Computation of the error functions erf and erfc in arbitrary precision with correct rounding

Sylvain Chevillard
Arenaire, LIP, ENS-Lyon, France
Sylvain.Chevillard@ens-lyon.fr

Nathalie Revol
INRIA, Arenaire, LIP, ENS-Lyon, France
Nathalie.Revol@ens-lyon.fr

ICMS, Castro Urdiales, Spain 1-3 September 2006
IEEE-754 standard for floating-point arithmetic

Floating-point number: \( s.m.\beta^e \) or 0, a denormal, \( \pm \infty \), NaN.
- \( s \): sign
- \( m \): mantissa \( \in [2^{p-1}/2^p, (2^p - 1)/2^p] \), \( p \) is the precision
- \( e \): exponent \( \in [e_{\text{min}}, e_{\text{max}}] \)
- \( \beta \): basis, usually equal to 2

IEEE-754 standard for floating-point arithmetic:
- fixed formats for single and double precision
- specifies 4 rounding modes
- specifies the arithmetic and algebraic operations \(+, -, \times, \div, \sqrt{\cdot}\).

Advantages:
- well-specified arithmetic, reproducible results
- error bounds can be established and proofs can be done
Desirable extensions of the IEEE-754 standard

**correctly rounded elementary functions**:
cf. current revision of the standard

**arbitrary precision**: (software)
cf. MPFR

**correctly rounded special functions**:

**correctly rounded functions in arbitrary precision**:
cf. MPFR for elementary functions
How can one return the correctly rounded evaluation of a function $f$?

To return the correctly rounded evaluation of $f(x)$ in precision $p$:
1- approximate $f(x)$ with larger precision $q$, error $< \varepsilon$
2- if can_round $(f(x), \varepsilon, p)$

then return it
3- otherwise increase $q$, decrease $\varepsilon$ and try again.
Outline of this talk

• the error function erf
• algorithm to return the correctly rounded evaluation of erf($x$)
• experimental results
• the complementary error function erfc
• conclusion and hints for improvements
• current work on interval arithmetic and algorithms: MPFI
The error function \( \text{erf} \)

\[
\text{erf}(x) = \frac{2}{\sqrt{\pi}} \int_0^x e^{-t^2} dt
\]

The error function is very useful in statistics (cf. Gaussian distribution), diffusion equation (special configurations) and other heat transfer problems. . .

**Goal:** return the correctly rounded value of \( \text{erf}(x) \) in arbitrary precision.
Possible formulas

\[
\text{erf}(x) = \frac{2}{\sqrt{\pi}} \int_0^x e^{-t^2} dt
\]

\[
= \frac{2}{\sqrt{\pi}} \sum_{n=0}^{+\infty} \frac{(-1)^n x^{2n+1}}{n!(2n+1)}
\]

\[
= \frac{2}{\sqrt{\pi}} e^{-x^2} \sum_{n=0}^{+\infty} \frac{2^n x^{2n+1}}{1.3\cdots(2n+1)}
\]

\[
= \sqrt{2} \sum_{n=0}^{+\infty} (-1)^n \left[ I_{2n+1/2}(x^2) - I_{2n+3/2}(x^2) \right]
\]

\[
= e^{-x^2} \frac{1}{\sqrt{\pi}} \frac{1}{x+} \frac{1/2}{x+} \frac{1}{x+} \frac{3/2}{x+} \frac{2}{x+} \cdots
\]
Possible formulas

\[
\text{erf}(x) = \frac{2}{\sqrt{\pi}} \int_{0}^{x} e^{-t^2} dt
\]

\[
= \frac{2}{\sqrt{\pi}} \sum_{n=0}^{+\infty} \frac{(-1)^n x^{2n+1}}{n!(2n+1)}
\]

\[
= \frac{2}{\sqrt{\pi}} e^{-x^2} \sum_{n=0}^{+\infty} \frac{2^n x^{2n+1}}{1.3\ldots(2n+1)}
\]

\[
= \sqrt{2} \sum_{n=0}^{+\infty} (-1)^n \left[ I_{2n+1/2}(x^2) - I_{2n+3/2}(x^2) \right]
\]

\[
= \frac{e^{-x^2}}{\sqrt{\pi}} \frac{1}{x} \frac{1}{x} \frac{1}{x} \frac{3/2}{x} \frac{2}{x} \ldots
\]

