A comparison of polynomial evaluation schemes

L. Fousse [a] S. Schmitt [b]

[a] *LORIA/INRIA Lorraine, 615 rue du jardin botanique, F-54602 Villers-lès-Nancy Cedex*

[b] *MPI für Informatik, 66123 Saarbrücken*

Abstract

The goal of this paper is to analyze two polynomial evaluation schemes for multiple precision floating point arithmetic. Polynomials are used extensively in numerical computations (Taylor series for mathematical functions, root finding) but a rigorous bound of the error on the final result is seldom provided. We provide such an estimate for the two schemes and find how to reduce the number of operations required at run-time by a dynamic error analysis. This work is useful for floating point polynomial arithmetic.

Key words: polynomial evaluation, bounded error

1 Goal and motivations

The goal is to compare two polynomial evaluation schemes. We want to compute the sum:

$$P(x) = \sum_{i=0}^{l} a_i x^i$$

and provide an error bound on the final result with respect to the number of summands $l + 1$ and the precision F used for the intermediate results (the "internal" precision).

Email addresses: `laurent@komite.net` (L. Fousse), `sschmitt@mpi-sb.mpg.de` (S. Schmitt).

[1] Supported by the Marie-Curie Training Site programme of the EU.

We make the following assumptions:

- $|x| \leq 2^{-k}$ with $k \geq 1$;
- $|a_i| \leq |a_0| = 1$;
- the final result is rounded to f bits with $f < F$: f is the precision expected by the user on the final result;
- during the computation, inputs as well as intermediate results are not denormalized numbers.

These conditions are not as restrictive as one might think first. Fast and accurate polynomial evaluation is needed in mathematical libraries for the elementary functions [4] (log, exp, cos, ...). In this context, there is first an argument reduction based on the properties of the function to evaluate. The goal is to have x in an interval as small as possible in which the polynomial approximation of the function is good. A typical value for k would be 5 in this case. The second hypothesis is needed to avoid cancellation to zero where no meaningful result on the final error can be given, and is actually often verified by Taylor expansions of elementary functions or minimax approximations.

In [1] such an evaluation scheme based on argument reduction and polynomial evaluation with increased precision is given for the exponential function for example.

First we present the algorithm used for each evaluation method. A bound on the error on the final result is then computed for each algorithm, and, following a discussion on how this evaluation can be made easier at runtime, actual error and time measurements are provided.

Throughout the paper we let $\epsilon = 2^{-F}$ the *machine epsilon* for F-digit floating point numbers, i.e. the relative difference between two consecutive normalized floating point numbers.

For a non-zero real number x we define the *exponent* $E(x) := 1 + \lfloor \log_2 |x| \rfloor$, such that $2^{E(x)-1} \leq |x| < 2^{E(x)}$. We further define $\mathrm{ulp}_F(x) = 2^{E(x)-F}$. When x is a floating point number with a mantissa of F bits, $\mathrm{ulp}_F(x)$ corresponds to the weight of the last mantissa bit.

2 Schemes

There are two well-known polynomial schemes.

Basic Scheme We compute $P(x)$ in increasing power order. At each step, y_i is an approximation of x^i and z_i of $a_i x^i$. Then t_l is an approximation of $P(x)$. More precisely, the algorithm used is the following:

$t_0 \leftarrow a_0$
$y_0 \leftarrow 1$
for $i \leftarrow 1$ **to** l
  $y_i \leftarrow \circ(y_{i-1}x)$
  $z_i \leftarrow \circ(y_i a_i)$
  $t_i \leftarrow \circ(t_{i-1} + z_i)$

Horner Scheme We compute $P(x)$ with the classical Horner method. Here, s_0 is an approximation of $P(x)$.

$s_l \leftarrow a_l$
for $i \leftarrow l-1$ **downto** 0
  $s_i \leftarrow a_i + xs_{i+1}$

3 Error estimate

3.1 Basic scheme

We estimate the error of the basic method $\varepsilon_{fin} = \left| t_l - \sum_{i=0}^{l} a_i x^i \right|$. To do this we introduce the following two sources of error:

- the error of the computation of z_i at each step. We call that the *evaluation error* and its value is

$$\varepsilon_{eval} = \left| \sum_{i=0}^{l} (a_i x^i - z_i) \right|.$$

  Here we define $z_0 = a_0$.
- the summation error. This is the error that is caused by the rounding in the addition at each step, and its value is

$$\varepsilon_{add} = \left| t_l - \sum_{i=0}^{l} z_i \right|.$$

Then $\varepsilon_{fin} \leq \varepsilon_{eval} + \varepsilon_{add}$.

