

- Introduction to numerical mathematics
- Computer arithmetics
- Conditioning and stability of an algorithm
- Direct and iterative methods
- Systems of linear equations

## Numerical mathematics

### Introduction

**Numerical mathematics** is devoted to methods that seek an approximate but sufficiently accurate solution of problems in various fields. A **simplified mathematical model** of the problem is used; its partial tasks consist of various mathematical problems.

The following mathematical problems are often involved:

1. solution of systems of linear equations,
2. solution of differential equations,
3. calculation of integrals,
4. evaluations of function values,
5. estimation of errors in calculations,
6. ...

Typically, a computer calculation is involved.

- Error in the Patriot missile system

$$(0.1)_{10} = (0.000110011001100110011001100110011\dots)_2$$

- Explosion of the Ariane 5 rocket conversion from a 64-bit floating point number to a 16-bit signed integer
- ...

This does not mean that approximation methods do not work. In the vast majority of cases they work well, but it is important to know how reliable they are.

## Origin of errors

---

Category of errors

We will use different **approximations** to design the algorithm. We will therefore make various kind of mistakes, which can be divided according to their origin:

1. errors in the **model**: the mathematical model to solve the problem is somehow simplified.
2. errors in the **data**: data often come from measurements that do not have absolute accuracy.
3. errors in the **algorithm**: we don't have to have an algorithm that finds the exact solution in a finite number of steps.
4. **rounding** errors: errors occur during the calculation itself (e.g., during arithmetic operations).

Apart from data errors, we will give examples of all other kinds of errors. We start with rounding errors, which are given by the fact that the algorithm need a computer to do the hard work.

## Computer arithmetics

### Representation with floating point

To store a number in computer we usually use the binary number system.

$$(6)_{10} = (110)_2 \quad (0.1)_{10} = (0.000110011001100110011001100110011...)_{2}$$

For non-integers, one can use the **scientific notation**. In the binary base a number  $x$  is represented as

$$x = \pm m \cdot 2^e.$$

$m$  - **mantissa/significand** having a fixed number of digits / fixed length; these digits are also called **significant digits**.

$e$  - **exponent** having a fixed number of digits / fixed length.

Representation  
with floating  
point

IEEE-754

A number  $x$  is represented by its sign  $s$  and by the numbers  $e$  and  $m$ .

The standard IEEE-754 defines the following lengths of  $e$  and  $m$  and their interpretation.

precision	length of $m$	$d =$ length of $e$	$b$
binary32 / single precision	23	8	127
binary64 / double precision	52	11	1023
binary128 / quadruple precision	112	15	16383

- if  $e = 2^d - 1$  and  $m \neq 0$ , then  $x = \text{NaN}$  (Not a Number)
- if  $e = 2^d - 1$  and  $m = 0$  and  $s = 0$ , then  $x = +\text{Inf}$
- if  $e = 2^d - 1$  and  $m = 0$  and  $s = 1$ , then  $x = -\text{Inf}$
- if  $0 < e < 2^d - 1$ , then  $x = (-1)^s \cdot (1.m)_2 \cdot 2^{e-b}$  (so-called **normalized numbers**)
- if  $e = 0$  and  $m \neq 0$ , then  $x = (-1)^s \cdot (0.m)_2 \cdot 2^{-b+1}$  (so-called **subnormal/unnormalized numbers**)

- if  $e = 0$  and  $m = 0$  and  $s = 0$ , then  $x = 0$
- if  $e = 0$  and  $m = 0$  and  $s = 1$ , then  $x = -0$

---

Machine numbers (1/3)

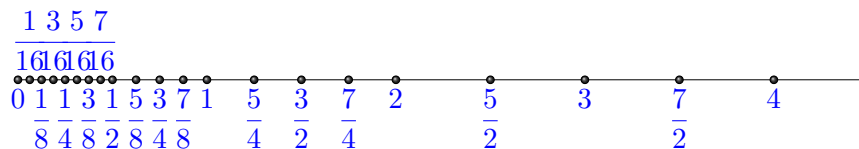
The numbers that can be represented as floating point numbers (with selected finite lengths of  $m$  and  $e$ ) are called **machine numbers**.

