Lecture 9

Error Analysis

Learning Outcomes

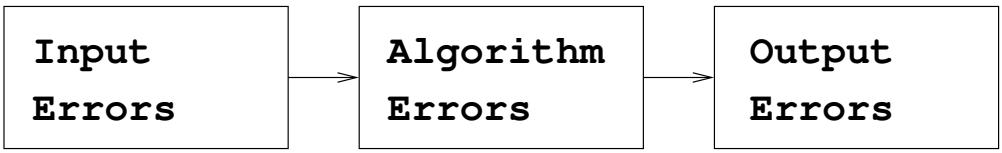
- **Proficiency in Identifying and Analyzing Numerical Errors:**
  - Develop proficiency in identifying and analyzing errors in numerical mathematics, understanding how inaccuracies can be introduced in computed solutions due to input data and algorithmic operations, and gaining insights into the sources of such errors like floating-point representation, data uncertainty, rounding, and approximation errors.
- **Mastery of Forward Error Analysis:**
  - Achieve mastery in forward error analysis, learning to define and differentiate between absolute and relative errors, and understanding the stability of algorithms. Acquire the ability to analyze the propagation of errors through algorithmic operations, recognizing the limitations and challenges of forward error analysis in large and complex problems.
- **Insight into Intrinsic Nature and Conditioning of Problems:**
  - Gain insights into how the intrinsic nature of some problems can inherently produce significant errors due to minor inaccuracies in input data, regardless of the algorithm used. Understand the concept of problem conditioning and acquire the skills to use nuanced methods like backward error analysis for deeper insights into the conditioning of problems and for identifying ill-conditioned problems effectively.

Introduction

In an abstract way, a problem consists of input data of a specific type that are transformed to output data of a specific type using an algorithm:

Example: solving $f : GL(n) \times \mathbb{C}^n \to \mathbb{C}^n$, $f(A, b) = A^{-1}b$ with **GEPP**.

The subject of NUMERICAL MATHEMATICS is to analyse and estimate errors in the result in terms of errors in the input data and occurring when performing operations:



The question is: how close is the computed solution to the correct one?

Input Errors: We have two sources in mind,

1. floating point representation of numbers in computers,
2. input data uncertainty, for example parameters obtained from experimental measurements.

Algorithm Errors: Again, there are mainly two sources,

1. rounding errors when performing instructions on a computer, e.g., elementary operations $+, -, \times, /$ in floating point arithmetic,
2. approximation errors, for example

$$f'(x) \approx \frac{f(x+h) - f(x-h)}{2h}, \quad \exp(x) \approx \sum_{n=0}^{N} \frac{1}{n!} x^n.$$

Backward Error

Before starting to talk about errors, two definitions: If ξ is an approximation to a datum x

$$\textbf{absolute error:} \quad \|\xi - x\|, \qquad \textbf{relative error:} \quad \frac{\|\xi - x\|}{\|x\|}.$$

By $\mathbb{F} \subset \mathbb{R}$ we denote the floating point numbers that can be represented on our computing machine (see Lecture 8).

Assumption 9.1. [3.19]

A1 *For all $x \in \mathbb{R}$ there is an $\varepsilon \in [-\varepsilon_m, \varepsilon_m]$ such that*

$$\xi = fl(x) = x(1 + \varepsilon).$$

This is a fundamental property of floating point representation. It means that for any real number x, there exists a floating point representation such that the relative error between the number, x and its floating point representation, $fl(x)$ is $< \epsilon_m$.

A2 *For each elementary executable operation $* \in \{+, -, \times, /, \sqrt{\cdot}\}$ and every $x, y \in \mathbb{F}$ there is a $\delta \in [-\varepsilon_m, \varepsilon_m]$ with*

$$x \circledast y = (x * y)(1 + \delta)$$

where $\circledast$ denotes the computed version of $*$.

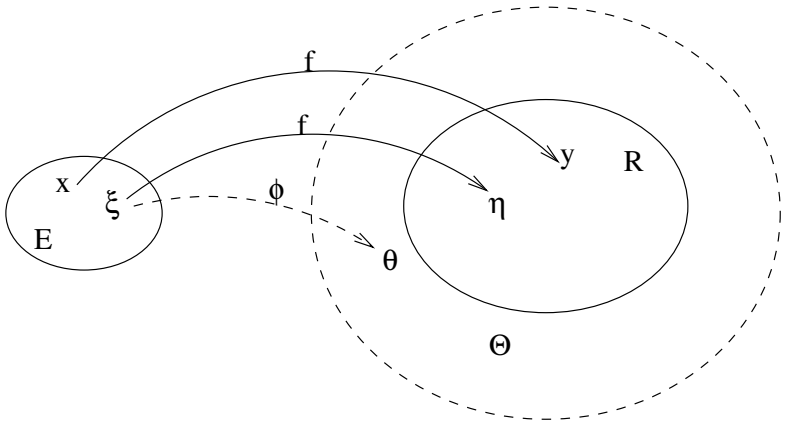
In particular, overflow and underflow are neglected. The above two assumptions are assumed to hold for any computing machine that satisfies the IEEE standards.

