MA398 Matrix Analysis and Algorithms

Mathematics Institute and Centre for Scientific Computing
University of Warwick
UNITED KINGDOM

December 10, 2019

Contents

Lecture 1

Introduction

Subject of this course are some problems that are central to numerical linear algebra and occur in many applications.

Systems of Linear Equations (SLE)

Given $A \in \mathbb{C}^{n\times n}$, $b \in \mathbb{C}^n$

$$\text{find } x \in \mathbb{C}^n \text{ such that } Ax = b.$$

Lots of theory has already been provided in courses on linear algebra such as solvability (regular matrices), characterisation of linear maps (Jordan canonical form).

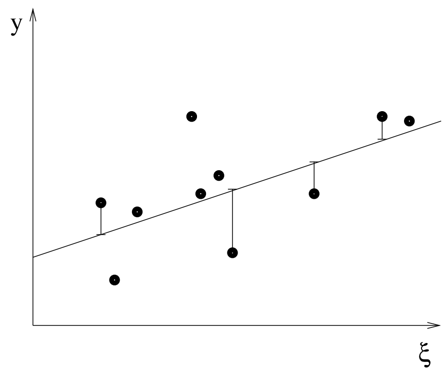
Least SQuares problems (LSQ)

Given $A \in \mathbb{R}^{m\times n}$, $b \in \mathbb{R}^m$ (typically $m \geq n$)

$$\text{find } x \in \mathbb{R}^n \text{ such that } \|Ax - b\|_2 \text{ is minimal.}$$

Example: Regression (linear). Given points $(\xi_i, y_i)_{i=1}^m$ find a linear function $\xi \mapsto x_1 + x_2\xi$ such that

$$g(x) := \Big(\sum_{i=1}^m \big(x_1 + x_2\xi_i - y_i\big)^2\Big)^{1/2} \quad \text{is minimal.}$$



In this case,

$$A = \begin{pmatrix} 1 & \xi_1 \\ \vdots & \vdots \\ 1 & \xi_m \end{pmatrix}, \quad b = \begin{pmatrix} y_1 \\ \vdots \\ y_m \end{pmatrix}.$$

EigenValue Problems (EVP)

Given $A \in \mathbb{C}^{n \times n}$

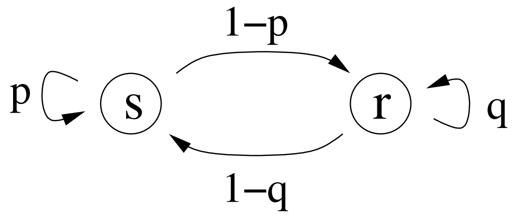
$$\text{find } (x, \lambda) \in (\mathbb{C}^n \backslash \{0\}) \times \mathbb{C} \text{ such that } Ax = \lambda x.$$

Sometimes, an eigenvector or an eigenvalue may be known, and the goal then is to compute the corresponding eigenvalue or, respectively, a corresponding eigenvector.

Example: We consider a highly simplified weather model. We distinguish only two states: $\underline{s}$un and $\underline{r}$ain. If one day's weather is s then the next day's weather is

- s with probability $p \in (0, 1)$,
- r with probability $1 - p$.

Similarly, there is a probability $q \in (0, 1)$ for no change of state r and $1 - q$ for a change from r to s.



Let us denote the actual state by a vector $\mu \in \mathbb{R}^2$ such that $(1, 0)$ corresponds to s (i.e., we associate state s with index 1) and $(0, 1)$ corresponds to r (associate r with index 2). Let us furthermore introduce the <u>probability matrix</u> $P = (p_{i,j})_{i,j=1}^2$ where $p_{i,j}$ denotes the probability that the system being in state i changes to state j, i.e.,

$$P = \begin{pmatrix} p & 1-p \\ 1-q & q \end{pmatrix}.$$

Multiplying the actual state with P from the right then yields a vector of probabilities for the next day's state, for instance if we start with a sunny day, $\mu^{(0)} := (1, 0)$, then

$$\mu^{(0)} P = (1, 0) \begin{pmatrix} p & 1-p \\ 1-q & q \end{pmatrix} = (p, 1-p) =: \mu^{(1)}$$

