MEMORANDUM
RM-5028-PR
SEPTEMBER 1966

JOSS: ARITHMETIC AND FUNCTION EVALUATION ROUTINES

I. D. Greenwald

PREPARED FOR:
UNITED STATES AIR FORCE PROJECT RAND

The RAND Corporation
SANTA MONICA • CALIFORNIA

MEMORANDUM
RM-5028-PR
SEPTEMBER 1966

JOSS: ARITHMETIC AND FUNCTION EVALUATION ROUTINES

I. D. Greenwald

This research is sponsored by the United States Air Force under Project RAND—Contract No. AF 49(638)-1700—monitored by the Directorate of Operational Requirements and Development Plans, Deputy Chief of Staff, Research and Development, Hq USAF. Views or conclusions contained in this Memorandum should not be interpreted as representing the official opinion or policy of the United States Air Force.

The RAND Corporation
1700 MAIN ST. • SANTA MONICA • CALIFORNIA • 90406

Published by The RAND Corporation

PREFACE

In response to requests from computer installations to provide material describing the programming of JOSS[†] arithmetic and function evaluation routines, this study is addressed to professional programmers. Attempt is made to provide answers to the kinds of questions such an audience might raise if they were interested in producing similar programs for their own computers. The description of each routine consists of an analysis section and/or a program flow and/or commentary on the program flow, the choice being dictated by what would best serve the needs of programmers.

[†]JOSS is the trademark and service mark of The RAND Corporation for its computer program and services using that program.

SUMMARY

This memorandum describes JOSS arithmetic and function evaluation routines from a programmer's point of view. JOSS is an experimental, on-line, time-shared computing service in daily use by staff members of The RAND Corporation for the solution of small numerical problems. It is currently operational on a Digital Equipment Corporation PDP-6 computer.

For the purposes of this study, the JOSS functions are divided into three groups: arithmetic operations (including exponential and square root), elementary transcendental functions, and number dissection functions.

The four arithmetic operations (add, subtract, multiply, divide) treat the operands as exact nine-digit numbers and produce true results rounded to nine digits. This is accomplished by doing integer arithmetic and scaling by powers of ten. Descriptions of the arithmetic routines are presented in gross verbal flowcharts, amplified by commentaries. The general exponential routine to compute A * B factors out error situations and the special cases of B = 0, A = 0, B = 1, and A an integer power of 10, B = .5, B = -.5, and B an integer with $2 \leq B \leq 29$, before resorting to $\exp[B \cdot \log(A)]$.

The discussion of the transcendental functions emphasizes the analysis done in order to achieve almost nine-significant-digit accuracy in the results and shows how special cases are handled in order to hit certain "magic" values on the nose (for example, the exponential of 0).

Program flows of the number dissection functions (integer part, digit part, fractional part, signum, exponent part) are next presented. Especially when coupled with the ability to append an "if" clause to virtually any statement, they demonstrate how a sophisticated tool may be supplied to the user with trivial expenditure of programming effort.

Appendixes to the study contain (1) a description of Routine S75, which converts and normalizes binary answers to JNF (JOSS Normal Form), and (2) two algorithms for evaluating series on a computer.

ACKNOWLEDGMENTS

J. C. Shaw created most of the techniques and algorithms described herein for the JOHNNIAC version of JOSS. The well-annotated listings of that version were the source of all of my initial insights. Many of the modifications and extensions arose from conversations with C. L. Baker, who also did much of the final testing of the routines.

CONTENTS

PREFACE iii

SUMMARY v

ACKNOWLEDGMENTS vii

Section
I. INTRODUCTION 1
Description of Approach 1
Salient Machine Characteristics 2

II. NUMBER REPRESENTATION 3

III. ARITHMETIC OPERATIONS 5
ADD: C = A + B 6
Flow 6
Commentary 7
MULTIPLY: C = A·B 8
Flow 8
Commentary 9
DIVIDE: C = A/B 10
Flow 10
Commentary 11
EXPONENTIATE: C = A * B 12
Flow 12
Commentary 13
SQRT: C = $\sqrt{A}$ 14
Flow 14
Commentary 15

IV. ELEMENTARY TRANSCENDENTAL FUNCTIONS 16
EXPONENTIAL: C = e^A 17
Analysis 17
Flow 18
Commentary 19
LOGARITHM: C = ln(A) 21
Analysis 21
Commentary 24
CIRCULAR SINE, COSINE: C = sin(A); C = cos(A) 25
Analysis 25
Commentary 27
ARGUMENT: C = arg(A, B) 29
Analysis 29
Commentary 31

V. NUMBER DISSECTION FUNCTIONS 32
INTEGER PART: C = ip(A) 32
Flow 32

DIGIT PART: C = dp(A) 32
Flow 32
FRACTIONAL PART: C = fp(A) 33
Flow 33
SIGNUM: C = sgn(A) 33
Flow 33
EXPONENT PART: C = xp(A) 34
Flow 34

Appendix
A. ROUTINE S75: CONVERT TO JNF 36
Flow 36
Commentary 37
B. TECHNIQUES FOR SERIES EVALUATION 38

BIBLIOGRAPHY 41

I. INTRODUCTION

This memorandum describes JOSS† arithmetic and function evaluation routines from a programmer's point of view. JOSS is a user-oriented time-sharing system for numerical computation. It is currently operational on a Digital Equipment Corporation PDP-6 computer. Descriptions of the system and its uses appear in the studies listed in the Bibliography to this memorandum.

