# A REALISTIC MODEL FOR ERROR ESTIMATES IN THE EVALUATION OF ELEMENTARY FUNCTIONS

Krzysztof S. Frankowski

Computer Science Department
University of Minnesota
Minneapolis, Minnesota

## Abstract

Floating point error analysis, as described by J. H. Wilkinson (1963) has two known drawbacks: it is too pessimistic and too cumbersome for everyday use. This paper describes a realistic model for error analysis, gives examples of simple formulae frequently used in the calculation of elementary functions, and analyses the error generated in single precision computations with these formulae, using the proposed model for error analysis. The paper also presents error bounds for various polynomial evaluations, as predicted by the model. Model verification is done using double precision arithmetic.

## Index Terms

Floating point error; computer arithmetic; evaluation of elementary functions; polynomial evaluation.

## Introduction

Wilkinson's (W63) publications solved many of the mysteries of floating point arithmetic. Error studies are found even in the most elementary publications, for example D. McCracken and W. Dorn (McC65). Unfortunately, the usual analysis suffers from two serious drawbacks:

1. Error bounds computed in the usual way are unrealistically pessimistic, and
2. The algebra required is so involved that it discourages many casual users from performing error analysis on a day to day basis. In fact, D. McCracken uses tree structures (called process graphs) to analyze errors, and D. Stoutemyer (S77) even proposes the use of an algebraic manipulator for this purpose.

The issue is further clouded by a lack of agreement about how to measure error. Some authors use ulps (units in last place), some logarithmic error and others relative error. In some library descriptions, even absolute error is used. Without getting into a discussion of which measure is best, we will use only the relative error measure. It is the author's claim that a few simple assumptions enable a user to perform error analysis simply and get realistic error estimates in many practical cases.

## The Basic Model

Assume that $x$ is a real number and $\bar{x}$ is its floating point representation. Then $E(x) = x - \bar{x}$ and $R(x) = \frac{E(x)}{x}$, for $x \neq 0$. When $E(x)$ is small ($E^2(x)=0$) we have:

$$R(x \pm y) = \frac{xR(x)}{x \pm y} \pm \frac{yR(y)}{x \pm y} + \varepsilon a \qquad (1)$$

$$R(x*y) = R(x) + R(y) + \varepsilon a \qquad (2)$$

$$R(x/y) = R(x) - R(y) + \varepsilon a \qquad (3)$$

where $\varepsilon = \pm 1$ only, and $a$ is the error due to performing arithmetic operations in a computer. Formulae (1-3) simply state that the total relative error in the four basic operations is composed of two parts: 1. Relative error in operands, and 2. Error due to chopping, rounding or whatever the arithmetic unit of a particular computer does while storing the result of an operation in a bounded memory cell or register. We will call that error arithmetic error. Note that $\varepsilon$ can be either $\pm 1$, and we choose the sign in such a way that the $a$'s do not cancel. During the actual computations in any numeric algorithm, each atomic operation generates an error, and some of these errors cancel each other. Since we wish to obtain an upper bound on the error of the numeric algorithm, we accumulate the errors of all the atomic operations, ignoring this cancellation. Thus, our bound is skewed toward the true upper bound. We can neglect the sign of $\varepsilon$ if we remember the rule $\alpha a + \beta a = (|\alpha|+|\beta|)\, a$ for any $\alpha$, $\beta$ real.

In addition to formulae (1-3) we shall assume the following properties of finite arithmetic:

$$R(i) = 0 \text{ for } -M \leq i \leq M,\ i \in \text{Integer} \qquad (4)$$

which states that sufficiently small integers are represented exactly, and

$$R(i \pm j) = 0 \text{ and } R(i*j) = 0 \qquad (5)$$

which states that if the absolute value of the results of an operation is less than or equal to M, the result is represented exactly. Propagation of error is well described in rules (1-5). Because the $\varepsilon a$ of rules (1-3) each represent the same quantity, we can obtain simpler formulae for the error of any numerical algorithm than those obtained when $\varepsilon a_i$ represents a different quantity in each of the rules (1-3).