indeed contains the correct probabilities for s and r. The nice thing about this notation is that we simply may go on multiplying with P from the right to obtain the probabilities for the subsequent days, for instance

$$\mu^{(2)} := \mu^{(1)} P = (p^2 + (1-p)(1-q), p(1-q) + (1-p)q)$$

contains the probabilities for s and r on the second day, and

$$\mu^{(k)} := \mu^{(k-1)} P \Big(= \mu^{(0)} P^k \Big) \tag{1.1}$$

the probabilities for the k^{th} day.
Experts will recognise this iteration as a discrete Markov chain. In applications, one often is interested in stationary distributions π satisfying

$$\pi = \pi P, \quad \pi_i \geq 0 \,\forall i, \quad \sum_i \pi_i = 1$$

which means that π is a left eigenvector of P for the eigenvalue 1. In fact, 1 always is an eigenvalue of such probability matrices so the goal is just to compute a corresponding eigenvector. And of further interest is when Markov chains often converge to such stationary states.
We will see such type of iterations as the Markov chain (1.1) again in a slightly more general context of the so-called power iteration. Apart from the convergence, another questions immediately arising is how long one has to iterate in order to obtain an acceptable approximation to a stationary state (as this is an iterative method one will not expect to get an exact solution).

Aims and Objectives

The general goal is to understand the mathematical principles underlying the design of algorithms for the previously mentioned problems in linear algebra, and their analysis with respect to accuracy and cost.
At the end of the module you will familiar with concepts and ideas related to:

- diverse **matrix factorisations**
  - to obtain analytical results,
  - as the theoretical basis for designing algorithms,
- assessing algorithms with respect to computational cost (**efficiency**),
- **error analysis**, splitting up into
  - conditioning of problems,
  - stablity of algorithms,
- differences in the analysis of **direct** versus **iterative** methods.

Lecture 2

Gaussian Elimination

Problem: Given $A \in \mathbb{C}^{n\times n}$ and $b \in \mathbb{C}^n$ find $x \in \mathbb{C}^n$ such that $Ax = b$.
<u>Observation:</u> If $A = (a_{i,j})_{i,j=1}^n$ is <u>(upper) triangular</u>, i.e.,

$$A = \begin{pmatrix} a_{1,1} & \cdots & \cdots & a_{1,n} \\ 0 & \ddots & & \vdots \\ \vdots & \ddots & \ddots & \vdots \\ 0 & \cdots & 0 & a_{n,n} \end{pmatrix}$$

then the x_i can be recursively computed as follows:

$$\begin{array}{rc} \text{last row :} & a_{n,n}x_n = b_n \Rightarrow x_n = b_n/a_{nn}, \\ \text{row } n-1 : & a_{n-1,n-1}x_{n-1} + a_{n-1,n}x_n = b_{n-1} \Rightarrow x_{n-1} = (b_{n-1} - a_{n-1,n}x_n)/a_{n-1,n-1}, \\ \vdots & \vdots \\ \text{first row:} & \ldots \Rightarrow x_1 = (b_1 - \sum_{j=2}^{n} a_{1,j}x_j)/a_{1,1}. \end{array}$$

Algorithm 1 BS (backward substitution)

input: $U = (u_{i,j})_{i,j=1}^n \in \mathbb{C}^{n\times n}$ upper triangular, $u_{i,i} \neq 0$ for all $i = 1, \ldots, n$, $b = (b_i)_{i=1}^n \in \mathbb{C}^n$.
output: $x \in \mathbb{C}^n$ solution to $Ux = b$.

```
for $i = n$ to 1 do
    $h := 0$
    for $j = n$ to $i + 1$ do
        $h := h + u_{i,j}x_j$
    end for
    $x_i := (b_i - h)/u_{i,i}$
end for
```

Assume now that A is an arbitrary matrix.
The following operations do not change the solution space:

- multiplying an equation with a number $r \in \mathbb{C}\backslash\{0\}$,
- adding an equation to another equation.

Idea: Use such operations to transform A into a triangular matrix.