One design criterion for JOSS was that it should give the same results for arithmetic operations as those provided by a desk calculator. This implies, for example, that the nine-decimal-digit inputs to the arithmetic routines are treated as exact and that the outputs are true nine-digit rounded results. To meet this criterion, JOSS numbers are represented internally as integers with associated powers of ten and all arithmetic operations are performed using integer arithmetic.

An added advantage of this technique is that "real" numbers and "integers" need not be differentiated, as is true for systems representing numbers as fractions with an associated exponent (for example, floating binary).

DESCRIPTION OF APPROACH

This study is addressed to professional programmers, in response to requests from computer installations to provide material describing how the JOSS programming was

---

†JOSS is the trademark and service mark of The RAND Corporation for its computer program and services using that program.

done. The general organization of the material into analysis/flow/commentary sections is intended to supply the programmer with specific answers on how the analysis was performed and what the program flow is, with descriptive comments to elucidate the flow. Superfluous information, however, has not been included merely for the sake of symmetry between sections, and therefore the description of each program is composed of an analysis section and/or a program flow and/or a commentary section, the choice being dictated solely by what would best serve the needs of programmers.

SALIENT MACHINE CHARACTERISTICS

In order to describe JOSS arithmetic and function evaluation routines, only two characteristics of the Digital Equipment Corporation PDP-6 computer need be set forth:

1. The machine is binary, with a fixed 36-bit word length, and represents negative numbers in 2's complement form.†

2. Multiplication of two fixed-point, 36-bit numbers (optionally) produces a 72-bit product in two adjacent registers. Division of a 72-bit dividend by a 36-bit divisor produces a quotient and a remainder of 36 bits each.

---

†The 2's complement of a number is obtained by inverting all of the bits (changing 0's to 1's and 1's to 0's) and then adding a 1 in the least significant bit.

II. NUMBER REPRESENTATION

The internal representation of a number, Y, in JOSS consists of

1. A positive integer, Y_m, such that

$$10^8 \leq Y_m < 10^9 \quad \text{or} \quad Y_m = 0;$$

2. An associated sign Y_s;

3. An exponent, Y_e, chosen so that

$$|Y| = Y_m \cdot 10^{Y_e - 8}.$$

We restrict Y_e to $-100 < Y_e < 100$.

Offsetting the exponent by eight is equivalent to assuming that Y_m consists of one whole digit and eight decimal digits. For example, Y = 123.456789 is equivalent to $Y = 1.23456789 \cdot 10^2$; this is represented internally as

$$Y_m = 123456789,$$

$$Y_s = +,$$

$$Y_e = 2.$$

This is JOSS Normal Form (JNF).

All inputs to the arithmetic and function routines in JOSS are assumed to be in JNF. Output from these routines is in JNF, except that the exponent range, Y_e, has not been checked.

In this study, we shall use the notation: A and B for input, C for output, and others (X, Y) for intermediate quantities. Thus, for example:

C_s = sign of result,

B_e = exponent of second input,

A_m = magnitude of first input.

III. ARITHMETIC OPERATIONS

The four arithmetic operations (add, subtract, multiply, divide) treat the operands as exact nine-digit numbers and produce true results, rounded to nine digits. This is accomplished by doing integer arithmetic and scaling by powers of ten.

Descriptions of the arithmetic routines will consist of gross verbal flowcharts, amplified by commentaries. The following general comments apply to the descriptions:

1. Subtraction (C = A - B) is merely addition with B_s inverted, and will not be described further.

2. The algebraic sign of the result can always be determined in advance. For addition, it is the sign of the operand of greater magnitude; for multiplication and division, it is the exclusive OR of the signs of the two operands.

3. Since the sign of the result is predetermined, multiplication and division are carried out exclusively with magnitudes. Addition of numbers with the same algebraic signs is performed with magnitudes; if the signs are not the same, the integer, Y_m, of the number of smaller magnitude is complemented.

4. The following abbreviations have been used:

   R = remainder function,
   Q = quotient function,
   ip = integer part (greatest integer in),
   Bn = (n is an integer) binary scale factor.

ADD: C = A + B

Flow

1. If B = 0, set C = A and exit.
2. If A = 0, set C = B and exit.
3. If $|A| < |B|$, interchange A and B.
4. Set $p = A_e - B_e$.
5. If $p > 10$, set C = A and exit.
6. If $A_s \neq B_s$, complement B_m.
7. Set $C_s = A_s$ and $C_e = A_e$.
8. Set $X_q = Q\left[\frac{2\cdot(10^p\cdot A_m + B_m)}{10^p}\right]$.
9. Set $X_r = R\left[\frac{2\cdot(10^p\cdot A_m + B_m)}{10^p}\right]$.
10. If $X_q = X_r = 0$, set C = 0 and exit.
11. If $X_q < 2\cdot 10^8$, decrement† C_e and set $X_q = 10\cdot X_q + (10\cdot X_r)/10^p$.
12. For as long as $X_q < 2\cdot 10^8$, decrement C_e and set $X_q = 10\cdot X_q$.
13. If $X_q \geq 2\cdot 10^9 - 1$, increment† C_e and set $X_q = X_q/10$.
14. Set $C_m = (X_q + 1)/2$ and exit.

---

†"Decrement" means reduce by one; "increment" means increase by one.

Commentary

3. As a result of this step, A will always be the number with the greater magnitude. This predetermines both the sign of the result for step 7, and a preliminary value for the exponent that will be adjusted during normalization.