**Example:** take  $m$  of length 2 bits,  $e$  of length 3 bits, and  $b = 3$ .

We obtain the following set of numbers (we consider only positive elements)

$$\left\{ 0, \frac{1}{16}, \frac{1}{8}, \frac{3}{16}, \frac{1}{4}, \frac{5}{16}, \frac{3}{8}, \frac{7}{16}, \frac{1}{2}, \frac{5}{8}, \frac{3}{4}, \frac{7}{8}, 1, \frac{5}{4}, \frac{3}{2}, \frac{7}{4}, 2, \frac{5}{2}, 3, \frac{7}{2}, 4, 5, 6, 7, 8, 10, 12, 14 \right\}$$

Subnormal numbers are in brown.



The set of all machine numbers with a given precision has little in common with the set of real numbers. It resembles more to a finite subset of integers.

---

Machine numbers (2/3)

Denote the set of machine numbers by  $F$ .

The set  $F$  has the largest and the smallest positive elements as follows:

precision	max. no.	min. pos. normalized	min. pos. subnormal
single	$(2 - 2^{-23}) \cdot 2^{127}$ $\approx 3.4 \cdot 10^{38}$	$2^{-126}$ $\approx 1.2 \cdot 10^{-38}$	$2^{-126-23} = 2^{-149}$ $\approx 1.4 \cdot 10^{-45}$
double	$(2 - 2^{-52}) \cdot 2^{1023}$ $\approx 1.8 \cdot 10^{308}$	$2^{-1022}$ $\approx 2.2 \cdot 10^{-308}$	$2^{-1022-52} = 2^{-1074}$ $\approx 4.9 \cdot 10^{324}$

---

Machine numbers (3/3)

$F$  is characterized by the **machine epsilon**  $\epsilon_F$ , which is the difference between 1.0 and the smallest number in  $F$  larger than 1.

For single precision we have  $\epsilon_F = 2^{-23}$ , for double  $2^{-52}$ .

**Proposition 1.** *The distance between any two neighboring normalized numbers in  $F$  is at least  $\frac{\epsilon_F}{2}$  and at most  $\epsilon_F$ .*

---

Let  $fl : \mathbb{R} \rightarrow F$  be the mapping which assigns to any  $x \in \mathbb{R}$  the closest machine number.

Representation  
of real numbers  
(1/3)

The “closest” is given by the method chosen: rounding (“ties to even”), chopping (rounding towards 0),...

When trying to represent a number which is out of the representable range, an **overflow** or **underflow** is returned.

**Definition 2.** *Let a number  $\alpha$  be an approximate value of a number  $a$ .*

- The **absolute error** is the value  $|\alpha - a|$ .
- For  $a \neq 0$ , the **relative error** is  $\frac{|\alpha - a|}{|a|}$ .

---

In single precision, suppose that a number  $x \in \mathbb{R}$  lies in the normalized range, i.e.,

Representation  
of real numbers  
(2/3)

$$x = q \cdot 2^\ell, \quad \text{where } 1 \leq q < 2 \text{ and } -126 \leq \ell \leq 127.$$

What is the **error** due to the rounding or chopping when the closest machine number is chosen?

Let's **round towards 0**, i.e., chop off bits which do not fit into the significand (for positive numbers).

$$\text{If } x = (1.b_1b_2b_3b_3\dots)_2 \cdot 2^\ell \quad \text{then } fl(x) = (1.b_1b_2\dots b_{23}) \cdot 2^\ell.$$

The absolute error and the absolute errors are respectively:

$$|x - fl(x)| \leq 2^{-23+\ell} \quad \text{and} \quad \frac{|x - fl(x)|}{|x|} \leq \frac{2^{-23+\ell}}{q \cdot 2^\ell} \leq 2^{-23}.$$

---

Representation  
of real numbers  
(3/3)

The threshold of relative error is called the **unit roundoff error** and is denoted by **u**. Thus, in the single precision with chopping we have **u** =  $2^{-23}$ .

