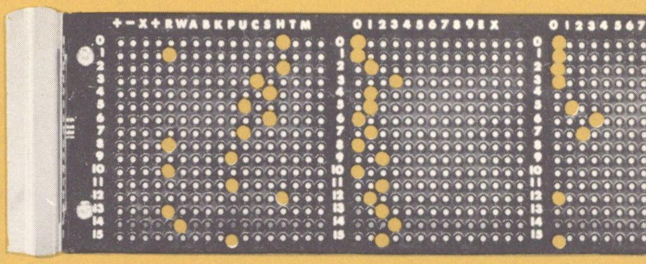

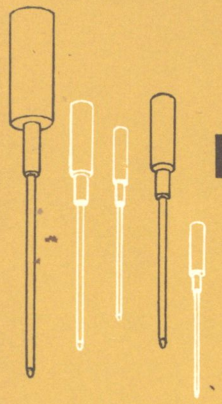

BURROUGHS E 101

HANDBOOK

Subroutines and Subroutine Methods

PREFACE

In computational problems it is common to find that certain functions (such as square roots, logarithms, sines, cosines, tangents, etc.) occur either repetitiously in a single problem or frequently from problem to problem. These functions are basic functions. Programs for computing these basic functions are generally referred to as "subroutines." Because of the frequent occurrence of these basic functions, it becomes expeditious to have available complete programs of subroutines and, in some cases, even to prepare paper templates of the programs.

This booklet consists of two sections.

Section I discusses the uses of subroutines in particular problems and presents the techniques employed in programming them and in extending their range and/or accuracy.

Section II is the main part of the booklet, the chief purpose of which is to consolidate in handy reference form some of the more useful subroutines for the Burroughs E101 Electronic Computer. The section contains the subroutines employed in evaluating the basic, more useful functions and gives programmed examples of techniques which increase the utility of the E101.

At the end is added, for further reference, a short bibliography of handbooks containing the series for common functions and of sources for general studies in functional approximation.

CONTENTS

SECTION I

Uses of, and Techniques for, Subroutines

A. Uses of Subroutines

Some of the various situations which arise in the handling of subroutines and some programming methods which can be considered for these situations are: single use of a subroutine; multiple use of a subroutine; constant increments; compound functions; and multi-purpose use of subroutines.

Before control is transferred to the subroutine, one must be sure that the "starting point" requirements given in the pinboard program for the subroutine have been met.

1. Single Use of a Subroutine

When a subroutine is used only once in the solution of a problem, there are two procedures, either of which may be followed:

(a) to incorporate the subroutine as part of the main program (the sequence of steps in the subroutine would not be altered, although the steps need not be located on a single pinboard; the constant required in the subroutine may be stored in any convenient memory addresses);

(b) to use a separate pinboard for the subroutine and one "U" instruction to transfer from the main part of the problem to the subroutine pinboard and a second "U" instruction to transfer back again.

The first of these two procedures fits the subroutine into the main part of the problem and, as a result, economizes programming space and running time. The second is a bit more flexible and makes better use of pinboards or templates on hand.

2. Multiple Use of a Subroutine

The multiple use of a subroutine is advisable when a subroutine is to be used in several places in the solution of a problem. Each use of the subroutine involves transfers, first, from the problem to the subroutine and, then, back to a different part of the problem. The E101, with the aid of a "subroutine exit instruction," has a simple procedure for handling these transfers.

Each call for the subroutine requires two steps in the main program:

(a) "H 0 b." (This instruction homes the E switch to "b," the "b" being pinned to correspond to the pinboard to which the program should return after the use of the subroutine);

(b) "U a 0." (This instruction transfers control to the beginning of the subroutine, the "a" being pinned to correspond to the pinboard which contains the subroutine).

Neither of these two instructions affects the contents of the accumulator.

The transfer back to the appropriate step in the main program, i.e., the step following the "U a 0" instruction, is effected by the "subroutine exit instruction," "U E *." The "U E *" instruction, which occurs as the last step in the subroutine, transfers control back to pinboard "b" of the last "H 0 b" instruction of the main program and then to the step following the "U a 0" instruction on the same pinboard "b."

To illustrate the procedure, let us assume that we wish to transfer to the sine subroutine (pinboard 4) from the main routine, which is first at step 5 of pinboard 1, later at step 4 of pinboard 2, and finally at step 3 of pinboard 3. After each use of the sine subroutine in pinboard 4 we must return to the pinboard (1, 2, or 3) from which the transfer was made.

Specifically, in the illustration below, the multiple use of the sine subroutine would necessitate initially a transfer from pinboard 1 (step 6) to the beginning of the sine subroutine on pinboard 4 (step 0); then from the end of the sine subroutine on pinboard 4 (step 15) back to the main routine on pinboard 1 (step 7); later from pinboard 2 (step 5) to the beginning of the sine subroutine on pinboard 4 (step 0); then from the end of the sine subroutine on pinboard 4 (step 15) back to the main routine on pinboard 2 (step 6); later from pinboard 3 (step 4) to the beginning of the sine subroutine on pinboard 4 (step 0); then from the end of the sine subroutine on pinboard 4 (step 15) back to the main routine on pinboard 3 (step 5).

Pinboard No.

Step No.	1	2	3	4
0				
1				
2				
3			H 0 3	
4		H 0 2	U 4 0	
5	H 0 1	U 4 0		Sine
6	U 4 0			subroutine
7				
8				
9				
10				
11				
12				
13				
14				
15				U E *

3. Constant Increments

When it is desirable to find a function of a variable that increases at a constant rate, the "chain formula" method should be considered. As an example, consider $\sin ku_0$ and $\cos ku_0$ (used frequently in Fourier Series and in other applications, k being an integer that steps from 1 to n). One method would be to use, by means of two pinboards, the sine and cosine subroutines, shown on pages 6 to 8.

A better method, however, is the constant-increments one, which makes use of the chain formulas for $\sin ku_0$ and $\cos ku_0$:

$$\sin ku_0 = \sin[(k-1)+1]u_0$$
$$= \cos u_0 \sin(k-1)u_0 + \sin u_0 \cos(k-1)u_0$$

and

$$\cos ku_0 = \cos[(k-1)+1]u_0$$
$$= \cos u_0 \cos(k-1)u_0 - \sin u_0 \sin(k-1)u_0.$$

Using these formulas, we can program a subroutine that computes both $\sin ku_0$ and $\cos ku_0$ in less than one pinboard. The program requires 14 steps and 2 seconds for each value of k as compared to 26 steps and 8 seconds required in the two-pinboard method (see page 18).

If only $\cos ku_0$ is desired, the chain formula,

$$\cos ku_0 = 2\cos(k-1)u_0 \cos u_0 - \cos(k-2)u_0,$$

may be used.

Another example in which the constant-increments approach may be used is e^{ku_0}. Once e^{u_0} has been computed, e^{ku_0} can be found very simply for any number of integral increments merely by adding a few steps to the program, i.e., by raising e^{u_0} to successive integer powers.

The constant-increments approach can be used effectively with other functions whenever the argument of the function is to be advanced by multiples of an integer. Because of the word length of the E101 (12 digits), there is rarely, if ever, a build-up of errors sufficient to cause trouble in engineering computations.

4. Compound Functions

When a problem involves a compound function, such as

$$e^u \sin u \quad \text{or} \quad \sin hu + \log u,$$

in which each of the component functions can be expressed as a power series, there are two possible methods of attack.

One method is to compute each component function separately and then find their combined value.

Another, and in some cases a better, method is, first, to combine the power series of each component function into a single new power series and, then, to program the new series. As an example of this method, we can multiply the e^u series by the $\sin u$ series to arrive at a new series (see page 19). The program for the $e^u \sin u$ series requires only 8 steps and 10 seconds as compared to 26 steps and 16 seconds required in the two-pinboard operation.

5. Multi-Purpose Use of Subroutines

When a problem involves several functions, each of which can be approximated with sufficient accuracy by a finite polynomial, an economy in the number of pinboard steps required can be effected by the use of a multi-purpose subroutine, i.e., one that will suffice for all the functions involved. Such a multi-purpose subroutine for $\sin u$, $\cos u$, $\arctan u$, and e^{-u}, illustrated on pages 19-20, requires only 10 steps as contrasted with the need for 4 pinboards to compute the functions separately.