4. This step determines the amount of decimal shift required to line up the decimal points.

5. If $p > 10$, B can have no effect on the result. On the other hand, for $p = 10$, consider

$$\begin{array}{r} 100\ 000\ 000 \\ -\ \underline{000\ 000\ 000\ 06}, \\ 99\ 999\ 999\ 94 \end{array}$$

which, when rounded, gives 999 999 999. Thus, exponent differences up to and including 10 must be considered for nine-digit results.

8. This is the actual addition step. The remainder is required when, for example,

$$\begin{array}{r} 100\ 000\ 000 \\ -\ \underline{99\ 999\ 999\ 3}. \\ 000\ 000\ 000\ 7 \end{array}$$

The multiplicative factor of 2 produces one additional bit for rounding purposes (see step 14). $A_m \cdot 10^p$ produces a 72-bit product. B_m is added to the low 36 bits. If the result of this addition causes an overflow or is negative, appropriate adjustments must be made. (The negative result can occur if B_m was complemented at step 6.)

11. If normalization is required, only one digit of the remainder could possibly contribute to the result (see the example in step 8 of this commentary).

12. This step continues the normalization process by bringing in trailing zeros.

13. In addition to normalizing downward (A and B had the same algebraic sign), this step also ensures against an overflow in the rounding step that follows.

MULTIPLY: $C = A \cdot B$

Flow

1. If A or B = 0, set C = 0 and exit.
2. Set $C_s = A_s \oplus B_s$ ($\oplus$ = exclusive OR).
3. Set $C_e = A_e + B_e$.
4. Set $X = 2A_m \cdot B_m$.
5. If $X < 2 \cdot 10^{17} - 10^8$, set $C_m = [(X/10^8) + 1]/2$ and exit.
6. Set $C_m = [(X/10^9) + 1]/2$.
7. Set $C_e = C_e + 1$.
8. Exit.

Commentary

4. Twice the product is computed to give a bit for rounding. The product has 72 bits. Note that the product $A \cdot B$ has two whole digits and 16 decimal digits. Further, since A and B are carried as integers, they both contain a multiplicative factor of 10^8, giving the product a factor of 10^{16}.

5-7. If the first whole digit is zero, divide by 10^8 to return to JNF. Otherwise, divide by 10^9 and increment the exponent by 1. The test also ensures that rounding will not propagate an overflow. The comparison is double length: The most significant half of X is compared against

$$Q\left[\frac{2\cdot 10^{17} - 10^8}{2^{35}}\right] = 5820766,$$

while the least significant half of X is compared against

$$R\left[\frac{2\cdot 10^{17} - 10^8}{2^{35}}\right] = 3038650112.$$

DIVIDE: C = A/B

Flow

1. Error if $B = 0$.
2. If $A = 0$, set $C = 0$ and exit.
3. Set $C_s = A_s \oplus B_s$.
4. Set $C_e = A_e - B_e$.
5. If $A_m \geq B_m$, set $p = 8$.
6. If $A_m < B_m$, set $p = 9$ and $C_e = C_e - 1$.
7. $C_m = \frac{1}{2}\left[\frac{2 \cdot A_m \cdot 10^p}{B_m} + 1\right]$ and exit.

Commentary

5. Since both numbers are in JNF, if $A \geq B$, $A \cdot 10^8/B$ will be in JNF; if $A < B$, $A \cdot 10^9/B$ will be in JNF. The factor of 2 is for rounding. An alternative approach to get the rounded result would be to compute

$$\frac{A \cdot 10^p + B/2}{B}.$$

The choice depends on the instruction repertoire of a given machine. In either case, the product is, of course, double length.

EXPONENTIATE: C = A * B

Flow

	Case	Result
1.	B = 0.	C = 1.
2.	A = 0; B < 0.	Error.
3.	A = 0; B > 0.	C = 0.
4.	A = 1.	C = 1.
5.	B = 1.	C = A.
6.	B = -1.	C = 1/A.
7.	A < 0; B not integral.	Error.
8.	A > 0; B = $\frac{1}{2}$.	C = SQRT(A).
9.	A > 0; B = $-\frac{1}{2}$.	C = 1/SQRT(A).
10.	A = $\pm 10^{p}$; B integral.	$C_e = 10^{p \cdot B}$.
		$C_s = A_s^B$.
11.	B integral; 1 < B ≤ 29.	Compute by multiplication.
12.	B integral; -29 ≤ B < -1.	Compute by multiplication on (1/A).
13.	B integral; 29 > |B|.	C_m = EXP[B·LØG(|A|)].
		$C_s = A_s^B$.
14.	All others.	C = EXP[B·LØG(A)].

Commentary

1. $0^0 = 1$.

7. B is integral if

   a. $B_e \geq 8$, or

   b. $B_e \geq 0$ and $\text{REM}\left[\frac{B_m}{10^{8-B_e}}\right] = 0$.

8,9. JOSS SQRT routine.

11,12. The rationale for choosing B = 29 as the crossover is that 2^{29} can be represented exactly in JNF, while 2^{30} cannot. Since $1 < |B| \leq 29$, B is at most 5 bits. The multiplication is done as follows:

   a. If $B < 0$, set $A = 1/A$ and $B = |B|$;
   b. Set $C = 1$;
   c. If B is odd, set $C = A \cdot C$;
   d. Shift B to the right one place (zeros enter at the left);
   e. Exit if $B = 0$;
   f. Set $A = A^2$;
   g. To step c.

   All multiplications and the division (step a) use the JOSS arithmetic routines.