3.1.1 The evaluation error

The value y_i is the result of i multiplications

$$y_i = x^i \prod_{j=1}^{i} (1 + \theta_j).$$

As $z_i = \circ(y_i a_i)$, we get for z_i

$$z_i = a_i \cdot x^i (1+\theta'_i) \prod_{j=1}^{i} (1+\theta_j).$$

The relative error at step i is given by

$$\frac{|z_i - a_i \cdot x^i|}{|a_i x^i|} \leq \left| 1 - (1+\theta'_i) \prod_{j=1}^{i} (1+\theta_j) \right|.$$

Assuming rounding is to nearest, we know that

$$\forall\ j \in \{1,\ldots,l\}, \quad |\theta_j| \leq 2^{-F} = \epsilon \quad \text{and} \quad \forall\ i \in \{1,\ldots,l\}, \quad |\theta'_i| \leq 2^{-F} = \epsilon.$$

Lemma 1 *Let θ_j for $j = 1,\ldots,i$ be given such that $\sum_{j=1}^{i} |\theta_j| \leq \frac{1}{2}$. Then*

$$1 - \sum_{j=1}^{i} |\theta_j| \leq \prod_{j=1}^{i} (1+\theta_j) \leq 1 + 2 \sum_{j=1}^{i} |\theta_j|.$$

PROOF. By induction on i. $\square$

As $y_0 = 1$ we know that $\theta_1 = 0$ (hence $y_1 = x$). Then, for $i \leq 2^{F-1}$, the hypothesis of the lemma is fulfilled:

$$|\theta'_i| + \sum_{j=2}^{i} |\theta_j| \leq \sum_{j=1}^{i} 2^{-F} = i2^{-F} \leq \frac{1}{2} \quad \Leftrightarrow \quad i \leq 2^{F-1}.$$

We get the following relative error bound on z_i:

$$\frac{|z_i - a_i \cdot x^i|}{|a_i x^i|} \leq 2i2^{-F} = i2^{1-F} \quad \text{if} \quad i \leq 2^{F-1}.$$

We will therefore assume that $l \leq 2^{F-1}$. The evaluation error can then be bounded the following way :

$$\begin{aligned} \varepsilon_{eval} &\leq \sum_{i=1}^{l} i2^{1-F} |a_i x^i| \leq 2^{1-F} \sum_{i=1}^{l} i2^{-ki} \\ &\leq 2^{1-F} \sum_{i=1}^{\infty} i2^{-ki} = \frac{2^{1-k-F}}{(1-2^{-k})^2} \\ &\leq 2^{3-k-F}. \end{aligned}$$

3.1.2 The summation error

The summation error

$$\varepsilon_{add} = \left| t_l - \sum_{j=0}^{l} z_j \right|$$

is naively bounded by $\varepsilon_{add} = \frac{1}{2} \sum_{i=1}^{l} ulp_F(t_i)$ since rounding is to nearest. A better bound can be given if for example the exponents of the summands decrease after the first error (which can be easily detected at run time), see [2].

3.1.3 The final error

Since the summation error is given relative to the ulp of the current sum at each step, we need to know how the ulp (or the exponent) of this sum changes in the computation. This is necessary as well to get a relative error at the end.

Each time the exponent of the current sum decreases, the relative error accumulated so far is multiplied by 2 (the basis of the computation), so we don't want to let this exponent decrease too much. For that we need an upper bound of the z_i.

We assume that we still use rounding to nearest. Then

$$\begin{aligned} |z_i| &\leq |a_i x^i|(1 + i2^{1-F}) \\ &\leq 2^{-ki}(1 + i2^{1-F}) \\ &= 2^{F-ki-1}(1 + i2^{1-F})\mathrm{ulp}_F(t_0). \end{aligned}$$

Let i_0 be the first index at which the exponent of t_i decreases (it can grow before), i.e.:

$$E(t_{i_0}) < E(t_{i_0-1}) \quad \text{and} \quad \forall\, j \in \{0, \ldots, i_0 - 1\},\ E(t_j) \geq E(t_{j-1})$$

and let $i_0 = l+1$ if there is no decrease in exponent. Then we have $\mathrm{ulp}_F(t_{i_0-1}) = 2^{\alpha}\mathrm{ulp}_F(t_0)$ with $\alpha \geq 0$.

