# HIGH BANDWIDTH EVALUATION OF ELEMENTARY FUNCTIONS*

P. Michael Farmwald†

S-1 Project

Lawrence Livermore National Laboratory

## Abstract

Among the requirements currently being imposed on high-performance digital computers to an increasing extent are the high-bandwidth computations of elementary functions, which are relatively time-consuming procedures when conducted in software. In this paper, we elaborate on a technique for computing piecewise quadratric approximations to many elementary functions. This method permits the effective use of large RAMs or ROMs and parallel multipliers for rapidly generating single-precision floating-point function values (e.g., 30-45 bits of fraction, with current RAM and ROM technology). The technique, based on the use of Taylor series, may be readily pipelined. Its use for calculating values for floating-point reciprocal, square root, sine, cosine, arctangent, logarithm, exponential and error functions is discussed.

## 1. Introduction

Consider the well known [1] technique for evaluating $1/z$ by generating an initial approximation, $a_0$, for $1/z$ using a table-lookup on the high bits of $z$ and then using the iteration $a_{i+1} = a_i(2 - a_i z)$ until the desired accuracy is reached. The combination of the table-lookup and the first iteration is the same as expanding the Taylor series for $1/z = 1/(x + y)$ about the initial point of approximation, $x$, i.e.

$$a_0(2 - a_0 z) = \frac{1}{x} - \frac{y}{x^2},$$

where $a_0 = 1/x$ is the initial approximation. Furthermore, by one multiplication ($y(1/x^2)$) we have doubled the precision of our approximation. We shall show that using further terms of the Taylor series leads to a fast method for evaluating many functions, whereby in one multiplication time (but with two parallel multiplications) we can triple the precision of the table-lookup approximation.

*Work performed under the auspices of the U.S. Department of Energy by Lawrence Livermore National Laboratory under contract number W-7405-ENG-48, with support from the Naval Electronic Systems Command, U. S. Navy.

†Fannie and John Hertz Foundation Fellow, Computer Science Dept., Stanford University

## 2. Mathematics of the approximation technique

We wish to compute a suitable function, $f(z)$, to $p$-bits of accuracy. More precisely, over some limited range of the argument $z$, we would like to compute an approximation $\hat{f}(z)$, such that $|f(z) - \hat{f}(z)| \leq 1/2^{p+1}$ for all $z$ in the restricted range.

For simplicity, assume that $0 \leq z < 1$ is the range in which we are interested. Thus $z$ is a fixed-point fraction with $p_z$ bits. We will break $z$ into two pieces, $x$ and $y$, where $x = \lfloor z2^{p_z} \rfloor / 2^{p_z}$ and $y = (z - x)2^{p_z}$, so that $z = x + y/2^{p_z}$. It should be clear that $0 \leq x < 1$ and $0 \leq y < 1$ and that $x$ and $y$ are fixed point fractions with $p_x$ and $p_z - p_x$ bits, resp. We are going to make a number of approximations to $x$ and $y$ which use various numbers of the leading bits of each. We will use the notation $p_{x_i}$ and $p_{y_i}$ for the number of bits of $x$ or $y$ used in the $i^{th}$ term of the approximation.

We have (using the Taylor expansion for $f$ at $x$)

$$\begin{aligned} f(z) = f(x + \frac{y}{2^{p_z}}) \\ = \quad & f(x) && (1a) \\ & + \frac{y}{2^{p_z}} f'(x) && (1b) \\ & + (\frac{y}{2^{p_z}})^2 f''(x)/2! && (1c) \\ & + (\frac{y}{2^{p_z}})^3 f'''(x)/3! && (1d) \\ & + (\frac{y}{2^{p_z}})^4 f''''(x + \beta)/4!, && (1e) \end{aligned}$$

where $0 \leq \beta \leq y/2^{p_z}$.