Since all four functions can be expressed as polynomials in u, with only the coefficients varying, the same pinboard steps can be used for the polynomials in u, with different memory addresses for the coefficients. The coefficients for each function are stored in a separate row in the memory, the E switch being used to select the proper row.

B. Techniques for Subroutines

1. Extension of Range and Accuracy

The programs for the subroutines given in Section II are valid, within the accuracy specified, only for the indicated range of the argument. Since, however, there are some problems that involve ranges of the argument greater than those indicated in the subroutines in Section II, additional programming can sometimes be used to cover the extended range.

Problems requiring an extension of range and/or an increase in accuracy are too varied and numerous, even to catalogue. Therefore, only general lines of approach can be indicated.

An example of a subroutine that can be easily extended in range is the exponential function, e^u. The basic program on page 12 is valid for $-1 \leq u \leq 1$, but the range can be extended by the use of either of the following identities:

$$e^u = \frac{1}{e^{-u}} \qquad \text{or} \qquad e^u = (e^{u/n})^n.$$

Furthermore, greater accuracy of e^u can be obtained if, for example, $e^{u/4}$ is computed and then raised to the fourth power.

An identical approach can be employed to extend the range and/or accuracy of the 10^u subroutine.

Another example of a subroutine that may be easily extended in range is the $\sin u$ subroutine, where the range, $-\pi/2 \leq u \leq \pi/2$, may be increased by the use of trigonometric identities, which replace u by an angle y in the first quadrant:

$$\sin u = \sin\left(\frac{\pi}{2} + y\right) = \cos y,$$

$$\sin u = \sin(\pi + y) = -\sin y,$$

$$\sin u = \sin\left(\frac{3\pi}{2} + y\right) = -\cos y,$$

$$\sin u = \sin(2\pi + y) = \sin y.$$

If u is a large angle, a standard procedure is to write

$$\sin u = \sin 2\pi\left(\frac{u}{2\pi} - \frac{u}{|u|}\left[\frac{|u|}{2\pi}\right]\right),$$

where the symbolism $\left[\frac{|u|}{2\pi}\right]$

means "the largest integer in $\frac{|u|}{2\pi}$."

Trigonometric identities can usually be employed to extend the range of any subroutine for a trigonometric function. For example, the range for the tan u subroutine can be significantly extended by use of the following identity:

$$\tan 2a = \frac{2 \tan a}{1 - \tan^2 a}.$$

In any of the subroutines that use Taylor Series, the range and/or accuracy may be extended by increasing the number of terms in the polynomial approximation. For example, to extend the range of the sin u subroutine, on page 6, to $-\pi < u < \pi$ and retain the accuracy of $\pm 0.000\ 05$, the following polynomial would be used:

$$\sin u = u - \frac{u^3}{3!} + \frac{u^5}{5!} - \frac{u^7}{7!} + \frac{u^9}{9!} - \frac{u^{11}}{11!} + \frac{u^{13}}{13!}.$$

Many of the subroutines in this booklet use polynomial approximations not derived from Taylor Series. It is nonetheless true that the range and accuracy of these subroutines may be extended by increasing the number of terms in the polynomial approximation. Hence, in order to aid the user in extending the range and/or accuracy of these subroutines, references which contain extensions of the polynomial approximation are given with each subroutine.

2. Programming

The subroutines in this booklet are based primarily on polynomial evaluation. The polynomials are written in terms of u, u^2, or $(u - a)/(u + a)$. The general method for programming such an approximating polynomial,

$$P(u) = \sum_{k=0}^{n} a_k u_k,$$

can best be shown by an example.

Suppose n = 4. Then the polynomial becomes

$$P(u) = a_0 + a_1 u + a_2 u^2 + a_3 u^3 + a_4 u^4,$$

which may be written as

$$P(u) = a_0 + u[a_1 + u[a_2 + u[a_3 + a_4 u]]].$$

This factored expression is the form leading to the most efficient programs for digital computers because $[a_3 + a_4 u]$ involves merely a_4 times u, added to a_3, which result becomes the new coefficient of u in $[a_2 + u[a_3 + a_4 u]]$. The process can be repeated, and we are thereby able to evaluate a high-degree polynomial by repeatedly performing a simple multiplication, followed by a simple addition.

Now suppose that the a's and the value of u, for which the polynomial is to be evaluated, are stored in memory addresses as follows: a_4 in 20, a_3 in 21, a_2 in 22, a_1 in 23, a_0 in 24, and u in 25; then the original polynomial P(u) can be easily evaluated by employing its factored form in the following program:

Step	*Instruction*	*Contents of Accumulator*
0	R 2 0	a_4
1	B	
2	× 2 5	$a_4 u$
3	+ 2 1	$a_3 + a_4 u$
4	B	
5	× 2 5	$u[a_3 + a_4 u]$
6	+ 2 2	$a_2 + u[a_3 + a_4 u]$
7	B	
8	× 2 5	$u[a_2 + u[a_3 + a_4 u]]$
9	+ 2 3	$a_1 + u[a_2 + u[a_3 + a_4 u]]$
10	B	
11	× 2 5	$u[a_1 + u[a_2 + u[a_3 + a_4 u]]]$
12	+ 2 4	$a_0 + u[a_1 + u[a_2 + u[a_3 + a_4 u]]] = P(u)$
13	P	Print P(u).

The foregoing program is linear (not looped). To save instruction space we can make use of the "H," "S," and "U" instructions to form a loop and thereby arrive at the following shorter program:

Step	*Instruction*	*Contents of Accumulator*
0	R 2 0	
1	H 1 1	at step 4, when
2	B	$F = 1$, is $a_3 + a_4 u$;
3	× 2 5	$F = 2$, is $a_2 + u[a_3 + a_4 u]$;
4	+ 2 F	$F = 3$, is $a_1 + u[a_2 + u[a_3 + a_4 u]]$;
5	S 1 4	$F = 4$, is $a_0 + u[a_1 + u[a_2 + u[a_3 + a_4 u]]]$.
6	U 0 2	
7	P	Print P(u).

So far, scaling has been neglected, but now we must consider the scaling that would be involved in such a problem. First, we must establish some rules of notation.

Suppose, for example, that we have a number, say 20, which we want to write in a memory address. If we enter this number into the machine as .200 000 000 00, where the decimal is the machine decimal, then we say that the given number, 20, times 10^{-2}, is in the memory. If we enter this number, 20, into the machine as .020 000 000 00, where the decimal is the machine decimal, then we say that the given number, 20, times 10^{-3}, is in the memory. In general, if we enter a given number into the E101 in such a way that the machine decimal is p places to the left of the real decimal, then we say that the given number times 10^{-p} is in the memory, B-register, or accumulator, as the case may be.

If, in the evaluation of the polynomial, u times 10^{-p} is in the memory, then a_4u and a_3 must be made, by correctly scaling a_4 and a_3, to agree in their power of 10, since they are to be added. For example, if a_4 times 10^8 is in the memory, then a_3 times 10^{8-p} must be in the memory since a_4u times 10^{8-p} will be in the accumulator and must be added to a_3. By following this procedure we can determine the appropriate power of 10 at which each of the a's should be written. (It is obvious that the printed answer will be of the same power of 10 as the constant a_0.)

Thus, in our example, if u times 10^{-1} is in the memory and if the a's have been stored as follows: a_4 times 10^0, a_3 times 10^{-1}, a_2 times 10^{-2}, a_1 times 10^{-3}, and a_0 times 10^{-4}, then all the steps used in evaluating the polynomial can be correctly performed.

Scaled Coefficients	*Partial Sums*
a_4 times 10^0	a_4u times 10^{-1}
a_3 times 10^{-1}	$[a_4u + a_3]$ times 10^{-1}
a_2 times 10^{-2}	$[a_4u + a_3]u$ times 10^{-2}
a_1 times 10^{-3}	$[[a_4u + a_3]u + a_2]$ times 10^{-2}
a_0 times 10^{-4}	$[[a_4u + a_3]u + a_2]u$ times 10^{-3}
	$[[[a_4u + a_3]u + a_2]u + a_1]$ times 10^{-3}
	$[[[a_4u + a_3]u + a_2]u + a_1]u$ times 10^{-4}

$$P(u) \text{ times } 10^{-4} = [[[a_4u + a_3]u + a_2]u + a_1]u + a_0 \text{ times } 10^{-4}$$

It is perhaps most efficient in the programming of polynomial evaluation to begin the scaling with the coefficient of the term involving the highest power of u, say a_n, and then to arrange the scale factors of the other coefficients, namely, a_{n-1}, a_{n-2}, a_{n-3}, $\cdots$, a_0, to be the same, respectively, as the scale factors of a_nu, a_nu^2, a_nu^3, $\cdots$, a_nu^n.