The following examples demonstrate the use of the model to select the best formula for a given computation, and to select a range of values on

CH1412-6/78/0000-0070$00.75 (c) 1978 IEEE

which to apply a given formula.

Example One

In computing the volume of a sphere, we have to calculate $y=\left(\frac{\sqrt{5}-2}{\sqrt{5}+2}\right)^3$. Using simple rules of algebra we obtain:

$$y_1=\left(\frac{\sqrt{5}-2}{\sqrt{5}+2}\right)^3 \quad y_2=(\sqrt{5}-2)^6 \quad y_3=(9-4\sqrt{5})^3 \quad y_4=(2889-1292\sqrt{5})$$

$$y_5=\left(\frac{1}{\sqrt{5}+2}\right)^6 \quad y_6=\left(\frac{1}{9+4\sqrt{5}}\right)^3 \quad y_7=\frac{1}{2889+1292\sqrt{5}}$$

We have seven different formulae for the same number and it is not obvious which one produces the smallest error.
Let us use our model to evaluate $R(y_1)$

$$R(y_1)=3R\left(\frac{\sqrt{5}-2}{\sqrt{5}+2}\right)+2a = 3R(\sqrt{5}-2) - 3R(\sqrt{5}+2) + 5a=$$

$$12\sqrt{5}\,R(\sqrt{5}) + 11a \simeq 27\,R(\sqrt{5}) + 11a$$

Table I gives a summary of the $R(y_i)$ for $i=1,\ldots,7$ and also the numerical results, assuming $a=0$, that is, that calculations are performed exactly, and $\sqrt{5}\simeq 9/4 = 2.25$. From this table it is clear that formula $y_7$ is the best and formula $y_4$ is the beast.

Example Two

We evaluate $y_1=\frac{1-x}{1+x}=y_2=1-\frac{2x}{1+x}$ for small $|x|$. This type of formula is often used in the evaluation of $\log(x)$, $e^x$, etc. Here

$$R(y_1)=\frac{-2xR(x)}{1-x^2}+3a,$$

and

$$R(y_2)=\frac{-2x}{1-x^2}\left(R(x)+2a(1+x)\right)+a \ . \ *$$

It is clear that as $x\to 0$ $R(y_1)\to 3a$, so $y_2$ is more accurate for small $|x|$. We wish to find out for which range of x formula $y_2$ is better. This involves solving the inequality $\left|\frac{-2xR(x)}{1-x^2}+3a\right| > \left|\frac{-2xR(x)}{1-x^2}-\frac{4ax}{1-x}+a\right|$ under different assumptions. The simplest one is that x is accurate. Then $3 > \left|\frac{1-5x}{1-x}\right|$ or $-1<x<1/2$.

*In most binary machines $R(2x)=R(x)$ whether the result is obtained by x+x or 2*x or an addition to the exponenet. (If $2.*x \neq x+x$ the operation which produces the exact answer should be chosen).

Thus our model tells us that for $|x|<1/2$, if x is accurate, then formula $y_2$ is better.

Error In Polynomial

Before we discuss errors in polynomials, let us examine the error in a sum. We

evaluate $S_n=\sum_{i=1}^{n} a_i$ by $\begin{cases} S_0=0 \\ S_i=S_{i-1}+a_i \text{ for } i=1,2,3\ldots n \end{cases}$

or by the equivalent formula $\begin{cases} S_0=a_i \\ S_i=S_{i-1}+a_{i+1} \\ \text{for } i=1,2,\ldots n-1 \end{cases}$