We will analyze the approximation to $f$ of (see Figure 1)

$$\begin{aligned} \hat{f}(z) = {} & \text{round}(f(x), p + 6) && (2a) \\ & + \frac{y}{2^{p_z}} \text{round}\left(f'(x), p + 6 - p_z\right) && (2b) \\ & + \text{round}\left(\left(\frac{\text{truncr}(y, p_{y_2})}{2^{p_z}}\right)^2, p + 6\right) \\ & \quad \times \text{round}\left(\frac{f''(x)}{2!}, p + 7 - 2p_z\right) && (2c) \\ & + \text{round}\left(R\left(\text{truncr}(x, p_{x_3}), \frac{\text{truncr}(y, p_{y_3})}{2^{p_z}}\right)\right. \\ & \quad \left., p + 7 - 3p_z\right) && (2d) \end{aligned}$$

where

$$\text{round}(z,q) = \lfloor z2^q + 1/2 \rfloor / 2^q,$$

(i.e., normal rounding to $p$-bits)

$$\text{truncr}(z,q) = (\lfloor z2^q \rfloor + 1/2)/2^q$$

(also known as von Neumann rounding or jamming) and

$$R(u,v) = (v/2^{p_x})^3 f'''(u)/3! + (v/2^{p_x})^4 f''''(u)/4!.$$

Of these four terms constituting the desired approximation, notice that (2a) and (2d) consist of table-lookups, (2b) of a table-lookup followed by a multiplication and (2c) of two table-lookups followed by a multiplication. Notice also that the table-lookups may be performed in parallel, as may the two multiplications.

Let us assume that the general term of the Taylor series of $f$ satisfies

$$\left| f^{(n)}(z) \right| \leq n!$$

for $0 \leq z < 1$ and $n \geq 0$. In addition , let us choose $p_x$ such that it is technologically feasible to perform fast table lookups on $x$ (e.g., about 9-16 bits using 1981 technology). The values of the terms $f(x)$, $f'(x)$ and $f''(x)$ thus can be obtained directly from ROM or RAM. (This implies that $p_{x_0} = p_{x_1} = p_{x_2} = p_x$.) Also note that $p_{y_1} = p_y = p_z - p_x$, since we use the full precision of $y$ in term (2b). This is not strictly necessary, since we could approximate $y$ with enough of its leading bits (at least $p + 7 - p_x$ to make the error of truncation small).

The term $(y/2^{p_x})^2$ can be obtained by table lookup on the high $p_{y_2}$-bits of $y$ if $2p_x + p_{y_2} \geq p + 2$. To see this, note that the approximation used in the table lookup is

$$y \approx \frac{\lfloor y2^{p_{y_2}} \rfloor + \frac{1}{2}}{2^{p_{y_2}}} = y + \frac{\alpha}{2^{p_{y_2}}},$$

where $-1/2 < \alpha \leq 1/2$. The error involved in using this to approximate $(y/2^{p_x})^2$ is

$$\left| \left(\frac{y}{2^{p_x}}\right)^2 - \left(\frac{y + \frac{\alpha}{2^{p_{y_2}}}}{2^{p_x}}\right)^2 \right| = \frac{1}{2^{2p_x}} \left| \frac{2\alpha y}{2^{p_{y_2}}} - \frac{\alpha^2}{2^{2p_{y_2}}} \right|$$

$$\leq \frac{1}{2^{2p_x + p_{y_2}}} + \frac{1}{2^{2p_x + 2p_{y_2} + 2}}.$$

Term (2d) is found by table lookup on the high bits of $x$ and $y$. We will use the high $p_{x_3}$ and $p_{y_3}$ bits of x and y. Using the same approach as before, we approximate $x \approx x + \tau/2^{p_{x_3}}$ and $y \approx y + \lambda/2^{p_{y_3}}$, where $-1/2 < \tau \leq 1/2$ and $-1/2 < \lambda \leq 1/2$. Then the error in this approximation to term (2d) will be (leaving out intermediate steps)

