BLOCK COURSE

Introduction to Numerical Methods for Computer Simulation

Roland Pulch, Andreas Bartel

Manuscript Winterterm 2005/06,

Bergische Universität Wuppertal, Applied Mathematics / Numerical Analysis

Contents:

Literature:

Stoer, J.; Bulirsch, R.: Introduction to Numerical Analysis, Springer.

Quateroni, A.; Sacco, R.; Saleri, F.: Numerical Mathematics, Springer.

Chapter 0

Introduction

Scientific Computing: Interdisciplinary subject consisting in mathematics, applied sciences and information technology.

Examples for applications: reaction kinetics, circuit simulation, multibody systems, fluid dynamics, financial products, etc.

Basis and core is *Simulation*. That needs: Modeling, Analysis, Algorithms, Implementation, Visualization.

Mathematical part is *Numerical Analysis*: deals with the development and investigation of computable models, and algorithms to compute the according numerical solution.

Using computers, one has to be careful: e.g. finite amount of accuracy, and error occurring in each step may falsify the solution. To underline this, we investigate the following example.

Example 0.1 (Integral recursion) We want to compute the integrals

$$I_n = \frac{1}{e} \int_0^1 x^n e^x \, \mathrm{d}x \quad \text{for} \quad n = 0, 1, 2, \dots \ .$$

Elementary integration fails for $n > 0$. Obviously, we have the bounds

$$0 < I_n < 1 \quad \text{for all } n.$$

Using integration by parts, we can deduce the following:

$$\begin{aligned} I_n &= \frac{1}{e}\int_0^1 x^n e^x \,\mathrm{d}x \\ &= \frac{1}{e}\left(x^n e^x\Big|_0^1 - \int_0^1 n x^{n-1} e^x \,\mathrm{d}x\right) \\ &= \frac{1}{e}\left(e - n\int_0^1 x^{n-1} e^x \,\mathrm{d}x\right) \\ &= 1 - nI_{n-1} \end{aligned}$$

Result: *2-Term-Recursion*

$$I_n = 1 - nI_{n-1} \tag{0.1}$$

Given I_0, we can compute from (0.1) any integral I_n for $n > 0$.

Start value: $I_0 = \frac{1}{e}\int_0^1 e^x \,\mathrm{d}x = \frac{e-1}{e} \doteq 0.632120558\ldots$

Pocket calculator or computer (using MATLAB):

$I_0 = 0.6321...,\quad \ldots\quad ,\ I_{29} = 1.0989...\cdot 10^{14},\ I_{30} = -3.2968...\cdot 10^{15}$?????

Can the numerical computer results be correct?

A variant: instead of *forward recursion* (0.1), which behaves awkwardly, one could try *backward recursion*:

$$I_{n-1} = \tfrac{1}{n}(1 - I_n). \tag{0.2}$$

As start value, we may pick: $I_{30} = \frac{1}{2}$ (wrong!) and compute I_0 (backward).

Surprisingly, result: $I_0 = 0.632120558\ldots$, correct in all digits!

Explanation (pre-fetch):

forward recursion:	errors are tremendously amplified $\rightarrow$ *ill-conditioned*
backwark recursion:	errors are strongly damped, i.e. errors cancel out $\rightarrow$ *well-conditioned*

$\diamond$

These phenomena are studied in the next chapter (before we explain the basic concepts and algorithms in numerics).

Chapter 1

Error Analysis

One of the *main tasks* in numerical mathematics is the *assessment of accuracy*, i.e. to investigate the quality of results produced by an algorithm.

Reasons:

- to guarantee a certain amount of accuracy in the resulting values.
- to give a correct interpretation of the results.
- to recognise the limitations of on algorithm.

During the whole procedure from modelling to computer simulation, errors occur. We can distinguish:

1. modelling error	simplification in physical problem (neglect of effects in setup of mathematical description)
2. error in input data	measurements
3. approximation error	mathematical model vs. computer model (e.g. continuous problem $\rightarrow$ discrete problem)
4. roundoff error	finite grid of machine numbers

1.1 Number Representation: Floating-Point and Roundoff

Fundamental difference:

- real numbers $\mathbb{R}$: without gap, unbounded, infinite
- machine representation: finite number of distinct values
  realization in digital computer: via digits

  power expansion to basis B; usual $B = 10$ (decimal), or $B = 2$ (dual)
  $19.25 = 1 \cdot 10^1 + 9 \cdot 10^0 + 2 \cdot 10^{-1} + 5 \cdot 10^{-2} = 1925 \cdot 10^{-2} = 0.1925 \cdot 10^2$

  $\rightsquigarrow$ error in transition (real to machine)

To formalise, we introduce two sets: *floating point numbers* $\mathbb{G}$ and *machine numbers* $\mathbb{M}$:

Definition 1.1 (Floating point) *We form the set $\mathbb{G}$ of normalised floating point numbers with t-significant digits for basis B as follows:*

$$\mathbb{G} := \left\{ g = M \cdot B^E : \ M, E \in \mathbb{Z} \ \text{and} \ M = 0 \ \text{or} \ B^{t-1} \leq |M| < B^t \right\}$$

using exponent E and mantissa M (M and E integers), where $B > 1$ and $t > 0$ are natural numbers. □

The property $B^{t-1} \leq |M| < B^t$ implies a *normalised* representation, where no leading zeros occur in the used digits. Furthermore, this condition implies a unique representation for each floating point number via mantissa and exponent, i.e. the map $\gamma : \mathbb{G} \to \mathbb{Z}^2, \ g \mapsto M, E$ becomes bijective.

Example 1.1 (Normalised Floats – decimal)
Firstly, we consider the set of decimal floats with 4-digits

$$\mathbb{G} = \left\{ M \cdot 10^E : \ M = 0 \ \text{or} \ 10^3 \leq |M| < 10^4 \right\}.$$

Hence we have for the mantissa: $1000 \leq |M| \leq 9999$.

Examples for this representation:

$$19.25 = 1925 \cdot 10^{-2}, \qquad 0.004589 = 4589 \cdot 10^{-6}, \quad \ldots$$ ◇

Example 1.2 (Normalised floats – dual)
Secondly, 7-digit, normalised set of duals:

$$\mathbb{G} = \left\{ M \cdot 2^E : \ M = 0 \ \text{ or } \ 2^6 \leq |M| < 2^7 \right\}.$$

We transform 19.25 to dual system: notationally $L \equiv 1$

$$\begin{aligned} 19.25 &= 1 \cdot 2^4 + 0 \cdot 2^3 + 0 \cdot 2^2 + 1 \cdot 2^1 + 1 \cdot 2^0 + 0 \cdot 2^{-1} + 1 \cdot 2^{-2} \\ &= L00LL.0L = L00LL0L \cdot 2^{-L0} \end{aligned}$$ ◇

Next, *machine numbers*: Subset of floats, where the exponent is bounded.

Definition 1.2 (Machine numbers) *The set* $\mathbb{M}$ *of machine numbers is defined by*

$$\mathbb{M} := \left\{ g = M \cdot B^E \in \mathbb{G} : \ \alpha \leq E \leq \beta \right\}.$$

for given integers $\alpha \leq \beta$. □

Consequences:

- $\mathbb{M}$ is finite.
- Quantities B, t, α, β are parameters fixed by implementation.
- Mantissa M and exponent E define (uniquely) the machine number.
- symmetry of $\mathbb{M}$ w.r.t. $g = 0$, i.e. $g \in \mathbb{M} \Leftrightarrow -g \in \mathbb{M}$ (same for $\mathbb{G}$).

Characteristics for a given set of machine numbers:

$$\begin{aligned} \sigma &:= B^{t-1} \cdot B^{\alpha} \quad \text{smallest positive machine number} \\ \lambda &:= (B^t - 1) \cdot B^{\beta} \quad \text{largest machine number} \end{aligned}$$

Exponent overflow / underflow occurs, if computed values exceed the set

$$[-\lambda, -\sigma] \cup \{0\} \cup [+\sigma, +\lambda].$$

Figure 1: Density of normalised machine numbers.

Resolution:

- machine numbers not equally distributed.
- relatively large gap occurs at 0.
- density reduces farther away from zero.

Formally, two succeeding floats $g = M \cdot B^E$ and $g' = (M+1) \cdot B^E$ exhibit the relative distance $(g' - g)/g = 1/M$. The largest relative distance

$$\rho := \max\{1/M : B^{t-1} \leq |M| < B^t\} = 1/B^{t-1} = B^{1-t}$$

is referred to as *resolution*.

IEEE-Standard 754

Coding of M and E by IEEE-standard since 1984; Basis is $B = 2$ (dual). There are several number formats: (a selection!)

format	bits sign	bits exponent	bits mantissa
4 byte - short real (single precision)	1	8	23
8 byte - long real (double precision)	1	11	52
10 byte - temp real (extended precision)	1	15	64

Hidden bit: Since all normalised dual representations have a leading L, we obtain an additional digit for the resolution!

Characteristics for these arithmetics:

format	σ	λ	ρ
short real	$1.2 \cdot 10^{-38}$	$3.4 \cdot 10^{38}$	$1.2 \cdot 10^{-7}$
long real	$2.2 \cdot 10^{-308}$	$1.8 \cdot 10^{308}$	$2.2 \cdot 10^{-16}$
temp real	$3.4 \cdot 10^{-4932}$	$1.2 \cdot 10^{4932}$	$1.1 \cdot 10^{-19}$

Example 1.3 (Short real)
Here we have 1 sign digit, 1 byte = 8 bit for exponent, and 3 byte = 1 + 23 bit (including the hidden bit) for mantissa.

Schematic storage: $\boxed{\pm}\boxed{a_1\, a_2,\, \ldots, a_8}\boxed{b_1, b_2, \ldots, b_{23}}$ (sign, exponent, mantissa). ◇

Specials in IEEE-Standard:

$\pm\infty$	larger/smaller than any real	devision by zero
`NaN`	(Not a Number), undefined, passed on	invalid operation ($\infty - \infty$, $0/0$)

Rounding

Since the set of machine numbers is finite, we cannot represent all real numbers exactly. Therefore we need to approximate the continuum $\mathbb{R}$. How? This problem occurs not only while reading data, also while computing. Generally, exact results of elementary operations $(+, -, \cdot, /)$ on $\mathbb{G}^2$ do not belong to $\mathbb{G}$.

The rounding function rd does the job:

Definition 1.3 (Round to nearest) *Correct rounding to nearest is the map* $\mathrm{rd} : \mathbb{R} \to \mathbb{G}$*, which assigns to a real number* x *a floating point number* g *such that*

$$|\mathrm{rd}(x) - x| \leq |g - x| \quad \textit{for all } g \in \mathbb{G}\,. \qquad \square$$

Rounding rd is surjective, idempotent and monotone.

For fixed x, let g_L denote the largest float, which is smaller or equal to x (left neighbour) and g_R smallest float, which is larger than x (right neighbour).

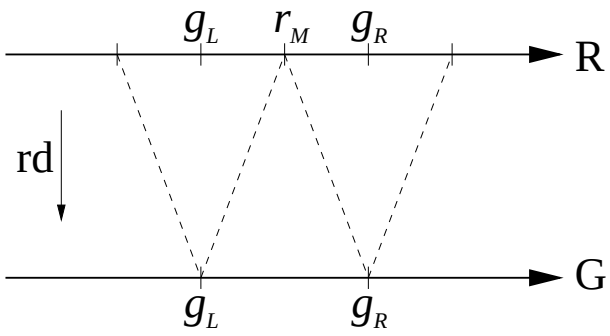


Figure 2: Sketch of rounding procedure.

Furthermore, we define the center point $r_M = \frac{1}{2}(g_L + g_R)$. Thus

$$\mathrm{rd}(x) = \begin{cases} g_L & \text{if } x \leq r_M \\ g_R & \text{if } x \geq r_M \end{cases}$$

In the case $x = r_M$, the correct round is not uniquely defined. IEEE-Standard puts:

$$\mathrm{rd}(r_M) = \begin{cases} g_L & \text{if } g_L \text{ has even mantissa} \\ g_R & \text{if } g_L \text{ has odd mantissa} \end{cases}$$

For $B = 2$, this condition is equivalent to the question, if the last bit of g_L is equal to zero or one.

Further rounding modes: (let $x \in \mathbb{R}$)

round down:	$x \mapsto g_L$			
round up:	$x \mapsto g_R$			
round towards zero:	$x \mapsto g_L$,	if $x \geq 0$,	$x \mapsto g_R$	if $x < 0$
round away from zero:	$x \mapsto g_R$,	if $x \geq 0$,	$x \mapsto g_L$	if $x < 0$

Rounding causes errors! We define

$$\Delta x := |\mathrm{rd}(x) - x| \qquad \textit{absolute error}$$

and

$$\varepsilon_x := \frac{\Delta x}{|x|} = \frac{|\mathrm{rd}(x) - x|}{|x|} \qquad \textit{relative error}.$$

For the absolute error, we find

$$|\mathrm{rd}(x) - x| \leq \tfrac{1}{2}|g_R - g_L| = \tfrac{1}{2}|M \cdot B^E - (M+1) \cdot B^E| = \tfrac{1}{2}B^E$$

and for the relative error

$$\frac{|\mathrm{rd}(x) - x|}{|x|} \leq \frac{B^E}{2|M|B^E} = \frac{1}{2|M|} \leq \tfrac{1}{2}B^{1-t}\,.$$

Thus the relative rounding error is bounded (from above) by half of the resolution (of the arithmetic). Formally, it holds

$$\mathrm{rd}(x) = x(1+\varepsilon) \quad \text{with} \quad |\varepsilon| \leq \tfrac{1}{2}B^{1-t}, \tag{1.1}$$

where ε depends on x.

Definition 1.4 (Machine precision) *Using t significant digits, the machine precision is defined as*

$$\varepsilon_0 := \tfrac{1}{2}B^{1-t}. \qquad \square$$

Thus the machine precision is half times the resolution.

Integral recursion (Ex. 0.1) revisited

Under the above perspective, we investigate Ex. 0.1 again:

$$\text{forward recursion:} \quad I_n = 1 - nI_{n-1} \quad \text{for} \quad I_n = \tfrac{1}{e}\textstyle\int_0^1 x^n e^x \,\mathrm{d}x.$$

We perform a simple analysis: the start value $I_0 = \frac{1}{e}\int_0^1 e^x \,\mathrm{d}x = \frac{e-1}{e}$ is transcendent (i.e., only an infinite number of digits define the value exactly). In our computation, we used $\tilde{I}_0 = \mathrm{rd}(I_0)$.

How does the input error $\Delta I_0 = \tilde{I}_0 - I_0$ propagates in this recursion?

For simplicity, we assume that the recursion is computed in exact arithmetic, i.e., without further roundoff errors:

$$\tilde{I}_k = 1 - k \cdot \tilde{I}_{k-1} \qquad \text{for } \; k = 1, 2, \ldots \; .$$

Thus in the first two steps, we deduce the errors

$$\begin{aligned}\Delta I_1 &= \tilde{I}_1 - I_1 = (1 - \tilde{I}_0) - (1 - I_0) = (-1)\,\Delta I_0 \\ \Delta I_2 &= \tilde{I}_2 - I_2 = (1 - 2\tilde{I}_1) - (1 - 2I_1) = -2\Delta I_1 = (-2)(-1)\Delta I_0.\end{aligned}$$

By induction, it follows

$$\Delta I_n = \tilde{I}_n - I_n = (-1)^n\, n!\, \Delta I_0.$$

Hence the input error ΔI_0 is amplified by a factor of $n!$ and oscillates. This explains the results in Ex. 0.1 very well. In addition, one obtains that the error in the backward recursion is damps by $\frac{1}{n!}$.

Floating point operations

As simplification, we assumed in the analysis of the integral recursion that the basic operations were performed without error. In practice, this is not the case. For the elementary operations $\odot \in \{+, -, \cdot, /\}$, we have

$$a,\, b \in \mathbb{G} \;\not\Rightarrow\; a \odot b \in \mathbb{G}$$

i.e., we need to round the result.

Following Wilkinson (1958), one denotes by $a \odot_{\mathrm{fl}} b$ the result of floating point operations and demands for an *ideal arithmetic*:

$$a \odot_{\mathrm{fl}} b = \mathrm{rd}(a \odot b) \quad \text{for } a, b \in \mathbb{G} \tag{1.2}$$

(i.e., floating point operations shall yield correct rounding of exact result). This is *IEEE-Standard 754* and it is realised in todays microprocessors.

From (1.2), we obtain

$$a \odot_{\mathrm{fl}} b = (a \odot b)(1 + \varepsilon), \quad \text{where } |\varepsilon| \le \varepsilon_0\,. \tag{1.3}$$

One refers to (1.3) as *strong hypothesis* of roundoff errors; it is fulfilled by any microprocessor with ideal arithmetic.

Property (1.3) is the basis for error analysis: here ε depends on arguments $a, b \in \mathbb{G}$, but the bound ε_0 is known a priori, and enables estimates.

Remarks:

- In analysing floating point operations, we neglect overflow and underflow.
- Correct rounding arithmetic is obtained using a computation based on $t + 3$ digits. (Two additional guard bits plus one sticky bit.)
- The elementary operations are implemented as microprograms.
  One multiplication is twice as expensive as an addition.

  Example addition (microprogram): choose $B = 2$, $t = 5$, and
  $3 = \texttt{L.L000} \cdot 2^0$ and $0.09375 = \texttt{L.L000} \cdot 2^{-4}$:

  i) exponent adaption, i.e., add two hidden bits and shift in mantissa by increase of smaller exponent: $\texttt{L.L000|00} \cdot 2^0$
  $\texttt{0.000L|L0} \cdot 2^0$

  ii) addition of mantissas: $\texttt{L.L000L|L0} \cdot 2^0$

  iii) normalise (if necessary: mantissa shift)

  iv) round down: $\texttt{L.L000L} \cdot 2^0$
  round up: $\texttt{L.L00L0} \cdot 2^0$
  i.e., result: $\texttt{L.L00L0} \cdot 2^0$

  Caution: By subtraction *cancellation* of leading digits may occur! (No rounding necessary!)

1.2 Roundoff Error Analysis

An algorithm (computer program) in numerics is a sequence of elementary operations, where the chronology is uniquely defined. In floating point arithmetic, roundoff errors alter and falsify any intermediate results and thus the final result.

The order of operations is crucial for error analysis, since floating point arithmetic is *not associative* and is *not distributive*: i.e.,

$$a +_{\mathrm{fl}} (b +_{\mathrm{fl}} c) \neq (a +_{\mathrm{fl}} b) +_{\mathrm{fl}} c, \qquad a \cdot_{\mathrm{fl}} (b \cdot_{\mathrm{fl}} c) \neq (a \cdot_{\mathrm{fl}} b) \cdot_{\mathrm{fl}} c,$$

$$a +_{\mathrm{fl}} (b \cdot_{\mathrm{fl}} c) \neq (a \cdot_{\mathrm{fl}} b) +_{\mathrm{fl}} (a \cdot_{\mathrm{fl}} c).$$

Forward- vs. Backward-Analysis

How to proceed for roundoff error analysis? *Forward analysis* is self-evident: that is, to compare floating point result with exact solution. Often this is very complicated. Therefore the situation is interchanged, that is, (roughly)

Definition 1.5 (Backward Analysis) *Let x denote input data, the exact result be denoted as $y(x)$ and the algorithm in floating point arithmetic yields $y^{\mathrm{fl}}(x)$. Now, backward analysis aims at the representation of the floating point result $y^{\mathrm{fl}}(x)$ as exact solution for (slightly) modified input data $\hat{x}$:*

$$y^{\mathrm{fl}}(x) = y(\hat{x})$$

(if possible). □

Backward analysis is standard.

Example 1.4 (Backward Analysis for $+_{\mathrm{fl}}/\cdot_{\mathrm{fl}}$)

$a +_{\mathrm{fl}} b = (a + b)(1 + \alpha)$, exact for arguments: $a(1 + \alpha), b(1 + \alpha)$,

$a \cdot_{\mathrm{fl}} b = (a \cdot b)(1 + \beta)$, exact for arguments: $a\sqrt{1 + \beta}, b\sqrt{1 + \beta}$. ◇

As an example, we investigate the roundoff error in the Horner-scheme.

Horner-Scheme

Task: Compute polynomial P at value x

$$P(x) = P(c_0, c_1, \ldots, c_n; x) = c_0 + c_1 x + c_2 x^2 + \ldots + c_n x^n$$

with given coefficients $c_0, \ldots, c_n$.

For a direct evaluation we need: n (ADD+MULT) plus all powers x^k.

Following Horner, the evaluation is computed as

$$P(x) = (\ldots((c_n x + c_{n-1}) \cdot x + c_{n-2}) \cdot x + \cdots + c_1) \cdot x + c_0\,, \qquad (1.4)$$

using n (ADD+MULT) – furthermore the risk of overflow is much smaller.

This reads:

Algorithm 1.1 (Horner-Scheme)
The result $y = P(x)$ is computed as:

$y := c_n$;
for $k = n-1 : -1 : 0$
    $y := y \cdot x + c_k$;
end; ◇

For the roundoff analysis, we assume coefficients c_i and value x to be exact. Using the strong hypothesis (1.3) for the values of floating point arithmetic:

$\tilde{y} := c_n$;
for $k = n-1 : -1 : 0$
    $\tilde{y} := ((\tilde{y} \cdot x) \cdot (1+\mu_k) + c_k)(1+\alpha_k)$;
end;

At the end, $\tilde{y} = \tilde{c}_0 + \tilde{c}_1 x + \ldots + \tilde{c}_n x^n$ with perturbed coefficients $\tilde{c}_k$

$$\tilde{c}_k := c_k \cdot (1+\alpha_k) \cdot (1+\mu_{k-1}) \cdot (1+\alpha_{k-1}) \cdot (1+\mu_{k-2}) \cdot \ldots \cdot (1+\alpha_0)$$

and $\alpha_n := 0$. Hence the value $\tilde{y}$ can be identified with the exact evaluation of a 'slightly' modified polynomial (at x).

The modification in coefficient $\tilde{c}_n$ is the largest. By linearisation, we can approximate the product of factors $1+\varepsilon$: using $|\alpha|, |\mu| \le \varepsilon_0$, we put

$$(1+\alpha) \cdot (1+\mu) \doteq 1 + \alpha + \mu$$

neglecting the product $\alpha \cdot \mu$ (small numbers!). Thus ($k > 1$)

$$\tilde{c}_n \doteq c_n \cdot (1 + \mu_{n-1} + \alpha_{n-1} + \ldots + \mu_0 + \alpha_0)$$

and for the absolute error in $\tilde{c}_n$

$$\Delta c_n = |\tilde{c}_n - c_n| \doteq |\mu_{n-1} + \alpha_{n-1} + \ldots + \mu_0 + \alpha_0||c_n| \leq 2n\varepsilon_0|c_n| \,.$$

And analogue for $\tilde{c}_k$: $\Delta c_k \dot{\leq} (2k+1)\varepsilon_0|c_k|$.

Thus we have found the relation

$$P^{\text{fl}}(c_0, c_1, \ldots, c_n; x) = P(\tilde{c}_0, \tilde{c}_1, \ldots, \tilde{c}_n; x).$$

In the view of backward analysis, the Horner-Scheme behaves well: with small degree n, the error is moderate.

