Lecture 0 - floating point arithmetic, sources of error

Dr. Dmitry Batenkov

March 7, 2021

1 Introduction

The subject of numerical analysis deals with inexact computations (usually done on a digital computer). The computations we are interested in usually approximate well-defined mathematical operations such as:

- Evaluation of transcendental functions: $\sin x$, $\int_a^x \frac{\sin t}{t} dt$, ...
- Solution of equations: linear ($Ax = b$), nonlinear ($f(x) = 0$), differential ($y'(t) = f(t, y)$)
- Integration/quadrature: compute $\int_a^b f(x)dx$
- Finding eigenvalues / eigenvectors of matrices
- Optimization problems: $\min_x f(x)$

In the vast majority of cases, these operations cannot be solved analytically (contrary to what you learn in high school). In order to get an answer, one needs to employ a *numerical method. We will learn some of these in the course.*

An inherent part of the subject is *error analysis*. Indeed, when we get an answer to a problem, we would like to know how far/close we are from the true (and unknown) solution to our problem. In order to do this, we need first to understand what are the possible sources of errors in computations.

1. Physical (inexact measurements, say of the vector b in $Ax = b$);
2. Modeling (the function $f(x)$ is not known exactly);
3. Algorithmic (the method converges slowly, errors are amplified during computation - stability);
4. **Rounding and finite precision computations:** inexact representation of numbers by a computer, even rational numbers such as $1/3 = (0.33(3))_{10}$ or $1.2 = (1.0011(0011))_2$.

2 Floating point arithmetic

The main goal of floating point arithmetic is to construct a system of representation of a wide range of numbers by a fixed amount of information (bits). The problem is that any such system will necessarily incur a rounding error, if we wish to represent, say, all real numbers in a certain range.

How would you represent a number such as $x = -32.14$?

$$-32.14 = (-1) \times \left(0.3214 \times 10^2\right) = (-1) \times 3214 \times 10^{-2} = (-1) \times \frac{3214}{10^4} \times 10^2$$

We fix a base β. In modern computers $\beta = 2$ but in these notes we will use for simplicity $\beta = 10$. Every floating point number is represented as follows:

$$[\sigma] \times \mathbf{0.a_1 a_2 \dots a_t} \times \beta^{\mathbf{p}} = [\sigma] \times \left\{ \frac{a_1 \dots a_t}{\beta^t} \right\} \times \beta^p$$

The elements in the representation **which need to be stored** are:

1. $\sigma = \pm 1$: the sign
2. $a_1 a_2 \dots a_t$: the mantissa. Every a_i is an integer between 0 and $\beta - 1$, and $a_1 \neq 0$.
3. p : the exponent. It is an integer between $L < p < U$

Because $a_1 \neq 0$ the representation is canonical, or normalized - so that every floating point number is uniquely represented.

Example 1. $123400 = 0.1234 \times 10^6$, $0.0001234 = 0.1234 \times 10^{-3}$ (here $\beta = 10$).

Remark 1. Depending on the computer architecture, the values of β, t, L, U. Most commonly used standard, called the **IEEE 754 double precision,** defines these values (overall 64 bits):

- $\beta = 2$ (binary) - not stored
- $t = 52$ (52 bit mantissa)
- $L = -1023, U = 1024$ (11 bit signed exponent).

In fact, the IEEE 754 representation is slightly different (pay attention to the leading 1 which is implied and not stored explicitly):

$$[\sigma] \times \mathbf{1.a_1 a_2 \dots a_t} \times \beta^{\mathbf{p}}$$

Some consequences:

- The largest representable number: $1.\underbrace{11\dots1}_{\times 52} \times 2^{1023} \approx 10^{307}$
- Closest to zero positive representable number: $1.\underbrace{00\dots0}_{\times 52} \times 2^{-1022} \approx 10^{-308}$

3 Errors

From now on we assume $\beta = 10$.

Given a real number $x \in \mathbb{R}$, we can always write

$$x = \lim_{t\to\infty} \text{sign}(x) \times \beta^{\text{exponent}(x)} \times \frac{\text{mantissa}_t(x)}{\beta^t}, \text{or}$$
$$x = \sigma.(a_1 \dots a_t a_{t+1} \dots)_\beta \times \beta^p$$

Question: how to get from x to its floating point representation $\text{fl}(x)$, and what is the error?

1. **Chopping:** choosing a finite $t$and disregard all a_k for $k \geq t+1$: $\text{chop}(x) = \sigma.(a_1 \dots a_t)_\beta \times \beta^p$
   The **absolute chopping** error:

$$\text{chopping error}(x) = |x - \text{chop}(x)| = 0.\left(\underbrace{00\dots}_{\times t} a_{t+1} \dots\right)_\beta \times \beta^p$$

   This can be very large depending on p. Therefore it is more important to control the **relative error**:

$$\left|\frac{x - \text{chop}(x)}{x}\right| \leq$$
$$\dots$$
$$\leq \beta^{1-t}$$

   And this is fixed (and small!) for every x.