Lemma 2 *Let $F \geq 5$ and $p \leq 2^{F-3}$. We further assume that $ki_0 + \alpha \geq 3$. Then $\forall j \in \{i_0, \ldots, i_0 + p\}$,*

$$\mathrm{ulp}_F(t_{i_0}) = \frac{\mathrm{ulp}_F(t_{i_0-1})}{2} \leq \mathrm{ulp}_F(t_j) \leq \mathrm{ulp}_F(t_{i_0-1}).$$

PROOF. The proof is done by induction on j. First we show that

$$\frac{\mathrm{ulp}_F(t_{i_0-1})}{2} = \mathrm{ulp}_F(t_{i_0}) < \mathrm{ulp}_F(t_{i_0-1}).$$

From the definition of t_{i_0} the last inequality follows directly. For the first equality it suffices to show that

$$\mathrm{ulp}_F(t_{i_0}) \geq \frac{\mathrm{ulp}_F(t_{i_0-1})}{2}.$$

We first estimate $|t_{i_0}|$.

$$\begin{aligned}|t_{i_0-1} + z_{i_0}| &\geq |t_{i_0-1}| - |z_{i_0}| \\ &\geq \left(2^{F-1} - 2^{F-ki_0-1-\alpha}(1 + i_0 2^{1-F})\right) \mathrm{ulp}_F(t_{i_0-1}) \\ &\geq 2^{F-2}\mathrm{ulp}_F(t_{i_0-1}).\end{aligned}$$

To show the last inequality we need to show

$$2^{F-ki_0-1-\alpha}(1 + i_0 2^{1-F}) \leq 2^{F-2}.$$

From the assumption it follows that $ki_0 + \alpha \geq 2$. Hence

$$2^{F-ki_0-1-\alpha}(1 + i_0 2^{1-F}) \leq 2^{F-3} + i_0 2^{-ki_0} \leq 2^{F-3} + \frac{1}{2} \leq 2^{F-2}$$

as $F \geq 2$. Since $t_{i_0} = \circ_F(t_{i_0-1} + z_{i_0})$ we know that after rounding the condition will still hold.

We now get

$$\mathrm{ulp}_F(t_{i_0}) = 2^{E(t_{i_0})-F} > 2^{-F}|t_{i_0}| \geq 2^{-2}\mathrm{ulp}_F(t_{i_0-1})$$

and because of the strict inequality it follows that

$$\mathrm{ulp}_F(t_{i_0}) \geq \frac{\mathrm{ulp}_F(t_{i_0-1})}{2}.$$

We now assume the property holds for $j \in \{i_0, \ldots, i_0 + p - 1\}$ and first show the inequality

$$\mathrm{ulp}_F(t_{i_0+p}) \geq \frac{\mathrm{ulp}_F(t_{i_0-1})}{2}.$$

To do this, we estimate $|t_{i_0+p}|$ similarly as above.

$$|t_{i_0+p-1} + z_{i_0+p}| \geq |t_{i_0-1}| - \sum_{j=i_0}^{i_0+p} |z_j| - \sum_{j=i_0}^{i_0+p-1} \frac{1}{2}\mathrm{ulp}_F(t_j)$$

$$\begin{aligned}
&\geq \left[2^{F-1} - \sum_{j=i_0}^{i_0+p} 2^{F-kj-1-\alpha}(1+j2^{1-F}) - \frac{p}{2}\right] \mathrm{ulp}_F(t_{i_0-1}) \\
&\geq 2^{F-2}\mathrm{ulp}_F(t_{i_0-1}).
\end{aligned}$$

We have to show the last inequality. If we extend the sum to infinity we get the following bound (using $ki_0+\alpha \geq 3$)

$$\begin{aligned}
&\sum_{j=i_0}^{i_0+p} 2^{F-kj-1-\alpha}(1+j2^{1-F}) + \frac{p}{2} \\
&\leq \sum_{j=i_0}^{\infty} 2^{F-1-kj-\alpha} + \sum_{j=i_0}^{\infty} j2^{-kj-\alpha} + \frac{p}{2} \\
&\leq \frac{2^{F-1-ki_0-\alpha}}{1-2^{-k}} + 2^{-ki_0-\alpha}\sum_{j=0}^{\infty}(j+i_0)2^{-kj} + \frac{p}{2} \\
&= \frac{2^{F-1-ki_0-\alpha}}{1-2^{-k}} + 2^{-ki_0-\alpha}\frac{2^{-k}}{(1-2^{-k})^2} + i_0 2^{-ki_0-\alpha}\frac{1}{1-2^{-k}} + \frac{p}{2} \\
&\leq \frac{2^{F-4}}{\frac{1}{2}} + 2^{-3}\frac{2^{-1}}{\frac{1}{4}} + i_0 2^{-i_0}\frac{1}{\frac{1}{2}} + \frac{p}{2} \\
&\leq 2^{F-3} + \frac{1}{4} + 1 + \frac{p}{2} \\
&\leq 2^{F-2}.
\end{aligned}$$

For the last inequality we use $F \geq 5$ and hence $2^{F-4} \geq 2$. Further, we use $p \leq 2^{F-3}$, and hence $\frac{p}{2} \leq 2^{F-4}$.