There is also a strict bound (i.e., without linearisation):

$$\tilde{c}_k = c_k(1+\varepsilon) \quad \text{with} \quad |\varepsilon| \leq \frac{2k\varepsilon_0}{1 - 2k\varepsilon_0} \quad \text{for} \;\; k = 1, \ldots, n.$$

This roundoff error analysis is *a priori*: a hypothesis on the roundoff error yields bounds for the global error. Since worst cases enter the rigorous bounds, the observed error is often much smaller.

There are also *a posteriori* error bounds for an already numerically computed approximation. For instance, a residual (of the computed solution) might be computed using some higher precision.

1.3 Error propagation and condition

Here we investigate the influence of some uncertainty in the input data. That is, we shift the focus from the algorithm to the problem itself. The results have consequences for numerics and the error propagation (propagation of roundoff errors):

1) solution procedure decomposed in a sequence of steps: roundoff error in ith step enter as input errors the $(i+1)$th step;

2) condition (problem inherent amplification of errors): yields gauge for assessment of roundoff errors.

We abstractly characterise a problem as follows:

input data	$x = (x_1, \ldots, x_n)^\top \in \mathbb{D} \subset \mathbb{R}^n$
result	$y = (y_1, \ldots, y_m)^\top \in \mathbb{R}^m$
problem	map $\varphi : \mathbb{D} \to \mathbb{R}^m$, which assigns to any (admissible) input data a unique result: $y = \varphi(x) = (\varphi_1(x), \ldots, \varphi_m(x))^\top$

To solve this problem is meant to find/compute the value of φ at x. Now, how sensitive is the (analytic) result y with respect to changes in the input data x?

The *differential error analysis* investigates the partial derivatives of φ_i with respect to the input data x_j: $\partial\varphi_i/\partial x_j$, – this quantifies the sensitivity (of φ_i) w.r.t. changes in the initial data.

Let $\Delta x_i, i = 1, \ldots, n$ denote the absolute perturbation, and $\varrho x_i := \Delta x_i / x_i$ the relative quantity for component i. In linear approximation for $\Delta y_i = \varphi_i(\widetilde{x}) - \varphi_i(x)$ and $\varrho y_i = \Delta y_i / y_i$, we have

$$\Delta y_i \doteq \sum_{j=1}^{n} \frac{\partial \varphi_i}{\partial x_j} \Delta x_j \tag{1.5}$$

$$\varrho y_i \doteq \sum_{j=1}^{n} \frac{x_j}{y_i} \frac{\partial \varphi_i}{\partial x_j} \varrho x_j \tag{1.6}$$

$\rightsquigarrow$ indeed amplification factors!

Note on derivation: For $n = m = 1$, Taylor expansion yields

$$\begin{aligned}\Delta y &= \varphi(\widetilde{x}) - \varphi(x) \\ &= \varphi(x) + \frac{\partial \varphi}{\partial x}(x)\Delta x + \mathcal{O}((\Delta x)^2) - \varphi(x) \\ &\doteq \frac{\partial \varphi}{\partial x}(x)\Delta x\end{aligned}$$

$$\varrho y = \frac{\Delta y}{y} \doteq \frac{\partial \varphi}{\partial x}(x)\frac{\Delta x}{y}\frac{x}{x} = \frac{x}{y}\frac{\partial \varphi}{\partial x}(x)\varrho x.$$

Corresponding to (1.5), there is a vector notation

$$\Delta y = \frac{\partial \varphi}{\partial x} \cdot \Delta x$$

with the Jacobian matrix $\frac{\partial \varphi}{\partial x} \in \mathbb{R}^{m \times n}$.

Quantity $x_j/y_i \cdot \partial \varphi_i / \partial x_j$ gauges the influence the relative error in x_j to the result y_i. Analog quantity $\partial \varphi_i / \partial x_j$ for the absolute error. These numbers are referred to as *condition numbers*.

Definition 1.6 (Condition numbers) *A problem is well-conditioned, if small changes in Δx, and ϱx yield small changes in Δy and ϱy, respectively. Otherwise it is ill-conditioned. As condition numbers, we use absolute and relative variants (coordinate-wise and vectorial):*

$$\left|\frac{\partial \varphi_i}{\partial x_j}\right|, \quad \left\|\frac{\partial \varphi}{\partial x}\right\|, \quad \left|\frac{x_j}{y_i}\frac{\partial \varphi_i}{\partial x_j}\right| . \qquad \square$$

$\rightsquigarrow$ 'small' is problem dependent.
$\rightsquigarrow$ describe how much domain is stretched
$\rightsquigarrow$ feature of the mathematical problem (see Fig. 3, Fig 4).

Also the elementary operations exhibit conditions. We obtain the absolute errors:

$$\Delta(a \pm b) = \Delta a \pm \Delta b \tag{1.7}$$

$$\Delta(a \cdot b) \doteq b\Delta a + a\Delta b \tag{1.8}$$

$$\Delta(a/b) \doteq \Delta a/b - a\Delta b/b^2 \tag{1.9}$$

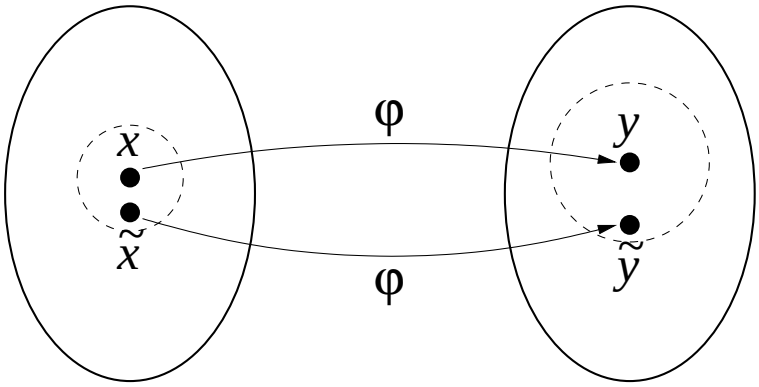


Figure 3: Sketch "well-conditioned".

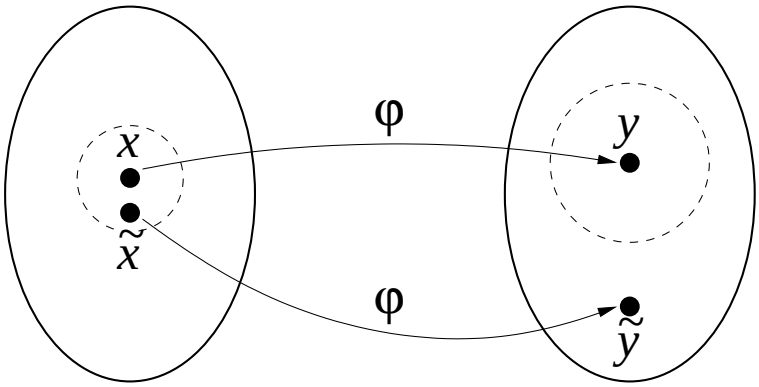


Figure 4: Sketch "ill-conditioned".

and for relative errors (with $\varrho x = \Delta x/x$):

$$\varrho(a \pm b) = \frac{a}{a \pm b}\varrho a + \frac{b}{a \pm b}\varrho b \tag{1.10}$$

$$\varrho(a \cdot b) \doteq \varrho a + \varrho b \tag{1.11}$$

$$\varrho(a/b) \doteq \varrho a - \varrho b \tag{1.12}$$

We find: the relative condition for multiplication and division is 1. That is, the operation may be called well-conditioned. For addition/subtraction, the absolute conditions are small, the relative conditions are *unbounded*! So-called *cancellation* poses a delicate situation: If $a \pm b \approx 0$, i.e. a and b have the some common leading digits, then these digits disappear.

Example 1.5 An ill-conditioned problem cannot be amended by an algorithm.

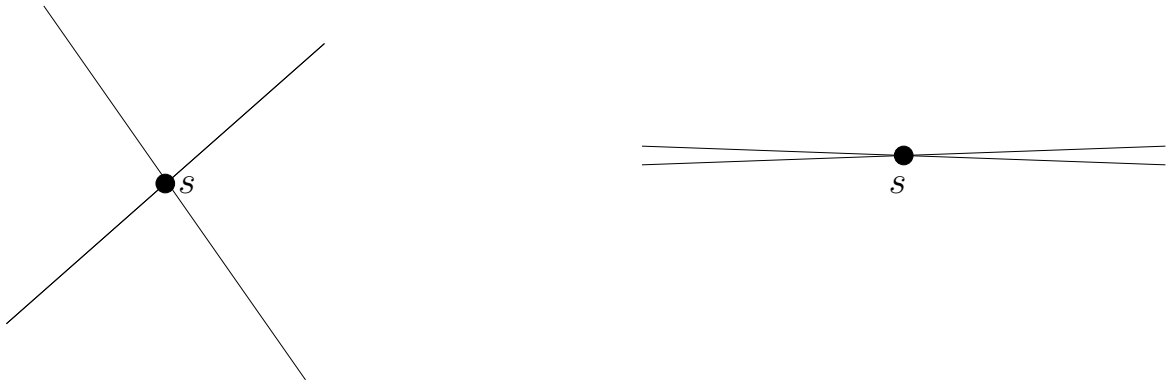


Figure 5: Computation of the intersection of lines

We investigate the intersection of two lines $y_i = a_i x + b_i$ $(i = 1, 2)$ in point s. Regarding Fig. 5, on the left-hand side, small data changes in a_i, b_i produce small changes in s. On the right-hand side, large changes arise even for small changes in the data – much harder to located the intersection s. $\diamond$

Condition plus roundoff error

If problem φ decomposes, then any subproblem has an own condition number. By the chain rule for $\varphi = \sigma \circ \tau$

$$\frac{\partial \varphi}{\partial x} = \left(\frac{\partial \sigma}{\partial z}\right)_{z=\tau(x)} \cdot \frac{\partial \tau}{\partial x},$$

we obtain the following relation for the condition numbers:

$$\mathrm{cond}_{\sigma\circ\tau} = \mathrm{cond}_{\sigma} \cdot \mathrm{cond}_{\tau}\,.$$

Notice, the total condition is independent of how the subproblems are formed. Generally, the smallness of the total parameter $\mathrm{cond}_{\sigma\circ\tau}$ does not guarantee, that both factors are small.

Often there are decompositions (algorithmic variants) with a large and small factor. Roundoff errors, which can be interpreted as input errors in a step, may cause such an algorithm to be useless.

$$\varphi(x) = \sigma(\tau(x) + \Delta\tau) + \Delta\sigma \tag{1.13}$$

with errors $\Delta\tau, \Delta\sigma$ arising from roundoff errors in the evaluation of the functions τ, σ. Considering a perturbance Δx in the initial values, we obtain by linearisation (with other $\Delta\tau, \Delta\sigma$ as in (1.13), but in the same order of magnitude)

$$\begin{aligned}
\varphi(x+\Delta x) &= \sigma(\tau(x+\Delta x)+\Delta\tau)+\Delta\sigma \\
&\doteq \sigma(\tau(x)+\tfrac{\partial\tau}{\partial x}(x)\Delta x+\Delta\tau)+\Delta\sigma \\
&\doteq \sigma(\tau(x))+\tfrac{\partial\sigma}{\partial x}(x)\tfrac{\partial\tau}{\partial x}(x)\Delta x+\tfrac{\partial\sigma}{\partial x}(x)\Delta\tau+\Delta\sigma \\
&= \varphi(x)+\tfrac{\partial\varphi}{\partial x}(x)\Delta x+\tfrac{\partial\sigma}{\partial x}(x)\Delta\tau+\Delta\sigma
\end{aligned}$$

Error in Algorithm

A numerical algorithm $\tilde{\varphi}$ to compute of $y = \varphi(x)$ is *numerically more trustworthy* than another algorithm, if the total influence of roundoff errors is smaller. This leads to two further notations:

Definition 1.7 (Acceptable result) *In the sense of backward analysis, a numerically computed solution $\tilde{y}$ for initial data x is called acceptable, if it can be interpreted as exact solution to modified data $\tilde{x}$ for given tolerance (ε_x):*

$$\tilde{y} \text{ acceptable } \Leftrightarrow\ \exists\tilde{x}:\ \tilde{y}=\varphi(\tilde{x}) \text{ and } \|\tilde{x}-x\|\le\|x\|\varepsilon_x \text{ or } \frac{\|\tilde{x}-x\|}{\|x\|}\le\varepsilon_x$$

(Tolerance ε_x in the order of magnitude of machine precision ε_0.) □

And in the overall

Definition 1.8 (Well behaved / Numerically stable algorithm) *An algorithm is said to be well behaved or numerically stable, if for any input data x the computed (approximated) solution $\tilde{y}$ is acceptable.* □

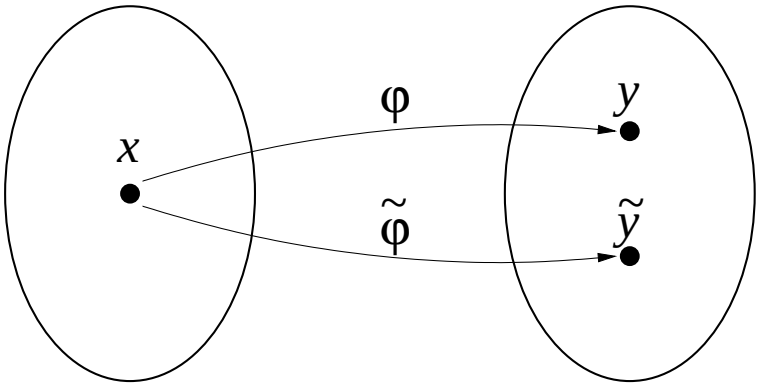


Figure 6: Sketch forward-analysis.

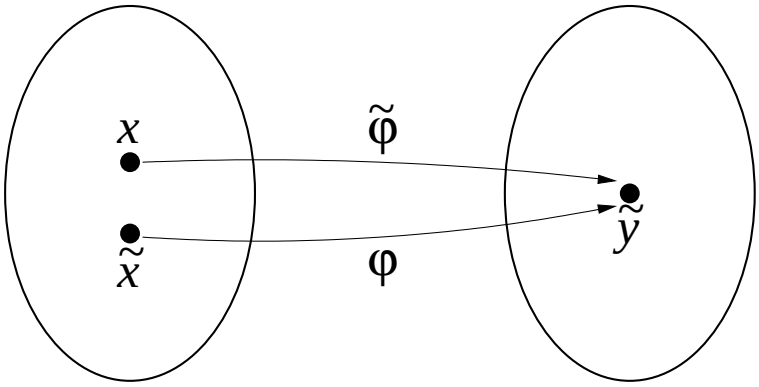


Figure 7: Sketch backward-analysis.

Remarks

- Def. 1.7 can be weakened: the result is acceptable, if there exists y' with $\|\tilde{y} - y'\|$ small and y' is exact for data $\tilde{x}$.
  ($\|\tilde{y} - y'\|$ in the order of magnitude: $\{\|\partial\varphi/\partial x\|\|x\| + \|y\|\}\varepsilon_0$, with machine precision ε_0.)
- Horner-Scheme and elementary operations are stable algorithms.
- Concatenation of two stable algorithms is sometimes not stable.
- For instability, a counterexample suffices. For stability, we have to estimate both errors in the algorithm and roundoff errors.

- For well-conditioned problems, forward analysis is conceivable; for ill-conditioned problems only backward analysis can be applied.

Chapter 2

Vectors and Matrices

Operations using vectors and matrices arise in many numerical methods, especially for solving linear and nonlinear systems. In this section, we introduce basic tools concerning vectors and matrices, which are important in numerical mathematics.

2.1 Notations

A vectors in $\mathbb{R}^n$ or $\mathbb{C}^n$ is written as column

$$x \in \mathbb{R}^n/\mathbb{C}^n \quad \Leftrightarrow \quad x = \begin{pmatrix} x_1 \\ \vdots \\ x_n \end{pmatrix} \quad \text{with } x_i \in \mathbb{R}/\mathbb{C}. \tag{2.1}$$

The transposed form represents a row, i.e. $x^T = (x_1, \ldots, x_n)$.

Matrices are used to describe *linear mappings* $\Phi : \mathbb{R}^n \to \mathbb{R}^m$ and $\Phi : \mathbb{C}^n \to \mathbb{C}^m$, respectively, via $\Phi(x) = Ax$ using $A \in \mathbb{R}^{m\times n}/\mathbb{C}^{m\times n}$. The corresponding notation is

$$A \in \mathbb{R}^{m\times n}/\mathbb{C}^{m\times n} \quad \Leftrightarrow \quad A = \begin{pmatrix} a_{11} & \cdots & a_{1n} \\ \vdots & & \vdots \\ a_{m1} & \cdots & a_{mn} \end{pmatrix} \quad \text{with } a_{ij} \in \mathbb{R}/\mathbb{C}. \tag{2.2}$$

In the case $m = n$, the matrix is said to be quadratic of order n. We will use the abbreviation $M(m, n)$ for the set of matrices in $\mathbb{R}^{m\times n}$ or $\mathbb{C}^{m\times n}$, respectively.

The *unit matrix* or *identity* $I \in M(n, n)$ is given by

$$I = \begin{pmatrix} 1 & & 0 \\ & \ddots & \\ 0 & & 1 \end{pmatrix}. \tag{2.3}$$

Diagonal matrices $D \in M(n, n)$ exhibit the form

$$D = \begin{pmatrix} d_1 & & 0 \\ & \ddots & \\ 0 & & d_n \end{pmatrix}, \tag{2.4}$$

where sometimes the abbreviation $D = \operatorname{diag}(d_1, \ldots, d_n)$ is employed.

2.2 Determinant, Inverse, Eigenvalues

For square matrices $A \in M(n, n)$, the *determinant* $\det A \in \mathbb{R}/\mathbb{C}$ can be defined by

$$\det A := \sum_{\sigma\in S_n} \operatorname{sign}(\sigma) \cdot a_{1\sigma(1)} \cdot \ldots \cdot a_{n\sigma(n)}, \tag{2.5}$$

where S_n is the set of all permutations on $\{1, \ldots, n\}$ and $\operatorname{sign}(\sigma)$ is the sign of a special permutation σ.

The property $\det A \neq 0$ implies that the rows as well as the columns of A are linearly independent vectors and vice versa. In this case, the linear mapping $\Phi(x) := Ax$ becomes bijective. Consequently, the inverse mapping Φ^{-1} exists and is linear. Thus $\Phi^{-1}(x) = A^{-1}x$ involving a unique matrix $A^{-1} \in M(n, n)$, which is called the *inverse* of A. It holds

$$A \cdot A^{-1} = A^{-1} \cdot A = I. \tag{2.6}$$

The inverse A^{-1} can be written in terms of determinants using A and submatrices of A. Likewise, Cramer's rule formulates the solution $x = A^{-1}b$ of the linear system $Ax = b$ applying determinants.

Further properties of the determinant are:

- $\det(A \cdot B) = (\det A) \cdot (\det B)$ for all $A, B \in M(n, n)$
- $\det A^{-1} = (\det A)^{-1}$ for all $A \in M(n, n)$ with $\det A \neq 0$
- $\det(\alpha A) = \alpha^n \det A$ for all $A \in M(n, n)$ and $\alpha \in \mathbb{R}/\mathbb{C}$
- $\det A^T = \det A$ for all $A \in \mathbb{R}^{n\times n}$, $\det A^H = \overline{\det A}$ for all $A \in \mathbb{C}^{n\times n}$
- If $U \in M(n, n)$ is an upper triangular matrix with elements $u_{11}, \ldots, u_{nn}$ on the diagonal, then it holds $\det U = u_{11} \cdot \ldots \cdot u_{nn}$.

The *characteristic polynomial* of a matrix $A \in M(n, n)$ is defined as

$$p_A(\lambda) := \det(A - \lambda I). \tag{2.7}$$

The polynomial is of order n and can be written in the form

$$p_A(\lambda) = (-1)^n(\lambda - \lambda_1) \cdot \ldots \cdot (\lambda - \lambda_n). \tag{2.8}$$

The zeros $\lambda_1, \ldots, \lambda_n \in \mathbb{C}$ are called *eigenvalues*, where recurrences are possible. For each eigenvalue λ, an *eigenvector* $v \in \mathbb{C}^n$ exists, which is defined by the property $Av = \lambda v$. The determinant of A is given by $\det A = \lambda_1 \cdot \ldots \cdot \lambda_n$.

Attention: The computation of the determinant via approaches like (2.5) is numerically unstable. Consequently, the inverse matrix is never computed by determinants and linear systems are never solved by Cramer's rule. Moreover, the use of determinants for these problems would demand a huge computational effort.

If we introduce another basis $\{b_1, \ldots, b_n\}$ ($b_i \in \mathbb{R}^n/\mathbb{C}^n$) of the space $\mathbb{R}^n/\mathbb{C}^n$ in terms of an original basis, then the arising *basis transformation* can be

described by the matrix

$$B = \begin{pmatrix} | & & | \\ b_1 & \cdots & b_n \\ | & & | \end{pmatrix}, \tag{2.9}$$

where $\det B \neq 0$ holds. Let x and y be the representation in the old and new basis, respectively. Then it follows $y = B^{-1}x$ and $x = By$. Accordingly, a linear mapping transforms via

$$\Phi(x) = Ax \quad \Rightarrow \quad \Phi(y) = B^{-1}ABy. \tag{2.10}$$

In general, *similarity transformations* are defined by

$$\tilde{A} = B^{-1}AB \quad \text{with} \quad \det B \neq 0 \tag{2.11}$$

and the matrices $A, \tilde{A}$ are called *similar*. Similarity transformations generate an equivalence relation in $M(n,n)$, i.e. $A \sim \tilde{A}$, if a matrix B exists satisfying (2.11). In particular, the eigenvalues of A are invariant in a similarity transformation, since the characteristic polynomial does not change

$$\begin{aligned} p_{\tilde{A}}(\lambda) &= \det(\tilde{A} - \lambda I) = \det(B^{-1}AB - \lambda B^{-1}B) \\ &= \det(B^{-1}(A - \lambda I)B) = (\det B^{-1}) \cdot (\det(A - \lambda I)) \cdot (\det B) \\ &= \det(A - \lambda I) = p_A(\lambda). \end{aligned} \tag{2.12}$$

Furthermore, a matrix $A \in M(n,n)$ is called *diagonalisable*, if a basis consisting of eigenvectors exists. Consequently, $D = B^{-1}AB$ holds, where D is a diagonal matrix containing the eigenvalues.

2.3 Special matrices

In this subsection, we introduce special classes of square matrices and discuss their properties.

Definition 2.1 *$A \in \mathbb{R}^{n\times n}$ is called a* symmetric matrix, *if $A = A^T$, i.e. $a_{ij} = a_{ji}$, holds. $A \in \mathbb{C}^{n\times n}$ is called a* hermitian matrix, *if $A = A^H$, i.e. $a_{ij} = \overline{a_{ji}}$, holds.*

Definition 2.2 *$A \in \mathbb{R}^{n\times n}$ is called an* orthogonal matrix*, if $A^{-1} = A^T$ holds. $A \in \mathbb{C}^{n\times n}$ is called a* unitary matrix*, if $A^{-1} = A^H$ holds.*

Definition 2.3 *$A \in \mathbb{R}^{n\times n}$ is called a* positive definite matrix*, if $x^T Ax > 0$ holds for all $x \in \mathbb{R}^n$ with $x \neq 0$. $A \in \mathbb{R}^{n\times n}$ is called a* positive semi-definite matrix*, if $x^T Ax \geq 0$ holds for all $x \in \mathbb{R}^n$.*

The definition of *negative (semi-)definite matrices* is analogue via replacing $>, \geq$ by $<, \leq$.