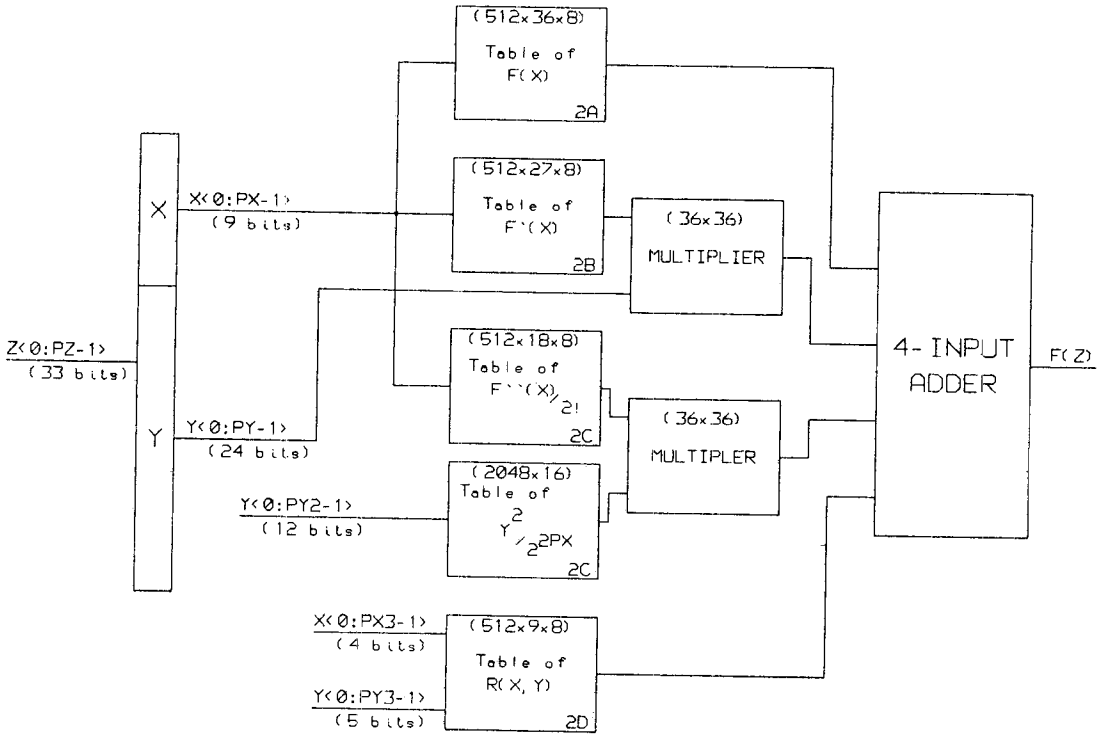


Figure 1

$$\left| R(x,y) - \left(\frac{y + \frac{\lambda}{2^{p_{y_3}}}}{2^{p_x}}\right)^3 \frac{f'''(x + \frac{r}{2^{p_{x_3}}})}{3!} - \left(\frac{y + \frac{\lambda}{2^{p_{y_3}}}}{2^{p_x}}\right)^4 \frac{f''''(x + \frac{r}{2^{p_{x_3}}})}{4!} \right|$$
$$\leq \frac{2}{2^{3p_x + p_{x_3}}} + \frac{3/2}{2^{3p_x + p_{y_3}}} + O\left(\frac{1}{2^{4p_x + \min(p_{x_3}, p_{y_3})}}\right) + O\left(\frac{1}{2^{3p_x + 2\min(p_{x_3}, p_{y_3})}}\right).$$

We now have all of the pieces to find the total error in our approximation to (1). The five rounding errors sum to at most $1/2^{p+5}$. Thus the error in (2c) can be made close to $1/2^{p+2}$ by making $2p_x + p_{y_2} \geq p + 2$. We can make the error in (2d) be less than $7/2^{p+5}$ if $3p_x + p_{x_3} \geq p + 4$ and $3p_x + p_{y_3} \geq p + 4$. Furthermore we ignore a number of smaller error terms because they make little or no contribution to the error. Summing these errors gives a total maximum error less than $1/2^{p+1}$, as desired. Notice that this is the maximum error — the average error will tend to be much less due to cancellation.

Summarizing the foregoing assumptions, we have

$$\left|f^{(n)}(z)\right| \leq n!, \tag{3a}$$
$$2p_x + p_{y_2} \geq p + 2, \tag{3b}$$
$$3p_x + p_{y_3} \geq p + 4, \tag{3c}$$
$$3p_x + p_{x_3} \geq p + 4, \tag{3d}$$

together with technology constraints on the choice of $p_x$, $p_{y_2}$, $p_{x_3}$ and $p_{y_3}$.