As for the case $j = i_0$ we can see that after rounding, $|t_{i_0+p}| \geq 2^{F-2}\mathrm{ulp}_F(t_{i_0-1})$. As before, it follows that $\mathrm{ulp}_F(t_{i_0+p}) \geq \frac{\mathrm{ulp}_F(t_{i_0-1})}{2}$.

Now we consider the other inequality

$$\mathrm{ulp}_F(t_{i_0+p}) \leq \mathrm{ulp}_F(t_{i_0-1}).$$

Again we estimate $|t_{i_0+p}|$.

$$\begin{aligned}
&|t_{i_0+p-1} + z_{i_0+p}| \\
&\leq |t_{i_0}| + \sum_{j=i_0+1}^{i_0+p} |z_j| + \sum_{j=i_0+1}^{i_0+p-1} \frac{1}{2}\mathrm{ulp}_F(t_j) \\
&\leq \left[2^{F-1} - \frac{1}{2} + \sum_{j=i_0+1}^{i_0+p} 2^{F-kj-1-\alpha}(1+j2^{1-F}) + \frac{p-1}{2}\right] \mathrm{ulp}_F(t_{i_0-1}).
\end{aligned}$$

For the last inequality we need

$$|t_{i_0}| \leq (2^F - 1)\mathrm{ulp}_F(t_{i_0}) = \left(2^{F-1} - \frac{1}{2}\right)\mathrm{ulp}_F(t_{i_0-1}).$$

We need to show that

$$\sum_{j=i_0+1}^{i_0+p} 2^{F-kj-1-\alpha}(1 + j2^{1-F}) + \frac{p}{2} \leq 2^{F-1}$$

which is trivial with what we've already proved for the other inequality. It follows that $|t_{i_0+p-1} + z_{i_0+p}| \leq (2^F - \frac{1}{2})\mathrm{ulp}_F(t_{i_0-1})$. Rounding leads to the same estimate for $|t_{i_0+p}|$:

$$|t_{i_0+p}| \leq \left(2^F - \frac{1}{2}\right)\mathrm{ulp}_F(t_{i_0-1}) < 2^F\mathrm{ulp}_F(t_{i_0-1}).$$

Then

$$\mathrm{ulp}_F(t_{i_0+p}) \leq 2^{-F+1}|t_{i_0+p}| < 2\mathrm{ulp}_F(t_{i_0-1})$$

and hence $\mathrm{ulp}_F(t_{i_0+p}) \leq \mathrm{ulp}_F(t_{i_0-1})$. □

Note that for $ki_0 + \alpha = 2$ the lemma is still true, but we need to be more careful with the estimates.

As $p \leq l \leq 2^{F-1}$, we know that the exponent of the final result is at least one less that the highest exponent of the partial sums. The final error on t_l is then:

$$\begin{aligned}
\varepsilon_{fin} &\leq \varepsilon_{eval} + \varepsilon_{add} \\
&\leq 2^{3-k-F} + \frac{l}{2}\mathrm{ulp}_F(t_{i_0-1}) \\
&= \left(2^{2-k-\alpha} + \frac{l}{2}\right)\mathrm{ulp}_F(t_{i_0-1}) \\
&\leq (2^{3-k-\alpha} + l)\mathrm{ulp}_F(t_l).
\end{aligned}$$

Here we used

$$\mathrm{ulp}_F(t_{i_0-1}) = 2^{\alpha}\mathrm{ulp}_F(t_0) = 2^{\alpha+1-F}.$$

3.2 Improvement of the basic method

From the error bound we see that the final error mostly comes from the rounding error at each step and not from the evaluation error. In order to decrease the evaluation cost it could be meaningful to use a reduced precision for the z_i.

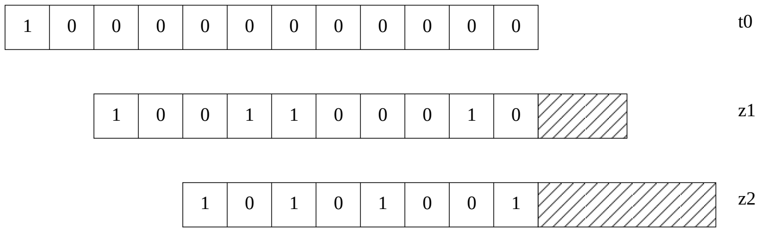

Fig. 1. Decreasing ulp of z_i.

This idea appears already in [5] for summing a series with decreasing terms. A gain of a factor of up to three is given in [5] for the summation time; a detailed error analysis is however not provided.

3.2.1 The evaluation error

As the value of z_i decreases by an order of 2^{-k} compared to the value of z_{i-1}, the first idea is to use only $F - ki$ bits of precision to compute z_i. For example Figure 1 shows how the ulp of the z_i decreases for $k = 2$ and $|a_i| = 1$ (the dashed boxes are the neglected bits in this improved method).