Using our model we obtain:

$$S_iR(S_i)=S_{i-1}R(S_{i-1})+a_iR(a_i)+a\cdot S_i \qquad (6)$$

Solving this difference equation we have

$$S_nR(S_n)=\sum_{i=1}^{n} a_iR(a_i)+a\sum_{i=1}^{n}|a_i|(n-i+1) - a\cdot|a_1| \text{ or}$$

$$S_nR(S_n)=\sum_{i=1}^{n} a_iR(a_i)+a[nT_n-\sum_{i=1}^{n}(i-1)|a_i|-|a_1|] \qquad (7)$$

where $T_n=\sum_{i=1}^{n}|a_i|$

Let $P_{n-1}=\sum_{i=1}^{n} b_ix^{i-1}=b_1+b_2x+\ldots+b_nx^{n-1}$

which we evaluate by:

$$P_0=b_1$$

$$t_0=1 \qquad (8)$$

$$\begin{cases} t_i=t_{i-1}\cdot x \\ P_i=P_{i-1}+t_i\cdot b_{i+1} \text{ for } i=1,2,3\ldots n-1 \end{cases}$$

From the model we have

$$P_iR(P_i)=P_{i-1}R(P_{i-1})+(t_i\cdot b_{i+1})R(t_ib_{i+1})+aP_i$$

This equation is equivalent to (6) with $a_i=t_{i-1}\cdot b_i$ so

$$P_{n-1}R(P_{n-1})=\sum_{i=1}^{n} t_{i-1}b_iR(t_{i-1}b_i)+a[nQ_{n-1}$$

$$-\sum_{i=1}^{n}(i-1)|t_{i-1}b_i|-|b_1|], \text{ where } Q_{n-1}^{(x)}=\sum_{i=1}^{n}|b_i||x|^{i-1}$$

But since $t_i=x^i$, $R(t_i)=i\,R(x)+(i-1)a$ (for $i=1,2\ldots n-1$), and

$$R(t_{i-1}b_i)=(i-1)(R(x)+a)+R(b_i)$$

we obtain

$$P_{n-1}R(P_{n-1}) = \sum_{i=1}^{n} x^{i-1} b_i[i-1)(R(x)+ R(b_i)]$$

$$+a[n\, Q_{n-1}-|b_1|-\sum_{i=1}^{n}(i-1)|x|^{i-1}|b_i|]$$

We notice that since the derivative $P'_{n-1}(x) = \sum_{i=1}^{n} b_i(i-1)x^{i-2}$, the term

$$\sum_{i=1}^{n}(i-1)x^{i-1}b_i = xP'_{n-1}(x)$$

This gives us the error formula in the form:

$$P_{n-1}R(P_{n-1}) = \sum_{i=1}^{n} x^{i-1}b_iR(b_i)+xR(x)P'_{n-1}(x) + a(nQ_{n-1}-|b_1|) \quad (9)$$

In many applications e.g., while evaluating elementary functions the coefficients $b_i$ are computed, rounded and stored in memory, then $R(b_i) \approx \varepsilon a$; and (9) becomes;

$$R(P_{n-1}) = \frac{XR(x)P'_{n-1}(x)}{P_{n-1}(x)} + a[(n+1)\frac{Q_{n-1}(x)}{P_{n-1}(x)} - \frac{|b_1|}{P_{n-1}(x)}] \quad (10)$$

Example Three

We evaluate $e^x = \sum_{i=1}^{n} \frac{x^{i-1}}{(i-1)!}$ according to equation (8) storing $b_i = 1/(i-1)!$ perturbed by a; Assuming that we take enought terms $p'_{n-1}(x) \approx e^x$ then

$$R(e^x) = xR(x) + a[(n+1)\, e^{|x|-x} - e^{-x}] \quad (11)$$

For $x>0$ the dominant term is $(n+1)$ a but if $x<0$ and $|x|$ large, catastrophic cancellation occurs.

e.g. $R(e^{-10}) \approx 10R(x) + a(n+1)e^{20} + a\cdot e^{10}$ where $e^{20} \approx 4.85\cdot E8$.

Table II gives computed and predicted errors for $|x| < 2$ in steps of .4 with $R(x)=0$.

Let us turn our attention to evaluation of polynomials by Horner's rule.