Orthogonal/unitary matrices have the advantage that the inverse can be directly obtained. Furthermore, they play a role in the context of the *scalar product*

$$\langle x, y\rangle := x^T y \quad x, y \in \mathbb{R}^n \quad \text{or} \quad \langle x, y\rangle := x^H y \quad x, y \in \mathbb{C}^n. \tag{2.13}$$

Thereby, the crucial property of a unitary matrix Q is

$$\langle x, y\rangle = x^H y = x^H Q^{-1} Qy = x^H Q^H Qy = (Qx)^H Qy = \langle Qx, Qy\rangle \tag{2.14}$$

and analogue for orthogonal matrices. Thus the linear mapping $\Phi(x) := Qx$ preserves the length of vectors as well as the angle between two vectors.

For symmetric/hermitian matrices, a basis of corresponding eigenvectors exists, i.e. these matrices are diagonalisable. Furthermore, the involved eigenvalues are always real numbers. In addition, the eigenvectors can be chosen orthogonal with respect to the scalar product, which yields an orthogonal/unitary matrix Q of basis vectors

$$D = Q^T AQ \quad \text{or} \quad D = Q^H AQ. \tag{2.15}$$

This property allows to characterise the positive definiteness of a symmetric matrix. We rearrange

$$x^T Ax = x^T QDQ^T x = (Q^T x)^T D(Q^T x). \tag{2.16}$$

Since $\det Q^T \neq 0$, it follows $\{x \in \mathbb{R}^n : x \neq 0\} = \{Q^T x \in \mathbb{R}^n : Q^T x \neq 0\}$. Thus the matrix is positive definite, if and only if all eigenvalues are positive.

Likewise, the matrix is positive semi-definite, if and only if all eigenvalues are non-negative. An according result follows for negative (semi-)definite matrices. If a positive as well as a negative eigenvalue exist, then the matrix is called *indefinite*. Positive and negative definite matrices A satisfy $\det A \neq 0$, because

$$\det A = 0 \quad \Rightarrow \quad \exists x \neq 0 : Ax = 0 \quad \Rightarrow \quad \exists x \neq 0 : x^H Ax = 0. \tag{2.17}$$

Thus corresponding linear systems have a unique solution.

2.4 Norms

Definition 2.4 *A function $\|\cdot\| : \mathbb{C}^n \to \mathbb{R}$ is called a (vector)* norm*, if the following three properties are satisfied.*

1. $\|x\| > 0$ *for all* $x \in \mathbb{C}^n$ *with* $x \neq 0$ *(positive definite)*
2. $\|\alpha x\| = |\alpha| \cdot \|x\|$ *for all* $\alpha \in \mathbb{C}$ *and* $x \in \mathbb{C}^n$ *(homogeneity)*
3. $\|x + y\| \leq \|x\| + \|y\|$ *for all* $x, y \in \mathbb{C}^n$ *(subadditivity)*

The same definition holds for the space $\mathbb{R}^n$ by specialisation.

All norms, which are defined on the same space $\mathbb{C}^n$ are equivalent in the following sense: For each pair of norms $\|\cdot\|_a, \|\cdot\|_b$ there are constants $c_1, c_2 > 0$ satisfying

$$c_1\|x\|_a \leq \|x\|_b \leq c_2\|x\|_a \quad \text{for all } x \in \mathbb{C}^n. \tag{2.18}$$

A class of norms is given by

$$\|x\|_p := \left(\sum_{j=1}^{n} |x_j|^p \right)^{1/p} \quad \text{for } 1 \leq p \leq \infty. \tag{2.19}$$

The following three special cases are most commonly used.

$$\begin{aligned} \|x\|_1 &= \textstyle\sum_{j=1}^n |x_j| && \text{sum norm} \\ \|x\|_2 &= \sqrt{\textstyle\sum_{j=1}^n |x_j|^2} && \text{Euclidian norm} \\ \|x\|_\infty &= \max_{j=1,\ldots,n} |x_j| && \text{maximum norm} \end{aligned} \tag{2.20}$$

The Euclidean norm can be written in terms of vector operations via $\|x\|_2 = \sqrt{x^H x}$, i.e. using the *scalar product* $\langle x, y\rangle := x^H y$.

Definition 2.5 *A function* $\|\cdot\| : M(m,n) \to \mathbb{R}$ *is called a* matrix norm*, if the following properties are satisfied.*

1. $\|A\| > 0$ *for all* $A \in M(m,n)$ *with* $A \neq 0$ *(positive definite)*
2. $\|\alpha A\| = |\alpha| \cdot \|A\|$ *for all* $\alpha \in \mathbb{C}$ *and* $A \in M(m,n)$ *(homogeneity)*
3. $\|A + B\| \leq \|A\| + \|B\|$ *for all* $A, B \in M(m,n)$ *(subadditive)*
4. *In case of* $m = n$*:*
   $\|A \cdot B\| \leq \|A\| \cdot \|B\|$ *for all* $A, B \in M(n,n)$ *(submultiplicative)*

Accordingly, all matrix norms defined on the same space $M(m,n)$ are equivalent.

The matrix norm $\|\cdot\|_M$ on $M(m,n)$ is said to be *consistent* with the vector norms $\|\cdot\|_a$ on $\mathbb{R}^n/\mathbb{C}^n$ and $\|\cdot\|_b$ on $\mathbb{R}^m/\mathbb{C}^m$, if

$$\|Ax\|_b \leq \|A\|_M \cdot \|x\|_a \quad \text{for all } A \in M(m,n),\ x \in \mathbb{R}^n/\mathbb{C}^n. \tag{2.21}$$

Definition 2.6 *Given arbitrary vector norms* $\|\cdot\|_a$ *on* $\mathbb{R}^n/\mathbb{C}^n$ *and* $\|\cdot\|_b$ *on* $\mathbb{R}^m/\mathbb{C}^m$*, the* subordinate matrix norm $\mathrm{lub} : \mathbb{R}^{m\times n}/\mathbb{C}^{m\times n} \to \mathbb{R}$ *is defined by*

$$\mathrm{lub}(A) := \max_{x\neq 0} \frac{\|Ax\|_b}{\|x\|_a} \tag{2.22}$$

(lub: least upper bound).

An equivalent expression is obtained via

$$\max_{x\neq 0}\frac{\|Ax\|_b}{\|x\|_a}=\max_{x\neq 0}\left\|A\frac{x}{\|x\|_a}\right\|_b=\max_{\|y\|_a=1}\|Ay\|_b. \tag{2.23}$$

The subordinate matrix norm indicates the maximum magnification of a vector x by multiplication with A. The four properties of a matrix norm can be verified. Furthermore, the subordinate norm is consistent with the vector norms

$$\|Ax\|_b=\frac{\|Ax\|_b}{\|x\|_a}\cdot\|x\|_a\leq \mathrm{lub}(A)\cdot\|x\|_a \quad \text{for all } \; x\neq 0. \tag{2.24}$$

Moreover, the subordinate norm is the smallest matrix norm, which is consistent with the vector norm, i.e.

$$\|Ax\|_b\leq\|A\|\cdot\|x\|_a \text{ for all } A,x \quad\Rightarrow\quad \mathrm{lub}(A)\leq\|A\| \text{ for all } A. \tag{2.25}$$

Considering square matrices ($m=n$), usually $\|\cdot\|_a=\|\cdot\|_b$ is presumed. In this context, another quantity plays a role.

Definition 2.7 *Let $\lambda_1,\ldots,\lambda_n\in\mathbb{C}$ be the eigenvalues of a matrix $A\in M(n,n)$, then the* spectral radius *of A is given by*

$$\rho(A):=\max_{j=1,\ldots,n}|\lambda_j|. \tag{2.26}$$

Let v be an eigenvector corresponding to an eigenvalue λ, it follows using an arbitrary vector norm and a consistent matrix norm

$$|\lambda|\cdot\|v\|=\|\lambda v\|=\|Av\|\leq\|A\|\cdot\|v\| \quad\Rightarrow\quad |\lambda|\leq\|A\|. \tag{2.27}$$

Consequently, it holds $\rho(A)\leq\|A\|$ for all subordinate matrix norms. The spectral radius itself is not a matrix norm, for example, matrices $A\neq 0$ with all eigenvalues equal to zero exist. Nevertheless, for fixed A and $\varepsilon>0$, a special vector norm exists, whose subordinate norm satisfies $\|A\|\leq\rho(A)+\varepsilon$.

The subordinate matrix norms corresponding to the common vector norms (2.20) are

$$\begin{aligned} \|A\|_1 &= \max_{j=1,\ldots,n} \sum_{i=1}^{n} |a_{ij}| \quad \text{column-sum norm} \\ \|A\|_2 &= \sqrt{\rho(A^H A)} \quad \text{spectral norm} \\ \|A\|_\infty &= \max_{i=1,\ldots,n} \sum_{j=1}^{n} |a_{ij}| \quad \text{row-sum norm} \end{aligned} \tag{2.28}$$

The column-sum and the row-sum norm can be computed easily. On the contrary, the computation of the spectral norm demands solving an eigenvalue problem. However, the spectral norm is known a priori for some matrix classes. Orthogonal/unitary matrices A exhibit $\|A\|_2 = 1$.

A matrix norm on $M(m,n)$, which is no subordinate norm for any vector norm, is given by

$$\|A\|_S := \sqrt{\sum_{i=1}^{m} \sum_{j=1}^{n} |a_{ij}|^2} \quad \text{(Schur norm)}. \tag{2.29}$$

Nevertheless, this matrix norm is consistent with the Euclidean vector norm. In contrast to the subordinate matrix norm $\|\cdot\|_2$, the Schur norm is easy to compute.

Definition 2.8 *The* condition *of a matrix $A \in M(n,n)$ with $\det A \neq 0$ is given by*

$$\kappa(A) := \frac{\max\limits_{\|x\|=1} \|Ax\|}{\min\limits_{\|x\|=1} \|Ax\|}, \tag{2.30}$$

which depends on the chosen norm.

The condition specifies the amount of deformation corresponding to the unit sphere with respect to the used norm. The favourable case $\kappa(A) = 1$ is fulfilled by orthogonal/unitary matrices using the Euclidean norm.

If $\det(A) = 0$ and $A \neq 0$, it follows $\kappa(A) = \infty$ in any norm, since an x with $Ax = 0$ but $\|x\| = 1$ exists. If $\det(A) \neq 0$, an equivalent expression of the

condition is obtained by employing A^{-1}. We have

$$\frac{1}{\min\limits_{\|x\|=1} \|Ax\|} = \max_{\|x\|=1} \frac{1}{\|Ax\|} = \max_{x\neq 0} \frac{\|x\|}{\|Ax\|} = \max_{y\neq 0} \frac{\|A^{-1}y\|}{\|y\|} = \|A^{-1}\|, \quad (2.31)$$

where the subordinate matrix norm is used. Hence it follows

$$\kappa(A) = \|A\| \cdot \|A^{-1}\|. \quad (2.32)$$

Furthermore, we conclude

$$1 = \|I\| = \|AA^{-1}\| \leq \|A\| \cdot \|A^{-1}\| = \kappa(A) \quad (2.33)$$

and thus 1 is the optimal value.

2.5 Elementary matrices

Now we give a list of so-called *elementary matrices*. In numerical linear algebra, algorithms can often be described by successive operations involving these matrices. However, these processes represent just a theoretical tool. In the following, the vectors and matrices may be real as well as complex.

Scaling:

We consider a diagonal- or *scaling matrix*

$$D = \text{diag}(d_1, d_2, \ldots, d_n). \quad (2.34)$$

operation on vectors: $(Dx)_j = d_j x_j, \quad j = 1, \ldots, n.$

operation on matrices:

$$(DA)_{ij} = d_i a_{ij}, \quad i\text{th row scaled with } d_i,\ i = 1, \ldots, n.$$

$$(AD)_{ij} = a_{ij} d_j, \quad j\text{th column scaled with } d_j,\ j = 1, \ldots, n.$$

inverse: $D^{-1} = \text{diag}(d_1^{-1}, d_2^{-1}, \ldots, d_n^{-1}) \quad (d_j \neq 0,\ j = 1, \ldots, n)$

similarity transformation: $A \mapsto DAD^{-1},\ a_{ij} \mapsto (d_i/d_j)a_{ij},\ i,j = 1,\ldots,n$

Transposition:

Transposition matrices P_{ij} are special permutation matrices. They agree to the unit matrix I except for the four entries $(i,i),(i,j),(j,i),(j,j)$, where $i,j \in \{1,\ldots,n\}$ represent a certain choice.

$$P_{ij} := \begin{pmatrix} 1 & & & & & & & & \\ & \ddots & & & & & & & \\ & & 1 & & & & & & \\ & & & 0 & & & 1 & & \\ & & & & 1 & & & & \\ & & & & & \ddots & & & \\ & & & & & & 1 & & \\ & & & 1 & & & & 0 & \\ & & & & & & & & 1 \\ & & & & & & & & & \ddots \\ & & & & & & & & & & 1 \end{pmatrix} \tag{2.35}$$

operation on vectors: $P_{ij}x$ interchanges components i and j.

operation on matrices:

$P_{ij}A$ interchanges rows i and j.

AP_{ij} interchanges columns i and j.

inverse: $P_{ij}^{-1} = P_{ij}$ (involution)

similarity transformation: $A \mapsto P_{ij}AP_{ij}$ interchanges rows and columns.

In particular: $\begin{matrix} a_{ii} & a_{ij} \\ a_{ji} & a_{jj} \end{matrix} \mapsto \begin{matrix} a_{jj} & a_{ji} \\ a_{ij} & a_{ii} \end{matrix}$ 'crosswise'

Since P_{ij} is symmetric $P_{ij} = P_{ij}^T$, P_{ij} is orthogonal due to $P_{ij}^{-1} = P_{ij}$. Hence

P_{ij} exhibits the optimal condition number $\kappa_2(P_{ij}) = 1$.

Permutation:

Let $\sigma : \{1, 2, \ldots, n\} \to \{1, 2, \ldots, n\}$ be bijective, i.e. σ represents a permutation of the set $\{1, 2, \ldots, n\}$. The corresponding permutation matrix is obtained by a permutation of the rows of the identity matrix

$$I = \begin{pmatrix} - & {e_1}^\top & - \\ & \vdots & \\ - & {e_n}^\top & - \end{pmatrix} \quad \longrightarrow \quad P_\sigma := \begin{pmatrix} - & {e_{\sigma(1)}}^\top & - \\ & \vdots & \\ - & {e_{\sigma(n)}}^\top & - \end{pmatrix}. \tag{2.36}$$

operation on vectors: $P_\sigma x$ permutes components of x.

operation on matrices:

$P_\sigma A$ permutes rows of A.

AP_σ permutes columns of A.

inverse: $P_\sigma^{-1} = P_{\sigma^{-1}} = P_\sigma^\top$

Each permutation matrix can be written as a product of transposition matrices, i.e.

$$P_\sigma = \prod_{l=1}^{L} P_{i(l),j(l)}. \tag{2.37}$$

Row/column operations:

The elementary operation of adding a multiple of one row/column to another row/column can be described by the matrix

$$N_{ij}(\alpha) := \begin{pmatrix} 1 & & & \\ & \ddots & & \\ & \alpha & \ddots & \\ & & & 1 \end{pmatrix}. \tag{2.38}$$

$N_{ij}(\alpha)$ is identical to the unit matrix except for the component (i, j), which contains the element α.

operation on vectors:

$$N_{ij}(\alpha)x: \; x_i \mapsto x_i + \alpha \cdot x_j \qquad \alpha\text{-times comp. } j \text{ added to comp. } i$$

operation on matrices:

$$N_{ij}(\alpha)A: \quad \alpha\text{-times row } j \text{ added to row } i$$

$$AN_{ij}(\alpha): \quad \alpha\text{-times column } i \text{ added to column } j$$

inverse: $N_{ij}(\alpha)^{-1} = N_{ij}(-\alpha)$

Similarity transformation are not of interest here.

On the other hand, products of the matrices $N_{ij}(\alpha)$ are important, since they arise in the Gaussian elimination.

$$\prod_{i=1, i\neq j}^{n} N_{ij}(\alpha_i) = \begin{pmatrix} 1 & & & \alpha_1 & & & \\ & \ddots & & \vdots & & & \\ & & \ddots & \alpha_{i-1} & & & \\ & & & \ddots & & & \\ & & & \alpha_{i+1} & \ddots & & \\ & & & \vdots & & \ddots & \\ & & & \alpha_n & & & 1 \end{pmatrix} \tag{2.39}$$

Thereby, the succession of the products does not influence the result. Fur-

ther successive products yield

$$\prod_{i>1} N_{i1}(\alpha_{i1}) \prod_{i>2} N_{i2}(\alpha_{i2}) \dots \prod_{i>n-1} N_{i,n-1}(\alpha_{i,n-1}) = \begin{pmatrix} 1 & & & \\ \alpha_{21} & \ddots & & \\ \vdots & & \ddots & \\ \alpha_{n,1} & & \alpha_{n,n-1} & 1 \end{pmatrix}. \tag{2.40}$$

2.6 BLAS package

The software package BLAS (Basic Linear Algebra Subprograms) was developed by J. Dongarra, G. Golub and other american scientists, which have been working on portability and standardisation of numerical software since the seventies.

The BLAS package includes exclusively elementary operations using vectors and matrices. Numerical algorithms can be composed of these established components. Each subroutine of BLAS is adapted a benchmark of a common computer architecture. Further optimisations with respect to a special architecture are possible. BLAS is implemented in FORTRAN but used in many connections to other programming languages by corresponding interfaces.

BLAS is subdivided into three levels:

Level 1 BLAS

These subroutines incorporate vector operations of complexity $\mathcal{O}(n)$ (realised by a `for`-loop). BLAS-1 subroutines are dedicated to scalar computers, where optimised versions exist, for example, by using Assembler programs. For vector computers, the BLAS-1 subroutines can be realised by one or several vector operations.
Examples for BLAS-1 subroutines:

`SDOT`, `DDOT`: $x^T y \to \beta$ scalar product
`SSCAL`, `DSCAL`: $\alpha,\ x \to \alpha x$ scaling a vector

Level 2 BLAS

The BLAS-2 subroutines are defined for constructing a portable software, which performs optimally on many computers. The time-consuming parts are isolated in just a few optimised subroutines. BLAS-2 subroutines execute matrix-vector operations and thus employ typically $\mathcal{O}(n^2)$ floating point operations.
Examples for BLAS-2 subroutines:
`STRSV`, `DTRSV`: $L^{-1}x \to x$ lin. sys. with triangular matrix on vector
`SGEMV`, `DGEMV`: $\alpha Ax + \beta y \to y$ common matrix-vector product

Level 3 BLAS

Using new computer architectures (e.g. several processors, hierarchical memory structures), the BLAS-2 subroutines are only partly suitable. Consequently, the BLAS-3 subroutines include matrix-matrix operations, which exhibit the complexity $\mathcal{O}(n^3)$ (three `for`-loops). Based on BLAS-3, the software LAPACK has been developed, which comprehends the scope of the packages EISPACK and LINPACK, i.e. the linear algebra operations for dense matrices.
Examples for BLAS-3 subroutines:
`STRSM`, `DTRSM`: $L^{-1}X \to X$ lin. sys. with triangular matrix on matrix
`SGEMM`, `DGEMM`: $\alpha AB + \beta C \to C$ common matrix-matrix product

Chapter 3

Direct Methods for Linear Systems

Linear systems occur in many numerical algorithms: e.g. for systems of ordinary differential equations, for systems of partial differential equations, for optimisation problems. Therefore linear systems need to be solved daily many million times.

Here we address Gaussian elimination, LU-decomposition and Cholesky as the standard numerical algorithms for linear systems. Furthermore we discuss the sensitivity of linear systems and the method of least squares for data fitting.

3.1 Gaussian Elimination and LU-Decomposition

Given a linear system of equations:

$$A \cdot x = b \tag{3.1}$$

with real matrix $A \in \mathbb{R}^{n \times n}$, right-hand side $b \in \mathbb{R}^n$ and unknowns $x \in \mathbb{R}^n$. Considering a regular matrix A, i.e. $\det(A) \neq 0$, the unique solution is

$$x = A^{-1}b.$$

And Cramer's rule yields

$$x_i = \det(A_{i,b}) / \det(A), \qquad i = 1, \ldots, n\,.$$

But these results from linear algebra are not suitable for a realisation on a computer. Important principles:

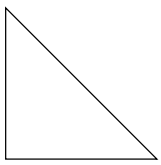
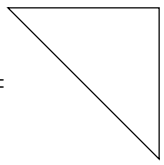
Principle 1: Never use Cramer's rule to compute solution of $A \cdot x = b$!

Principle 2: Never compute (numerically) the inverse A^{-1} explicitly!

Principle 3: Never use $\det(A)$!

To solve (3.1), we use Gaussian Elimination and LU-Decomposition.

A special case are lower and upper triangular matrices:

$A = (a_{ij}) =$  $(a_{ij} = 0 \text{ for } j > i)$ $A =$  $(a_{ij} = 0 \text{ for } i > j)$

Here, we can compute the solution x successively: forward substitution for lower tr. matrices and backward substitution for upper tr. matrices.

Algorithm 3.1 (Forward Substitution)

for $i = 1 : n$

$$x_i := \Big(b_i - \sum_{j=1}^{i-1} a_{ij}\, x_j\Big)/a_{ii}\,;$$

end

$\diamond$

Algorithm 3.2 (Backward Substitution)

for $i = n : -1 : 1$

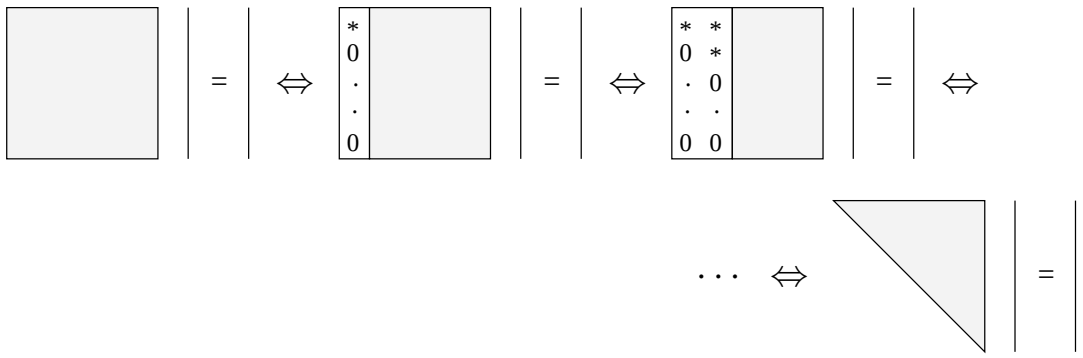
$$x_i := \Big(b_i - \sum_{j=i+1}^{n} a_{ij}\, x_j\Big)/a_{ii}\,;$$

end

$\diamond$

Explanation of names: Forward subst. calculates sequence $x_1, x_2, \ldots, x_n$, whereas backward subst. calculates sequence $x_n, x_{n-1}, \ldots, x_1$.