13. EXP and LØG are the JOSS exponential and logarithm routines. If B is $\begin{Bmatrix}\text{even}\\ \text{odd}\end{Bmatrix}$, C_s will be $\begin{Bmatrix}+\\ A_s\end{Bmatrix}$.

14. The LØG will error exit if $A \leq 0$; EXP will error exit if the result $> 2^{36} - 1$.

SQRT: $C = \sqrt{A}$

Flow

1. Error if $A < 0$.
2. Set $C = 0$ and exit if $A = 0$.
3. If A_e = odd, set $A_m = 10 \cdot A_m$ and $C_m = 10^9$.
4. If A_e = even, set $C_m = 10^8 \cdot \sqrt{10}$ + epsilon.
5. Set $C_e = ip(A_e/2)$ and $C_s = +$.
6. Iterate by

$$D = C_m - \frac{10^8 \cdot A_m}{C_m},$$

$$C_m = \frac{2 \cdot C_m - D}{2}.$$

   Exit from iteration when $D < 10^4$.

7. Exit if $(10^8 \cdot A_m)/C_m - C_m < 1$.
8. Set $C_m = C_m + 1$ and exit.

Commentary

3,4. C_m is the initial guess for iteration. The addition of an epsilon ensures that convergence will be from the high side. This is necessary so that D (step 6) does not go negative before converging. Epsilon may be determined empirically for a given implementation.

5. ip = integer part. This step is a right shift of 1 bit.

6. This is merely Newton's method. D is kept separate to test for convergence. Newton's method for square root doubles the number of correct digits at each iteration. Hence, convergence is based on five correct digits in D.

7,8. It is given that X is an approximate, but possibly low, value of $\sqrt{N}$. Test to see which value, X^2 or $(X + 1)^2$, is closer to N. If

$$N - X^2 > (X + 1)^2 - N,$$

then

$$N - X^2 > X^2 + 2X + 1 - N,$$

$$2N - 2X^2 > 2X + 1,$$

$$N - X^2 > X + \frac{1}{2} > X, \qquad X > 0,$$

$$\frac{N}{X} - X > 1.$$

Thus, if this last condition is true, $(X + 1)^2$ is closer to N than X^2, so add one to the approximate value X to obtain the required square root.

IV. ELEMENTARY TRANSCENDENTAL FUNCTIONS

We wished, if possible, to use single-precision arithmetic for the evaluation of the JOSS functions. In surveying the field of rational approximations, we could not find ones that (1) yield the desired accuracy and (2) had coefficients that could be represented in 36-bit fixed point.

Continued fractions were considered and finally discarded for several reasons:

1. For a given precision of a restricted range of argument, series evaluation appeared to be faster and easier (primarily because the latter lends itself so well to computer implementation).

2. For us, at least, it was much easier to determine the precision of a given truncated series than for a continued fraction.

3. The numerical range of results of intermediate computations for series is both easier to determine and smaller.

In general, the analyses were oriented toward achieving a relative error on the order of 10^{-9}. On the other hand, since we were dealing with a finite word length and the approximations required many operations, we attempted to obtain 35-bit accuracy wherever possible--that is, $(3 \cdot 10^{-11})$ in all intermediate calculations. Since we always had a restricted range of arguments, the latter choice usually yielded better results. It is important to recognize that no amount of analysis can substitute for a great deal of care in coding, particularly in scaling for maximum precision.

EXPONENTIAL: $C = e^A$

Analysis

The exponential is computed using the series approximation

$$e^x \sim \sum_{n=0}^{k} \frac{X^n}{n!}.$$

We always evaluated $e^{|x|}$, then took the reciprocal if $X < 0$. The problem was to restrict the range of x, so that a fixed and relatively small number of terms, k, would give the required precision. Since

$$y = e^x = 10^{x/C} = 10^W \cdot 10^F = 10^W \cdot e^{F \cdot C},$$

where $C = \ln(10)$,

W = whole digits,

F = decimal digits,

$x > 0$,

we could immediately restrict the range to $F \cdot C$, taking care of 10^W by adjusting the exponent in the JNF result.

The range $0 \le F \cdot C \le 2.30258509$ is still rather large, but

$$e^x = (e^{x/4})^4.$$

With $F \cdot C/4 < .576$, $k = 12$ will give 36-bit precision. (Due to the many operations involved and the use of single-precision arithmetic, the precision is unattainable; however, care in scaling and rounding nearly achieves it.)

Flow

1. If A = 0, set C = 1 and exit.

2. If |A| > 230.258509: set C = 0 and exit if A < 0; error exit if A > 0.

3. If $A_e < -5$, set C = 1 + A and exit.

4. If $A_e < 0$, compute

$$X = \frac{A_m \cdot 2^{-11}}{10^{8+A_e}}, \qquad \text{(denominator scaled at B44)}$$

$$C_e = 0.$$

5. If $A_e > 0$, compute

$$C_e = Q\left[\frac{A_m \cdot 2^4 \cdot 10^{A_e}}{10^8 \cdot \ln(10)}\right], \qquad \begin{matrix}(10^{A_e} \text{ scaled at B32}) \\ (10^8 \cdot \ln(10) \text{ scaled at B28})\end{matrix}$$

$$X = R\left[\frac{A_m \cdot 2^4 \cdot 10^{A_e}}{10^8 \cdot \ln(10)}\right] \Big/ 10^8 \cdot \ln(10),$$

$$X = [X \cdot \ln(10)] \cdot 2. \qquad (\ln(10) \text{ scaled at B3})$$

6. Compute

$$y_0 = \frac{X}{2!} + \frac{X^2}{3!} + \cdots + \frac{X^{11}}{12!}, \qquad \text{(scaled at B1)}$$

$$y_1 = X \cdot y_0 + 1, \qquad \text{(rounded and scaled at B1)}$$

$$y_2 = X \cdot y_1 + 1, \qquad \text{(rounded and scaled at B1)}$$

$$y_3 = (y_2^2)^2. \qquad \text{(rounded each step, scaled at B2, B4)}$$

7. If A_s = -:

   Set $C_e = -C_e$,

   Set $y_3 = 1/y^3$. (1 scaled at B8)

8. Set C_s = +.

9. Convert to JNF (routine S75) B4.†

10. Exit.

Commentary

2. Since a JNF number must be less than 10^{100}, A must be less than $100 \cdot \ln(10) = 230.258509$. Underflow is treated as a zero.

