3 conditioning and error analysis

3.1 error measures

slide 18
slide 19
slide 20

Numerical simulations contain errors that come from various sources:

- Modelling error: when describing a problem with mathematical equations, various effects are typically neglected (e.g., continuum models versus the atomic structure of gases or solids)
- measurement errors: models typically contain parameters that have to be measured
- roundoff errors: computers work with finite precision numbers (typically floating point numbers), so that an error is made in each floating point operation
- discretization errors: numerical methods are not exact. Examples we have encountered are numerical differentiation and integration

errors are typically measured using *norms*:

slide 19a

Definition 3.1 (norm) *A mapping* $\|\cdot\| : \mathbb{R}^n \to \mathbb{R}_0^+$ *is called a* norm, *if*

(i) (triangle inequality) $\|\mathbf{x}+\mathbf{y}\| \le \|\mathbf{x}\| + \|\mathbf{y}\|$ *for all* $\mathbf{x}, \mathbf{y} \in \mathbb{R}^n$

(ii) (homogeneity) $\|\lambda\mathbf{x}\| = |\lambda|\|\mathbf{x}\|$ *for all* $\mathbf{x} \in \mathbb{R}^n$, $\lambda \in \mathbb{R}$

(iii) (definiteness) $\|\mathbf{x}\| = 0$ *implies* $\mathbf{x} = 0$.

Example 3.2 *Important norms on* $\mathbb{R}^n$ *are:*

1. *the* euklidian norm $\|\mathbf{x}\|_2 := \sqrt{\sum_{i=1}^n |\mathbf{x}_i|^2}$
2. *the* ∞-norm $\|\mathbf{x}\|_\infty := \max_{i=1,\dots,n} |\mathbf{x}_i|$
3. *the* 1-norm $\|\mathbf{x}\|_1 := \sum_{i=1}^n |\mathbf{x}_i|$ ∎

3.2 conditioning and stability of algorithms

The condition number of a problem measures how the (exact) mathematical problem deals with perturbations/errors in the input data:

Definition 3.3 *The condition number of a problem (described as the evaluation of a function* f*) is the factor by which input perturbations are amplified in the worst case. One distinguishes:*

(a) absolute condition number $\kappa_{abs}(x)$ *is the smallest number such that for all sufficiently small* Δx*:*

$$\|f(x) - f(x+\Delta x)\| \le \kappa_{abs}(x)\|\Delta x\|.$$

(b) relative condition number $\kappa_{rel}(x)$ *is the smallest number such that for all sufficiently small* Δx*:*

$$\frac{\|f(x) - f(x+\Delta x)\|}{\|f(x)\|} \leq \kappa_{rel}(x) \frac{\|\Delta x\|}{\|x\|}$$

finis 5.DS

In practice, one can compute the condition number in terms of the derivative of f. In the interest of simplicity, we consider the simple case $f : \mathbb{R} \to \mathbb{R}$. If $f \in C^1$, then Taylor expansion yields $f(x+\Delta x) \approx f(x) + f'(x)\Delta x + \cdots$ so that (approximately) $|f(x+\Delta x) - f(x)| \leq |f'(x)|\,|\Delta x|$. Hence, we see that (essentially)

$$\kappa_{abs}(x) = |f'(x)|$$

For the relative condition number we obtain analogously (for $f(x) \neq 0$)

$$\frac{|f(x+\Delta x) - f(x)|}{|f(x)|} \approx \frac{|f'(x)\Delta x|}{|f(x)|} = \frac{|f'(x)|x}{|f(x)|}\frac{|\Delta x|}{|x|}.$$

That is, we expect

$$\kappa_{rel}(x) = \frac{|f'(x)|}{|f(x)|}|x|.$$

In the following, we consider the relative condition number of a problem. We say that a problem is *well conditioned*, if $\kappa_{rel}(x)$ is "moderate" and it is called *ill conditioned*, if $\kappa_{rel}(x)$ is "large". The notion of "moderate" and "large" are vague, since it depends on the setting and the ultimate goal of the calculation whether a certain amplification of input errors is acceptable or not.

Example 3.4 *The addition of two* positive *numbers is well conditioned: Let* x, $y > 0$ *and* Δx, Δy *with* $|\Delta x|/x \leq \delta$ *and* $|\Delta y|/y \leq \delta$*. Then*

$$\frac{|(x+\Delta x) + (y+\Delta y) - (x+y)|}{|x+y|} \leq \frac{|\Delta x| + |\Delta y|}{x+y} \overset{x,y>0}{\leq} \frac{\delta x + \delta y}{x+y} \leq \delta,$$

i.e. $\kappa_{rel} \leq 1$*. The (relative) error in the result is at most as large as the (relative) input error.*

■

Example 3.5 **slide 21**

Subtracting two numbers of similar size *is ill-conditioned ("cancellation"). Consider the subtraction*

$$x_1 = 1.23456\mathbf{89}? \cdot 10^0$$
$$x_2 = 1.23456\mathbf{79}? \cdot 10^0$$

where ? *stands for an error/uncertainty in the input. The relative input error is thus of size* 10^{-8}*. For the difference*

$$x_1 - x_2 = 0.0000011? \cdot 10^0 = 1.1? \cdot 10^{-6}$$

we get a relative error/uncertainty of 10^{-2}*. Thus, we have lost* 6 *digits. Correspondingly, the (relative) condition number is* $\kappa_{rel} \approx 1.8 \cdot 10^6$*:* *auxiliary computation:*

$$\left|\frac{(x+\Delta x) - (y+\Delta y) - (x-y)}{x-y}\right| = \left|\frac{\Delta x - \Delta y}{x-y}\right| \leq \frac{|\Delta x| + |\Delta y|}{|x-y|}.$$