Discussion of the use of quadrature :

• numerous evaluations of \( \exp \) : costly (either in computing time or in development time)
• many sources of error : evaluation of \( \exp \), quadrature, roundoff
Possible formulas

\[
\text{erf}(x) = \frac{2}{\sqrt{\pi}} \int_0^x e^{-t^2} dt
\]

\[
= \frac{2}{\sqrt{\pi}} \sum_{n=0}^{+\infty} \frac{(-1)^n x^{2n+1}}{n!(2n+1)}
\]

\[
= \frac{2}{\sqrt{\pi}} e^{-x^2} \sum_{n=0}^{+\infty} \frac{2^n x^{2n+1}}{1.3\cdots(2n+1)}
\]

\[
= \sqrt{2} \sum_{n=0}^{+\infty} (-1)^n \left[ I_{2n+1/2}(x^2) - I_{2n+3/2}(x^2) \right]
\]

\[
= \frac{e^{-x^2}}{\sqrt{\pi}} \frac{1}{x} \frac{1}{x} \frac{1}{x} \frac{3/2}{x} \frac{2}{x} \cdots
\]

Discussion of the use of alternate power series:

- the remainder is easy to bound
- sum of terms of alternate signs: cancellation
Possible formulas

\[ \text{erf}(x) = \frac{2}{\sqrt{\pi}} \int_0^x e^{-t^2} dt \]  

\( (1) \)

\[ = \frac{2}{\sqrt{\pi}} \sum_{n=0}^{+\infty} \frac{(-1)^n x^{2n+1}}{n!(2n+1)} \]  

\( (2) \)

\[ = \frac{2}{\sqrt{\pi}} e^{-x^2} \sum_{n=0}^{+\infty} \frac{2^n x^{2n+1}}{1.3\cdots(2n+1)} \]  

\( (3) \)

\[ = \sqrt{2} \sum_{n=0}^{+\infty} (-1)^n \left[ I_{2n+1/2}(x^2) - I_{2n+3/2}(x^2) \right] \]  

\( (4) \)

\[ = \frac{e^{-x^2}}{\sqrt{\pi}} \frac{1}{x+} \frac{1/2}{x+} \frac{1}{x+} \frac{3/2}{x+} \frac{2}{x+} \cdots \]  

\( (5) \)

Discussion of the use of this power series:

- sum of positive terms: numerical stability
- the remainder is less easy to bound
Possible formulas

\[
\text{erf}(x) = \frac{2}{\sqrt{\pi}} \int_0^x e^{-t^2} dt
\]

\[
= \frac{2}{\sqrt{\pi}} \sum_{n=0}^{+\infty} \frac{(-1)^n x^{2n+1}}{n!(2n+1)}
\]

\[
= \frac{2}{\sqrt{\pi}} e^{-x^2} \sum_{n=0}^{+\infty} \frac{2^n x^{2n+1}}{1.3\cdots(2n+1)}
\]

\[
= \sqrt{2} \sum_{n=0}^{+\infty} (-1)^n \left[ I_{2n+1/2}(x^2) - I_{2n+3/2}(x^2) \right]
\]

\[
= \frac{e^{-x^2}}{\sqrt{\pi}} \frac{1}{x} \frac{1}{x+} \frac{1}{x+} \frac{3/2}{x+} \frac{2}{x+} \cdots
\]

Discussion of the use of Bessel functions of fractional order:

- the problem is now to evaluate the Bessel functions of fractional order
Possible formulas

\[
erf(x) = \frac{2}{\sqrt{\pi}} \int_0^x e^{-t^2} dt \tag{1}
\]
\[
= \frac{2}{\sqrt{\pi}} \sum_{n=0}^{+\infty} \frac{(-1)^n x^{2n+1}}{n!(2n+1)} \tag{2}
\]
\[
= \frac{2}{\sqrt{\pi}} e^{-x^2} \sum_{n=0}^{+\infty} \frac{2^n x^{2n+1}}{1.3\cdots(2n+1)} \tag{3}
\]
\[
= \sqrt{2} \sum_{n=0}^{+\infty} (-1)^n \left[ I_{2n+1/2}(x^2) - I_{2n+3/2}(x^2) \right] \tag{4}
\]
\[
= \frac{e^{-x^2}}{\sqrt{\pi}} \frac{1}{x+} \frac{1/2}{x+} \frac{1}{x+} \frac{3/2}{x+} \frac{2}{x+} \cdots \tag{5}
\]