Forward Error Analysis

Similarly here we will discuss an algorithm of a function which will be referred to as $\phi : x \longrightarrow y$, where ϕ is an algorithm that approximates the computations of an actual function $f : x \longrightarrow y$. The question to be answered here is how good an algorithm is.
To re-emphasise the subject here is the analysis of $\Theta = \phi(E)$ where ϕ is an algorithm for f. The set E represents input data that cannot be distinguished from x,

$$E := \{\xi \,|\, \|\xi - x\| \leq \delta\|x\|\} \text{ with a given accuracy / tolerance } \delta. \tag{9.1}$$

The set R will be defined later on.



Definition 9.2. [3.17] *The forward error is defined by:*

$$\textit{absolute:} \quad \|\phi(\xi) - f(x)\| = \|\theta - y\|, \quad \textit{relative:} \quad \frac{\|\phi(\xi) - f(x)\|}{\|f(x)\|} = \frac{\|\theta - y\|}{\|y\|}.$$

An algorithm is forward stable if

$$\frac{\|\theta - y\|}{\|y\|} = \frac{\|\phi(\xi) - f(x)\|}{\|f(x)\|} = O(\varepsilon_m) \quad \textit{whenever } \frac{\|\xi - x\|}{\|x\|} = O(\varepsilon_m) \qquad \textit{as } \varepsilon_m \searrow 0.$$

A straight forward way towards an error analysis is to take the errors of order ε_m for every input number and every elementary executable operation into account and to drop terms of order $o(\varepsilon_m)$ as $\varepsilon_m \to 0$.

Example: Given three numbers $x_1, x_2, x_3 \neq 0$ we want to compute $f(x_1, x_2, x_3) = x_1x_2/x_3$.
A possible algorithm is to first compute the product x_1x_2 and then to divide the result by x_3. By assumption A1, there are floating point numbers ξ_i and small reals $\varepsilon_i = O(\varepsilon_m)$ as $\varepsilon_m \to 0$ such that $\xi_i = x_i(1 + \varepsilon_i)$, $i = 1, 2, 3$. With assumption A2, the first step of the algorithm yields a number $\xi_1\xi_2(1 + \delta_1)$ with some $\delta_1 = O(\varepsilon_m)$. The second step involves a $\delta_2 = O(\varepsilon_m)$ associated with the small error due to the division and results in $(\xi_1\xi_2/\xi_3)(1 + \delta_1)(1 + \delta_2) = (\xi_1\xi_2/\xi_3)(1 + \delta_1 + \delta_2 + O(\varepsilon_m^2))$ where we used that $\delta_1\delta_2 = O(\varepsilon_m^2) = o(\varepsilon_m)$. Together with the

erroneous input data the algorithm computes

$$\begin{aligned}(x_1, x_2, x_3) \mapsto & \frac{\xi_1\xi_2}{\xi_3}(1+\delta_1+\delta_2+O(\varepsilon_m^2)) \\ &= \frac{x_1(1+\varepsilon_1)x_2(1+\varepsilon_2)}{x_3(1+\varepsilon_3)}(1+\delta_1+\delta_2+O(\varepsilon_m^2)) \\ &= \frac{x_1x_2}{x_3}(1-\varepsilon_3+O(\varepsilon_m^2))(1+\varepsilon_1+\varepsilon_2+\delta_1+\delta_2+O(\varepsilon_m^2)) \\ &= \frac{x_1x_2}{x_3}(1+\varepsilon_1+\varepsilon_2-\varepsilon_3+\delta_1+\delta_2+O(\varepsilon_m^2)) \quad =: \tilde{f}(x)\end{aligned}$$

Hence we obtain for the relative forward error that

$$\frac{|\tilde{f}(x)-f(x)|}{|f(x)|} = |\varepsilon_1+\varepsilon_2-\varepsilon_3+\delta_1+\delta_2+O(\varepsilon_m^2)| \leq 5\varepsilon_m + O(\varepsilon_m^2) = O(\varepsilon_m) \quad \text{as } \varepsilon_m \to 0.$$

One can imagine that this type of error analysis becomes quite tedious when the problems become large and the algorithms complicated. An advantage is that it is possible to perform them with a computer again.
Nevertheless, it turns out that the estimates usually are quite rough and, even more important, yield no particular insight. As we will see there are problems which intrinsically lead to relatively large errors when the input data involve small errors independently of the algorithm used to solve it. The above method of forward error analysis does not care about this so-called conditioning of problems let alone that ill-conditioned problems are detected.
A more sophisticated way that meanwhile is common in numerical linear algebra is the backward error analysis. But before turning to this idea we have a look at this notion of conditioning.

