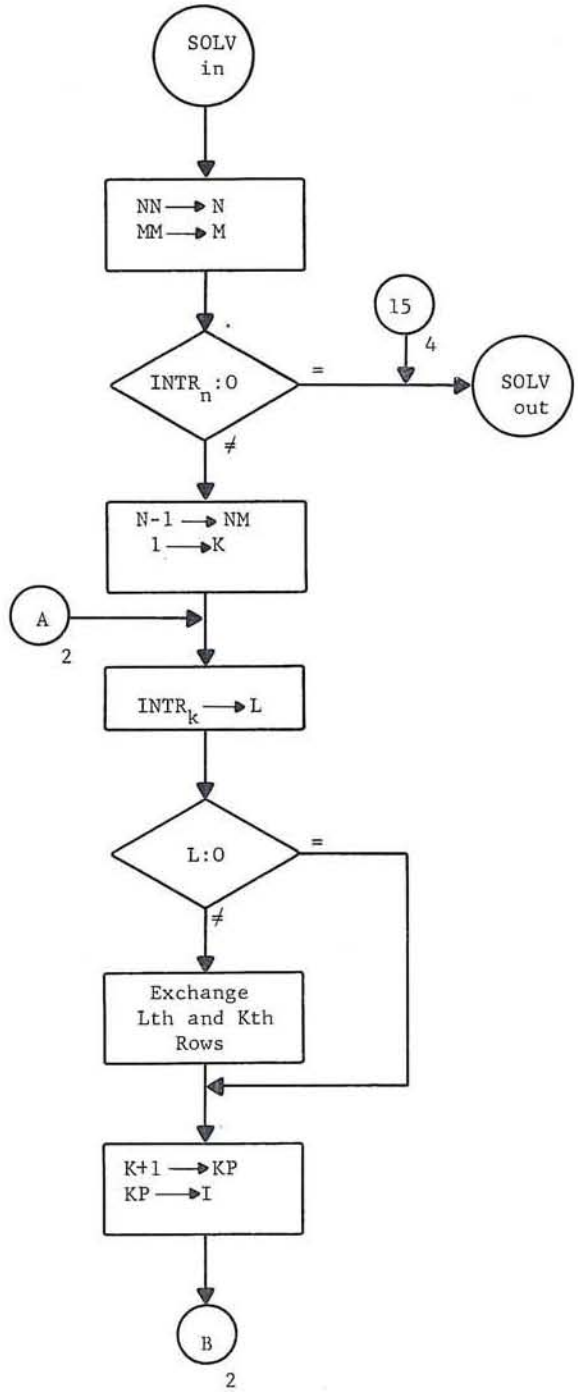
DECØM

(2)



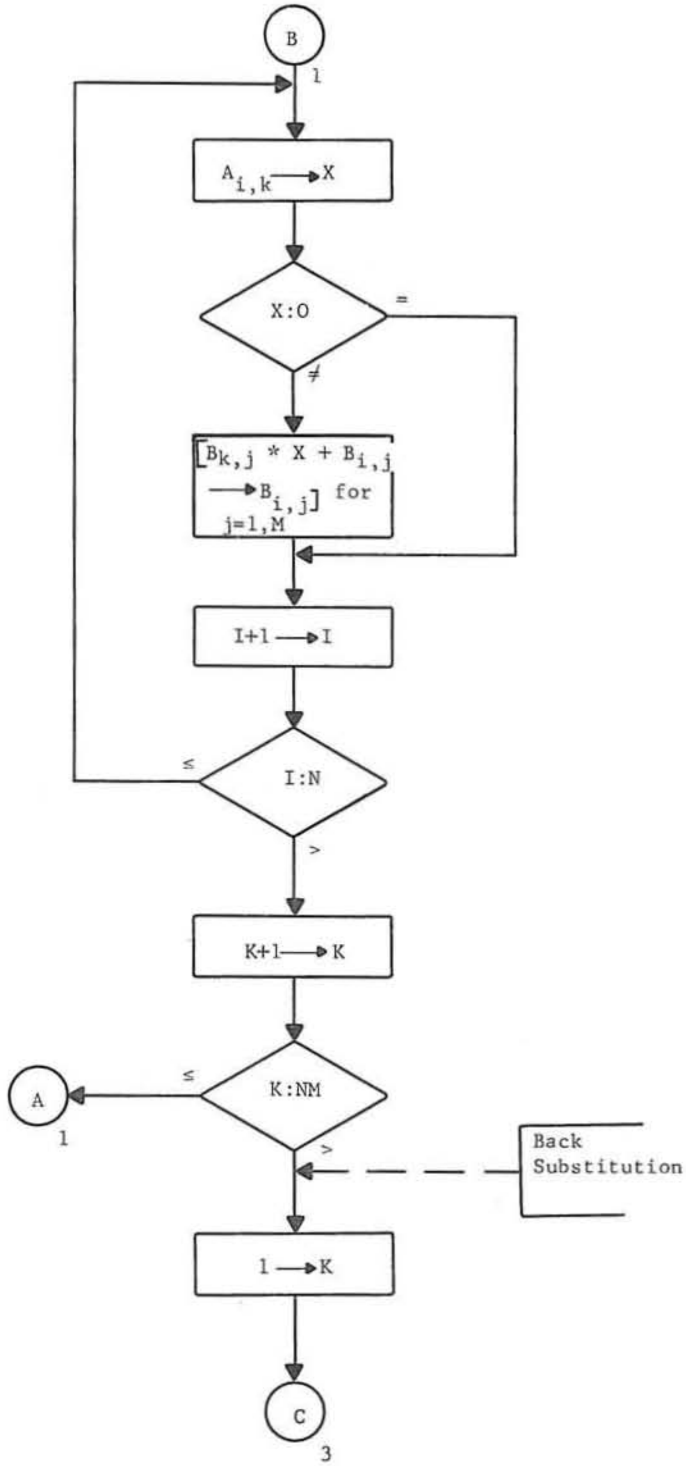
DECØM

3


SOLV
in
NN → N
MM → M
15
4
INTR
n
:0
=
SOLV
out
≠
N-1 → NM
1 → K
A
2
INTR
k
→ L
L:0
=
≠
Exchange
Lth and Kth
Rows
K+1 → KP
KP → I
B
2

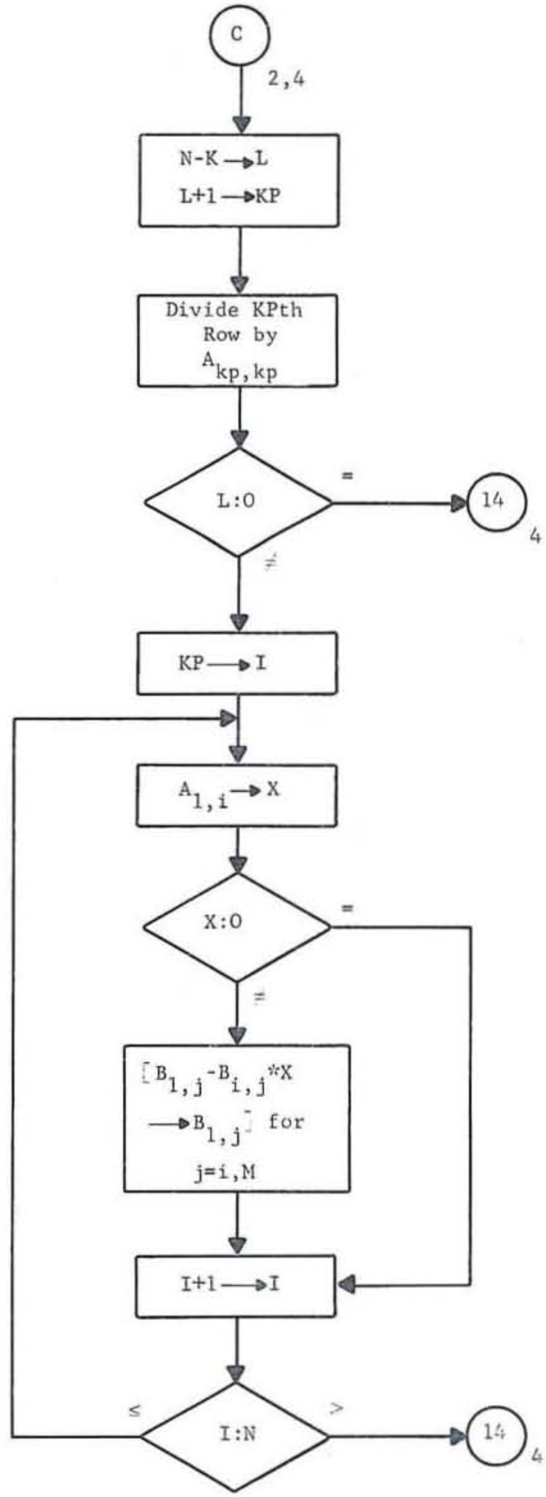
SØLV

(1)


B
1
A i,k → X
X:O
=
≠
[B k,j * X + B i,j → B i,j] for j=1,M
I+1 → I
≤
I:N
>
K+1 → K
≤
K:NM
>
A
1
Back Substitution
1 → K
C
3

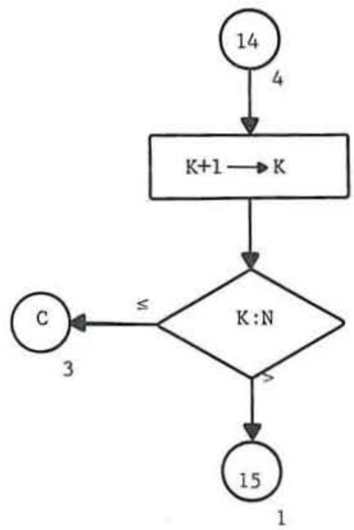
SØLV

(2)


C
2,4
N-K → L
L+1 → KP
Divide KPth Row by $A_{kp,kp}$
L:O
=
≠
14
4
KP → I
$A_{1,i}$ → X
X:O
=
≠
[$B_{1,j}$-$B_{i,j}$*X → $B_{1,j}$] for j=i,M
I+1 → I
I:N
≤
>
14
4

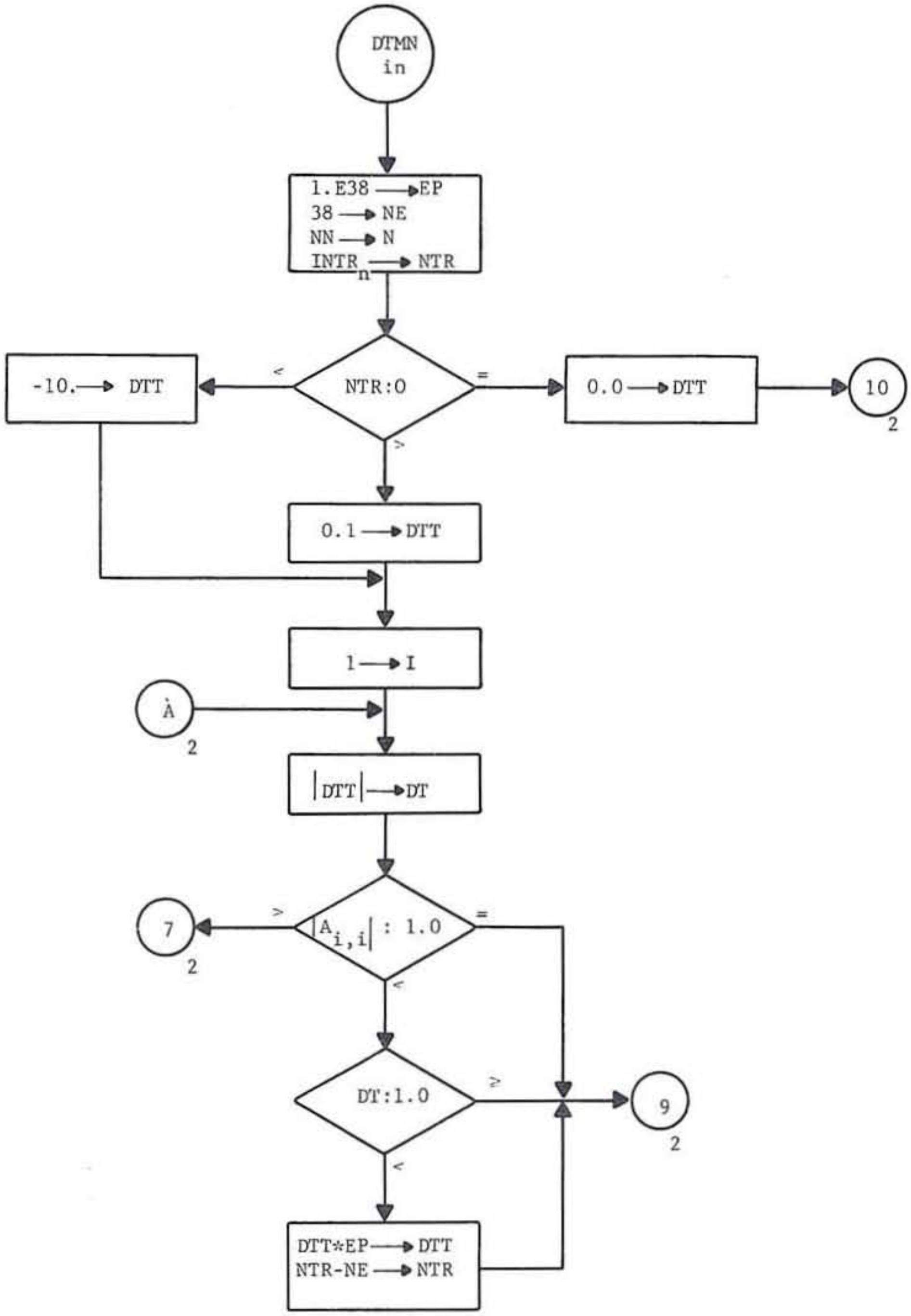
SØLV

(3)



SØLV

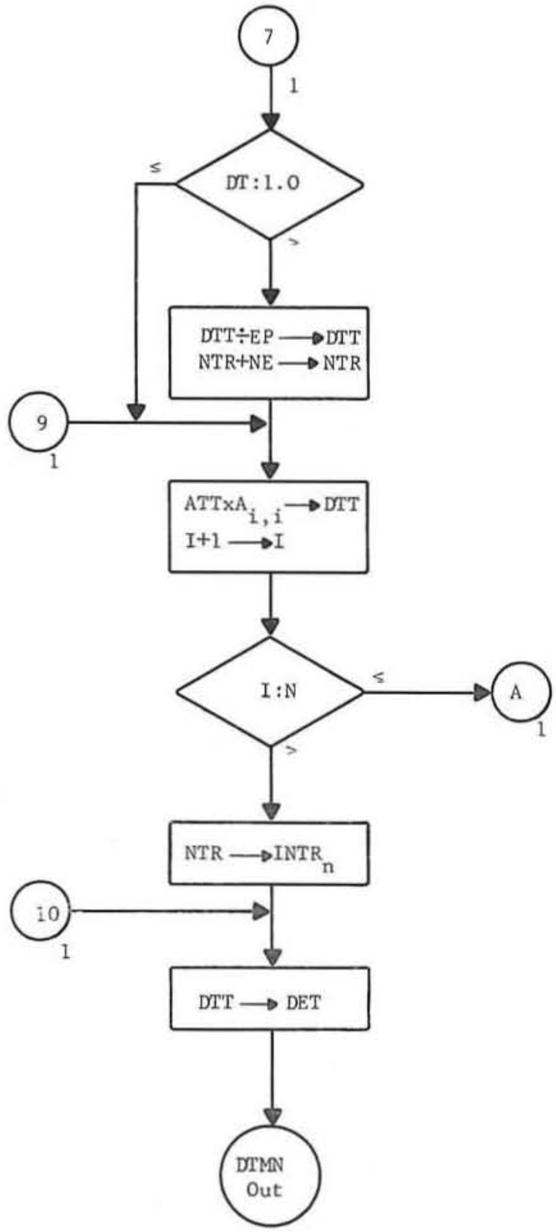
(4)



DTMN in
1.E38 → EP
38 → NE
NN → N
INTR_n → NTR
NTR:0
-10. → DTT
0.0 → DTT
10
2
0.1 → DTT
1 → I
A
2
|DTT| → DT
|A_i,i| : 1.0
7
2
DT:1.0
9
2
DTT*EP → DTT
NTR-NE → NTR

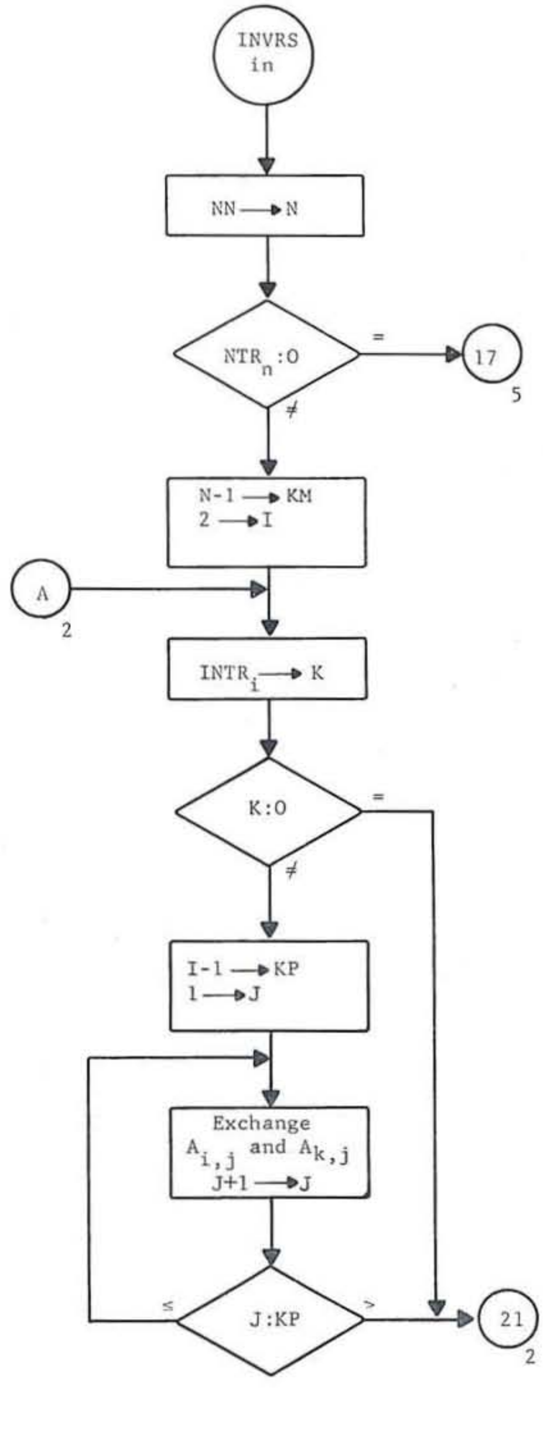
DTMN

(1)


7
1
DT:1.0
≤
>
DTT÷EP → DTT
NTR+NE → NTR
9
1
ATTxA$_{i,i}$ → DTT
I+1 → I
I:N
≤
>
A
1
NTR → INTR$_n$
10
1
DTT → DET
DTMN
Out

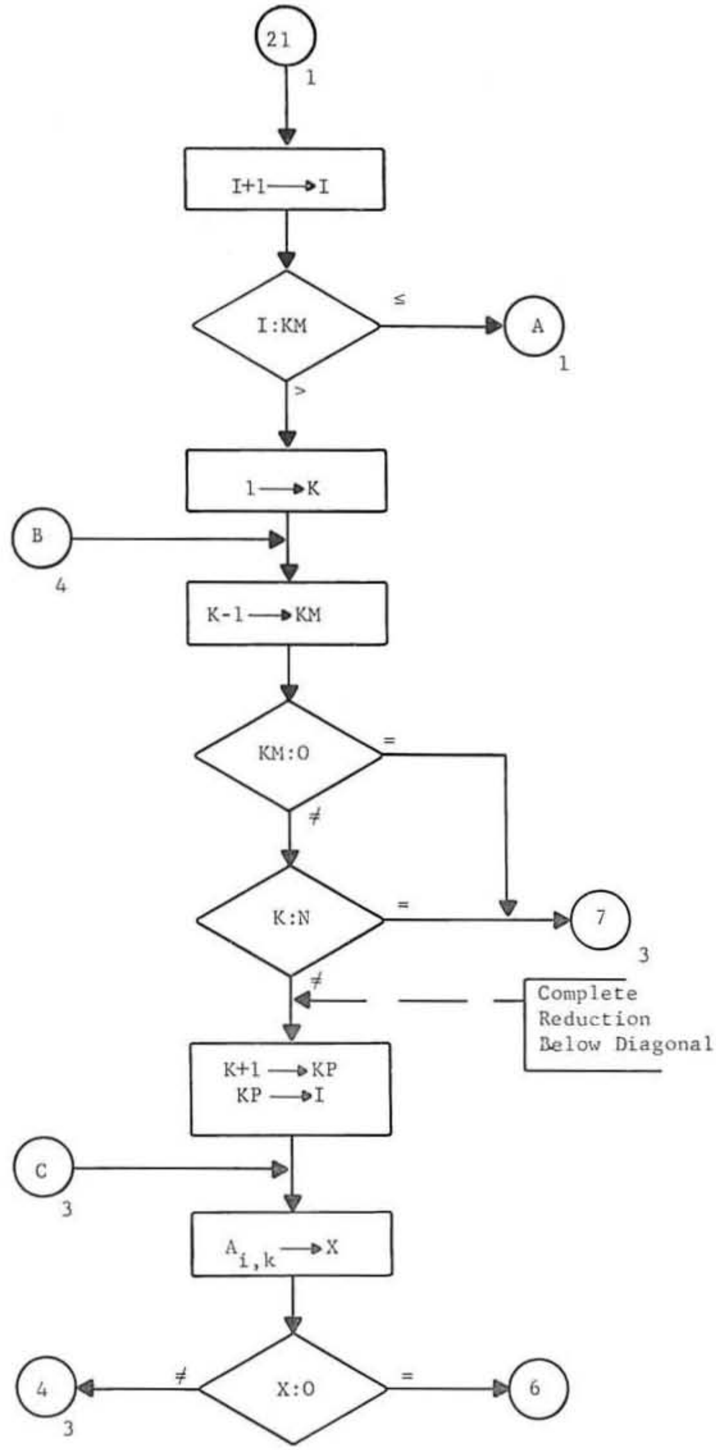
DTMN

(2)


INVRS
in
NN ⟶ N
NTR_n:0
=
17
5
≠
N-1 ⟶ KM
2 ⟶ I
A
2
$INTR_i$ ⟶ K
K:0
=
≠
I-1 ⟶ KP
1 ⟶ J
Exchange
$A_{i,j}$ and $A_{k,j}$
J+1 ⟶ J
J:KP
≤
>
21
2

INVRS

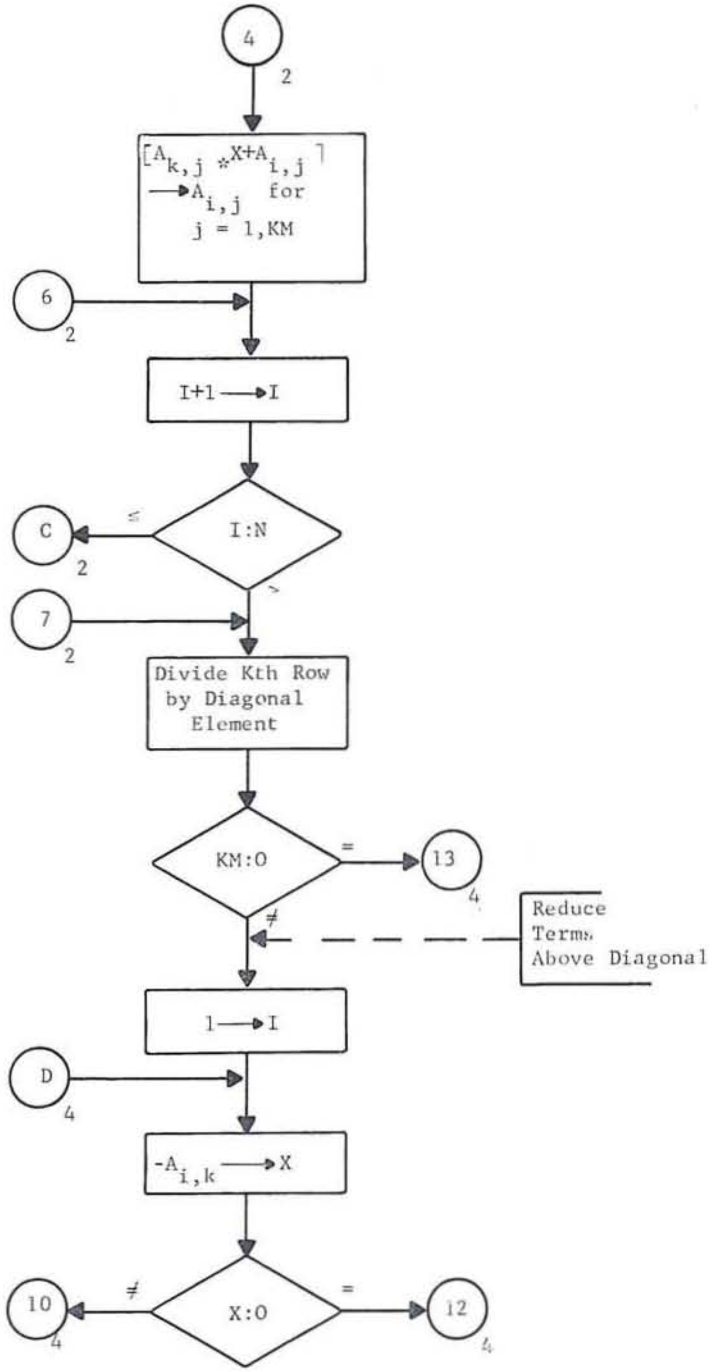
(1)



21
1
I+1 → I
I:KM
≤
A
1
>
1 → K
B
4
K-1 → KM
KM:0
=
≠
K:N
=
7
3
≠
Complete
Reduction
Below Diagonal
K+1 → KP
KP → I
C
3
$A_{i,k}$ → X
4
3
≠
X:0
=
6

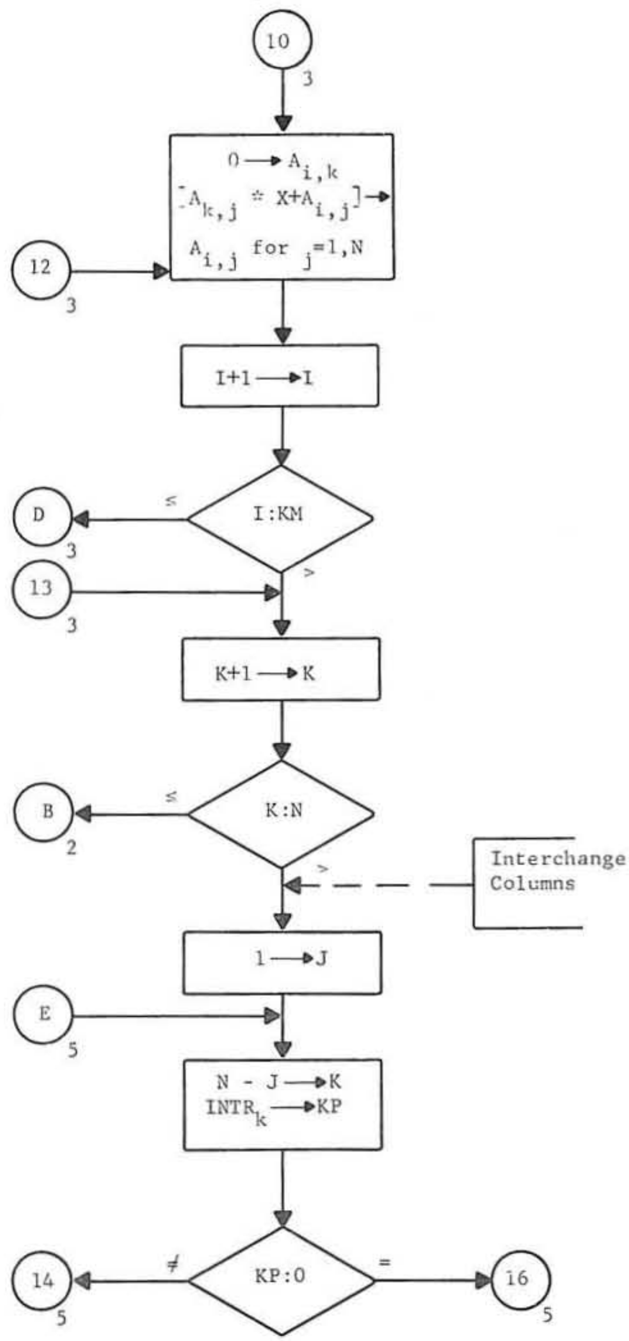
INVRS

(2)


4
2
[A k,j * X+A i,j] → A i,j for j = 1,KM
6
2
I+1 → I
≤
C
2
I:N
>
7
2
Divide Kth Row by Diagonal Element
KM:0
=
13
4
≠
Reduce Terms Above Diagonal
1 → I
D
4
-A i,k → X
X:0
≠
10
4
=
12
4

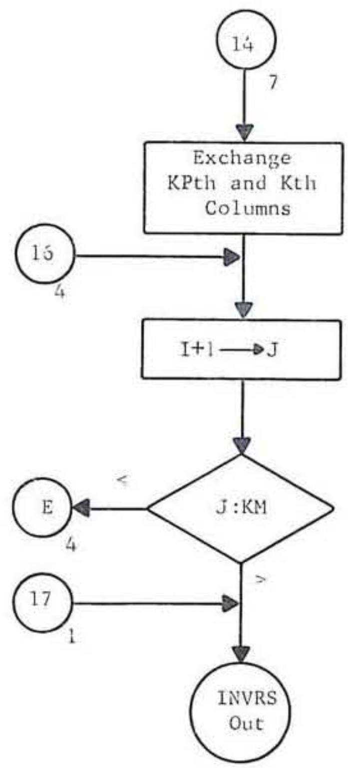
INVRS

(3)


10
3
0 → A i,k
[A k,j * X+A i,j] → A i,j for j=1,N
12
3
I+1 → I
I:KM
≤
>
D
3
13
3
K+1 → K
K:N
≤
>
B
2
Interchange Columns
1 → J
E
5
N - J → K
INTR k → KP
KP:0
≠
=
14
5
16
5

INVRS

(4)


14
7
Exchange KPth and Kth Columns
16
4
I+1 ——▶J
J:KM
<
E
4
>
17
1
INVRS Out

INVRS

5

SIMEQ
INPUT CODING FORM

Col
2

$P/NAME=60H____
,
CASE=60H____
,
TITLE=60H____
,
N=____, M=____, NEW=____$

$X/L___/C=___,___, ________, ________, ________, ________, ________,
________, ________, ________, ________, ________, ________$

$X/L___/C=___,___, ________, ________, ________, ________, ________,
________, ________, ________, ________, ________, ________$

$X/L___/C=___,___, ________, ________, ________, ________, ________,
________, ________, ________, ________, ________, ________$

$X/L___/C=___,___, ________, ________, ________, ________, ________,
________, ________, ________, ________, ________, ________$

$X/L___/C=___,___, ________, ________, ________, ________, ________,
________, ________, ________, ________, ________, ________$

$X/L___/C=___,___, ________, ________, ________, ________, ________,
________, ________, ________, ________, ________, ________$

$X/L___/C=___,___, ________, ________, ________, ________, ________,
________, ________, ________, ________, ________, ________$

$X/L___/C=___,___, ________, ________, ________, ________, ________,
________, ________, ________, ________, ________, ________$

$X/L___/C=___,___, ________, ________, ________, ________, ________,
________, ________, ________, ________, ________, ________$

$X/L___/C=___,___, ________, ________, ________, ________, ________,
________, ________, ________, ________, ________, ________$

$X/L___/C=___,___, ________, ________, ________, ________, ________,
________, ________, ________, ________, ________, ________$

$X/L___/C=___,___, ________, ________, ________, ________, ________,
________, ________, ________, ________, ________, ________$

$X/L___/C=___,___, ________, ________, ________, ________, ________,
________, ________, ________, ________, ________, ________$

$X/L___/C=___,___, ________, ________, ________, ________, ________,
________, ________, ________, ________, ________, ________$

$X/L___/C=___,___, ________, ________, ________, ________, ________,
________, ________, ________, ________, ________, ________$

$X/L___/C=___,___, ________, ________, ________, ________, ________,
________, ________, ________, ________, ________, ________$

Notes:
1. Values not entered are set to zero
2. Punch all data starting in column 2
3. Discontinue punching after handwritten $

SIMEQ
INPUT CODING FORM

Col
2

$P/NAME=60H____
,
CASE=60H____
,
TITLE=60H____
,
N=____, M=____, NEW=____$

$X/L___/C=___,___, ________, ________, ________, ________, ________,
________, ________, ________, ________, ________, ________$

$X/L___/C=___,___, ________, ________, ________, ________, ________,
________, ________, ________, ________, ________, ________$

$X/L___/C=___,___, ________, ________, ________, ________, ________,
________, ________, ________, ________, ________, ________$

$X/L___/C=___,___, ________, ________, ________, ________, ________,
________, ________, ________, ________, ________, ________$

$X/L___/C=___,___, ________, ________, ________, ________, ________,
________, ________, ________, ________, ________, ________$

$X/L___/C=___,___, ________, ________, ________, ________, ________,
________, ________, ________, ________, ________, ________$

$X/L___/C=___,___, ________, ________, ________, ________, ________,
________, ________, ________, ________, ________, ________$

$X/L___/C=___,___, ________, ________, ________, ________, ________,
________, ________, ________, ________, ________, ________$

$X/L___/C=___,___, ________, ________, ________, ________, ________,
________, ________, ________, ________, ________, ________$

$X/L___/C=___,___, ________, ________, ________, ________, ________,
________, ________, ________, ________, ________, ________$

$X/L___/C=___,___, ________, ________, ________, ________, ________,
________, ________, ________, ________, ________, ________$

$X/L___/C=___,___, ________, ________, ________, ________, ________,
________, ________, ________, ________, ________, ________$

$X/L___/C=___,___, ________, ________, ________, ________, ________,
________, ________, ________, ________, ________, ________$

$X/L___/C=___,___, ________, ________, ________, ________, ________,
________, ________, ________, ________, ________, ________$

$X/L___/C=___,___, ________, ________, ________, ________, ________,
________, ________, ________, ________, ________, ________$

$X/L___/C=___,___, ________, ________, ________, ________, ________,
________, ________, ________, ________, ________, ________$

Notes:
1. Values not entered are set to zero
2. Punch all data starting in column 2
3. Discontinue punching after handwritten $

SIMEQ
INPUT CODING FORM

Col
2

$P/NAME=60H ___

,
CASE=60H ___

,
TITLE=60H ___

,
N= ___ , M= ___ , NEW= ___ $

$X/L ___ /C= ___ , ___ , ___ , ___ , ___ , ___ , ___ ,

___ , ___ , ___ , ___ , ___ , ___ $

$X/L ___ /C= ___ , ___ , ___ , ___ , ___ , ___ , ___ ,

___ , ___ , ___ , ___ , ___ , ___ $

$X/L ___ /C= ___ , ___ , ___ , ___ , ___ , ___ , ___ ,

___ , ___ , ___ , ___ , ___ , ___ $

$X/L ___ /C= ___ , ___ , ___ , ___ , ___ , ___ , ___ ,

___ , ___ , ___ , ___ , ___ , ___ $

$X/L ___ /C= ___ , ___ , ___ , ___ , ___ , ___ , ___ ,

___ , ___ , ___ , ___ , ___ , ___ $

$X/L ___ /C= ___ , ___ , ___ , ___ , ___ , ___ , ___ ,

___ , ___ , ___ , ___ , ___ , ___ $

$X/L ___ /C= ___ , ___ , ___ , ___ , ___ , ___ , ___ ,

___ , ___ , ___ , ___ , ___ , ___ $

$X/L ___ /C= ___ , ___ , ___ , ___ , ___ , ___ , ___ ,

___ , ___ , ___ , ___ , ___ , ___ $

$X/L ___ /C= ___ , ___ , ___ , ___ , ___ , ___ , ___ ,

___ , ___ , ___ , ___ , ___ , ___ $

$X/L ___ /C= ___ , ___ , ___ , ___ , ___ , ___ , ___ ,

___ , ___ , ___ , ___ , ___ , ___ $

$X/L ___ /C= ___ , ___ , ___ , ___ , ___ , ___ , ___ ,

___ , ___ , ___ , ___ , ___ , ___ $

$X/L ___ /C= ___ , ___ , ___ , ___ , ___ , ___ , ___ ,

___ , ___ , ___ , ___ , ___ , ___ $

$X/L ___ /C= ___ , ___ , ___ , ___ , ___ , ___ , ___ ,

___ , ___ , ___ , ___ , ___ , ___ $

$X/L ___ /C= ___ , ___ , ___ , ___ , ___ , ___ , ___ ,

___ , ___ , ___ , ___ , ___ , ___ $

$X/L ___ /C= ___ , ___ , ___ , ___ , ___ , ___ , ___ ,

___ , ___ , ___ , ___ , ___ , ___ $

$X/L ___ /C= ___ , ___ , ___ , ___ , ___ , ___ , ___ ,

___ , ___ , ___ , ___ , ___ , ___ $

Notes:
1. Values not entered are set to zero
2. Punch all data starting in column 2
3. Discontinue punching after handwritten $

SIMEQ
INPUT CODING FORM

Col
2

$P/NAME=60H____
,
CASE=60H____
,
TITLE=60H____
,
N=____, M=____, NEW=____$

$X/L___/C=___,___, ______, ______, ______, ______, ______,
______, ______, ______, ______, ______, ______$

$X/L___/C=___,___, ______, ______, ______, ______, ______,
______, ______, ______, ______, ______, ______$

$X/L___/C=___,___, ______, ______, ______, ______, ______,
______, ______, ______, ______, ______, ______$

$X/L___/C=___,___, ______, ______, ______, ______, ______,
______, ______, ______, ______, ______, ______$

$X/L___/C=___,___, ______, ______, ______, ______, ______,
______, ______, ______, ______, ______, ______$

$X/L___/C=___,___, ______, ______, ______, ______, ______,
______, ______, ______, ______, ______, ______$

$X/L___/C=___,___, ______, ______, ______, ______, ______,
______, ______, ______, ______, ______, ______$

$X/L___/C=___,___, ______, ______, ______, ______, ______,
______, ______, ______, ______, ______, ______$

$X/L___/C=___,___, ______, ______, ______, ______, ______,
______, ______, ______, ______, ______, ______$

$X/L___/C=___,___, ______, ______, ______, ______, ______,
______, ______, ______, ______, ______, ______$

$X/L___/C=___,___, ______, ______, ______, ______, ______,
______, ______, ______, ______, ______, ______$

$X/L___/C=___,___, ______, ______, ______, ______, ______,
______, ______, ______, ______, ______, ______$

$X/L___/C=___,___, ______, ______, ______, ______, ______,
______, ______, ______, ______, ______, ______$

$X/L___/C=___,___, ______, ______, ______, ______, ______,
______, ______, ______, ______, ______, ______$

$X/L___/C=___,___, ______, ______, ______, ______, ______,
______, ______, ______, ______, ______, ______$

$X/L___/C=___,___, ______, ______, ______, ______, ______,
______, ______, ______, ______, ______, ______$

Notes:
1. Values not entered are set to zero
2. Punch all data starting in column 2
3. Discontinue punching after handwritten $

SIMEQ
INPUT CODING FORM

Col
2

$P/NAME=60H____
,
CASE=60H____
,
TITLE=60H____
,
N=____, M=____, NEW=____$

$X/L___/C=___,___, ________, ________, ________, ________, ________,
________, ________, ________, ________, ________, ________$

$X/L___/C=___,___, ________, ________, ________, ________, ________,
________, ________, ________, ________, ________, ________$

$X/L___/C=___,___, ________, ________, ________, ________, ________,
________, ________, ________, ________, ________, ________$

$X/L___/C=___,___, ________, ________, ________, ________, ________,
________, ________, ________, ________, ________, ________$

$X/L___/C=___,___, ________, ________, ________, ________, ________,
________, ________, ________, ________, ________, ________$

$X/L___/C=___,___, ________, ________, ________, ________, ________,
________, ________, ________, ________, ________, ________$

$X/L___/C=___,___, ________, ________, ________, ________, ________,
________, ________, ________, ________, ________, ________$

$X/L___/C=___,___, ________, ________, ________, ________, ________,
________, ________, ________, ________, ________, ________$

$X/L___/C=___,___, ________, ________, ________, ________, ________,
________, ________, ________, ________, ________, ________$

$X/L___/C=___,___, ________, ________, ________, ________, ________,
________, ________, ________, ________, ________, ________$

$X/L___/C=___,___, ________, ________, ________, ________, ________,
________, ________, ________, ________, ________, ________$

$X/L___/C=___,___, ________, ________, ________, ________, ________,
________, ________, ________, ________, ________, ________$

$X/L___/C=___,___, ________, ________, ________, ________, ________,
________, ________, ________, ________, ________, ________$

$X/L___/C=___,___, ________, ________, ________, ________, ________,
________, ________, ________, ________, ________, ________$

$X/L___/C=___,___, ________, ________, ________, ________, ________,
________, ________, ________, ________, ________, ________$

$X/L___/C=___,___, ________, ________, ________, ________, ________,
________, ________, ________, ________, ________, ________$

Notes:
1. Values not entered are set to zero
2. Punch all data starting in column 2
3. Discontinue punching after handwritten $

Progress Is Our Most Important Product

GENERAL ELECTRIC

COMPUTER DEPARTMENT • PHOENIX, ARIZONA

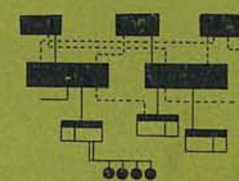
LITHO U.S.A.

