JOHNNIAC LIBRARY LISTING

SEPT. 1959

J103B,E	PUNCH STYLE B, ABSOLUTE -- 11 PER CARD
J104B,E	BLOCK TRANSFER ROUTINE
J105B,E	PUNCH STYLE D, ABSOLUTE -- 22 PER CARD
J106B,E	SQUARE ROOT
J112A	LOAD STYLE D CARDS
J116B,E	LOG-BASE TWO
J120B,E	BLOCK TRANSFER TO AND FROM DRUM
J122A	LOAD STYLE B CARDS - LOW
J123A	LOAD STYLE B CARDS - HIGH
J124B	EIGENVALUES AND EIGENVECTORS OF A REAL STMMETRIC MATRIX
J126B,E	SOLUTION OF A SYSTEM OF ORDINARY DIFFERENTIAL EQUATIONS
J127B,E	ARCTANGENT X
J129B,E	CARD IMAGE TO CHARACTERS
J130B,E	DECIMAL INPUT VARIABLE FORMAT
J131B,E	DECIMAL INPUT FIXED FORMAT ONE/CARD
J132B,E	PRESET OR RESET ROUTINE
J133B,E	PSEUDO B-BOX
J135A	LOADER FOR STYLE F ABSOLUTE BINARY CARDS
J136A	LOADER FOR STYLE E
J137B,E	MANIPULATION OF ARRAYS
J138B,E	MULTIPLICATION OF ARRAYS
J140A	CLEAR H.S.S. TO ZERO
J142E	PUNCH STYLE F BINARY CARDS
J148E	PUNCH STYLE F WITH I.D.
J149B,E	BLOCK TRANSFER TO AND FROM DRUM WITH TALLY
J151B,E	SIN Y, COS Y, SIN 2Y, COS 2Y
J153A	CLEAR H.S.S. TO PRESCRIBED WORD
J154B,E	SIN PI Y, COS PI Y
J156B,E	INTEGRAL ROOT X1/P
J159E	INVERSE INTERPOLATION, A REAL ROOT OF F(X)=0
J160E	CARD TO MOUSE MATRIX
J161E	CARD TO MOUSE MATRIX WITH INTERMEDIATE COMPUTATION
J162E	MOUSE MATRIX TO H.S.S.
J163E	H.S.S. TO MOUSE MATRIX
J164E	MOUSE MATRIX TO PUNCH
J165E	MOUSE MATRIX TO PUNCH WITH INTERMEDIATE COMPUTATION
J166E	MOUSE MATRIX TO PUNCH AND ECHO CHECK
J167E	MOUSE MATRIX TO PUNCH AND ECHO CHECK WITH INTERMEDIATE COMPUTATION
J178E	LEAST SQUARES POLYNOMIAL APPROXIMATION -- FL. POINT
J180A	LOADER FOR STYLE F CARDS WITH BLOCK CHECK SUMS
J181E	PUNCH STYLE F CARDS WITH BLOCK CHECK SUMS
J182E	PUNCH STYLE F CARDS WITH BLOCK CHECK SUMS AND I.D.
J185E	INTEGRATION OF N SIMULTANEOUS SECOND ORDER DIFFERENTIAL EQUATIONS WITH INITIAL CONDITIONS SPECIFIED
J186E	FIXED POINT SQUARE ROOT WITH DOUBLE PRECISION ARGUMENT
J188B,E	K TO THE X
J189B,E	INTERPRET T-SWITCHES
J190E	LOG - BASE E OR 10
J201B,E	PUNCH DECIMAL CARDS

J202A	DUMP MEMORY TO DRUM
J203E	INPUT ADDRESSABLE FLOATING-POINT DATA
207A	AUTOMATIC POLYNOMIAL FUNCTION EVALUATION - FL. PT.
208A	AUTOMATIC COMPUTATION OF THE SOLUTION OF A MATRIX EQUATION AX=B, WHERE A IS N X N AND NON-SINGULAR -- FLOATING POINT
J209E	PUNCH STYLE E
J213A	CHANGE ADDRESS FIELDS
J214E	NATURAL LOGARITHM
J215E	10 PT. GAUSSIAN INTEGRATION -- FLOATING POINT
J216A	JOHNNIAC MUSIC
J217A	STYLE D LOWER LOADER
J218A	STYLE D UPPER LOADER
J219A	STYLE D CHECK SUM CORRECTOR
J223E	PRINT DECIMAL NUMBERS - UNIFORM FORMAT AND SCALING
J224A	SYMBOLIC-RELATIVE ASSEMBLY WITH DRUM
J225E	MOUSE MATRIX TO PRINTER ALPHANUMERIC
J226E	PRINT DECIMAL NUMBERS, DECIMAL FORMAT AND SCALING
J227E	MESSAGE PRINTER
J228A	OCTAL MEMORY DUMP - 4 PER LINE
J230A	OCTAL MEMORY DUMP W/O DRUM
J232A	INSERT BLOCK CHECK SUMS INTO STYLE F DECK
J233E	DECIMAL PRINT ROUTINE
J234A	OCTAL MEMORY DUMP
J235E	PRINT POSITIVE DECIMAL INTEGERS
J238E	PRINT DECIMAL - ONE PER LINE
J241E	ADDRESSABLE INPUT - FLOATING PT
242E	INPUT DATA CARDS - FLOATING PT
243E	PUNCH DATA CARDS - FLOATING PT
244E	PRINT AND TRACE - FLOATING PT
J246E	SINE-COSINE - FLOATING PT
J247E	ARC TANGENT - FLOATING PT
J248E	EXPONENTIAL - FLOATING PT
J249E	LOGARITHM - FLOATING PT
J250E	INDEXING - FLOATING PT SYSTEM
J253E	FIXED POINT SQUARE ROOT 1
J254E	P-TH ROOT, P IN THE RANGE FROM 1 TO 39
J255B,E	CAT READ ROUTINE
J256B,E	CAT READ COMPILER
J257B,E	CAT BCD FROM IMAGE
J258B,E	CAT OUTPUT ROUTINE
J259B,E	CAT ALPHANUMERIC PRINT
J260B,E	CAT DECIMAL PRINT
J261B,E	CAT PUNCH
J262B,E	CAT MOVE IMAGE
J263B,E	CAT SELECTIVELY CLEAR IMAGE
J264A	AUTOMATIC LEAST SQUARES POLYNOMIAL APPROXIMATION -- FLOATING POINT
J265B,E	ADDRESSABLE DECIMAL INPUT
J266B,E	120 COLUMN VARIABLE FORMAT DECIMAL PRINT
J267A	STYLE E LOADER
J268E	ZEROS OF A POLYNOMIAL WITH COMPLEX COEFFICIENTS - FLOATING PT
J269E	COMPLEX MULTIPLY - FLOATING PT
270E	COMPLEX DIVIDE - FLOATING PT
71E	SCALAR PRODUCT OF TWO VECTORS - FLOATING PT

J272E	SUM OF THE COMPONENTS OF A VECTOR - FLOATING PT
J273E	LINEAR COMBINATION OF TWO VECTORS - FLOATING PT
J274E	POLYNOMIAL FUNCTION EVALUATION - FLOATING PT
J275E	ITERATED SYNTHETIC DIVISION - FLOATING PT
J276E	MAXIMUM ELEMENT OF A VECTOR - FLOATING PT
J277E	SOLUTION OF A MATRIX EQUATION AX=B WHERE A IS N*N AND NON-SINGULAR - FLOATING PT
J278E	EVALUATION OF A ROOT OF A POLYNOMIAL WITH COMPLEX COEF-FICIENTS - FLOATING PT
J279F	AUTOMATIC COMPUTATION OF THE ZEROS OF A POLYNOMIAL WITH COM-PLEX COEFFICIENTS - FLOATING PT
J280E	CAT PUNCH WITH ECHO CHECKING
J281E	ORDINARY LEAST SQUARES - FLOATING PT
J282F	ORDINARY LEAST SQUARES - FLOATING PT
J283E	BACK AND FORTH BLOCK TRANSFER
J284E	LEGENDRE POLYNOMIAL EVALUATION - FLOATING PT
J285E	LEGENDRE POLYNOMIAL APPROXIMATION - FLOATING PT
J286E	LEGENDRE POLYNOMIAL COEFFICIENTS - FLOATING PT
J287E	(TRIANGULAR MATRIX)*VECTOR - FLOATING PT
J288F	LEGENDRE POLYNOMIAL APPROXIMATION - FLOATING PT
J289E	ORTHOGONAL POLYNOMIAL LEAST SQUARES - FLOATING PT
J290E	ORTHOGONAL POLYNOMIAL COEFFICIENTS - FLOATING PT
J291E	ORTHOGONAL POLYNOMIAL EVALUATION - FLOATING PT
J292E	TRIANGULAR MATRIX * VECTOR - FLOATING PT
J293F	ORTHOGONAL POLYNOMIAL LEAST SQUARES - FLOATING PT
J294E	ORTHOGONAL POLYNOMIAL LEAST SQUARES, EQUALLY SPACED POINTS
J295E	ORTHOGONAL POLYNOMIAL COEFFICIENTS, EQUALLY SPACED POINTS
J296E	ORTHOGONAL POLYNOMIAL EVALUATION, EQUALLY SPACED POINTS
J297F	ORTHOGONAL POLYNOMIAL LEAST SQUARES, EQUALLY SPACED POINTS
J298E	INTEGRATION BY SIMPSONS RULE WITH PRESCRIBED ERROR - FLOAT-ING PT
J299F	JOHNNIAC FLOATING-POINT INTERPRETIVE SYSTEM
J300E	JOHNNIAC FLOATING-POINT INTERPRETIVE SYSTEM
J301E	SQUARE ROOT - FLOATING PT
J302E	COMPLETE ELLIPTIC INTEGRAL OF THE FIRST KIND - FLOATING PT
J303E	COMPLETE ELLIPTIC INTEGRAL OF THE SECOND KIND - FLOATING PT
J304E	COMPLETE ELLIPTIC INTEGRAL OF THE THIRD KIND - FLOATING PT
J307A	CLEAR THE DRUM
J308A	OCTAL DRUM DUMP
J310E	SOLUTION OF A SYSTEM OF ORDINARY DIFFERENTIAL EQUATIONS - FLOATING PT
J311E	ORTHOGONAL POLYNOMIAL LEAST SQUARES WITH WEIGHTS
J312F	ORTHOGONAL POLYNOMIAL LEAST SQUARES WITH WEIGHTS

JOHNNIAC LIBRARY LISTING SEPT. 1959

ASSEMBLY

J216A	JOHNNIAC MUSIC
J224A	SYMBOLIC-RELATIVE ASSEMBLY WITH DRUM

LOADING

J122A	LOAD STYLE B CARDS - LOW
J123A	LOAD STYLE B CARDS - HIGH
J135A	LOADER FOR STYLE F ABSOLUTE BINARY CARDS
J136A	LOADER FOR STYLE E
J180A	LOADER FOR STYLE F CARDS WITH BLOCK CHECK SUMS
J217A	STYLE D LOWER LOADER
J218A	STYLE D UPPER LOADER
J267A	STYLE E LOADER

INPUT

J112A	LOAD STYLE D CARDS
J129B,E	CARD IMAGE TO CHARACTERS
J130B,E	DECIMAL INPUT VARIABLE FORMAT
J131B,E	DECIMAL INPUT FIXED FORMAT ONE/CARD
J160E	CARD TO MOUSE MATRIX
J161E	CARD TO MOUSE MATRIX WITH INTERMEDIATE COMPUTATION
J162E	MOUSE MATRIX TO H.S.S.
J203E	INPUT ADDRESSABLE FLOATING-POINT DATA
J241E	ADDRESSABLE INPUT - FLOATING PT
J242E	INPUT DATA CARDS - FLOATING PT
J255B,E	CAT READ ROUTINE
J256B,E	CAT READ COMPILER
J257B,E	CAT BCD FROM IMAGE
J262B,E	CAT MOVE IMAGE
J263B,E	CAT SELECTIVELY CLEAR IMAGE
J265B,E	ADDRESSABLE DECIMAL INPUT

PRINT

J223E	PRINT DECIMAL NUMBERS - UNIFORM FORMAT AND SCALING
J225E	MOUSE MATRIX TO PRINTER ALPHANUMERIC
J226E	PRINT DECIMAL NUMBERS, DECIMAL FORMAT AND SCALING
J227E	MESSAGE PRINTER
J233E	DECIMAL PRINT ROUTINE
J235E	PRINT POSITIVE DECIMAL INTEGERS
J238E	PRINT DECIMAL - ONE PER LINE
J244E	PRINT AND TRACE - FLOATING PT
J258B,E	CAT OUTPUT ROUTINE
J259B,E	CAT ALPHANUMERIC PRINT
J260B,E	CAT DECIMAL PRINT
J266B,E	120 COLUMN VARIABLE FORMAT DECIMAL PRINT

PUNCH

J103B,E	PUNCH STYLE B, ABSOLUTE -- 11 PER CARD
J105B,E	PUNCH STYLE D, ABSOLUTE -- 22 PER CARD
J142E	PUNCH STYLE F BINARY CARDS
J148E	PUNCH STYLE F WITH I.D.
J163E	H.S.S. TO MOUSE MATRIX
J164E	MOUSE MATRIX TO PUNCH
J165E	MOUSE MATRIX TO PUNCH WITH INTERMEDIATE COMPUTATION
J166E	MOUSE MATRIX TO PUNCH AND ECHO CHECK
J167E	MOUSE MATRIX TO PUNCH AND ECHO CHECK WITH INTERMEDIATE COMPUTATION
J181E	PUNCH STYLE F CARDS WITH BLOCK CHECK SUMS
J182E	PUNCH STYLE F CARDS WITH BLOCK CHECK SUMS AND I.D.
J201B,E	PUNCH DECIMAL CARDS
J209E	PUNCH STYLE E
J243E	PUNCH DATA CARDS - FLOATING PT
J261B,E	CAT PUNCH
J280E	CAT PUNCH WITH ECHO CHECKING

DEBUGGING

J228A	OCTAL MEMORY DUMP - 4 PER LINE
J230A	OCTAL MEMORY DUMP W/O DRUM
J234A	OCTAL MEMORY DUMP
J308A	OCTAL DRUM DUMP

RED TAPE

J104B,E	BLOCK TRANSFER ROUTINE
J120B,E	BLOCK TRANSFER TO AND FROM DRUM
J132B,E	PRESET OR RESET ROUTINE
J133B,E	PSEUDO B-BOX
J137B,E	MANIPULATION OF ARRAYS
J138B,E	MULTIPLICATION OF ARRAYS
J140A	CLEAR H.S.S. TO ZERO
J149B,E	BLOCK TRANSFER TO AND FROM DRUM WITH TALLY
J153A	CLEAR H.S.S. TO PRESCRIBED WORD
J189B,E	INTERPRET T-SWITCHES
J202A	DUMP MEMORY TO DRUM
J213A	CHANGE ADDRESS FIELDS
J219A	STYLE D CHECK SUM CORRECTOR
J232A	INSERT BLOCK CHECK SUMS INTO STYLE F DECK
J307A	CLEAR THE DRUM

INTERPRETIVE

J250E	INDEXING - FLOATING PT SYSTEM
J299F	JOHNNIAC FLOATING-POINT INTERPRETIVE SYSTEM
J300E	JOHNNIAC FLOATING-POINT INTERPRETIVE SYSTEM

FUNCTIONS

J106B,E	SQUARE ROOT
J116B,E	LOG-BASE TWO
J127B,E	ARCTANGENT X
J151B,E	SIN Y, COS Y, SIN 2Y, COS 2Y
J154B,E	SIN PI Y, COS PI Y
J156B,E	INTEGRAL ROOT X1/P
J186E	FIXED POINT SQUARE ROOT WITH DOUBLE PRECISION ARGUMENT
J188B,E	K TO THE X
J190E	LOG - BASE E OR 10
J207A	AUTOMATIC POLYNOMIAL FUNCTION EVALUATION - FL. PT.
J214E	NATURAL LOGARITHM
J246E	SINE-COSINE - FLOATING PT
J247E	ARC TANGENT - FLOATING PT
J248E	EXPONENTIAL - FLOATING PT
J249E	LOGARITHM - FLOATING PT
J253E	FIXED POINT SQUARE ROOT 1
J254E	P-TH ROOT, P IN THE RANGE FROM 1 TO 39
J269E	COMPLEX MULTIPLY - FLOATING PT
J270E	COMPLEX DIVIDE - FLOATING PT
J274E	POLYNOMIAL FUNCTION EVALUATION - FLOATING PT
J275E	ITERATED SYNTHETIC DIVISION - FLOATING PT
J284E	LEGENDRE POLYNOMIAL EVALUATION - FLOATING PT
J286E	LEGENDRE POLYNOMIAL COEFFICIENTS - FLOATING PT
J290E	ORTHOGONAL POLYNOMIAL COEFFICIENTS - FLOATING PT
J291E	ORTHOGONAL POLYNOMIAL EVALUATION - FLOATING PT
J295E	ORTHOGONAL POLYNOMIAL COEFFICIENTS, EQUALLY SPACED POINTS
J296E	ORTHOGONAL POLYNOMIAL EVALUATION, EQUALLY SPACED POINTS
J301E	SQUARE ROOT - FLOATING PT
J302E	COMPLETE ELLIPTIC INTEGRAL OF THE FIRST KIND - FLOATING PT
J303E	COMPLETE ELLIPTIC INTEGRAL OF THE SECOND KIND - FLOATING PT
J304E	COMPLETE ELLIPTIC INTEGRAL OF THE THIRD KIND - FLOATING PT

MATRICES AND LINEAR EQUATIONS

J124B	EIGENVALUES AND EIGENVECTORS OF A REAL STMMETRIC MATRIX
J208A	AUTOMATIC COMPUTATION OF THE SOLUTION OF A MATRIX EQUATION AX=B, WHERE A IS N X N AND NON-SINGULAR -- FLOATING POINT
J271E	SCALAR PRODUCT OF TWO VECTORS - FLOATING PT
J272E	SUM OF THE COMPONENTS OF A VECTOR - FLOATING PT
J273E	LINEAR COMBINATION OF TWO VECTORS - FLOATING PT
J276E	MAXIMUM ELEMENT OF A VECTOR - FLOATING PT
J277E	SOLUTION OF A MATRIX EQUATION AX=B WHERE A IS N*N AND NON-SINGULAR - FLOATING PT
J283E	BACK AND FORTH BLOCK TRANSFER
J287E	(TRIANGULAR MATRIX)*VECTOR - FLOATING PT
J292E	TRIANGULAR MATRIX * VECTOR - FLOATING PT

APPROXIMATIONS

J178E	LEAST SQUARES POLYNOMIAL APPROXIMATION -- FL. POINT
J264A	AUTOMATIC LEAST SQUARES POLYNOMIAL APPROXIMATION -- FLOATING POINT

J281E	ORDINARY LEAST SQUARES - FLOATING PT
J282F	ORDINARY LEAST SQUARES - FLOATING PT
J285E	LEGENDRE POLYNOMIAL APPROXIMATION - FLOATING PT
J288F	LEGENDRE POLYNOMIAL APPROXIMATION - FLOATING PT
J289E	ORTHOGONAL POLYNOMIAL LEAST SQUARES - FLOATING PT
J293F	ORTHOGONAL POLYNOMIAL LEAST SQUARES - FLOATING PT
J294E	ORTHOGONAL POLYNOMIAL LEAST SQUARES, EQUALLY SPACED POINTS
J297F	ORTHOGONAL POLYNOMIAL LEAST SQUARES, EQUALLY SPACED POINTS
J311E	ORTHOGONAL POLYNOMIAL LEAST SQUARES WITH WEIGHTS
J312F	ORTHOGONAL POLYNOMIAL LEAST SQUARES WITH WEIGHTS

NON-LINEAR EQUATIONS

J159E	INVERSE INTERPOLATION, A REAL ROOT OF F(X)=0
J268E	ZEROS OF A POLYNOMIAL WITH COMPLEX COEFFICIENTS - FLOATING PT
J278E	EVALUATION OF A ROOT OF A POLYNOMIAL WITH COMPLEX COEFFICIENTS - FLOATING PT
J279F	AUTOMATIC COMPUTATION OF THE ZEROS OF A POLYNOMIAL WITH COMPLEX COEFFICIENTS - FLOATING PT

INTEGRATION AND DIFFERENTIAL EQUATIONS

J126B,E	SOLUTION OF A SYSTEM OF ORDINARY DIFFERENTIAL EQUATIONS
J185E	INTEGRATION OF N SIMULTANEOUS SECOND ORDER DIFFERENTIAL EQUATIONS WITH INITIAL CONDITIONS SPECIFIED
J215E	10 PT. GAUSSIAN INTEGRATION -- FLOATING POINT
J298E	INTEGRATION BY SIMPSONS RULE WITH PRESCRIBED ERROR - FLOATING PT
J310E	SOLUTION OF A SYSTEM OF ORDINARY DIFFERENTIAL EQUATIONS - FLOATING PT

J. C. Shaw

J100A - J118A - J134A

JOHNNIAC SYMBOLIC-RELATIVE ASSEMBLY NO. 1

by J. Schwartz

May 6, 1955

INTRODUCTION

This program is designed to handle both symbolic and regional addresses. The program includes provisions for input of decimal data with specified scaling, obviates the necessity of incorporating library subroutine decks at assembly time, and provides a listing of the deck being assembled, the binary deck produced and some of the errors, of a common type, which appeared in the deck. The output is either a relative or an absolute binary deck, of the first location, check sum type, eleven full words on a card. Style B

During assembly time, instructions are given the assembler by means of a four-letter mnemonic code punched in columns 16-19 of a card.

DEFINITIONS OF TERMS

In discussion of this program, certain terms are used frequently. These terms and their meanings as applied to this program are:

(1) SYMBOLIC CODING: Location designations are necessary only when referred to by an address. Output values of locations are assigned by designating an output address to the first card of a sequence of cards. Each card following will have as final location a value one larger than the preceding card. Addresses are assigned output values by using the output value of the location with the same symbol.

(2) REGIONAL (RELATIVE) CODING: Every location and address is assigned a region and sequence number. A base for

every region must be assigned. The output value of addresses and locations is computed by adding the sequence number to the base of the region for the address or location.

(3) SYMBOL: A symbol in this program consists of a two digit number and an alphabetic part, consisting of a letter or its equivalent number.* The numbers range from 00-99, and letters from A through Z (1-26). Examples are 13C, 99Z, 00A, 1303, 9926, 0001.* A blank location field is considered a symbol. ?

(4) REGIONAL (RELATIVE) ADDRESS (LOCATION): In the input deck a regional address consists of a letter or number* for a base, and a three-digit sequence field. Bases are from A-Z (01-26); with base zero reserved for shift orders, etc. Sequences range from 000-999.

(5) COUNTER: This is a cell reserved in memory for assignment of output values for symbolic locations. This is set before a sequence of cards by a control card and increases by one for every symbolic location entering the reader, assigning its current value to the present location.

(6) ALPHABETIC CHARACTERS: In this program, every location and address consists of one alphabetic character and two or three numeric. The alphabetic character is either a number or a letter. A number can be used in place of a letter, the number being considered the same as the letter with this

* See Definition of Alphabetic characters (6) p. 2

sequence in the alphabet. For example, A = 01, Z = 26. The program does not differentiate between A and 01, B and 02, etc. <u>This definition of alphabetic characters applies only to locations and addresses</u>.

(7) INPUT CARDS: Cards which are keypunched and are being assembled.

(8) OUTPUT CARDS: Binary cards to be punched by assembly. Both absolute and relative binary cards are considered, although the card form for both is the same, absolute being distinguished from relative only by having blanks in the binary region fields.

Assembly instructions and their definitions, to be explained in full later, are:

1) Instructions pertaining to symbolic coding:
   - a) SGEN: Symbol generation
   - b) SAME: Reset counter to same value it had at this symbol.
   - c) SETC: Set counter.
   - d) SKIP: Skip counter.

2) Instructions pertaining to relative coding.
   - a) CTOR: Counter value assigned to origin.
   - b) RDOR: Read origin from card.

3) Instructions pertaining to symbolic and relative coding.
   - a) DDAT: Decimal data.
   - b) ODAT: Octal data.

c) LIBR: Library subroutine origin assignment

d) LAST: End of deck to be assembled.

4) Instructions pertaining to deck for loading.

a) TRFR: Punch transfer card.

b) LDOR: Punch origin cards.

GENERAL CHARACTERISTICS OF INPUT CARDS

Address and location fields are five characters in length.

In symbolic coding, the third and fourth positions are the alphabetic character position (i.e., if a one-digit number is used, the fourth position contains this number, a two-digit number uses the third and fourth positions, a letter uses the third position only. There is never a punch in the fifth position of the address or location field in symbolic coding. There are 2600 symbols, with 00-99 the range in positions 1 and 2, and 01-26 or (A-Z) in the third and fourth positions.

In regional coding, the first and second positions are the alphabetic character position. If numbers are used, the first and second positions are used. Letters use the second position only. Sequence numbers are in the third, fourth and fifth positions of the field. A regional address always has a punch in the fifth position. There are again 26 alphabetic characters, and the sequence ranges from 000-999.

Aside from the fifth position, zero punches or blanks are treated in the same manner.

Blanks in location positions are treated as symbols. ?.
The counter is increased by one and the full word with this location is assigned this counter value as an output location.

When an integer such as a shift order is used in an address field, the units position <u>must</u> be punched. No field of this kind can have over 999 as its value.

Examples of address and location fields.

	1	2	3	4	5
Equivalent	0	0	A		
Symbols	0	0	0	1	
				1	
	0		A		
Equivalent	9	9	Z		
Symbols	9	9	2	6	
		A	0	0	0
Equivalent	0	1	0	0	0
Regional		A			0
Addresses		1			0
Equivalent		Z	9	9	9
Regional	2	6	9	9	9
Addresses					

Operations can be alphabetic or octal numeric. Zeros or blanks can be used interchangeably. Letters cannot be used as direct replacements for numbers. For instance, A-- means ADD or 024, not 100.

Assembly instructions are always letters. However, to prevent errors due to keypunching, when the letter "O" is in an instruction, it is permissible to use either the letter "O" punch or a zero punch.

ASSIGNMENT OF OUTPUT ADDRESSES AND LOCATIONS DURING ASSEMBLY

This is a two pass assembly program both for symbolic and relative coding. On the first pass, the program reads and converts cards, and decides whether locations and addresses are symbolic or relative.

1) When symbolic locations are used, the counter must be set preceding the entrance of any symbolic locations into the reader. After the counter is set, it will continue to increase by one for every symbolic location, unless it is reset. The counter does not increase by one when a relative location appears. After a location symbol is assigned an output location, this output location is put in a table containing the output value of all symbols. The full word associated with this symbol is then stored on the drum. When all cards have been read, the words on the drum are recalled, and symbolic addresses are assigned the output value which is found in the table for that symbol.

An arbitrary sequencing of symbols is allowed. For example, 99Z can follow 02B, etc. Instructions used to control the counter are as listed on page 3 No. 1, and are used as follows:

a) SGEN: If it is desired to designate groups of words with symbolic locations, i.e. data, erasable, the SGEN instruction may be used to avoid punching a card with every symbol on it for assigning each symbol an output value.

The card form for this instruction is as follows:

Cols.	16-19	20-24	25-27	28-32	36-40
	SGEN	1st Symb.	ΔS	N	ΔC

1st SYMB. is first symbol desired to be generated.

ΔS is change in each symbol desired.

N is number of symbols to be generated.

ΔC is the change in the counter for each symbol.

Example: To generate a group of erasable cells 1.0000 - 1.0009 in output, using symbols 13A - 13J in input, the following procedure is used:

Set the counter to 1.0000 (A.0000). Then read in the following card:

Cols.	16-19	20-24	25-27	28-32	36-40
	SGEN	13A (1301)	001	0010	00001

Note that the change in the symbol is added to the letter position and not the numeric portion. In the example above, 13A would receive the counter value 1.0000, 13B = 1.0001, 13J = 1.0009. If the number of symbols is large enough to pass the letter value Z or 26, the next symbol generated will contain an increase in the numeric part of the symbol by the increment 1, and the letter portion will be A (01). For example, if N had equalled 27 in the example above, the 27th symbol generated would be 14A, which would be assigned the value 1.0026 by the counters. The counter is left at the last position generated plus one after the symbols are generated. Following this order, an address can refer to any of the generated symbols and be assigned an output value.

b) SAME: If a program is such that different blocks of words will use the same portion of memory at various times, the SAME instruction is used to reset the counter to the same value it was when the previous block was assigned output values.

The card form is as follows:

Cols.	16-19	20-24
	SAME	Symbol
Examples:	SAME	15B

This Symbol is the same as the one used previously. The counter is reset to the value it had at this symbol, and the next card will be assigned this counter value as a location.

c) SET C: This instruction sets counter for regional output (Relative binary) or absolute output (absolute binary). There are 26 regions possible; (A-Z) (1-26), and/or a sequence of 0000-$(4095)_{10}$.

The card form is as follows:

Cols.	16-19	20-21	29-32
	SETC	Region	Sequence
Example:	SETC	A	3139
		01	3139

d) SKIP: This instruction permits an increase in value of the counter a specified number of places. The skip can be in a region or sequence.

Card form is as follows:

```
Cols.          1 - 4 ?      1 - 5 ?
                SKIP          N
```

N = numbers of places to be skipped, either in region or sequence. Greatest skip allowed in sequence is 999.

Example:

```
1) Cols. 16-19     20-24
         SKIP      01.000
         Counter will change to next region.

2) Cols. 16-19     20-24
         SKIP      00.190
         Counter will increase by 190 in the sequence.
```

2) When regional coding is used, the bases may be assigned at any time during the first pass. The regional addresses and locations are stored on the drum until the second pass, then the bases are added to the sequences. Regional input bases may be assigned the same or different regional output bases (relative binary), or absolute bases (absolute binary).

Instructions used to assign bases for origins are:

a) CTOR: This instruction assigns a particular region a base equal to the counter value at this time. As an $\boxed{\text{option,}}$ it also allows the skipping of the counter N places or n regions <u>after</u> assigning the base, N having the range 000-999.

Card form is as follows:

Cols.	16-20	20-21	28-29	30-32
	CTOR	REG.	n	N

n = Change in counter region ← can it be negative?
N = Change in counter sequence.

Example: Assume counter at 4.1234

Cols.	16-19	20-21	28-29	30-32
	CTOR	F (06)	01	105

After reading this card, input region F (06) will be assigned the output base and sequence 4.1234, and the counter will move to 5.1339.

b) RDOR: This is a base card which assigns a particular region an output base (relative or absolute).

Card form is:

Cols.	16-19	20-21	26-27	29-32
	RDOR	INPUT REGION	OUTPUT REGION	OUTPUT SEQUENCE

Output sequence has the range 0000-4095.

Example:	Cols. 16-19	20-21	26-27	29-32
	RDOR	01	D	3123
		A	. 04	

For this case, any input address or location with region A (01) will be assigned the base 4.3123 for the binary cards.

CARD FORM FOR FULL WORD ORDERS

For both regional and symbolic coding, ordinary JOHNNIAC full words (consisting of location, left and right operation, left and right address), will be designated by a blank assembly instruction. Operations can be octal numeric or alphabetic (see list of JOHNNIAC operations), both cases using three character fields. In the numeric operations, zero punches or blanks are treated similarly. In alphabetic operations, zero punches and the letter "O" punches are treated similarly.

Card form for full words is:

Cols.	16-19	20-24	25-27	28-32	33-35	36-40
		LOC.	L.OP.	L.ADD.	R. OP.	R.ADD.

Examples: (Blanks represented by -.)

Cols.	16-19	20-24	25-27	28-32	33-35	36-40
		13.A--	020	17C--	S--	-B.000
		-----	SHL	01.01-	120	----0
		02.26-	RA-	02.Z--	TRL	99.B--
		-B.000	---	----1	TRL	----0

CARD FORMS FOR DATA TO BE ASSEMBLED

This assembler recognizes two types of data, decimal and octal. A card with decimal data has DDAT punched in the assembly instruction field. The data is entered in the following manner. The location of the word is in columns 20-24, the sign of the data in column 25. The full word, from columns 26 to 36 contains the data. Columns 37 and 38 contain the number of places from the sign position to the decimal point of the word, designated as p. Columns 39 and 40 contain the number of places from the 2^0 position, excluding 2^0, of memory to the binal point of the full word. Columns 39 and 40 are designated as q. ? limits?

Card form is as follows:

Cols.	16-19	20-24	25	26-36	37-38	39-40
	DDAT	LOC.	$\pm$	DATA	p	q

Example:	16-19	20-24	25	26-36	37-38	39-40
	DDAT	13Z--	+	1374206989	03	11
	DDAT	02.100	-	11500000000	06	20

Octal data can be entered through the use of an ODAT in the assembly instruction field. The data consists of a 14 digit word with a 0 or 1 in the first position and any octal number in each of the next 13. The program will put the binary image of this word in the designated location.

Card form is:

Cols.	16-19	20-24	25-38
	ODAT	LOC.	FULL WORD

Examples:

Cols.	16-19	20-24	25-38
	ODAT	-J.400	17770000111000
	ODAT	15.02--	01110101777130

INCORPORATION OF LIBRARY SUBROUTINES

JOHNNIAC library routines will be in relative binary and will consist of region 3 as the instruction part, and other regions, usually 1 and 2 as erasable and constants. The only reference to a library subroutine in the main program is in the calling sequence and this reference will be to cell 3.000. ?

This assembly takes advantage of the above facts by having the programmer refer in his program to a symbol representing cell 3.000 for each library routine. The program then punches a card which contains the library program's identification and the current counter value which acts as the base for region 3 in the library program, and other information if requested. During loading time, this card is placed in front of its respective library routine, and proper bases are assigned the ?

Describe loader.

different library programs.

This library feature also allows the sharing of regions 1 and 2, or other regions, among the library programs, and with the main program if desired. This is done by allowing different bases for regions 1 and 2 to be assigned with each new library routine or not allowing different bases. If no origins are assigned for regions 1 and 2 for all library routines, all routines will share regions 1 and 2 with the main program.

If it is desired to load all library routines in a block of memory, this library incorporation permits skipping the counter the necessary number of cells for each program. This is done by punching the number of words allotted to region 3 in a particular program.

The card form for library incorporation is as follows:

Cols.	16-19	20-24	25-29	33-35	37-40
	LIBR	SYMBOL	REG.NOS.	N	RAND I.D.

SYMBOL = symbol used as reference to program in calling sequence.

REG. NOS. = the regions of the library program for which new bases are desired. 1, 2, 4, 5, 6 and 7 are possible, one per column.

N = number of words in region 3.

RAND I.D. = the numeric part of the <u>RAND Library Identification</u>.

An example of this type of library incorporation is:

Given two library routines with RAND identification <u>RJ0020</u> and <u>RJ0027</u>:<u>RJ0020</u> has 48 words in region 3, 4 words in region 2, and 35 words in region 1; <u>RJ0027</u> has 35 words ✓

in region 3, 4 words in region 2, and 40 words in region 1. The current counter number is 6.1234. There is no input region 1 and 2 in the main program.

One method of incorporating these two programs is the following:

1) In the main program have the following calling sequences:

α_1 RA - α_1 TRL 13S--

α_2 RA - α_2 TRL 25Z--

2) Have following LIBR cards punched:

Cols.	16-19	20-24	25-29	33-35	~~37~~ 38-40
a)	LIBR	13S--	-----	048	ø020
b)	LIBR	25Z--	-----	035	ø027

3) Given this information, the assembly program will punch two binary cards. In the first ~~6~~ 5 columns of one will appear ~~RJ0020~~ J020B. An $(077)_8$ will appear in the left operation position of the right-hand side of the 9's row on both cards. The number "3" and the base 6.1234 will appear in the left half of the 8's row of RJ0020. The other card will contain ~~RJ0027~~ J027B as an identification and the number "3" with 6.1282 in the left half of the 8's row.

4) At load time these cards should be placed at the front of their respective library routines. A relative binary origin card should contain origins for regions 1 and 2, even though there are no regions 1 and 2 in the main program. Also an origin card for region 6 must be loaded with the program.

5) Then the two library programs will be loaded, with regions 1 and 2 being shared in memory, and region 3 for each program going into the cells specified by the base for region 6 and the respective sequence numbers.

6) The binary cards containing the calling sequences for these programs will appear with the transfers to 6.1234 and 6.1282 respectively.

If it is desired to put regions 1 and 2 of the library programs into regions other than 1 and 2 in the main program, the following procedure can be used.

1) Preceding the LIBR cards for these programs read in RDOR or CTOR cards containing bases for regions 1 and 2, for example Region 1 has base 7.0000, Region 2 has base 7.0050.

2) Punch LIBR cards, this time of the following form:

Cols.	16-19	20-24	25-29	33-35	38 ~~37~~-40
	LIBR	135--	12---	048	~~0~~020
	LIBR	25Z--	12---	035	~~0~~027

3) The results of these operations will be the same as previously, except that now the library header cards punched will contain a "1" and 7.0000 in the 7's row, and a "2" and 7.0050 in the 6 row of the left half of the card. ?

4) Now at load time, a relative binary origin card for region 7 is necessary as well as the one for region 6 as previously. No origin for regions 1 and 2 is now necessary.

5) Once the input bases for regions 1 and 2 have been made equal to 7.0000 and 7.0050, they will remain this way for the remainder of the assembly.

INSTRUCTIONS USED FOR PREPARATION OF LOAD DECK

These assembly instructions are used to make possible the immediate loading of a program after assembly, by punching relative binary origin cards at the front of the output deck and a transfer control card at the end.

LDOR: This is a card which contains the absolute base of an output region, which will be used in loading the final program. A series of these cards will produce one binary card containing the designated regions and their respective bases. This card is punched immediately preceding the main deck and has an 020 in binary in the left operation of the right half of the 9's row.

The LDOR card has the following form:

```
          Cols. 16-19    20-21     29-32
                LDOR     REGION  ABSOLUTE BASE
       Example: LDOR      01        1299
                         [illegible]A
                          [illegible]
```

[illegible]

TRFR: The TRFR card is a card containing the desired address to transfer control to after loading the assembled deck. This produces a card recognizable by the loader, which will transfer control to the left half-word of the designated address. This address can be symbolic or regional on the TRFR card, and regional or absolute on the binary transfer card.

✓

Card form:

```
          Cols.    16-19     20-24
                   TRFR    SYMBOL OR REGIONAL ADDRESS
```

MARKER FOR END OF DECK TO BE ASSEMBLED

Because the JOHNNIAC has no end of file return, it is necessary to have a card mark the end of the input cards. For this program, the marker is a card with a LAST punched in the assembly instruction field.. This card also can be used to form the identification field for the output cards. Columns 20-27 of the LAST card will be reproduced in columns 1-8 of all the output cards except the origin, library, and transfer cards.

Card form of LAST card is:

```
Cols.     16-19    20-27
          LAST      I.D.
```

INSERTIONS AND DELETIONS

The method of making insertions and deletions depends on the type of input and output cards used.

Insertions and deletions can be made in a symbolic deck simply by adding or deleting cards and reassembling, thereby getting a new binary deck and an up-to-date listing.

There is no provision for making insertions or deletions in a regional input deck. Relative binary output cards can be inserted and deleted by means of the program for that purpose. This program will produce a new relative binary deck and also an up-to-date octal listing.

No insertions or deletions can be made in an absolute binary deck.

Binary corrections can be made with either absolute or relative binary cards.

LISTING

The listing is composed of three parts. First is the listing of the library header cards that are punched. This will consist of every region and the base assigned for each library program. The heading will be the RAND Identification for each library routine.

The second part of the listing contains the output program deck. This consists of two lines for every full word on a binary card. There are four spaces at the end of the listing of each binary card. The listing for each full word contains on the first line the location and two addresses, or data exactly as they appeared in the input deck. The second line contains the octal image of the location and full word as they appear in the binary deck.

At present, some fractional data will be listed -1 in the least position. In some cases, for example, .4999 will be listed for .5000. why?

The third part of the listing contains a list of the errors found during assembly.

ERRORS AND ERROR DETECTION

Wherever possible, this assembly, upon finding an error will make a note of it, replace the error, and continue with the assembly. These types of errors will then be listed as the third part of the listing. Each type of error will be listed under a numerical heading which represents this error. The contents of the list will be information which allows one to locate the error and describes what correction was made, where applicable.

Errors that are listed, and their headings are:

a) Two equal location symbols: 7777

This error occurs when the same symbol is used in two different location fields. When this error is detected any address using this symbol will have a blank on the binary card where this address appears. Each line under this error will contain the symbol which is used twice, the counter value assigned this symbol previously and the current counter value.

b) No assigned base for input region: 1234

This error will be detected by the assembly if no RDOR or CTOR card was used and there are regional addresses or locations used on input cards. When this error is found for one region, the assembler assigns the region base $(6000)_8$. For each succeeding unassigned region, this base assignment is increased by $(400)_8$; i.e., the second region will be assigned base $(6400)_8$, third $(7000)_8$, etc.

Better to assign it's own region?

Each line under the error will contain the region which was unassigned and the base assigned by the assembly program.

c) Illegal operation: 3443

This error is detected when an alphabetic operation is misspelled, so that there is no such operation. Wrong numerical operations are not detected. 000 or NO-OP will replace the wrong operation on the output binary deck.

~~Listed under the heading will be the symbol.~~

~~Each line under the heading will contain the output location in octal of the full word where the error is located.~~

d) Improper q for DDAT: 1717

If, on a decimal data card, the specification of the binal point is incompatible with the number of decimal places, an error will be detected. In this case, a blank will be left on the binary output card in place of the full word where the error occurs.

Listed under the heading will be the output location, in octal, of the wrong data.

e) Heading greater than 9926: 2700 ?

If the third and fourth characters of a symbol are greater than or equal to 27 (greater than Z), an error will be recognized. A blank will replace the symbol in the output cards, and also in the symbol table, so that if this symbol was used as a location, the addresses with this symbol will have no location to assign them a counter value.

Listed under the heading will be the symbol in error and the counter value in octal where it occurs.

f) No counter value for address: 6666

This error occurs when there has been no symbol in a location field for a corresponding symbol in the address field.

Listed under the heading will be the symbol. ~~and the output location of the full word where it occurs.~~

Certain errors, such as errors in assembly instructions, are not correctable by the assembly program and will cause stops in the program. In most of these cases, it will be necessary to reassemble.

When a counter region is overrun, it is possible to continue with the assembly by taking the last card read out of the reader and placing it in front of the remaining cards in the hopper and pressing the "GO" button. ?

JOHNNIAC SYMBOLIC-RELATIVE ASSEMBLY

JOB NO. .... DATE .... PAGE ...... OF ......

PROGRAM I D .... (COL 1 - 15) PROGRAMMER ....

INST	LOC	OP	ADDR	OP	ADDR	COMMENTS
16 17 18 19	20 21 22 23 24	25 26 27	28 29 30 31 32	33 34 35	36 37 38 39 40	41 — 80
LIBR	Symbol	Reg Nos →		N	1D	
LIBR	13S	12		048	020	
LIBR	25Z	12		035	027	
LDOR	Region		Absolute Base			
LDOR	A		1299			
TRFR	Loc'n					
TRFR	10A					
TRFR	C000					
AST	← ID →					
LAST	TEST	PRG				
FDAT	Loc'n	± EXP	MANTISSA →			← Decimal listing will
FDAT	K 15	-51	10000	000	00	not be correct (recognizable).
ERRORS:						
3443	Illegal operation.					
1717	Improper q for DDAT.					
2700	Improper symbol.					
6666	Undefined symbol.					

JOHNNIAC SYMBOLIC-RELATIVE ASSEMBLY

JOB NO. ____________ DATE ____________ PAGE ____ OF ____

PROGRAM I D ____ J 100 ____ (COL 1 - 15) PROGRAMMER ____________

INST	LOC	OP	ADDR	OP	ADDR	COMMENTS
16 17 18 19	20 21 22 23 24	25 26 27	28 29 30 31 32	33 34 35	36 37 38 39 40	41 — 80
SGEN	1st Symbol	ΔS	N		ΔC	
SGEN	13A	001	10		1	
SAME	SYMBOL					
SAME	15B					
SETC	Region		Sequence			
SETC	A		3139			
SKIP	N					
KIP	01000					
SKIP	00190					
CTOR	Region		n N			
CTOR	F		01105			
RDOR	Input Region	Output Region	Output Sequence			
RDOR	A	D	3123			
DDAT	Loc'n	± ←	DATA		→ p q	
DDAT	13Z	+13	74206	989	10311	
DDAT	02100	−11	50000	000	00620	
ODAT	Loc'n	←	FULL WORD		→	
DAT	J400	177	70000	111	000	
DAT	1502	011	10101	777	130	

J100A , J118A

Additions and Corrections to Write-up on JOHNNIAC Symbolic Relative Assembly No. 1

(1) New Title

In the JOHNNIAC library, the program as described in this write-up (making use of the drum) is titled J100A.

(2) J118A (Assembly without drum):

J100A has been modified so that the same assembly can be done without use of the drum. This program is called J118A, and is identical with J100A in every other respect.

(3) Input of floating data

The instruction FDAT can be used with both J100A and J118A to make the assembling of floating data possible.

The form for this type card is:

Cols.	16 - 19	20 - 24	25	26 - 27	28 - 36
	FDAT	LOC'N	SIGN	EXP.	MANTISSA

The decimal listing for this type input will not be correct. The octal listing, however, will be exactly as on the binary card.

(4) Corrections

(a) Page 12, under examples of ODAT

The examples given should contain 14 octal digits, zero or one in the first positions and 13 other digits.

Cols.	16 - 19	20 - 24	25 - 38
	ODAT	-J400	17770000111003
	ODAT	1502	01110101777135

(b) Incorporation of Library Subroutines, pp. 12-15.

All references to subroutines should be changed from RJXXXX to JXXXB, so that the first five columns of the I.D. field are used for subroutine identification. Thus, all references to RAND I.D. in the examples should use only columns 38-40.

(c) Page 15, the first line of the example should read:

Cols.	16 - 19	20-24	25 - 29	33 - 35	38 - 40
	LIBR	13S	12---	048	020

(d) Page 16, example, middle of page:

The A under the REGION field should be under column 21.

(e) Page 17, first paragraph, line 6, should read:

Columns 20-27 of the LAST card will be reproduced . . .

(f) Page 20, paragraph (f):

The second sentence should read, "Listed under the heading will be the symbol."

The output location will not be listed.

6-30-55

J101A LOAD STYLE B CARDS OBSOLETE
REPLACED BY J122A 8-9-55

(1) Description

J101 is a program to load the absolute or relative check sum binary cards produced by J100 (style B cards) and incorporate the library, origin, and transfer cards punched by J100.

(2) Binary card form to be loaded: Style B

(3) Origin Cards

The origin card for the regions on relative binary cards is recognized by a punch in column 43 of the 9's row and the remainder of the 9's row blank. There is room to place 44 origins (including repetitions) on this card by designating columns 1 - 7, 20 - 28, 41 - 47, 60 - 68 as the region positions, and columns 8 -19, 29 - 40, 48 - 59, 69 - 80 as their respective bases in rows 8 - 12 of the origin card. The origin card can be entered at any point in the program deck, but must <u>precede</u> cards with references to the regions on the origin card. More than one origin card is allowed. The origins are loaded from left to right beginning with the 8's row, so that a change in origin can be made by punching the region and new base in a higher row of the card or to the right on the same row. Twenty-six regions are allowed $(1 - 26)_{10}$, in addition to region zero (absolute binary).

(4) Incorporation of Sub-routines

Library heading cards, which contain the origin for region 3 for the particular library routine and any other necessary region are recognized by an $(077)_8$ in columns 41-47 of the 9's row and the remainder of the 9's row blank, except possibly for some identification in the first 8 columns.

The regions specified are in columns 17 - 19 beginning with row 8, and the bases (relative or absolute) in columns 23 - 40 in the respective rows. The sequence of the bases is in columns 29 - 40, and region for the bases in columns 23 - 28. The region specified in columns 17 - 19 is the actual region in which the library program is coded (0 - 7), while the regions specified in columns 23 - 28 are the regions used in the program to be loaded other than the library programs $(0 - 26)_{10}$. The load program uses two tables of origins. One consists of 27 places $(0 - 26)_{10}$ for the main body of the program, for which each origin is assigned a base by origin cards (see section 3). The second table consists of 8 places $(0 - 7)_{10}$, for the origins of the particular library routines. Thus, the origins for the library routines can contain as bases a region which is one of the 26 program regions. In this case, the library address will be increased by the sequence in the library base plus the base of the region specified in the library origin.

The library header card should be placed in front of its respective library program so that each library program will be placed in its proper position in memory. Library subroutines are in style B card form. The last card in every subroutine is recognized by a punch in column 10 of the 9's row, which is included in the check sum of this card.

(5) Binary Corrections

To set up the load program to accept binary corrections, a card with a punch in column ~~10~~ 80 in the 9's row, and the remainder blank, is necessary. The cards following may or may not be binary corrections. If they are not (i.e., if regular check sums cards are used again) any further binary corrections must again be preceded by a card with a 9 punch in column ~~10~~ 80.

The binary corrector cards are all recognized by a 9 punch in column 10. The location of the correction is in columns 11 - 28 of any of the 12 rows from 9 - 12, columns 11 - 16 for the region of the correction, and columns 17 - 28 for the sequence. The full word of the correction is in the same form as the style B check sum cards (i.e., columns 29 - 40 for the two address regions, and columns 41 - 80 for the two operations and sequences.

(6) Transfer Control Card

The transfer card is recognized by the absence of any punches in columns 41 - 80. The transfer or halt transfer instruction is placed in columns 1 - 7 of the 9's row and the transfer address in columns (23 - 40), (23 - 28) for the region, and (29 - 40) for the sequence.

(7) Repeated Use of J101

By retaining cells $(0 - 335)_8$ in memory after a program has been loaded, more cards can be loaded by J101 by executing a transfer to address $(60)_8$ and having the cards to be loaded at the read position of the primary feed of the collator.

(8) Program Stops

(a) $(172)_8$: Check sum stop

By pressing "GO" button, the program will continue, but the card following the bad check sum card will not be loaded.

(b) $(127)_8$: Blank 9's row

Press "GO" to restart.

(9) Program length: $(336)_8$ full words

Memory occupied $(0\text{-}335)_8$

Program available as J101A.

Available in style B absolute binary cards.

Program Storage: 2960_{10} - 4095_{10} (5620_8 - 7777_8).

LIST OF FREQUENTLY USED LOCATIONS

Dec.	Octal	Contents
2960	5620	First Location of Interpreter
2963	5623	020 [I CTR.] 050 5623
2980	5644	I Register
4041	7711	Error Halt Location
3040	5740	Trap Register - BRKPT
3041	5741	" " - All Orders
3042	5742	" " - Transfer
3656	7110	AMQ Exponent
3658	7112	AMQ Mantissa
3708	7174	072 [q_1] 025 7016 - Square Root Shift*
3733	7225	022 7122 076 [q_2] - Series Shift*
3999	7637	Series Test Number*
3064	5770	Index Register A
3056	5760	" " B
3052	5754	" " C
3050	5752	" " D
3049	5751	" " E
3048	5750	" " F
3597	7015	Integer - Zero
3598	7016	" - One
⋮	⋮	⋮ ⋮ ⋮
3606	7026	Integer - Nine
3615	7037	" - Fifty
3634	7062	" - Ten exp zero
3635	7063	" - " " One
⋮	⋮	⋮ - ⋮ ⋮ ⋮
3643	7073	Integer - Ten exp. Nine

* Let $\Delta_n = f_{(n+1)} - f_n$, where f_n is the nth approximation to the function f. The test for convergence of f_{n+1} to f is then made to depend upon the sign of $|\Delta_n| \cdot 2^{-q} - Z$. Z is always 1 for the SQR operation. For the ART, EXP, and LOG operations Z is the series test number. The series test number is ordinarily equal to 1 for these three opera-

J102B,E

LIST OF FLOATING-POINT OPERATIONS

The Operation Codes refer to the two least significant octal digits of the representation of the JOHNNIAC Operation Codes. The most significant octal digit of the latter is always used for control purposes in the Floating Point Interpretive System.

00	---	(No Operation)	04	PCH	(Punch Cards)
01	TNL		05	TNR	
02	TPL		06	TPR	
03	TL	(Transfer Left)	07	TR	(Transfer Right)
10	EXL	(Exit Interpreter Left)	14	EXR	(Exit Interpreter Right)
11	TZL	(Transfer on Zero Left)	15	TZR	(Transfer on Zero Right)
12			16		
13	INP	(Read Cards)	17	PNT	(Print)
20	RA		24	A	
21	RS		25	S	
22	RAV		26	AV	
23	RSV		27	SV	
30			34		
31			35		
32	M		36		
33	MN		37		

J102 B,E

40	DS		44		
41	DNS		45		
42			46		
43			47		
50	ST		54	ART	
51	SQR		55	EXP	(e^x)
52	SIN		56	LOG	(Natural Log.)
53	COS		57		
60			64		
61			65		
62			66		
63			67		
70	RAX	(Reset Add Index)	74	AX	(Add Index)
71	TNX	(Transfer on Negative Index)	75		
72	TPX	(Transfer on Positive Index)	76		
73	ENX	(Enter X Mode)	77		

40 COL. A EX PRINTER

JOHNNIAC FLOATING POINT SYSTEM

X - DECIMAL DIGIT
Y - OCTAL DIGIT

S { BLANK - PLUS
S { ⊕ - MINUS

```
 1 2 3 4 5 6 7 8 9 10 11 12 13 14 15 16 17 18 19 20 21 22 23 24 25 26 27 28 29 30 31 32 33 34 35 36 37 38 39 40   FORMAT FOR:

   L O C       L O P   L   A D                    C (AMQ)                      C (MEM)
 Y Y Y Y       Y Y Y   Y Y Y Y   S X X X X X X X X X X X   S X X X X X X X X X X X    TRACING LEFT
                                                                                      ORDER

   L O C       R O P   R   A D                    C (AMQ)                      C (MEM)
 Y Y Y Y 2     Y Y Y   Y Y Y Y   S X X X X X X X X X X X   S X X X X X X X X X X X    TRACING RIGHT
                                                                                      ORDER
                                                                                      NON-INDEXING
   L O C       T A G   R   A D                  X( )                          ΔX( )
 Y Y Y Y 2     Y Y Y   Y Y Y Y                X X X X                        X X X X    TRACING RIGHT
                                                                                      ORDER INDEXING

                     ONE EXTRA LINE IS SPACED PRIOR TO
                     PRINTING INFORMATION FOR THE LEFT
                     ORDER.

                A                            B                            C
             C (α)                      C (α + 1)                  C ( α + 2)
   S X X X X X X X X X X X      S X X X X X X X X X X X      S X X X X X X X X X X X   PRINTING WITH
                                                                                      TAG OR
                                                                                      ROP = 1A 1B 1C

                     SENSE SWITCH SETTINGS FOR TRACING

       T2 Off     T3 Off            NO TRACING
       T2 On      T3 Off            BREAKPOINT
       T2 Off     T3 On             TRANSFER
       T2 On      T3 On             FLOATING POINT (All Orders)
```

Left Margin of Paper

Column Marker
Manually Set

Column Marker
Manually Set

Column Marker
Manually Set

Right Margin of Paper

J102B, E

J103B PUNCH BINARY PROGRAM - 11 PER CARD
J103E

Eleven words per card are punched in style B absolute cards, beginning with first location specified and ending with last location specified.

Region 1 (erasable) - 7 words

Region 3 - 47 words

Calling sequence:

```
        α    020    α    010    XXX (arbitrary symbol)
    α + 1    000 first loc.    000 last loc.
    α + 2    Control returns here
```

Program available as J103B,E

J104B BLOCK TRANSFER ROUTINE
J104E

Calling sequence:

```
        α    RA      α     TRL    XXX   (Arbitrary symbol)
    α + 1   [                        ]  arbitrary instruction pair
    α + 2    000    ΔL     000    ΔR    increments to addresses in
                                                    α + 1
    α + 3    000    0000   000     n    number of executions
    α + 4    Control returns here
```

J104 executes the instruction pair in α + 1 n times. After each execution the addresses are modified by adding the increments in α + 2 then storing both addresses, therefore, the addresses can be decreased by adding complements. The call sequence itself remains unmodified. J104 can be used to read, print or punch images and to clear equally spaced locations in addition to its more frequent use as a block transfer routine within high-speed storage.

Examples:

```
            α   RA      α     TRL     XXX  (Arbitrary symbol)
        α + 1   RA    1000    ST     2000
        α + 2            2           4095
        α + 3                          20
```

This copies the contents of 1000(2)1038 into 2000(-1)1981.

```
        α - 1   SEL     0
        α       RA      α     TRL     XXX  (Arbitrary symbol)
        α + 1   C     1000    ST     1001
        α + 2            2              2
        α + 3                          12
```

This reads a card image from the primary feed into 1000(1)1023.

Program Length: 14 words - all in region 3

Program Available: J104B,E

No program stops.

Timing: Approximately .8 ms before first execution of the pair.
Approximately .5 ms between executions of the pair.

J105B PUNCH BINARY PROGRAM - 22 PER CARD
J105E

Twenty-two full words per card are punched in style D absolute cards, from first location to last location specified.

Region 3 - 56 words ?

Region 1 (erasable) - 8 words

Calling sequence:

```
        α     020   α    010   XXX (arbitrary symbol)
    α + 1     000 first loc.   000 last loc.
    α + 2     Control returns here
```

Program available as J105B,E

J106B SQUARE ROOT
J106E

Take the square root of the fraction in the MQ; leave the answer in the Accumulator.

Calling Sequence:

```
α     RA    α    TRL    XXX (Arbitrary Symbol)
α+1   Control returns here.
```

Program length:

38 words of region 3
5 words of region 1

Program Stop:

3.0036	Argument < 0 Hit start to set answer = 0.

Timing:

4.093 ms $\leq$ total time ≤ 21.830 ms
Approximate expected time = 14 ms

Precision: error $\leq 2^{-39}$

Program available as J106B,E

J107B K^x
J107E

$K = 2$, e, or 10. X in MQ.

Calling Sequence:

α	020	α	010	XXX	(Arbitrary symbol)
$\alpha+1$	$q_1 \cdot 2^{-18}$	+	$k \cdot 2^{-39}$		
$\alpha+2$	Control returns here.				

$k = 0$	2^x
1	e^x
2	10^x

leaves K^x in MQ normalized.

leaves $q_2 \cdot 2^{-18}$ in Accumulator where q_2 is scale factor for K^x.

q_1 = location of binary point in X.

$-90 \leq q_1 \leq 37$

Approx. Maximum time: 25 ms

Accuracy: Approximately 9 S.D.

Program Stop: 3.009 q_1 exceeds 37

Program Length: 56 words in region 3
3 words in region 1

Program available as J107B,E

6-30-55

J108A LOAD STYLE B CARDS

OBSOLETE
REPLACED BY J123A 8-9-55

J108 is identical with J101 except for the following features:

Sections 1 - 6 of the J101 write-up are the same for J108.

(7) Repeated Use of J108

By retaining cells $(7446 - 7777)_8$ in memory after a program has been loaded, more cards can be loaded by J108 by executing a transfer to address $(7511)_8$ and having the cards to be loaded at the read position of the primary feed of the collator.

(8) Program Stop

(a) 7623: Check sum stop

(b) 7560: Blank 9's row

(9) Program length $(332)_8$ full words

Memory occupied $(7446 - 7777)_8$.

In the initial loading of J108, cells $(0 - 57)_8$ are used, but not when J108 itself is in operation.

Program available as J108A.

7-15-55

Obsolete -
replaced by J131B

J109B DECIMAL DATA INPUT

J109 will input any number of decimal numbers, one per card, at full speed, from the primary reader and place them in consecutive locations of H.S.S. beginning with the location specified in the calling sequence. Scaling of the numbers is specified by p and q field for each number to be input. p is the number of digits in the word from the sign position to the decimal point. q is the number of binary bits to the left of the binal point, excluding the sign position.

One word on each card, with its sign, eleven digits, p, and q can be read. The remaining columns, except column 80, of the card are ignored by J109. error, Reading is terminated when a 12 punch in column 80 is encountered. The last card (12 punch in column 80) is included in the input, and the information on it is stored in the location following the preceding card's location.

Card form:

Cols.	25	26 - 36	37 - 38	39 - 40
	Sign	Word	p	q

Example:

Cols.	25	26 - 36	37 - 38	39 - 40
	+	02345678900	03	06

In the number, p, and q fields, zeros or blanks are considered to be the same. In the sign position, a blank is treated as a plus sign. A minus sign is a punch in row 11. A plus sign punch is a punch in row 12.

Call sequence:

```
    α     RA   α   TRL   XXX (arbitrary symbol)
    α+1                  YYY (YYY = 1st location in H.S.S.)
    α+2   Control returns here
```

Program stop: $(3.0137)_8$: Wrong q for given p. Program will not continue.

Program length:

Region 3: $(130)_{10}$ words: $(3.0000 - 3.0201)_8$

Region 1: $(16)_{10}$ words: $(1.0000 - 1.0017)_8$

Program available as J109B.

$0 \le q \le 99$

$0 \le p \le$ "

Limits on p, q ± ?

Accuracy of input: ~~rounded?~~ truncated (8)

J110B DECIMAL PRINT ROUTINE
J110E

J110 converts to decimal and prints a block of H.S.S. beginning at a specified location, according to scale factors and other specifications stored somewhere in H.S.S. The resulting printed page will have one space at the left, sign and first word, space, sign and second word, space, etc.

Let

a	=	number of words per line
b	=	location of first word to be printed
c	=	number of lines
d	=	location in H.S.S. of scaling factors, etc. (see below)
p_i	=	number of decimal digits left (right if negative) of the decimal point of the i th word.
q_i	=	location of binary point in i th word [+ if right, - if left]
k_i	=	$\begin{cases} 0 \text{ if } p_i \text{ and } q_i \text{ are positive} \\ 1 \text{ '' } p_i \text{ '' } q_i \text{ '' negative} \end{cases}$
n_i	=	number of decimal digits of i th word to be printed.

Thus $\sum_{1}^{a} n_i + 2a \leq 40.$

Calling sequence:

```
    α   020  α   010   XXX (arbitrary symbol)
α + 1    a   b    c     d
α + 2   Control returns here
```

(continued)

Locations d (1) d+a-1 in H.S.S. have

d	k_1	p_1	q_1	n_1
d+1	k_2	p_2	q_2	n_2
.	.	.	.	.
.	.	.	.	.
.	.	.	.	.
d+i-1	k_i	p_i	q_i	n_i
.	.	.	.	.
.	.	.	.	.
.	.	.	.	.
d+a-1	k_a	p_a	q_a	n_a

A minus sign appears as Φ (0 and 1 overprinted). If a number is incorrectly scaled, θ (0 and 2 overprinted) will appear in the sign position followed by all zeros. The program will not stop.

Region 1 (erasable) - 18 words

Region 3 - 102 words

Program available as J110B,E

7-19-55

J112A SELF LOADING BINARY LOADER FOR STYLE D

J112 will load itself and any number of style D cards which follow it into HSS, checking the check sum on each card. Loading will cease when a card with a positive right half 9's row is encountered. The control will then execute the instruction made up by the operation in 9's row C_{1-7} and the address in 9's row C_{29-40}.

Program Location:	0000 - $(0057)_8$.
Program Stop:	$(0050)_8$ Check sum stop.

Program available as J112A

TRL $(13)_8$ to enter J112 again.

7-19-55

J113B SINE-COSINE PROGRAM

OBSOLETE
SEE J157

The four trigonometric functions developed and their locations are:

$\frac{\sin y}{2}$ in location 1.001

$\frac{\cos y}{2}$ in location 1.002

$\frac{\sin 2y}{2}$ in location 1.000 and in MQ

$\frac{\cos 2y}{2}$ in Accumulator

Range: $-1 \leq y < 1$ (in radians)

Region 3 - 31 words

Region 1 (erasable) - 4 words

y in MQ

Calling sequence:

```
α    020   α   010   XXX (arbitrary symbol)
α+1 Control returns here
```

Program available as J113B

Error ?

Time ?

J114A OCTAL MEMORY DUMP

This program prints two full words and the location of the first word in octal on one line. Lines having both words zero are omitted. The section of memory to be printed is specified by a control card or set of control cards.

The program uses a special loader which writes the contents of high-speed memory from 0002_{10} to 0499_{10} onto drum (drum 0, position 0, band 0), so that all of memory except locations 0000 and 0001 may be printed.

Printing is terminated by a blank control card. At the conclusion of printing, memory is restored (except locations 0000 and 0001).

Deck:	J114A Control cards 3 blanks	
Program length:	effectively 2 words in high-speed storage $(764)_8 = (500)_{10}$ on drum	
Stops:	0000	Printing complete, memory restored.
Control Cards:	One for each block of memory desired. First address punched in binary in 9 row, col.48-59 Last address punched in binary in 9 row, col.69-80	

Program available as J114A.

O'FLO IGN. ON

8-31-55

J115B PRINT POSITIVE DECIMAL INTEGERS
J115E

Calling sequence:

Loc	OP	ADDR	OP	ADDR
α		Q	RA	α
α+1	TRL	XXX		N
α+2	Control returns here			

where: Q is the location of the first word.

N, the number of consecutive words to be printed.

XXX, any arbitrary symbol.

J115 will convert consecutive positive words to decimal integers and print them one per line.

Region 3 - 36 words

Region 1 - 10 words

Program available as J115B,E

Speed: 5 digits or less per line - 900 lines/min.

6 or more digits per line - 600 lines/min.

J116B LOGARITHM - BASE 2
J116E

$2^{-39} \log_2 x$ is developed in A, MQ.

$MQ_0 = 1$ is not a part of the answer.

Range: $0 < X < 1$

If $X = 0$ the program will stop at 3.019.

The routine is entered with X in MQ.

Calling sequence:

```
α    020  α  010  XXX
α+1  Control returns here
```

Region 1 (erasable) - 4 words

Region 3 - 23 words

Error less than 2^{-38} assuming X exact.

Program available as J116B, E

J117B LOGARITHM - BASE e or 10
J117E

2^{-39} log X base e or 10 is developed in A, MQ. MQ_0 is not part of the answer.

Range: $0 < X < 1$

If X = 0 the program will stop at 3.030.

Calling sequence:

α	020	α	010	XXX	

α+1	000	L(X)	000	~~X~~ 000^{0}_{1}	for base e
					for base 10

α+2	Control returns here				

Region 1 (erasable) - 6 words

Region 3 - 39 words

Logarithms accurate to 9 decimal places.

Program available as J117B, E

12-15-55

J118A RELATIVE SYMBOLIC ASSEMBLY WITHOUT DRUM

J118A is designed to perform the same functions as J100A, at times when the drum is not available on the JOHNNIAC, making the use of J100A impossible.

All the card forms and versatility of J100A are available without exception on J118A. However, the lack of a drum necessitates somewhat more physical exertion on the part of the programmer or machine operator at assembly time. The procedure for assembling a program with J118A is the following:

The deck to be assembled, with one blank in front and three behind is placed in the secondary card feed. PART I of J118A is then put in the primary feed. The LOAD button should then be pushed. After all cards are read and the punch stops punching, both feeds of the collator should be cleared. The cards that have been punched should be placed in the secondary feed of the collator (with one blank in front and three behind). PART 2 of J118A is then put in the primary feed of the collator. After starting the machine again, the assembly process will continue as in J100A.

Jules Schwartz

8-31-55

OBSOLETE

J119B PRINT DECIMAL - ONE PER LINE
J119E

Calling Sequence:

LOC	OP	ADDR	OP	ADDR
α		A	RA	α
α+1	TRL	XXX		N
α+2		Q		K
α+3	Control returns here			

where: XXX is any arbitrary symbol.

A is the loc. of first word to be printed.

N, the number of consecutive words to be printed.

Q, the scaling or location of the binary point for <u>all N</u> words $0 \leq Q \leq 39$.

K, the number of digits to be printed in the fractional part of the number $1 \leq K \leq 20$.

J119 converts to and prints in decimal N consecutive words of HSS beginning with A. All numbers must be scaled the same in a block, and the same number of digits are converted in the fractional part for the block. A space is left between the integral and fractional parts of the number denoting the decimal point and the minus sign (ⓓ) when appropriate, is printed at the left in the 10^{13} position. Since K can never be less than 1, a number with a Q of 39 will appear as XXXXX 0 and a number with a Q of zero will appear as 0 XXXX . . . Conversion of the decimal part terminates as soon as the quotient of a division by ten is zero so that one digit integers appear as X 0, two digit integers XX 0, etc.

Region C - 59 words

Region 1 - 11 words

Program available as J119B,E

Speed: For integers equal to or less than 4 digits - 900 lines/min.

For integers greater than 4 digits - 600 lines/min.

For fractional numbers with K = 10 possibly 600 lines/min.