The relative error is now :

$$z_i = a_i x^i (1 + \theta'_i) \prod_{j=1}^{i} (1 + \theta_j)$$

with $\forall i \in \{1, \ldots, l\}, |\theta'_i| \leq 2^{ki-F}$ and $\forall j \in \{1, \ldots, i\}, |\theta_j| \leq 2^{kj-F}$.

To apply Lemma 1, we need $F \geq ki + 3$. Then

$$|\theta'_i| + \sum_{j=1}^{i} |\theta_j| \leq 2^{ki-F} + \sum_{j=1}^{i} 2^{kj-F} = 2^{ki-F} + \frac{2^k}{2^k - 1} 2^{-F}(2^{ki} - 1) \leq 3 \cdot 2^{ki-F} \leq \frac{1}{2}.$$

(Note that we don't use $\theta_1 = 0$ here, which does not improve the estimates.) Using the lemma and the assumptions on $|\theta'_i|$, $|\theta_i|$, we get the two inequalities

$$1 - 2^{ki-F} - \sum_{j=1}^{i} 2^{kj-F} \leq (1 + \theta'_i) \prod_{j=1}^{i} (1 + \theta_j) \leq 1 + 2 \left(2^{ki-F} + \sum_{j=1}^{i} 2^{kj-F} \right).$$

The relative error at each step is given by

$$\frac{|z_i - a_i x^i|}{|a_i x^i|} = \left| 1 - (1 + \theta'_i) \prod_{j=1}^{i} (1 + \theta_j) \right| \leq 2^{-F} e(i)$$

with

$$e(i) = 2 \left(2^{ki} + \sum_{j=1}^{i} 2^{kj} \right) = 2 \left(2^{ki} + \frac{2^k}{2^k - 1} (2^{ki} - 1) \right).$$

The following estimate (for $F \geq ki + 3$ for $i = 1, \ldots, l$) is useful.

$$2^{-ki} e(i) = 2 \left(1 + \frac{2^k}{2^k - 1}(1 - 2^{-ki})\right) \leq 2(1 + 2) = 6.$$

The evaluation error is then bounded by

$$\varepsilon_{eval} \leq \sum_{i=1}^{l} 2^{-F} e(i) |a_i x^i| \leq 2^{-F} \sum_{i=1}^{l} e(i) 2^{-ki} \leq 6l 2^{-F}.$$

3.2.2 The final error

To get a similar bound as before on the final error, we need to compare the ulp of the final result with the accumulated error. Again we assume rounding to nearest. Then

$$\begin{aligned} |z_i| &\leq |a_i x^i| (1 + 2^{-F} e(i)) \\ &\leq 2^{-ki} (1 + 2^{-F} e(i)) \\ &\leq 2^{F-ki-1} (1 + 2^{-F} e(i)) \mathrm{ulp}_F(t_0). \end{aligned}$$

Using $F \geq ki + 3$ for all i, we can further estimate $2^{-F} \leq 2^{-ki-3}$ and

$$|z_i| \leq 2^{F-ki-1} \left(1 + \frac{6}{8}\right) \mathrm{ulp}_F(t_0) \leq 2^{F-ki} \mathrm{ulp}_F(t_0).$$

Let i_0 as before be the first index at which the exponent of t_i decreases ($i_0 = l + 1$ if there is no decrease). Also let $\alpha \geq 0$ with $\mathrm{ulp}_F(t_{i_0-1}) = 2^{\alpha} \mathrm{ulp}_F(t_0)$. The next lemma is similar to Lemma 2.

Lemma 3 *Let $p \leq 2^{F-2}$ and $F \geq kl + 3$. We further assume that $ki_0 + \alpha \geq 4$. Then $\forall j \in \{i_0, \ldots, i_0 + p\}$,*

$$\mathrm{ulp}_F(t_{i_0}) = \frac{\mathrm{ulp}_F(t_{i_0-1})}{2} \leq \mathrm{ulp}_F(t_j) \leq \mathrm{ulp}_F(t_{i_0-1}).$$

PROOF. The proof is exactly the same as the proof of Lemma 2, using the estimates from above. □

We can now estimate the partial sum exponent as before and get the final error

$$\varepsilon_{fin} \leq \varepsilon_{eval} + \varepsilon_{add} \leq 7l \, \mathrm{ulp}_F(t_l).$$

The ideas of the error estimate for the Horner method are taken from [3]. Looking at the relative error, in step i we have $s_i = (a_i+(x\cdot s_{i+1})(1+\theta_i))(1+\theta_i')$ with $|\theta_i|, |\theta_i'| \leq 2^{-F}$. The general formula is then

$$s_0 = \sum_{i=0}^{l}(1+\theta_i')\left(\prod_{j=0}^{i-1}(1+\theta_j)(1+\theta_j')\right)a_i x^i.$$