Example:

$$A = \begin{pmatrix} 2 & 1 & 1 \\ 4 & 3 & 3 \\ 8 & 7 & 9 \end{pmatrix}, \quad b = \begin{pmatrix} 4 \\ 10 \\ 24 \end{pmatrix}.$$

$$\leadsto \quad \begin{array}{rcrcrcr} 2x_1 & + & x_2 & + & x_3 & = & 4 \\ 4x_1 & + & 3x_2 & + & 3x_3 & = & 10 \\ 8x_1 & + & 7x_2 & + & 9x_3 & = & 24 \end{array}$$

multiply the first equation with $-2 = -a_{2,1}/a_{1,1}$ and add to the second equation,
multiply the first equation with $-4 = -a_{3,1}/a_{1,1}$ and add to the third equation,

$$\leadsto \quad \begin{array}{rcrcrcr} 2x_1 & + & x_2 & + & x_3 & = & 4 \\ 0 & + & x_2 & + & x_3 & = & 2 \\ 0 & + & 3x_2 & + & 5x_3 & = & 8 \end{array}$$

multiply the second equation with -3 and add to the third equation,

$$\leadsto \quad \begin{array}{rcrcrcr} 2x_1 & + & x_2 & + & x_3 & = & 4 \\ & & x_2 & + & x_3 & = & 2 \\ & & 0 & + & 2x_3 & = & 2 \end{array}$$

Now, use **BS** to compute the solution $x = (1, 1, 1)^T$.

Observation: The first step corresponds to multiplying $Ax = b$ with

$$\tilde{L}_1 := \begin{pmatrix} 1 & 0 & 0 \\ -2 & 1 & 0 \\ -4 & 0 & 1 \end{pmatrix}$$

from the left:

$$\tilde{L}_1 A x = \begin{pmatrix} 1 & 0 & 0 \\ -2 & 1 & 0 \\ -4 & 0 & 1 \end{pmatrix} \begin{pmatrix} 2 & 1 & 1 \\ 4 & 3 & 3 \\ 8 & 7 & 9 \end{pmatrix} x = \begin{pmatrix} 2 & 1 & 1 \\ 0 & 1 & 1 \\ 0 & 3 & 5 \end{pmatrix} x,$$

$$\tilde{L}_1 b = \begin{pmatrix} 1 & 0 & 0 \\ -2 & 1 & 0 \\ -4 & 0 & 1 \end{pmatrix} \begin{pmatrix} 4 \\ 10 \\ 24 \end{pmatrix} = \begin{pmatrix} 4 \\ 2 \\ 8 \end{pmatrix}$$

Similarly, the second step corresponds to a multiplication with

$$\tilde{L}_2 := \begin{pmatrix} 1 & 0 & 0 \\ 0 & 1 & 0 \\ 0 & -3 & 1 \end{pmatrix}.$$

A matrix $A = (a_{i,j}) \in \mathbb{C}^{n \times n}$ is unit lower triangular if

1. $a_{i,j} = 0$ if $1 \le i < j \le n$,
2. $a_{i,i} = 1$ for all $i = 1, \dots, n$.

Lemma 2.1. [5.1] *(a) Let $L \in \mathbb{C}^{n\times n}$ unit lower triangular with non-zero entries only in column k. Then L^{-1} is unit lower triangular with non-zero entries only below the diagonal in column k with entries*

$$(L^{-1})_{i,k} = -L_{i,k}, \quad i = k+1, \dots, n.$$

(b) Let $L, M \in \mathbb{C}^{n\times n}$ be unit lower triangular , $k \in \{1, \dots, n\}$, and
L has non-zero entries below the diagonal only in columns $1, \dots, k$,
M has non-zero entries below the diagonal only in columns $k+1, \dots, n$.
Then LM is unit lower triangular with

$$(LM)_{i,j} = \begin{cases} L_{i,j} & \text{if } j \le k, \\ M_{i,j} & \text{else.} \end{cases}$$

Proof. Exercise. □

Consequence:

$$(\tilde{L}_1)^{-1} = \begin{pmatrix} 1 & 0 & 0 \\ 2 & 1 & 0 \\ 4 & 0 & 1 \end{pmatrix} =: L_1, \quad (\tilde{L}_2)^{-1} := \begin{pmatrix} 1 & 0 & 0 \\ 0 & 1 & 0 \\ 0 & 3 & 1 \end{pmatrix} =: L_2,$$

and we see that

$$A = (\tilde{L}_1)^{-1}\tilde{L}_1 A = L_1(\tilde{L}_2)^{-1}\tilde{L}_2\tilde{L}_1 A = L_1 L_2 U = LU$$

with

$$L = \begin{pmatrix} 1 & 0 & 0 \\ 2 & 1 & 0 \\ 4 & 3 & 1 \end{pmatrix}, \quad U = \begin{pmatrix} 2 & 1 & 1 \\ 0 & 1 & 1 \\ 0 & 0 & 2 \end{pmatrix}$$