Lecture 10

Conditioning

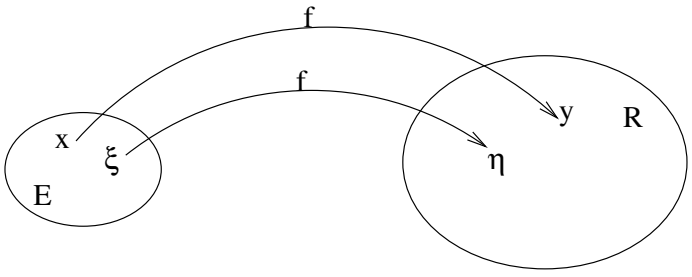
Learning Outcomes

- **Proficiency in Understanding Conditioning:**
  - Acquire proficiency in exploring and understanding the concept of conditioning, learning how it reflects the sensitivity of the solution to small perturbations in the input data and distinguishing between well-conditioned and ill-conditioned problems.
- **Mastery of Condition Number Concept:**
  - Achieve mastery in the concept of a condition number, understanding its significance as a measure to determine the conditioning of a problem and its implications in scalar and higher dimensional problems.
- **Skill in Analyzing Impact of Perturbations:**
  - Develop the skill to analyze the impact of perturbations in the input data on the solution, focusing on understanding the relationship between the condition number of a matrix, perturbations in the matrix, and the resultant errors in the solution.

Understanding conditioning

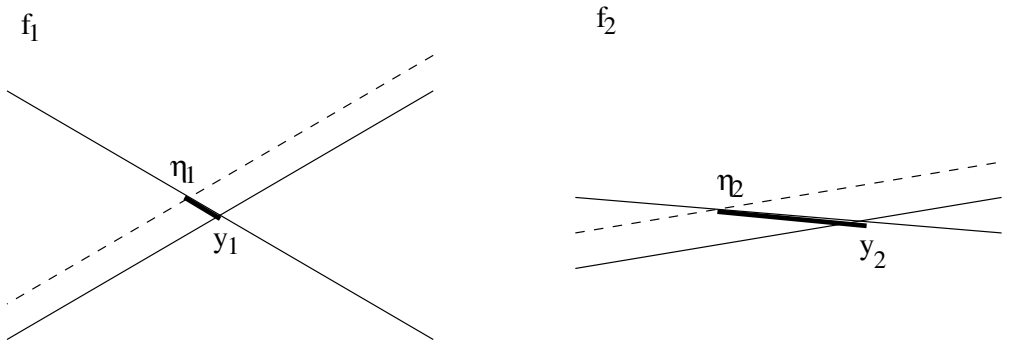
The key question we are going to answer in this lecture is how the solution of a problem change when its data is perturbed. This is known as conditioning.

Conditioning indicates how sensitive the solution of a problem $f : x \mapsto y = f(x)$ may be to small perturbations in the input data.



The result set is $R = f(E)$. The Conditioning depends on the size of R. The problem f is called well-conditioned if R small and ill-conditioned if R is rather big.

Example: Consider a 2×2 linear system of equations or, as an equivalent geometric problem, the cut of two lines.



Errors in the input data correspond to changes of the lines. Problem f_1 is better conditioned than problem f_2 since the change of the intersection point is less drastic when shifting one of the lines by about the same amount. In this particular example, a small angle between the lines is deemed unfavourable.

Example: compute $f(x) = x + 1$.
Reminder: floating point representation guarantees that,

$$\left|\frac{fl(x) - x}{x}\right| = \epsilon, \quad \text{for some } 0 \leq \epsilon \leq \epsilon_m \implies$$
$$\frac{fl(x) - x}{x} = \pm\epsilon \implies$$
$$fl(x) = x(1 \pm \epsilon)$$

Thus,

$$f(fl(x)) = x(1 \pm \epsilon) + 1$$

which in turn means that,

$$\begin{aligned} f(fl(x)) - f(x) &= x(1 \pm \epsilon) + 1 - x - 1 \\ &= \pm\epsilon x \end{aligned}$$

Now considering the relative error yields,

$$\begin{aligned} \frac{f(fl(x)) - f(x)}{f(x)} &= \frac{\pm\epsilon x}{1 + x} \\ &= \frac{\epsilon|x|}{|1 + x|} \end{aligned}$$

which means that the perturbation in the result is proportional to the perturbation in the data.