GENERAL ELECTRIC
COMPUTERS

GE-625/635 Math Routines

DIFFE

DIFFE



GENERAL ELECTRIC

CPB-1168

GE-625/635
MATH ROUTINES
DIFFE

SIMULTANEOUS LINEAR DIFFERENTIAL EQUATIONS

Program Number
CD600D8.001

October 1965

GENERAL ELECTRIC
COMPUTER DEPARTMENT

CREDITS

The source material used in this manual is taken from a document, published by General Electric, titled DIFFE--Computer Program for Solution of a System of N First-Order Ordinary (Linear or Nonlinear) Differential Equations, by R. G. Claussen. Permission to use the original document was given by D. L. Shell, Manager, Computer Applications and Processing, G-E's Telecommunications and Information Processing Department.

Comments on this publication may be addressed to Technical Publications, Computer Department, General Electric Company, P. O. Box 2961, Phoenix, Arizona, 85002.

INSTRUCTIONS

Clip and add the following note to pages 18 and 20 of the GE-625/635 Math Routines - DIFFE, CPB-1168.

Page 18.

Special Use of NAMELIST Input

In this case, the first two lines of identification are retained from the previous case (page 16). The 60H on the third line of the form is changed to 59H to allow the input with the additional dollar sign to be placed within 72 columns (see NAMELIST in the GE-625/635 FORTRAN IV Reference Manual, CPB-1006.)

Page 20.

Special Use of NAMELIST Input

In this case, the first two lines of identification are retained from the previous case (page 16). The 60H on the third line of the form is changed to 59H to allow the input with the additional dollar sign to be placed within 72 columns (see NAMELIST in the GE-625/635 FORTRAN IV Reference Manual, CPB-1006.

TIB DISPOSITION

The revised pages will appear in the next edition of GE-625/635 Math Routines-DIFFE, CPB-1168.

CONTENTS

APPENDICES

1. INTRODUCTION

DIFFE is a program for the solution of a system of N first-order ordinary (linear or nonlinear) differential equations. Equations to be solved are written in the FORTRAN language, observing certain minor rules. Nth order equations are written as N first-order equations. Input data, such as initial conditions, error bounds, and values of the independent variable at which print-out is required are entered on a simple input sheet.

Some of the features of the program are: automatic restart when singularities are encountered in the dependent variable calculations, negative integration, relative error bounds, easily coded input sheets and a simple output format.

2. RESTART

A signal is set to tell the main program when a singularity occurs in the calculation of the dependent variables. Then, depending upon the direction of integration, new initial conditions are set up and integration is continued.

Suppose singularity at x_i. The values of y (dependent variables) at x_i are assumed to be the conditions at x_o $(x_i + E)$

$$E = (x_f - x_i)\ (0.1)\ (ERI)$$

where ERI is the number of times singularity occurred at x_i. The present limit on this number is 3.

The calculated values of the dependent variables (y) are compared with the predicted values (yp) and, if each agrees within EMAX, the calculated value is assumed correct.

If EMAX is positive: $|yp - y| < |EMAX|$ to pass.

If EMAX is negative: $|yp - y| < |EMAX \cdot y|$ to pass.

Note: When y goes to zero, care must be taken in using relative error bounds.

3. MATHEMATICAL METHOD

DIFFE is programmed using the Adams-Moulton method as modified by Shell. This is a polynomial predictor-corrector method in which the interval size is automatically controlled by desired accuracy.

Since the Adams-Moulton method requires several starting values, the integration is initiated by using a special start-up procedure to obtain the first set of derivatives. Then, calling the user's routine for derivative calculations, the program predicts and corrects a further point and checks the result against the given tolerance. If more accuracy is needed, the program reduces the interval size and tries again. If excess accuracy is found, the program increases the interval size. The calculation proceeds in this manner until the given final value of the independent variable is reached.

The Adams-Moulton method is discussed in "Advanced Calculus for Engineers" by F. B. Hildebrand, Prentice Hall, 1948.

4. EXPLANATION OF INPUT SHEETS

There are three input sheets; one describes the differential equations, and two are devoted to program control, covering initial conditions and constants, respectively.

In filling out the sheets, the independent variable is X and the dependent variable is Y. Therefore, all equations are to be in the $\frac{dy}{dx}$ format. For the following equation:

$$\frac{dx}{dt} + x = 0$$

simply substitute and get

$$\frac{dy}{dx} + y = 0$$

INPUT SHEET 1--FORTRAN DIFFERENTIAL EQUATIONS

The differential equations used in the program must be written in FORTRAN notation observing these rules:

1. Signs are denoted as follows:

plus	+
minus	-
multiplication	*
division	/
exponentiation	**

2. F is the value of the derivative.

3. Y is the dependent variable.

Thus,

$$F(1) = \frac{dy}{dx}$$

$$F(2) = \frac{d^2y}{dx^2}$$

$$F(N) = \frac{d^ny}{dx^n}$$

and

$$Y(1) = Y$$

$$Y(2) = \frac{dy}{dx}$$

$$Y(3) = \frac{d^2y}{dx^2}$$

$$Y(N) = \frac{d^{n-1}y}{dx^{n-1}}$$

As shown later in the example equation

$$\frac{d^2x}{dt^2} + \frac{dx}{dt} + x + x^2 + x^3 = \sin t$$

This may be rewritten in one of the following two ways:

$$\frac{d^2y}{dx} + \frac{dy}{dx} + y + y^2 + y^3 = \sin x$$

or

$$\frac{d^2y}{dx} = -\frac{dy}{dx} - y - y^2 - y^3 + \sin x$$

Then, using the FORTRAN notation, the above equation can be written as two first-order equations as follows:

```
F (1)  =  Y (2)
F (2)  =  -Y (2)  -  Y (1)  -  Y (1)**2  -  Y (1)**3  +  SIN (x)
```

INPUT SHEETS D1 AND D2

The input sheet formats are shown below.

Input Sheet D1

ORDER must be filled in and it signifies the highest order of the equation or the number of first-order equations. SIZMAX* and SIZMIN are optional. If they are not filled in, they are set to 10^{-3} and 10^{-10}, respectively. XO and as many YO's as necessary (depending on the size of ORDER) must be filled in. As many EMAX's as necessary must be filled in.

$/NAME=60H	(Alphanumeric identification)	Comma denotes separate card.
, ADDRES=60H	(More identification)	
, IDENT=60H	(Further identification)	

*If SIZMAX is negative, integration will be negative.

ORDER= , SIZMAX= , SIZMIN= ,
XO= ,
YO= , , , , , (as needed)
EMAX= , , , , (as needed)

Input Sheet D2

The step size for printout may be selected in two ways, constant step or variable step. DELTA is the constant step size terminated at FINAL. If DELTA = 0, the values found in VAR are used. The last VAR terminates integration.

```
DELTA=        , FINAL=         ,
VAR=      ,       ,      ,       ,     ,  (as needed)
A=        ,       ,      ,       ,  (optional constants for
                                     the equations)
IOF=      $    (T if last case, F if more cases follow.)
```

Input Coding

After the input sheets are completed, the program is run as a FORTRAN compile and execute job. Of course, the binary deck from the equations can be retained for further use with new input sheets D1 and D2.

As an example, the FORTRAN coding for the previous example is shown:

```
SUBROUTINE DE (X, Y, F)
DIMENSION Y (25), F(25)
COMMON A(200)
F(1)=Y(2)
F(2)=-Y(2)-Y(1)-Y(1)**2-Y(1)**3+A(1)*SIN(X)
RETURN
END
```

In this example, the term A(1) is used because two cases are to be solved, the previous example and a similar equation without sin x. On the input sheets for one case A(1) is entered as 1, and for the other, as 0.

5. PRODUCTION DECK SETUP

The following deck setup is used for compiling and executing the DIFFE program:

```
$       IDENT
$       COMMENT
$       OPTION   FORTRAN
$       FORTRAN  DECK , LSTOU
$       INCODE   IBMF
```

Binary Deck (main program, other subroutines)
DE Subroutine deck (input sheet 1)

```
$       EXECUTE
```

(input sheets D1, and D2)

```
$       ENDJOB
***EOF
```

The following deck setup is used for executing a previously compiled DE program:

```
$       IDENT
$       COMMENT
$       OPTION   FORTRAN
```

BINARY DECK (main program, other subroutines including DE)

```
$       EXECUTE
```

(input sheets D1, D2)

```
$       ENDJOB
***EOF
```

6. TEST CASE EXAMPLES

The following examples show that the same derivative (DE) subroutine can be used for several different equations:

Eq. 1) $\frac{d^2x}{dt^2} + \frac{dx}{dt} + x + x^2 + x^3 = \sin(t)$

Eq. 2) $\frac{d^2x}{dt^2} + \frac{dx}{dt} + x + x^2 + x^3 = 0$

Note that all terms in the equations above are alike except those to the right of the equals sign. The equation can, therefore, be determined by controlling A in the term A sin(t).

$$\frac{d^2x}{dt^2} + \frac{dx}{dt} + x + x^2 + x^3 = A \sin(t)$$

Thus, in equation 1) A = 1 and in equation 2) A = 0

Example (1)

Solve the differential equation

$$\frac{d^2x}{dt^2} + \frac{dx}{dt} + x + x^2 + x^3 = \sin t,$$

with the initial condition that at t = 0

$$x = 0 \text{ and } \frac{dx}{dt} = 0$$

The required values of the independent variables are:

VAR = 0.878, 1.216, 1.469, 1.700, 1.941, 2.262, 2.589, 2.911, 3.200, 3.438, 3.626, 3.795, 3.953, 4.105, 4.253, 4.397, 4.538, 4.678, 4.819, 5.106, 5.938, 6.220

Thus, for this example,

ORDER = 2

SIZMAX = 1/-3 (10^{-3})

SIZMIN = 1/-10 (10^{-10})

DELTA = 0

FINAL = 6.220

XO = 0

EMAX = 1/-7, 1/-7 (10^{-7})

YO = 0, 0 $(x = 0, \frac{dx}{dt} = 0)$

A = 1.0

Example (2)

Solve the differential equation

$$\frac{d^2x}{dt^2} + \frac{dx}{dt} + x + x^2 + x^3 = 0,$$

with the initial condition that at t = 0

$$x = 1 \text{ and } \frac{dx}{dt} = 0$$

The required values of the independent variable are:

VAR = 0.290, 0.418, 0.527, 0.629, 0.729, 0.829, 0.933, 1.043, 1.163, 1.297, 1.451, 1.635, 1.875, 2.295, 2.643, 3.069, 3.741, 4.260, 4.812, 5.552, 6.797

Thus, for this example,

ORDER = 2

SIZMAX = 1/-3 (10^{-3})

SIZMIN = 1/-10 (10^{-10})

DELTA = 0

FINAL = 6.797

XO = 0

EMAX = 1/-7, 1/-7 (10^{-7})

YO = 1.0, 0 $(x = 1, \frac{dx}{dt} = 0)$

A = 0

SAMPLE INPUT FORMS

GENERAL ELECTRIC FORTRAN CODING FORMS

PROBLEM DIFFE INPUT (EQUATIONS)

PROGRAMMER DATE PAGE OF

Statement No. (1–5)	Continuation (6)	FORTRAN STATEMENT (7–)
		SUBROUTINE DE(X, Y, F)
		COMMON A
		DIMENSION Y(25), F(25)
		F(1) = Y(2)
		F(2) = -Y(2) -Y(1) - Y(1)**2 -Y(1)**3 + A(1)*SIN(X)
		RETURN
		END

DIFFE INPUT (SHEET D1)

$S/NAME=60H DIFFE TEST CASE

,ADDRES=60H USING THE TEST CASES FROM EVENDALE WRITEUP

,IDENT=60H EXAMPLE 1

,ORDER= 2, SIZMAX= 1.E-3, SIZMIN= 1.E-10,

XO= 0,

YO= 0, 0, ___, ___, ___,

EMAX= 2*1.E-7, ___, ___, ___, ___,

DIFFE INPUT (SHEET D2)

DELTA= ______________, FINAL= ______________,

VAR= .878, 1.216, 1.469, 1.7, 1.941, 2.262,

2.589	2.911	3.22	3.438	3.626	3.795
3.953	4.105	4.253	4.397	4.538	4.678
4.819	5.106	5.43	5.938	6.22	

A= 1.0

IOF= F $

T if last case
F is not last case

DIFFE INPUT (SHEET D1)

$S/NAME=60H ______________________________

,
ADDRES=60H ______________________________

,
$S/IDENT=~~60H~~ 59H   EXAMPLE 1B

,
ORDER= 2 , SIZMAX= 1.E-3 , SIZMIN= 1.E-10,

XO= 0 ,

YO= 0 , 0 , ______ , ______ , ______ ,
______ , ______ , ______ , ______ , ______ ,
______ , ______ , ______ , ______ , ______ ,
______ , ______ , ______ , ______ , ______ ,
______ , ______ , ______ , ______ , ______ ,

EMAX= 2*1.E-7 , ______ , ______ , ______ , ______ ,
______ , ______ , ______ , ______ , ______ ,
______ , ______ , ______ , ______ , ______ ,
______ , ______ , ______ , ______ , ______ ,
______ , ______ , ______ , ______ , ______ ,

DIFFE INPUT (SHEET D2)

DELTA= 1.0 , FINAL= 25 ,

VAR= ______ , ______ , ______ , ______ , ______ , ______ ,

A= 1 , ______ , ______ , ______ , ______ , ______ ,

IOF= F $

T if last case
F if not last case

$S/NAME=60H ________________________________
,
ADDRES=60H ________________________________
,
$S/IDENT=~~60H~~ 59H EXAMPLE 2
,
ORDER= 2 , SIZMAX= 1.E-3 , SIZMIN= 1. E-10,

XO= 0 ,

YO= 1.0 , 0 , ____ , ____ , ____ ,
____ , ____ , ____ , ____ , ____ ,
____ , ____ , ____ , ____ , ____ ,
____ , ____ , ____ , ____ , ____ ,
____ , ____ , ____ , ____ , ____ ,

EMAX= 1.E-7 , 1.E-7 , ____ , ____ , ____ ,
____ , ____ , ____ , ____ , ____ ,
____ , ____ , ____ , ____ , ____ ,
____ , ____ , ____ , ____ , ____ ,
____ , ____ , ____ , ____ , ____ ,

DIFFE INPUT (SHEET D2)

DELTA=________, FINAL=________,

VAR= .29 , .418 , .527 , .629 , .729 , .829 ,
.933 , 1.043 , 1.163 , 1.297 , 1.451 , 1.635 ,
1.875 , 2.295 , 2.643 , 3.069 , 3.741 , 4.26 ,
4.812 , 5.552 , 6.797 , ____ , ____ , ____ ,

A= 0 , ____ , ____ , ____ , ____ , ____ ,

IOF= T $

T if last case
F if not last case

LISTING OF INPUT CARDS

```
*DE                                  DE ROUTINE FOR DIFFEE TEST CASE          DE  0020
*                                    CD600.XX       DATE 05/06/65             DE  0030
      SUBROUTINE DE(X,Y,F)                                                    DE  0040
      COMMON A(200)                                                           DE  0050
      DIMENSION Y(25),F(25)                                                   DE  0060
      F(1)=Y(2)                                                               DE  0070
      F(2)=-Y(2)-Y(1)-Y(1)**2-Y(1)**3+A(1)*SIN(X)                             DE  0080
      RETURN                                                                  DE  0090
      END                                                                     DE  0100
$       EXECUTE
$       INCODE  IBMF
 $S/NAME=60HDIFFE TEST CASE
 ,
 ADDRES=60HUSING THE TEST CASES FROM EVENDALE WRITE-UP
 ,
 IDENT=60HEXAMPLE 1
 ,
 ORDER=2,SIZMAX=1.E-3,SIZMIN=1.E-10,
 XO=0,YO=0,0,EMAX=2*1.E-7,
 VAR=.878,1.216,1.469,1.7,1.941,2.262,2.589,2.911,3.22,3.438,3.626,
 3.795,3.953,4.105,4.253,4.397,4.538,4.678,4.819,5.106,5.43,5.938,6.22,
 A=1.0,
 IOF=F$
 $S/IDENT=54HEXAMPLE 1B
 ,
 ORDER=2,SIZMAX=1.E-3,SIZMIN=1.E-10,
 XO=0,YO=0,0,EMAX=2*1.E-7,
 DELTA=1.0,FINAL=25,
 A=1,
 IOF=F$
 $S/IDENT=54HEXAMPLE 2
 ,
 ORDER=2,SIZMAX=1.E-3,SIZMIN=1.E-10,
 XO=0,YO=1.0,0,EMAX=1.E-7,1.E-7,
 VAR=.29,.418,.527,.629,.729,.829,.933,1.043,1.163,1.297,1.451,1.635,
 1.875,2.295,2.643,3.069,3.741,4.26,4.812,5.552,6.797,
 A=0,
 IOF=T$
$        ENDJOB
```

```
                SOLUTION OF A SYSTEM OF ORDINARY LINEAR OR          PAGE  1
                   NON=LINEAR DIFFERENTIAL EQUATIONS

   DATE OF RUN    052365

        DIFFE TEST CASE

        USING THE TEST CASES FROM EVENDALE WRITE-UP

        EXAMPLE 1

 *    *    *    *    *    *    *    INPUT   DATA    *    *    *    *    *    *    *


      NUMBER OF EQATIONS,          2
      MAXIMUM STARTING INTERVAL SIZE     (SIZMAX)           1.00000E-03
      MINIMUM PERMITTED INTERVAL SIZE    (SIZMIN)           1.00000E-10

 INITIAL VALUE OF THE INDEPENDENT VARIABLE         0.
      INITIAL CONDITIONS OF THE DEPENDENT VARIABLES

       1)     0.                  2)     0.


       ERROR BOUNDS OF THE DEPENDENT VARIABLES

       1)     1.00000E=07         2)     1.00000E-07


       INTEGRATE  FROM            0.                  AT THESE VALUES
       1)     8.78000E=01         2)     1.21600E 00          3)     1.46900E 00
       4)     1.70000E 00         5)     1.94100E 00          6)     2.26200E 00
       7)     2.58900E 00         8)     2.91100E 00          9)     3.22000E 00
      10)     3.43800E 00        11)     3.62600E 00         12)     3.79500E 00
      13)     3.95300E 00        14)     4.10500E 00         15)     4.25300E 00
      16)     4.39700E 00        17)     4.53800E 00         18)     4.67800E 00
      19)     4.81900E 00        20)     5.10500E 00         21)     5.43000E 00

      22)     5.93800E 00        23)     6.22000E 00

              CONSTANTS USED IN THE   DE  SUBROUTINE  EQUATIONS

       1)     1.00000E 00
```

APPENDIX A
PROGRAM LISTING

```
***EOF
$       IDENT    AAV,GE,FORTRAN,DIFFE                                           DIFF0000
$       OPTION   FORTRAN,GO                                                     DIFF0010
$       FORTRAN  LSTOU,DECK,STAB                                                DIFF0020
$       INCODE   IBMF                                                           DIFF0030
*DIFFE           DIFFERENTIAL EQUATIONS SOLVER                                  DIFF0040
*                                     CD600D7.002      DATE 05/06/65            DIFF0050
      COMMON A(200)                                                             DIFF0060
      DIMENSION YO(25),EMAX(25),Y(25),VAR(300)                                  DIFF0070
      DIMENSION NAME(10),ADDRES(10),IDENT(10)                                   DIFF0080
      LOGICAL IOF                                                               DIFF0090
      INTEGER HOURS,SECNDS,STIME,SSEC,STIMEP,SSECP,STIMEL,SSECL,STIMEN          DIFF0100
      INTEGER SSECN,ETIME,ESEC                                                  DIFF0110
      NAMELIST/S/NAME,ADDRES,IDENT,ORDER,EMAX,SIZMAX,SIZMIN,XO,YO,DELTA,DIFF0120
     1FINAL,VAR,A,N,HO,HMIN,IOF                                                 DIFF0130
      CALL EFMOUT                                                               DIFF0140
      NER=3                                                                     DIFF0150
      CALL CLOCK(DATE,HOURS,SECNDS)                                             DIFF0160
      STIME=HOURS                                                               DIFF0170
      SSEC=SECNDS                                                               DIFF0180
      IP=0                                                                      DIFF0190
      CALL MAB(BITS)                                                            DIFF0200
      SIZMAX=1.E-3                                                              DIFF0210
      SIZMIN=1.E-10                                                             DIFF0220
    1 NA=0                                                                      DIFF0230
      NV=0                                                                      DIFF0240
```

```
      CALL CLOCK(DATE,HOURS,SECNDS)                                    DIFF0250
      STIMEP=HOURS                                                     DIFF0260
      SSECP=SECNDS                                                     DIFF0270
      STIMEL=HOURS                                                     DIFF0280
      SSECL=SECNDS                                                     DIFF0290
      DO 405 I=1,200                                                   DIFF0300
  405 A(I)=BITS                                                        DIFF0310
      DO 406 I=1,300                                                   DIFF0320
  406 VAR(I)=BITS                                                      DIFF0330
      DELTA=BITS                                                       DIFF0340
      IER=0                                                            DIFF0350
      IOF=.FALSE.                                                      DIFF0360
      READ(5,S)                                                        DIFF0370
      DO 407 I=1,200                                                   DIFF0380
      IF(A(I).EQ.BITS)GO TO 408                                        DIFF0390
  407 CONTINUE                                                         DIFF0400
  408 NA=I-1                                                           DIFF0410
      IF(DELTA.EQ.BITS)GO TO 411                                       DIFF0420
  420 IF(DELTA)412,411,412                                             DIFF0430
  411 DELTA=0.0                                                        DIFF0440
      DO 409 I=1,300                                                   DIFF0450
      IF(VAR(I).EQ.BITS)GO TO 410                                      DIFF0460
  409 CONTINUE                                                         DIFF0470
  410 NV=I-1                                                           DIFF0480
      IF(NV)2017,2017,2040                                             DIFF0490
 2040 CONTINUE                                                         DIFF0500
      FINAL=VAR(NV)                                                    DIFF0510
      GO TO 2042                                                       DIFF0520
  412 IF(DELTA)2045,411,2046                                           DIFF0530
 2045 IF(XO+DELTA-FINAL)2017,2017,2042                                 DIFF0540
 2046 IF(XO+DELTA-FINAL)2042,2017,2017                                 DIFF0550
 2042 N=ORDER                                                          DIFF0560
```

```
      NLL=ORDER/3.+.9
      IF(N.GT.10)GO TO 2017
      HO=SIZMAX
      HMIN=SIZMIN
      IF(N.GT.6)GO TO 41
      K=N
      GO TO 42
   41 K=6
   42 IP=IP+1
      WRITE(6,100)IP,DATE,NAME,ADDRES,IDENT
  100 FORMAT(1H1,14X,42HSOLUTION OF A SYSTEM OF ORDINARY LINEAR OR,8X,
     A4HPAGE,I3/18X,33HNON-LINEAR DIFFERENTIAL EQUATIONS//2X,11HDATE OF
     BRUN,3X,A6///8X,10A6//8X,10A6//8X,10A6///)
      WRITE(6,99)N,HO,HMIN,XO,(I,YO(I),I=1,N)
   99 FORMAT(1X,7(4H*    ),12HINPUT   DATA,7(4H    *)///6X,19HNUMBER OF EQ
     1ATIONS,I8/6X,41HMAXIMUM STARTING INTERVAL SIZE    (SIZMAX)1PE17.5/6
     2X,41HMINIMUM PERMITTED INTERVAL SIZE  (SIZMIN)1PE17.5//42H INITIAL
     3 VALUE OF THE INDEPENDENT VARIABLE,1PE17.5/6X,45HINITIAL CONDITION
     4S OF THE DEPENDENT VARIABLES//(1X,I7,1H)1PE15.5,I7,1H)1PE15.5,I7,1
     5H)1PE15.5))
      WRITE(6,98)(I,EMAX(I),I=1,N)
   98 FORMAT(1H0/7X,39HERROR BOUNDS OF THE DEPENDENT VARIABLES//(1X,I7,1
     1H)1PE15.5,I7,1H)1PE15.5,I7,1H)1PE15.5))
      NLA=0
      NLV=0
      IF(NV)2017,2000,2001
 2001 VN=NV
      VLN=VN/3.+.9
      NLV=VLN
      IF(NLV.GT.16)GO TO 2003
      WRITE(6,96)XO,(I,VAR(I),I=1,NV)
   96 FORMAT(1H0/7X,15HINTEGRATE  FROM,1PE19.5,7X,15HAT THESE VALUES/(1X
     A,I7,1H)1PE15.5,I7,1H)1PE15.5,I7,1H)1PE15.5))
```

```
      NPV=20-NLV                                                      DIFF0900
      GO TO 2007                                                      DIFF0910
 2003 WRITE(6,96)XO,(I,VAR(I),I=1,48)                                 DIFF0920
      IF(NV.GT.201)GO TO 2005                                         DIFF0930
      IP=IP+1                                                         DIFF0940
      WRITE(6,961)IP,(I,VAR(I),I=49,NV)                               DIFF0950
  961 FORMAT(1H1,58X,4HPAGE,I7/9X,37HINTEGRATION  (INDEPENDENT)  VARIABLDIFF0960
     1ES//(1X,I7,1H)1PE15.5,I7,1H)1PE15.5,I7,1H)1PE15.5))             DIFF0970
      NPV=67-NLV                                                      DIFF0980
      GO TO 2007                                                      DIFF0990
 2005 IP=IP+1                                                         DIFF1000
      WRITE(6,961)IP,(I,VAR(I),I=49,201)                              DIFF1010
      IF(NV.GT.300)GO TO 2017                                         DIFF1020
      IP=IP+1                                                         DIFF1030
      WRITE(6,951)IP,(I,VAR(I),I=202,NV)                              DIFF1040
      NPV=118-NLV                                                     DIFF1050
      GO TO 2007                                                      DIFF1060
 2000 WRITE(6,97)XO,FINAL,DELTA                                       DIFF1070
   97 FORMAT(1H0/7X,15HINTEGRATE  FROM1PE16.5,5H   TO1PE19.5/14X,11HIN SDIFF1080
     1TEPS OF1PE19.5)                                                 DIFF1090
      NPV=16                                                          DIFF1100
 2007 IF(NA)2008,2009,2008                                            DIFF1110
 2008 AN=NA                                                           DIFF1120
      ALN=AN/3.+.9                                                    DIFF1130
      NA=AN                                                           DIFF1140
      IF(NPV.GT.16)GO TO 2011                                         DIFF1150
      IF(NLA.GT.12)GO TO 2013                                         DIFF1160
      WRITE(6,95)(I,A(I),I=1,NA)                                      DIFF1170
   95 FORMAT(1H0/12X,49HCONSTANTS USED IN THE   DE  SUBROUTINE  EQUATIONDIFF1180
     1S//(1X,I7,1H)1PE15.5,I7,1H)1PE15.5,I7,1H)1PE15.5))              DIFF1190
      GO TO 2021                                                      DIFF1200
 2013 WRITE(6,95)(I,A(I),I=1,36)                                      DIFF1210
```

```
      IF(NA.GT.189)GO TO 2015                                           DIFF1220
      IP=IP+1                                                           DIFF1230
      WRITE(6,951)IP,(I,A(I),I=37,NA)                                   DIFF1240
  951 FORMAT(1H1,11X,49HCONSTANTS USED IN THE   DE  SUBROUTINE  EQUATIONDIFF1250
     1S//(1X,I7,1H)1PE15.5,I7,1H)1PE15.5,I7,1H)1PE15.5))                DIFF1260
      GO TO 2021                                                        DIFF1270
 2015 IP=IP+1                                                           DIFF1280
      WRITE(6,951)IP,(I,A(I),I=37,189)                                  DIFF1290
      IF(NA.GT.200)GO TO 2017                                           DIFF1300
      IP=IP+1                                                           DIFF1310
      WRITE(6,951)IP,(I,A(I),I=190,NA)                                  DIFF1320
      GO TO 2021                                                        DIFF1330
 2011 IF(NA.GT.153)GO TO 2019                                           DIFF1340
      IP=IP+1                                                           DIFF1350
      WRITE(6,951)IP,(I,A(I),I=1,NA)                                    DIFF1360
      GO TO 2021                                                        DIFF1370
 2019 IP=IP+1                                                           DIFF1380
      WRITE(6,951)IP,(I,A(I),I=1,153)                                   DIFF1390
      IF(NA.GT.200)GO TO 2017                                           DIFF1400
      IP=IP+1                                                           DIFF1410
      WRITE(6,951)IP,(I,A(I),I=154,NA)                                  DIFF1420
      GO TO 2021                                                        DIFF1430
 2009 NPA=0                                                             DIFF1440
 2021 IF(NLA+NLV-5)2022,2022,2023                                       DIFF1450
 2022 WRITE(6,101)                                                      DIFF1460
  101 FORMAT(1H0//1X,24(3H*  )//13X,9HVARIABLES//2X,11HINDEPENDENT9X,9HDDIFF1470
     1EPENDENT//)                                                       DIFF1480
      NCT=45                                                            DIFF1490
      GO TO 2024                                                        DIFF1500
 2023 IP=IP+1                                                           DIFF1510
      WRITE(6,1010)IP                                                   DIFF1520
 1010 FORMAT(1H164X,4HPAGEI3/13X,9HVARIABLES//2X,11HINDEPENDENT9X,9HDEPEDIFF1530
     1NDENT//)                                                          DIFF1540
```

```
      NCT=5                                                             DIFF1550
 2024 IX=1                                                              DIFF1560
      IF(DELTA)5,6,5                                                    DIFF1570
    5 XF=XO+DELTA                                                       DIFF1580
      GO TO 9                                                           DIFF1590
    6 L=1                                                               DIFF1600
      XF=VAR(L)                                                         DIFF1610
    9 WRITE(6,102)XO,(I,YO(I),I=1,N)                                    DIFF1620
  102 FORMAT(1PE13.4,2X,3(I4,1H)1PE13.5)/(15X,I4,1H)1PE13.5,I4,1H)1PE13.DIFF1630
     A5,I4,1H)1PE13.5))                                                 DIFF1640
   10 IERR=0                                                            DIFF1650
      CALL AMSINT(IX,XF,Y,EMAX,N,HO,XO,YO,HMIN,X,IERR)                  DIFF1660
      IF(IERR)2060,2070,2060                                            DIFF1670
 2060 IER=IER+1                                                         DIFF1680
      IF(IER-1)2062,2061,2062                                           DIFF1690
 2061 XFP=XF                                                            DIFF1700
      IER=1                                                             DIFF1710
      GO TO 2064                                                        DIFF1720
 2062 IF(XFP-XF)2061,2063,2061                                          DIFF1730
 2063 IF(IER-NER)2064,2064,301                                          DIFF1740
  301 WRITE(6,105)                                                      DIFF1750
  105 FORMAT(1H0//12(6H      *)/6X,55HERROR FOUND IN THIS CASE  --  WILL DIFF1760
     1PROCEED TO NEXT CASE/12(6H      *))                               DIFF1770
      GO TO 80                                                          DIFF1780
 2064 DO 2065 J=1,N                                                     DIFF1790
 2065 YO(J)=Y(J)                                                        DIFF1800
      ERI=IER                                                           DIFF1810
      XO=X+(XF-X)*0.1*ERI                                               DIFF1820
      IX=1                                                              DIFF1830
      GO TO 10                                                          DIFF1840
 2070 CONTINUE                                                          DIFF1850
      WRITE(6,102)XF,(I,Y(I),I=1,N)                                     DIFF1860
```

```
      NCT=NCT+NLL
      IF(NCT.LT.50)GO TO 61
      IP=IP+1
      WRITE(6,1010)IP
      NCT=5
   61 CONTINUE
   30 IF(HO)2051,2017,2050
 2050 IF(FINAL-XF)12,12,11
 2051 IF(FINAL-XF)11,12,12
   11 IF(DELTA)15,16,15
   15 XF=XF+DELTA
      IX=2
      GO TO 10
   16 L=L+1
      XF=VAR(L)
      IX=2
      GO TO 10
   12 CONTINUE
   80 CALL CLOCK(DATE,HOURS,SECNDS)
      STIMEN=HOURS
      SSECN=SECNDS
      ETIME=STIMEN-STIMEP
      ESEC=SSECN-SSECP
      IP=IP+1
      ITIME=ESEC
      WRITE(6,103)IP,ETIME,ITIME
  103 FORMAT(1H163X,4HPAGEI4/6X,44HALL DEPENDENT VARIABLES HAVE BEEN CAL
     1CULATED//6X,33HFOR THE PREVIOUS INPUT QUANTITIES////14X,19HELAPSED
     2 TIME OF RUNI10,12H  HOURS  ***I6,12H  SECONDS   )
      IF(IOF)GO TO 300
      GO TO 1
 2017 WRITE(6,955)NA,NV,N,DELTA,HO
```

```
  955 FORMAT(1H15X,34HERROR IN NUMBER OF INPUT VARIABLES/6X,3HNA=I7/6X,3DIFF2190
     1HNV=I7/6X,2HN=I8/6X,6HDELTA=1PE12.5/6X,3HH0=1PE15.5)              DIFF2200
  300 IETIME=STIMEN-STIME                                                DIFF2210
      IESEC=SSECN-SSEC                                                   DIFF2220
      ITIME=IESEC                                                        DIFF2230
      WRITE(6,104)IETIME,ITIME                                           DIFF2240
  104 FORMAT(1H0//8X,25HTOTAL ELAPSED TIME OF RUN,I10,12H  HOURS  ***,I6DIFF2250
     A,12H  SECONDS   )                                                  DIFF2260
      STOP                                                               DIFF2270
      END                                                                DIFF2280
$       FORTRAN LSTOU,DECK,STAB                                          AMSN0000
$       INCODE  IBMF                                                     AMSN0010
*AMSINT                      DIFFERENTIAL EQUATION SUBROUTINE            AMSN0020
*                             CD600D7.002      DATE 05/06/65             AMSN0030
*       SOLVES A SET OF N SIMULTANEOUS FIRST ORDER DIFFERENTIAL EQUATIONSAMSN0040
*       USING THE GENERALIZED ADAMS-MOULTON METHOD                       AMSN0050
      SUBROUTINE AMSINT(IX,XF,Y,EMAX,N,H1,X0,Y0,HF,X,IERR)              AMSN0060
      DIMENSION Y(25),EMAX(25),Y0(25),YP(25),Y1(25),Y2(25)               AMSN0070
      DIMENSION F(25,6),DYP(25),DYC(25),E(25),P(6,27),C(5,9)             AMSN0080
      DATA P/16128.,22808.,8218.,1645.,107.,.0203,3840.,5512.,2135.,     AMSN0090
     A570.,107.,.0277,1152.,1675.,694.,278.,107.,.0339,768.,1163.,       AMSN0100
     B648.,278.,25.,.0335,192.,297.,187.,107.,25.,.043,96.,151.,106.,    AMSN0110
     C91.,40.,.0502,360.,589.,625.,455.,59.,.0497,96.,161.,206.,200.,    AMSN0120
     D59.,.0597,126.,215.,322.,469.,236.,.0663,2016.,3880.,2366.,539.,   AMSN0130
     E37.,.0358,480.,952.,625.,190.,37.,.0477,144.,293.,206.,94.,37.,    AMSN0140
     F.0573,96.,211.,200.,94.,9.,.0563,24.,55.,59.,37.,9.,.0704,30.,     AMSN0150
     G71.,85.,80.,36.,.0804,90.,229.,415.,320.,44.,.0792,6.,16.,35.,     AMSN0160
     H36.,11.,.0924,126.,347.,889.,1372.,704.,.1008,504.,1648.,1526.,    AMSN0170
     J413.,31.,.0588,120.,416.,415.,150.,31.,.0756,36.,131.,140.,76.,    AMSN0180
     K31.,.0882,6.,25.,36.,19.,2.,.0861,6.,27.,44.,31.,8.,.1033,30.,     AMSN0190
     L143.,260.,275.,128.,.1148,45.,242.,665.,550.,82.,.1127,6.,35.,     AMSN0200
     M116.,128.,41.,.1268,63.,388.,1505.,2492.,1312.,.1353/             AMSN0210
      DATA C/2016.,856.,1246.,91.,5.,96.,40.,61.,6.,1.,144.,59.,94.,     AMSN0220
```

```
     A14.,5.,96.,37.,72.,14.,1.,24.,9.,19.,5.,1.,30.,11.,25.,10.,4.,    AMSN0230
     B90.,31.,95.,40.,4.,6.,2.,7.,4.,1.,126.,41.,161.,140.,64./          AMSN0240
      GO TO (1,3),IX                                                     AMSN0250
*     COMPUTE A STARTING H                                               AMSN0260
    1 ASSIGN 12 TO ICHOCE                                                AMSN0270
      H=32.                                                              AMSN0280
   10 H=H/2.                                                             AMSN0290
      IF(ABS(H1)-H)10,11,11                                              AMSN0300
   11 H=H*ABS(H1)/H1                                                     AMSN0310
*     GO THROUGH INITIAL EXTRAPOLATION                                   AMSN0320
   12 CALL DE(X0,Y0(1),F(1,2))                                           AMSN0330
      DO 101 I=1,N                                                       AMSN0340
  101 Y1(I)=Y0(I)+H*F(I,2)/4.                                            AMSN0350
      X=X0+H/4.                                                          AMSN0360
      CALL DE(X,Y1(1),F(1,3))                                            AMSN0370
      DO 102 I=1,N                                                       AMSN0380
  102 Y2(I)=Y1(I)+H*(3.*F(I,3)-F(I,2))/8.                                AMSN0390
      X=X+H/4.                                                           AMSN0400
      CALL DE(X,Y2(1),F(1,4))                                            AMSN0410
      DO 103 I=1,N                                                       AMSN0420
  103 YP(I)=Y2(I)+H*(19.*F(I,4)-20.*F(I,3)+7.*F(I,2))/12.                AMSN0430
 1031 X=X+H/2.                                                           AMSN0440
*     PERFORM INITIAL CORRECTION AND TEST                                AMSN0450
      CALL DE(X,YP(1),F(1,5))                                            AMSN0460
      ASSIGN 107 TO ITEST                                                AMSN0470
      DO 104 I=1,N                                                       AMSN0480
      Y(I)=Y0(I)+H*(37.*F(I,2)+72.*F(I,3)-14.*F(I,4)+F(I,5))/384.        AMSN0490
      IF(EMAX(I))1039,1040,1040                                          AMSN0500
 1039 IF(ABS(Y1(I)-Y(I))-ABS(EMAX(I)*Y(I)))104,104,1041                  AMSN0510
 1040 IF(ABS(Y1(I)-Y(I))-EMAX(I))104,104,1041                            AMSN0520
 1041 ASSIGN 1031 TO ITEST                                               AMSN0530
  104 Y1(I)=Y(I)                                                         AMSN0540
      X=X0+H/4.                                                          AMSN0550
```

```
      CALL DE(X,Y1(1),F(1,3))
      DO 105 I=1,N
      Y(I)=Y1(I)+H*(-5.*F(I,2)+56.*F(I,3)+46.*F(I,4)-F(I,5))/384.
      IF(EMAX(I))1049,1050,1050
 1049 IF(ABS(Y2(I)-Y(I))-ABS(EMAX(I)*Y(I)))105,105,1051
 1050 IF(ABS(Y2(I)-Y(I))-EMAX(I))105,105,1051
 1051 ASSIGN 1031 TO ITEST
  105 Y2(I)=Y(I)
      X=X+H/4.
      CALL DE(X,Y2(1),F(1,4))
      DO 106 I=1,N
      Y(I)=Y2(I)+H*(F(I,2)-4.*F(I,3)+7.*F(I,4)+2.*F(I,5))/12.
      IF(EMAX(I))1059,1060,1060
 1059 IF(ABS(YP(I)-Y(I))-ABS(EMAX(I)*Y(I)))106,106,1061
 1060 IF(ABS(YP(I)-Y(I))-EMAX(I))106,106,1061
 1061 ASSIGN 1031 TO ITEST
  106 YP(I)=Y(I)
      GO TO ITEST,(1031,107)
  107 X=X+H/2.
      IT1=-1
      IT2=-1
      IT3=0
   20 CALL DE(X,YP(1),F(1,5))
  200 ITC=3*IT1+IT2
      ITP=-3*ITC-IT3+14
      ITC=5-ITC
      DO 201 I=1,N
      DYP(I)=(P(2,ITP)*F(I,5)-P(3,ITP)*F(I,4)+P(4,ITP)*F(I,3)-P(5,ITP)*
     1F(I,2))*H/P(1,ITP)
  201 Y1(I)=YP(I)+DYP(I)
      X=X+H
      CALL DE(X,Y1(1),F(1,6))
      DO 202 J=1,N
```

```
      DYC(J)=(C(2,ITC)*F(J,6)+C(3,ITC)*F(J,5)-C(4,ITC)*F(J,4)+C(5,ITC)* AMSN0890
     1F(J,3))*H/C(1,ITC)                                                 AMSN0900
      E(J)=P(6,ITP)*(DYC(J)-DYP(J))                                      AMSN0910
      IF(EMAX(J)-ABS(E(J)))210,202,202                                   AMSN0920
  202 Y(J)=DYC(J)-E(J)+YP(J)                                             AMSN0930
      ASSIGN 204 TO LOW                                                  AMSN0940
      DO 203 I=1,N                                                       AMSN0950
      Y2(I)=YP(I)                                                        AMSN0960
      YP(I)=Y(I)                                                         AMSN0970
      F(I,1)=F(I,2)                                                      AMSN0980
      F(I,2)=F(I,3)                                                      AMSN0990
      F(I,3)=F(I,4)                                                      AMSN1000
      F(I,4)=F(I,5)                                                      AMSN1010
      F(I,5)=F(I,6)                                                      AMSN1020
      IF(EMAX(I)/40.-ABS(E(I)))2031,2031,203                             AMSN1030
 2031 ASSIGN 205 TO LOW                                                  AMSN1040
  203 CONTINUE                                                           AMSN1050
      ITT=1                                                              AMSN1060
      ITO=0                                                              AMSN1070
      GO TO LOW,(204,205)                                                AMSN1080
  204 ITO=-1                                                             AMSN1090
      H=H*2.                                                             AMSN1100
  205 IT4=IT3                                                            AMSN1110
      IT3=IT2                                                            AMSN1120
      IT2=IT1                                                            AMSN1130
      IT1=ITO                                                            AMSN1140
      ASSIGN 211 TO ICHOCE                                               AMSN1150
    2 IF((X-XF)/H)20,206,250                                             AMSN1160
  206 RETURN                                                             AMSN1170
    3 IF((X-XF)/H)20,30,250                                              AMSN1180
   30 DO 31 I=1,N                                                        AMSN1190
   31 Y(I)=YP(I)                                                         AMSN1200
      GO TO 206                                                          AMSN1210
```