Discussion of the use of this continued fraction:

- the remainder is less easy to bound
- numerical stability is not ensured
Chosen formula

\[ \text{erf}(x) = \frac{2}{\sqrt{\pi}} \int_0^x e^{-t^2} dt \]  \hspace{1cm} (1)

\[ = \frac{2}{\sqrt{\pi}} \sum_{n=0}^{+\infty} \frac{(-1)^n x^{2n+1}}{n!(2n+1)} \]  \hspace{1cm} (2)

\[ = \frac{2}{\sqrt{\pi}} e^{-x^2} \sum_{n=0}^{+\infty} \frac{2^n x^{2n+1}}{1.3\ldots(2n+1)} \]  \hspace{1cm} (3)

\[ = \sqrt{2} \sum_{n=0}^{+\infty} (-1)^n \left[ I_{2n+1/2}(x^2) - I_{2n+3/2}(x^2) \right] \]  \hspace{1cm} (4)

\[ = \frac{e^{-x^2}}{\sqrt{\pi}} \frac{1}{x+} \frac{1/2}{x+} \frac{1}{x+} \frac{3/2}{x+} \frac{2}{x+} \ldots \]  \hspace{1cm} (5)

Motivation for the choice of this alternate power series:

- the remainder is easy to bound
- special care to avoid cancellation in the sum of terms of alternate signs
Other useful formulas

\[ \operatorname{erf}(-x) = -\operatorname{erf}(x) \]

No argument reduction possible (cf. \( \sin(x + 2\pi) = \sin x \) or \( \exp(2x) = (\exp x)^2 \)).

\[ \frac{2}{\sqrt{\pi}} \cdot \frac{e^{-x^2}}{x + \sqrt{x^2 + 2}} < \operatorname{erfc}(x) \leq \frac{2}{\sqrt{\pi}} \cdot \frac{e^{-x^2}}{x + \sqrt{x^2 + \frac{4}{\pi}}} \quad \text{for } x \geq 0. \]
Outline

- the error function erf
- algorithm to return the correctly rounded evaluation of \( \text{erf}(x) \)
- experimental results
- the complementary error function \( \text{erfc} \)
- conclusion and hints for improvements
- current work on interval arithmetic and algorithms: MPFI
Algorithm

Input : $x, p$
Output : correctly rounded value of erf($x$) with $p$ bits

1. handle special cases : $x = 0$, $x = \pm \infty$, $x < 0$
2. check whether the last enclosure gives rapidly the answer
3. determine the truncation rank $N$ needed to have an absolute error $\leq \varepsilon$
4. determine the computing precision $q$ to have roundoff error $\leq \varepsilon$
5. evaluate $y$ that approximates erf($x$) using the alternate power series ; bound from above the roundoff error $\varepsilon' \leq \varepsilon$, on the fly
6. if `can_round($y, \varepsilon' + \varepsilon', p$)` then
   return $y$
   else increase $N$ and $q$ and do steps (5) and (6) again
Algorithm: step (2)

Input: $x, p$
Output: correctly rounded value of $\text{erf}(x)$ with $p$ bits

1. handle special cases: $x = 0$, $x = \pm\infty$, $x < 0$
2. check whether the last enclosure gives rapidly the answer
3. determine the truncation rank $N$ needed to have an absolute error $\leq \varepsilon$
4. determine the computing precision $q$ to have roundoff error $\leq \varepsilon$
5. evaluate $y$ that approximates $\text{erf}(x)$ using the alternate power series; bound from above the roundoff error $\varepsilon' \leq \varepsilon$, on the fly
6. if can_round($y, \varepsilon + \varepsilon', p$) then
   return $y$
   else increase $N$ and $q$ and do steps (5) and (6) again
Algorithm : step (2)