3. For $|X| < 10^{-5}$, $X^2/2 < 5 \cdot 10^{-11}$ and contributes nothing. Use the JOSS ADD routine to compute 1 + X.

4. If there are no whole digits, set $C_e = 0$ and convert $A_m/4$ to a fraction by dividing by the appropriate power of 10. A_m is an integer and hence is scaled at B = 35:

$$\frac{A_m \cdot 2^{-35} \cdot 2^{-11}}{10^P \cdot 2^{-44}} = A \cdot 2^{-2} = \frac{A}{4}.$$

5. If there are whole digits, convert A to nonscientific notation by multiplying by the appropriate power of 10. Keep a double precision product (scaled for the division that is to follow):

---

†Routine S75 (see Appendix A) has as input C_e, a binary fraction, and a scale factor ($-3 \le B \le 4$); it outputs a JNF number.

$$A_m \cdot 2^{-35} \cdot 2^4 \cdot 10^{A_e} \cdot 2^{-32} = A \cdot 2^{-63}.$$

Convert to whole numbers and fraction by dividing by 10^8, then dividing the remainder by 10^8. To get A/ln(10), divide by $10^8 \cdot \ln(10)$:

$$W = Q\left[\frac{A \cdot 2^{-63}}{10^8 \ln(10) \cdot 2^{-28}}\right] = A \cdot 2^{-35}$$

(that is, W = integer part),

$$F = \frac{R\left[\frac{A \cdot 2^{-63}}{10^8 \ln(10) \cdot 2^{-28}}\right]}{10^8 \ln(10) \cdot 2^{-28}} = A \cdot 2^0$$

(that is, F = fraction part).

Multiply the fraction part by ln(10) and divide by 4:

$$\frac{F \cdot C}{4} = F \cdot \ln(10) \cdot 2^{-3} \cdot 2 = F \cdot \ln(10) \cdot 2^{-2}.$$

6. Evaluate the series and raise to the fourth power.

7. $e^{-X} = 1/e^X$.

LOGARITHM: C = ln(A)

Analysis

Log is computed using the series approximation:

$$\log(x) \sim 2\cdot\left[z + \frac{z^3}{3} + \frac{z^5}{5} + \cdots\right],$$

where

$$z = \frac{X - 1}{X + 1}.$$

The problem is to reduce the range of z. The solution is based on the following relation: If $X = \nu\cdot 10^P$,

$$\ln(X) = \ln(\nu) + P\cdot\ln(10). \qquad (1)$$

In JNF, $P = A_e$; hence $P\cdot\ln(10)$ is easy to obtain. Since $1 \le \nu \le 10$, the range of

$$w = \left|\frac{\nu - 1}{\nu + 1}\right|$$

is $0 \le w \le .82$, necessitating a considerable number of terms in the series. A reduction may be obtained by applying the definition:

$$\ln(\nu) = \ln(2^a\cdot\nu) - a\cdot\ln(2) \qquad (2)$$

(we choose powers of 2 since they are merely a shift on a binary computer). The minimum range on the interval $\nu' < 2^a\cdot\nu < 2\nu'$ occurs when

$$\left|\frac{\nu' - 1}{\nu' + 1}\right| = \left|\frac{2\cdot\nu' - 1}{2\cdot\nu' + 1}\right|,$$

which yields $\nu' = \sqrt{2}/2$.

Choosing a such that

$$\sqrt{2}/2 \le 2^a\cdot\nu < \sqrt{2}$$

yields $0 \le w < .172$, which reduces the number of terms of the series to 7.

However, ν lies in the range $1 \le \nu < 10$, so that $a \le 0$ (that is, a right shift and lost precision). To change the range of $2^a\cdot\nu$, rewrite Eq. (2) as follows:

$$\ln(\nu) = \ln\left[\frac{2^a}{2^b}\cdot\nu\right] + (b - a)\cdot\ln(2) \qquad (3)$$

and

$$z = \frac{2^a\cdot\nu - 2^b}{2^a\cdot\nu + 2^b}.$$

The range of $2^a\cdot\nu$ would be

$$2^b\sqrt{2}/2 \le 2^a\cdot\nu < 2^b\sqrt{2}.$$

Since $\nu < 10$, $b = 3$ ensures that $a \ge 0$ (that is, a left shift).