1. O'FLO IGN <u>ON</u> OTHERWISE FAILS ON −1.

2. THE SIGN CLOBBERS THE 12th SIGNIFICANT DIGIT TO LEFT OF POINT.

8-10-55

J120B JOHNNIAC BLOCK TRANSFER TO AND FROM DRUM
J120E

J120 transfers any block of information from 1 to 4096 words in length to or from HSS from or to any one position of the drum. One position of the drum is considered to be numbered with addresses 0000(1)7777 octal. "Stepping over" bands is done automatically. The calling sequence is as follows:

α	RA	α	TRL	XXX (Arbitrary Symbol)
$\alpha+1$	Op.	β	OpO	γ
$\alpha+2$	000	f	000	000
$\alpha+3$	Control returns here			

where Op. = {WD if transferring from HSS to drum; RD if transferring from drum to HSS

β = first HSS loc.

γ = last HSS loc.

p = drum position (0,1, or 2)

f = first drum address of position p.

[if b is the band and a the band address for starting, then $f = (2000)_8 \cdot b + a$.]

It is assumed that $\gamma \geq \beta$ and that $f + (\gamma-\beta) \leq (7777)_8$. If the latter does not hold, information will be written (read) up to the end of position p and then starting from the beginning of position p.

Reg. A (erasable) - 3 words, seq. 000,001,002

Reg. C - 30 words

Program available as J120B, E

8-10-55

Errors

OBSOLETE
SEE J149

J121B JOHNNIAC BLOCK TRANSFER TO AND FROM DRUM WITH TALLY

J121 is a "fancy" version of J120. The calling sequence is as follows:

α	RA	α	TRL	XXX (Arbitrary Symbol)
$\alpha+1$	Op.	β	Op0	γ
$\alpha+2$	000	f	000	ℓ
$\alpha+3$	Controls returns either left or right. (See below.)			

where Op = $\begin{cases}\text{WD if transferring from HSS to drum}\\ \text{RD if transferring from drum to HSS}\end{cases}$

β = first HSS loc.

γ = last HSS loc.

p = drum position (0, 1, or 2)

f = first drum address of position p

ℓ = last drum address of position p allowable.

J121 does not assume $\gamma \geq \beta$. However, if $\beta > \gamma$, then $f > \ell$ must also hold. In this case, the interval $[\gamma, \beta]$ in HSS is stored on the drum so that contents of β go to f, contents β-1 go to f-1, etc.

The purpose of ℓ is to provide a check against writing over good information or going past the end of the position. If the interval $[f,\ell]$ (or $[\ell, f]$) is smaller than $[\beta,\gamma]$ (or $[\gamma, \beta]$), then reading or writing will not occur and control returns to the left command of α+3. Otherwise, control returns to the right command of α+3 with the left address part of the accumulator containing the next available drum address, f*, i.e.

$$f^* = f + (\gamma-\beta)+1 \text{ if } \gamma \geq \beta$$
$$= f + (\gamma-\beta)-1 \text{ if } \beta > \gamma.$$

Thus by using an SAL α+2 instruction on returning, C(α+2) acts as an automatic tally on available drum locations.

Note that when storing successive blocks on the drum with $\beta > \gamma$, the blocks are stored in reverse order of their generation. However one-word blocks ($\beta = \gamma$) are always stored in ascending order.

Reg. A (erasable) - 5 words, seq. 000, 001, 002, 003, 004.
Reg. C. - 47 words

Program available as J121B.

8-8-55

J 122A   J 123A

LOADERS FOR STYLE B CARDS

The first few weeks of heavy usage of J101A and J108A (Style B loaders) turned up a few malfunctions in them. Corrections have been made in these loaders, and we have two new loaders for Style B cards. J122A replaces J101A and J123A replaces J108A, J101A and J108A now being obsolete. No new writeup will be made for J122A and J123A. The writeups for J122A and J123A are the same as the writeups on J101A and J108A, respectively, with the following exceptions:

(1) Neither J122A nor J123A check check sums. Thus there are no check sum stops. However, the loaders do look for a negative right half nine's row (punch in column 41) in the program cards.

(2) On page 3 of the J101A writeup the following corrections should be made:

<u>Paragraph (5), Line 2:</u>

Should read: "a card with a punch in column <u>80</u> in the 9's row . . ."

<u>Paragraph (5), Line 7:</u>

Should read: "punch in column 80."

J126B SOLUTION OF A SYSTEM OF ORDINARY DIFFERENTIAL
J126E EQUATIONS

J126 solves a set of differential equations with the following call sequence:

```
α      RA    α   TRL    (XXX) [any symbol]
α+1    000   m   000      d
```

where m is a scaling constant and d is the location or symbol of the first word of the auxiliary subroutine.

The following library region bases must be designated before input of this subroutine in the following manner:

Region	Contents	Use
D	a	The numbers in a+i are the variables y_i (i=0, ..., n-1). Originally the initial values are placed here.
E	b-a	The numbers in b+i are the scaled derivatives, $2^m hy_i$, calculated by the auxiliary subroutine. $b > a+n-1$.
F	c-b	Locations c+i are used as temporary storage for this subroutine. These locations must be cleared to zero before this subroutine is entered for the first time. $c > b+n-1$.
G	c+n	n is the number of differential equations to be solved.

Region 1 (erasable) 4 words

Region 3 $(60)_8 = (48)_{10}$ words

Duration: $T = 6.7+n\ (19.9 + .1\ m) + 4t$ ms.

where T = time in milliseconds to perform one step of integration.

t = time in milliseconds for the auxiliary subroutine.

Description

This subroutine will handle a set of n simultaneous first order ordinary differential equations, in which each derivative is expressed explicitly in terms of the variables.

$$y_0' = f_0 (y_0, y_1, \ldots, y_{n-1})$$
$$y_1' = f_1 (y_0, y_1, \ldots, y_{n-1})$$
$$y_{n-1}' = f_{n-1} (y_0, y_1, \ldots, y_{n-1}).$$

Any differential equation or set of differential equations to be solved must first be expressed in the above form before this subroutine can be applied. For example, the second order differential equation

$$y'' = w^2 y$$

must be written as two first order differential equations

$$y_0' = wy_1 \qquad y_1' = wy_0$$

where $y_1 = y$ and $y_0 = y'/w$.

Each time this subroutine is called into use, it will carry out one integration step of length h. Each of the integrals y_i $(i = 0, 1, 2, \ldots, n-1)$ is replaced by its value at the end of a step of length h. In doing so, this subroutine employs an auxiliary closed subroutine which evaluates the functions $f_0, f_1, f_2, \ldots, f_{n-1}$ from the given values of y_i. The coder must write this auxiliary subroutine for his individual problem since it defines the equations being solved and this depends entirely on his specific problem.

The purpose of the auxiliary subroutine is to calculate and store in locations $b + i$, $(i = 0, 1, 2, \ldots, n-1)$, the quantities hf_i multiplied by a suitable scale factor 2^m. h is the increment of the independent variable and m is a positive integer to be chosen as large as possible without having any of the quantities 2^mhf_i exceed capacity anywhere throughout the range of integration. The factor 2^m is introduced to increase the accuracy of the integration subroutine. The variables, y_i, must all be scaled so that they are less than one throughout the range of integration before they are used in the auxiliary subroutine. Also for maximum accuracy, one should store 2^mh instead of just h. This auxiliary subroutine must be located in a sequence of locations beginning with location d. In integrating over one step, the integration subroutine will call in the auxiliary subroutine four times.

This integration subroutine requires 3n arbitrary storage locations. The n consecutive locations a + i, ($i = 0, 1, 2, \ldots, n-1$; a arbitrary), are used to store the variables y_i. It is in these locations that the initial values are to be placed. It is also in these locations that the final results are found. The n consecutive locations b + i, ($i = 0, 1, 2, \ldots, n-1$; $b > a + n-1$), are used to store the scaled derivatives, $2^mh_y'$ ($= 2^mhf_i$), which are calculated by the auxiliary subroutine. The n consecutive locations c + i ($i = 0, 1, 2, \ldots, n-1$; $c > b + n-1$) are used for temporary storage by the integration subroutine. These locations will hold the quantities 2^mq_i. The numbers left in these locations at the end of an integration step are 3.2^m times the roundoff errors of the quantities y_i. These numbers are taken into account during the following step and serve to prevent the rapid accumulation of roundoff errors. As a result the effective numerical accuracy is m digits more than the capacity of the storage locations. Therefore, it is important that the locations $c+i$, ($i = 0, 1,2,\ldots,n-1$) be cleared to zero before the integration subroutine is entered. Otherwise, this integration subroutine will add spurious corrections to the variables. Thus before the integration subroutine can be entered, the main routine must clear the temporary storage locations $c + i$ to zero and set the initial values of the variables y_i in locations $a + i$.

Summary

Supposing that, in the course of his routine, a coder has to solve a set of differential equations over a specified range given the initial value of the independent variable, a possible procedure would be the following:

(1) Reduce the given set of differential equations to a set of n first order differential equations.

(2) Calculate the initial values of the dependent variables, y_i.

(3) Scale all the functions so that all the values y_i are less than one throughout the range of integration.

(4) Choose a proper value of h (see note I).

(5) Chose m properly.

(6) Determine the parameters to be assigned D-G, observing that $a < b < c$.

(7) Write an auxiliary subroutine which evaluates the functions $2^m hf_i$ and stores them in locations b + i.

(8) Make certain that the main routine sets the scaled initial values in locations a + i, and clears the temporary storage locations c + i to zero before the integration subroutine is entered.

With respect to the solution of a set of differential equations, a program can be broken up into three parts:

(1) The main routine,

(2) The integration subroutine (Code J126)

(3) The auxiliary subroutine.

The Independent Variable

If the independent variable x occurs in the functions f_i or if it is required during an integration as an index, then it may be obtained by integrating the equation $x' = 1$. The independent variable x is then treated as an additional dependent variable, for which the auxiliary subroutine has to provide the quantity $2^m hx' = 2^m h$. However, this latter quantity may be planted at the beginning of the integration in the appropriate location (e.g. in location b) and left there, so that the auxiliary subroutine is relieved of the task. If the independent variable does not appear in any of the f_i's but is merely wanted for indication purposes, it is quicker to use a simple counter in the main routine.

Notes

I) Accuracy: The truncation error in one step is of the order of h^5. Ordinarily, that is for a small set of well-behaved equations, its magnitude is about $10^{-2}h^5$; for large sets or difficult equations it may be greater. Over the range of integration this error will amount to about $h^4/100$. Roundoff errors accumulate at a rate corresponding to the keeping of (39 + m) binary digits. The choice of the length of the increment h is governed largely by the accuracy desired. An increase in the length of h will result in a decrease in accuracy and in operating time. Likewise, a decrease in the length of h will result in an increase in both accuracy and operating time. However, no further increase in accuracy can be gained by choosing $h < 2^{-8}$ because of the introduced truncation error. But, if the functions are very sensitive to variations in y_i, or if the number of equations is very large, smaller steps will probably be necessary with, of course, a corresponding increase in the time required. Now, the process used in the integration subroutine is a fourth order one. Thus, 1/15 of the following difference,

(the value of y_e calculated using an interval of length h)

-(the value of y_e calculated using an interval of length $2h$)

is an approximation of the error.

II) Adjustment of the increment h: There exist essentially two ways of adjusting the increment

a) One may double or halve the increment by varying the value of m in the main routine. This may be done over the complete range of integration or just over part of it. When only the parameter m in the link between the main routine and the auxiliary subroutine is changed to $m+1$ and the auxiliary subroutine is unaltered, the length of the increment is halved. Likewise, when only the parameter m is changed to $m-1$ the length of the increment is doubled. The auxiliary subroutine is not altered since $2^m h = 2^{m+1} h/2$. If one adjusts the increment over the complete range, adjusting only the value of m is sufficient. However, if one wishes to adjust the length of the increment within the range of integration, one must also adjust all the quantities in locations $c + i$. Otherwise, one will introduce roundoff errors in y_i of the magnitude,

$$2 \text{ (old value of } h \text{ - new value of } h) \times 2^{-40}.$$

Now by also doubling the quantities in $c + i$ when one halves the increment one will introduce no roundoff error. Similarly, by halving the quantities in $c + i$ when one doubles the length of the increment, one will introduce no roundoff error. If one clears the locations c + 1, one introduces roundoff errors of magnitude 2^{-40}.

b) One may alter the length of the increment in any ratio by adjusting the scaling factor $2^m h$ in the auxiliary subroutine. Here also one may adjust the length of the increment within the range of integration. Now it is not necessary to adjust the quantities in c + 1. If, however, $2^m h$ becomes small, then roundoff errors are introduced by inaccuracies in the auxiliary subroutine. Thus one should not keep $2^m h$ small when integrating over large ranges unless the loss of accuracy and time does not matter.

III) Often it is desired to evaluate functions involving expressions like sin x or $J_m(x)$. These expressions can be evaluated by solving extra distinct differential equations along with the desired ones. For example,

$$d^2/dx^2 \ (\sin x)/2 = -(\sin x)/2.$$

Thus we can evaluate (sin x)/2 by using the extra pair of equations

$$y'_{n+1} = 2^m h y_n \qquad y'_n = -2^m h y_{n+1}$$

and suitable initial conditions.

Method Used for Integration in the Routine

Given a set of differential equations,

$$y_i' = f_i\,(y_0, y_1, y_2, \ldots, y_{n-1}),\quad (i = 0, 1, 2, \ldots, n-1)$$

the process used in the integration is defined by the following equations

$$k_{ij} = 2^m h f_i\,(y_{0j},\ y_{1j}, \ldots, y_{n-1\ j})$$

$$r_{i,j+1} = (A_{j+1}+1)\quad (k_{i,j} - B_j\,q_{i,j})$$

$$y_{i,j+1} = y_{i,j} + 2^{-m}\,r_{i,j+1}$$

$$q_{i,j+1} = q_{ij} + 3r_{i,j} + (C_j - 1)\,k_{i,j+1}$$

with the following table of values

j	A_{j+1}	B_j	C_j
0	$-1/2$	2	$1/2$
1	$-(1/2)^{1/2}$	1	$(1/2)^{1/2}$
2	$(1/2)^{1/2}$	1	$-(1/2)^{1/2}$
3	$-5/6$	2	$1/2$

Of the double subscripts used in the above equations, the first subscript, i, indicates which variable is being considered, and the second subscript, j, indicate which of the four parts of one step is being performed. The auxiliary subroutine evaluates the quantities $k_{i,j}$. In the above equations, only the quantities $q_{i,4}$ and $y_{i,4}$ are carried over from step to step. The quantities $r_{i,j}$ are calculated in the course of one step; they are not carried directly from step to step. When $j \doteq 4$, we replace it by zero, increase i by 1, and terminate the step.

For one step, the sequence of operations is as follows:

$j = 0$	$i = 0, 1, 2, \ldots, n-1$
$j = 1$	$i = 0, 1, 2, \ldots, n-1$
$j = 2$	$i = 0, 1, 2, \ldots, n-1$
$j = 3$	$i = 0, 1, 2, \ldots, n-1$

References

Gill, S., "A process for the Step-by-Step Integration of Differential Equations in an Automatic Digital Computing Machine", Proceedings of the Cambridge Philosophical Society vol. 47 (1951) pp. 96-108

Wilkes, M.V., Wheeler, D.J., and Gill, S., The Preparation of Programs for an Electronic Digital Computer Addison-Wesley Press, Inc. Cambridge, Mass., (1951) pp. 32-33, 56-57, 86-87, 132-134.

Program available as J126B, E

J127B ARCTANGENT X
J127E

Arctangent X (in radians) is developed in MQ.

Range: $-1 < X < 1$.

The routine is entered with X in MQ.

Calling sequence:

```
α     020  α    010   XXX
α+1   Control returns here
```

Region 1 (erasable) - 3 words

Region 3 - 28 words

Arctangents correct to 11 decimal places, assuming an exact argument.

Program available as J127B,E

J128B,E PUNCH DECIMAL CARDS

J128 punches one decimal card containing 6 twelve-place fields and a three-digit sequence number.

Calling sequence:

α	020	α	010	XXX
$\alpha+1$	—	—	—	SEQ
	0 1-6	7-19	20-27	28-39
$\alpha+2$	k_1p_1	q_1	—	L_1
$\alpha+3$	k_2p_2	q_2	—	L_2
$\alpha+4$	k_3p_3	q_3	—	L_3
$\alpha+5$	k_4p_4	q_4	—	L_4
$\alpha+6$	k_5p_5	q_5	—	L_5
$\alpha+7$	k_6p_6	q_6	—	L_6
$\alpha+8$	Control returns here			

where:

$k_i = \begin{cases} 0 \text{ (in sign position) if p and q positive} \\ 1 \text{ (if the negative values of } p_i \text{ and } q_i \text{ are to be used)} \end{cases}$

p_i
- positive = number of decimal digits to the left of the decimal point.
- negative = number of lead zeros desired to be between the decimal point and the first digit of the word in memory.

q_i
- positive = number of positions right from 2^0 (excluding 2^0) to binary point.
- negative = number of positions left from 2^0 (excluding 2^0) where binary point is assumed to be.

L_i = location of the i th number.

continued

The punched card form is:

Column	
1	blank
2-4	sequence number
5-16	1st number
17-28	2nd number
29-40	3rd number
41-44	blank
45-56	4th number
57-68	5th number
69-80	6th number

If p_i and q_i are positive field, i contains 11 digits and a decimal point after the integer part of the number.

If p_i and q_i are negative, the absolute value of p_i is punched in the first column of the field followed by 11 digits.

An 11 is overpunched in the 12th column of the field for negative numbers. No punch is used to designate positive numbers.

If any cell in $\alpha+2$ through $\alpha+7$ is left blank (zero), the corresponding field on the card will be blank.

Program length: Region 1 (erasable) - 30 words

Region 3 - 170 words

No program stops.

Timing: Cards will be punched at half-speed with a reasonable amount of manipulation between entering of the program.

Program available as J128B,E

J129B, E CARD IMAGE TO CHARACTERS

J129 will convert a card image in storage at D 0 - D 23 in 9L, 9R, 8L, ..., 12R order to 80 words in E 1 - E 80 one character per word corresponding to the columns of the card image. The character code is the sum of the values of the punches for the character. A "12" = 16, "11" = 32, "0" = 48 and numeric 1-9 = 1-9 respectively. All "legal" characters of the key punch will be converted to unique character codes.

The routine is designed for "legal" characters only. The card image is clobbered in the conversion process.

Calling sequence:	α	RA α TRL XXX
	α+1	Control returns here
Program storage:	C 0 - C 51	program
	D 0 - D 23	card image
	E 1 - E 80	characters
Timing:	Approx. 135 ms.	

No program stops.

Program available as J129B, E

J130B
J130E DECIMAL LOADER

J130 is a decimal loader with the following calling sequence:

```
α      020   α    010    XXX
α+1    00a  0000 000     c
α+2    Control returns here
```

where a = {0 if primary feed of card reader used; 1 if secondary " " " " "}

c = starting location (absolute or relative)

The program reads decimal cards starting each card with column 5 [cols. 1(1)4 are for I.D. and are ignored], and proceeding to the right, converting and storing numbers, until a blank column is encountered, which causes the next card to be read, or an end of file 12 punch in col. 80 is encountered.

The program will store the numbers in the starting location and succeeding locations unless it hits a specified location on a card indicated by L followed by the location number. At this time it will store the number following in the specified location and proceed as before with the new location as a starting location. The numbers to be loaded must be punched in the cards in the following form:

L c ($\pm$ N $^{x}/$ p F) + q

where (= 11, 4, 8 punch

and) = 12, 4, 8 punch

(continued)

and

c = location in which to store the number following. This will always be preceded by the symbol L. Absence of L and location will result in the number being stored in (location of preceding number) +1.

N = number to be converted and stored (must include a decimal point and be p eceded by a sign). N limited to eleven decimal digits. [9 decimal digits if floating point numbers.]

xp or /p = These are optional. If absent, N will be handled as it appears in the card. xp will cause N to be multiplied by 10^p. /p will cause N to be divided by 10^p. $0 \leq p \leq 18$.

F = F punched at this point will cause the number to be converted into the floating point form (power + 50 and magnitude), described in J102.

2^{-q} is the scale factor. $-8 \leq q \leq 39$.

A plus sign is a 12 punch and a minus sign an 11 punch. For example, a program entered with calling sequence:

```
α      020    α      010    XX   (arbitrary symbol)
α+1    000    0000   000    100
```

and first card punched (starting in col. 5) as follows:

```
(+325.14)+15(-2.935x4)+22
L200(-5.62901/3)-6(+.0004F)
(+1.5604x2F)
```

will result in

$+325.14\times2^{-15}$	being	stored	as	a fraction	in	100
-29350×2^{-22}	"	"	"	"	"	"	101
$-.00562901\times2^{6}$	"	"	"	"	"	"	200
+47 400000000	"	"	"	an integer	"	201
+53 156040000	"	"	"	"	"	"	202

Program stop at $\left\{\begin{matrix}3.(088)_{10}\\3.(130)_{8}\end{matrix}\right\}$ when a card without a punch in column 5 is encountered.

Region 1 (erasable)--$(46)_{10}$ words

Region 3 -- $(314)_{10}$ words

Program available as J130B
J130E

Marvin Shapiro

J131B DECIMAL DATA INPUT
J131E

J131 will convert and store in consecutive locations of H.S.S., decimal numbers, one per card at full speed from the primary feed. The first location is specified in the calling sequence and there is no limit to the number of cards to be read. The scaling of each number is specified in each card by p and q, where p is the location of the decimal point counting right from the sign position (but not including the sign position) and q is the location of the binary point counting right from the zero-th bit (not including the zero-th bit). Fractional binary representations are truncated in absolute value. This routine will not input -1×2^0.

Card form:

Col.	25	26-36	37-38	39-40
	sign	number	p	q
		$0 \leq p \leq 11$	$0 \leq q \leq 99$	

Example:

Col.	25	26-36	37-38	39-40
	$\pm$	00326947300	04	07

Zeros and blanks are interchangeable. An 11-punch in col. 25 is a minus sign and a 12-punch or blank is a plus sign. Reading is terminated when a 12-punch in column 80 is read. The remaining columns of the card may contain any other information desired.

(continued)

Calling sequence:

LOC	OP	ADDR	OP	ADDR	
α	RA	α	TRL	XXX	(Arbitrary symbol)
$\alpha+1$				YYY	(1st location in H.S.S.)
$\alpha+2$	Control returns here				

Program stop: $(3.0140)_8$ wrong q for given p. Program will not continue.

Program length:

Region 3: $(131)_{10}$ words: $(3.0000 - 3.0202)_8$

Region 1: $(16)_{10}$ words: $(1.0000 - 1.0017)_8$

Program available as J131B.
J131E.

9-20-55

J132B PRESET OR RESET ROUTINE
J132E

J132 sets any number of specified addresses to specified values. Its primary intended use is in connection with J133.

Calling Sequence:

α	RA	α	TRL	XXX
$\alpha+1$	$00k_1$	A_1	000	B_1
.	.	.	.	.
.	.	.	.	.
.	.	.	.	.
$\alpha+i$	$00k_i$	A_i	000	B_i
.	.	.	.	.
.	.	.	.	.
.	.	.	.	.
$\alpha+p$	$00k_p$	A_p	000	B_p
$\alpha+p+1$	100	0000	Control returns here	

If $k_i = 0$, A_i will be stored in the left address of the word in B_i.
If $k_i = 1$, B_i " " " " " right " " " " " A_i.

Program length: 9 words -- all in region 3

Program available: J132B,E

9-20-55

J133B PSEUDO B-BOX
J133E

J133 provides for a rudimentary form of internally programmed B-Boxes. It will successively add (or subtract) specified increments to specified addresses, allowing for intermediate calculations using the transformed words, and will control the number of repetitions of this process.

Calling Sequence:

α	RA	α	TRL	XXX
α+1				n
α+2	$00k_1$	A_1	000	B_1
.	.	.	.	.
.	.	.	.	.
.	.	.	.	.
α+i+1	$00k_i$	A_i	000	B_i
.	.	.	.	.
.	.	.	.	.
.	.	.	.	.
α+p+1	100	C	000	D

If $\begin{Bmatrix} k_i = 0 \\ k_i = 1 \end{Bmatrix}$ then $\begin{Bmatrix} A_i \\ B_i \end{Bmatrix}$ will be added (modulo 2^{12}) to the $\begin{Bmatrix} \text{left} \\ \text{right} \end{Bmatrix}$ address of the word in $\begin{Bmatrix} B_i \\ A_i \end{Bmatrix}$; n-1 will be stored in α+1, and if the result is ≥ 1 control will go to the left of D while if it is < 1 control will go to the left of C.

Suggested Use:

Have a routine, S, containing the words whose addresses are to be modified, which exits to the left of α. Have the variable addresses set to their desired initial values -- the values for the first execution of S -- by use of J132. Let

D = the entry point of S (it is required that S be enterable on the left). Let n = the desired number of executions of S, Transfer control to the entry of S.

Then control will eventually proceed to the left of C, S having been executed exactly n times, with the required address modifications. $\alpha+1$ will contain a zero and the variable addresses of S will be set up for an (n+1)st execution.

Caution: If the process is begun by transferring to α, the first execution of S will not take place until after one address modification and there will be only n-1 executions of S.

S may itself contain call sequences using J133.

Program length: 17 words - all in region 3.

Program available: J133B,E

No program stops.

1-23-56

J134A SYMBOLIC RELATIVE ASSEMBLY

J134A has the same features as J100A with the following exception. If no RDOR card is put in for a region in the decimal deck, the binary equivalent of the decimal region will be used as the origin for this region. Thus there is no 1234 stop with J134A.

Jules I. Schwartz

Easy Fox

J 135	Style F loader
J 136	Style E loader

These writeups are in the Easy-Fox envelope.

X267

REVISED STYLE E LOADER TO REPLACE J136

I'd like to have your assistance in checking out this new style E loader by your using an interim version (X267) for a few days to see if any bugs develop or if any desirable changes are suggested before the routine is placed in the library under its ultimate identification (J267).

The new loader is roughly campatible with J136 but the following differences should be understood before using it:

1. The new loader occupies 100 words more than J136 for a total of 600 words (0000-0599).

2. The permitted symbolic locations and addresses are again * 0 thru * 99 but only 100 forward symbolic references are allowed instead of 155.

3. There is no longer a restriction against using a forward symbolic reference in words being loaded into absolute or relative locations.

4. The initial condition of the counter ($) is 600, $(1130)_8$.

5. It is no longer necessary to put in a redundant comma card at the beginning of a deck just to clean up the tables. The initial condition of the new loader is such that it is ready to load the first routine.

6. Certain kinds of errors are detected by the new loader and corresponding error messages are typed out on the printer. The left half of the current card is always printed out as part of the message except in the event of more than one error message for that card in which case the card is printed only with the first message. The error message pertains to the card which is printed except for undefined symbolic addresses. Here a comma card is printed out to identify the routine in which the undefined symbols occurred. A full-table error message occurs when the 101st forward symbolic reference is encountered. A "CLOBBERED" error message occurs when an attempt is made to use the counter to load beyond the end of memory or anywhere in the loader area. Both of these error types end in drop dead halts. The other error messages for ambiguous symbols, undefined symbols, undefined characters, and scaling errors are followed by setting an error flag in the loader but the loading process continues to edit the deck for other errors. When the period card is encountered (period in C36 that is) if the error flag has been set the loader will drop dead.

7. The new loader will clean up (substitute and initialize) only on a comma card. The period card now causes a transfer directly to the period cell without a clean up. The initial condition of the period cell is a drop dead halt.

8. Style F binary cards with no ID can be called for in two ways:

```
LOCN OPN ADDR OPN ADDR
   (     aaaa     bbbb)
```

Such a card will cause the punching of style F binaries to cover the range aaaa thru bbbb. The aaaa and bbbb parameters must be explicit--that is, they cannot be forward symbolic references.

```
LOCN OPN ADDR OPN ADDR
   (                 (
```

Such a card will set the beginning of range parameter to $000.

```
LOCN OPN ADDR OPN ADDR
   )                 )
```

Such a card will set the end of range parameter to $000 less one and will then cause the punching of style F binaries from the beginning of range thru the end of range. Note that it is now simple to punch out routines individually or in a block but that the resulting binaries will not in general include the directory which should be punched explicitly as in the following example:

```
LOCN OPN ADDR OPN ADDR
   (      = 0     = 13)
```

Obviously one should not insert such a card between a pair of double-parenthesis cards.

Punching will be suppressed if the error flag has been set.

9. Data input is the same as in J136. Floating point numbers intended for the Johnniac floating point system must be punched with the decimal point (which separates the translated exponent from the significant digits) in C21. C18 must be punched + ("12" punch) or - ("11" punch).

```
LOCN OPN ADDR OPN ADDR
    +52.123456000                +12.3456
    -47.123456789                  -.000123456789
    +53.123                     +123.
    -59.      123               -123.    (unnormalized
```

Fixed point data is recognized by the binary scaling punched in the right address field. Call it b. The limits are 0 thru 40 for b. The number times two to the minus b rounded to Johnniac word size must be a <u>proper</u> fraction. A scaling error message will be printed if this is not the case or if b is greater than 40.

```
LOCN OPN ADDR OPN  ADDR
    +123456789012   39        +123456789012 x 2^-39
     -.12345678901   0        -.12345678901 x 2^0
    +          123  39        +          123 x 2^-39
    -         123.  39        -          123 x 2^-39
    +         123   39        +         1230 x 2^-39
    -   123.45678    7        -123.45678     x 2^-7
```

REVISED STYLE E LOADER TO REPLACE J136 J. C. Shaw 1-13-58

I'd like to have your assistance in checking out this new style E loader by your using an interim version (X267) for a few days to see if any bugs develop or if any desirable changes are suggested before the routine is placed in the library under its ultimate identification (J267).

The new loader is roughly compatible with J136 but the following differences should be understood before using it:

1. The new loader occupies 100 words more than J136 for a total of 600 words (0000-0599).

2. The permitted symbolic locations and addresses are again * 0 thru * 99 but only 100forward symbolic references are allowed instead of 155.

3. There is no longer a restriction against using a forward symbolic reference in words being loaded into absolute or relative locations.

4. The initial condition of the counter ($) is 600, $(1130)_8$.

5. It is no longer necessary to put in a redundant comma card at the beginning of a deck just to clean up the tables. The initial condition of the new loader is such that it is ready to load the first routine.

6. Certain kinds of errors are detected by the new loader and corresponding error messages are typed out on the printer. The left half of the current card is always printed out as part of the message except in the event of more than one error message for that card in which case the card is printed only with the first message. The error message pertains to the card which is printed except for undefined symbolic addresses. Here the comma card is printed out to identify the routine in which the undefined symbols occurred. A full-table error message occurs when the 101st forward symbolic reference is encountered. A "CLOBBERED" error message occurs when an attempt is made to use the counter to load beyond the end of memory or anywhere in the loader area. Both of these error types end in drop dead halts. The other error messages for ambiguous symbols, undefined symbols, undefined characters, and scaling errors are followed by setting an error flag in the loader but the loading process continues to edit the deck for other errors. When the period card is encountered (period in C36 that is) if the error flag has been set the loader will drop dead.

7. The new loader will clean up (substitute and initialize) only on a comma card. The period card now causes a transfer directly to the period cell without a clean up. The initial condition of the period cell is a drop dead halt.

8. Style F binary cards with no ID can be called for in ~~###~~ two ways:

```
LOCN OPN ADDR OPN ADDR
  (       aaaa      bbbb)
```

Such a card will cause the punching of style F binaries to cover the range aaaa thru bbbb. The aaaa and bbbb parameters must be explicit--that is they cannot be forward symbolic references.

```
LOCN OPN ADDR OPN ADDR
   (                 (
```

Such a card will set the beginning of range parameter to $000.

```
LOCN OPN ADDR OPN ADDR
   )                 )
```

Such a card will set the end of range parameter to $000 less one and will then cause the punching of style F binaries from the beginning of range thru the end of range. Note that it is now simple to punch out routines individually or in a block but that the resulting binaries will not in general include the directory which should be punched~~XX~~ explicitly as in the following example:

```
LOCN OPN ADDR OPN ADDR
   (     = 0     = 13)
```

Obviously one should not insert such a card between a pair of double parenthesis cards.

Punching will be suppressed if the error flag has been set.

9. Data input is the same as in J136. Floating point numbers intended for the Johnniac floating point system must be punched with the decimal point(which separates the translated exponent from the significant digits) in C21. C18 must be punched + ("12" punch) or - ("11" punch).

```
LOCN OPN ADDR OPN ADDR
    +52.123456000                 +12.3456
    -47.123456789                   -.000123456789
    +53.123                       +123.
    -59.      123                 -123.    (unnormalized)
```

Fixed point data is recognized by the binary scaling punched in the right address field. Call it b. The limits are 0 thru 40 for b. The number times two to the minus b rounded to Johnniac word size must be a <u>proper</u> fraction. A scaling error message will be printed if this is not the case or if b is greater than 40.

```
LOCN OPN ADDR OPN ADDR
    +123456789012     39        +123456789012 x 2^-39
    -.12345678901      0        -.12345678901 x 2^0
    +           123   39        +         123 x 2^-39
    -          123.   39        -         123 x 2^-39
    +          123    39        +        1230 x 2^-39
    -     123.45678    7        -123.45678    x 2^-7
```

10-17-55

J135A LOADER FOR STYLE F ABSOLUTE BINARY CARDS

J135 is a one-card self-loading routine which loads Style F absolute binary cards from the primary feed. After bootstrapping its way into HSS the routine selects the primary feed and executes the control instruction which is punched in C_{20-40} of the 9 row of the incoming card with the information word in C_{41-80} of the 9 row in the accumulator. C_{20-40} will usually contain a store instruction. The routine then processes the 8, 7, ..., 12 row in the same way unless control is taken away from J135 by a transfer instruction in C_{20-40} of some row. If the control remains in J135 after processing all rows of the incoming card, then the routine will select the primary feed for another card and continue. The MQ is used by J135 to count the 12 rows.

After bootstrapping into HSS, J135 looks like:
(in octal)

LOCN	OPN	ADDR	OPN	ADDR					
0001	100	0000	004	0005	1	SEL	0	LDQ	5
0002	101	0003	014	0003	2	LPY	3	TRC	3
0003					3	****		****	
0004	075	0117	002	0002	4	CLH	117	TLP	2
0005	010	0001	000	4000	5	TRL	1	NØP	4000

Storage: 0001-0005

No error stops

Program available as J135A.

It is interesting to note that although J135 is a self-loader, it carries identification, and further that the J135 card itself is in style F.

J.C. Shaw

J136A LOADER FOR STYLE E PROGRAM AND DATA CARDS

J136 is a self-loading routine which loads Style E program and data cards. The loader loads instructions with absolute, relative, and symbolic locations and addresses in either octal or decimal. Operation codes must be in octal. Data words of two forms are accepted: fixed point decimal data punched with decimal point and a specified binary scale factor, and floating decimal data of the form used by J102.

The principle of the loader is straightforward: convert the location, convert and store the word to be loaded, continue this process with the other words of the routine to be loaded until the end of a routine is indicated by a punch in C_{36} at which time sweep back over the routine just loaded filling in absolute equivalents for symbolic addresses which were undefined when first encountered, wipe out the Symbol Table in preparation for another routine, then begin loading the next routine or begin execution according to the punch in C_{36}.

Locations and addresses are four character fields and will be converted base ten unless prefixed by a fifth character (by convention a comma) to indicate base eight. Blank columns are everywhere numerically equivalent to zeros but logically distinct from zeros. Relative locations and addresses consist of a character (A through Z, #, $) followed by a three-digit number. When a relative location or address is converted its absolute equivalent is formed by adding to the three-digit number the contents of the cell reserved for the character by the loader. A character is "defined" by loading a non-negative integer < 4096 into the cell reserved for the character. If at conversion time the character part of a relative location or address is still undefined (as indicated by a negative number in the cell reserved for the character) the loader will halt. A character may be redefined at any time in the loading process by loading the new number into the cell reserved for the character.

The initial condition of the loader allows for 100 symbols (* 0 through * 99) but this number may be changed at will by the programmer. A symbolic location or address consists of an asterisk followed by a three-digit number. One word is allowed for each symbol in the Symbol Table. When a symbolic location is encountered the loader verifies that the symbol has not previously been defined, then defines the symbol by storing the value of a counter (cell 43 corresponding to the character "$") in the Symbol Table. The loader prepares to store the word to be loaded in the cell indicated by the counter, converts the word, advances the counter by one and stores the word. When a blank location is encountered the treatment is identical to that for a symbolic location except, of course, there is no symbol to define. Notice that the counter is automatically advanced only for symbolic and blank locations. In converting a symbolic address the loader replaces the symbolic address by its absolute equivalent if the symbol has been defined, otherwise an entry is made in the Forward Reference Table so that when loading of the current routine is complete the loader may go back and fill in an absolute address for the symbolic address. If the symbolic address is still undefined (as indicated by a negative number in the Symbol Table) the loader will halt.

When C_{18} is punched ("12" for +, "11" for -), the loader will interpret the card as containing a data word of the following format:

C_{18}	contains the sign
C_{19-30}	contains the number with decimal point or an integer $< 2^{39}$.
C_{31-35}	is blank for floating decimal numbers or, in the case of fixed point numbers, contains "q" $(0 \leq q \leq 40)$ where 2^{-q} is the desired scale factor to apply to the number to scale it down to a proper fraction for internal use. Floating point numbers are converted exactly, fixed point numbers are rounded.

The loader interprets a punch in C_{36} as indicating the end of a routine. The loader fills in absolute addresses for symbolic addresses noted in the Forward Reference Table, then it clears the Symbol Table to -1's in preparation for another routine and finally transfers to the cell corresponding to the punch in C_{36}. If the punch is a comma, the loader will be reentered for loading another routine. If the punch is a period, JOHNNIAC control will pass to cell 27 which according to convention will have been loaded with a transfer instruction to the beginning of the master routine for the program just loaded.

J136A is designed for a master routine-subroutine approach to a problem with relative addresses for blocks of data, symbolic addresses for internal references in routines and (by convention) a special set of relative addresses using the character "#" for links between routines through a directory provided by the programmer.

A small limitation of J136A is that forward reference symbolic addresses are not permitted in words having absolute or relative locations.

Timing: Loads at full speed except for cards punched in C_{36}.

Storage: 0 - $(499)_{10}$ can be varied.

Program available as: J136A.

J. C. Shaw

```
IDENTIF.      LOCN OPN ADDR OPN ADDR     COMMENTS

         1111111111222222222233333333334444444444555555555566666666667777777777778
12345678901234567890123456789012345678901234567890123456789012345678901234567890

SAMPLE        1000 020 1234 024 4095     ABSOLUTE DECIMAL LOCATION AND ADDRESSES.
SAMPLE       ,1750 020,2322 024,7777     ABSOLUTE OCTAL LOCATION AND ADDRESSES.
SAMPLE        A123 020 A123 010 # 13     RELATIVE LOCATION AND ADDRESSES.
SAMPLE        * 35 020 * 50 024 * 51     SYMBOLIC LOCATION AND ADDRESSES.
SAMPLE           $               1000,   SET THE COUNTER TO 1000 AND PRESET TABLE
SAMPLE           $               $100    SKIP THE COUNTER 100.
SAMPLE           $               A  0    SET THE COUNTER EQUAL TO A000.
SAMPLE           A               2000    SET A EQUAL TO 2000.
SAMPLE           A               $  0    SET A EQUAL TO COUNTER VALUE.
SAMPLE        # 10 010 $  0 014 $  0     SET DIRECTORY FOR ROUTINE WHICH FOLLOWS.
SAMPLE            &1234.5678901   11     FIXED POINT DATA, Q EQUALS 11.
SAMPLE            -123456789012   39     FIXED POINT DATA, Q EQUALS 39.
SAMPLE            &52.123456789          FLOATING POINT DATA FOR J102.
SAMPLE             . 010 #  1        .   SET LINK TO ROUTINE, CLEAN UP, KICK OFF.
```

~~Fix up to allow 52.123 to be entered for floating point no matter where decimal point appears.~~

Test for improper q

J136A PROGRAMMED HALTS
(in octal)

NIA	IR	Reason
0034	130 0033 134 0033	No link to beginning of program.
0167	010 0173 134 0166	Ambiguous symbolic location somewhere in the last region, or symbol table wasn't preset to -1's. PR, 010 0161, PR
0220	001 0220 134 0217	Too many forward references in the current region.
0244	002 0244 134 0243	Undefined symbol in left address somewhere in last region. PR, 010 0242, PR
0252	130 0251 020 ----	Undefined symbol in right address somewhere in last region. PR, [illegible]
0275	002 0275 134 0274	Undefined relative location or address on last card. PR, 010 0273, PR

J137B MANIPULATION OF ARRAYS
J137E

J137 is intended for use in manipulating sets of quantities systematically distributed in the memory. The following definitions are basic:

An array is a doubly indexed system $\{a_{i,j}\}$ of cell addresses, such that

$$a_{i,j} = a + \alpha i + \beta j \text{ (modulo } 2^{12})$$
$$0 \leq i \leq m$$
$$0 \leq j \leq n \quad .$$

An array is completely specified by the quantities α, β, m, n and a. By the designator of such an array we shall mean an address, A, such that information defining the array is stored in A, A+1, A+2 as follows:

```
A      000   α   000 β
A+1    000   m   000 n
A+2    000 0000  000 a
```

J137 allows the manipulation of the contents of arrays(1), by referring only to their designators.

J137 will place $f((a_{i,j}), (a'_{i,j}))$ in $a'_{i,j}$, for each i,j, where $\{a_{i,j}\}$ and $\{a'_{i,j}\}$ are arrays and $f(x,y)$ is an arbitrary function.

Calling Sequence for J137

```
θ      RA    θ     TRL    XXXX
θ+1    000   A     000    A'
θ+2    (← W  →)    Control returns here.
```

XXXX is the first word of J137. A and A' are designators of

(1) The contents of an array $\{a_{i,j}\}$ is the doubly indexed set of numbers $\{(a_{i,j})\}$. We denote the contents of x by (x), for any address x.

arrays $\left\{a_{i,j}\right\}$ and $\left\{a'_{i,j}\right\}$.

W is an instruction of the form TRL YYYY or TRR YYYY.

YYYY is an address at the left or right (depending on W) of which is situated the entry order of a routine, F, which computes f(x,y) given x in the accumulator and y in 2.0000. F must exit to the <u>right</u> of XXXX with f (x,y) in the accumulator.

<u>Note 1</u>: J137 will not use the m' and n' associated with A' but will assume m' = m and n' = n. There need be no relation between a, a', α, α', β, β'.

<u>Note 2</u>: Each of the quantities $(a_{i,j})$ and $a'_{i,j})$ will be operated upon by F exactly once so that F may compute various types of cumulative information, such as $\sum_{i,j} (a_{i,j})$.

<u>Note 3</u>: At the time that $(a_{i,j})$ and $(b_{i,j})$ are operated upon by F

m-i is stored in 2.0002

n-j is stored in 2.0001

Hence, F may compute the indices of those $(a_{i,j})$ and/or $(b_{i,j})$ having certain properties.

<u>Note 4</u>: F can be made vacuous by having W = TRR XXXX. This will result in the contents of $a'_{i,j}$ being replaced by those of $a_{i,j}$, i.e. the array $\left\{(a_{i,j})\right\}$ being transferred to $\left\{a'_{i,j}\right\}$.

<u>Note 5</u>: If F uses a subroutine having an erasable region, this region must be different from the erasable region of J137. For this reason, region 2, rather than region 1, was used.

continued

Region 2 - erasable - (see note 5) 6 words

Region 3 - 37_{10} words

Program Stops: None (except possibly in F).

Program available as J137B.
J137E.

Norman Shapiro

J138B
J138E

MULTIPLICATION OF ARRAYS

J138 forms the scaled matrix product of two arrays in a third array(1).

Calling Sequence:

```
        θ      RA      θ    TRL     XXXX
        θ+1    000     A    000     A'
        θ+2    (←  S  →)    000     A"
        θ+3    Control returns here.
```

where A, A', A" are array designators and S is a shift instruction. After execution of J138(2).

$$(a''_{i,k}) = \sum_{j=0}^{j=n} s(a_{i,j}, a'_{j,k}). \quad 0 \leq i \leq m$$
$$0 \leq k \leq n'$$

s denotes the effect of the shift operation, S, on the double precision product $a_{i,j}$, $a_{j,k}$.

Note 1: J138 will not use m', m", or n", but will assume m' = n, m" = m, and n" = n'.

Note 2: An additive overflow will turn on the overflow toggle with control at the right of 3.0041.

Note 3: If the scaling is such that it is preferable to shift right and add from the MQ rather than shifting left and adding from the accumulator, then change two words of J138 as follows:

```
    3.0041    000    0000      RA    2.0005
    3.0042    AQS    2.0005    RA    3.0040
```

(continued)

(1) See the write-up for J137 for terminology.

(2) Let A designate $\{a_{i,j}\}$, A' designate $\{a'_{i,j}\}$, etc.

Region 1 - erasable - 6 words

Region 3 - 52_{10} words

Program Stops: See note 2.

Program available as J138B
J138E

Norman Shapiro

12-5-55

OBSOLETE

OFLO 1ON ON

J139A MATRIX INVERSION

J139A will solve systems of equations and/or invert matrices of order n $(1 \leq n \leq 40)$. All arithmetic for this routine is carried out in double-precision fixed-point to guarantee maximum accuracy.

The matrix of coefficients and the right-hand side for this routine are input in decimal, one per card, having the following format:

Col.	Contents
17-19	j $(1 \leq j \leq n)$ $j = 0$ for the right-hand side
20-22	i $(1 \leq i \leq n)$
24-27	Integral portion of a_{i_j}
28-33	Fractional portion of a_{i_j}
34	Sign (Blank is +; 1 is minus)

The remainder of the card may contain any information desired.

It is not necessary to enter the zero elements of the system since HSS is cleared to zero from 610 through 4095 each time the Input routine is called for. The elements may be in any order. Each matrix or system of equations must be headed by a single card with the order of the system, n, in j and cols. 20-34 blank, and followed by a blank card to signify the end of the deck. (This input format is the same as the JOHNNIAC simplex code.)

As many systems of equations to be solve or matrices to be inverted as desired may be entered behind the program deck, just so each deck has the proper header card and is followed by a blank. Two additional blank cards are necessary at the very end of the deck.

The following things may be done with J139:

1) To obtain just the solution to one set of equations and no inverse, put H1 on.

2) To obtain the solutions to more than one set of equations and no inverses, put T1 on.

3) The inverse of the matrix is punched in binary for later use with J141A; if this is not desired, put T2 on.

4) The inverse of the matrix is printed row by row; if this is not desired put T3 on. (The solution to the system of equations is always printed; in the case where no right-hand side was entered, an n-order vector of zeros is printed. In addition, following the computation of the actual inverse a check vector of order n is printed which should be all 1's. The difference between 1 and the actual values printed is a measure of the accuracy obtained.)

5) To invert a single matrix, put H2 on; for more than one matrix,leave H2 off.

The deck is put together as follows: J139, header card (n in j) coefficients (and right-hand side), a blank, header card, etc., two blanks.

Program Stops:

NIA = $(0055)_8$,	No header card for matrix or i or j is greater than n.
NIA = $(0025)_8$	H1 on, System solved. Hit GO to go on and invert the matrix if T1 is not on, or read in next matrix if T1 is on.

NIA = $(0152)_8$ The matrix is singular or poorly conditioned.

NIA = $(0317)_8$ Overflow, rescale the matrix.

NIA = $(0052)_8$ H2 on, Matrix inverted, hit GO to read in next matrix.

J139 uses all of HSS and the Drum.

Program available as J139A.

Time: The only estimate available is for inverting a 24 order matrix -- this took roughly three minutes of computing time.

M. I. Bernstein

J140A CLEAR HSS TO ZERO

J140 is a one-card self-loading routine which clears HSS to zero and then effectively hits the load button for the next card in the primary feed.

Timing: One second

Program available as J140A.

J.C. Shaw

11-30-55

O'FLO IGN ON

J141A INVERSE TIMES VECTORS

J141 will multiply a matrix (as obtained from J139A or J143A) onto column vectors of the same order n ($1 \leq n \leq 40$). The vectors are punched one element per card in the same format as the inputs for J139A. The elements of a vector do not have to be in order on i. The matrix is in binary style D.

Each vector in read in and the product is computed and printed before the next vector is read in. In printing the product vector, the j which appeared in the last card of the vector is the j indicative printed with the product vector. j is ignored in the other vector cards.

The deck is made up as follows:

1. J141A
2. Binary matrix followed by one blank card.
3. First vector followed by one blank card.
4. Second vector " " " " "

.
.
.

k kth vector " " " " "

.
.
.

n Last vector followed by one blank card.

n+1 A "stop" card with $i > n$ and 2 blank cards.

The product vectors are printed $\left[\frac{41}{n+1}\right]$ per page where [ ] denotes the greatest integer in, and n the order of the system.

(continued)

Program stops:

NIA = $(0174)_8$ an i of the vector being read in is greater than n, or the end of the computation.

J141 uses all of HSS.

Program available as J141A.

Morton I. Bernstein

11-28-55

J142E PUNCH STYLE F BINARY CARDS

J142 will punch a block of storage from location "a" through location "b" in style F ready for reloading. C_{1-19} are left blank in the cards punched.

```
Calling sequence:   020 $̸  0 010 #  0
                    000 a    000  b
                    Control returns here.
```

Program length: 16 words (completely self-contained).

Highest symbol used: * 23

No program stops.

Program available as J142E

Timing: Punches at full speed, 100 cards per minute.

J. C. Shaw

11-30-55

OBSOLETE

J143A MATRIX ASSEMBLY O'FLO IGN ON

J143 assembles a matrix of order n $(1 \leq n \leq 40)$ into binary for use with J141A in place of the binary inverse obtained from J139A.

The inputs for J143 follow exactly the same format and rules as the inputs for J139. It is possible to assemble more than one matrix with J143, just so each matrix has the proper header card (see J139 write-up) and is followed by a blank. After each matrix is punched there is a 13X stop with NIA equal to $(0127)_8$. To assemble the next matrix, hit GO.

Program stops:

NIA = $(0055)_8$, No header card for matrix or i or j is greater than n.

NIA = $(0127)_8$, Assembly completed.

J143 uses all of HSS.

Program available as J143A.

M. I. Bernstein

J144B
J144E

DEBUGGER -- CONTINUOUS PRINTING

J144 is intended to serve as an aid in debugging programs. Its purpose is the tracing of the activities of another program, periodically printing information which the programmer may use in diagnosing his difficulties. In order to facilitate the description of J144 we will adopt certain definitions. The concepts involved correspond, loosely speaking, to the information on a P. R. card. Suppose that a JOHNNIAC instruction has just been completed.

Control Counter: The NIA diminished by one, i.e., the address in memory from which the last instruction was obtained. Note the meaning of this in the case that the last instruction was a transfer order.

Orientation Character: A symbol which is zero if the instruction just completed was a left instruction and one if it was a right instruction.

Overflow Character: A symbol which is zero if the overflow toggle is off and seven if it is on. (See Note 4)

Accumulator: The word in the accumulator.

MQ: The word in the MQ register.

Referenced Word: Roughly speaking the contents of the memory register, more precisely the contents of the cell whose address is the address part of the last instruction.

Instruction Word: If the last instruction was at the left (i.e. if the orientation is zero) this is simply the contents of the instruction register. If the last instruction was at the right this is the contents of the instruction register shifted half a word (21 bits) to the left. Note that in either case the last instruction appears at the left of the instruction word.

To use J144, have in the left half of the accumulator a transfer instruction which, if executed, would transfer control to the point in the programmer's program (hereafter called "the program") at which tracing is to begin and transfer control to the left half of the first word of J144 (3.0000). Then, insofar as the programmer need be concerned, his program will be executed just as though he had transferred to the appropriate point in his own program. This statement is true only with the qualifications in the notes below.

Note 1. The tracer will periodically print three lines as follows:

First line: an indentation, a five-character identification word (giving the cause of the print-out), the control counter, the orientation character,and the overflow character.

Second line: The instruction word and the referenced word.

Third line: The Accumulator and the MQ.

All these quantities will be with respect to the program being traced and will therefore be distinct from the control panel readings. The conditions under which a print-out will occur appear in the notes below.

Note 2. The tracer will not attempt to supervise the execution of input-output (10X) instructions. When such an instruction is encountered it will print out (see note 1), with an ID of 66666 and then transfer control to the point in the program from which the instruction came. Note that tracing will therefore stop; the program will then be executed directly; even after input-output is over.

Note 3. All control panel switches should be in the same position they would be in if the program were being directly executed. The tracer will cause a machine stop if and only if the program has a stop. The sole exceptions to the above statements are the overflow ignore switch and overflow stops; the overflow ignore switch should be on during tracing and there will be no overflow stops (see note 4). At a stop, pressing the GO button will restart the tracer. The only registers which will be correct (from the programmer's point of view) are the MQ and Accumulator. The stop will have the same cause as in the direct execution of the program.

Note 4. If an unexpected overflow occurs, i.e. if an instruction which causes an overflow is followed by an instruction other than TFL or TFR the tracer will print out (with an ID of 77777), turn off the overflow toggle and resume tracing. This feature can be suppressed by having a negative quantity in the 4th word (3.0003) of J144. This can be done either by internal programming or adding a single punch to the deck on which J144 is loaded. If this is done the overflow character will always appear as zero on print-outs.

Note 5. In addition to the print-outs mentioned above, there will also be a print-out (ID = 00000) after the completion of every instruction. The programmer has the option of suppressing some of these print outs. This can be done by changing the left address of 3.0001 to the address of an entry word to a routine which selects the conditions under which print-out is to occur. This routine should transfer to the left of 3.0002 if a print-out is desired, and to the right of 3.0002 if no print-out is desired.

Region 3. $230_8 = 152_{10}$ words.

Program stops - See Note 3.

Program available as J144B
J144E

Norman Shapiro

J145B
J145E DEBUGGER -- PRINT ON TRANSFER

The write-up for J145 is the same as that for J144 up to but not including Note 5.

Note 5. In addition to the print-outs mentioned above, there will be a print-out (ID = 00000) after the completion of every transfer.

Region 3 - $226_8 = 150_{10}$ words.
Program stops - See Note 3.

Program available as J145B
J145E

Norman Shapiro

12-8-55

J146B
J146E

DEBUGGER -- BREAKPOINT PRINTING

The write-up for J146 is the same as that for J144 up to but not including Note 5.

<u>Note 5</u>. In addition to the print-outs mentioned above there will be a print-out (ID = 00000) after the completion of every left instruction such that the word from which the instruction was obtained has a 1 in the 2^{-19} bit (the left-most of the two unused bits).

Region 3- $227_8 = 151_{10}$ words
Program Stops. See note 3.

Program available as: J146B
J146E

Norman Shapiro

12-8-55

J147B
J147E

DEBUGGER -- SELECTIVE CELL PRINTING

The write-up for J147 is the same as that for J144 up to but not including Note 5.

Note 5. In addition to the print-outs mentioned above there will be a print-out after the completion of every non-shift instruction (left or right) whose address part coincides with one of a specified list of addresses. This list is specified as follows:

Θ	000	A_1	000	0000
$\Theta+1$	⋮			
$\Theta+n$	000	A_n	000	0000
$\Theta+n+1$	100	0000	000	0000

where the specified addresses are $A_1, \ldots, A_n$. At the time of the original entry to the tracer (in 3.0000) the MQ should have Θ in the left address.

Region 3 - $241_8 = 161_{10}$ words

Program stops - See note 3.

Program available as: J147B
J147E

Norman Shapiro

J148E PUNCH STYLE F WITH I.D.

J148 will punch a block of words in style F from location "a" through location "b". Identification specified in the calling sequence is punched in C_{1-5} of each card and a sequence number 001(1)n is punched in C_{6-8}.

Calling Sequence:

α	020	$	0	010	# 0
$\alpha+1$			a		b
$\alpha+2$					x_1
$\alpha+3$					x_2
$\alpha+4$					x_3
$\alpha+5$					x_4
$\alpha+6$					x_5
$\alpha+7$	control returns here				

where x_1 x_2 x_3 x_4 x_5 is the identification in the Easy Fox character code.

Program length: Main program - 104 words
Region A (erasable) - 15 words

Highest symbol used: * 50

Timing: Punches at full speed, 100 cards per minute.

Program available as: J148E

M. Shapiro

J149B
J149E JOHNNIAC BLOCK TRANSFER TO AND FROM DRUM WITH TALLY

J149 is a "fancy" version of J120. The calling sequence is as follows:

α	RA	α	TRL	XXX (Arbitrary Symbol)
α+1	Op.	β	OpO	γ
α+2	000	f	000	ℓ
α+3	Controls returns either left or right. (See below.)			

where Op = {WD if transferring from HSS to drum; RD if transferring from drum to HSS

β	=	first HSS loc.
γ	=	last HSS loc.
p	=	drum position (0, 1, or 2)
f	=	first drum address of position p
ℓ	=	last drum address of position p allowable.

J149 does not assume $\gamma \geq \beta$. However, if $\beta > \gamma$, then $f > \ell$ must also hold. In this case, the interval $[\gamma, \beta]$ in HSS is stored on the drum so that contents of β go to f, contents β-1 go to f-1, etc.

The purpose of ℓ is to provide a check against writing over good information or going past the end of the position. If the interval $[f,\ell]$ (or $[\ell, f]$) is smaller than $[\beta,\gamma]$ (or $[\gamma, \beta]$), then reading or writing will not occur and control returns to the left command of α+3. Otherwise, control returns to the right command of α+3 with the left address part of the accumulator containing the next available drum address, f*, i.e.

$$f^* = f + (\gamma-\beta)+1 \text{ if } \gamma \geq \beta$$
$$= f + (\gamma-\beta)-1 \text{ if } \beta > \gamma.$$

J150B
J150F

JOHNNIAC FLOATING-POINT INTERPRETIVE SYSTEM

Program Storage: 2960_{10} - 4095_{10} (5620_8 - 7777_8).

All of the Floating Point Operations are recognized in their mnemonic forms by J100, J118.

LIST OF FREQUENTLY USED LOCATIONS

Dec.	Octal	Contents
2960	5620	First Location of Interpreter
2963	5623	020 [I CTR.] 050 5644
2980	5644	I Register
4041	7711	Error Halt Location
3040	5740	Trap Register - BRKPT
3041	5741	" " - All Orders
3042	5742	" " - Transfer
3656	7110	AMQ Exponent
3658	7112	AMQ Mantissa
3708	7174	072 [q_1] 025 7016 - Square Root Shift*
3733	7225	022 7122 076 [q_2] - Series Shift*
3999	7637	Series Test Number*
3064	5770	Index Register A
3056	5760	" " B
3052	5754	" " C
3050	5752	" " D
3049	5751	" " E
3048	5750	" " F
3597	7015	Integer - Zero
3598	7016	" - One
⋮	⋮	⋮ - ⋮
3606	7026	Integer - Nine
3615	7037	" - Fifty
3634	7062	" - Ten exp zero
3635	7063	" - " " One
⋮	⋮	⋮ - ⋮ ⋮ ⋮
3643	7073	Integer - Ten exp. Nine

Program available as: J150B
J150F

* Let $\Delta_n = f_{(n+1)} - f_n$, where f_n is the nth approximation to

the function f. The test for convergence of f_{n+1} to f is then made to depend upon the sign of $|\Delta_n| \cdot 2^{-q} - Z$. Z is always 2^{-39} for the SQR operation. For the ART, EXP, and LOG operations Z is the series test number. The series test number is ordinarily equal to 2^{-39} for these three operations, but it can be modified by the user. However, the series test number is usually reset to 2^{-39} upon execution of a SIN or COS operation. For the SQR operation q is q_1 and q_1 is ordinarily set to 5. However, q_1 can be modified by the user. The same situation holds for the SIN, COS, ART, LOG, and EXP operations with q equal to q_2.

John Derr

J151B
J151E SINE-COSINE

The four trigonometric functions developed and their locations are:

$\frac{\sin y}{2}$ in location 1.001

$\frac{\cos y}{2}$ in location 1.002

$\frac{\sin 2y}{2}$ in location 1.000 and in MQ

$\frac{\cos 2y}{2}$ in Accumulator

Range: $-1 \leq y < 1$ (in radians)

Entered with y in MQ

Calling sequence:

α	020	α	010	XXX
α+1	Control returns here.			

Region 3 - 32 words

Region 1 (erasable) - 3 words

Program available as J151B
J151E

M. Shapiro

J152B
J152E DEBUGGER -- BREAKPOINT PRINTING

The write-up for J152 is the same as that for J144 up to but not including Note 5.

Note 5. In addition to the print-outs mentioned above there will be a print-out (ID = 00000) after the completion of every left instruction such that the word from which the instruction was obtained has a 1 in the 2^{-20} bit (the right-most of the two unused bits).

Region 3 - $227_8 = 151_{10}$ words
Program Stops. See note 3.

Program available as J152B
J152E

Norman Shapiro

12-29-55

J153A   CLEAR HSS TO PRESCRIBED WORD

J153A is a one-card self-loading routine which clears the HSS to the word placed in the right half of the 12's row and then effectively hits the load button for the next card in the primary feed.

Its principal suggested use is to clear the HSS to: 130 0000 130 0000. If this is done and a program error causes transfer to a word not previously affected by the program, there will be a halt and no information which may be useful for debugging purposes will be destroyed.

Program available as J153A.

Norman Shapiro

J154B
J154E

SINE - COSINE ROUTINE FOR Y IN RADIANS/π

Depending upon the argument y supplied in MQ, this routine develops $\frac{1}{2}\sin\theta$ or $\frac{1}{2}\cos\theta$ in A (θ in radians).

Argument	Range	Function Developed
$y = \theta/\pi$	$-1 \le y < 1$	$\frac{1}{2}\sin\theta$
$y = \frac{1}{2} \pm \frac{\theta}{\pi}$	$-1 \le y < 1$	$\frac{1}{2}\cos\theta$

If $\frac{\theta}{\pi}$ or $\frac{1}{2} \pm \frac{\theta}{\pi}$ overflows out of the 2^0 position from left shifts or additions, the result is unaffected because $\sin(\theta + 2n\pi) = \sin\theta$.

Accuracy: Error less than 3×2^{-39}, in particular:

y	|error|
0	0×2^{-39}
$\frac{1}{2}$	0×2^{-39}
$-\frac{1}{2}$	0×2^{-39}
-1	0×2^{-39}
1/6	0×2^{-39}
5/6	1×2^{-39}
-1/6	1×2^{-39}
-5/6	1×2^{-39}

Duration: Approximately 15 milliseconds maximum.

Entered with y in MQ.

Calling Sequence:

```
α       020α   010   XXX
α+1     Control returns here
```

Region 3: 25 words
Region 1: 3 words (erasable)

Program available as J154B
J154E

Note: This program was developed from the University of Illinois Code T5-157 prepared for the ILLIAC.

Nancy Brooks

1-19-56

J155E PRINT INSTRUCTIONS WITH DECIMAL LOCATIONS, OCTAL OPERATIONS, DECIMAL ADDRESSES

J155 is a closed subroutine which prints the contents of a block of storage from "a" through "b" ($b \geq a$) as instructions. The instructions are printed one word per line with octal operations and decimal address parts. The location of each word is printed on the left in decimal. A block of one or more zero words is indicated by a single blank line. The printing rate is 900 lines per minute except when passing over zeros.

```
Calling Sequence:     020  $ 0  010  #  0
                      000    a  000     b
                      Return
```

Highest symbol used: * 57

Storage requirements: 104 words

Program available: J155E

J. C. Shaw

J156B
J156E

INTEGRAL ROOT $x^{1/p}$

This routine computes the p th root (p, a positive integer, $1 \leq p \leq 4095$) of a 39 binary digit real argument x, $-1 \leq x < 1$. If p is even and x is positive, the positive root is found.

Special cases:

	x	p	root	Time required
1.	0	$0 \leq p \leq 4095$	0	0.5 ms.
2.	Negative	even	halt, 0 if GO	
3.	-1	odd	-1	1.6 ms.
4.	$\lvert x \rvert < 1$	1	x	x neg. 2.0 ms. x pos. 1.0 ms.
5.	$\lvert x \rvert < 1$	0	halt; 0 if GO	
6.	$\lvert x \rvert > 1-px2^{-39}$	$p > 1$	$\pm(1-2^{-39})$ with sign of x	x neg. 2.3 ms. x pos. 1.4 ms.

Program stops: 3.017 p equal to zero. (If GO is pressed, root is set to zero.)

3.029 p even and x negative. (If GO is pressed, root is set to zero.)

Calling Sequence: x in MQ

```
α      020  α  010  XXX
α+1                 p
α+2    Control returns here
```

Exits with $x^{1/p}$ in A.

Accuracy: $|\text{error}| \leq 2^{-39}$

Approximate Time:

For the special cases, the time required is given above. In general, the time required varies directly with p and indirectly with x. Some typical times in milliseconds are given in the following table:

p	x = 0.1	0.2	0.3	0.5	0.8
2	35	30	30	30	25
3	45	45	40	40	35
4	55	50	50	45	40
10	130	125	120	105	90

For large values of p and small values of x the time increases rapidly.

Program length: 33 words in region 3.
4 words in region 1.

Note: This routine was adapted from Illinois Code R2-105 as programmed for the ILLIAC.

Nancy Brooks

J157E ZEROS OF A POLYNOMIAL WITH COMPLEX COEFFICIENTS -- FLOATING POINT

J157 will evaluate all of the roots of an nth degree polynomial equation of the form

$$\sum_{i=0}^{n} C_i Z^i = 0,$$

where C_i is a complex number for $i = 0, \ldots, n$ and $n < 50$. Each root is computed as a complex number and, furtherfore, the norm of the remainder obtained by dividing the reduced polynomial by the computed zero is supplied for each zero evaluated. J157 must be used in conjunction with the Floating Point Interpretive System.

J157 is entered by basic linkage. Before the calling sequence is executed, however, the coefficients of the given polynomial must be in consecutive H. S. S. locations as follows:

C_n^R = real component of C_n in floating point internal form

C_n^I = imaginary " " " " " " " "

$\vdots$

C_o^R = real component of C_o in floating point internal form

C_o^I = imaginary " " " " " " " "

Now if $C_n = 0$, let N be such that $C_N \neq 0$ and $C_{N+1} = C_{N+2} = \cdots = C_n = 0$.

<u>Calling Sequence</u>:

α	020	\$ 0	010	# 0	Link to J157
α+1	000	$L(C_n^R)$	n	L(N)	
α+2	Control returns here				

Let Ω_j be the j^{th} root of $f(Z) = \sum_{i=0}^{N} C_i Z^i = 0$. Upon execution of the basic link, J157 will proceed to find the zeros $(\Omega_1, \ldots, \Omega_N)$ of the reduced polynomials $g(Z)$ of $f(Z)$, one at a time, except in the case when Ω is a zero of K^{th} order multiplicity. (In the latter event Ω is evaluated only one time, but it is counted as K roots with respect to the output of J157). The roots Ω_j, together with $|g(\Omega_j{}^R)|+|g(\Omega_j{}^I)|$, are stored in consecutive H.S.S. locations beginning with L(N)+1 and ending with L(N)+3N as follows:

L(N)+1	$\Omega_1{}^R$	$\Omega_1{}^I$	$\left\|f(\Omega_1{}^R)\right\|+\left\|f(\Omega_1{}^I)\right\|$
L(N)+4	$\Omega_2{}^R$	$\Omega_2{}^I$	$\left\|g(\Omega_2{}^R)\right\|+\left\|g(\Omega_2{}^I)\right\|$
$\vdots$	$\vdots$	$\vdots$	$\vdots$
L(N)+3N-2	$\Omega_N{}^R$	$\Omega_N{}^I$	$\left\|g(\Omega_N{}^R)\right\|+\left\|g(\Omega_N{}^I)\right\|$

N is a fixed-point integer. All other numbers are in the floating-point internal form.

<u>Description of the Method</u>

The basic process used is Newton's method, i.e. if Z_n is the n^{th} approximation to a root Ω, then

$$Z_{n+1} = Z_n - \frac{g(Z_n)}{g'(Z_n)} .$$

The criteria for convergence of Z_n to Ω requires that

$$\left|Z_{n+1} - Z_n\right| = \left|\Delta Z_n\right| = \left|\frac{g(Z_n)}{g'(Z_n)}\right|$$ satisfy the inequalities

$|\Delta Z_n| \leq |Z_n| \cdot 10^{-7}$ and $|\Delta Z_n^R| \leq |Z_n^R| \cdot 10^{-7}$.* Hence $|g(\Omega^R)|+|g(\Omega^I)|$ yields an independent check on the accuracy of the calculation of Ω. Note that this latter check involves an absolute error with respect to the polynomial, whereas the convergence criteria is based on a relative error with respect to Z_n.