Defining

$$\delta_i := (1+\theta_i')\left(\prod_{j=0}^{i-1}(1+\theta_j)(1+\theta_j')\right) - 1,$$

we can write

$$s_0 = \sum_{i=0}^{l}(1+\delta_i)a_i x^i.$$

We then can estimate the difference between s_0 and $P(x)$:

$$|s_0 - P(x)| = \left|\sum_{i=0}^{l}(1+\delta_i)a_i x^i - \sum_{i=0}^{l} a_i x^i\right| = \left|\sum_{i=0}^{l}\delta_i a_i x^i\right| \leq \sum_{i=0}^{l}|\delta_i||a_i||x|^i.$$

Now we need a bound for $|\delta_i|$. Using Lemma 1, we get

$$1-(2i+1)2^{-F} \leq \delta_i + 1 = (1+\theta_i')\left(\prod_{j=0}^{i-1}(1+\theta_j)(1+\theta_j')\right) \leq 1+2(2i+1)2^{-F}$$

as long as $2i+1 \leq 2^{F-1}$. Therefore, for $l \leq 2^{F-2} - \frac{1}{2}$, we get the estimate for all $i = 0, \ldots, l$:

$$|\delta_i| \leq 2(2i+1)2^{-F} = (2i+1)2^{1-F}.$$

The error estimate for the Horner scheme is then

$$|s_0 - P(x)| \leq \sum_{i=0}^{l}(2i+1)2^{1-F}|a_i||x|^i.$$

Using $|a_i| \leq 1$ and $|x| \leq 2^{-k}$ we can further estimate

$$|s_0 - P(x)| \leq 2^{1-F}\sum_{i=0}^{l}(2i+1)2^{-ik} \leq 2^{1-F}\frac{1+2^{-k}}{(1-2^{-k})^2}.$$

The last inequality is obtained by extending the sum to infinity. The goal is to get a lower bound of $|s_0|$ so we can bound the relative error.

$$|P(x)| \geq 1 - \sum_{i=1}^{l}|a_i x^i| \geq 1 - \sum_{i=1}^{l}2^{-ki} = 1 - 2^{-k}\frac{1-2^{-kl}}{1-2^{-k}}.$$

We need to distinguish two cases here:

- If $k \geq 2$, we get
$$|P(x)| \geq 1 - 2^{-k}\frac{1}{1-2^{-k}} \geq \frac{2}{3}.$$
Using the estimate from above we get the lower bound of $|s_0|$:
$$|s_0| \geq \frac{2}{3} - 2^{1-F}\frac{1+2^{-k}}{(1-2^{-k})^2} = \frac{2}{3}\cdot\frac{(1-2^{-k})^2 - 3\cdot 2^{-F}(1+2^{-k})}{(1-2^{-k})^2}.$$
And the relative error is then bounded by :
$$\epsilon_{rel} = \frac{|s_0 - P(x)|}{|s_0|} \leq \frac{3\cdot 2^{-F}(1+2^{-k})}{(1-2^{-k})^2 - 3\cdot 2^{-F}(1+2^{-k})}.$$
The worst case is $k = 2$ which gives $\epsilon_{rel} \leq \frac{5\cdot 2^{-F}}{\frac{3}{4}-5\cdot 2^{-F}}$. For example with $F \geq 6$ this means we will lose at most 3 bits of precision in the final result [2] .
- $k = 1$. We get $|P(x)| \geq 2^{-l}$. The lower bound of s_0 is then $|s_0| \geq 2^{-l} - 12\cdot 2^{-F}$ which is only meaningful if $F \geq l + 4$. The relative error is then
$$\epsilon_{err} \leq \frac{12\cdot 2^{-F}}{2^{-l} - 12\cdot 2^{-F}}.$$

For example for $F = 53$ and $l = 10$ we estimate that we can lose up to 14 bits in the final result. This estimate is quite bad but this comes from the static lower bound for $P(x)$ and s_0. At run time the final value of s_0 is known and a better error estimate can almost always be given.

3.4 Improvement of the Horner scheme

Similar to the basic method, we estimate the errors after cutting the number of bits in the internal representation. We start the computation with low accuracy and increase the size of the partial result at each step. Here we have the relative errors $|\theta_i'| \leq 2^{ki-F}$ and $\theta_i \leq 2^{ki-F}$ for $i = 0, \ldots, l$. Again we define

$$\delta_i := (1+\theta_i')\left(\prod_{j=0}^{i-1}(1+\theta_j)(1+\theta_j')\right) - 1.$$

To apply Lemma 1, we need the inequality

$$2^{ki-F} + 2\sum_{k=0}^{i-1} 2^{kj-F} \leq \frac{1}{2}.$$

[2] $\epsilon = 2^{-6}$.