Combining Eqs. (1) and (3) yields

$$\ln(x) = P\cdot\ln(10) + (3 - a)\cdot\ln(2) + 2\cdot\left[z + \frac{z^3}{3} + \cdots + \frac{z^{13}}{13}\right], \qquad (4)$$

where

$$z = \frac{2^a\cdot\nu - 8}{2^a\cdot\nu + 8},$$

and a is chosen such that

$$8 \cdot \frac{\sqrt{2}}{2} \leq 2^a \cdot \nu < 8 \cdot \sqrt{2}.$$

The chosen series yields very poor results for X close to 1 ($X = 1 \pm \epsilon$). Experimentation† has shown that the range $.997885258 \leq X \leq 1.00211474$ can be considerably improved by using the first three terms of the series

$$\log(1 + X) = X - \frac{X^2}{2} + \frac{X^3}{3}.$$

We set $Z = X - 1$ and then evaluate

$$Z - \frac{Z^2}{2} + \frac{Z^3}{3}$$

using "pseudo" JOSS arithmetic for the multiplications and divisions, and JOSS arithmetic for the additions and subtractions. ("Pseudo" indicates that we take advantage of our knowledge of ranges and that the results are not always normalized or rounded.)

Finally, compute the first two terms of Eq. (4) to "double precision"; that is, carry a whole number and a fraction. The summing of the three terms, $P \cdot \ln(10)$, $(3 - a) \cdot \ln(2)$, and the series, can be tricky in double precision on a 2's complement machine, though the second term cannot be negative.

---

†The crossover points are a function of the number of digits in the JNF number, the precision of intermediate quantities, and the order of computation. Hence, the final determinations were done empirically.

Commentary

a. Error exit if $A \leq 0$.

b. If $|A - 1| > \varepsilon$: $(\varepsilon = 2.11474 \cdot 10^{-3})$

Shift A_m to the left until $A_m \geq 8 \cdot (\sqrt{2}/2)$;

Set q = 3 - (number of shifts);

Set $X = \dfrac{[A_m - 10^8 \cdot 8] \cdot 2^2}{A_m + 10^8 \cdot 8}$; (X will be scaled at B-2)

Set $z = X^2$; (rounded and scaled at B0)

Set $s = z^6/13 + z^5/11 + \ldots z/3$; (all constants scaled at B-1)

Set $y = (s \cdot X \cdot 2^{-1} + X \pm 2^{-35})2^{-1} = \ln(A_m)$; (y scaled at B0)

$$\ln(A) = q \cdot \ln(2) + A_e \cdot \ln(10) + y,$$

where the first two terms are double precision, B35.

Set C_s = sign of ln(A); X = |ln(A)|.

$$\text{Set } C_e = \begin{cases} 3 \text{ if } 100 \leq X, \\ 2 \text{ if } 10 \leq X < 100, \\ 1 \text{ if } 1 \leq X < 10, \\ 0 \text{ if } X < 1. \end{cases}$$

Set $X = X/10^{C_e}$ (single precision, B0).

Convert to JNF by means of routine S75 and exit.

CIRCULAR SINE, COSINE: C = sin(A); C = cos(A)

Analysis

We arbitrarily decided that $|A| \geq 100$ would result in an error exit, to warn the user of potential loss of precision. The series

$$\sin(X) \sim X - \frac{X^3}{3!} + \frac{X^5}{5!} - \cdots - \frac{X^{11}}{11!},$$

$$\cos(x) \sim 1 - \frac{X^2}{2!} + \frac{X^4}{4!} - \cdots - \frac{X^{10}}{10!}$$

can be evaluated by means of two tables:

$$T: \quad 1, \frac{1}{2}, \frac{1}{3}, \ldots, \frac{1}{11},$$

and

$$T^1: \quad \frac{1}{2}, \frac{1}{3.2}, \frac{1}{4.3}, \ldots, \frac{1}{10.9},$$

and the following algorithm:

1. Input n.
2. Set $z = X^2$.
3. Set $s = T_n = 1/n$.
4. Set $n = n - 2$.
5. Set $s = -s \cdot z \cdot T^1_n + T_n$.
6. If $n = 2$, set $s = (1 - z \cdot s)$ and exit.
7. If $n = 1$, set $s = X \cdot z$ and exit.
8. To step 4.

Thus, for n = 10, the cos series will be evaluated; for n = 11, the sin series will be evaluated.

For $X \leq \pi/4$, the error in truncating the series will be less than

$$\frac{X^{12}}{12!} \sim 1.2 \cdot 10^{-10}.$$

The algorithm for reducing the input, A, is as follows:

1. If sin, set $C_s = A_s$ and n = 11.
2. If cos, set C_s = + and n = 10.
3. Set $y = R(A/2\pi)$.
4. If $y > \pi$:

   Set $y = 2\pi - y$;

   Invert C_s, if n = 11.
5. If $y > \pi/2$:

   Set $y = \pi - y$;

   Invert C_s, if n = 10.
6. If $y > \pi/4$:

   Set $y = \pi/2 - y$;

   Set n = 11, if n = 10;

   Set n = 10, if n = 11.

Steps 1 to 5 reduce the argument to the first quadrant adjusting the sign of the answer as needed. Step 6 takes the opposite function of $(\pi/2 - y)$ if $y > \pi/4$.

For $X = \epsilon$, the sin will not be as accurate as necessary. Hence (for sin only) we have the following special case: If $X \leq 2 \cdot 10^{-2}$,

$$\sin(X) = \left(1 - \frac{X^2}{6}\right) \cdot X.$$

The evaluation of $-X^2/6$ is done using "pseudo" JOSS arithmetic, and the remainder of the evaluation with JOSS arithmetic.

Commentary

a. If $A_e \leq 0$, compute

$$y = \frac{A_m \cdot 2^{-4}}{10^8}. \qquad \text{(y scaled at B4)}$$

Then if $A_e < 0$, compute

$$y = \frac{y \cdot 2^{-32}}{10^{|A_e|}}. \qquad \text{(result scaled at B1)}$$

Go to compare against $\pi/4$.