In the case of a repeated root Ω of degree K, $K > 1$, Newton's method is rendered inadequate on two accounts. The convergence becomes linear rather than quadratic and the number of good significant digits in Z_n is reduced by a factor of at least two since $g'(Z_n) \rightarrow 0$ as $Z_n \rightarrow \Omega$. As a result, $g(Z_n)/g'(Z_n)$ becomes the ratio of two relatively small numbers. Here, the basic process is modified as follows: Let L be the first integer, $L > 1$, such that $|g^{L+1}(Z_n)| \geq Q \cdot \epsilon$ (Here we will omit the details concerning the explicit identity of Q and ϵ. Let it suffice to say that the test is relative versus absolute and that $g^{L+1}(Z_n)$ is bounded away from zero.) Now

$$Z_{n+1} = Z_n - \frac{(K-L)\, g^{L}(Z_n)}{g^{L+1}(Z_n)} \quad \text{and } \Delta Z_n = - \frac{(K-L)\, g^{L}(Z_n)}{g^{L+1}(Z_n)} .$$

Using this new form of ΔZ_n the test for convergence is the same as described in the preceding paragraph. Also the modified process is still at least a second order process.

Convergence and the Choice of Z

When the computation for any root Ω begins Z_0 is computed either as a function of C_0 or as the conjugate of the immediately

*The second inequality is designed to cover the situation where the imaginary component of Ω can be much larger than the real component of Ω and the real component is the only one desired. Problems exist in the field of dynamic analysis, for example, where this type of convergence is desired.

preceding root. The iteration will then proceed to compute Z_{n+1} as indicated above until either Z_{n+1} converges to Ω or $n = 13$. In the former case (the ordinary case), we are through. In the latter event a new Z_o is computed as a different function of C_o. This time n is allowed to reach $13 + 10 = 23$ before a new Z_o is computed. This process can be repeated until the 7th value of Z_o has been computed. If convergence has not been achieved when $n = 13 + 6 \cdot 10 = 73$, then the program will halt.

In order to give the user some insight into the reasons for selecting such an elaborate system for computing Z_o, we might outline some of the pathological cases which might arise and do not fit into the standard convergence pattern. We first distinguish two fundamentally different causes for abnormal convergence.

I. Newton's method is not, precisely speaking, an "error-squaring" process. In fact, it need not even be convergent at all. Basically $(\Delta Z)_{n+1} \doteqdot q \cdot (\Delta Z)_n^2$. Convergence is guaranteed if $|q \cdot (\Delta Z)_n| < 1$, but the rate of convergence still depends directly on q. If $|q| \doteqdot 1$, the convergence might be very slow. As an example of non-convergence consider the case where $g'(Zn) \leq Q_1 \cdot \epsilon_1$ and $g(Z_n) \geq Q_2 \cdot \epsilon_2$, i.e. Z_n is very near a root of $g'(Z) = 0$ which is not a root of $g(Z) = 0$.

II. Cycling can occur in this method due to two diverse causes. (By cycling we mean that a finite sequence $(Z_{\alpha_1}, Z_{\alpha_2}, \ldots, Z_{\alpha_m})$ of root approximation values are taken on in a cyclic manner.)

A. The propagated error which accumulates in the calculation of $g(Z_n)$ and $g'(Z_n)$ is so great that the convergence criteria cannot be met. This sort of cycling becomes more probable as the degree N of $g(Z)$ increases.

B. A geometrical symmetry exists among the roots of $g(Z)$ relative to the approximation value Z_n.

<u>Notes</u>

(1) A Halt (130 operation) will occur if the process does not converge. If the GO button is pressed, Z_{n+1} will be placed in the root output storage and the program will proceed to evaluate the roots of the reduced polynomial.

(2) The roots are usually in error only in the low order two digits of the mantissa. The accuracy of a root varies, naturally, with the degree of the original polynomial and the position of the root in the sequence $(\Omega_1, \ldots, \Omega_N)$.

(3) The evaluation of the nth roots of unity is probably more difficult than the "average" polynomial occurring in practice due to the geometrical symmetry of the roots. The following approximate times were recorded while operating in the SD Mode:

$X^3 - 1$	13 sec.
$X^5 - 1$	18 sec.
$X^8 - 1$	1 min. 45 sec.
$X^9 - 1$	45 sec.
$X^{16} - 1$	8 min. 50 sec.
$X^{17} - 1$	4 min. 20 sec.

Reg. A 332_{10} words (A 0 - A 331)*

Symbolic Code: $422_{10} = 646_8$ words

Highest symbol used: * 99

J150F $5620_8 - 7777_8$ (Current version of Floating Point Interpretive System)

Program available as J157E.

John I. Derr

*A132-A331 are available to the user for storing coefficients for input to J157, i.e. it is permissible to have $L(C_n^R) = A132$. However, these coefficients will not be available upon reentry to the main program.

J158A AUTOMATIC COMPUTATION OF THE ZEROS OF A POLYNOMIAL WITH COMPLEX COEFFICIENTS -- FLOATING POINT

J158A is an automatic version of J157. The user needs only to record the necessary input data in punched cards, place these cards immediately behind J158A, and press the LOAD button. The roots of the given polynomials will be computed one root at a time and all of the roots for one polynomial will be printed after the last root has been computed.

<u>Input Form</u>

Information should be punched in the Floating Point Data Form as follows:

<u>1st card for a problem</u>

```
      (1)               (2)               (3)                   (6)
± EXP MANTISSA    ± EXP MANTISSA    ± EXP MANTISSA  ...  ± EXP MANTISSA
|   n     ID         ---    BLANK            ↑
|   |     |                            COEFFICIENTS
|   |     |__ Any decimal              BEGIN  HERE
|   |         information
|   |        for problem ID
|   |
|   |__ order of polynomial
|
|__[+  if all coefficients are real
   [-  if any coefficient is not real
```

The coefficients are punched into consecutive positions of the cards beginning with position (3) of the first card and continuing into the succeeding cards beginning with position (1). The last card for each problem <u>must</u> contain a 12-punch (+) in column 80. If the sign position of the word punched in position (1) of the first card is (+) or blank, then only the real coefficients are to be punched and these <u>must</u> be in consecutive positions (no gaps). Otherwise, the real and imaginary components of <u>each</u> coefficient must be punched.

The roots for a sequence of polynomials can be evaluated by placing the input cards for the polynomials in sequential order behind J158A. Recall that the last card of each problem must contain a (+) in column 80. If the input cards are followed by two blank cards, the collator will be on Select following completion of the last problem.

<u>Output Form</u>

The page will be ejected prior to the beginning of the first problem. One space will be skipped before the printing for each problem begins. The numbers printed will be in the external floating-point form. The print format is that described for the 013 (PNT) order in the Floating-Point write-up. The printing for one problem takes place as follows:

	A	B	C
space - - - - - - - - - - - - - -	——	——	————
Information in position (1) of 1st Input Card } - - -	——	——	$\pm$ n ID
Roots begin here - - - - - - - -	Ω_1^R	Ω_1^I	$\lvert f(\Omega_1^R)\rvert + \lvert f(\Omega_1^I)\rvert$
	⋮	⋮	⋮
Last Root - - - - - - - - - - -	Ω_N^R	Ω_N^I	$\lvert g(\Omega_N^R)\rvert + \lvert g(\Omega_N^I)\rvert$

<u>Notes</u>:

(1) The program will halt at 1012_8 if n exceeds 49. If the GO button is pressed, the program will eject the page and begin the computation of the next problem.

Notes (cont.)

(2) A programmed halt at 1370_8 indicates that the iteration has not converged. If the GO button is pressed, the current root approximation will be placed in the root output storage for printing and the program will proceed to evaluate the roots of the reduced polynomial.

(3) If $N < n$, only N roots will be printed.

(4) The user is referred to the J157 write-up for a description of the method.

Program available as J158A

John I. Derr

J159E INVERSE INTERPOLATION, A REAL ROOT OF $f(x) = 0$

This routine searches for the real root x in the specified interval $x_1 < x < x_2$ of a function $f(x) = 0$. An auxiliary closed subroutine must be provided which, when entered with x in the MQ with the following calling sequence:

```
γ     020   γ  010   β
γ+1   Control return
```

returns control to the left order in $\gamma+1$ with $f(x)$ in A.

Calling sequence:

x_1 in B 0

x_2 in B 1

```
α     020   α   010 δ
α+1         β   Control return with  x  in  A
```

where β is the location of the auxiliary routine.

Restrictions:

1. If $f(x_1)$ and $f(x_2)$ are of the same sign or if either is zero, no root is found. Such a situation is indicated by a control return with $x = -1$ in A.
2. In the interval $x_1 \leq x \leq x_2$ $f(x)$ must be continuous and in the range $-1 \leq f(x) < 1$.

Range: $-1 < x < 1$

Method: If $(x_1)_n$ and $(x_2)_n$ are two arguments of $f(x)$ such that $f(x_1)_n$ and $f(x_2)_n$ are of opposite sign, linear interpolation is used to find the next approximation to the root x_{n+1}. The sign of $f(x_{n+1})$ is then compared with the signs of $f(x_1)_n$ and $f(x_2)_n$ to determine the interval between which the root x lies. If

$(x_1)_n < x < x_{n+1}$, then $(x_2)_n$ is replaced by x_{n+1}, $f(x_2)_n$ is replaced by $f(x_{n+1})$, and $f(x_1)_n$ is replaced by $1/2 f(x_1)_n$. If, however, $x_{n+1} < x < (x_2)_n$, then $(x_1)_n$ is replaced by x_{n+1}, $f(x_1)_n$ is replaced by $f(x_{n+1})$, and $f(x_2)_n$ is replaced by $1/2\ f(x_2)_n$. The process is repeated until either

$$\left|1/2\ f(x_{n+1})\right| \leq \epsilon_1 \quad \text{or} \quad \left|(x_1)_n - (x_2)_n\right| \leq \epsilon_2$$

where ϵ_1 and ϵ_2 are located in * 11 and * 12 respectively. In the library copy $\epsilon_1 = 2^{-39}$ and $\epsilon_2 = 2^{-38}$.

Accuracy: for the root x ; ϵ_2

for the function $f(x) = 0$; $2\epsilon_1$

Duration: $[2\ t + 3 + n(t-6.5)]$ ms where n = number of iterative linear interpolations required to find root, and t = duration in ms of auxiliary routine which computes f(x).

Symbolic Code: 38 words

Region B: 7 words (must be preserved by auxiliary routine; otherwise erasable).

Highest symbol used: * 13

Program available as J159E.

Nancy Brooks

J160E CARD TO MOUSE MATRIX(1)

J160 encodes 80 columns of a card into Mouse matrix form. Columns 1-40 are encoded into cells B0, ..., B6 and columns 41-80 are encoded into C0, ..., C6. Each of the 80 columns may contain any 026 keypunch character.

Before entering J160 an appropriate select order must be given (100 0000 to read a card from the primary feed or 100 0001 to read a card from the secondary feed). J160 must be entered within 75 milliseconds of the select order or a late copy will result(2). To read cards at full speed, the select order must be given within 3.6 milliseconds of the exit from J160.

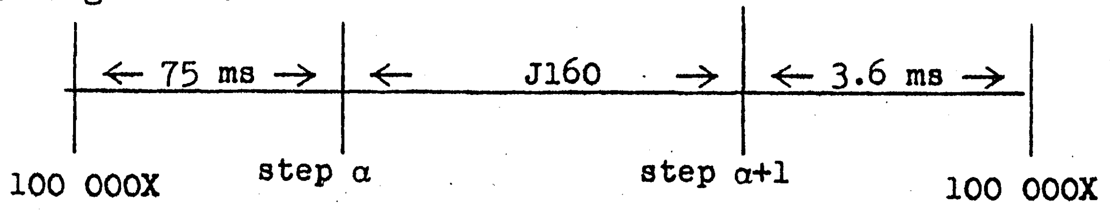
Calling sequence:

```
        α      020   α   010   β
        α+1    Control returns here
```

Highest symbol used: * 1

Program will stop at the left of word 2 if there is a late copy.

Timing chart:



A late copy will result if the allowed time between the 100 000X order and step α is exceeded. The reader will run at $\frac{1}{k}$ speed if the time used between step α+1 and the 100 000X order is less than 3.6 + 250(k-1) and greater than 3.6 + 250(k-2) milliseconds.

Program length:

27 words

Region A (erasable) - 4 words, A3 to A6

Region B - 7 words, B0 to B6

Region C - 7 words, C0 to C6

Available as J160E

M. B. Shapiro and N. Z. Shapiro

(1) For an explanation of the intended use of J160, see JOHNNIAC Note No. 42

(2) These times are based on 84 μs for an 02X operation and 89 μs for an 06X operation

J161E CARD TO MOUSE MATRIX WITH INTERMEDIATE COMPUTATION(1)

J161 encodes 80 columns of a card into Mouse matrix form. Columns 1-40 are encoded into B0 - B6 and columns 41-80 are encoded into C0 - C6. Each of the 80 columns may contain any 026 keypunch character.

J161 must be entered 12 times in order to complete the reading and encoding of a single card. The time between successive exits from J161 and succeeding entries is available to the programmer. Before the first entry to J161 an appropriate select order must be given (100 0000 to read a card from the primary feed and 100 0001 to read a card from the secondary feed). If too much time is taken between the select order and the first entry or between successive exits and succeeding entries a late copy will result. (See timing chart below for detailed times.)(2) To read cards at full speed, the select order must be given within 3 milliseconds of the 12th exit from J161.

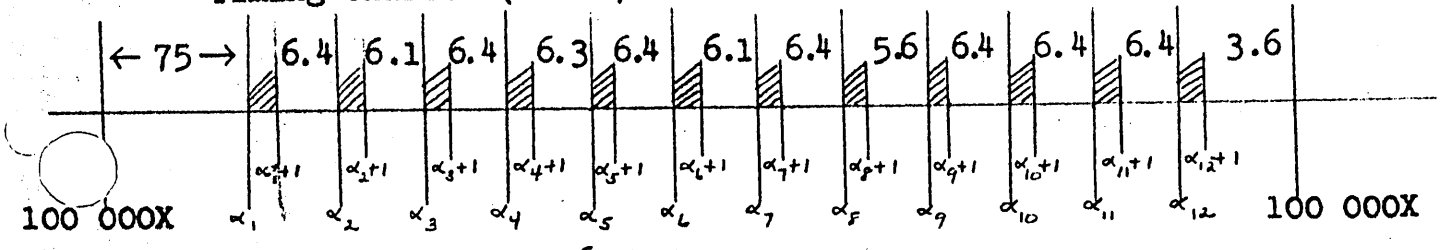
Calling sequence:

$$\left.\begin{array}{llll} \alpha_i & 020 \quad \alpha & 010 & \beta \\ \alpha_i+1 & \text{Control returns here} & & \end{array}\right\} \begin{array}{c} \text{12 times for each card} \\ i = 1, \ldots, 12 \end{array}$$

Highest symbol used: * 3

Program will stop at the left of word 2 if there is a late copy.

Timing chart: (in ms)



Shaded areas are J161.

A late copy will result if the allowed time between the 100 000X order and step α_1 or between steps α_i+1 and α_{i+1} ($i = 1, \ldots, 11$) is exceeded. The reader will run at $\frac{1}{k}$ speed if the time used between step $\alpha_{12}+1$ and the 100 000X order is less than $3.6 + 250(k-1)$ and greater than $3.6 + 250(k-2)$ milliseconds.

Program length:

40 words

Region A (erasable) - 4 words, A3 to A6

Region B - 7 words, B0 - B6

Region C - 7 words, C0 - C6

Available as J161E

M. B. Shapiro and N. Z. Shapiro

(1) For an explanation of the intended use of J161, see JOHNNIAC Note No. 42

(2) These times are based on 84 μs for an 02X operation and 89 μs for an 06X operation.

4-9-56

J162E MOUSE MATRIX TO H.S.S.(1)

J162 forms

$$(10^n y+10^{n-1}x_1+\ldots+10^{n-i}x_i+\ldots+10x_{n-1}+x_n)2^{-39}$$

where n is a positive integer specified by the calling sequence, A 0 initially contains $y\cdot 2^{-39}$ and $x_1, \ldots, x_n$ are integers formed from the numeric parts of the left-most n columns of the Mouse matrix in B0, ..., B6. The result is placed in A 0 and the entire matrix is shifted left n places.

Calling sequence:

```
α      020   α   010    β
α+1          n·2^-39
α+2    Control returns here.
```

Highest symbol used: * 3

Program will stop at the right of words 11 or 12 because of an overflow if A 0 initially contains a number which will cause the polynomial to overflow at some step in the accumulation.

Timing:(2) (.20 + 2.10n) ms.

Program length:

21 words

Region A (erasable) - 3 words, A0 to A2

Region B - 7 words, B0 to B6

Available as J162E

M. B. Shapiro and N. Z. Shapiro

(1) For an explanation of the intended use of J162, see JOHNNIAC Note No. 42

(2) These times are based on 84 μs for an 02X operation and 89 μs for an 06X operation.

J163E H.S.S. TO MOUSE MATRIX(1)

J163 shifts the Mouse matrix in B0, ..., B6 to the left n places, placing x_1, x_2, ..., x_n in the right-most n columns. The integers x_1, ... x_n are defined by the recursion:

x_i = Integral part of $10|y_i|$, $0 \le i \le n$

y_o = Initial contents of A0

$y_{i+1} = 10|y_i| - x_i \quad 0 \le i \le n$

At exit, y_{n+1} will be in A0.

Any x_i which equal zero enter the matrix as blanks.(2)

Calling sequence:

```
α      020   α   010   β
α+1    n·2^-39
α+2    Control returns here
```

Highest symbol used: * 2

No program stops.

Timing:(3) (.10 + 2.10n) ms.

Program length:

23 words

Region A (erasable) - 3 words, A0 to A2

Region B - 7 words, B0 to B6

Available as J163E

M. B. Shapiro and N. Z. Shapiro

(1) For an explanation of the intended use of J163 see JOHNNIAC Note No. 42.

(2) The routine for inserting zeros is listed in JOHNNIAC Note No. 42.

(3) These times are based on 84 μs for an 02X operation and 89 μs for an 06X operation.

J164E MOUSE MATRIX TO PUNCH(1)

J164 punches 80 columns corresponding to the 80 columns of characters encoded in Mouse matrix form. Columns 1-40 are punched from B0 - B6 and columns 41-80 are punched from C0 - C6. Each of the 80 columns may contain any 026 keypunch character.

Before entering J164 a 100 0002 order must be given. J164 must be entered within 42 milliseconds(2) of the select order or a late copy will result. To read cards at full speed the select order must be given within 17.7 milliseconds of the exit from J164.

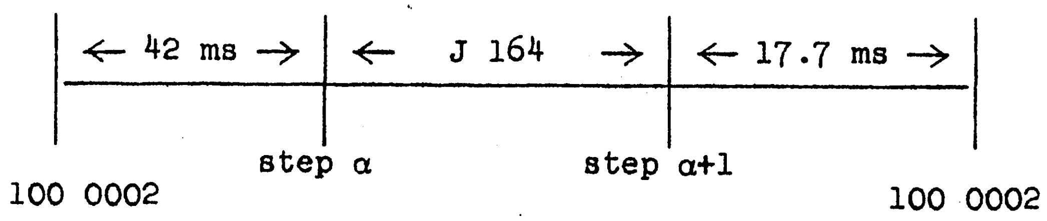
Calling sequence:

```
        α      020   α    010   β
        α+1    Control returns here
```

Highest symbol used: * 2

Program will stop at the right of word 6 if there is a late copy.

Timing chart:



A late copy will result if the allowed time between the 100 0002 order and step α is exceeded. The punch will run at $\frac{1}{k}$ speed if the time used between step α+1 and the 100 0002 order is less than 17.7 + 600 (k-1) and greater than 17.7 + 600 (k-2) milliseconds.

J165E MOUSE MATRIX TO PUNCH WITH INTERMEDIATE COMPUTATION(1)

J165 punches 80 columns corresponding to the 80 columns of characters encoded in Mouse matrix form. Columns 1-40 are punched from the matrix in B0, ..., B6 and columns 41-80 are punched from the matrix in C0, ..., C6. Each of the 80 columns may contain any 026 keypunch character. J165 must be entered 12 times in order to complete the punching of a single card. The time between successive exits from J165 and succeeding entries is available to the programmer. Before the first entry to J165, a 100 0002 order must be given. If too much time is taken between the select order and the first entry or between successive exits and succeeding entries, a late copy will result. (See timing chart below for detailed times.)(2) To punch cards at full speed the select order must be given within 17.2 milliseconds of the 12th exit from J165.

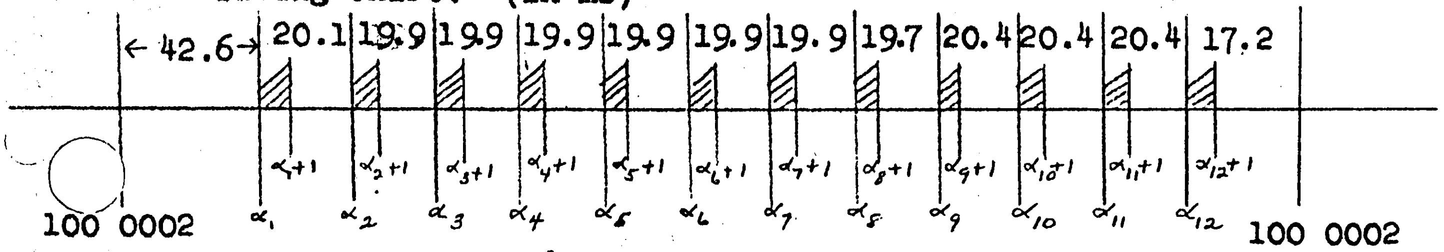
Calling sequence:

α_i	020 α 010 β	12 times for each card
α_i+1	Control returns here	(i=1, ..., 12)

Highest symbol used: * 3

Program will stop at the left of one of the following words if there is a late copy: 8, 12, 17, 22, 27,32, 37, 42, 48, 50, 52, or 54.

Timing chart: (in ms)



Shaded area is J165.

A late copy will result if the allowed time between the 100 0002 order and step α, or between steps α_i+1 and α_{i+1} ($i = 1, \ldots, 11$) is exceeded. The punch will run at $\frac{1}{k}$ speed if the time used between step $\alpha_{12}+1$ and the 100 0002 order is less than 17.2 + 600 (k-1) and greater than 17.2 + 600 (k-2) milliseconds.

Program length:

56 words

Region A (erasable) - 3 words, A3 to A5

Region B - 7 words, B0 to B6

Region C - 7 words, C0 to C6

Available as J165E

M. B. Shapiro and N. Z. Shapiro

(1) For an explanation of the intended use of J165 see JOHNNIAC Note No. 42

(2) These times are based on 84 μs for an 02X operation and 89 μs for an 06X operation.

J166E MOUSE MATRIX TO PUNCH AND ECHO CHECK(1)

J166 punches and echo checks 80 columns corresponding to the 80 columns of characters encoded in Mouse matrix form. Columns 1-40 are punched from the matrix in B0, ..., B6 and columns 41-80 are punched from the matrix in C0, ..., C6. Each of the 80 columns may contain any 026 keypunch character.

As one card is being punched, the previous card is being echo checked. Region D contains the binary image of the card being echo checked. The last (absolutely) card punched by J166 can be echo-checked only by entering J166 again and punching an extra (blank) card. If it is desired to suppress echo checking, the left operation of the word in * 0(2) must be made 014. If this is done, the next use of J166 will not cause an echo check stop (although the select order may be either 100 0002 or 100 0003). If there is no further alteration of * 0, there will be echo checking on subsequent cards.

Roughly speaking, once J166 has been entered, region 0 ? cannot be used until the final exit from J166. The exception to this statement occurs when punching with echo check suppression.(3)

Before entering J166, a 100 0003 order must be given. The entry to J166 must be completed within 40.4 milliseconds(4) of the select order or a late copy will result. To read cards at full speed the select order must be given within 17.0 milliseconds of the exit from J166.

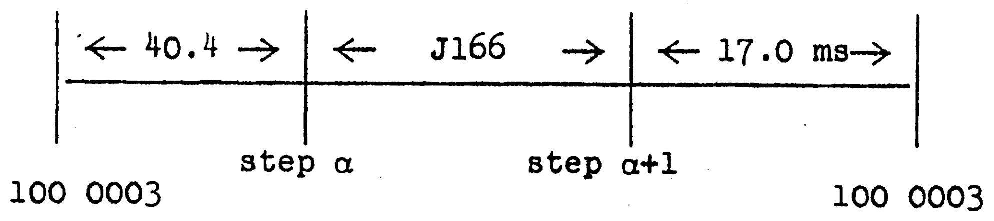
Calling sequence:

```
        α      020   α   010   β
        α+1    Control returns here
```

Highest symbol used: * 8

Program will stop at the right of word 74 (* 0) if there is an echo check (there will be no way of proceeding) or at the right of word 68 if there is a late copy.

Timing chart:



A late copy will result if the allowed time between the 100 0003 order and step α is exceeded. The punch will run at $\frac{1}{k}$ speed if the time used between step α+1 and 100 0003 order is less than 17 + 600 (k-1) and greater than 17 + 600 (k-2) milliseconds.

Program length:

76 words

Region A (erasable) - 6 words, A3 to A8

Region B - 7 words, B0 to B6

Region C - 7 words, C0 to C6

Region D (semi-erasable) - 24 words, D0 to D23

Available as J166E

M. B. Shapiro and N. Z. Shapiro

(1) For an explanation of the intended use of J166, see JOHNNIAC Note No. 42.

(2) * 0 is the 74th word of J166E.

(3) See JOHNNIAC Note No. 42.

(4) These times are based on 84 μs for an 02X operation and 89 μs for an 06X operation.

4-9-56

J167E MOUSE MATRIX TO PUNCH AND ECHO CHECK WITH INTERMEDIATE COMPUTATION(1)

J167 punches and echo checks 80 columns corresponding to the 80 columns of characters encoded in Mouse matrix form. Columns 1-40 are punched from the matrix in B0, ..., B6 and columns 41-80 are punched from the matrix in C0, ..., C6. Each of the 80 columns may contain any 026 keypunch character.

J167 must be entered 12 times in order to complete the punching of a single card. The time between successive exits from J167 and succeeding entries is available to the programmer. Before the first entry to J167 a 100 0003 order must be given. If too much time is taken between the select order and the first entry or between successive exits and succeeding entries a late copy will result. (See timing chart below for detailed times.)(2) To punch cards at full speed, the select order must be given within 16.1 milliseconds of the 12th exit from J167.

As one card is being punched the previous card is being echo checked. Region D contains the binary image of the card being echo checked. The last (absolutely) card punched by J167 can be echo checked only by entering J166 again and punching an extra (blank) card. If it is desired to suppress echo checking the left operation of the word in * 0^2 must be made 014. If this is done, the next use of J167 will not cause an echo check stop (although the select order may be either 100 0002 or 100 0003) during the first card it punches after this. If there is no further alteration of * 0 there will be echo checking on subsequent cards.

Roughly speaking, once J167 has been entered, region D cannot be used until the final exit from J167. The exception to this statement occurs when punching with echo check suppression.(3)

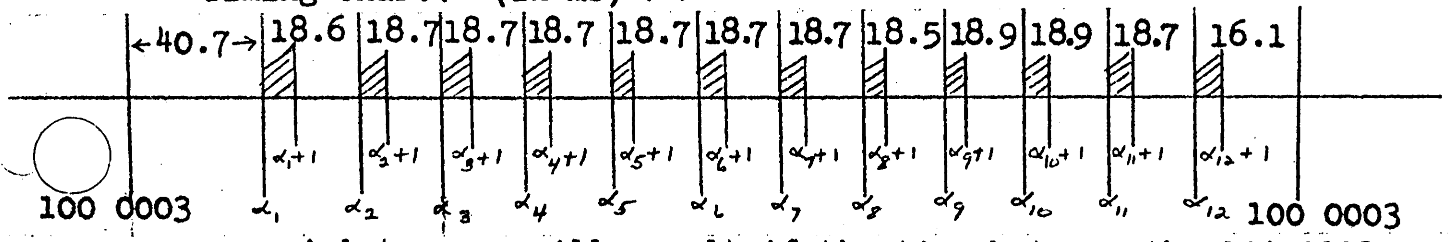
Calling sequence:

α_i	020 α 010 β	12 times for each card
α_i+1	Control returns here	(i = 1, ..., 12)

Highest symbol used: * 11

Program will stop at the right of word 75 (* 0)(4) if there is an echo check (there will be no way of proceeding) or at the right of word 69 if there is a late copy.

Timing chart: (in ms) (5)



A late copy will result if the time between the 100 0003 order and step α_1 or between steps α_i+1 and α_{i+1} (i=1, ..., 11) is exceeded. The punch will run at $\frac{1}{k}$ speed if the time used between step α_{12}+1 and the 100 0003 order is less than 16.1 + 600 (k-1) and greater than 16.1 + 600 (k-2) milliseconds.

Program length:

78 words

Region A (erasable) - 6 words, A3 to A8

Region B - 7 words, B0 to B6

Region C - 7 words, C0 to C6

Region D (semi-erasable) - 24 words, D0 to D23

Available as J167E

M. B. Shapiro and N. Z. Shapiro

(1) For an explanation of the intended use of J167, see JOHNNIAC Note No. 42.

(2) These times are based on 84 μs for an 02X operation and 89 μs for an 06X operation.

(3) See JOHNNIAC Note No. 42.

(4) * 0 is the 75th word of J167E.

(5) Shaded area is J167.

J168E MOUSE MATRIX TO PRINTER(1)

J168 prints 40 columns of information encoded in B0, B1, B2, B3, and B6 of the Mouse matrix. B4 and B5 are not used since only numeric information is printed.

Before entering J168 an appropriate select order must be given (100 0004)to select the printer and space or 100 0005 to select the printer). The entry to J168 must be completed within 17.8 milliseconds(2) of the select order or a late copy will result. To print at full speed the select order must be given within 12.5 milliseconds of the exit from J168.

If a character other than blank, 0, 1, 2, ..., 9 is called for, J168 will produce hash, although it will not hang up a printer column.

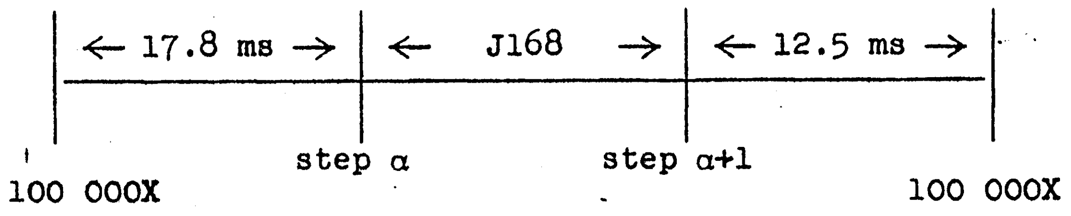
Calling sequence:

```
        α      020   α   010   β
        α+1    Control returns here
```

Highest symbol: * 1

Program will stop at the left of word 2 if there is a late copy.

Timing chart:



Program length:

29 words

Region A (erasable) - 4 words, A3 to A6

Region B - 5 words, Bo, B1, B2, B3, and B6

Available as J168E

M. B. Shapiro and N. Z. Shapiro

(1) For an explanation of the intended use of J168, see JOHNNIAC Note No. 42.

(2) These times are based on 84 μs for an 02X operation and 89 μs for an 06X operation.

J169E COMPLEX FLOATING-POINT MULTIPLY

J169 is a closed sub-routine which will multiply two complex floating-point numbers a and b. The arguments must be in region A prior to the execution of the basic link. The resulting complex floating-point product c = a · b will be found in region A when J169 links back to the main program. The arguments a and b will not be destroyed by J169.

<u>Calling Sequence</u>

```
α      020   $ 0   010  # 0   (Link to J169)
α+1    Control returns here.
```

The region A locations of a, b, and c are as follows:

A_0	contains	a^R = real	component	of	a	in	fl.	pt.	internal	form.
A_1	"	a^I = imaginary	"	"	a	"	"	"	"	"
A_2	"	b^R = real	"	"	b	"	"	"	"	"
A_3	"	b^I = imaginary	"	"	b	"	"	"	"	"
A_4	"	c^R = real	"	"	c	"	"	"	"	"
A_5	"	c^I = imaginary	"	"	c	"	"	"	"	"

Region A (erasable) 6 words A 0 - A 5

Symbolic Code $10_{10} = 12_8$ words

Highest symbol used: * 0

Program available as: J169E

John I. Derr

J170E COMPLEX FLOATING-POINT DIVIDE

J170E is a closed sub-routine which will obtain the complex floating-point quotient q of two complex floating-point numbers a and b. The dividend a and the divisor b must be in region A prior to the execution of the basic link to J170. The quotient q and the square of the norm of b $\left(|b|^2 = (b^R)^2 + (b^I)^2\right)$ will be found in region A when J170 links back to the main program provided that $b \neq 0$. If $b = 0$, q will not be computed. The arguments a and b will not be destroyed by J170.

Calling Sequence:

```
α      020   $ 0   010  # 0   (Link to J170)
α+1   Control returns here if b ≠ 0
α+2     "        "      "  "  b = 0.
```

The region A locations of a, b, q, and $|b|^2$ are as follows:

A_0	contains	a^R = real	component	of	a	in	floating-pt.	internal	form
A_1	"	a^I = imaginary	"	"	a	"	"	"	"
A_2	"	b^R = real	"	"	b	"	"	"	"
A_3	"	b^I = imaginary	"	"	b	"	"	"	"
A_4	"	q^R = real	"	"	q	"	"	"	"
A_5	"	q^I = imaginary	"	"	q	"	"	"	"
A_6	"	$\|b\|^2$ = norm squared of b.							

Region A (erasable): 7 words A 0 - A 6
Symbolic Code $17_{10} = 21_8$ words
Highest symbol used: * 1
Program available as: J170E

John I. Derr

J171E FIXED POINT SQUARE ROOT

J171 is a closed subroutine which will evaluate the non-negative solution of the equation $x^2 = |a|$, i.e. $x = +\sqrt{|a|}$. Before executing the basic link into J171, a <u>must</u> be in the <u>MQ</u>, where $0 \leq |a| < 1$. When J171 links back to the main program x will be in the <u>Accumulator</u> and $0 \leq x < 1$.

Calling sequence:

```
        020  $  0   010   #  0   (Link to J171E)
```

Description of the Method:

J171 first normalizes a so that $2^{-2} \leq a \cdot 2^{2q} < 1$, provided that $a \neq 0$. Let $y = a \cdot 2^{2q}$. Then $x_o = 1/2 + y/2$. Then if $x_o \neq y$, the following form of the Newton iteration scheme is used until $X_{n+1} \doteqdot X_n$:

$$X_{n+1} = X_n + 1/2 \left[y/X_n - X_n \right].$$

Finally, $x = X_n \cdot 2^{-q}$.

Accuracy: Let $\epsilon(a) = x - \sqrt{|a|}$ be the absolute error. Then $\epsilon(a) = 0$ or 2^{-39}. Observe that x has approximately q more significant binary digits than a.

Timing: Let T(a) denote the time in ms required to evaluate $\sqrt{|a|}$. Then .6 ms. < T(a) < 23 ms. for every admissable a. In particular, we list the following extreme cases:

$T(0) \doteqdot .6$ ms.

$T(1-2^{-39}) = T(1-2^{-38}) \doteqdot 1.49$ ms.

$T(2^{-2}) \doteqdot 13.14$ ms. (requires the maximum number (5) of iteration cycles).

$T(2^{-38}) \doteq 23.04$ ms (requires the maximum number (5) of iteration cycles <u>and</u> the maximum number (18) of normalization cycles).

Region A (erasable) - 3 words, A 0 - A 2

Symbolic Code: 20 words

Highest symbol used is * 5.

Program available as J171E

John I. Derr

J172E SCALAR PRODUCT OF TWO VECTORS -- FLOATING POINT

J172 computes the usual inner product (x, y) w.r.t. the standard orthonormal basis over a Euclidean Vector Space, i.e.

$$(x,y) = \sum_{i=1}^{n} x_i y_i \ .$$

Denote (x,y) by γ. All of the numbers $\{x_i\}$ and $\{y_i\}$ must be in the packed internal form recognized by the Floating-Point Interpretive System. Upon exit from J172 γ will be in the same form.

The elements $\{x_i\}$ of the vector $x = (x_1, \ldots, x_n)$ must be placed in H.S.S. as follows:

$$L(x_1),\ L(x_2) = L(x_1) + \Delta_x,\ \ldots,\ L(x_n) = L(x_1) + (n-1)\,\Delta_x,$$

where Δ_x is an integer and $|\Delta_x| \leq (377)_8 = 255_{10}$.

A similar statement holds for y.

Calling sequence:

α	020	α	010	β	(Link to J172)
$\alpha+1$	$\pm$	$L(\gamma)$	$\lvert\Delta_x\rvert$	$L(x_1)$	
$\alpha+2$	$\pm$	n	$\lvert\Delta_y\rvert$	$L(y_1)$	
$\alpha+3$	Control returns here				

Here the $\pm$'s in steps $\alpha+1$ and $\alpha+2$ correspond only to Δ_x and Δ_y, respectively. The usage is non-standard in the sense that the word in $\alpha+1$ and $\alpha+2$ must not be complemented in the case of a negative sign. More precisely, for step $\alpha+1$ take

$$\Delta[L(x_i)] = \begin{cases} |\Delta_x| & \text{if the sign position of the word in } \alpha+1 \text{ is } 0 \\ -|\Delta_x| & \text{if the sign position of the word in } \alpha+1 \text{ is } 1 \end{cases}$$

Do likewise in the case of $\alpha+2$.

Region A (erasable): 6 words A 0 - A 5

Symbolic Code: 31 words

Highest symbol used: * 5

Program available as: J172E

John Derr

J173E A LINEAR NORM OF TWO VECTORS -- FLOATING POINT

If we define an m^{th} degree norm of the vector $x = (x_1,\ldots,x_n)$ w.r.t. the standard orthonormal basis over a Euclidean Vector Space by

$$|x| = \left[\sum_{i=1}^{n} |x_i|^m\right]^{1/m},$$

then J173 computes the first degree norm of x, i.e.

$$|x| = \sum_{i=1}^{n} |x_i| .$$

Denote $|x|$ by γ. All of the numbers $\{x_i\}$ must be in the packed internal form recognized by the Floating-Point Interpretive System. Upon exit from J173 γ will be in the same form.

The elements $\{x_i\}$ of the vector x must be placed in H.S.S. as follows:

$L(x_1)$, $L(x_2) = L(x_1) + \Delta$, ..., $L(x_n) = L(x_1) + (n-1)\Delta$, where Δ is an integer and $|\Delta| \le 377_8 = 255_{10}$.

<u>Calling Sequence</u>:

α	020	α	010	β	(Link to J173)
$\alpha+1$	$\pm$	$L(\gamma)$	$\|\Delta\|$	$L(x_1)$	
$\alpha+2$	000	n	000	0000	
$\alpha+3$	Control returns here				

Here the $\pm$ in step $\alpha+1$ corresponds only to $|\Delta|$. The usage is non-standard in the sense that the word in $\alpha+1$ <u>must</u> not be complemented in the case of a negative sign. More precisely, take

$$\Delta\left[L(x_1)\right] = \begin{cases} |\Delta| & \text{if the sign position of the word in } \alpha+1 \text{ is } 0 \\ -|\Delta| & \text{if the sign position of the word in } \alpha+1 \text{ is } 1 \end{cases}$$

Region A (erasable): 4 words A0 - A3

Symbolic Code: 23 words

Highesty symbol used: * 5

Program available as J173E.

John Derr

J174E MAXIMUM ELEMENT OF A VECTOR

Let $x = (x_1, \ldots, x_n)$ be an n-tuple with real components. J174 will find the element $x_k \in \{x_i\}$ such that $|x_k| \geq |x_i|$ for $i=1, \ldots, n$ and if for some $\ell \neq k$, $|x_\ell| \geq |x_i|$ for $i=1, \ldots, n$, then $k < \ell$. The element x_k will be in the MQ upon exit from J174. In addition, $L(x_k) \cdot 2^{-39}$ will be in the A upon exit from J174.

The elements $\{x_i\}$ of the vector x must be placed in H.S.S. as follows:

$L(x_1)$, $L(x_2) = L(x_1) + \Delta$, ..., $L(x_n) = L(x_1) + (n-1)\Delta$, where Δ is an integer and $|\Delta| \leq 377_8 = 255_{10}$.

Calling Sequence:

```
α      020   α   010   β     (link to J174)
α+1     ±    n   |Δ|   L(x1)
α+2    Control returns here
```

Here the $\pm$ in step $\alpha+1$ corresponds only to Δ. The usage is non-standard in the sense that the word in $\alpha+1$ must not be complemented in the case of a negative sign. More precisely, for step $\alpha+1$ take

$$\Delta[L(x_1)] = \begin{cases} |\Delta| & \text{if the sign position of the word in } \alpha+1 \text{ is } 0 \\ -|\Delta| & \text{if the sign position of the word in } \alpha+1 \text{ is } 1 \end{cases}$$

Region A (erasable): 5 words (A0 - A4)

Symbolic Code: 23 words

Highest symbol used: * 8

Program available as J174E.

John Derr

J175E INTERCHANGE TWO VECTORS

Let $x = (x_1, \ldots, x_n)$ and $y = (y_1, \ldots, y_n)$ be two n-tuples with real components. Upon entry to J175 the following pair of instruction words will be executed for $i = 1, 2, \ldots, n$:

```
004   L(x_i)   020   L(y_i)
050   L(x_i)   060   L(y_i),
```

i.e. $y_i \rightarrow L(x_i)$ and $x_i \rightarrow L(y_i)$ is performed in the natural order $i=1, 2, \ldots, n$.

The elements $\{x_i\}$ of the vector x must be placed in H.S.S. as follows:

$L(x_1), L(x_2) = L(x_1) + \Delta_x, \ldots, L(x_n) = L(x_1) + (n-1)\Delta_x$,

where Δ_x is an integer and $|\Delta_x| \leq 7777_8 = 4095_{10}$. A similar statement holds for y.

Calling sequence:

α	020	α	010	β	(link to J175)
$\alpha+1$	000	$L(x_1)$	n	$L(y_1)$	
$\alpha+2$	$\pm$	Δ_x	000	Δ_y	
$\alpha+3$	Control returns here				

Here the $\pm$ in step $\alpha+2$ corresponds only to Δ_x and Δ_y. The usage is non-standard in the sense that the word in $\alpha+2$ must not be complemented in the case of a negative sign. More precisely, for step $\alpha+2$ take

$$\Delta\left[L(x_i)\right] = \begin{cases} |\Delta_x| & \text{if the sign position of the word in } \alpha+2 \text{ is } 0 \\ -|\Delta_x| & \text{if the sign position of the word in } \alpha+2 \text{ is } 1 \end{cases}$$

The same decision holds for $\Delta\left[L(y_i)\right]$.

Region A (erasable): 3 words A0 - A2

Symbolic Code: 21 words

Highest symbol used: * 6

Program available as: J175E.

John Derr

J176E SOLUTION OF A MATRIX EQUATION AX = B WHERE A IS n x n AND NON-SINGULAR -- FLOATING POINT

A link to J176 will result in the solution of the k systems of n simultaneous linear equations

$$\sum_{j=1}^{n} a_{ij} x_j^{(\ell)} = b_i^{(\ell)}, \; i = 1, 2, \ldots, n,$$

in n unknowns for $\ell = 1, \ldots, k$. If we define the matrices A, X, B by

$$A = \begin{pmatrix} a_{11} & a_{12} & \cdots & a_{1n} \\ a_{21} & & & \vdots \\ \vdots & & & \\ a_{n1} & & \cdots & a_{nn} \end{pmatrix}, X = \begin{pmatrix} x_1^{(1)} & x_1^{(2)} & \cdots & x_1^{(k)} \\ x_2^{(1)} & & & \vdots \\ \vdots & & & \\ x_n^{(1)} & & \cdots & x_n^{(k)} \end{pmatrix}, B = \begin{pmatrix} b_1^{(1)} & b_1^{(2)} & \cdots & b_1^{(k)} \\ b_2^{(1)} & & & \\ \vdots & & & \\ b_n^{(1)} & & \cdots & b_n^{(k)} \end{pmatrix}$$

then the two statements concerning the function of J176 are equivalent. J176 also computes the determinant of A which we denote as det A.

<u>Calling Sequence</u>:

```
α      020   α    010   β          (Link to J176)
α+1     k    n     △    L(a₁₁)
α+2    Control returns here if det A = 0
α+3    Control returns here if det A ≠ 0
```

Before linking into J176 the user must place the elements a_{ij} of A and $b_i^{(\ell)}$ of B into H.S.S. locations as follows:

$$\begin{array}{l} a_{11} \; a_{12} \; \cdots \; a_{1n} \; b_1^{(1)} \; b_1^{(2)} \; \cdots \; b_1^{(k)} \quad \_\_\_ \\ a_{21} \; a_{22} \qquad\qquad\qquad\qquad\qquad\qquad\quad \vdots \\ \vdots \\ a_{n1} \; a_{n2} \; \cdots \; a_{nn} \; b_n^{(1)} \; b_n^{(2)} \; \cdots \; b_n^{(k)} \quad \_\_\_ \end{array}$$

Basically, these locations differ by Δ. More precisely,

$$\text{loc } (a_{i,j+1}) = \text{loc } (a_{ij}) + \Delta, \; j=1, 2, \ldots, n-1,$$
$$\text{loc } (a_{i,1}) = \text{loc } (b_{i-1}{}^{(k)}) + 2\Delta, \; i \neq 1,$$
$$\text{loc } (b_i{}^{(1)}) = \text{loc } (a_{i,n}) + \Delta, \text{ and}$$
$$\text{loc } (b_i{}^{(\ell+1)}) = \text{loc } (b_i{}^{(\ell)}) + \Delta, \; \ell=1, 2, \ldots, k-1.$$

Upon exit from J176 these same H.S.S. locations will contain

$$\begin{array}{llllllll}
\alpha_{11} & \alpha_{12} & \cdots & \alpha_{1n} & x_1{}^{(1)} & x_1{}^{(2)} & \cdots & x_1{}^{(k)} \quad c_1 \\
\alpha_{21} & \alpha_{22} & \cdots & & & & & \quad\;\; \vdots \\
\vdots & & & & & & & \\
\alpha_{n1} & \alpha_{n2} & \cdots & \alpha_{nn} & x_n{}^{(1)} & x_n{}^{(2)} & \cdots & x_n{}^{(k)} \quad c_n
\end{array}$$

In addition, det A will be in A 19. All of the numbers referred to are in the packed internal form recognized by the Floating-Point Interpretive System. The quantities k, n, and Δ, in the Calling Sequence are positive integers with the obvious upper bounds $(77)_8$, $(7777)_8$, and $(377)_8$, respectively. Clearly, these are not least upper bounds.

The matrix $\alpha = (\alpha_{ij})$ is the n x n matrix obtained from A by the forward solution part of the elimination process. The n x 1 matrices $\left(x_1{}^{(\ell)}, x_2{}^{(\ell)}, \ldots, x_n{}^{(\ell)}\right)$ are obtained in the back solution part of the method. The n x 1 matrix $(c_1, c_2, \ldots c_n)$ should be ignored by the user. It functions only as a check on the computational accuracy of the forward and backward solutions.

Description of the Method

The basic method used is the Gauss elimination method. In the forward solution the matrix A is reduced to a triangular matrix T. The solution of Tx = Eb is equivalent to the solution of the original equation Ax = b, where $E = \prod_i E_i$ and the E_i are elementary matrices, i.e. $E_i M$ results in performing elementary row operations on M. The back solution can be performed by a simple substitution using the equation Tx = Eb.

The rule of formation of the sequence $(E_1, E_2, \ldots)$ is known as Crout's Method*. However, here the method is modified so as to permute the rows of the resulting matrix, if necessary, to use the element $\alpha_{ki} = \underset{i \leq j \leq n}{\text{Max}} \left(|\alpha_{ji}| \right)$ as the α_{ii} pivot element for i = 1, 2, ..., n.

The computation of the i^{th} row of T is checked versus $(i+1)^2 \cdot 10^{-7}$ according to the Crout criteria for checking. Similarly, the calculation of the i^{th} row of X is checked versus $(n+1)^2 \cdot 10^{-7}$. This method of checking will probably suffice in most cases w. r. t. testing for machine malfunctions. Note that this check does not guarantee that the solution is "correct" in any normal sense, since the ultimate accuracy of the computed solution is limited by the inherent error of the original system of equations.

*See Numerical Calculus, pp. 17-29, by W. E. Milne

NOTES

(i) J172-175 are slave routines w.r.t. J176 and therefore must be used along with the Floating-Point System in conjunction with J176. Further, J172-5 have the origins WO, XO, YO, and ZO, respectively, w.r.t. J176. Hence, these regions must be assigned locations before J176 is loaded.

(ii) A 13X halt will occur in the 119th word of J176 if the computational checking criteria is not met in any row during the forward solution. The calculation will continue if the GO button is pressed.

(iii) A 13X halt will occur in the 181st word of J176 if the computational checking criteria is not met in any row during the back solution. The calculation will continue if the GO button is pressed.

(iv) A 13X halt will occur in the 75th word of J176 if det $A=0$. Control will link back to the main program if the GO button is pressed.

(v) Clearly, the original matrices A and B are destroyed by J176.

(vi) The relation

$$n(n+k+1) \leq (4096-F) - 288 - 26 - M$$

must always be satisfied when using J176, where $F =$ the number of words used by the Floating-Point Interpretive System and $M =$ the number of words in H.S.S. occupied by programs other than J172-6 and the Floating-Point System.

In order to get some real-live numbers, suppose $(4096 - F) = 2960$, $\Delta = 1$, and $M = 115$. Then $n(n+k+1) \leq 2531$ must hold. If $k=1$, $n \leq 49$. If $k=n$, $k=n\leq 35$.

Timing

Since at the time of this writing no explicit times were known for the floating-point operations, it follows that no explicit timing estimates could be made for the general case. However, the following floating-point operation counts should yield a reasonable measure of how the time will vary as a function of n and k.

Operation Counts for the Forward Solution

(÷) $\frac{n(n+2k+1)}{2}$

(+) and (x) $\frac{n^3}{3} + \frac{k\,n^2}{2}$ for each (approximate for large n or k)

Operation Counts for the Backward Solution

(+) and (x) $\frac{n(n-1)(k+1)}{2}$ for each

While operating in the SD Mode five seconds were required for a solution with n=7 and k=1. This example can be used as a reference point in the (n,k) plane to compute approximate absolute times using the above formulae.

J176 Storage (without slaves)

Reg. A (erasable) 20 words (A 6 - A 25)

Symbolic Code: 190 words

Highest Symbol Used: * 41.

Storage Used by Slave Routines (J172-J175)

Reg. A 6 words (A 0 - A 5) (Note that the erasable storage used by J172 - J175 does not overlap that used by J176.)

Symbolic Code: 98 words

Highest Symbol Used: * 8

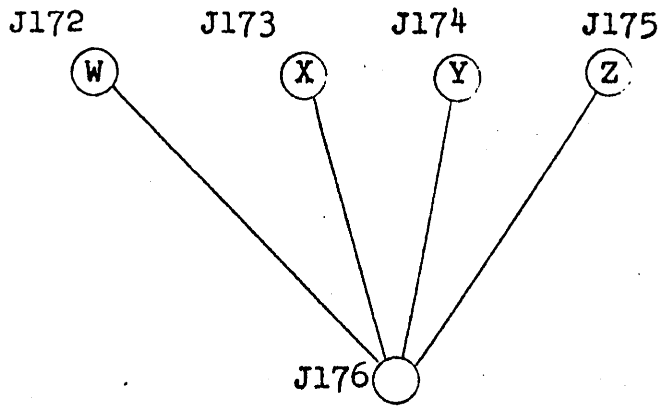
Combined Storage Used by J172 - J176

Reg. A 26 words (A o - A 25)

Symbolic Code: 288 words

Highest Symbol Used: * 41

Lattice Diagram Describing Master-Slave Relations



John Derr

J177A AUTOMATIC SOLUTION OF A MATRIX EQUATION AX=B WHERE A IS n x n AND NON-SINGULAR -- FLOATING POINT

J177 is an automatic version of J176. The user needs only to record the necessary input data in punched cards, place these cards immediately behind J177, and press the LOAD button. The solution matrix, X*, which approximates X in the equation AX=B will be computed and printed for each equation.

<u>Input Form</u>

Information should be punched in the Floating Point Data Form as follows:

<u>1st card for a problem</u>

```
            (1)                                   (2)
±  EXP  MANTISSA                       ±  EXP  MANTISSA
|   n      ID                          |   k     ______
|   |      └ Any decimal               |   └    number of columns
|   |        information               |        in the matrices
|   |        for problem ID            |        X and B
|   └order of matrix  A                |
└[+ if A is not to be printed          └[+ if (αij) is not to be printed
 [- if A is to be printed               [- if (αij) is to be printed
```

The elements of A must be punched by rows into consecutive positions of the cards beginning with the second card. Each row must begin with position (1) of a card and if an EF mark (12 punch (+) in column 80) is encountered prior to the reading of the n^{th} element of a given row, then J177 will assume that the remaining elements of the row are 0. (This does not, of course, affect the six elements of A which were punched

into the card with the EF mark.)

The elements of B must be punched by <u>columns</u> into consecutive positions of the cards beginning with the first card following the cards for A. Each column must begin with position (1) of a card.

The solutions for a sequence of equations AX=B can be evaluated by placing the input cards for the equations in sequential order behind J177. If the input cards are followed by two blank cards, the collator will be on Select following the completion of the last problem.

<u>Output Form</u>

The page will be ejected prior to the beginning of each problem. The numbers printed will be in the external floating point form. The print format is that described for the 013 (PNT) order in the Floating-Point writeup. The printing for one problem takes place as follows:

	A	B	C
Information in positions (1) and (2) of 1st Input Card	——	$\pm$ n ID	$\pm$ k ——
Space	——	——	——

If A is printed, then the printing for the i^th row is as follows:

Row number	——	i	——
Elements of the row begin here	a_{i1} ⋮	a_{i2} ⋮	a_{i3} ⋮

If A is printed, the page is ejected following the printing of the last row.

	A	B	C
Determinant	—	—	det A
Space	—	—	—

If (α_{ij}) is printed, the same procedure is followed as for A.

S is printed by columns The printing for the ℓth column is as follows:

Column number	ℓ	—	—
Elements of the column begin here	$x_1^{(\ell)}$ ⋮	$x_2^{(\ell)}$ ⋮	$x_3^{(\ell)}$ ⋮

<u>Notes</u>:

(1) The inequality $n(n+k+1) \leq 2529$ must be satisfied by n and k. In particular, if k=1, then n must not exceed 49 and if k=n, then n must not exceed 35. If $n(n+k+1) > 2529$, the program will halt at 5331_8 with a 130 5331 order.

(2) A 13X halt will occur at 5172_8 if the computational checking criteria is not met in any row during the forward solution. The calculation will continue if the GO button is pressed.

(3) A 13X halt will occur at 5270_8 if the computational checking criteria is not met in any row during the back solution. The calculation will continue if the GO button is pressed.

(4) A 13X halt will occur at 5116_8 if det A=0. If the GO button is pressed, then det A and the current state of the (α_{ij}) and X (or B) matrices will be printed. The next

problem in sequence will then be started if it exists.

(5) The user is referred to the J176 write-up for a description of the method.

Program available as J177A

John Derr

J178E LEAST SQUARES POLYNOMIAL APPROXIMATION - FLOATING POINT

J178 will compute the best possible polynomial approximation to a set of data in the sense of least squares. The data consists of a set of number pairs (x_i, y_i) for $i = 1, 2, \ldots, N$. The set of number pairs determine a relation which we denote as y (x) and J178 approximates y(x) by a polynomial,

$$\sum_{j=0}^{M} a_j x^j = y^*(x) \text{ for } 1 \leq M \leq 5.$$

J178 must be used in conjunction with the current version of the Floating Point System. All of the numbers x_i and y_i must be in the packed internal form and must be placed in consecutive H.S.S. locations as follows:

$x_1, y_1,$ ——, $x_2, y_2,$ ——, ..., $x_N, y_N,$ ——.

OK

Calling Sequence:

α	020	α	010	β		(Link to J178)
$\alpha+1$	0δ	$L(a_0)$	N	$L(x_1)$		
$\alpha+2$	Error Return -- det T = 0					
$\alpha+3$	Control returns here					

Here, $\delta = \delta_0\ \delta_1\ \delta_2 \quad \delta_3\ \delta_4\ \delta_5$ and

$$\delta_j = \begin{cases} 1 \text{ if } 0 \leq j \leq M \\ 0 \text{ if } j > M \end{cases}$$

Upon exit from J178, the coefficients, a_j for $j = 0, 1, \ldots, M$, of y^* (x) will be in consecutive

H.S.S. locations beginning with $L(a_0)$. J178 also computes $y^*(x_i) = y_i^*$ for $i = 1, 2, \ldots, N$ and the y_i^*, together with the corresponding pairs (x_i, y_i) are stored as triplets in the form: $x_1, y_1, y_1^*, x_2, y_2, y_2^*, \ldots, x_N, y_N, y_N^*$.
The a_j and y_i^* are in the packed internal form.

Method:

The fundamental least squares method involves finding the $\{a_i\}$ which will minimize

$$\sum_{i=1}^{N} \left[y(x_i) - y^*(x_i)\right]^2 .$$

The values of a_i are found explicitly as the solution of a matrix equation $Ax = b$[1].

As the degree, M, of the polynomial $y^*(x)$ increases, the accuracy of the computed a_i decreases rapidly. The loss of significant digits occurs in the solution of the matrix equation $Ax = b$.

In order to mitigate this inherent loss of information, the basic least squares method has been modified according to a process developed at Lockheed Aircraft Co. for their FLOP coding system. A description of this modified process follows below.

First normalize the $\{x_i\}$ to the interval $[-1, +1]$ by setting $x'_i = x_i \div |x_i|_{max}$. The modification then consists of replacing the $\{x_i^j\}$ used in the standard method by the corresponding Chebyshev polynomials $\{T_j\ (x'_i)\}$ for $j = 0, 1, \ldots, M$. The $\{T_j\ (x'_i)\}$ are defined inductively by $T_0\ (x'_i) = 1$,

1 See Milne, Numerical Calculus, pp. 242,3

$T_1(x'_i) = x'_i$, $T_j(x'_i) = 2x'_i \cdot T_{j-1}(x'_i) - T_{j-2}(x'_i)$ for $j = 2, 3, \ldots, M$.

Then proceed in the ordinary least squares fashion to find the coefficients $\{t_j\}$ of the polynomial $\sum_{j=0}^{M} t_j \cdot T_j(x)$. The matrix equation $Ax = b$ is replaced by the new equation $Tt = d$ under the transformation $x^j \longrightarrow T_j(x')$ where

$$T = \begin{bmatrix} \sum T_0T_0 & \sum T_0T_1 & \cdots & \sum T_0T_M \\ \sum T_1T_0 & & & \\ \vdots & & & \\ \sum T_MT_0 & \sum T_MT_1 & \cdots & \sum T_MT_M \end{bmatrix}, \quad d = \begin{bmatrix} \sum T_0y_i \\ \sum T_1y_i \\ \vdots \\ \sum T_My_i \end{bmatrix}, \quad t = \begin{bmatrix} t_0 \\ t_1 \\ \vdots \\ t_M \end{bmatrix}.$$

In the above we have used the shorthand notation

$$\sum T_p\, T_q = \sum_{i=1}^{N} T_p(x'_i) \cdot T_q(x'_i) \text{ and } \sum T_p y = \sum_{i=1}^{N} T_p(x'_i) \cdot y_i .$$

It remains to compute the $\{a_i\}$ from the $\{t_i\}$ by essentially performing the inverse of the transformation $x^j \longrightarrow T_j(x')$. We set

$$\sum_{j=0}^{M} t_j\, T_j(x') = \sum_{j=0}^{M} a_j x^j$$ and expanding the T_j we have

$$t_0 + t_1 \cdot x' + t_2\left(2(x')^2 - 1\right) + t_3\left(4(x')^3 - 3(x')\right)$$
$$+ t_4\left(8(x')^4 - 8(x')^2 + 1\right) + t_5\left[16(x')^5 - 20(x')^3 + 5(x')\right]$$
$$= \sum_{j=0}^{M} a_j x^j .$$

Recalling that $x' = x \div |x_i|_{max}$ and equating like coefficients of x^j we get by proceeding inductively on M:

for $M=1$, $a_0 = t_0$ and $a_1 = t_1 \div |x_i|_{max}$

for $M=2$, $a_0 = t_0 - t_2$, $a_1 = t_1 \div |x_i|_{max}$ and $a_2 = 2t_2 \div \left(|x_i|_{max}\right)^2$

$\vdots$

for $M=5$, $a_0 = t_0 - t_2 + t_4$, $a_1 = (t_1 - 3t_3 + 5t_5) \div |x_i|_{max}$,

$a_2 = \left(2t_2 - 8t_4\right) \div \left(|x_i|_{max}\right)^2$, $a_3 = \left(4t_3 - 20t_5\right) \div \left(|x_i|_{max}\right)^3$

$a_4 = 8t_4 \div \left(|x_i|_{max}\right)^4$, and $a_5 = 16t_5 \div \left(|x_i|_{max}\right)^5$.

NOTES

(1) J172, J173, J175, and J195 (or J174) are slaves to J196 (as well as to J178) and hence must be assigned locations prior to the loading of J196. Further, J196, J191, and J187 are slaves to J178 and must be assigned locations before J178 is loaded. The regions associated with all of the slaves can be ascertained from the lattice diagram.

(2) J174 may be used in place of J195 if all of the numbers $\left\{x_i, y_i\right\}$ are normalized and the floating point system is in the N mode.

(3) For an explanation of a 13X halt which occurs in J196, see Notes (ii), (iii), and (iv) of the J196 write-up. The computational checking criteria can fail to be met due to a loss of significant digits in the solution of the matrix equation Tt = d. The chance that a halt will occur increases with the degree M of the polynomial approximation.

(4) In general, the accuracy of the computed $\left\{a_i\right\}$ can be increased by first translating the set of $\left\{x_i\right\}$ into the set of $\left\{x_i - m\right\}$ where m is the mean of the $\left\{x_i\right\}$, i.e.

$$m = \left(\sum_{i=1}^{N} x_i \right) \div N.$$

Then enter J178 with the number pairs $\left(x_i - m, y_i \right)$. Upon exit from J178 use the computed coefficients with the argument x-m or perform the inverse transformation on the coefficients to get the coefficients w.r.t. the $\left\{x_i\right\}$. Usually, this additional procedure will be necessary only for large M and strongly non-zero m.

Timing

The Legendre polynomials of degree one through five were used as test problems with 21 points, i.e. $M = 1, 2, 3, 4, 5$ and $N = 21$. The following times were recorded while operating in the SD mode:

$M = 1$	4 sec.	$M = 2$	7 sec.
$M = 3$	10 sec.	$M = 4$	14 sec.
$M = 5$	18 sec.		

Then, roughly speaking, the operating time increases directly as M. In the case $M = 1$, 7 seconds were required for $N = 42$. Apparently,then the variation of time with N is also approximately direct.

Accuracy

Using $x_i = 0(.05)1$ and the exact corresponding values of y_i obtained from a table of Legendre polynomials of degree M the number of correct significant digits obtained in the computed coefficients were as follows:

$M = 1$	9 digits		
$M = 2$	6 digits	$M = 3$	5 digits
$M = 4$	3 digits	$M = 5$	2 digits

By translating the interval $[0,1]$ to $[-.5, .5]$ the following results were obtained with all other conditions unchanged:

$M = 2$	9 digits	$M = 3$	9 digits
$M = 4$	8(-) digits	$M = 5$	7 digits

Observe that in each of the cases listed, the accuracy obtained for the translated interval $[-.5,.5]$ is better than that obtained for the original interval $[0,1]$.

<u>J178 Storage (without slaves)</u>

Region A (erasable) 76 words (A12-A87)

Symbolic Code: 201 words

Highest Symbol Used: * 56

<u>Storage Used by Slaves (J172, J173,J175,J187,J195,J196 and J191)</u>

Region A (erasable) 26 words (A0 - A25)

Symbolic Code: 381 words

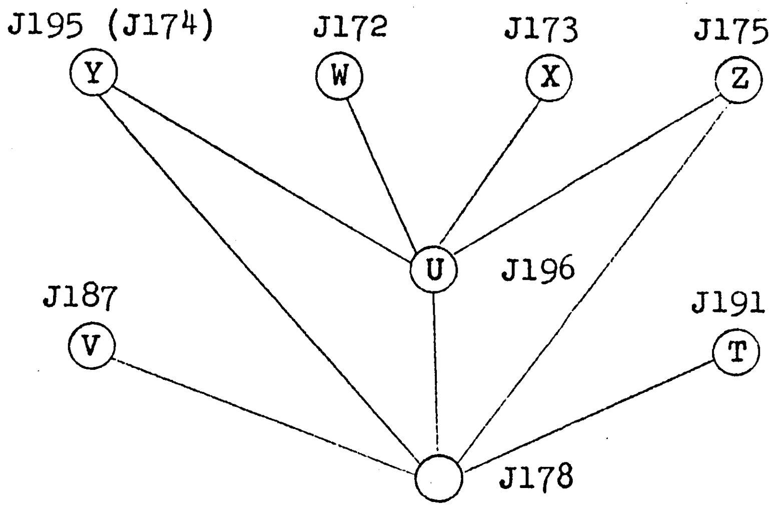
Highest Symbol Used: * 41

<u>Combined Storage Used by J178 and Slaves</u>

Region A (erasable) 88 words (A0 - A87)

Symbolic Code: 582 words

<u>Lattice Diagram Describing Master-Slave Relations</u>



John Derr

J179A AUTOMATIC LEAST SQUARES POLYNOMIAL APPROXIMATION -- FLOATING POINT

J179 is an automatic version of J178. The user needs only to record the necessary input data in punched cards, place these cards immediately behind J179, and press the LOAD button. The coefficients $\{a_i\}$ of the approximating polynomial

$$f(x) = \sum_{i=0}^{M} a_i x^i$$

will be computed and printed for each header card.

Input Form

Information should be punched in the Floating Point Data Form as follows:

pos (1)	S	$\begin{cases} + \text{ if data triplets } \underline{\text{are}} \text{ to be printed} \\ - \text{ if data triplets are } \underline{\text{not}} \text{ to be printed} \end{cases}$
	EXP	M = degree of polynomial f(x)
	MANTISSA	ID (any decimal information)
pos (2)	S	$\begin{cases} + \text{ if data cards follow} \\ - \text{ if no data cards follow} \end{cases}$
	EXP	Blank
	MANTISSA	N = number of points
pos (3)	S	Blank (or +)
	EXP	$\begin{cases} 00 \text{ if } (x_i, y_i, r_i^1) \text{ is to be printed} \\ 01 \text{ if } (x_i, y_i, y_i^*) \text{ is to be printed} \end{cases}$
	MANTISSA	$\begin{cases} \delta \text{ if } \delta_j = 0 \text{ for } A \leq j \leq M^2. \\ 0 \text{ otherwise} \end{cases}$

Positions (4), (5), and (6) of the header card are not interpreted by J179A.

1 $r_i = y_i^* - y_i$. Hence, $y_i^* = r_i + y_i$.

2 See J178 write-up.

(x_i,y_i)Data Cards

The number pairs (x_i,y_i) must be punched in consecutive three-position blocks beginning with the first card following the header card.

	(1)	(2)	(3)	(4)	(5)	(6)
1st Card	x_1	y_1	---	x_2	y_2	---
⋮	⋮	⋮	⋮	⋮	⋮	⋮
last card (N even)	x_{N-1}	y_{N-1}	---	x_N	y_N	---+
(N odd)	x_N	y_N	---	---	---	---+

Further, there must be an end of file mark (12 punch) in column 80 of the last data card.

There is a one-to-one correspondence between header cards and problems. A sequence of problems can be solved by placing the associated header cards together with the data cards following (if any) in sequential order behind J179. If the entire deck of input cards is followed by two blank cards, the console will show "READER" following the completion of the last problem.

Output Form

The page will be ejected prior to the beginning of each problem. The numbers printed will be in the external floating point form. The print format is that described for the 017 (PNT) order in the Floating-Point write-up. The printing for one problem takes place as follows:

	A	B	C
	S EXP MAN	S EXP MAN	S EXP MAN
Information in header card	$\pm$ M ID	$\pm$ 00 N	$\pm$ 0 $^0/_1$ $^\delta/_0$
Space	—	—	—
Coefficients	a_0	a_1	a_2
	$\vdots$	$\vdots$	$\vdots$
Space	—	—	—

If the number triplets are to be printed, they are printed one triplet per line in the natural order.

x_1	y_1	r_1/y_1*
x_2	y_2	r_2/y_2*
$\vdots$	$\vdots$	$\vdots$
x_N	y_N	r_N/y_N* .