Reminder :

\[
\frac{2}{\sqrt{\pi}} \cdot \frac{e^{-x^2}}{x + \sqrt{x^2 + 2}} < \text{erfc}(x) \leq \frac{2}{\sqrt{\pi}} \cdot \frac{e^{-x^2}}{x + \sqrt{x^2 + \frac{4}{\pi}}} \quad \text{for } x \geq 0.
\]

Step (2) :
compute both sides of this enclosure : \(y_L, y_R\)
if \text{can\_round}(y_L, y_R - y_L, p) then return it.
Algorithm: step (3)

Input: $x, p$
Output: correctly rounded value of $\text{erf}(x)$ with $p$ bits

1. handle special cases: $x = 0$, $x = \pm \infty$, $x < 0$
2. check whether the last enclosure gives rapidly the answer
3. determine the truncation rank $N$ needed to have an absolute error $\leq \varepsilon$
4. determine the computing precision $q$ to have roundoff error $\leq \varepsilon$
5. evaluate $y$ that approximates $\text{erf}(x)$ using the alternate power series; bound from above the roundoff error $\varepsilon' \leq \varepsilon$, on the fly
6. if $\text{can\_round}(y, \varepsilon + \varepsilon', p)$ then
   return $y$
   else increase $N$ and $q$ and do steps (5) and (6) again
Algorithm : step (3)

Reminder : power series

\[
\text{erf}(x) = \frac{2}{\sqrt{\pi}} \sum_{n=0}^{+\infty} \frac{(-1)^n x^{2n+1}}{n!(2n+1)} = \sum_{n=0}^{+\infty} a_n \text{ with } a_n = \frac{2}{\sqrt{\pi}} \frac{(-1)^n x^{2n+1}}{n!(2n+1)}
\]

Property : alternate power series \( \sum_{n=0}^{+\infty} a_n \) with non-increasing term \( a_n \) (for \( n \) large enough)
\[ \Rightarrow \text{remainder } | \sum_{n=N}^{+\infty} a_n | = | \text{erf}(x) - \sum_{n=0}^{N-1} a_n | \leq |a_N| . \]

Step (3) :
\[ \varepsilon = 2^{-p-8} : 8 \text{ extra bits} \]
evaluate \( a_n \) until \( a_N \leq \varepsilon : N \) is the truncation rank.
Algorithm : step (4)

Input : $x, p$
Output : correctly rounded value of $\text{erf}(x)$ with $p$ bits

1. handle special cases : $x = 0$, $x = \pm\infty$, $x < 0$
2. check whether the last enclosure gives rapidly the answer
3. determine the truncation rank $N$ needed to have an absolute error $\leq \varepsilon$
4. determine the computing precision $q$ to have roundoff error $\leq \varepsilon$
5. evaluate $y$ that approximates $\text{erf}(x)$ using the alternate power series; bound from above the roundoff error $\varepsilon' \leq \varepsilon$, on the fly
6. if can_round($y, \varepsilon + \varepsilon', p$) then
   return $y$
   else increase $N$ and $q$ and do steps (5) and (6) again
Algorithm : step (4)

Goal : computing precision $q$ such that roundoff error $\leq \varepsilon$.


Step (4) :

$$q = 1 + \log_2 \left( \frac{5N+1}{2\alpha} \right) \text{ where } \alpha = \min \left( \frac{1}{2}, \frac{\varepsilon x}{e^{x^2} - 1 + \frac{\varepsilon x}{2}} \right).$$
Algorithm: step (5)

Input: \( x, p \)
Output: correctly rounded value of \( \text{erf}(x) \) with \( p \) bits

1. handle special cases: \( x = 0, x = \pm \infty, x < 0 \)
2. check whether the last enclosure gives rapidly the answer
3. determine the truncation rank \( N \) needed to have an absolute error \( \leq \varepsilon \)
4. determine the computing precision \( q \) to have roundoff error \( \leq \varepsilon \)
5. evaluate \( y \) that approximates \( \text{erf}(x) \) using the alternate power series; bound from above the roundoff error \( \varepsilon' \leq \varepsilon \), on the fly
6. if \( \text{can\_round}(y, \varepsilon + \varepsilon', p) \) then
   return \( y \)
   else increase \( N \) and \( q \) and do steps (5) and (6) again
Algorithm : step (5)

Power series :

\[ \text{erf}(x) = \frac{2}{\sqrt{\pi}} \sum_{n=0}^{+\infty} \frac{(-1)^n x^{2n+1}}{n!(2n + 1)} \]

Problem : cancellation \(\Rightarrow\) numerical unstability.