Attention, this number is sometimes called **machine epsilon**.

If we use **mathematical rounding**, we obtain **u** =  $2^{-24}$ .

**Proposition 3.** *Let  $x \in \mathbb{R}$  be greater than the smallest normalized number of  $F$  and smaller than the greatest normalized number of  $F$ . We have*

$$fl(x) = x(1 + \delta), \quad \text{where } |\delta| \leq \mathbf{u},$$

## Arithmetic operations

---

Arithmetic op-  
erations - error

**Proposition 4.** *Let  $x, y \in F$  and  $\odot$  be the operation of addition, multiplication or division. If there is no overflow or underflow, then we have*

$$fl(x \odot y) = (x \odot y)(1 + \delta), \quad \text{where } |\delta| \leq \mathbf{u},$$

In general: If we operate with more numbers, it is better to start with the smallest ones.

---

Arithmetic  
operations - a  
demonstration

Let  $f : \mathbb{R}^2 \mapsto \mathbb{R}$  be a mapping given by

$$f(x, y) = 333.75y^6 + x^2 \left( 11x^2y^2 - y^6 - 121y^4 - 2 \right) + 5.5y^8 + \frac{x}{2y}.$$

Let us evaluate  $f(77617, 33096)$ :

SageMath (precision 23 bits)	1.17260
SageMath (precision 24 bits)	$-6.33825 \cdot 10^{-29}$
SageMath (precision 53 bits)	$-1.18059162071741 \cdot 10^{21}$
SageMath (precision 54 bits)	$1.18059162071741 \cdot 10^{21}$
SageMath (precision 100 bits)	1.1726039400531786318588349045
SageMath (precision 121 bits)	1.17260394005317863185883490452018371
SageMath (precision 122 bits)	-0.827396059946821368141165095479816292

The exact solution is  $-\frac{54767}{66192} \approx -0.827396$ .

[S. M. Rump: *Algorithms for verified inclusions - theory and practice*, ..., 1988]

---

Loss of significant digits (1/3)

Errors while doing arithmetical operations can accumulate.

Big problems can be caused by the so-called **cancellation**.

Let us illustrate this on an example. Imagine that our computer calculates in basis 10 and uses 10 significant digits.

We want to evaluate  $x - \sin(x)$  for  $x = \frac{1}{15}$ .

$$\begin{aligned} x &\leftarrow 6.6666\ 66667 \cdot 10^{-2} \\ \sin(x) &\leftarrow 6.6617\ 29492 \cdot 10^{-2} \\ x - \sin(x) &\leftarrow 0.0049\ 37175 \cdot 10^{-2} \\ x - \sin(x) &\leftarrow 4.9371\ 75000 \cdot 10^{-5} \end{aligned}$$

The last 3 zeros are not *correct* significant digits.

Let us calculate the relative error.

---

Loss of significant digits (2/3)

$$\frac{\left| \frac{1}{15} - \sin\left(\frac{1}{15}\right) - fl\left(fl\left(\frac{1}{15}\right) - \sin\left(fl\left(\frac{1}{15}\right)\right)\right) \right|}{\left| \frac{1}{15} - \sin\left(\frac{1}{15}\right) \right|} \approx 1.4 \cdot 10^{-7}.$$

That is a lot in comparison to

$$\frac{|x - fl(x)|}{|x|} \leq 5 \cdot 10^{-10}.$$

**Proposition 5.** *Let  $x$  and  $y$  be normalized machine numbers and  $x > y > 0$ .*

*If  $2^{-p} \leq 1 - \frac{y}{x} \leq 2^{-q}$  for some positive integers  $p$  and  $q$ , then **at most**  $p$  and **at least**  $q$  significant binary bits are lost when performing the operation  $x - y$ .*

---

Loss of significant digits (3/3)

Cancellation can be avoided by using the following techniques:

- rationalizing the problem, i.e., using rational numbers and avoiding the subtraction in floating points arithmetics,
- using series expansions (such as Taylor series),
- using other identities, . . .