SECTION II

The Commonly Used Subroutines

A. Preliminary Information

This section contains programs of the more useful subroutines and furnishes programmed examples of 3 techniques that increase the utility of the E101.

Approximations by power series or rational functions have been used. The maximum error indicated for each program is the absolute difference between the function and the approximation.

For each subroutine these items have been given: the range of u (i.e., the independent variable of the function for which the program is usable), the method by which the given function is approximated, the constants used and their number, the number of temporary memory addresses, the scale factors, the switches used, the time required, and the detailed pinboard program.

The constants required for each program are shown exactly as they are stored in the memory, but, if in any given problem it should be convenient to use memory addresses other than those suggested, the memory addresses can be changed, provided the program steps referring to them are also changed.

The exit instruction back to the main program has been omitted except in the case of the conditional transfer.

The constants in the subroutines are scaled according to the scaling techniques discussed in Section I. Changes in the scaling of these constants should be made with caution.

Emphasis has been placed on economy of program steps rather than on economy of running time. Program steps have been saved by the use of "S" and "U" instructions to iterate the basic steps of each subroutine. Another approach, that of stretching the program out in a linear fashion, instead of iterating with "S" and "U," requires less running time but more program steps. The linear program for cos u, for instance, requires 18 steps and 3 seconds as against the iterative ("S" and "U") requirement of only 12 steps (a saving of 6 steps) and 4 seconds (a loss of only 1 second in time).

B. Subroutines

1. sin u ($-\pi/2 \leq u \leq \pi/2$, u in radians)
2. sin u ($-90° \leq u \leq 90°$ or $-\pi/2 \leq u \leq \pi/2$, u in degrees or radians, respectively)
3. sin A (for large angles, A, in radians or degrees)
4. cos u ($-\pi/2 \leq u \leq \pi/2$, u in radians)
5. cos u ($-90° \leq u \leq 90°$ or $-\pi/2 \leq u \leq \pi/2$, u in degrees or radians, respectively)
6. cos A (for large angles, A, in radians or degrees)
7. tan u ($-\pi/4 \leq u \leq \pi/4$ or $-1 < u < 1$, u in radians
8. arccos u and arcsin u ($0 < u < 1$, result in radians)
9. arccos u and arcsin u ($-1/2 \leq u \leq 1/2$) or $-\sqrt{2}/2 \leq u \leq \sqrt{2}/2$, result in radians)
10. arctan u ($-1 < u < 1$, result in radians)
11. arctan u ($-1 < u < 1$, result in radians)
12. arctan u ($0 < u < 999$, result in radians)
13. e^u ($-1 \leq u \leq 1$)
14. e^{-u} ($0 \leq u \leq 10$)
15. 10^u ($0 \leq u \leq 1$)
16. $\log_{10} u$ ($1 \leq u \leq 10$)
17. $\log_e u$ ($1 \leq u \leq 10$)
18. $\log_e u$ ($1 \leq u \leq 10$)
19. sinh u ($-4.5 \leq u \leq 4.5$)
20. cosh u ($-4.5 \leq u \leq 4.5$)
21. tanh u ($-2 \leq u \leq 2$)
22. $\sqrt{u}$
23. $\sqrt{u}$
24. Multiplication of complex numbers: $(a + bi)(c + di)$ for $-.9999 \leq a, b, c, d \leq .9999$
25. Division of complex numbers: $(a + bi)/(c + di)$ for $-.9999 \leq a, b, c, d \leq .9999$

1. sin u
(u in radians)

Range: $-\pi/2 \leq u \leq \pi/2$

Accuracy: Maximum error: $\pm 0.000\ 05$

Method: Reference—Taylor Series

$$\sin u = u - \frac{u^3}{3!} + \frac{u^5}{5!} - \frac{u^7}{7!} + \frac{u^9}{9!}$$

Number of Constants: 5

Number of Temporary Memory Addresses: 2

Switches Used: F (4 to 8)

Time on E101: 4 seconds

Special Feature: easier to extend in range and accuracy than other sine routine on this page.

Constants:

Constant	Scale Factor	As Constant Appears in Memory	Assumed Memory Address
−1/7!	3	0.198 412 698 41−	94
1/5!	1	0.083 333 333 33	95
−1/3!	−1	0.016 666 666 67−	96
1	−3	0.001 000 000 00	97
1/9!	5	0.275 573 192 24	98

Pinboard Program:

Starting point: u times 10^{-1} is in the accumulator. u should be shifted to meet this requirement.

0	W 91
1	B
2	× 91
3	W 92
4	R 98
5	H 14
6	B
7	× 92
8	+ 9F
9	S 17
10	U 06
11	B
12	× 91
13	
14	
15	

Result:

sin u times 10^{-4} is in the accumulator. The result can then be shifted to meet the requirements of any problem.

2. sin u
(u in radians or degrees)

Range: $-90° \leq u \leq 90°$ or $-\pi/2 \leq u \leq \pi/2$

Accuracy: Maximum error: $\pm 0.000\ 000\ 1$

Method: Reference—ElectroData. See also Hastings, pp. 138–140.

Let $x = u/90$ or $u/\pi/2$

then

$$\sin u = \sin \frac{\pi}{2} x = \sum_{k=0}^{4} C_{2k+1} x^{2k+1}$$

Number of Constants: 7

Number of Temporary Memory Addresses: 2

Switches Used: F (3 to 7)

Time on E101: 4.0 seconds

Constants:

Constant	Scale Factor	As Constant Appears in Memory	Assumed Memory Address
C_9	−1	0.000 015 148 42	97
C_7	−1	0.000 467 376 56−	93
C_5	−1	0.007 968 967 93	94
C_3	−1	0.064 596 371 11−	95
C_1	−1	0.157 079 631 85	96
1/90	2	1.111 111 111 10	98 (for u in degrees)
2/π	1	6.366 197 720 00	98 (for u in radians)

Pinboard Program:

Starting point: u (in degrees) times 10^{-2} or u (in radians) times 10^{-1} is in the accumulator. u should be shifted to meet this requirement.

0	B
1	× 98
2	W 91
3	B
4	× 91
5	W 92
6	R 97
7	H 13
8	B
9	× 92
10	+ 9F
11	S 16
12	U 08
13	B
14	× 91
15	

Result:

sin u times 10^{-1} is in the accumulator. The result can then be shifted to meet the requirements of any problem.

3. sin A

(for large angles)

Range: $-10{,}000° < A < 10{,}000°$ or
-100 radians $< A < 100$ radians

Accuracy: Maximum error: $\pm 0.000\ 000\ 2$

Method: Reference—ElectroData

$$\sin 2\pi u = \sum_{k=0}^{8} c_{2k+1} u^{2k+1}$$

$$\text{where } u = \frac{A}{2\pi} - \frac{A}{|A|}\left[\frac{|A|}{2\pi}\right]$$

Number of Constants: 10

Number of Temporary Memory Addresses: 2

Switches Used: F (1 to 10)

Time on E101: 8 seconds

Constants:

Constant	Scale Factor	As Constant Appears in Memory	Assumed Memory Address
c_{17}	−2	0.000 606 752 76	90
c_{15}	−2	0.006 406 975 47—	91
c_{13}	−2	0.037 414 651 35	92
c_{11}	−2	0.150 465 962 59—	93
c_9	−2	0.420 407 965 50	94
c_7	−2	0.767 019 335 43—	95
c_5	−2	0.816 047 826 56	96
c_3	−2	0.413 416 770 70—	97
c_1	−2	0.062 831 849 10	98
1/360	3	2.777 777 777 80	99 (for A in degrees)
$1/2\pi$	1	1.591 549 430 00	99 (for A in radians)

Pinboard Program:

Starting point: A (in degrees) times 10^{-4} or A (in radians) times 10^{-2} is in the B register. A should be shifted to meet this requirement.

