then the ϕ's and ψ's defined in this way form a solution to (d), the ϕ's being mutually orthogonal and the ψ's being mutually orthogonal. The computations can be carried on sequentially. Let λ_1 be the largest eigenvalue of K, ϕ_1 a corresponding normalized eigenfunction and ψ_1 defined as above. Having found ϕ_1 and ψ_1 we form the symmetric kernel, K_1 associated with the residual function $z(x, y) - \phi_1(x)\psi_1(y)$. Then, the largest eigenvalue of K_1 is the next largest eigenvalue of K and we take ϕ_2 to be the corresponding normalized eigenfunction with ψ_2 defined as above. We continue in this manner until we have the required number of functions. Incidentally, for $n = 1$, a variational technique yields the necessary conditions for an extremum:

$$\phi(x) = \int_0^1 z(x, y)\psi(y)dy \Big/ \int_0^1 \psi^2(y)dy,$$

$$\psi(y) = \int_0^1 z(x, y)\phi(x)dx \Big/ \int_0^1 \phi^2(x)dx.$$

These can be used to generate an iterative computation for ϕ and ψ. However, questions of convergence, proper initiating functions to achieve the largest eigenvalue, etc. seem to be difficult. One final remark—the minimum value of $\int_0^1 \int_0^1 [z(x, y) - \sum_{j=1}^{n} \phi_j(x)\psi_j(y)]^2 dxdy$ is precisely the sum of the remaining eigenvalues of K, i.e. the sum of the eigenvalues minus the sum of the largest n eigenvalues, the former sum being equal to $\int\int z^2 dxdy$.

OLIVER GROSS

The RAND Corporation
Santa Monica, California

[1] S. P. DILIBERTO & E. G. STRAUS, "On the approximation of a function of several variables by the sum of functions of fewer variables," *Pacific Jn. of Math.*, v. 1, p. 195–210, 1951.

A Logarithm Algorithm

The method of calculating logarithms given in this paper is quite unlike anything previously known to the author and seems worth recording because of its mathematical beauty and its adaptability to high-speed computing machines. Although there are well known methods[1] which involve continued fractions, these methods invariably utilize the *analytic* properties of the logarithm *function* and not the *arithmetic* properties of the *individual* logarithm. The first version of this algorithm is based directly upon such arithmetic continued fractions. In a subsequent skeletonized modification, however, continued fractions no longer appear explicitly.

Let $a_0 > a_1 > 1$ be given. To find $\log_{a_0} a_1$ we determine the two sequences

$$a_2, a_3, \cdots$$
$$n_1, n_2, \cdots,$$

where the n's are positive integers, by the relations

$$a_i^{n_i} < a_{i-1} < a_i^{n_i+1}$$
$$a_{i+1} = a_{i-1}/a_i^{n_i}. \tag{1}$$

We define the complete quotient $x_1 > 1$ by

$$a_0 = a_1^{n_1+1/x_1}.$$

Since

$$a_1 = a_2^{x_1}$$

we write

$$x_1 = n_2 + 1/x_2$$

and so on. Thus we have the continued fraction

$$\log_{a_0} a_1 = \frac{1}{n_1} + \frac{1}{n_2} + \frac{1}{n_3} + \cdots.$$

For example let $a_0 = 10$, $a_1 = 2$. Then

$$2^3 < 10 < 2^4.$$

Therefore $n_1 = 3$ and $a_2 = 10/2^3 = 1.25$. Further:

$$1.25^3 < 2 < 1.25^4.$$

Thus $n_2 = 3$ and $a_3 = 2/1.25^3 = 1.024$. Continuing we obtain

i	n_i	a_i
0	—	10
1	3	2
2	3	1.25
3	9	1.024
4	2	1.009741958
5	2	1.004336279
6	4	1.001041545

It follows that

$$\log_{10}2 = \frac{1}{3} + \frac{1}{3} + \frac{1}{9} + \frac{1}{2} + \frac{1}{2} + \frac{1}{4} + \cdots.$$

Breaking off the continued fraction at successive terms we find:

i	rational approximation to $\log_{10}2$
1	1/3 = .33333333
2	3/10 = .30000000
3	28/93 = .30107527
4	59/196 = .30102041
5	146/485 = .30103093
6	643/2136 = .30102996.