Now, the idea of Gaussian elimination (LU-decomposition) is to transform to a triangular form. This is done with (at most) $n-1$ transformation steps. Symbolically:



How does this work? – Perform steps $j := 1, \ldots, n-1$:

1. if necessary swap jth-row with any row: $j+1, \ldots, n$
   chosen diagonal elements a_{jj} are called *pivots*

2. swap jth-column with any column: $j+1, \ldots, n$
   ($\rightsquigarrow$ descreases roundoff errors)

3. subtract suitable multiple of jth row from row $i = j+1, \ldots, n$:
   define factor l_{ij}
   $$l_{ij} = a_{ij}/a_{jj}$$
   $\rightsquigarrow$ transform matrix and right-hand side
   ($\rightsquigarrow$ l_{ij} form lower triangular matrix)

Algorithm 3.3 (Gaussian Elimination without pivoting)

for $j = 1 : n-1$

$$
\begin{array}{l}
\quad \textit{for } i = j+1 : n \\
\qquad \textit{for } k = j+1 : n \\
\qquad\quad a_{ik} := a_{ik} - (a_{ij}/a_{jj})\, a_{jk}; \qquad\qquad (*) \\
\qquad \textit{end} \\
\qquad b_i := b_i - (a_{ij}/a_{jj})\, b_j; \\
\quad \textit{end} \\
\textit{end} \\
\textit{for } i = n : -1 : 1 \\
\qquad x_i := \Big(b_i - \sum_{j=i+1}^{n} a_{ij}\, x_j\Big)/a_{ii}\,; \\
\textit{end}
\end{array}
$$

◇

In step j (outer loop j), a zero column is generated below a_{jj} and the problem reduces. In the next step only the block $j+1, \ldots, n$ is worked on.

If all *pivots* are nonzero, we can compute a solution via Gaussian elimination for systems of arbitrary large size.

In the successive generation of the upper triangular matrix, zeros are not explicitly computed.

Example 3.1 A simple computation:

$$
\left(\begin{array}{rrr|r} 1 & -1 & 0 & 1 \\ -1 & 6 & -3 & 0 \\ 0 & -3 & 3.6 & 0 \end{array}\right) \rightsquigarrow \left(\begin{array}{rrr|r} 1 & -1 & 0 & 1 \\ 0 & 5 & -3 & 1 \\ 0 & -3 & 3.6 & 0 \end{array}\right) \rightsquigarrow \left(\begin{array}{rrr|r} 1 & -1 & 0 & 1 \\ 0 & 5 & -3 & 1 \\ 0 & 0 & 1.8 & 0.6 \end{array}\right)
$$

backward substitution yields: $u_3 = \frac{1}{3},\ u_2 = \frac{2}{5},\ u_1 = \frac{7}{5}$. ◇

The marked row $(*)$ in Alg. 3.3 defines the elimination step. It is equivalent to

$$A := N_{ij}(-l_{ij}) \cdot A \tag{3.2}$$

with $l_{ij} = a_{ij}/a_{jj}$ and the elementary matrix N_{ij} (see above, Sect. 2) – the sum of $(-l_{ij}) \times j$th row and ith row.

Notice, in the jth step the leading elements in any row $a_{i,1}, \ldots, a_{i,j-1}$ $(i = j, \ldots, n)$ are already put to zero, by the previous elimination steps. Thus these have no influence in the product $N_{ij}(-l_{ij}) \cdot A$.

In the overall the algorithm can be described by a product of elementary matrices: for the resulting upper triangular matrix U, we obtain

$$U = N_{n,n-1}(-l_{n,n-1}) \cdot \ldots \cdot N_{21}(-l_{21}) \cdot A\,.$$

Since $N_{ij}(\alpha)^{-1} = N_{ij}(-\alpha)$, we have

$$A = N_{21}(l_{21}) \cdot \ldots \cdot N_{n,n-1}(l_{n,n-1}) \cdot U\,,$$

and from the previous section, we know that the product yields a lower triangular matrix $L = (l_{ij})$:

$$N_{21}(l_{21}) \cdot \ldots \cdot N_{n,n-1}(l_{n,n-1}) = \left(\prod_{i>1} N_{i1}(l_{i1})\right) \cdot \ldots \cdot \left(\prod_{i>n-1} N_{i,n-1}(l_{i,n-1})\right) = L$$

Furthermore, all diagonal elements are equal to 1, i.e. $l_{ii} = 1$ for all i. A triangular matrix with this property shall be called *normed*.

That is,

Theorem 3.1 (LU-Decomposition) *The Gaussian elimination (Alg. 3.3) is equivalent to the triangular decomposition: $A = L \cdot U$, including a normed lower triangular L and an upper triangular U.*

The decomposition is unique, the sequence of operations is not. The uniqueness follows from

$$A = L_1 U_1 = L_2 U_2 \quad \Rightarrow \quad L_2^{-1} L_1 = U_1 U_2^{-1}.$$

Since $L_2^{-1} L_1$ is a normed lower and $U_1 U_2^{-1}$ an upper triangular matrix, we obtain

$$L_2^{-1} L_1 = U_1 U_2^{-1} = I \quad \Rightarrow \quad L_1 = L_2,\ U_1 = U_2.$$

Now having computed $A = L \cdot U$, we solve for x in two steps: $Ax = LUx = L(Ux) = b$

1.) Solve $L \cdot y = b$ by *forward substitution*:

$$\text{for } i = 1 : n \qquad y_i = b_i - \sum_{j=1}^{i-1} l_{ij} y_j \tag{3.3}$$

$$\begin{pmatrix} 1 & & & 0 \\ l_{21} & \ddots & & \\ \vdots & & \ddots & \\ l_{n1} & \cdots & l_{n,n-1} & 1 \end{pmatrix} \cdot \begin{pmatrix} y_1 \\ \vdots \\ \vdots \\ y_n \end{pmatrix} = \begin{pmatrix} b_1 \\ \vdots \\ \vdots \\ b_n \end{pmatrix}$$

2.) Solve $U \cdot x = y$ by *backward substitution*:

$$\text{for } i = n : -1 : 1 \qquad x_i = \Big(y_i - \sum_{j=i+1}^{n} u_{ij} x_j \Big) / u_{ii} \tag{3.4}$$

$$\begin{pmatrix} u_{11} & \cdots & u_{1n} \\ & \ddots & \vdots \\ 0 & & u_{nn} \end{pmatrix} \cdot \begin{pmatrix} x_1 \\ \vdots \\ x_n \end{pmatrix} = \begin{pmatrix} y_1 \\ \vdots \\ y_n \end{pmatrix}$$

These operations coincide with Alg. 3.3. Both ways are equivalent. The LU-decomposition is computed as:

Algorithm 3.4 (LU-Decomposition)

for $j = 1 : n$
    $l_{jj} = 1$;
    for $k = j : n$
        $u_{jk} := a_{jk}$;
    end
    for $i = j + 1 : n$
        $l_{ij} := a_{ij} / u_{jj}$;
        for $k = j + 1 : n$
            $a_{ik} := a_{ik} - l_{ij} u_{jk}$;

```
        end
    end
end
```
◇

In an implementation, the factors L and U are stored in one matrix, actually replacing the entries of A.

Often the linear system (3.1) has to be solved with a set of different right-hand sides: b, b', etc. Then the LU-decomposition is performed once, and forward/backward substitution is done for all right-hand sides. For this reason, there are usually two subroutines (modules) for Gaussian elimination.

Complexity

How many operations are necessary for LU-decomposition and substitutions?

We count operations (multiplications, usually $a \cdot b + c$)

$$\begin{aligned}
\sum_{k=1}^{n-1} k^2 &= \tfrac{1}{6}(n-1)n(2n-1) \doteq \frac{n^3}{3} && \text{LU-decomposition} \\
\sum_{k=1}^{n-1} k &= \tfrac{1}{2}(n-1)n \doteq \frac{n^2}{2} && \text{per substitution.}
\end{aligned}$$

Hence the decomposition needs the mayor part of the work. The substitutions need n^2 operations like a matrix-vector multiplication.

Pivoting

In Alg. 3.3 and Alg. 3.4, we assumed inherently that pivots a_{jj} or u_{jj} are nonzero. This is the case for symmetric positive definite and strictly diagonal dominate matrices, but not in general.

That is, we need *pivoting*.

Example 3.2 In the matrix

$$A = \begin{pmatrix} 0 & 1 \\ 1 & 2 \end{pmatrix},$$

we need simply to interchange rows. ◇

There is a further complication by small pivots. These yield large roundoff errors.

The choice of the largest absolute value in the actual column is referred to as *partial pivoting*. If it is the element a_{ij} $(i > j)$, then we have to interchange row i and row j.

In the complete pivoting, we look for the largest absolute element in the whole remainder. Swapping of of rows and columns is then necessary. (More effort, less common.)

The following result ensures that we are on the correct track: LU-decomposition does only fail for singular A (in exact arithmetic).

Theorem 3.2 *For any regular matrix A, a permutation matrix P and a decomposition*

$$P \cdot A = L \cdot U$$

with normed lower triangular L and upper triangular U exist. It is possible to choose P, such that all entries of L have modulus less or equal 1.

Proof:

The elimination of all elements in the jth column below the pivot is described by a multiplication with the matrix

$$K_j = \prod_{i>j} N_{ij}(-l_{ij}) = I + q_j \cdot {e_j}^T \quad \text{with} \quad q_j := (0, \ldots, 0, -l_{j+1,j}, \ldots, -l_{n,j})^T.$$

Since the product of these matrices commutes, we obtain

$$K_j^{-1} = \prod_{i>j} N_{ij}(l_{ij}),$$

which represents exactly the jth column of the matrix L. Without pivoting , the LU-decomposition is given by

$$U = K_{n-1} \cdot \ldots \cdot K_1 \cdot A \quad \text{or} \quad L \cdot U = K_1^{-1} \cdot \ldots \cdot K_{n-1}^{-1} \cdot U = A.$$

Interchanging row i and row j can be described by a multiplication with a transposition matrix P_{ij} (see previous section). Using partial pivoting, an element $a_{k_1,1}$ with $|a_{k_1,1}| > |a_{i,1}|$ for all i is chosen. It holds $a_{k_1,1} \neq 0$, since a column with only zeros implies a singular matrix. The interchange of rows and the following elimination can be written as

$$A \rightsquigarrow K_1 \cdot P_{1,k_1} \cdot A = \left(\begin{array}{c|ccc} a_{k_1,1} & * & \cdots & * \\ \hline 0 & & & \\ \vdots & & A' & \\ 0 & & & \end{array}\right)$$

using the transposition matrix P_{1,k_1} and a matrix K_1 with a structure like above. The choice of the pivot guarantees $|l_{j,1}| = |a_{j,1}/a_{k_1,1}| \leq 1$ for $j = 2, \ldots, n$. Since it holds $\det(A) = a_{k_1,1} \cdot \det(A')$, the remaining matrix A' is regular, too.

Likewise, the following elimination steps yield

$$K_{n-1} \cdot P_{n-1,k_{n-1}} \cdot K_{n-2} \cdot P_{n-2,k_{n-2}} \cdot \ldots \cdot K_1 \cdot P_{1,k_1} \cdot A = U$$

with a upper triangular U. The product of transposion and elimination matrices can be modified. Using the abbreviation $P_i := P_{i,k_i}$, it holds for $j < i$

$$P_i K_j P_i = I + \tilde{q}_j {e_j}^T \qquad \text{with} \quad \tilde{q}_j = P_i q_j,$$

which represents a lower triangular matrix with the same structure as K_j. Since $P_i^{-1} = P_i$, we can expand

$$\begin{aligned} K_{n-1} \cdot \underbrace{(P_{n-1} \cdot K_{n-2} \cdot P_{n-1})}_{=:\tilde{K}_{n-2}} \cdot \underbrace{(P_{n-1} \cdot P_{n-2} \cdot K_{n-3} \cdot P_{n-2} \cdot P_{n-1})}_{=:\tilde{K}_{n-3}} \\ \cdot \ldots \cdot \underbrace{(P_{n-1} \cdot \ldots \cdot P_2 \cdot K_1 \cdot P_2 \cdot \ldots \cdot P_{n-1})}_{=:\tilde{K}_1} \cdot \underbrace{(P_{n-1} \cdot \ldots \cdot P_1)}_{=:P} \cdot A = U. \end{aligned}$$

Thus defining $\tilde{K}_i := P_{n-1} \cdots P_{i+1} K_i P_{i+1} \cdots P_{n-1}$ and $P := P_{n-1} \cdots P_1$, we obtain

$$K_{n-1} \cdot \tilde{K}_{n-2} \cdot \ldots \cdot \tilde{K}_1 \cdot P \cdot A = U.$$

The matrices $\tilde{K}_i$ exhibit the same structure as the matrices K_i (just elements in the ith column beyond the pivot are modified). Thus

$$L := \tilde{K}_1^{-1} \cdot \ldots \cdot \tilde{K}_{n-2}^{-1} \cdot K_{n-1}^{-1}$$

represents a normed lower triangular matrix. The matrix P describes the permutation of the rows. $\square$

Cholesky-Decomposition

For the special class of *symmetric positive definite* (*s.p.d.*) matrices, which occur often in applications, there is no pivoting necessary.

To understand this, we investigate the first elimination step. Let A be a s.p.d. matrix, then $e_1^\top A e_1 = a_{11} > 0$, and the first diagonal element is feasible as pivot. Therefore we obtain

$$A = \left(\begin{array}{c|c} a_{11} & v^\top \\ \hline v & B \end{array}\right) \quad \rightsquigarrow \quad K_1 \cdot A = \left(\begin{array}{c|c} a_{11} & v^\top \\ \hline 0 & B' \end{array}\right) .$$

The remainder B' is composed as

$$B' = B - \frac{vv^\top}{a_{11}} ,$$

which is symmetric again.

Notice:

$$vv^\top = \begin{pmatrix} v_1 \cdot v_1 & \dots & v_1 \cdot v_n \\ \vdots & & \vdots \\ v_n \cdot v_1 & \dots & v_n \cdot v_n \end{pmatrix}$$

Now, for showing definiteness, we put

$$x := \begin{pmatrix} -v^\top y / a_{11} \\ y \end{pmatrix} \quad \text{with arbitrary } \; y \in \mathbb{R}^{n-1},\; y \neq 0$$

and obtain

$$Ax = \begin{pmatrix} 0 \\ B'y \end{pmatrix} \quad \Rightarrow \quad 0 < x^\top A x = y^\top B' y \, .$$

Thus by induction:

Theorem 3.3 *Let A be symmetric positive definite. Then the LU-decomposition yields in each step a symmetric positive definite remainder and the diagonal element can be used as pivot.*

Moreover, it can be shown that a pivoting with respect to numerical stability (choice of large elements) is not necessary for s.p.d. matrices (see later).

In addition (to no pivoting), it is possible to half the computational effort due to symmetry. That will be to define symmetric factors. To this end, notice that

$$A = L \cdot U = LD \underbrace{D^{-1}U}_{\tilde{U}}, \quad D := \operatorname{diag}(u_{11}, \ldots, u_{nn}) .$$

yields a normed upper triangular matrix $\tilde{U} := D^{-1}U$. Thus $\tilde{U}^\top$ represents a normed lower triangular matrix. By symmetry, we have

$$LD\tilde{U} = A = A^\top = \tilde{U}^\top D L^\top .$$

Since the LU-decomposition is unique, it holds $\tilde{U}^\top = L$, and we have the so-called *rational Cholesky-Decomposition*

$$A = L \cdot D \cdot L^\top . \tag{3.5}$$

Since A is s.p.d. and $\det L \neq 0$, we know $(L^{-\top} := (L^\top)^{-1} = (L^{-1})^\top)$

$$0 < (L^{-\top} e_i)^\top (LDL^\top) L^{-\top} e_i = {e_i}^\top D e_i = d_i = u_{ii}.$$

Thus we can define the root of D as

$$D^{1/2} := \operatorname{diag}(\sqrt{u_{11}}, \ldots, \sqrt{u_{nn}}).$$

Now the *Cholesky-Decomposition* reads

$$A = \hat{L} \cdot \hat{L}^\top , \quad \hat{L} := L \cdot D^{1/2} \tag{3.6}$$

with lower triangular matrix $\hat{L}$, which is not normed in general.

Computation: With $\hat{L} = (\hat{\lambda}_{ij})$ and $\hat{L}^\top = (\hat{\hat{\lambda}}_{ij})$, we have

$$a_{ii} = \sum_{k=1}^{i} \hat{\lambda}_{ik} \hat{\hat{\lambda}}_{ki} \qquad \Rightarrow \qquad |\hat{\lambda}_{ii}|^2 = a_{ii} - \sum_{k=1}^{i-1} |\hat{\lambda}_{ik}|^2 \qquad (> 0)$$

$$j > i: \quad a_{ji} = \sum_{k=1}^{i} \hat{\lambda}_{jk} \hat{\hat{\lambda}}_{ki} \qquad \Rightarrow \qquad \hat{\lambda}_{ji} = \big(a_{ji} - \sum_{k=1}^{i-1} \hat{\lambda}_{jk} \hat{\lambda}_{ik}\big) / \hat{\lambda}_{ii}$$

Computation is uniquely defined for $\hat{\lambda}_{ii} > 0$. And $\hat{L}$ can be obtained columnwise ($i = 1, \ldots, i = n$).

Algorithm 3.5 (Cholesky-Decomposition)

for $i = 1 : n$
  $\hat{\lambda}_{ii} := \left(a_{ii} - \sum_{k=1}^{i-1} |\hat{\lambda}_{ik}|^2\right)^{1/2}$
  for $j = (i+1) : n$
    $\hat{\lambda}_{ji} := \left(a_{ji} - \sum_{k=1}^{i-1} \hat{\lambda}_{jk}\, \hat{\lambda}_{ik}\right)/\hat{\lambda}_{ii}$
  end
end ◇

The effort reduces to about $n^3/6$ operations (using symmetry) in comparison to about $n^3/3$ for LU-decomposition.

The property

$$a_{ii} - \sum_{k=1}^{i-1} |\hat{\lambda}_{ik}|^2 > 0 \quad \Rightarrow \quad \sqrt{a_{ii}} \geq |\hat{\lambda}_{ik}| \text{ for all } k \neq i$$

shows that all entries of $\hat{L}$ are small in comparison to the respective diagonal elements in A. Thus the Cholesky-decomposition exhibits very good stability properties.

3.2 Condition and Roundoff error

The aim of this section is two answer the following questions: Defines the linear system a well-conditioned problem? How do roundoff errors influence the LU-decomposition?

First the condition: We compare the solution x of $Ax = b$ ($\det(A) \neq 0$) with the solution of the perturbed problem

$$\tilde{A}\tilde{x} = \tilde{b}, \quad \text{where } \tilde{A} = A + \Delta A,\ \tilde{x} = x + \Delta x,\ \tilde{b} = b + \Delta b. \tag{3.7}$$

We assume that the perturbation ΔA is small enough, such that $A + \Delta A$ is still a regular matrix. This is the case, if

$$\|A^{-1}\| \cdot \|\Delta A\| < 1 \quad \Leftrightarrow \quad \|\Delta A\| < \frac{1}{\|A^{-1}\|}$$

holds for an arbitrary matrix norm, which is consistent to some vector norm.

Proof:

From $(A + \Delta A)x = 0$, we have $x = -A^{-1}\Delta A x$ and thus

$$\|x\| \leq \|A^{-1}\| \cdot \|\Delta A\| \cdot \|x\| \quad \Rightarrow \quad (1 - \|A^{-1}\| \cdot \|\Delta A\|) \cdot \|x\| \leq 0.$$

Hence $\|\Delta A\| < 1/\|A^{-1}\|$ implies $x = 0$ and $A + \Delta A$ is regular. □

Now, starting from $(A + \Delta A)(x + \Delta x) = b + \Delta b$, we have (using $Ax = b$)

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

and in the norm

$$\|\Delta x\| \leq \|A^{-1}\| \cdot (\|\Delta b\| + \|\Delta A\| \cdot \|x\| + \|\Delta A\| \cdot \|\Delta x\|).$$

Consequently, we obtain

$$\|\Delta x\| \leq \frac{\|A^{-1}\|}{1 - \|A^{-1}\| \cdot \|\Delta A\|} \cdot (\|\Delta b\| + \|\Delta A\| \cdot \|x\|)\,.$$

Thus we have found an upper bound on the change in solution x for perturbation in ΔA, Δb (of the problem).

Now we can use the condition number $\mathrm{cond}(A) = \|A\| \cdot \|A^{-1}\|$ (see previous section). Furthermore, if $\|\Delta A\| \leq \varepsilon_A \|A\|$ and $\|\Delta b\| \leq \varepsilon_b \|b\|$ holds for some sufficiently small $\varepsilon_A > 0$, then we find

$$\|\Delta x\| \leq \frac{\|A^{-1}\|}{1 - \|A^{-1}\| \cdot \varepsilon_A \cdot \|A\|} \cdot (\varepsilon_b \cdot \|b\| + \varepsilon_A \|A\| \cdot \|x\|)\,.$$

and thus

$$\frac{\|\Delta x\|}{\|x\|} \leq \frac{\text{cond}(A)}{1 - \varepsilon_A \cdot \text{cond}(A)} \cdot \left(\varepsilon_b \cdot \frac{\|b\|}{\|A\| \cdot \|x\|} + \varepsilon_A \right).$$

Using $\|b\| = \|Ax\| \leq \|A\| \cdot \|x\|$ yields the final result

$$\frac{\|\Delta x\|}{\|x\|} \leq \frac{(\varepsilon_A + \varepsilon_b) \cdot \text{cond}(A)}{1 - \varepsilon_A \cdot \text{cond}(A)}. \tag{3.8}$$

The condition $\text{cond}(A) \geq 1$ is indeed the amplification factor, which describes how changes in the data transform the solution. Assuming $\varepsilon_A = \varepsilon_b = \varepsilon$, we perform the rough estimate

$$\frac{\|\Delta x\|}{\|x\|} \leq \frac{2\varepsilon \cdot \text{cond}(A)}{1 - \varepsilon \cdot \text{cond}(A)} \approx \varepsilon \cdot \text{cond}(A) \tag{3.9}$$

just to observe the order of magnitude in the relative error.

Notice, as always for condition numbers, here we do not investigate roundoff errors, but the inherent problem features! (For propagation of errors)

The user must estimate the condition number and decide whether or not $\text{cond} \cdot \varepsilon \ll 1$ hold. Only in that case, it is feasible to compute an accurate approximation of x.

Formula (3.9) enables a rough backward analysis of algorithms for computing the solution of linear systems. Let $\varepsilon_0 \approx 10^{-t}$ (t digits) be the used machine precision. Gaussian elimination on a computer yields an approximation $\tilde{x}$ to the linear system $Ax = b$. Following backwark analysis, we see this approximation as the exact solution of the system $\tilde{A}\tilde{x} = \tilde{b}$ with $\|\Delta A\| \leq \varepsilon_A \|A\|$ and $\|\Delta b\| \leq \varepsilon_b \|b\|$. Since the result was calculated on a computer, we assume $\varepsilon_A \approx \varepsilon_b \approx \varepsilon_0$ (this is not always the case). Let $\text{cond}(A) \approx 10^c$ with an integer $c \geq 0$. Consequently, we have

$$\frac{\|\Delta x\|}{\|x\|} \approx \varepsilon_0 \cdot \text{cond}(A) \approx 10^{c-t},$$

i.e. we loose about c correct digits in the computed result $\tilde{x}$.

Residual

For an approximation $\tilde{x}$ to the solution x, we define the corresponding *residual* $r := b - A\tilde{x}$. Is there a way to assess the quality of $\tilde{x}$ from the knowledge of r?

The idea: "small residual $r \Rightarrow$ small difference $x - \tilde{x}$" is *wrong*!

