Error Propagation and Roundoff Error

In general our problem has a certain number of input values $x_1, \ldots, x_n$ and a certain number of output values $y_1, \ldots, y_m$, and some (possibly complicated) formulas describe how the output values depend on the input values. Let us consider the simplest case of one input and one output value where we have $y = f(x)$.

Error propagation

If we only have an approximation $\tilde{x}$ of our input value x available (e.g., because of measurement errors), the best thing we can do is to compute $\tilde{y} := f(\tilde{x})$. For the resulting relative error we obtain

$$\varepsilon_y := \frac{\tilde{y} - y}{y} = \frac{f(\tilde{x}) - f(x)}{f(x)} \approx \frac{(\tilde{x} - x)f'(x)}{f(x)} = \frac{xf'(x)}{f(x)} \frac{\tilde{x} - x}{x} = c_f \varepsilon_x$$

where the magnification factor $c_f := \frac{xf'(x)}{f(x)}$ is called the **condition number** of the function f at x. The condition number determines how sensitive a problem is to small perturbations of input values. If $|c_f|$ is not much larger than 1 we call the problem **well-conditioned**, in the case of $|c_f| \gg 1$ we call the problem **ill-conditioned**. For example, the function $f(x) = \frac{1}{x}$ has the condition number $c_f = \frac{x(-x^{-2})}{x^{-1}} = -1$ and is therefore well conditioned.

Error propagation for arithmetic operations

If we multiply two numbers to compute $z := xy$ we get for perturbed input values $\tilde{x}, \tilde{y}$ the result $\tilde{z} := \tilde{x}\tilde{y} = x(1 + \varepsilon_x)y(1 + \varepsilon_y) = z(1 + \varepsilon_x + \varepsilon_y + \varepsilon_x\varepsilon_y)$. Hence we have $\varepsilon_z = \varepsilon_x + \varepsilon_y + \varepsilon_x\varepsilon_y \approx \varepsilon_x + \varepsilon_y$.

For the quotient $z = x/y$ we obtain $\varepsilon_z \approx \varepsilon_x + \varepsilon_{1/y} \approx \varepsilon_x - \varepsilon_y$ using the above result for $f(x) = 1/x$.

For the addition $z = x + y$ we have $\varepsilon_z = \frac{x(1+\varepsilon_x)+y(1+\varepsilon_y)-(x+y)}{x+y} = \frac{x}{x+y}\varepsilon_x + \frac{y}{x+y}\varepsilon_y$. This also covers subtraction since x, y can have arbitrary signs. However, for the case that x and y have the same sign we observe that $|\varepsilon_z| \leq \max\{|\varepsilon_x|, |\varepsilon_y|\}$.

Machine numbers and algorithms in machine arithmetic

Instead of arbitrary real numbers in $\mathbb{R}$ we only have a finitely many machine numbers available. As long as $x_{min} \leq |x| \leq x_{max}$ (i.e., no underflow or overflow) we can approximate a real number x with a machine number $fl(x)$ such that $\left|\frac{fl(x)-x}{x}\right| \leq \varepsilon_M$. Here ε_M denotes the so-called machine accuracy (aka unit roundoff).

An algorithm in machine arithmetic is a function $\hat{y} = \hat{f}(\hat{x})$ which takes an input machine number $\hat{x}$, performs certain operations in machine arithmetic and finally gives an output machine number $\hat{y}$. If we want to compute $y = f(x)$ for some $x \in \mathbb{R}$ we would use this algorithm and compute $\hat{y} = \hat{f}(\hat{x})$ with $\hat{x} = fl(x)$. What is the best accuracy $\left|\frac{\hat{y}-y}{y}\right|$ we can hope to achieve?

The **ideal algorithm** would compute $\tilde{y} := f(\hat{x})$ exactly (or at least with lots of extra precicion), and then approximate the result $\tilde{y}$ by the closest machine number $\hat{y} := fl(\tilde{y})$, i.e., we would use $\hat{f}(\hat{x}) := fl(f(\hat{x}))$. If we compare this with $y = f(x)$ we obtain $\left|\frac{\hat{x}-x}{x}\right| \leq \varepsilon_M$, $\left|\frac{\tilde{y}-y}{y}\right| \leq |c_f|\varepsilon_M$ and $\left|\frac{\hat{y}-y}{y}\right| \leq |c_f|\varepsilon_M + \varepsilon_M$. This expression is called the **unavoidable error**. As the algorithm $\hat{f}$ uses machine numbers for input and output we must accept a relative error of size ε_M for both the input and output values, and this means that the relative error in the result can be as high as $|c_f|\varepsilon_M + \varepsilon_M$. Hence the ideal algorithm

would achieve for well-conditioned functions an error of not much more than ε_M, and for ill-conditioned functions we would obtain an error of order $|c_f|\varepsilon_M$.