The matrix A is factorised into a unit lower triangular matrix L and an upper triangular regular matrix U. Given this, we may solve $Ly = b$ by forward substitution (**exercise**, formulate this in analogy to **BS**) and then $Ux = y$ by **BS** in order to compute the solution to $Ax = b$.

Lecture 3

LU factorisation

The j^{th} principal sub-matrix of $A \in \mathbb{C}^{n\times n}$ is by $A_j \in \mathbb{C}^{j\times j}$ given by $(A_j)_{k,l} = a_{k,l}$, $k, l = 1, \dots, j$.

Theorem 3.1. *Let $A \in \mathbb{C}^{n\times n}$.*
(a) Assume that A_j is invertible for all $j = 1, \dots, n$. Then there is a unique factorisation $A = LU$ with L unit lower triangular and U regular and upper triangular.
(b) If A_j is singular for some j then there is no such factorisation.

Proof. (a) Proof is based on induction, similar proofs will follow $\rightarrow$ **Exercise.**
(b) Assume that $A = LU$ exists but A_j is singular. In block form:

$$A = \begin{pmatrix} A_{11} & A_{12} \\ A_{21} & A_{22} \end{pmatrix} = \begin{pmatrix} L_{11} & 0 \\ L_{21} & L_{22} \end{pmatrix} \begin{pmatrix} U_{11} & U_{12} \\ 0 & U_{22} \end{pmatrix} = \begin{pmatrix} L_{11}U_{11} & \star \\ \star & \star \end{pmatrix}$$

with $A_{11} = A_j$, and since L_{11} is unit lower triangular and U_{11} is regular and upper triangular, $A_j = L_{11}U_{11}$ is the LU factorisation of A_j. But then

$$\det(A_j) = \det(L_{11}U_{11}) = \underbrace{\det(L_{11})}_{=1}\underbrace{\det(U_{11})}_{\neq 0} \neq 0$$

in contradiction to the singularity of A_j. □

Algorithm 2 LU

input: $A = (a_{ij})_{i,j=1}^{n} \in \mathbb{C}^{n\times n}$ with $\det(A_k) \neq 0$, $k = 1, \dots, n$.
output: $L \in \mathbb{C}^{n\times n}$ unit lower triangular, $U \in \mathbb{C}^{n\times n}$ upper triangular and regular with $A = LU$.

```
1: $U = A$, $L = I$.
2: for $k = 1$ to $n-1$ do
3:   for $j = k+1$ to $n$ do
4:     $l_{j,k} := u_{j,k}/u_{k,k}$
5:     $u_{j,k} := 0$
6:     for $i = k+1$ to $n$ do
7:       $u_{j,i} := u_{j,i} - l_{j,k}u_{k,i}$
8:     end for
9:   end for
10: end for
```

We have $u_{k,k} \neq 0$ in line 4:
After $k-1$ steps we have the matrix

$$U^{(k-1)} = \begin{pmatrix} \star & \cdots & \cdots & \star & \star & \cdot & \star \\ 0 & \ddots & & \vdots & \vdots & & \vdots \\ \vdots & \ddots & \star & \star & \vdots & & \vdots \\ 0 & \cdots & 0 & u_{k,k} & \star & \cdots & \star \\ 0 & \cdots & 0 & \star & \star & \cdots & \star \\ \vdots & & \vdots & \vdots & \vdots & & \vdots \\ 0 & \cdots & 0 & \star & \star & \cdots & \star \end{pmatrix} = (L^{(k-1)})^{-1}A$$

with $L^{(k-1)}$ unit lower triangular. Therefore

$$u_{k,k}\det(U^{(k-1)}_{k-1}) = \det(U^{(k-1)}_k) = \underbrace{\det((L^{(k-1)})^{-1}_k)}_{=1}\det(A_k) \neq 0$$

since $\det(A_k) \neq 0$ by assumption. Hence $u_{k,k} \neq 0$ and we can divide in line 4.