Solution : group the terms by pair (assume \(N\) is odd) :

\[ \text{erf}(x) = \sum_{n=0}^{N-1} a_n = a_{N-1} + \frac{2x}{\sqrt{\pi}} \sum_{n=0}^{\frac{N-1}{2}} \frac{x^{4n}}{(2n)!} \left( \frac{1}{4n + 1} - \frac{x^2}{(2n+1)(4n+3)} \right) \]

Step (5) :
sum using Horner-like scheme.
Algorithm: remaining issues

Increase $N$ and $q$:

cf. Kreinovich and Rump 2006: optimal overhead $= 4$ if the computing time is doubled

$N_{i+1} \simeq (2 - \alpha_i) N_i$ with $\alpha_i = 2/(1 + q_i)$,

$q_{i+1}$ depends on $N_{i+1}$.

Termination:

if there exists a floating-point number $x$ such that $\text{erf}(x)$ is a floating-point number, then can_round can never answer "yes".

In other words, infinite loop.
Outline

• the error function erf
• algorithm to return the correctly rounded evaluation of erf($x$)
• experimental results
• the complementary error function erfc
• conclusion and hints for improvements
• current work on interval arithmetic and algorithms: MPFI
Experimental results: small values of $x$

<table>
<thead>
<tr>
<th>$x$</th>
<th>$p$</th>
<th>mpfr_erf</th>
<th>Maple 6</th>
<th>my_erf</th>
</tr>
</thead>
<tbody>
<tr>
<td>0.25</td>
<td>100</td>
<td>0.21 ms</td>
<td>&lt; 0.10 ms</td>
<td>0.53 ms</td>
</tr>
<tr>
<td>0.25</td>
<td>1000</td>
<td>7.51 ms</td>
<td>20 ms</td>
<td>3.28 ms</td>
</tr>
<tr>
<td>0.25</td>
<td>10000</td>
<td>1020 ms</td>
<td>580 ms</td>
<td>365 ms</td>
</tr>
</tbody>
</table>

(2003: on a Pentium II 400MHz, 64MB RAM)

Comments:
for small precisions, our pre-computations (determination of the truncation rank and of the computing precision) is costly; this cost is compensated by the gain it provides, for larger precisions.
Experimental results: intermediate values of $x$

<table>
<thead>
<tr>
<th>$x$</th>
<th>$p$</th>
<th>mpfr_erf</th>
<th>Maple 6</th>
<th>my_erf</th>
</tr>
</thead>
<tbody>
<tr>
<td>$\pi$</td>
<td>100</td>
<td>0.73 ms</td>
<td>&lt; 0.10 ms</td>
<td>1.33 ms</td>
</tr>
<tr>
<td>$\pi$</td>
<td>1000</td>
<td>19.3 ms</td>
<td>60 ms</td>
<td>8.7 ms</td>
</tr>
<tr>
<td>$\pi$</td>
<td>10000</td>
<td>2040 ms</td>
<td>7320 ms</td>
<td>560 ms</td>
</tr>
<tr>
<td>$\pi$</td>
<td>100000</td>
<td>340.7 s</td>
<td>1692 s</td>
<td>149.6 s</td>
</tr>
</tbody>
</table>

(times in seconds, Pentium II 400MHz, 64MB RAM)

Comments: ibidem

for small precisions, our pre-computations (determination of the truncation rank and of the computing precision) is costly; this cost is compensated by the gain it provides, for larger precisions.
Experimental results: large values of $x$

<table>
<thead>
<tr>
<th>$x$</th>
<th>$p$</th>
<th>mpfr_erf</th>
<th>Maple 6</th>
<th>my_erf</th>
</tr>
</thead>
<tbody>
<tr>
<td>100</td>
<td>1 000</td>
<td>0.0022 ms</td>
<td>&lt; 0.10 ms</td>
<td>0.22 ms</td>
</tr>
<tr>
<td>100</td>
<td>10 000</td>
<td>0.0065 ms</td>
<td>&lt; 0.10 ms</td>
<td>0.22 ms</td>
</tr>
<tr>
<td>100</td>
<td>14 446</td>
<td>191 200 ms</td>
<td>0.20 ms</td>
<td>0.22 ms</td>
</tr>
<tr>
<td>100</td>
<td>15 000</td>
<td>196 100 ms</td>
<td>0.40 ms</td>
<td>75 600 ms</td>
</tr>
</tbody>
</table>