Correct is the estimate:

$$\|x - \tilde{x}\| \leq \mathrm{cond}(A) \frac{\|r\|}{\|A\|}\,, \tag{3.10}$$

which follows from $x - \tilde{x} = A^{-1}r$. Assuming standardised A, i.e. $\|A\| = 1$, the condition number gives the correct insight – small $\|r\|$ yields arbitrary large errors for a larger and larger condition number.

Can we obtain something else from the residual? From $r = b - A\tilde{x}$, we have

$$A\tilde{x} = b - r\,,$$

that is, $\tilde{x}$ is the exact solution to the right-hand side $b - r$. If $\|r\|$ is of same order of magnitude as Δb (uncertainty in b), then the result is acceptable (cf. Def. 1.7).

An additional application of r is present in *a posteriori iteration*: an enhancement of the approximation $\tilde{x}$ by a combination of single and double precision.

Roundoff errors in decompositions

Having studied conditions (of the problem), which are independent of the realisation on a computer, we turn to the influence of roundoff errors. A short glance.

Let $A = LU$ be the exact decomposition (without pivoting). Considering roundoff errors we obtain factors $\tilde{L}$ and $\tilde{U}$. For an entry of L, we have the

computation

$$l_{ik} = \Big(\cdots ((a_{ik} - l_{i1}\, u_{1k}) - l_{i2}\, u_{2k}) - \cdots - l_{i,k-1}\, u_{k-1,k} \Big) / u_{kk}\,.$$

For the strong hypothesis (1.3), this transforms

$$\begin{aligned} \tilde{l}_{ik} = \Big(\Big(\cdots ((a_{ik} - \tilde{l}_{i1}\, \tilde{u}_{1k}(1+\mu_1))(1+\sigma_1) - \cdots \\ - \tilde{l}_{i,k-1}\, \tilde{u}_{k-1,k}(1+\mu_{k-1}) \Big) (1+\sigma_{k-1}) / \tilde{u}_{kk} \Big) \cdot (1+\delta)\,. \end{aligned}$$

The coefficients marked by $\tilde{}$ are already computed with roundoff errors.

Using backward analysis (Sautter 1971), one can prove that the numerical approximation $\tilde{x}$ is an exact solution to modified coefficient matrix $\tilde{A}$ (and previous right-hand side b):

$$\tilde{A} \cdot \tilde{x} = b\,.$$

For $\tilde{A}$ we have the estimate

$$|\tilde{A} - A| \le |\tilde{L}| \cdot |\tilde{U}| \cdot (3\varepsilon + \varepsilon^2)\,, \qquad \varepsilon := \frac{n\varepsilon_0}{1 - n\varepsilon_0}\,, \tag{3.11}$$

where $|A|$ denotes the componentwise modulus.

The bound (3.11) is not sufficient to have the stability of Gaussian elimination. We would further need: $|\tilde{L}| \cdot |\tilde{U}| \approx |A|$. But there are indeed decompositions, such that

$$|\tilde{L}| \cdot |\tilde{U}| \gg |\tilde{L} \cdot \tilde{U}| \approx |L \cdot U| = |A|\,,$$

where (3.11) yields a large difference $|\tilde{A} - A|$.

That is, the aim of pivoting is, to find a decomposition such that the product $|\tilde{L}| \cdot |\tilde{U}|$ is small. Partial pivoting yields $|l_{ij}| \le 1$ (Th.3.2), but for the entries u_{ij} exists only the bound

$$|u_{ij}| \le 2^{n-1} a_0 \qquad \text{with} \qquad a_0 := \max_{1 \le i,j \le n} |a_{ij}|\,,$$

which is most times too pessimistic. (But there are example, where this estimate is sharp!)

For special matrices (e.g. tridiagonal), much better estimates can be given.

Total pivoting yields slightly better bounds, but still today there are no strategies to minimise $|\tilde{L}| \cdot |\tilde{U}|$.

Concluding remarks to decompositions

- Gaussian elimination / LU-decomposition with partial pivoting is THE *algorithm to solve linear systems.*
- Gaussian elimination and LU-decomposition represent identical methods even including roundoff errors. They yield solutions with 'small' residuals $r = Ax - b$, but this does not allow to conclude on $\|\tilde{x} - x\|$.
- For known condition number $\text{cond}(A)$, we can roughly estimate the precision of the solution via (3.8) setting ε to machine precision.
  Rule of thumb: $\text{cond}(A) = 10^t \Rightarrow t$ digits are lost!
- For the computation of the condition, there are special algorithms. Exact computation demands $O(n^3)$ operations, whereas rough estimates can be computed cheaply.
- *Rescaling* $A \mapsto D_1 A D_2$ with diagonal matrices D_1, D_2 may reduce the condition significantly. (The aim is to put all values of A in approximately the same order of magnitude.)
- Numerical Software:

  LINPACK (first) library for numerical linear algebra (1979); based on BLAS Level 1 and 2. C- and FORTRAN Codes.

  Special subroutines for LU-decomposition are: SGEFA, SGESL (or DGEFA, DGESL for double precision). SGECO estimates the condition.

  LAPACK (successor) uses in addition BLAS Level 3.

  Subroutines for LU-decomposition: SGETRF, SGETRS (or DGETRF, DGETRS for double precision).

Internet: `http://elib.zib.de/netlib/master/readme.html`

3.3 Linear Least Squares

In this section, we address the solution of overdetermined linear systems. This will apply the method of least squares – dating back to C. F. Gauss (about 1800), where he minimised measurement errors. A special descomposition $A = Q \cdot U$ with orthogonal Q will is suitable for this task.

Problem description

A linear system with more equations than unknowns is called *overdetermined*:

$$Ax = b \qquad \text{with} \quad A \in \mathbb{R}^{m \times n},\ x \in \mathbb{R}^n,\ b \in \mathbb{R}^m \quad \text{and} \quad m > n. \tag{3.12}$$

Such problems occur in data fitting: here measurement data shall be used to calibrate a mathematical model. Thus x denotes the unknown model parameters, b measurements, and A describes the relation of both.

Example 3.3 (Radio-active decay) Given a radio-active substance (e.g. from medical application), which consists of 2 isotopes with constants λ_1, λ_2 for the decay. At times t_j, $j = 1, \ldots, m$ the radio activity $b = (b_1, \ldots, b_m)^\top$ is measured.

We look for the initial concentration x_1 and x_2 of the two isotopes.

To solve this problem, we first need the *functional relation* of measured data b_j and the concentrations x_1, x_2. In this example, it is the exponential: The radio-active substance shows the *decay* $y(t) = x \exp(\lambda t)$.

Consequently, we have for both substances

$$y(t) = x_1 \exp(\lambda_1 t) + x_2 \exp(\lambda_2 t)\,.$$

At the measurement time t_j we obtain the relation

$$y(t_j) = x_1 \exp(\lambda_1 t_j) + x_2 \exp(\lambda_2 t_j)\,, \quad j = 1, \ldots, m$$

which reads in matrix notation ($x \in \mathbb{R}^2,\ \tilde{b} \in \mathbb{R}^m,\ A \in \mathbb{R}^{m\times 2}$)

$$\bar{b} = A \cdot x \quad \text{with}\, \bar{b} := \begin{pmatrix} y(t_1) \\ \vdots \\ y(t_m) \end{pmatrix},\ A := \begin{pmatrix} \exp(\lambda_1 t_1) & \exp(\lambda_2 t_1) \\ \vdots & \vdots \\ \exp(\lambda_1 t_m) & \exp(\lambda_2 t_m) \end{pmatrix},\ x := \begin{pmatrix} x_1 \\ x_2 \end{pmatrix}.$$

In addition, we do not have the exact data $\bar{b}$, but measurements b (which will be used to track back the initial concentration). That is, we need to solve

$$A \cdot x = b\,,$$

which is over-determinant.

Since also the measurements are faulty, we do not expect to find x, such that $A \cdot x = b$ is indeed fulfilled. Therefore we are satisfied to find a minimum of the problem

$$\|A \cdot x - b\| \to \text{min!}$$

$\diamond$

From the example (radio-active decay), to find an exact solution x to (3.12) is useless, since it will often not exist. Therefore we demand that the residual $r(x) := b - Ax$ is as small as possible in some applied norm.

The Euclidian norm is specially suitable, since it can be written using vector operations ($\|x\|_2 = \sqrt{x^\top x}$). Accordingly, the *linear least squares method* reads:

$$\text{Find } \hat{x},\ \text{such that} \quad \|r(\hat{x})\|_2 \le \|r(x)\|_2 \quad \text{for all } \ x \in \mathbb{R}^n. \tag{3.13}$$

For the exact (but unknown) data $\bar{b} := Ax$, problem (3.13) is equivalent to

$$\sum_{i=1}^{m} (\bar{b}_i - b_i)^2 \to \text{ min!} \tag{3.14}$$

– that is, the contradictive data are fit by a method of least squares (see Fig. 8, where the least squares are visualised for Ex. 3.3).

We summarise the problem of *linear least squares*:

Given data:

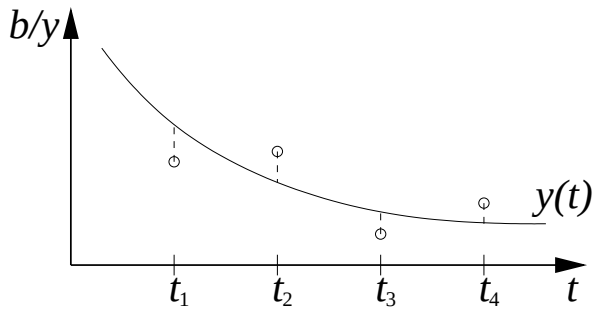


Figure 8: Linear least squares for $y(t) = x_1 e^{\lambda_1 t} + x_2 e^{\lambda_2 t}$

- measurements (t_j, b_j), $j = 1, \ldots, m$
- model (ansatz) functions $g_i(t)$, $i = 1, \ldots, n$
- functional relation $y(t) = x_1 g_1(t) + \ldots + x_n g_n(t)$

Find: the (linear) coefficients $x_1, \ldots, x_n$,

$$F(x_1, \ldots, x_n) := \sum_{i=1}^{m} (y(t_i) - b_i)^2 \rightarrow \text{ min!}$$

Normal Equations

The solution of the linear least squares problem (3.12) is given by the *normal equations*. For a deduction, we can consider geometry or analysis:

Geometrical Approach:
Let $\hat{y} \in \mathbb{R}^m$ be the orthogonal projection of b onto the image of A ($\mathrm{Im}\,(A)$) – considering the Euclidian scalar product $\langle \cdot, \cdot \rangle$. Then an $\hat{x} \in \mathbb{R}^n$ exists satisfying $A\hat{x} = \hat{y}$. If A has full rank n, then $\hat{x}$ is unique. We estimate using the Euclidian norm $\| \cdot \|_2$:

$$\begin{aligned} \|Ax - b\|^2 &= \|Ax - b + A\hat{x} - A\hat{x}\|^2 \\ &= \|A\hat{x} - b\|^2 + \|Ax - A\hat{x}\|^2 + 2\langle A\hat{x} - b, A(x - \hat{x})\rangle \\ &= \|A\hat{x} - b\|^2 + \|Ax - A\hat{x}\|^2 \\ &\geq \|A\hat{x} - b\|^2. \end{aligned}$$

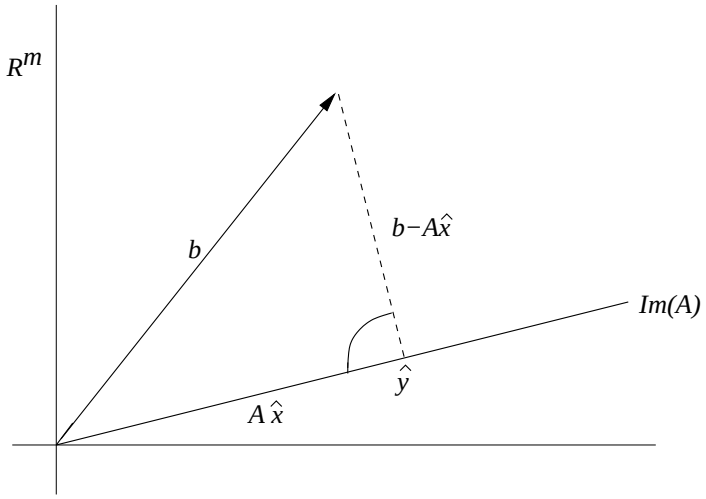


Figure 9: Normal equations: the smallest residual $r(\hat{x}) := b - A\hat{x}$ is orthogonal to the columns of A

Consequently, $\hat{x}$ solves the least squares problem

$$\|A\hat{x} - b\|^2 \leq \|Ax - b\|^2 \quad \text{f.a. } x \in \mathbb{R}^n \quad \Leftrightarrow \quad \langle A\hat{x} - b, Az\rangle = 0 \quad \text{f.a. } z \in \mathbb{R}^n.$$

The conclusion from the right-hand to the left-hand property is shown above. Vice versa, the left-hand property implies

$$\|Ax - A\hat{x}\|^2 + 2\langle A\hat{x} - b, A(x - \hat{x})\rangle \geq 0 \ \text{ f.a. } x \quad \Leftrightarrow \quad \|Az\|^2 + 2\langle A\hat{x} - b, Az\rangle \geq 0 \ \text{ f.a. } z.$$

If $\langle A\hat{x} - b, Az^*\rangle \neq 0$ holds for some z^*, then setting $z = \mu z^*$ for $\mu \in \mathbb{R}$ yields

$$\mu^2\|Az^*\| + 2\mu\langle A\hat{x} - b, Az\rangle \geq 0.$$

For $\mu \to 0$ with appropriate sign, the left-hand side becomes negative, i.e. a contradiction arises.

From orthogonality of the residual $r = b - A\hat{x}$ and $\mathrm{Im}(A)$, we conclude that

$$\hat{x}^T A^T Ax = b^T Ax \quad \text{for all } x \in \mathbb{R}^n,$$

and therefore $\hat{x}$ fulfills the normal equations

$$A^T A\hat{x} = A^T b. \tag{3.15}$$

Notice, the equivalence to $A^T r(\hat{x}) = 0$.

Minimisation Approach:
To demand (3.13) is equal to minimising the function $r(x)^T r(x)$, that is,

$$F(x) = r(x)^T r(x) = x^T A^T A x - 2x^T A^T b + b^T b \;\rightarrow\; \min!$$

Necessary for a minimum of a function $F : \mathbb{R}^n \rightarrow \mathbb{R}$ at $\hat{x}$ is the condition grad $F(\hat{x}) = 0$. The derivative of F with respect to the unknowns x yields

$$\text{grad } F(x)^\top = \left(\frac{\partial F(x)}{\partial x_1}, \ldots \frac{\partial F(x)}{\partial x_n} \right)^\top = 2A^T A x - 2A^T b \,.$$

Componentweise:

$$\begin{aligned} \frac{\partial F(x)}{\partial x_j} &= \frac{\partial}{\partial x_j} \left(x^T A^T A x + 2x^T A^T b \right) = e_j^T A^T A x + x^T A^T A e_j - 2e_j^T A^T b \\ &= 2 \left(\left[A^T A x \right]_j - \left[A^T b \right]_j \right) \end{aligned}$$

From that we have (3.15).

The normal equations are not only necessary, they are also sufficient! Write $r(x) = r(\hat{x}) + A(x - \hat{x})$, then we deduce

$$r(x)^T r(x) = r(\hat{x})^T r(\hat{x}) + 0 + (x - \hat{x})^T A^T A (x - \hat{x}) \geq r(\hat{x})^T r(\hat{x}) \,,$$

i.e., $\hat{x}$ is a minimum. Equality holds only for $A(x - \hat{x}) = 0$.

The next theorem gives the according conclusion.

Theorem 3.4 *Every solution $\hat{x}$ of the linear least squares problem (3.13) satisfies the normal equations (3.15). The minimiser $\hat{x}$ unique, if and only if the columns of A are linear independent, i.e., rank$(A) = n$. The minimal residual $r(\hat{x})$ is always unique.*

In the case rang$(A) = n$, $A^T A$ is positive definite, and therefore the normal equations exhibit a unique solution $\hat{x}$. This case we assume in the following.

Condition of the least squares problem

Linear least squares problem may be more sensitive to perturbations in the data than linear systems. Without proof, we quote the following results:

We consider the linear least squares problem with matrix A ($\operatorname{rank}(A) = n$) and vector b. A QR-decomposition of A shall be given by

$$A = Q \begin{pmatrix} R \\ 0 \end{pmatrix}. \tag{3.16}$$

Let $\tilde{A} = A + \Delta A$ and $\tilde{b} = b + \Delta b$ denote the modified data. If the perturbances are small, then we obtain w.r.t. the Euclidean norm (by neglecting higher-order terms)

$$\begin{aligned} \frac{\|\Delta x\|_2}{\|x\|_2} \quad \dot{\leq} \quad & \operatorname{cond}(R)\frac{\|\Delta A\|}{\|A\|} + \operatorname{cond}(R)^2 \frac{\|r\|}{\|Ax\|}\frac{\|\Delta A\|}{\|A\|} \\ & + \operatorname{cond}(R)\frac{\|b\|}{\|Ax\|}\frac{\|\Delta b\|}{\|b\|}. \end{aligned} \tag{3.17}$$

Remark that the subordinate matrix norm is defined for rectangular matrices, too. We define the angle φ via

$$\tan\varphi = \frac{\|r\|}{\|Ax\|}, \qquad 0 \leq \varphi < \tfrac{\pi}{2}.$$

Then it follows

$$\begin{aligned} \frac{\|\Delta x\|_2}{\|x\|_2} \quad \dot{\leq} \quad & \operatorname{cond}(R)\frac{\|\Delta A\|}{\|A\|} + \operatorname{cond}(R)^2 \tan\varphi \frac{\|\Delta A\|}{\|A\|} \\ & + \operatorname{cond}(R)\sqrt{1+\tan^2\varphi}\frac{\|\Delta b\|}{\|b\|}. \end{aligned} \tag{3.18}$$

For small φ, i.e. $\operatorname{cond}(R)\tan\varphi \leq 1$, the condition of the problem is determined by $\operatorname{cond}(R)$. For $\varphi \to \frac{\pi}{2}$, the number $\operatorname{cond}(R)^2$ is dominating, which may result in an ill-conditioned problem. Remark that the discussion above considers the least squares problem and not a special algorithm to solve it.

For comparison, we observe the normal equations to solve the problem. Using (3.16), we obtain w.r.t. the Euclidean norm

$$\text{cond}(A^\top A) = \text{cond}(R^\top R) = \text{cond}(R)^2.$$

Hence the condition number of the normal equations is always dominated by the factor $\text{cond}(R)^2$.

Computation

The cheapest way to compute the solution $\hat{x}$ is via the normal equations (3.15). One can combine the computation of $A^T A$ and the Cholesky-decomposition, as well as the the computation of $A^T b$ and the forward substitution.

For $m = n$, it holds $\text{cond}(A^T A) = \text{cond}(A)^2$ w.r.t. the Euclidean norm. Therefore this strategy is quite imprecise. But, the factor $\text{cond}(A)^2$ is somehow problem inherent. A posteriori iteration can enhance the results (significantly). But one has to be careful using Cholesky-decomposition!

Example 3.4 Given matrix A

$$A = \begin{pmatrix} 1 & 1 & 1 \\ \varepsilon & 0 & 0 \\ 0 & \varepsilon & 0 \\ 0 & 0 & \varepsilon \end{pmatrix} \quad \Rightarrow \quad A^\top A = \begin{pmatrix} 1+\varepsilon^2 & 1 & 1 \\ 1 & 1+\varepsilon^2 & 1 \\ 1 & 1 & 1+\varepsilon^2 \end{pmatrix}$$

Which rank of $A^T A$ is analytically computed, which rank is obtained for $\varepsilon = 10^{-5}$ and machine precision $\varepsilon_0 = 6 \cdot 10^{-8}$? The exact computation yields $\text{rank}(A^T A) = 3$. Using $\varepsilon = 10^{-5}$ and $\varepsilon_0 = 6 \cdot 10^{-8}$ yields $\text{fl}(1 + \varepsilon^2) = 1$ and therefore $\text{rank}(\text{fl}(A^T A)) = 1$, i.e., the normal equations are singular.

Another way to solve the least squares problem is standard: It is based on *orthogonal eliminations*.

For an orthogonal matrix $Q \in \mathbb{R}^{m \times m}$ holds:

$$\|r(x)\|_2 = \|Q \cdot r(x)\|_2 = \|Q \cdot b - Q \cdot A \cdot x\|_2$$

the norm of the residual is unchanged.

If we assume, that we can choose Q, such that $Q \cdot A$ is a regular upper triangular matrix $R \in \mathbb{R}^{n \times n}$ and a block of zeros $0 \in \mathbb{R}^{m-n \times n}$,

$$Q \cdot A = \begin{pmatrix} R \\ 0 \end{pmatrix},$$

then the linear least squares problem can be rewritten as:

$$\|Q \cdot b - Q \cdot A \cdot x\|_2 = \left\| \begin{pmatrix} c \\ d \end{pmatrix} - \begin{pmatrix} R \cdot x \\ 0 \end{pmatrix} \right\|_2 \rightarrow \text{ min !}$$

Since

$$\left\| \begin{pmatrix} c \\ d \end{pmatrix} - \begin{pmatrix} R \cdot x \\ 0 \end{pmatrix} \right\|_2^2 = \|c - R \cdot x\|_2^2 + \|d\|_2^2$$

the minium in $\hat{x}$ can only be attained if

$$\|c - R \cdot \hat{x}\|_2^2 = 0 \quad \Leftrightarrow \quad R \cdot \hat{x} = c\,.$$

The solution $\hat{x}$ of the least squares problem is then obtained by a simple backward substitution.

Are there such orthogonal transforms, which can be constructed and stable be computed? The answer is YES.

Householder Transformation and QR-Decomposition

The matrix

$$T := I - 2 \cdot v \cdot v^H$$

with $v \in \mathbb{C}^m$ and $\|v\|_2 = 1$ is called a reflection.

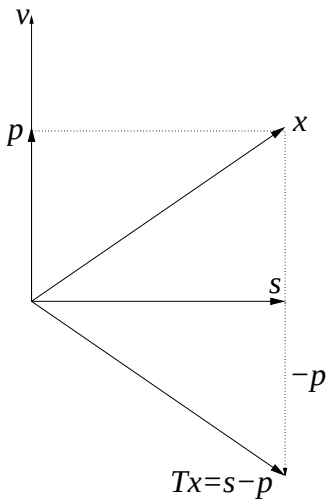


Figure 10: Geometric interpretation of reflection T.

Scheme for T:

$$T = \begin{pmatrix} 1-2v_1^2 & -2v_1v_2 & \dots & -2v_1v_{n-1} & -2v_1v_n \\ -2v_2v_1 & 1-2v_2^2 & \dots & -2v_2v_{n-1} & -2v_2v_n \\ & & \ddots & & \\ -2v_{n-1}v_1 & -2v_{n-1}v_2 & \dots & 1-2v_{n-1}^2 & -2v_{n-1}v_n \\ -2v_nv_1 & -2v_nv_2 & \dots & -2v_nv_{n-1} & 1-2v_n^2 \end{pmatrix}$$

An arbitrary vector $x \in \mathbb{C}^m$ can be decomposed in $x = p + s$ with $p := (v^H x)v$ parallel to v and $s := x - p$ orthogonal to v.

The matrix T reflects the parallel part p on s,

$$T \cdot (s+p) = T \cdot (s + (v^H x)v) = (I - 2vv^H) \cdot (s + (v^H x)v) = s - p\,.$$

The reflection T has the following properties:

(i) T is hermitian (in real: symmetric), $T^H = T$

(ii) T is involutory, $T^{-1} = T$

(iii) T is unitary (in real: orthogonal).