This is satisfied for $F \geq ki + 3$:

$$\begin{aligned} 2^{ki-F} + 2\sum_{k=0}^{i-1} 2^{kj-F} &= 2^{-F}\left(2^{ki} + \frac{2}{2^k - 1}(2^{ki} - 1)\right) \\ &\leq 2^{-F}\left(2^{ki} + 2 \cdot 2^{ki}\right) \\ &= 3 \cdot 2^{ki-F} \leq \frac{1}{2}. \end{aligned}$$

Now we can give a bound for $|\delta_i|$. Using Lemma 1, we get

$$1 - 2^{-F}\left(2^{ki} + 2\sum_{k=0}^{i-1} 2^{kj}\right) \leq \delta_i + 1 \leq 1 + 2 \cdot 2^{-F}\left(2^{ki} + 2\sum_{k=0}^{i-1} 2^{kj}\right)$$

as long as $F \geq ki + 3$. We write

$$e(i) = 2\left(2^{ki} + 2\sum_{k=0}^{i-1} 2^{kj}\right) = 2\left(2^{ki} + \frac{2}{2^k - 1}(2^{ki} - 1)\right)$$

and get, for $F \geq kl + 3$, the estimate for all $i = 0, \ldots, l$:

$$|\delta_i| \leq 2^{-F} e(i).$$

We later need the estimate

$$2^{-ki} e(i) = 2\left(1 + \frac{2}{2^k - 1}(1 - 2^{-ki})\right) \leq 6.$$

Now we can estimate the difference between s_0 and $P(x)$:

$$|s_0 - P(x)| = \left|\sum_{i=0}^{l} \delta_i a_i x^i\right| \leq \sum_{i=0}^{l} 2^{-F} e(i) |a_i| |x|^i.$$

Using $|a_i| \leq 1$ and $|x| \leq 2^{-k}$ we can further estimate

$$|s_0 - P(x)| \leq 2^{-F} \sum_{i=0}^{l} 2^{-ik} e(i) \leq 6 \cdot 2^{-F}(l + 1).$$

We can use the same lower bound for $|P(x)|$ as above:

$$|P(x)| \geq 1 - 2^{-k}\frac{1 - 2^{-kl}}{1 - 2^{-k}}.$$

We need to distinguish two cases here:

- If $k \geq 2$, we get
$$|P(x)| \geq \frac{2}{3}.$$

Using the estimate from above we get the lower bound of $|s_0|$:

$$|s_0| \geq \frac{2}{3} - 6 \cdot 2^{-F}(l+1).$$

(For the right hand side to be positive we need $2^{-F}(l+1) \leq \frac{1}{9}$.) The relative error is then bounded by :

$$\epsilon_{rel} = \frac{|s_0 - P(x)|}{|s_0|} \leq \frac{6 \cdot 2^{-F}(l+1)}{\frac{2}{3} - 6 \cdot 2^{-F}(l+1)}.$$

- $k = 1$. We get $|P(x)| \geq 2^{-l}$. The lower bound of s_0 is then $|s_0| \geq 2^{-l} - 6 \cdot 2^{-F}(l+1)$. (The right hand side is positive for example for $F \geq 2l+3$.) The relative error is then

$$\epsilon_{err} \leq \frac{6 \cdot 2^{-F}(l+1)}{2^{-l} - 6 \cdot 2^{-F}(l+1)}.$$

4 Providing a dynamic error bound

Dynamic error bounds are bounds that can be deduced from the partial results, as opposed to *static bounds* which are estimated before the actual computation.

The static error bounds we provided are good to have an idea of the maximum error these algorithms can yield in the worst case (although we don't provide examples showing that these bounds are optimal).

However, the static analysis shows its limits when we want to give an error bound that is relative to the final result. The lower bound of the final result is rather pessimistic and is not relevant at run-time since we know the final result. This is confirmed by our experiments where the static error is worse by a factor of two. The algorithms were therefore written to provide the evaluation of $P(x)$ and a bound on the final error at the same time so that we know how many bits are significant.

For the Horner and improved Horner schemes nothing needs to be done, we only have to compare the value of the error given by the static analysis with the value of the final result.

For the basic and improved basic methods, we compute the maximum of the exponents of the intermediate results t_i. We know that this exponent is the exponent of t_{i_0-1} and can then compute α (see section 3). We also know whether the exponent of t_l is that of t_{i_0} or of t_{i_0-1} and we don't overestimate the error too much.

5 Conclusion and future work

In this paper we presented and analyzed two polynomial evaluation schemes and improved them. The results confirm the reputation of the Horner method having more numerical stability.