<u>Notes</u>

(1) Prior to entry into J178, the set of $\{x_i\}$ is replaced by the set of $\{x_i - m\}$ where

$$m = \left(\sum_{i=1}^{N} x_i\right) \div N.$$

Upon exit from J178, J194 is used to compute the coefficients $\{a_i\}$.

(2) $1 \leq M \leq 5$ and $M+1 \leq N \leq 120$[1] must hold.

(3) If $\delta \neq 0$, then the <u>decimal</u> <u>equivalent</u> of the octal number $\delta_0\ \delta_1\ \delta_2\ \ \delta_3\ \delta_4\ \delta_5$ must be presented to J179. For example, suppose M=5 and it is desired to impose the condition that

1 The upper bound of 120 on N was artificially imposed.

the odd powers of x^1 shall vanish. Hence, we want $a_1 = a_3 = a_5 = 0$. Then $\delta = (101\ 010)_2 = (52)_8 = (42)_{10}$. $\delta = 42$ is presented to J179.

(4) A 13X halt will occur at $(2434)_8$ if the computational checking criteria is not met in any row during the forward solution in J196. The calculation will continue if the GO button is pressed. This halt can occur due to the loss of significant digits in the matrix solution.

(5) A 13X halt will occur at $(2533)_8$ if the computational checking criteria is not met in any row during the back solution. The calculation will continue if the GO button is pressed. This halt can occur due to the loss of significant digits in the matrix solution.

(6) A 13X halt will occur at $(2357)_8$ if det T = 0. If the GO button is pressed, then J179 will exit J196 and J178. If printing occurs upon completion of this problem, the results will be incorrect. J179 will continue ahead to the next problem if there is one.

(7) A check sum 13X halt at $(0014)_8$ indicates that J179 has probably been loaded incorrectly under the control of J180A. The entire loading process should be restarted.

Accuracy

Using $x_i = 0(.05)1$ and the exact corresponding values of y_i obtained from a table of Legendre polynomials of degrees M = 2, 3, 4, 5 the number of correct significant digits obtained in the computed coefficients was as follows:

M=2	9 digits	M=3	9 digits
M=4	8(-) digits	M=5	7 digits

Timing

See the discussion of timing in the J178 write-up. The times there do not include input-output times, however.

Program available as J179A.

John Derr

J180A   LOADER FOR STYLE F CARDS WITH BLOCK CHECK SUMS

J180 is a one-card self-loading routine which loads style F cards from the primary feed. J180 works exactly as J135 does, except that when a row containing 010 0006 in the control field is encountered, it sums the high-speed storage (modulo 2^{40}) between previously specified cells a and b, inclusively. If this sum agrees with a previously specified check sum, it proceeds to the next card. If the sum does not agree with the check sum a 13X stop occurs. To continue reading, press the GO button. The row containing the transfer instruction must have: 1XX XXXX XXX a in the information field and information must have previously been placed (presumably by the preceding two rows) in 0015_8 and 0016_8 as follows:

0015_8	XXX XXXX XXX b-a
0016_8	The check sum

J181 and J182 prepare cards for loading by J180. The preparation, of course, includes the extra three rows. J180 will also load style F cards without check sums or mixed cards.

Storage: 0001 to 0016 Octal.

Error Stops: Left 0014 Octal. To proceed, press GO button.

Timing: Reads cards at 240 cards per minute--except when check summing.

Available as J180A.

Norman Shapiro

J181E PUNCH STYLE F CARDS WITH BLOCK CHECK SUMS

J181 will punch a block of storage from a to b in Style F, with three extra rows containing the block check sum information, ready for reloading by J180. C_{1-19} are left blank in the cards punched.

As an option the block a to b can be punched so as to be reloaded into the block (a+c) to (b+c) (modulo 2^{12}) by placing c in the right address of * 0. Once c is placed there, it will remain unless further modified. The restriction (a+c) (modulo 2^{12}) $\leq$ (b+c) (modulo 2^{12}) must be satisfied.

Calling sequence:

```
        α      020   α    010   β    Link to J181
        α+1          a          b
        Control returns to left of α+2
```

Symbolic Code: 31 words

Location of * 0: Last word of routine

Region A (erasable): A 0 to A 2

Highest Symbol Used: * 13

Timing: Punches cards at 100 cards per minute.

Program available as J181E

Norman Shapiro

J182E PUNCH STYLE F WITH I.D. AND BLOCK CHECK SUM

J182 will punch a block of storage from a to b in style F --with a block check sum ready for reloading by J180. Identification specified in the calling sequence is punched in C_{1-5} of each card. A sequence number beginning with 001 and increasing by one with each card is punched in C_{6-8} of each card.

Option 1: The block from a to b can be punched so as to be reloaded into the block from a+c to b+c (mod 2^{12}) by placing c in the right address of *0, the last word of J182, before entering.

Option 2: The three digits of the sequence number are stored by J182 in cells A11, A12 and A13. They are normally preset to zero and then modified so as to increase the sequence number by one prior to punching every card including the first card. The presetting to zero (but not the modification) can be suppressed by entering the routine at the right of the first word instead of the left. This option can be used to begin sequencing with any value (including 000 by presetting to 0, 0 and -1) and to punch several blocks into one consecutively sequenced deck (by suppressing the presetting to zero and not otherwise modifying A11, A12 and A13). The sequence number digits are scaled by 2^{-5}.

Calling sequence:

```
        α      020    α    01kβ
        α+1           a    b
        α+2                x₁
        α+3                x₂
        α+4                x₃
        α+5                x₄
        α+6                x₅
        α+7  Control returns to left of α+7
```

where $x_1, \ldots, x_5$ are the identification in the character code used by Easy Fox and by J148. This code is attained by assigning a value of 16, 32 and 48 to 12, 11 and 0 punches, respectively. This code differs from the Mouse code which assigns a value of 64 to 0 punches. k=4 suppresses sequence presetting to zero. Otherwise, set k=0.

Symbolic Code: 115 words

Region A Erasable - A0 to A21

Highest Symbol Used: *41

Timing: Punches cards at half speed, 50 cards per minute.

Note: J182 is the only program which punches style F cards at anything less than full speed.

Available as J182E

Norman Shapiro

J183A MATRIX INVERSION

J183 will solve systems of equations and/or invert matrices of order n ($1 \leq n \leq 40$). All arithmetic for this routine is carried out in double-precision fixed-point to guarantee maximum accuracy.

The matrix of coefficients and the right-hand side for this routine are input in decimal, one per card, having the following format:

Col.	Contents
17-19	j ($1 \leq j \leq n$) j = 0 for the right-hand side
20-22	i ($1 \leq i \leq n$)
24-27	Integral portion of a_{ij}
28-33	Fractional portion of a_{ij}
34	Sign (Blank is +; 1 is minus)

The remainder of the card may contain any information desired. The matrix is then printed column by column. The right-hand side is <u>not</u> printed.

It is not necessary to enter the zero elements of the system since HSS is cleared to zero from 610 through 4095 each time the Input routine is called for. The elements may be in any order. Each matrix or system of equations <u>must</u> be headed by a single card with the order of the system, n, in j and cols. 20-34 blank, and followed by a blank card to signify the end of the deck. (This input format is the same as the JOHNNIAC simplex code.)

As many systems of equations to be solved or matrices to be inverted as desired may be entered behind the program deck, just so each blank deck has the proper header card and is followed by a blank. Two additional blank cards are necessary

NIA = $(0152)_8$ The matrix is singular or poorly conditioned.

NIA = $(0317)_8$ Overflow, rescale the matrix.

NIA = $(0052)_8$ H2 on, Matrix inverted, hit GO to read in next matrix.

J183 uses all of HSS and the Drum.

Program available as J183A.

Time: The only estimate available is for inverting a 24 order matrix -- this took roughly three minutes of computing time.

M. I. Bernstein

3-26-56

J184A MATRIX ASSEMBLY

J184 assembles a matrix of order n ($1 \leq n \leq 40$) into binary for use with J141A in place of the binary inverse obtained from J183A.

The inputs for J184 follow exactly the same format and rules as the inputs for J183. It is possible to assemble more than one matrix with J184, just so each matrix has the proper header card (see J183 write-up) and is followed by a blank. After each matrix is punched, and printed row by row, there is a 13X stop with NIA equal to $(0165)_8$. To assemble the next matrix, hit GO.

Program stops:

NIA = $(0055)_8$, No header card for matrix or i or j is greater than n.

NIA = $(0165)_8$, Assembly completed.

J184 uses all of HSS.

Program available as J184A.

M. I. Bernstein

3-27-56

J185E INTEGRATION OF n SIMULTANEOUS SECOND-ORDER DIFFERENTIAL EQUATIONS WITH INITIAL CONDITIONS SPECIFIED

J185 integrates n simultaneous second-order differential equations of the form:

$$\ddot{y}_i = f_i(y_0, y_1, \dots y_{n-1}, \dot{y}_0, \dot{y}_1, \dots \dot{y}_{n-1}, x)$$

$$i = 0, 1, \dots n-1$$

where differentiation is with respect to x. Each time J185 is entered, integration is carried out over one step Δx, replacing the values of y_{ij}, $(\Delta x)\dot{y}_{ij}$, and $(\Delta x)^2\ddot{y}_{ij}$ at $x = x_j$ by $y_{i,\ j+1}$, $(\Delta x)\ \dot{y}_{i,\ j+1}$, and $(\Delta x)^2\ddot{y}_{i,\ j+1}$ at $x = x_{j+1} = x_j + \Delta x$. An auxiliary closed subroutine is required, which when entered with the calling sequence:

```
α      020   α   010   F 0
α+1    Control returns here
```

evaluates $(\Delta x)^2\ddot{y}_i$ and stores these quantities in G.0+i. The y_i and $(\Delta x)\dot{y}$ are located in E.0+i and E.0+i+n respectively. If required, the independent variable x should be calculated by the master routine.

Before J185 is entered for the first time the values of y_{i0}, $(\Delta x)\dot{y}_{i0}$, and $(\Delta x)^2\ddot{y}_{i0}$ at $x = x_0$ must be located in the H. S. S. as follows:

Location	Contents
E.0 + i	y_{i0}
E.0 + i + n	$(\Delta x)\ \dot{y}_{i0}$
E.0 + i + 2n	$(\Delta x)^2\ddot{y}_{i0}$

$$i = 0, 1, \dots n-1$$

After integration at any $x = x_j$, these storage positions contain y_{ij}, $(\Delta x)\,\dot{y}_{ij}$, and $(\Delta x)^2\ddot{y}_{ij}$. Two constants must be provided at all times in region D as follows:

Location	Contents	
D.0	$n \times 2^{-39}$	(n is number of equation, n = 1, 2 ...)
D.1	ϵ	(ϵ is the tolerance on convergence, see Method)

Calling sequence:

```
α      020   α   010   γ
α+1    Control returns here
```

H. S. S. Storage Requirements:

Symbolic Code	41 words	(* 19 highest symbol used)
Region A	3 words	erasable
Region D	2 words	constants
Region E	3n words	variables
Region F		auxiliary routine
Region G	n words	erasable except for auxiliary routine
Total:	(46+4n) words	+ auxiliary routine

Duration: 0.6 + 2.2n + I (0.3 + 3.8n + D) ms where

I = number of iterations for an interval,

D = time of auxiliary routine and

n = number of equations.

Method: The numerical procedure, a modified Euler procedure, is given by equations 1, 2, and 3.

$$\dot{y}_{i,j+1,k} = \dot{y}_{i,j} + \frac{\Delta x}{2}\left\{\ddot{y}_{i,j} + \ddot{y}_{i,j+1,k}\right\} \qquad (1)$$

$$y_{i,j+1,k} = y_{i,j} + \Delta x\, \dot{y}_{i,j} + (\Delta x)^2 \left\{\frac{1}{2}-\beta\right\}\left\{\ddot{y}_{i,j}\right\} + (\Delta x)^2\beta\, \ddot{y}_{i,j+1,k} \qquad (2)$$

$$\left.\begin{array}{l} y_{i,j+1} = y_{i,j+1,k} \\ \dot{y}_{i,j+1} = \dot{y}_{i,j+1,k} \\ \ddot{y}_{i,j+1} = \ddot{y}_{i,j+1,k} \end{array}\right] \qquad (3)$$

if for all i the relation

$$(\Delta x)^2 \left|\ddot{y}_{i,j+1,k} - \ddot{y}_{i,j+1,k-1}\right| \leq \epsilon \text{ is satisfied.}$$

In the above equations:

i indicates the variable of integration,

j indicates the step of integration,

k indicates the number of iterations.

The constant β is set equal to 1/6 in * 18 of the library routine. Other values of β in the range $0 \leq \beta \leq 1/4$ are acceptable.

Sufficient conditions to guarantee that the procedure will converge and that numbers stay in bounds can be stated, but use of these conditions leads to extremely short intervals. In practice, if units of measure and a value of Δx are chosen so that numbers are within bounds within the auxiliary routine, the value of Δx so chosen is usually satisfactory for convergence of the numerical procedure.

This program was adapted from University of Illinois Code F3. Program available as J185E.

Nancy Brooks

[illegible]

J186E FIXED POINT SQUARE ROOT WITH DOUBLE PRECISION ARGUMENT

J186 is a "fancy" version of J171E in that it evaluates the positive square root of a double-precision number $a = a_0 + 2^{-39}a_1$.

Calling sequence: a_0 in A.0

a_1 in A.1

```
020   $   0   010 γ
```

Control return: to the left of \$ 1 with the single precision square root x of the double precision number a in the accumulator.

Symbolic Code: 25 words

Region A: 4 words erasable

Error Halt: (* 9 highest symbol used) a_0 negative, or $a_0 = 0$ and a_1 negative

Program available as J186E

Nancy Brooks

J187E LINEAR COMBINATION OF TWO VECTORS -- FLOATING POINT

J187 computes the linear combination $z = \beta x + y$ of the vectors x and y over a Euclidean Vector Space. Here x, y, and z are denoted by the n-tuples $(x_1, \ldots, x_n)$, $(y_1, \ldots, y_n)$, and $(z_1, \ldots, z_n)$. β is a scalar. All of the numbers $\{x_i\}$, $\{y_i\}$, and β must be in the packed internal form recognized by the Floating-Point Interpretive System. Upon exit from J187, the numbers $\{z_i\}$ will be in the same form.

The elements $\{x_i\}$ of the vector x must be placed in H.S.S. as follows:

$L(x_1)$, $L(x_2) = L(x_1) + \triangle_x$, ..., $L(x_n) = L(x_1) + (n-1)\triangle_x$, where $\triangle_x$ is an integer and $|\triangle_x| \leq 377_8 = 255_{10}$. Similar statements hold for y and z.

Calling Sequence:

α	020	α	010	----	(Link to J187)
$\alpha+1$	$\pm$	$L(\beta)$	$\lvert\triangle_x\rvert$	$L(x_1)$	
$\alpha+2$	$\pm$	n	$\lvert\triangle_y\rvert$	$L(y_1)$	
$\alpha+3$	$\pm$	0000	$\lvert\triangle_z\rvert$	$L(z_1)$	
$\alpha+4$	Control returns here				

Here the $\pm$ in steps $\alpha+1$, $\alpha+2$, and $\alpha+3$ correspond only to $\triangle_x$, $\triangle_y$, and $\triangle_z$ respectively. The usage is non-standard in the sense that the words in $\alpha+1$, $\alpha+2$, and $\alpha+3$ must not be complemented in the case of a negative sign. More precisely, for step $\alpha+1$ take

$$\triangle\left[L(x_1)\right] = \begin{cases} |\triangle_x| & \text{if the sign position of the word in } \alpha+1 \text{ is } 0 \\ -|\triangle_x| & \text{if the sign position of the word in } \alpha+1 \text{ is } 1 \end{cases}$$

Steps $\alpha+2$ and $\alpha+3$ can be interpreted in an analagous manner.

Two variations in the use of J187 are permitted. Together with the most general form already described there are three mutually exclusive functions available:

(i) $\beta x + y$, $L(\beta)$, $L(y_1)$ <u>must</u> be non-zero.

(ii) $x + y$, $L(\beta)$ <u>must</u> be zero (blank).

(iii) βx , $L(\beta)$ <u>must</u> be non-zero <u>and</u>
$L(y)$ <u>must</u> be zero (blank).

Essentially, J187 consists of three programs in one. The basic computational loop for each of the three functions described is optimal in the sense that the number of program steps is minimized subject to the condition that the smallest possible number of program steps shall be executed in the Floating-Point system.

Region A (erasable): 12 words (A0 - A11)

Symbolic Code: 65 words

Highest symbol used: * 15

Program available as: J187E.

John Derr

J188E K^x

$K = 2$, e, or 10. X in MQ.

Calling Sequence:

α	020 α 010 XXX	(arbitrary symbol)
$\alpha+1$	$q_1 \cdot 2^{-18} + k \cdot 2^{-39}$	
$\alpha+2$	Control returns here.	

$k = 0$	2^x
1	e^x
2	10^x

leaves K^x in MQ normalized.

leaves $q_2 \cdot 2^{-18}$ in Accumulator where q_2 is scale factor for K^x.

If $q_2 \geq 128$ both the Accumulator and MQ are left zero.

q_1 = location of binary point in X.

$-90 \leq q_1 \leq 37$

Approximate maximum time: 23 ms.

Accuracy: Approximately 9 S.D.

Program stop: 3.009 q_1 exceeds 37

Highest Symbol Used: * 60

Symbolic Code: 61 words

Region A (erasable): 3 words

Program available as J188E

Marvin Shapiro

J189B
J189E IT (INTERPRET T-SWITCHES)

J189 is an <u>open</u> subroutine which decodes the condition of the three T-switches into an eight-way optional transfer. A table of eight transfer orders must be provided in memory locations δ through δ+3 as follows:

LOC	OP	ADDR	OP	ADDR
δ	01_4^0	γ_0	01_4^0	γ_1
$\delta+1$	01_4^0	γ_2	01_4^0	γ_3
$\delta+2$	01_4^0	γ_4	01_4^0	γ_5
$\delta+3$	01_4^0	γ_6	01_4^0	γ_7

Calling sequence: $\begin{pmatrix} 010 \\ 130 \end{pmatrix} \delta) \times 2^{-18}$ in ACC

$\alpha_{L \text{ or } R}$ 010 β

Control return:

To	T_1	T_2	T_3
δ_L	●	●	●
δ_R	●	●	○
$\delta+1_L$	●	○	●
$\delta+1_R$	●	○	○
$\delta+2_L$	○	●	●
$\delta+2_R$	○	●	○
$\delta+3_L$	○	○	●
$\delta+3_R$	○	○	○

● = On
○ = Off

Program length: 7 words {Style B region C
Style E symbolic (* 5 highest symbol)

4 words δ through δ+3 for exit table

Program stop: 134 β+4 in word β+4 as a result of an illegal entry.

Program:

LOC	OP	ADDR	OP	ADDR
β	015	β+1	024	β+5
β+1	024	β+5	016	β+2
β+2	024	β+5	017	β+3
β+3	024	β+6	053	β+4
β+4	---	---	134	β+4
β+5		1		
β+6	004	0		

Program available as J189B
J189E

Nancy Brooks and Mort Bernstein

J190E LOGARITHM - BASE e or 10

2^{-39} log X base e or 10 is developed in A, MQ. MQ_0 is not part of the answer.

Range: $0 < X < 1$

If X = 0 the program will stop at 3.030.

Calling sequence:

α	020	α	010	XXX	

α+1	000	L(X)	000	000^0_1	for base e
					for base 10

α+2	Control returns here				

Region A (erasable): 6 words

Symbolic code: 39 words

Highest symbol used: * 38

Logarithms accurate to 9 decimal places.

Program available as J190E.

Marvin Shapiro

J191E POLYNOMIAL FUNCTION EVALUATION -- FLOATING POINT

For a given value of x, J191 will compute

$$p(x) = \sum_{i=0}^{N} a_i x^i, \; N \geq 1.$$

J191 must be used in conjunction with the current version of the Floating Point System and x, $p(x)$, and the coefficients $a_0, a_1, \ldots, a_N$ must be in the packed internal form. Further, the coefficients must be placed in H.S.S. locations as follows: $L(a_0)$, $L(a_1) = L(a_0) + \Delta$, ---, $L(a_N) = L(a_0) + N\cdot\Delta$, where Δ is an integer and $|\Delta| \leq (377)_8 = (255)_{10}$.

Calling Sequence:

```
α      020     α        010     β           (Link to J191)
α+1     ±      L(a_N)    Δ       N
α+2            L(x)          L[p(x)]
α+3    Control returns here
```

Here, the $\pm$ in step $\alpha+1$ corresponds only to Δ. The usage is non-standard in the sense that the word in $\alpha+1$ must not be complemented in case of a minus sign. More precisely, for step $\alpha+1$ take

$$\Delta[L(a_i)] = \begin{cases} |\Delta| & \text{if the sign position of the word in } \alpha+1 \text{ is } 0. \\ -|\Delta| & \text{if the sign position of the word in } \alpha+1 \text{ is } 1. \end{cases}$$

Method:

$p(x) = p_N(x)$, where $p_N(x)$ is generated by recursively defining $p_1(x) = a_{N-1} + a_N x$ and $p_{i+1}(x) = a_{N-i-1} + xp_i(x)$ for $i=1, 2, ---, N-1$.

Timing:

No absolute times have been recorded, but N floating-point additions and multiplications are performed. Hence, for sufficiently large N the time will vary directly as N.

Region A (erasable): 3 words A0 - A2

Symbolic code: 26 words

Highest symbol used: * 8

Program available as: J191E

John I. Derr

J192E INTEGRATION BY SIMPSON'S RULE WITH PRESCRIBED ERROR FLOATING POINT

J192 computes $\int_a^b f(x)dx$, where f(x) is computed by an auxiliary subroutine written by the programmer. J192 divides the domain from a to b, a < b, into N intervals of length Δ_i, $i = 1, 2, \ldots, N$, and computes the area $\int_i^{(5)}$ for each subinterval by Simpson's 5 point formula. The $\{\Delta_i\}$ are chosen small enough so that the relative error E_M in the interval a_1 to $a_M + \Delta_M$ is less than the error prescribed by the programmer, i.e. so that

$$E_M = \frac{\int_M^{(3)} - \int_M^{(5)}}{\sum_{i=1}^{M} \int_i^{(5)}} < \epsilon, \tag{1}$$

where $\int_i^{(3)}$ = the area over the i th interval as computed by Simpsons' 3 pt. formula.

$\int_i^{(5)}$ = the area over the i th interval as computed by Simpson's 5 pt. formula.

and ϵ = the error prescribed by the programmer.

The maximum error possible over the entire interval from a to b is $N\epsilon \max F(x)$ where $F(x) = \int_a^x f(t)dt$. The method is a modification of that described in Scarborough[1].

1 Scarborough, Numerical Mathematical Analysis, 3rd edition, p. 178.

Calling sequence:

```
α     020   α   010   β
α+1             a
α+2             b
α+3             Δ1
α+4             ε
α+5   Control returns here in floating point
```

a, b, Δ_1, and ϵ must be in the floating point data form. If the starting Δ is not prescribed but is to be determined by J192, step α+3 must be negative.

The computational procedure is as follows:

Define $h_1 = \frac{\Delta_1}{4}$. Let h_1' be a temporary value for h_1. If step α+3 is minus, make $\Delta_1' = b-a$, otherwise make Δ_1' the contents of α+3.

If $E_1 > \epsilon$ replace h_1' by $h_1'/2$. Repeat this halving procedure until either $E_1 \leq \epsilon$ or $h_1'/2 \leq 10^{-4}$ $(a_1 + \Delta_1')$. In both cases the last h_1' is used in computing $\int_1^{(5)}$. In the latter case a row of 11 nines is printed to indicate that the error criterion was not met. This process is used for each interval $i = 1, 2, \ldots, N$.

The constant 10^{-4} is located in the 97th(last) word. Thus the 10^{-4} $(a_M+\Delta_M)$ criterion can be changed by the programmer to fit a particular problem. If $500\ E_1 \leq \epsilon$, set $h_{1+1}' = 2h_1$. The constant 500, located in word 96 can also be changed.

The auxiliary closed subroutine is entered by J192, with x in the floating point AMQ by means of the calling sequence:

```
γ     020   γ   010   F   0
γ+1   Control returns here in floating point
```

At step γ+1 f(x) should be located in the AMQ and the floating point system should be in control, i.e. the exit from the auxiliary subroutine must not be made with an 010 (EXL) order.

The final exit from J192 to step $\alpha+5$ is made with the floating point system in control.

If cell A0 is positive, after the computation on the Mth interval is completed, the following 2 lines of information will be printed in the Floating Point print format:

A	B	C
$\sum_{i=1}^{M} \int_i^{(5)}$	a	$a_M+\Delta_M$
$\int_M^{(5)}$	a_M	Δ_M

where a_M and $a_M+\Delta_M$ are the first and last points of the Mth subinterval. Hence $a_1=a$ and $a_N+\Delta_N=b$.

If cell A0 is negative after the computation on the Mth interval is completed, there will be no printing. On exit from J192 $\sum_{i=1}^{N} \int_i^{(5)}$ will be in cell B0.

Symbolic code: 97 words

Region A (erasable): 8 words, A0 to A7

Region B (semi-erasable): 19 words, B0 to B18.
Region B is erasable except for the f(x) routine.

Region F: f(x) routine

Highest Symbol Used: * 30

Program available as J192E.

Marvin Shapiro

J193E FLOATING POINT DATA DUMP

The contents of the memory between specified locations, a and b, are printed in the floating point data format. Each line consists of two words and the location of the first word. The location is in octal or decimal and is preceded by 7 extra zeros. If both the exponent and the mantissa of both of the data words on a line are zero (i.e., if both of the words are machine zeros), that line is not printed.

J193 uses the JOHNNIAC floating point interpreter and the right address of the 20th word of J193 must be the entry location of the interpreter. It is input as 5620_8, the appropriate location for use in connection with the current version of the Floating Point System.

Calling Sequence:

```
α      020   α   010   XXXX   (Link to J193)
α+1    k00   a   000    b
α+2    Control returns to left of α+2

             k = 0 for octal locations
             k = 1 for decimal locations
```

Region A (erasable): A0 to A9

Symbolic Code: 37 words

Program available as J193E

Highest Symbol Used: * 12

Norman Shapiro

J194E ITERATED SYNTHETIC DIVISION - FLOATING POINT

Given a polynomial $f(x) = \sum_{i=0}^{N} a_i x^i$, J194 will divide $f(x)$ by $x-c$ M times, where $1 \leq M \leq N$. More precisely, define the quotient polynomials $q_i(x)$ by recursion. Set $q_0(x) = f(x)$ and let $q_j(x)$ be defined by $q_{j-1}(x) = (x-c)\, q_j\,(x) + q_{j-1}(c)$ for $j=1, 2, \ldots, M$. Now let

$$q_M(x) = \sum_{j=0}^{N-M} q_j^M x^j \text{ define } \left\{q_j^M\right\}.$$ Then J194 computes the set of N numbers $\left\{b_i\right\}$, where $b_i = \begin{cases} q_i(c), & i=0,1,\ldots,M-1 \\ q_{i-M}^M, & i=M,M+1,\ldots,N \end{cases}$.

All of the numbers $\{a_i\}$, $\{b_i\}$, c, $\{q_i(c)\}$, and $\{q_{i-M}^M\}$ are in the packed internal form recognized by the current version of the floating point system. The $\{a_i\}$ and $\{b_i\}$ must be arranged in H.S.S. as follows:

$L(a_0)$, $L(a_1) = L(a_0) + \triangle_a$, $\ldots$, $L(a_N) = L(a_0) + N\cdot\triangle_a$.

$L(b_0)$, $L(b_1) = L(b_0) + \triangle_b$, $\ldots$, $L(b_N) = L(b_0) + N\cdot\triangle_b$.

Here, $\triangle_a$ and $\triangle_b$ are integers whose absolute values are bounded below $(400)_8 = (256)_{10}$.

<u>Calling Sequence</u>:

α	020	α	010	β	(Link to J194)
$\alpha+1$	a00	N	$\lvert\triangle_a\rvert$	$L(a_0)$	
$\alpha+2$	b00	M	$\lvert\triangle_b\rvert$	$L(b_0)$	
$\alpha+3$		$L(c)$			
$\alpha+4$	Control returns here				

$$\Delta_a = \begin{cases} |\Delta_a| & \text{if } a = 0 \\ -|\Delta_b| & \text{if } a = 1 \end{cases} \quad \text{and } \Delta_b = \begin{cases} |\Delta_b| & \text{if } b = 0 \\ -|\Delta_b| & \text{if } b = 1 \end{cases}$$

serve to define a and b .

Notes

(i) Making use of the general relationship $f^{(k)}(c) = k!\cdot q_k(c)$ for $0 \leq k \leq N$, one can use J194 to compute the values of the derivatives of a polynomial $f(x)$ evaluated at $x = c$. In particular, for $k=1$ J194 duplicates the function of J191.

(ii) If $M = N$, then the $\{b_i\}$ computed by J194 also satisfy the equation

$$f(x) \overset{\text{def}}{=\!=} \sum_{i=0}^{N} a_i x^i = \sum_{i=0}^{N} b_i (x+c)^i \overset{\text{def}}{=\!=} g\ (x+c).$$

(iii) The $\{a_i\}$ are not destroyed by J194 if $\{L(a_i)\}$ are disjoint from $\{L(b_i)\}$. However, setting $\Delta_a = \Delta_b$ and $L(a_0) = L(b_0)$ is permissible if the $\{a_i\}$ are expendable.

Timing

$\frac{N!}{(N-M)!}$ pairs of floating-point add and multiply orders are performed for each entry.

Region A (erasable): 7 words A0 - A6

Symbolic Code: 46 words

Highest Symbol Used: * 10

Program available as: J194E

John Derr

?

J195E MAXIMUM ELEMENT OF A VECTOR -- FLOATING POINT

J195 is a floating-point version of J174E. Let $x = (x_1, \ldots, x_n)$ be an n-tuple with real components. J195 will find the element $x_k \in \{x_i\}$ such that $|x_k| \geq |x_i|$ for $i=1, \ldots, n$ and if for some $\ell \neq k$, $|x_\ell| \geq |x_i|$ for $i=1, \ldots, n$, then $k < \ell$. The element x_k will be in A4 upon exit from J174. ? In addition, $L(x_k) \cdot 2^{-39}$ will be in the $\overset{\text{accumulator}}{\text{A}}$ upon exit from J195.

The elements $\{x_i\}$ of the vector x must be in the packed internal form recognized by the Floating Point system and must be placed in H.S.S. as follows:

$L(x_1)$, $L(x_2) = L(x_1) + \Delta$, ..., $L(x_n) = L(x_1) + (n-1)\Delta$, where Δ is an integer and $|\Delta| \leq 377_8 = 255_{10}$.

Calling Sequence:

```
α      020     α     010    β              (Link to J195)
α+1     ±      n     |Δ|    L(x₁)
α+2    Control returns here
```

Here the $\pm$ in step $\alpha+1$ corresponds only to Δ. The usage is non-standard in the sense that the word in $\alpha+1$ must not be complemented in the case of a negative sign. More precisely, for step $\alpha+1$ take

$$\Delta[L(x_i)] = \begin{cases} |\Delta| & \text{if the sign position of the word in } \alpha+1 \text{ is 0.} \\ -|\Delta| & \text{if the sign position of the word in } \alpha+1 \text{ is 1.} \end{cases}$$

Region A (erasable): 5 words (A0 -- A4)

Symbolic Code: 25 words

Highest symbol used: * 8

Program available as J195E

John Derr

J196E SOLUTION OF A MATRIX EQUATION AX = B WHERE A IS n x n AND NON-SINGULAR -- FLOATING POINT

A link to J196 will result in the solution of the k systems of n simultaneous linear equations

$$\sum_{j=1}^{n} a_{ij} x_j^{(\ell)} = b_i^{(\ell)}, \ i = 1, 2, \ldots, n,$$

in n unknowns for $\ell = 1, \ldots, k$. If we define the matrices A, X, B by

$$A = \begin{pmatrix} a_{11} & a_{12} & \cdots & a_{1n} \\ a_{21} & & & \vdots \\ \vdots & & & \vdots \\ a_{n1} & & \cdots & a_{nn} \end{pmatrix}, X = \begin{pmatrix} x_1^{(1)} & x_1^{(2)} & \cdots & x_1^{(k)} \\ x_2^{(1)} & & & \vdots \\ \vdots & & & \vdots \\ x_n^{(1)} & & \cdots & x_n^{(k)} \end{pmatrix}, B = \begin{pmatrix} b_1^{(1)} & b_1^{(2)} & \cdots & b_1^{(k)} \\ b_2^{(1)} & & & \\ \vdots & & & \\ b_n^{(1)} & & \cdots & b_n^{(k)} \end{pmatrix}$$

then the two statements concerning the function of J196 are equivalent. J196 also computes the determinant of A which we denote as det A.

Calling Sequence:

```
α      020    α    010    β        (Link to J196)
α+1     k     n     Δ     L(a11)
α+2    Control returns here if det A = 0
α+3    Control returns here if det A ≠ 0
```

Before linking into J196 the user must place the elements a_{ij} of A and $b_i^{(\ell)}$ of B into H.S.S. locations as follows:

$$\begin{array}{l} a_{11} \ a_{12} \ \cdots \ a_{1n} \ b_1^{(1)} \ b_1^{(2)} \ \cdots \ b_1^{(k)} \ \_\_\_ \\ a_{21} \ a_{22} \qquad\qquad\qquad\qquad\qquad \vdots \\ \vdots \\ a_{n1} \ a_{n2} \ \cdots \ a_{nn} \ b_n^{(1)} \ b_n^{(2)} \ \cdots \ b_n^{(k)} \ \_\_\_ \end{array}$$

Basically, these locations differ by Δ. More precisely,

$$\text{loc } (a_{i,j+1}) = \text{loc } (a_{ij}) + \Delta, \; j=1, 2, \ldots, n-1,$$
$$\text{loc } (a_{i,1}) = \text{loc } (b_{i-1}{}^{(k)})+2\Delta, \; i \neq 1,$$
$$\text{loc } (b_i{}^{(1)}) = \text{loc } (a_{i,n}) + \Delta, \text{ and}$$
$$\text{loc } (b_i{}^{(\ell+1)}) = \text{loc } (b_i{}^{(\ell)}) + \Delta, \; \ell=1, 2, \ldots, k-1.$$

Upon exit from J196 these same H.S.S. locations will contain

$$\begin{array}{l}
\alpha_{11} \; \alpha_{12} \cdots \alpha_{1n} \; x_1{}^{(1)} \; x_1{}^{(2)} \cdots x_1{}^{(k)} \quad c_1 \\
\alpha_{21} \; \alpha_{22} \cdots \qquad\qquad\qquad\qquad\qquad\quad \vdots \\
\vdots \\
\alpha_{n1} \; \alpha_{n2} \cdots \alpha_{nn} \; x_n{}^{(1)} \; x_n{}^{(2)} \cdots x_n{}^{(k)} \quad c_n
\end{array}$$

In addition, det A will be in A 19. All of the numbers referred to are in the packed internal form recognized by the Floating-Point Interpretive System. The quantities k, n, and Δ, in the Calling Sequence are positive integers with the obvious upper bounds $(77)_8$, $(7777)_8$, and $(377)_8$, respectively. Clearly, these are not least upper bounds.

The matrix $\alpha = (\alpha_{ij})$ is the n x n matrix obtained from A by the forward solution part of the elimination process. The n x 1 matrices $\left(x_1{}^{(\ell)}, x_2{}^{(\ell)}, \ldots, x_n{}^{(\ell)}\right)$ are obtained in the back solution part of the method. The n x 1 matrix $(c_1, c_2, \ldots c_n)$ should be ignored by the user. It functions only as a check on the computational accuracy of the forward and backward solutions.

Description of the Method

The basic method used is the Gauss elimination method. In the forward solution the matrix A is reduced to a triangular matrix T. The solution of $Tx = Eb$ is equivalent to the solution of the original equation $Ax = b$, where $E = \prod_i E_i$ and the E_i are elementary matrices, i.e. $E_i M$ results in performing elementary row operations on M. The back solution can be performed by a simple substitution using the equation $Tx = Eb$.

The rule of formation of the sequence $(E_1, E_2, \ldots)$ is known as Crout's Method*. However, here the method is modified so as to permute the rows of the resulting matrix, if necessary, to use the element $\alpha_{ki} = \underset{i \leq j \leq n}{\text{Max}} \left(|\alpha_{ji}|\right)$ as the α_{ii} pivot element for $i = 1, 2, \ldots, n$.

The computation of the i^{th} row of T is checked versus $(i+1)^2 \cdot 10^{-6}$ according to the Crout criteria for checking. Similarly, the calculation of the i^{th} row of X is checked versus $(n+1)^2 \cdot 10^{-6}$. This method of checking will probably suffice in most cases w. r. t. testing for machine malfunctions. Note that this check does not guarantee that the solution computed is "correct" in any normal sense, since the ultimate accuracy of the computed solution is limited by the inherent error of the original system of equations. An exact specification of bounds for inherent error is beyond the scope of J196. However, loss of significant digits due to ill-conditioning of the original matrix A can cause the checking criteria not to be satisfied.

*See Numerical Calculus, pp. 17-29, by W. E. Milne

NOTES

(i) J172, 3, 5 and J195 (or J174 in place of J195 if all calculations are performed in the N mode) are slave routines w.r.t. J196 and therefore must be used along with the Floating-Point System in conjunction with J196. Further, J172, 3, 5 and J195 (or J174) have the origins WO, XO, ZO, and YO, respectively, w.r.t. J196. Hence, these regions must be assigned locations before J196 is loaded.

(ii) A 13X halt will occur in the 120th word of J196 if the computational checking criteria is not met in any row during the forward solution. The calculation will continue if the GO button is pressed.

(iii) A 13X halt will occur in the 183rd word of J196 if the computational checking criteria is not met in any row during the back solution. The calculation will continue if the GO button is pressed.

(iv) A 13X halt will occur in the 75th word of J196 if det $A=0$. Control will link back to the main program if the GO button is pressed.

(v) Clearly, the original matrices A and B are destroyed by J196.

(vi) The relation

$n(n+k+1) \leq (4096-F) - 292 - 26 - M$

must always be satisfied when using J196, where $F =$ the number of words used by the Floating-Point Interpretive System and $M =$ the number of words in H.S.S. occupied by programs other than J172, 3, 5 and J195 and the Floating-Point System.

In order to get some real-live numbers, suppose $(4096 - F) = 2960$, $\Delta = 1$, and $M = 115$. Then $n(n+k+1) \leq 2527$ must hold. If $k=1$, $n \leq 49$. If $k=n$, $k=n\leq 35$.

<u>Timing</u>

Since at the time of this writing no explicit times were known for the floating-point operations, it follows that no explicit timing estimates could be made for the general case. However, the following floating-point operation counts should yield a reasonable measure of how the time will vary as a function of n and k.

Operation Counts for the Forward Solution

(÷) $\frac{n(n+2k+1)}{2}$

(+) and (x) $\frac{n^3}{3} + \frac{k\ n^2}{2}$ for each (approximate for large n or k)

Operation Counts for the Backward Solution

(+) and (x) $\frac{n(n-1)(k+1)}{2}$ for each

While operating in the SD Mode five seconds were required for a solution with $n=7$ and $k=1$. This example can be used as a reference point in the (n,k) plane to compute approximate absolute times using the above formulae.

<u>J196 Storage (without slaves)</u>

Reg. A (erasable) 20 words (A 6 - A 25)

Symbolic Code: 192 words

Highest Symbol Used: * 41.

<u>Storage Used by Slave Routines</u> (J172,3,5 and J195 [or J174])

Region A 6 words (A 0 - A 5) (Note that the erasable storage used by J172,3,5 and J195 (or J174) does not overlap that used by J196.)

Symbolic Code: 100 words (or 98 words if J174 used in place of J195)

Highest Symbol Used: * 8

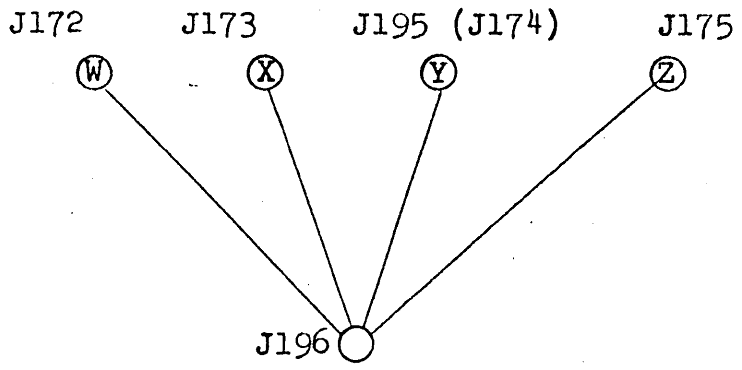
<u>Combined Storage Used by J172,3,5 and J195 (or J174)</u>

Reg. A 26 words (A 0 - A 25)

Symbolic Code: 292 words (or 290 words if J174 is used in place of J195)

Highest Symbol Used: * 41

<u>Lattice Diagram Describing Master-Slave Relations</u>



J197A AUTOMATIC SOLUTION OF A MATRIX EQUATION AX=B WHERE A IS n x n AND NON-SINGULAR -- FLOATING POINT

J197 is an automatic version of J196. The user needs only to record the necessary input data in punched cards, place these cards immediately behind J197, and press the LOAD button. The solution matrix, X*, which approximates X in the equation AX=B will be computed and printed for each equation.

<u>Input Form</u>

Information should be punched in the Floating Point Data Form as follows:

<u>1st card for a problem</u>

(1)

±	EXP	MANTISSA
	n	ID

ID: Any decimal information for problem ID

n: order of matrix A

±: + if A is not to be printed; − if A is to be printed

(2)

±	EXP	MANTISSA
	k	———

k: number of columns in the matrices X and B

±: + if (α_{ij}) is not to be printed; − if (α_{ij}) is to be printed

The elements of A must be punched by rows into consecutive positions of the cards beginning with the second card. Each row must begin with position (1) of a card and if an EF mark (12 punch (+) in column 80) is encountered prior to the reading of the nth element of a given row, then J197 will assume that the remaining elements of the row are 0. (This does not, of course, affect the six elements of A which were punched

into the card with the EF mark.)

The elements of B must be punched by columns into consecutive positions of the cards beginning with the first card following the cards for A. Each column must begin with position (1) of a card.

The solutions for a sequence of equations AX=B can be evaluated by placing the input cards for the equations in sequential order behind J197. If the input cards are followed by two blank cards, the collator will be on Select following the completion of the last problem.

Output Form

The page will be ejected prior to the beginning of each problem. The numbers printed will be in the external floating point form. The print format is that described for the 013 (PNT) order in the Floating-Point writeup. The printing for one problem takes place as follows:

	A	B	C
Information in positions (1) and (2) of 1st Input Card	——	± n ID	± k ——
Space	——	——	——

If A is printed, then the printing for the i^{th} row is as follows:

Row number	——	i	——
Elements of the row begin here	a_{11}	a_{12}	a_{13}
	⋮	⋮	⋮

If A is printed, the page is ejected following the printing of the last row.

	A	B	C
Determinant	——	——	det A
Space	——	——	——

If (α_{ij}) is printed, the same procedure is followed as for A.

X is printed by columns The printing for the ℓth column is as follows:

Column number	ℓ	——	——
Elements of the column begin here	$x_1^{(\ell)}$	$x_2^{(\ell)}$	$x_3^{(\ell)}$
	$\vdots$	$\vdots$	$\vdots$

Notes:

(1) The inequality $n(n+k+1) \leq 2526$ must be satisfied by n and k. In particular, if k=1, then n must not exceed 49 and if k=n, then n must not exceed 35. If $n(n+k+1) > 2526$, the program will halt at 5474_8 with a 130 5474 order.

(2) A 13X halt will occur at 5334_8 if the computational checking criteria is not met in any row during the forward solution. The calculation will continue if the GO button is pressed.

(3) A 13X halt will occur at 5433_8 if the computational checking criteria is not met in any row during the back solution. The calculation will continue if the GO button is pressed.

(4) A 13X halt will occur at 5257_8 if det A=0. If the GO button is pressed, then det A and the current state of the (α_{ij}) and X (or B) matrices will be printed. The next

problem in sequence will then be started if it exists.

(5) A check sum BX halt at 0014_8 indicates that J197 has probably been loaded incorrectly under the control of J180A. The entire loading process should be restarted.

(6) The user is referred to the J196 write-up for a description of the method.

Program available as J197A.

John Derr

J198E SERIAL DRUM SIMULATOR

J198 transfers any block of information from 1 to 9216 words in length to or from HSS from or to the drum. "Stepping over" bands and positions is done automatically.

Calling sequence:

```
        α      020    α      010     β
        α+1                  a·2^-39
        α+2                  b·2^-39
        α+3                  00c     n
        α+4    Control returns here
```

where a = first drum address

b = last drum address

$c = \begin{cases} 0 \text{ if transferring from drum to HSS} \\ 1 \text{ if transferring from HSS to drum} \end{cases}$

n = first memory address

Words are transferred to or from HSS modulo the address 4096.

Starting at 0000 and going to 9215 inclusive, the order of the drum positions and bands used is as follows:

Drum Addresses	Position	Band
0000-1023	0	0
1024-2047	0	2
2048-3071	2	1
3072-4095	1	0
4096-5119	2	0
5120-6143	1	2
6144-7167	0	1
7168-8191	2	2
8192-9215	1	1

The fact that consecutively numbered blocks of words are not stored on consecutively numbered positions or bands need not concern the programmer as long as he uses J198 for both reading

J199E SERIAL DRUM SIMULATOR WITH CHECK SUM

J199 transfers any block of information from 1 to 9216 words in length to or from HSS from or to the drum. "Stepping over" bands and positions is done automatically. On exiting from J199 the sum, modulo 2^{40}, of the block of words transferred to or from the drum is left in the accumulator.

Calling sequence:

α	020	α	010	β
$\alpha+1$			$a \cdot 2^{-39}$	
$\alpha+2$			$b \cdot 2^{-39}$	
$\alpha+3$			00c	n
$\alpha+4$	Control returns here			

where a = first drum address

b = last drum address

c = 0 if transferring from drum to HSS
    1 if transferring from HSS to drum

n = first memory address

Words are transferred to or from HSS modulo the address 4096.

Starting at 0000 and going to 9215 inclusive, the order of the drum positions and bands used is as follows:

Drum Addresses	Position	Band
0000-1023	0	0
1024-2047	0	2
2048-3071	2	1
3072-4095	1	0
4096-5119	2	0
5120-6143	1	2
6144-7167	0	1
7168-8191	2	2
8192-9215	1	1

The fact that consecutively numbered blocks of words are not stored on consecutively numbered positions or bands need not concern the programmer as long as he uses J199 for both reading and writing. This ordering must, of course, be taken into account if the contents of the drum are dumped in the normal position and band order, i.e. (0,0), (0,1), (0,2), (1,0), ..., (2,2).

Region A (erasable) - 7 words, A0 to A6

Symbolic Code: 44 words

Highest Symbol Used: * 19

Program Stops: None

Program available as J199E.

Marvin B. Shapiro

J200B
J200F

JOHNNIAC FLOATING-POINT INTERPRETIVE SYSTEM

Program Storage: 2960_{10} - 4095_{10} (5620_8 - 7777_8).

All of the Floating Point Operations are recognized in their mnemonic forms by J100, J118.

LIST OF FREQUENTLY USED LOCATIONS

Dec.	Octal	Contents
2960	5620	First Location of Interpreter
2963	5623	020 [I CTR.] 050 5644
2980	5644	I Register
4041	7711	Error Halt Location
3040	5740	Trap Register - BRKPT
3041	5741	" " - All Orders
3042	5742	" " - Transfer
3656	7110	AMQ Exponent
3658	7112	AMQ Mantissa
3708	7174	072 [q_1] 025 7016 - Square Root Shift*
3733	7225	022 7122 076 [q_2] - Series Shift*
3999	7637	Series Test Number*
3064	5770	Index Register A
3056	5760	" " B
3052	5754	" " C
3050	5752	" " D
3049	5751	" " E
3048	5750	" " F
3597	7015	Integer - Zero
3598	7016	" - One
⋮	⋮	⋮ - ⋮
3606	7026	Integer - Nine
3615	7037	" - Fifty
3634	7062	" - Ten exp zero
3635	7063	" - " " One
⋮	⋮	⋮ - ⋮ ⋮ ⋮
3643	7073	Integer - Ten exp. Nine

* Let $\Delta_n = f_{(n+1)} - f_n$, where f_n is the nth approximation to the function f. The test for convergence of f_{n+1} to f

is then made to depend upon the sign of $|\Delta_n| \cdot 2^{-q} - Z$. Z is always 2^{-39} for the SQR operation. For the ART, EXP, and LOG operations Z is the series test number. The series test number is ordinarily equal to 2^{-39} for these three operations, but it can be modified by the user. However, the series test number is usually reset to 2^{-39} upon execution of a SIN or COS operation. For the SQR operation q is q_1 and q_1 is ordinarily set to 5. However, q_1 can be modified by the user. The same situation holds for the SIN, COS, ART, LOG, and EXP operations with q equal to q_2.

Coding Aids Using J200F

1. To normalize the number in the AMQ, execute the following instructions outside interpreter control:

```
α-1    ---    ---    004    7112
α      020    α      010    7575   (Link to count number of SD.
                                     Leaves number of SD in MQ.)
α+1    020    α+1    010    7624   (Link to normalize mantissa)
α+2    Control returns here
```

2. To suppress Error Halt:

   Replace the left operation of 7711 by 010.

3. To suppress Error Halt and the subsequent trace printing, replace the left operation and left address of 7711 by 010 5667.

Program available as J200B
J200F

John Derr

J201B
J201E

PUNCH DECIMAL CARDS

J201, punches one decimal card containing 6 twelve-place fields and a three-digit sequence number.

Calling sequence:

α	020	α	010	XXX
$\alpha+1$	—	—	—	SEQ
	0 1-6	7-19	20-27	28-39
$\alpha+2$	$k_1 p_1$	q_1	—	L_1
$\alpha+3$	$k_2 p_2$	q_2	—	L_2
$\alpha+4$	$k_3 p_3$	q_3	—	L_3
$\alpha+5$	$k_4 p_4$	q_4	—	L_4
$\alpha+6$	$k_5 p_5$	q_5	—	L_5
$\alpha+7$	$k_6 p_6$	q_6	—	L_6
$\alpha+8$	Control returns here			

where:

k_i = 0 (in sign position) if p and q positive
1 (if the negative values of p_i and q_i are to be used)

p_i positive = number of decimal digits to the left of the decimal point.
negative = number of lead zeros desired to be between the decimal point and the first digit of the word in memory.

q_i positive = number of positions right from 2^0 (excluding 2^0) to binary point.
negative = number of positions left from 2^0 (excluding 2^0) where binary point is assumed to be.

L_i = location of the i th number.

continued

The punched card form is:

Column	
1	blank
2-4	sequence number
5-16	1st number
17-28	2nd number
29-40	3rd number
41-44	blank
45-56	4th number
57-68	5th number
69-80	6th number

If p_i and q_i are positive field, i contains 11 digits and a decimal point after the integer part of the number.

If p_i and q_i are negative, the absolute value of p_i is punched in the first column of the field followed by 11 digits.

An 11 is overpunched in the 12th column of the field for negative numbers. No punch is used to designate positive numbers.

If any cell in $\alpha+2$ through $\alpha+7$ is left blank (zero), the corresponding field on the card will be blank.

Program length: Region A (erasable) - 30 words
Region C - 171 words

No program stops.

Timing: Cards will be punched at half-speed with a reasonable amount of manipulation between entering of the program.

Program available as J201B
J201E

Marvin Shapiro

9-12-56

J202A DUMP MEMORY TO DRUM

J202A is a one-card self-loading routine which clobbers cells 0000-0002, transfers HSS cells $(3\text{-}1023_{10})$* to $(3\text{-}1023_{10})$* of the drum (position 0, band 0)* and then effectively hits the load button. (The MQ is not cleared.)

* Note: The action of J202A can be changed in several ways. (For example, any band position combination may be used.) In particular, the right half of the 8's row and the left address of the right half of the 6's row are flexible.

Norman Shapiro

NS:reg

9-12-56

J203E INPUT ADDRESSABLE FLOATING-POINT DATA

J203E will read cards containing addressable floating point data punched in the external floating-point form as described in JOHNNIAC Note #66.

Calling Sequence:

```
        α      020   α    010   β       Link to J203E
        α+1    k00   a
        α+2    Control returns to left of α+2
```

Cards will be read from the primary if k=0 and from the secondary if k=1. Locations will be relative to a .

J203 <u>does not</u> require the presence of a floating-point interpreter.

<u>Note</u>: J203E will destroy the origins of regions B, C, D while loading, and must be loaded by J136A. J203 is available only in style E. The J203E deck consists of 146 cards.

Timing: Cards are read at full speed. Each reference to J203 will lose one card cycle.

Region A (Erasable): A0 - A40

Program: 132 words

Highest Symbol Used: *99

Program available as J203E

Norman Shapiro

J204A AUTOMATIC FLOATING POINT DATA DUMP

J204A is a self-loading program which effectively uses only the first three cells of HSS (0000-0002). The contents of band 0, position 0 of the drum are clobbered.

J204A requires the presence of the current version of the floating point system in H.S.S. If this condition is not already met, it can be attained by placing a style F version of the system between J204A002 and J204A003. The floating-point system will then be loaded into memory and the information previously present in the cells used by the system will be destroyed.

Any number of specified sections of memory may be printed. Each line printed consists of two words in the floating-point data format and the location of the first word. The locations may be in octal or in decimal and are preceded by 7 extra zeros. If both of the data words on a line are machine zeros, that line is not printed.

The sections to be printed are determined by one or more control cards, having the first and last addresses in the address positions of the right half of the 9's row. A 9 punch in col. 41 causes decimal locations to be printed. The absence of a 9 punch in col. 41 causes octal locations to be printed.

Printing is terminated by a blank card. After reading the blank card, J204A effectively hits the load button.

Preparation of Deck:

J204A
Control cards
Blank card.

<u>HALT</u> NIA = 0015_8. Check sum stop. J204A has not read in correctly and portions of the memory may have been clobbered.

<u>Drum Used</u>: Position 0, band 0

Available as J204A.

Norman Shapiro

J205B
J205F

JOHNNIAC FLOATING-POINT INTERPRETIVE SYSTEM

Program Storage: $2960_{10} - 4095_{10}$ ($5620_8 - 7777_8$).

All of the Floating Point Operations are recognized in their mnemonic forms by J100, J118.

LIST OF FREQUENTLY USED LOCATIONS

Dec.	Octal	Contents
2960	5620	First Location of Interpreter
2963	5623	020 [I CTR.] 050 5644
2980	5644	I Register
4041	7711	Error Halt Location
3040	5740	Trap Register - BRKPT
3041	5741	" " - All Orders
3042	5742	" " - Transfer
3656	7110	AMQ Exponent
3658	7112	AMQ Mantissa
3708	7174	072 [q_1] 025 7016 - Square Root Shift*
3733	7225	022 7122 076 [q_2] - Series Shift*
3999	7637	Series Test Number*
3064	5770	Index Register A
3056	5760	" " B
3052	5754	" " C
3050	5752	" " D
3049	5751	" " E
3048	5750	" " F
3597	7015	Integer - Zero
3598	7016	" - One
⋮	⋮	⋮ - ⋮
3606	7026	Integer - Nine
3615	7037	" - Fifty
3634	7062	" - Ten exp zero
3635	7063	" - " " One
⋮	⋮	⋮ - ⋮ ⋮ ⋮
3643	7073	Integer - Ten exp. Nine

* Let $\Delta_n = f_{(n+1)} - f_n$, where f_n is the nth approximation to the function f. The test for convergence of f_{n+1} to f

is then made to depend upon the sign of $|\Delta_n| \cdot 2^{-q} - Z$. Z is always 2^{-39} for the SQR operation. For the ART, EXP, and LOG operations Z is the series test number. The series test number is ordinarily equal to 2^{-39} for these three operations, but it can be modified by the user. However, the series test number is usually reset to 2^{-39} upon execution of a SIN or COS operation. For the SQR operation q is q_1 and q_1 is ordinarily set to 5. However, q_1 can be modified by the user. The same situation holds for the SIN, COS, ART, LOG, and EXP operations with q equal to q_2.

Coding Aids Using J205F

1. To normalize the number in the AMQ, execute the following instructions outside interpreter control:

```
    α-1    ---    ---    004    7112
    α      020    α      010    7575   (Link to count number of SD.
                                        Leaves number of SD in MQ.)
    α+1    020    α+1    010    7624   (Link to normalize mantissa)
    α+2    Control returns here
```

2. To suppress Error Halt:
   Replace the left operation of 7711 by 010.
3. To suppress Error Halt and the subsequent trace printing, replace the left operation and left address of 7711 by 010 5667.

Program available as J205B
J205F

John Derr

J206F JOHNNIAC FLOATING-POINT INTERPRETIVE SYSTEM #2

Program Storage: 2960_{10} - 4095_{10} (5620_8 - 7777_8).

All of the Floating Point Operations are recognized in their mnemonic forms by J100, J118.

LIST OF FREQUENTLY USED LOCATIONS

Dec.	Octal	Contents
2960	5620	First Location of Interpreter
2963	5623	020 [I CTR.] 050 5644
2980	5644	I Register
4041	7711	Error Halt Location
3040	5740	Trap Register - BRKPT
3041	5741	" " - All Orders
3042	5742	" " - Transfer
3656	7110	AMQ Exponent
3658	7112	AMQ Mantissa
3708	7174	072 [q_1] 025 7016 - Square Root Shift*
3733	7225	022 7122 076 [q_2] - Series Shift*
3999	7637	Series Test Number*
3064	5770	Index Register A
3056	5760	" " B
3052	5754	" " C
3050	5752	" " D
3049	5751	" " E
3048	5750	" " F
3597	7015	Integer - Zero
3598	7016	" - One
⋮	⋮	⋮ ⋮ ⋮
3606	7026	Integer - Nine
3615	7037	" - Fifty
3634	7062	" - Ten exp zero
3635	7063	" - " " One.
⋮	⋮	⋮ - ⋮ ⋮ ⋮
3643	7073	Integer - Ten exp. Nine

* Let $\Delta_n = f_{(n+1)} - f_n$, where f_n is the nth approximation to the function f. The test for convergence of f_{n+1} to f

is then made to depend upon the sign of $|\triangle_n| \cdot 2^{-q} - Z$. Z is always 2^{-39} for the SQR operation. For the ART, EXP, and LOG operations Z is the <u>series test number</u>. The series test number is ordinarily equal to 2^{-39} for these three operations, but it can be modified by the user. However, the series test number is usually reset to 2^{-39} upon execution of a SIN or COS operation. For the SQR operation q is q_1 and q_1 is ordinarily set to 5. However, q_1 can be modified by the user. The same situation holds for the SIN, COS, ART, LOG, and EXP operations with q equal to q_2.

Coding Aids Using J206

1. To normalize the number in the AMQ, execute the following instructions outside interpreter control:

```
α-1    ---   ---   004   7112
α      020   α     010   7575  (Link to count number of SD.
                                Leaves number of SD in MQ.)
α+1    020   α+1   010   7624  (Link to normalize mantissa)
α+2    Control returns here
```

2. To suppress Error Halt:
   Replace the left operation of 7711 by 010.
3. To suppress Error Halt and the subsequent trace printing, replace the left operation and left address of 7711 by 010 5667.

Program available as J206F

Norman Shapiro

J207A AUTOMATIC POLYNOMIAL FUNCTION EVALUATION - FLOATING POINT

J207 is an automatic version of J191. The user needs only to insert the current version of the JOHNNIAC Floating-Point System between the second and third cards of J207, to record the necessary input data in punched cards, place these cards immediately behind J207, and press the LOAD button. The quantities p(x), where

$$p(x) = \sum_{i=0}^{N} a_i x^i, \ N \geq 1, \text{ will be computed and printed}$$

for each header card.

Input Form

Information should be punched in the Floating Point Data Form as follows:

Header card for a problem

(1)	(2)		(3)
x	$\pm$ EXP	MANTISSA	
	+ 00	N	d

d: < 0 if the $\{a_i\}$ are to be printed.
≥ 0 if the $\{a_i\}$ are not to be printed.

Positions (4), (5), and (6) of the header card are not interpreted by J207.

Coefficients $\{a_i\}$

If new coefficients $\{a_i\}$ are to be used for this problem, then they must be punched 6/card into consecutive positions of the cards beginning with the second card. Each row must begin with position (1) of a card.

End-of-file card for a problem

The cards for one problem begin with a header card and end with a card containing an end-of-file mark (12 punch, +) in col. 80 (called an end-of-file card). The end-of-file card usually coincides with the last coefficient card, but if the coefficients from a previous problem are being used again (for example, if x is a parameter), the end-of-file card can coincide with the header card.* In any event, though, there must be an end-of-file card for each problem.

Sequence of problems

A sequence of polynomial function evaluations can be performed by placing the input cards for the individual problems in sequential order behind J207. If the input cards are followed by two blank cards, the console will show "READER" following the completion of the last problem.

Output Form

The page will be ejected prior to the beginning of the first problem. The numbers printed will be in the external floating-point form. The print format is that described for the 017 (PNT) order in the Floating-Point write-up. The printing for one problem takes place as follows:

A	B	C
p(x)	x	N

---

* The storage reserved for the $\{a_i\}$ is cleared to zero prior to beginning the first problem and that storage is altered only by reading in $\{a_i\}$.

If the $\{a_i\}$ are to be printed ($d < 0$), they are printed 3/line in the natural order:

Space	A	B	C
Coefficients	a_0	a_1	a_2
	$\vdots$	$\vdots$	$\vdots$

One space is skipped at the end of each problem.

Program available as J207A.

John Derr and Norman Shapiro

J208A AUTOMATIC SOLUTION OF A MATRIX EQUATION AX=B WHERE A IS n x n AND NON-SINGULAR -- FLOATING POINT

J208 is an automatic version of J196. The user needs only to insert the current version of the Floating-Point System preceding the last card of J208, to record the necessary input data in punched cards, place these cards immediately behind J208, and press the LOAD button. The solution matrix, X*, which approximates X in the equation AX=B will be computed and printed for each equation.

<u>Input Form</u>

Information should be punched in the Floating-Point Data Form as follows:

<u>Header card for a problem</u>

```
            (1)                                    (2)
±  EXP  MANTISSA                       ±  EXP  MANTISSA
|                                      |
|   n       ID                         |   k    ______
|   |       |                          |   |
|   |       └ Any decimal              |   └     number of columns
|   |         information              |         in the matrices
|   |         for problem ID           |         X and B
|   |                                  |
|   └-order of matrix  A               |
|                                      |
└[+ if A is not to be printed          └[+ if (α_ij) is not to be printed
 [- if A is to be printed               [- if (α_ij) is to be printed
```

<u>Data cards for a problem</u>

The elements of A must be punched by rows into consecutive positions of the cards beginning with the second card. Each row must begin with position (1) of a card and if an EF mark (12 punch (+) in column 80) is encountered prior to the reading of the n^th element of a given row, then J208 will assume that the remaining elements of the row are 0. (This does not, of course, affect the six elements of A which were punched

into the card with the EF mark.)

The elements of B must be punched by <u>columns</u> into consecutive positions of the cards beginning with the first card following the cards for A. Each column must begin with position (1) of a card.

Cards for a Sequence of Problems

The solutions for a sequence of equations AX=B can be evaluated by placing the input cards for the equations in sequential order behind J208. If the input cards are followed by two blank cards, the collator will be on Select following the completion of the last problem.

Output Form

The page will be ejected prior to the beginning of each problem. The numbers printed will be in the external floating point form. The print format is that described for the 017 (PNT) order in the Floating-Point writeup.

Printing for a problem

	A	B	C
Information in positions (1) and (2) of header Card	—	± n ID	± k —
Space	—	—	—

If A is printed, then the printing for the ith row is as follows:

Row number	—	i	—
Elements of the row begin here	a_{11} ⋮	a_{12} ⋮	a_{13} ⋮

If A is printed, the page is ejected following the printing of the last row.

	A	B	C
Determinant	——	——	det A
Space	——	——	——

If (α_{ij}) is printed, the same procedure is followed as for A.

X is printed by columns The printing for the ℓ^{th} column is as follows:

Column number	ℓ	——	——
Elements of the column begin here	$x_1^{(\ell)}$ ⋮	$x_2^{(\ell)}$ ⋮	$x_3^{(\ell)}$ ⋮

<u>Notes</u>:

(1) The inequality $n(n+k+1) \leq 2526$ must be satisfied by n and k. In particular, if k=1, then n must not exceed 49 and if k=n, then n must not exceed 35. If $n(n+k+1) > 2526$, the program will halt at 5474_8 with a 130 5474 order.

(2) A 13X halt will occur at 5334_8 if the computational checking criteria is not met in any row during the forward solution. The calculation will continue if the GO button is pressed.

(3) A 13X halt will occur at 5433_8 if the computational checking criteria is not met in any row during the back solution. The calculation will continue if the GO button is pressed.

(4) A 13X halt will occur at 5257_8 if det A=0. If the GO button is pressed, then det A and the current state of the (α_{ij}) and X (or B) matrices will be printed. The next

problem in sequence will then be started if it exists.

(5) A check sum 13X halt at 0014_8 indicates that J208 has probably been loaded incorrectly under the control of J180A. The entire loading process should be restarted.

(6) The user is referred to the J196 write-up for a description of the method.

Program available as J208A.

John Derr

J209E PUNCH STYLE E

J209 will punch the block of words from a to b into style E cards, in octal, one word per card. The location fields of the cards will be left blank. Since the information will be punched in octal, commas will be punched into columns 22 and 31.

Calling sequence:

```
    α     020   α   010   β          Link to J209
    α+1         a         b
    α+2   Control returns here
```

Program length: 52 words

Erasable: A0 - A12

Highest symbol used: *13

Timing: Cards are punched at full speed, 100 cards per minute.

Available as: J209E

Norman Shapiro

J210E EVALUATION OF A ROOT OF A POLYNOMIAL WITH COMPLEX COEFFICIENTS -- FLOATING POINT

J210E is a more flexible version of J157E. J210 will compute, by an iterative process, up to n roots of an n^{th} degree polynomial equation of the form

$$f(z) = \sum_{i=0}^{n} C_i z^i = 0, \text{ where } C_i$$

is a complex number for i=0, ..., n. Along with each root Ω computed, the multiplicity k of Ω, $|g(\Omega)|$ and the new reduced polynomial g(z) are available to the user. The user may, for each root Ω, constrain the convergence of the process by specifying an initial guess z_0 and by modifying certain sensitive quantities.

J210 must be used in conjunction with the current version of the Floating-Point System. J210 is entered in two ways. The main entry (polynomial entry) is performed by basic linkage and results in normalizing the polynomial f(z) into monic form, i.e.

$$g(z) = \sum_{i=0}^{N} a_i z^{N-i}, \text{ where } a_0 = 1 = \frac{C_N}{C_N} \text{ and } C_{N+1} = C_{N+2} = \dots$$

$= C_n = 0$. In all succeeding re-entries (root entries), an attempt is made to compute a new root of the reduced polynomial.

Calling Sequence:

α	020	α	010	β	(polynomial entry to 1st word of J210)
α+1	a00	$L(C_n)$	$\|\Delta\|$	n	
α+2	Error return				(Control returns here if N=0 for the polynomial entry or if the process does not converge for a root reentry.)

(Calling Sequence continued)

Calling Sequence (continued)

$\alpha+3$ First Root return (Control returns here following polynomial entry to J210.)

$\alpha+4$ Intermediate root return (Control returns here following the evaluation of a root of f(z).)

Before the polynomial entry to J210 is executed, the coefficients $\left\{C_i\right\}$ of f(z) must be stored in H.S.S. locations as follows, where $\Delta = \begin{cases} |\Delta| & \text{if } a = 0 \\ -|\Delta| & \text{if } a = 1 \end{cases}$ (Note that $|\Delta| \geq 2$):

$L(C_n)$—C_r^R = real component of C_n in floating point internal form

$L(C_n)+1$—C_n^I = imag. component of C_n in floating pt. internal form

$\vdots$

$L(C_n)+n\cdot\Delta$—C_0^R = real component of C_0 in floating pt. internal form

$L(C_n)+n\cdot\Delta+1$—C_0^I = imag. component of C_0 in fl. pt. internal form

The following locations contain the current values of quantities which are of interest to the user and to J210:

B00 – z_i^R = real component of the current approximation to Ω.

B01 – z_i^I = imag. component of the current approximation to Ω.

B02 – $|g(\Omega)|$, where Ω is the last root evaluated.

B03 – k = current estimate of multiplicity of z_i.

B04 – N = current degree of $g(z)$

B05 – $(n-N)$ = number of leading zero coefficients of f(z).

B10 +6j (j=0,...,N-1) – a_{1+j}^r = real comp. of the coefficient of z^{N-1-j} in g(z).