We shall now select $p_x$, $p_{y_2}$, $p_{x_3}$ and $p_{y_3}$ to minimize the total number of bits in the tables. There are five tables used, and the sum of their sizes (in order of their appearance in (2)) is

$$2^{p_x}(p+6) + 2^{p_x}(p+6-p_x) + 2^{p_{y_2}}(p+6-2p_x) + 2^{p_x}(p+7-2p_x) + 2^{p_{x_3}+p_{y_3}}(p+7-3p_x).$$

Clearly, we can let $p_{y_2} = p + 2 - 2p_x$ and $p_{x_3} = p_{y_3} = p + 5 - 3p_x$, since these are the minimum values which satisfy the constraints. Thus we must minimize

$$2^{p_x}(3p + 19 - 3p_x) + 2^{p+2-2p_x}(p+6-2p_x) + 2^{2p+10-6p_x}(p+7-2p_x)$$

as a function of $p_x$. Without proof, we state the result that the minimum occurs at $p_x \approx p/3$, which satisfies the constraints if $p_x \geq 3$. The table involving only bits of $y$ need not be duplicated when using this technique for evaluating multiple functions; this drives the optimum value of $p_x$ even lower, since the $y^2$ table is common to all of the functions. The total table size (in bits) at $p_x = p/3$ is approximately

$$2^{\frac{p}{3}}(3p + 45) + 7000$$

or if $k$ functions are implemented:

$$2^{\frac{p}{3}}((2k+1)p + 20k + 25) + 7000k.$$

## 3. Some functions which satisfy the constraints

Table 1 lists some common functions which satisfy the necessary conditions developed above. The examples assume a radix-2 floating-point format. Since the dynamic range of the result must be limited to maintain relative accuracy (the dynamic range is limited essentially to 2), we are forced in some cases to compute another function which can easily be transformed to the correct result. In fact, the correct way to regard this method is to consider it a technique for computing a fixed-point function of a fixed-point number. To maintain relative precision in floating-point may involve additional work. This is why we compute $\sin(z)/z$ instead of $\sin(z)$ (since sin has a zero at zero). Also notice that $1 - \mathrm{erf}(z) \leq \pi e^{-z^2}/(2z)$ for $z \geq 1$. Thus for $z > 8$ we have $\mathrm{erf}(z) = 1$ to within $2^{-92}$ or $10^{-28}$. Also the derivatives of $\mathrm{erf}(z)$ are so small when $z > 1$ that much coarser tables can be used than for the range $0 \leq z < 1$. The tables used in computing $2/z$ and $\sqrt{z}$ are double-sized, since in computing $2/z$ we need to look at the sign of $z$ and in computing $\sqrt{z}$ we must examine the low exponent bit (i.e., we use different approximations, depending upon whether the exponent is odd or even).

## 4. Practical implementation

The technique just described has been implemented in the S-1 Mark IIA processor [4] to evaluate elementary functions in single-precision floating-point (which has a sign bit, 9 exponent bits, a hidden fraction bit, and 26 bits of fraction). The numbers in parenthesis (in Figure 1) are the actual sizes of the tables used. The Taylor series is actually evaluated to 29 bits of accuracy and then rounded to 27 bits. This implies that the functions satisfy an equation of the form $\hat{f}(x) = f(x)(1+\epsilon)$ where $\epsilon \leq 0.62/2^{27}$. All of the approximations used satisfy the correct monotonicity properties (partly as a consequence of evaluating it to extra precision before rounding). In addition, some of the functions satisfy necessary special properties, such as $|\sin| \leq 1$. Evaluation to extra precision is also valuable for computing double-precision reciprocation and square-root. The S-1 double-precision floating-point format has 57 fraction bits. Since we have an approximation which has more than one-half of the desired precision, one Newton iteration will suffice to finish the approximation.

All of the table lookups are done in the first stage of the multiplier functional unit, with the normal operation of the succeeding stages of the multipier being delayed by 25 nsec (to make time for the table lookups). The multiplier has four 18 x 36 multipliers which can be reconfigured as two 36 x 36 multipliers. The subsequent pipe stages (accumulate, normalize, round and normalize) are shared by the floating-point multiply and elementary function evaluation functional units. The multiplier has a total latency (time from input to output) of either 125 or 150 nsec, depending on how it is being used. This arrangement readily supports the pipelined evaluation of $1/z$, $\sqrt{z}$, $\lg z$, and $2^z$ at 25 nsec per datum and $y/z$, $\ln z$,

$\log z$, $e^z$, $\arctan z$, $\sin z$, $\cos z$ at 50 nsec per datum.