Instead of a normed vector v, we can define T as

$$T := I - uu^H/\kappa\,, \quad \kappa := \frac{1}{2}u^H u$$

with $u \in \mathbb{C}^m/\{0\}$.

The operation of T on vector x:

$$y = T \cdot x = x - uu^H x/\kappa$$

needs no matrix vector multiplication! First, form the scalar product $\sigma := u^H x/\kappa$ and then $y := x - \sigma u$. Analog for the operation on matrices

$$T \cdot A = A - u(u^H A/\kappa)\,, \quad \text{each column as before.}$$

The matrix A is not necessary quadratic.

In our application (for least squares), one has to deal with a special case. T and u are to be determined, such that the operation on given x is a multiple of the first unit vector e_1

$$T \cdot x = y = -\zeta e_1 = \begin{pmatrix} -\zeta \\ 0 \\ \vdots \\ 0 \end{pmatrix}.$$

The following choice does the job:

$$\begin{aligned} \zeta &:= \begin{cases} \|x\|_2 x_1/|x_1| & \text{if } x_1 \neq 0 \\ \|x\|_2 & \text{if } x_1 = 0 \end{cases} \\ u &:= x + \zeta e_1 = (x_1 + \zeta, x_2, \ldots, x_m)^T \\ \kappa &:= x^H x + \|x\|_2 \cdot |x_1| \\ T &:= I - uu^H/\kappa \end{aligned} \tag{3.19}$$

This reflection T is then called *Householder transformation.*

Theorem 3.5 (Householder transformation) *The Householder transform T from (3.19) reflects x onto $Tx = -\zeta e_1 = (-\zeta, 0, \ldots, 0)^T$.*

Proof: Exercise. □

Now, a sequence of Householder transformations can be used to transform matrix A to an upper triangular form.

Step 1: For x in (3.19) choose first column of $A \in \mathbb{C}^{m\times n}$ and form T_1. Then

$$T_1 A = \left(\begin{array}{c|ccc} -\zeta_1 & \star & \cdots & \star \\ \hline 0 & & & \\ \vdots & a_2^{(1)} & \cdots & a_n^{(1)} \\ 0 & & & \end{array}\right).$$

The first column reduces and all others are transformed.

Step 2: Apply Householder transformation of dimension $m-1$ to the first column of the remaining matrix.

This step can be described by using Householder transformation $T_2 \in \mathbb{C}^{m\times m}$, where the first component of u is put to zero. Thus the operation of T_2 keeps the first row and column in $T_1 A$ unchanged:

$$T_2 T_1 A = \left(\begin{array}{c|c|ccc} -\zeta_1 & \star & \star & \cdots & \star \\ \hline 0 & -\zeta_2 & \star & \cdots & \star \\ \hline 0 & 0 & & & \\ \vdots & \vdots & a_3^{(2)} & \cdots & a_n^{(2)} \\ 0 & 0 & & & \end{array}\right).$$

After n steps all columns are reduced and A is transformed in an upper triangular matrix and a zero block:

$$T_n \cdot \ldots \cdot T_1 \cdot A = \begin{pmatrix} R \\ 0 \end{pmatrix} \quad \text{and } A = T_1 \cdot \ldots \cdot T_n \begin{pmatrix} R \\ 0 \end{pmatrix} = Q \cdot \begin{pmatrix} R \\ 0 \end{pmatrix}$$

With $Q := T_1 \cdot \ldots \cdot T_n$ one has the so-called *QR-Decomposition* of A.

The product $Q = T_1 \cdot \ldots \cdot T_n$ of Householder transformations is as well unitary or orthogonal: $Q^H Q = I$.

That is, we have reached the aim: we can solve the linear least squares problem via QR-decomposition (notice Q^T instead Q):

$$\begin{aligned} \|b - Ax\|_2^2 &= \|Q^T b - Q^T A x\|_2^2 \\ &= \left\| \begin{pmatrix} c \\ d \end{pmatrix} - \begin{pmatrix} R \cdot x \\ 0 \end{pmatrix} \right\|_2^2 \\ &= \|c - R \cdot x\|_2^2 + \|d\|_2^2 \end{aligned}$$

The solution $\hat{x}$ of the least squares is deduced by a backward substitution $R\hat{x} = c$.

For implementation, one has be take care, that matrices T_j are not formed explicitly, but just the operations on A are computed. In the jth step, T_j is applied to $A^{(j-1)} = T_{j-1} \ldots T_1 A$ and $b^{(j-1)} = T_{j-1} \ldots T_1 b$. Since $T_j = I - u_j u_j^T / \kappa_j$, we have the computation

$$A^{(j-1)} \mapsto A^{(j)} = T_j A^{(j-1)} = A^{(j-1)} - u_j y_j^T \,, \quad y_j^T := u_j^T A^{(j-1)} / \kappa_j \,.$$

Complexity: In the jth step we need

$(n-j+1)(m-j+1)$ additions/multiplications for $u_j^T A^{(j-1)}$
$(n-j+1)(m-j+1)$ additions/multiplications for $u_j y_j^T$

(the first $j-1$ components of u_j are zero).

In total

$$\sum_{j=1}^{n} 2(n-j+1)(m-j+1) \approx mn^2 - \frac{n^3}{3}$$

operations. For $m \gg n$ this is the double as for Cholesky-decomposition and normal equations.

Remarks

- In the special case $m = n$, the QR-decomposition is an alternative to LU-decomposition (the zero block vanishes). If A has complete rank, then QR-decomposition yields the solution of the linear system $Ax = b$. The effort is $2/3n^3$ operations (multiplications), approximately the double of the LU-decomposition. For stability reasons, one applies from time to time indeed QR-decomposition.
- For positive diagonal entries $r_{ii} > 0, i = 1, \ldots, n$, the QR-decomposition is unique.
- Pivoting: Businger/Golub (1965) have introduced a technique for the rank deficient case (i.e., if zero columns occur as first column of the remaining part).
- Instead of using Householder transformations, we can obtain the QR-decomposition by Givens-Rotations.

Chapter 4

Iterative Methods for Linear Systems

In this section, we discuss the numerical solution of linear systems

$$Ax = b, \quad A \in \mathbb{R}^{n\times n}, \quad \det A \neq 0, \quad x, b \in \mathbb{R}^n \tag{4.1}$$

by means of *iterative methods*. Since an LU-decomposition of A demands $\mathcal{O}(n^3)$ floating point operations, the direct solving becomes extremely costly for large n. In practical applications, large matrices A are typically sparse, i.e. just a few elements are not equal to zero. To exploit the sparsity, special algorithms for LU-decomposition try to minimise the number of fill-ins (non-zero numbers created during elimination) by permutations of rows and columns using a restricted pivoting. Since floating point operations are realised only for non-zero elements in these algorithms, such *direct sparse solvers* are suitable for $500 \leq n \leq 50000$. In the following, we introduce iterative methods, which are successfully applied for significantly larger dimensions n. The computational effort of one step must not be much larger than a matrix-vector-multiplication (where the sparsity is considered and operations including zeros are omitted).

Motivation for iterative methods:
For $\text{cond}(A) \approx 1$, Gaussian elimination on a computer yields an approxi-

mation $\tilde{x}$ for $x^* = A^{-1}b$ with

$$\frac{\|\tilde{x} - x^*\|}{\|x^*\|} \approx \varepsilon_0 \approx 10^{-16}$$

in some norm. In practice, an accuracy

$$\frac{\|\tilde{x} - x^*\|}{\|x^*\|} \approx 10^{-4}$$

is often sufficient, for example, if the exact solution of the linear system is only an approximation of another problem. Hence the idea is to use an alternative method, which is less accurate ($\varepsilon \approx 10^{-4}$ instead of $\varepsilon \approx 10^{-16}$) but needs significantly lower computational effort ($\mathcal{O}(n^2)$ instead of $\mathcal{O}(n^3)$).

4.1 Illustrative example

We consider the *Dirichlet problem* (special boundary-value problem of an elliptic partial differential equation) for a function $u : \Omega \to \mathbb{R}$, involving the unit-square $\Omega := \{(x, y) : 0 < x, y < 1\}$, with $u \in C^2$, i.e.

$$\begin{aligned} -\Delta u = -u_{xx} - u_{yy} &= f(x, y), \qquad (x, y) \in \Omega \\ u(x, y) &= 0, \qquad (x, y) \in \partial\Omega \end{aligned} \tag{4.2}$$

with predetermined function $f : \Omega \to \mathbb{R}$, $f \in C^0$. Now the partial derivatives are discretised on a uniform grid Ω_h using the step size $h := 1/(M+1)$ for $M \in \mathbb{N}$

$$\Omega_h := \{(x_i, y_j) = (ih, jh) : i, j = 1, \ldots, M\}. \tag{4.3}$$

Symmetric differences of second order yield an approximation for $u_{i,j} = u(x_i, y_j)$

$$-\frac{u_{i-1,j} - 2u_{i,j} + u_{i+1,j}}{h^2} - \frac{u_{i,j-1} - 2u_{i,j} + u_{i,j+1}}{h^2} \doteq f(x_i, y_j) \tag{4.4}$$

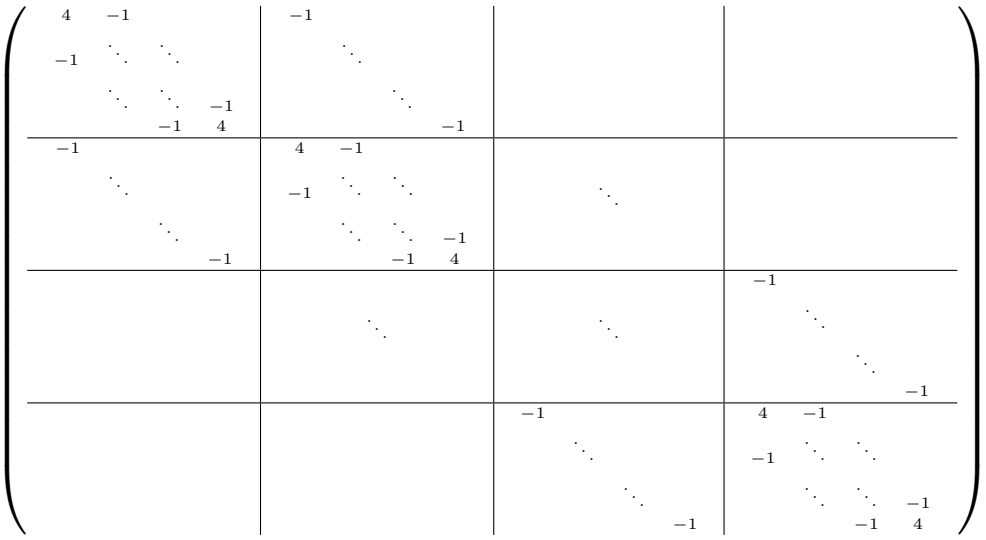
or equivalent using $f_{i,j} = f(x_i, y_j)$

$$4u_{i,j} - u_{i-1,j} - u_{i+1,j} - u_{i,j-1} - u_{i,j+1} = h^2 f_{i,j} \tag{4.5}$$

for $i, j = 1, \ldots, M$. We arrange the unknowns and the right-hand side in the form

$$\begin{aligned} u &= (u_{1,1}, u_{2,1}, \ldots, u_{M,1}, u_{1,2}, \ldots, u_{1,M}, \ldots, u_{M,M})^T \\ b &= h^2(f_{1,1}, f_{2,1}, \ldots, f_{M,1}, f_{1,2}, \ldots, f_{1,M}, \ldots, f_{M,M})^T, \end{aligned} \tag{4.6}$$

which results in the linear system $Ax = b$ of dimension $n = M^2$. The matrix A exhibits the structure



. (4.7)

The matrix A is sparse, because each row contains at most five non-zero elements. Moreover, A is symmetric and positive definite. However, a Cholesky-decomposition $A = LL^T$ would create a large number of fill-ins, i.e. L is not sparse. On the contrary, matrix-vector-multiplications using A are relatively cheap, since just non-zeros elements have to be considered.

4.2 Classical iterative methods

A classical iterative method for the problem (4.1) features the form

$$x^{k+1} = \Phi(x^k), \qquad k = 0, 1, 2, \ldots \tag{4.8}$$

for given starting value x^0. In classical methods, the function Φ remains the same in all steps. On the contrary, further types of iterative methods

use different functions Φ^k in each step. We assume that the exact solution x^* of (4.1) is the unique fixed point of the iteration (4.8).

A classical iterative method is produced by the choice of a matrix $B \in \mathbb{R}^{n\times n}$, $\det B \neq 0$ using the splitting

$$Bx + (A - B)x = b \quad \Rightarrow \quad Bx^{k+1} + (A - B)x^k = b, \tag{4.9}$$

which represents a linear system with solution x^{k+1} and thus it holds

$$x^{k+1} = x^k - B^{-1}(Ax^k - b) = (I - B^{-1}A)x^k + B^{-1}b. \tag{4.10}$$

To compute x^{k+1}, we require a matrix-vector-multiplication with respect to A and we have to solve a linear system including the matrix B. The iteration matrix $I - B^{-1}A$ is used exclusively for the analysis of convergence.

In this context, the matrix B shall have two properties:

1. Linear systems involving the matrix B are cheap to solve, which results in a low computational effort of the iteration.
2. The matrix B is a reasonable approximation of A, i.e. B contains essential information of A, which ensures the convergence of the iteration.

Theorem 4.1 (Convergence of classical methods) *The iterative technique (4.10) is convergent for any starting value, if and only if it holds*

$$\rho(I - B^{-1}A) < 1. \tag{4.11}$$

Sufficient for the convergence of (4.10) using any starting value is the property

$$\|I - B^{-1}A\| < 1, \tag{4.12}$$

where $\|\cdot\|$ is the subordinate matrix norm corresponding to an arbitrary vector norm.

Proof:

Considering the error $f^k := x^k - x^*$ ($x^* = A^{-1}b$) and using

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

we obtain the recursion

$$f^k = (I - B^{-1}A)f^{k-1} \quad \Rightarrow \quad f^k = (I - B^{-1}A)^k f^0,\ k = 0, 1, 2, \ldots.$$

Let (4.10) be convergent, i.e. $\lim f^k = 0$. Given an eigenvalue λ of $I - B^{-1}A$, we set f^0 to the corresponding eigenvector. Thus it holds $f^k = \lambda^k f^0$ and the convergence implies $|\lambda| < 1$.

Vice versa, let $\rho(I - B^{-1}A) < 1$. For the special matrix $I - B^{-1}A$ and a given $\varepsilon > 0$, a vector norm $\|\cdot\|_\varepsilon$ exists such that the corresponding subordinate matrix norm satisfies

$$\|I - B^{-1}A\|_\varepsilon \leq \rho(I - B^{-1}A) + \varepsilon.$$

Therefore we obtain

$$\|(I - B^{-1}A)^k\|_\varepsilon \leq \|I - B^{-1}A\|_\varepsilon^k \leq \big(\rho(I - B^{-1}A) + \varepsilon\big)^k = \mu^k$$

involving $\mu < 1$ for small ε. Hence $\lim(I - B^{-1}A)^k = 0$ and $\lim f^k = 0$ holds for all f^0.

If λ is an eigenvalue of a matrix $C \in \mathbb{R}^{n\times n}$ and v a corresponding eigenvector, then it holds

$$|\lambda| \cdot \|v\| = \|\lambda v\| = \|Cv\| \leq \|C\| \cdot \|v\| \quad \Rightarrow \quad |\lambda| \leq \|C\|$$

and thus $\rho(C) < \|C\|$ for any matrix norm, which is consistent to some vector norm. Hence $\|I - B^{-1}A\| < 1$ implies $\rho(I - B^{-1}A) < 1$ and the method is convergent owing to the first part of the theorem. □

In particular, the proof shows the estimate

$$\|x^{k+1} - x^*\| \leq \|I - B^{-1}A\| \cdot \|x^k - x^*\| \tag{4.13}$$

for an arbitrary vector norm and subordinate matrix norm. Thus the property $\|I - B^{-1}A\| < 1$ implies global linear convergence in the used norm. The speed of convergence using the method (4.10) increases, the smaller the norm of the iteration matrix becomes. The following theorem illustrates a statement, which is independent from the applied matrix norm.

Theorem 4.2 (Speed of convergence) *In the iterative method (4.10), the error $f^k = x^k - x^*$ satisfies*

$$\sup_{f^0 \neq 0} \limsup_{k \to \infty} \sqrt[k]{\frac{\|f^k\|}{\|f^0\|}} = \rho(I - B^{-1}A) \tag{4.14}$$

using an arbitrary vector norm $\|\cdot\|$.

Proof :

Let σ denote the left-hand side of (4.14). By choosing f^0 as eigenvector corresponding to an eigenvalue with $|\lambda| = \rho(I - B^{-1}A)$, we see $\sigma \geq \rho(I - B^{-1}A)$. For given $\varepsilon > 0$, a vector norm $\|\cdot\|_\varepsilon$ exists such that the subordinate matrix norm yields

$$\|I - B^{-1}A\|_\varepsilon \leq \rho(I - B^{-1}A) + \varepsilon.$$

Since all norms are equivalent in $\mathbb{R}^n$, we obtain constants $C_1, C_2 > 0$ with

$$C_1 \cdot \|x\| \leq \|x\|_\varepsilon \leq C_2 \cdot \|x\| \quad \text{for all } x \in \mathbb{R}^n.$$

For arbitrary $f^0 \neq 0$, it follows that

$$\begin{aligned} \|f^k\| &\leq \tfrac{1}{C_1}\|f^k\|_\varepsilon = \tfrac{1}{C_1}\|(I - B^{-1}A)^k f^0\|_\varepsilon \\ &\leq \tfrac{1}{C_1}\|I - B^{-1}A\|_\varepsilon^k \|f^0\|_\varepsilon \\ &\leq \tfrac{C_2}{C_1}\left(\rho(I - B^{-1}A) + \varepsilon\right)^k \|f^0\| \end{aligned}$$

or

$$\sqrt[k]{\frac{\|f^k\|}{\|f^0\|}} \leq \sqrt[k]{\frac{C_2}{C_1}} \cdot \left(\rho(I - B^{-1}A) + \varepsilon\right).$$

Since $\lim \sqrt[k]{C_2/C_1} = 1$, we obtain $k \leq \rho(I - B^{-1}A) + \varepsilon$ for arbitrary $\varepsilon > 0$, which implies $\sigma \leq \rho(I - B^{-1}A)$. □

In the following, we introduce the prominent classical methods, which arise in view of the splitting

$$A = D + L + U, \tag{4.15}$$

where D is a diagonal matrix including the diagonal of A and L, U contain the lower and upper triangular part (without diagonal) of A, respectively.

Jacobi method

A simple iterative method is generated by choosing $B = D$ in (4.10). We assume $a_{ii} \neq 0$ for all $i = 1, \ldots, n$, which implies $\det B \neq 0$. For each component $i = 1, \ldots, n$, the formula (4.10) results in

$$x_i^{k+1} = \frac{1}{a_{ii}} \left(b_i - \sum_{j \neq i} a_{ij} x_j^k \right). \tag{4.16}$$

Thereby, each component can be calculated separately.

If the diagonal of A represents the crucial part of the whole matrix, then the Jacobi method is convergent.

Definition 4.1 *The matrix A satisfies the* strong row sum criterion*, if it holds*

$$|a_{ii}| > \sum_{j \neq i} |a_{ij}| \quad \textit{for} \quad i = 1, \ldots, n. \tag{4.17}$$

The matrix A satisfies the strong column sum criterion*, if it holds*

$$|a_{jj}| > \sum_{i \neq j} |a_{ij}| \quad \textit{for} \quad j = 1, \ldots, n. \tag{4.18}$$

Both criteria imply the convergenge of the Jacobi method by Theorem 4.1. From condition (4.17), we obtain

$$\|I - B^{-1}A\|_\infty = \max_{i=1,\ldots,n} \frac{1}{|a_{ii}|} \sum_{j \neq i} |a_{ij}| < 1 \tag{4.19}$$

and, from condition (4.18), it follows

$$\|I - B^{-1}A\|_1 = \max_{j=1,\ldots,n} \frac{1}{|a_{jj}|} \sum_{i \neq j} |a_{ij}| < 1. \tag{4.20}$$

Demanding additional properties, the convergence of Jacobi's method can also be guaranteed for weak sum criteria (replace $>$ by $\geq$ in (4.17),(4.18)).

Gauss-Seidel method

Now we apply more information from the matrix A by considering $B = D + L$. Thus the linear system corresponding to a triangular matrix has to be solved, which can be done directly by forward substitution in case of $a_{ii} \neq 0$ for all $i = 1, \ldots, n$. For each component $i = 1, \ldots, n$, we obtain the formula

$$x_i^{k+1} = \frac{1}{a_{ii}} \left(b_i - \sum_{j<i} a_{ij} x_j^{k+1} - \sum_{j>i} a_{ij} x_j^k \right). \tag{4.21}$$

Hence the components have to be computed successively.

The conditions (4.17) and (4.18) are sufficient for the convergence of the Gauss-Seidel method, too. For some matrix classes, the Gauss-Seidel technique converges faster than the Jacobi method.

The Gauss-Seidel method can also be defined by $B = D + U$. Furthermore, symmetric techniques exist alternating between a step with $B = D + L$ and next step with $B = D + U$, which can be written in classical form, too.

Relaxation methods

More general, the matrix B in (4.10) may depend on a parameter $\omega \in \mathbb{R}$. Consequently, the aim is to choose ω such that $\rho(I - B(\omega)^{-1}A)$ is minimal and thus covergence becomes fast. In relaxation methods, the matrix

$$B(\omega) = \frac{1}{\omega}(D + \omega L) \qquad (\omega > 0) \tag{4.22}$$

is selected and ω is called the *relaxation parameter*. Let $\tilde{x}^{k+1}$ be the result from the Gauss-Seidel method (4.21). By (4.22), it can be shown that

$$x^{k+1} = (1 - \omega)x^k + \omega \tilde{x}^{k+1}. \tag{4.23}$$

Setting $0 < \omega \leq 1$ is called *underrelaxation* and results in a convex combination of the old approximation and the Gauss-Seidel approximation. If

$\omega > 1$ is applied, the procedure is called *overrelaxation* and the *successive overrelaxation (SOR) method* arises.

For some matrix classes, an optimal relaxation parameter $1 \leq \hat{\omega} < 2$ can be computed explicitly. Accordingly, the convergence becomes much faster than for the Gauss-Seidel method.

Application to illustrative example

The convergence of classical methods for linear systems involving the example (4.7) is analysed now. For the three introduced techniques, the spectral radius of the iteration matrix $I - B^{-1}A$ can be calculated directly. We obtain the following values.

$$\begin{aligned} \text{Jacobi method :} \qquad \rho(I - D^{-1}A) &= \cos\left(\tfrac{\pi}{M+1}\right) \\ \text{Gauss-Seidel method :} \quad \rho(I - (D+L)^{-1}A) &= \cos^2\left(\tfrac{\pi}{M+1}\right) \\ \text{SOR method :} \qquad \rho(I - B(\hat{\omega})^{-1}A) &= \frac{\cos^2\left(\frac{\pi}{M+1}\right)}{\left(1+\sin\left(\frac{\pi}{M+1}\right)\right)^2} \end{aligned}$$

By Theorem 4.1, the convergence is guaranteed in all three schemes. Theorem 4.2 shows that the magnitude of the spectral radius determines the speed of convergence, i.e. the number of required steps for achieving a given accuracy. For $M \to \infty$, we have $\rho \to 1$, which slows down the convergence in all three methods. However, considering a fixed M, the Gauss-Seidel method is faster than Jacobi's method. Furthermore, we conclude for large M

$$\begin{aligned} &\rho(I - D^{-1}A)^{\sigma} = \rho(I - B(\hat{\omega})^{-1}A) \\ \Rightarrow \quad &\sigma = \tfrac{\ln \rho(I - B(\hat{\omega})^{-1}A)}{\ln \rho(I - D^{-1}A)} \approx \tfrac{4(M+1)}{\pi}, \end{aligned} \tag{4.24}$$

which implies that the SOR method using optimal relaxation parameter is more than M times faster than the Jacobi method.