Subtractive cancellation: if $x \to -1$, then $|1 + x| \to 0$, resulting in a large relative error in response to a small perturbation.

Now, the general goal is to find a significant number to measure how well a problem is conditioned. To keep things simple, Consider a simple scalar problem $f : \mathbb{R} \to \mathbb{R}$. The exact value is x and its floating point representation is ξ, thus,

$$f(\xi) - f(x) \longleftarrow \text{change in result}$$
$$\frac{f(\xi) - f(x)}{f(x)} \longleftarrow \text{relative change in result}$$

Since this is proportional to relative change of data, we get

$$\frac{\frac{f(\xi)-f(x)}{f(x)}}{\frac{\xi-x}{x}} = \frac{\frac{f(x(1+\epsilon))-f(x)}{f(x)}}{\epsilon}$$

which can be further simplified by considering,

$$= \lim_{\epsilon \to 0} \left(\frac{f(x+\epsilon x) - f(x)}{\epsilon x} \cdot \frac{x}{f(x)} \right)$$
$$= f'(x) \frac{x}{f(x)}$$

A meaningful definition of a condition number (of problem f in a point x) therefore is,

$$\kappa_f(x) := \left\| \frac{x f'(x)}{f(x)} \right\|.$$

For higher dimensional problems $f : \mathbb{R}^m \to \mathbb{R}^n$ this reads

$$\kappa_f(x) = \|J(x)\| \frac{\|x\|}{\|f(x)\|}$$

where $J(x)$ is the Jacobian of f and $\|J(x)\|$ is a matrix norm induced by the vector norms used for x on $\mathbb{C}^m$ and $f(x)$ on $\mathbb{C}^n$.

Example: $f(x) = \arcsin(x)$, then $f'(x) = 1/\sqrt{1-x^2}$ and $\kappa_f = x/(\sqrt{1-x^2}\arcsin(x))$, $\kappa_f \to \infty$ as $x \to \pm 1$, hence the evaluation of arcsin close to $x = 1$ is an ill-conditioned problem.

Conditioning of (SLE)

Consider the problem of computing the matrix-vector product $f(A, x) = Ax =: b$, $x, b \in \mathbb{C}^n$, $A \in \mathbb{C}^{n \times n}$. Then $f'(x) = A$, and the condition number becomes $\kappa_f(x) = \|A\|\|x\|/\|Ax\|$ (we omit the discussion of the trivial case $x = 0$). Assume now that A is invertible and observe that then

$$\|x\| = \|A^{-1}Ax\| \leq \|A^{-1}\|\|Ax\|.$$

We deduce that

$$\kappa_f(x) = \|A\| \frac{\|x\|}{\|Ax\|} \leq \|A\|\|A^{-1}\|$$

Occasionally, when considering the problem $g(A, b) = A^{-1}b =: x$ we just have to replace A by A^{-1} in the above analysis to obtain the same estimate for the condition number.

Definition 10.1. [3.2] *The condition number of a square matrix* $A \in \mathbb{C}^{n\times n}$ *is*

$$\kappa(A) = \begin{cases} \|A\|\|A^{-1}\| & \text{if } A \text{ is regular,} \\ \infty & \text{otherwise.} \end{cases}$$

Remark: Conditioning applies to other types of errors or uncertainties. Examples uncertainties are

- parameter uncertainty.
- sensors' noises.
- environment.

Now that we understand conditioning, we are going to discuss next how to use this concept in order to analyse the solutions of $Ax = b$.