B11 +6j (j=0, ..., N-1) — a_{1+j}^{I} = imag. comp. of the coefficient of z^{N-1-j} in g(z).

E281 (*88) — I = maximum number of iterations in the first cycle diminished by 1/2N (mod 4) (I=17 unless changed by user)

E282 (*89) — C = maximum number of iteration cycles

(C=7 unless changed by user

E285 (*96) — ϵ ($\epsilon = 10^{-7}$ unless changed by user)

E286 (*97) — η ($\eta = 10^{-4}$ unless changed by user)

E289 (*94) — δ in floating-point ($\delta = 10^{-3}$ unless changed by user)

E290 (*95) — δ in fixed point scaled 2^0 ($\delta = 10^{-3}$ unless changed by user)

All root entries (reentries) to J210 are made by transferring to E046. If the root return is made to the right of E046, then J210 will use the contents of B0 and B1 for the initial guess, z_0, to Ω. Otherwise, J210 will set $z_0 = \sqrt[N]{|a_N|} + i \sqrt[N]{|a_N|}$. Prior to any root re-entry to J210, the user is free to alter z_0, I, C, ϵ, η, and δ as he sees fit. Upon return of control to $\alpha+2$, $\alpha+3$, or $\alpha+4$ the user will probably want to utilize some of the quantities z_1, $|g(\Omega)|$, k, N, n-N, and the $\{a_i\}$. If control returns to the Intermediate root return, the user may assume that $\Omega = z_1$, k = the multiplicity of Ω and N is the degree and the $\{a_i\}$ are the coefficients of the new reduced polynomial g(z). It is important to observe that the user must program the decision to terminate the computation of roots of f(z). This can usually be done by testing N=0 for every intermediate root return if all of the roots of f(z) are desired.

Description of the Method

Set $q_0(z) = g(z)$ and let $q_j(z)$ be defined by $q_{j-1}(z) = (z-z_i)q_j(z) + q_{j-1}(z_i)$ for $j=1, 2, \ldots, N$. Let the non-negative integer ℓ be defined by $\ell=0$ if $|g'(z_i)| \geq \eta\,|q_1(0)|$ and by $|g^m(z_i)| < \eta\,|q_m(0)|$ for $m=0, 1, \ldots, \ell$ and $|g^{\ell+1}(z_i)| \geq \eta\,|q_{\ell+1}(0)|$*. Then the basic iterative process is defined by

$$(\Delta z)_i = \frac{(k-\ell)f^{\ell}(z_i)}{f^{\ell+1}(z_i)}$$, where the determination of

$k-\ell$ is yet to be explained. If $\ell=0$, then $k-\ell=1$. Hence, when $\ell=0$, the basic process reduces to the Newton-Raphson process. If $\ell>0$, then make use of the fact that

$$\frac{g^{\ell}(z_i)}{g^{\ell+1}(z_i)} \rightarrow \frac{z_i - \Omega}{k-\ell} \quad \text{as } z_i \rightarrow \Omega \text{ for } 0 \leq \ell < k,$$

provided Ω is a root of $g(z)$ of multiplicity k.

Compute $x = \dfrac{g^{\ell}(z_i)/g^{\ell+1}(z_i)}{g^{\ell}(z_i)/g^{\ell+1}(z_i) - g^{\ell-1}(z_i)/g^{\ell}(z_i)}$. Suppose

$x \doteq$ an integer $I \geq 2$. More precisely, if $x^R \geq 2-\delta$, $|x^I| + x^R < \delta$, $f > 1-\delta$ or $f \leq \delta$ (where f is the fractional part of x^R), and $x^R \leq N - \ell+1$, then set $k-\ell=x-1$ and $k=x+\ell-1$. Otherwise, set $k-\ell=1$ and $k=\ell+1$.**

The basic iterative process is modified***at every fourth

* In J157E, $\eta = 10^{-2}$.

** The δ used here is 10^{-1} in J157.

*** In J157, this modification does not exist.

iteration of a cycle, provided $\ell = 0$. Compute

$$x = \frac{(\Delta x)_{i-1}}{(\Delta x)_{i-1} - (\Delta x)_i} .$$

Determine as above whether or not $x \doteq$ an integer I such that $2 \leq I \leq N$. If so, then set $\ell = 1$ and carry out the basic process for $\ell > 0$. If not, then replace $(\Delta x)_i$ by $(\Delta x)_i^3/(\Delta x)_{i-1}^2$.

In the former case ($x \doteq I$ such that $2 \leq I \leq N$), multiple roots can be detected prematurely although sometimes a set of distinct roots will look like one big multiple root to the process with multiplicity equal to the sum of the respective multiplicities.

In the latter case, the convergence for the pair of steps $i-1$ and i is accelerated. In fact, the convergence becomes of the third order. Hence, half of the time (more than half in the case of multiple roots) the combined process is a third order one. Simultaneous with the jump to third order convergence, however, the domain of convergence is decreased.

Convergence and the Choices of z_0

The criteria for convergence of z_i to Ω requires that $|(\Delta z)_i| \leq |z_i| \cdot \epsilon$, i.e. that the relative error in Ω shall not exceed ϵ. $|z(\Omega)|$ provides an independent check on the computation. Note though that J210 does not substitute Ω back into $f(z)$.

The system which J210 uses for starting the iterative process breaks down into C iteration cycles. The first iteration cycle allows a maximum of I+N/2 (mod 4) iterations and this maximum is increased by 4 for each succeeding iteration cycle. On the first cycle z_0 is chosen as described above. On successive cycles z_0 is set equal to a complex number with modulus $\doteq \sqrt[N]{a_N}$ according to a fixed pattern, where a_N is the constant coefficient of g(z). The set of values of z_0 includes points in all four quadrants and contains at most seven distinct points.

Any cycle can be interrupted by convergence. If all C cycles are executed in their entirety, then the process has not converged and control will return to the Error return.

NOTES

(1) If f(z) has real coefficients, and if Ω is a proper complex root, then set $z_0 = \Omega^*$ (the conjugate of Ω), provided Ω^* has not already been evaluated.

(2) In general, do not set z_0 = a real number unless only real roots are desired.

(3) In general, ϵ should not be less than 10^{-8}, since 9 digit arithmetic is used.

(4) In general, as η is increased, the process will tend to produce more multiple roots. For example, as η is increased, distinct roots (as well as true multiple roots) are more likely to be identified as multiple roots. Conversely, as η is decreased, multiple roots (as well as distinct roots) will tend more to be identified as sets of distinct roots.

(5) Essentially, the statement of (4) for η holds for δ as well, although not for the same reason.

(6) The values of the roots, as well as their multiplicities, can vary significantly with the order in which they are determined.

(7) The nature of an error return can be determined by an examination of N.

(8) J169 and J170 are slave routines w.r.t. J210 and therefore must be used along with the current version of the Floating-Point System in conjunction with J210. Further, J169 and J170 have the origins Y0 and Z0 respectively, w.r.t. J210. Hence, these regions must be assigned locations before J210 is loaded.

(9) If the process does not converge, B0 and B1 contain the last approximation to a root Ω. Also N and the $\{a_{1+j}\}$ give the coefficients of g(z).

(10) The reader is referred to the J157 write-up for information of a more general nature.

J210 Storage (without slaves)

Region A (erasable) 27 words (A00-A26)

Region B (semi-erasable) 10+6N words (B0 - B9 + 6N)

Symbolic Code: 320 words

Highest Symbol Used: *99

Storage Used by Slave Routines (J169E and J170E)

Region A (erasable) 7 words (A0 - A6)

Symbolic Code: 27 words

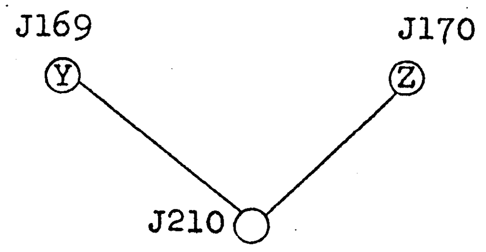
Combined Storage Used by J210, J169 and J170

Region A (erasable) 27 words (A00 - A26)

Region B (semi-erasable) 10 + 6N words (B0 - B9 + 6N)

Symbolic Code: 347 words

Lattice Diagram Describing Master-Slave Routines



John Derr

J211A AUTOMATIC COMPUTATION OF THE ZEROS OF A POLYNOMIAL WITH COMPLEX COEFFICIENTS -- FLOATING POINT

J211A makes use of J210 to compute the roots of polynomials. The user needs only to place a copy of the current version of the Floating Point System before the last card of J211, record the necessary input data in punched cards, place these cards immediately behind J211, and press the LOAD button. The roots of the polynomials will be computed and printed one at a time.

Input Form

Information should be punched in the Floating Point Data Form as follows:

Header card for a problem

Pos (1)	S	+ (blank) if all coefficients are real - if any coefficient is not real
	EXP	m = the maximum number of distinct roots to be evaluated (usually m=n). If m=0, then m will be set = n.
	MANTISSA	ID (If ID=0, then the coefficients of f(z) will be printed.)
Pos (2)	S	Iff -, set $\Omega^I=0$ when $\lvert\Omega^I\rvert < \epsilon \lvert\Omega^R\rvert$.
	EXP	Iff 0, the final reduced polynomial g(z) will be printed.
	MANTISSA	n = degree of f(z).
Pos (3)	z_0^R	
Pos (4)	z_0^I	Iff $z_0 \neq 0$, the starting value for the first iteration cycle and for the first root will be z_0.

Pos (5) Iff $\epsilon \neq 0$, then ϵ will replace the ϵ (10^{-7}) in J210.

Pos (6) Iff $\eta \neq 0$, then η will replace the η (10^{-4}) in J210.

Coefficient cards for a problem

The coefficients are punched 6/card into consecutive positions of the cards beginning with position (1) of the next card after the header card. If the sign in position (1) of the first card is + (blank), then only the real components of the coefficients are to be punched and these must be in consecutive positions (no gaps). Otherwise, the real and imaginary components of each coefficient must be punched. The last card for each problem must contain an EF mark (12 punch (+) in col. 80).

A sequence of problems

The roots for a sequence of polynomials can be evaluated by placing the input cards for the polynomials in sequential order behind J211A. If the input cards are followed by two blank cards, the console will show READER upon completion of the last problem.

Output Form

The page will be ejected prior to the beginning of each problem. The numbers printed will be in the external Floating Point form. The print format is that described for the 017 (PNT) order in the Floating-Point write-up.

Printing for one problem

	A	B	C
Information in positions 1,2,3 of header card	$\pm m$ ID	$\pm 0/1$ n	z_0^R
Information in positions 4,5,6 of header card	z_0^I	ϵ	η

If ID = 0, the coefficients of f(z) will be printed in positions A and B as follows:

	A	B	C
Space	——	——	——
Coefficients begin here	c_n^R	c_n^I/c_{n-1}^R	——
	⋮	⋮	
Coefficients end here	c_0^R/c_1^R	c_0^I/c_0^R/blank	——
Space	——	——	——
Roots begin here	——	——	k_1
⋮	Ω_1^R	Ω_1^I	$\lvert g(\Omega_1)\rvert$
	⋮	⋮	⋮
	——	——	k_p
(pth) last root computed	Ω_p^R	Ω_p^I	$\lvert g(\Omega_p)\rvert$
Space	——	——	——
Space	——	——	——

If 0/1 = 0, then the coefficients $\{a_i\}$ of g(z) will be printed as follows:

Coefficient of z^N	a_0^R	a_0^I	N
	⋮	⋮	⋮
Constant coefficient	a_N^R	a_N^I	0

NOTES

(1) If n exceeds 450, then following the printing of the contents of the header card and the printing (if any) of the coefficients of f(z), J211 will begin the next problem (if any).

(2) $p \leq \min (m, n)$ and $N \leq n - \sum_{j=1}^{p} k_j$, where here N denotes the degree of the last reduced polynomial g(z).

(3) If f(z) has real coefficients, $\Omega^I \neq 0$, and $|\Omega^I| \geq |\Omega^k| \cdot \epsilon$, then J211 will set $z_0 = \Omega^R - \Omega^I$, provided $\Omega^R - \Omega^I$ has not been evaluated previously.

(4) If the process does not converge, then the printer will be spaced twice, z_1^R, z_1^I, and $|g(z_1)|$ will be printed on one line, and the printer will be spaced twice again. Then if 0/1 = 0, the coefficients of g(z) will be printed as described above. In any event, J211 will then begin the next problem (if any).

(5) The following approximate times were recorded while operating in the SD mode:

$z^3 - 1$ required 10 sec. $z^5 - 1$ required 34 sec.
$z^9 - 1$ required 1 min. 10 sec. $z^{17} - 1$ required 5 min. 10 sec.
$z^{33} - 1$ required 30 min.

(6) The user is referred to the J210 write-up for a description of the method.

(7) The computation for any one problem can be interrupted by manually transferring to 5560_8. The result will be the same as in (4). Note that the coefficients of g(z) will be printed only if 0/1 = 0.

(8) J210 has been modified by placing breakpoint tracing marks in certain key words. Thus, if T2 is on, the following information will be printed:

5160 5162	halves computed $(\Delta z)i$ and subtracts from z_1, if $\left\|f^{\ell+1}\right\| < n \left\|q_{\ell+1}(0)\right\|$ and $\left\|f^{\ell}\right\| \geq n^2\left\|q_{\ell}(0)\right\|$
5211 5213	x^R is compared with 2-6 and $\left\|x^I\right\|$ is compared with x^R
5251 5253	the computed $(\Delta z)_1$ is multiplied by $k-\ell$
5254 5256	the computed $(\Delta z)_1$ is added to z_1
5435 5436 5437 5440	the computed $(\Delta z)_1$ is replaced by $(\Delta z)_1^3 / (\Delta z)_1^2$ and added to z_1.

Program available as J211A.

John Derr

J212A AUTOMATIC LEAST SQUARES POLYNOMIAL APPROXIMATION -- FLOATING POINT

J212 is an automatic version of J178. The user needs only to insert the current version of the Floating-Point System preceding the last card of J212, to record the necessary input data in punched cards, place these cards immediately behind J212, and press the LOAD button. The coefficients $\{a_i\}$ of the approximating polynomial

$$f(x) = \sum_{i=0}^{M} a_i x^i$$

will be computed and printed for each header card.

Input Form

Information should be punched in the Floating Point Data Form as follows:

Header card for a problem

pos (1)	S	+ if data triplets **are** to be printed
		- if data triplets are **not** to be printed
	EXP	M = degree of polynomial f(x)
	MANTISSA	ID (any decimal information)
pos (2)	S	+ if data cards follow
		- if no data cards follow
	EXP	Blank
	MANTISSA	N = number of points
pos (3)	S	Blank (or +)
	EXP	00 if $(x_i, y_i, r_i$[1]$)$ is to be printed
		01 if (x_i, y_i, y_i^*) is to be printed
	MANTISSA	δ if $\delta_j = 0$ for $A \leq j \leq M$[2].
		0 otherwise

Positions (4), (5), and (6) of the header card are not interpreted by J212A.

1 $r_i = y_i^* - y_i$. Hence, $y_i^* = r_i + y_i$.

2 See J178 write-up.

(x_i, y_i) Data Cards for a problem

The number pairs (x_i, y_i) must be punched in consecutive three-position blocks beginning with the first card following the header card.

		(1)	(2)	(3)	(4)	(5)	(6)
1st Card		x_1	y_1	---	x_2	y_2	---
⋮		⋮	⋮	⋮	⋮	⋮	⋮
last card	(N even)	x_{N-1}	y_{N-1}	---	x_N	y_N	---+
	(N odd)	x_N	y_N	---	---	---	---+

Further, there must be an end of file mark (12 punch) in column 80 of the last data card.

Cards for a sequence of problems

There is a one-to-one correspondence between header cards and problems. A sequence of problems can be solved by placing the associated header cards together with the data cards following (if any) in sequential order behind J212. If the entire deck of input cards is followed by two blank cards, the console will show "READER" following the completion of the last problem.

Output Form

The page will be ejected prior to the beginning of each problem. The numbers printed will be in the external floating point form. The print format is that described for the 017 (PNT) order in the Floating-Point write-up.

Printing for a problem

	A	B	C
	S EXP MAN	S EXP MAN	S EXP MAN
Information in header card	$\pm$ M ID	$\pm$ 00 N	$\pm$ 0 δ/1 δ/0
Space	—	—	—
Coefficients	a_0	a_1	a_2
	$\vdots$	$\vdots$	$\vdots$
Space	—	—	—

If the number triplets are to be printed, they are printed one triplet per line in the natural order.

x_1	y_1	$r_1/y_1{}^*$
x_2	y_2	$r_2/y_2{}^*$
$\vdots$	$\vdots$	$\vdots$
x_N	y_N	$r_N/y_N{}^*$.

Notes

(1) Prior to entry into J178, the set of $\{x_i\}$ is replaced by the set of $\{x_i - m\}$ where

$$m = \left(\sum_{i=1}^{N} x_i \right) \div N.$$

Upon exit from J178, J194 is used to compute the coefficients $\{a_i\}$.

(2) $1 \leq M \leq 5$ and $M+1 \leq N \leq 120$[1] must hold.

(3) If $\delta \neq 0$, then the decimal equivalent of the octal number $\delta_0\ \delta_1\ \delta_2\ \delta_3\ \delta_4\ \delta_5$ must be presented to J212. For example, suppose M=5 and it is desired to impose the condition that

1 The upper bound of 120 on N was artificially imposed.

the odd powers of x^1 shall vanish. Hence, we want $a_1 = a_3 = a_5 = 0$. Then $\delta = (101\ 010)_2 = (52)_8 = (42)_{10}$. $\delta = 42$ is presented to J212.

(4) A 13X halt will occur at $(2434)_8$ if the computational checking criteria is not met in any row during the forward solution in J196. The calculation will continue if the GO button is pressed. This halt can occur due to the loss of significant digits in the matrix solution.

(5) A 13X halt will occur at $(2533)_8$ if the computational checking criteria is not met in any row during the back solution. The calculation will continue if the GO button is pressed. This halt can occur due to the loss of significant digits in the matrix solution.

(6) A 13X halt will occur at $(2357)_8$ if det $T = 0$. If the GO button is pressed, then J212 will exit J196 and J178. If printing occurs upon completion of this problem, the results will be incorrect. J212 will continue ahead to the next problem if there is one.

(7) A check sum 13X halt at $(0014)_8$ indicates that J212 has probably been loaded incorrectly under the control of J180A. The entire loading process should be restarted.

Accuracy

Using $x_1 = 0(.05)1$ and the exact corresponding values of y_1 obtained from a table of Legendre polynomials of degrees $M = 2, 3, 4, 5$ the number of correct significant digits obtained in the computed coefficients was as follows:

M=2	9 digits	M=3	9 digits
M=4	8(-) digits	M=5	7 digits

Timing

See the discussion of timing in the J178 write-up. The times there do not include input-output times, however.

Program available as J212A.

John Derr

J213A CHANGE ADDRESS FIELDS

J213A will reproduce style E cards changing the location and address fields which have specified contents to specified quantities.

For the purpose of this write-up an address may be regarded as a sequence of four alpha-numeric characters. In using J213A a sequence of pairs of addresses α_1, β_1, ..., α_i, β_i, ..., α_n, β_n is specified on control cards. A deck of style E cards will then be processed. All information except the information in the location and address fields will be duplicated. Any location or address field not equal to any of the α_i will be duplicated. A location or address field equal to one of the α_i will be replaced by the corresponding β_i.

The pairs α_i, β_i are placed in the left and right address fields of successive style E control cards.

In determining if two addresses are equal it is required that they be equal, character by character. In particular, addresses which differ only in that zeros are replaced by blanks will not be regarded as equal. This can be changed by turning T1 on. If this is done, addresses which can be made equal by changing zeros to blanks will be regarded as equal and no zeros will be punched in location or address fields.

The cards punched are ordinarily echo checked. T2 on will suppress echo checking.

Preparation of Deck:

J213A
Control Cards
Blank Card
Deck to be Processed
2 Blank Cards

13X stops: While reading in Program; check sum stop.

While processing cards; Echo check stop.

Program available as: J213A

Norman Shapiro

J214E NATURAL LOGARITHM

J214 computes the natural logarithm of a number y, where $0 < y < 2^{39}$.

J214 is entered with $2^{-q}y$ in MQ, where $0 < 2^{-q}y < 1$ and $-39 \leq q \leq 39$. On exit from J214 $2^{-5}\ln y$ is in the accumulator.

Calling sequence:

α	020	α	010	β
$\alpha+1$	$q \cdot 2^{-39}$			
$\alpha+2$	Control returns here			

Program stop at word 27, right, if $y \leq 0$. Hit GO to exit from J214, accumulator will contain $|2^{-q}y|$.

Region A (erasable): 5 words, A0 to A4.

Symbolic code: 36 words

Highest symbol used: * 10

Timing: J214 has 7 multiplications and a division as compared to 40 multiplications for J190.

Accuracy: Logarithm accurate to 9 decimal places.

Program available as J214E.

The power series used was taken from the ILLIAC manual, Code 43.

Marvin Shapiro

J215E 10 PT. GAUSSIAN INTEGRATION -- FLOATING POINT

J215 computes $\int_a^b f(x)dx$ using a 10 point Gaussian formula. This is equivalent to approximating f(x) by a polynomial of degree 19.

Auxiliary routine:

J215 utilizes an auxiliary subroutine written by the programmer to compute f(x). This subroutine is entered from J215 with x in the floating-point AMQ. The calling sequence to the subroutine is:

```
γ      020   γ   010   F 0
γ+1    Control returns to left of γ+1 in floating point
```

At step γ+1 f(x) must be in the floating point AMQ. The exit from the auxiliary subroutine must be made with the floating point system in control.

Calling sequence:

```
α      020   α   010   β
α+1          a           (in floating point form)
α+2          b           (in floating point form)
α+3    Control returns to the left of α+3 in floating point.
```

a and b must be in the floating point data form. The final exit from J215 to step α+3 is made with the floating point system in control.

On exit from J215 $\int_a^b f(x)dx$ will be in the floating point AMQ and in cell A 0.

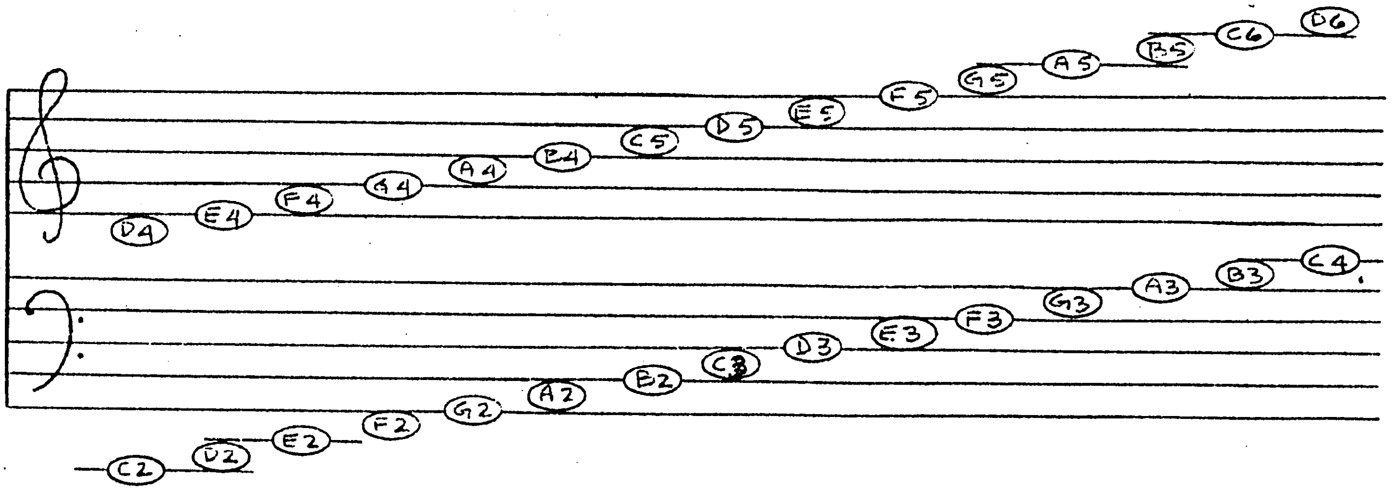
Region A (semi-erasable): 4 words, A 0 to A 3. Region A is erasable except for the f(x) routine.

Region F: f(x) routine
Symbolic code: 46 words
Highest symbol used: * 16
Program available as J215E.

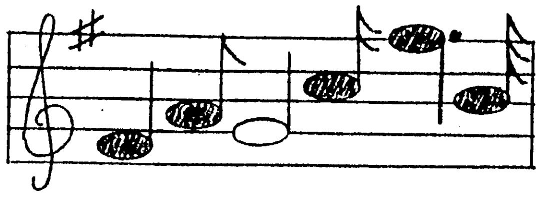
Marvin Shapiro

J216A. JOHNNIAC MUSIC

Anyone who can read music can write or transcribe music for the JOHNNIAC. The following staffs show the allowable notes and their JOHNNIAC Music Assembler designation:



Sharps are denoted by an asterisk (*), flats by a comma (,). Any valued note may be designated for the assembler. For example:



would be written as:

Col. 9	10	11	13	14	16	17
F	*	4	0	1	0	4
A		4	0	1	0	8
G		4	0	1	0	2
C		5	0	1	1	6
F	*	5	0	3	0	8
B	,	4	0	1	3	2

The time signature need only be entered upon the first card and the beats per minute also need only be entered on the first card unless it changes during the progress of the song, whence it should be entered again.

The decimal card form for input is as follows:

Cols.	Description
1 and 3	Time signature
5-7	Beats per minute
9	Note (Alphabetic designation)
10	Sharp (Asterisk (*) if note is sharped; comma (,) if note is flatted)
11	Octave (Numeric designation shown in above staff)
13-14	Numerator of value
16-17	Denominator of value
18-80	Comments

A rest is denoted by an R in the note field, sharp and octave must be blank.

An End of Song (or File) is a blank card and reading ceases at that point.

The Deck is made up as follows: The JOHNNIAC Music Assembler, the decimal song deck, and 3 blank cards. The whole deck is read from the primary feed of the reader. When the end of file card is reached, the machine will stop unless T3 is on. If T3 is on, a complete self-loading binary deck will be punched, and then the machine will stop.

In order to play the tune, put the Hoot Selector on 105, press the reset button next to the Hoot Selector and hit GO. To stop the song at the end, put H1 on, to repeat, hit GO. The above instructions also hold when playing the song from the binary deck.

The switches T1 and T2 may be used to alter the speed of playing the song, as follows:

T1 off	T2 off	- as written
T1 on	T2 off	- play twice as fast
T1 off	T2 on	- play 4/3 as fast
T1 on	T2 on	- play 4/5 as fast.

Decimal input forms are available, and any questions will be answered by M. Bernstein.

JOHNNIAC MUSIC ASSEMBLY

TIME SIG.				BEATS PER MIN.		NOTE	SHARP	OCTAVE		VALUE			COMMENTS
1	/	3		5 6 7		9	10	11		13 14	/	16 17	18 — 80

J217A STYLE D LOWER LOADER

J217 will load itself and the following style D binary cards into H.S.S. checking the check sum of most cards. The loader will not check the check sum of any card which has a 9 punch in col. 1.

Loading will cease when the loader encounters a card which has no 9 punch in col. 41. The loader will then execute the operation in the 9's row cols. 1-7 in association with the address in the 9's row cols. 29-40 of this card.

If the check sum fails to check, there will be a program stop with NIA = 0037_8. The accumulator will contain $-\left|\frac{1}{2}\right.$ (card check sum - computed check sum)$\left.\right|$. To continue, hit GO; no cards are lost in this process.

To re-enter the loader, transfer to the left of 0001_8.

Program Location: 0000 - 0043_8

Program Stop: NIA = 0037_8 - check sum stop.

Program available as J217A.

See style D definition sheet for definition of check sum.

M. I. Bernstein

12-17-56

J218A STYLE D UPPER LOADER

J218 will load itself and the following style D binary cards into H.S.S. checking the check sum on most cards. The loader will not check the check sum on those cards which have a 9 punch in col. 1.

Loading will cease when the loader encounters a card which has no 9 punch in col. 41, the loader will then execute the operation in the 9's row col. 1-7 in association with the address in the 9's row cols. 29-40 of this card.

If the check sum fails to check, there will be a program stop with NIA = 7777_8. The accumulator will contain $-\left|\frac{1}{2}\text{ (card check sum - computed check sum)}\right|$. To continue loading, hit GO. No cards are lost in this process.

To re-enter the loader, transfer to the left of 7741_8.

Program Location: 7734_8 - 7777_8

Program Stop NIA = 7777_8 Check Sum Stop

Program available as J218A.

M. I. Bernstein

12-19-56

J219A STYLE D CHECK SUM CORRECTOR

J219 is a self-loading program which will correct check sums in Style D binary cards. If a 9 punch is present in col. 1 to prevent the check sum from being checked while loading with J217A or J218A, it will be eliminated before the new check sum is formed.

The cards which are to be corrected should be loaded immediately behind J219 plus three blanks at the end. Card for card each one is read and a new one is punched until a card with <u>no 9 punch in col. 41</u> is encountered, at which time a program stop occurs with NIA = 0144_8.

Program Stops: NIA = 0144_8 - end of program

Program available as J219A.

M. I. Bernstein

J220E SOLUTION OF A SYSTEM OF ORDINARY DIFFERENTIAL EQUATIONS - FLOATING POINT [1]

J220 solves a set of differential equations with the following call sequence:

```
α     020     α     010     β
α+1                         d
α+2   Return (outside floating point control)
```

where d is the location of the first word of the auxiliary subroutine.

The following library region bases must be designated before input of this subroutine in the following manner:

Region	Origin	Use
D	a	The numbers in a+i are the variables y_i (i=0, ..., n-1). Originally the initial values are placed here.

The origins of regions E, F, and G are used as preassigned parameters by J220.

E	b-a	The numbers in b+i are the scaled derivatives, hy'_i, calculated by the auxiliary subroutine. $b > a+n-1$.
F	c-b	Locations c+i are used as temporary storage for this subroutine. These locations must be cleared to zero before this subroutine is entered for the first time. $c > b+n-1$.
G	c+n	n is the number of differential equations to be solved.

Region A (erasable): 4 words

Symbolic Code: 45 words

Highest symbol used: * 48

Program available as J220E

1 This routine is a floating point version of J126. For a description of the method see the writeup for J126.

J221E PRINT DECIMAL NUMBERS - VARIABLE FORMAT AND SCALING

Calling seq:

Loc	OP	ADDR	OP	ADDR
α	020	α	010	β
$\alpha+1$	000	0000	000	s_1
$\alpha+2$	000	L_1	000	q_1
$\alpha+3$	000	p_1	000	k_1
⋮	⋮	⋮	⋮	⋮
$\alpha+3w-2$	000	0000	000	s_w
$\alpha+3w-1$	000	L_w	000	q_w
$\alpha+3w$	000	p_w	000	k_w
$\alpha+3w+1$	100	ΔL	000	n
$\alpha+3w+2$	Control returns to left of $\alpha+3w+2$			

L_i:	Location of i^{th} print word on first line	$0000 \leq L_i \leq 7777_8$
ΔL:	Increments of locations of print words(1)	$0000 \leq \Delta L \leq 7777_8$
s_i:	Space to left of i^{th} word on each line(1)	$0 \leq s_i \leq 120_{10}$
q_i:	No. of bits (excluding 2^0 bit) to left of binary point in i^{th} word on each line	$0 \leq q_i \leq 39_{10}$
p_i:	Number of decimal digit positions to left of decimal point in i^{th} word on each line	$0 \leq p_i \leq 11_{10}$
k_i:	Number of decimal digits to be printed in i^{th} word on each line(2)	$1 \leq k_i \leq 11_{10}$
n :	Number of lines	

<u>Restriction</u> $\sum_{i=1}^{w} (k_i+s_i+1)+(\text{the number of } i \text{ with } p_i \leq k_i) \leq 120$

J221 will print n lines, w words per line. The scaling and format specification must be the same for the i^{th} word on each line ($1 \leq i \leq w$) and is specified in the calling sequence, by $\alpha+3i-2$, $\alpha+3i-1$ and $\alpha+3i$. The locations of the i^{th} word on successive lines must differ by ΔL, i.e., the locations of the i^{th} word on successive

(1) A more precise definition appears below.

(2) If $p \geq k$, the k most significant digits will be printed.

lines must be L_1, $L_1+\Delta L$, $\cdots$, $L_1+(n-1)\Delta L$.

Leading zeros but not following zeros will be suppressed. Decimal points will be printed except when $p \geq k$ in which case no space will be allocated for the decimal point. Signs will be printed immediately to the left of the first digit position. Plus signs will appear as blanks.

s_i is the number of blanks to appear to the left of the sign position of the i^{th} word on each line.

An incorrectly scaled number will appear as a sequence of decimal points. The numbers to be printed will be approximately rounded. Number which can be printed exactly within the specified format will be so printed (regardless of rounding). Numbers which are correctly scaled before rounding but incorrectly scaled after rounding will not be rounded.

J221 <u>post spaces</u>

Region A (erasable)-40 words, A0-A39

Symbolic code - 168 words

Available as J221E

M.and N. Shapiro

J222E - PRINT DECIMAL NUMBERS - UNIFORM FORMAT AND SCALING

Calling sequence:

Loc.	OP	ADDR	OP	ADDR
α	020	α	010	β
$\alpha+1$		ΔL		s
$\alpha+2$		L		q
$\alpha+3$		p		k
$\alpha+4$		w		n
$\alpha+5$	Control returns to left of $\alpha+5$.			

L :	Location of first print word	$0000 \leq L \leq 7777_8$
ΔL:	Increment of locations of print words[1]	$0000 \leq \Delta L \leq 7777_8$
s :	Space between printed words[1]	$0 \leq s \leq 120_{10}$
q :	No. of bits (excluding 2^0 bit) to left of binary point	$0 \leq q \leq 39_{10}$
p :	Number of decimal digit positions to left of decimal point	$0 \leq p \leq 11_{10}$
k:	Number of decimal digits per word to be printed[2]	$0 \leq k \leq 11_{10}$
w :	Number of words per line	
n :	Number of lines.	

<u>Restriction</u> $w(k+s+1) \leq 120$ if $p \geq k$, $w(k+s+2) \leq 120$ if $p < k$.

J222 will print n lines, w words per line. Each word must have the same scale factor and will be printed with the same format. The words to be printed must have locations

(mod. 2^{12}): L, $L+\Delta L, \cdots, L+(nw-1)\Delta L$.

Leading zeros but not following zeros will be suppressed. Decimal points will be printed except when $p \geq k$, in which case no space

[1] A more precise definition appears below.

[2] If $p \geq k$, the k most significant digits will be printed.

will be allocated for the decimal point. Signs will be printed immediately to the left of the first digit position. Plus signs will appear as blanks.

s is the number of blank columns to appear to the left of the sign position of each word.

Incorrectly scaled numbers will be printed as a series of decimal points. The numbers to be printed will be approximately rounded. Numbers which can be printed exactly within the specified format will be so printed (regardless of the rounding). Numbers which are correctly scaled before rounding but incorrectly scaled after rounding will not be rounded.

J222 Post spaces

Region A (erasable)- 36 words, A0-A35

Symbolic code - 166 words

Available as J222E

M. and N. Shapiro

J223E - PRINT DECIMAL NUMBERS - UNIFORM FORMAT AND SCALING

Calling sequence:

Loc.	OP	ADDR	OP	ADDR
α	020	α	010	β
$\alpha+1$		ΔL		s
$\alpha+2$		L		q
$\alpha+3$		p		k
$\alpha+4$		w		n
$\alpha+5$	Control returns to left of $\alpha+5$.			

L :	Location of first print word	$0000 \leq L \leq 7777_8$
ΔL:	Increment of locations of print words(1)	$0000 \leq \Delta L \leq 7777_8$
s :	Space between printed words(1)	$0 \leq s \leq 120_{10}$
q :	No. of bits (excluding 2^0 bit) to left of binary point	$0 \leq q \leq 39_{10}$
p :	Number of decimal digit positions to left of decimal point	$0 \leq p \leq 11_{10}$
k:	Number of decimal digits per word to be printed(2)	$0 \leq k \leq 11_{10}$
w :	Number of words per line	
n :	Number of lines.	

<u>Restriction</u> $w(k+s+1) \leq 120$ if $p \geq k$, $w(k+s+2) \leq 120$ if $p < k$.

J223 will print n lines, w words per line. Each word must have the same scale factor and will be printed with the same format. The words to be printed must have locations

(mod. 2^{12}): $L, L+\Delta L, \cdots, L+(nw-1)\Delta L$.

Leading zeros but not following zeros will be suppressed. Decimal points will be printed except when $p \geq k$, in which case no space

(1) A more precise definition appears below.

(2) If $p \geq k$, the k most significant digits will be printed.

will be allocated for the decimal point. Signs will be printed immediately to the left of the first digit position. Plus signs will appear as blanks.

s is the number of blank columns to appear to the left of the sign position of each word.

Incorrectly scaled numbers will be printed as a series of decimal points. The numbers to be printed will be approximately rounded. Numbers which can be printed exactly within the specified format will be so printed (regardless of the rounding). Numbers which are correctly scaled before rounding but incorrectly scaled after rounding will not be rounded.

J223 Post spaces

Region A (erasable)- 36 words, A0-A35

Symbolic code - 167 words

Available as J223E

M. and N. Shapiro

7-31-59

J225 - MOUSE MATRIX TO PRINTER ALPHA NUMERIC

J225 prints 80 columns corresponding to the 80 columns encoded in MOUSE MATRIX form. The left 40 columns are printed from B0 - B6, and the right 40 columns are printed from C0 - C6. Each of the 80 columns may contain any alpha-numeric character.

Before entering J225 a 100 0004 order (to print in C_{1-80}) or a 100 005 order (to print in C_{41-120}) must be given. The programmer must give appropriate 106 orders at the appropriate times as J225 does not space the printer.

Calling Sequence:

```
     α   020   α   010 β
   α+1   Control returns at left of α+1
```

J225 does not space the printer

Space required:

```
Symbolic code - 139 words
Region A - 28 words (A3 - A30)
Region B -  7 words (B0 - B6)
Region C -  7 words (C0 - C6)
```

M Shapiro

8-4-59

J226E* PRINT DECIMAL NUMBERS - VARIABLE FORMAT AND SCALING

Calling Sequence:

Loc	OP	ADDR	OP	ADDR
α	020	α	010	β
$\alpha+1$	000	0000	000	s_1
$\alpha+2$	000	L_1	000	q_1
$\alpha+3$	000	p_1	000	k_1
.	.	.	.	.
.	.	.	.	.
.	.	.	.	.
$\alpha+3w-2$	000	0000	000	s_w
$\alpha+3w-1$	000	L_w	000	q_w
$\alpha+3w$	000	p_w	000	k_w
$\alpha+3w+1$	100	ΔL	000	n
$\alpha+3w+2$	Control returns to left of $\alpha+3w+2$			

L_i	:	Location of ith print word on first line	$0000 \leq L_i \leq 7777_8$
ΔL	:	Increments of locations of print words [1]	$0000 \leq \Delta L \leq 7777_8$
s_i	:	Space to left of ith word on each line [1]	$0 \leq s_i \leq 120_{10}$
q_i	:	No. of bits (excluding 2^0 bit) to left of binary point in ith word on each line	$0 \leq q_i \leq 39_{10}$
p_i	:	Number of decimal digit positions to left of decimal point in ith word on each line	$0 \leq p_i \leq 11_{10}$
k_i	:	Number of decimal digits to be printed in ith word on each line [2]	$1 \leq k_i \leq 11_{10}$
n	:	Number of lines	

<u>Restriction:</u>

$$\sum_{i=1}^{w} (k_i+s_i+1) + (\text{th number of } i \text{ with } p_i \leq k_i) \leq 120$$

J226 will print n lines, w words per line. The scaling and format specification must be the same for the ith word on each line ($1 \leq i \leq w$) and is specified in the calling sequence, by $\alpha+3i-2$, $\alpha+3i-1$ and $\alpha+3i$. The locations of the ith word on successive lines must differ by ΔL, i.e., the locations of the ith word on successive

---

(1) A more precise definition appears below.
(2) If $p > k$, the k most significant digits will be printed.

lines must be L_1, $L_1 + \Delta L$, ..., $L_1 + (n-1)\ \Delta L$.

Leading zeros but not following zeros will be suppressed. Decimal points will be printed except when $p \geq k$ in which case no space will be allocated for the decimal point. Signs will be printed immediately to the left of the first digit position. Plus signs will appear as blanks.

s_i is the number of blanks to appear to the left of the sign position of the i^{th} word on each line.

An incorrectly scaled number will appear as a sequence of decimal points. The numbers to be printed will be approximately rounded. Number which can be printed exactly within the specified format will be so printed (regardless of rounding). Numbers which are correctly scaled before rounding but incorrectly scaled after rounding will not be rounded.

J226 post spaces

Region A (erasable) - 40 words, A0-A39

Symbolic Code - 169 words

Available as J266E

M. Shapiro

* This routine replaces J221E, which had an error in it.

J227 Message Printer

J227 is designed to print headings, messages to the operator, etc. A message consists of up to 120-alpha-numeric characters and is encoded into a block of 21 words. A message is read in by calling on J227 at a time when the message is in C_{41-80} of up to three cards at the read position of the primary feed of the collator. Several messages may be stored in distinct blocks, and the same message may be printed several times.

Calling sequence:

α	020	α	01k	β
$\alpha+1$	xyz	0000	000	L
$\alpha+2$	Control returns to left of $\alpha+2$			

k=0	for message reading
k=4	for message printing

x,y,z = 0 or 1

Message Reading.

Referring to the blocks (L+0 through L+6), (L+7 through L+13) (L+14 through L+20) as α, β and γ respectively. If x = 1 the contents of C_{41-80} of a card will be encoded into α; if x = 0, α will not be altered. Similarly, no corresponding card will be read.

Thus, if, e.g. x = 1, y = 0, z = 1, then 2 cards will be read, C_{41-80} of the first card be encoded into α, C_{41-80} of the second card will be encoded into γ and β will be unaltered. If x = 0, y = 0, z = 0 then neither α, β, nor γ will be altered and no cards will be read.

Message Printing.

If $x = 1$, the message encoded in α will be printed in columns 1-40 of the printer; if $x = 0$ C_{1-40} will be left blank. Similarly for y, β and Cols 41-80 as well as z, γ and Cols 81-120. If $x = 0$, $y = 0$, $z = 0$, there will be no printing but the printer will be spaced.

CAUTION: In many cases it will be better to use several library routines rather than J227.

NOTE: The 40 columns of information stored in each block α, β, and γ are in Mouse form.

J227 Postspaces (i.e., it spaces after printing)

Symbolic Code: 213 words

Region A (erasable): A0 to A43

Program available as J227

M. Shapiro

J228A OCTAL MEMORY DUMP - 4 PER LINE

J228 prints 4 full words and the location of the first word in octal on one line. All of memory is printed except that only the first line of a block of zero or equal lines is printed. The printer is spaced one to indicate such a block.

The first card of J228 is a special loader which puts cells 0003-1024 onto the drum, band 0, position 0. Only cells 0000-0002, which are used by this loader, are printed incorrectly.

Program length: effectively 3 words (0000-0002) in HSS.

Program halt: 0000 Printing complete, memory restored.

Program available at J228A.

Marvin Shapiro

J230A OCTAL MEMORY DUMP W/O DRUM 2-24-58

J230 is a modification of J228 for use without the drum. J230 prints four full words and the location of the first word in octal on a line. Only the first line of a block of equal lines is printed.

One or more control cards are required by J230. The locations of the first and last words to be printed are punched in the 9's row right of the control card. The first location goes in the left address, the last location in the right address.

Restrictions:

(1) Only one control card may refer to addresses $\geq 7450)_8$ and it must be the last control card.

(2) If no reference is made to locations ≥ 7450, only the control cards and 2 blanks should be placed after J230 (see below).

Use:

(1) Load in primary. Overflow Ign. OFF.

(2) J230 will punch from $7750)_8$ to $7777)_8$.

(3) Press LOAD.

(4) J230 will punch cells $7450)_8$ to $7747)_8$, and run the punched cards followed by 2 blanks into the stacker of the punch.

(5) Place the control card(s), the binaries which were just punched,and the 2 blanks after J230.

(6) Press LOAD or GO.

(7) J230 will stop with the reader on SELECT.

Cells 0000, 0001, 0002 are printed incorrectly.

Memory is not restored.

Program Length:

$0000(1)0002$, $7120)_8(1)7777)_8$ or $7450)_8(1)7777)_8$ if no location is $\geq 7450)_8$.

Program Halts:

NIA = 0003, overflow indicator ON.
"Sneak on" is complete.

NIA = $7764)_8$. Memory from $7450)_8$ to $7747)_8$ has been punched.

NIA = $7510)_8$, reader on SELECT. Final stop.

W. L. Sibley

4-11-57

J232A INSERT BLOCK CHECK SUMS INTO STYLE F DECK

J232 converts a style F deck (with or without check sums or with incorrect check sums) into a style F deck with block check sums. To use J232 place

1 Blank Card

J232A

Control Card

Style F Deck to be Converted

in primary; load and go. When the computer stops with the reader selected run the cards out UNTIL THE COMPUTER STOPS WITH A 13X.

The punched cards will carry an I.D in C_{1-5} as determined by C_{1-5} of the control card, and will be sequenced in C_{6-8} beginning with one greater than the number in C_{6-8} of the control card. To begin sequencing with 001, leave C_{6-8} of the control card blank. To begin sequencing with 000, punch 999 into C_{6-8} of the control card. The punched cards will be echo checked.

T1 Suppresses Echo Checking

T2 Suppresses Sequence Numbering.

T1 and T2 may not be changed once the routine begins.

Stops	Echo Check Stop	- Reload	NIA = 1202_8
	Loading Error	- Reload	NIA = 0015_8
	Job Over		NIA = 1056_8

Speed Normally about 50 cards per minute.

For full details concerning what J232 will do under all circumstances consult the decimal listing or see Norman Shapiro.

Available as J232A

Norman Z. Shapiro

J233E DECIMAL PRINT ROUTINE*

J233 converts to decimal and prints a block of H.S.S. beginning at a specified location, according to scale factors and other specifications stored somewhere in H.S.S. The resulting printed page will have one space at the left, sign and first word, space, sign and second word, space, etc. There will be printing only in the first 40 columns.

J233 will assume the printer has been prespaced and will itself prespace the printer.

Let

a	=	number of words per line
b	=	location of first word to be printed
c	=	number of lines
d	=	location in H.S.S. of scaling factors, etc. (see below)
p_i	=	number of decimal digits left of the decimal point of the i th word.
q_i	=	location of binary point in i th word
n_i	=	number of decimal digits of i th word to be printed.

Thus $\sum_{i}^{a} n_i + 2a \leq 40$.

Calling sequence:

```
    α   020   α   010   β
α + 1    a    b    c    d
α + 2  Control returns here
```

(continued)

Locations d (1) d+a-1 in H.S.S. have

d	p_1	q_1	n_1 ← what's the format?
d+1	p_2	q_2	n_2
.	.	.	.
.	.	.	.
.	.	.	.
d+i-1	p_i	q_i	n_i
.	.	.	.
.	.	.	.
.	.	.	.
d+a-1	p_a	q_a	n_a

If a number is incorrectly scaled, ⌀ (0 and 7 overprinted) will appear in the sign position followed by all zeros. The program will not stop.

Region 1 (erasable) - 18 words

Region 3 - 102 words

Program available as J233B,E

* This routine is J110 modified for the new printer.

11-22-57 J.C. Shaw

J234A

OCTAL MEMORY DUMP*

This program prints two full words and the location of the first word in octal on one line in C_{1-40}. Lines having both words zero are omitted. The section of memory to be printed is specified by a control card or set of control cards.

The program uses a special loader which writes the contents of high-speed memory from 0002_{10} to 0499_{10} onto drum (drum 0, position 0, band 0), so that all of memory except locations 0000 and 0001 may be printed.

Printing is terminated by a blank control card. If the cards have not been run out of the reader at the conclusion of printing, memory is restored (except locations 0000 and 0001).

Deck: J234A
Control cards
3 blanks

Program length: effectively 2 words in high-speed storage
$(764)_8 = (500)_{10}$ on drum

Stops: 0000 Printing complete, memory restored.

Control Cards: One for each block of memory desired.
First address punched in binary in 9 row, col.48-59
Last address punched in binary in 9 row, col.69-80

Program available as J234A.

* This routine is J114A modified for the new printer.

J235E PRINT POSITIVE DECIMAL INTEGERS*

Calling sequence:

Loc	OP	ADDR	OP	ADDR
α		A	020	α
$\alpha+1$	010	β		N
$\alpha+2$	Control returns here			

where: A is the location of the first word.

N, the number of consecutive words to be printed.

J115 will convert consecutive positive words to decimal integers and print them one per line.

Region 3 - 36 words

Region 1 - 10 words

Program available as J235B,E

* This routine is J115 modified for the new printer.

J238E

PRINT DECIMAL - ONE PER LINE*

Calling Sequence:

LOC	OP	ADDR	OP	ADDR
α		A	020	α
$\alpha+1$	010	β		N
$\alpha+2$		Q		K
$\alpha+3$	Control returns here			

where: A is the loc. of first word to be printed.

N, the number of consecutive words to be printed.

Q, the scaling or location of the binary point for <u>all N</u> words $0 \leq Q \leq 39$.

K, the number of digits to be printed in the fractional part of the number $1 \leq K \leq 20$.

J238 converts to and prints in decimal N consecutive words $a_1, \ldots, a_N$ (where $|a_i| < 1$) of HSS beginning with location A. All numbers must be scaled the same in a block, and the same number of digits are converted in the fractional part for the block. The minus sign, when appropriate, is printed at the left in the 10^{13} position. Since K can never be less than 1, a number with a Q of 39 will appear as XXXXX.0 and a number with a Q of zero will appear as 0.XXXX . . . Conversion of the decimal part terminates as soon as the quotient of a division by ten is zero so that one digit integers appear as X.0, two digit integers XX.0, etc.

(continued)

Region 3 - 59 words

Region 1 - 11 words

Program available as J238B,E.

* This routine is J119 modified for the new printer.

6-1-56

J239F JOHNNIAC FLOATING-POINT INTERPRETIVE SYSTEM

J239F is an absolute binary assembly of the style E programs J240-J250, inclusive. The absolute location of the first word of the program storage for each routine can be found below.

	OCTAL	DECIMAL
J240 E 000	5500	2880
J241 E 000	6262	3250
J242 E 000	6461	3377
J243 E 000	6607	3463
J244 E 000	6733	3547
J245 E 000	7306	3782
J246 E 000	7341	3809
J247 E 000	7415	3853
J248 E 000	7467	3895
J249 E 000	7544	3940
J250 E 000	7612	3978
A 000	7713	4043
E 000	7763	4083

At the time of this writing, the Error Halt Location corresponds to NIA = 6160_8.

Reference: JOHNNIAC Floating-Point Interpretive System Manual

Program Storage (octal): 5500 - 7704 inclusive
7713 - 7777 inclusive

J. Derr

J240E INTERPRETER-JOHNNIAC FLOATING-POINT SYSTEM

J240E interprets and executes the following floating-point operations:

(i) the "zero class" operations

(ii) 010, 011, 014, and 015 operations

(iii) the "two," "three," and "four class" operations

(iv) the 050 operation

In addition, J240E "interprets" all floating-point operations, and contains some constants, machine-language code, and storage registers which are at least shared with other operations. These other operations correspond to JOHNNIAC Literary programs which are (<u>at loading time</u>) masters to J240E. They include J241E, J242E, ..., J250E. The master routines refer to the slave routine, J240E, as region F, i.e. J240E000 = F000. It is natural then for the user to cause the first word of J240E to coincide with F000. It is clear that region F must be assigned an absolute location before any of the master routines are loaded. Moreover, it is also necessary that the loading of J240E itself precede that of any of the master routines.

<u>Basic Link to J240E</u>   020 $0 010 F000.

<u>Key Words in J240E</u>

J240E003	Instruction counter (in Left Address)

J240E140	Trap Register - Breakpoint tracing
J240E141	Trap Register - All Orders tracing
J240E142	Trap Register - Transfer tracing
J240E148	Series Test Number
J240E209	AMQ Exponent
J240E210	AMQ Mantissa
J240E228	First word of closed routine which counts significant digits
J240E245	First word of closed routine which normalizes AMQ mantissa
J240E303	Error Halt Location

<u>Remarks on the Use of J240E</u>

1. The series test number is involved in the series calculations for the 054, 055, and 056 floating-point operations. If the series is denoted by $\sum_{1}^{\infty} x_n$ and the series test number by #, the test for convergence depends on the sign of $|x_n| \cdot 2^{-5} - \#$. Unless altered by the user, $\# = 2^{-39}$.

2. To normalize the AMQ execute the following machine language instructions with the AMQ mantissa in the MQ.

```
020    $0    010    F228
020    $0    010    F245
```

3. To suppress the Error Halt, replace the left operation of F303, (13X), by the corresponding unconditional transfer operation, (01X). Originally J240E303 = F303 contains

    134 F191 000 0000.

   Following the halt, the interpreter omits the trace printing step, but otherwise proceeds as described in the JOHNNIAC Floating Point Interpretive System manual. However, F303 is altered by the loading of J244E. The reader is in this connection referred to the J244E writeup.

4. J240 cannot be loaded by X267 (to become J267) without modification since too many forward references occur. Suggestion: Replace the contents of 0045 (absolute decimal) by 0200 (decimal); e.g. place immediately behind the last card of X267 (X267A038) a style E card with 0045 in columns 14-17, 0200 in columns 32-35, and , in column 36. Note that the length of the X267 program storage is thereby increased by 100 words.

<u>Program</u> <u>Storage</u>:

Region E (Semieraseable): 13 words, E000-E012.

Region E is available to the user at all times when outside interpreter control.

Region F: 370 words, F000 - F369.

J. Derr

11 August 1958

J241E ADDRESSABLE INPUT - JOHNNIAC FLOATING POINT SYSTEM

When used in conjunction with J240E, J241E will execute the C12 floating point operation as described in the JOHNNIAC Floating-Point Interpretive System manual. J240E must be loaded prior to the loading of J241E, and the first word of J240E must coincide with F000.

Program Storage: 127 words (there are 128 style E cards - J241E000 to J241E127 inclusive.)

Erasable Storage: 34 words - A000 to A033

Norman Shapiro

11 August 1958

J242E INPUT DATA CARDS – JOHNNIAC FLOATING POINT SYSTEM

When used in conjunction with J240E, J242E will execute the 013 floating point operation as described in the JOHNNIAC Floating Point Interpretive System manual. J240E must be loaded prior to the loading of J242E, and the first word of J240E must coincide with F000.

Program Storage: 86 words (87 style E cards – J242E000 to J242E086).

Erasable Storage: 30 words – A000 to A029.

Robert Mercer

11 August 1958

J243E PUNCH DATA CARDS - JOHNNIAC FLOATING POINT SYSTEM

When used in conjunction with J240E, J243E will execute the 016 floating point operation as described in the JOHNNIAC Floating Point Interpretive System manual. J240E must be loaded prior to the loading of J243E, and the first word of J240E must coincide with F000.

Program Storage: 84 words (85 style E cards - J243E000 to J243E084).

Erasable Storage: 27 words — A000 to A026.

Robert Mercer

11 August 1958

J244E PRINT AND TRACE – JOHNNIAC FLOATING POINT SYSTEM

When used in conjunction with J240E, J244E will execute the 017 floating point operation and the tracing as described in the JOHNNIAC Floating Point Interpretive System manual. J240E must be loaded prior to the loading of J244E, and the first word of J240E must coincide with F000.

One result of the loading of J244E is to modify the Error Halt Location – J240E303 $\equiv$ F303 – so as to become

130 F183 000 0000.

1. To suppress the Error Halt, replace the 130 in F303 by 010 after loading J244.
2. To suppress the tracing following the Error Halt, replace 130 F183 in F303 by 134 F191 after loading J244.

Program Storage: 235 words (239 style E cards – J244E000 to J244E238).

Region E(Semierasable): 13 words, E000 – E012.

Region A(Erasable): 40 words, A000 – A039.

Note: A39 = 7777_8 will result in printing 1 word per line.

Marvin Shaprio

8-11-58

J245E SQUARE ROOT - JOHNNIAC FLOATING POINT SYSTEM

When used in conjunction with J240E, J245E will execute the 051 floating point operation as described in the JOHNNIAC Floating Point Interpretive System manual. J240E must be loaded prior to J245E, and the first word of J240E must coincide with F000.

Program Storage: 27 words (28 style E cards - J245E000 to J245E027).

Region E: 13 words, E000 - E012.

John Derr

11 August 1958

J246E SINE - COSINE - JOHNNIAC FLOATING POINT SYSTEM

When used in conjunction with J240E, J246E will execute the 052 and/or 053 floating point operations as described in the JOHNNIAC Floating Point Interpretive System manual. J240E must be loaded prior to J246E, and the first word of J240E must coincide with F000.

Program Storage: 44 words (46 style E cards - J246E000 to J246E045).

Region E: 13 words, E000 - E012

John Derr

11 August 1958

J247E ARC TANGENT - JOHNNIAC FLOATING POINT SYSTEM

When used in conjunction with J240E, J247E will execute the 054 floating point operation as described in the JOHNNIAC Floating Point Interpretive System manual. J240E must be loaded prior to J247E, and the firstword of J240 must coincide with F000.

Program Storage: 42 words (43 style E cards - J247E000 to J247E042).

Region E: 13 words, E000 - E012.

John Derr

11 August 1958

J248E EXPONENTIAL – JOHNNIAC FLOATING POINT SYSTEM

When used in conjunction with J240E, J248 will execute the 055 floating point operation as described in the JOHNNIAC Floating Point Interpretive System manual. J240E must be loaded prior to J248E, and the first word of J240E must coincide with F000.

Program Storage: 45 words (46 style E cards – J248E000 to J248E045).

Region E: 13 words, E000–E012.

John Derr

11 August 1958

J249E LOGARITHM - JOHNNIAC FLOATING POINT SYSTEM

When used in conjunction with J240E, J249 will execute the 056 floating point operation as described in the JOHNNIAC Floating Point Interpretive System manual. J240E must be loaded prior to J249E, and the first word of J240E must coincide with F000.

Program Storage: 38 words (40 style E cards - J249E000 to J249E039).

Region E: 13 words, E000 - E012.

John Derr

11 August 1958

J250E INDEXING - JOHNNIAC FLOATING POINT SYSTEM

When used in conjunction with J240E, J250E will execute the 070-074 floating point operations as described in the JOHNNIAC Floating Point Interpretive System manual. J240E must be loaded prior to J250E, and the first word of J240E must coincide with E000.

The Indexing Registers are stored in the following words of J250E.

Index Register	Location
A	J250E028 (29th word of J250E)
B	J250E020 (21st word of J250E)
C	J250E016 (17th word of J250E)
D	J250E014 (15th word of J250E)
E	J250E013 (14th word of J250E)
F	J250E012 (13th word of J250E)

Program Storage: 59 words (64 style E cards - J250E000 to J250E063).

Region E: 13 words, E000 - E012

John Derr

7-10-57

J253E FIXED POINT SQUARE ROOT[1]

J253E is a closed subroutine which will evaluate the non-negative solution of the equation $x^2 = |a|$, i.e. $x = +\sqrt{|a|}$. Before executing the basic link into J253, a must be in the MQ, where $0 \leq |a| < 1$. When J253 links back to the main program x will be in the Accumulator and $0 \leq x < 1$.

Calling sequence:

```
020   $   0   010   #   0   (Link to J253E)
```

Description of the Method:

J253 first normalizes a so that $2^{-2} \leq a \cdot 2^{2q} < 1$, provided that $a \neq 0$. Let $y = a \cdot 2^{2q}$. $y_o = \begin{cases} 1/4 + y \text{ if } 1/4 \leq y < 1/2 \\ 1/2 + y/2 \text{ if } 1/2 \leq y < 1 \end{cases}$

Then if $y_o \neq y$, the following form of the Newton iteration scheme is used until $y_{n+1} \doteq y_n$:

$$y_{n+1} = y_n + 1/2 \left[y/y_n - y_n \right].$$

Finally, $x = y_n \cdot 2^{-q}$.

Accuracy: Let $\epsilon(a) = x - \sqrt{|a|}$ be the absolute error. Then $\epsilon(a) = 0$ or 2^{-39}. Observe that x has approximately q more significant binary digits than a.

Timing: Let T(a) denote the time in ms required to evaluate $\sqrt{|a|}$. Then .3 ms. < T(a) < 14 ms. for every admissable a. In particular, we list the following extreme cases:

1 J253E replaces J171E

$T(0) \doteq .3$ ms.

$T(1-2^{-39}) \doteq$ <u>1.23 ms.</u>

$T(2^{-1}) \doteq$ <u>6 ms.</u> (requires the maximum number (3) of iteration cycles).

$T(2^{-39}) \doteq 14$ ms (requires the maximum number (3) of iteration cycles <u>and</u> 19 normalization cycles).

Region A (erasable) - 3 words, A 0 - A 2

Symbolic Code: 23 words

Program available as J253E.

John I. Derr

J254E P^th ROOT, $1 \leq P \leq 39$ (Newton's Method with Sophisticated FIRST GUESS).

J254E will evaluate the non-negative solution of the equation $x^P = |a|$, i.e. $x = \sqrt[P]{|a|}$, when $|a|<1$. If $a = -1$, x is set equal to -1.

J254 is entered by basic link with a in the MQ. When J254 links back to the main program x will be in the Accumulator.

Calling Sequence:

α	020 α 010 β	(Link to J254E)
$\alpha+1$	P	
$\alpha+2$	Control returns here with -1 in Accumulator if $a = -1$.	
$\alpha+3$	Control returns here with x in Accumulator if $|a|<1$.	

Description of the Method:

(i) If $a = 0$, proceed to step (xi).

(ii) If $a = -1$, control returns to $\alpha + 2$ with -1 in Accumulator.

(iii) Normalize a so that $2^{-P} \leq a \cdot 2^{Pq} < 1$. Let $y = a \cdot 2^{Pq}$.

(iv) Set $x_0 = \begin{cases} \dfrac{P-1+y}{P} & \text{if } \dfrac{P-1}{2^P-2} \leq y < 1 \\ \dfrac{P-1+2^P y}{2P} & \text{if } 2^{-P} \leq y < \dfrac{P-1}{2^P-2} \end{cases}$.

(v) If $x_0 \leq y$, proceed to step (x).

(vi) Comp $P(\Delta x)_n = \left[y/x_n^{P-1} - x_n \right]$.

(vii) If $P(\Delta x)_n \geq 0$, proceed to step (x).

(viii) Replace x_n by $x_n + (\Delta x)_n$.

(ix) If $(\Delta x)_n + C < 0$, repeat step (vi) $(0<C<2^{-20})$.

(x) Place $x = x_n \cdot 2^{-q}$ in Accumulator.

(xi) Return to $\alpha+3$.

Accuracy:

$x \cdot 2^q$ has about as many correct significant bits as does y.

x has approximately $(P-1)q$ more significant bits than a.

The maximum error ϵ incurred in steps (viii) and (x) satisfies $\epsilon \leq 2^{-39}$. By rounding the number residing in the combined Accumulator and MQ upon return to $\alpha+3$ the user can reduce ϵ so that $\epsilon \leq 2^{-40}$.

Timing:

Let $\epsilon(y)$ denote $x_0 - y^{1/P}$. For fixed P $\max\limits_{2^{-P} \leq y < 1} \epsilon(y)$ occurs when $y = \frac{P-1}{2^P-2}$. For $y = \frac{P-1}{2^P-2}$, the time required to compute x varies roughly as the square of P since the number of iterations as well as the time per iteration varies approximately as P. For all P $\epsilon(2^{-P}) = \epsilon(1-2^{-39}) = 0$. On $[2^{-P}, 1]$, $\epsilon(y)$ satisfies $0 \leq \epsilon(y) \leq \epsilon(\frac{P-1}{2^P-2})$ and is cusplike in appearance with a sharp peak at $(P-1)/(2^P-2)$.

P	$(P-1)/(2^P-2) = y$	$x_0 - y^{1/P}$	# iterations
2	.500	.043	4
3	.333	.084	4
4	.214	.123	5
5	.133	.158	5
7	.048	.217	7
10	.0088	.277	8
15	.00043	.337	11
20	.000018	.371	14
25	.00000072	.392	18
30	.000000027	.407	21
35	.00000000099	.418	26
38	.00000000013	.424	26

Region A (eraseable) - 7 words, A0 - A6
Symbolic Code: 44 words

Program available as J254E.
John Derr

J255 — CAT READ ROUTINE 12-5-57

Calling Sequence:

LOC	OP	ADDR	OP	ADDR
α-k	SEL	0 or 1		
⋮				
α	RA	α	TRL	β
α+1	Control Returns			

The CAT read routine will do one of two things: either read a card image undisturbed into Region B (see below) without using the CAT read compiler or, having compiled the CAT read routine, it will read and convert to integers scaled 2^{-39} the fields defined in the CAT read compiler calling sequence and store them in Region A as follows:

Field 1 $\rightarrow$ A 0
⋮ ⋮
Field n $\rightarrow$ A n-1

The retained columns of the card image are stored in Region B as follows:

B 1	C_{1-40}	12's row
B 2	C_{1-40}	11's row
⋮		
B 12	C_{1-40}	9's row
B 13	C_{41-80}	12's row
⋮	⋮	
B 24	C_{41-80}	9's row

Storage Requirements:

Reg A, A0 - A25 (26 words)

Reg B, B1 - B24 (24 words)

Read Routine - C0 - C130 (131 words)

Timing:

5.6 ms. are used before first copy is given and 0.32 ms. are used after the twelfth copy.

J256 — CAT READ COMPILER 12-5-57

Calling Sequence:

LOC	OP	ADDR	OP	ADDR
α	RA	α	TRL	β
$\alpha+1$		P		S
$\alpha+2$	$\longleftarrow$	M_L	$\longrightarrow$	
$\alpha+3$	$\longleftarrow$	M_R	$\longrightarrow$	
$\alpha+4$		F_1		L_1
$\vdots$		$\vdots$		$\vdots$
$\alpha+n+3$	100	F_n		L_n
$\alpha+n+4$	Control Returns here			

Where: P is the origin of the CAT Read Routine;

S is the sign convention indicator,

S = 0, signs are ignored;

S = 1, each negative field bears an 11 overpunch in its high order position;

S = 2, each negative field bears an 11 overpunch in its units position;

M_L is a 40 bit mask with 1's indicating columns in C_{1-40} which are to be retained in the image, 0's indicating columns to be deleted. One need not save fields or columns being converted by the CAT read routine, the masks are independent of this process;

M_R is the same as M_L except for C_{41-80};

F_i is the first column of the i$^{\underline{th}}$ field;

L_i is the last column of the i$^{\underline{th}}$ field ($L_i - F_i \leq 8$, $F_{i+1} > L_i$)

n is the number of fields defined ($1 \leq n \leq 12$)

Note that the left operation of the n$^{\underline{th}}$ field definition contains the termination signal 100.

The CAT read compiler does the following things.

1) Compiles the conversion portion of the CAT read routine to read and convert fields to binary integers (scaled 2^{-39}) complying with the n field definitions in the calling sequence.

2) Compiles the sign detection portion of the CAT read routine according to the sign convention (S) specified and the field definitions.

3) Generates a decimal input table beginning at location 0000 of length 2^k where k is the length of the largest field defined.

Some general information:

1) During number conversion overpunched 12's are ignored.

2) Blanks and zeros are interchangeable in numeric fields.

3) Multiple numeric punches (e.g., decimal point = 12, 3, 8) result in the sum of the decimal digits (e.g., eleven).

4) It is impossible to convert numbers from the card without first compiling the CAT read routine. On the other hand the CAT read routine used without prior compilation will read and retain an 80 column image in B1 thru B24.

Program Stop:

C $\underline{0}$ $\underline{1}$ $\underline{3}$ $\underline{4}$, largest field > 9 cols no recovery procedure.

Storage Requirements:

CAT read compiler, Reg C C0 thru C209 (210 words)

Reg A, A0 (1 word)

Reg B, not actually used, but must be defined before loading J256.

Timing:

5.4 + 4.9n + 7.1k ms. where n is the number of fields defined and k is the length of the longest field.

J257 – CAT BCD FROM IMAGE 12-5-57

Calling Sequence:

LOC	OP	ADDR	OP	ADDR
α	RA	α	TRL	β
α+1		F		L
α+2	Control Returns here			

Where F is the first column of the field and L the last column.

This routine converts the defined field (F and L) to BCD characters (see appendix) packed 6/word in bits 2^{-4} thru 2^{-39}, six bits per character and stores them beginning with A 0 thru A n. (The determination of n is left as an exercise for the student for all cases from one column fields thru 80 column fields.)

Space Required:

C0 thru C127 (128 words)

A0 thru A15 (16 words)

B1 thru B24 (24 words - the image)

Timing:

Slow!