This leads to $2 \cdot 10^{-8}/(1.1 \cdot 10^{-6}) \approx 1.8 \cdot 10^{-2}$*.* ■

Exercise 3.6 *Show that multiplication and division are well conditioned (relative conditioning).* ■

3.3 stability of algorithms

The algorithmic realization of a mathematical function f is typically done as a concatination

$$f = f_1 \circ f_2 \cdots \circ f_N$$

of functions $f_1, \ldots, f_N$, where one may think of the functions f_i as "elementary functions" such as the addition, subtraction, multiplication, division or as more complex subproblems such as the evaluation of integrals, finding zeros of functions, solutions of differential equations. An algorithm will typically not realize a function exactly, i.e., f will be approximated by

$$\widehat{f} = \widehat{f_1} \circ \widehat{f_2} \cdots \circ \widehat{f_N}.$$

Examples of such approximations are:

- A computer realizes numbers typically as floating point numbers. Hence, already the input is rounded. The elementary operations $+$, $-$, $*$, / cannot be realized exactly.
- Subproblems f_i such as the evaluation of integrals are not exact but are tainted with discretization errors.

An inaccuracy/error that results from using an approximation $\widehat{f_i}$ instead of f_i is potentially amplified by the subsequent functions $\widehat{f_1}, \ldots, \widehat{f_{i-1}}$. A stability analysis of algorithms tries to identify ill-conditioned subproblems $\widehat{f_i}$ and will possibly modify them. Modifying subproblems $\widehat{f_i}$ (or choosing a different decomposition $f_1 \circ \cdots \circ f_{N'}$) is a sensible approach if some subproblems are ill-conditioned but if at the same time the corresponding "exact" functions are well conditioned. We illustrate this procedure with some simple examples in which cancellation (cf. Example 3.5) is the culprit.

Example 3.7 slide 21
Consider the evaluation of the function $f(x) = \log(1+x)$ for small x. The problem is well-conditioned since

$$\kappa_{rel}(x) = \frac{|f'(x)||x|}{|f(x)|} = \frac{|x|}{(1+x)|\log(1+x)|} \leq 2 \qquad \textit{(for } x \textit{ sufficiently close to } 0\textit{)}$$

The "naive" numerical realization is

$$x \overset{f_2}{\mapsto} w := (x+1) \overset{f_1}{\mapsto} \log w.$$

The mapping f_1 is ill-conditioned near $w = 1x + 1$:

$$\kappa_{rel}(w) \approx \frac{w}{w|\log w|} = \frac{1}{|\log w|} = \frac{1}{\log(1+x)} \approx \frac{1}{x}.$$

Hence, we observe the following: The intermediate result $1 + x$ has a relative accuracy of 16 digits but the subsequent application of f_2 may amplify (relative) inaccuracies by a factor $\approx 1/x$. For example, for $x = 10^{-10}$ one has to fear that one loses 10 digits. Indeed, in `matlab`*:*

```
>> x=1.234567890123456e-10;
>> w=1+x; f=log(w)
f =
     1.234568003306966e-10
```

The true value (rounded to 16 digits) is $f = \mathbf{1.23456}7890047248e-10$. *That is, although the IEEE-floating point arithmetic of* `matlab` *uses 16 digits, the result has only 6 corret digits, i.e., 10 digits were lost.*
Since the original function f *is well-conditioned, one may hope to find another algorithm that circumvents this cancellation problem. Indeed, using, e.g., the Taylor approximation of* f *for small* x *gives*

$$f(x) = x - \frac{1}{2}x^2 + \frac{1}{3}x^3 - \cdots \tag{3.1}$$

and one obtains for $x - x^2/2$ *the value* $1.234567890047248e-10$, *which is correct to all digits. This example is not untypical. The situation is such that the final result (here:* x*) is small but that the intermediate results (here:* $1 + x \approx 1$*) are large relative to the final result. One should fear that the small final result is then somehow obtained by subtracting numbers of similar size. A different way of understanding the problem is: by (3.1) the final result is approximately* x *so that one shouldn't lose information contained in the digits of* x. *However, the intermediate results remove information about* x *as the following calculation with 16 digits shows:*

$$\begin{array}{l} \mathbf{1.000000000000000} \\ \mathbf{0.0000000001234567}890123456 \\ \hline \mathbf{1.0000000001234568} \end{array}$$

■

Example 3.8 *The two zeros of the quadratic equation* $x^2 - 2px - q = 0$ *are given by*

$$x_0 = p - \sqrt{p^2 + q}, \qquad x_1 = p + \sqrt{p^2 + q}. \tag{3.2}$$

A (mathematically equivalent) alternative formula is given by

$$x_1 = p + \sqrt{p^2 + q}, \tag{3.3a}$$

$$x_0 = p - \sqrt{p^2 + q} = \frac{(p - \sqrt{p^2 + q})(p + \sqrt{p^2 + q})}{p + \sqrt{p^2 + q}} = \frac{-q}{p + \sqrt{p^2 + q}} = -\frac{q}{x_1} \tag{3.3b}$$

Consider the case $p,\ q > 0$. *If* $p^2 \gg q$ *we expect again cancellation when computing* x_0. *Indeed, in* `matlab`*:*

```
>> p = 400000; q = 1.234567890123456;
>> r = sqrt(p^2+q); x0=p-r
x0 =
    -1.543201506137848e-06
```

The exact solution is $-\mathbf{1.54320}9862651343129e-06$. *The reason is again cancellation in the last step of the realization of the formula for* x_0. *The alternative formula (3.3b) avoids this subtraction and yields a result with 16 correct digits:*

```
>> x1=p+sqrt(p^2+q);x0=-q/x1
x0 =
    -1.543209862651343e-06
```

■