Example:

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

is regular since $\det(A) = 2$ but $A_1 = 0$ and A_2 both are singular.
Idea: Permutation of the rows of A, $\sigma : (1,2,3) \mapsto (3,1,2)$, which is equivalent to a multiplication with the permutation matrix

$$P = \begin{pmatrix} 0 & 1 & 0 \\ 0 & 0 & 1 \\ 1 & 0 & 0 \end{pmatrix}, \quad PA = \begin{pmatrix} 1 & 1 & 0 \\ 0 & 2 & 1 \\ 0 & 0 & 1 \end{pmatrix}$$

and PA has regular submatrices. In fact, as PA is upper triangular, the LU factorisation is obtained by setting $U = PA$ and choosing L to be the identity.

Definition 3.2. *$P \in \mathbb{C}^{n\times n}$ is a permutation matrix if every row and every column contains $n-1$ zeros and 1 one.*

We may write $P = (e_{\sigma(1)}, e_{\sigma(2)}, \ldots, e_{\sigma(n)})$ where e_j is the j^{th} vector of the standard basis of $\mathbb{C}^n$. With $\pi = \sigma^{-1}$ we also may write

$$P = \begin{pmatrix} — & e_{\pi(1)} & — \\ & \vdots & \\ — & e_{\pi(n)} & — \end{pmatrix}.$$

This means that if we write

$$A = \begin{pmatrix} — & a_1 & — \\ & \vdots & \\ — & a_n & — \end{pmatrix}$$

with the row vectors $a_i \in \mathbb{C}^n$ then

$$PA = \begin{pmatrix} — & a_{\pi(1)} & — \\ & \vdots & \\ — & a_{\pi(n)} & — \end{pmatrix}$$

hence, a multiplication with a permutation matrix from the *left* exchanges the *rows* according to the associated permutation. Similarly, a multiplication with a permutation matrix from the *right* exchanges the *columns*.

Theorem 3.3. *Let $A \in \mathbb{C}^{n\times n}$ be regular. Then there are a permutation matrix $P \in \mathbb{C}^{n\times n}$, $L \in \mathbb{C}^{n\times n}$ unit lower triangular, and $U \in \mathbb{C}^{n\times n}$ upper triangular with $PA = LU$.*

Proof. By induction on n. The case $n = 1$ is trivial as one just has to choose $P = L = 1$ and $U = A$.
Let $n > 1$ and assume that the assertion is true for $n-1$. Choose a permutation matrix P_1 such that $a := (P_1A)_{1,1} \neq 0$. Such a matrix exists because A is regular, whence the first column of A will contain a nonzero entry. We write

$$P_1A = \begin{pmatrix} a & u^* \\ l & B \end{pmatrix} = \begin{pmatrix} 1 & 0 \\ \frac{1}{a}l & I \end{pmatrix} \begin{pmatrix} a & u \\ 0 & \tilde{A} \end{pmatrix}$$

where $l, u \in \mathbb{C}^{(n-1)\times 1}$, u^* is the adjoint of u, and $\tilde{A} = -\frac{1}{a}lu^* \in \mathbb{C}^{(n-1)\times(n-1)}$. The matrix $\tilde{A}$ is regular since

$$0 \neq \det(P_1A) = \underbrace{\det \begin{pmatrix} 1 & 0 \\ \frac{1}{a}l & I \end{pmatrix}}_{=1} \det \begin{pmatrix} a & u \\ 0 & \tilde{A} \end{pmatrix} = a \det(\tilde{A}).$$