Iterative refinement

The class of iterative methods (4.10) includes the *iterative refinement*, too. If a linear system $Ax = b$ is solved directly by LU-decomposition using a computer, then rounding errors cause a perturbed solution $\tilde{x} \approx A^{-1}b$. If the condition of A is not too large, then this approximation can be improved to machine precision.

Gaussian elimination employing a computer yields a perturbed decomposition $A \approx \hat{L}\hat{U}$, i.e.

$$A = \hat{L}\hat{U} + E \tag{4.25}$$

with an unknown error matrix E. Now we choose $B = \hat{L}\hat{U}$ in (4.10). Remark that $\hat{L}, \hat{U}$ do not stand for L, U in (4.15) here. Thus we obtain the iteration

$$x^{k+1} = x^k - (\hat{L}\hat{U})^{-1}(Ax^k - b) = x^k - (\hat{L}\hat{U})^{-1}r^k, \tag{4.26}$$

where r^k is the residual of the kth approximation. The iteration (4.26) is cheap, since the decomposition $\hat{L}\hat{U}$ is already computed and thus just forward- and backward substitutions are necessary. Furthermore, we have $(\hat{L}\hat{U})^{-1}A \approx I$, because just rounding errors interfere with $\hat{L}, \hat{U}$. Consequently, the spectral radius of the iteration matrix is small and convergence becomes very fast. Moreover, we hold a good starting value by the direct solution $x^0 = (\hat{L}\hat{U})^{-1}b$. In general, only two steps are sufficient to produce a result with accuracy corresponding to machine precision.

Computing the residual r^k, the subtraction causes cancellation due to the property $Ax^k \approx b$. Hence this value has to computed using higher precision than the machine precision applied in the other operations. Otherwise, the result can not be obtain up to machine precision. If x^0 represents a rough approximation, then iterative refinement with exclusively constant precision may increase the number of correct digits, but can not achieve all digits to be correct.

4.3 Conjugate gradient method

The *conjugate-gradient (CG) method* of Hestenes and Stiefel represents an iterative scheme for solving the linear system $Ax = b$ including a symmetric and positive definite matrix $A \in \mathbb{R}^{n\times n}$. Like for classical methods, just one maxtrix-vector multiplication using A is necessary in each step.

In the kth step, the difference $x^k - x^0$ shall be situated in the *Krylov space*

$$\mathcal{K}_k := \operatorname{span}\{r^0, Ar^0, A^2r^0, \ldots, A^{k-1}r^0\}, \tag{4.27}$$

where $r^0 = b - Ax^0$ represents the residual for the starting value. This requirement follows from the Theorem of Cayley-Hamilton, which implies that a polynomial q of degree $n-1$ exists satisfying $A^{-1} = q(A)$. Hence it follows

$$x^* - x^0 = A^{-1}(b - Ax^0) = A^{-1}r^0 = q(A)r^0 \in \mathcal{K}_n. \tag{4.28}$$

If the case $\mathcal{K}_l = \mathcal{K}_{l+1}$ arises for some $l < n$, then it holds $\mathcal{K}_l = \mathcal{K}_n$, too. Thus the space $x^0 + \mathcal{K}_l$ already contains the exact solution x^*.

Starting from x^0, we can try to obtain the exact solution by increasing successively the dimension of the Krylov space and determining x^k in an optimal sense with respect to $\mathcal{K}_k$. Not later than n steps, the method reaches the exact solution. This approach yields a class of iterative methods, which are called *Krylov-subspace methods*.

In this subsection, we consider a symmetric, positive definite matrix A. Thus the definition

$$\langle x, y\rangle_A := \langle Ax, y\rangle = x^T Ay \tag{4.29}$$

yields a scalar product and the corresponding norm

$$\|x\|_A := \sqrt{\langle x, x\rangle_A} \tag{4.30}$$

is called the *energy norm*.

Let $x^0, \ldots, x^{k-1}$ and the subspace $\mathcal{K}_k$ be already determined. In the CG method, the new approximation x^k is fixed by the demand

$$\|x^k - x^*\|_A = \min_{y \in x^0 + \mathcal{K}_k} \|y - x^*\|_A, \tag{4.31}$$

which is equivalent to the condition

$$\langle x^k - x^*, u \rangle_A = 0 \quad \text{for all } \ u \in \mathcal{K}_k. \tag{4.32}$$

This can be seen via the relation

$$\|y - x^*\|_A^2 = \|y - x^k\|_A^2 + \|x^k - x^*\|_A^2 + 2\langle x^k - x^*, y - x^k \rangle_A, \tag{4.33}$$

since $y - x^k \in \mathcal{K}_k$ holds. Thus x^k represents the orthogonal projection of x^* onto the space $x^0 + \mathcal{K}_k$ w.r.t. $\langle \cdot, \cdot \rangle_A$. Remark that the approximation x^k is determined uniquely by the demands above.

The residual $r^k = b - Ax^k$ satisfies

$$0 = \langle x^* - x^k, u \rangle_A = \langle r^k, u \rangle \quad \text{for all } \ u \in \mathcal{K}_k. \tag{4.34}$$

Thus the residual is orthogonal to the space $\mathcal{K}_k$ with respect to the Euclidean scalar product. Furthermore, it holds

$$\mathcal{K}_k = \operatorname{span}\{r^0, Ar^0, \ldots, A^{k-1}r^0\} = \operatorname{span}\{r^0, r^1, \ldots, r^{k-1}\} \tag{4.35}$$

and

$$\langle r^i, r^j \rangle = 0 \ \text{ for } i \neq j \quad (i, j = 0, 1, \ldots, k). \tag{4.36}$$

Thereby, we assume $r^k \neq 0$. Otherwise, the exact solution is reached and the method terminates.

The residuals, which span the Krylov space $\mathcal{K}_k$ are orthogonal to each other. Thus we might think of an algorithm, which determines x^k via the demand (4.31) using projections. Unfortunately, this approach would require the knowledge of the solution x^* and thus it is not feasible.

On the other hand, we are able to provide an A-orthogonal basis in the space $\mathcal{K}_k$, i.e.

$$\mathcal{K}_k = \text{span}\{p^0, p^1, \ldots, p^{k-1}\}, \quad \langle p^i, p^j \rangle_A = 0 \ \text{ for } \ i \neq j. \tag{4.37}$$

The new approximation simply results in

$$x^k = x^{k-1} + \alpha^{k-1} p^{k-1}, \quad \alpha^{k-1} = \frac{\langle r^{k-1}, r^{k-1} \rangle}{\langle p^{k-1}, p^{k-1} \rangle_A} \tag{4.38}$$

and the residual is computed via

$$\begin{aligned} r^k &= b - Ax^k = A(x^* - x^k) = A(x^* - x^{k-1} - \alpha^{k-1} p^{k-1}) \\ &= r^{k-1} - \alpha^{k-1} A p^{k-1}. \end{aligned} \tag{4.39}$$

The new A-orthogonal basis vector is computed employing

$$p^k = r^k + \beta^{k-1} p^{k-1}, \quad \beta^{k-1} = -\frac{\langle r^k, p^{k-1} \rangle_A}{\langle p^{k-1}, p^{k-1} \rangle_A}. \tag{4.40}$$

The formulae (4.38)-(4.40) form the basis of the CG method.

Theorem 4.3 *Let $p^0 := r^0 = b - Ax^0$. If $r^k \neq 0$ holds, then the sequences defined by (4.38)-(4.40) satisfy*

(i) $\quad \mathcal{K}_k = \text{span}\{p^0, p^1, \ldots, p^{k-1}\} = \text{span}\{r^0, r^1, \ldots, r^{k-1}\}$

(ii) $\quad \langle p^i, p^j \rangle_A = 0 \ \text{ for } \ i \neq j$

(iii) $\quad \|x^k - x^*\|_A = \min_{y \in x^0 + \mathcal{K}_k} \|y - x^*\|_A.$

Proof: see J. Stoer, R. Bulirsch: Introduction to Numerical Analysis. 2nd Ed. Springer, New York, 1993.

Due to property (ii), the vectors p^k are called *conjugate directions*. Thus the CG method is also called *method of conjugate directions*.

The computation of β^{k-1} can be done applying

$$\alpha^{k-1} \langle r^k, p^{k-1} \rangle_A = \langle \alpha^{k-1} A p^{k-1}, r^k \rangle = \langle r^{k-1} - r^k, r^k \rangle = -\langle r^k, r^k \rangle \tag{4.41}$$

and thus

$$\beta^{k-1} = \frac{\langle r^k, r^k \rangle}{\langle r^{k-1}, r^{k-1} \rangle}. \tag{4.42}$$

Hence the complete algorithm reads as follows.

Algorithm 4.1 Conjugate Gradient method

$r^0 = b - Ax^0, \quad p^0 = r^0$
for $k = 0, 1, \ldots, k_{\max}$
  $\alpha^k = \frac{{r^k}^T r^k}{{p^k}^T A p^k}$
  $x^{k+1} = x^k + \alpha^k p^k$
  $r^{k+1} = r^k - \alpha^k A p^k$
  if $\|r^{k+1}\|_2 < TOL$*: exit*
  $\beta^k = \frac{{r^{k+1}}^T r^{k+1}}{{r^k}^T r^k}$
  $p^{k+1} = r^{k+1} + \beta^k p^k$
end

In practical computations, the result x^n will not be the exact solution due to rounding errors. The iteration can be continued and it takes about $4n$ steps to obtain the result accurately w.r.t. machine precision. However, in many applications, we do not require a high accuracy, i.e. an approximation of the exact solution is sufficient. Consequently, we hope to require $k \ll n$ steps to achieve a given accuracy.

Interpretation via Optimisation:

Considering the function

$$f : \mathbb{R}^n \to \mathbb{R}, \quad f(x) := \tfrac{1}{2} x^T A x - b^T x, \tag{4.43}$$

the gradient is $\mathrm{grad} f(x) = Ax - b$. A necessary condition for a local optimum $\hat{x}$ of f is $\mathrm{grad} f(\hat{x}) = 0$. The exact solution $x^* := A^{-1}b$ of the linear system satisfies this requirement. Since A is positive definite, the point x^* represents the global minimum of f. Given a starting value x^0, the CG

method chooses p^0 as the negative gradient of A. The point x^1 is the minimum point of f on line line $x^0 + \gamma p^0$. Thus the first step corresponds to the method of steepest descent. Later the approximation x^k is chosen as the minimum point of f on line $x^{k-1} + \gamma p^{k-1}$, too. The vectors p^k represent directions of decreasing f, i.e.

$$p^{k^T} \text{grad} f(x^k) < 0.$$

However, they do not represent the steepest descent. Nevertheless, the CG method yields the exact solution within n steps, whereas the method of steepest descent may take much longer to generate a reasonable approximation.

Preconditioning:

Using the energy norm, the estimate

$$\|x^k - x^*\|_A \leq 2 \left(\frac{\sqrt{\text{cond}(A)} - 1}{\sqrt{\text{cond}(A)} + 1} \right)^k \|x^0 - x^*\|_A \tag{4.44}$$

can be proved, where the condition number refers to the Euclidean norm. Thus the method converges faster the more the condition number of A decreases. One can try to scale down the condition number by the transformations

$$Ax = b \quad \rightsquigarrow \quad MANy = Mb, \;\; y = N^{-1}x \tag{4.45}$$

such that $\text{cond}(MAN) \ll \text{cond}(A)$ holds. Instead of performing the transformations explicitely, linear systems including M and N are solved in each step of the CG algorithm. Accordingly, the matrices M and N shall have a structure, which enables a fast direct solution of the arising linear systems. Simple choices for M, N are diagonal matrices, for example.

Since the matrix A is symmetric and positive definite, we perform the preconditioning technique

$$A \quad \rightsquigarrow \quad (\tilde{L})^{-1} A (\tilde{L}^{-1})^{-T}, \tag{4.46}$$

using a lower triangular matrix $\tilde{L}$, which approximates the Cholesky factor in $A = LL^T$. Such an approximation can be obtained by an incomplete Cholesky decomposition, for example.

Another preconditioning strategy, which is successfully used in practice, is to define the matrix M in (4.45) implicitely via an application of the symmetric Gauss-Seidel method, whereas $N = I$ is chosen. Therefore the classical iterative methods can be used as preconditioning steps in more sophisticated iterative techniques.

4.4 GMRES method

If the matrix $A \in \mathbb{R}^{n\times n}$ is not symmetric or symmetric but not positive definite, then the CG method can not be used to solve the linear system $Ax = b$. For general matrices A with $\det A \neq 0$, iterative techniques can be designed, which are based on the minimisation of the residual

$$h(x) := \|Ax - b\|_2^2 \tag{4.47}$$

with respect to Krylov spaces. The method of *generalised minimal residuals (GMRES)* is the most popular technique of this kind. In the kth step, the approximation x^k is determined by the demand

$$h(x^k) = \min_{y\in x^0+\mathcal{K}_k} h(y) = \min_{y\in\mathcal{K}_k} \|Ay + r^0\|_2^2 \qquad \left(r^0 := Ax^0 - b\right). \tag{4.48}$$

In the CG method, the requirement (4.31) minimises the distance of the approximation x^k to the exact solution with respect to the energy norm. Alternatively, the GMRES method tries to minimise the residual of x^k in the Euclidean norm via (4.48). The GMRES scheme yields the exact solution in at most n steps, too.

In contrast to the CG method, suitable basis vectors for the optimisation problem can not be obtained directly. Thus a new basis vector in the Krylov space has to be orthogonalised with respect to the old basis vectors.

Consequently, the computational effort increases linearly with the number of Newton steps, which represents a significant drawback of the GMRES method.

In practical application, the computational effort can be reduced by restarting the GMRES algorithm after 10 – 100 steps and use the last result as new starting value. Consequently, convergence slows down and alltogether more steps are required. However, the complete computational effort will be lower in general. Furthermore, preconditioning techniques may increase the speed of convergence in the GMRES method, too.

Chapter 5

Methods for Nonlinear Systems

In this section, we discuss the numerical solution of *nonlinear systems*, which represents *finding zeros* of a given function $F : \mathbb{R}^n \to \mathbb{R}^n$. In contrast to solving linear systems, direct techniques are not feasible and thus *iterative methods* have to be applied. The case of a univariate function already indicates the involved concepts and methods. Generalisations to the multidimensional case are straightforward. However, the analysis demands some sophisticated considerations. We focus on methods of Newton type, which are the most commonly used techniques in solving nonlinear systems.

5.1 Newton's Method

5.1.1 Univariate Case

Given a sufficiently smooth function $f : \mathbb{R} \to \mathbb{R}$, we are searching for one or more zeros $\hat{x}$, i.e.

$$f(\hat{x}) = 0. \tag{5.1}$$

Since an analytical solution of the problem (5.1) is impossible in most cases, we require numerical methods. The numerical techniques are working iter-

atively by producing a sequence of approximations

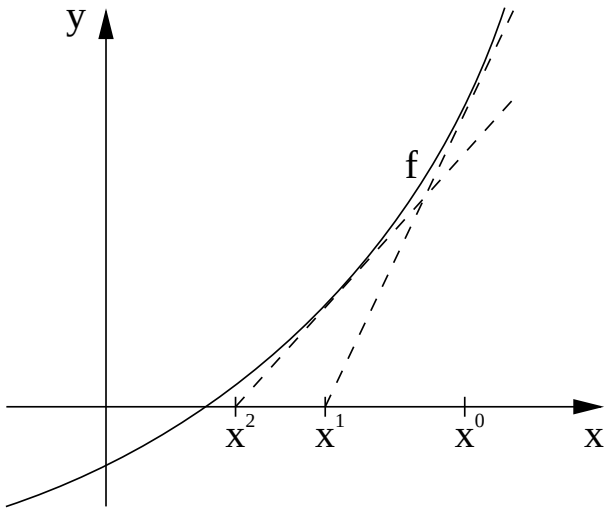
$$x^0, x^1, x^2, \ldots, x^k \in \mathbb{R}.$$

A simple and robust method represents the bisection. The scheme starts using two points $a < b$, where f exhibits opposite signs, i.e. $f(a) \cdot f(b) < 0$. Consequently, a continuous function f owns at least one zero in $[a, b]$. By evaluating f in the middle of the interval, we can exclude one half of the interval due to the arising sign. Thus $[a, b]$ is cut into halves and the procedure restarts using the approriate half.

Algorithm 5.1 Bisection

$A := f(a);\ B := f(b);$
while $b - a > tol$
$\quad t := (a + b)/2;\ T := f(t);$
$\quad$ *if* $A \cdot T > 0 :\ a := t;\ A := T;$
$\quad$ *else* $:\ b := t;\ B := T;$
$\hat{x} := (a + b)/2$

Newton's method applies information from the derivative of f. This technique corresponds to approximate f locally by a tangent and to compute the zero of this straight line. The following figure sketches this approach.



Algorithm 5.2 Newton's method

choose x^0*;*
for $k = 0, 1, 2, \ldots$
$\quad x^{k+1} := x^k - f(x^k)/f'(x^k)$*;*
$\quad$ *if* $|x^{k+1} - x^k| < tol$*: exit*
$\hat{x} := x^{k+1}$

To discuss Newton's method, we analyse shortly one step methods in general. A one step iterative method can be written in the form

$$x^{k+1} = \Phi(x^k) \tag{5.2}$$

using an iteration function $\Phi : D \to D$ ($D \subset \mathbb{R}$). If this iteration converges to some $\hat{x}$ and Φ is continuous, then $\hat{x}$ is a *fixed point* of Φ, because

$$\hat{x} = \lim_{k\to\infty} x^{k+1} = \lim_{k\to\infty} \Phi(x^k) = \Phi(\lim_{k\to\infty} x^k) = \Phi(\hat{x}). \tag{5.3}$$

Definition 5.1 *Let $\hat{x} \in D$ be a fixed point of Φ. The method (5.2) is* locally convergent*, if a neighbourhood $U \subset D$ of $\hat{x}$ exists such that $\lim_{k\to\infty} x^k = \hat{x}$ holds for all starting values $x^0 \in U$. The method (5.2) is* globally convergent*, if $\lim_{k\to\infty} x^k = \hat{x}$ holds for all starting values $x^0 \in D$.*

Remark: The local and global convergence imply the uniqueness of the fixed point in U and D, respectively.

Definition 5.2 *Let $\hat{x} \in D$ be a fixed point of Φ. The method (5.2) is locally convergent of at least* order $p \geq 1$*, if a neighbourhood $V \subset D$ of $\hat{x}$ exists such that*

$$|x^{k+1} - \hat{x}| \leq C|x^k - \hat{x}|^p$$

holds for all starting values $x^0 \in V$ and a constant $C \geq 0$. The arising constant has to satisfy $C < 1$ in case of $p = 1$.

Remark: Def. 5.2 can be generalised to functions $\Phi : \mathbb{R}^n \to \mathbb{R}^n$ by using a norm $\|\cdot\|$ instead of the absolute value $|\cdot|$. In this section, the Euclidean norm is applied, if not otherwise labelled.

Rule-of-thumb: If the method is convergent of order $p \geq 2$, then the number of correct digits increases in each step by the factor p. (Hence $p = 2$ implies a doubling of correct digits in each step.)

The order of convergence is mostly determined via Taylor expansion

$$\begin{aligned} \Phi(x) &= \Phi(\hat{x}) + \Phi'(\hat{x})(x - \hat{x}) + \tfrac{1}{2}\Phi''(\hat{x})(x - \hat{x})^2 + \cdots \\ \Rightarrow x^{k+1} = \Phi(x^k) &= \Phi(\hat{x}) + \Phi'(\hat{x})(x^k - \hat{x}) + \tfrac{1}{2}\Phi''(\hat{x})(x^k - \hat{x})^2 + \cdots . \end{aligned} \tag{5.4}$$

For Newton's method, we obtain the formulae (provided that $f'(x) \neq 0$)

$$x^{k+1} = x^k - \frac{f(x^k)}{f'(x^k)} \quad \Rightarrow \quad \Phi(x) = x - \frac{f(x)}{f'(x)} \tag{5.5}$$

$$\Phi'(x) = 1 - \frac{f'(x)^2 - f(x)f''(x)}{f'(x)^2} = f(x)\frac{f''(x)}{f'(x)^2}. \tag{5.6}$$

Since $f(\hat{x}) = 0$ implies $\Phi(\hat{x}) = \hat{x}$ and $\Phi'(\hat{x}) = 0$, it follows

$$x^{k+1} = \Phi(x^k) = \underbrace{\Phi(\hat{x})}_{=\hat{x}} + \underbrace{\Phi'(\hat{x})}_{=0}(x^k - \hat{x}) + \tfrac{1}{2}\Phi''(\hat{x} + \vartheta^k(x^k - \hat{x}))(x^k - \hat{x})^2 \tag{5.7}$$

$$\Rightarrow \quad x^{k+1} - \hat{x} = \tfrac{1}{2}\Phi''(\hat{x} + \vartheta^k(x^k - \hat{x}))(x^k - \hat{x})^2 \tag{5.8}$$

with $\vartheta^k \in (0, 1)$. If Φ'' is continuous, a compact neighbourhood $U \subset D$ of $\hat{x}$ exists, where $|\Phi''(x)| < 2C$ holds for all $x \in U$ and thus

$$|x^{k+1} - \hat{x}| < C|x^k - \hat{x}|^2. \tag{5.9}$$

Consequently, Newton's method is locally convergent of at least order 2 (quadratic convergence). In general, the order is exactly 2, since $\Phi''(\hat{x}) \neq 0$ mostly occurs. On the contrary, Newton's method is not globally convergent (counter-example: arctan-function). Therefore the convergence of Newton's method depends essentially on the used starting value.

5.1.2 Multivariate Case

Now we consider the numerical solution of nonlinear systems

$$F(x) = 0 \tag{5.10}$$

with a sufficiently smooth function $F : D \to \mathbb{R}^n$ ($D \subset \mathbb{R}^n$). Problems of the form (5.10) arise in many applications, for example, directly by equilibrium conditions in a discrete problem of indirectly by discretisations of differential equations.

Newton's method in the univariate case can be generalised to nonlinear systems. However, a simple geometric interpretation is not feasible any more. Alternatively, we employ Taylor expansion with respect to some starting value x^0 and obtain

$$0 = F(\hat{x}) = F(x^0) + DF(x^0)(\hat{x} - x^0) + \mathcal{O}(\|\hat{x} - x^0\|^2). \tag{5.11}$$

Thereby, $DF \in \mathbb{R}^{n \times n}$ represents the Jacobian matrix of F. Neglecting the last term, we obtain a new approximation x^1 for $\hat{x}$ via

$$0 = F(x^0) + DF(x^0)(x^1 - x^0) \quad \Rightarrow \quad DF(x^0)(x^1 - x^0) = -F(x^0). \tag{5.12}$$

Consequently, the kth step of Newton's method reads

$$\begin{array}{lrcl} \text{Solve:} & DF(x^k)\Delta x^k & = & -F(x^k) \\ \text{Update:} & x^{k+1} & = & x^k + \Delta x^k. \end{array} \tag{5.13}$$

This approach is also called the *Newton-Raphson* method (to distinguish between the variants of Newton's method).