Let $p_{n-1} = \sum_{i=1}^{n} b_i x^{i-1} = b_1 + b_2 x + \ldots b_n x^{n-1}$

which we evaluate by

$$p_0 = b_n \quad (12)$$

$p_i = p_{i-1}x + b_{n-1}$ for $i=1,2,3\ldots n-1$

Using the model equations (1) to (5) we obtain

$$p_iR(p_i) = xp_{i-1}R(p_{i-1}) + (R(x)+a)xp_{i-1} + b_{n-1}R(b_{n-1}) + ap_i \quad (13)$$

Solution of (13), though elementary, involves more algebra. After some manipulations we obtain

$$P_{n-1}R(p_{n-1}) = \sum_{i=1}^{n} x^{i-1}b_iR(b_i) + xR(x)P'_{n-1}(x) + a(Q_{n-1}+2|x|\, Q'_{n-1}+|x|^{n-1}|b_n|) \quad (14)$$

$$P_{n-1}R(P_{n-1}) = \sum_{i=1}^{n} x^{i-1}b_iR(b_i) + xR(x)P'_{n-1}(x) + a(nQ_{n-1}-|b_1|) \quad (9)$$

Comparing the two solutions, we see that the first two terms are identical but $nQ_{n-1}$ changed to $Q_{n-1}$ and $|b_1|$ changed to $|x|^{n-1}|b_n|$; both of these changes are beneficial for evaluation of elementary functions where the coefficients $b_i$ are stored and decrease rapidly and $|x|$ is small.[1] Again, assume that $R(b_i)=a$. Then

$$R(P_{n-1})_H = \frac{xR(x)P'_{n-1}(x)}{P_{n-1}(x)} + a[\frac{2\cdot Q_{n-1}(x)}{P_{n-1}(x)} + 2\frac{|x|Q'_{n-1}(x)}{P_{n-1}(x)} + \frac{|x|^{n-1}\cdot|b_n|}{P_{n-1}(x)}] \quad (15)$$

Applying (15) to our previous example we have

$$R(e^x)_H = xR(x) + a(2+2|x|)\, e^{|x|-x}$$

The results of predicted errors are given in Table II.

According to formula (15) even under the most favorable conditions our error bound for the evaluation of polynomials by Horner's method is bounded by 2a. One of these a's comes from evaluating the first sum $\sum_{i=1}^{n} x^{i-1}b_iR(b_i)$ in formula (14) with the assumption that $R(b_i)=a$. If $|x|$ is sufficiently small and accurate, and $R(b_1)=0$, then

$$\sum_{i=1}^{n} x^{i-1}b_iR(b_i) = b_1R(b_1) + xb_2R(b_2) + \ldots = 0 + xb_2R(b_2) + \ldots,$$

showing that the contribution of $\frac{\sum_{i=1}^{n} x^{i-1}b_iR(b_i)}{P_{n-1}}$ $< a$. The second a comes from $a\sum_{i=1}^{n-1} x^{i-1}P_{n-i}$, which corresponds to the $aP_i$ in (13). For very small $|x|$, the sum is equal to $aP_{n-1}$ and again the contribution of this term to the error is traced to the same source: the accuracy of $b_1$.

So in computing elementary functions using polynomial evaluation by Horner's method, it is essential to represent the first coefficient exactly, even though we obtain the rest of the terms from a

minmax approximation. For example, we must use an exact 1 in $e^x=1 + (\ )$, or $\cos(x)=1 - (\ )$, and an exact x for $\sin(x) = x - (\ )$ to minimize the total computational error.

## Conclusions

We have shown that our simplified method of error analysis reflects error behavior quite well. It is especially useful when many mathematically equivalent forms with different computational characteristics exist for a numerical algorithm. Example one shows how easy it is to choose the "correct" computational formula, that is, the one which gives the smallest computational error.

Secondly, our error analysis helps to obtain computationally sound algorithms for use in a library of mathematical functions, where an additional bit lost a the beginning of the computation might have serious consequences toward the end of the computation. Thirdly, our model helps trace lost accuracy to its origin, as was shown in the discussion at the end of the last section.