0	× 99
1	A 12
2	A 21
3	B
4	W 88
5	× 88
6	W 89
7	R 90
8	H 11
9	B
10	× 89
11	+ 9F
12	S 19
13	U 09
14	× 88
15	

Result:

sin A times 10^{-2} is in the accumulator. The result can then be shifted to meet the requirements of any problem.

4. cos u

(u in radians)

Range: $-\pi/2 \leq u \leq \pi/2$

Accuracy: Maximum error: $\pm 0.000\ 05$

Method: Reference—Taylor Series

$$\cos u = 1 - \frac{u^2}{2!} + \frac{u^4}{4!} - \frac{u^6}{6!} + \frac{u^8}{8!}$$

Number of Constants: 5

Number of Temporary Memory Addresses: 1

Switches Used: F (4 to 8)

Time on E101: 4 seconds

Special Feature: easier to extend in range and accuracy than cos routine on p. 8

Constants:

Constant	Scale Factor	As Constant Appears in Memory	Assumed Memory Address
−1/6!	2	0.138 888 888 89—	84
1/4!	0	0.041 666 666 67	85
−1/2!	−2	0.005 000 000 00—	86
1	−4	0.000 100 000 00	87
1/8!	4	0.248 015 873 02	88

Pinboard Program:

Starting point: u times 10^{-1} is in the accumulator. u should be shifted to meet this requirement.

0	W 89
1	B
2	× 89
3	W 89
4	R 88
5	H 14
6	B
7	× 89
8	+ 8F
9	S 17
10	U 06
11	
12	
13	
14	
15	

Result:

cos u times 10^{-4} is in the accumulator. The result can then be shifted to meet the requirements of any problem.

5. cos u

(u in radians or degrees)

Range: $-90° \leq u \leq 90°$ or $-\pi/2 \leq u \leq \pi/2$

Accuracy: Maximum error: ±0.000 000 1

Method: Reference—ElectroData

Let $x = u/90$ or $u/\pi/2$

$$\text{Then } \cos u = \cos \frac{\pi}{2} x = \sum_{k=0}^{5} c_{2k} x^{2k}$$

Number of Constants: 7

Number of Temporary Memory Addresses: 1

Switches Used: F (4 to 9)

Time on E101: 4.5 seconds

Constants:

Constant	Scale Factor	As Constant Appears in Memory	Assumed Memory Address
1/90	2	1.111 111 111 10	83 (for u in degrees)
$2/\pi$	1	6.366 197 720 00	83 (for u in radians)
c_8	−1	0.000 091 785 85	84
c_6	−1	0.002 086 279 50−	85
c_4	−1	0.025 366 935 70	86
c_2	−1	0.123 370 053 81−	87
c_0	−1	0.100 000 000 00	88
c_{10}	−1	0.000 002 388 30−	89

Pinboard Program:

Starting point: u (in degrees) times 10^{-2} or u (in radians) times 10^{-1} is in the accumulator. u should be shifted to meet this requirement.

0	B
1	× 83
2	W 82
3	B
4	× 82
5	W 82
6	R 89
7	H 14
8	B
9	× 82
10	+ 8F
11	S 18
12	U 08
13	
14	
15	

Result:

cos u times 10^{-1} is in the accumulator. The result can then be shifted to meet the requirements of any problem.

6. cos A

(for large angles)

Range: $-10{,}000° < A < 10{,}000°$ or -100 radians $< A < 100$ radians

Accuracy: Maximum error: ±0.000 000 7

Method: Reference—ElectroData

$$\cos 2\pi u = \sum_{k=0}^{8} c_{2k} u^{2k}$$

$$\text{where } u = \frac{A}{2\pi} - \frac{A}{|A|}\left[\frac{|A|}{2\pi}\right]$$

Number of Constants: 10

Number of Temporary Memory Addresses: 1

Switches Used: F (1 to 9)

Time on E101: 7 seconds

Constants:

Constant	Scale Factor	As Constant Appears in Memory	Assumed Memory Address
c_{16}	−2	0.001 590 159 38	90
c_{14}	−2	0.015 108 285 35−	91
c_{12}	−2	0.077 136 624 40	92
c_{10}	−2	0.263 212 932 60−	93
c_8	−2	0.602 103 183 15	94
c_6	−2	0.854 504 972 05−	95
c_4	−2	0.649 388 111 90	96
c_2	−2	0.197 391 881 51−	97
c_0	−2	0.009 999 998 80	98
1/360	3	2.777 777 777 80	99 (for A in degrees)
$1/2\pi$	1	1.591 549 430 00	99 (for A in radians)

Pinboard Program:

Starting point: A (in degrees) times 10^{-4} or A (in radians) times 10^{-2} is in the accumulator. A should be shifted to meet this requirement.

0	B
1	× 99
2	A 12
3	A 21
4	B
5	W 89
6	× 89
7	W 89
8	H 11
9	R 90
10	B
11	× 89
12	+ 9F
13	S 18
14	U 0 10
15	

Result:

cos A times 10^{-2} is in the accumulator. The result can then be shifted to meet the requirements of any problem.

7. tan u

(u in radians)

Range: (a) $-\pi/4 \leq u \leq \pi/4$

(b) $-1 < u < 1$

Accuracy: Maximum error: (a) $\pm 0.000\ 000\ 7$

(b) $\pm 0.000\ 2$

Method: Reference—Taylor Series

$$\tan u = u + c_1 u^3 + c_2 u^5 + c_3 u^7 + \cdots + c_9 u^{19}$$

Number of Constants: 10

Number of Temporary Memory Addresses: 2

Switches Used: F (0 to 8)

Time on E101: 9 seconds

Constants:

Constant	Scale Factor	As Constant Appears in Memory	Assumed Memory Address
c_8	0	0.000 590 027 44	90
c_7	0	0.001 455 834 38	91
c_6	0	0.003 592 128 00	92
c_5	0	0.008 863 235 50	93
c_4	0	0.021 869 488 50	94
c_3	0	0.053 968 253 00	95
c_2	0	0.133 333 333 33	96
c_1	0	0.333 333 333 33	97
1	0	1.000 000 000 00	98
c_9	0	0.000 239 129 11	99

Pinboard Program:

Starting point: u times 10^0 is in the accumulator. u should be shifted to meet this requirement.

0	W 89
1	B
2	× 89
3	B
4	× 99
5	H 10
6	+ 9F
7	W 88
8	× 88
9	S 17
10	U 06
11	+ 98
12	A 21
13	B
14	× 89
15	

Result:

tan u times 10^{-1} is in the accumulator. The result can then be shifted to meet the requirements of any problem.

8. arccos u and arcsin u

(result in radians)

Range: $0 < u < 1$

Accuracy: Maximum error in either function:

$\pm 0.000\ 000\ 2$

Method: Reference—Hastings, page 162

$$\arccos u = \left(\sum_{k=0}^{6} c_k u^k \right) \sqrt{1 - u}$$

$$\arcsin u = \pi/2 - \arccos u$$

Number of Constants: 9 or 10

Number of Temporary Memory Addresses: 3

Switches Used: F (3 to 9)

Time on E101: 17 seconds for arccos u

Constants:

Constant	Scale Factor	As Constant Appears in Memory	Assumed Memory Address
$^*\pi/2$	−1	0.157 079 632 68	50 (necessary only for arcsin u)
1/2	0	0.500 000 000 00	51
−1	−2	0.010 000 000 00—	52
c_5	0	0.011 146 229 40—	53
c_4	0	0.026 899 948 20	54
c_3	0	0.048 802 504 30—	55
c_2	0	0.088 755 628 60	56
c_1	0	0.214 585 264 70—	57
c_0	0	1.570 796 172 80	58
c_6	0	0.002 295 964 80	59

Pinboard Program:

Starting point: u times 10^0 is in the accumulator. u should be shifted to meet this requirement.

	Pinboard 1	*Pinboard 2*	
0	W 49	× 51	
1	R 59	W 49	
2	H 13	A 12	
3	B	B	
4	× 49	R 49	
5	+ 5F	÷ 47	
6	S 18	× 51	
7	U 03	+ 47	
8	W 48	S 18	
9	R 49	U 03	
10	A 22	B	
11	+ 52	× 48	(arccos u times 10^{-1})
12	A 5	*A 5	
13	H 10	*+ 50	
14	B		(arcsin u times 10^{-1})
15	U 20		

Result:

The result ($0 < \arccos u < \pi/2$ or $0 < \arcsin u < \pi/2$) times 10^{-1} is in the accumulator. The result can then be shifted to meet the requirements of any problem.

