Implementing the exponential function in C

T.T. Lindkvist

1 Making the computer understand

As we know the exponential function `exp(x)`, is simply Euler's number, $e \approx 2.71828$, raised to the power of x - `exp(x)` $= e^x$. However, computers really only know the simple operators +, -, * and /. Using the Taylor series for the exponential function, we can easily implement it

$$e^x = \sum_{n=0}^{\infty} \frac{x^n}{n!} = 1 + x + \frac{x^2}{2} + \frac{x^3}{6} + \cdots \tag{1}$$

Of course we cannot do an infinite sum, but luckily this converges fast for small x. So we take the first $N = 10$ terms. This leaves a problem for larger x's, where the approximation is not so good. To combat this, we can use a rule of exponents

$$e^x = e^{x/2} \cdot e^{x/2} \, . \tag{2}$$

Suppose we want to compute `exp(x)` for some x larger than some x_{max}, where we know precision becomes an issue, we simply apply this rule of exponents recursively until we can get by, by calculating `exp(x')` for some $x' < x_{max}$. We can also make use of another property of exponents

$$e^{-x} = \frac{1}{e^x} \, , \tag{3}$$

and hereby only need to evaluate the function for positive argument x - making all terms in the series positive. Summing terms of alternating sign can make the calculation lose precision, since the terms are of comparable order of magnitude. Summing large to small terms lose precision. Avoid many multiplication by making a convoluted expression.

1.1 Pseudocode

2 C Implementation

3 Comparison of implementations