The ideal algorithm is usually impossible to implement or too costly (but note that IEEE 754 requires that the elementary operations $+, -, \cdot, /, \sqrt{}$ are implemented in that way). However, we can expect that the actual implementation performs not much worse than the ideal algorithm: We call an algorithm **numerically stable** if it yields in machine arithmetic a result $\hat{y} = \hat{f}(\hat{x})$ such that $\left|\frac{\hat{y}-y}{y}\right| \leq C(|c_f|\varepsilon_M + \varepsilon_M)$ where C is not much larger than 1 (say, not larger than 10).

One way to show that an algorithm is numerically stable is called forward error analysis.

Forward Error Analysis

We try to find upper bounds for the absolute values of the relative error at each stage of the algorithm, moving forward through the algorithm. We start with bounds for the errors in the given data. When a function f is applied, we multiply the error bound by the condition number $|c_f|$. When two values are added, subtracted, multiplied, divided we use the above formulas for error propagation. Each time a result is rounded we add $|\varepsilon_M|$ to the error bound.

Example: Consider $y = f(x) := 1 - \cos x$ for $x = 10^{-5}$ and double precision machine numbers. We find that $c_f = \frac{x \sin x}{1-\cos x} \approx \frac{x \cdot x}{x^2/2} = 2$, hence the function is well-conditioned and the unavoidable error is $|c_f|\varepsilon_M + \varepsilon_M \approx 3 \cdot 10^{-16}$.

Consider the **first algorithm**

$$y_1 := \cos x, \quad y := 1 - y_1$$

Evaluating this in machine arithmetic gives $\hat{x} := fl(x)$, $\tilde{y}_1 := \cos \hat{x}$, $\hat{y}_1 := fl(\tilde{y}_1)$, $\tilde{y} := 1 - \hat{y}_1$, $\hat{y} := fl(\tilde{y})$. For the relative errors we obtain $\left|\frac{\hat{x}-x}{x}\right| \leq \varepsilon_M$, $\left|\frac{\tilde{y}_1-y_1}{y_1}\right| \leq c_1\varepsilon_M$ with $c_1 = \left|\frac{x(-\sin x)}{\cos x}\right| \approx 10^{-10}$, $\left|\frac{\hat{y}_1-y_1}{y_1}\right| \leq c_1\varepsilon_M + \varepsilon_M$, $\left|\frac{\tilde{y}-y}{y}\right| \leq c_2\,(c_1\varepsilon_M + \varepsilon_M)$ with $c_2 = \left|\frac{y_1(-1)}{1-y_1}\right| \approx 2 \cdot 10^{10}$ and finally

$$\left|\frac{\hat{y}-y}{y}\right| \leq c_2\,(c_1\varepsilon_M + \varepsilon_M) + \varepsilon_M \approx 2\varepsilon_M + 2 \cdot 10^{10}\varepsilon_M + \varepsilon_M \approx 2 \cdot 10^{-6}$$

which is much larger than the unavoidable error. This algorithm is numerically unstable.

To find a better algorithm we can use that $1 - \cos x = 1 - \cos(\frac{x}{2} + \frac{x}{2}) = 1 - \cos(\frac{x}{2})^2 + \sin(\frac{x}{2})^2 = 2\sin(\frac{x}{2})^2$. This yields the **second algorithm**

$$y_1 := x/2, \quad y_2 := \sin y_1, \quad y_3 := y_2^2, \quad y_4 := 2y_3$$

Note that multiplication by 2 and division by 2 is exact in machine arithmetic, so the first and last step introduce no roundoff error. We only have to find the condition numbers $c_1 = \frac{y_1 \cos y_1}{y_1} \approx 1$ and $c_2 = \frac{y_2 2 y_2}{y_2^2} = 2$ for the second and third steps and obtain in the same way as above $\left|\frac{\hat{y}-y}{y}\right| \leq c_2\,(c_1\varepsilon_M + \varepsilon_M) + \varepsilon_M \approx 2\varepsilon_M + 2\varepsilon_M + \varepsilon_M \approx 5 \cdot 10^{-16}$. This is not much more than the unavoidable error, and this algorithm is numerically stable.