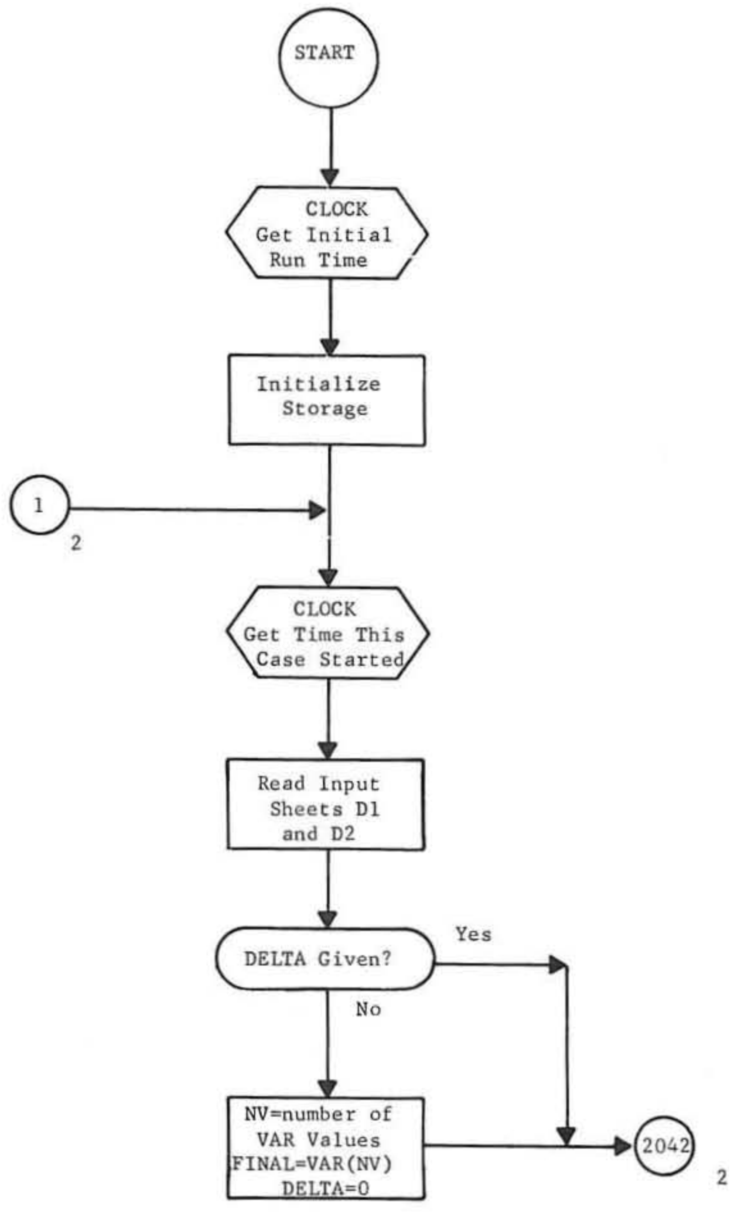
```
*       REDUCE INTERVAL SIZE AND RECALCULATE                              AMSN1220
  210 H=H/2.                                                              AMSN1230
      IF(ABS(HF)-ABS(H))2100,2100,220                                     AMSN1240
 2100 GO TO ICHOCE,(12,211)                                               AMSN1250
  211 IF(IT1)212,212,213                                                  AMSN1260
  212 IT1=IT1+1                                                           AMSN1270
      X=X-2.*H                                                            AMSN1280
      GO TO 200                                                           AMSN1290
  213 IF(ITT)220,2131,2131                                                AMSN1300
 2131 IF(IT2)214,214,220                                                  AMSN1310
  214 IT1=IT2+1                                                           AMSN1320
      IT2=IT3                                                             AMSN1330
      IT3=IT4                                                             AMSN1340
      H=2.*H                                                              AMSN1350
      X=X-3.*H                                                            AMSN1360
      DO 215 I=1,N                                                        AMSN1370
      YP(I)=Y2(I)                                                         AMSN1380
      F(I,5)=F(I,4)                                                       AMSN1390
      F(I,4)=F(I,3)                                                       AMSN1400
      F(I,3)=F(I,2)                                                       AMSN1410
  215 F(I,2)=F(I,1)                                                       AMSN1420
      ITT=-1                                                              AMSN1430
      GO TO 200                                                           AMSN1440
*       TAKE CARE OF SINGULARITY OUTPUT                                   AMSN1450
  220 CALL SING(X,Y,N,J)                                                  AMSN1460
      IERR=J                                                              AMSN1470
      GO TO 206                                                           AMSN1480
*       COMPUTE SPECIAL VALUES OF Y                                       AMSN1490
  250 IF(IT1)2503,2501,2502                                               AMSN1500
 2501 HI=H                                                                AMSN1510
      GO TO 251                                                           AMSN1520
 2502 HI=2.*H                                                             AMSN1530
      GO TO 251                                                           AMSN1540
```

```
 2503 HI=H/2.                                                          AMSN1550
  251 IF(IT2)2511,2513,2512                                            AMSN1560
 2511 H2=HI/2.                                                         AMSN1570
      GO TO 252                                                        AMSN1580
 2512 H2=2.*HI                                                         AMSN1590
      GO TO 252                                                        AMSN1600
 2513 H2=HI                                                            AMSN1610
  252 IF(IT3)2521,2523,2522                                            AMSN1620
 2521 H3=H2/2.                                                         AMSN1630
      GO TO 253                                                        AMSN1640
 2522 H3=2.*H2                                                         AMSN1650
      GO TO 253                                                        AMSN1660
 2523 H3=H2                                                            AMSN1670
  253 Q1=H2+HI                                                         AMSN1680
      Q2=H2+H3                                                         AMSN1690
      R1=H3+Q1                                                         AMSN1700
      U=XF-X+HI                                                        AMSN1710
      U1=((U/4.+(H2+Q2)/3.)*U+H2*Q2/2.)*U/(HI*Q1*R1)                   AMSN1720
      U2=((U/4.-(HI-H2-Q2)/3.)*U-(HI*H2-H2*Q2+Q2*HI)/2.)*U/(HI*H2*Q2)-1.AMSN1730
      U3=((U/4.-(HI-Q2)/3.)*U-HI*Q2/2.)*U/(H2*H3*Q1)                   AMSN1740
      U4=((U/4.-(HI-H2)/3.)*U-HI*H2/2.)*U/(H3*Q2*R1)                   AMSN1750
      DO 260 I=1,N                                                     AMSN1760
  260 Y(I)=Y2(I)+U*(U1*F(I,5)-U2*F(I,4)+U3*F(I,3)-U4*F(I,2))           AMSN1770
      GO TO 206                                                        AMSN1780
      END                                                              AMSN1790
$       GMAP    DECK                                                   EFM00000
*EFMOUT         SUBROUTINE TO INHIBIT FLOATING FAULT                   EFM00010
*                               CD600D7.002     DATE 05/06/65          EFM00020
        LBL     EFMOUT                                                 EFM00030
        SYMDEF  EFMOUT                                                 EFM00040
EFMOUT  STI     IND                                                    EFM00050
        LDA     (04000,DL                                              EFM00060
        ORSA    IND                                                    EFM00070
```

```
        LDI     IND                                                    EFM00080
        TRA     0,1                                                    EFM00090
    IND BSS     1                                                      EFM00100
        END                                                            EFM00110
$       GMAP    DECK                                                   CLCK0000
*CLOCK          FORTRAN CLOCK ROUTINE                                  CLCK0010
*                                CD600D7.002      DATE 05/06/65        CLCK0020
*                               CALL CLOCK[DATE,IHOUR,ISEC]            CLCK0030
        LBL     CLOCK                                                  CLCK0040
        SYMDEF  CLOCK                                                  CLCK0050
  CLOCK STX1    A                                                      CLCK0060
        MME     GETIME                                                 CLCK0070
      A LDX1    0,DU                                                   CLCK0080
        STA     2,1*            STORE MODAYR IN DATE                   CLCK0090
        DIV     (64000,DL       TOTAL SECONDS TO Q                     CLCK0100
        DIV     (3600,DL        HOURS TO Q, SECONDS TO A               CLCK0110
        STQ     3,1*            PUT HOURS IN IHOUR                     CLCK0120
        STA     4,1*             PUT SECONDS IN ISEC                   CLCK0130
        TRA     0,1                                                    CLCK0140
        END                                                            CLCK0150
$       GMAP    DECK                                                   MAB 0000
*MAB            SUBROUTINE TO GIVE PLUS BITS                           MAB 0010
*                                CD600D7.002      DATE 05/06/65        MAB 0020
        LBL     MAB                                                    MAB 0030
        SYMDEF  MAB                                                    MAB 0040
*               CALL MAB[BITS]                                         MAB 0050
    MAB LDA     BITS                                                   MAB 0060
        STA     2,1*                                                   MAB 0070
        TRA     0,1                                                    MAB 0080
   BITS OCT     377777777777                                           MAB 0090
        END                                                            MAB 0100
$       FORTRAN LSTOU,DECK,STAB                                        SING0000
$       INCODE  IBMF                                                   SING0010
```

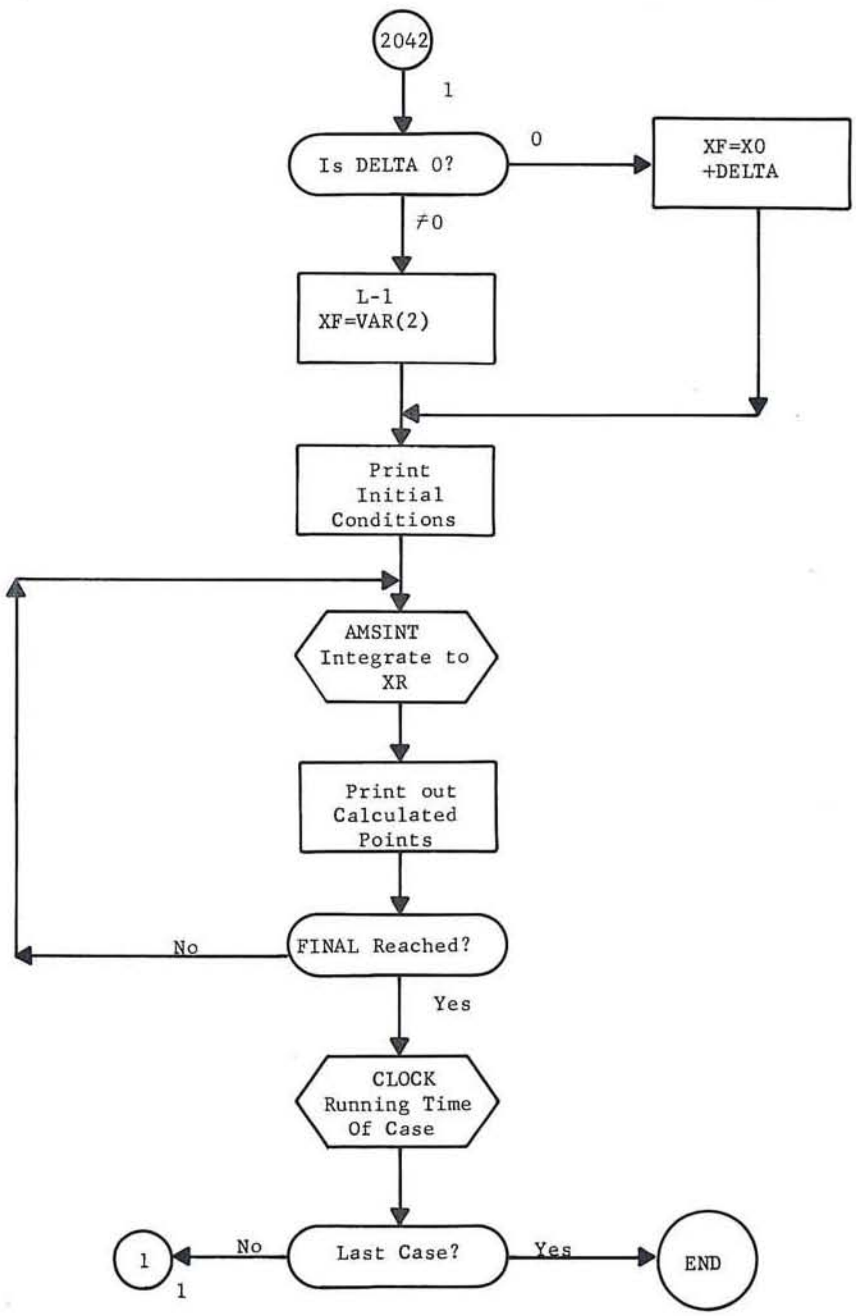
```
*SING                     SINGULARITY SUBROUTINE FOR TEST CASE              SING0020
*                         CD600D7.002       DATE 05/06/65                   SING0030
      SUBROUTINE SING(X,Y,N,J)                                              SING0040
      DIMENSION Y(25)                                                       SING0050
      WRITE(6,100)J,X,(Y(I),I=1,N)                                          SING0060
  100 FORMAT(1H05X,25HSINGULARITY DETECTED IN YI5/6X,4HTIME1PE15.6/6X,      SING0070
     A9HFUNCTIONS/(4(1PE16.6)))                                             SING0080
      RETURN                                                                SING0090
      END                                                                   SING0100
$      FORTRAN LSTOU,DECK,STAB                                              DE  0000
$      INCODE  IBMF                                                         DE  0010
*DE                       DE ROUTINE FOR DIFFEE TEST CASE                   DE  0020
*                         CD600D7.002       DATE 05/06/65                   DE  0030
      SUBROUTINE DE(X,Y,F)                                                  DE  0040
      COMMON A(200)                                                         DE  0050
      DIMENSION Y(25),F(25)                                                 DE  0060
      F(1)=Y(2)                                                             DE  0070
      F(2)=-Y(2)-Y(1)-Y(1)**2-Y(1)**3+A(1)*SIN(X)                           DE  0080
      RETURN                                                                DE  0090
      END                                                                   DE  0100
```

APPENDIX B
FLOW CHARTS



DIFFE

(1)


2042
1
Is DELTA 0?
0
XF=X0
+DELTA
≠0
L-1
XF=VAR(2)
Print
Initial
Conditions
AMSINT
Integrate to
XR
Print out
Calculated
Points
FINAL Reached?
No
Yes
CLOCK
Running Time
Of Case
Last Case?
No
Yes
1
1
END

DIFFE

(2)

DIFFE INPUT (SHEET D2)

DELTA= ________ , FINAL= ____________,

VAR= ________ , ________ , ________ , ________ , ________ , ________ ,

A= ________ , ________ , ________ , ________ , ________ , ________ ,

IOF= ________ $

T if last case
F if not last case

DIFFE INPUT (SHEET D2)

DELTA= ________ , FINAL= ____________ ,

VAR= ________ , ________ , ________ , ________ , ________ , ________ ,

A= ________ , ________ , ________ , ________ , ________ , ________ ,

IOF= ________ $

T if last case
F if not last case

DIFFE INPUT (SHEET D2)

DELTA= ________ , FINAL= ____________ ,

VAR= ________ , ________ , ________ , ________ , ________ , ________ ,
________ , ________ , ________ , ________ , ________ , ________ ,
________ , ________ , ________ , ________ , ________ , ________ ,
________ , ________ , ________ , ________ , ________ , ________ ,
________ , ________ , ________ , ________ , ________ , ________ ,
________ , ________ , ________ , ________ , ________ , ________ ,
________ , ________ , ________ , ________ , ________ , ________ ,
________ , ________ , ________ , ________ , ________ , ________ ,
________ , ________ , ________ , ________ , ________ , ________ ,
________ , ________ , ________ , ________ , ________ , ________ ,
________ , ________ , ________ , ________ , ________ , ________ ,
________ , ________ , ________ , ________ , ________ , ________ ,
________ , ________ , ________ , ________ , ________ , ________ ,
________ , ________ , ________ , ________ , ________ , ________ ,
________ , ________ , ________ , ________ , ________ , ________ ,
________ , ________ , ________ , ________ , ________ , ________ ,
________ , ________ , ________ , ________ , ________ , ________ ,
________ , ________ , ________ , ________ , ________ , ________ ,
________ , ________ , ________ , ________ , ________ , ________ ,
________ , ________ , ________ , ________ , ________ , ________ ,

A= ________ , ________ , ________ , ________ , ________ , ________ ,
________ , ________ , ________ , ________ , ________ , ________ ,
________ , ________ , ________ , ________ , ________ , ________ ,
________ , ________ , ________ , ________ , ________ , ________ ,
________ , ________ , ________ , ________ , ________ , ________ ,
________ , ________ , ________ , ________ , ________ , ________ ,
________ , ________ , ________ , ________ , ________ , ________ ,
________ , ________ , ________ , ________ , ________ , ________ ,
________ , ________ , ________ , ________ , ________ , ________ ,
________ , ________ , ________ , ________ , ________ , ________ ,
________ , ________ , ________ , ________ , ________ , ________ ,
________ , ________ , ________ , ________ , ________ , ________ ,
________ , ________ , ________ , ________ , ________ , ________ ,
________ , ________ , ________ , ________ , ________ , ________ ,
________ , ________ , ________ , ________ , ________ , ________ ,
________ , ________ , ________ , ________ , ________ , ________ ,
________ , ________ , ________ , ________ , ________ , ________ ,
________ , ________ , ________ , ________ , ________ , ________ ,
________ , ________ , ________ , ________ , ________ , ________ ,

IOF= ________ $

T if last case
F if not last case

DIFFE INPUT (SHEET D2)

DELTA= __________ , FINAL= ______________,

VAR= __________ , __________ , __________ , __________ , __________ , __________ ,
__________ , __________ , __________ , __________ , __________ , __________ ,
__________ , __________ , __________ , __________ , __________ , __________ ,
__________ , __________ , __________ , __________ , __________ , __________ ,
__________ , __________ , __________ , __________ , __________ , __________ ,
__________ , __________ , __________ , __________ , __________ , __________ ,
__________ , __________ , __________ , __________ , __________ , __________ ,
__________ , __________ , __________ , __________ , __________ , __________ ,
__________ , __________ , __________ , __________ , __________ , __________ ,
__________ , __________ , __________ , __________ , __________ , __________ ,
__________ , __________ , __________ , __________ , __________ , __________ ,
__________ , __________ , __________ , __________ , __________ , __________ ,
__________ , __________ , __________ , __________ , __________ , __________ ,
__________ , __________ , __________ , __________ , __________ , __________ ,
__________ , __________ , __________ , __________ , __________ , __________ ,
__________ , __________ , __________ , __________ , __________ , __________ ,
__________ , __________ , __________ , __________ , __________ , __________ ,
__________ , __________ , __________ , __________ , __________ , __________ ,
__________ , __________ , __________ , __________ , __________ , __________ ,
__________ , __________ , __________ , __________ , __________ , __________ ,
__________ , __________ , __________ , __________ , __________ , __________ ,

A= __________ , __________ , __________ , __________ , __________ , __________ ,
__________ , __________ , __________ , __________ , __________ , __________ ,
__________ , __________ , __________ , __________ , __________ , __________ ,
__________ , __________ , __________ , __________ , __________ , __________ ,
__________ , __________ , __________ , __________ , __________ , __________ ,
__________ , __________ , __________ , __________ , __________ , __________ ,
__________ , __________ , __________ , __________ , __________ , __________ ,
__________ , __________ , __________ , __________ , __________ , __________ ,
__________ , __________ , __________ , __________ , __________ , __________ ,
__________ , __________ , __________ , __________ , __________ , __________ ,
__________ , __________ , __________ , __________ , __________ , __________ ,
__________ , __________ , __________ , __________ , __________ , __________ ,
__________ , __________ , __________ , __________ , __________ , __________ ,
__________ , __________ , __________ , __________ , __________ , __________ ,
__________ , __________ , __________ , __________ , __________ , __________ ,
__________ , __________ , __________ , __________ , __________ , __________ ,
__________ , __________ , __________ , __________ , __________ , __________ ,
__________ , __________ , __________ , __________ , __________ , __________ ,
__________ , __________ , __________ , __________ , __________ , __________ ,
__________ , __________ , __________ , __________ , __________ , __________ ,

IOF= __________ $

T if last case
F if not last case

DIFFE INPUT (SHEET D2)

DELTA= __________ , FINAL= ________________ ,

VAR= __________ , __________ , __________ , __________ , __________ , __________ ,
__________ , __________ , __________ , __________ , __________ , __________ ,
__________ , __________ , __________ , __________ , __________ , __________ ,
__________ , __________ , __________ , __________ , __________ , __________ ,
__________ , __________ , __________ , __________ , __________ , __________ ,
__________ , __________ , __________ , __________ , __________ , __________ ,
__________ , __________ , __________ , __________ , __________ , __________ ,
__________ , __________ , __________ , __________ , __________ , __________ ,
__________ , __________ , __________ , __________ , __________ , __________ ,
__________ , __________ , __________ , __________ , __________ , __________ ,
__________ , __________ , __________ , __________ , __________ , __________ ,
__________ , __________ , __________ , __________ , __________ , __________ ,
__________ , __________ , __________ , __________ , __________ , __________ ,
__________ , __________ , __________ , __________ , __________ , __________ ,
__________ , __________ , __________ , __________ , __________ , __________ ,
__________ , __________ , __________ , __________ , __________ , __________ ,
__________ , __________ , __________ , __________ , __________ , __________ ,
__________ , __________ , __________ , __________ , __________ , __________ ,
__________ , __________ , __________ , __________ , __________ , __________ ,
__________ , __________ , __________ , __________ , __________ , __________ ,
__________ , __________ , __________ , __________ , __________ , __________ ,

A= __________ , __________ , __________ , __________ , __________ , __________ ,
__________ , __________ , __________ , __________ , __________ , __________ ,
__________ , __________ , __________ , __________ , __________ , __________ ,
__________ , __________ , __________ , __________ , __________ , __________ ,
__________ , __________ , __________ , __________ , __________ , __________ ,
__________ , __________ , __________ , __________ , __________ , __________ ,
__________ , __________ , __________ , __________ , __________ , __________ ,
__________ , __________ , __________ , __________ , __________ , __________ ,
__________ , __________ , __________ , __________ , __________ , __________ ,
__________ , __________ , __________ , __________ , __________ , __________ ,
__________ , __________ , __________ , __________ , __________ , __________ ,
__________ , __________ , __________ , __________ , __________ , __________ ,
__________ , __________ , __________ , __________ , __________ , __________ ,
__________ , __________ , __________ , __________ , __________ , __________ ,
__________ , __________ , __________ , __________ , __________ , __________ ,
__________ , __________ , __________ , __________ , __________ , __________ ,
__________ , __________ , __________ , __________ , __________ , __________ ,
__________ , __________ , __________ , __________ , __________ , __________ ,
__________ , __________ , __________ , __________ , __________ , __________ ,
__________ , __________ , __________ , __________ , __________ , __________ ,

IOF= __________ $

T if last case
F if not last case

GENERAL ELECTRIC
COMPUTERS

GE-625/635 Math Routines

GENERAL ELECTRIC

CPB-1165

GE-625/635
MATH ROUTINES
LSPF

LEAST SQUARES POLYNOMIAL FIT

Program Number
CD600D6.001

October 1965

GENERAL ELECTRIC
COMPUTER DEPARTMENT

CREDITS

The source material used in this manual is taken from a document published by the General Electric Telecommunications and Information Processing Department, titled Sequential Least Squares for Polynomials by C. B. Chandler and J. T. Godfrey. Permission to use the original document was given by D. L. Shell, Manager, Computer Applications and Processing of the Telecommunications and Information Processing Department, General Electric Company.

Comments on this publication may be addressed to Technical Publications, Computer Department, General Electric Company, P. O. Box 2961, Phoenix, Arizona, 85002.

CONTENTS

APPENDICES

1. GENERAL DESCRIPTION

The sequential least squares for polynomials program determines the coefficients of a polynomial in X which give the least squares fit to given data.

Five features make this least squares curve fitting program unique:

(1) Virtually any polynomial model in one variable may be specified. For example, a model such as $y = a_0 + a_1 x^7 + a_2 x^5 + a_3 x^2$ could be specified.

(2) All reduced models of the original can be obtained with a few additional calculations. Thus, in the example $y = b_0 + b_1 x^7 + b_2 x^5$, $y = c_0 + c_1 x^7$ and $y = d_0$ could also be obtained.

(3) Predictions are computed for all models.

(4) A complete set of statistical parameters are computed for each model such as F-test, t-test for coefficients and standardized deviates for predictions.

(5) The program is free standing and simple to use. No programming is required of the user.

2. MATHEMATICAL METHOD

The method used to solve the simultaneous equations represented by the product moment matrix and vector is called left-right decomposition. A complete inverse of the product-moment matrix is not needed. This method is as efficient as a normal approach for a given M and provides a much faster procedure for the calculation of the reduced models.

Briefly, the method is as follows:

The original equation is $A\hat{x} = \hat{b}$

where A = product-moment matrix S
$\hat{b}$ = product-moment vector Sy
$\hat{x}$ = vector of coefficients--the unknowns

The matrix A, which is symmetric, can be reduced to the product of 2 matrices L and R. That is, A = LR where

$$L = \begin{bmatrix} 1 & 0 & 0 & . & 0 \\ l_{21} & 1 & 0 & . & 0 \\ l_{31} & l_{32} & 1 & . & 0 \\ . & . & . & . & . \\ l_{N1} & l_{N2} & l_{N3} & . & 1 \end{bmatrix} \text{ and } R = \begin{bmatrix} r_{11} & r_{12} & r_{13} & . & r_{1N} \\ 0 & r_{22} & r_{23} & . & r_{2N} \\ 0 & 0 & r_{33} & . & r_{3N} \\ . & . & . & . & . \\ 0 & 0 & 0 & . & r_{NN} \end{bmatrix}$$

and where $l_{ij} = r_{ji}/r_{jj}$

and for $i = 1,\ r_{ij} = A_{ij}$ with $j = 1, N$

for $i > 1,\ r_{ij} = A_{ij} - \sum_{K=1}^{i-1} r_{Ki}\, r_{Kj}/r_{KK}$ with $j = i, N$

This gives the equation $A\hat{x} = LR\hat{x} = \hat{b}$

Let $\hat{y} = R\hat{x}$

Then $L\hat{y} = \hat{b}$

and $\hat{y}$ can be determined by forward substitution. That is, consider $L\hat{y} = \hat{b}$.

$$\begin{bmatrix} 1 & 0 & 0 & . & 0 \\ l_{21} & . & 0 & . & 0 \\ l_{31} & l_{32} & 1 & . & 0 \\ . & . & . & . & . \\ l_{N1} & l_{N2} & l_{N3} & . & 1 \end{bmatrix} \begin{bmatrix} y_1 \\ y_2 \\ y_3 \\ . \\ y_N \end{bmatrix} = \begin{bmatrix} b_1 \\ b_2 \\ b_3 \\ . \\ b_N \end{bmatrix}$$

Then $y_1 = b_1$

$y_2 = b_2 - l_{21} y_1$

$y_3 = b_3 - l_{31} y_1 - l_{32} y_2$

etc.

Now, since $R\hat{x} = \hat{y}$, $\hat{x}$ -- the desired coefficients -- can be determined by backward substitution.

$$\begin{bmatrix} r_{11} & . & r_{1,N-1} & r_{1,N} \\ . & . & . & . \\ . & . & . & . \\ 0 & . & r_{N-1,N-1} & r_{N-1,N} \\ 0 & . & 0 & r_{N,N} \end{bmatrix} \begin{bmatrix} X_1 \\ . \\ . \\ X_{N-1} \\ X_N \end{bmatrix} = \begin{bmatrix} y_1 \\ . \\ . \\ y_{N-1} \\ y_N \end{bmatrix}$$

Then, $X_N = y_N / r_{N,N}$

$X_{N-1} = (y_{N-1} - r_{N-1,N} X_N) / r_{N-1,N-1}$

etc.

The diagonal of the inverse of A, needed for the variance and t-test calculations, is obtained by the same method. That is, $A\hat{x} = LR\hat{x} = \hat{b}$.

If b is successively $\begin{bmatrix} 1 \\ 0 \\ 0 \\ . \\ 0 \end{bmatrix}, \begin{bmatrix} 0 \\ 1 \\ 0 \\ . \\ 0 \end{bmatrix}, \begin{bmatrix} 0 \\ 0 \\ 1 \\ . \\ 0 \end{bmatrix}, , \begin{bmatrix} 0 \\ 0 \\ 0 \\ . \\ 1 \end{bmatrix}$

then the resulting $\hat{x}$ vectors will give the inverse although only the diagonal is saved.

The primary advantage of this method is provided by its use of left-right decomposition. For the reduced models, the representation discussed above (reduced accordingly) is correct. Therefore, the forward substitution need not be repeated.

SAMPLE PROBLEM

A sample output is included in Chapter 3, Usage. The model was

$$\hat{Y} = C_{11} + C_{12}X^5 + C_{13}X^3 + C_{14}X + C_{15}X^2 + C_{16}X^4$$

The reduced models were then

$$\hat{Y} = C_{21} + C_{22} X^5 + C_{23} X^3 + C_{24} X + C_{25} X^2$$

$$\hat{Y} = C_{31} + C_{32} X^5 + C_{33} X^3 + C_{34} X$$

$$\hat{Y} = C_{41} + C_{42} X^5 + C_{43} X^3$$

$$\hat{Y} = C_{51} + C_{52} X^5$$

$$\hat{Y} = C_{61}$$

The data for this problem was taken casually from e^x; that is $y \sim e^x$.
The data, with weights used, was:

X	Y	W
-2.50	0.05	1.0
-1.50	0.20	2.0
-0.50	0.50	3.0
0.75	2.10	3.0
1.00	2.7182818	4.0
1.75	5.50	4.0
2.00	7.3890560	5.0
2.75	15.50	5.0
3.00	20.085536	5.0

The additional output under the option IOP2 includes the moment matrix S (NS) and the moment vector SY(NT). The listing also includes the coefficients desired and the diagonal of the inverse of the product moment matrix of each reduced model in sequence.

3. USAGE

RESTRICTIONS

The restrictions listed below govern the construction of the polynomial model:

1. The number of terms in the model being fitted may not exceed 15; that is, $NT \leq 15$.

2. The number of observations specified may be from NX = NT up to NX = 9999.

3. If the number of observations is greater than 200, a tape 3 is required.

4. If the number of observations is less than 201, all data is stored in core memory rather than on tape. Therefore, tapes will not be used.

5. The exponents K_1, K_2, K_3, ..., K_n must be positive integers or 0. No two of these exponents may be equal.

6. The fit obtained for lesser order models will not necessarily suffer from ill-conditioning which can occur with higher order models.

7. Conditioning of the product-moment matrix may be improved by subtracting a number X_0, close to the mean of the X_i's, from each of the $_i$'s prior to obtaining the model. The model would then be of the form

   $$\hat{Y} = C_1 (X - X_0)^{K_1} + C_2 (X - X_0)^{K_2} + \ldots + C_n (X - X_0)^{K_n}$$

   This program permits the user to specify an arbitrary "X_0" which will be subtracted from each of the X_i's as they are read.

8. The maximum value for any "K_i" is 30.

9. The weighting factor W, if used, may be any floating-point number -- not necessarily an integer.

DEFINITIONS AND OUTPUT DESIGNATION

The glossary on the following page defines the statistical parameters computed and gives the symbols used. If no weighting is used, the definitions are true for W = 1.

Parameter	Program Symbol	Definition
Degree of Freedom Total	NX	$NX_{total} = \Sigma 1$
Sum of Squares Total	SYSQ	$SYSQ = \Sigma WY^2$
Sum of Squares due to regression	SSREG	$SSREG = C_1 \Sigma WX^{K_1} Y + C_2 \Sigma WX^{K_2} Y + \ldots + C_n \Sigma WX^{K_n} Y$
Sum of Squares due to residual	SSYX	SSYX = SYSQ-SSREG
Degrees of Freedom due to regression	DFREG	DFREG = M
Degrees of Freedom due to residual	DFYX	DFYX = NX-DFREG
Mean Square due to regression	S2REG	S2REG = SSREG/DFREG
Mean Square due to residual	S2YX	S2YX = SSYX/DFYX
F-value	F	F = S2REG/S2YX

In addition, the term NS gives the number of elements in the upper half of the product moment matrix.

NS = (NT*NT - NT)/2 + NT.

When the analysis of variance table is complete, the coefficients and their variances are printed and a t-test for statistical significance different from zero is computed and printed.

Coefficient	Variance of Coefficient	t-test
C_1	$v(C_1) = c_{11} \cdot S2YX$	$t_{c1} = C_1/\sqrt{v(C_1)}$
C_2	$v(C_2) = c_{22} \cdot S2YX$	$t_{c2} = C_2/\sqrt{v(C_2)}$
C_3	$v(C_3) = c_{33}$	$t_{c3} = C_3/\sqrt{v(C_3)}$
.	.	.
C_M	$v(C_M) = c_{MM} \cdot S2YX$	$t_{c_M} = C_M/\sqrt{v(C_M)}$

The "t_{c_K}" in the table has DFYX degrees of freedom. The c_{11}, c_{22}, c_{33}, . . . , c_{MM} form the diagonal of the inverse of the product-moment matrix.

This information, including the above analysis of variance calculations, is repeated for each of the lesser models (if that option was taken). Finally, X_i and Y_i are printed with the determined $\hat{Y}_i$. The standardized deviate, $(Y_i - \hat{Y}_i)/\sqrt{S2YX}$, is calculated and printed.

All of the models and their corresponding standardized deviates are calculated for each point X_i, Y_i.

INPUT/OUTPUT

Input is as follows:

$PARA/NX = , NT = , NP = , NW = , IOP1 = , IOP2 = , XZER = $

Where

NX is the number of points to be read in
NT is the number of terms in the model
NP is the highest exponent of X
NW is 1 for weighting, 0 for no weighting
IOP1 is 1 for reduced models, 0 for original model only
IOP2 is 1 for intermediate printout, 0 for results only
XZER, if nonzero, is the value of X to be used as zero for the resulting curve.

$EXP/KSEQ = , , , , $

A list of the exponents in the desired order

$DATA/X = , Y = , W = , $

One card per input point. W is optional, depending on NW

Another case may be entered by beginning with another $PARA card. If no more cases are to be run, a $PARA card with NX=0 will stop the run.

Input for a sample case is illustrated on the following two pages. Additional input sheets are furnished at the back of this manual for the user's convenience. A listing of input cards and an output listing are also included in this section.