## Errors - conclusion

---

Errors - conclusion

Origins of errors:

- rounding errors and their accumulation,
- cancellation.

The errors on the inputs may also play an important role. Those errors are given by the origin of the input which may be the output of another calculation or a measurement.

A few final notes:

- increased precision may not lead to a more precise result,
- cancellation can be useful - it may cancel rounding errors,
- few operations with small numbers do not imply a small error.



One of the problems of machine numbers (IEEE-754) is in the ignorance of the created error.

There are some alternatives:

- Exact arithmetics:  $\mathbb{Z}$ ,  $\mathbb{Q}$  or  $GF(p)$  (it is not always possible or suitable).
- [Interval arithmetics](#) (we work with intervals instead of points). (IEEE 1788-2015).
- [Unum](#).

## Conditioning and stability of an algorithm

### Example

Consider two systems of linear equations with 2 unknowns:

$$\begin{pmatrix} 1 & 1/2 \\ 1/2 & 1/3 \end{pmatrix} \begin{pmatrix} x \\ y \end{pmatrix} = \begin{pmatrix} 3/2 \\ 1 \end{pmatrix} \quad \text{and} \quad \begin{pmatrix} 1 & 1/5 \\ 1/5 & -1 \end{pmatrix} \begin{pmatrix} x \\ y \end{pmatrix} = \begin{pmatrix} 3/2 \\ 1 \end{pmatrix}.$$

The solutions are

$$(x, y)^T = (0, 3)^T \quad \text{and} \quad (x, y)^T = (85/52, -35/52)^T \approx (1.6346, -0.67308)^T.$$

Let us try to simulate an error on the input, or during a calculation, by changing the right side to  $\begin{pmatrix} 3/2 \\ 5/6 \end{pmatrix}$ .

$$\begin{pmatrix} 1 & 1/2 \\ 1/2 & 1/3 \end{pmatrix} \begin{pmatrix} x \\ y \end{pmatrix} = \begin{pmatrix} 3/2 \\ 5/6 \end{pmatrix} \quad \text{and} \quad \begin{pmatrix} 1 & 1/5 \\ 1/5 & -1 \end{pmatrix} \begin{pmatrix} x \\ y \end{pmatrix} = \begin{pmatrix} 3/2 \\ 5/6 \end{pmatrix}.$$

The solutions change to

$$(x, y)^T = (1, 1)^T \quad \text{and} \quad (x, y)^T = (125/78, -20/39)^T \approx (1.6026, -0.51282)^T.$$

The change in the right side was

$$\begin{pmatrix} 3/2 \\ 1 \end{pmatrix} - \begin{pmatrix} 3/2 \\ 5/6 \end{pmatrix} = \begin{pmatrix} 0 \\ 1/6 \end{pmatrix},$$

a vector of Euclidean length  $1/6$  (the relative error is 0.09).

The change in the solution of **the first equation** was

$$\begin{pmatrix} 0 \\ 3 \end{pmatrix} - \begin{pmatrix} 1 \\ 1 \end{pmatrix} = \begin{pmatrix} -1 \\ 2 \end{pmatrix}$$

---

Example:  
system of linear  
equations (1/2)

---

Example:  
system of linear  
equations (2/2)

(the relative error is 0.75) and the one in the solution of **the second equation**

$$\begin{pmatrix} 85/52 \\ -35/52 \end{pmatrix} - \begin{pmatrix} 125/78 \\ -20/39 \end{pmatrix} = \begin{pmatrix} 5/156 \\ -25/156 \end{pmatrix}$$

(the relative error is 0.09).

Why is it that the first system is more sensitive to this change? Why are the two relative errors so different?

### Forward and backward error

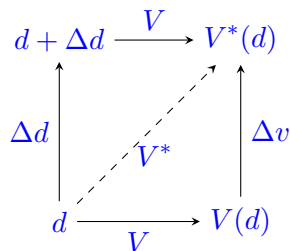
---

Forward and backward error

Let  $V$  be a numerical algorithm whose theoretical (accurate) output is denoted by  $V^*(d)$  where  $d$  is the input.