For large inputs, the improved methods are faster than the original methods. This gain was predictable from the theoretical complexity: depending on the time complexity of the multiplication, the truncated basic method gains a factor ranging from 2 to 3 on the time spent doing multiplications compared to the basic method. The actual error of the improved methods is comparable to the error of the original methods. It is therefore not straightforward which method to chose as there is a trade-off between very good accuracy and good efficiency [3] (Horner) and not so good accuracy but even higher efficiency (improved Horner method in the last experiment). The choice is highly context-dependent.

As a future work, Smith gives in [5] a method to sum series where the terms are "related". In our tests the coefficient were precomputed once but we could apply our error analysis to Smith's "concurrent series" summing to get an efficient method to sum such series with bounded error.

References

[1] David Defour, Florent de Dinechin, and Jean-Michel Muller. Correctly rounded exponential function in double precision arithmetic. In *Proceedings of SPIE 46th Annual Meeting, International Symposium on Optical Science and Technology*, San Diego, USA, 2001.

[2] Demmel J. and Hida Y. Accurate floating-point summation, 2002. `http://www.cs.berkeley.edu/~yozo/papers/csd-02-1180.ps.gz`.

[3] D. Manocha. Error analysis for polynomial evaluation. `http://www.cs.unc.edu/~smp/COMP205/LECTURES/ERROR/lec4.ps`.

[4] Jean-Michel Muller. *Elementary Functions. Algorithms and Implementation.* Birkhauser, 1997. 232 pages.

[5] David M. Smith. Algorithm 693. a Fortran package for floating-point multiple-precision arithmetic. *ACM Transactions on Mathematical Software*, 17(2):273–283, 1991.

[6] INRIA. Spaces Project. The MPFR library. `http://www.mpfr.org/`.

[3] We're interested in the time efficiency.

A Experiments

The different algorithms were written with the MPFR [6] floating-point library and ran on a P4 processor at 3GHz, taking the l first term of the exponential series for P, with several values of the different parameters. The values of l were chosen so that every term z_i in the basic method is greater than the ulp of the current result (no term is completely useless). The polynomials are computed by evaluating the first terms of the exponential series in sequence and rounding each term to the current precision f. The error is computed with respect to the correct value, that is the value computed with infinite precision. The tables show the runtime, the predicted accuracy and the actual accuracy. Each method is run with F as the working precision and returns the final result on F bits. The errors are given in ulp of the final result.

We took $x = \circ(\frac{1}{\sqrt{42}})$ rounded to f bits.

$F = 53, f = 40, l = 11$

Method	measured error	dynamic error	static error	Time(μs)
basic	1.063269	6.5	13	5.646
basic improved	1.063269	3.85e1	77	10.834
Horner	0.063269	2.22	4.44	4.234
Horner improved	0.063269	3.60e1	7.20e1	9.918

$F = 410, f = 400, l = 57$

Method	measured error	dynamic error	static error	Time(ms)
basic	0.100492	2.95e1	59	0.135
basic improved	0.100492	2.00e2	3.99e2	0.149
Horner	0.100492	2.22	4.44	0.071
Horner improved	0.100492	1.74e2	3.48e2	0.099

$F = 4010, f = 4000, l = 404$

Method	measured error	dynamic error	static error	Time(ms)
basic	1.672861	2.03e2	4.06e2	39.375
basic improved	1.672861	1.41e3	2.83e3	39.062
Horner	0.327139	2.22	4.44	20.156
Horner improved	0.672861	1.22e3	2.44e3	17.812

One test with $x = \circ(\frac{1}{\sqrt{4200}})$ which verifies $k = 6$:

$$F = 4010, f = 4000, l = 311$$

Method	measured error	dynamic error	static error	Time(ms)
basic	6.589485	1.56e2	3.12e2	30.469
basic improved	6.589485	1.09e3	2.18e3	25.312
Horner	0.410515	1.05	2.10	15.469
Horner improved	0.410515	9.36e2	1.87e3	10.781

From the results we see that no method ever underestimated the actual error it did but rather provided safe error bounds usually by several orders of magnitude larger than the measured error in the case of the first two methods. When computing the static error, the ulp of the final result is usually underestimated by one, which explains the factor of two between the static and the dynamic errors.

The basic and basic improved methods achieve the same accuracy in our tests confirming the intuition that computing too many bits for the higher terms of the series is inefficient. The improved method is worse than the basic method for small inputs because we lose more time evaluating the error and rounding the partial result than actually computing the evaluation.

The Horner method has always the best accuracy (measured and predicted) but it not always as efficient as the truncated (improved) methods.

The same experiments were done with the cosine series but were not included as the results are similar.