2. **Roundoff/rounding:** as learned in high school! This is what is (almost) used in practice in modern computers.

$$\text{fl}(x) = \begin{cases} \sigma.(a_1 \dots a_t)_\beta \times \beta^p & \text{if } a_{t+1} < \frac{\beta}{2}; \\ \sigma\left[.(a_1 \dots a_t)_\beta + .\left(\underbrace{0\dots01}_{\times t}\right)_\beta\right] \times \beta^p & \text{if } \frac{\beta}{2} \leq a_{t+1} < \beta. \end{cases}$$

   Exercise: check that

$$\left|\frac{x - \text{fl}(x)}{x}\right| \leq \frac{1}{2}\beta^{1-t}.$$

Remark 2. The number β^{1-t} is called the *machine epsilon*. In IEEE 754 this is $\epsilon_M = 2^{-52} \approx 2 \times 10^{-16}$. (notice the hidden bit convention here!)

4 Error propagation

Suppose we have $x, y \in \mathbb{R}$ and their approximations $\tilde{x}, \tilde{y}$ (for example $\tilde{x} = \mathrm{fl}(x)$, but not necessarily), so that $\tilde{x} - x = \Delta x = e(x)$, $\tilde{y} - y = \Delta y = e(y)$. Denote: $\delta(x) = \frac{e(x)}{x}$ and $\frac{e(y)}{y}$ - the relative errors.

In a long computation, we might have several (in fact MANY) arithmetic operations such as $+, -, /, *$. So instead of computing $x \circ y$ we actually have $\widetilde{x \circ y} = \tilde{x} \circ \tilde{y}$. What is the relative and absolute error?

1. Addition: $|\delta(x+y)| = \left|\frac{\tilde{x}+\tilde{y}-(x+y)}{x+y}\right|$. If x, y have same sign then
$$|\delta(x+y)| \leq |\delta(x)| + |\delta(y)|$$
2. Subtraction: will see in tutorial that $|\delta(x-y)| \leq \left|\frac{e(x)}{x-y}\right| + \left|\frac{e(y)}{x-y}\right|$. If $x \approx y$ we have **loss of relative accuracy**, which is an undesirable thing.
3. Multiplication: $|\delta(xy)| = \left|\frac{(x+e(x))(y+e(y))-xy}{xy}\right| \leq \cdots \leq |\delta(x)| + |\delta(y)| + |\delta(x)\delta(y)| \approx |\delta(x)| + |\delta(y)|$.
4. Division: show that $|\delta(x/y)| \leq \left|\frac{y}{\tilde{y}}\right| (|\delta(x)| + |\delta(y)|)$. What happens when $y, \tilde{y} \approx 0$? Can we lose accuracy?

Now if we want to compute a function $f(x_1, \ldots, x_n)$ of n variables $x_1, \ldots, x_n$ where $e(x_i) = \Delta_i$, we can employ the mean value theorem to get

$$|\Delta f| = |f(\tilde{x}_1, \ldots, \tilde{x}_n) - f(x_1, \ldots, x_n)| \leq \sum_{i=1}^{n} \left|\frac{\partial f}{\partial x_i}(\hat{x}_1, \hat{x}_2, \ldots, \hat{x}_n)\right| |\Delta_i| \leq \sum_{i=1}^{n} M_i |\Delta_i|$$

where $\left|\frac{\partial f}{\partial x_i}(\hat{x}_1, \hat{x}_2, \ldots, \hat{x}_n)\right| \leq M_i$. Try to obtain from this bound the expressions for the absolute error of addition (i.e. $f(x, y) = x + y$) and relative error of multiplication

Remark 3. In IEEE 754, it is guaranteed that for every operation $\circ \in \{+, -, /, *\}$ and every two floating point numbers x, y (i.e. $\mathrm{fl}(x) = x$ and $\mathrm{fl}(y) = y$) we have

$$\left|\frac{\mathrm{fl}(x \circ y) - x \circ y}{x \circ y}\right| \leq \epsilon_M.$$

5 Loss of accuracy

Example 2. Subtraction of two close numbers can result in a large relative error and therefore loss of accuracy: $x = 0.11124, y = 0.11111$ and $\tilde{x} = 0.1112, \tilde{y} = 0.1111$. We get $\tilde{x} - \tilde{y} = 10^{-4}$ and $x - y = 1.3 \times 10^{-4}$, therefore $\delta(x-y) > 23\%$.

Example 3. Inefficient calculations: analytically $y = x + y - x$, but if x, y have different scales the right hand side will not be equal to y. For instance $x = 0.1 \times 10^{18}$ and $y = 0.1 \times 10^{1}$ with $t = 16$ then $\mathrm{fl}(x+y) = x$ and $\mathrm{fl}(x+y-x) = 0$. **So, on a computer, it is not necessarily true that** $a + b + c = a + c + b$**!**

How can we prevent loss of accuracy?

Example 4. Sometimes just rewrite the expression:

$$\sqrt{x+1} - \sqrt{x} = \frac{\left(\sqrt{x+1} - \sqrt{x}\right)\left(\sqrt{x+1} + \sqrt{x}\right)}{\sqrt{x+1} + \sqrt{x}} = \frac{1}{\sqrt{x+1} + \sqrt{x}}.$$