The result in the finite arithmetic is denoted  $V(d)$ . Furthermore, denote the so-called **forward error** by  $\Delta v := V^*(d) - V(d)$ .

The least (in a norm) number  $\Delta d$  such that  $V(d + \Delta d) = V^*(d)$  is the **backward error**.



If for all considerable inputs  $d$  the backward error is relatively small, we say that the algorithm is **backward stable**. “Small” depends again on the context.

## Conditioning

---

 Conditioning

The **conditioning** of a problem expresses the dependence of the output on the inputs - given a little perturbation  $\delta d$  of the input, we look how the output changes.

The **relative condition number** of a problem is

$$C_r = \lim_{\varepsilon \rightarrow 0^+} \sup_{\substack{d + \delta d \in D \\ \|\delta d\| \leq \varepsilon}} \frac{\|V(d + \delta d) - V(d)\|}{\frac{\|V(d)\|}{\|d\|}},$$

where  $D$  is the domain of  $V$ .

If  $C_r \approx 1$ , then we say that the problem is **well-conditioned**.

If it is large, we say the problem is **ill-conditioned**.

## Direct and iterative methods

### Directive methods

---

 Direct methods

A **direct method** calculates a solution of a problem in finitely many steps such that in absolute theoretical precision it gives the exact solution.

### Iterative methods

---

 Idea of iterative methods

**Iterative methods** look for approximate solutions to mathematical problems by constructing a sequence of approximate solutions:

$$x_0, x_1, x_2, \dots$$

Every following (approximate) solution is derived from the previous:

$$x_k = T(x_{k-1}),$$

for  $k > 0$  and some mapping  $T$ .

The mapping  $T$  is chosen so that the sequence  $(x_i)$  has a limit which is the (exact) solution of the problem.

If  $T$  is the same for all  $k$ , the method is called **stationary**.

## Systems of linear equations

### Notation

---

System of linear equations

We want to solve a system of  $n$  linear equations. We write the system in matrix representation

$$Ax = b,$$

where  $A \in \mathbb{R}^{n,n}$  is regular and  $b \in \mathbb{R}^{n,1}$ .

This is often a partial subproblem of a larger problem.

---

Norm - reminder

A **norm** on a vector space  $V$  is a mapping  $\|\cdot\| : V \mapsto \mathbb{R}_0^+$  which satisfies

1.  $\|x\| = 0 \Rightarrow x = 0$ ,
2.  $\|\alpha x\| = |\alpha| \cdot \|x\|$ ,
3.  $\|x + y\| \leq \|x\| + \|y\|$  (triangle inequality),

for all  $x, y \in V$  and all scalars  $\alpha$ .

On  $\mathbb{R}^n$  (or  $\mathbb{C}^n$ ) the most used norm is probably the **Euclidean norm**:

$$\|x\| = \left( \sum_{i=1}^n x_i^2 \right)^{\frac{1}{2}},$$

where  $x = (x_1, x_2, \dots, x_n) \in \mathbb{R}^n$ .

Other commonly used norms include

- $\|x\|_\infty = \max \{|x_i| : i \in \{1, \dots, n\}\}$  **maximum norm**,

- $\|x\|_1 = \sum_{i=1}^n |x_i|$  taxicab or  $L_1$  norm.

---

Matrix norm

Given a vector norm  $\|\cdot\|$ , we define the **induced matrix norm** of the matrix  $A \in \mathbb{C}^{n,n}$  as follows

$$\|A\| = \sup \{ \|Ax\| : x \in \mathbb{C}^{n,1} \text{ and } \|x\| = 1 \}.$$

Such norm satisfies

- $\|I\| = 1$ ,
- $\|Ax\| \leq \|A\| \cdot \|x\|$  (norm consistency),
- $\|AB\| \leq \|A\| \cdot \|B\|$ .

## Conditioning of the problem

---

Conditioning  
of the problem  
(1/2)

Let us see the conditioning of  $Ax = b$ . We suppose that the right side  $b$  is the input of the problem, and  $x$  is the output.

