1 Introduction: Errors and where they come from

This course is called “Introduction to Numerical Analysis”. For most real-life problems we cannot find a simple answer using pencil and paper. We want to use a computer to find an approximate answer. The central question is: how large is the error, and how does it depend on the amount of work the computer does.

1.1 Exact symbolic results vs. approximate numerical results

For simple problems (such as problems in calculus text books as opposed to real-life problems) one can find a formula for the result, e.g.,

$$I_1 = \int_0^1 \sin(x)dx = -\cos(1) + \cos(0) = 1 - \cos(1)$$

In many applications we would like to get a numerical answer with a certain number of digits. We can evaluate $1-\cos(1)$ with a calculator or Matlab and obtain $I_1 \approx .45969769413186$.

Now let us consider a slightly more complicated problem: We want to find

$$I_2 = \int_0^1 \sin(\sin(x))dx.$$

We cannot proceed as above since the antiderivative of $\sin(\sin(x))$ cannot be expressed using finitely many “elementary functions” like $\sin(x), \ln(x), \ldots$. We do know from calculus that the integral of a continuous function like $\sin(\sin(x))$ exists, so there should be a way to find a numerical value. Actually, we know from calculus that the Riemann sums of rectangles converge to the exact integral. Therefore we can divide the interval $[0,1]$ into e.g. $n = 1000$ subintervals of size $h = \frac{1}{n}$, and use on each subinterval the function value in the midpoint. This gives an approximate value

$$\hat{I}_2 := h \sum_{j=1}^{1000} \sin(\sin((j-\frac{1}{2})h)) \approx 0.430606129785715$$

where I used Matlab to compute the result. This is an example of a numerical algorithm.We would like to know: How large is the error of $\hat{I}_2$ compared to I_2 ? How fast does the error decrease if we increase the number n of subintervals? Are there more efficient methods which give smaller errors with fewer operations?

1.2 How to measure errors

Assume that the exact value is x , and that we have an approximation $\hat{x}$. Then we call $\hat{x} - x$ the **absolute error**, and we would like $|\hat{x} - x|$ to be small. E.g., for $x = 0.02$ and $\hat{x} = 0.03$ we have an absolute error of 0.01 .

In many applixations an absolute error of, say, 0.01 is more relevant in the case $x = 0.02$ than in the case $x = 200000$. Therefore one is interested in the **relative error**

$$\varepsilon_x = \frac{\hat{x} - x}{x} \qquad \Longleftrightarrow \qquad \hat{x} = x(1 + \varepsilon_x).$$

E.g., for $x = 0.02$ and $\hat{x} = 0.03$ we have $\varepsilon_x = 0.5 = 50\%$. Note that the relative error may be positive or negative. We would like $|\varepsilon_x|$ to be small.

Note that for $x = 0$ the relative error is not defined, and we can only use the absolute error.

What happens if we combine two relative errors? Assume that x is the exact value, $\hat{x}$ has a relative error of ε_1 with respect to x , and $\tilde{x}$ has a relative error of ε_2 with respect to $\hat{x}$. What is the relative error of $\tilde{x}$ with respect to x ? We have $\hat{x} = x(1 + \varepsilon_1)$ and

$$\tilde{x} = \hat{x}(1 + \varepsilon_2) = x(1 + \varepsilon_1)(1 + \varepsilon_2) = x(1 + \varepsilon_1 + \varepsilon_2 + \varepsilon_1\varepsilon_2),$$

hence $\tilde{x} = x(1 + \varepsilon)$ with $\varepsilon = \varepsilon_1 + \varepsilon_2 + \varepsilon_1\varepsilon_2 \approx \varepsilon_1 + \varepsilon_2$ for small values of $|\varepsilon_1|, |\varepsilon_2|$.