Finally, the simplicity of the computations involved in the error analysis might encourage people to perform error analysis on their own numerical algorithms more often.


## Acknowledgments

I am indebted to my wife Elaine for her critical review of the form of this paper and to L. Liddiard of the UCC for many discussions about computational errors.


## References


[1] Wilkinson, J.H.; Rounding Errors in Algebraic Processes; 1963; Prentice-Hall.

[2] McCracken, D.D. and Dorn, W.S.; Numerical Methods and Fortran Programming; 1964; John Wiley.

[3] Stoutemyer, D.R.; Automatic Error Analysis Using Computer Algebraic Manipulation; 1977; ACM Trans on Math. Software; Vol. 3, N1.


| i | $R(y_i)$ | $y_i$ for $\sqrt{5}=9/4$ | $R(y_i)$ observed | $R(y_i)$ predicted |
|---|---|---|---|---|
| 1 | $27R(\sqrt{5}) + 11a$ | $(1/17)^3=2.03542E-4$ | -.176 | -.178 |
| 2 | $57R(\sqrt{5}) + 11a$ | $(1/4)^6=2.44141E-4$ | -.411 | -.36 |
| 3 | $-481.R(\sqrt{5}) + 486a$ | $0 = 0.$ | 1.000 | 3.00 |
| 4 | $-1.67*10^7R(\sqrt{5})+1.67*10^7a$ | $-18 =-1.80000E+1$ | 1.0400 E+5 | 1.04E+5 |
| 5 | $-3.17R(\sqrt{5}) + 11a$ | $(4/17)^6=1.69694E-4$ | 1.95E-2 | 1.97E-2 |
| 6 | $-1.5R(\sqrt{5}) + 6.5a$ | $(1/18)^3=1.71468E-4$ | 9.26E-3 | 9.34E-3 |
| 7 | $-.5R(\sqrt{5}) + 1.5a$ | $(1/5796)=1.72533E-4$ | 3.10E-3 | 3.12E-3 |

Table I

Comparison of different formulae for the evaluation of

$$y=\left(\frac{\sqrt{5}-2}{\sqrt{5}+2}\right)^3 \simeq 1.73070E-4$$

with $R(\sqrt{5})= -6.2306E-3$

| X | $R_p(e^X)$ | $R_p(e^X)$ | $R_H(e^X)$ | $R_H(e^X)$ | N |
|---|---|---|---|---|---|
| | observed | predicted | observed | predicted | |
| -2.0 | .55E-12 | .82E-11 | .43E-12 | .23E-11 | 22 |
| -1.6 | .22E-12 | .33E-11 | .23E-12 | .90E-12 | 20 |
| -1.2 | .87E-13 | .14E-11 | .98E-13 | .34E-12 | 18 |
| -.8 | .50E-13 | .50E-12 | .50E-13 | .12E-12 | 15 |
| -.4 | .22E-13 | .18E-12 | .22E-13 | .44E-13 | 12 |
| 0 | .98E-14 | .14E-13 | .98E-14 | .14E-13 | 2 |
| .4 | .21E-15 | .82E-13 | .49E-14 | .20E-13 | 12 |
| .8 | .49E-14 | .10E-12 | .49E-14 | .26E-13 | 15 |
| 1.2 | .57E-14 | .11E-12 | .57E-14 | .31E-13 | 17 |
| 1.6 | .12E-13 | .13E-12 | .68E-14 | .37E-13 | 19 |
| 2.0 | .92E-14 | .14E-12 | .92E-14 | .43E-13 | 20 |

Table II

Comparisons of error estimates for the evaluation of $e^X$ by power method and by Horner's method according to formulae (8) and (12). $R_p(e^X)$-relative error in the power method $R_H(e^X)$-relative error in Horner's method and N - number of terms used. (Note that the coefficients $a_i$ are perturbed to make comparison more realistic).