Algorithm 5.3 Newton's method

choose x^0*;*
for $k = 0, 1, 2, \ldots$
    Solve: $DF(x^k)\Delta x^k = -F(x^k)$
    Update: $x^{k+1} = x^k + \Delta x^k$

if $\|x^{k+1} - x^k\| < tol$*: exit*

$\hat{x} := x^{k+1}$

Newton's method demands the computation of the Jacobian matrix of F in each step, which is usually done by numerical differentiation. In addition, a corresponding linear system has to be solved in each step. Thus the problem of solving a nonlinear system is reduced to the successive solution of linear systems.

Concerning the convergence of the multivariate Newton method, the following theorem gives informations.

Theorem 5.1 *Let* $\hat{x}$ *be a zero of* $F : D \to \mathbb{R}^n$ $(D \subset \mathbb{R}^n)$,

$$F(x) = (f_1(x), \ldots, f_n(x))^T, \quad x = (x_1, \ldots, x_n)^T$$

and $F \in C^2(D)$*. Define*

$$\mathcal{K} := \{x \in \mathbb{R}^n : \|x - \hat{x}\|_\infty \leq r\} \subset D$$

and

$$M := \max \left\{ \left| \frac{\partial^2 f_l}{\partial x_i \partial x_j}(x) \right| : 1 \leq l, i, j \leq n, \;\; x \in \mathcal{K} \right\}.$$

If $DF(\hat{x})$ *is invertible and*

$$\beta r \leq \tfrac{1}{2} \quad \textit{with} \quad \beta := n^2 M \|DF(\hat{x})^{-1}\|_\infty$$

holds, then the sequence $(x^k)_{k\in\mathbb{N}}$ *defined by Newton's method converges for any* $x^0 \in \mathcal{K}$ *to* $\hat{x}$ *and it follows*

$$\|x^{k+1} - \hat{x}\|_\infty \leq \beta \|x^k - \hat{x}\|_\infty^2 \leq \tfrac{1}{2}\|x^k - \hat{x}\|_\infty.$$

Proof:

Firstly, we show

$$\|DF(x) - DF(y)\|_\infty \leq n^2 M \|x - y\|_\infty \quad \text{for all } \; x, y \in \mathcal{K}.$$

By the mean value theorem, it follows with $\vartheta_l \in (0,1)$

$$\frac{\partial f_l}{\partial x_j}(x) - \frac{\partial f_l}{\partial x_j}(y) = \sum_{q=1}^{n} \frac{\partial^2 f_l}{\partial x_q \partial x_j}(x + \vartheta_l(x-y))(x_q - y_q)$$

$$\left|\frac{\partial f_l}{\partial x_j}(x) - \frac{\partial f_l}{\partial x_j}(y)\right| \leq nM \max_{i=1,\dots,n}(x_i - y_i) = nM\|x-y\|_\infty$$

$$\sum_{j=1}^{n}\left|\frac{\partial f_l}{\partial x_j}(x) - \frac{\partial f_l}{\partial x_j}(y)\right| \leq n^2 M\|x-y\|_\infty$$

for each l, which yields the assumption.

By induction, we assume $x^k \in \mathcal{K}$ ($x^0 \in \mathcal{K}$ is a premise). Now Taylor expansion yields

$$F(x^k) = \underbrace{F(\hat{x})}_{0} + DF(\hat{x})(x^k - \hat{x}) + R^k,$$

where the lth component of $R^k \in \mathbb{R}^n$ reads

$$R_l^k = \frac{1}{2}\sum_{i=1}^{n}\sum_{j=1}^{n} \frac{\partial^2 f_l}{\partial x_i \partial x_j}(\hat{x} + \vartheta_{l,k}(x^k - \hat{x}))(x_i^k - \hat{x}_i)(x_j^k - \hat{x}_j).$$

Therefore we obtain directly

$$\|R^k\|_\infty \leq \tfrac{1}{2}n^2 M\|x^k - \hat{x}\|_\infty^2.$$

To apply the Newton iteration, we have to show $\det(DF(x^k)) \neq 0$. We define the matrix $H^k := DF(x^k) - DF(\hat{x})$. Since $\hat{x}, x^k \in \mathcal{K}$ holds, it follows

$$\|H^k\|_\infty = \|DF(x^k) - DF(\hat{x})\|_\infty \leq n^2 M\|x^k - \hat{x}\|_\infty \leq n^2 Mr.$$

Furthermore, we have

$$DF(x^k) = DF(\hat{x}) + DF(x^k) - DF(\hat{x}) = DF(\hat{x})(I + DF(\hat{x})^{-1}H^k).$$

Thus $\det(DF(x^k)) \neq 0$ if and only if $\det(I + DF(\hat{x})^{-1}H^k) \neq 0$. This is guaranteed by the property

$$\|DF(\hat{x})^{-1}H^k\|_\infty \leq \|DF(\hat{x})^{-1}\|_\infty \cdot \|H^k\|_\infty \leq \tfrac{\beta}{n^2 M} \cdot n^2 Mr = \beta r \leq \tfrac{1}{2} < 1.$$

The formula of the Newton iteration and the above Taylor expansion yields

$$x^{k+1} - \hat{x} = x^k - \hat{x} - DF(x^k)^{-1}F(x^k) = DF(x^k)^{-1}R^k.$$

The general property $\|(I+B)^{-1}\| \leq 1/(1-\|B\|)$ for $\|B\| < 1$ gives

$$\|DF(x^k)^{-1}\|_\infty = \|(I + DF(\hat{x})^{-1}H^k)^{-1}DF(\hat{x})^{-1}\|_\infty \leq 2\|DF(\hat{x})^{-1}\|_\infty.$$

Finally, we obtain

$$\begin{aligned}
\|x^{k+1} - \hat{x}\|_\infty &\leq \|DF(x^k)^{-1}\|_\infty \cdot \|R^k\|_\infty \\
&\leq 2\|DF(\hat{x})^{-1}\|_\infty \tfrac{1}{2} n^2 M \|x^k - \hat{x}\|_\infty^2 \\
&= \beta \|x^k - \hat{x}\|_\infty^2 \\
&\leq \beta r \|x^k - \hat{x}\|_\infty \\
&\leq \tfrac{1}{2}\|x^k - \hat{x}\|_\infty
\end{aligned}$$

and thus the predicated convergence results hold. □

Conclusions / remarks :

- Newton's method is locally convergent of at least order 2.
- Existence of zero $\hat{x}$ and $\det(DF(\hat{x})) \neq 0$ is postulated.
- Assumptions can not be verified in practical applications.

Another statement, which does not presume the existence of a zero but other assumptions concerning the starting value, is given by the following theorem.

Theorem 5.2 (Newton-Kantorovich) *Let $D \subset \mathbb{R}^n$ be open as well as convex and $F : D \to \mathbb{R}^n$ smooth ($F \in C^1$). Let $x^0 \in D$ denote a given starting value with invertible Jacobian $DF(x^0)$. Constants $\alpha, \beta, \gamma \geq 0$ shall exist, which satisfy*

$$\begin{aligned}
&(i) \quad \|DF(x^0)^{-1}F(x^0)\| \leq \alpha \\
&(ii) \quad \|DF(x^0)^{-1}\| \leq \beta \\
&(iii) \quad \|DF(x) - DF(y)\| \leq \gamma\|x - y\| \text{ for all } x, y \in D.
\end{aligned}$$

Consider the quantities

$$h := \alpha\beta\gamma, \quad \rho_{1,2} := \frac{\alpha}{h}(1 \mp \sqrt{1-2h}).$$

If $h \leq \frac{1}{2}$ *and* $\overline{S_{\rho_1}(x^0)} \subset D$ *holds, then the sequence* $(x^k)_{k\in\mathbb{N}}$ *defined by Newton's method (5.13) is a subset of* $S_{\rho_1}(x^0)$ *and converges to the unique zero* $\hat{x}$ *of* F *with* $\hat{x} \in D \cap S_{\rho_2}(x^0)$.

Proof: see J. M. Ortega, W. C. Rheinboldt: Iterative Solution of Non-linear Equations in Several Variables. Academic Press, New York, 1970.

5.2 Simplified Newton Method

Newton's method (5.13) for nonlinear systems demands the computation of the Jacobian matrix as well as solving the arising linear system in each step. Both parts may become costly in some applications. Therefore the idea is to use just the Jacobian corresponding to the starting value.

The algorithm of the *simplified Newton method* reads like the algorithm of the ordinary Newton method including the modification

$$\begin{array}{lrcl} \text{Solve:} & DF(x^0)\Delta x^k & = & -F(x^k) \\ \text{Update:} & x^{k+1} & = & x^k + \Delta x^k \end{array} \tag{5.14}$$

in the kth step.

Now the advantage is that just a single Jacobian matrix has to be evaluated. If Gaussian elimination yields the solutions of the linear systems, then the LU-decomposition has to be computed only once, too. Further solving of the linear systems by forward- and backward-substitution is cheap. If the linear systems are solved by iterative methods, then a sophisticated matrix for preconditioning can be reused in each step.

Theorem 5.3 *Let* $D \subset \mathbb{R}^n$ *be open as well as convex and* $F : D \to \mathbb{R}^n$ *smooth* ($F \in C^1$). *Let* $x^0 \in D$ *denote a given starting value with invertible*

Jacobian $DF(x^0)$. The following assumptions shall be fulfilled:

(i) $\|DF(x^0)^{-1}(DF(x) - DF(x^0))\| \leq \theta_0 \|x - x^0\|$ *for all* $x \in D$ *with some* $\theta_0 \geq 0$

(ii) $h_0 := \theta_0 \|\Delta x^0\| \leq \frac{1}{2}$

(iii) $t^* := 1 - \sqrt{1 - 2h_0}$, $\rho := \frac{t^*}{\theta_0}$ *yield* $\overline{S_\rho(x^0)} \subset D$.

Then the sequence $(x^k)_{k \in \mathbb{N}}$ defined by the simplified Newton method (5.14) remains in $\overline{S_\rho(x^0)}$ and converges to some $\hat{x}$ with $F(\hat{x}) = 0$. Furthermore, it holds

$$\|x^{k+1} - x^k\| \leq \tfrac{1}{2}(t^k + t^{k-1}) \|x^k - x^{k-1}\|, \quad i = 1, 2, \ldots$$

and

$$\|x^k - \hat{x}\| \leq \frac{t^* - t^k}{\theta_0}, \quad k = 0, 1, 2, \ldots,$$

where $t^0 := 0$ and $t^{k+1} := h_0 + \frac{1}{2}(t^k)^2$ for $k = 0, 1, 2, \ldots$.

Proof: see P. Deuflhard: Newton Methods for Nonlinear Problems. Springer, Berlin, 2004.

Conclusions / remarks:

- Simplified Newton method is locally convergent of order 1.
  $\rightarrow$ linear convergence
- rate of convergence: $\Theta_k := \frac{\|\Delta x^{k+1}\|}{\|\Delta x^k\|} \leq \frac{1}{2}(t^{k+1} + t^k) < t^* \leq 1$
  $\rightarrow$ convergence may slow down
- Quantity $\Theta_0 = \frac{\|\Delta x^1\|}{\|\Delta x^0\|} \leq \frac{1}{2} h_0 \leq \frac{1}{4}$ characterises the local convergence domain. In contrast to $\Theta_0 < 1$ for ordinary Newton method (5.13), the domain of convergence is reduced. $\rightarrow$ good starting values required

Further simplified techniques of Newton type:

- *Broyden's rank-one method*: Jacobian matrix used in the first step is modified slightly by a cheap formula in each subsequent step.

- *quasi Newton method*: Jacobian matrix is replaced by rough approximations.

5.3 Modified Newton Method

The convergence domain of Newton's method (5.13) is often tiny. Accordingly, the method frequently does not converge for given starting values. The convergence domain can be enlarged by a *damping strategy*, which results in the *modified Newton method*.

The finding of zeros belonging to a smooth function $F : D \to \mathbb{R}^n$ ($D \subset \mathbb{R}^n$) is closely connected to problems of minimisation. We define the *test function*

$$h : D \to \mathbb{R}_0^+, \quad h(x) := F(x)^T F(x) = \|F(x)\|^2. \tag{5.15}$$

Since $h(x) \geq 0$ for all x and

$$F(\hat{x}) = 0 \quad \Leftrightarrow \quad h(\hat{x}) = 0, \tag{5.16}$$

a zero of F is equivalent to a global minimum point of h. The method of *steepest descent* represents an approach to find the minimum of h. Thereby, the information from the gradient (written as column vector)

$$Dh(x) = \text{grad}h(x) = 2DF(x)^T F(x) \tag{5.17}$$

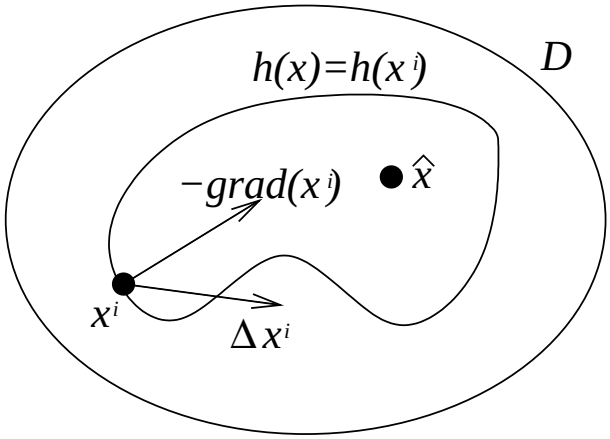
is applied. Unfortunately, the rate of convergence is very slow in this method. On the contrary, we want to exploit Newton's method, which is locally quadratic convergence. The Newton correction

$$\Delta x^k = -DF(x^k)^{-1} F(x^k) \tag{5.18}$$

is a direction of descent, i.e. $h(x^k + \omega \Delta x^k) < h(x^k)$ holds for small values $\omega \in (0, \omega_0)$. Because, let α be the angle between Δx^k and $-\text{grad}h(x^k)$, it follows

$$\begin{aligned} \cos\alpha &= \frac{-\text{grad}h(x^k) \cdot \Delta x^k}{\|\text{grad}h(x^k)\| \cdot \|\Delta x^k\|} = \frac{2F(x^k)^T DF(x^k) DF(x^k)^{-1} F(x^k)}{2\|DF(x^k)^T F(x^k)\| \cdot \|DF(x^k)^{-1} F(x^k)\|} \\ &\geq \frac{\|F(x^k)\|^2}{\|DF(x^k)\| \cdot \|DF(x^k)^{-1}\| \cdot \|F(x^k)\|^2} = \frac{1}{\text{cond}_2(DF(x^k))} > 0, \end{aligned}$$

i.e. $|\alpha| < \frac{\pi}{2}$. If $\text{cond}_2(DF(x^k))$ is huge, then h decays very slowly in direction Δx^k. Often the complete Newton step is too large, i.e. $h(x^k + \Delta x^k) > h(x^k)$ holds. The following draft indicates this problem.



Now the coefficient ω^k is chosen such that

$$\|F(\tilde{x}^{k+1})\|^2 < (1 - \tau\omega^k)\|F(x^k)\|^2 \tag{5.19}$$

is satisfied, where $0 < \tau < \frac{1}{2}$ is selected. The factor τ demands, that the residual decreases significantly. A common choice represents $\tau = \frac{1}{100}$ or $\tau = \frac{1}{4}$. By choosing ω^k sufficiently small, (5.19) is fulfilled. However, if ω^k becomes too small, we see the iteration as divergent and quit. The parameter ω^k satisfying (5.19) can be determined by trying subsequently the values $\omega = 1, \frac{1}{2}, \frac{1}{4}, \frac{1}{8}, \ldots, \omega_{\min}$. Consequently, the method reduces to the ordinary Newton method, if (5.19) is always satisfied for $\omega^k = 1$.

Algorithm 5.4 modified Newton method

choose x^0
choose $0 < \tau < \frac{1}{2}$
for $k = 0, 1, 2, \ldots$
    Solve: $DF(x^k)\Delta x^k = -F(x^k)$
    $\omega^k := 1$;
    $\tilde{x}^{k+1} := x^k + \omega^k \Delta x^k$;

while $\|F(\tilde{x}^{k+1})\|^2 \geq (1 - \tau\omega^k)\|F(x^k)\|^2$ *:*
$\quad \omega^k := \omega^k/2;$
$\quad \tilde{x}^{k+1} := x^k + \omega^k \Delta x^k;$
$x^{k+1} := \tilde{x}^{k+1};$
if $\|x^{k+1} - x^k\| <$ *tol: exit*

Concerning the convergence behaviour of this modified Newton method, the following theorem yields informations.

Theorem 5.4 *Let* $F : D \to \mathbb{R}^n$ *(*$D \subset \mathbb{R}^n$ *open),* $F \in C^2(D)$ *and* $y^0 \in D$*. If the set*

$$L := \{x \in D : \|F(x)\|_2 \leq \|F(y^0)\|_2\}$$

is compact and $\det DF(x) \neq 0$ *for all* $x \in L$*, then the sequence* $(x^k)_{k\in\mathbb{N}}$ *defined by the modified Newton method using* $0 < \tau < \frac{1}{2}$ *converges to a zero* $\hat{x}$ *of* F *for any starting value* $x^0 \in L$*. For sufficiently large* k*, the value* $\omega^k = 1$ *is chosen automatically, i.e. the method is locally convergent of at least order 2.*

Proof: see P. Deuflhard: A modified Newton method for the solution of illconditioned systems of nonlinear equations with application to multiple shooting. Num. Math. 22 (1974) S. 289-316.

Conclusions / remarks:

- Modified Newton method is globally convergent in L.
- Assumption $\det DF(x) \neq 0$ for all $x \in L$ is strong.
  $\to$ size of appropriate L may be small
- Computational effort of modified Newton method increases only by additional evaluations of function F.

Natural scaling:

The convergenge behaviour of the above modified Newton method can often be improved, if the requirement $\|F(\tilde{x}^{k+1})\|^2 < (1-\tau\omega^k)\|F(x^k)\|^2$ is replaced by

$$\|DF(x^k)^{-1}F(\tilde{x}^{k+1})\|^2 < (1-\tau\omega^k)\|DF(x^k)^{-1}F(x^k)\|^2. \tag{5.20}$$

Thus the residuals are scaled by the matrix $DF(x^k)^{-1}$ in the kth step, which is called *natural scaling*. This operation is not expensive, since an LU-decomposition of $DF(x^k)$ has been computed before to determine Δx^k.

5.4 Gauss-Newton Scheme

Nonlinear data-fitting problems are described by a function $F : D \to R^m$ with domain $D \subset \mathbb{R}^n$ and a set of data $y \in \mathbb{R}^m$, where $m > n$ holds. The problem is to determine a vector $x^* \in \mathbb{R}^n$, which minimises

$$\|y - F(x)\|^2 = \sum_{l=1}^{m}(y_l - F_l(x_1, \ldots, x_n))^2. \tag{5.21}$$

For a linear function $F(x) = Ax$ ($A \in \mathbb{R}^{m\times n}$), the problem reduces to the linear least-squares problem.

More general, we write the nonlinear least-squares problem in the form

$$\|G(x)\|^2 \to \min! \qquad \text{with} \qquad G : D \to \mathbb{R}^m \quad (D \subset \mathbb{R}^n). \tag{5.22}$$

An iteration scheme for the numerical solution of (5.22) is obtained by Taylor expansion using a starting value x^0

$$G(x) = G(x^0) + DG(x^0)(x - x^0) + \mathcal{O}(\|x - x^0\|^2), \tag{5.23}$$

where $DG \in \mathbb{R}^{m\times n}$ denotes the Jacobian matrix of G. We assume the property rank$DG = n$. If the starting value x^0 is close to the minimum point x^*, then the solution z in the linear least-squares problem

$$G(x^0) + DG(x^0)(z - x^0) \to \min! \tag{5.24}$$

will often still be closer to x^* than x^0, i.e. $\|G(z)\|^2 < \|G(x^0)\|^2$. This relation is not always true. However, using $s := z - x^0$ and $a(\omega) := \|G(x^0 + \omega s)\|^2$, we conclude

$$a'(0) = \frac{d}{d\omega} G(x^0 + \omega s)^T G(x^0 + \omega s)\Big|_{\omega=0} = 2(DG(x^0)s)^T G(x^0). \tag{5.25}$$

Since s is the solution of the normal equations

$$DG(x^0)^T DG(x^0)s = -DG(x^0)^T G(x^0) \tag{5.26}$$

of the linear least-squares problem, it follows that

$$\|DG(x^0)s\|^2 = s^T DG(x^0)^T DG(x^0)s = -(DG(x^0)s)^T G(x^0). \tag{5.27}$$

Thus rank$DG(x^0) = n$ and $s \neq 0$ implies

$$a'(0) = -2\|DG(x^0)s\|^2 < 0. \tag{5.28}$$

Hence the function $a(\cdot)$ decreases for small $\omega \in (0, \omega_0)$. Consequently, we apply a damping strategy like in the modified Newton method. Due to the similar derivation as in Newton's approach, the resulting technique is called *Gauss-Newton method*.

An appropriate choice of the parameter ω^k is given by a damping strategy employing the condition

$$\|G(x^k)\|^2 - \|G(x^{k+1})\|^2 \geq 2\tau\omega^k G(x^k)^T DG(x^k)s^k, \tag{5.29}$$

where $0 < \tau < \frac{1}{2}$ is chosen and s^k represents the solution of the linear problem in the kth step. Thereby, ω^k is selected as the maximum value in the sequence $1, \frac{1}{2}, \frac{1}{4}, \ldots$ again.

Algorithm 5.5 Gauss-Newton method

choose x^0
choose $0 < \tau < \frac{1}{2}$
for $k = 0, 1, 2, \ldots$

Compute minimum s^k of linear least-squares problem
$\|G(x^k) + DG(x^k)s^k\|^2 \to min!$
$\omega^k := 1;$
$\tilde{x}^{k+1} := x^k + \omega^k s^k;$
while $\|G(x^k)\|^2 - \|G(x^{k+1})\|^2 < 2\tau\omega^k G(x^k)^T DG(x^k)s^k$:
    $\omega^k := \omega^k/2;$
    $\tilde{x}^{k+1} := x^k + \omega^k s^k;$
$x^{k+1} := \tilde{x}^{k+1};$
if $\|x^{k+1} - x^k\| < tol$: *exit*

Theorem 5.5 *Let $G : D \to \mathbb{R}^m$ ($D \subset \mathbb{R}^n$ open), where $m \geq n$, $G \in C^2(D)$ and $y^0 \in D$. If the set*

$$L := \{x \in D : \|G(x)\|_2 \leq \|G(y^0)\|_2\}$$

is compact and rank$DG(x) = n$ for all $x \in L$, then the sequence $(x^k)_{k\in\mathbb{N}}$ defined by the damped Gauss-Newton method converges for any $x^0 \in L$ to a minimum point of $\|G(\cdot)\|^2$.

Conclusions / remarks:

- The assumption rank$DG = n$ in L is not strong.
- If several local minima exist, the method may converge to a local minimum, which is not a global minimum.
- More sophisticated damping strategies are feasible.