Often it is easier to consider $\frac{\hat{x}-x}{\hat{x}}$ instead of $\frac{\hat{x}-x}{x}$. Since $\hat{x} = x + (\hat{x}-x)$ we have $|\hat{x}| \le |x| + |\hat{x}-x|$. Hence $\left|\frac{\hat{x}-x}{\hat{x}}\right| \le \delta$ implies

$$\begin{aligned} |\hat{x}-x| &\le \delta\,|\hat{x}| \le \delta\,(|x| + |\hat{x}-x|) \\ (1-\delta)\,|\hat{x}-x| &\le \delta\,|x| \end{aligned}$$

If $\delta < 1$ we have $1-\delta > 0$ and hence

$$\left|\frac{\hat{x}-x}{\hat{x}}\right| \le \delta \qquad \Longrightarrow \qquad \left|\frac{\hat{x}-x}{x}\right| \le \frac{\delta}{1-\delta} \tag{1}$$

If δ is small using the Taylor series shows that $\frac{\delta}{1-\delta} \approx \delta + \delta^2$, hence $\frac{\hat{x}-x}{\hat{x}}$ and $\frac{\hat{x}-x}{x}$ are approximately equal.

1.3 The big picture: From real-life problem to computer output

In real life one does not make up problems like "Find $I_2 = \int_0^1 \sin(\sin(x))dx$ " out of thin air. One starts with a certain problem from an area such as engineering, physics, biology, finance,... and wants to obtain a numerical answer. In order to understand how meaningful or meaningless the final computer output is, we have to investigate all possible sources of errors.

1.3.1 Example: Dropping a mass from a height h

I drop a piece of chalk from a height of $h = 5$ feet and want to find the time t_0 it takes until the chalk hits the floor. I only have a simple 4-function calculator.

Newton's law says:

mass $\times$ acceleration $=$ sum of all forces acting on the mass.

I drop the chalk at time $t = 0$. Let us denote the height of the chalk at time t by $y(t)$. Then $y'(t)$ is the velocity, and y'' is the acceleration. The force from gravity acting on the mass is $-mg$ where $g \approx 9.81 \text{meter}/\text{second}^2 \approx 32 \text{ feet}/\text{second}^2$. Therefore we have $my''(t) = -mg$ and $y(0) = h$, $y'(0) = 0$. Taking antiderivatives we obtain

$$y'(t) = -gt, \qquad y(t) = h - \frac{g}{2}t^2$$

and the time t_0 is obtained by solving

$$y(t_0) = 0 \quad \Longleftrightarrow \quad \frac{g}{2}t_0^2 = h \quad \Longleftrightarrow \quad t_0 = \sqrt{\frac{2h}{g}}.$$

In our case the time t_0 in seconds is obtained as $t_0 = \sqrt{\frac{2\cdot 5}{32}}$.

I need to compute $x = \sqrt{a}$ for $a = \frac{5}{16}$. Unfortunately my calculator does not have a $\sqrt{\ }$ -button. I therefore use the following algorithm: I start with an initial guess, e.g., $x = 1$. If x is not the exact value of $\sqrt{a}$, we must have that $\sqrt{a}$ is located between x and $\frac{a}{x}$, and I can hope that the arithmetic mean $(x + \frac{a}{x})/2$ is an improved guess for $\sqrt{a}$. I therefore use the following algorithm:

```
$x := 1$
for $i = 1$ to 5:
    $x := (x + \frac{a}{x})/2$
```

(Note that in general $x = 1$ for the initial guess and 5 for the number of iterations may not be good choices). This is called the "Babylonian algorithm" because it was supposedly known to the ancient Babylonians. If I use this algorithm for $a = \frac{5}{16}$ I obtain an approximation 0.5590169944 for $\sqrt{a}$ using my calculator. Is this really the exact value for the time t_0 ?

1.3.2 Error in given data

For every problem there are some input values. In general they are not known exactly because of measurement or other errors. In our example the initial height h is an input parameter, and in practice I might be able to measure it with a relative error of 10^{-5} , but not exactly. Also the value of the acceleration $g \approx 9.81\text{meter}/\text{second}^2$ is an input value which I do not know exactly.