## 5. Rounding

There is a minor problem with this form of function evalution, in that it doesn't round "perfectly", i.e. it doesn't allow rounding of the nearest representable floating point number to the exact result. (Here "perfect" rounding means that the error in the result is less than or equal to one-half the least significant bit.) Of the functions discussed above, reciprocation and division (and occasionally square root) are the only ones for which serious attempts are ever made to do "perfect" rounding. The error due to the approximate method for elementary function evaluation presented here can be considered as additional rounding error, and can be made as small as desired by making $p$ as large as necessary before rounding. This type of error has been well studied by numerical analysts, and is therefore quite acceptable where function evaluation speed is important. There is no a priori limit known to the precision to which one might have to evaluate sine, cosine, logarithm, exponential and arctangent in order to round correctly [6]. Of course, for a fixed size word, one could determine the precision needed to round each value ahead of time and so place a bound on the precision necessary to round any value. This is clearly not worth the small increment in precision that it actually gives. For division, reciprocation, and square root, it is interesting to note that in order to round "perfectly" to $p$ bits it is sufficient to evaluate the result to $2p$ bits before rounding.

## 6. Summary

We have shown that it is possible to compute many common functions to $p$ bits of accuracy, using a technique comprised of table look-ups and two parallel fixed-point multiplications with precisions $2p/3$ bits and $p/3$ bits. The size of the required tables is approximately $2(k+1)p2^{\frac{p}{3}}$ bits, where $k$ is the number of functions to be so approximated. With the fastest 1981 RAM or ROM technology, $p$ can feasibly range from 30 to about 45. This structure can be readily pipelined (as easily as can a multiplication), and it is very similar to that of a high-precision multiplier unit implemented from a pair of lower-precision mulipliers.

## References


[1] M. J. Flynn, "On Division by Functional Iteration", *IEEE Trans. Computers*, vol. C-19, pp. 702-706, August 1970.

[2] P. Rabinowitz, "Multiple Precision Division", *Commun. ACM*, vol. 4, p. 98, February 1961.

[3] Z. Shaham and Z. R. Riesel, "A Note on Division Algorithms Based on Multiplication", *IEEE Trans. Computers*, vol. C-21, pp. 513-514, May 1972.

[4] Staff, *S-1 Project FY1979 Annual Report*, University of California Lawrence Livermore National Lab., UCID 18619 (1979).

[5] C. T. Fike, *Computer Evaluation of Mathematical Functions*, Prentice-Hall, 1968.

[6] W. Kahan, private communication, March 1981


| Base function | Domain | Example of use |
|---|---|---|
| $\sin(\frac{\pi z}{2})/z$ | $0 \le z < 1$ | $\sin(z) = \begin{cases} if(i), & (0 \le j < 1) \\ (1-i)f(1-i), & (1 \le j < 2) \\ -if(i), & (2 \le j < 3) \\ -(1-i)f(1-i), & (3 \le j < 4) \end{cases}$ where $i = \frac{2z}{\pi} \bmod 1$ and $j = \frac{2z}{\pi} \bmod 4$ |
| $\lg(z)(= \log_2(z))$ | $1 \le z < 2$ | $\lg(2^e m) = e + f(m)$ |
| $2/z$ | $1 \le \lvert z \rvert < 2$ | $1/2^e m = 2^{-e-1} f(m)$ |
| $\sqrt{z}$ | $1 \le z < 4$ | $\sqrt{2^{2e_1+e_2} m} = 2^{e_1} f(2^{e_2} m)$ |
| $2^z$ | $0 \le z < 1$ | $2^z = 2^{\lfloor z \rfloor} f(z - \lfloor z \rfloor)$ |
| $\arctan(z)/z$ | $0 \le z < 1$ | $\arctan z = \begin{cases} f(\frac{1}{z})/z - \frac{\pi}{2}, & (z < -1) \\ -zf(z), & (-1 \le z < 0) \\ zf(z), & (0 \le z < 1) \\ \frac{\pi}{2} - f(\frac{1}{z})/z, & (1 \le z) \end{cases}$ |
| $\operatorname{erf}(z)/z$ | $0 \le z < 8$ | $\operatorname{erf} z = \begin{cases} -1, & (z < -8) \\ -zf(z), & (-8 \le z < 0) \\ zf(z), & (0 \le z < 8) \\ 1, & (8 \le z) \end{cases}$ |

Table 1