If $A_e = 0$, compute

$$y = R\left[\frac{y \cdot 2^{-34}}{2\pi}\right]. \qquad \text{(result, } 2\pi \text{ scaled at B3)}$$

Go to compare against π.

b. If $A_e > 0$, compute

$$x = \frac{10 \cdot A_m \cdot 2^{-7}}{10^8};$$

$$y = R\left[\frac{x \cdot 2^{-31}}{2\pi}\right]. \qquad \text{(y, } 2\pi \text{ scaled at B3)}$$

Go to compare against π.

c. Note that

$$2\pi(B3) = \pi(B2) = \frac{\pi}{2}(B1),$$

and

$$\pi(B3) = \frac{\pi}{2}(B2) = \frac{\pi}{4}(B1).$$

d. (Refer to step 4 of the reduction algorithm given on page 26.)

If $y(B3) > \pi(B3)$, set $y = 2\pi - y$; invert C_s if $n = 11$.

Set $y = 2 \cdot y$.

If $y(B2) > \pi/2(B2)$, and so forth.

ARGUMENT: C = arg(A, B)

Analysis

The arg function computes the angle between the x axis and the line from the origin to the point x, y. The result is positive in the first and second quadrants and negative in the third and fourth quadrants. Hence $-\pi < \arg(x, y) \leq \pi$, and arg(0, 0) is defined to be equal to 0. Use the series

$$\tan^{-1} Z = Z - \frac{Z^3}{3} + \frac{Z^5}{5} - , \ldots, \qquad Z^2 < 1,$$

where $Z = y/x$. To meet the constraint that $Z^2 < 1$, compute

$$Z = \frac{x}{y} \quad \text{if} \quad x^2 < y^2$$

and

$$\arg(x, y) = \frac{\pi}{2} - \tan^{-1} Z$$

($(x^2 = y^2)$ is treated as a special case).

Since this series converges slowly, again reduce the range. The following relations are pertinent:

1. Let $\alpha = \tan^{-1} Z = \theta + \beta$. Then $Z = \tan(\theta + \beta)$.
2. Let $k = \tan \theta$. Then

$$Z = \frac{k + \tan \beta}{1 - k \tan \beta},$$

$$\beta = \tan^{-1} \left(\frac{Z - k}{1 + Z \cdot k} \right),$$

$$\alpha = \theta + \tan^{-1} \left(\frac{Z - k}{1 + Z \cdot k} \right). \qquad (5)$$

Though the derivation is not shown here, Eq. (5) holds

equally well for $\alpha = \theta - \beta$. For

$$\theta_i = 0, \frac{\pi}{32}, \frac{2\pi}{32}, \ldots, \frac{\pi}{4},$$

$$\max |k_i - k_{i-1}| < .098.$$

Hence, for

$$\theta_j = 0, \frac{\pi}{16}, \frac{2\pi}{16}, \ldots, \frac{\pi}{4},$$

if we choose k_j such that

$$|k_j - Z| \leq |k_i - Z| \quad \text{for} \quad i \neq j,$$

then

$$|k_j - Z| < .098,$$

and for $Z > 0$,

$$\omega = \frac{|Z - k_j|}{1 + Z \cdot k_j} < .098,$$

and six terms of the series will suffice.

The following are treated as special cases:

$\arg(0, 0) = 0$;

$\arg(0, y) = \pm\pi/2$ (depending on sign of y);

$\arg(x, 0) = 0$ or π (depending on sign of x);

$\arg(x, y) = \pm n \cdot (\pi/4)$, if $x^2 = y^2$ (n and sign determined by quadrant);

$\tan^{-1} \omega = \omega$ if $\omega \leq 10^{-5}$.

Commentary

The correct quadrant is determined from A_s and B_s. If $|A| < |B|$, interchange A and B and adjust the quadrant by adding 4. Compute α using $Z = |B|/|A|$. Then

Quadrant	arg(A, B)
1	α
2	$\pi - \alpha$
3	$-\pi + \alpha$
4	$-\alpha$
5	$\pi/2 - \alpha$
6	$\pi/2 + \alpha$
7	$-\pi/2 - \alpha$
8	$-\pi/2 + \alpha$

Note that $\arg(A, B) = d + m \cdot \alpha$. On a complement machine, with an index register and an "execute" instruction, one can execute an indexed table that complements or not, as m is + or -, and a second table that adds the appropriate d.

V. NUMBER DISSECTION FUNCTIONS

The number dissection functions in JOSS, especially when coupled with the ability to append an "if" clause to virtually any statement, provide an example of how a sophisticated tool may be supplied to the user with trivial expenditure of programming effort.

INTEGER PART: C = ip(A)

Flow

1. If A = 0, set C = 0 and exit.
2. If $A_e < 0$, set C = 0 and exit.
3. If $A_e \geq 8$, set C = A and exit.
4. Set $C_m = Q\left[\frac{A_m}{10^{8-A_e}}\right] \cdot 10^{8-A_e}$;

   Set $C_s = A_s$;

   Set $C_e = A_e$.
5. Exit.

DIGIT PART: C = dp(A)

Flow

1. Set $C_m = A_m$;

   Set $C_s = A_s$;

   Set $C_e = 0$.
2. Exit.