Appendix to CAT-BCD from Image

The Hollerith characters will have the following binary representations in the CAT system:

Hollerith	BCD	Hollerith	BCD
Blank	00	P	47
0	60	Q	50
1	01	R	51
2	02	S	62
3	03	T	63
4	04	U	64
5	05	V	65
6	06	W	66
7	07	X	67
8	10	Y	70
9	11	Z	71
A	21	=	13
B	22	+	20
C	23	.	33
D	24	)	34
E	25	-(11)	40
F	26	$	53
G	27	*	54
H	30	/	61
I	31	,	73
J	41	(	74
K	42	-(8,4)	14
L	43		
M	44		
N	45		
Ø	46		

J258 – CAT OUTPUT ROUTINE

12-5-57

Calling Sequence:

LOC	OP	ADDR	OP	ADDR
α	RA	α	TRL	β
$\alpha+1$		N		
$\alpha+2$		F_1	T_1	L_1
$\vdots$		$\vdots$	$\vdots$	$\vdots$
$\alpha+n+1$	100	F_n	T_n	L_n
$\alpha+n+2$	Control Returns here			

Where: N is the origin of the block where words to be converted to the image are stored

F_i, the first column of the ith field

L_i, the last column of the ith field.

T_i, the type of information in the ith word of block N.

T can have the following values and corresponding meanings:

T = 0 or blank = Error (see Error stop)

T = 1, Integer, scaled 2^{-39}

T = 2, Fraction, scale 2^{0} $(-1 < \chi < 1)$

T = 3, BCD packed 6/word (2^{-4} thru 2^{-39}, 6 bits/character)

This routine will convert and store in the Image the following types of information:

1) Integers scaled 2^{-39}
2) Fractions scaled 2^{0} $(-1 < \chi < 1)$
3) BCD characters packed 6/word.

There is no limit to the number of fields defined (except of course the 80 cols. of the Image) nor to the length of a field (except BCD fields where $L_i - F_i \leq 5$).

Integers will be placed in the image with leading zeros eliminated.

Fractions will be placed in the image truncated; any rounding should be done prior to storing in block N.

Negative numbers will have an 11 punch placed over the last column of the defined field.

Program stop:

C122 - $T_1 = 0$, no recovery procedure

Storage Requirements:

C0 - C143 (144 words)

B1 - B24 (24 words - the image)

Timing:

A rough estimate is about 2 ms per character stored in the image.

J259 - CAT ALPHANUMERIC PRINT 12-5-57

Calling Sequence:

LOC	OP	ADDR	OP	ADDR
α	RA	α	TRL	β
$\alpha+1$	Control Returns			

The CAT print routine will print the prepared card image (via CAT output or otherwise) into either the left or right 80 columns of the ANelex 56-120 printer. The select and space orders must be given before entering this routine.

Overpunched signs (such as produced by the CAT output routine) will appear as alphabetic characters, therefore, special care must be taken with negative numbers when placing them in the image.

Space Required:

CO - C143 (144 words)

AO - A104 (105 words)

B1 - B24 (24 words - the image)

Timing:

Slow

NOTE:- An 8-4 combination in the image will print as < instead of –.

J260 - CAT DECIMAL PRINT 12-5-57

Calling sequence:

LOC	OP	ADDR	OP	ADDR
α	RA	α	TRL	β
α+1	Control Returns			

The CAT decimal print routine will print a CAT card image with numeric information only. The following exceptions are included in the routine; 11 bits or punches are printed as minus signs and 12 bits or punches are printed as periods.

In this routine overpunched signs will cause overprinting, thyratron hangup or both, therefore negative numbers require special handling before using the CAT Output Routine to prepare an image.

The proper select and space instructions must be given before entering this routine

Space Required:

C0 - C11 (12 words)

B1 - B24 (24 words - the image)

Timing:

The routine uses $\sim$ .25 ms before the first copy order is given and exits $\sim$ .06 ms after the 12th copy is given.

J261 — CAT PUNCH 12-5-57

Calling Sequence:

LOC	OP	ADDR	OP	ADDR
α	RA	α	TRL	β
$\alpha+1$	Control Returns			

This routine will punch the stored image 80-80. The proper select order must be given before entering this routine. Echo-checking is impossible using this routine since control is not returned to the main program between copies.

Time Used:

Before 9's copy $\sim$ 0.4ms.

After 12's copy $\sim$ 0.2ms.

Space Required:

C0-C13 (14 words)

B1-B24 (24 words - the image)

J262 - CAT MOVE IMAGE 12-5-57

Calling Sequence:

LOC	OP	ADDR	OP	ADDR
α	RA	α	TRL	β
α+1		T		F
α+2	Control Returns			

Where T is the origin of the block the image is to be moved to, and F the origin it is to be moved from. Either T or F (not both) must be equal to zero; if T = 0, the routine interprets this to mean the image is to be moved from storage to the original residence (Reg. B) of the image, and if F = 0, the image is to be moved from Reg. B to some other space specified by T.

Space Required:

C0 thru C30 (31 words)

B1 thru B24 (24 words - the image)

Time Required:

9.6 ms.

J263 - CAT SELECTIVELY CLEAR IMAGE 12-5-57

Calling Sequence:

LOC	OP	ADDR	OP	ADDR
α	RA	α	TRL	α
α+1	Left Mask			
α+2	Right Mask			
α+3	Control Returns			

This routine will clear those columns designated by zeros in the left and right mask for C_{1-40} and C_{41-80} respectively and retain information in those columns designated by 1's.

Space Required:

C0 - C16 (17 words)

B1 - B24 (24 words - the image)

Time Required:

9.2 ms.

J264A[2] AUTOMATIC LEAST SQUARES POLYNOMIAL APPROXIMATION -- FLOATING POINT

J264 is an automatic version of J178. The user needs only to insert the current version of the Floating-Point System preceding the last card of J264, to record the necessary input data in punched cards, place these cards immediately behind J264, and press the LOAD button. The coefficients $\{a_i\}$ of the approximating polynomial

$$f(x) = \sum_{i=0}^{M} a_i x^i$$

will be computed and printed for each header card.

Input Form

Information should be punched in the Floating Point Data Form as follows:

Header card for a problem

pos (1)	S	+ if data triplets are to be printed
		- if data triplets are not to be printed
	EXP	M = degree of polynomial $f(x)$
	MANTISSA	ID (any decimal information)
pos (2)	S	+ if data cards follow
		- if no data cards follow
	EXP	Blank
	MANTISSA	N = number of points
pos (3)	S	Blank (or +)
	EXP	00 if (x_i, y_i, r_i)[1] is to be printed
		01 if (x_i, y_i, y_i^*) is to be printed
	MANTISSA	0

Positions (4), (5), and (6) of the header card are not interpreted by J264A.

1 $r_i = y_i^* - y_i$.

2 J264A is J212A modified for the new JOHNNIAC printer.

(x_i, y_i) Data Cards for a problem

The number pairs (x_i, y_i) must be punched in consecutive three-position blocks beginning with the first card following the header card.

	(1)	(2)	(3)	(4)	(5)	(6)
1st Card	x_1	y_1	---	x_2	y_2	---
⋮	⋮	⋮	⋮	⋮	⋮	⋮
last card (N even)	x_{N-1}	y_{N-1}	---	x_N	y_N	---+
(N odd)	x_N	y_N	---	---	---	---+

Further, there must be an end of file mark (12 punch) in column 80 of the last data card.

Cards for a sequence of problems

There is a one-to-one correspondence between header cards and problems. A sequence of problems can be solved by placing the associated header cards together with the data cards following (if any) in sequential order behind J264. If the entire deck of input cards is followed by two blank cards, the console will show "READER" following the completion of the last problem.

Output Form

The page will be ejected prior to the beginning of each problem. The numbers printed will be in the external floating point form. The print format is that described for the 017 (PNT) order in the Floating-Point write-up.

Printing for a problem

	A	B	C
	S EXP MAN	S EXP MAN	S EXP MAN
Information in header card	± M ID	± 00 N	± 0 $^0/1$ $^\delta/0$
Space	—	—	—
Coefficients	a_0	a_1	a_2
	⋮	⋮	⋮
Space	—	—	—

If the number triplets are to be printed, they are printed one triplet per line in the natural order.

x_1	y_1	$r_1\ (y_1{}^*)$
x_2	y_2	$r_2\ (y_2{}^*)$
⋮	⋮	⋮
x_N	y_N	$r_N\ (y_N{}^*)$.

Notes

(1) Prior to entry into J178, the set of $\left\{x_i\right\}$ is replaced by the set of $\left\{x_i - m\right\}$ where

$$m = \left(\sum_{i=1}^{N} x_i\right) \div N.$$

Upon exit from J178, J194 is used to compute the coefficients $\left\{a_i\right\}$.

(2) $1 \leq M \leq 5$ and $M+1 \leq N \leq 120$[1] must hold.

1 The upper bound of 120 on N was artificially imposed.

(3) A check sum 13X halt at $(0014)_8$ indicates that J264 has probably been loaded incorrectly under the control of J180A. The entire loading process should be restarted.

(4) A 13X halt will occur at $(2434)_8$ if the computational checking criteria is not met in any row during the forward solution in J196. The calculation will continue if the GO button is pressed. This halt can occur due to the loss of significant digits in the matrix solution.

(5) A 13X halt will occur at $(2533)_8$ if the computational checking criteria is not met in any row during the back solution. The calculation will continue if the GO button is pressed. This halt can occur due to the loss of significant digits in the matrix solution.

(6) A 13X halt will occur at $(2357)_8$ if det T = 0. If the GO button is pressed, then J264 will exit J196 and J178. If printing occurs upon completion of this problem, the results will be incorrect. J264 will continue ahead to the next problem if there is one.

Accuracy

Using $x_i = 0(.05)1$ and the exact corresponding values of y_i obtained from a table of Legendre polynomials of degrees M = 2, 3, 4, 5 the number of correct significant digits obtained in the computed coefficients was as follows:

M=2	9 digits	M=3	9 digits
M=4	8(-) digits	M=5	7 digits

Timing

The Legendre polynomials of degree one through five were used as test problems with 21 points, i.e., M=1, 2, 3, 4, and 5 and N=21. The following times (which do not include input-output times) were recorded while operating in the SD mode:

M=1	4 sec.	M=2	7 sec.
M=3	10 sec.	M=4	14 sec.
M=5	18 sec.		

Program available as J264A.

John Derr

J265B-E ADDRESSABLE DECIMAL INPUT

J265 will read and convert decimal numbers, one per card, into H.S.S. at the designated word relative to an origin specified in the calling sequence.

Calling Sequence:

LOC	OP	ADDR	OP	ADDR
α	RA	α	TRL	β
$\alpha+1$		F		R
$\alpha+2$	End of file return			

Where R is the origin relative to which input numbers are addressed, and F, the feed designation; F=0, primary feed; F=1, secondary feed.

Card Format:

C_1	- sign of address
C_{2-4}	- relative address
C_6	- sign of number
C_{7-18}	- number (including decimal point)
C_{20-21}	- b-the binary scaling at which the number is to be stored in H.S.S. $(0 \leq b \leq 40)$
C_{80}	- End of file (12-punch - see below)

Signs are denoted as follows, minus signs are 11-punches, plus signs, 12-punches or blank. Zeros and blanks are interchangeable in all fields.

In the number field, the integral and fractional parts of the numbers are separated by a decimal point. The decimal point may be left out for integers if the units digit is placed in C_{18}; in this case 12 digit integers are input modulo 549,755,813,888 (2^{39}).

b is defined as the number of binary digits between the sign bit (2^0) and the binary point in the word. For b=0, the number range is $-1 < x < 1$. All integers are converted exactly; fractions are rounded such that the error in representation is not greater than 2^{-k}, where k = 40-b.

End of file is denoted by a 12 punch in C_{80} of a separate card, it is read but the information on the card is not stored in H.S.S. C_{22-79} of all cards are not read.

Timing:

Reader is run at full speed for each file (240 cards/min.)

Storage:

C0 thru C120 (121 words)

A0 thru A19 (20 words)

Error Stops:

C82 - HTL C82 : b is too small, no recovery procedure.

M. I. Bernstein

J266B-E 120 COLUMN VARIABLE FORMAT DECIMAL PRINT

J266 will print up to 120 characters on a line in the format specified in the calling sequence.

Calling Sequence:

LOC	OP	ADDR	OP	ADDR
α	RA	α	TRL	β
$\alpha+1$		L_1		B_1
$\alpha+2$		T_1		N_1
$\vdots$				
$\alpha+2n-1$		L_n		B_n
$\alpha+2n$	100	T_n		N_n
$\alpha+2n+1$	Control Returns here			

Where L_i is the location of the ith specified word;

B_i is the binary scaling of the ith word ($0 \leq B_i \leq 39$);

T_i is the column in which the decimal point of the ith word is to appear ($1 \leq T_i \leq 120$); T_{i+1} need not be greater than T_i;

N_i is the number of digits to be printed to the right of the decimal point of ith word ($0 \leq N_i \leq 10$).

Note that the left operation of $\alpha+2n$ contains the termination signal (100) for the calling sequence.

All numbers are rounded by adding $.5 \times 10^{-n}$ prior to printing.

When N=0, the decimal point is not printed.

All leading zeros are suppressed with the exception that at least one digit is printed to the left of the decimal point, even when B=0.

Minus signs are printed to the left of the highest order digit printed, plus signs are not printed.

No provision is made for checking to see if an overlap of any two adjacent numbers has occured; the programmer must provide ample space to prevent this; it can cause the printer to hang up or overprinting, or erroneous printing.

All selects and spaces are given in the routine.

Timing:

The maximum print speed of this routine is 400 lines/min.

Space Required:

C0 - C187 (188 words)

A0 - A35 (36 words)

M. I. Bernstein

J268E* ZEROS OF A POLYNOMIAL WITH COMPLEX COEFFICIENTS -- FLOATING POINT SYSTEM

J268 will evaluate all of the roots of an n^{th} degree polynomial equation of the form

$$\sum_{i=0}^{n} C_i Z^i = 0,$$

where C_i is a complex number for $i = 0, \ldots, n$ and $n < 50$. Each root is computed as a complex number and, furthermore, the norm of the remainder obtained by dividing the reduced polynomial by the computed zero is supplied for each zero evaluated. J268 must be used in conjunction with the Floating Point Interpretive System. J268 makes references to the interpreter (J240E or J239F) as region F.

J268 is entered by basic linkage. Before the calling sequence is executed, however, the coefficients of the given polynomial must be in consecutive H. S. S. locations as follows:

C_n^R = real component of C_n in floating point internal form
C_n^I = imaginary " " " " " " " "

⋮

C_o^R = real component of C_o in floating point internal form
C_o^I = imaginary " " " " " " " "

Now if $C_n = 0$, let N be such that $C_N \neq 0$ and $C_{N+1} = C_{N+2} \cdots = C_n = 0$.

<u>Calling Sequence</u>:

```
α      020   $ 0       010   # 0  Link to J268
α+1    000   L(C_n^R)   n    L(N)
α+2    Control returns here
```

*J268 is J157 as modified for the revised JOHNNIAC FLOATING POINT SYSTEM.

Let Ω_j be the jth root of $f(Z) = \sum_{i=0}^{N} C_i Z^i = 0$. Upon execution of the basic link, J268 will proceed to find the zeros $(\Omega_1, \ldots, \Omega_N)$ of the reduced polynomials $g(Z)$ of $f(Z)$, one at a time, except in the case when Ω is a zero of Kth order multiplicity. (In the latter event Ω is evaluated only one time, but it is counted as K roots with respect to the output of J268). The roots Ω_j, together with $|g(\Omega_j)^R| + |g(\Omega_j)^I|$, are stored in consecutive H.S.S. locations beginning with L(N)+1 and ending with L(N)+3N as follows:

L(N)+1	Ω_1^R	Ω_1^I	$\|f(\Omega_1)^R\| + \|f(\Omega_1)^I\|$
L(N)+4	Ω_2^R	Ω_2^I	$\|g(\Omega_2)^R\| + \|g(\Omega_2)^I\|$
$\vdots$	$\vdots$	$\vdots$	$\vdots$
L(N)+3N-2	Ω_N^R	Ω_N^I	$\|g(\Omega_N)^R\| + \|g(\Omega_N)^I\|$

N is a fixed-point integer. All other numbers are in the floating-point internal form.

Description of the Method

The basic process used is Newton's method, i.e. if Z_n is the nth approximation to a root Ω, then

$$Z_{n+1} = Z_n - \frac{g(Z_n)}{g'(Z_n)} .$$

The criterion for convergence of Z_n to Ω requires that

$$\left|Z_{n+1} - Z_n\right| = \left|\Delta Z_n\right| = \left|\frac{g(Z_n)}{g'(Z_n)}\right|$$ satisfy the inequalities

$|\Delta Z_n| \leq |Z_n| \cdot 10^{-7}$ and $|\Delta Z_n{}^R| \leq |Z_n{}^R| \cdot 10^{-7}$.* Hence $|g(\Omega)^R| + |g(\Omega)^I|$ yields an independent check on the accuracy of the calculation of Ω. Note that this latter check involves an absolute error with respect to the polynomial, whereas the convergence **criterion** is based on a relative error with respect to Z_n.

In the case of a repeated root Ω of degree K, $K > 1$, Newton's method is rendered inadequate on two accounts. The convergence becomes linear rather than quadratic and the number of good significant digits in Z_n is reduced by a factor of at least two since $g'(Z_n) \to 0$ as $Z_n \to \Omega$. As a result, $g(Z_n)/g'(Z_n)$ becomes the ratio of two relatively small numbers. Here, the basic process is modified as follows: Let L be the first integer, $L > 1$, such that $|g^{L+1}(Z_n)| \geq Q \cdot \epsilon$ (Here we will omit the details concerning the explicit identity of Q and ϵ. Let it suffice to say that the test is relative versus absolute and that $g^{L+1}(Z_n)$ is bounded away from zero.) Now

$$Z_{n+1} = Z_n - \frac{(K-L)\, g^L(Z_n)}{g^{L+1}(Z_n)} \quad \text{and } \Delta Z_n = - \frac{(K-L)\, g^L(Z_n)}{g^{L+1}(Z_n)} .$$

Using this new form of ΔZ_n the test for convergence is the same as described in the preceding paragraph. Also the modified process is still at least a second order process.

Convergence and the Choice of Z_o

When the computation for any root Ω begins Z_o is computed either as a function of C_o or as the conjugate of the immediately

*The second inequality is designed to cover the situation where the imaginary component of Ω can be much larger than the real component of Ω and the real component is the only one desired. Problems exist in the field of dynamic analysis, for example, where this type of convergence is desired.

preceding root. The iteration will then proceed to compute Z_{n+1} as indicated above until either Z_{n+1} converges to Ω or $n = 13$. In the former case (the ordinary case), we are through. In the latter event a new Z_c is computed as a different function of C_o. This time n is allowed to reach $13 + 10 = 23$ before a new Z_o is computed. This process can be repeated until the 7th value of Z_o has been computed. If convergence has not been achieved when $n = 13 + 6 \cdot 10 = 73$, then the program will halt.

In order to give the user some insight into the reasons for selecting such an elaborate system for computing Z_o, we might outline some of the pathological cases which might arise and do not fit into the standard convergence pattern. We first distinguish two fundamentally different causes for abnormal convergence.

I. Newton's method is not, precisely speaking, an "error-squaring" process. In fact, it need not even be convergent at all. Basically $(\Delta Z)_{n+1} \doteqdot q \cdot (\Delta Z)^2_n$. Convergence is guaranteed if $|q \cdot (\Delta Z)_n| < 1$, but the rate of convergence still depends directly on q. If $|q| \doteqdot 1$, the convergence might be very slow. As an example of non-convergence consider the case where $g'(Zn) \leq Q_1 \cdot \epsilon_1$ and $g(Z_n) \geq Q_2 \cdot \epsilon_2$, i.e. Z_n is very near a root of $g'(Z) = 0$ which is not a root of $g(Z) = 0$.

II. Cycling can occur in this method due to two diverse causes. (By cycling we mean that a finite sequence $(Z_{\alpha_1}, Z_{\alpha_2}, \ldots, Z_{\alpha_m})$ of root approximation values are taken on in a cyclic manner.)

A. The propagated error which accumulates in the calculation of $g(Z_n)$ and $g'(Z_n)$ is so great that the convergence criterion cannot be met. This sort of cycling becomes more probable as the degree of N of $g(Z)$ increases.

B. A geometrical symmetry exists among the roots of $g(Z)$ relative to the approximation value Z_n.

Notes

(1) A Halt (130 operation) will occur if the process does not converge. If the GO button is pressed, Z_{n+1} will be placed in the root output storage and the program will proceed to evaluate the roots of the reduced polynomial.

(2) The roots are usually in error only in the low order two digits of the mantissa. The accuracy of a root varies, naturally, with the degree of the original polynomial and the position of the root in the sequence $(\Omega_1, \ldots, \Omega_N)$.

(3) The evaluation of the nth roots of unity is probably more difficult than the "average" polynomial occurring in practice due to the geometrical symmetry of the roots. The following approximate times in seconds were recorded while operating in the SD Mode:

$x^3 - 1$	17
$x^5 - 1$	23
$x^8 - 1$	122
$x^9 - 1$	59
$x^{16} - 1$	465
$x^{17} - 1$	329

(4) J268E requires only the interpreter (J240E) of the JOHNNIAC Floating-Point System. The first word of the interpreter (J240E000 or 5500_8) must coincide with F000. Region F must be

assigned on absolute location prior to the loading of J268.

Reg. A 332_{10} words (A 0 - A 331)*

Symbolic Code: $422_{10} = 646_8$ words

Program available as J268E.

John I. Derr

* A132-A331 are available to the user for storing coefficients for input to J268, i.e., it is permissible to have $L(C_n^R) = A132$. However, these coefficients will not be available upon re-entry to the main program.

J269E* COMPLEX MULTIPLY -- FLOATING POINT SYSTEM

J269 is a closed sub-routine which will multiply two complex floating-point numbers a and b. The arguments must be in region A prior to the execution of the basic link. The resulting complex floating-point product $c = a \cdot b$ will be found in region A when J269 links back to the main program. The arguments a and b will not be destroyed by J269. J269 makes references to the interpreter (J240E or J239F) as region F.

Calling Sequence

```
α       020   $ 0    010  # 0   (Link to J269)
α + 1 Control returns here.
```

The region A locations of a, b, and c are as follows:

A_0	contains	a^R = real component of a in fl. pt. internal form.								
A_1	"	a^I = imaginary	"	"	a	"	"	"	"	"
A_2	"	b^R = real	"	"	b	"	"	"	"	"
A_3	"	b^I = imaginary	"	"	b	"	"	"	"	"
A_4	"	c^R = real	"	"	c	"	"	"	"	"
A_5	"	c^I = imaginary	"	"	c	"	"	"	"	"

J269E requires only the interpreter (J240E) of the Floating Point System The first word of the interpreter (J240E000 or 5500_8) must coincide with F000. Region F must be assigned an absolute location prior to the loading of J268.

Region A (erasable) 6 words A 0 - A 5

Symbolic Code $10_{10} = 12_8$ words

Program available as: J269E

John I. Derr

* J269 is J169 as modified for the revised JOHNNIAC Floating Point System.

9-12-58

J270E* COMPLEX DIVIDE -- FLOATING POINT SYSTEM

J270E is a closed sub-routine which will obtain the complex floating-point quotient q of two complex floating-point numbers a and b. The dividend a and the divisor b must be in region A prior to the execution of the basic link to J270. The quotient q and the square of the norm of b $\left(|b|^2 = (b^R)^2 + (b^I)^2\right)$ will be found in region A when J270 links back to the main program provided that $b \neq 0$. If $b = 0$, q will not be computed. The arguments a and b will not be destroyed by J270. J270 makes references to the interpreter (J240E or J239F) as region F.

<u>Calling Sequence:</u>

```
α      020    $ 0    010 #0 (Link to J270)
α+1    Control returns here if b ≠ 0
α+2    Control returns here if b = 0.
```

The region A locations of a, b, q, and $|b|^2$ are as follows:

A_0	contains	a^R = real component	of	a	in	floating-pt. internal form
A_1	"	a^I = imaginary "	"	a	"	" " " "
A_2	"	b^R = real "	"	b	"	" " " "
A_3	"	b^I = imaginary "	"	b	"	" " " "
A_4	"	q^R = real "	"	q	"	" " " "
A_5	"	q^I = imaginary "	"	q	"	" " " "
A_6	"	$|b|^2$ = norm squared of b.				

J270E requires only the interpreter (J240E) of the Floating Point System. The first word of the interpreter (J240E000 or 5500_8) must coincide with F000.

Region A (erasable): 7 words A 0 - A 6
Symbolic Code $17_{10} = 21_8$ words
Program available as: J170E

John I. Derr

* J270 is J170 as modified for the revised JOHNNIAC Floating Point System.

9-16-58

J271E* SCALAR PRODUCT OF TWO VECTORS -- FLOATING POINT SYSTEM

J271 computes the usual inner product (x, y) w.r.t. the standard orthonormal basis over a Euclidean Vector Space, i.e.

$$(x,y) = \sum_{i=1}^{n} x_i y_i \ .$$

Denote (x, y) by γ. All of the numbers $\{x_i\}$ and $\{y_i\}$ must be in the packed internal form recognized by the Floating-Point Interpretive System. Upon exit from J271 γ will be in the same form. J271 makes references to the interpreter (J240E or J239F) as region F.

The elements $\{x_i\}$ of the vector $x = (x_1, \ldots, x_n)$ must be placed in H.S.S. as follows:

$$L(x_1),\ L(x_2) = L(x_1) + \Delta_x,\ \ldots,\ L(x_n) = L(x_1) + (n-1)\,\Delta_x,$$

where Δ_x is an integer and $|\Delta_x| \leq (377)_8 = 255_{10}$.

A similar statement holds for y.

Calling sequence:

α	020	α	010	β	(Link to J271)
$\alpha+1$	x00	$L(\gamma)$	$\lvert\Delta_x\rvert$	$L(x_1)$	
$\alpha+2$	y00	n	$\lvert\Delta_y\rvert$	$L(y_1)$	
$\alpha+3$	Control returns here				

Here x is defined by

$$\Delta[L(x_1)] = \begin{cases} |\Delta_x| & \text{if } x = 0 \\ -|\Delta_x| & \text{if } x = 1 \end{cases},$$

and a similar statement holds for y .

* J271 is J172 as modified for the revised JOHNNIAC Floating Point System.

J271 requires only the interpreter (J240E) of the Floating Point System. The first word of the interpreter (J240E000 or 5500_8) must coincide with F000.

Region A (erasable): 6 words A 0 - A 5

Symbolic Code: 31 words

Program available as: J271E

John I. Derr

J272E* SUM OF THE COMPONENTS OF A VECTOR -- FLOATING POINT SYSTEM

The function of J272 depends upon the contents of the left operation field of its 16th word which is 024 unless altered by the user. In particular, J272 computes

$$\gamma = \begin{cases} \sum_{i=1}^{n} x_i & \text{, if the field contains 024} \\ \sum_{i=1}^{n} |x_i| & \text{, if the field contains 026} \end{cases} .$$

All of the numbers $\{x_i\}$ must be in the packed internal form recognized by the Floating-Point Interpretive System. Upon exit from J272 γ will be in the same form. J272 makes reference to the interpreter (J240E or J239F) as region F.

The elements $\{x_i\}$ of the vector x must be placed in H.S.S. as follows:

$L(x_1)$, $L(x_2) = L(x_1) + \Delta$, ..., $L(x_n) = L(x_1) + (n-1)\Delta$, where Δ is an integer and $|\Delta| \leq 377_8 = 255_{10}$.

Calling Sequence:

α	020	α	010	β	(Link to J272)
$\alpha+1$	σ00	$L(\gamma)$	$\lvert\Delta\rvert$	$L(x_1)$	
$\alpha+2$	000	n	000	0000	
$\alpha+3$	Control returns here				

$$\Delta \equiv \Delta[L(x_i)] = \begin{cases} |\Delta| & \text{if } \sigma = 0 \\ -|\Delta| & \text{if } \sigma = 1 \end{cases} .$$

---

*J272 is J173 as modified for the revised JOHNNIAC Floating Point System.

J272 requires only the interpreter (J240E) of the Floating Point System. The first word of the interpreter (J240E000 or 5500_8) must coincide with F000.

Region A (erasable): 4 words A0 - A3

Symbolic Code: 23 words

Program available as J173E.

John Derr

J273E* LINEAR COMBINATION OF TWO VECTORS -- FLOATING POINT SYSTEM

J273 computes the linear combination $z = \beta x + y$ of the vectors x and y over a Euclidean Vector Space. Here x, y, and z are denoted by the n-tuples $(x_1, \ldots, x_n)$, $(y_1, \ldots, y_n)$, and $(z_1, \ldots, z_n)$. β is a scalar. All of the numbers $\{x_i\}$, $\{y_i\}$, and β must be in the packed internal form recognized by the Floating-Point Interpretive System. Upon exit from J273, the numbers $\{z_i\}$ will be in the same form. J273 makes reference to the interpreter (J240E or J239F) as region F.

The elements $\{x_i\}$ of the vector x must be placed in H.S.S. as follows:

$L(x_1)$, $L(x_2) = L(x_1) + \Delta_x$, ..., $L(x_n) = L(x_1) + (n-1)\Delta_x$, where Δ_x is an integer and $|\Delta_x| \leq 377_8 = 255_{10}$. Similar statements hold for y and z.

Calling Sequence:

α	020	α	010	----	(Link to J273)
$\alpha+1$	$\sigma_x 00$	$L(\beta)$	$\lvert\Delta_x\rvert$	$L(x_1)$	
$\alpha+2$	$\sigma_y 00$	n	$\lvert\Delta_y\rvert$	$L(y_1)$	
$\alpha+3$	$\sigma_z 00$	0000	$\lvert\Delta_z\rvert$	$L(z_1)$	
$\alpha+4$	Control returns here				

$$\Delta_x \equiv \Delta[L(x_1)] = \begin{cases} |\Delta_x| & \text{if } \sigma_x = 0 \\ -|\Delta_x| & \text{if } \sigma_x = 1 \end{cases} .$$

*J273 is J187 as modified for the revised JOHNNIAC Floating Point System.

Steps $\alpha+2$ and $\alpha+3$ can be interpreted in an analogous manner.

Two variations in the use of J187 are permitted. Together with the most general form already described there are three mutually exclusive functions available:

(i) $\beta x + y$, $L(\beta)$, $L(y_1)$ <u>must</u> be non-zero.

(ii) $x + y$, $L(\beta)$ <u>must</u> be zero (blank).

(iii) βx , $L(\beta)$ <u>must</u> be non-zero <u>and</u>
$L(y)$ <u>must</u> be zero (blank).

Essentially, J187 consists of three programs in one. The basic computational loop for each of the three functions described is optimal in the sense that the number of program steps is minimized subject to the condition that the smallest possible number of program steps shall be executed in the Floating-Point System.

J273 requires only the interpreter (J240E) of the Floating Point System. The first word of the interpreter (J240E000 or 5500_8) must coincide with F000.

Region A (erasable): 12 words (A0 - A11)

Symbolic Code: 65 words

Program available as: J187E.

John Derr

J274E* POLYNOMIAL FUNCTION EVALUATION -- FLOATING POINT SYSTEM

For a given value of x, J274 will compute

$$p(x) = \sum_{i=0}^{N} a_i x^i \ , \ N \geq 1.$$

J274 must be used in conjunction with the current version of the Floating Point System and x , $p(x)$, and the coefficients $a_0, a_1, \ldots, a_N$ must be in the packed internal form. Further, the coefficients must be placed in H.S.S. locations as follows: $L(a_0)$, $L(a_1) = L(a_0) + \Delta$, ..., $L(a_N) = L(a_0) + N\cdot\Delta$, where Δ is an integer and $|\Delta| \leq (377)_8 = (255)_{10}$. J274 makes references to the interpreter (J240E or J239F) as region F.

<u>Calling Sequence</u>:

α	020	α	010	β	(Link to J274)
$\alpha+1$	σ00	$L(\alpha_N)$	Δ	N	
$\alpha+2$		$L(x)$		$L[p(x)]$	
$\alpha+3$	Control returns here				

$$\Delta \equiv \Delta[L(a_1)] = \begin{cases} |\Delta| \text{ if } \sigma = 0 \\ -|\Delta| \text{ if } \sigma = 1 \end{cases} .$$

<u>Method</u>:

$p(x) = p_N(x)$, where $p_N(x)$ is generated by recursively defining $p_1(x) = a_{N-1} + a_N x$ and $p_{i+1}(x) = a_{N-i-1} + xp_i(x)$ for $i=1, 2, \ldots, N-1$.

---

*J274 is J191 as modified for the revised JOHNNIAC Floating Point System.

<u>Timing</u>:

No absolute times have been recorded, but N floating-point additions and multiplications are performed. Hence, for sufficiently large N the time will vary directly as N.

<u>Note</u>: J274 requires only the interpreter (J240E) of the Floating Point System. The first word of the interpreter (J240E000 or 5500_8) must coincide with F000.

Region A (erasable): 3 words A0 - A2

Symbolic code: 26 words

Program available as: J191E

John Derr

J275E* ITERATED SYNTHETIC DIVISION - FLOATING POINT SYSTEM

Given a polynomial $f(x) = \sum_{i=0}^{N} a_i x^i$, J275 will divide f(x) by x-c M times, where $1 \le M \le N$. More precisely, define the quotient polynomials $q_i(x)$ by recursion. Set $q_0(x) = f(x)$ and let $q_j(x)$ be defined by $q_{j-1}(x) = (x-c)\, q_j(x) + q_{j-1}(c)$ for j=1, 2, ..., M. Now let

$q_M(x) = \sum_{j=0}^{N-M} q_j^M x^j$ define $\left\{q_j^M\right\}$. Then J194 computes the set of N numbers $\left\{b_i\right\}$, where $b_i = \begin{cases} q_i(c), & i=0, 1, \dots, M-1 \\ q_{i-M}^M, & i=M, M+1, \dots, N \end{cases}$.

All of the numbers $\left\{a_i\right\}$, $\left\{b_i\right\}$, c, $\left\{q_i(c)\right\}$, and $\left\{q_{i-M}^M\right\}$ are in the packed internal form recognized by the current version of the floating point system. The $\left\{a_i\right\}$ and $\left\{b_i\right\}$ must be arranged in H.S.S. as follows:

$$L(a_0),\ L(a_1) = L(a_0) + \Delta_a,\ \dots,\ L(a_N) = L(a_0) + N\cdot\Delta_a \ .$$
$$L(b_0),\ L(b_1) = L(b_0) + \Delta_b,\ \dots,\ L(b_N) = L(b_0) + N\cdot\Delta_b \ .$$

Here, Δ_a and Δ_b are integers whose absolute values are bounded below $(400)_8 = (256)_{10}$. J275 makes references to the interpreter (J240E or J239F) as region F.

<u>Calling Sequence</u>:

α	020	α	010	β	(Link to J275)
$\alpha+1$	$\sigma_a 00$	N	$\lvert\Delta_a\rvert$	$L(a_0)$	
$\alpha+2$	$\sigma_b 00$	M	$\lvert\Delta_b\rvert$	$L(b_0)$	
$\alpha+3$		L(c)			
$\alpha+4$	Control returns here				

*J275 is J194 as modified for the revised JOHNNIAC Floating Point System.

$$\Delta_a = \begin{cases} |\Delta_a| & \text{if } \sigma_a = 0 \\ -|\Delta_a| & \text{if } \sigma_a = 1 \end{cases} \quad \text{and } \Delta_b = \begin{cases} |\Delta_b| & \text{if } \sigma_b = 0 \\ -|\Delta_b| & \text{if } \sigma_b = 1 \end{cases} .$$

Notes:

(i) Making use of the general relationship $f^{(k)}(c) = k!\cdot q_k(c)$ for $0 \leq k \leq N$, one can use J275 to compute the values of the derivatives of a polynomial $f(x)$ evaluated at $x = c$. In particular, for $k=0$ J275 duplicates the function of J274.

(ii) If $M = N$, then the $\{b_i\}$ computed by J275 also satisfy the equation

$$f(x) \overset{\text{def}}{=} \sum_{i=0}^{N} a_i x^i = \sum_{i=0}^{N} b_i (x-c)^i \overset{\text{def}}{=} g\ (x-c) .$$

(iii) The $\{a_i\}$ are not destroyed by J275 if $\{L(a_i)\}$ are disjoint from $\{L(b_i)\}$. However, setting $\Delta_a = \Delta_b$ and $L(a_0) = L(b_0)$ is permissible if the $\{a_i\}$ are expendable.

(iv) J275 requires only the interpreter (J240E) of the Floating Point System. The first word of the interpreter (J240E000 or 5500_8) must coincide with F000.

Timing:

$\frac{1}{2}M(2N-M+1)$ pairs of floating-point add and multiply orders are performed for each entry.

Region A (erasable): 7 words A0 - A6

Symbolic Code: 46 words

John Derr

J276E* MAXIMUM ELEMENT OF A VECTOR -- FLOATING POINT SYSTEM

J276 is a floating-point version of J174E. Let $x = (x_1, \ldots, x_n)$ be an n-tuple with real components. J276 will find the first element $x_k \in \{x_i\}$ such that $|x_k| \geq |x_i|$ for $i=1, \ldots, n$. The element x_k will be in A4 upon exit from J276. In addition, $L(x_k) \cdot 2^{-39}$ will be in the A upon exit from J276.

The elements $\{x_i\}$ of the vector x must be in the packed internal form recognized by the Floating Point system and must be placed in H.S.S. as follows:

$L(x_1)$, $L(x_2) = L(x_1) + \Delta$, ..., $L(x_n) = L(x_1) + (n-1)\Delta$, where Δ is an integer and $|\Delta| \leq 377_8 = 255_{10}$.

Calling Sequence:

α	020	α	010	β	(Link to J276)
α+1	σ00	n	$\|\Delta\|$	$L(x_1)$	
α+2	Control returns here				

$$\Delta \equiv \Delta[L(x_i)] = \begin{cases} |\Delta| \text{ if } \quad = 0 \\ -|\Delta| \text{ if } \quad = 1 \end{cases} .$$

Notes:

(1) By modifying the 18th word of J276, J276 can produce the maximum or minimum of the $\{x_i\}$ or $\{|x_i|\}$.

(2) J276 requires only the interpreter (J240E) of the Floating Point System. The first word of the interpreter (J240E000 or 5500_8) must coincide with F000.

*J276 is J195 or modified for the revised JOHNNIAC Floating Point System

Region A (erasable): 5 words (A0 - A4)

Symbolic Code: 25 words

John Derr

J277E* SOLUTION OF A MATRIX EQUATION AX = B WHERE A IS n x n AND NON-SINGULAR -- FLOATING POINT SYSTEM

A link to J277 will result in the solution of the k systems of n simultaneous linear equations

$$\sum_{j=1}^{n} a_{ij}\, x_j^{(\ell)} = b_i^{(\ell)},\ i = 1, 2, \ldots, n,$$

in n unknowns for $\ell = 1, \ldots, k$. If we define the matrices A, X, B by

$$A = \begin{pmatrix} a_{11} & a_{12} & \cdots & a_{1n} \\ a_{21} & & & \vdots \\ \vdots & & & \vdots \\ a_{n1} & \cdot & \cdot\ \cdot & a_{nn} \end{pmatrix},\ X = \begin{pmatrix} x_1^{(1)} & x_1^{(2)} & \cdots & x_1^{(k)} \\ x_2^{(1)} & & & \vdots \\ \vdots & & & \vdots \\ x_n^{(1)} & & \cdots & x_n^{(k)} \end{pmatrix},\ B = \begin{pmatrix} b_1^{(1)} & b_1^{(2)} & \cdots & b_1^{(k)} \\ b_2^{(1)} & & & \vdots \\ \vdots & & & \vdots \\ b_n^{(1)} & & \cdots & b_n^{(k)} \end{pmatrix}$$

then the two statements concerning the function of J277 are equivalent. J277 also computes the determinant of A which we denote as det A.

<u>Calling</u> <u>Sequence</u>:

α	020	α	010	β	(Link to J277)
$\alpha+1$	k	n	Δ	$L(a_{11})$	
$\alpha+2$	Control returns here if det A = 0				
$\alpha+3$	Control returns here if det A $\neq$ 0				

Before linking into J196 the user must place the elements a_{ij} of A and $b_i^{(\ell)}$ of B into H.S.S. locations as follows:

$$\begin{array}{llllllll} a_{11} & a_{12} & \cdots & a_{1n} & b_1^{(1)} & b_1^{(2)} & \cdots & b_1^{(k)} \\ a_{21} & a_{22} & & & & & & \\ \vdots & & & & & & & \\ a_{n1} & a_{n2} & \cdots & a_{nn} & b_n^{(1)} & b_n^{(2)} & \cdots & b_n^{(k)} \end{array}$$

*J277 is J196 as modified for the revised JOHNNIAC Floating Point System.

Basically, these locations differ by Δ. More precisely,

$$\text{loc } (a_{i,j+1}) = \text{loc } (a_{ij}) + ,, \quad j=1, 2, \ldots, n-1,$$

$$\text{loc } (a_{i,1}) = \text{loc } (b_{i-1}{}^{(k)}) + 2\Delta, \; i \neq 1,$$

$$\text{loc } (b_i{}^{(1)}) = \text{loc } (a_{i,n}) + \Delta, \text{ and}$$

$$\text{loc } (b_i{}^{(\ell+1)}) = \text{loc } (b_i{}^{(\ell)}) + \Delta, \; \ell=1, 2, \ldots, k-1.$$

Upon exit from J277 these same H.S.S. locations will contain

$$\begin{array}{llllllll}
\alpha_{11} & \alpha_{12} & \cdots & \alpha_{1n} & x_1{}^{(1)} & x_1{}^{(2)} & \cdots\ x_1{}^{(k)} & c_1 \\
\alpha_{21} & \alpha_{22} & \cdots & & & & & \vdots \\
\vdots & & & & & & & \\
\alpha_{n1} & \alpha_{n2} & \cdots & \alpha_{nn} & x_n{}^{(1)} & x_n{}^{(2)} & \cdots\ x_n{}^{(k)} & c_n
\end{array}$$

In addition, det A will be in A 19. All of the numbers referred to are in the packed internal form recognized by the Floating-Point Interpretive System. The quantities k, n, and Δ, in the Calling Sequence are positive integers with the obvious upper bounds $(77)_8$, $(7777)_8$, and $(377)_8$, respectively. Clearly, these are not least upper bounds.

The matrix $\alpha = (\alpha_{ij})$ is the n x n matrix obtained from A by the forward solution part of the elimination process. The n x 1 matrices $(x_1{}^{(\ell)}, x_2{}^{(\ell)}, \ldots, x_n{}^{(\ell)})$ are obtained in the back solution part of the method. The n x 1 matrix $(c_1, c_2, \ldots c_n)$ should be ignored by the user. It functions only as a check on the computational accuracy of the forward and backward solutions.

Description of the Method

The basic method used is the Gauss elimination method. In the forward solution the matrix A is reduced to a triangular matrix T. The solution of Tx = Eb is equivalent to the solution of the original equation $Ax = b$, where $E = \pi_i E_i$ and the E_i are elementary matrices, i.e. $E_i M$ results in performing elementary row operations on M. The back solution can be performed by a simple substitution usingthe equation $Tx = Eb$.

The rule of formation of the sequence $(E_1, E_2, \ldots)$ is known as Crout's Method*. However, here the method is modified so as to permute the rows of the resulting matrix, if necessary, to use the element $\alpha_{ki} = \underset{i \leq j \leq n}{\text{Max}} \ (|a_{ji}|)$ as the α_{ii} pivot element for $i = 1, 2, \ldots, n$.

The computation of the ith row of T is checked versus $(i+1)^2 \cdot 10^{-6}$ according to the Crout criterion for checking. Similarly, the calculation of the ith row of X is checked versus $(n+1)^2 \cdot 10^{-6}$. This method of checking will probably suffice in most cases w. r. t. testing for machine malfunctions. Note that this check does not guarantee that the solution computed is "correct" in any normal sense, since the ultimate accuracy of the computed solution is limited by the inherent error of the original system of equations. An exact specification of bounds for inherent error is beyond the scope of J277. However, loss of significant digits due to ill-conditioning of the original matrix A can cause the checking criterion not to be satisfied.

NOTES

(i) J271-2, J175, J276 (or J174 in place of J276 if all calculations are performed in the N mode) and the interpreter (J240E or J239F) of the Floating Point System are slave routines

*See Numerical Calculus, pp. 17-29, by W. E. Milne

w.r.t. J277 and therefore must be used in conjunction with J277. Further, J271-2, J175, J276 (or J174) and the interpreter have the origins WO, XO, ZO, YO, and FO respectively, w.r.t. J277. Hence, these regions must be assigned locations before J277 is loaded.

(ii) A 13X halt will occur in the 120th word of J277 if the computational checking criterion is not met in any row during the forward solution.. The calculation will continue if the GO button is pressed.

(iii) A 13X halt will occur in the 183rd word of J277 if the computational checking criterion is not met in any row during the back solution. The calculation will continue if the GO button is pressed.

(iv) A 13X halt will occur in the 75th word of J277 if det $A=0$. Control will link back to the main program if the GO button is pressed.

(v) Clearly, the original matrices A and B are destroyed by J277.

(vi) The relation

$$n(n+k+1) \leq (4096-F) - 292 - 26 - M$$

must always be satisfied when using J277, where $F =$ the number of words used by the Floating-Point Interpretive System and $M =$ the number of words in H.S.S. occupied by programs other than J271-2, J175, J276 and the Floating-Point System.

In order to get some real-live numbers, suppose $(4096 - F) = 2880$, $\Delta = 1$, and $M = 115$. Then $n(n+k+1) \leq 2447$ must hold. If $k=1$, $n \leq 48$. If $k=n$, $k=n \leq 34$.

Timing

Since at the time of this writing no explicit times were known for the floating-point operations, it follows that no explicit timing estimates could be made for the general case. However, the following floating-point operation counts should yield a reasonable measure of how the time will vary as a function of n and k.

Operation Counts for the Forward Solution

($\div$) $\frac{n(n+2k+1)}{2}$

(+) and (x) $\frac{n^3}{3} + \frac{k\,n^2}{2}$ for each (approximate for large n or k)

Operation Counts for the Backward Solution

(+) and (x) $\frac{n(n-1)(k+1)}{2}$ for each

J277 Storage (without slaves)

Reg. A (erasable) 20 words (A 6 - A 25)

Symbolic Code: 192 words

Storage Used by Slave Routines (Interpreter excepted)

Region A 6 words (A 0 - A 5) (Note that the erasable storage used by J271-2, J175 and J276 (or J174) does not overlap that used by J277.)

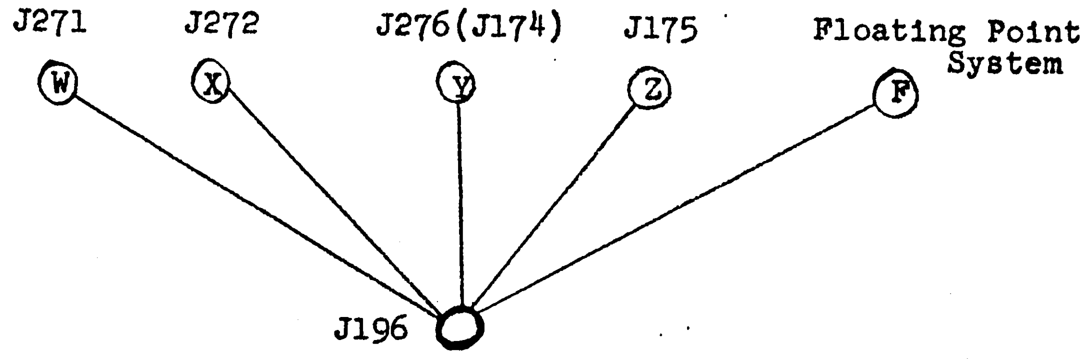
Symbolic Code: 100 words (or 98 words if J174 is used)

Combined Storage Used (Interpreter excepted)

Reg. A 26 words (A 0 - A 25)

Symbolic Code: 292 words (or 290 words if J174 is used)

Lattice Diagram Describing Master-Slave Relations



J278E* EVALUATION OF A ROOT OF A POLYNOMIAL WITH COMPLEX COEFFICIENTS -- FLOATING POINT SYSTEM

J278E is a more flexible version of J268E. J278 will compute, by an iterative process, up to n roots of an nth degree polynomial equation of the form

$$f(z) = \sum_{i=0}^{n} C_i z^i = 0, \text{ where } C_i$$

is a complex number for $i=0, \ldots, n$. Along with each root Ω computed, the multiplicity k of Ω, $|g(\Omega)|$ and the new reduced polynomial g(z) are available to the user. The user may, for each root Ω, constrain the convergence of the process by specifying an initial guess z_0 and by modifying certain sensitive quantities.

J278 must be used in conjunction with the current version of the Floating-Point System. J278 makes references to the interpreter (J240E or J239F) as region F. The main entry (polynomial entry) is performed by basic linkage and results in normalizing the polynomial f(z) into monic form, i.e.

$$g(z) = \sum_{i=0}^{N} a_i z^{N-i}, \text{ where } a_0 = 1 = \frac{C_N}{C_N} \text{ and } C_{N+1} = C_{N+2} = \ldots$$

$= C_n = 0$. In all succeeding re-entries (root entries), an attempt is made to compute a new root of the reduced polynomial.

Calling Sequence for Polynomial Entry:

```
        α    020    α    010    β    (polynomial entry to 1st
                                       word of J278)
```

*J278 is an improved version of J210 as modified for the revised JOHNNIAC Floating Point System.

$\alpha+1$	a00 $L(C_n)$ $\lvert\Delta\rvert$ n	
$\alpha+2$	Error return	(Control returns here if N=0 for the polynomial entry or if the process does not converge for a root reentry.)
$\alpha+3$	First Root return	(Control returns here following polynomial entry.)
$\alpha+4$	Intermediate root return	(Control returns here following the evaluation of a root.)

Before the polynomial entry to J278 is executed, the coefficients $\{C_i\}$ of f(z) must be stored in H.S.S. locations as follows, where $\Delta = \begin{cases} \lvert\Delta\rvert & \text{if } a = 0 \\ -\lvert\Delta\rvert & \text{if } a = 1 \end{cases}$ (Note that $\lvert\Delta\rvert \geq 2$):

$L(C_n)$—C_n^R = real component of C_n in floating pt. internal form

$L(C_n)+1$—C_n^I = imag. component of C_n in floating pt. internal form

$\vdots$

$L(C_n)+n\cdot\Delta$—C_0^R = real component of C_0 in floating pt. internal form

$L(C_n)+n\cdot\Delta+1$—C_0^I = imag. component of C_0 in fl. pt. internal form

The following locations contain the current values of quantities which are of interest to the user and to J278:

B00 - z_i^R = real component of the current approximation to Ω.

B01 - z_i^I = imag. component of the current approximation to Ω.

B02 - $\lvert g(\Omega)\rvert$, where Ω is the last root evaluated.

B03 - k_i = current estimate of multiplicity of z_i.

B04 - N = degree of g(z)

B05 - (n-N) = number of leading zero coefficients of f(z).

B10 + 6j (j=0,...,N-1) - a_{1+j}^{R} = real comp. of the coefficient of z^{N-1-j} in g(z).

B11 + 6j (j=0,...,N-1) - a_{1+j}^{I} = imag. comp. of the coefficient of z^{N-1-j} in g(z).

J278E283 (*88) - I(I=17 unless changed by user). The maximum number of iterations in the first cycle is I + N/2 (mod 4).

J278E284 (*89) - C = maximum number of iteration cycles (C=7 unless changed by user)

J278E285 (*96) - ϵ(ϵ=10^{-7} unless changed by user)

J278E286 (*97) - η(η=10^{-3} unless changed by user)

J278E289 (*94) - δ in floating-point (δ=10^{-2} unless changed by user)

J278E290 (*95) - δ in fixed point scaled 2^0 (δ=10^{-2} unless changed by user)

Root Entry (Re-entry):

All root entries (re-entries) to J278 are made by transferring to J278E046. If the root return is made to the right of J278E046, then J278 will use the contents of B0 and B1 for the initial guess, z_0, to Ω. Otherwise, J278 will set $z_0 = \sqrt[N]{|a_N|} + i \sqrt[N]{|a_N|}$. Prior to any root re-entry to J278, the user is free to alter z_0, I, C, ϵ, η, and δ as he sees fit. Upon return of control to α+2, α+3, or α+4 the user will probably want to utilize some of the quantities z_1, $|g(\Omega)|$, k, N, n-N, and the $\{a_1\}$. If control returns to the Intermediate root return, the user may assume that $\Omega = z_1$, k_1 = the multiplicity of Ω and N is the degree and the $\{a_1\}$ are the coefficients of the new reduced polynomial $g(z)$. It is important to observe that the user must program the decision to terminate the computation of roots of $f(z)$. This can usually be done by testing $N=0$ for every intermediate root return if all of the roots of $f(z)$ are desired.

Description of the Method

Set $q_0(z) = g(z)$ and let $q_j(z)$ be defined by $q_{j-1}(z) = (z-z_1)q_j(z) + q_{j-1}(z_1)$ for $j=1,2,\ldots,N$. Define ℓ_1 to be the smallest non-negative integer such that

(i) $|q_{\ell_1+1}(z_1)| \geq \eta\, |q_{\ell_1+1}(0)|$ and

(ii) $\dfrac{|q_{\ell_1-1}(z_1)|}{|q_{\ell_1-1}(0)|} \leq \ell_1 \dfrac{|q_{\ell_1}(z_1)|}{|q_{\ell_1}(0)|}$, whenever $\ell_1 > 0$,

if such a number exists.* Otherwise, replace z_1 by $z_1 + \dfrac{(\Delta z)_{i-1}}{2}$ and repeat the decision process until an ℓ_1 satisfies (i) and (ii) for some z_1.

We next define k_1 to be 1 or $\ell_1+\hat{x}-1$ depending on whether ℓ_1 is respectively zero or positive. To obtain $\hat{x}$ first compute

$$x = \frac{g^{(\ell_1)}(z_1)/g^{(\ell_1+1)}(z_1)}{g^{(\ell_1)}(z_1)/g^{(\ell_1+1)}(z_1) - g^{(\ell_1-1)}/g^{(\ell_1)}(z_1)},$$

provided the right hand side of the above equation exists. Then

(i) Set $\hat{x} = j$ if x is well-defined and there exists an integer j, $2 \leq j \leq N-\ell_1+1$, such that $|x-j| < \delta$.**

---

*In J268E, $=10^{-2}$.
**The δ used here is 10^{-1}in J268.

(ii) Set $\hat{x} = 2$ otherwise (in this event $k_i = \ell_i + 1$).

Finally, the basic iterative process consists of recursively computing z_{i+1} from z_i by means of the relation

$$z_{i+1} = z_i - (\Delta z)_i \text{ , where } (\Delta z)_i = \frac{(k_i - \ell_i) g^{(\ell_i)}(z_i)}{g^{(\ell_i+1)}(z_i)} \; .$$

The basic iterative process is modified * when $\ell_i = 0$ and with x now defined by

$$x = \frac{(\Delta x)_{i-1}}{(\Delta x)_{i-1} - (\Delta x)_i} \text{ , when}$$

there exists an integer j, $2 \leq j \leq N$, such that $|x-j| < \delta$. If such is the case set $k_i = j$ and $k_i - \ell_i = j-1$. With this modification multiple roots can be detected prematurely.

<u>Convergence and the Choices of z_0</u>

The criterion for convergence of z_i to Ω requires that $|(\Delta z)_i| \leq |z_i| \cdot \epsilon$, i.e. that the relative error in Ω shall not exceed ϵ. $|g(\Omega)|$ provides an independent check on the computation. Note though that J278 does not substitute Ω back into the <u>original</u> polynomial $f(z)$.

The system which J278 uses for starting the iterative process breaks down into C iteration cycles. The first iteration cycle allows a maximum of I+N/2 (mod 4) iterations and this maximum is increased by 4 for each succeeding iteration cycle.

*In J268, this modification does not exist.

On the first cycle z_0 is chosen as described above. On successive cycles z_0 is set equal to a complex number with modulus $\doteq \sqrt[N]{a_N}$ according to a fixed pattern, where a_N is the constant coefficient of g(z). The set of values of z_0 includes points in all four quadrants.

Any cycle can be interrupted by convergence. If all C cycles are executed in their entirety, then the process has not converged and control will return to the Error return.

NOTES

(1) If f(z) has real coefficients, and if Ω is a proper complex root, then set $z_0 = \Omega^*$ (the conjugate of Ω), provided Ω^* has not already been evaluated.

(2) In general, do not set $z_0 =$ a real number unless only real roots are desired.

(3) In general, ϵ should not be less than 10^{-8}, since 9 digit arithmetic is used.

(4) In general, as η is increased, the process will tend to produce more multiple roots. For example, as η is increased, distinct roots (as well as true multiple roots) are more likely to be identified as multiple roots. Conversely, as η is decreased, multiple roots (as well as distinct roots) will tend more to be identified as sets of distinct roots.

(5) Essentially, the statement of (4) for η holds for δ as well.

(6) The values of the roots, as well as their multiplicities, can vary significantly with the order in which they are determined.

(7) The reader is referred to the J268 write-up for additional information of a more general nature.

(8) J269 and J270 and the interpreter (J240E or J239F) of the Floating Point System are slave routines w.r.t. J278 and therefore <u>must</u> be used in conjunction with J278. Further, J269, J270, and the interpreter have the origins YO, ZO, and FO respectively, w.r.t. J278. Hence, these regions must be assigned locations before J278 is loaded. The first word of the interpreter (J240E000 or 5500_8) must coincide with F000.

(9) If the process does not converge, B0 and B1 contain the last approximation to a root Ω. Also N and the $\{a_{1+j}\}$ suffice to describe g(z).

(10) J278E requires only the Exponential (J248E) and Logarithm (J249E) programs of the JOHNNIAC Floating Point System in addition to the interpreter.

(11) J278E can't be loaded by X267 without modification since too many forward references occur.

<u>J278 Storage (without slaves)</u>

Region A (erasable) 27 words (A00-A26)

Region B (semi-erasable) 10+6N words (B0 - B9 + 6N)

Symbolic Code: 305 words

<u>Storage Used by Slave Routines (J269E and J270E)</u>

Region A (erasable) 7 words (A0 - A6)

Symbolic Code: 27 words

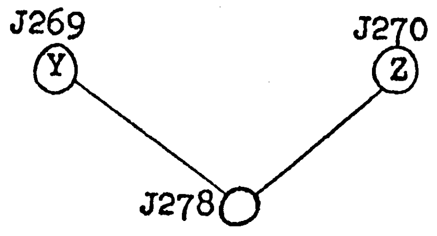
<u>Combined Storage Used by J278, J269 and J270</u>

Region A (erasable) 27 words (A00 - A26)

Region B (semi-erasable) 10 + 6N words (B0 - B9 + 6N)

Symbolic Code: 332 words

<u>Lattice Diagram Describing Master-Slave Routines</u>



John Derr

J279F AUTOMATIC COMPUTATION OF THE ZEROS OF A POLYNOMIAL WITH COMPLEX COEFFICIENTS -- FLOATING POINT SYSTEM

J279 makes use of J278 to compute the roots of polynomials. The roots of the polynomials will be computed and printed one at a time.

Formation of Input Deck:

1. blank card
2. J140A (1 card)
3. J135A (1 card)
4. J239F or current version of the Floating Point System
5. J279F
6. Data
7. 2 blank cards

Input Form - Data:

Information should be punched in the Floating Point Data Form as follows:

Header card for a problem

Pos (1)	S	+ (blank) if all coefficients are real - if any coefficient is not real
	EXP	m = the maximum number of distinct roots to be evaluated (usually m=n). If m=0, then m will be set = n.
	MANTISSA	ID (Iff ID=0, the coefficients of f(z) will be printed.)
Pos (2)	S	Iff -, set $\mathcal{S}^I=0$ when $\|\mathcal{S}^I\| < \epsilon \|\mathcal{S}^R\|$.
	EXP	Iff 0, the final reduced polynomial g(z) will be printed.

MANTISSA n = degree of $f(z)$.

Pos (3) z_0^R }
Pos (4) z_0^I } Iff $z_0 \neq 0$, the starting value for the first iteration cycle will be z_0.

Pos (5) Iff $\epsilon \neq 0$, then ϵ will replace the $\epsilon(10^{-7})$ in J278.

Pos (6) Iff $\eta \neq 0$, then η will replace the $\eta(10^{-3})$ in J278.

Coefficient cards for a problem

The coefficients are punched 6/card into consecutive positions of the cards beginning with position (1) of the next card after the header card. If the sign in position (1) of the header card is + (blank), then only the real components of the coefficients are to be punched and these must be in consecutive positions (no gaps). Otherwise, the real and imaginary components of each coefficient must be punched. The last card for each problem must contain an EF mark (12 punch (+) in col. 80).

A sequence of problems

The roots for a sequence of polynomials can be evaluated by placing the input cards for the polynomials in sequential order behind J279F. If the input cards are followed by two blank cards, the console will show READER upon completion of the last problem or a hang up condition will occur.

Output Form

The page will be ejected prior to the beginning of the first problem. The numbers printed will be in the external Floating Point form. The print format is that described for the 017 (PNT) order in the Floating-Point manual.

Printing for one problem

	x_1	x_2	x_3	x_4	x_5	x_6
Information in header card	S EXP MAN	S EXP MAN	z_0^R	z_0^I	ϵ	η
	$\pm$ m ID	$\pm$ 0/1 n				

If ID = 0, the printer will be spaced one line. Then the coefficients will be printed 8/line, if the sign position of the first word of the header card is +, and pairs of real and imaginary coefficients will be printed in positions (x_1, x_2), (x_4, x_5), and (x_7, x_8), otherwise.

	x_1	x_2	x_3	x_4	x_5	x_6
Space	———	———		———		
Roots begin here						
⋮	Ω_1^R	Ω_1^I	———	$\lvert g(\Omega_1)\rvert$	———	k
	⋮	⋮		⋮		
(pth) last root computed	Ω_p^R	Ω_p^I	———	$\lvert g(\Omega_p)\rvert$	———	k
Space	———	———		———		
Space	———	———		———		

If 0/1 = 0, then the coefficients $\{a_i\}$ of g(z) will be printed as follows:

	x_1	x_2	x_3	x_4
Coefficient of z^N	a_0^R	a_0^I	———	N
⋮	⋮	⋮		⋮
Constant coefficient	a_N^R	a_N^I		0

The printer is spaced four lines. If there is another problem, the header card will be read and printed and the above process for printing will be repeated.

NOTES

(1) If n exceeds 450, then following the printing of the contents of the header card and the printing (if any) of the coefficients of f(z), J279 will begin the next problem (if any).

(2) $p \leq \min(m,n)$ and $N \leq n - \sum_{j=1}^{p} k_j$, where here N denotes the degree of the last reduced polynomial g(z).

(3) If f(z) has real coefficients, $\Omega^I \neq 0$, and $|\Omega^I| \geq |\Omega^R| \cdot \epsilon$, then J279 will set $z_0 = \Omega^R - \Omega^I$, provided $\Omega^R - \Omega^I$ has not been evaluated previously.

(4) If the process does not converge, then the printer will be spaced twice, z_1^R, z_1^I, and $|g(z_1)|$ will be printed on one line, and the printer will be spaced twice again. Then if 0/1 = 0, the coefficients of g(z) will be printed as described above. In any event, J279 will then begin the next problem (if any).

(5) The following approximate times were recorded while operating in the SD mode:

$z^3 - 1$ required 9 sec.	$z^5 - 1$ required 36 sec.
$z^9 - 1$ required 76 sec.	$z^{17} - 1$ required 401 sec.

(6) The user is referred to the J278 write-up for a description of the method and the definitions of quantities not defined in this writeup.

(7) The computation for any one problem can be interrupted by manually transferring to 5116_8. The result will be the same as in (4). Note that the coefficients of g(z) will be printed only if 0/1 = 0.

(8) J278 has been modified by placing breakpoint tracing marks in certain key words. Thus, if T2 is on, the following information will be printed:

4243	The real part of $(\Delta z)_{i-1}$ is divided by -2 and replaces $(\Delta z)_i$.
4244	The imaginary part of $(\Delta z)_{i-1}$ is divided by -2.
4273	2 is subtracted from the real part of x.
4275	The modulus of the imaginary part of x is divided by the real part of x.
4335	The real part of $(\Delta z)_i$ is multiplied by $k_i - \ell_i$.
4337	The imaginary part of $(\Delta z)_i$ is multiplied by $k_i - \ell_i$.
4340	The real part of $(\Delta z)_i$ is subtracted from the real part of z_i.
4342	The imaginary part of $(\Delta z)_i$ is subtracted from the imaginary part of z_i.

Program available as J279F--53 cards (J79F001 - J279F053).

John Derr

J280E CAT Punch with Echo Checking

Calling Sequence:

$\alpha - k$	100	3		
⋮	⋮			
α	020	α	010	β
$\alpha + 1$	Normal Return			

J280 punches the CAT image* from B1 thru B24, then transfers the image to B25 thru B48 where it is used for Echo checking. Note that B25 thru B48 must be clear before the <u>first</u> usage and must <u>not</u> be disturbed until the punch is cleared. Also note that a card other than a blank (in the punch) will cause an echo failure at the time of first usage.

In the event of an echo failure, one blank card is fed, and the failed card is re-punched. The machine will then stop at the right of the origin of J280 + 23. At this time the penultimate card in the stacker is the failed card. Remove the last two cards from the stacker, hit GO, and the repunched card will be echo checked and normal operation will continue until the next failure. Note that the first card into the stacker after hitting GO will be a blank.

---

* CAT card image is as follows:

B 1	C_{1-40}	12's row
B 2	C_{1-40}	11's row
⋮	⋮	⋮
B 12	C_{1-40}	9's row
B 13	C_{41-80}	12's row
⋮	⋮	⋮
B 24	C_{41-80}	9's row

Timing: There are approximately 39 ms. available for computing between the 100 3 instruction and the entry to this routine.

Program length: 34 words

Highest symbol used: * 18

Region B: B1-B48

C. P. Martin
10-21-58

J281E   Ordinary Least Squares -- Floating Point System

The program computes the best possible polynomial approximating a set of data in the sense of least squares. The data consist of number pairs (x_i, y_i) for $i = 1, 2, \ldots, N$ which determine a relation (x,y). The program approximates (x,y) by a polynomial

$$y^*(x) = \sum_{j=0}^{M} a_j x^j = \sum_{0}^{M} b_j (x-m)^j ,$$

where $1 \leq M \leq \min(10, N-1)$ and $m = \sum_{1}^{N} x_i/N$. This program must be used in conjunction with the interpreter of the Floating Point System. The interpreter is referred to as region F.

<u>Calling Sequence</u> (<u>x_i entry</u>):

α	020	α	010	J281E001
$\alpha+1$	M	N	Δ_x	$L(x_1)$
$\alpha+2$	Error Return - ($1 \leq M \leq \min(10, N-1)$ is not satisfied or 13X in J277)			
$\alpha+3$	Control returns here.			

<u>Calling Sequence</u> (<u>y_i entry</u>):

α	020	α	010	J281E104
$\alpha+1$		$L(b_0)$	Δ_y	$L(y_1)$
$\alpha+2$	Control returns here			

<u>Application</u>:

The employment of a double entry is motivated by the case where many sets of $\{y_i\}$ correspond to one set of $\{x_i\}$. The locations of the $\{x_i\}$ and $\{y_i\}$ are equally spaced; i.e.

$L(x_{i+1}) = L(x_i) + \Delta_x$ for $1 \leq i \leq N-1$, where $\Delta_x > 0$, and similarly for $\{y_i\}$ and Δ_y. An x_i entry followed by a y_i entry yields $b_0, b_1, \ldots, b_M$. Following an x_i entry $-m$ will be in A026. A link to J275E with $L(c)$ = A026 will yield $a_0, a_1, \ldots, a_M$.

Method:

The x_i entry produces the following results:

(i) Compute m.

(ii) Form the normal matrix A for $\{x_i'\}$, where $x_i' = x_i - m$ for $i=1, 2, \ldots, N$, i.e.

$$A = \begin{pmatrix} N & 0 & \sum_1^N (x_i')^2 & \cdots & \sum_1^N (x_i')^M \\ 0 & \sum_1^N (x_i')^2 & & \cdots & \vdots \\ \sum_1^N (x_i')^2 & & & \cdots & \\ \vdots & & & & \\ \sum_1^N (x_i')^M & & & \cdots & \sum_1^N (x_i')^{2M} \end{pmatrix}$$

(iii) Compute A^{-1}.