Input Coding Form (With Sample Data)

LSPF INPUT

Col
2

```
$PARA/NX= 9 , NT= 6 , NP= 5 , NW= 1 , IØP1= 1 , IØP2= 0 , XZER= 0 $
$EXP/KSEQ= 0 , 5 , 3 , 1 , 2 , 4 $
$DATA/X= -2.5 , Y= 0.05 , W= 1 $
$DATA/X= -1.5 , Y= 0.2 , W= 2 $
$DATA/X= -0.5 , Y= 0.5 , W= 3 $
$DATA/X= 0.75 , Y= 2.1 , W= 3 $
$DATA/X= 1 , Y= 2.7182818 , W= 4 $
$DATA/X= 1.75 , Y= 5.5 , W= 4 $
$DATA/X= 2 , Y= 7.389056 , W= 5 $
$DATA/X= 2.75 , Y= 15.5 , W= 5 $
$DATA/X= 3 , Y= 20.085536 , W= 5 $
$DATA/X= ________ , Y= ____________ , W= __________ $
$DATA/X= ________ , Y= ____________ , W= __________ $
$DATA/X= ________ , Y= ____________ , W= __________ $
$DATA/X= ________ , Y= ____________ , W= __________ $
$DATA/X= ________ , Y= ____________ , W= __________ $
$DATA/X= ________ , Y= ____________ , W= __________ $
$DATA/X= ________ , Y= ____________ , W= __________ $
$DATA/X= ________ , Y= ____________ , W= __________ $
```

LSPF INPUT

Col
2

$PARA/NX=Ø, NT=___, NP=___, NW=___, IØP1=___, IØP2=___, XZER=___$

$EXP/KSEQ=___, ___, ___, ___, ___, ___$

$DATA/X=__________, Y=__________, W=__________$

$DATA/X=__________, Y=__________, W=__________$

$DATA/X=__________, Y=__________, W=__________$

$DATA/X=__________, Y=__________, W=__________$

$DATA/X=__________, Y=__________, W=__________$

$DATA/X=__________, Y=__________, W=__________$

$DATA/X=__________, Y=__________, W=__________$

$DATA/X=__________, Y=__________, W=__________$

$DATA/X=__________, Y=__________, W=__________$

$DATA/X=__________, Y=__________, W=__________$

$DATA/X=__________, Y=__________, W=__________$

$DATA/X=__________, Y=__________, W=__________$

$DATA/X=__________, Y=__________, W=__________$

$DATA/X=__________, Y=__________, W=__________$

$DATA/X=__________, Y=__________, W=__________$

$DATA/X=__________, Y=__________, W=__________$

$DATA/X=__________, Y=__________, W=__________$

Listing of Input Cards

```
                                                                    FOE***
$       EXECUTE
$       INCODE   IBMF
 $PARA/NX=9,NT=6,NP=5,NW=1,IOP1=1,IOP2=0,XZER=0$
 $EXP/KSEQ=0,5,3,1,2,4$
 $DATA/X=-2.5,Y=0.05,W=1$
 $DATA/X=-1.5,Y=0.2,W=2$
 $DATA/X=-0.5,Y=0.5,W=3$
 $DATA/X=0.75,Y=2.1,W=3$
 $DATA/X=1,Y=2.7182818,W=4$
 $DATA/X=1.75,Y=5.5,W=4$
 $DATA/X=2,Y=7.389056,W=5$
 $DATA/X=2.75,Y=15.5,W=5$
 $DATA/X=3,Y=20.085536,W=5$
 $PARA/NX=0$
$       ENDJOB
$       EXECUTE
$       INCODE   IBMF
```

Output Listing

```
POLYNOMIAL FITTING BY
SEQUENTIAL LEAST SQUARES

GIVEN MODEL
     0     5     3     1     2     4
Y=C X  +C X  +C X  +C X  +C X  +C X
  1     2     3     4     5     6

SYSQ= 3,65600320E 03  NX=   9  M=   6  NS=  21
ANALYSIS OF VARIANCE
   SSREG          SSYX              DFREG          DFYX            S2REG            S2YX             F
 3.65579779E 03  2,05413818E-01                6              3  6.09299629E 02  6,84712725E-02  8,89861694E 03

          K    COEFFICIENT      VARIANCE        T-TEST
          1   9,37171698E-01  1,72527863E-02  7,13492715E 00
          2   2,08703391E-02  6,19765460E-05  2,65103719E 00
          3   7,38368556E-02  4,44022584E-03  1,10807885E 00
          4   1,12666719E 00  1,67327859E-02  8,70986700E 00
          5   4,90458202E-01  1,45713576E-02  4,06304836E 00
          6   5,25407619E-02  3,17335940E-04  2,94942078E 00
ANALYSIS OF VARIANCE
   SSREG          SSYX              DFREG          DFYX            S2REG            S2YX             F
 3,65520215E 03  8,01055908E-01                5              4  7.31040428E 02  2,00263977E-01  3,65038403E 03

          K    COEFFICIENT      VARIANCE        T-TEST
          1   6,68115020E-01  2,61214166E-02  4,13383245E 00
          2   4,02950835E-02  5,44062609E-05  5.46295494E 00
          3  -5,91219487E-02  7,04304385E-03 -7,04479806E-01
          4   1,23179391E 00  4,52240999E-02  5,79232717E 00
          5   8,35810430E-01  2,51803122E-03  1,66562500E 01
ANALYSIS OF VARIANCE
   SSREG          SSYX              DFREG          DFYX            S2REG            S2YX             F
 3.59964279E 03  5,63604126E 01                4              5  8.99910698E 02  1,12720824E 01  7,98353539E 01

          K    COEFFICIENT      VARIANCE        T-TEST
          1   2,63474232E 00  6,85596123E-01  3,18202782E 00
          2   5,18953409E-02  3,03501615E-03  9,41993408E-01
          3   9,28254686E-02  3,91741440E-01  1,48308961E-01
          4   7,88094081E-01  2,50554755E 00  4,97882362E-01
ANALYSIS OF VARIANCE
   SSREG          SSYX              DFREG          DFYX            S2REG            S2YX             F
 3.59684860E 03  5,91546021E 01                3              6  1.19894952E 03  9,85910034E 00  1,21608410E 02

          K    COEFFICIENT      VARIANCE        T-TEST
          1   2,75364906E 00  5,49767420E-01  3,71380496E 00
          2   2,89138339E-02  7,91038539E-04  1,02803254E 00
          3   3,76819748E-01  5,80593478E-02  1,56385894E 00
ANALYSIS OF VARIANCE
   SSREG          SSYX              DFREG          DFYX            S2REG            S2YX             F
 3,57273666E 03  8,32665405E 01                2              7  1.78636833E 03  1,18952200E 01  1,50175306E 02

          K    COEFFICIENT      VARIANCE        T-TEST
          1   3,23539412E 00  5,48814580E-01  4,36731350E 00
          2   7,19571728E-02  4,03931740E-05  1,13219205E 01
ANALYSIS OF VARIANCE
```

```
     SSREG              SSYX              DFREG           DFYX         S2REG             S2YX               F
2.04793738E 03    1.60806583E 03              1              8    2.04793738E 03    2.01008228E 02    1.01883260E 01

            K     COEFFICIENT        VARIANCE         T-TEST
            1     7.99987769E 00   6.28150713E 00   3.19191575E 00

PREDICTED Y AND STANDARDIZED DEVIATE
        MODEL        YHAT             ST.DEV                  MC        X                 Y
                                                              1   -2.50000000E 00   4.99999998E-02
            6     4.64210035E-02   1.36775103E-02
            5    -1.98840929E-01   5.56058384E-01
            4    -5.85379529E 00   1.75844671E 00
            3    -5.95777613E 00   1.91335298E 00
            2    -3.79167354E 00   1.11386925E 00
            1     7.99987769E 00  -5.60729668E-01
                                                              2   -1.50000000E 00   1.99999999E-01
            6     2.09005948E-01  -3.44171803E-02
            5     5.94543397E-01  -8.81644219E-01
            4     7.45234981E-01  -1.62398361E-01
            3     1.26231796E 00  -3.38326398E-01
            2     2.68896931E 00  -7.21661121E-01
            1     7.99987769E 00  -5.50149702E-01
                                                              3   -5.00000000E-01   5.00000000E-01
            6     4.89854638E-01   3.87715683E-02
            5     2.67301686E-01   5.19986197E-01
            4     2.22747031E 00  -5.14527410E-01
            3     2.70564300E 00  -7.02451885E-01
            2     3.23314545E 00  -7.92458467E-01
            1     7.99987769E 00  -5.28989762E-01
                                                              4    7.50000000E-01   2.09999999E 00
            6     2.11078155E 00  -4.12028506E-02
            5     2.04672393E 00   1.19050356E-01
            4     3.27728859E 00  -3.50655667E-01
            3     2.91948122E 00  -2.60987896E-01
            2     3.25246987E 00  -3.34151447E-01
            1     7.99987769E 00  -4.16136771E-01
                                                              5    1.00000000E 00   2.71828181E 00
            6     2.70154500E 00   6.39614603E-02
            5     2.71689248E 00   3.10457620E-03
            4     3.56755716E 00  -2.52956852E-01
            3     3.15938261E 00  -1.40481526E-01
            2     3.30735129E 00  -1.70797020E-01
            1     7.99987769E 00  -3.72527428E-01
                                                              6    1.75000000E 00   5.50000000E 00
            6     5.64190835E 00  -5.42317566E-01
            5     5.72793370E 00  -5.09339236E-01
            4     5.36315608E 00   4.07589911E-02
            3     5.24773258E 00   8.03419789E-02
            2     4.41643333E 00   3.14173389E-01
            1     7.99987769E 00  -1.76324170E-01
                                                              7    2.00000000E 00   7.38905603E 00
            6     7.25153667E 00   5.25544681E-01
            5     7.29141164E 00   2.18195541E-01
            4     6.61418509E 00   2.30795477E-01
            3     6.69344968E 00   2.21536299E-01
            2     5.53802365E 00   5.36695279E-01

            1     7.99987769E 00  -4.30831569E-02
                                                              8    2.75000000E 00   1.55000000E 01
            6     1.55674634E 01  -2.57818464E-01
            5     1.54842770E 01   3.51344952E-02
            4     1.48943912E 01   1.80380719E-01
            3     1.51377850E 01   1.15358032E-01
            2     1.45525568E 01   2.74705254E-01
            1     7.99987769E 00   5.29007010E-01
                                                              9    3.00000000E 00   2.00855360E 01
            6     2.00521860E 01   1.27450494E-01
            5     2.00812030E 01   9.68253775E-03
            4     2.01158798E 01  -9.03789885E-03
            3     1.99538438E 01   4.19412446E-02
            2     2.07209871E 01  -1.84245074E-01
            1     7.99987769E 00   8.52439165E-01

END OF DATA

END OF PROGRAM
```

APPENDIX A
PROGRAM LISTING

```
***EOF
$      IDENT    AAV,GE,FORTRAN,LSPF
$      OPTION   FORTRAN,GO
$      FORTRAN LSTOU,DECK,STAB
$      INCODE   IBMF
*LSPF       LEAST SQUARES POLYNOMIAL CURVE FIT
*                           CD600D6.001       DATE 05/05/65
*     SEQUENTIAL LEAST SQUARES
*     DIMENSIONS FOR NX=ANY VALUE, NT=15,NP=30
*     FOR NX LESS THAN 201, NO TAPES ARE USED
*     AND XX PRODUCTS ARE REGENERATED
*     OTHERWISE X,Y, AND XX ARE WRITTEN ON TAPE 3
*     AND READ BACK FOR PREDICTIONS IN LOTS OF 25
*      FOR NW=1, READ X,Y,W WHERE W=WEIGHT
*     FOR NW=0, READ X,Y   WITH NO WEIGHT
      DIMENSION YY(15),B(15),KSEQ(15),JXSEQ(15),XXT(15),SXX(15),A(120)
      DIMENSION KORD(31),SRSYX(15),XYT(15),SXY(15),C(120)
      DIMENSION STXN(15),YSAVE(15),XYS(426),SY(15),S(120)
      DIMENSION WXXT(15),WXYT(15),WSXX(15),WSXY(15)
      DIMENSION CNX(15)
      DATA CNX/6H=C X  ,6H+C X  ,6H+C X  ,6H+C X  ,6H+C X  ,6H+C X  ,6H+
     AC X  ,6H+C X  ,6H+C X  ,6H+C  X ,6H+C  X ,6H+C  X ,6H+C  X ,6H+C
     BX ,6H+C  X /
      NAMELIST/PARA/NX,NT,NP,NW,IOP1,IOP2,XZER
      NAMELIST/DATA/X,Y,W/EXP/KSEQ
    2 FORMAT(I8,I6,I6,I6,I6,I6,I6,I6,I6,I7,I7,I7,I7,I7,I7)
    3 FORMAT(2H YA6,A6,A6,A6,A6,A6,A6,A6,A6,A6,1X,A6,1X,A6,1X,A6,1X,A6,1
     AX,A6)
    4 FORMAT(12H0GIVEN MODEL/)
    5 FORMAT(25H SEQUENTIAL LEAST SQUARES)
    6 FORMAT(22H POLYNOMIAL FITTING BY)
    7 FORMAT(I5,I6,I6,I6,I6,I6,I6,I6,I6,I7,I7,I7,I7,I7,I7)
   79 FORMAT(15H0END OF PROGRAM)
```

```
   89 FORMAT(12H0END OF DATA)                                              LSPF0320
  102 FORMAT(1PE17.8,1PE17.8,1PE17.8,1PE17.8,1PE17.8,1PE17.8)              LSPF0330
  109 FORMAT(13H0MATRIX S(NS))                                             LSPF0340
  112 FORMAT(14H0VECTOR SY(NT))                                            LSPF0350
  113 FORMAT(7H0 SYSQ=1PE15.8,5H   NX=I4,4H  M=I4,5H   NS=I4)              LSPF0360
  114 FORMAT(21H ANALYSIS OF VARIANCE)                                     LSPF0370
  115 FORMAT(109H        SSREG             SSYX              DFREG         LSPF0380
     1  DFYX              S2REG             S2YX              F)           LSPF0390
  120 FORMAT(1PE17.8,1PE17.8,2I17,1PE17.8,1PE17.8,1PE17.8)                 LSPF0400
  121 FORMAT(63H0                   K     COEFFICIENT        VARIANCE      LSPF0410
     1 T-TEST)                                                             LSPF0420
  124 FORMAT(I17,1PE17.8,1PE17.8,1PE17.8)                                  LSPF0430
  128 FORMAT(37H0PREDICTED Y AND STANDARDIZED DEVIATE)                     LSPF0440
  129 FORMAT(92H                 MODEL       YHAT              ST.DEV      LSPF0450
     1          MC       X                 Y)                              LSPF0460
  134 FORMAT(I68,1PE17.8,1PE17.8)                                          LSPF0470
  142 FORMAT(29H0RIGHT DECOMPOSITION OF S(NS))                             LSPF0480
  143 FORMAT(21H0PRODUCT VECTOR B(NT))                                     LSPF0490
  148 FORMAT(14H0FORWARD YY(N))                                            LSPF0500
  162 FORMAT(13H0COEFFICIENTS)                                             LSPF0510
  163 FORMAT(5H    J=I3,6H    YJ=1PE15.8)                                  LSPF0520
  164 FORMAT(10H0DIAGONALS)                                                LSPF0530
  165 FORMAT(5H   IB=I3,6H   YIB=1PE15.8)                                  LSPF0540
  166 FORMAT(37H0COEFFICIENTS AND DIAGONALS COMPLETED)                     LSPF0550
  175 FORMAT(7H0 SSYX=1PE15.8,27H   SET SSYX=1.0 AND CONTINUE)             LSPF0560
*         BEGIN                                                            LSPF0570
   97 READ(5,PARA)                                                         LSPF0580
      IF(NX)75,75,31                                                       LSPF0590
*         READ MODEL DESCRIPTION                                           LSPF0600
```

```
   31 READ(5,EXP)
      NS=(NT*NT-NT)/2+NT
      IOP1=IOP1-1
      IOP2=IOP2-1
      NW=NW-1
*     ZERO S,SY,KORD
      LK=0
      DO 33 LR=1,15
      DO 32 L=LR,15
      LK=LK+1
   32 S(LK)=0.0
   33 SY(LR)=0.0
      DO 80 LR=1,31
   80 KORD(LR)=0
*     FORM KORD FROM KSEQ
      DO 34 M=1,NT
      KL=KSEQ(M)
   34 KORD(KL+1)=1
      WRITE(6,6)
      WRITE(6,5)
      WRITE(6,4)
      WRITE(6,2)(KSEQ(I),I=1,NT)
      WRITE(6,3)(CNX(I),I=1,NT)
      WRITE(6,7)(I,I=1,NT)
*     FORM JXSEQ
      NST=NT
   18 KMX=0
      IS=1
      DO 20 IL=1,NT
      KST=KSEQ(IL)
      IF(KST-KMX)20,19,19
   19 KMX=KST
      IS=IL
```

```
   20 CONTINUE
      JXSEQ(NST)=IS
      NST=NST-1
      IF(NST)98,98,91
   91 KSEQ(IS)=-1
      GO TO 18
*     START
   98 N=0
      NP1=NP+1
      NXYMX=201
      IF(NX-NXYMX)11,101,101
*     X,Y,XX ON TAPE 3
*     POSITION TAPE 3
  101 ICASE=0
      REWIND 3
      NXMAX=26
      GO TO 85
*     X,Y, IN CORE, XX REGENERATED
   11 ICASE=-1
      NXMAX=NXYMX
   85 NT1=NXMAX-1
      LN1=NT1+NT1
      NT2=LN1+NT1*NT
      IDY=NT1
      NTR=0
      NXX=0
      NXY=IDY
      LN=LN1
      SYSQ=0.0
*     READ X,Y OR X,Y,W
   99 N=N+1
   41 READ (5,DATA)
   56 X=X-XZER
```

```
      NXX=NXX+1
      NXY=NXY+1
      IF(NXX-NXMAX)45,44,44
*     WRITE X,Y,XX ON TAPE 3
   44 WRITE(3)(XYS(I),I=1,NT2)
      NXX=1
      NXY=NXMAX
      LN=LN1
      NTR=NTR+1
   45 XYS(NXX)=X
      XYS(NXY)=Y
*     SEQUENTIAL POWERS OF X
      L=1
      M=0
      XX=1.0
      XY=Y
   35 IF(KORD(L))36,36,103
  103 M=M+1
      XXT(M)=XX
      IF(NW)76,57,57
   57 WXXT(M)=XX*W
      WXYT(M)=XY*W
      GO TO 36
   76 XYT(M)=XY
   36 IF(L-NP1)104,37,37
  104 L=L+1
      XX=XX*X
      XY=XY*X
      GO TO 35
*     DESIRED ORDERS
   37 DO 38 LJ=1,NT
      JX=JXSEQ(LJ)
      IF(NW)111,77,77
```

```
   77 WSXY(JX)=WXYT(LJ)                                                LSPF1600
      WSXX(JX)=WXXT(LJ)                                                LSPF1610
      GO TO 38                                                         LSPF1620
  111 SXY(JX)=XYT(LJ)                                                  LSPF1630
   38 SXX(JX)=XXT(LJ)                                                  LSPF1640
*     FORM S AND SY                                                    LSPF1650
      LK=0                                                             LSPF1660
      DO 10 LI=1,NT                                                    LSPF1670
      IF(NW)168,167,167                                                LSPF1680
  167 SXXLI=WSXX(LI)                                                   LSPF1690
      GO TO 169                                                        LSPF1700
  168 SXXLI=SXX(LI)                                                    LSPF1710
  169 DO 39 LJ=LI,NT                                                   LSPF1720
      LK=LK+1                                                          LSPF1730
   39 S(LK)=S(LK)+SXXLI*SXX(LJ)                                        LSPF1740
      IF(NW)173,172,172                                                LSPF1750
  172 SXYLI=WSXY(LI)                                                   LSPF1760
      GO TO 10                                                         LSPF1770
  173 SXYLI=SXY(LI)                                                    LSPF1780
   10 SY(LI)=SY(LI)+SXYLI                                              LSPF1790
      WYTY=Y*Y                                                         LSPF1800
      IF(NW)171,170,170                                                LSPF1810
  170 WYTY=W*WYTY                                                      LSPF1820
  171 SYSQ=SYSQ+WYTY                                                   LSPF1830
      IF(ICASE)86,26,26                                                LSPF1840
*     STORE XX IN XYS                                                  LSPF1850
   26 DO 27 L=1,NT                                                     LSPF1860
      LN=LN+1                                                          LSPF1870
   27 XYS(LN)=SXX(L)                                                   LSPF1880
*     IF LAST X,Y READ, GO TO SECTION JTG WITH S AND SY                LSPF1890
   86 IF(N-NX)99,105,105                                               LSPF1900
  105 IF(NTR)28,28,106                                                 LSPF1910
*     WRITE X,Y,XX AND EOF ON TAPE 3 AND REWIND                        LSPF1920
```

```
  106 NTR=NTR+1                                                    LSPF1930
      IXX=NXX                                                      LSPF1940
      IXY=NT1                                                      LSPF1950
      ITST=NXY                                                     LSPF1960
      LASTX=IXX                                                    LSPF1970
      ICNT=-1                                                      LSPF1980
   87 IXX=IXX+1                                                    LSPF1990
      IXY=IXY+1                                                    LSPF2000
      XYS(IXX)=XYS(IXY)                                            LSPF2010
      IF(IXY-ITST)87,107,107                                       LSPF2020
  107 IF(ICNT)108,88,88                                            LSPF2030
  108 LASTY=IXX                                                    LSPF2040
      IXY=LN1                                                      LSPF2050
      ITST=LN                                                      LSPF2060
      ICNT=0                                                       LSPF2070
      GO TO 87                                                     LSPF2080
   88 WRITE(3)(XYS(I),I=1,IXX)                                     LSPF2090
      END FILE 3                                                   LSPF2100
      REWIND 3                                                     LSPF2110
      LSTXX=IXX                                                    LSPF2120
   28 IF(IOP2)100,81,81                                            LSPF2130
   81 WRITE(6,109)                                                 LSPF2140
      WRITE(6,102)(S(MP),MP=1,NS)                                  LSPF2150
      WRITE(6,112)                                                 LSPF2160
      WRITE(6,102)(SY(MP),MP=1,NT)                                 LSPF2170
      GO TO 100                                                    LSPF2180
*     RETURN FROM SECTION JTG WITH A(COEFF) AND C(DIAG)            LSPF2190
*     ANALYSIS OF VARIANCE TABLE                                   LSPF2200
   13 NF=NT                                                        LSPF2210
      LK=1                                                         LSPF2220
      MK=1                                                         LSPF2230
      M=NT                                                         LSPF2240
      WRITE(6,113)SYSQ,NX,M,NS                                     LSPF2250
```

```
      IVFIT=-1
   14 NY=1
      WRITE(6,114)
      WRITE(6,115)
      IFIT=-1
      SSREG=0.0
   15 SSREG=SSREG+A(LK)*SY(NY)
      IF(NY-NF)116,16,16
  116 NY=NY+1
      LK=LK+1
      GO TO 15
   16 SSYX=SYSQ-SSREG
      IF(SSYX)176,176,177
  176 WRITE(6,175)SSYX
      SSYX=1.0
  177 DFREG=NF
      NFYX=NX-NF
      S2REG=SSREG/DFREG
      IF(NFYX)117,117,118
  117 S2YX=0.0
      FF=0.0
      TTEST=0.0
      VAR=0.0
      SRYX=0.0
      IFIT=0
      IVFIT=0
      GO TO 119
  118 DFYX=NFYX
      S2YX=SSYX/DFYX
      FF=S2REG/S2YX
      SRYX=SQRT(S2YX)
  119 SRSYX(NF)=SRYX
      WRITE(6,120)SSREG,SSYX,NF,NFYX,S2REG,S2YX,FF
```

```
      WRITE(6,121)
      MN=0
   21 MN=MN+1
      COMK=A(MK)
      IF(IFIT)122,123,123
  122 VAR=C(MK)*S2YX
      SRVAR=SQRT(VAR)
      TTEST=COMK/SRVAR
  123 WRITE(6,124)MN,COMK,VAR,TTEST
      MK=MK+1
      IF(MN-NF)21,125,125
  125 IF(IOP1)17,126,126
  126 NF1=NF-1
      IF(NF1)17,17,127
  127 NF=NF1
      LK=LK+1
      GO TO 14
*     PREDICTED Y AND STANDARDIZED DEVIATE
   17 WRITE(6,128)
      WRITE(6,129)
      MT=0
      IF(NTR)130,130,29
*     XX REGENERATED
  130 LX=0
      MCX=0
      MCY=NT1
      MSTRT=LN1
      NT1=NX
   90 IF(ICASE)131,22,22
  131 LN=LN1
      LX=LX+1
      X=XYS(LX)
*     SEQUENTIAL POWERS OF X
      L=1
```

```
      M=0
      XX=1.0
   92 IF(KORD(L))93,93,132
  132 M=M+1
      XXT(M)=XX
   93 IF(L-NP1)133,94,94
  133 L=L+1
      XX=XX*X
      GO TO 92
*     DESIRED ORDER
   94 DO 95 LJ=1,NT
      JX=JXSEQ(LJ)
   95 SXX(JX)=XXT(LJ)
*     STORE XX IN XYS
      DO 96 L=1,NT
      LN=LN+1
   96 XYS(LN)=SXX(L)
      MM=MSTRT
      GO TO 22
   12 MCX=0
      MCY=NXY
      MM=MSTRT
   22 MCX=MCX+1
      MCY=MCY+1
      MT=MT+1
      DO 23 NN=1,NT
      MM=MM+1
   23 STXN(NN)=XYS(MM)
      X=XYS(MCX)
      Y=XYS(MCY)
      WRITE(6,134)MT,X,Y
      IFIT=IVFIT
      MF1=NT
```

```
      MK=0
   24 MF=MF1
      SRYX=SRSYX(MF)
      YHAT=0.0
      DO 25 MN=1,MF
      MK=MK+1
      COMK=A(MK)
      XNP=STXN(MN)
   25 YHAT=YHAT+COMK*XNP
      IF(IFIT)135,110,110
  110 IFIT=-1
      STD=0.0
      GO TO 136
  135 STD=(Y-YHAT)/SRYX
  136 WRITE(6,124)MF,YHAT,STD
      IF(IOP1)43,137,137
  137 MF1=MF-1
      IF(MF1)43,43,24
   43 IF(MCX-NT1)90,138,138
  138 IF(ICASE)40,139,139
  139 IF(NTR)40,40,29
   29 NTR=NTR-1
      IF(NTR)30,30,82
*     READ X,Y,XX FROM TAPE 3
   82 READ(3)(XYS(I),I=1,NT2)
      NXY=NT1
      MSTRT=LN1
      GO TO 12
*     READ X,Y,XX FROM TAPE 3-LAST ENTRY
   30 READ(3)(XYS(I),I=1,LSTXX)
      NXY=LASTX
      MSTRT=LASTY
      NT1=LASTX
```

```
      GO TO 12
*     SEQ LEAST SQUARES SECTION JTG
*     RIGHT DECOMPOSITION OF S(NS) AND SY(NT)
  100 N=NT
      I=1
      L=N
   51 IF(I-N)140,54,54
  140 J=I
      I=I+1
   53 J=J+1
      K=1
      ID=0
      IT=N
      SUM=0.0
   52 IID=I+ID
      JID=J+ID
      KID=K+ID
      SUM=SUM+S(IID)*S(JID)/S(KID)
      IT=IT-1
      ID=ID+IT
      K=K+1
      IF(K-I)52,141,141
  141 L=L+1
      S(L)=S(L)-SUM
      IF(J-N)53,51,51
   54 KS=0
      IB=0
      IPM=0
      IDS=0
      IKS=0
      IP=1
      DO 55 J=1,N
   55 B(J)=SY(J)
```

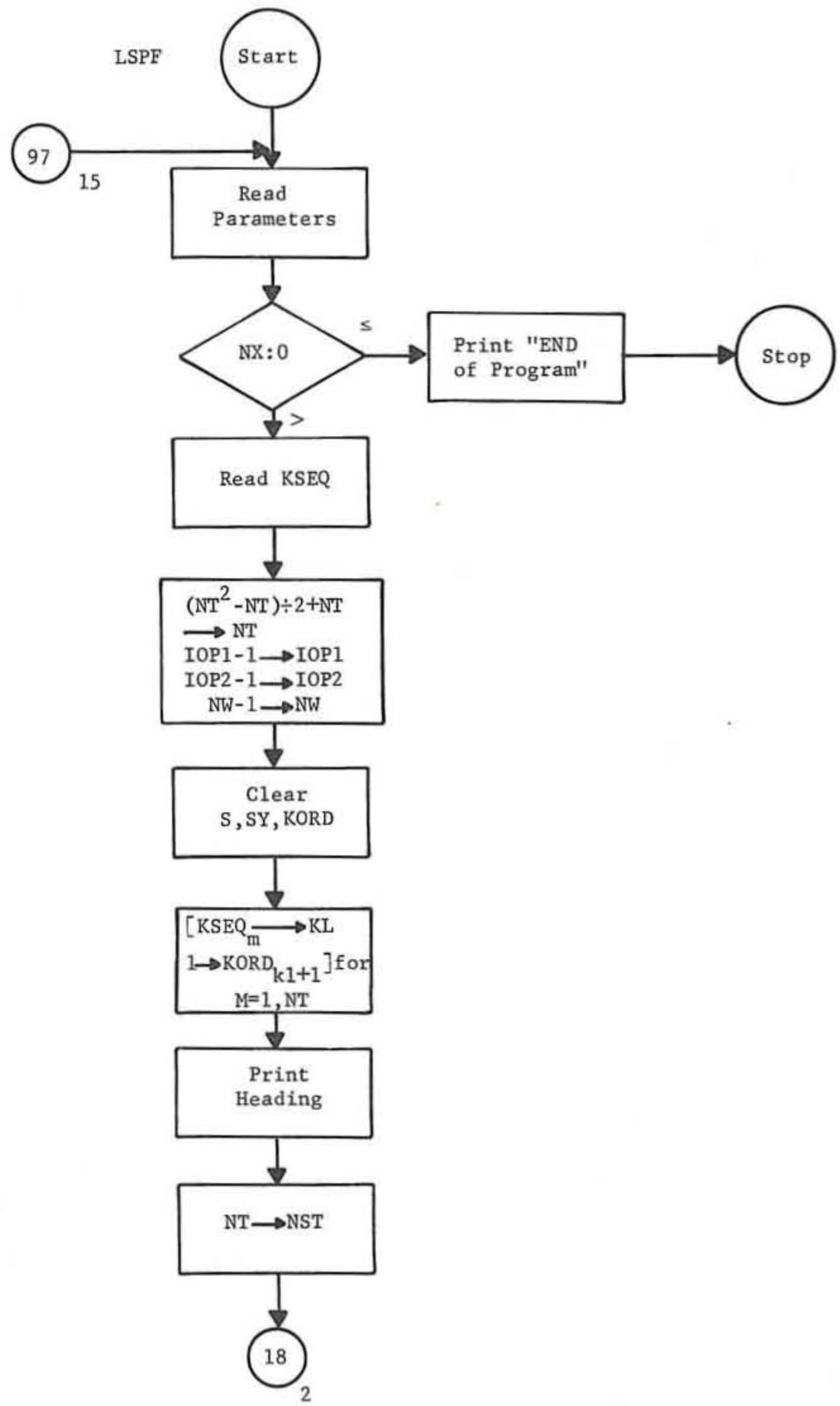
```
      IF(IOP2)49,50,50                                                  LSPF3920
   50 WRITE(6,142)                                                      LSPF3930
      WRITE(6,102)(S(MP),MP=1,NS)                                       LSPF3940
      WRITE(6,143)                                                      LSPF3950
      WRITE(6,102)(B(MP),MP=1,NT)                                       LSPF3960
*     FORWARD SUBSTITUTION                                              LSPF3970
   49 IPP=-1                                                            LSPF3980
   58 J=IP-1                                                            LSPF3990
   59 J=J+1                                                             LSPF4000
      SUM=0.0                                                           LSPF4010
      K=0                                                               LSPF4020
      ID=0                                                              LSPF4030
      IT=NT                                                             LSPF4040
   60 K=K+1                                                             LSPF4050
      IF(K-J)144,61,61                                                  LSPF4060
  144 JID=J+ID                                                          LSPF4070
      KID=K+ID                                                          LSPF4080
      SUM=SUM+S(JID)*YY(K)/S(KID)                                       LSPF4090
      IT=IT-1                                                           LSPF4100
      ID=ID+IT                                                          LSPF4110
      GO TO 60                                                          LSPF4120
   61 YY(J)=B(J)-SUM                                                    LSPF4130
      IF(J-N)59,145,145                                                 LSPF4140
  145 IF(IPP)146,63,63                                                  LSPF4150
  146 IF(IOP2)48,147,147                                                LSPF4160
  147 WRITE(6,148)                                                      LSPF4170
   48 IDDS=ID                                                           LSPF4180
      IDD=ID                                                            LSPF4190
      DO 62 J=1,N                                                       LSPF4200
      YJ=YY(J)                                                          LSPF4210
      YSAVE(J)=YJ                                                       LSPF4220
      IF(IOP2)62,47,47                                                  LSPF4230
   47 WRITE(6,102)YJ                                                    LSPF4240
```

```
   62 CONTINUE                                                          LSPF4250
   63 ID=IDD                                                            LSPF4260
      DO 64 I=1,N                                                       LSPF4270
   64 B(I)=0.0                                                          LSPF4280
*     BACKWARD SUBSTITUTION                                             LSPF4290
   65 SUM=0.0                                                           LSPF4300
      K=J                                                               LSPF4310
      IT=N                                                              LSPF4320
      ID=ID-IKS                                                         LSPF4330
   66 K=K+1                                                             LSPF4340
      IF(K-N)149,149,67                                                 LSPF4350
  149 JID=J+ID                                                          LSPF4360
      SUM=SUM+S(JID)*YY(IT)                                             LSPF4370
      IT=IT-1                                                           LSPF4380
      ID=ID-1                                                           LSPF4390
      GO TO 66                                                          LSPF4400
   67 JID=J+ID                                                          LSPF4410
      YY(J)=(YY(J)-SUM)/S(JID)                                          LSPF4420
      J=J-1                                                             LSPF4430
      IF(J-IPM)150,150,65                                               LSPF4440
  150 IF(IPP)151,69,69                                                  LSPF4450
*     COEFFICIENTS                                                      LSPF4460
  151 IPP=0                                                             LSPF4470
      IF(IOP2)68,152,152                                                LSPF4480
  152 WRITE(6,162)                                                      LSPF4490
   68 DO 174 J=1,N                                                      LSPF4500
      YJ=YY(J)                                                          LSPF4510
      IF(IOP2)83,153,153                                                LSPF4520
  153 WRITE(6,163)J,YJ                                                  LSPF4530
   83 KSJ=KS+J                                                          LSPF4540
      A(KSJ)=YJ                                                         LSPF4550
  174 CONTINUE                                                          LSPF4560
```

```
*      DIAGONALS                                                LSPF4570
       IF(IOP2)70,154,154                                       LSPF4580
   154 WRITE(6,164)                                             LSPF4590
       GO TO 70                                                 LSPF4600
    69 YIB=YY(IB)                                               LSPF4610
       IF(IOP2)84,155,155                                       LSPF4620
   155 WRITE(6,165)IB,YIB                                       LSPF4630
    84 KSB=KS+IB                                                LSPF4640
       C(KSB)=YIB                                               LSPF4650
    70 IF(IB-N)156,71,71                                        LSPF4660
   156 IB=IB+1                                                  LSPF4670
       B(IB)=1.0                                                LSPF4680
       IP=IB                                                    LSPF4690
       IPM=IP-1                                                 LSPF4700
       IF(IPM)58,58,157                                         LSPF4710
   157 DO 78 J=1,IPM                                            LSPF4720
    78 YY(J)=0.0                                                LSPF4730
       GO TO 59                                                 LSPF4740
*      COMPLETION OF COEFF AND DIAG FOR ONE N                   LSPF4750
    71 KS=KS+N                                                  LSPF4760
       IF(IOP2)46,158,158                                       LSPF4770
   158 WRITE(6,166)                                             LSPF4780
       DO 72 J=1,KS                                             LSPF4790
       WRITE(6,102)A(J),C(J)                                    LSPF4800
    72 CONTINUE                                                 LSPF4810
    46 IF(IOP1)74,159,159                                       LSPF4820
   159 N=N-1                                                    LSPF4830
       IF(N)74,74,160                                           LSPF4840
   160 IPM=0                                                    LSPF4850
       IPP=-1                                                   LSPF4860
       IB=0                                                     LSPF4870
       IKS=IKS+1                                                LSPF4880
       IDS=IDS+IKS                                              LSPF4890
```

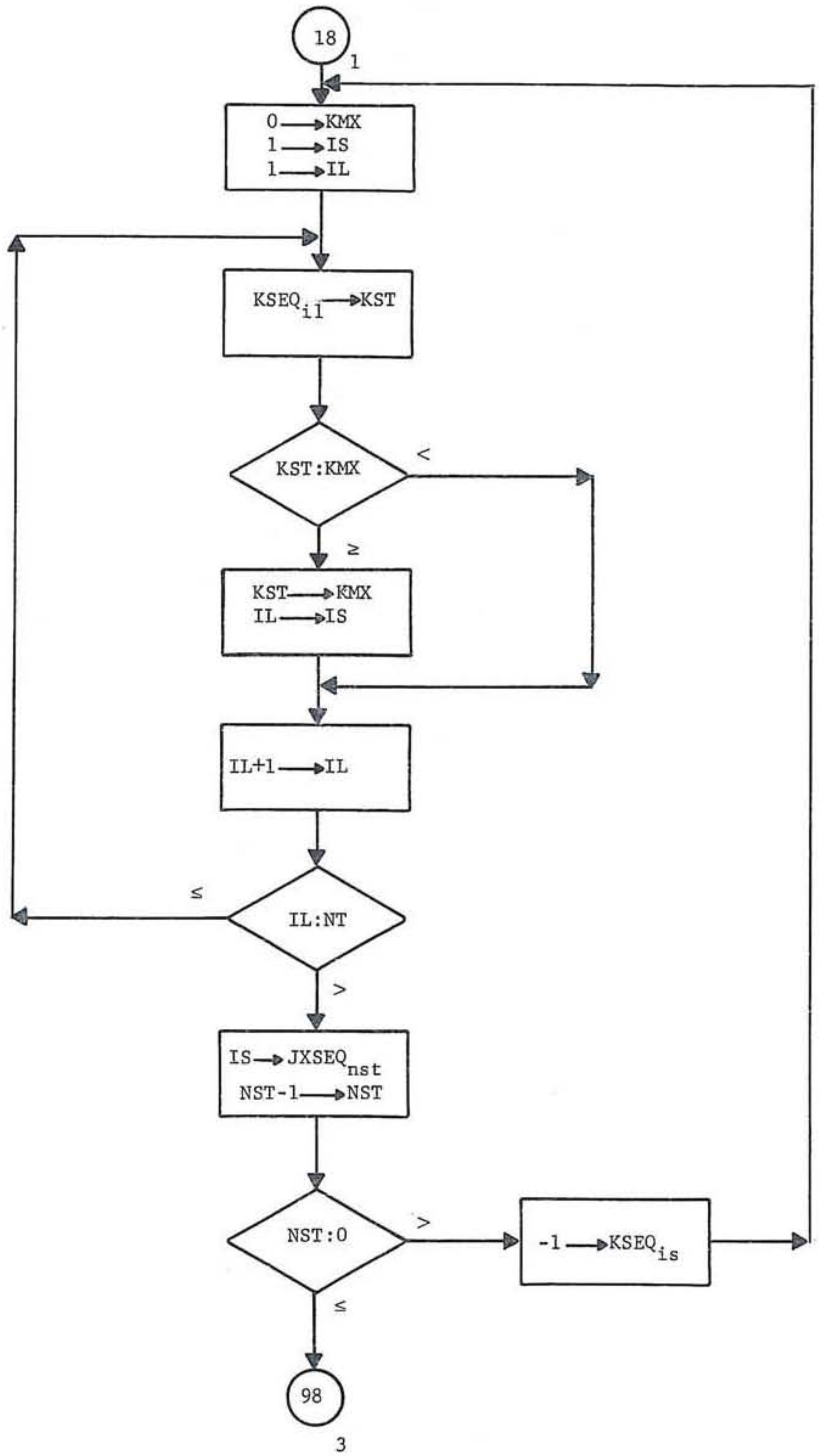
```
      IDD=IDDS+IKS-IDS                                                  LSPF4900
      ID=IDD                                                            LSPF4910
      DO 73 J=1,N                                                       LSPF4920
   73 YY(J)=YSAVE(J)                                                    LSPF4930
      GO TO 65                                                          LSPF4940
   74 IF(IOP2)13,161,161                                                LSPF4950
  161 WRITE(6,166)                                                      LSPF1960
      GO TO 13                                                          LSPF4970
   40 WRITE(6,89)                                                       LSPF4980
      GO TO 97                                                          LSPF4990
   75 WRITE(6,79)                                                       LSPF5000
      STOP                                                              LSPF5010
      END                                                               LSPF5020
```