*These items necessary only for arcsin u.

9. arccos u and arcsin u

(result in radians)

Range: (a) $-1/2 \leq u \leq 1/2$

(b) $-\sqrt{2}/2 \leq u \leq \sqrt{2}/2$

Accuracy: Maximum error in either function:

(a) $\pm 0.000\ 000\ 1$

(b) $\pm 0.000\ 3$

Method: Reference—Langdon, page 13

$$\arcsin u = \sum_{k=0}^{4} c_{2k+1} u^{2k+1}$$

$$\arccos u = \pi/2 - \arcsin u$$

Number of Constants: 5 or 6

Number of Temporary Memory Addresses: 2

Switches Used: F (0 to 4)

Time on E101: 5 seconds for arccos u

Special Feature: much shorter over a limited range than routine on p. 9.

Constants:

Constant	Scale Factor	As Constant Appears in Memory	Assumed Memory Address
c_7	—1	0.003 822 898 56	70
c_5	—1	0.007 574 473 31	71
c_3	—1	0.016 663 218 82	72
c_1	—1	0.100 000 044 17	73
c_9	—1	0.005 315 614 62	74
*$\pi/2$	—1	0.157 079 632 68	75 (necessary only for arccos u)

Pinboard Program:

Starting point: u times 10^0 is in the accumulator. u should be shifted to meet this requirement.

```
 0   W 76
 1   B
 2   × 76
 3   W 77
 4   R 74
 5   H 10
 6   B
 7   × 77
 8   + 7F
 9   S 13
10   U 06
11   B
12   × 76  (arcsin u times 10⁻¹)
*13  A 5
*14  + 75  (arccos u times 10⁻¹)
15
```

Result:

The result ($-\pi/6 \leq \arcsin u \leq \pi 6$ or $\pi/3 \leq \arccos u \leq 2\pi/3$) times 10^{-1} is in the accumulator. The result can then be shifted to meet the requirements of any problem.

*These items necessary only for arccos u.

10. arctan u

(result in radians)

Range: $-1 < u < 1$

Accuracy: Maximum error: $\pm 0.000\ 002$

Methods: Reference—Hastings, page 135

$$\arctan u = \sum_{k=0}^{5} c_{2k+1} u^{2k+1}$$

Number of Constants: 6

Number of Temporary Memory Addresses: 2

Switches Used: F (0 to 4)

Time on E101: 6 seconds

Constants:

Constant	Scale Factor	As Constant Appears in Memory	Assumed Memory Address
c_7	—1	0.011 643 287 00—	90
c_5	—1	0.019 354 346 00	91
c_3	—1	0.033 262 347 00—	92
c_1	—1	0.099 997 726 00	93
c_{11}	—1	0.001 172 120 00—	94
c_9	—1	0.005 265 332 00	95

Pinboard Program:

Starting point: u times 10^0 is in the accumulator. u should be shifted to meet this requirement.

```
 0   B
 1   W 96
 2   × 96
 3   B
 4   × 94
 5   + 95
 6   H 10
 7   W 97
 8   × 97
 9   + 9F
10   S 13
11   U 07
12   B
13   × 96
14
15
```

Result:

arctan u times 10^{-1} is in the accumulator. The result can then be shifted to meet the requirements of any problem.

11. arctan u
(result in radians)

Range: $-1 < u < 1$

Accuracy: Maximum error: $\pm 0.000\ 08$

Method: Reference—Hastings, page 133

$$\arctan u = \sum_{k=0}^{3} c_{2k+1} u^{2k+1}$$

Number of Constants: 5

Number of Temporary Memory Addresses: 3

Switches Used: F (0 to 4)

Time on E101: 5 seconds

Constants:

Constant	Scale Factor	As Constant Appears in Memory	Assumed Memory Address
c_5	−1	0.014 627 660 00	80
c_3	−1	0.032 118 190 00—	81
c_1	−1	0.099 921 500 00	82
zero	0	0.000 000 000 00	83
c_7	−1	0.003 899 290 00—	90

Pinboard Program:

Starting point: u times 10^0 is in the accumulator. u should be shifted to meet this requirement.

0	W 93
1	B
2	× 93
3	W 92
4	W 91
5	H 10
6	B
7	× 9F
8	+ 8F
9	S 13
10	U 06
11	
12	
13	
14	
15	

Result:

arctan u times 10^{-1} is in the accumulator. The result can then be shifted to meet the requirements of any problem.

12. arctan u
(result in radians)

Range: $0 < u < 999.0$

Accuracy: Maximum error: $\pm 0.000\ 09$

Method: Reference—Hastings, page 133

$$\arctan u = \pi/4 + \sum_{k=0}^{3} c_{2k+1} \left(\frac{u-1}{u+1}\right)^{2k+1}$$

Number of Constants: 7

Number of Temporary Memory Addresses: 3

Switches Used: F (0 to 4)

Time on E101: 6 seconds

Special Feature: true for greater range of argument than other routine on this page and the one on p. 10.

Constants:

Constant	Scale Factor	As Constant Appears in Memory	Assumed Memory Address
c_5	−1	0.014 627 660 00	80
c_3	−1	0.032 118 190 00—	81
c_1	−1	0.099 921 500 00	82
$\pi/4$	−1	0.078 539 800 00	83
2	−3	0.002 000 000 00	84
1	−3	0.001 000 000 00	85
c_7	−1	0.003 899 290 00—	90

Pinboard Program:

Starting point: u times 10^{-3} is in the accumulator. u should be shifted to meet this requirement.

0	+ 85
1	B
2	− 84
3	÷ 93
4	R 93
5	B
6	× 93
7	W 91
8	W 92
9	H 10
10	B
11	× 9F
12	+ 8F
13	S 13
14	U 0 10
15	

Result:

arctan u times 10^{-1} is in the accumulator. The result can then be shifted to meet the requirements of any problem.

13. e^u

Range: $-1 \leq u \leq 1$

Accuracy: Maximum error: ±0.000 005

Method: Reference—ElectroData

$$e^u = \sum_{k=0}^{6} c_k u^k$$

Number of Constants: 7

Number of Temporary Memory Addresses: 2

Switches Used: F (0 to 6)

Time on E101: 6 seconds

Constants:

Constant	Scale Factor	As Constant Appears in Memory	Assumed Memory Address
c_5	1	0.086 805 555 50	60
c_4	0	0.041 635 664 70	61
c_3	—1	0.016 649 305 60	62
c_2	—2	0.005 000 062 00	63
c_1	—3	0.001 000 021 70	64
c_0	—4	0.000 100 000 00	65
c_6	2	0.143 849 206 30	66

Pinboard Program:

Starting point: u times 10^{-1} is in the accumulator. u should be shifted to meet this requirement.

0	W 67
1	R 66
2	H 10
3	B
4	× 67
5	+ 6F
6	S 15
7	U 03
8	A 13
9	B
10	W 68
11	× 68
12	
13	
14	
15	

Result:

e^{2u} times 10^{-2} is in the accumulator; e^u times 10^{-1} is in memory address 68. The results can then be shifted to meet the requirements of any problem.

14. e^{-u}

Range: $0 \leq u \leq 10$

Accuracy: Maximum error: ±0.000 03

Method: Reference—Hastings, page 182

$$e^{-u} = 1/\left(\sum_{k=0}^{8} c_k u^k\right)^2$$

Number of Constants: 10

Number of Temporary Memory Addresses: 1

Switches Used: F (0 to 8)

Time on E101: 8 seconds

Constants:

Constant	Scale Factor	As Constant Appears in Memory	Assumed Memory Address
c_7	+4	0.012 158 000 00	70
c_6	+3	0.022 049 330 00	71
c_5	+2	0.027 728 680 00	72
c_4	+1	0.026 697 599 60	73
c_3	0	0.020 341 621 28	74
c_2	—1	0.012 562 628 30	75
c_1	—2	0.004 998 207 00	76
c_0	—3	0.001 000 000 00	77
1	—6	0.000 001 000 00	78
c_8	+5	0.007 129 530 25	79

Pinboard Program:

Starting point: u times 10^{-1} is in the accumulator. u should be shifted to meet this requirement.