The y_i entry produces the following results:

(i) Form the vector $y = \begin{pmatrix} \sum_1^N y_i \\ \sum_1^N x_i' y_i \\ \vdots \\ \sum_1^N (x_i')^M y_i \end{pmatrix}$

(ii) Compute the vector $b = A^{-1}y = \begin{Bmatrix} b_0 \\ b_1 \\ \vdots \\ b_m \end{Bmatrix}$

Notes:

(1) J271, J272, J175, and J276 (or J174) are slaves to J277 and hence must be assigned locations prior to the loading of J277. Further, J273, in addition to J277 and the interpreter, are slaves to J281 and must be assigned locations prior to the loading of J281.

(2) For a 13X halt which occurs in J277 see the J277 writeup.

(3) See the J282 writeup for some timing and accuracy samples.

J281 Storage

Region A (eraseable) : A000 - 4N + A283

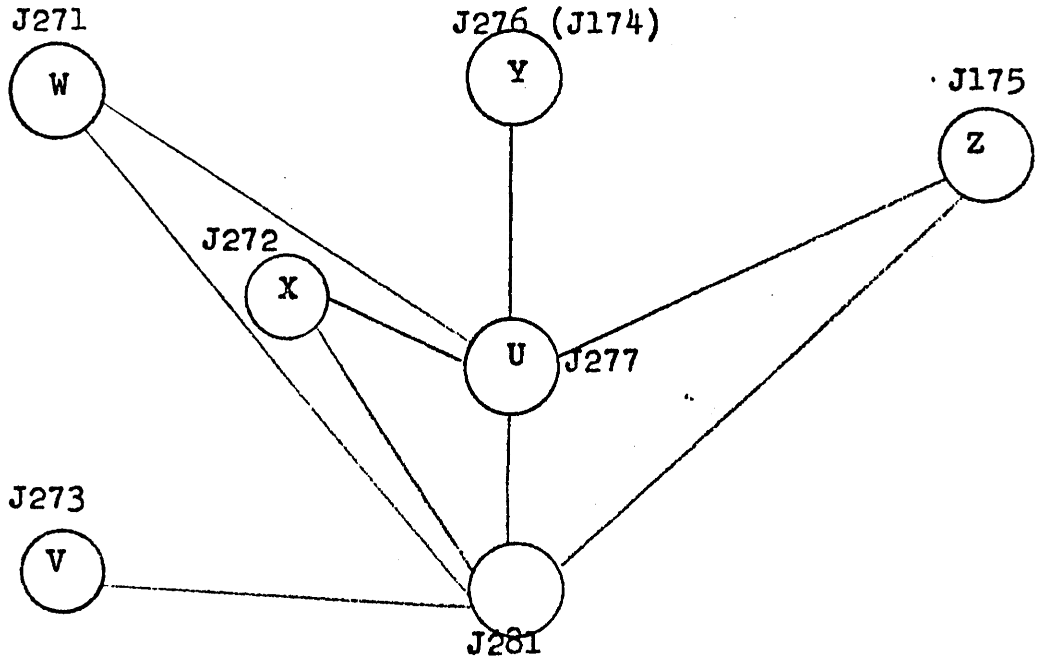
Program : 153 words (J281E001 - J281E153)

Total Storage Including Slaves - Interpreter Excepted

Region A : A000 - 4N + A283

Program : 510 words

Lattice Diagram Describing Master-Slave Relations



John Derr

J282F Ordinary Least Squares -- Floating Point System

J282F is an absolute binary assembly of J281E together with J175E, J271E - J273E, J275E - J277E and a program which loads, computes and prints in a manner to be described. To use J282F form a deck consisting of

(i)	one blank card
(ii)	J140A
(iii)	J135A
(iv)	J239F
(v)	J282F
(vi)	data
(vii)	2 blank cards.

Header Card Format - 013 style input:

pos	(1)		ID
pos	(2)	S	blank or +
		EXP	blank or zero
		MANTISSA	M
pos	(3)	S	+ if x_i cards follow; - if y_i cards follow
		EXP	blank or zero
		MANTISSA	N
pos	(4)	< 0	if printing of x_i, y_i, $y^*(x_i)$, r_i, and r_i^2 is to be suppressed.
		≥ 0	otherwise

pos (4) is examined only if the sign position (3) is – .

Data Card Format

If the sign of position (3) of a header card is + , then the N values of x_i will be loaded 6/card beginning with pos (1) of the next card. The card containing the Nth x_i must have an end of file mark in col. 80. Upon loading the latter card an x_i entry to J281E occurs. A similar statement holds for loading $\{y_i\}$ Following the

y_i entry a link to J275E yields $a_0, a_1, \ldots, a_M$.

Ordering of Data Cards

A set of number pairs (x_i, y_i) is specified by an x_i header card followed by the x_i cards and a y_i header card followed by the y_i cards. The x_i cards must precede the y_i cards and the values of M and N on the corresponding header cards must agree. Many sets of y_i cards can follow a set of x_i cards.

Printing

The page is ejected following the reading of each x_i header card. The printer is spaced one line following the reading of each y_i header card. In both cases the contents of the first four positions of the header card is printed immediately thereafter. Following the calculation of $a_0, a_1, \ldots, a_M$ the printer is spaced twice and $a_0, a_1, \ldots, a_M$ are printed 8/line. Then if position (4) of the y_i header card is non-negative, the program computes $\sum_{j=0}^{M} a_j(x_i)^j = y^*(x_i)$, $y^*(x_i) - y_i = r_i$ and $\sum_{j=1}^{i} r_j^2$, for i = 1, ..., N.

An illustrative printing layout follows:

	x_1	x_2	x_3	x_4	x_5
	ID	+ 00 M	+ 00 N	+ 00 ——	
one space					
	ID	+ 00 M	− 00 N	+ 00 ——	
two spaces					
	a_0	a_1	a_2	a_3	a_4 ...
two spaces					
	x_1	y_1	$y^*(x_1)$	r_1	r_1^2
	x_2	y_2	$y^*(x_2)$	r_2	$r_1^2 + r_2^2$
	$\vdots$	$\vdots$	$\vdots$	$\vdots$	$\vdots$

Notes:

(1) $N \leq 54$ has been artificially imposed.

(2) In the event of a 13X halt consult the J277E and J281E writeups.

Timing and Accuracy:

The Legendre polynomials on $[0,1]$ with N = 21 points, 0 (.05) 1, were used as test problems.

M	Time for x_i entry	Time for y_i entry	Accuracy
1	3 secs.	1 secs.	8.5 SD
2	5 "	2 "	8.8 "
3	8 "	3 "	7.9 "
4	12 "	4 "	5.5 "
5	17 "	5 "	5.5 "

The above times are exclusive of loading and printing times. The accuracy refers to the number of correct SD in the least accurate of the coefficients.

Program Available in style F - 53 cards (J282F001 - J282F053)

John Derr

29 October 1958

J283E[*] BACK AND FORTH BLOCK TRANSFER

Let $x = (x_1,\ldots,x_n)$ and $y = (y_1,\ldots,y_n)$ be two n-tuples with real components. Upon entry to J283 the following pair of instruction words will be executed for $i = 1,2,\ldots,n$:

J283E020	004	$L(x_i)$	020	$L(y_i)$
J283E021	050	$L(x_i)$	060	$L(y_i)$,

i.e. $y_i \rightarrow L(x_i)$ and $x_i \rightarrow L(y_i)$ is performed in the natural order $i = 1,2,\ldots,n$. It is clear that J283 can be modified to function as a standard block transfer routine by deleting the 050 or 060 operation in J283E021.

The vector x must be placed in H.S.S. as follows:
$L(x_1)$, $L(x_2) = L(x_1) + \Delta_x$, ..., $L(x_n) = L(x_1) + (n-1)\,\Delta_x$,
where Δ_x is an integer and $|\Delta_x| \leq 377_8$. A similar statement holds for y.

Calling Sequence:

α	020	α	010	β	(Link to J283)
$\alpha+1$	$\sigma_x 00$	n	$\lvert\Delta_x\rvert$	$L(x_1)$	
$\alpha+2$	$\sigma_y 00$		$\lvert\Delta_y\rvert$	$L(y_1)$	

In the above

$$\Delta_x = \begin{cases} |\Delta_x| & \text{if } \sigma_x = 0 \\ -|\Delta_x| & \text{if } \sigma_y = 1 \end{cases} \quad \text{and similarly for y.}$$

Storage

Region A: 5 words, A0 - A4

Symbolic Code: 28 words, J283E001 - J283E028

[*]J283 differs functionally from J175 only in the calling sequence.

John Derr

29 October 1958

J284E LEGENDRE POLYNOMIAL EVALUATION - FLOATING POINT SYSTEM

The program computes $p_k(x)$ for $k = 2,3,\ldots,M \leq 15$, where $p_o(x) = 1$, $p_1(x) = x$, and

$$p_{n+1}(x) = \frac{2n+1}{n+1} x\, p_n(x) - \frac{n}{n+1} p_{n-1}(x) \quad \text{for} \quad n \geq 1.$$

Prior to entering the program by basic linkage the user must place 1 in A12 and x in A13 and A35, both numbers being in floating point form. In addition, M+1 (scaled 2^{-39}) must be placed in A32.* Upon exit from the program A12+k will contain $p_k(x)$ for $k = 0,1,\ldots,M$. It is clear that these cells will contain $y \cdot p_k(x)$ if A12 and A13 contain y and yx, respectively.

Calling Sequence:

```
              α       020       α       010       J284E001
              α+1     Control returns here
```

The program makes references only to the interpreter of the floating point system and to that as region F. The user must cause F000 to coincide with the first word of the interpreter.

Storage:

Region A: A8, A9, A12 - A27, A31, A32, and A35

Program: 24 words (J284E001 - J284E024).

*Upon exit from J285, A32 will already contain M+1.

John Derr

29 October 1958

J285E LEGENDRE POLYNOMIAL APPROXIMATION -- FLOATING POINT SYSTEM

The program computes the least squares approximation to a set of data with respect to the Legendre polynomials. The data consist of number pairs (x_i, y_i) for $i = 1,2,\ldots,N$ which determine a relation (x,y). The program approximates (x,y) by a polynomial

$$p(x) = \sum_{j=0}^{M} b_j \, p_j(\overline{x}) = \sum_{j=0}^{M} a_j \, x^j ,$$

where $\overline{x} = (x-m)/(\max_i |x_i - m|)$, $1 \leq M \leq \min(15, N-1)$, $m = \sum_{1}^{N} x_i/N$,

$$p_o(x) = 1 \quad , \; p_1(x) = x, \text{ and}$$

$$p_{n+1}(x) = \frac{2n+1}{n+1} \, x \, p_n(x) - \frac{n}{n+1} \, p_{n-1}(x)$$

for $n \geq 1$. The $\{p_k\}$ satisfy

$$\int_{-1}^{+1} p_j(x) \, p_k(x) = \delta_{jk} \left(\frac{2}{2k+1}\right).$$

<u>Calling Sequence (x_i entry):</u>

```
α      020      α      010      J285E001
α+1     M       N      Δx        L(x1)
α+2    Error Return (13X in J277)
α+3    Control returns here
```

<u>Calling Sequence (y_i entry):</u>

```
α      020      α      010      J285E105
α+1           L(bo)    Δy        L(y1)
α+2     Control returns here
```

Application: The employment of a double entry is motivated by the case where many sets of $\{y_i\}$ correspond to one set of $\{x_i\}$. The location of the $\{x_i\}$ and $\{y_i\}$ are equally spaced; i.e. $L(x_{i+1}) = L(x_i) + \Delta_x$ for $1 \leq i \leq N-1$, where $\Delta_x > 0$, and similarly for the $\{y_i\}$ and Δ_y. An x_i entry followed by a y_i entry yields $b_o, b_1, \ldots, b_M$. Following an x_i entry m will be in A29 and $\max\limits_i |x_i - m|$ will be in A30.

$p(x)$ can be readily computed by the expression $\sum\limits_{j=0}^{M} b_j p_j(\overline{x})$ if the user links successively to J284 and J271. Alternatively, $a_o, a_1, \ldots, a_M$ can be obtained by successive links to J286, J287, and J275.

Method: The x_i entry produces the following results:

(i) Compute m and $\max\limits_i |x_i - m|$.

(ii) Form the normal matrix A with respect to the $\{p_j(\overline{x}_i)\}$, i.e. if $A = (a_{jk})$, $0 \leq j,k \leq M$, then $a_{jk} = \sum\limits_{i=1}^{N} p_j(\overline{x}_i)\, p_k(\overline{x}_i)$.

(iii) Compute A^{-1}.

The y_i entry produces the following results:

(i) Form the vector $y = \begin{pmatrix} \sum\limits_{i=1}^{N} p_o(\overline{x}_i) y_i \\ \vdots \\ \sum\limits_{i=1}^{N} p_M(\overline{x}_i) y_i \end{pmatrix}$.

(ii) Compute the vector $b = A^{-1}y = \begin{pmatrix} b_o \\ \vdots \\ b_M \end{pmatrix}$.

NOTES:

(1) J271, J272, J175, and J276 (or J174) are slaves to J277 and hance must be assigned locations prior to the loading of J277. Further, J273, J283, and J284, in addition to J277 and the interpreter, are slaves to J285 and must be assigned locations prior to its loading.

(2) The program makes references only to the interpreter of the floating point system and to that as region F. The user must cause F000 to coincide with the first word of the interpreter.

(3) For a 13X halt which occurs in J277 see the J277 writeup.

(4) See the J288 writeup for some timing and accuracy samples.

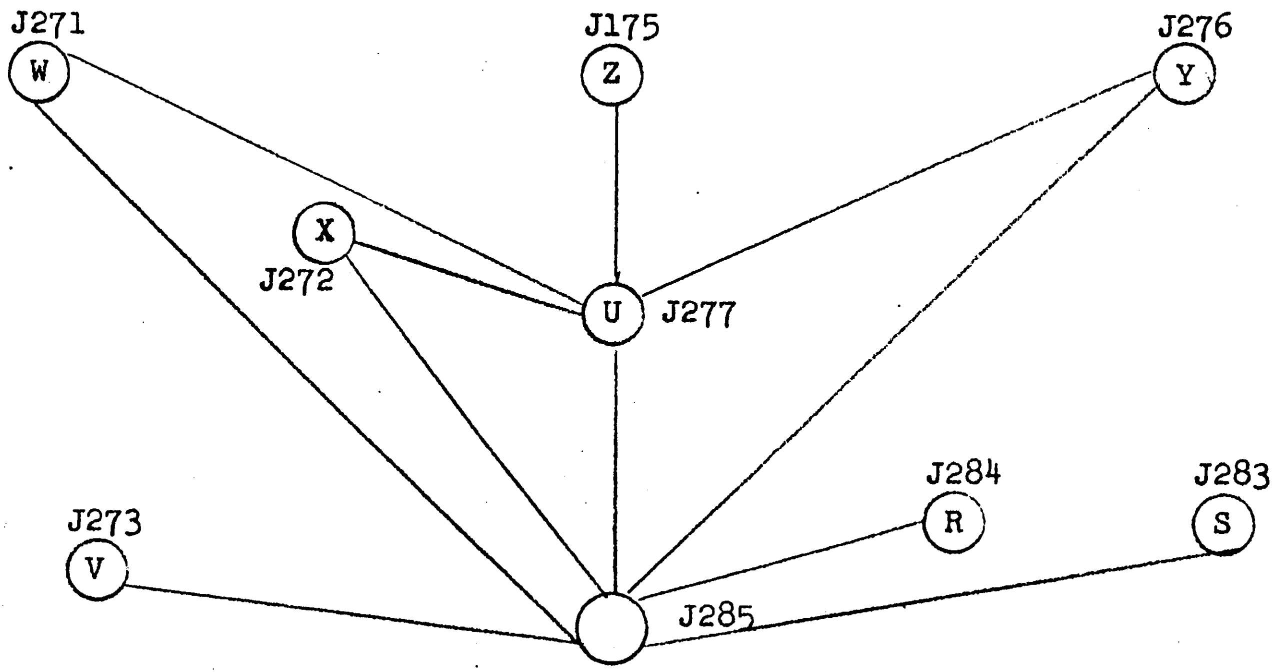
J285 Storage:

Region A:	A000 - A037 + $2(M+1)^2$
Program:	146 words (J285E001 - J285E146)

Total Storage Including Slaves - Interpreter Excepted

Region A:	A000 - A037 + $2(M+1)^2$
Program:	555 words

Lattice Diagram Describing Master - Slave Relations



29 October 1958

J286E LEGENDRE POLYNOMIAL COEFFICIENTS -- FLOATING POINT SYSTEM

An entry to this program will produce the coefficients p_{kj} of the Legendre polynomials p_k on $[-1,1]$ for $k=0,1,\ldots,M$; i.e. $p_0(x) = 1 = p_{00}$, $p_1(x) = x = p_{10}x^0 + p_{11}x$, and

$$p_{n+1}(x) = \frac{2n+1}{n+1}\, x\, p_n(x) - \frac{n}{n+1}\, p_{n-1}(x) = \sum_{j=0}^{n+1} p_{n+1,j}x^j \text{ for } n \geq 1.$$

<u>Calling Sequence</u>:

α	020	α	010	J286E001
$\alpha+1$		M		$L(p_{00})$
$\alpha+2$	Control returns here			

Prior to entering the program the user should clear cells $L(p_{00})$ to $L(p_{00}) + M(M+1)$, inclusive, to zero since only the non-zero p_{kj} are computed and stored. Upon exit from the program the p_{kj} will be in H.S.S. cells in floating point form as follows:

$$\begin{pmatrix}
p_{00} & p_{01} & p_{02} & \cdots & p_{0,M-1} & p_{0,M} \\
p_{11} & p_{12} & p_{13} & \cdots & p_{1,M} & - \\
\vdots & \vdots & \vdots & & \vdots & \vdots \\
p_{M-1,M-1} & p_{M-1,M} & - & \cdots & - & - \\
p_{M,M} & - & - & \cdots & - & -
\end{pmatrix}$$

The cells corresponding to — and to zero p_{kj} are never read into the program. The program makes references only to the interpreter of the floating point system and to that as region F. The user must cause F000 to coincide with the first word of the interpreter.

<u>Storage</u>:

Region A: 6 words, A0 - A5

Program: 62 words (J286E001 - J286E062).

John Derr

J287E (TRIANGULAR MATRIX) X VECTOR - FLOATING POINT SYSTEM

The program computes for $k = 0,1,\ldots,M$,

$$b_k = \sum_{j=k}^{M} a_j \, p_{kj}.$$

CALLING SEQUENCE:

α	020	α	010	J287E001
$\alpha+1$		$L(b_0)$	Δ	$L(a_0)$
$\alpha+2$		M		$L(p_{00})$
$\alpha+3$	Control returns here			

Here $L(a_{j+1}) = L(a_j) + \Delta$, $\Delta > 0$, $L(b_{j+1}) = L(b_j) + 1$, and the p_{kj} must be in consecutive H.S.S. cells as follows:

$$\begin{pmatrix} p_{00} & p_{01} & p_{02} & \cdots & p_{0,M-1} & p_{0,M} \\ p_{11} & p_{12} & p_{13} & \cdots & p_{1,M} & - \\ \vdots & \vdots & \vdots & & \vdots & \vdots \\ p_{M-1,M-1} & p_{M-1,M} & - & \cdots & - & - \\ p_{M,M} & - & - & \cdots & - & - \end{pmatrix}$$

The contents of those cells corresponding to a "—" are irrelevant. Although the storage assignment is unusual the set of (p_{kj}) do constitute an upper triangular matrix in which the elements of each row have been moved to the left until the elements on the principal diagonal are at the extreme left.

The program makes references only to the interpreter of the floating point system and to that as region F. The user must cause F000 to coincide with the first word of the interpreter.

J271E is a slave to the program and is referred to as region W. The user must cause W000 to coincide with J271E000.

<u>STORAGE</u>:

Program: 19 words (J287E001 - J287E019).

John Derr

J288F LEGENDRE POLYNOMIAL APPROXIMATION -- FLOATING POINT SYSTEM

This program is an absolute binary assembly of J285E together with J175, J271E - J273E, J275E - J277E, J283E, J284E, J286E, J287E, and a program which loads, computes and prints in a manner to be described. To use J288F form a deck consisting of

(i) one blank card
(ii) J140A
(iii) J135A
(iv) J239F
(v) J288F
(vi) data
(vii) 2 blank cards

<u>Header Card Format (013 style input)</u>:

pos (1)		ID
pos (2)	S	blank or +
	EXP	blank or zero
	MANTISSA	M
pos (3)	S	$\begin{cases} + \text{ if } x_i \text{ cards follow} \\ - \text{ if } y_i \text{ cards follow} \end{cases}$
	EXP	blank or zero
	MANTISSA	N

pos (4) $\begin{cases} < 0 \text{ if printing of } x_i, y_i, p(x_i), r_i, \text{ and } \sum r_i^2 \text{ is to be \underline{suppressed}.} \\ \geq 0 \text{ otherwise.} \end{cases}$

Position (4) is examined only if pos (3) < 0.

Data Card Format

If the sign of position (3) of a header card is +, then the N values of x_i will be loaded 6/card beginning with position (1) of the next card. The card containing the N^{th} x_i must have an end of file mark in column 80. Upon loading the latter an x_i entry to J285E occurs. A similar statement holds for loading y_i. Following the y_i entry successive links to J286E, J287E, and J275E yield $a_0,\ldots,a_M$.

Ordering of Data Cards

A set of data points (x_i,y_i) is defined by an x_i header card followed by the x_i cards and a y_i header card followed by the y_i cards. The x_i cards must precede the y_i cards and values of M and N on the corresponding header cards must agree. Many sets of y_i cards can follow a set of x_i cards.

Printing

The page is ejected following the reading of each x_i header card. The printer is spaced one line following the reading of each y_i header card. In both cases the contents of positions 1-4 of the header card is printed immediately thereafter. Following the calculation of $b_0,b_1,\ldots,b_M$ the printer is spaced twice and $b_0,b_1,\ldots,b_M$ are printed 8/line. Following the calculation of $a_0,\ldots,a_M$ the printer is spaced twice and $a_0,\ldots,a_M$ are printed 8/line. Then if position (4) of the y_i header card is non-negative, the program computes $p(x_i) = \sum_{j=0}^{M} b_j p_j(\overline{x}_i)$, $p(x_i)-y_i = r_i$, and

$\sum_{j=1}^{i} r_j^2$, and prints x_i, y_i, $p(x_i)$, r_i, and $\sum_{j=1}^{i} r_j^2$ for $i=1,\ldots,N$.

An illustrative printing layout follows:

x_1	x_2	x_3	x_4	x_5	$\cdots$
ID	+ 00 M	+ 00 N	+ 00 —		
ID	+ 00 M	- 00 N	+ 00 —		
b_0	b_1	b_2	b_3	b_4	$\cdots$
a_0	a_1	a_2	a_3	a_4	$\cdots$
x_1	y_1	$p(x_1)$	r_1	r_1^2	
x_2	y_2	$p(x_2)$	r_2	$r_1^2+r_2^2$	
$\vdots$	$\vdots$	$\vdots$	$\vdots$	$\vdots$	

NOTES:

(1) $M \leq 14$ has been artificially imposed. Recall that $M \leq 15$ for J285.

(2) In the event of a 13x halt consult the J277 and J281 writeups.

Timing and Accuracy

The Legendre polynomials on [0,1] with N=21 points, 0(.05)1, were used as test problems.

M	x_i entry	y_i entry	Printing	Accuracy
1	5 secs.	2 secs.	7 secs.	8 SD
2	8 "	3 "	10 "	9 "
3	14 "	4 "	12 "	7.9"
4	20 "	6 "	14 "	8.8"
5	28 "	8 "	17 "	7.5"

The times for the x_i entry and y_i entry are exclusive of loading time. The printing times cover only the printing of x_i, y_i, $p(x_i)$, r_i, and r_i^2. The accuracy refers to the number of correct SD in the least accurate of the coefficients.

Program: 66 cards (J288F001 - J288F066).

John Derr

J289E ORTHOGONAL POLYNOMIAL LEAST SQUARES--FLOATING POINT SYSTEM

Given a set of points $x_1, \ldots, x_N$ and an integer $M_x \geq 1$, the program computes

(i) c_j for $j = 0, 1, \ldots, M_x$ and d_j for $j = 1, \ldots M_x$,

(ii) $p_j(x_i)$ for $1 \leq i \leq N$ and $0 \leq j \leq M_x$, and

(iii) (p_j, p_j) for $j = 0, 1, \ldots, M_x$,

where $(f,g) = \sum_{i=1}^{N} f(x_i)g(x_i)$ for any functions f and g defined on $\{x_i\}$, and the $\{p_j\}$ are defined by

$$p_0(x) = 1, p_1(x) = x + c_0, \text{ and for } j \geq 1$$
$$p_{j+1}(x) = (x+c_j)p_j(x) + d_j p_{j-1}(x),$$
and $(p_j, p_k) = 0$ if $j \neq k$.

If, in addition, a set of points $y_1, \ldots, y_N$ is given, the program computes $a_0, a_1, \ldots, a_{M_y}$, provided $M_y \leq M_x$. The $\{a_j\}$ minimize $\|y - \sum_{j=0}^{M} a_j p_j\|^2 = (y - \sum_{j=0}^{M} a_j p_j, y - \sum_{j=0}^{M} a_j p_j)$, where $y(x_i) = y_i$.

Calling Sequence (x_i entry)

α	020	α	010	J289E001
α+1	M_x	N	Δ_x	$L(x_1)$
α+2	Control returns here			

Calling Sequence (y_i entry)

β	020	β	010	J289E095
$\beta+1$	M_y	$L(a_o)$	Δ_y	$L(y_1)$
$\beta+2$	Control returns here			

The above notation is defined by $L(x_{i+1}) = L(x_i) + \Delta_x$, $L(y_{i+1}) = L(y_i) + \Delta_y$, and $L(a_{j+1}) = L(a_j) + 1$.

Application: The double entry is motivated by the case where many sets of $\{y_i\}$ correspond to one set of $\{x_i\}$. Following an x_i entry

(i) (p_j,p_j) will be in A18 + 3j for $j=0,1,\ldots,M_x$,

(ii) c_j will be in A19 + 3j for $j=0,1,\ldots,M_x$,

(iii) d_j will be in A20 + 3j for $j=1,\ldots,M_x$, and

(iv) $p_j(x_i)$ will be in B0 + jN + i - 1 for

$1 \leq i \leq N$ and $0 \leq j \leq M_x$.

A consistent x_i entry must precede the first y_i entry and $M_y \leq M_x$ must always hold. Following a y_i entry $\|y - \sum_{j=0}^{M_y} a_j p_j\|^2$ will be in A14.

Let $p(x) = \sum_{j=0}^{M_y} a_j p_j(x)$. Assuming that a double entry to J289 has occurred and the $\{a_j\}$, $\{c_j\}$, and $\{d_j\}$ are available, $p(x)$ can be readily computed by linking to J291.

Alternatively, let $b_0, b_1, \ldots, b_{M_y}$ denote the coefficients of $p(x)$ with respect to the basis $\{1,x,\ldots,x^{M_y}\}$,

$$\text{i.e.} \quad \sum_{j=0}^{M_y} b_j x^j = \sum_{j=0}^{M_y} a_j p_j(x).$$

Then the $\{b_j\}$ can be obtained by successive links to J290 and J292. In case $M_x = M_y$, the following sequence of links is permissible. J289 (x_i entry), J290, J289 (y_i entry), and J292. The first two links are performed for x_i entries and the last two for y_i entries. As a result in the case of a many-one correspondence between the sets $\{y_i\}$ and $\{x_i\}$, only one link to J290 is required.

An additional flexibility is furnished by a convenient arrangement for re-entries, whereby given a larger value of M the calculation proceeds from where it left off at the smaller value of M. A typical application of re-entries is the following: Suppose we are given a set of number pairs, $\{(x_i, y_i): i = 1,\dots N\}$, and $\varepsilon>0$, and that we seek the smallest M_y such that

$$||y - \sum_{j=0}^{M_y} a_j p_j|| \leq \varepsilon,$$

or we seek the smallest M_y such that

$$||y - \sum_{j=0}^{M_y+1} a_j p_j|| \geq ||y - \sum_{j=0}^{M_y} a_j p_j||.$$

An x_i (y_i) re-entry is effected by

(i) storing $M_x(M_y)$ in the left address of A16,

(ii) storing $\alpha+2$ $(\beta+2)$ in the left address of J289E055, and

(iii) transferring control to the left of J289E056 (J289E107)

Re-entries are governed by the following rules:

(i) An entry must precede the first corresponding re-entry.

(ii) $M_x(M_y)$ must increase with each $x_i(y_i)$ re-entry.

(iii) With the exception of A15, A16, and A20, cells A12

- A18 + $3M_x$, B0 - $BN(M_x+1) - 1$, and those cells occupied by $\{a_j\}$ should not be read into between entries.

(iv) $M_y \leq M_x$ must be valid where M_x refers to the last (and largest) value prescribed for an x_i entry or re-entry.

An entry with $M = M_1$ followed by re-entries with $M = M_2, \ldots, M_e$, $M_1 < M_2 < \ldots < M_e$, requires essentially the same computing time as a single entry with $M = M_e$.

<u>Method:</u>

The x_i entry produces the following results:

(i) Compute $c_0 = -(\sum_1^N x_i)/N$.

(ii) Set $p_1(x_i) = x_i + c_0$ for $i = 1,\ldots,N$.

(iii) Set $p_0(x_i) = 1$ for $i = 1,\ldots,N$, and set $j = 1$.

(iv) Compute $||p_j||^2$.

(v) Exit if $j - M_x \geq 0$; otherwise proceed to the next step.

(vi) Compute $d_j = -||p_j||^2/||p_{j-1}||^2$. (An x_i re-entry begins here).

(vii) Compute $\{p_1 p_j\}$.

(viii) Compute $(p_1 p_j, p_j)$.

(ix) Compute $c_j = -(p_1 p_j, p_j)/||p_j||^2$.

(x) Compute $\{p_1 p_j + d_j p_{j-1}\}$.

(xi) Compute $\{p_{j+1}\} = \{c_j p_j\} + \{p_1 p_j + d_j p_{j-1}\}$.

(xii) Increase j by 1, and recycle beginning with step (iv).

The y_i entry produces the following results:

(i) Set $||y - \sum_{j=0}^{M_y} a_j p_j||^2 = ||y||^2$, and set $j-1 = -1$.

(ii) Compute (y, p_j). (A y_i re-entry begins here).

(iii) Compute $a_j = (y, p_j)/||p_j||^2$.

(iv) Decrease $||y - \sum_{j=0}^{M_y} a_j p_j||^2$ by $a_j(y, p_j)$.

(v) Increase $j-1$ by 1. Exit if $(j-1)-M_y \geq 0$. Otherwise, recycle beginning with step (ii).

Notes:

(1) J271, J272, and J273 are slaves to J289 and hence must be assigned locations prior to the loading of this program.

(2) The program makes references only to the interpreter of the Floating Point System and to that as region F. Region F must be assigned a location coinciding with the first word of the interpreter prior to the loading of this program.

(3) $||p_0||^2 = N$

(4) See the J293 write-up for some timing and accuracy samples.

(5) Good results should not be expected unless the number of distinct x_i exceeds M_x.

J289 Storage

Region A:	A000 - A018 + $3M_x$
Region B:	B000 - B000 + $(M+1)N - 1$
Program:	123 words (J289E001 - J289E123)

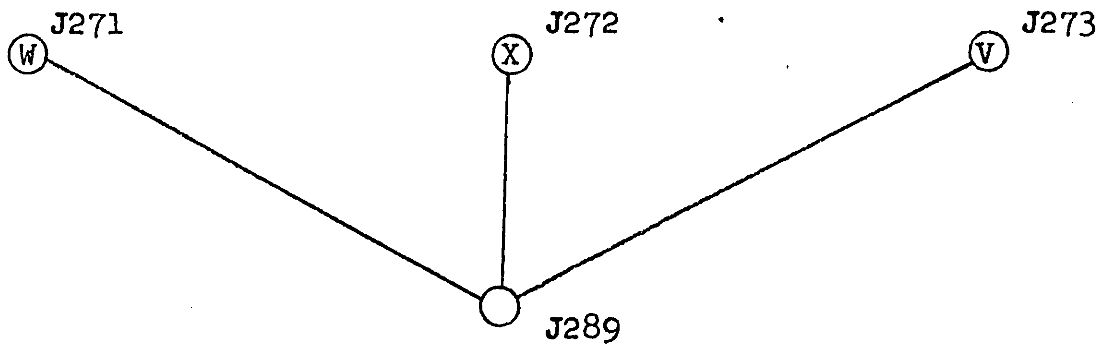
Total Storage Including Slaves - Interpreter Excepted

Region A (the same)

Region B (the same)

Program: 242 words

Lattice Diagram Describing Master - Slave Relations



J290E ORTHOGONAL POLYNOMIAL COEFFICIENTS--FLOATING POINT SYSTEM

An entry to this program produces the coefficients p_{kj} of the first M + 1 polynomials $p_k(x)$ defined by

$$p_0(x) = 1 = p_{00},\ p_1(x) = x + c_0 = p_{01}x^0 + p_{11}x, \text{ and}$$

$$p_{n+1}(x) = (x+c_n)\ p_n(x) + d_n p_{n-1}(x) = \sum_{j=0}^{n+1} p_{j,n+1}x^j \quad \text{for } n \geq 1.$$

Calling Sequence:

α	020	α	010	J290E001
$\alpha+1$	Δ_c	$L(c_o)$		$L(p_{00})$
$\alpha+2$	Δ_d	$L(d_1)$		M
$\alpha+3$	Control returns here			

Here, Δ_c, $\Delta_d \geq 0$, and $L(c_{n+1}) = L(c_n) + \Delta_c$, $L(d_{n+1}) = L(d_n) + \Delta_d$. Upon exit from the program the p_{kj} will be in the (M+1)(M+2)/2 consecutive H.S.S. cells in floating point form as follows:

$$\begin{pmatrix} p_{00} & p_{01} & \cdots & p_{0,\,M-1} & p_{0\,M} \\ & p_{11} & \cdots & p_{1,\,M-1} & p_{1\,M} \\ & & & \vdots & \vdots \\ & & & p_{M-1,M-1} & p_{M-1,M} \\ & & & & p_{MM} \end{pmatrix}$$

The program makes references only to the interpreter of the floating point system and to that as region F. The user must cause F000 to coincide with the first word of the interpreter.

Storage:

Region A : 6 words (A0 - A5)

Program : 71 words (J290E001 - J290E071)

John Derr

J291E ORTHOGONAL POLYNOMIAL EVALUATION -- FLOATING POINT SYSTEM

An entry to this program with x in the MQ yields

$$f(x) = \sum_{j=0}^{M} a_j p_j(x), \text{ where}$$

$$p_0(x) = 1,\ p_1(x) = x + c_0, \text{ and for } n \geq 1$$

$$p_{n+1}(x) = (x+c_n)\ p_n(x) + d_n\ p_{n-1}(x).$$

Upon exit f(x) will be in A6.

Calling Sequence:

α	020	α	010	J291E001	(enter with x in MQ)
$\alpha+1$	Δ_a	$L(a_0)$	Δ_c	$L(c_0)$	
$\alpha+2$		M	Δ_d	$L(d_1) - \Delta_d$	
$\alpha+3$	Control returns here with f(x) in A6.				

Here, Δ_a, Δ_c, $\Delta_d \geq 0$, and $L(a_{j+1}) = \Delta_a + L(a_j)$, $L(c_{j+1}) = \Delta_c + L(c_j)$, and $L(d_j+1) = \Delta_d + L(d_j)$. Only the interpreter of the Floating Point System is required, and it is referred to as region F. The user must cause F000 to coincide with the first word of the interpreter.

Storage:

Erasable	:	7 words	(A0-A6)
Program	:	38 words	(J291E001 - J291E038)

J292E TRIANGULAR MATRIX X VECTOR - FLOATING POINT SYSTEM

The program computes for $k = 0,1,\ldots,M$,

$$b_k = \sum_{j=k}^{M} a_j \, p_{kj} \, .$$

CALLING SEQUENCE:

α	020	α	010	J292E001
$\alpha+1$		$L(b_0)$	Δ_a	$L(a_0)$
$\alpha+2$		M		$L(p_{00})$
$\alpha+3$	Control returns here.			

Here $L(a_{j+1}) = L(a_j) + \Delta_a$, $\Delta_a > 0$, $L(b_{j+1}) = L(b_j) + 1$, and the p_{kj} must be in the consecutive (M+1)(M+2)/2 H.S.S. cells as follows:

$$\begin{pmatrix} p_{00} & p_{01} & p_{02} & \cdots & p_{0,M-1} & p_{0M} \\ & p_{11} & p_{12} & \cdots & p_{1,M-1} & p_{1M} \\ & & & & \vdots & \vdots \\ & & & & p_{M-1,M-1} & p_{M-1,M} \\ & & & & & p_{MM} \end{pmatrix}$$

The $\{p_{kj}\}$ constitute an upper triangular matrix. The storage assignment is compressed so as to eliminate the cells corresponding to matrix elements below the main diagonal.

The program makes references only to the interpreter of the floating point system and to that as region F. The user must cause F000 to coincide with the first word of the interpreter.

J271E is a slave to this program and is referred to as region W. The user must cause W000 to coincide with J271E000.

<u>STORAGE</u>: 20 words (J292E001 - J292E020).

John Derr

J293F ORTHOGONAL POLYNOMIAL LEAST SQUARES--FLOATING POINT SYSTEM

J293F is an absolute binary assembly of J289E together with J271-3, J290, J292, J240, J242, J244, and a program which loads, computes and prints in a manner to be described. To use J293F form a deck consisting of

(i) one blank card

(ii) J140A

(iii) J135A

(iv) J293F

(v) data

(vi) 2 blank cards

<u>Header Card Format - 013 Style Input:</u>

pos	(1)		
pos	(2)	S	$\begin{cases} + \text{ if an entry} \\ - \text{ if a re-entry} \end{cases}$
		EXP	blank or zero
		MANTISSA	M
pos	(3)	S	$\begin{cases} + \text{ if } x_1 \\ - \text{ if } y_1 \end{cases}$
		EXP	blank or zero
		MANTISSA	N

$$\text{pos}\quad(4)\quad \begin{cases} < 0 \text{ and pos } (3) < 0 & \text{if the printing of } x_i, y_i, p(x_i), r_i, \text{ and } \Sigma r_i^2 \text{ is to be suppressed.} \\ < 0 \text{ and pos } (3) \geq 0 & \text{if the sheet eject prior to the printing of the } x_i \text{ header card is to be replaced by skipping 4 lines.} \\ \geq 0 & \text{otherwise} \end{cases}$$

Data Card Format

If pos (2) $\geq$ 0 and pos (3) $\geq$ 0 (pos (3) < 0) for a header card, then N values of x_i (y_i) will be loaded 6/card beginning with pos (1) of the next card. The card containing x_N (y_N) must have a 12 punch in col. 80. Upon loading the latter card an x_i(y_i) entry to J289 occurs with M_x (M_y) = M.

Ordering of Header Cards and Data Cards

A set of number pairs (x_i, y_i) is specified by an x_i header card with pos (2) $\geq$ 0 followed by x_i data cards and a y_i header card with pos (2) $\geq$ 0 followed by y_i data cards. The x_i cards must precede the y_i cards. In the case of re-entries no data cards follow the corresponding header cards. See the J289 write-up for a discussion of re-entries and a many-one correspondence between the sets $\{y_i\}$ and $\{x_i\}$.

<u>Printing</u>

The page is ejected (the page is spaced four lines) following the reading of each x_i header card if pos (4) ≥ 0 (<0). The page is spaced one line following the reading of each y_i header card. In all cases the contents of pos (1) - (4) of the header card are printed immediately thereafter. The page is spaced two lines prior to a y_i entry or re-entry. Following that entry $\|y - \sum_{j=0}^{M} a_j p_j\|^2$ is printed in the x_2 field. Following the calculation of $b_0, b_1, \ldots, b_{M_y}$ the printer is spaced one line and the $\{b_j\}$ are printed 8/line. Then if pos (4) ≥ 0, the printer is spaced two lines, and the program computes $p(x_i) = \sum_{j=0}^{M} a_j p_j(x_i)$, $r_i = p(x_i) - y_i$, and $\sum_{k=1}^{i} r_k^2$, and prints x_i, y_i, $p(x_i)$, r_i, and $\sum_{k=1}^{i} r_k^2$ in positions $x_1, \ldots, x_5$ for $i = 1, \ldots, N$. An illustrative printing layout follows:

```
                 x1        x2        x3        x4        x5   ...

                 ID   +  00  3  + 00  5  +  00 —
one space
                 ID   +  00  3  - 00  5  +  00 —
two spaces
```

$$\|y - \sum_{j=0}^{3} a_j p_j\|^2$$

```
one space
                 a0        a1        a2        a3
```

two spaces	x_1	x_2	x_3	x_4	x_5
	x_1	y_1	$p(x_1)$	r_1	r_1^2
	x_2	y_2	$p(x_2)$	r_2	$r_1^2+r_2^2$
	x_3	y_3	$p(x_3)$	r_3	$r_1^2+r_2^2+r_3^2$
	x_4	y_4	$p(x_4)$	r_4	$r_1^2+\ldots+r_4^2$
	x_5	y_5	$p(x_5)$	r_5	$r_1^2+\ldots+r_5^2$

Notes: (1) $M \leq 25$ and $N \leq 90$ must hold.

(2) The number of distinct x_i should exceed M.

Timing and Accuracy:

(1) The Legendre polynomials on $[0,1]$ with N = 21 points, 0(.05)1 yielded the following information operating in the SD mode:

M	x_i entry	y_i entry	compute $\{b_j\}$	Print	Accuracy
1	2	1	1	7	9
2	4	2	1	7	8.8
3	6	3	1	8	7.2
4	8	3	2	8	6.8
5	11	4	2	9	6.1

The times in seconds under x_i entry to print inclusive are exclusive of loading times. The accuracy refers to the number of correct SD in the least accurate of the coefficients.

(2) With the duplication of the points 0, .05, .5, .95, and 1 (i.e. N = 26), the following results were obtained assuming entries with M = 5 have occurred previously:

M	x_1 re-entry	y_1 re-entry	compute $\{b_j\}$	Print
10	15	3	5	14
15	15	3	9	18
20	15	3	16	22
25	15	3	25	25

Program: 102 cards (J293F001 - J293F102).

J294E ORTHOGONAL POLYNOMIAL LEAST SQUARES--EQUALLY SPACED POINTS

Given a set of points $x_1,\ldots,x_N$ and an integer $M_x \geq 1$ such that $\sum_1^N (x_{i-m}^{2k+1}) = 0$ for $2k+1 = 1,3,5,\ldots,M_x/(M_{x-1})$, where $m = (\sum_1^N x_i)/N$, the program computes

(i) d_j for $j = 1,\ldots M_x$,

(ii) $p_j(\overline{x}_i)$ for $1 \leq i \leq N$ and $0 \leq j \leq M_x$, and

(iii) (p_j,p_j) for $j = 0,1,\ldots,M_x$,

where $(f,g) = \sum_{i=1}^N f(\overline{x}_i)g(\overline{x}_i)$ for any functions f and g defined on $\{\overline{x}_i\}, \overline{x} = x-m$, and the $\{p_j\}$ are defined by

$$p_0(\overline{x}) = 1, p_1(\overline{x}) = \overline{x}$$

$$p_{j+1}(\overline{x}) = \overline{x}p_j(\overline{x}) + d_j p_{j-1}(\overline{x}) \text{ for } j \geq 1,$$

and $(p_j,p_k) = 0$ if $j \neq k$.

If, in addition, a set of points $y_1,\ldots,y_N$ is given, the program computes $a_0, a_1,\ldots,a_{M_y}$, provided $M_y \leq M_x$. The $\{a_j\}$ minimize $||y - \sum_{j=0}^M a_j p_j||^2 = (y - \sum_{j=0}^M a_j p_j,\ y - \sum_{j=0}^M a_j p_j)$, where $y(\overline{x}_i) = y_i$.

<u>Calling Sequence (x_i entry)</u>

α	020	α	010	J294E001
$\alpha+1$	M_x	N	Δ_x	$L(x_1)$
$\alpha+2$	Control returns here			

Calling Sequence (y_i entry)

β	020	β	010	J294E075
$\beta+1$	M_y	$L(a_0)$	Δ_y	$L(y_1)$
$\beta+2$	Control returns here			

The above notation is defined by $L(x_{i+1}) = L(x_i) + \Delta_x$,

$L(y_{i+1}) = L(y_i)+\Delta_y$, and $L(a_{j+1}) = L(a_j) + 1$.

Application: The double entry is motivated by the case where many sets of $\{y_i\}$ correspond to one set of $\{x_i\}$. Following an x_i entry

(i) (p_j,p_j) will be in A18 + 3j for $j=0,1,\ldots,M_x$,

(ii) -m will be in A15.

(iii) d_j will be in A20 + 3j for $j=1,\ldots,M_x$, and

(iv) $p_j(\overline{x}_i)$ will be in B0 + jN + i - 1 for $1 \le i \le N$ and $0 \le j \le M_x$.

A consistent x_i entry must precede the first y_i entry and $M_y \le M_x$ must always hold. Following a y_i entry

$||y - \sum_{j=0}^{M_y} a_j p_j||^2$ will be in A14.

Let $p(x) = \sum_{j=0}^{M_y} a_j p_j(\overline{x})$. Assuming that a double entry to J294 has occurred and the $\{a_j\}$, $\{c_j\}$, and $\{d_j\}$ are available, p(x) can be readily computed by linking to J296 with $\overline{x}$ in the MQ.

Alternatively, let $b_0, b_1,\ldots,b_{M_y}$ denote the coefficients of p(x) with respect to the basis $\{1,x,\ldots,x^{M_y}\}$,

i.e. $\sum_{j=0}^{M_y} b_j x^j = \sum_{j=0}^{M_y} a_j p_j(\overline{x})$.

Then the $\{b_j\}$ can be obtained by successive links to J295, J292, and J275. In case $M_x = M_y$, the following sequence of links is permissible. J294 (x_i entry), J295, J294 (y_i entry), J292 and J275. The first two links are performed for x_i entries and the last three for y_i entries. As a result in the case of a many-one correspondence between the sets $\{y_i\}$ and $\{x_i\}$, only one link to J295 is required.

An additional flexibility is furnished by a convenient arrangement for re-entries, whereby given a larger value of M the calculation proceeds from where it left off at the smaller value of M. A typical application of re-entries is the following: Suppose we are given a set of number pairs, $\{(x_i, y_i): i = 1,\ldots N\}$, and $\varepsilon>0$, and that we seek the smallest M_y such that

$$||y - \sum_{j=0}^{M_y} a_j p_j|| \leq \varepsilon,$$

or we seek the smallest M_y such that

$$||y - \sum_{j=0}^{M_y+1} a_j p_j|| \geq ||y - \sum_{j=0}^{M_y} a_j p_j||.$$

An x_i (y_i) re-entry is effected by

(i) storing $M_x(M_y)$ in the left address of A16,

(ii) storing $\alpha+2$ ($\beta+2$) in the left address of J294E050, and

(iii) transferring control to the left of J294E051 (J294E087)

Re-entries are governed by the following rules:

(i) An entry must precede the first corresponding re-entry.

(ii) $M_x(M_y)$ must increase with each $x_i(y_i)$ re-entry.

(iii) With the exception of A16, and A20, cells A12

- A18 + $3M_x$, BO - BN(M_x+1) - 1, and those cells occupied by $\{a_j\}$ should not be read into between entries.

(iv) $M_y \leq M_x$ must be valid where M_x refers to the last (and largest) value prescribed for an x_i entry or re-entry.

An entry with $M = M_1$ followed by re-entries with $M = M_2, \ldots, M_e$, $M_1 < M_2 < \ldots < M_e$, requires essentially the same computing time as a single entry with $M = M_e$.

Method:

The x_i entry produces the following results:

(i) Compute $-m = -(\sum_1^N x_i)/N$.

(ii) Set $p_1(\bar{x}_i) = \bar{x}_i$ for $i = 1,\ldots,N$.

(iii) Set $p_0(\bar{x}_i) = 1$ for $i = 1,\ldots,N$, and set $j = 1$.

(iv) Compute $||p_j||^2$.

(v) Exit if $j - M_x \geq 0$; otherwise proceed to the next step.

(vi) Compute $d_j = -||p_j||^2/||p_{j-1}||^2$. (An x_i re-entry begins here).

(vii) Compute $\{p_1 p_j\}$.

(viii) Compute $(p_1 p_j, p_j)$.

(ix) Compute $\{p_{j+1}\} = \{p_1 p_j + d_j p_{j-1}\}$.

(x) Increase j by 1, and recycle beginning with step (iv).

The y_i entry produces the following results:

(i) Set $||y - \sum_{j=0}^{M_y} a_j p_j||^2 = ||y||^2$, and set $j-1 = -1$.

(ii) Compute (y,p_j). (A y_i re-entry begins here).

(iii) Compute $a_j = (y,p_j)/||p_j||^2$.

(iv) Decrease $||y - \sum_{j=0}^{M_y} a_j p_j||^2$ by $a_j(y,p_j)$.

(v) Increase $j-1$ by 1. Exit if $(j-1)-M_y \geq 0$. Otherwise, recycle beginning with step (ii).

Notes:

(1) J271, J272, and J273 are slaves to J294 and hence must be assigned locations prior to the loading of this program.

(2) The program makes references only to the interpreter of the Floating Point System and to that as region F. Region F must be assigned a location coinciding with the first word of the interpreter prior to the loading of this program.

(3) $||p_0||^2 = N$

(4) See the J297 write-up for some timing and accuracy samples.

(5) Good results should not be expected unless the number of distinct x_i exceeds M_x.

J294 Storage

Region A:	A000 - A018 + $3M_x$
Region B:	B000 - B000 + $(M+1)N - 1$
Program:	103 words (J294E001 - J294E103)

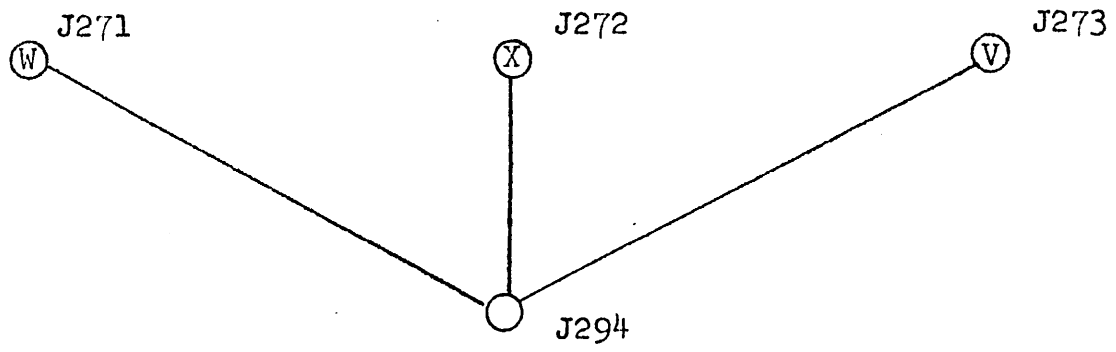
Total Storage Including Slaves - Interpreter Excepted

Region A (the same)

Region B (the same)

Program: 222 words

Lattice Diagram Describing Master - Slave Relations



J295E ORTHOGONAL POLYNOMIAL COEFFICIENTS--EQUALLY SPACED POINTS

An entry to this program produces the coefficients p_{kj} of the first M + 1 polynomials $p_k(x)$ defined by

$$p_0(x) = 1 = p_{00},\ p_1(x) = x = p_{01}x^0 + p_{11}x, \text{ and}$$

$$p_{n+1}(x) = xp_n(x) + d_n p_{n-1}(x) = \sum_{j=0}^{n+1} p_{j,n+1}x^j \quad \text{for } n \geq 1.$$

Calling Sequence:

α	020	α	010	J295E001
$\alpha+1$				$L(p_{00})$
$\alpha+2$	Δ_d	$L(d_1)$		M
$\alpha+3$	Control returns here			

Here, $\Delta_d \geq 0$, and $L(d_{n+1}) = L(d_n) + \Delta_d$. Upon exit from the program the p_{kj} will be in the (M+1)(M+2)/2 consecutive H.S.S. cells in floating point form as follows:

$$\begin{pmatrix} p_{00} & p_{01} & \cdots & p_{0,M-1} & p_{0\,M} \\ & p_{11} & \cdots & p_{1,M-1} & p_{1\,M} \\ & & & \vdots & \vdots \\ & & & p_{M-1,M-1} & p_{M-1,M} \\ & & & & p_{MM} \end{pmatrix}$$

Those p_{kj} for which j+k is odd are zero, but zeros are not stored by this program in the corresponding cells. A good procedure is to clear the (M+1)(M+2)/2 cells to zero prior to each entry to this program with a new value for M.

The program makes references only to the interpreter of the floating point system and to that as region F. The user must cause F000 to coincide with the first word of the interpreter.

Storage:

Region A	:	4 words	(A0-A1, A2,A4)
Program	:	51 words	(J295E001 - J295E051)

J296E ORTHOGONAL POLYNOMIAL EVALUATION--EQUALLY SPACED POINTS

An entry to this program with x in the MQ yields

$$f(x) = \sum_{j=0}^{M} a_j p_j(x), \quad \text{where}$$

$$p_0(x) = 1, \; p_1(x) = x, \text{ and for } n \geq 1$$

$$p_{n+1}(x) = xp_n(x) + d_n \, p_{n-1}(x).$$

Upon exit f(x) will be in A6.

Calling Sequence:

α	020	α	010	J296E001	(enter with x in MQ
$\alpha+1$	Δ_a	$L(a_0)$			
$\alpha+2$		M	Δ_d	$L(d_1) - \Delta_d$	
$\alpha+3$	Control returns here with f(x) in A6.				

Here, $\Delta_a, \Delta_d \geq 0$, $L(a_{j+1}) = \Delta_a + L(a_j)$, and $L(d_j+1) = \Delta_d + L(d_j)$. Only the interpreter of the Floating Point System is required, and it is referred to as region F. The user must cause F000 to coincide with the first word of the interpreter.

Storage:

Erasable	:	7 words	(A0-A6)
Program	:	33 words	(J296E001 - J296E033)

J297F ORTHOGONAL POLYNOMIAL LEAST SQUARES--EQUALLY SPACED POINTS

J297F is an absolute binary assembly of J294E together with J271-3, J275, J295, J292, J240, J242, J244, and a program which loads, computes and prints in a manner to be described. To use J297F form a deck consisting of

(i) one blank card

(ii) J140A

(iii) J135A

(iv) J297F

(v) data

(vi) 2 blank cards

<u>Header Card Format - 013 Style Input:</u>

pos	(1)		
pos	(2)	S	+ if an entry - if a re-entry
		EXP	blank or zero
		MANTISSA	M
pos	(3)	S	+ if x_i - if y_i
		EXP	blank or zero
		MANTISSA	N

$$\text{pos}\quad (4)\quad \begin{cases} < 0 \text{ and pos } (3) < 0 & \text{if the printing of } x_i, y_i, p(x_i), r_i, \text{ and } \Sigma r_i^2 \text{ is to be suppressed.} \\ < 0 \text{ and pos } (3) \geq 0 & \text{if the sheet eject prior to the printing of the } x_i \text{ header card is to be replaced by skipping 4 lines.} \\ \geq 0 & \text{otherwise} \end{cases}$$

Data Card Format

If pos (2) ≥ 0 and pos (3) ≥ 0 (pos (3) < 0) for a header card, then N values of x_i (y_i) will be loaded 6/card beginning with pos (1) of the next card. The card containing x_N (y_N) must have a 12 punch in col. 80. Upon loading the latter card an $x_i(y_i)$ entry to J294 occurs with M_x (M_y) = M.

Ordering of Header Cards and Data Cards

A set of number pairs (x_i, y_i) is specified by an x_i header card with pos (2) ≥ 0 followed by x_i data cards and a y_i header card with pos (2) ≥ 0 followed by y_i data cards. The x_i cards must precede the y_i cards. In the case of re-entries no data cards follow the corresponding header cards. See the J294 write-up for a discussion of re-entries and a many-one correspondence between the sets $\{y_i\}$ and $\{x_i\}$.

Printing

The page is ejected (the page is spaced four lines) following the reading of each x_1 header card if pos (4) ≥ 0 (<0). The page is spaced one line following the reading of each y_1 header card. In all cases the contents of pos (1) - (4) of the header card are printed immediately thereafter. The page is spaced two lines prior to a y_1 entry or re-entry. Following that entry $||y - \sum_{j=0}^{M} a_j p_j||^2$ is printed in the x_2 field. Following the calculation of $b_0, b_1, \ldots, b_{M_y}$ the printer is spaced one line and the $\{b_j\}$ are printed 8/line. Then if pos (4) ≥ 0, the printer is spaced two lines, and the program computes $p(x_i) = \sum_{j=0}^{M} a_j p_j(x_i)$, $r_i = p(x_i) - y_i$, and $\sum_{k=1}^{i} r_k^2$, and prints x_i, y_i, $p(x_i)$, r_i, and $\sum_{k=1}^{i} r_k^2$ in positions $x_1, \ldots, x_5$ for $i = 1, \ldots, N$. An illustrative printing layout follows:

```
              x1      x2      x3      x4      x5   ...
              ID   +  00  3 + 00  5 + 00 —
one space
              ID   +  00  3 - 00  5 + 00 —
two spaces
```

$$||y - \sum_{j=0}^{3} a_j p_j||^2$$

one space

a_0 a_1 a_2 a_3

	x_1	x_2	x_3	x_4	x_5
two spaces	x_1	y_1	$p(x_1)$	r_1	r_1^2
	x_2	y_2	$p(x_2)$	r_2	$r_1^2+r_2^2$
	x_3	y_3	$p(x_3)$	r_3	$r_1^2+r_2^2+r_3^2$
	x_4	y_4	$p(x_4)$	r_4	$r_1^2+\ldots+r_4^2$
	x_5	y_5	$p(x_5)$	r_5	$r_1^2+\ldots+r_5^2$

Notes: (1) $M \leq 25$ and $N \leq 90$ must hold.

(2) The number of distinct x_i should exceed M.

Timing and Accuracy:

(1) The Legendre polynomials on $[0,1]$ with N = 21 points, 0(.05)1 yielded the following information operating in the SD mode:

M	x_i entry	y_i entry	compute $\{b_j\}$	Print	Accuracy
1	2	1	1	7	9
2	3	2	1	7	8.6
3	4	3	1	8	8.5
4	5	3	2	8	8.7
5	7	4	2	9	7.5

The times in seconds under x_i entry to print inclusive are exclusive of loading times. The accuracy refers to the number of correct SD in the least accurate of the coefficients.

(2) With the duplication of the points 0, .05, .5, .95, and 1 (i.e. N = 26), the following results were obtained assuming entries with M = 5 have occurred previously:

M	x_i re-entry	y_i re-entry	compute b_j	Print
10	9	3	4	14
15	9	3	7	18
20	9	3	14	22
25	9	3	21	25

Program: 103 cards (J293F001 - J293F103).

J298E* INTEGRATION BY SIMPSON'S RULE WITH PRESCRIBED ERROR--FLOATING POINT SYSTEM

J298 computes $\int_a^b f(x)dx$, where f (x) is computed by an auxiliary subroutine written by the programmer. J298 divides the domain from a to b, $a < b$, into N intervals of length Δ_i, $i = 1, 2, \ldots, N$, and computes the area $\int_i^{(5)}$ for each subinterval by Simpson's 5 point formula. The $\{\Delta_i\}$ are chosen small enough so that the relative error E_M in the interval a_M to $a_M + \Delta_M$ is less than the error prescribed by the programmer. i.e., so that

$$(1) \qquad E_M = \frac{\int_M^{(3)} - \int_M^{(5)}}{\sum_{i=1}^{M} \int_i^{(5)}} < \epsilon,$$

where $\int_i^{(3)}$ = the area over the ith interval as computed by Simpson's 3 pt. formula.

$\int_i^{(5)}$ = the area over the ith interval as computed by Simpson's 5 pt. formula.

and ϵ = the error prescribed by the programmer.

The maximum error possible over the entire interval from a to b is $N\epsilon \max F(x)$ where $F(x) = \int_a^x f(t)dt$. The method is a modification of that described in Scarborough.[1]

[1] Scarborough, <u>Numerical Mathematical Analysis</u>, 3rd edition, p. 178.

*J298 is J192 as modified for the revised JOHNNIAC Floating Point System.

Calling sequence:

```
α       020     α       010     β
α+1                     a
α+2                     b
α+3                     Δ1
α+4
α+5     Control returns here in floating point
```

a, b, Δ_1, and ϵ must be in the floating point data form. If the starting Δ is not prescribed but is to be determined by J298, step $\alpha+3$ must be negative.

The computational procedure is as follows:

Define $h_1 = \frac{\Delta_1}{4}$. Let h_1' be a temporary value for h_1. If step $\alpha+3$ is minus, make $\Delta_1' = b-a$, otherwise make Δ_1' the contents of $\alpha+3$.

If $E_1 > \epsilon$ replace h_1' by $h_1'/2$. Repeat this halving procedure until either $E_1 \leq \epsilon$ or $h_1'/2 \leq 10^{-4}\ (a_1 + \Delta_1')$. In both cases the last h_1' is used in computing $\int_1 (5)$. In the latter case a row of 11 nines is printed to indicate that the error criterion was not met. This process is used for each interval $i = 1, 2, \ldots, N$.

The constant 10^{-4} is located in the 96th (last) word. Thus the $10^{-4}\ (a_M+\Delta_M)$ criterion can be changed by the programmer to fit a particular problem. If $500\ E_1 \leq \epsilon$, set $h_{1+1}' = 2h_1$. The constant 500, located in word 95 can also be changed.

The auxiliary closed subroutine is entered by J298, with x in the floating point AMQ by means of the calling sequence:

```
γ       020     γ       010     1000
γ+1     Control returns here in floating point
```

At step $\gamma+1$ f(x) should be located in the AMQ and the floating point system should be in control, i.e., the exit from the auxiliary subroutine must not be made with an 010 (EXL) order.

The final exit from J298 to step $\alpha+5$ is made with the floating point system in control.

If cell A0 is positive, after the computation on the M th interval is completed, the following line of information will be printed in the Floating Point print format:

x_1	x_2	x_3	x_4	x_5	x_6	x_7
$\sum_{i=1}^{M} \int_i^{(5)}$	a	$a_M+\Delta_M$	—	$\int_M^{(5)}$	a_M	Δ_M

where a_M and $a_M+\Delta_M$ are the first and last points of the M th subinterval, i.e., $a_1=a$ and $a_N+\Delta_N=b$. The printer will be spaced one line following each such line of printing.

If cell A0 is negative after the computation on the M th interval is completed, there will be no printing. On exit from J298 $\sum_{i=1}^{N} \int_i^{(5)}$ will be in cell B0.

Symbolic code: 96 words

Region A (erasable): 6 words, A0, A1, A4-A7.

Region B (semi-erasable): 19 words, B0 to B18.
Region B is erasable except for the f(x) routine.

Region F: Interpreter of Floating Point System. The user must cause F0 to coincide with the first word of the interpreter.

Region I: f(x) routine

Program available as J298E.

Marvin Shapiro
Revised by John Derr

12-22-58

J301E SQUARE ROOT - JOHNNIAC FLOATING POINT SYSTEM

When used in conjunction with J300E, J301E will execute the 051 floating point operation as described in the JOHNNIAC Floating Point Interpretive System manual.* J300 must be loaded prior to J301E, and the first word of J300E must coincide with F000.

<u>Program Storage:</u> 27 words (28 style E cards - J301E000 to J301E027).

Region E: 13 words, E000 - E012.

* J301 differs from J245 as follows: The operation $\sqrt{y} \cdot 10^9$ described in step 6 on page 49 is now rounded. Step 7 on page 49 should be changed to read,

7) The remaining steps of this operation are like steps 6-10 for the ADD operation.

John Derr

J302E COMPLETE ELLIPTIC INTEGRAL OF FIRST KIND - FLOATING POINT

An entry to this program with $1-k^2$ in the AMQ will result in the computation of

$$K(k) = \int_0^{\pi/2} \frac{d\alpha}{\sqrt{1-k^2 \sin^2 \alpha}} \text{, where } 0 \leq k < 1 .$$

Upon exit from the program $K(k)$ will be in the AMQ.

CALLING SEQUENCE

α 020 α 010 J302E001 (Execute with $1-k^2$ in AMQ)

$\alpha+1$ Control returns with $K(k)$ in the AMQ.

METHOD: The program approximates $K(k)$ by

$$K^*(k) = a_0 + \eta\left[a_1 + a_2\eta\right] - \left\{b_0 + \eta\left[b_1 + b_2\eta\right]\right\} \log \eta ,$$

where $\eta = 1-k^2$ and

a_0	$= 1.3862944$	b_0	$= .5$
a_1	$= .1119723$	b_1	$= .1213478$
a_2	$= .0725296$	b_2	$= .0288729$.*

ACCURACY: $\max_k |K^*(k) - K(k)| \leq .00003^*$.

NOTES: This program requires only the interpreter of the Floating Point System, and it is referred to as region F.

*Reference: Cecil Hastings, Jr., Approximations for Digital Computers, R-264, 1954, p.170.

The user must cause F000 to coincide with its first word.

Program Storage: 17 words (J302E001 - J302E017)

Erasable Storage (Region A): 2 words (A000-A001).

J303E COMPLETE ELLIPTIC INTEGRAL OF SECOND KIND - FLOATING POINT

An entry to this program with $1-k^2$ in the AMQ will result in the computation of $E(k)-1$ for $0 \leq k < 1$, where

$$E(k) = \int_0^{\pi/2} \sqrt{1-k^2 \sin^2 \alpha} \quad d\alpha .$$

Upon exit from the program $E(k)-1$ will be in the AMQ.

CALLING SEQUENCE:

```
α     020     α     010     J303E001  (Execute with 1-k²
                                        in AMQ)

α+1  Control returns here with  E(k)-1  in the AMQ
```

METHOD: The program approximates $E(k)-1$ by

$$E^*(k)-1 = \eta\left[a_1+\eta(a_2+a_3\eta)\right] - \left\{\eta\left[b_1+\eta(b_2+b_3\eta)\right]\right\} \log \eta ,$$

where $\eta = 1-k^2$ and

$a_1 = .44479204$	$b_1 = .249697949$
$a_2 = .085099193$	$b_2 = .08150224$
$a_3 = .040905094$	$b_3 = .01382999$.*

ACCURACY: $\max_k |E^*(k) - E(k)| \leq .0000007.$*

NOTES: This program requires only the interpreter of the Floating Point System, and it is referred to as region F. The user must cause F000 to coincide with its first word.

Program Storage: 18 words (J303E001 - J303E018)

Erasable Storage (region A): 2 words (A000-A001)

*Reference: Cecil Hastings, Jr., Approximations for Digital Computers, R-264, 1954, p.174.

J304E COMPLETE ELLIPTIC INTEGRAL OF THIRD KIND--FLOATING POINT

An entry to this program with n in A2 and k^2 in A3 will yield the evaluation of

$$\pi(n,k) = \int_0^{\pi/2} \frac{d\varphi}{(1+n \sin^2\varphi)\sqrt{1-k^2 \sin^2\varphi}}$$

where $n > -1$ and $0 \leq k < 1$. Upon exit from the program $\pi(n,k)$ will be in B0.

Calling Sequence:

```
        α        020        α        010        J304E001
        α+1      Control returns here.
```

Method: The integral is evaluated by linking to J298 with $a = 0$, $b = 1.5707963$, $\Delta_1 = .1$, and $\epsilon = 10^{-6}$. The quantities a, b, Δ_1, and ϵ are in J304E003 - J304E006 respectively.

Accuracy: The error is bounded by $N\epsilon\pi(n,k)$ and is generally much smaller, where N is as defined in J298--the number of intervals Δ_i into which $[0,\pi/2]$ is divided. N increases as $n \longrightarrow -1$ or $k \longrightarrow 1$, but for an "average" case $\Delta_i \geq .1$ for $i = 1, 2, \ldots, N-1$.

Notes:

(1) $\pi(0,k) = K(k)$

(2) $\pi(-k^2,k) = E(k)/(1-k^2)$.

(3) J304 makes references to the interpreter of the

Floating Point System as region F and to J298 as region Z. Hence regions F and Z must be assigned prior to loading J304.

(4) J304 uses the 051 and 052 floating point operations.

(5) Region I begins at J304E008. See the J298 write-up.

(6) If cell A0 is positive printing will occur as described in the J298 write-up.

(7) Cells A0 and J304E005-6 can be manipulated by the user within the limitations imposed by J298.

Storage:

Symbolic Code:	18 words; 19 cards--J304E001-J304E019		
Region A (erasable):	2 words,	A2-A3	for J304;
	8 words,	A0-A7	including J298.
Region B (semi-erasable):	1 word,	B0	for J304;
	19 words,	B0-B18	including J298.
Region F:	Interpreter of the Floating Point System.		
Region I:	f(x) auxiliary routine.		