Given a small perturbation  $\delta x$  we have:

$$A(x + \delta x) = Ax + A\delta x = b + \delta b,$$

where  $A\delta x = \delta b$ .

We have  $\|b\| = \|Ax\| \leq \|A\| \cdot \|x\|$ , which implies  $\frac{1}{\|x\|} \leq \frac{\|A\|}{\|b\|}$ .

Furthermore,  $\|\delta x\| = \|A^{-1}\delta b\| \leq \|A^{-1}\| \cdot \|\delta b\|$ .

Finally,

$$\frac{\|\delta x\|}{\|x\|} \leq \|A\| \cdot \|A^{-1}\| \frac{\|\delta b\|}{\|b\|},$$

---

Conditioning  
of the problem  
(2/2)

$$\frac{\|\delta x\|}{\|x\|} \leq (\|A\| \cdot \|A^{-1}\|) \frac{\|\delta b\|}{\|b\|}$$

The number  $\kappa(A) = \|A\| \cdot \|A^{-1}\|$  is the **condition number** of the matrix  $A$ .

The above inequality reads: the relative error of the results is less than the relative error of the input times the condition number.

The greater  $\kappa(A)$  is, the more ill-conditioned the problem is.

Note that  $b$  may contain an error coming from its origin, for instance a measurement.

Of course, the condition number depends on the chosen norm.

---

Let us revisit the example we saw earlier:

$$A = \begin{pmatrix} 1 & 1/2 \\ 1/2 & 1/3 \end{pmatrix} \quad \text{and} \quad B = \begin{pmatrix} 1 & 1/5 \\ 1/5 & -1 \end{pmatrix},$$

The inverses are

$$A^{-1} = \begin{pmatrix} 4 & -6 \\ -6 & 12 \end{pmatrix} \quad \text{and} \quad B^{-1} \approx \begin{pmatrix} 0.961538 & 0.192308 \\ 0.192308 & -0.961538 \end{pmatrix},$$

To calculate the condition number  $\kappa(A) = \|A\| \cdot \|A^{-1}\|$  we use the norm  $\|A\|_\infty$ :

$$\kappa(A) = \|A\|_\infty \cdot \|A^{-1}\|_\infty = \frac{3}{2} \cdot 18 = 27 \quad \text{and} \quad \kappa(B) = \frac{18}{13} \approx 1.3846056.$$

The problem with the matrix  $A$  is significantly more **ill-conditioned** than with the matrix  $B$ . This is in accordance with our previous observations.

Example of  
two sets of  
linear equations  
revisited

## Description of the iterative method

We will construct a sequence of vectors  $x_0, x_1, x_2, \dots$  which will approximate the solution of  $Ax = b$ .

Basic iterative  
methods for  
 $Ax = b$

The vector  $x_0$  is chosen randomly.

We choose a regular matrix  $Q$  and the following terms will be calculated as

$$Qx_k = (Q - A)x_{k-1} + b$$

for all  $k > 0$ .

**The idea:** we choose the matrix  $Q$  so that the sequence  $(x_k)$  converges to some  $x^*$ . Then,

$$Qx^* = (Q - A)x^* + b$$

and thus

$$Ax^* = b.$$

Convergence  
choice of  $Q$

We use the equality  $x_k = Q^{-1}((Q - A)x_{k-1} + b)$  in

$$\begin{aligned} x_k - x &= Q^{-1}((Q - A)x_{k-1} + b) - x \\ &= (I - Q^{-1}A)x_{k-1} - x + Q^{-1}b \\ &= (I - Q^{-1}A)x_{k-1} - (I - Q^{-1}A)x \\ &= (I - Q^{-1}A)(x_{k-1} - x), \end{aligned}$$

where  $x$  is the exact solution satisfying  $Ax = b$ .

Denote  $W = I - Q^{-1}A$ .

We denote the **error vector**  $e_k = x_k - x$ . We have  $e_k = We_{k-1}$ .

The vector  $e_k$  will be “smaller” than  $e_{k-1}$  if  $W$  is “small”.  
“Small” can be determined using norms.