0	W 89
1	R 79
2	H 10
3	B
4	× 89
5	+ 7F
6	S 17
7	U 03
8	W 89
9	B
10	× 89
11	B
12	R 78
13	÷ 89
14	
15	

Result:

e^{-u} times 10^0 is in memory address 89. The result can then be shifted to meet the requirements of any problem.

15. 10^u

Range: $0 \leq u \leq 1$

Accuracy: Maximum error: $\pm 0.000\ 000\ 05$

Method: Reference—Hastings, page 144

$$10^u = [1 + c_1u + c_2u^2 + \cdots + c_7u^7]^2$$

Number of Constants: 8

Number of Temporary Memory Addresses: 1

Switches Used: F (2 to 8)

Time on E101: 7 seconds

Constants:

Constant	Scale Factor	As Constant Appears in Memory	Assumed Memory Address
c_7	0	0.000 932 642 67	90
c_6	0	0.002 554 917 96	91
c_5	0	0.017 421 119 88	92
c_4	0	0.072 951 736 66	93
c_3	0	0.254 393 574 84	94
c_2	0	0.662 730 884 29	95
c_1	0	1.151 292 776 03	96
1	0	1.000 000 000 00	97

Pinboard Program:

Starting point: u times 10^0 is in the accumulator. u should be shifted to meet this requirement.

```
0   H 12
1   B
2   × 90
3   + 91
4   W 98
5   × 98
6   + 9F
7   S 17
8   U 04
9   W 98
10  A 21
11  B
12  × 98
13
14
15
```

Result:

10^u times 10^{-1} is in the accumulator. The result can then be shifted to meet the requirements of any problem.

16. $\log_{10} u$

Range: $1 \leq u \leq 10$

Accuracy: Maximum error: $\pm 0.000\ 05$

Method: Reference—Hastings, page 126

$$\log_{10} u = 1/2 + \sum_{k=0}^{2} c_{2k+1} \left(\frac{u - \sqrt{10}}{u + \sqrt{10}}\right)^{2k+1}$$

Number of Constants: 6

Number of Temporary Memory Addresses: 2

Switches Used: F (0 to 3)

Time on E101: 5 seconds

Constants:

Constant	Scale Factor	As Constant Appears in Memory	Assumed Memory Address
c_5	—1	0.025 432 750 00	50
c_3	—1	0.027 738 390 00	60
c_1	—1	0.086 902 860 00	61
1/2	—1	0.050 000 000 00	62
$\sqrt{10}$	—3	0.003 162 278 00	63
$2\sqrt{10}$	—3	0.006 324 556 00	64

Pinboard Program:

Starting point: u times 10^{-3} is in the accumulator. u should be shifted to meet this requirement.

```
0   + 63
1   B
2   − 64
3   ÷ 52
4   R 52
5   B
6   × 52
7   W 51
8   H 10
9   B
10  × 5F
11  + 6F
12  S 12
13  U 09
14
15
```

Result:

$\log_{10} u$ times 10^{-1} is in the accumulator. The result can then be shifted to meet the requirements of any problem.

17. $\log_e u$

Range: $1 \leq u \leq 10$

Accuracy: Maximum error: $\pm 0.000\ 05$

Method: Reference—Hastings, page 126

$$\log_e u = \frac{1}{M} \log_{10} u$$

$$= \frac{1}{M}\left[\frac{1}{2} + \sum_{k=0}^{2} c_{2k+1} \left(\frac{u - \sqrt{10}}{u + \sqrt{10}}\right)^{2k+1}\right]$$

Number of Constants: 6

Number of Temporary Memory Addresses: 2

Switches Used: F (0 to 3)

Time on E101: 5 seconds

Constants:

Constant	Scale Factor	As Constant Appears in Memory	Assumed Memory Address
c_5/M	—1	0.058 561 070 00	50
c_3/M	—1	0.063 870 000 00	60
c_1/M	—1	0.200 101 230 00	61
$1/2M$	—1	0.115 129 260 00	62
$\sqrt{10}$	—3	0.003 162 278 00	63
$2\sqrt{10}$	—3	0.006 324 556 00	64

Pinboard Program:

The program is the same as that for $\log_{10} u$.

18. $\log_e u$

Range: $1 \leq u \leq 10$

Accuracy: Maximum error: $\pm 0.000\ 000\ 003$

Method: Reference—Langdon, page 10

$$\log_e u = \log_e c + \sum_{k=0}^{6} c_{2k+1} y^{2k+1}, \text{ where}$$

$$y = \frac{u - c}{u + c}$$

Number of Constants: 10

Number of Temporary Memory Addresses: 2

Switches Used: F (0 to 6)

Time on E101: 10 seconds

Special Feature: increased accuracy over other log routine on this page.

Constants:

Constant	Scale Factor	As Constant Appears in Memory	Assumed Memory Address
c_{11}	—2	0.000 572 283 27	80
c_9	—2	0.002 503 410 93	81
c_7	—2	0.002 824 335 71	82
c_5	—2	0.004 001 930 33	83
c_3	—2	0.006 666 617 10	84
c_1	—2	0.020 000 000 37	85
$\log_e c$	—2	0.011 512 925 46	86
c	—2	0.031 622 776 60	87
2c	—2	0.063 245 553 20	88
c_{13}	—2	0.004 105 970 44	89

Pinboard Program:

Starting point: u times 10^{-2} is in the accumulator. u should be shifted to meet this requirement.

Pinboard 1		Pinboard 2	
0	+ 87	0	B
1	B	1	× 99
2	− 88	2	+ 86
3	÷ 99	3	
4	R 99	4	
5	B	5	
6	× 99	6	
7	B	7	
8	H 10	8	
9	R 89	9	
10	W 98	10	
11	× 98	11	
12	+ 8F	12	
13	S 15	13	
14	U 0 10	14	
15	U 20	15	

Result:

$\log_e u$ times 10^{-2} is in the accumulator. The result can then be shifted to meet the requirements of any problem.

19. sinh u

Range: $-4.5 \leq u \leq 4.5$

Accuracy: Maximum error: ±0.000 05

Method: Reference—Taylor Series

$$\sinh u = u + \frac{u^3}{3!} + \frac{u^5}{5!} + \cdots + \frac{u^{2n-1}}{(2n-1)!},$$

where n = 11

Number of Constants: 11

Number of Temporary Memory Addresses: 2

Switches Used: F (0 to 9)

Time on E101: 10 seconds

Constants:

Constant	Scale Factor	As Constant Appears in Memory	Assumed Memory Address
1/21!	17	0.001 957 294 11	69
1/19!	15	0.008 220 635 25	70
1/17!	13	0.028 114 572 54	71
1/15!	11	0.076 471 637 32	72
1/13!	9	0.160 590 438 38	73
1/11!	7	0.250 521 083 85	74
1/9!	5	0.275 573 192 24	75
1/7!	3	0.198 412 698 41	76
1/5!	1	0.083 333 333 33	77
1/3!	−1	0.016 666 666 67	78
1	−3	0.001 000 000 00	79

Pinboard Program:

Starting point: u times 10^{-1} is in the accumulator. u should be shifted to meet this requirement.

0	B
1	W 68
2	× 68
3	B
4	× 69
5	H 10
6	+ 7F
7	W 67
8	× 67
9	S 18
10	U 06
11	+ 79
12	B
13	× 68
14	
15	

Result:

sinh u times 10^{-4} is in the accumulator. The result can then be shifted to meet the requirements of any problem.

20. cosh u

Range: $-4.5 \leq u \leq 4.5$

Accuracy: Maximum error: ±0.000 05

Method: Reference—Taylor Series

$$\cosh u = 1 + \frac{u^2}{2!} + \frac{u^4}{4!} + \cdots + \frac{u^{2n}}{(2n)!},$$

where n = 10

Number of Constants: 11

Number of Temporary Memory Addresses: 1

Switches Used: F (1 to 10)

Time on E101: 9 seconds

Constants:

Constant	Scale Factor	As Constant Appears in Memory	Assumed Memory Address
1	−4	0.000 100 000 00	69
1/20!	16	0.004 110 317 62	70
1/18!	14	0.015 619 206 97	71
1/16!	12	0.047 794 773 32	72
1/14!	10	0.114 707 455 98	73
1/12!	8	0.208 767 569 88	74
1/10!	6	0.275 573 192 24	75
1/8!	4	0.248 015 873 02	76
1/6!	2	0.138 888 888 89	77
1/4!	0	0.041 666 666 67	78
1/2!	−2	0.005 000 000 00	79

Pinboard Program:

Starting point: u times 10^{-1} is in the accumulator. u should be shifted to meet this requirement.