(2003: on a Pentium II 400MHz, 64MB RAM)

Comments:
erf($x$) is very close to 1;
here, Maple computes erfc($x$) and then erf($x$) = 1 − erfc($x$).
Outline

• the error function erf
• algorithm to return the correctly rounded evaluation of erf($x$)
• experimental results
• the complementary error function erfc
• conclusion and hints for improvements
• current work on interval arithmetic and algorithms : MPFI
The complementary error function erfc

\[ \text{erfc}(x) = 1 - \text{erf}(x) = \frac{2}{\sqrt{\pi}} \int_{x}^{+\infty} e^{-t^2} dt \]

Why not use \( \text{erf} \) for large \( x \)?
because this would require to compute a large number of bits of \( \text{erf}(x) \) to cancel them

\[ 1 - \text{erf}(x) = 1.000 \cdots - 0.111 \cdots 10b_1 b_2 \cdots b_p = 0.000 \cdots 01 \cdots \]
Possible formulas

\[ \text{erfc}(x) = \frac{2}{\sqrt{\pi}} \int_x^{+\infty} e^{-t^2} dt \]  
\[ = 1 - \frac{2}{\sqrt{\pi}} \sum_{n=0}^{+\infty} \frac{(-1)^n x^{2n+1}}{n!(2n+1)} \]  
\[ = 1 - \frac{2}{\sqrt{\pi}} e^{-x^2} \sum_{n=0}^{+\infty} \frac{2^n x^{2n+1}}{1.3\cdots(2n+1)} \]  
\[ = \frac{e^{-x^2}}{x\sqrt{\pi}} \left( 1 + \sum_{n=1}^{+\infty} (-1)^n n \frac{1.3\cdots(2n-1)}{(2x^2)^n} \right) \]  
\[ = \frac{e^{-x^2}}{\sqrt{\pi}} \frac{1}{x + \frac{1/2}{x + \frac{2/2}{x + \frac{3/2}{x + \cdots}}}} \]  

Chosen formula: the continued fraction (5).
Algorithm

Input : $x, p$
Output : correctly rounded value of $\text{erfc}(x)$ with $p$ bits

1. handle special cases : $x = 0$, $x = \pm \infty$, $x < 0$
2. determine the truncation rank $N$ needed to have an absolute error $\leq \varepsilon$
3. determine the computing precision $q$ to have roundoff error $\leq \varepsilon$
4. evaluate $y$ that approximates $\text{erfc}(x)$ using the continued fraction;
   (two methods : compute the fraction or compute separately the numerator and the denominator)
5. if $\text{can\_round}(y, 2\varepsilon, p)$ then
   return $y$
   else increase $N$ and $q$ and do steps (5) and (6) again
Outline

• the error function erf
• algorithm to return the correctly rounded evaluation of erf($x$)
• experimental results
• the complementary error function erfc
• conclusion and hints for improvements
• current work on interval arithmetic and algorithms : MPFI
Conclusion and future work

Realization:
efficient implementation of correctly rounded error functions erf and erfc.
Termination? Table’s Maker Dilemma.

Future work:
• small precisions: avoid costly determination of the truncation rank
• choose between erf and erfc depending on the value of $x$, to avoid cancellation
• when can_round fails, change the heuristic to increase $N$ to reach an optimal overhead factor of 4
• exceptions (under/overflow) not totally taken care of yet
Current work on interval arithmetic and algorithms: MPFI

**MPFI (Multiple Precision Floating-point Interval arithmetic library):**
- development of this C library, based on MPFR.
- design of MPFI vs standardization of intervals in C++ (proposal)

**Algorithms:**
- automatic adaptation of the computing precision
- Newton method for univariate problems
- linear recurrences
- global optimization for univariate problems (approximations of an elementary function by a polynomial)
Bibliography for the evaluation of erf and erfc