Even if I do everything else exactly, the error of the given data propates through the whole problem and pollutes my final result. If my problem is very sensitive to small perturbations in my given data ("*ill-conditioned*" problem) this will cause very large errors in the result, and there is no way to avoid this.

1.3.3 Modeling error

I have to translate my real life problem into a mathematical problem. That means that I have to make some simplifications. For our example problem I assumed that the only force is $-mg$. In reality there is another force which comes from the friction with the air. Also, the force from gravity is not really constant, e.g., for $h = 1000\,\text{miles}$ I need to use a different model.

Finding a mathematical model for a real life problem is a topic of application areas like engineering, biology, finance etc. Therefore we will not investigate modeling errors in this class, but we should be aware that they are often the largest source of error in a computed result.

1.3.4 Approximation error

Typically I cannot solve my mathematical problem exactly with a finite number of operations. Therefore I have to pick an algorithm which gives a good approximation. In the example, we used 5 iterations of the Babylonian algorithm instead of the exact value of $\sqrt{a}$. Another example is the approximation of functions using Taylor expansions (see below) where the error is also called "*truncation error*". Yet another example was the approximation of an integral I_2 by a sum of rectangles in section 1.1.

1.3.5 Roundoff error

Theoretically I can imagine that I perform my algorithm using arithmetic with infinite precision. In practice I have to pick a certain computer which has finite precision machine numbers, and finite precision machine arithmetic. Therefore I will obtain a different result, and the error between the infinite precision computation and the machine arithmetic computation is called *roundoff error*.

We will investigate how much roundoff error we can expect for a "good algorithm" (the so-called "*unavoidable error*"). "Bad algorithms" can give unnecesserarily large roundoff errors ("*numerically unstable*" algorithms).

1.4 Taylor series

An important tool for approximation is the Taylor expansion. We will use this many times, and we will need a theorem for the remainder term.

Theorem 1.1 *Assume that f has continuous derivatives up to order $n+1$ between x and x_0 . Then*

$$f(x) = f(x_0) + f'(x_0)(x-x_0) + \cdots + f^{(n)}(x_0)\frac{(x-x_0)^n}{n!} + R_{n+1}$$

with the remainder term

$$R_{n+1} = f^{(n+1)}(t)\frac{(x-x_0)^{n+1}}{(n+1)!}$$

where t is between x and x_0 .

Example We want to compute $y = \sin(0.1)$ using a simple four function calculator. We choose $x_0 = 0$ (since we know the values of $\sin 0$ and $\cos 0$, and 0 is close to $x = 0.1$). We approximate $y = \sin(0.1)$ by the Taylor series with terms up to order $n = 6$. Note that $f(x_0) = \sin(0) = 0$, $f'(x_0) = \cos(0) = 1$, $f''(x_0) = 0$, $f^{(3)}(x_0) = -1$, $f^{(4)}(x_0) = 0$, $f^{(5)}(x_0) = 1$, $f^{(6)}(x_0) = 0$ and therefore the approximation is

$$\tilde{y} = 0 + x + 0 - \frac{x^3}{3!} + 0 + \frac{x^5}{5!} + 0 = 0.1 + \frac{0.1^3}{6} - \frac{0.1^5}{120} = 0.099833416666\ldots$$

and

$$|y - \tilde{y}| = |R_7| = \left| -\cos(t)\frac{x^7}{7!} \right| \leq 1 \cdot \frac{0.1^7}{5040} \approx 1.98 \cdot 10^{-11}.$$

This is an estimate for the absolute error $|\tilde{y} - y|$.

For the relative error $\left|\frac{\tilde{y}-y}{y}\right|$ we first note that $\left|\frac{\tilde{y}-y}{\tilde{y}}\right| \leq \frac{0.1^7/5040}{\tilde{y}} =: \delta \approx 1.98744 \cdot 10^{-10}$. Then we obtain with (1) that

$$\left|\frac{\tilde{y} - y}{y}\right| \leq \frac{\delta}{1 - \delta} \approx 1.98744 \cdot 10^{-10}.$$