0	B
1	W 68
2	× 68
3	B
4	× 70
5	H 11
6	+ 7F
7	W 68
8	× 68
9	S 19
10	U 06
11	+ 69
12	
13	
14	
15	

Result:

cosh u times 10^{-4} is in the accumulator. The result can then be shifted to meet the requirements of any problem.

21. tanh u

Range: $-2 \leq u \leq 2$

Accuracy: Maximum error: $\pm 0.000\ 8$

Method: Reference—ElectroData

$$\tanh u = \frac{a}{d + .1}, \quad \text{where } d = \frac{c + .245}{c} b,$$

$$c = b + .105, b = a^2, \text{and } a = .10u$$

Number of Constants: 3

Number of Temporary Memory Addresses: 3

Switches Used: none

Time on E101: 1.9 seconds

Constants:

Constant	Scale Factor	As Constant Appears in Memory	Assumed Memory Address
.105	0	0.105 000 000 00	00
.245	0	0.245 000 000 00	01
.100	0	0.100 000 000 00	02

Pinboard Program:

Starting point: u times 10^{-1} is in the accumulator. u should be shifted to meet this requirement.

0	B
1	W 03
2	× 03
3	W 04
4	+ 00
5	B
6	+ 01
7	÷ 05
8	R 04
9	B
10	× 05
11	+ 02
12	B
13	R 03
14	÷ 05
15	

Result:

tanh u times 10^0 is in memory address 05. The result can then be shifted to meet the requirements of any problem.

22. $\sqrt{u}$

(see note below)

Range: Unrestricted, provided u times 10^{-n} is scaled so that n is an even integer and u times 10^{-n} is less than 1.

Accuracy: Maximum error in $\sqrt{u}$ times $10^{-n/2}$ is less than $\pm 0.000\ 000\ 000\ 01$, if $b = 6$ in step 9.

Method: Newton approximation:

$$x_{i+1} = \frac{1}{2}\left(x_i + \frac{u}{x_i}\right)$$

Number of Constants: 1

Number of Temporary Memory Addresses: 2

Switches Used: F (0 to 6)

Time on E101: 1.17 seconds per iteration

Special Feature: fewer program steps than routine on p. 17.

Constants:

Constant	Scale Factor	As Constant Appears in Memory	Assumed Memory Address
1/2	0	0.500 000 000 00	90

Pinboard Program:

Starting point: u times 10^{-n} is in the accumulator. u should be shifted to meet this requirement.

0	B
1	× 90
2	W 91
3	H 10
4	R 91
5	÷ 92
6	× 90
7	+ 92
8	B
9	S 1b
10	U 04
11	
12	
13	
14	
15	

Result:

$\sqrt{u}$ times $10^{-n/2}$ is in the accumulator and in the B register.

Note: u times 10^{-n} must be scaled and placed in the accumulator before control is transferred to the sub-routine.

If the leading digit of u times 10^{-n} is the first or second digit after the machine decimal point, then 7 iterations (obtained by setting "b" = 6 in step 9) will give the accuracy specified above.

Study of the range of numbers whose square roots are to be taken will give some indication of the number of iterations required to achieve the desired accuracy; the "b" of step 9 can be pinned accordingly.

23. $\sqrt{u}$

(see note below)

Range: Unrestricted, provided u times 10^{-n} is scaled so that n is an even integer and u times 10^{-n} is less than 1.

Accuracy: Maximum error in $\sqrt{u}$ is 5 in the least significant digit of the accumulator.

Method: Newton approximation:

$$x_{i+1} = \frac{1}{2}\left(x_i + \frac{u}{x_i}\right)$$

$$x_{i+1} \doteq \sqrt{u} \text{ when } |x_i - x_{i+1}| \leq 0.000\ 000\ 000\ 05$$

Number of Constants: 1

Number of Temporary Memory Addresses: 3

Switches Used: none

Time on E101: 6 seconds

Constants:

Constant	Scale Factor	As Constant Appears in Memory	Assumed Memory Address
1/2	0	0.500 000 000 00	90

Pinboard Program:

Starting point: u times 10^{-n} is in the accumulator. u should be shifted to meet this requirement.

0	B
1	× 90
2	W 91
3	R 91
4	÷ 92
5	× 90
6	+ 92
7	W 93
8	B
9	− 92
10	− 92
11	A 4
12	A 24
13	C 03
14	R 93
15	

Result:

$\sqrt{u}$ times $10^{-n/2}$ is in the accumulator, in the B register, and in memory address 93.

Note: u times 10^{-n} must be scaled and placed in the accumulator before control is transferred to the subroutine.

If the leading digit of u times 10^{-n} is the first or second digit after the machine decimal point, then a good first approximation to $\sqrt{u}$ times $10^{-n/2}$ is obtained, and consequently the sequence of approximations converges very rapidly.

24. Multiplication of complex numbers

Range: $u + vi = (a + bi)(c + di)$

$-.9999 \leq a, b, c, d \leq .9999$

Accuracy: Depends on the accuracy of a, b, c, and d and on the scale factors at which these numbers are entered.

Method: $(a + bi)(c + di) = (ac - bd) + (ad + bc)i$

Number of Constants: 4

Number of Temporary Memory Addresses: 2

Switches Used: none

Time on E101: 2 seconds

Constants:

Constant	Scale Factor	As Constant Appears in Memory	Assumed Memory Address
a	Both a and b must be entered with the		20
b	same scale factor; similarly, c and d		21
c	must be entered with the same scale		22
d	factor.		23

Pinboard Program:

Starting point: a, b, c, and d in the assumed memory addresses.

0	R 21
1	B
2	× 23
3	W 24
4	× 22
5	W 25
6	R 20
7	B
8	× 23
9	+ 25
10	W 20
11	× 22
12	− 24
13	W 21
14	
15	

Result:

The real part of the result, (ac − bd), is stored in memory address 21 and in the accumulator. The imaginary part of the result, (ad + bc), is stored in memory address 20.

25. Division of complex numbers

Range: $u + vi = (a + bi)/(c + di)$

$-.9999 \leq a, b, c, d \leq .9999$

Accuracy: Depends on the accuracy of a, b, c, and d and on the scale factors at which these numbers are entered.

Method: $$\frac{a + bi}{c + di} = \frac{ac + bd}{c^2 + d^2} + \frac{bc - ad}{c^2 + d^2} i$$

Number of Constants: 4

Number of Temporary Memory Addresses: 2

Switches Used: none

Time on E101: 4.5 seconds

Constants:

Constant	Scale Factor	As Constant Appears in Memory	Assumed Memory Address
a	Both a and b must be entered with the		20
b	same scale factor; similarly, c and d		21
c	must be entered with the same scale		22
—d	factor.		23

Pinboard Program:

Starting point: a, b, c, and $-d$ in assumed memory addresses.

	Pinboard 1	*Pinboard 2*
0	R 21	R 22
1	B	B
2	× 23	× 22
3	W 24	W 24
4	× 22	R 23
5	W 25	B
6	R 20	× 23
7	B	+ 24
8	× 23	B
9	+ 25	R 20
10	W 20	÷ 20
11	× 22	R 21
12	− 24	÷ 21
13	W 21	
14	U 20	
15		

Result:

The real part of the result, $(ac + bd)/(c^2 + d^2)$, is stored in memory address 21. The imaginary part, $(bc - ad)/(c^2 + d^2)$, is stored in memory address 20.

C. Techniques Which Increase the Utility of the E101

1. Example of Constant Increments Approach
   $\sin ku_0$ and $\cos ku_0$ for unit increments of k
2. Example of Compound Functions Approach
   $e^u \sin u$
3. Example of Multi-purpose Approach

1. Example of Constant Increments Approach sin ku_0 and cos ku_0 for unit increments of k

Range: All u_0 and k within the limits of the E101.

Accuracy: Depends on the number of increments used. An example of the maximum error to be expected is as follows: $u_0 = .04$ radians; number of increments $= 160$; maximum error of $\pm 2 \times 10^{-7}$, assuming $\sin u_0$ and $\cos u_0$ are accurate to 10^{-10}.