Since  $e_k = W^k e_0$ , to lower the error at each step we need to have  $\lim_{k \rightarrow +\infty} W^k = 0$ .



## Convergence

---

Convergence vs.  
spectral radius

The **Spectral radius** of a matrix  $M$  is the number  $\rho(M)$  defined as the greatest eigenvalues (in absolute value), i.e.,

$$\rho(M) = \max\{|\lambda| : \lambda \text{ is an eigenvalue of } M\},$$

**Theorem 6.** *If  $M \in \mathbb{C}^{n,n}$ , then*

$$\lim_{k \rightarrow +\infty} M^k = 0 \Leftrightarrow \rho(M) < 1,$$

Thus, in our case, the method converges **if and only if**

$$\rho(W) < 1,$$

i.e., all the eigenvalues of the matrix  $W = I - Q^{-1}A$  are in absolute value less than 1.

---

Speed of conver-  
gence of  $e_k$

How fast is the error vector  $e_k$  converging to 0?

We have

$$e_k = W^k e_0.$$

We estimate in norm

$$\|e_k\| = \|W^k e_0\| \leq \|W^k\| \cdot \|e_0\| \leq \|W\|^k \cdot \|e_0\|.$$

The condition of convergence  $\rho(W) < 1$  does not imply anything on the speed from the previous estimate.

However, the estimate on the right side is strictly decreasing if  $\|W\| < 1$ .

---

When to stop?  
(1/2)

The iterative method is stopped in the step  $k$  if  $x_k$  reaches some desired precision.

The desired precision is given by the nature of the problem.

In the case  $\|W\| < 1$ , we know that the sequence  $(\|e_k\|)_k$  is strictly decreasing and we may stop iterating when

$$\|e_k - e_{k-1}\| < \varepsilon,$$

where  $\varepsilon$  is a constant supplied by the user.

This is impractical since we do not have the exact solution.

In the step  $k$  we can calculate the so-called **residue**  $Ax_k - b$  and the **convergence criterion** can be set to

$$\|Ax_k - b\| < \varepsilon.$$

---

When to stop?  
(2/2)

Instead of calculating the residues, one may use a more efficient criterion

$$\|x_{k+1} - x_k\| < \varepsilon.$$

We have

$$\begin{aligned} \|e_k\| &= \|x_k - x\| = \|x_k - x_{k+1} + x_{k+1} - x\| \\ &\leq \|x_k - x_{k+1}\| + \underbrace{\|x_{k+1} - x\|}_{=e_{k+1}} \\ &< \varepsilon + \|W\| \cdot \|e_k\|, \end{aligned}$$

where, supposing  $\|W\| < 1$ , the last inequality gives

$$\|e_k\| < \frac{\varepsilon}{1 - \|W\|}.$$

Thus, this criterion can be effectively used if  $\|W\| < 1$ , but not too close to 1.

---

Finite precision  
calculations

All ideas so far were made in the theoretical absolute precision.

In a finite precision, the method may not converge even if  $\|W\| < 1$  due to rounding errors.

However, an advantage of iterative methods in a finite precision arithmetic is that at each step the rounding errors from the previous step are “forgotten”. We start the new iteration with a better approximate solution.

In finite arithmetic the method can diverge even if the problem is not ill-conditioned.

Thus, in practice, we need another parameter of the method - a maximum number of iterations. If we reach this number of iterations without satisfying a convergence criterion, then the method outputs failure.

## Concrete algorithms

---

Choices of  $Q$

Denote by  $a_{i,j}$  the entries of the matrix  $A$  and denote

$$L = \begin{pmatrix} 0 & 0 & \cdots & 0 \\ a_{2,1} & 0 & \cdots & 0 \\ \vdots & \ddots & \ddots & \vdots \\ a_{n,1} & \cdots & a_{n,n-1} & 0 \end{pmatrix} \text{ and } D = \begin{pmatrix} a_{1,1} & 0 & \cdots & 0 \\ 0 & a_{2,2} & \ddots & \vdots \\ \vdots & \ddots & \ddots & 0 \\ 0 & \cdots & 0 & a_{n,n} \end{pmatrix}.$$

Denote  $U$  so that  $A = L + D + U$ .