Since $\log_{10}2 = .30102999566$ we see that each successive approximation gives us approximately one more correct decimal place than the previous one. This rate of convergence, one decimal place per cycle, is typical. If one chooses at random an a_0 and an a_1, one will "almost always" find such a rate of convergence. A method of estimating this universal rate is given below.

Both phases of the recurrence rules given in (1) may be carried out simultaneously. One divides a_{i-1} by a_i; then the quotient by a_i; then that quotient

by a_i, etc., until the resulting quotient is less than a_i. Then this last quotient is a_{i+1} and the number of divisions is n_i. In fact, this part of the process involves *only division.*

The second part of the process, namely, that of obtaining the successive fractions, may similarly be reduced to *only addition*, but at first we note that from the theory of continued fractions if P_{i-1}/Q_{i-1} and P_i/Q_i are the rational approximations which include the terms up to $1/n_{i-1}$ and $1/n_i$ respectively; then

$$\frac{P_{i+1}}{Q_{i+1}} = \frac{P_{i-1} + n_{i+1}P_i}{Q_{i-1} + n_{i+1}Q_i}. \tag{2}$$

For example, above we had

$$P_2/Q_2 = 3/10,\ P_3/Q_3 = 28/93,\ n_4 = 2;$$

and therefore

$$P_4/Q_4 = 59/196.$$

However, for an automatic computing machine we recommend the following variation which abstracts the contents of the last two paragraphs. We maintain six registers A, B, C, D, E, and F. At each inning we do one of two things:

Operation I (if $A \geq B$):
We put A/B in A, $C + E$ in C, and $D + F$ in D.
Operation II (if $A < B$):
We interchange A and B, C and E, D and F. We start with a_0 in A, a_1 in B, 1 in C and F, and 0 in D and E. The latest approximation to $\log_{a_0} a_1$ is always E/F.

In the example above we would obtain:

	A	B	C	D	E	F
	10	2	1	0	0	1
Op. I	5	2	1	1	0	1
	2.5	2	1	2	0	1
	1.25	2	1	3	0	1
Op. II →						
	2	1.25	0	1	1	3
Op. I	1.6	1.25	1	4	1	3
	1.28	1.25	2	7	1	3
	1.024	1.25	3	10	1	3
Op. II →						
	1.25	1.024	1	3	3	10

It is readily seen that in this variation we need *not* assume $a_0 > a_1$, as was done in (1), but merely that $a_0 \geq 1$ and $a_1 \geq 1$. Further, if one or both of these numbers are less than 1, then since

$$\log_a x^{-1} = -\log_a x, \quad \log_{1/a} x = -\log_a x, \quad \log_{1/a} x^{-1} = \log_a x,$$

we may proceed as before after taking the reciprocals of those numbers less than 1. We then multiply the fraction obtained by $(-1)^m$ where m is the number of reciprocals taken.

If it happens that the logarithm is a rational number, for instance $\log_8 4 = 2/3$, then at some point B becomes 1, the exact log is obtained and no further changes in E or F occurs. For example:

A	B	C	D	E	F
8	4	1	0	0	1
2	4	1	1	0	1
4	2	0	1	1	1
2	2	1	2	1	1
1	2	2	3	1	1
2	1	1	1	2	3
2	1	3	4	2	3
2	1	5	7	2	3

If only four registers are available then one may (with some care) economize by keeping C and D side by side in a register $\bar{C}$ and similarly E and F in a register $\bar{E}$.

Operation I now reads:

We put A/B in A and $\bar{C} + \bar{E}$ in $\bar{C}$.