APPENDIX B
FLOW CHARTS



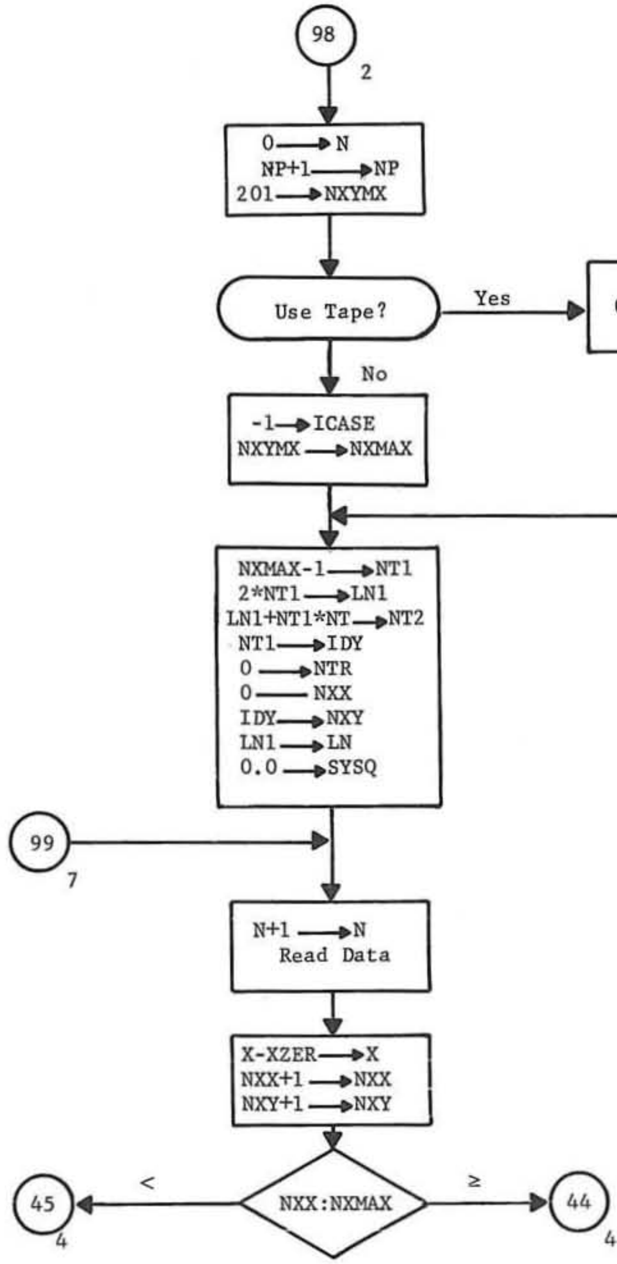
LSPF

1


18
1
0 → KMX
1 → IS
1 → IL
KSEQ_il → KST
KST:KMX
<
≥
KST → KMX
IL → IS
IL+1 → IL
IL:NT
≤
>
IS → JXSEQ_nst
NST-1 → NST
NST:0
>
-1 → KSEQ_is
≤
98
3

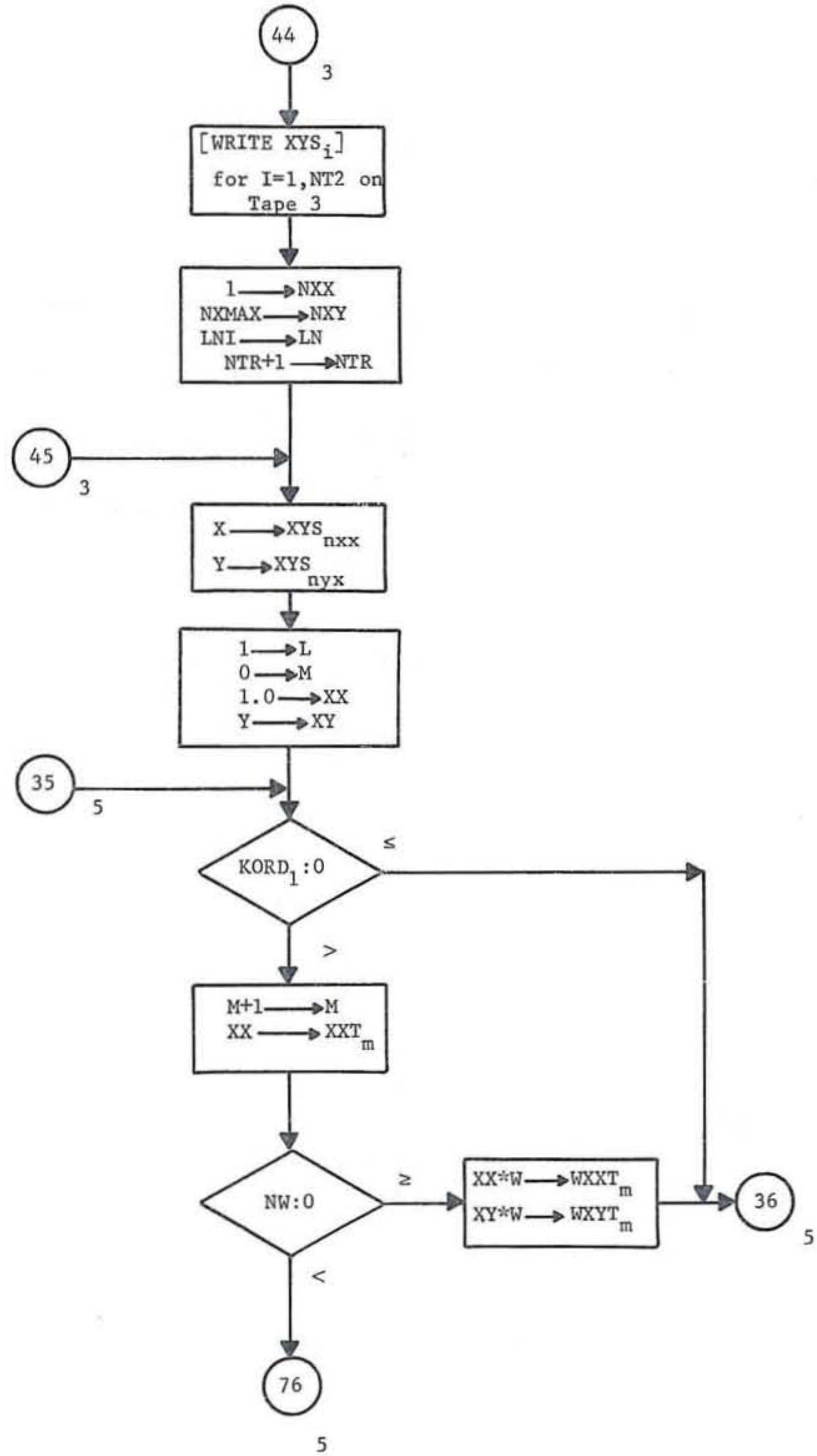
LSPF

(2)



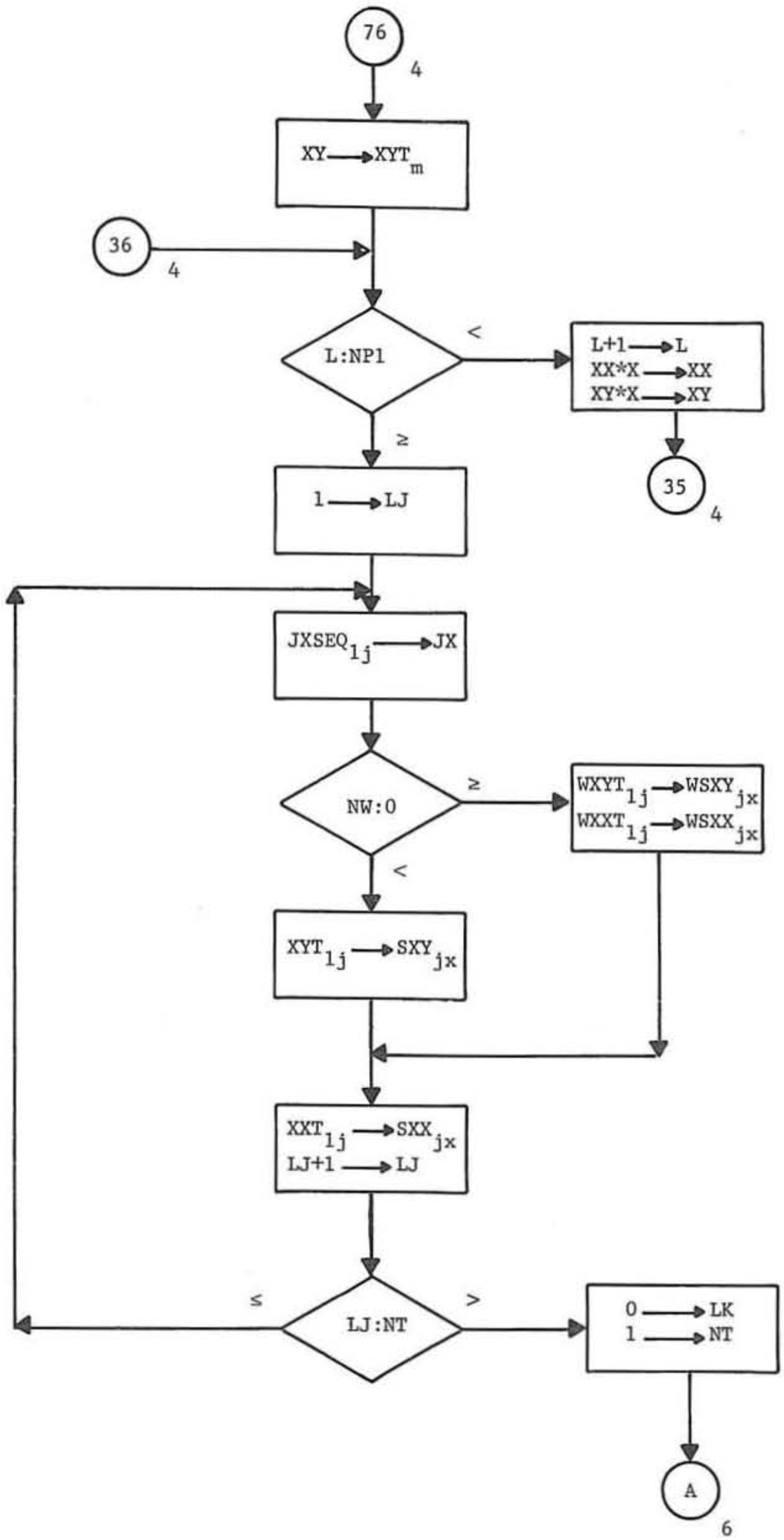
98
2
0 → N
NP+1 → NP
201 → NXYMX
Use Tape?
Yes
Rewind 3
0 → ICASE
26 → NXMAX
No
-1 → ICASE
NXYMX → NXMAX
NXMAX-1 → NT1
2*NT1 → LN1
LN1+NT1*NT → NT2
NT1 → IDY
0 → NTR
0 — NXX
IDY → NXY
LN1 → LN
0.0 → SYSQ
99
7
N+1 → N
Read Data
X-XZER → X
NXX+1 → NXX
NXY+1 → NXY
NXX:NXMAX
<
≥
45
4
44
4

LSPF
(3)


44
3
[WRITE XYS_i]
for I=1,NT2 on
Tape 3
1 → NXX
NXMAX → NXY
LNI → LN
NTR+1 → NTR
45
3
X → XYS nxx
Y → XYS nyx
1 → L
0 → M
1.0 → XX
Y → XY
35
5
KORD1:0
≤
>
M+1 → M
XX → XXT m
NW:0
≥
<
XX*W → WXXT m
XY*W → WXYT m
36
5
76
5

LSPF

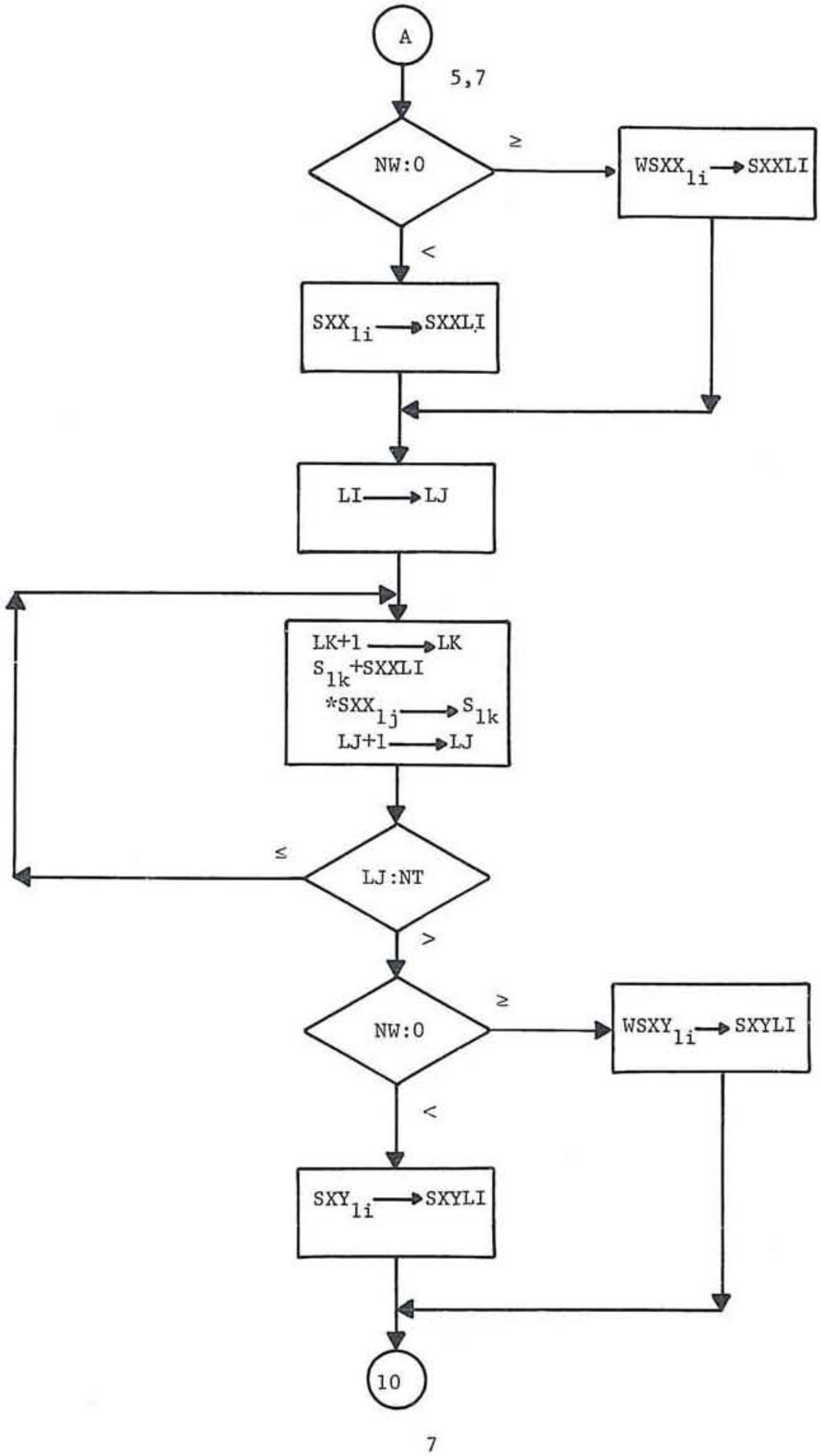
(4)



76
4
XY → XYT m
36
4
L:NP1
<
L+1 → L
XX*X → XX
XY*X → XY
35
4
≥
1 → LJ
JXSEQ 1j → JX
NW:0
≥
WXYT 1j → WSXY jx
WXXT 1j → WSXX jx
<
XYT 1j → SXY jx
XXT 1j → SXX jx
LJ+1 → LJ
≤
LJ:NT
>
0 → LK
1 → NT
A
6

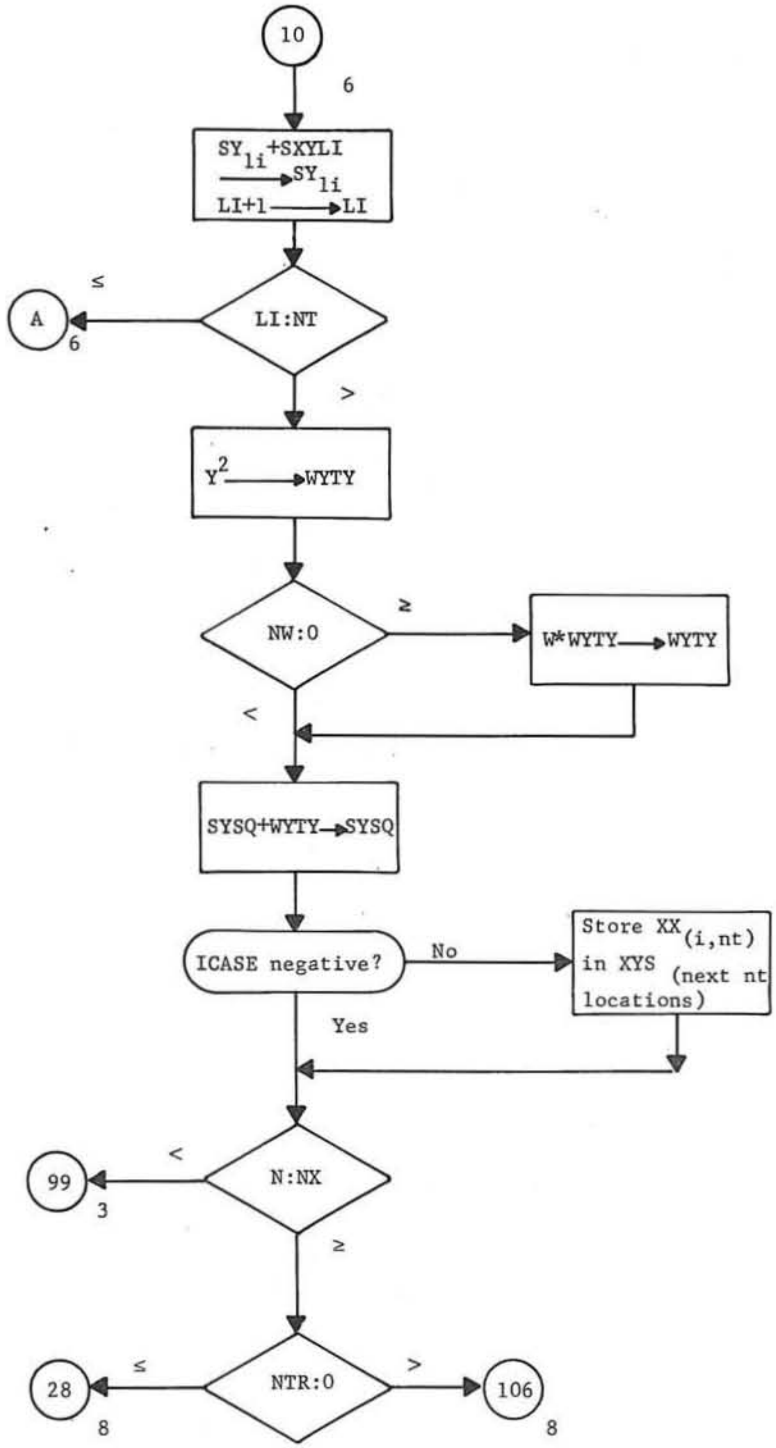
LSPF

(5)


A
5,7
NW:0
≥
WSXX1i → SXXLI
<
SXX1i → SXXLI
LI → LJ
LK+1 → LK
S1k+SXXLI
*SXX1j → S1k
LJ+1 → LJ
≤
LJ:NT
>
NW:0
≥
WSXY1i → SXYLI
<
SXY1i → SXYLI
10
7

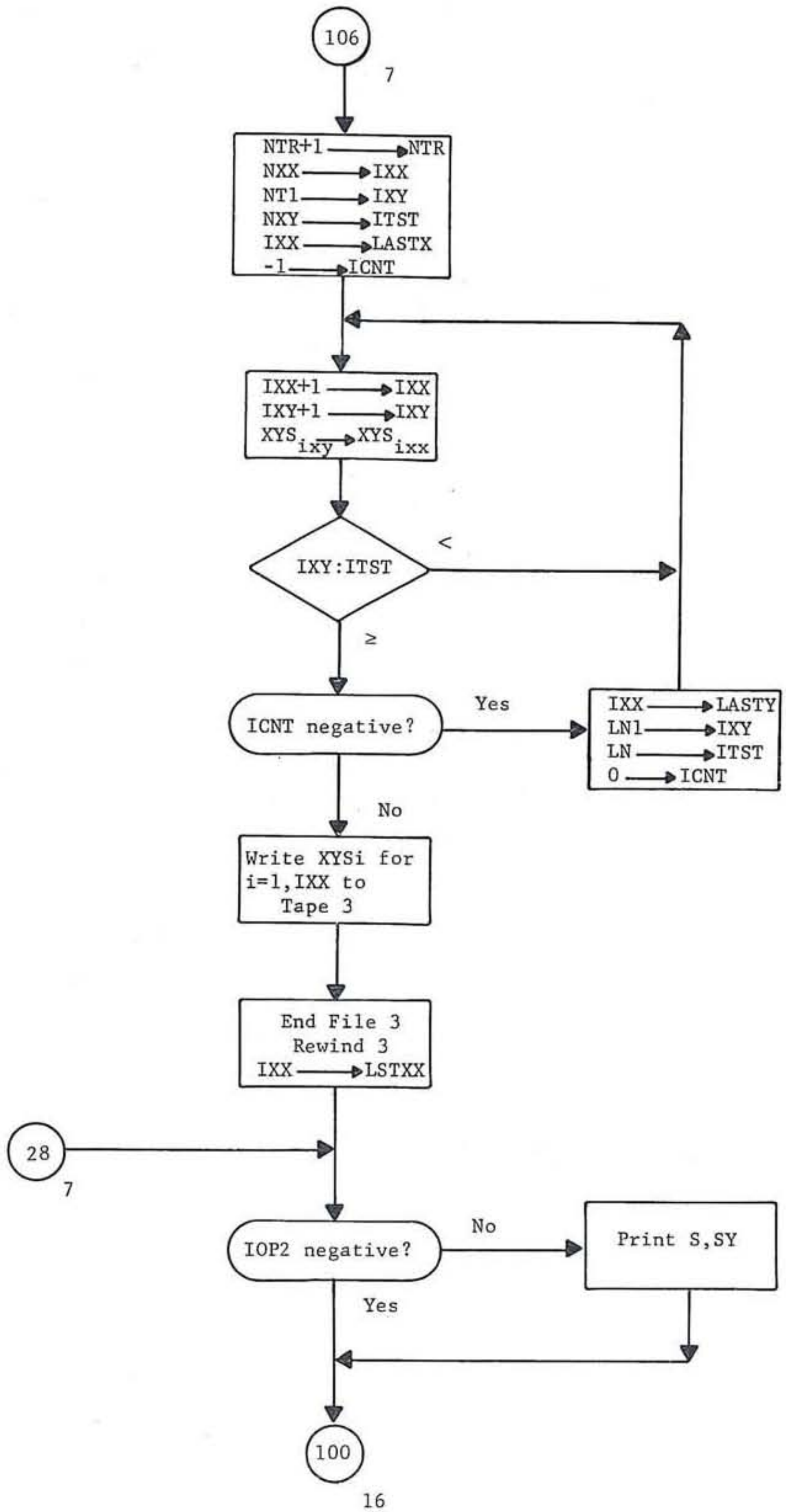
LSPF

(6)


10
6
SY_li+SXYLI → SY_li
LI+1 → LI
≤
A
6
LI:NT
>
Y^2 → WYTY
NW:0
≥
W*WYTY → WYTY
<
SYSQ+WYTY → SYSQ
ICASE negative?
No
Store XX(i,nt) in XYS (next nt locations)
Yes
<
99
3
N:NX
≥
≤
28
8
NTR:0
>
106
8

LSPF

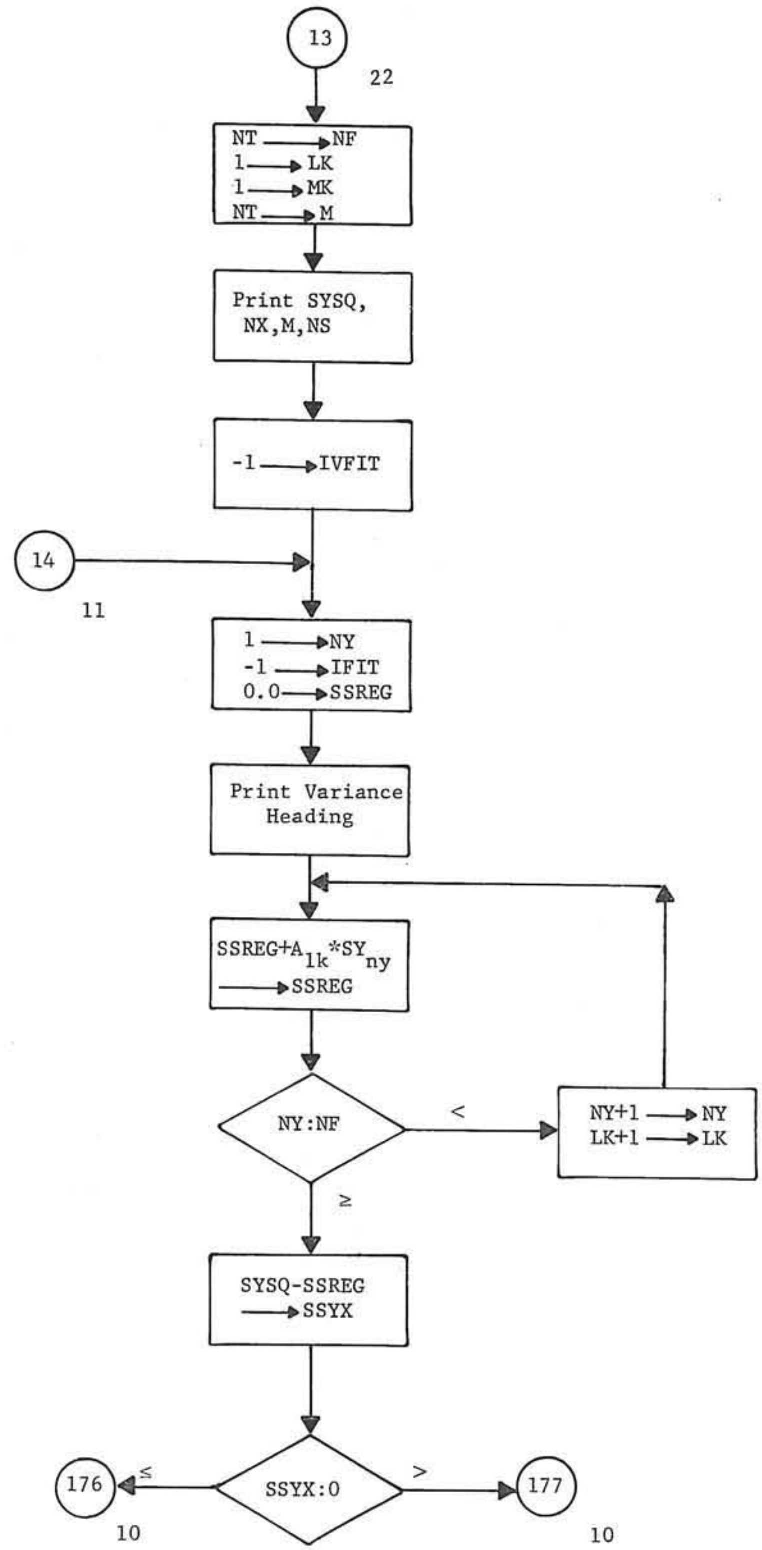
⑦



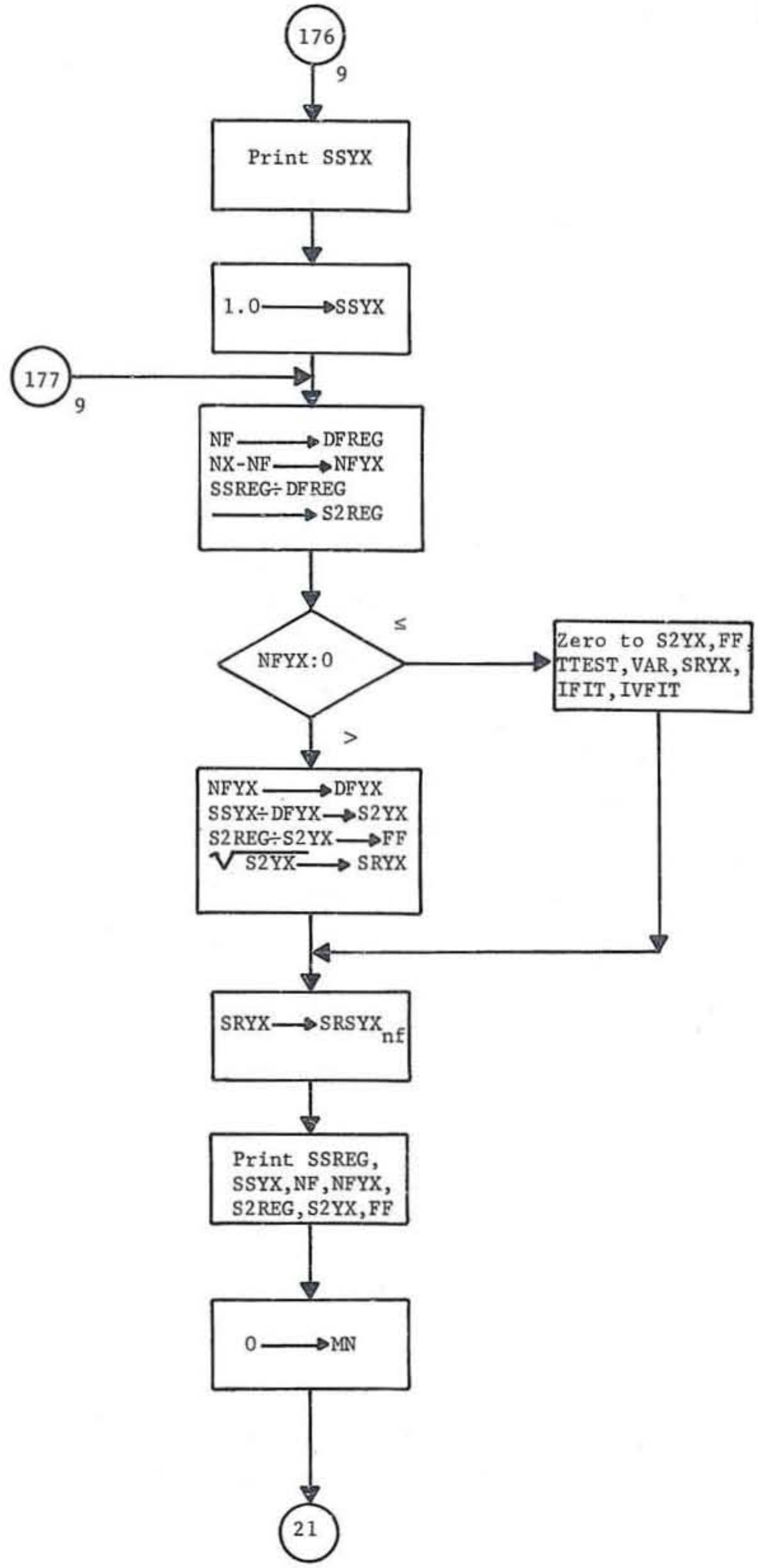
106
7
NTR+1 → NTR
NXX → IXX
NT1 → IXY
NXY → ITST
IXX → LASTX
-1 → ICNT
IXX+1 → IXX
IXY+1 → IXY
XYS ixy → XYS ixx
IXY:ITST
<
≥
ICNT negative?
Yes
IXX → LASTY
LN1 → IXY
LN → ITST
0 → ICNT
No
Write XYSi for i=1,IXX to Tape 3
End File 3
Rewind 3
IXX → LSTXX
28
7
IOP2 negative?
No
Print S,SY
Yes
100
16

LSPF

8


13
22
NT → NF
1 → LK
1 → MK
NT → M
Print SYSQ, NX,M,NS
-1 → IVFIT
14
11
1 → NY
-1 → IFIT
0.0 → SSREG
Print Variance Heading
SSREG+A1k*SYny → SSREG
NY:NF
<
NY+1 → NY
LK+1 → LK
≥
SYSQ-SSREG → SSYX
SSYX:0
≤
176
10
>
177
10

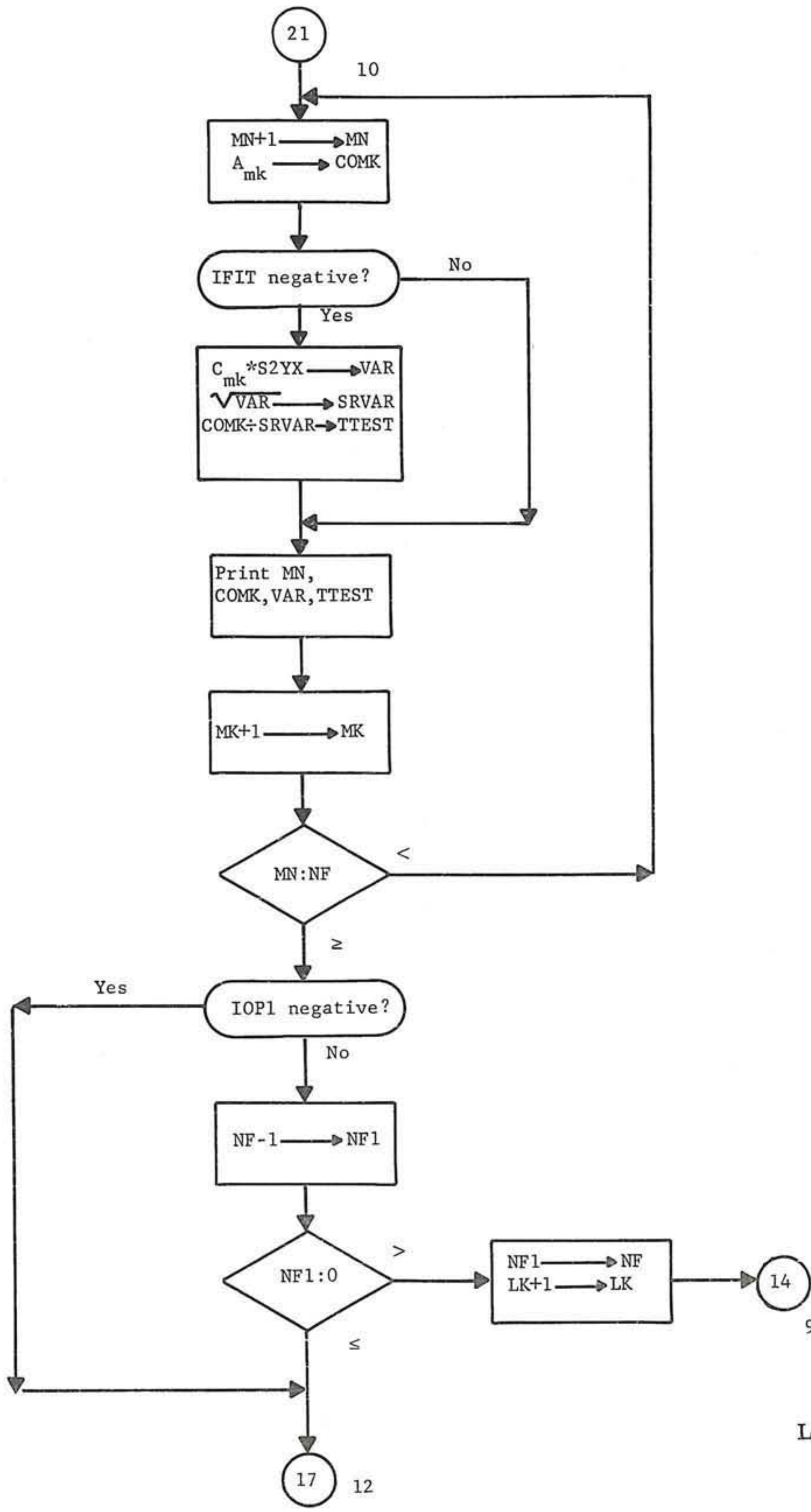
LSPF
(9)


176
9
Print SSYX
1.0 → SSYX
177
9
NF → DFREG
NX-NF → NFYX
SSREG÷DFREG → S2REG
NFYX:0
≤
Zero to S2YX,FF, TTEST,VAR,SRYX, IFIT,IVFIT
>
NFYX → DFYX
SSYX÷DFYX → S2YX
S2REG÷S2YX → FF
√S2YX → SRYX
SRYX → SRSYX nf
Print SSREG, SSYX,NF,NFYX, S2REG,S2YX,FF
0 → MN
21

11

LSPF

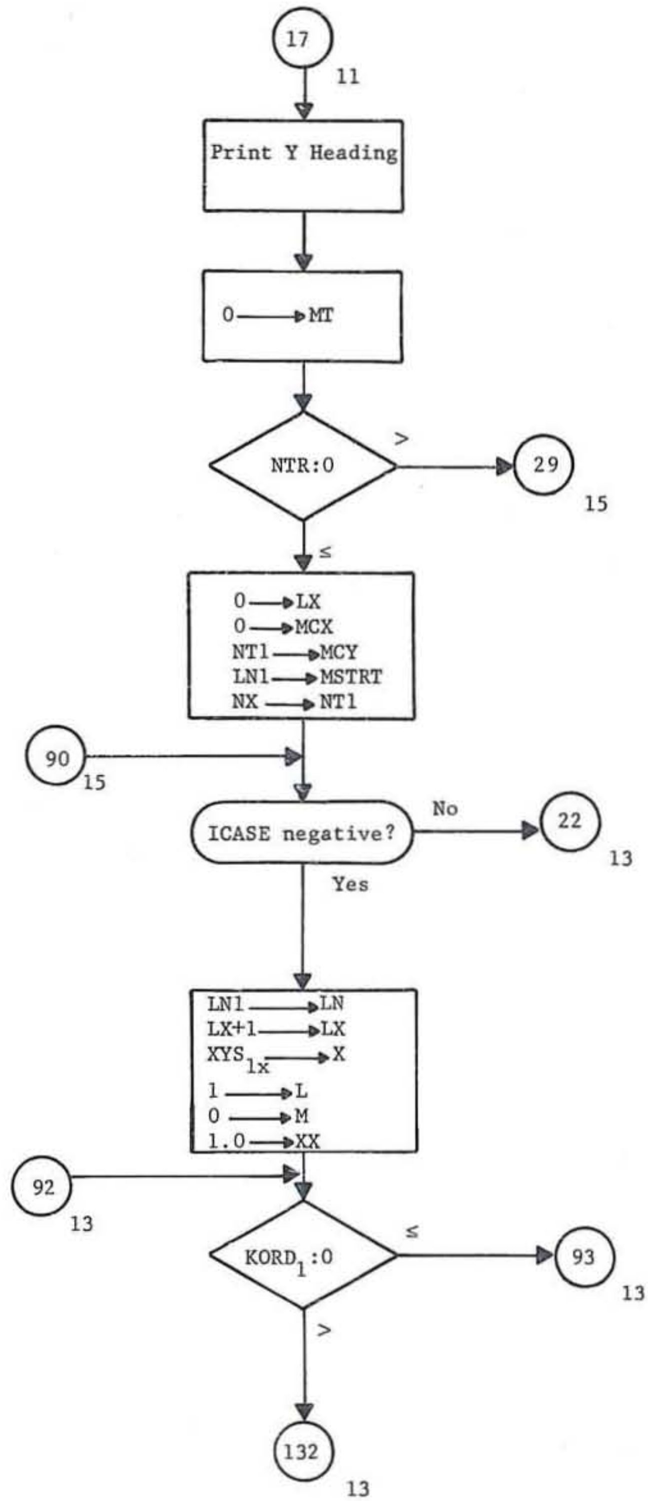
(10)