FRACTIONAL PART: C = fp(A)

Flow

1. If A = 0, set C = 0 and exit.
2. If $A_e < 0$, set C = A and exit.
3. If $A_e \geq 8$, set C = 0 and exit.
4. Set $C_m = R\left[\frac{A_m}{10^{8-A_e}}\right] \cdot 10^{A_e+1}$;

   Set $C_s = A_s$;

   Set $C_e = -1$.
5. If $C_m = 0$, set C = 0.
6. Exit.

SIGNUM: C = sgn(A)

Flow

1. If A = 0, set C = 0 and exit.
2. If A_s = -, set C = -1 and exit.
3. Set C = +1 and exit.

EXPONENT PART: C = xp(A)

Flow

1. If A = 0, set C = 0 and exit.
2. If $A_e < 0$:

   Set C_s = -;

   Set $C_m = |A_e|$.
3. If $A_e \geq 0$:

   Set C_s = +;

   Set $C_m = A_e$.
4. If $C_m \geq 10$:

   Set $C_e = 1$;

   Set $C_m = 10^7 \cdot C_m$.
5. If $C_m < 10$:

   Set $C_e = 0$;

   Set $C_m = 10^8 \cdot C_m$.
6. Exit.

Appendix

Appendix A

ROUTINE S75: CONVERT TO JNF

Most function routines compute a binary answer at some scale factor and must be converted and normalized to JNF. Routine S75 performs this function.

Input X = positive fraction,

B = binary scale factor ($-3 \leq B \leq 4$),

C_e = existing scale factor.

Flow

1. If X = 0, set C = 0 and exit.
2. Set $C_e = C_e - 1$.
3. Compute $y = X \cdot (2 \cdot 10^9) \cdot 2^B$ (double length). ($2 \cdot 10^9$ scaled at B35)
4. If $y < 2 \cdot 10^8$:
   a. Compute $y = 10 \cdot y_1$ + most significant half of $(10 \cdot y_2)$,
      where y_1 = most significant half of y,
      y_2 = least significant half of y.
      Set $C_e = C_e - 1$.
   b. If y = 0, set C = 0 and exit.
   c. If $y < 2 \cdot 10^8$, set $y = 10 \cdot y$ and $C_e = C_e - 1$.
   d. Repeat step c until $y \geq 2 \cdot 10^8$.
5. If $y \geq 2 \cdot 10^9 - 1$, set $y = y/10$ and $C_e = C_e + 1$.
6. Repeat step 5 until $y < 2 \cdot 10^9 - 1$.
7. Set $C_m = (y + 1)/2$ and exit.

(C_s is handled externally.)

Commentary

2. C_e must be adjusted, since 10^9 is the multiplier rather than 10^8.

3. The result of this step will be a double-length product scaled at B35. The factor of 2 supplies an extra bit for rounding.

4. If normalization is required, bring in only one decimal digit from the least significant half. To determine the effect of this, first note that the multiplier is $2 \cdot 10^{10}$; hence, look for an n such that

$$2^{-n} \cdot 2 \cdot 10^{10} < 1,$$

$$2 \cdot 10^{10} < 2^n,$$

$$n = 35.$$

Since the maximum right shift in step 3 is 3, we lose, at most, 3 bits from the input 35-bit X.

5-6. Scale down, ensuring that rounding will not propagate a carry.

7. Round.

Appendix B

TECHNIQUES FOR SERIES EVALUATION

The following two algorithms for evaluating series on a computer should aid the novice. Extension to other series should not be difficult, once these are understood.

1. $$\sum_{n=1}^{k} \frac{X^n}{n}$$

Set n = k,

Set s = 1/n.

LOOP: Set n = n - 1.

If n = 0, set s = s·X and exit.

Set s = s·X + 1/n and go to LOOP.

2. $$\sum_{n=1}^{k} \frac{X^n}{n!}$$

Set n = k,

Set s = 1/n.

LOOP: Set n = n - 1.

If n = 0, set s = s·X and exit.

Set s = (s·X + 1)·(1/n) and go to LOOP.

For most series, 1/n is stored in a table and stepped through by indexing, although some instruction sets may make it appropriate to do division by n.

The series are summed from k down to 1. In general, this aids in ensuring that the contribution of the smaller terms is not ignored--especially when variable scaling is used from term to term.

BIBLIOGRAPHY

DOCUMENTS OF HISTORICAL INTEREST

Baker, C. L., JOSS: Scenario of a Filmed Report, The RAND Corporation, RM-4162-PR, June 1964 (AD 602074).

"The JOSS System: Time-Sharing at RAND," Datamation, Vol. 10, No. 11, November 1964, pp. 32-36. (This article is based on RM-4162-PR above.)

Shaw, J. C., JOSS: A Designer's View of an Experimental On-Line Computing System, The RAND Corporation, P-2922, August 1964; also published in AFIPS Conference Proceedings (1964 FJCC), Vol. 26, Spartan Books, Baltimore, Maryland, 1964, pp. 455-464.

-----, JOSS: Conversations with the Johnniac Open-Shop System, The RAND Corporation, P-3146, May 1965 (AD 615604).

-----, JOSS: Examples of the Use of an Experimental On-line Computing Service, The RAND Corporation, P-3131, April 1965 (AD 614992).

-----, JOSS: Experience with an Experimental Computing Service for Users at Remote Typewriter Consoles, The RAND Corporation, P-3149, May 1965 (AD 615943).

DOCUMENTS OF CURRENT INTEREST

Baker, C. L., JOSS: Introduction to a Helpful Assistant, The RAND Corporation, RM-5058-PR, July 1966.

Greenwald, I. D., JOSS: Console Service Routines (The Distributor), The RAND Corporation, RM-5044-PR, September 1966.