Proposition 10.2. [3.3] *Let* A *be regular,* $Ax = b \neq 0$ *and* $A(x + \Delta x) = b + \Delta b$. *Then*

$$\frac{\|\Delta x\|}{\|x\|} \leq \kappa(A)\frac{\|\Delta b\|}{\|b\|}.$$

Proof. Known or given to us is,

$$\begin{aligned} Ax &= b \\ A(x + \Delta x) &= b + \Delta b \end{aligned}$$

Expanding the second equation yields,

$$\begin{aligned} Ax &= b \\ Ax + A\Delta x &= b + \Delta b \end{aligned} \tag{10.1}$$

subtracting the second equation from the first yields,

$$A\Delta x = \Delta b \tag{10.2}$$

10.1 implies that,

$$\begin{aligned} &||b|| = ||Ax|| \\ &\implies ||b|| = ||Ax|| \leq ||A||||x|| \\ &\implies \frac{1}{||x||} \leq ||A||\frac{1}{||b||} \end{aligned} \tag{10.3}$$

rewriting and applying the norm to (10.2) yields,

$$\begin{aligned} \Delta x &= A^{-1}\Delta b \\ ||\Delta x|| &= ||A^{-1}\Delta b|| \leq ||A^{-1}||||\Delta b|| \end{aligned} \tag{10.4}$$

Multiplying (10.3) and (10.4) yield,

$$\begin{aligned} \frac{\|\Delta x\|}{\|x\|} &\leq \|A\|\|A^{-1}\|\frac{\|\Delta b\|}{\|b\|} \\ \frac{\|\Delta x\|}{\|x\|} &\leq \kappa(A)\frac{\|\Delta b\|}{\|b\|} \end{aligned}$$

□

The following proposition provides an estimate with respect to perturbations of the matrix A.

Proposition 10.3. [3.5] *Let A be regular, $Ax = b \neq 0$ and $(A + \Delta A)(x + \Delta x) = b$. If $\kappa(A)\|\Delta A\| < \|A\|$ then*

$$\frac{\|\Delta x\|}{\|x\|} \leq \frac{\kappa(A)}{1 - \frac{\kappa(A)\|\Delta A\|}{\|A\|}} \frac{\|\Delta A\|}{\|A\|}.$$

Proof.

$$Ax = b \quad \text{and} \quad (A + \Delta A)(x + \Delta x) = b$$

implies that,

$$(A + \Delta A)(x + \Delta x) = Ax$$

or equivalently,

$$\Delta A x + A\Delta x + \Delta A \Delta x = 0$$

which can be rewritten as,

$$\Delta x = A^{-1}(-\Delta A x - \Delta A \Delta x)$$

Applying the norm to this equation yields,

$$\|\Delta x\| = \|A^{-1}(-\Delta A x - \Delta A \Delta x)\| \leq \|A^{-1}\|\|\Delta A\|\|x\| + \|A^{-1}\|\|\Delta A\|\|\Delta x\|$$

Subtracting $\|A^{-1}\|\|\Delta A\|\|\Delta x\|$ from both sides of the equation yields,

$$\begin{aligned} &||\Delta x|| - \|A^{-1}\|\|\Delta A\|\|\Delta x\| \leq \|A^{-1}\|\|\Delta A\|\|x\| \\ \implies &(1 - \|A^{-1}\|\|\Delta A\|)\|\Delta x\| \leq \|A^{-1}\|\|\Delta A\|\|x\| \end{aligned}$$

which can be rewritten as,

$$\frac{\|\Delta x\|}{\|x\|} \leq \frac{\|A^{-1}\|\|\Delta A\|}{(1 - \|A^{-1}\|\|\Delta A\|)}$$

or equivalently,

$$\begin{aligned} \frac{\|\Delta x\|}{\|x\|} &\leq \frac{\|A\|\|A^{-1}\|\frac{\|\Delta A\|}{\|A\|}}{(1 - \|A\|\|A^{-1}\|\frac{\|\Delta A\|}{\|A\|})} \\ &\leq \frac{\kappa(A)\|\frac{\|\Delta A\|}{\|A\|}}{\left(1 - \kappa(A)\|\frac{\|\Delta A\|}{\|A\|}\right)} \end{aligned}$$

□

Bringing the results of the propositions 10.2 and 10.3 together one can show

Theorem 10.1. *Let $A, \Delta A \in \mathbb{C}^{n\times n}$ regular such that $\kappa(A)\|\Delta A\| \leq \|A\|$ and let $b \in \mathbb{C}^n\backslash\{0\}$, $\Delta b \in \mathbb{C}^n$. Assume that $x \in \mathbb{C}^n$ solves $Ax = b$ and $\hat{x}$ solves $(A + \Delta A)\hat{x} = b + \Delta b$. Then*

$$\frac{\|\hat{x} - x\|}{\|x\|} \leq \frac{\kappa(A)}{1 - \frac{\kappa(A)\|\Delta A\|}{\|A\|}} \Big(\frac{\|\Delta A\|}{\|A\|} + \frac{\|\Delta b\|}{\|b\|}\Big).$$

Hints:

- One can start from $A\hat{x} = b + \Delta b - \Delta A\hat{x}$, and subtract Ax and b (given they are equal to one another) from the lhs and rhs, respectively.
- Multiply from the left by A^{-1} and use norm properties (separate out terms as much as possible with relevant inequalities). It also helps at this stage to notice that $||\hat{x}|| \leq ||x|| + ||\hat{x} - x||$ on the rhs. The *extra* terms can then be suitably shifted to the lhs and re-cast into a term involving A and ΔA.
- Division by $||x||$ and using the definition of the condition number $\kappa(A)$ to replace relevant terms involving A and its inverse lead us to an almost final result.
- At the final stage note one of our assumptions (i.e. $\kappa(A)\|\Delta A\| < \|A\|$) to ensure that a suitable division does not become problematic and we reach our conclusion.

Exercise: (Conditioning of linear systems)
Consider a linear systems described by,

$$\underbrace{\begin{pmatrix} 2 & -1.1 \\ -4 & 2.1 \end{pmatrix}}_{A} \underbrace{\begin{pmatrix} x_1 \\ x_2 \end{pmatrix}}_{x} = \underbrace{\begin{pmatrix} 1 \\ -1 \end{pmatrix}}_{b}$$

- Compute the solution of this linear system.
- Compute the condition number of the matrix A.
- Analyse the solution when,
$$b = \begin{pmatrix} 1+\delta \\ -1 \end{pmatrix}$$
where $0 < \delta < 1$
- Analyse the solution when,
$$A = \begin{pmatrix} 2+\epsilon & -1.1 \\ -4 & 2.1 \end{pmatrix}$$
where $0 < \epsilon < 1$
- Compute the solution when $\epsilon = 0.1$.

Exercise: (Conditioning number of a matrix)
Using a norm of your choice, find the condition number of the following matrix,

$$A = \begin{pmatrix} \epsilon & 1 \\ 0 & 1 \end{pmatrix},$$

where $0 < \epsilon < 1$. Explain what happens to the condition number when ϵ varies.

Lecture 11

Backward Error Analysis

Learning Outcomes

- **Proficiency in Algorithm Definition and Decomposition:**
  - Attain proficiency in defining algorithms, emphasizing understanding of their finiteness, effectivity, termination, and determinacy, and acquire the ability to decompose algorithms into elementary executable operations.
- **Mastery of Backward Error Analysis Concept:**
  - Achieve mastery in the concept of backward error analysis, learning to consider the computed value as the exact result for perturbed input data, and understanding the conditions under which an algorithm is deemed backward stable.
- **Insightful Understanding of Error Propagation and Problem Conditioning:**
  - Develop insightful understanding of how backward error analysis can provide sharper estimates and deeper insights into error propagation and problem conditioning, learning to distinguish between problem intrinsic and algorithm intrinsic parts.

Algorithm Definition

Before we can proceed with an error analysis we have to say what an algorithm is. We give a general definition first.

Definition 11.1. *An algorithm is a process consisting of a set of elementary components (called steps) to derive specific output data from specific input data where the following conditions have to be fulfilled:*

- *finiteness: the whole process is described by a finite text,*
- *effectivity: each step can be executed on a computing machine,*
- *termination: the process terminates after a finite number of steps,*
- *determinacy: the course of the process is completely and uniquely prescribed.*

Elementary executable operations are the operations $\{+, -, \times, /, \sqrt{\cdot}\}$. *An algorithm for a map* f *is a decomposition*

$$f = f^{(l)} \circ \cdots \circ f^{(1)}, \quad l \in \mathbb{N},$$

into maps $f^{(i)}$ *involving at most one elementary executable operation.*

Occasionally, a step will contain a reference to another algorithm.
In a realisation of an algorithm, elementary executable operations involve errors (assumption A2), and we denote by $\phi^{(i)}$ a realisation of $f^{(i)}$ so that $\phi = \phi^{(l)} \circ \cdots \circ \phi^{(1)}$ is a realisation of f.

General types of errors that occur in the process of computing an approximation to the exact solution are:

- data error is the error in the data x. In real world applications, numerical analysis involves solving a problem with perturbed input data ζ. The exact data is often unavailable because it must be obtained by measurements which are usually susceptible to noises and environmental effects or other computations that fail to be exact due to limited precision.
- computational error which refers to the error that occurs when attempting to compute $f(\zeta)$. Effectively, we must approximate $f(\zeta)$ by the quantity $\phi(\zeta)$, where ϕ is a function that approximates f. This approximation may be the result of truncation, which occurs when it is not possible to evaluate f exactly using a finite sequence of steps, and therefore a finite sequence that evaluates f approximately must be used instead. This particular source of computational error will be discussed in this lecture.
- Another source of computational error is round off error, which was discussed in the previous lecture, thus we are computing $\phi(\xi)$.