J305E ORTHOGONAL POLYNOMIAL LEAST SQUARES WITH WEIGHTS

Given a set of points $x_1, \ldots, x_N$, a set of non-negative points $w_1, \ldots, w_N$ satisfying $\sum_{1}^{n} w_i > 0$, and an integer $M_x \geq 1$, the program computes

(i) c_j for $j = 0, 1, \ldots, M_x$ and d_j for $j = 1, \ldots M_x$,

(ii) $p_j(x_i)$ for $1 \leq i \leq N$ and $0 \leq j \leq M_x$, and

(iii) (p_j, p_j) for $j = 0, 1, \ldots, M_x$,

where $(f,g) = \sum_{i=1}^{N} w_i\, f(x_i)g(x_i)$ for any functions f and g defined on $\{x_i\}$, and the $\{p_j\}$ are defined by

$$p_0(x) = 1, p_1(x) = x + c_0, \text{ and for } j \geq 1$$

$$p_{j+1}(x) = (x+c_j)p_j(x) + d_j p_{j-1}(x),$$

and $(p_j, p_k) = 0$ if $j \neq k$.

If, in addition, a set of points $y_1, \ldots, y_N$ is given, the program computes $a_0, a_1, \ldots, a_{M_y}$, provided $M_y \leq M_x$. The $\{a_j\}$ minimize

$$||y - \sum_{j=0}^{M} a_j p_j||^2 = (y - \sum_{j=0}^{M} a_j p_j,\ y - \sum_{j=0}^{M} a_j p_j), \text{ where } y(x_i) = y_i.$$

<u>Calling Sequence (x_i entry)</u>

α	020	α	010	J305E001
α+1	M_x	N	Δ_x	$L(x_1)$
α+2			Δ_w	$L(w_1)$
α+3	Control returns here			

Calling Sequence (y_i entry)

β	020	β	010	J305E105
$\beta+1$	M_y	$L(a_o)$	Δ_y	$L(y_1)$
$\beta+2$	Control returns here			

The above notation is defined by $L(x_{i+1}) = L(x_i) + \Delta_x$, $L(w_{i+1}) = L(w_i) + \Delta_w$, $L(y_{i+1}) = L(y_i) + \Delta_y$, and $L(a_{j+1}) = L(a_j) + 1$.

Application: The double entry is motivated by the case where many sets of $\{y_i\}$ correspond to one set of $\{x_i\}$ and $\{w_i\}$. Following and x_i entry

(i) (p_j, p_j) will be in A18 + 3j for $j=0, 1, \ldots, M_x$,

(ii) c_j will be in A19 + 3j for $j=0,1,\ldots,M_x$,

(iii) d_j will be in A20 + 3j for $J=1,\ldots,M_x$, and

(iv) $p_j(x_i)$ will be in B0 + (j+1)N + i - 1 for

$1 \leq i \leq N$ and $0 \leq j \leq M_x$.

A consistent x_i entry must precede the first y_i entry and $M_y \leq M_x$ must always hold. Following a y_i entry

$||y - \sum_{j=0}^{M_y} a_j p_j||^2$ will be in A14.

Let $p(x) = \sum_{j=0}^{M_y} a_j p_j(x)$. Assuming that a double entry to J305 has occurred and the $\{a_j\}$, $\{c_j\}$, and $\{d_j\}$ are available, p(x) can be readily computed by linking to J291.

Alternatively, let $b_0, b_1, \ldots, b_{M_y}$ denote the coefficients of $p(x)$ with respect to the basis $\{1, x, \ldots, x^{M_y}\}$, i.e. $$\sum_{j=0}^{M_y} b_j x^j = \sum_{j=0}^{M_y} a_j p_j(x).$$

Then the $\{b_j\}$ can be obtained by successive links to J290 and J292. In case $M_x = M_y$, the following sequence of links is permissible. J305 (x_i entry), J290, J305 (y_i entry), and J292. The first two links are performed for x_i entries and the last two for y_i entries. As a result in the case of a many-one correspondence between the sets $\{y_i\}$ and $\{x_i\} \cup \{w_i\}$, only one link to J305 is required.

An additional flexibility is furnished by a convenient arrangement for re-entries, whereby given a larger value of M the calculation proceeds from where it left off at the smaller value of M. A typical application of re-entries is the following: Suppose we are given the set $\{w_i\}$ and a set of number pairs, $\{(x_i, y_i) : i = 1, \dots N\}$, and $\varepsilon > 0$, and that we seek the smallest M_y such that

$$||y - \sum_{j=0}^{M_y} a_j p_j|| \leq \varepsilon,$$

or we seek the smallest M_y such that

$$||y - \sum_{j=0}^{M_y+1} a_j p_j|| \geq ||y - \sum_{j=0}^{M_y} a_j p_j||.$$

An x_i (y_i) re-entry is effected by

(i) storing $M_x(M_y)$ in the left address of A16,

(ii) storing $\alpha+2$ ($\beta+2$) in the left address of J305E064, and

(iii) transferring control to the left of J305E065 (J305E125)

Re-entries are governed by the following rules:

(i) An entry must precede the first corresponding re-entry.

(ii) $M_x(M_y)$ must increase with each $x_i(y_i)$ re-entry.

(iii) With the exception of A15, A16, and A20, cells A12 - A18 + $3M_x$, BO - BN(M_x+2) - 1, and those cells occupied by $\{a_j\}$ should not be read into between entries.

(iv) $M_y \leq M_x$ must be valid where M_x refers to the last (and largest) value prescribed for an x_i entry or re-entry.

An entry with $M = M_1$ followed by re-entries with $M = M_2$, $\ldots, M_e$, $M_1 < M_2 < \ldots < M_e$, requires essentially the same computing time as a single entry with $M = M_e$.

Method

The x_i entry produces the following results:

(i) Compute $c_0 = - \left(\sum_1^N w_i x_i \right) / \left(\sum_1^N w_i \right)$.

(ii) Set $p_1(x_i) = x_i + c_0$ for $i = 1, \ldots, N$.

(iii) Set $p_0(x_i) = 1$ for $i = 1, \ldots, N$, and set $j = 1$.

(iv) Compute $\{wp_j\}$.

(v) Compute $||p_j||^2$.

(vi) Exit if $j - M_x \geq 0$; otherwise proceed to the next step.

(vii) Compute $d_j = - ||p_j||^2/||p_{j-1}||^2$. (An x_i re-entry begins here.)

(viii) Compute $\{xp_j\}$.

(ix) Compute (xp_j, p_j).

(x) Compute $c_j = - (xp_j, p_j)/||p_j||^2$.

(xi) Compute $\{xp_j + d_j p_{j-1}\}$.

(xii) Compute $\{p_{j+1}\} = \{c_j p_j\} + \{xp_j + d_j p_{j-1}\}$.

(xiii) Increase j by 1, and recycle beginning with step (iv).

The y_i entry produces the following results:

(i) Compute $\{w_i y_i\}$.

(ii) Set $||y - \sum_{j=0}^{M_y} a_j p_j{}^2 = ||y||^2$, and set $j-1 = -1$.

(iii) Compute (y,p_j). (A y_i re-entry begins here.)

(iv) Compute $a_j = (y,p_j)/||p_j||^2$.

(v) Decrease $||y - \sum_{j=0}^{M_y} a_j p_j||^2$ by $a_j(y,p_j)$.

(vi) Increase j-1 by 1. Exit if $(j-1)-M_y \geq 0$. Otherwise, recycle beginning with step (iii).

Notes

(1) J271, J272, and J273 are slaves to J305 and hence must be assigned locations prior to the loading of this program.

(2) The program makes references only to the interpreter of the Floating Point System and to that as region F. Region F must be assigned a location coinciding with the first word of the interpreter prior to the loading of this program.

(3) $||p_0||^2 = \sum_{1}^{N} w_i$

(4) See the J306 write-up for some timing and accuracy samples.

(5) Good results should not be expected unless the number of distinct x_i exceeds M_x.

J305 Storage

Region A: A000 - A018 + $3M_x$

Region B: B000 - B000 + (M+3)N - 1

Program : 141 words (J305E001 - J305E141)

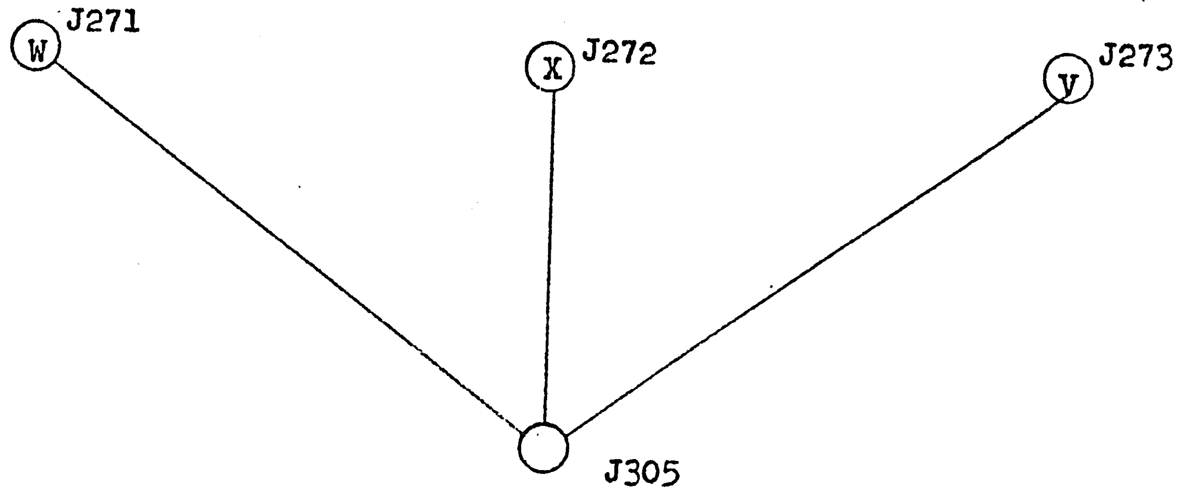
Total Storage Including Slaves - Interpreter Excepted

Region A (the same)

Region B (the same)

Program: 260 words

Lattice Diagram Describing Master-Slave Relations



John I. Derr

J306F ORTHOGONAL POLYNOMIAL LEAST SQUARES WITH WEIGHTS

J306F is an absolute binary assembly of J305E together with J271-3, J290, J292, J240, J242, J244, and a program which loads, computes and prints in a manner to be described. To use J306F form a deck consisting of

(i) one blank card
(ii) J140A
(iii) J135A
(iv) J306F
(v) data
(vi) 2 blank cards

<u>Header Card Format - 013 Style Input:</u>

pos	(1)		
pos	(2)	S	+ if an entry - if a re-entry
		EXP	blank or zero
		MANTISSA	M
pos	(3)	S	+ if x_1 - if y_1
		EXP	blank or zero
		MANTISSA	N
pos	(4)	< 0 and pos	(3) < 0 if the printing of x_1, y_1, $p(x_1)$, r_1, and Σr_1^2 is to be suppressed.
		< 0 and pos	(3) $\geq$ 0 if the sheet eject prior to the printing of the x_1 header card is to be replaced by skipping 4 lines.
		$\geq$ 0	otherwise

pos (5) $\left\{ \begin{array}{l} \text{if } = 0\text{, proceed as in J293F.} \\ \text{if } \neq 0 \text{ and pos (3)} \geq 0\text{, then following the printing of the } x_i \text{ header card load } \{w_i\}\text{, space printer two lines, and print } \{w_i\}\text{, then proceed as in J293F.} \end{array} \right.$

Data Card Format

If pos (2) $\geq$ 0 and pos (3) $\geq$ 0 (pos (3) $<$ 0) for a header card, then N values of x_i (y_i) will be loaded 6/card beginning with pos (1) of the next card. The card containing x_N (y_N) must have a 12 punch in col. 80. Upon loading the latter card an x_i (y_i) entry to J305 occurs with M_x (M_y) = M. If pos (2) $\geq$ 0, pos (3) $\geq$ 0, and, in addition, pos (5) $\neq$ 0, then N values of w_i will be loaded 6/card in the same manner.

Ordering of Header Cards and Data Cards

A set of number pairs (x_i, y_i) is specified by an x_i header card with pos (2) $\geq$ 0 followed by x_i data cards and a y_i header card with pos (2) $\geq$ 0 followed by y_i data cards. The x_i cards must precede the y_i cards. If pos (5) $\neq$ 0 for an x_i header card, then the w_i data cards must be inserted between the header card and the x_i data cards. In the case of re-entries no data cards follow the corresponding header cards. See the J305 write-up for a discussion of re-entries and a many-one correspondence between the sets $\{y_i\}$ and $\{x_i\} \cup \{w_i\}$.

Printing

The page is ejected (the page is spaced four lines) following the reading of each x_i header card if pos (4) $\geq$ 0 (<0). The page is spaced one line following the reading of each y_i header card. In the case of an x_i (y_i) header card the contents of pos (1) - (5) -- pos (1) - (4) -- of the header card are printed immediately thereafter. The page is spaced two lines prior to a y_i entry or re-entry. Following that entry

$$\| y - \sum_{j=0}^{M} a_j p_j \|^2$$ is printed in the x_2 field. Following the calculation of $b_0, b_1, \ldots, b_{M_y}$ the printer is spaced one line and the $\{b_j\}$ are printed 8/line. Then if pos (4) ≥ 0, the printer is spaced two lines, and the program computes

$$p(x_i) = \sum_{j=0}^{M} a_j p_j(x_i), \quad r_i = p(x_i) - y_i, \text{ and } \sum_{k=1}^{i} r_k^2,$$ and prints

$$x_i,\ y_i,\ p(x_i),\ r_i, \text{ and } \sum_{k=1}^{i} r_k^2$$ in positions $x_1, \ldots, x_5$ for

$i = 1, \ldots, N$. An illustrative printing layout follows:

	x_1	x_2	x_3	x_4	x_5	...
	ID	+ 00 3	+ 00 5	+ 00 —	+ 00 1	
two spaces						
	w_1	w_2	w_3	w_4	w_5	
one space	ID	+ 00 3	- 00 5	+ 00 —		
two spaces		$\|y - \sum_{j=0}^{3} a_j p_j\|^2$				
one space	a_0	a_1	a_2	a_3		

	x_1	x_2	x_3	x_4	x_5
two spaces					
	x_1	y_1	$p(x_1)$	r_1	r_1^2
	x_2	y_2	$p(x_2)$	r_2	$r_1^2 + r_2^2$
	x_3	y_3	$p(x_3)$	r_3	$r_1^2 + r_2^2 + r_3^2$
	x_4	y_4	$p(x_4)$	r_4	$r_1^2 + \ldots + r_4^2$
	x_5	y_5	$p(x_5)$	r_5	$r_1^2 + \ldots + r_5^2$

Notes: (1) $M \leq 25$, $N \leq 90$, and $N(M+3) \leq 1952$ must hold.

(2) The number of distinct x_i should exceed M.

Timing and Accuracy

(1) The Legendre polynomials on $[0,1]$ with N = 21 points, 0(.05)1 yielded the following information operating in the SD mode:

M	x_i entry	y_i entry	compute $\{b_j\}$	Print	Accuracy
1	2	1	1	7	9
2	5	3	1	7	8.8
3	8	4	1	8	7.2
4	12	4	2	8	6.8
5	14	5	2	9	6.1

The times in seconds under x_i entry to print inclusive are exclusive of loading times. The accuracy refers to the number of correct SD in the least accurate of the coefficients.

(2) With the duplication of the points 0, .05, .5, .95, and 1 (i.e. N = 26), the following results were obtained assuming entries with M = 5 have occurred previously:

J307A CLEAR THE DRUM

J307 is a one card self-loading routine which clears the 12,288 words of the Johnniac drum to zeros and effectively presses the load button by executing the sequence of instructions.

```
100  0000   101  0000
010  0000
```

HSS is not cleared.

J307 may be modified to preset every word on the drum to any desired bit pattern by punching the 12's row right of J307 with the pattern.

Note: J307 should be placed ahead of J140 if it is necessary to have HSS cleared to zero before loading a program deck.

Requires all of HSS

M. I. Bernstein

J308A OCTAL DRUM DUMP

J308 is a self-loading routine which will print specified portions of a band on the drum in octal. The printed format is band address followed by up to 4 drum words. Uncalled words are not printed. Specification of drum information is made via a control card which must contain a legitimate drum control word (see Johnniac manual if in doubt) in the 9's row left. A control card with 9's row left = 000 0000 000 0000 will terminate dumping.

J308 occupies the first 208 cells of HSS plus the largest block called from the drum - there is no recovery procedure to restore the lost portions of memory.

Each specified block starts a new page of printing headed by the control card information in octal.

The deck is made up as follows:

J308A

control cards

3 blank cards.

It is suggested that J308 not be used as a console routine, but that the user provide a complete deck with control cards inserted instead of sending just control cards to the operator, Program available as J308A.

M. I. Bernstein

J309E CONVERSION FROM GEOGRAPHIC TO UTM COORDINATES

This program converts the latitude and longitude in degrees and minutes of a point on the earth's surface to the UTM zone number and the UTM northing and easting in 100 meters. The point should have a latitude from 20° to 60°, and it must have a zone number between 10 and 19 inclusive. Any point in the continental component of the United States containing Los Angeles satisfies these requirements.

<u>Use</u>: Form a deck of cards as follows:

```
      (i)  one blank card
     (ii)  J140A001
    (iii)┌ X267A001
     (iv)│ J299F001 - J299F097
      (v)└ X267A002 - X267A038
     (vi)┌ J309E001 - J309E070
    (vii)│ J274E001 - J274E026
   (viii)└┌J309E071
     (ix) │J269E001 - J269E010
      (x) └J309E072 - J309E140
     (xi)  Data cards
    (xii)  Two blank cards.
```

Horrors!

Hit load.

<u>Data Card Format</u>:

For each point to be converted punch a card as follows:

59 —> columns 10-11, 22-23, 34-35, 46-47

longitude in degrees ⟶ columns 18-20

longitude in minutes ⟶ columns 31-32

latitude in degrees ⟶ columns 43-44

latitude in minutes ⟶ columns 55-56

ID (floating point) ⟶ position 5 of floating point data form

Print Format:

The sheet is ejected prior to printing for the first point. Following the loading of a data card the information in positions 1-5 of the input data form is printed in positions 1-5 of the print format. On the next line, the following information will be printed:

UTM zone in the two low order digits of the mantissa in position 1;

UTM northing in the five low order digits of the mantissa in position 2;

UTM easting[*] in the four low order digits of the mantissa in position 3;

contents of input position 5 in position 4;

00.000000000 in position 5;

$\Delta\lambda$ in position 6;

$\Delta\phi$ in position 7;

Δq in position 8.

The sheet is spaced one line following the second line of printing for a point.

[*]The easting is augmented by the false easting of 5000.

Punch Format:

Following the printing of the second line for a point, the information in positions 1-4 of the second line of printing is punched in positions 1-4 in the floating point form.

Method:

The method used is described in [1]. A USE program based on the method is being submitted in the near future. See [2]. Let λ denote the longitude in radians and ϕ the latitude in seconds. Let $\Delta\lambda = \lambda_o - \lambda$ and $\Delta\phi = \phi - \phi_o$, where λ_o is the longitude for the central meridian of the zone in question and ϕ_o is 50^o or 30^o (converted to seconds) depending on whether $\phi \geq 40^o$ or $\phi < 40^o$, respectively.

The zone number is computed as $19 - (\lambda - 66)/6$. The zone number is in turn used to pick the corresponding entry in a table which yields λ_o. Compute

$$\Delta q = \sum_{k=1}^{8} A_k (\Delta\phi)^k, \text{ where}$$

A_k is stored in Q000 + k + ϕ_o in the program. Next compute

$$N + iE = a_o + ie_r + \sum_{k=1}^{7} a_k (\Delta q + i\Delta\lambda)^k,$$

where a_k is stored in Uooo + k + ϕ_o in the program, $e_r = 5000$, N = UTM northing in 100 meters, and E = UTM easting in 100 meters.

J310E[2] SOLUTION OF A SYSTEM OF ORDINARY DIFFERENTIAL EQUATIONS - FLOATING POINT[1]

J310 solves a set of differential equations with the following call sequence:

α	020	α	010	β
$\alpha+1$				d
$\alpha+2$	Return (outside floating point control)			

where d is the location of the first word of the auxiliary subroutine.

The following library region bases must be designated before input of this subroutine in the following manner:

Region	Origin	Use
D	a	The numbers in a+i are the variables y_i (i=0, ..., n-1). Originally the initial values are placed here.

The origins of regions E, F, and G are used as preassigned parameters by J310.

E	b-a	The numbers in b+i are the scaled derivatives, hy'_i, calculated by the auxiliary subroutine and and $b > a+n-1$.
F	c-b	Locations c+i are used as temporary storage for this subroutine. These locations must be cleared to zero before this subroutine is entered for the first time. $c > b+n-1$.
G	c+n	n is the number of differential equations to be solved.

1 This routine is a floating point version of J126. For a description of the method see the writeup for J126.

2 This program is J220E as modified for the revised Johnniac Floating Point System.

Region A (erasable): 4 words

Symbolic Code: 45 words

Marvin B. Shapiro

J311E* ORTHOGONAL POLYNOMIAL LEAST SQUARES WITH WEIGHTS

Given a set of points $x_1, \ldots, x_N$, a set of non-negative points $w_1, \ldots, w_N$ satisfying $\sum_{1}^{n} w_i > 0$, and an integer $M_x \geq 1$, the program computes

(i) c_j for $j = 0, 1, \ldots, M_x$ and d_j for $j = 1, \ldots M_x$,

(ii) $p_j(x_i)$ for $1 \leq i \leq N$ and $0 \leq j \leq M_x$, and

(iii) (p_j, p_j) for $j = 0, 1, \ldots, M_x$,

where $(f,g) = \sum_{i=1}^{N} w_i\, f(x_i) g(x_i)$ for any functions f and g defined on $\{x_i\}$, and the $\{p_j\}$ are defined by

$$p_0(x) = 1, p_1(x) = x + c_0, \text{ and for } j \geq 1$$

$$p_{j+1}(x) = (x+c_j)p_j(x) + d_j p_{j-1}(x),$$

and $(p_j, p_j) = 0$ if $j \neq k$.

If, in addition, a set of points $y_1, \ldots, y_N$ is given, the program computes $a_0, a_1, \ldots, a_{M_y}$, provided $M_y \leq M_x$. The $\{a_j\}$ minimize

$$||y - \sum_{j=0}^{M} a_j p_j||^2 = (y - \sum_{j=0}^{M} a_j p_j, y - \sum_{j=0}^{M} a_j p_j), \text{ where}$$

$y(x_i) = y_i$.

<u>Calling Sequence (x_i entry)</u>

α	020	α	010	J311E001
$\alpha+1$	M_x	N	Δ_x	$L(x_1)$
$\alpha+2$			Δ_w	$L(w_1)$
$\alpha+3$	Control returns here			

* J311E is a corrected version of J305 E.

Calling Sequence (y_i entry)

β	020	β	010	J311E106
β+1	M_y	$L(a_o)$	Δ_y	$L(y_1)$
β+2	Control returns here			

The above notation is defined by $L(x_{i+1}) = L(x_i) + \Delta_x$,

$L(w_{i+1}) = L(w_i) + \Delta_w$, $L(y_{i+1}) = L(y_i) + \Delta_y$, and $L(a_{j+1}) = L(a_j) + 1$.

Application: The double entry is motivated by the case where many sets of $\langle y_i \rangle$ correspond to one set of $\langle x_i \rangle$ and $\langle w_i \rangle$.
Following an x_i entry

(i) (p_j, p_j) will be in A18 + 3j for j=0, 1, ..., M_x,

(ii) c_j will be in A19 + 3j for j=0,1,...,M_x,

(iii) d_j will be in A20 + 3j for j=1, ...,M_x, and

(iv) $p_j(x_i)$ will be in B0 + (j+1)N + i - 1 for

$1 \leq i \leq N$ and $0 \leq j \leq M_x$.

A consistent x_i entry must precede the first y_i entry and

$M_y \leq M_x$ must always hold. Following a y_i entry

$$||y - \sum_{j=0}^{M_y} a_j p_j||^2$$ will be in A14.

Let $p(x) = \sum_{j=0}^{M_y} a_j p_j(x)$. Assuming that a double entry

to J311 has occurred and the $\langle a_j \rangle$, $\langle c_j \rangle$, and $\langle d_j \rangle$ are available, p(x) can be readily computed by linking to J291.

Alternatively, let $b_0, b_1, \ldots, b_{M_y}$ denote the coefficients of $p(x)$ with respect to the basis $\langle 1, x, \ldots, x^{M_y} \rangle$, i.e. $$\sum_{j=0}^{M_y} b_j x^j = \sum_{j=0}^{M_y} a_j p_j(x).$$

Then the $\{b_j\}$ can be obtained by successive links to J290 and J292. In case $M_x = M_y$, the following sequence of links is permissible. J311 (x_i entry), J290, J311E(y_i entry), and J292. The first two links are performed for x_i entries and the last two for y_i entries. As a result in the case of a many-one correspondence between the sets $\{y_i\}$ and $\{x_i\} \cup \{w_i\}$, only one link to J311 is required.

An additional flexibility is furnished by a convenient arrangement for re-entries, whereby given a larger value of M the calculation proceeds from where it left off at the smaller value of M. A typical application of re-entries is the following: Suppose we are given the set $\{w_i\}$ and a set of number pairs, $\{(x_i, y_i) : i = 1,\ldots N\}$, and $\varepsilon > 0$, and that we seek the smallest M_y such that

$$||y - \sum_{j=0}^{M_y} a_j p_j|| \leq \varepsilon,$$

or we seek the smallest M_y such that

$$||y - \sum_{j=0}^{M_y+1} a_j p_j|| \geq ||y - \sum_{j=0}^{M_y} a_j p_j||.$$

An x_i (y_i) re-entry is effected by

(i) storing $M_x(M_y)$ in the left address of A16,

(ii) storing $\alpha+2$ ($\beta+2$) in the left address of J311E065, and

(iii) transferring control to the left of J311E066,(J311E126)

Re-entries are governed by the following rules:

(i) An entry must precede the first corresponding re-entry.

(ii) $M_x(M_y)$ must increase with each $x_i(y_i)$ re-entry.

(iii) With the exception of A15, A16, and A20, cells A12 - A18 + $3M_x$, BO - BN(M_x+2) - 1, and those cells occupied by $\left\{ a_j \right\}$ should not be read into between entries.

(iv) $M_y \leq M_x$ must be valid where M_x refers to the last (and largest) value prescribed for an x_i entry or re-entry.

An entry with $M = M_1$ followed by re-entries with $M = M_2, \ldots, M_e$, $M_1 < M_2 < \ldots < M_e$, requires essentially the same computing time as a single entry with $M = M_e$.

Method

The x_i entry produces the following results:

(i) Compute $c_0 = - \left(\sum_1^N w_i x_i \right) / \left(\sum_1^N w_i \right)$.

(ii) Set $p_1(x_i) = x_i + c_0$ for $i = 1, \ldots, N$.

(iii) Set $p_0(x_i) = 1$ for $i = 1, \ldots, N$, and set $j = 1$.

(iv) Compute $\left\{ wp_j \right\}$.

(v) Compute $||p_j||^2$.

(vi) Exit if $j - M_x \geq 0$; otherwise proceed to the next step.

(vii) Compute $d_j = - ||p_j||^2 / ||p_{j-1}||^2$. (An x_i re-entry begins here.)

(viii) Compute $\left\{ xp_j \right\}$.

(ix) Compute (xp_j, p_j).

(x) Compute $c_j = - (xp_j, p_j) / ||p_j||^2$.

(xi) Compute $\left\{ xp_j + d_j p_{j-1} \right\}$.

(xii) Compute $\{p_{j+1}\} = \{c_j p_j\} + \{x p_j + d_j p_{j-1}\}$.

(xiii) Increase j by 1, and recycle beginning with step (iv).

The y_i entry produces the following results:

(i) Compute $\{w_i y_i\}$.

(ii) Set $||y - \sum_{j=0}^{M_y} a_j p_j{}^2|| = ||y||^2$, and set $j-1 = -1$.

(iii) Compute (y,p_j). (A y_i re-entry begins here.)

(iv) Compute $a_j = (y,p_j)/||p_j||^2$.

(v) Decrease $||y - \sum_{j=0}^{M_y} a_j p_j||^2$ by $a_j(y,p_j)$.

(vi) Increase j-1 by 1. Exit if $(j-1)-M_y \geq 0$. Otherwise, recycle beginning with step (iii).

Notes

(1) J271, J272, and J273 are slaves to J311 and hence must be assigned locations prior to the loading of this program.

(2) The program makes references only to the interpreter of the Floating Point System and to that as region F. Region F must be assigned a location coinciding with the first word of the interpreter prior to the loading of this program.

(3) $||p_0||^2 = \sum_{1}^{N} w_i$

(4) See the J312 write-up for some timing and accuracy samples.

(5) Good results should not be expected unless the number of distinct x_i exceeds M_x.

J305 Storage

Region A: A000 - A018 + $3M_x$

Region B: B000 - B000 + (M+3)N - 1

Program : 142 words (J311E001 - J311E142)

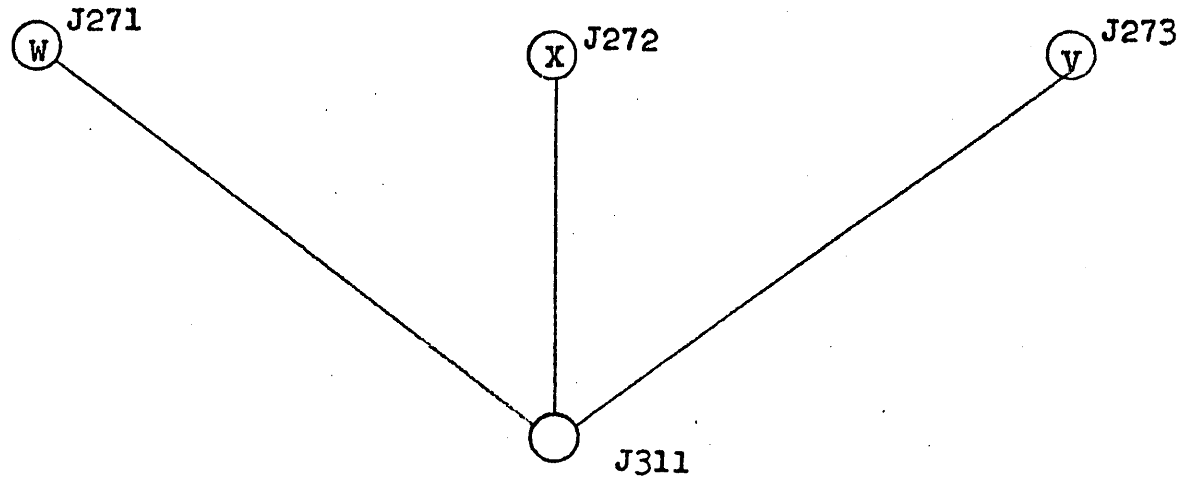
Total Storage Including Slaves - Interpreter Excepted

Region A (the same)

Region B (the same)

Program: 261 words

Lattice Diagram Describing Master-Slave Relations



John I. Derr

J312F ORTHOGONAL POLYNOMIAL LEAST SQUARES WITH WEIGHTS *

J312F is an absolute binary assembly of J311E together with J271-3, J290, J292, J240, J242, J244, and a program which loads, computes and prints in a manner to be described. To use J312F form a deck consisting of

(i) one blank card
(ii) J140A
(iii) J135A
(iv) J312F
(v) data
(vi) 2 blank cards

<u>Header Card Format - 013 Style Input:</u>

pos	(1)		
pos	(2)	S	+ if an entry - if a re-entry
		EXP	blank or zero
		MANTISSA	M
pos	(3)	S	+ if x_i - if y_i
		EXP	blank or zero
		MANTISSA	N
pos	(4)	< 0 and pos (3) < 0	if the printing of x_i, y_i, $p(x_i)$, r_i, and $\sum r_i^2$ is to be suppressed.
		< 0 and pos (3) ≥ 0	if the sheet eject prior to the printing of the x_i header card is to be replaced by skipping 4 lines.
		≥ 0	otherwise

* J312F is a corrected version of J306F.

pos (5) $\left\{ \begin{array}{l} \text{if} = 0\text{, proceed as in J293F.} \\ \text{if} \neq 0 \text{ and pos (3)} \geq 0\text{, then following} \\ \text{the printing of the } x_i \text{ header card} \\ \text{load } \{w_i\}\text{, space printer two lines,} \\ \text{and print } \{w_i\}\text{, then proceed as in} \\ \text{J293F.} \end{array} \right.$

Data Card Format

If pos (2) ≥ 0 and pos (3) ≥ 0 (pos (3) < 0) for a header card, then N values of x_i (y_i) will be loaded 6/card beginning with pos (1) of the next card. The card containing x_N (y_N) must have a 12 punch in col. 80. Upon loading the latter card an x_i (y_i) entry to J311 occurs with M_x (M_y) = M. If pos (2) ≥ 0, pos (3) ≥ 0, and, in addition, pos (5) $\neq 0$, then N values of w_i will be loaded 6/card in the same manner.

Ordering of Header Cards and Data Cards

A set of number pairs (x_i, y_i) is specified by an x_i header card with pos (2) ≥ 0 followed by x_i data cards and a y_i header card with pos (2) ≥ 0 followed by y_i data cards. The x_i cards must precede the y_i cards. If pos (5) $\neq 0$ for an x_i header card, then the w_i data cards must be inserted between the header card and the x_i data cards. In the case of re-entries no data cards follow the corresponding header cards. See the J311 write-up for a discussion of re-entries and a many-one correspondence between the sets $\{y_i\}$ and $\{x_i\} \cup \{w_i\}$.

Printing

The page is ejected (the page is spaced four lines) following the reading of each x_i header card if pos (4) ≥ 0 (<0). The page is spaced one line following the reading of each y_i header card. In the case of an x_i (y_i) header card the contents of pos (1) - (5) -- pos (1) - (4) -- of the header card are printed immediately thereafter. The page is spaced two lines prior to a y_i entry or re-entry. Following that entry

$|| y - \sum_{j=0}^{M} a_j p_j ||^2$ is printed in the x_2 field. Following the calculation of $b_0, b_1, \ldots, b_{M_y}$ the printer is spaced one line and the $\left\{ b_j \right\}$ are printed 8/line. Then if pos (4) ≥ 0, the printer is spaced two lines, and the program computes

$p(x_i) = \sum_{j=0}^{M} a_j p_j(x_i)$, $r_i = p(x_i) - y_i$, and $\sum_{k=1}^{i} r_k^2$, and prints

x_i, y_i, $p(x_i)$, r_i, and $\sum_{k=1}^{i} r_k^2$ in positions $x_1, \ldots, x_5$ for $i = 1, \ldots, N$. An illustrative printing layout follows:

	x_1	x_2	x_3	x_4	x_5	...
two spaces	ID	+ 00 3	+ 00 5	+ 00 —	+ 00 1	
one space	w_1	w_2	w_3	w_4	w_5	
	ID	+ 00 3	- 00 5	+ 00 —		
two spaces						
		$\|\|y - \sum_{j=0}^{3} a_j p_j\|\|^2$				
one space						
	a_0	a_1	a_2	a_3		
two spaces						

x_1	x_2	x_3	x_4	x_5
x_1	y_1	$p(x_1)$	r_1	r_1^2
x_2	y_2	$p(x_2)$	r_2	$r_1^2 + r_2^2$
x_3	y_3	$p(x_3)$	r_3	$r_1^2 + r_2^2 + r_3^2$
x_4	y_4	$p(x_4)$	r_4	$r_1^2 + \ldots + r_4^2$
x_5	y_5	$p(x_5)$	r_5	$r_1^2 + \ldots + r_5^2$

Notes: (1) $M \leq 25$, $N \leq 90$, and $N(M+3) \leq 1952$ must hold.

(2) The number of distinct x_1 should exceed M.

Timing and Accuracy

(1) The Legendre polynomials on $[0,1]$ with N = 21 points, 0(.05)1,yielded the following information operating in the SD mode:

M	x_i entry	y_i entry	compute $\{b_j\}$	Print	Accuracy
1	2	1	1	7	9
2	5	3	1	7	8.8
3	8	4	1	8	7.2
4	12	4	2	8	6.8
5	14	5	2	9	6.1

The times in seconds under x_i entry to print inclusive are exclusive of loading times. The accuracy refers to the number of correct SD in the least accurate of the coefficients.

(2) With the duplication of the points 0, .05, .5, .95, and 1 (i.e. N = 26), the following results were obtained assuming entries with M = 5 have occurred previously:

M	x_i re-entry	y_i re-entry	compute b_j	Print
10	18	4	5	14
15	18	4	9	18
20	18	4	16	22
25	18	4	25	25

Program: 103 cards (J312F001 - J312F103).

John I. Derr

J. C. Shaw

J-313A Variable Format Input-Output Package

J-313A is a routine for providing flexible format of card input and card output and makes use of X-267 (JOHNNIAC Style E Loader) as a self-contained subroutine. It makes use of format cards to define the fields on the input and output cards.

There are three types of format cards, each format being identified by the condition of column one as follows:

FORMAT CARD	COL. ONE
Read	No identifying punch
Punch	"4"
Blank-out	"0"

A field is defined by punching a "1" in each column of the field. The low order position of the field must be underpunched with a "2" for both punch and read format cards.

It should be noted that field definition does not interfere with the identification used in column one. Following are a few examples:

```
READ FORMAT                   COLS.:    1 2 3 4 5 6 . . .

Example I        punch:                 1 1 1
                                            2

Example II       punch:                 1 1 1 1 1
                                        2 2     2

Example III      punch:                     1 1
                                              2

PUNCH FORMAT

Example IV       punch:                 1     1 1 1
                                        2         2
                                        4

Example V        punch:                 4 1 1 1 1
                                                2
```

```
BLANK-OUT FORMAT                  COLS.:   1 2 3 4 5 6 . . .

Example VI         punch:                  0
                                           1 1     1

Example VII        punch:                  0   1 1
```

The programmer writes his complete program using JOHNNIAC Style E instructions, some of which will be short calling sequences to instruct J-313 to read, print, or punch according to the prepared format cards. (Such a short calling sequence might look like the following:

```
α       020 $  0 010 Z 10
α + 1   control returns
```

The use of such subroutines will be explained later.) His complete program should be placed between card 122 and 123 of J-313. Card 122 sets the counter ($) to 2000)$_8$, and card 123 is a period (.) card (therefore the regular period card should be omitted), and transfers control to that section of J-313 which will read the format cards. Format cards (placed in any order) must follow card 123. The last format card must be followed by a card containing an 8-punch in column one. This card causes a transfer of control to location 2000)$_8$ which is the beginning of the program, however, if H-1 is on, a 13x stop at NIA = 6504)$_8$ will occur before this transfer is made. Hit GO to continue.

The loader routine contained in J-313 is stored in the region 0000)$_8$ through 1127)$_8$ making it impossible to use this region until after loading is complete. J-313 uses this region and more (up to 1777)$_8$) for the compiling of format card subroutines and the conversion of input/output data which takes place after loading. Generally then, it is necessary that the programmer not attempt to use portions of memory below 2000)$_8$.

Flexibility in operation of the program is under control of the T-switches as follows:

T-1	ON	1) One data card read; one data card punched
		2) Words available for use by the programmer are from $2000)_8$ to $6107)_8$.
	OFF	1) Fifty data cards read; fifty data cards punched.
		2) Words available for use by the programmer are from $2000)_8$ to $3407)_8$.
T-2	ON	80-80 printout instead of punching
	OFF	- - -

It might be pointed out that when using T-1 off, the routine is expecting 50 data cards to be available for reading and then after computing on these fifty cards, it will punch 50 cards. If the punch block of fifty cards is not filled up (such as when there are 95 data cards), then some action must be taken such as filling up the reader with cards of the type which the programmer knows will eventually cause punching, or to cause dummy numbers to be generated. For more information on this point, see the explanation of the punch format on an X-72 card at the end of this write-up.

J-313 interprets the format cards and from each one compiles a series of calling sequences into a subroutine. Each card is numbered sequentially (internally, only) by J-313 beginning with 1; the number will correspond to the number of the subroutine since exactly one subroutine is generated for each format card, and it is generated in the order in which its corresponding format card is placed in the program deck.

Placement in the deck, however, in no way governs the order in which they may be referenced; any format card may be referenced at will.

A directory to these subroutines is formed and assigned region "Z. "Z 1" is the directory number for the subroutine corresponding to the first format card, "Z 2" the directory number for the subroutine corresponding to the second format card, etc. "Z 0" is the directory number for the subroutine which causes a card to be read, and puts the information from the card in position for conversion according to one of the format cards and corresponding calling sequence in the "Z" region to be determined by the programmer.

There are some limitations on the number of format cards, and since no check is made to determine if available space is exceeded, some approximate guides are provided here, and generally must be considered together for each problem in determining if available space is exceeded. If computation indicates the limit is being reached, the programmer should make some independent checks for himself.

1. Number of fields 427
2. Number of format cards 31
3. In addition to these maximum limitations, the words available for the calling sequences compiled as a result of each format card are from $1000)_8$ through $1777)_8$. If a = the number of defined fields on a <u>read</u> or <u>punch</u> format card, and b = the number of undefined fields on a <u>read</u> or <u>punch</u> format card, then the number of words (x) required by a <u>read</u> or <u>punch</u> format card would be = a + b + 5.
4. The number of words (y) required by a <u>blank-out</u> format card is 4.

The sum of the X's for the read or punch format cards added to the sum of the Y's for the blank-out format cards must therefore be less than or equal to 512.

The fields of the read and punch format cards are also numbered (internally, only) beginning with 1 for the first field of the first format card, through "n" for the last field of the last format card. Blank-out format cards are not included in this numbering.

Thereafter, whenever the subroutine for a read format card is "fired", the contents of each field is converted into a binary integer and placed in memory at "G 0 + the number of the field." For a punch format card, each binary number in memory at "G 0 + the number of the field" is converted into a decimal integer and placed in the field. Leading zeros are included and all information previously in the field is cleared. This process essentially makes the JOHNNIAC a decimal machine. Ten decimal digits are usually the maximum for any field. Information in a field containing an 11-punch over the low-order digit is converted into a negative number and, conversely, when a negative number is punched out it contains an 11-punch over the low-order digit. A field containing a 12-punch over the low-order digit will be converted as a positive integer; in the absence of either an 11-punch or 12-punch, the field will be converted as a positive integer.

All punching of cards except the last is echo checked. If an error is detected, a 13X stop at NIA = $7330)_8$ will occur with the error card on top in the punch stacker. This means that the error indication does not occur until after two more cards have been punched. To get full echo checking and indication at least two blank cards should be punched after the last good card. After an echo check error stop, hit GO to continue.

The following regions are defined by J-313.

1. G 0 = $0124)_8$ (Maximum is G427 = $0777)_8$)

   The contents of the fields on the format cards are read into and punched (printed) from this G region.

2. X 0 = $0000)_8$ (Maximum is X 80 = $0120)_8$)

   This X region has a cell for each of the 80 columns (X 16 for column 16) of an input card. A cell in this region will be non-zero if the corresponding column contains either an 11-punch or a 12-punch; zero if the corresponding column does not contain either an 11-punch or a 12-punch. Punching data cards with 11 or 12 punches in particular columns can give the programmer "flags" in this X region to enable him to choose in his code the appropriate format card for reading or punching (see example at end of write-up where X 72 determines format card).

3. K 0 = $7720)_8$ (Maximum is K 47 = $7777)_8$)

   The K region is the location of constants used by J-313. The following are listed for what use they may be to the programmer:

```
K  1 000 0001 000 0001
K 14 000 0000 000 0001
K 20 000 0000 000 0010
K 21 100 0000 000 0000
K 22 077,7777 777,7777
K 29 000 0001 000 0000
```

4. Z 0 = $7640)_8$ (Maximum is Z 31 = $7677)_8$)

   The Z region is the directory to the format card subroutines. The following instruction word will always cause a card to be read, and prepared for conversion according to one of the read format cards:

```
020 $  0 010 Z  0
```

5. B 0 = $7700)_8$ (Maximum is B 6 = $7706)_8$)
(See note below.)

6. C 0 = $7710)_8$ (Maximum is C 6 = $7716)_8$)
(See note below.)

NOTE: The B and C regions noted above are the regions where the Mouse Matrix is located for those who are familiar with the Mouse Input and Output routines. The instruction in the description of the Z region above is essentially a calling sequence to the subroutine which causes a card to be read into Mouse Matrix (J160).

An example:

Suppose that there are five format cards as indicated below:

```
      0000000001111111111222222222233333333334444444----77778
      1234567890123456789012345678901234567890123456----67890

1    /    11111  111111111        111  11111111          1111
              2  2       2          2        22             2

2    /    11     11111           1111111  111111111       1
           2         2                 2    2     2       2

3    /0     1111     11111111111       11          11----1 111

4    /1111111     11111111   11   1111111111   1111   --11111
           2            2    2      2     2      2         2
      4

5    /0      11111        111  111          111    1--
```

The first two cards are read format cards. The third and fifth are blank-out format cards and the fourth is a punch format card. Also suppose that if a card contains an 11-punch in column 72 that it should be converted by using format card 1 and that if it does not contain an 11-punch in column 72 it should be converted by using format card 2.

The following instruction causes a card to be read:

```
α        020 $  0 010 Z  0
α + 1    control returns
```

The following instructions determine whether there is an 11-punch in column 72 and convert it accordingly:

```
α        023 X 72 001 α + 3
α + 1    020 $  0 010 Z   2
α + 2    010 α + 4
α + 3    020 $  0 010 Z   1
α + 4    control returns
```

If the Z 1 subroutine was fired, fields 1 thru 7 of the card would be converted into binary and placed in G 1 thru G 7 respectively. If the Z 2 subroutine was fired, fields 8 thru 13 of the card would be converted into binary and placed in G 8 thru G 13 respectively.

The following two instructions would cause fields 14 thru 20 to be cleared, would cause the numbers in G 14 thru G 20 to be converted into decimal and placed in the respective fields, would cause blanks to be placed in all other columns, and would cause a card to be punched:

```
α       020 $  0 010 Z  5      (blank-out)
α  + 1  020 $  0 010 Z  4      (convert and punch)
α  + 2  control returns
```

Bill Sibley
Chuck Smith

J315A J'ACASS II' (JOHNNIAC ASSEMBLER)

INTRODUCTION

This write-up contains the description of a coding language for the Johnniac and the associated assembly conventions. It is not intended to be a description of the Johnniac or an instruction manual on coding or programming. For a description of the Johnniac see DL3058 and D3518.

Though J'ACASS does not contain all of the features and capabilities that the current state of the programming art is capable of generating, it is felt to be a significant improvement over the systems currently available on the Johnniac.

J'ACASS is a two pass, load and go "symbolic" assembler. Briefly, pass one of the assembly process reads each block of a program and creates a label table, and stores the lines of code for pass two. Pass two, transforms each line of code from its symbolic form into absolute binary and stores it on the drum for subsequent loading and execution. Concurrently, pass two generates a program listing, displaying on each line the absolute and symbolic code, and an absolute **style D binary deck of** the block. J'ACASS will produce: 1) a reloadable label table (see p. 13); 2) an absolute deck for each program block in style D; 3) a label map for each block; 4) a full assembly listing; and 5) a J'AM'D version of any set of symbolic cards specified (see pg. 15). The assembler will accept as inputs: 1) "symbolic" instructions; 2) J'AM'D decks (see pg. 15); 3) previously produced label tables (see (1) above); and 4) absolute binary decks in styles B, D and F.

Blocks and Labels

Certain capabilities have been built into J'ACASS which are intended to facilitate the combining of programs. Of prime importance in this respect is the ability to create program blocks with local labels. One may ignore the block structuring capabilities of the system with small, but possibly important, penalties.

A block is a set of consecutive lines of code, any line belonging to, at most, one block. A label is an identifier for a line of code. It provides a means for referring to the line in an address field. (What is called a "label" here is often called a "symbol" or "location symbol" in other assemblers.)

A program may be divided into various sized, semi-independent blocks, each block containing one or more routines or data sets. Each of the blocks is of one of three types depending upon how it is related to other blocks; it may be: 1) always referred to but never referring; 2) always referring but never referred to; and 3) referred to and referring. Type (1) blocks are typically those routines called "closed subroutines" and data blocks. The type (2) category can be characterized by the master routine which directs or controls the activity of a program. Type (3) includes the rest of the multitude of routines.

Within blocks of type (1) there are two kinds of labels: those that are used within the block only, and those that are associated with external reference points (enabling reference from outside to the routines in the block). The two kinds of labels are called "local" and "global." Local labels have significance within a block only, and global labels have meaning or signifcance beyond the boundaries of the block. In type (2) blocks, then, all labels may be local, and in type (3) there are both local and global labels.

One may divide the routines of a program into the three kinds of blocks referred to above. He must then order the blocks so that any one of them is assembled before it is referred to by another. Generally speaking, those of type (1) must be assembled first, those of type (3) next, and, finally, those of type (2).

One bit of care must be taken in structuring the routines of a program into blocks of type (2) -- all of those routines whose addresses refer to each other must appear in the same block, and all of the non-global labels in that block must be local to the whole block (not to one routine) and, therefore, unique. The meaning of this will become clearer after the presentation of the method by which the assembler actually handles local and global labels.

J'ACASS Card Format

(See Appendix A for a sample coding form.)

C_{1-4}	Label (C_1 designates a comment card if it contains an asterisk).
C_5	Global label designator (*).
C_{6-8}	Left operation.
C_9	Listing format control (an asterisk indicates a new page), and indirect address field for Left address (indicated by a prime).
C_{10-17}	Left address.
C_{19-21}	Right operation.
C_{22}	Indirect Address field for Right Address (indicated by a prime).
C_{23-30}	Right address.
C_{31-80}	Comments
C_{10-30}	Data field for certain pseudo-ops.

The Character Set

J'ACASS uses the character set common to all RAND keypunches. This set consists of the characters: 0 through 9, A through Z + - * / $ = ' . , ( ). The last eleven symbols will be called "punctuation marks." Thus we have ten numerals, twenty-six capital letters, and eleven punctuation marks.

Labels

With few exceptions, any line of code may have a label. A label may be from one to four characters long. The first character of a label must be a letter or one of the following six punctuation marks: . , /) ' =. Successive characters of a label may be numerals, letters, or punctuation marks with the exception of "+" and "-".

Global labels are designated by an asterisk in column 5 of the coding line, no other distinction between local and global labels is made. When a global label is used in an address field, the asterisk is dropped.

Operations

Numbers appearing in the left and right operation fields are treated as octal (each digit is taken mod 8). Zero and blank are interchangeable in the operation fields, but in no other place.

A complete set of mnemonics (see Appendix B) for the Johnniac machine instructions are provided. Those specified in the Johnniac manual are included as a subset with one exception: the mnemonic for the shift command 074 is given in the Johnniac manual as SRH; this mnemonic has been usurped for another command, namely 057.

Addresses

Left and right addresses are composed as follows:

1. A blank field which is equivalent to zero.

2. A string of from one to eight characters, the first of which is an asterisk is also equivalent to zero.

3. A decimal integer (signed or unsigned) taken mod 2^{12}. The absence of a sign is equivalent to the presence of a plus sign; a minus sign indicates the 2's complement.

4. An octal integer (signed or unsigned) written as follows: a sign (if desired), a left paren, the digits of the number (each of which is considered mod 8), and a terminating blank or non-numeral. A right paren is aesthetically pleasing but not required.

5. A label.

6. A label followed by a signed decimal or octal constant.

7. The dollar sign ($), which designates the current value of the location counter.

8. The dollar sign followed by a signed decimal or octal constant, which designates the appropriately incremented (or decremented) value of the location counter.

No checks are made to determine the appropriateness of a particular address for the given operation. An address may begin in any column within the address field and is considered ended by either a blank or the right limit of the field.

Indirect Addresses

The assembler will provide the appropriate indirect address flag bit for either address field when it is flagged with a prime (or single quote mark) in the column between the op field and the address field--column 9 for the left address and column 22 for the right address.

Pseudo-Ops

There are two uses for pseudo-ops. The first is to specify constants for a routine; the other is to control the assembling process. All pseudo-ops must appear in the left operation field of an instruction.

There are three data specifiers, NUM, DEC, BCI, and BCT that function as follows:

<u>NUM</u> or <u>DEC</u> are used to indicate that the characters in the Data Field constitute a numeric constant. The constant is converted by the assembler into one of three forms: to fixed point binary, or to floating point decimal compatible with the JOSS programming system, or to octal.

Fixed point conversion occurs when the Data Field contains either a signed or unsigned integer or a signed or unsigned mixed number, followed by an appropriate binary scale. A member of the first type is written as a string of digits in the Data Field with no imbedded blanks. It is converted to a binary integer scaled at B=39. A mixed number is written in the Data Field as a string of digits, with a decimal point in the appropriate place, followed immediately by the letter "B" followed by a (signed or unsigned) decimal integer. Imbedded blanks cause errors. A positive scale indicates positioning of the number to the right of the Johnniac sign position. Thus, "1B2" in the instruction "DEC 1B2" appears as 001000...0 in the output program. Improper scaling in the sense of not allowing a large enough number of bits to accommodate the number (resulting in an overflow) causes an error message to be printed and the result of the "DEC" to be set to 0.

Floating decimal numbers may be written in one of two ways. Any mixed number (that is a string of digits and one decimal point) without a binary scale specified is converted to floating decimal. Any number followed by a decimal scale designated by an "E" followed by a signed or unsigned integer whose absolute value is less than 99 is also converted to floating decimal. Imbedded blanks cause errors. If the resulting power of the floating decimal number is outside the range $-99 \leq P \leq 99$, the number will be set to zero and an error message printed. In both cases the binary representation is truncated, not rounded.

Octal is distinguished from decimal by punctuation - namely, enclosing the number in the data field in parenthesis. Each digit within the parenthesis is taken mod 8, with the exception of the high order digit if there are fourteen digits in the number. Blanks may be included (and will be ignored) in order to separate subfields of an octal constant. If the constant contains less than fourteen digits, it will be converted as an integer scaled at 2^{-39}. A minus sign to the left of the opening parenthesis will indicate that the 2's compliment of the octal number is required.

<u>BCI</u> is used to input binary coded Hollerith characters as they are converted by J'ACASS (see Appendix C). The argument to this instruction must be in the <u>first</u> six columns of the left address field. The characters in these

columns are converted and the result occupies bits 4 through 39 of a machine word. The first two columns of the Right Operation Field are taken to form a four bit prefix placed in bits 0-3 of the word. The prefix is formed by taking the first character mod 2 and appending to it the second character taken mod 8.

BCT is similar to BCI. It converts the first four characters of the left address, placing the result in bits 4 through 27 of a machine word. A prefix taken from the first two columns of the Right Operation Field (in the same manner as described for BCI) is placed in bits 0-3 of the word. An address from the Right Address Field is placed in bits 28-39. The right address may contain any legitimate address combination.

The pseudo-ops which control the assembly function, and some of the assembly processes, will now be discussed.

SET is used to control the value of the location counter. When this instruction has a label, that label is defined to have the value of the location counter prior to its modification by the SET instruction. The counter is set according to the left address of the instruction. The left address may be any legitimate address; however, using as the address a yet to be encountered label will cause the location counter to be set to zero and will create problems.

The SET instruction can be used to reserve a block of storage by designating an increment for the location counter in the Left Address Field. An instruction to do this would have the form "L SET $ + n", where "L" is any label and n is an integer of appropriate value.

<u>EQU</u> is used to assign a value to a label. The value to be assigned is designated in the Left Address Field of the instruction. Any legitimate address may be used. Thus a label may be assigned the value of (or declared synonymous with) a constant, another label, etc. If the dollar sign is used as a part of the address of an EQU instruction, its value is taken not as that of the location counter where the EQU appeared, but rather as that which the location counter will have at the end of the block.

An EQU may have as its address any other label in the block or any retained global label, without care as to the position of occurrence of the label with respect to the EQU. The exception to this is that forward reference must not be made to a label defined by an EQU.

<u>END</u> terminates each block of code. When the assembler encounters an END, the following sequence of events takes place: On pass 1 the location counter (which has a value one greater than it had after the previous line of code) is saved; the label table is sorted; EQU's are processed and their labels added to the label table; the label table is searched for duplicates and if any are found, they are printed with the error message "AMBIGUOUS LABEL"; if label output is not suppressed, a label map is printed and the

global labels contributed to the table by this block are punched. Control then goes to pass 2.

When the END is again encountered on pass 2, the value of the location counter is compared to the value saved from pass 1. If they are different, processing stops and the the message "J'ACASS HAS ERRED" is printed. Otherwise, all local labels are purged from the label table, all conditions are set to normal, and reading of the next block begins.

*** Control Cards

***** is a special pseudo-op used to specify various other assembly control functions. The name of the function appears in the left address and the first five characters of the name must be as they are given below (everything over five is for understandability only). This, by the way, is the only instruction for which the label field is ignored. It, therefore, should not have a label that is referred to in the program.

The instructions in this set are divided into two groups: conditioning and action instructions.

The conditioning instructions are primarily used to control the output of the assembler and may appear anywhere within the block for which they are to have meaning. They are written below as they should appear in a program.

*** **NØLIST** instructs the assembler not to produce an assembly listing; it will not inhibit the printing of error messages.

*** <u>NΦPUNCH</u> inhibits the assembler from punching both the global labels for the block and the absolute style D binaries.

*** <u>NΦLABELS</u> inhibits the printing of the label map and the punching of global labels from the block. It is not redundant if <u>NΦLIST</u> has been specified since the label map is printed independently of the assembly listing.

The instructions which specify actions are of two kinds, those which act upon the label table and those which control other activities of the assembler. The label table instructions must be used with care. They are:

*** <u>DELETE</u> L, where "L" is a global label left in the label table from a prior block. "L" must appear in the right address. If there is no label "L" in the label table, the instruction is ignored. DELETE must appear before any instructions which have legitimate labels.

*** <u>ERASE</u> clears the label table of all global labels remaining from prior blocks. It should be used with caution, because it is capable of obliterating communication with prior blocks. Like DELETE, ERASE should occur in the block prior to any legitimately labeled instruction--in this case because any labels which appeared ahead of it would also be wiped out.

*** <u>SAVE</u> is really a conditioning instruction, but is concerned only with the label table. It instructs the assembler to make a copy of the label table (including local labels) as it appears just prior to the execution of pass 2 of the assembler. The last label table saved is available to the programmer at execution time of his own program. The table is on the drum at band 1, position 3, from location 1 through n. Band 1, position 3, location 0 contains n in both the left and right address fields of the word.

*** <u>RESTΦRE</u> is executed whenever it occurs. It restores the last saved copy of the label table and wipes out any existing label table. Note then that a RESTΦRE without a SAVE in a prior block is the same as an ERASE. Also note that since all labels are saved, that the RESTΦRE allows reference to the local labels of the prior block. There exists some danger of creating ambiguities this way - so let the programmer beware.

The other action instructions specify specific tasks for the assembler.

*** <u>LΦAD</u> informs the assembler that the immediately following cards are binaries (possibly produced by the assembler) to be loaded unmodified. Any global labels associated with the deck (produced by the assembler) are added to the label table. The program deck is in style D binaries. The format for a binary label card is a negative 9's row left, a blank 9's row right and labels (in BDT internal form) in sucessive words until either there are no more or the card is full. There may be as many label cards as are needed with a binary deck. Reading is terminated by a card with a blank 9's row and control returns to pass 1 of the assembler to continue reading "symbolic" instructions. The location counter is not affected by the LΦAD instructions.

*** <u>LΦADB</u> is exactly like LΦAD except that the program deck is assumed to be in style B <u>absolute</u> binaries. All other things are the same. Style B binaries are produced by the various versions of the J100 assembler, the latest of which is J224.

*** <u>LΦADF</u> is, again, exactly like LΦAD, except that it expects the program deck to be in style F binaries. Style F binaries are produced by J267.

The above three instructions permit previously assembled routines to be loaded with a symbolic program. Unfortunately, the routines are in absolute and cannot be re-arranged in storage.

*** <u>STΦP</u> informs the assembler to terminate activity <u>now</u>. It causes a program stop with NIA equal to 1751. In order to complete the assembly of a block, then, the STΦP must occur after the END for the block. Any instructions occurring after the STΦP will not be read, and any instructions in the same block read prior to the STΦP will not be assembled.

*** <u>GΦ</u> carries the same conditions of location as STΦP. It tells the assembler to load the assembled program (from the drum to core) and execute it beginning with the left instruction at the location specified by the right address (which may be any legitimate address). If <u>any</u> errors have been detected during any part of the assembly, the program will not be loaded, the message

"ERRØRS PREVENT EXECUTIØN" will be printed, and the computer will stop.

*** <u>GØGØ</u> is the imperative form of GØ which says to ignore any errors and attempt the execution anyway.

*** <u>TRANCARD</u> <u>0</u> <u>A</u> produces a binary transfer and compatibility with all style D loaders containing the instruction of the right operation and the address in the right address (which may be any legitimate address). This instruction should normally appear prior to the GØ or STØP but outside the END of the last block.

*** <u>START</u> and *** <u>FINISH</u> are in essence a pair of pseudo-brackets used to surround a block of "symbolic" code to indicate that all those lines included should be output as a <u>J'AM'D</u> deck. One should <u>not</u> include any of the *** pseudo-ops or SET instructions with other than addresses relative to the current value of the location counter ($ ± numeric constant).

<u>J'AM'D Decks</u>

The J'AM'D deck produced by the START - FINISH pair of instructions retains its full "symbolic" flexibility having as its primary advantage compactness thus saving card reading time. A J'AM'D deck <u>cannot</u> be modified.

A J'AM'D deck consists of up to three parts; the first and second parts do not necessarily appear. Each part is preceded by an identification card* with the identifier contained in the 9's row left in the BCD. The first part (if present) is the EQU's. The 9's row left contains the word "EQU's" and the 9 right word contains the count in both left and right address. Succeeding cards contain the EQU's. The next part contains levels; the identification card contains "LABELS" and a word count and is followed by a deck of labels contained in the text. The last part consists of the text which was enclosed in the pseudo-brackets. The identifier is "TEXT" without the double word count. The text is punched in BCI with trailing blanks eliminated from each line but including comments. An extra word containing all 1's is added at the end of the text for identification. An extra card is punched at the end with the BCI word "TH'END" and a single word count in 9's right address.

When a J'AM'D deck is punched, it precedes the label and style D binary decks punched for the block in which the request is made. Note that more than one J'AM'D deck may be produced within one block. Also note that it is up to the user to separate out the J'AM'D decks from the label and style D binary decks produced from a program before attempting to reload any of the decks.

*** <u>READ</u> is the pseudo-op which must precede <u>each</u> J'AM'D deck when it is to be read <u>in place of</u> the symbolic

*Identification cards are coded with punches in the 9's row of cols 1,2,3.

binaries from which it was produced. J'AM'D decks and symbolic decks may be combined and intermixed--the only restriction being that each J'AM'D deck be preceded by a ***READ card. Remember, there is no way to modify a J'AM'D deck.

Comments

Comments may be appended to a symbolic program in two ways. First, they may be written in the "Comments" field to the right of any instruction. Second, they may fill the entire line, if column 1 contains an asterisk. The assembler ignores all cards that contain an asterisk in column 1, and it deletes the comments appearing in column 31 on from the assembly listing.

Assembly Listing Format

When the instruction *** NΦLIST has not occurred and the instruction *** NΦLABELS occurred, the label map is not printed. If a *** NΦPUNCH instruction is given, the listing appears as a continuous block with double spaces surrounding the instructions SET and END. If binaries are punched, the assembly listing is blocked (separated by blank lines) according to the instructions punched on each card.

A given instruction can be made to begin on a new page by placing an asterisk in column 9 of the card--that is between the left operation field and the left address field. This does not affect the arrangement of the binary deck.

Restrictions and Limitations

1) The number of lines of code contained in a block may vary from 300 to 2000 depending upon the density or average line length in six character groups.

2) No block may generate a label table containing more than 511 labels, global and local, including those retained from prior blocks.

3) No block may contain more than 100 EQU's.

The assembled program is assumed to be able to reside in the 4096 word core storage of the Johnniac. No check is made for the number of lines of code assembled or for overlapping routines. On the other hand, there is no sacred place in the core storage that the programmer must reserve unless he plans to load the absolute program with one of the available loaders, which must have its own area during the loading process.

Error Messages

In every case of an error message the line of code being processed when the error is discovered is printed to the left of the message.

Tϕϕ MANY INSTRUCTIϕNS - you have more than 1200 lines of code in the block. The assembler stops and the stop is not recoverable.

Tϕϕ MANY LABELS - you have more than 511 labels in this block (see Restrictions and Limitations). The assembler stops and the stop is not recoverable.

Tϕϕ MANY EQU's - there are more than 100 EQU's in the block. The assembler stops in an unrecoverable stop.

ILLEGAL *** INSTRUCTIϕN - the first five non-blank characters of the left address are not a legitimate instruction. The instruction is ignored.

UNDEFINED LABEL - the line of code in which either the left or right address refers to a label which is not in label table is printed. The appropriate address is set to 0.

AMBIGUOUS LABEL - a label in the table appears more than once, in the Label Field. The label is printed with its two defined values. If it occurs more than twice the errors are printed in pairs, each with this message. No distinction is made between global and local labels.

IMPRϕPER LABEL - is caused by an EQU with a blank label. The assembler stops in an unrecoverable stop.

NϕN-EXISTENT ϕPERATIϕN - is caused by the misspelling of a mnemonic op code. The operation is set to zero.

ILLEGAL RIGHT ϕPERATIϕN - a pseudo-op has been put in the right operation field - the operation is set to zero and the pseudo-op ignored.

PϕSSIBLE J'AC ERRϕR - the assembler has missed the END instruction during pass 2. The assembler stops in an unrecoverable stop.

J'ACASS HAS ERRED - is caused by disagreement in the value of the location counter value at the end of pass 1 and pass 2 for a block. This is usually caused by a coding error. The assembler stops in an unrecoverable stop.

ERRØRS PREVENT EXECUTIØN - is printed when GØ is given and there has been at least one error.

IMPRØPER DEC, SET TØ 0 - an error in the binary or decimal scaling on a DEC results in a 0 result.

*** FINISH, NO *** START - The pseudo-brackets used to identify a block of symbolic code from which to produce a J'AM'D deck are not properly arranged.

APPENDIX B JOHNNIAC OP CODE MNEMONICS FOR J'ACASS

OCTAL	MNEMONICS
000	NOP ---
001	TLM TNL TML
002	TLP TPL
003	TLO TFL TΦL
004	LDQ LMQ LM
005	TRM TNR TMR
006	TRP TPR
007	TRO TFR TΦR
010	TRL TRA
011	T1L TL1
012	T2L TL2
013	T3L TL3
014	TRR
015	T1R TR1
016	T2R TR2
017	T3R TR3
020	CLA LAC LDA RA
021	CLS RS
022	CAM RAV
023	CSM CSV RSV
024	ADD A
025	SUB S
026	ADM AV
027	SBM SV
030	MPR MR
031	MNR
032	MPY M
033	MPN MN
034	MAR MB
035	MNB
036	MPA MA
037	MNA
040	DVS DS
041	DNS
044	DIV D
045	DVN DN
050	STO ST
051	SLO SOL
052	SLA SAL
053	SLH SHL
054	SBA SAB
055	SRO SOR
056	SRA SAR
057	SRH SHR

OCTAL	MNEMONICS
060	STQ
061	SNQ
062	SVQ
063	SNV
064	AQS
065	SQS
066	AVS
067	SVS
070	ARC SRC
071	CLC
072	LRC
073	LLC
074	ARH
075	CLH
076	LRH
077	LLH
100	SEL
101	CPY C
104	DIS
105	HUT
106	SPA EJ
107	RCA
110	RDD RD
111	WRD WD
120	ZAC ZTA
124	ANA AND PI
125	NAN CAA NI
126	AVA PMI
127	CAV NMI
130	HTL
131	H1L
132	H2L
133	H3L
134	HTR
135	H1R
136	H2R
137	H3R
140	WRC
141	RDC
142	WRR
143	RRA
144	SRM
145	SRN

Appendix C - Hollerith characters and J'ACASS octal equivalents

Character	Octal	Character	Octal
blank	00		
0	60	N	45
1	01	ϕ	46
2	02	P	47
3	03	Q	50
4	04	R	51
5	05	S	62
6	06	T	63
7	07	U	64
8	10	V	65
9	11	W	66
A	21	X	67
B	22	Y	70
C	23	Z	71
D	24	+	20
E	25	-	40
F	26	*	54
G	27	/	61
H	30	$	53
I	31	'	14
J	41	(	74
K	42	)	34
L	43	.	33
M	44	,	73
		=	13

J315A

J'ACASS II' JOHNNIAC ASSEMBLER

Index