We will mention the following choices of  $Q$ :

- Richardson method  $Q = I$ ,
- Jacobi method  $Q = D$ ,
- successive overrelaxation / SOR method  $Q = \frac{1}{\omega}D + L$ .

---

Richardson method

Set  $Q = I$ .

The recurrence relation is given by

$$x_k = (I - A)x_{k-1} + b$$

The convergence is for a narrow class of matrices:  $A$  must be close to  $I$  so that

$$\|I - A\| < 1.$$

---

Jacobi method

Set  $Q = D$ .

The recurrence relation is given by

$$Dx_k = (D - A)x_{k-1} + b = -(L + U)x_{k-1} + b.$$

**Proposition 7.** *If the matrix  $A$  is diagonally dominant, then the Jacobi method converges for any choice of  $x_0$ .*

A matrix is **diagonally dominant** if, for each row, the sum of the absolute values of all the entries except the one on the diagonal is less than the absolute value of the entry on the diagonal.

---

SOR method

Set  $Q = \frac{1}{\omega}D + L$ , where  $\omega \in \mathbb{R} \setminus \{0\}$ .

The recurrence relation is given by

$$\left(\frac{1}{\omega}D + L\right)x_k = \left(\frac{1}{\omega}D + L - A\right)x_{k-1} + b = \left(\left(-1 + \frac{1}{\omega}\right)D - U\right)x_{k-1} + b.$$

**Proposition 8.** *For  $0 < \omega < 2$  the SOR method converges if  $A$  is symmetric, positive definite and has positive diagonal entries.*

The parameter  $\omega$  is used to speed up the convergence.

The choice  $\omega = 1$  is the *Gauss-Seidel* method.

---

Algorithm

**Inputs:** matrices  $A, Q$ , vector  $b$ , precision  $\varepsilon$ , maximum number of iterations  $K$ .

1. choose  $\hat{x}_0$  at random
2. for  $k$  from 1 to  $K$  do
  - 2.1  $\hat{x}_{k+1} = Q^{-1}(Q - A)\hat{x}_k + Q^{-1}b$
  - 2.2 if  $\|A\hat{x}_k - b\| < \varepsilon$ , **return**  $\hat{x}_k$  (or in general if any convergence criterion is satisfied)
3. **return** “no solution found after  $K$  steps”.

---

Demonstration -  
Jacobi method  
(1/2)

Let  $A = \begin{pmatrix} 2 & 1 \\ 1 & 4 \end{pmatrix}$ .  $\|I - D^{-1}A\| = \frac{1}{2}$ .

We use the Jacobi method to calculate a solution for  $b = (3, 5)^T$ .  
The exact solution is  $(1, 1)^T$ .

The convergence criterion used is  $\|A\hat{x}_k - b\| < 10^{-2}$ .

$k$	$\hat{x}_k$	$\ A\hat{x}_k - b\ $
0	(0.5, 1.5)	1.58113883008
1	(0.75, 1.125)	0.450693909433
2	(0.9375, 1.0625)	0.197642353761
3	(0.96875, 1.015625)	0.0563367386791
4	(0.9921875, 1.0078125)	0.0247052942201
5	(0.99609375, 1.001953125)	0.00704209233489

---

Demonstration -  
Jacobi method  
(2/2)

...the same problem but with a different  $\hat{x}_0$ , which is further from the exact solution.

$k$	$\hat{x}_k$	$\ A\hat{x}_k - b\ $
0	(-10, 10)	28.1780056072
1	(-3.5, 3.75)	9.01734439844
2	(-0.375, 2.125)	3.5222507009
3	(0.4375, 1.34375)	1.1271680498
4	(0.828125, 1.140625)	0.440281337613
5	(0.9296875, 1.04296875)	0.140896006226
6	(0.978515625, 1.017578125)	0.0550351672016
7	(0.9912109375, 1.00537109375)	0.0176120007782
8	(0.997314453125, 1.002197265625)	0.0068793959002