Suppose that we compute an approximation $\theta = \phi(\xi)$ of the value $y = f(x)$ for a given function f and given problem data x. In order to analyse the accuracy of this approximation the following definitions from previous lectures are required,

Errors definitions

Definition 11.2. *The forward error: Let* x *be a real number and let* $f : R \longrightarrow R$ *be a function. If* θ *is a real number that is an approximation to* $y = f(x)$*, then the forward error in* θ *is,*

$$\textit{absolute forward error:} \quad \|\theta - y\|, \quad \textit{relative forward error:} \quad = \frac{\|\theta - y\|}{\|y\|}.$$

Clearly, our primary goal in error analysis is to obtain an estimate of the forward error. Unfortunately, it can be difficult to obtain this estimate directly. An alternative approach is to instead view the computed value θ as the exact solution of a problem with modified data; i.e., $\theta = f(\zeta)$ where ζ is a perturbation of x.

Definition 11.3. [3.17],[3.20] *The backward error: Let* x *be a real number and let* $f : R \longrightarrow R$ *be a function. Suppose that the real number* θ *is an approximation to* $y = f(x)$ *and that* θ *is in the range of* f*, that is* $\theta = f(\zeta)$ *for some real number* ζ*. Then*

$$\textit{absolute backward error:} \quad \|\zeta - x\|, \qquad \textit{relative backward error:} \quad \frac{\|\zeta - x\|}{\|x\|}.$$

An algorithm is backward stable if

$$\frac{\|\zeta - x\|}{\|x\|} = O(\varepsilon_m) \quad \textit{as } \varepsilon_m \searrow 0. \tag{11.1}$$

In most cases, the goal of error analysis is to obtain an estimate of the forward relative error $(f(\zeta)-f(x))/f(x)$, but it is often easier to instead estimate the relative backward error $(\zeta-x)/x$. Therefore, it is necessary to be able to estimate the forward error in terms of the backward error. The following definition address this need.

Definition 11.4. *The condition number: Let x be a real number and let $f : R \longrightarrow R$ be a function. The relative condition number of the problem of computing $y = f(x)$ is the ratio of the magnitude of the relative forward error to the magnitude of the relative backward error,*

$$\frac{\|\theta - y\|}{\|y\|} = \frac{\|\phi(\xi) - f(x)\|}{\|f(x)\|} = \frac{\|f(\zeta) - f(x)\|}{\|f(x)\|} \leq \kappa_f(x)\frac{\|\zeta - x\|}{\|x\|}.$$

Which means that,

$$\textit{relative forward error} \leq \kappa_f \textit{ relative backward error.}$$

Recall that we finally are interested in estimating the forward error. The above outlined backward error analysis typically yields sharper estimates than the standard forward error analysis and, in addition, splits into a problem intrinsic part (the conditioning) and an algorithm intrinsic part (the backward error). For a deeper discussion on error analysis, see (Higham, 2002).

Concluding remarks

1. determining the condition, or sensitivity, of a problem is an important task in the error analysis of an algorithm designed to solve the problem, but it does not provide sufficient information to determine whether an algorithm will yield an accurate approximate solution.

2. recall that the condition number of a function f depends, among other things, on the absolute forward error $f(\zeta) - f(x)$. However, an algorithm for evaluating $f(x)$ actually evaluates a function ϕ that approximates f, producing an approximation $\theta = \phi(x)$ to the exact solution $y = f(x)$. In our definition of backward error, we have assumed that $\phi(x) = f(\zeta)$ for some perturbed input ζ that is close to x. In another word, obtaining the solution with an algorithm involving approximations instead of performing exact computation is converted to exactly solving the problem with perturbed initial data.

3. this assumption has allowed us to define the condition number of f independently of any approximation ϕ. This independence is necessary, because the sensitivity of a problem depends solely on the problem itself and not any algorithm that may be used to approximately solve it.

Example 1: The subtraction $\ominus$ is backward stable.