21
10
MN+1 → MN
A mk → COMK
IFIT negative?
No
Yes
C mk *S2YX → VAR
√VAR → SRVAR
COMK÷SRVAR → TTEST
Print MN,
COMK,VAR,TTEST
MK+1 → MK
MN:NF
<
≥
Yes
IOP1 negative?
No
NF-1 → NF1
NF1:0
>
NF1 → NF
LK+1 → LK
14
9
≤
17
12

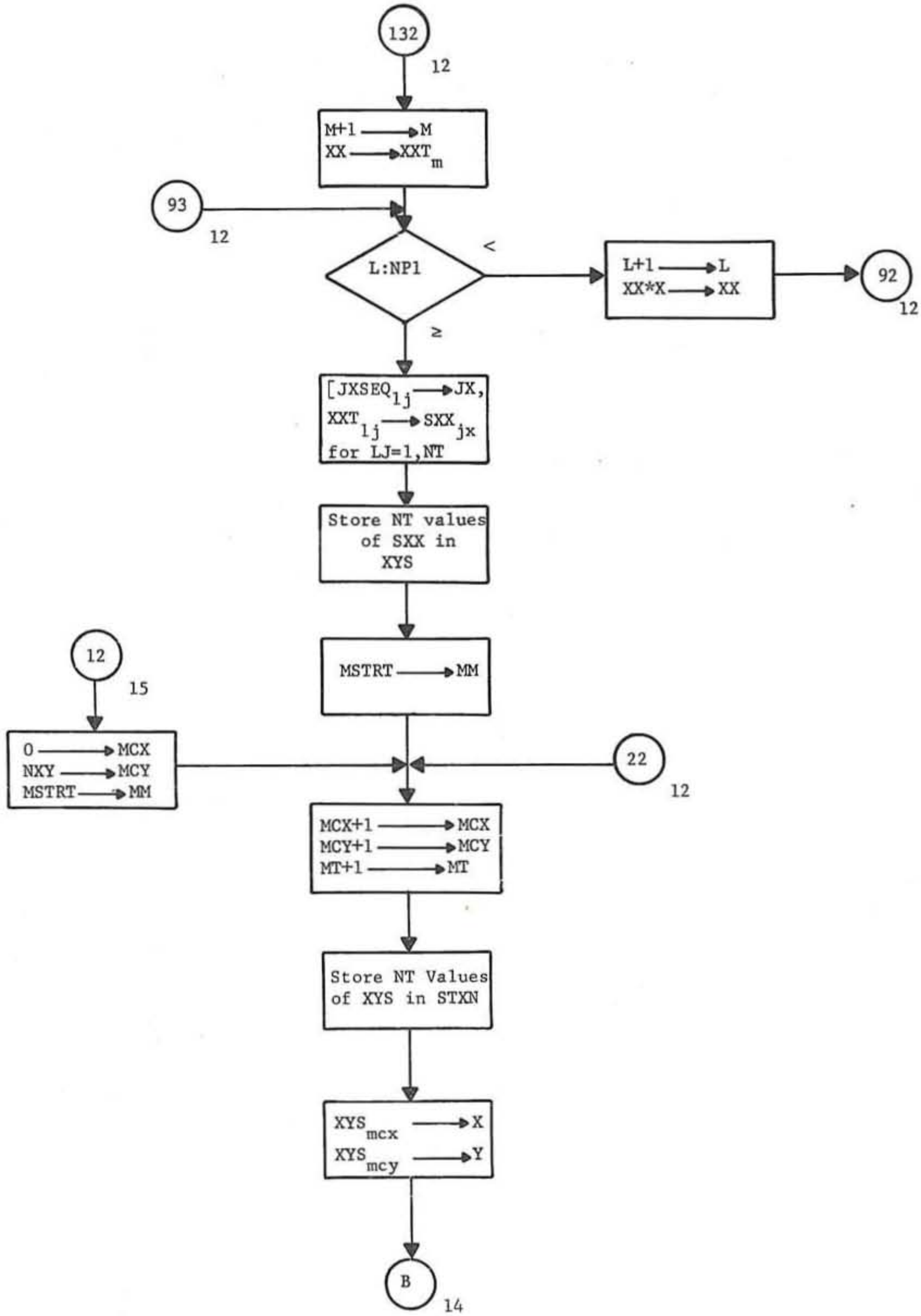
LSPF

(11)


17
11
Print Y Heading
0 → MT
NTR:0
>
29
15
≤
0 → LX
0 → MCX
NT1 → MCY
LN1 → MSTRT
NX → NT1
90
15
ICASE negative?
No
22
13
Yes
LN1 → LN
LX+1 → LX
XYS1x → X
1 → L
0 → M
1.0 → XX
92
13
KORD1:0
≤
93
13
>
132
13

LSPF

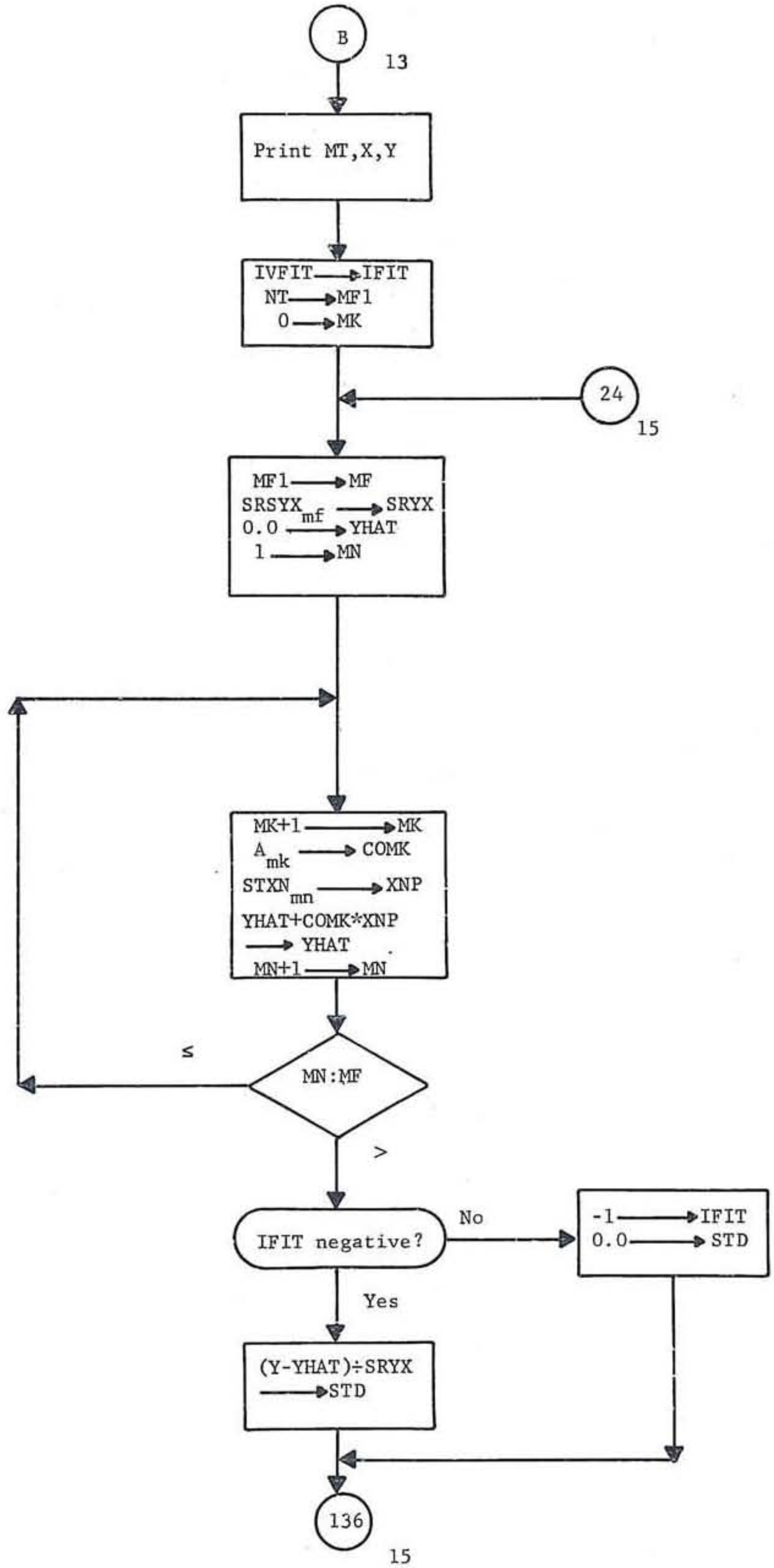
(12)



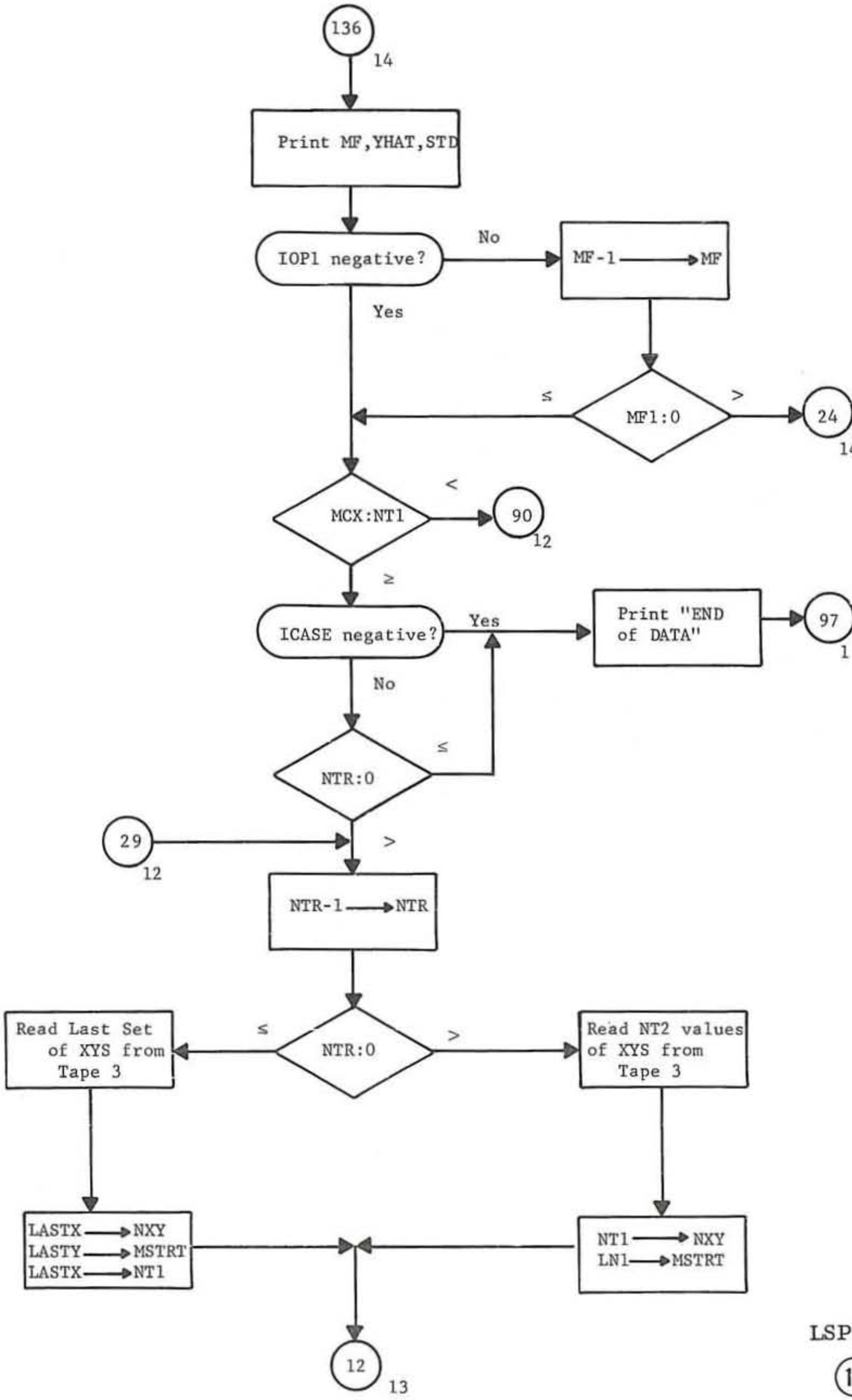
132
12
M+1 → M
XX → XXT m
93
12
L:NP1
<
≥
L+1 → L
XX*X → XX
92
12
[JXSEQ 1j → JX,
XXT 1j → SXX jx
for LJ=1,NT
Store NT values of SXX in XYS
MSTRT → MM
12
15
0 → MCX
NXY → MCY
MSTRT → MM
22
12
MCX+1 → MCX
MCY+1 → MCY
MT+1 → MT
Store NT Values of XYS in STXN
XYS mcx → X
XYS mcy → Y
B
14

LSPF

(13)


B
13
Print MT,X,Y
IVFIT → IFIT
NT → MF1
0 → MK
24
15
MF1 → MF
SRSYX mf → SRYX
0.0 → YHAT
1 → MN
MK+1 → MK
A mk → COMK
STXN mn → XNP
YHAT+COMK*XNP
→ YHAT
MN+1 → MN
≤
MN:MF
>
IFIT negative?
No
-1 → IFIT
0.0 → STD
Yes
(Y-YHAT)÷SRYX
→ STD
136
15

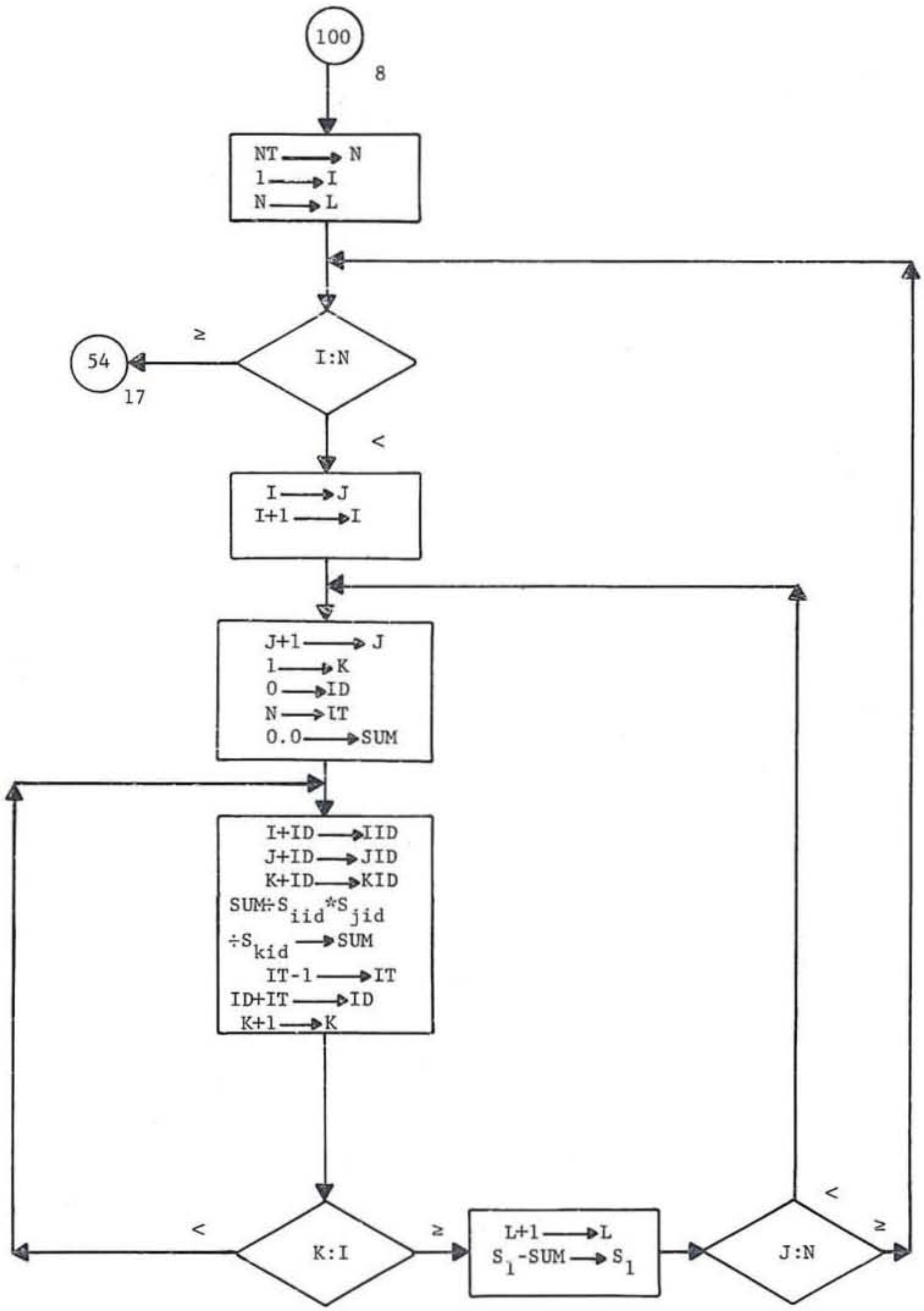
LSPF
(14)



136
14
Print MF,YHAT,STD
IOP1 negative?
No
Yes
MF-1 → MF
MF1:0
≤
>
24
14
MCX:NT1
<
90
12
≥
ICASE negative?
Yes
No
Print "END of DATA"
97
1
NTR:0
≤
29
12
>
NTR-1 → NTR
NTR:0
≤
>
Read Last Set of XYS from Tape 3
Read NT2 values of XYS from Tape 3
LASTX → NXY
LASTY → MSTRT
LASTX → NT1
NT1 → NXY
LN1 → MSTRT
12
13

LSPF

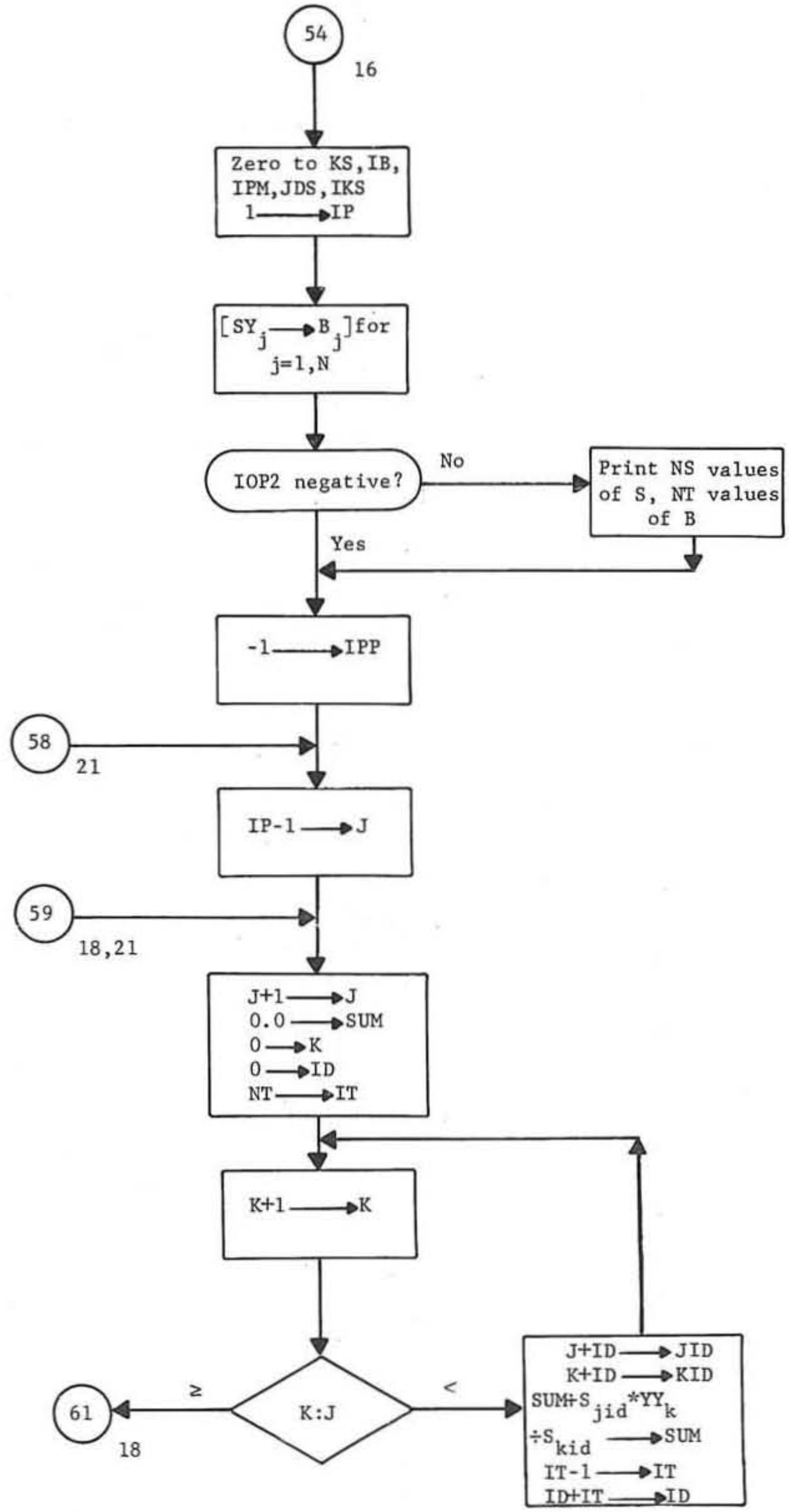
(15)



100
8
NT → N
1 → I
N → L
I:N
≥
54
17
<
I → J
I+1 → I
J+1 → J
1 → K
0 → ID
N → IT
0.0 → SUM
I+ID → IID
J+ID → JID
K+ID → KID
SUM+S iid *S jid
÷S kid → SUM
IT-1 → IT
ID+IT → ID
K+1 → K
<
K:I
≥
L+1 → L
S 1 -SUM → S 1
<
J:N
≥

LSPF

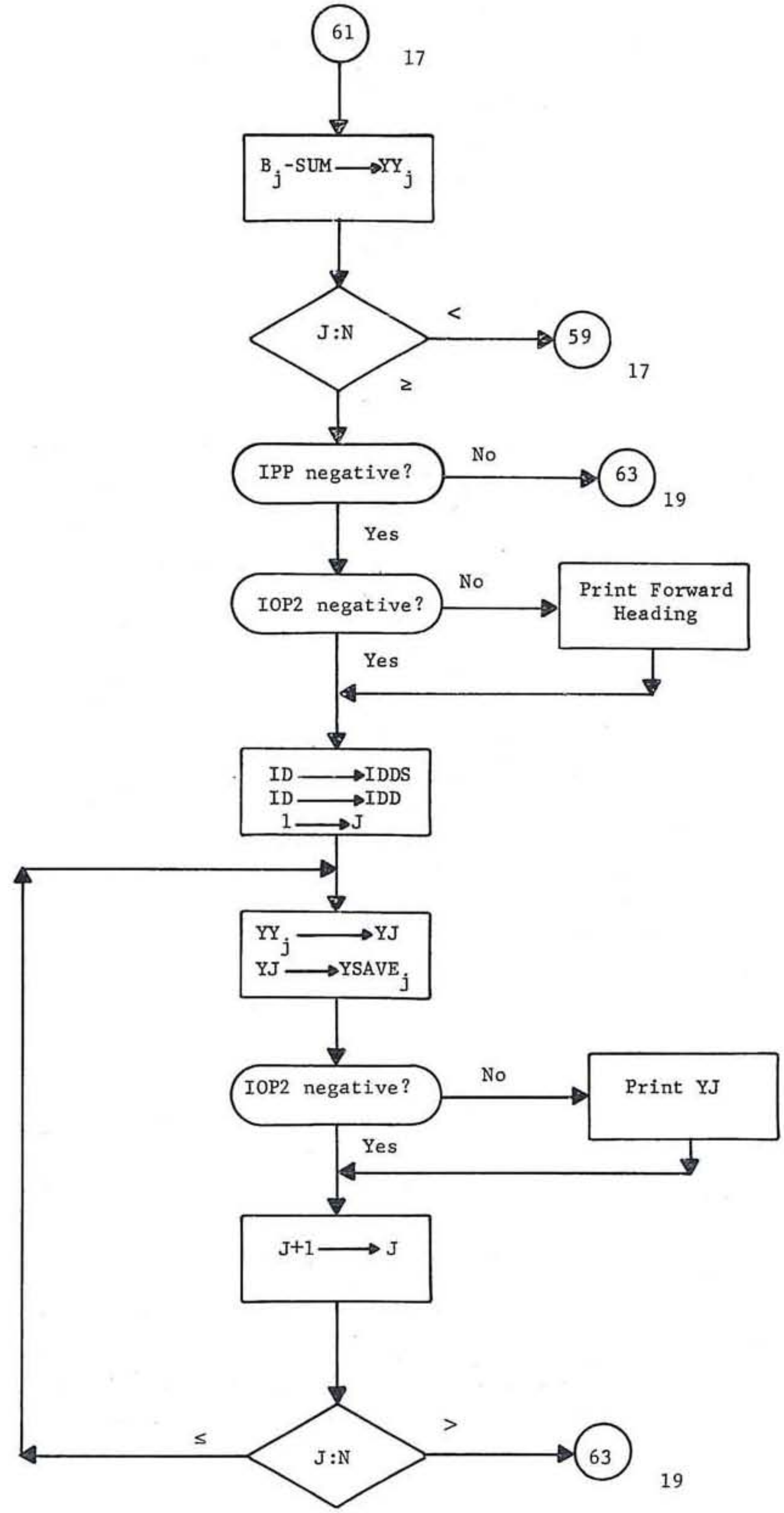
(16)



54
16
Zero to KS,IB, IPM,JDS,IKS
1 → IP
[SY_j → B_j] for j=1,N
IOP2 negative?
No
Print NS values of S, NT values of B
Yes
-1 → IPP
58
21
IP-1 → J
59
18,21
J+1 → J
0.0 → SUM
0 → K
0 → ID
NT → IT
K+1 → K
K:J
≥
<
61
18
J+ID → JID
K+ID → KID
SUM+S_jid*YY_k ÷S_kid → SUM
IT-1 → IT
ID+IT → ID

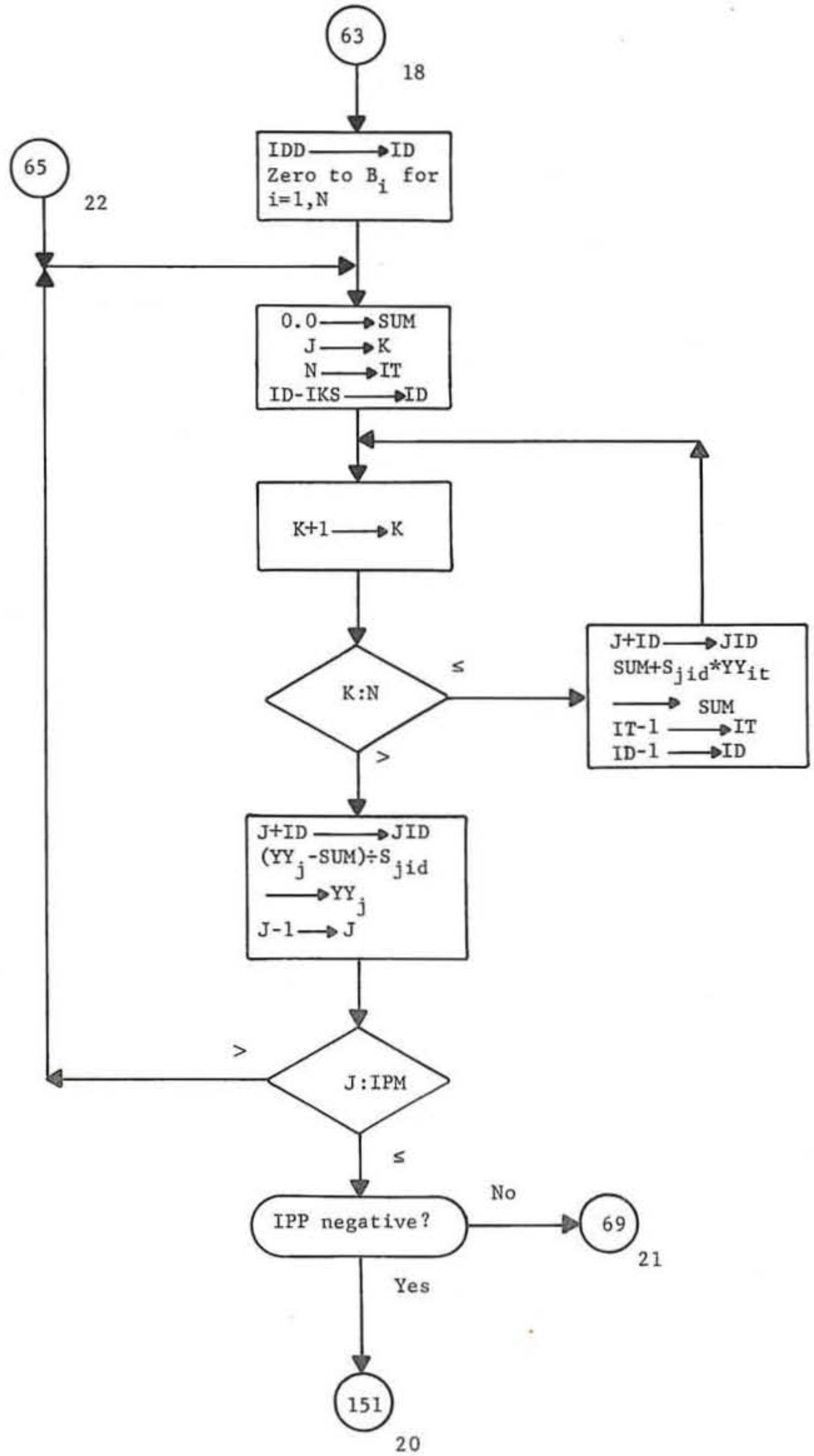
LSPF

(17)


61
17
B_j-SUM → YY_j
J:N
<
59
17
≥
IPP negative?
No
63
19
Yes
IOP2 negative?
No
Print Forward Heading
Yes
ID → IDDS
ID → IDD
1 → J
YY_j → YJ
YJ → YSAVE_j
IOP2 negative?
No
Print YJ
Yes
J+1 → J
≤
J:N
>
63
19

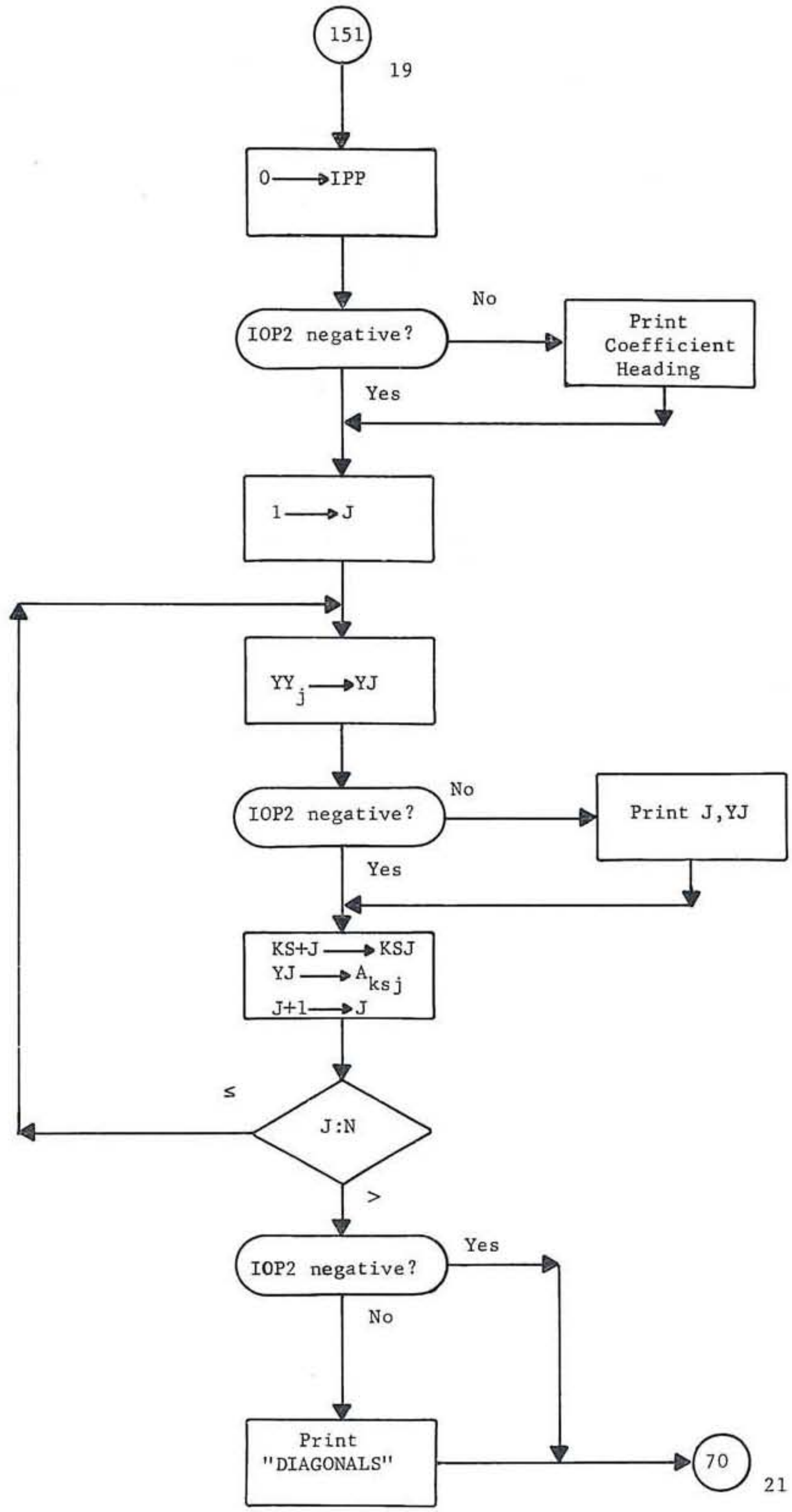
LSPF

(18)



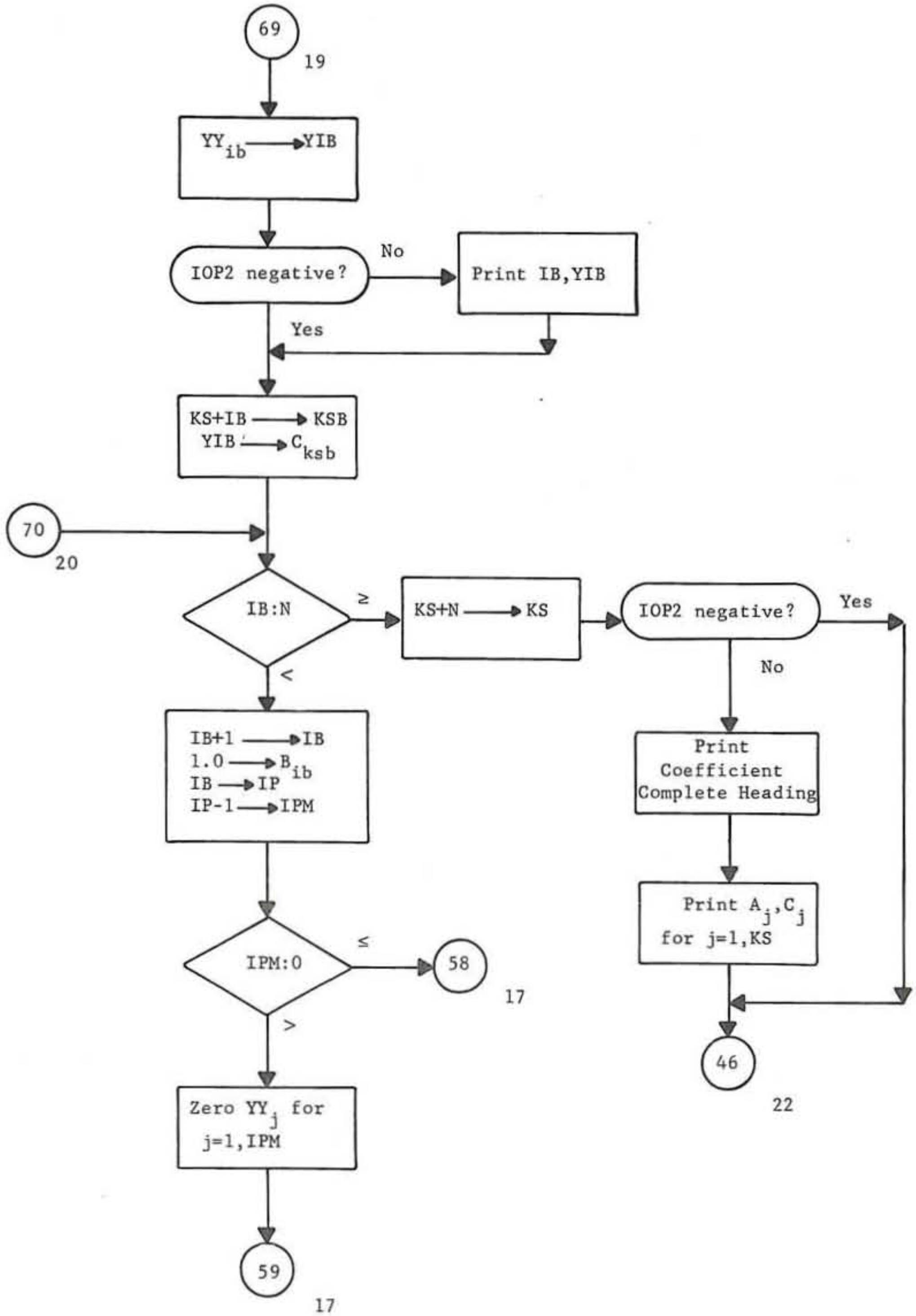
63
18
65
22
IDD → ID
Zero to B_i for
i=1,N
0.0 → SUM
J → K
N → IT
ID-IKS → ID
K+1 → K
K:N
≤
>
J+ID → JID
SUM+S_{jid}*YY_{it}
→ SUM
IT-1 → IT
ID-1 → ID
J+ID → JID
(YY_j-SUM)÷S_{jid}
→ YY_j
J-1 → J
J:IPM
>
≤
IPP negative?
No
69
21
Yes
151
20

LSPF
(19)


151
19
0 → IPP
IOP2 negative?
No
Print Coefficient Heading
Yes
1 → J
YYj → YJ
IOP2 negative?
No
Print J,YJ
Yes
KS+J → KSJ
YJ → Aksj
J+1 → J
≤
J:N
>
IOP2 negative?
Yes
No
Print "DIAGONALS"
70
21