Method:

$$\sin ku_0 = \cos u_0 \sin (k - 1)u_0 + \sin u_0 \cos (k - 1)u_0$$

$$\cos ku_0 = \cos u_0 \cos (k - 1)u_0 - \sin u_0 \sin (k - 1)u_0$$

Number of Constants: 4

Number of Temporary Memory Addresses: 2

Switches Used: none

Time on E101: 2 seconds for each k

Constants:

Constant	Scale Factor	As Constant Appears in Memory	Assumed Memory Address
sin o	0	0.000 000 000 00	40
sin u_0	0	sin u_0 times 10^0	41
cos u_0	0	cos u_0 times 10^0	42
cos o	0	1.000 000 000 00	43

Pinboard Program:

Starting point: sin 0, sin u_0, cos 0, and cos u_0 times 10^0 are in assumed memory addresses.

0	R 41
1	B
2	× 43
3	W 44
4	× 40
5	W 45
6	R 42
7	B
8	× 40
9	+ 44
10	W 40
11	× 43
12	− 45
13	W 43
14	
15	

Result:

sin ku_0 times 10^0 appears in memory address 40; cos ku_0 times 10^0, in memory address 43.

2. Example of Compound Functions Approach
$e^u \sin u$

Range: $-2 \leq u \leq 2$

Accuracy: Maximum error: $\pm 0.000\ 1$

Method: Reference—Taylor Series

$$e^u \sin u = u + u^2 + \frac{2}{3!}u^3 + 0u^4 - \frac{4}{5!}u^5$$
$$- \frac{8}{6!}u^6 - \frac{8}{7!}u^7 + 0u^8 + \frac{16}{9!}u^9$$
$$+ \frac{32}{10!}u^{10} + \frac{32}{11!}u^{11}$$

Number of Constants: 11

Number of Temporary Memory Addresses: 1

Switches Used: F (0 to 10)

Time on E101: 10 seconds

Constants:

Constant	Scale Factor	As Constant Appears in Memory	Assumed Memory Address
32/10!	5	0.881 834 215 04	30
16/9!	4	0.440 917 107 52	31
0	0	0.000 000 000 00	32
−8/7!	2	0.158 730 158 72—	33
−8/6!	1	0.111 111 111 11—	34
−4/5!	0	0.033 333 333 32—	35
0	0	0.000 000 000 00	36
2/3!	−2	0.003 333 333 33	37
1	−3	0.001 000 000 00	38
1	−4	0.000 100 000 00	39
32/11!	6	0.801 667 468 48	49

Pinboard Program:

Starting point: u times 10^{-1} is in the accumulator. u should be shifted to meet this requirement.

0	B
1	× 49
2	H 10
3	+ 3F
4	W 48
5	× 48
6	S 19
7	U 03
8	
9	
10	
11	
12	
13	
14	
15	

Result:

$e^u \sin u$ times 10^{-5} is in the accumulator. The result can then be shifted to meet the requirements of any problem.

3. Example of Multi-Purpose Approach

This program, initially mentioned on page 2 of Section I, is an example of a multi-purpose subroutine that can be used to compute sin u, cos u, arctan u, and e^{-u}. It is assumed here that these four functions occur in one problem. The illustration shows how to arrange the memory and how to transfer into, and out of, the multi-purpose subroutine.

Further, the illustration assumes that sin u is to be called for in pinboard 1; cos u in pinboard 2; arctan u in pinboard 4; and e^{-u} in pinboard 6.

Constants:

The constants for sin u are stored in row 1 of the memory; the constants for cos u are stored in row 2 of the memory; the constants for arctan u are stored in row 4 of the memory; and the constants for e^{-u} are stored in row 6 of the memory. It is important to note that the number of the row in which the constants for a particular function are stored is the same as the number of the pinboard in which the function is called for.

MEMORY*

	0	1	2	3	4	5	6	7	8	9
0										
1	1/9!	0	−1/7!	0	1/5!	0	−1/3!	0	1	0
2	0	1/8!	0	−1/6!	0	1/4!	0	−1/2!	0	1
3										
4	0	0	C_7	0	C_5	0	C_3	0	C_1	0
5										
6	0	C_8	C_7	C_6	C_5	C_4	C_3	C_2	C_1	C_0

*For range, accuracy, scale factor, time on E101, and constants as they appear in memory, of sin u, see page 6; of cos u, see pages 7 and 8; of arctan u, see pages 10 and 11; and of e^{-u}, see page 12.

Program:

The program for the multi-purpose subroutine is shown in pinboard 8. We assume, of course, that each time we enter the subroutine, u has been properly scaled and placed in the accumulator.

Just as in any multiple use of a subroutine, the E switch is homed to the number of the pinboard to which the control is to return when the subroutine has ended. But, because of the arrangement of the memory, as designated above, the E switch is also used to reference the proper row of constants. The F switch is then used to iterate along the chosen row.

Pinboard No.

Step No.	1	2	3	4	5	6	7	8
0								H 1 2
1								B
2						H 0 6		× E 0
3				H 0 4		U 8 0		+ E 1
4				U 8 0		W 9 9		W 9 9
5				(arctan u)		B		× 9 9
6						× 9 9		+ E F
7	H 0 1					B		S 1 9
8	U 8 0					R 6 8		U 0 4
9	(sin u)					÷ 9 9		U E *
10		H 0 2				(e^{-u})		
11		U 8 0						
12		(cos u)						
13								
14								
15								

This program is an illustration of a technique. Other polynomial approximations may, of course, be adapted to the same general procedure. However, one word of advice may be in order: it is perhaps most convenient first to arrange the memory and then adapt the program to exit from the proper pinboards, rather than to arrange the memory to correspond to the program exits.

REFERENCES

A

Handbooks Containing Series for Common Functions

Burington, Richard S., *Handbook of Mathematical Tables and Formulas*. Sandusky, Ohio: Handbook Publishers, Inc. 1954.

Dwight, Herbert B., *Tables of Integrals and Other Mathematical Data*. New York: The Macmillan Co., 1934.

Handbook of Chemistry and Physics. Cleveland, Ohio: Chemical Rubber Publishing Co., 1951.

Jahnke, E., and Emde, F., *Tables of Functions with Formulae and Curves*. New York: Dover Publications, 1945.

Peirce, B. O., *A Short Table of Integrals*. Boston: Ginn and Co., 1929.

B

General Studies in Functional Approximation

Approximations in Numerical Analysis. Santa Monica, California: The Rand Corporation.

Clenshaw, C. W., "Polynomial Approximations to Elementary Functions," *Mathematical Tables and Other Aids to Computation*. Washington, D. C.: The National Research Council, 1952.

Hastings, Jr., Cecil, *Approximations for Digital Computers*. Princeton, N. J.: Princeton University Press, 1955.

Lanczos, C., "Trigonometric Interpolation of Empirical and Analytical Functions," *Journal of Mathematics and Physics*, Vol. 17, 1938.

Langdon, Lyle, "Approximating Functions for Digital Computers" (dittoed copy). Detroit, Michigan: Wayne University Computation Laboratory, 1956.

Milne, William E., *Numerical Calculus*. Princeton, N. J.: Princeton University Press, 1950.

Scarborough, James B., *Numerical Mathematical Analysis*. Baltimore, Maryland: Johns Hopkins University Press, 1930.

Tables of Chebyshev Polynomials. Washington, D. C.: U. S. Government Printing Office, 1952.

Teichroew, D., "Use of Continued Fractions in High Speed Computing," *Mathematical Tables and Other Aids to Computation*. Washington, D. C.: The National Research Council, 1952.

C

ElectroData Division of Burroughs Corporation, 460 Sierra Madre Villa, Pasadena, Calif.

Burroughs Corporation
ELECTRODATA DIVISION
460 SIERRA MADRE VILLA,
PASADENA, CALIFORNIA

OFFICES

BOSTON
NEW YORK
PHILADELPHIA
PITTSBURGH
ROCHESTER
WASHINGTON

DENVER
LOS ANGELES
SAN FRANCISCO
SEATTLE

CHICAGO
CLEVELAND
DALLAS
DETROIT
KANSAS CITY
MINNEAPOLIS-ST. PAUL

OTTAWA
MONTREAL
TORONTO

Printed in U.S.A.