The exact version is $x = (x_1, x_2)^T$, $f(x) = x_1 - x_2 = y$. With some numbers $|\varepsilon^{(i)}| \le \varepsilon_m$, $i = 1, 2, 3,$, the computed version is

$$\theta = fl(x_1) \ominus fl(x_2) = \xi_1 \ominus \xi_2 = (\xi_1 - \xi_2)(1 + \varepsilon^{(3)}) = \\ = \big(x_1(1 + \varepsilon^{(1)}) - x_2(1 + \varepsilon^{(2)})\big)(1 + \varepsilon^{(3)}) = x_1(1 + \varepsilon^{(4)}) - x_2(1 + \varepsilon^{(5)})$$

where $\varepsilon^{(4)} = \varepsilon^{(1)} + \varepsilon^{(3)} + \varepsilon^{(1)}\varepsilon^{(3)} = O(\varepsilon_m)$ as $\varepsilon_m \searrow 0$, $\varepsilon^{(5)}$ similarly.
Hence $f(\zeta) = \theta$ for $\zeta = (x_1(1 + \varepsilon^{(4)}), x_2(1 + \varepsilon^{(5)}))^T$, and we conclude that

$$\|\zeta - x\| = \|(\varepsilon^{(4)}x_1, \varepsilon^{(5)}x_2)^T\| \le \varepsilon_m \|x\|.$$

Example 2: Let $x, y \in \mathbb{C}^n$ where, for simplicity, we assume that they are not defective. Computing the standard inner product recursively by

$$f_n\big(x, y\big) = \langle (x_1, \dots, x_n)^T, (y_1, \dots, y_n)^T \rangle = \overline{x}_n y_n + f_{n-1}\big(\underbrace{(x_1, \dots, x_{n-1})^T}_{=:x^{n-1}}, \underbrace{(y_1, \dots, y_{n-1})^T}_{=:y^{n-1}}\big)$$

is a backward stable algorithm in the sense that

$$\phi_n(x, y) = \langle \zeta, y \rangle$$

with a $\zeta \in \mathbb{C}^n$ satisfying $|\zeta_k - x_k| \le (n\varepsilon_m + o(\varepsilon_m))|x_k|$ as $\varepsilon_m \to 0$, $k = 1, \dots, n$.

Proof. The recursive definition of the algorithm for the scalar product invites for a proof by induction. If $n = 1$ we have the ordinary product on $\mathbb{C}$ which is backward stable as can be shown similarly to the subtraction in the previous example.
Let $n > 1$ and assume that the assertion is true for $n - 1$. We then have

$$\phi_n(x, y) = \big(\overline{x}_n y_n(1 + \varepsilon^{(1)}) + \phi_{n-1}(x^{n-1}, y^{n-1})\big)(1 + \varepsilon^{(2)})$$

with $|\varepsilon^{(i)}| \le \varepsilon_m$, $i = 1, 2$. By the induction hypothesis there is a $\tilde{\zeta}^{n-1} := (\tilde{\zeta}_1, \dots, \tilde{\zeta}_{n-1})^T \in \mathbb{C}^{n-1}$ with $\|\tilde{\zeta}^{n-1} - x^{n-1}\|_\infty / \|x^{n-1}\|_\infty \le (n-1)\varepsilon_m + o(\varepsilon_m)$ such that $\phi_{n-1}(x^{n-1}, y^{n-1}) = f_{n-1}(\tilde{\zeta}^{n-1}, y^{n-1})$. Setting

$$\overline{\zeta}_n := \overline{x}_n(1 + \varepsilon^{(1)})(1 + \varepsilon^{(2)}), \qquad \zeta_k := \tilde{\zeta}_k(1 + \varepsilon^{(2)}) \quad k = 1, \dots, n-1,$$

we obtain $\phi_n(x, y) = f_n(\zeta, y)$. Furthermore

$$|\zeta_n - x_n| \le (2\varepsilon_m + O(\varepsilon_m^2))|x_n| \le (n\varepsilon_m + o(\varepsilon_m))|x_n|,$$

as well as

$$|\zeta_k - x_k| \le |\zeta_k - \tilde{\zeta}_k| + |\tilde{\zeta}_k - x_k| \\ \le \varepsilon_m|\tilde{\zeta}_k| + ((n-1)\varepsilon_m + o(\varepsilon_m))|x_k| = (n\varepsilon_m + o(\varepsilon_m))|x_k|, \quad k < n-1$$

which yields the asserted estimate of the relative backward error. □

Exercise: This exercise demonstrate point 1 in the concluding remarks,
Consider the following SLE,

$$\underbrace{\begin{pmatrix} 0.254 \\ 0.127 \end{pmatrix}}_{b} = \underbrace{\begin{pmatrix} 0.913 & 0.659 \\ 0.457 & 0.33 \end{pmatrix}}_{A} \underbrace{\begin{pmatrix} x_1 \\ x_2 \end{pmatrix}}_{x}$$