LSPF

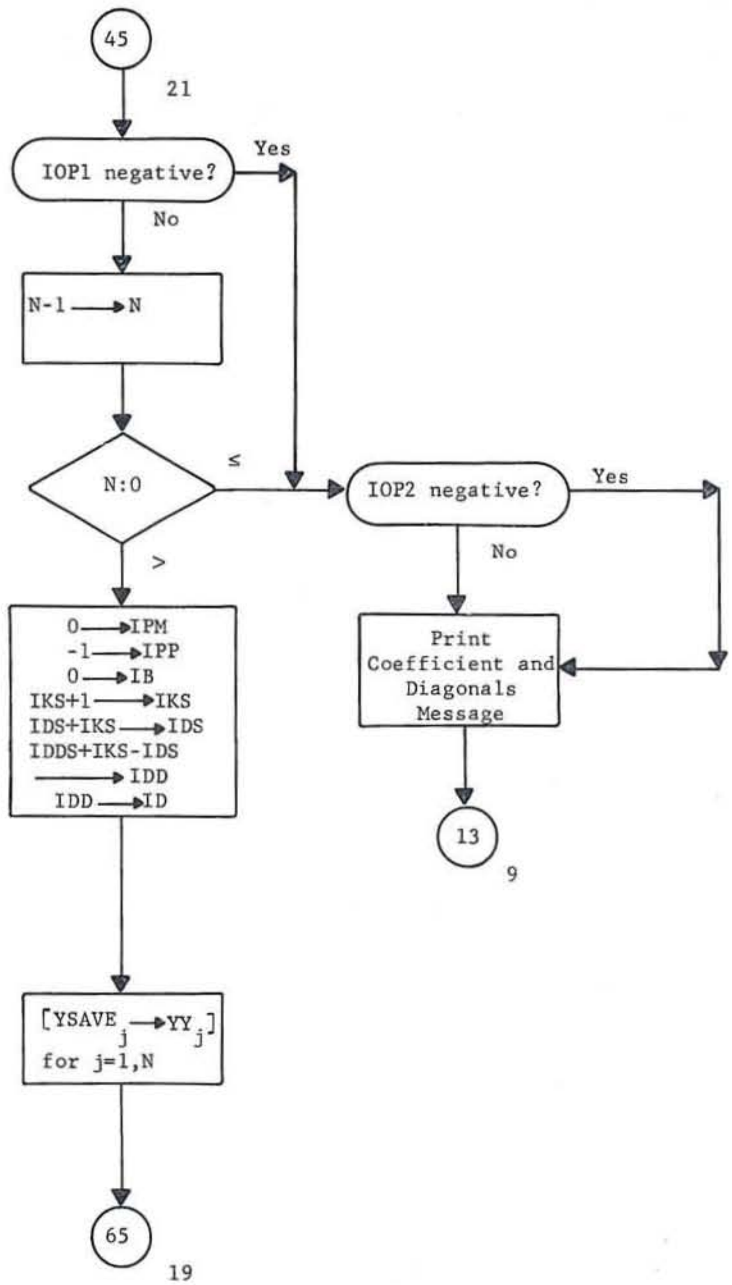
(20)



69
19
YY_ib → YIB
IOP2 negative?
No
Print IB,YIB
Yes
KS+IB → KSB
YIB → C_ksb
70
20
IB:N
≥
KS+N → KS
IOP2 negative?
Yes
No
<
IB+1 → IB
1.0 → B_ib
IB → IP
IP-1 → IPM
Print
Coefficient
Complete Heading
Print A_j,C_j
for j=1,KS
IPM:0
≤
58
17
>
46
22
Zero YY_j for
j=1,IPM
59
17

LSPF

(21)



45
21
IOP1 negative?
Yes
No
N-1 → N
N:0
≤
>
IOP2 negative?
Yes
No
0 → IPM
-1 → IPP
0 → IB
IKS+1 → IKS
IDS+IKS → IDS
IDDS+IKS-IDS
→ IDD
IDD → ID
Print
Coefficient and
Diagonals
Message
13
9
[YSAVE j → YY j]
for j=1,N
65
19

LSPF
(22)

LSPF INPUT

Col
2

$PARA/NX___, NT=___, NP=___, NW=___ , IØP1=___, IØP2=___, XZER=___$

$EXP/KSEQ=___, ___, ___, ___, ___, ___$

$DATA/X=__________, Y=__________, W=__________$

$DATA/X=__________, Y=__________, W=__________$

$DATA/X=__________, Y=__________, W=__________$

$DATA/X=__________, Y=__________, W=__________$

$DATA/X=__________, Y=__________, W=__________$

$DATA/X=__________, Y=__________, W=__________$

$DATA/X=__________, Y=__________, W=__________$

$DATA/X=__________, Y=__________, W=__________$

$DATA/X=__________, Y=__________, W=__________$

$DATA/X=__________, Y=__________, W=__________$

$DATA/X=__________, Y=__________, W=__________$

$DATA/X=__________, Y=__________, W=__________$

$DATA/X=__________, Y=__________, W=__________$

$DATA/X=__________, Y=__________, W=__________$

$DATA/X=__________, Y=__________, W=__________$

$DATA/X=__________, Y=__________, W=__________$

$DATA/X=__________, Y=__________, W=__________$

LSPF INPUT

Col
2

$PARA/NX___, NT=___, NP=___, NW=___ , IØP1=___, IØP2=___, XZER=___$

$EXP/KSEQ=___, ___, ___, ___, ___, ___$

$DATA/X=__________, Y= __________, W= __________$

$DATA/X=__________, Y= __________, W= __________$

$DATA/X=__________, Y= __________, W= __________$

$DATA/X=__________, Y= __________, W= __________$

$DATA/X=__________, Y= __________, W= __________$

$DATA/X=__________, Y= __________, W= __________$

$DATA/X=__________, Y= __________, W= __________$

$DATA/X=__________, Y= __________, W= __________$

$DATA/X=__________, Y= __________, W= __________$

$DATA/X=__________, Y= __________, W= __________$

$DATA/X=__________, Y= __________, W= __________$

$DATA/X=__________, Y= __________, W= __________$

$DATA/X=__________, Y= __________, W= __________$

$DATA/X=__________, Y= __________, W= __________$

$DATA/X=__________, Y= __________, W= __________$

$DATA/X=__________, Y= __________, W= __________$

$DATA/X=__________, Y= __________, W= __________$

LSPF INPUT

Col
2

$PARA/NX___, NT=___, NP=___, NW=___ , IØP1=___, IØP2=___, XZER=___$

$EXP/KSEQ=___, ___, ___, ___, ___, ___$

$DATA/X=__________, Y=__________, W=__________$

$DATA/X=__________, Y=__________, W=__________$

$DATA/X=__________, Y=__________, W=__________$

$DATA/X=__________, Y=__________, W=__________$

$DATA/X=__________, Y=__________, W=__________$

$DATA/X=__________, Y=__________, W=__________$

$DATA/X=__________, Y=__________, W=__________$

$DATA/X=__________, Y=__________, W=__________$

$DATA/X=__________, Y=__________, W=__________$

$DATA/X=__________, Y=__________, W=__________$

$DATA/X=__________, Y=__________, W=__________$

$DATA/X=__________, Y=__________, W=__________$

$DATA/X=__________, Y=__________, W=__________$

$DATA/X=__________, Y=__________, W=__________$

$DATA/X=__________, Y=__________, W=__________$

$DATA/X=__________, Y=__________, W=__________$

$DATA/X=__________, Y=__________, W=__________$

LSPF INPUT

Col
2

$PARA/NX___, NT=___, NP=___, NW=___ , IØP1=___, IØP2=___, XZER=___$

$EXP/KSEQ=___, ___, ___, ___, ___, ___$

$DATA/X=__________, Y= __________, W= __________$

$DATA/X=__________, Y= __________, W= __________$

$DATA/X=__________, Y= __________, W= __________$

$DATA/X=__________, Y= __________, W= __________$

$DATA/X=__________, Y= __________, W= __________$

$DATA/X=__________, Y= __________, W= __________$

$DATA/X=__________, Y= __________, W= __________$

$DATA/X=__________, Y= __________, W= __________$

$DATA/X=__________, Y= __________, W= __________$

$DATA/X=__________, Y= __________, W= __________$

$DATA/X=__________, Y= __________, W= __________$

$DATA/X=__________, Y= __________, W= __________$

$DATA/X=__________, Y= __________, W= __________$

$DATA/X=__________, Y= __________, W= __________$

$DATA/X=__________, Y= __________, W= __________$

$DATA/X=__________, Y= __________, W= __________$

$DATA/X=__________, Y= __________, W= __________$

LSPF INPUT

Col
2

$PARA/NX___, NT=___, NP=___, NW=___ , IØP1=___, IØP2=___, XZER=___$

$EXP/KSEQ=___, ___, ___, ___, ___, ___$

$DATA/X=__________, Y=__________, W=__________$

$DATA/X=__________, Y=__________, W=__________$

$DATA/X=__________, Y=__________, W=__________$

$DATA/X=__________, Y=__________, W=__________$

$DATA/X=__________, Y=__________, W=__________$

$DATA/X=__________, Y=__________, W=__________$

$DATA/X=__________, Y=__________, W=__________$

$DATA/X=__________, Y=__________, W=__________$

$DATA/X=__________, Y=__________, W=__________$

$DATA/X=__________, Y=__________, W=__________$

$DATA/X=__________, Y=__________, W=__________$

$DATA/X=__________, Y=__________, W=__________$

$DATA/X=__________, Y=__________, W=__________$

$DATA/X=__________, Y=__________, W=__________$

$DATA/X=__________, Y=__________, W=__________$

$DATA/X=__________, Y=__________, W=__________$

$DATA/X=__________, Y=__________, W=__________$

Progress Is Our Most Important Product

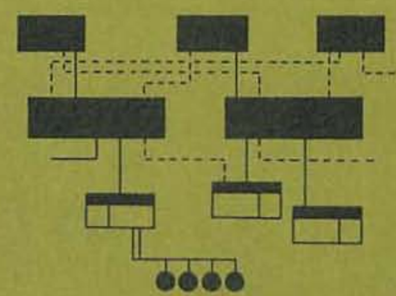
GENERAL ELECTRIC

COMPUTER DEPARTMENT • PHOENIX, ARIZONA

LITHO U.S.A.

GENERAL ELECTRIC
COMPUTERS

GE-625/635
Math Routines
POLRTS



GENERAL ELECTRIC

CPB-1152A

GE-625/635
MATH ROUTINES
POLRTS

ROOTS OF A POLYNOMIAL

Program Number
CD600D5.001

September 1965

Rev. June 1967

GENERAL ELECTRIC

INFORMATION SYSTEMS DIVISION

CPB-1152A

ACKNOWLEDGMENT

The source material used in this manual is taken from a document, published by General Electric, titled POLRTS--Bairstow's Method For Finding Polynomial Roots in FORTRAN, by R. F. Jordan and H. W. Moore. Permission to use the original document was given by D. L. Shell, Manager, Computer Applications and Processing, General Electric's Telecommunications and Information Processing Department.

PREFACE

This revised manual includes information previously published in CPB-1152, and supplemented with information previously published in Technical Information Bulletin 600-98. In this revised edition, changes in technical content from the previous edition are identified with a bar in the margin opposite the change.

Suggestions and criticisms relative to form, content, purpose, or use of this manual are invited. Comments may be sent on the Document Review Sheet in the back of this manual or may be addressed directly to Documentation Standards and Publications, B-90, Computer Equipment Department, General Electric Company, 13430 North Black Canyon Highway, Phoenix, Arizona 85029.

CONTENTS

APPENDIXES

ILLUSTRATIONS

1. GENERAL DESCRIPTION

The POLRTS subroutine is a FORTRAN subprogram for finding all the roots (both real and complex) of polynomials with real coefficients. The method used is a form of the Bairstow iteration algorithm for reducing the polynomial to quadratic factors which are readily solved by the quadratic formula. The subroutine accepts polynomials of any positive degree, subject only to the trivial restriction that the polynomial must have no zero roots. The initial guesses required to start the iteration are provided by the subroutine.

The POLRTS subroutine uses the two library routines, ABS and SQRT. In addition, the user must provide storage for three arrays as part of his main program. Each array must be of dimension at least n + 1, if n is the highest degree polynomial to be solved.

These arrays are used to store the coefficients of the polynomial and the real and imaginary parts of each root, as well as for intermediate working storage. The subroutine destroys the coefficients of the original polynomial in the course of the root-finding process.

The roots are usually found to at least six significant figures and the iteration normally converges at every stage. However certain ill-conditioned polynomials and polynomials having roots of high multiplicity require more elaborate routines employing multiple-precision arithmetic. In practice, the POLRTS subroutine has given very satisfactory results for polynomials of degree as high as 20 and, no doubt, will succeed at even higher degrees in favorable cases.

The subroutine performs approximately 4n + 10 multiplications and 4n + 10 additions per iteration in removing a quadratic factor from a polynomial of degree n. Once a factor has been removed, the following set of iterations will be performed on the reduced polynomial of degree n-2, and so forth, so that the number of operations will decrease rapidly as the factors are successively removed. The number of iterations required to find one quadratic factor cannot be predicted in advance. However, in practice a figure of 5 to 20 iterations is typical for a factor of a polynomial of moderate degree. If a factor is not found within 100 iterations, the subroutine executes a return to the main program with any roots previously found.

The Bairstow method appears to be the most satisfactory technique now known for finding roots of arbitrary polynomials. In practice, it has been found to be several times as fast as the more familiar Newton Raphson method, succeeding at least as often and giving results of equal or greater reliability. The program POLY is written to perform input/output for POLRTS, creating a free-standing package.

2. MATHEMATICAL METHOD

PROBLEM ANALYSIS

The discussion below describes Bairstow's method for finding the roots of the following polynomial:

$$P(z) = c_1 z^n + c_2 z^{n-1} + \ldots + c_n z + c_{n+1}.$$

In Bairstow's approach, the roots are not looked for directly. Instead, an attempt is made to find an exact quadratic divisor of P (z), say Z (z) = $z^2 + p^*z + q^*$ so that P(z) = Q(z) · R(z) where R(z) is of degree n - 2. The roots of Q(z) are readily found from the familiar quadratic formula. Any such root is clearly also a root of P(z), for if Q(r) = 0, then P(r) = Q(r) · R(r) = 0.

The problem may be solved by repeating this process (that is, finding an exact quadratic divisor of R(z) whose roots will similarly be roots of P(z) and continuing in this fashion until all the roots of P(z) are found.

The first step in finding the exact quadratic divisor of P(z) is to make an initial guess (p, q) and divide P(z) by the factor $z^2 + pz + q$):

$$P(z) = (z^2 + pz + q) \cdot R(z) + Az + B.$$

If A = B =0, then $z^2 + pz + q$ is the desired factor. In general, however, this will not be the case, and, therefore, the initial guess of (p, q) must be improved to make A and B as close to 0 as possible.

Since A and B are functions of p and q possessing continuous partial derivatives of all orders (as will be seen below), A (p*, q*) and B (p*, q*) may be represented by Taylor's series about the point (p, q):

$$A(p^*, q^*) = A(p,q) + \left.\frac{\partial A}{\partial p}\right|_{p,q} \cdot \Delta p + \left.\frac{\partial A}{\partial q}\right|_{p,q} \cdot \Delta q + \ldots$$

$$B(p^*, q^*) = B(p,q) + \left.\frac{\partial B}{\partial p}\right|_{p,q} \cdot \Delta p + \left.\frac{\partial B}{\partial q}\right|_{p,q} \cdot \Delta q + \ldots$$

Because the expression $B(p^*, q^*) = A(p^*, q^*) = 0$ must be true, if the higher order terms in the Taylor's series are ignored, a first approximation shows:

$$\left.\frac{\partial A}{\partial P}\right|_{p,\,\dot{q}} \Delta p + \left.\frac{\partial A}{\partial q}\right|_{p,\,\dot{q}} \Delta q = -A(p, q)$$

$$\left.\frac{\partial B}{\partial P}\right|_{p,\,\dot{q}} \Delta p + \left.\frac{\partial B}{\partial q}\right|_{p,\,\dot{q}} \Delta q = -B(p, q).$$

Since A (p, q) and B (p, q) are simply the coefficients of the remainder when P (z) is divided by $z^2 + pz + q$, these two linear equations may easily be solved by Cramer's rule for Δp and Δq, after the partial derivatives are evaluated. This gives the corrections to be made to p and q to cause $A(p + \Delta p, q + \Delta q)$ and $B(p + \Delta p, q + \Delta q)$ to be 0. Naturally, since only the first order terms of the Taylor's series have been retained, A and B are not expected to be exactly 0. However, $p + \Delta p$ and $q + \Delta q$ should be closer to p^* and q^* than p and q are, and may, therefore, be used as a basis for a further prediction. Convergence to p^* and q^* may be considered to have taken place when the magnitude of the change in Δp and Δq has ceased to be significant. The values of p and q at that point make $z^2 + pz + q$ an almost exact divisor of P(z), and, thus, two of the roots of P(z) may be found by solving the quadratic.

The following discussion describes the evaluation of the partial derivatives:

$$\frac{\partial A}{\partial p},\ \frac{\partial A}{\partial q},\ \frac{\partial B}{\partial p},\ \frac{\partial B}{\partial q}.$$

Suppose that the division of P(z) by $z^2 + pz + q$ gives the quotient:

$$b_1 z^{n-2} + b_2 z^{n-3} + \ldots + b_{n-2} z + b_{n-1}$$

and the remainder:

$$b_n z + b_{n+1} = Az + B.$$

It can be easily verified that the coefficients b_i are given by the relations:

$$b_1 = c_1$$

$$b_2 = c_2 - pb_1$$

$$b_i = c_i - pb_{i-1} - qb_{i-2} \qquad i=3, n$$

$$b_{n+1} = c_{n+1} - qb_{n-1}$$

where the c_i are the coefficients of P (z). Differentiating these relations with respect to p and q gives the partial derivatives:

$$\frac{\partial b_1}{\partial p} = \frac{\partial c_1}{\partial p} = 0$$

$$\frac{\partial b_2}{\partial p} = -b_1 - p\frac{\partial b_1}{\partial p} = -b_1$$

$$\frac{\partial b_i}{\partial p} = -b_{i-1} - p\frac{\partial b_{i-1}}{\partial p} - q\frac{\partial b_{i-2}}{\partial p} \qquad i = 3, n$$

$$\frac{\partial b_{n+1}}{\partial p} = -q\frac{\partial b_{n-1}}{\partial p}$$

and:

$$\frac{\partial b_1}{\partial q} = \frac{\partial c_1}{\partial q} = 0$$

$$\frac{\partial b_2}{\partial q} = \frac{\partial b_1}{\partial q} = 0$$

$$\frac{\partial b_i}{\partial q} = -b_{i-2} - p\frac{\partial b_{i-1}}{\partial q} - q\frac{\partial b_{i-2}}{\partial q}, \qquad i = 3, n$$

$$\frac{\partial b_{n+1}}{\partial q} = -b_{n-1} - q\frac{\partial b_{n-1}}{\partial q}$$

where:

$$\frac{\partial A}{\partial p} = \frac{\partial b_n}{\partial p}$$

$$\frac{\partial A}{\partial q} = \frac{\partial b_n}{\partial q}$$

$$\frac{\partial B}{\partial p} = \frac{\partial b_{n+1}}{\partial p}$$

$$\frac{\partial B}{\partial q} = \frac{\partial b_{n+1}}{\partial q}$$

Hence, the partial derivatives of A and B may be evaluated recursively by means of the above formulas. However, an inductive argument shows that:

$$\frac{\partial b_{i+1}}{\partial q} = \frac{\partial b_i}{\partial p} \qquad i = 1, \ldots, n$$

Thus, the partial derivatives must be calculated only with respect to p. This may be done by an algorithm whose form is similar to that of the calculation of the coefficients b_i. In fact, the complete calculation of the remainder Az + B and the partial derivatives of A and B is essentially accomplished by two successive divisions of P(z) by the factor $a^2 + pz + q$. If the symbol D_i is used to represent the partial derivative of b_i, with respect to p, the calculation may be arranged as shown on the following page.

i	b_i	D_i
1	c_1	0
2	$c_2 - pb_1$	$-b_1$
3	$c_3 - pb_2 - qb_1$	$-b_2 - pD_2 - qD_1$
4	$c_4 - pb_3 - qb_2$	$-b_3 - pD_3 - qD_2$
.	.	.
.	.	.
.	.	.
.	.	.
.	.	.
n	$c_n - pb_{n-1} - qb_{n-1}$	$-b_{n-1} - pD_{n-1} - qD_{n-2}$
n+1	$c_{n+1} - qb_{n-2}$	$-q\, D_{n-1}$

Then, use of the relation

$$\frac{\partial b_{i+1}}{\partial q} = \frac{\partial b_i}{\partial p}$$

gives

$$\frac{\partial A}{\partial p} = D_n$$

$$\frac{\partial A}{\partial q} = D_{n-1}$$

$$\frac{\partial B}{\partial p} = D_{n+1}$$

$$\frac{\partial B}{\partial q} = D_n$$

and the equations for Δp and Δq become

$$D_n \cdot \Delta p + D_{n-1} \cdot \Delta q = -b_n$$

$$D_{n+1} \cdot \Delta p + D_n \cdot \Delta q = -b_{n+1}.$$

Δp and Δq may readily be found from this equation.

Bairstow's method is discussed further in Numerical Methods for Scientists and Engineers by R.W. Hamming, McGraw Hill 1962, pp. 356-359, and in most standard texts on numerical analysis.

SUBROUTINE DESCRIPTION

The POLRTS subroutine begins by testing the degree of the polynomial to insure that it is positive. If it is 0 or negative, a return to the main program is executed with the IND indicator set to 0 (see page 9). Otherwise, the polynomial is converted to a monic polynomial by dividing every coefficient by the leading coefficient. The leading coefficient is thereafter assumed to be equal to 1 and is, consequently, not represented in the array of coefficients. Every coefficient is moved down into the preceding coefficient location to fill the vacancy thus created. The indicator ISW is calculated; it is -1, if the degree of the polynomial is odd, and 0, otherwise. The starting guess for p and q is formed by setting $p = c_{n-1}/c_{n-2}$ and $q = c_n/c_{n-2}$, unless c_{n-2} is 0: in this case, $p = q = c_n$. In order to prevent the initial guess for p from being 0, a small quantity is added to it.

If the polynomial at this point is of first degree, its real root, found without calculation, is placed in the appropriate position in the array of real roots and 0 is placed in the corresponding position of the array of imaginary roots (see Chapter 3, POLRTS Subroutine). A return to the main program is executed with the IND indicator set equal to N. If the degree is two, a branch is made to the section of the subroutine which solves for the roots of the quadratic factors. (This section is described below.)

Otherwise, the iteration counter IC is set equal to 1 and the iteration loop is entered. Using the algorithm explained in the discussion of the Sample Problem and the notation used in that discussion, the iteration loop calculates the values of A, B, D_{n-1}, D_n, D_{n+1}. The loop uses the following as temporary storage locations: the variable B1, B2, B3, B4, and the portions of the real and imaginary arrays which have not yet been occupied by roots. The b_i coefficients of the first quotient are stored in the RR array after each division and become the coefficients of the reduced polynomial when the iteration converges. Using Cramer's rule, the two simultaneous linear equations defining Δp and Δq are solved and the results are stored in B2 and B3, respectively.

The magnitudes of B2 and B3 are tested relative to P and Q, respectively, when the magnitudes of P and Q are greater than 1, and are tested absolutely, otherwise. When the tested magnitudes of B2 and B3 are both less than 5×10^{-6} convergence is considered to have taken place. Otherwise, B2 is added to P; B3 is added to Q; and IC, the iteration counter, is increased by 1. Assuming that IC is not now greater than 100, the iteration is repeated, using the new P and Q as a starting guess.

When IC becomes greater than 100, a convergence failure has occurred. Convergence failures can sometimes be caused by the presence of a single real root among complex roots; therefore, if ISW is equal to -1, indicating that the polynomial is of odd degree and, hence, has a real root, the polynomial is multiplied by the factor $(z + 1)$. This introduces the extra real root -1. ISW is then set equal to +1 as an indicator for future use, the IND indicator is set equal to the number of roots found prior to the convergence failure, and the present degree of the polynomial is increased by 1. The iteration procedure is then restarted with the new polynomial.

However, if ISW is equal to 0 or +1 after a convergence failure, indicating that the polynomial is of even degree, a return is executed to the main program; the IND indicator is set equal to the number of roots found before the convergence failure occurred, if ISW is 0, or to the number of roots found before the extra root was introduced, if ISW is +1.

When the test of B2 and B3 indicates that convergence has taken place, the polynomial could be reduced by dividing it by the factor $z^2 + Pz + Q$, using the last computed values of P and Q.

However, this division was already performed as the first step in the iteration loop, and the results were stored in the unused portion of the array of real roots. They may now be transferred from the array of real roots to the coefficient array (see Chapter 3, POLRTS Subroutine), thus destroying the coefficients of the previous polynomial. The coefficients are always stored up to location N of the coefficient array, and N remains constant. The counter I, which will be incremented by 2 before the next set of iterations, is used to mark the location of the first coefficient of the new polynomial in the coefficient array. Hence, I may also be used to mark the location of the last root found in the real and imaginary arrays.

Once the polynomial has been reduced in this fashion, the factor $z^2 + Pz + Q$ is solved by the quadratic formula. The two roots are stored in locations I and I + 1 of the real and imaginary arrays and I is incremented by 2. If I is not yet greater than N, a return is made to the iteration section with the reduced polynomial, in order to remove another quadratic factor. If I is greater than N, the polynomial has been completely reduced and all roots have been found.

Therefore, a return is made to the main program unless ISW is equal to +1. In this case, the extra root -1, which was previously introduced by the subroutine, must be removed from the array of real roots and the array of imaginary roots before a return to the main program can be made. If for any reason this root cannot be found, the return is made with the IND indicator set equal to the number of roots found before the extra root was introduced.

3. PROGRAM USAGE

POLRTS SUBROUTINE

Before calling the POLRTS subroutine, the user must dimension three arrays in his main program to contain the coefficients and the roots of the polynomials. These arrays are called the C (coefficient), RR (real-root), and RI (imaginary-root) arrays, and must be dimensioned at least n + 1, if n is the degree of the polynomial to be solved. Larger dimensions than necessary are acceptable. The POLY subroutine, which performs the input/output for POLRTS, is written to allow a 30th degree polynomial.

The polynomial itself must be written in descending powers of the independent variable:

$$c_1 z^n + c_2 z^{n-1} + \ldots + c_n z + c_{n+1}.$$

In addition, the polynomial must have no 0 roots; that is, the coefficient c_{n+1} must not be 0. If c_{n+1} is 0, the polynomial must be rewritten as a polynomial of degree m:

$$c_1 z^m + c_2 z^{m-1} + \ldots + c_m z + c_{m+1}$$

where m = n - 1. If c_{m+1} is 0, this must be repeated.

The coefficients are placed in the C array in order with c_i = C(I) for i = 1, 2, . . . n + 1. Since the subroutine will destroy the orginal coefficients in the course of solving the polynomial, they should also be stored in some other array if they are to be used again by the main program. The user does not have to specify the elements of the RR and RI arrays before calling the subroutine.

The subroutine is called by the statement:

```
CALL POLRTS (C, RR, RI, N, IND)
```

where N is the degree of the polynomial and IND is a dummy fixed point variable. Before returning to the main program, the subroutine sets the variable IND equal to the number of roots which have been found, and stores the real and imaginary parts of the roots in the first IND locations of the RR and RI arrays, respectively.

SAMPLE PROBLEMS

As many cases as desired may be stacked in one run.

The following six sample cases were used to test POLRTS. The first five sample cases have roots of 1, 2, 3, 4, and 5. The sixth case has both real and imaginary roots.

Case 1: $x - 1 = 0$

Case 2: $x^2 - 3x + 2 = 0$

Case 3: $x^3 - 6x^2 + 11x - 6 = 0$

Case 4: $x^4 - 10x^3 + 35x^2 - 50x + 24 = 0$

Case 5: $x^5 - 15x^4 + 85x^3 - 225x^2 + 274x - 120 = 0.$

Case 6: $1.5x^7 + 2.906x^6 + 10.6x^5 + 25.877x^4 + 2.3x^3 + 33x^2 + 1.234x + 543.2 = 0$

Input Preparation

Figure 1 shows how the sample cases are coded. Coding sheets are provided in the back of this manual for the user's convenience.

Case 1	$DATA/NX= 1 ,
	NC/C= 1 , -1 $
Case 2	$DATA/NX= 2 ,
	NC/C= 1 , -3 , 2 $
Case 3	$DATA/NX= 3 ,
	NC/C= 1 , -6 , 11 , -6 $
Case 4	$DATA/NX= 4 ,
	NC/C= 1 , -10 , 35 , -50 , 24 $
Case 5	$DATA/NX= 5 ,
	NC/C= 1 , -15 , 85 , -225 ,
	274 , -120 $
Case 6	$DATA/NX= 7 ,
	NC/C= 1.5 , 2.906 , 10.6 , 25.877 ,
	2.3 , 33 , 1.234 , 543.2 $
Last card	$DATA/NX=0$

Figure 1. POLRTS Coding Form with Sample Input

Column 1 is always blank.

Each case starts with $DATA/NX = highest power of x (maximum of 30) followed by a comma.

The list of coefficients starts with NC/C = followed by the coefficients of the equation in descending order of the powers of x. Each coefficient is followed by a comma except the last entry which is followed by a $. Columns 73 through 80 may not be used for coefficient entries but may be used for identification. Coefficients may be continued on as many cards as necessary as shown in cases 5 and 6. All coefficients must be entered even though they are zero.

Leading or trailing blanks are allowable in any of the numeric entries.

The last card in the deck must have NX set equal to zero followed by a $ to indicate the end of file.

Figure 2 is a listing of the input data deck.

```
$       EXECUTE
$       INCODE  IBMF
 $DATA/NX=1,NC/C=1,-1$
 $DATA/NX=2,NC/C=1,-3,2$
 $DATA/NX=3,NC/C=1,-6,11,-6$
 $DATA/NX=4,NC/C=1,-10,35,-50,24$
 $DATA/NX=5,NC/C=1,-15,85,-225,274,-120$
 $DATA/NX=0$
$       ENDJOB
```

Figure 2. Input Data Deck

Output Listings

In the output each polynomial to be solved and its roots is printed on a new page.

Figure 3 shows the output for the sample problems.

```
POLYNOMIAL TO BE SOLVED IS

      1.000000E 00 X

     -1.000000E 00

 ROOTS ARE AS FOLLOWS

         REAL PORTION    IMAGINARY PORTION

      1    1.000000E 00    0.            I
```

Case 1

Figure 3. Output from Sample Problems

```
POLYNOMIAL TO BE SOLVED IS
         1.000000E 00 X**2
        -3.000000E 00 X
         2.000000E 00

ROOTS ARE AS FOLLOWS
              REAL PORTION    IMAGINARY PORTION
         1    1.999998E 00    0.              I
         2    1.000001E 00    0.              I
```

Case 2

```
POLYNOMIAL TO BE SOLVED IS
         1.000000E 00 X**3
        -6.000000E 00 X**2
         1.100000E 01 X
        -6.000000E 00

ROOTS ARE AS FOLLOWS
              REAL PORTION    IMAGINARY PORTION
         1    2.000000E 00    0.              I
         2    1.000000E 00    0.              I
         3    3.000000E 00    0.              I
```

Case 3

Figure 3. Output from Sample Problems (cont.)

```
POLYNOMIAL TO BE SOLVED IS
      1.000000E 00 X**4
     -1.000000E 01 X**3
      3.500000E 01 X**2
     -5.000000E 01 X
      2.400000E 01

ROOTS ARE AS FOLLOWS
         REAL PORTION   IMAGINARY PORTION
    1    2.000000E 00    0.             I
    2    1.000000E 00    0.             I
    3    3.999996E 00    0.             I
    4    3.000003E 00    0.             I
```

Case 4

```
POLYNOMIAL TO BE SOLVED IS
      1.000000E 00 X**5
     -1.500000E 01 X**4
      8.500000E 01 X**3
     -2.250000E 02 X**2
      2.740000E 02 X
     -1.200000E 02

ROOTS ARE AS FOLLOWS
         REAL PORTION   IMAGINARY PORTION
    1    2.000000E 00    0.             I
    2    1.000000E 00    0.             I
    3    3.999995E 00    0.             I
    4    3.000003E 00    0.             I
    5    5.000003E 00    0.             I
```

Output Listing

Case 5

Figure 3. Output from Sample Problems (cont.)

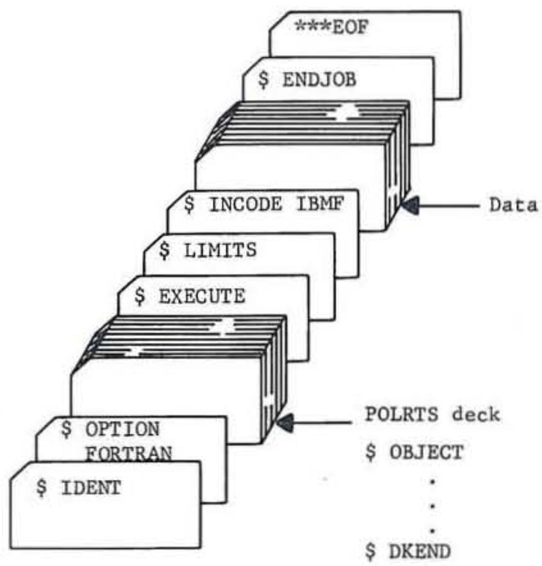
```
POLYNOMIAL TO BE SOLVED IS
        1,500000E 00 X**7
        2,906000E 00 X**6
        1,060000E 01 X**5
        2,587700E 01 X**4
        2,300000E 00 X**3
        3,300000E 01 X**2
        1,234000E 00 X
        5,432000E 02

ROOTS ARE AS FOLLOWS
         REAL PORTION   IMAGINARY PORTION
      1   2,137941E-01   2,873626E 00I
      2   2,137941E-01  -2,873626E 00I
      3   1,441992E 00   1,153711E 00I
      4   1,441992E 00  -1,153711E 00I
      5  -1,244203E 00   1,756273E 00I
      6  -1,244203E 00  -1,756273E 00I
      7  -2,760499E 00   0,            I
```

Case 6

Figure 3. Output from Sample Problems (cont.)

Control Cards and Deck Setup

Figure 4 shows the cards used to run the sample problem on the GE-625/635 computers.