Operation II now reads:

We interchange A and B, $\bar{C}$ and $\bar{E}$.

The example now reads:

	A	B	C	E
	10	2	.00010000	.00000001
	5	2	.00010001	.00000001
	2.5	2	.00010002	.00000001
	.....			
	1.0010415	1.0043363	.06432136	.01460485
Op. II→				
	1.0043363	1.0010415	.01460485	.06432136

We now split .06432136 into two parts and divide: $\log_{10} 2 \approx 0643/2136 = .30102996$. Of course the "$D$" and "$F$" parts of the numbers must not be allowed to overlap the first halves.

The simplicity of the rules given in either pair of *Operations I, II* is a recommendation for use of this logarithm in automatic computing machines. So also is the fact mentioned above that for "almost all" a_0 and a_1, the rate of convergence is essentially independent of these numbers and is, on the average, about one decimal place per complete cycle (that is, for each *Operation II*). We justify this claim as follows. KHINTCHINE[2] has shown that for "almost all" real numbers, $0 < x < 1$, the regular continued fraction has the property that the geometric mean of the n's is an absolute constant. This number, known as Khintchine's constant, has been computed by the author to ten figures[3]. Its numerical value is approximately

$$K = 2.685452001.$$

Now we saw in (2) that the n's regulate the rate of growth of the Q's. To estimate this growth we assume that the Q's are governed (in the mean) by the linear difference equation:

$$Q_{i+1} = Q_{i-1} + KQ_i.$$

From the theory of such equations, we find that for large i:

$$Q_i \doteq a\{\tfrac{1}{2}(K + [4 + K^2]^{\frac{1}{2}})\}^i$$

Now it is easily seen that the error in any approximation P_i/Q_i is of the order of $(Q_i)^{-2}$ and therefore, per average cycle, the error should decrease by a factor of $4\{K + (4 + K^2)^{\frac{1}{2}}\}^{-2} \doteq 1/9.1$ or approximately one decimal place.

DANIEL SHANKS

Naval Ordnance Laboratory
White Oak, Maryland

[1] See for example, D. TEICHROEW, "Use of continued fractions in high speed computing," *MTAC*, v. 6, 1952, p. 127.
[2] A. KHINTCHINE, "Zur metrische Kettenbruchtheorie," *Compositio Math.*, v. 3, 1936. p. 276–285.
[3] DANIEL SHANKS, "Note on an absolute constant of Khintchine," *MTAC*, v. 4, 1950, p. 28.

The Product Form for the Inverse in the Simplex Method

Summary: When a matrix is represented as a product of "elementary" matrices, the matrix, its transpose, its inverse and inverse transpose are readily available for vector multiplication. By an "elementary matrix" is meant one formed from the identity matrix by replacing one column; thus an elementary matrix can be compactly recorded by the subscript of the altered column and the values of the elements in it. In the revised simplex method,[1] both the inverse and inverse transpose of a "basic" matrix are needed; more significant, however, is the fact that each iteration replaces one of the columns of the basis. In the product form of representation, this change can be conveniently effected by multiplying the previous matrix by an elementary matrix; thus, only one additional column of information need be recorded with each iteration. This approach places relatively greater emphasis on "reading" operations than "writing" and thereby reduces computation time. Using the I.B.M. Card Programmed Calculator, a novel feature results: when the inverse matrix is needed at one stage and its transpose at another, this is achieved simply by turning over the deck of cards representing the inverse.

Introduction:

The simplex method is an algorithm for determining values for a set of n non-negative variables which minimizes a linear form subject to m linear restraints.[1,2a,3] It may be characterized briefly as a finite iterative procedure. Each iteration produces a new special solution to the restraint equations involving a subset of m of the variables, only one element of the subset changing on successive iterations; the remaining $n - m$ variables are equated to zero. The vectors of coefficients corresponding to the subset of m variables are linearly independent and constitute a *basis* in m-dimension real vector space. In the original simplex method[2a] (as coded for the SEAC[4] or as found