APPENDIX A
PROGRAM LISTING

```
$         IDENT    AAV,GE,FORTRAN,FOOTS OF A POLYNOMIAL
$         OPTION   FORTRAN,GO
$         FORTRAN LSTOU,DECK,STAB
$         INCODE   IBMF
*POLY                         POLYNOMIAL ROOTS PROGRAM
*                             CD600D5.001      DATE 05/05/65
      DIMENSION C(31),RR(31),RI(31),X(31)
      DATA X/6H X**30,6H X**29,6H X**28,6H X**27,6H X**26,6H X**25,
     A6H X**24,6H X**23,6H X**22,6H X**21,6H X**20,6H X**19,6H X**18,
     B6H X**17,6H X**16,6H X**15,6H X**14,6H X**13,6H X**12,6H X**11,
     C6H X**10,6H X**9 ,6H X**8 ,6H X**7 ,6H X**6 ,6H X**5 ,6H X**4 ,
     D6H X**3 ,6H X**2 ,6H X     ,6H       /
      NAMELIST/DATA/NC,NX,C
*     SAMPLE INPUT
*     $DATA/NX=3,NC/C=12.3,4.45,7.5,123$
*     NX IS HIGHEST POWER, C IS LIST OF NX+1 COEFFICIENTS (DESCENDING)
*     NX=0 SIGNALS END OF DATA
    1 READ(5,DATA)
      IF(NX.EQ.0)GO TO 16
      IF(NC.EQ.(NX+1))GO TO 3
    2 WRITE(6,4)
    4 FORMAT(34H1       WRONG NUMBER OF COEFFICIENTS)
      GO TO 1
    3 IF(C(NC))5,6,5
    6 WRITE(6,7)
    7 FORMAT(29H1       ZERO ROOT, REDUCE POWER)
      GO TO 1
    5 WRITE(6,8)
    8 FORMAT(31H1      POLYNOMIAL TO BE SOLVED IS/)
      J=31-NX
      DO 9 I=1,NC
      WRITE(6,10)C(I),X(J)
    9 J=J+1
```

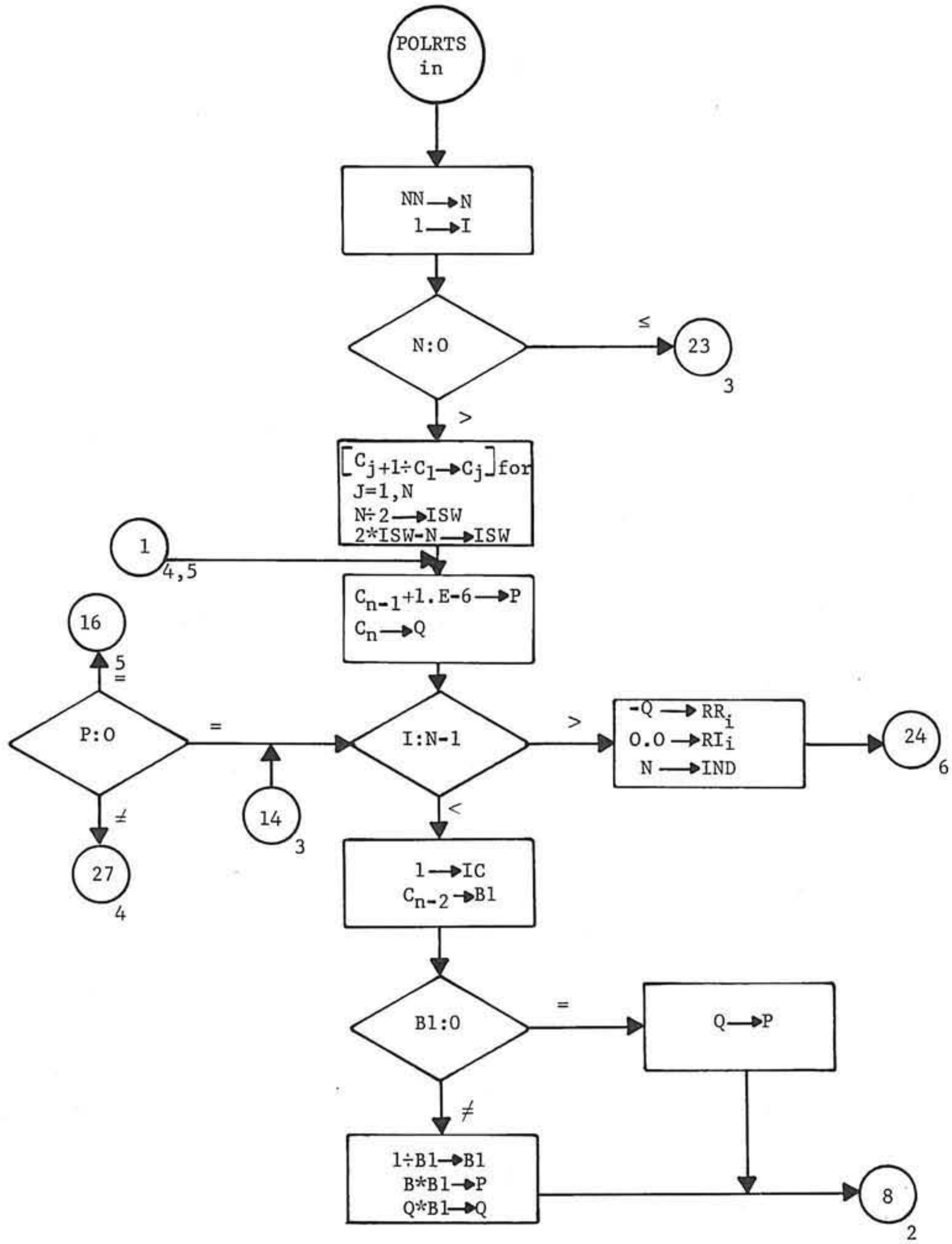
```
   10 FORMAT(1H010X,1PE13.6,A6)                                        POLY0330
      CALL POLRTS(C,RR,RI,NX,IND)                                      POLY0340
      IF(IND.LT.NX)GO TO 11                                            POLY0350
      WRITE(6,12)                                                      POLY0360
      WRITE(6,17)                                                      POLY0370
   12 FORMAT(//1H0,5X,20HROOTS ARE AS FOLLOWS)                         POLY0380
      GO TO 13                                                         POLY0390
   11 WRITE(6,14)                                                      POLY0400
      WRITE(6,17)                                                      POLY0410
   14 FORMAT(//1H0,5X,27HPARTIAL SOLUTION AS FOLLOWS)                  POLY0420
   13 WRITE(6,15)(I,RR(I),RI(I),I=1,IND)                               POLY0430
   15 FORMAT(1H010X,I2,1PE15.6,1PE15.6,1HI)                            POLY0440
   17 FORMAT(1H013X,32HREAL PORTION   IMAGINARY PORTION)               POLY0450
      GO TO 1                                                          POLY0460
   16 STOP                                                             POLY0470
      END                                                              POLY0480
$       FORTRAN LSTOU,DECK,STAB                                        POLR0000
$       INCODE  IBMF                                                   POLR0010
*POLRTS       POLYNOMIAL ROOTS BY BAIRSTOW,S METHOD                    POLR0020
*                                 CD600D5.001                          POLR0030
      SUBROUTINE POLRTS(C,RR,RI,NN,IND)                                POLR0040
*     BAIRSTOWS METHOD FOR FINDING POLYNOMIAL ROOTS                    POLR0050
*     POLYNOMIALS HAVE N+1 COEFFICIENTS STORED IN C IN                 POLR0060
*     ORDER OF DESCENDING POWERS, ZERO ROOTS NOT PERMITTED             POLR0070
      DIMENSION C(1),RR(1),RI(1)                                       POLR0080
      N=NN                                                             POLR0090
      I=1                                                              POLR0100
      IF(N)23,23,30                                                    POLR0110
   30 P=1.0/C(1)                                                       POLR0120
      DO 4 J=1,N                                                       POLR0130
    4 C(J)=P*C(J+1)                                                    POLR0140
      ISW=N/2                                                          POLR0150
```

```
      ISW=ISW+ISW-N
    1 P=C(N-1)+1.E-6
      Q=C(N)
      IF(I-N+1)5,14,2
    2 RR(I)=-Q
      RI(I)=0.0
      IND=N
      GO TO 24
    5 IC=1
      B1=C(N-2)
      IF(B1)7,28,7
    7 B1=1.0/B1
      P=P*B1
      Q=Q*B1
    8 B1=1.0
      B3=1.0
      B2=0.0
      B4=0.0
      DO 10 J=I,N
      RR(J)=C(J)-P*B1-Q*B2
      IF(J-N)9,10,10
    9 RI(J)=RR(J)-P*B3-Q*B4
      B2=B1
      B4=B3
      B1=RR(J)
      B3=RI(J)
   10 CONTINUE
      RI(N-1)=RI(N-1)-RR(N-1)
      B2=1.0
      B3=RI(N-1)
      B4=RI(N-2)
      IF(I-N+2)3,6,3
    3 B2=RI(N-3)
```

```
    6 B1=B4*B4-B3*B2                                         POLR0490
   11 B1=1.0/B1                                              POLR0500
      B2=(RR(N-1)*B4-RR(N)*B2)*B1                            POLR0510
      B3=(RR(N-1)*B3-RR(N)*B4)*B1                            POLR0520
      IF(ABS(B2)/(ABS(P)+1.0)-5.E-6)12,12,13                 POLR0530
   12 IF(ABS(B3)/(ABS(Q)+1.0)-5.E-6)21,21,13                 POLR0540
   13 P=P+B2                                                 POLR0550
      Q=Q-B3                                                 POLR0560
      IC=IC+1                                                POLR0570
      IF(IC-100)8,8,31                                       POLR0580
   31 IF(ISW)32,23,24                                        POLR0590
   32 ISW=1                                                  POLR0600
      IND=I-1                                                POLR0610
      J=N                                                    POLR0620
      N=N+1                                                  POLR0630
      C(N)=0.                                                POLR0640
   33 C(J+1)=C(J+1)+C(J)                                     POLR0650
      J=J-1                                                  POLR0660
      IF(J)34,34,33                                          POLR0670
   34 C(1)=C(1)+1.0                                          POLR0680
      GO TO 1                                                POLR0690
   14 IF(P)27,16,27                                          POLR0700
   27 B4=4.0*Q/(P*P)                                         POLR0710
      IF(ABS(B4)-1.E-6)15,15,16                              POLR0720
   15 RR(I)=-P                                               POLR0730
      RR(I+1)=-Q/P                                           POLR0740
      GO TO 19                                               POLR0750
   16 RR(I)=-.5*P                                            POLR0760
      RR(I+1)=RR(I)                                          POLR0770
      B1=P*P-4.0*Q                                           POLR0780
      IF(B1)17,19,18                                         POLR0790
   17 RI(I)=.5*SQRT(-B1)                                     POLR0800
      RI(I+1)=-RI(I)                                         POLR0810
```

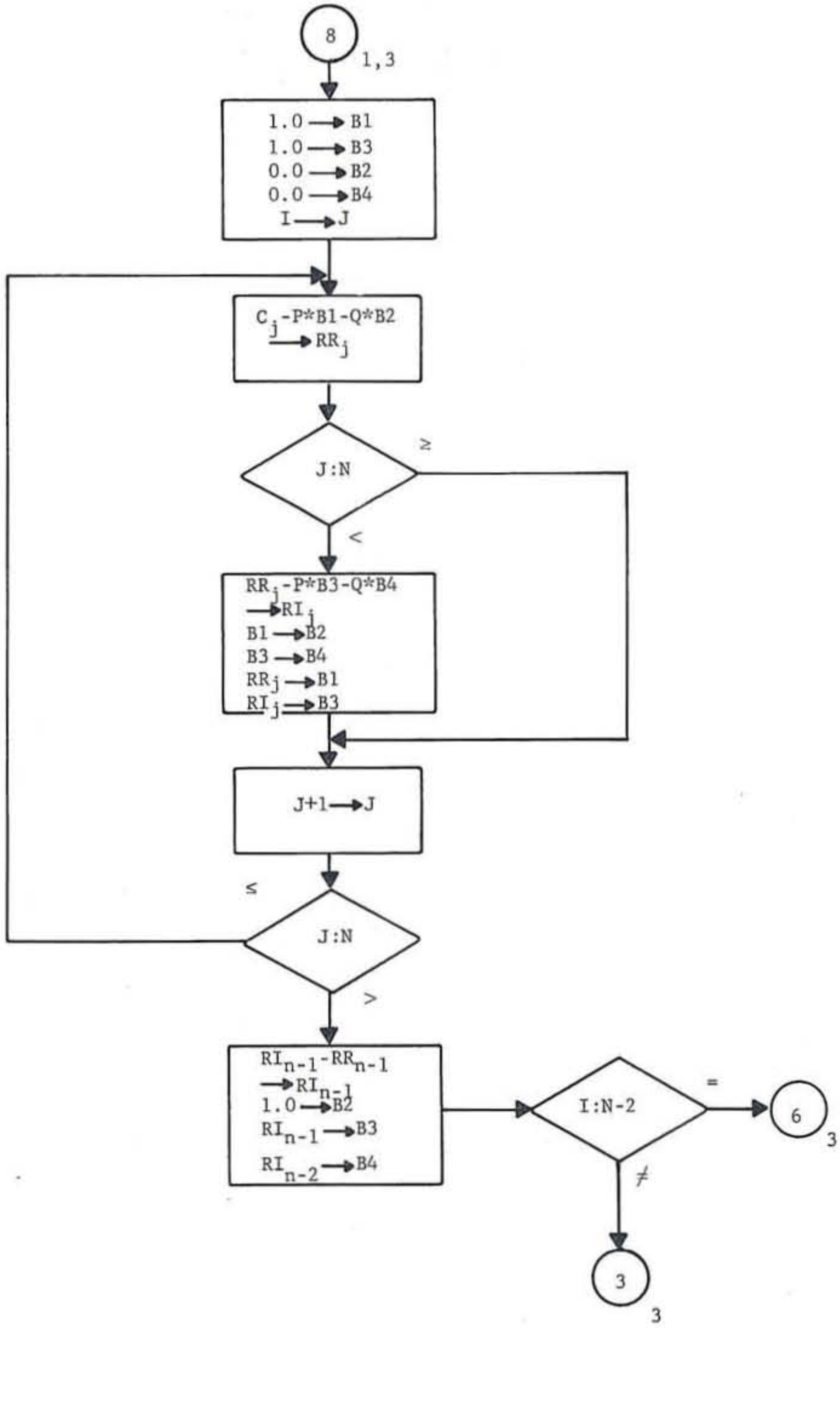
```
      GO TO 20                                                          POLR0820
   18 B1=.5*SQRT(B1)                                                    POLR0830
      RR(I)=RR(I)+B1                                                    POLR0840
      RR(I+1)=RR(I+1)-B1                                                POLR0850
   19 RI(I)=0.0                                                         POLR0860
      RI(I+1)=0.0                                                       POLR0870
   20 I=I+2                                                             POLR0880
      IF(I-N)1,1,35                                                     POLR0890
   35 IF(ISW)23,23,36                                                   POLR0900
   36 K=IND+1                                                           POLR0910
      DO 38 J=K,N                                                       POLR0920
      IF(ABS(RI(J))-1.E-6)37,37,38                                      POLR0930
   37 IF(ABS(RR(J)+1.0)-1.E-6)39,39,38                                  POLR0940
   38 CONTINUE                                                          POLR0950
      GO TO 24                                                          POLR0960
   39 DO 40 K=J,N                                                       POLR0970
      RR(K)=RR(K+1)                                                     POLR0980
   40 RI(K)=RI(K+1)                                                     POLR0990
      IND=N-1                                                           POLR1000
      GO TO 24                                                          POLR1010
   21 DO 22 J=I,N                                                       POLR1020
   22 C(J)=RR(J-2)                                                      POLR1030
      GO TO 14                                                          POLR1040
   23 IND=I-1                                                           POLR1050
   24 RETURN                                                            POLR1060
   28 P=Q                                                               POLR1070
      GO TO 8                                                           POLR1080
      END                                                               POLR1090
```

APPENDIX B
FLOWCHARTS



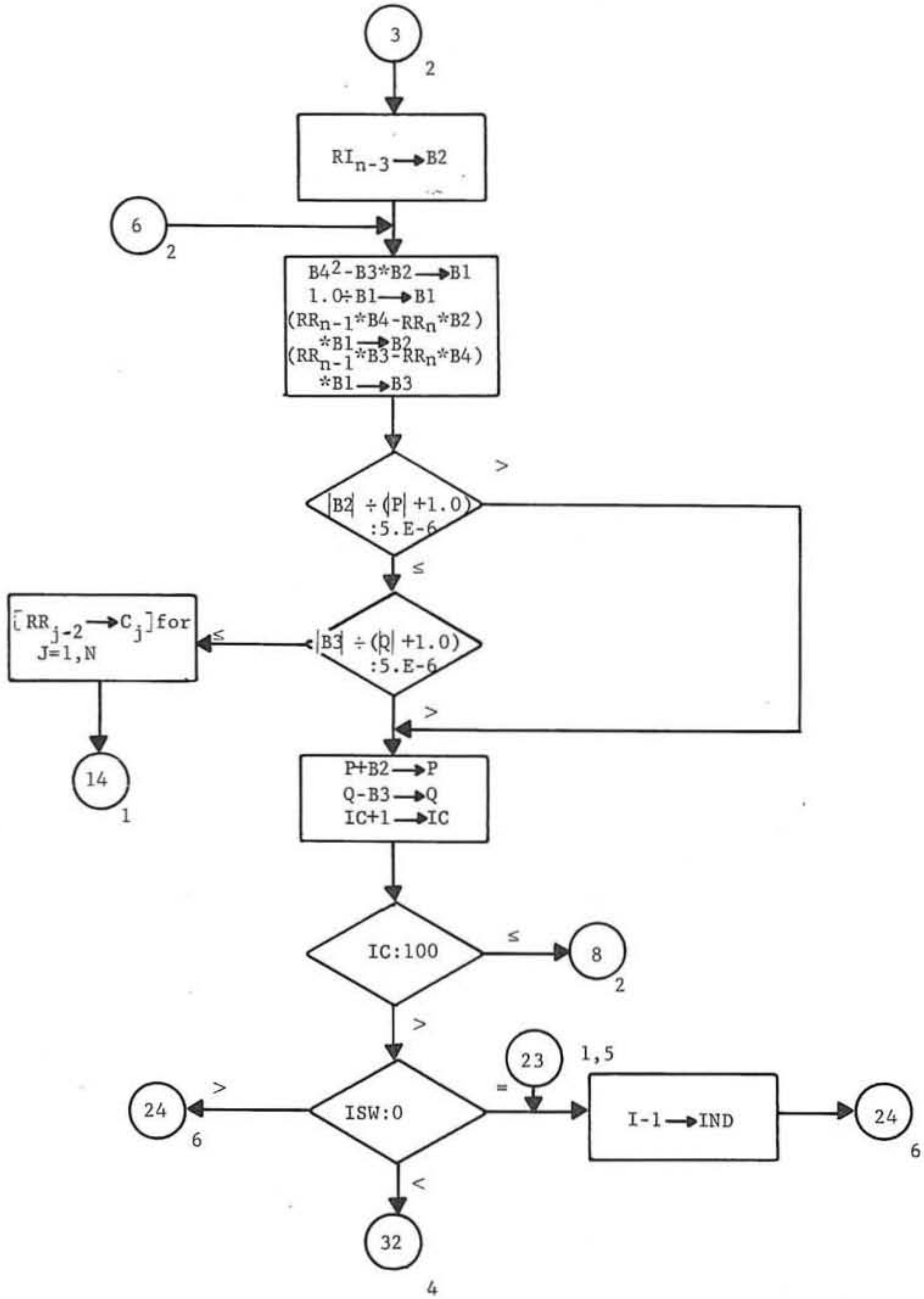
POLRTS

(1)


8
1,3
1.0 → B1
1.0 → B3
0.0 → B2
0.0 → B4
I → J
C_j-P*B1-Q*B2
→ RR_j
J:N
≥
<
RR_j-P*B3-Q*B4
→ RI_j
B1 → B2
B3 → B4
RR_j → B1
RI_j → B3
J+1 → J
≤
J:N
>
RI_n-1-RR_n-1
→ RI_n-1
1.0 → B2
RI_n-1 → B3
RI_n-2 → B4
I:N-2
=
6
3
≠
3
3

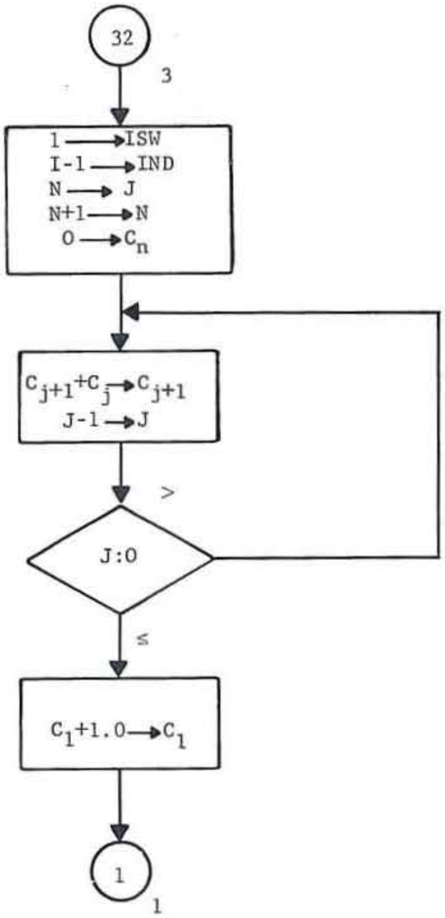
POLRTS

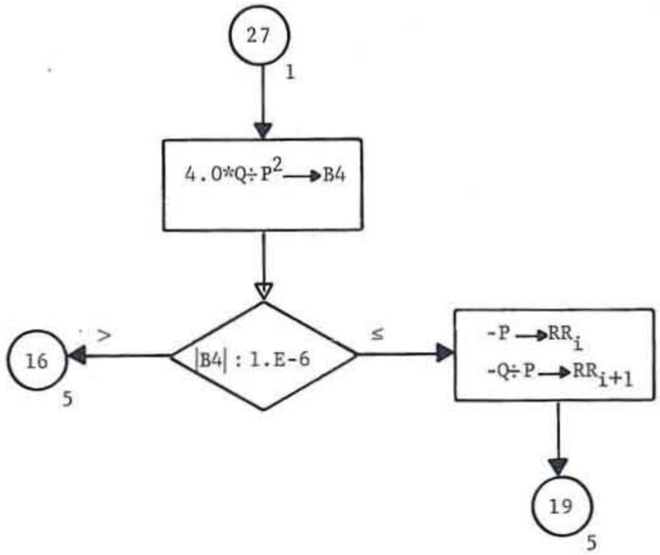
(2)


3
2
RIn-3 → B2
6
2
B4² - B3*B2 → B1
1.0÷B1 → B1
(RRn-1*B4-RRn*B2)
*B1 → B2
(RRn-1*B3-RRn*B4)
*B1 → B3
>
|B2| ÷ (|P| +1.0)
:5.E-6
≤
[RRj-2 → Cj] for
J=1,N
≤
|B3| ÷ (|Q| +1.0)
:5.E-6
>
14
1
P+B2 → P
Q-B3 → Q
IC+1 → IC
IC:100
≤
8
2
>
23
1,5
24
6
>
ISW:0
=
I-1 → IND
24
6
<
32
4

POLRTS

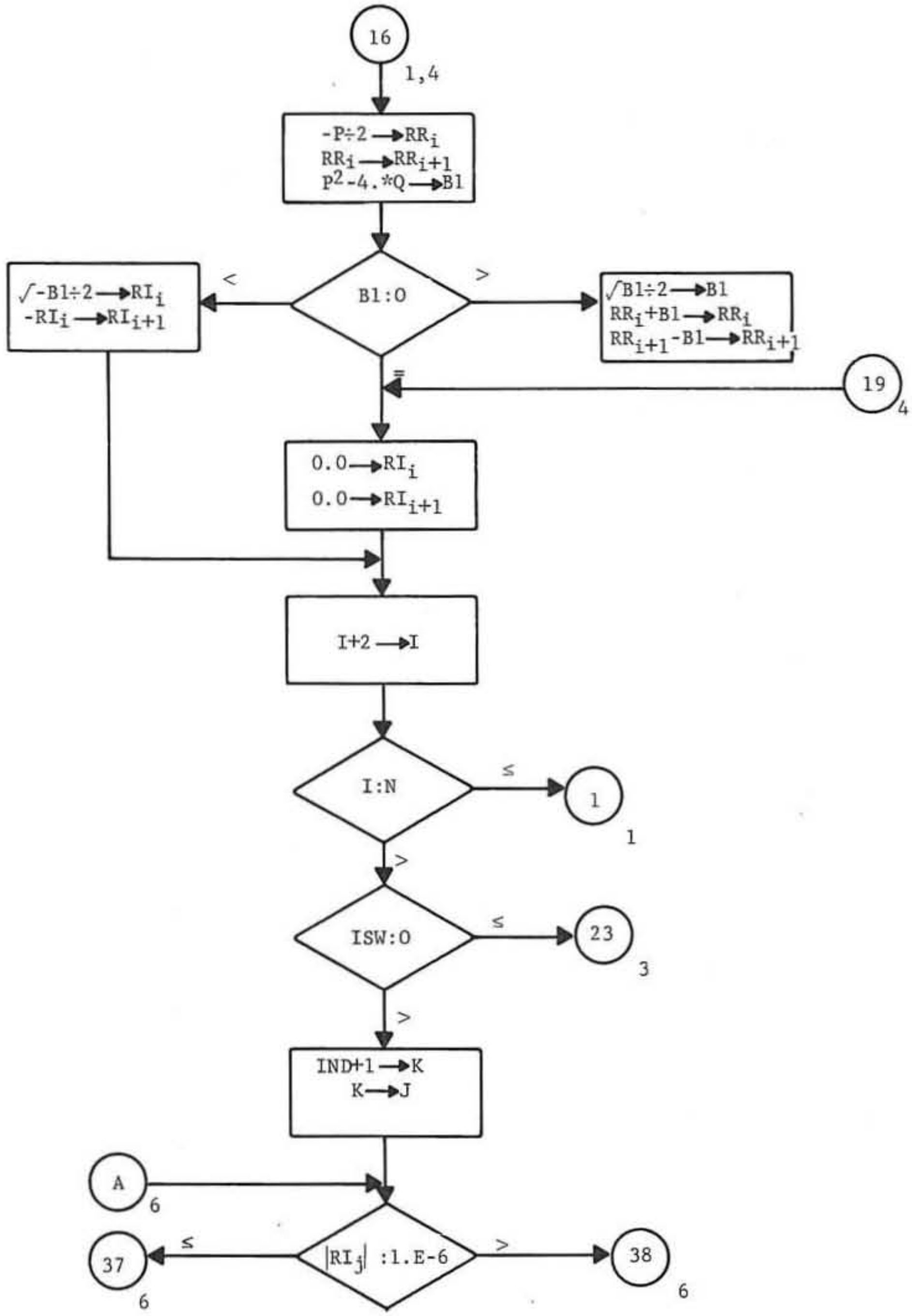
(3)


32
3
1 → ISW
I-1 → IND
N → J
N+1 → N
0 → Cn
Cj+1+Cj → Cj+1
J-1 → J
>
J:0
≤
C1+1.0 → C1
1
1


27
1
4.0*Q÷P2 → B4
>
|B4| : 1.E-6
≤
16
5
-P → RRi
-Q÷P → RRi+1
19
5

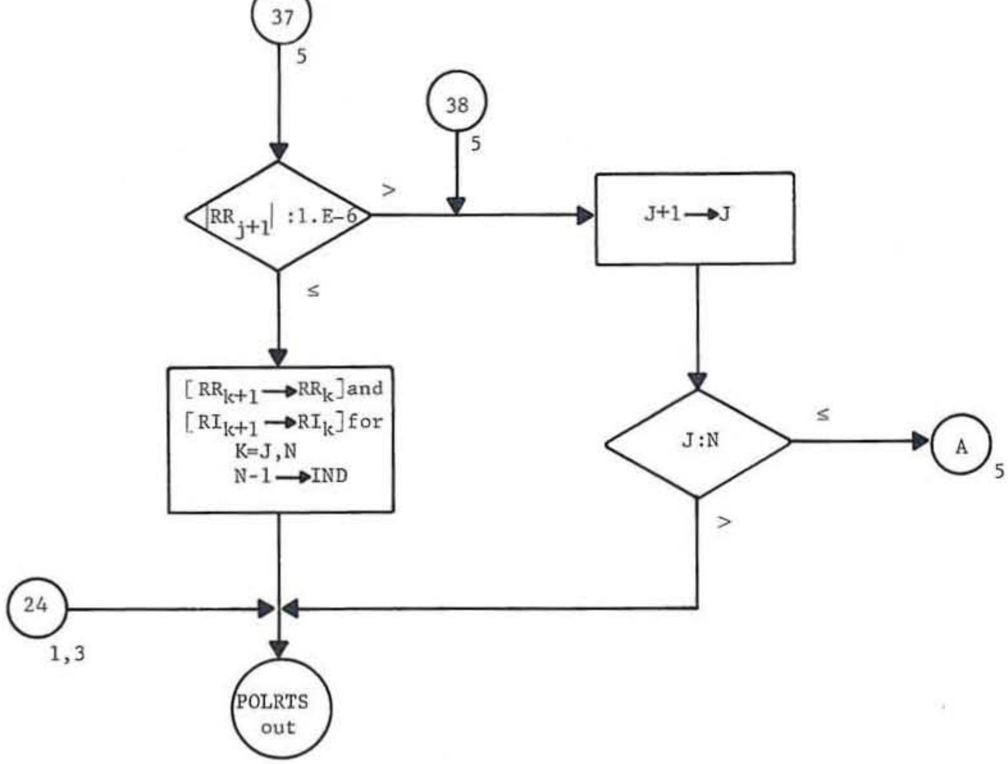
POLRTS

(4)


16
1,4
-P÷2 → RRi
RRi → RRi+1
P2-4.*Q → B1
B1:0
<
>
=
√-B1÷2 → RIi
-RIi → RIi+1
√B1÷2 → B1
RRi+B1 → RRi
RRi+1-B1 → RRi+1
19
4
0.0 → RIi
0.0 → RIi+1
I+2 → I
I:N
≤
1
1
>
ISW:0
≤
23
3
>
IND+1 → K
K → J
A
6
|RIj| :1.E-6
≤
37
6
>
38
6

POLRTS

5


37
5
38
5
|RR j+1| :1.E-6
>
≤
J+1 → J
[RR k+1 → RR k] and
[RI k+1 → RI k] for
K=J,N
N-1 → IND
J:N
≤
A
5
>
24
1,3
POLRTS
out

POLRTS

(6)

POLRTS Coding Sheet Instructions

For the general equations:

$$A_n x^n + A_{n-1} x^{n-1} + \ldots\ldots\ldots + A_1 x + A_0 = 0$$

Enter data as:

NX = highest power of x (maximum value of 30)

NC/C = coefficients in descending order starting with A_n through A_0 followed by $ after A_0

NX = 0 after last case to terminate run.

Example:

Equations to be solved:

Case 1: $x - 1 = 0$

Case 2: $x^2 - 3x + 2 = 0$

Case 3: $x^3 - 6x^2 + 11x - 6 = 0$

Case 4: $x^4 - 10x^3 + 35x^2 - 50x + 24 = 0$

Case 5: $x^5 - 15x^4 + 85x^3 - 225x^2 + 274x - 120 = 0$

Case 6: $1.5x^7 + 2.906x^6 + 10.6x^5 + 25.877x^4 + 2.3x^3 + 33x^2 + 1.234x + 543.2 = 0$

Above equations coded for POLRTS:

Case 1	$DATA/NX=1,
	NC/C=1,-1$
Case 2	$DATA/NX=2,
	NC/C=1,-3,2$
Case 3	$DATA/NX=3,
	NC/C=1,-6,11,-6$
Case 4	$DATA/NX=4,
	NC/C=1,-10,35,-50,24$
Case 5	$DATA/NX=5,
	NC/C=1,-15,85,-225,
	274,-120$
Case 6	$DATA/NX=7,
	NC/C=1.5,2.906,10.6,25.877,
	2.3,33,1.234,543.2$
Last card	$DATA/NX=0$

POLRTS Coding Sheet

Col. CASE 1
2
$DATA/NX=______,

NC/C= ______, ______, ______, ______, ______, ______,
A_n A_{n-1} A_{n-2} A_{n-3} A_{n-4} . . .

______, ______, ______, ______, ______, ______,
______, ______, ______, ______, ______, ______,
______, ______, ______, ______, ______, ______,

Col. CASE 2
2
$DATA/NX=______,

NC/C= ______, ______, ______, ______, ______, ______,
A_n A_{n-1} A_{n-2} A_{n-3} A_{n-4} . . .

______, ______, ______, ______, ______, ______,
______, ______, ______, ______, ______, ______,
______, ______, ______, ______, ______, ______,

Col. CASE N
2
$DATA/NX=______,

NC/C= ______, ______, ______, ______, ______, ______,
A_n A_{n-1} A_{n-2} A_{n-3} A_{n-4} . . .

______, ______, ______, ______, ______, ______,
______, ______, ______, ______, ______, ______,
______, ______, ______, ______, ______, ______,

TERMINATION CARD

Col.
2
$DATA/NX= 0$

NOTE TO KEYPUNCH OPERATOR:

Start all cards in column 2.
Terminate punching after $ in a given case.

POLRTS Coding Sheet Instructions

For the general equations:

$$A_n x^n + A_{n-1} x^{n-1} + \ldots\ldots\ldots + A_1 x + A_0 = 0$$

Enter data as:

NX = highest power of x (maximum value of 30)

NC/C = coefficients in descending order starting with A_n through A_0 followed by $ after A_0

NX = 0 after last case to terminate run.

Example:

Equations to be solved:

Case 1: $x - 1 = 0$

Case 2: $x^2 - 3x + 2 = 0$

Case 3: $x^3 - 6x^2 + 11x - 6 = 0$

Case 4: $x^4 - 10x^3 + 35x^2 - 50x + 24 = 0$

Case 5: $x^5 - 15x^4 + 85x^3 - 225x^2 + 274x - 120 = 0$

Case 6: $1.5x^7 + 2.906x^6 + 10.6x^5 + 25.877x^4 + 2.3x^3 + 33x^2 + 1.234x + 543.2 = 0$

Above equations coded for POLRTS:

Case 1	$DATA/NX= 1 ,
	NC/C= 1 , -1 $
Case 2	$DATA/NX= 2 ,
	NC/C= 1 , -3 , 2 $
Case 3	$DATA/NX= 3 ,
	NC/C= 1 , -6 , 11 , -6 $
Case 4	$DATA/NX= 4 ,
	NC/C= 1 , -10 , 35 , -50 , 24 $
Case 5	$DATA/NX= 5 ,
	NC/C= 1 , -15 , 85 , -225 ,
	274 , -120 $
Case 6	$DATA/NX= 7 ,
	NC/C= 1.5 , 2.906 , 10.6 , 25.877 ,
	2.3 , 33 , 1.234 , 543.2 $
Last card	$DATA/NX=0 $

POLRTS Coding Sheet

Col. CASE 1
2
$DATA/NX=______,

NC/C=__________, __________, __________, __________, __________, __________,
A_n A_{n-1} A_{n-2} A_{n-3} A_{n-4} . . .

__________, __________, __________, __________, __________, __________,

__________, __________, __________, __________, __________, __________,

__________, __________, __________, __________, __________, __________,

---

Col. CASE 2
2
$DATA/NX=______,

NC/C=__________, __________, __________, __________, __________, __________,
A_n A_{n-1} A_{n-2} A_{n-3} A_{n-4} . . .

__________, __________, __________, __________, __________, __________,

__________, __________, __________, __________, __________, __________,

__________, __________, __________, __________, __________, __________,

---

Col. CASE N
2
$DATA/NX=______,

NC/C=__________, __________, __________, __________, __________, __________,
A_n A_{n-1} A_{n-2} A_{n-3} A_{n-4} . . .

__________, __________, __________, __________, __________, __________,

__________, __________, __________, __________, __________, __________,

__________, __________, __________, __________, __________, __________,

---

TERMINATION CARD

Col.
2
$DATA/NX= 0$

NOTE TO KEYPUNCH OPERATOR:

Start all cards in column 2.
Terminate punching after $ in a given case.

POLRTS Coding Sheet Instructions

For the general equations:

$$A_n x^n + A_{n-1} x^{n-1} + \dots\dots\dots + A_1 x + A_0 = 0$$

Enter data as:

NX = highest power of x (maximum value of 30)

NC/C = coefficients in descending order starting with A_n through A_0 followed by $ after A_0

NX = 0 after last case to terminate run.

Example:

Equations to be solved:

Case 1: $x - 1 = 0$

Case 2: $x^2 - 3x + 2 = 0$

Case 3: $x^3 - 6x^2 + 11x - 6 = 0$

Case 4: $x^4 - 10x^3 + 35x^2 - 50x + 24 = 0$

Case 5: $x^5 - 15x^4 + 85x^3 - 225x^2 + 274x - 120 = 0$

Case 6: $1.5x^7 + 2.906x^6 + 10.6x^5 + 25.877x^4 + 2.3x^3 + 33x^2 + 1.234x + 543.2 = 0$

Above equations coded for POLRTS:

Case 1	$DATA/NX= 1 ,
	NC/C= 1 , -1 $
Case 2	$DATA/NX= 2 ,
	NC/C= 1 , -3 , 2 $
Case 3	$DATA/NX= 3 ,
	NC/C= 1 , -6 , 11 , -6 $
Case 4	$DATA/NX= 4 ,
	NC/C= 1 , -10 , 35 , -50 , 24 $
Case 5	$DATA/NX= 5 ,
	NC/C= 1 , -15 , 85 , -225 ,
	274 , -120 $
Case 6	$DATA/NX= 7 ,
	NC/C= 1.5 , 2.906 , 10.6 , 25.877 ,
	2.3 , 33 , 1.234 , 543.2 $
Last card	$DATA/NX=0 $

POLRTS Coding Sheet

Col.
2
$DATA/NX=______,

CASE 1

NC/C= ________, ________, ________, ________, ________, ________,

A_n A_{n-1} A_{n-2} A_{n-3} A_{n-4} . . .

________, ________, ________, ________, ________, ________,
________, ________, ________, ________, ________, ________,
________, ________, ________, ________, ________, ________,

Col.
2
$DATA/NX=______,

CASE 2

NC/C= ________, ________, ________, ________, ________, ________,

A_n A_{n-1} A_{n-2} A_{n-3} A_{n-4} . . .

________, ________, ________, ________, ________, ________,
________, ________, ________, ________, ________, ________,
________, ________, ________, ________, ________, ________,

Col.
2
$DATA/NX=______,

CASE N

NC/C= ________, ________, ________, ________, ________, ________,

A_n A_{n-1} A_{n-2} A_{n-3} A_{n-4} . . .

________, ________, ________, ________, ________, ________,
________, ________, ________, ________, ________, ________,
________, ________, ________, ________, ________, ________,

TERMINATION CARD

Col.
2
$DATA/NX= 0$

NOTE TO KEYPUNCH OPERATOR:

Start all cards in column 2.
Terminate punching after $ in a given case.

POLRTS Coding Sheet Instructions

For the general equations:

$$A_n x^n + A_{n-1} x^{n-1} + \ldots\ldots\ldots + A_1 x + A_0 = 0$$

Enter data as:

NX = highest power of x (maximum value of 30)

NC/C = coefficients in descending order starting with A_n through A_0 followed by $ after A_0

NX = 0 after last case to terminate run.

Example:

Equations to be solved:

Case 1: $x - 1 = 0$

Case 2: $x^2 - 3x + 2 = 0$

Case 3: $x^3 - 6x^2 + 11x - 6 = 0$

Case 4: $x^4 - 10x^3 + 35x^2 - 50x + 24 = 0$

Case 5: $x^5 - 15x^4 + 85x^3 - 225x^2 + 274x - 120 = 0$

Case 6: $1.5x^7 + 2.906x^6 + 10.6x^5 + 25.877x^4 + 2.3x^3 + 33x^2 + 1.234x + 543.2 = 0$

Above equations coded for POLRTS:

```
Case 1      $DATA/NX= 1 ,
            NC/C= 1 , -1 $
Case 2      $DATA/NX= 2 ,
            NC/C= 1 , -3 , 2 $
Case 3      $DATA/NX= 3 ,
            NC/C= 1 , -6 , 11 , -6 $
Case 4      $DATA/NX= 4 ,
            NC/C= 1 , -10 , 35 , -50 , 24 $
Case 5      $DATA/NX= 5 ,
            NC/C= 1 , -15 , 85 , -225 ,
                 274 , -120 $
Case 6      $DATA/NX= 7 ,
            NC/C= 1.5 , 2.906 , 10.6 , 25.877 ,
                 2.3 , 33 , 1.234 , 543.2 $
Last card   $DATA/NX=0$
```

POLRTS Coding Sheet

Col. CASE 1
2
$DATA/NX=______,

NC/C=__________, __________, __________, __________, __________, __________,
A_n A_{n-1} A_{n-2} A_{n-3} A_{n-4} . . .

__________, __________, __________, __________, __________, __________,
__________, __________, __________, __________, __________, __________,
__________, __________, __________, __________, __________, __________,

---

Col. CASE 2
2
$DATA/NX=______,

NC/C=__________, __________, __________, __________, __________, __________,
A_n A_{n-1} A_{n-2} A_{n-3} A_{n-4} . . .

__________, __________, __________, __________, __________, __________,
__________, __________, __________, __________, __________, __________,
__________, __________, __________, __________, __________, __________,

---

Col. CASE N
2
$DATA/NX=______,

NC/C=__________, __________, __________, __________, __________, __________,
A_n A_{n-1} A_{n-2} A_{n-3} A_{n-4} . . .

__________, __________, __________, __________, __________, __________,
__________, __________, __________, __________, __________, __________,
__________, __________, __________, __________, __________, __________,

---

TERMINATION CARD

Col.
2
$DATA/NX= 0$

NOTE TO KEYPUNCH OPERATOR:

Start all cards in column 2.
Terminate punching after $ in a given case.

POLRTS Coding Sheet Instructions

For the general equations:

$$A_n x^n + A_{n-1} x^{n-1} + \dots\dots\dots + A_1 x + A_0 = 0$$

Enter data as:

NX = highest power of x (maximum value of 30)

NC/C = coefficients in descending order starting with A_n through A_0 followed by \$ after A_0

NX = 0 after last case to terminate run.

Example:

Equations to be solved:

Case 1: $x - 1 = 0$

Case 2: $x^2 - 3x + 2 = 0$

Case 3: $x^3 - 6x^2 + 11x - 6 = 0$

Case 4: $x^4 - 10x^3 + 35x^2 - 50x + 24 = 0$

Case 5: $x^5 - 15x^4 + 85x^3 - 225x^2 + 274x - 120 = 0$

Case 6: $1.5x^7 + 2.906x^6 + 10.6x^5 + 25.877x^4 + 2.3x^3 + 33x^2 + 1.234x + 543.2 = 0$

Above equations coded for POLRTS:

Case 1	$DATA/NX= 1 ,
	NC/C= 1 , -1 $
Case 2	$DATA/NX= 2 ,
	NC/C= 1 , -3 , 2 $
Case 3	$DATA/NX= 3 ,
	NC/C= 1 , -6 , 11 , -6 $
Case 4	$DATA/NX= 4 ,
	NC/C= 1 , -10 , 35 , -50 , 24 $
Case 5	$DATA/NX= 5 ,
	NC/C= 1 , -15 , 85 , -225 ,
	274 , -120 $
Case 6	$DATA/NX= 7 ,
	NC/C= 1.5 , 2.906 , 10.6 , 25.877 ,
	2.3 , 33 , 1.234 , 543.2 $
Last card	$DATA/NX=0$

POLRTS Coding Sheet

Col. CASE 1
2
$DATA/NX=______,

NC/C=__________, __________, __________, __________, __________, __________,
A_n A_{n-1} A_{n-2} A_{n-3} A_{n-4} . . .

__________, __________, __________, __________, __________, __________,

__________, __________, __________, __________, __________, __________,

__________, __________, __________, __________, __________, __________,

---

Col. CASE 2
2
$DATA/NX=______,

NC/C=__________, __________, __________, __________, __________, __________,
A_n A_{n-1} A_{n-2} A_{n-3} A_{n-4} . . .

__________, __________, __________, __________, __________, __________,

__________, __________, __________, __________, __________, __________,

__________, __________, __________, __________, __________, __________,

---

Col. CASE N
2
$DATA/NX=______,

NC/C=__________, __________, __________, __________, __________, __________,
A_n A_{n-1} A_{n-2} A_{n-3} A_{n-4} . . .

__________, __________, __________, __________, __________, __________,

__________, __________, __________, __________, __________, __________,

__________, __________, __________, __________, __________, __________,

---

TERMINATION CARD

Col.
2
$DATA/NX= 0$

NOTE TO KEYPUNCH OPERATOR:

Start all cards in column 2.
Terminate punching after $ in a given case.

TITLE: GE-625/635 Math Routine

CPB #: 1152A

FROM:

Name: ______________________

Position: ______________________

Address: ______________________

____

Comments concerning this publication are solicited for use in improving future editions. Please provide any recommended additions, deletions, corrections, or other information you deem necessary for improving this manual. The following space is provided for your comments.

COMMENTS: ______________________

Please cut along this line

NO POSTAGE NECESSARY IF MAILED IN U.S.A.
Fold on two lines shown on reverse
side, staple, and mail.

Progress Is Our Most Important Product

GENERAL ELECTRIC

INFORMATION SYSTEMS DIVISION

LITHO U.S.A.