By the induction hypothesis there are $\tilde{P}$ (permutation), $\tilde{L}$ (unit lower triangular) and $\tilde{U}$ (regular upper triangular) with $\tilde{P}\tilde{A} = \tilde{L}\tilde{U}$. Therefore

$$\begin{aligned} P_1A &= \begin{pmatrix} 1 & 0 \\ \frac{1}{a}l & I \end{pmatrix} \begin{pmatrix} 1 & 0 \\ 0 & (\tilde{P})^{-1}\tilde{L} \end{pmatrix} \underbrace{\begin{pmatrix} a & u \\ 0 & \tilde{U} \end{pmatrix}}_{=:U} \\ &= \begin{pmatrix} 1 & 0 \\ \frac{1}{a}l & (\tilde{P})^{-1}\tilde{L} \end{pmatrix} U \\ &= \begin{pmatrix} 1 & 0 \\ 0 & (\tilde{P})^{-1} \end{pmatrix} \underbrace{\begin{pmatrix} 1 & 0 \\ \frac{1}{a}\tilde{P}l & \tilde{L} \end{pmatrix}}_{=:L} U \end{aligned}$$

$$\Rightarrow \quad \underbrace{\begin{pmatrix} 1 & 0 \\ 0 & (\tilde{P})^{-1} \end{pmatrix}^{-1} P_1}_{=:P} A = LU$$

where P is a permutation matrix and L and U are of the desired structure, too. □

Lecture 4

Gaussian Elimination with Pivoting

Before step k:

$$U^{(k-1)} = \begin{pmatrix} \star & \cdots & \cdots & \star & \star & \cdot & \star \\ 0 & \ddots & & \vdots & \vdots & & \vdots \\ \vdots & \ddots & \star & \star & \vdots & & \vdots \\ 0 & \cdots & 0 & u_{k,k} & \star & \cdots & \star \\ 0 & \cdots & 0 & \star & \star & \cdots & \star \\ \vdots & & \vdots & \vdots & \vdots & & \vdots \\ 0 & \cdots & 0 & \star & \star & \cdots & \star \end{pmatrix}$$

and we have a **problem** if the $\underline{\text{pivot}}$ $u_{k,k}$ is zero.
In fact, a small $u_{k,k}$ is undesirable as it leads to stability problems $\rightarrow$ will see this later on.
There are basically two strategies to overcome this problem.

1. **GEPP**, Gaussian Elimination with Partial Pivoting: Swap rows to maximise $|u_{k,k}|$ among the entries $u_{l,k}$, $l = k, \dots, n$.

2. **GECP**, Gaussian Elimination with Complete Pivoting: Swap rows and columns to maximise $|u_{k,k}|$ among the entries $u_{l,m}$, $l, m = k, \dots, n$.

In the following we will only consider **GEPP** since, in practice, partial pivoting usually is sufficient and the gain in stability due to complete pivoting negligible.
With an appropriate permutation matrix P_k that realises the swap of the rows we then compute $U^{(k)} = \tilde{L}_k P_k U^{(k-1)}$ so that in the end

$$U = \tilde{L}_{n-1} P_{n-1} \cdots \tilde{L}_1 P_1 A. \tag{4.1}$$

The permutation associated with P_l exchanges l with some number $i_l > l$ and leaves the other entries unchanged,

$$(1, \dots, l-1, l, l+1, \dots, i_l - 1, i_l, i_l + 1, \dots, n) \mapsto (1, \dots, l-1, i_l, l+1, \dots, i_l - 1, l, i_l + 1, \dots, n).$$

For some $k < l$, consider the unit lower triangular matrix

$$M = \begin{pmatrix}
1 & 0 & \cdots & \cdots & \cdots & \cdots & \cdots & \cdots & \cdots & \cdots & 0 \\
0 & \ddots & \ddots & & & & & & & & \vdots \\
\vdots & \ddots & \ddots & \ddots & & & & & & & \vdots \\
\vdots & & 0 & 1 & \ddots & & & & & & \vdots \\
\vdots & & \vdots & m_{k+1,k} & \ddots & \ddots & & & & & \vdots \\
\vdots & & \vdots & \vdots & 0 & \ddots & \ddots & & & & \vdots \\
\vdots & & \vdots & m_{l,k} & \vdots & \ddots & \ddots & \ddots & & & \vdots \\
\vdots & & \vdots & \vdots & \vdots & & \ddots & \ddots & \ddots & & \vdots \\
\vdots & & \vdots & m_{i_l,k} & \vdots & & & \ddots & \ddots & \ddots & \vdots \\
\vdots & & \vdots & \vdots & \vdots & & & & \ddots & \ddots & 0 \\
0 & \cdots & 0 & m_{n,k} & 0 & \cdots & \cdots & \cdots & \cdots & 0 & 1
\end{pmatrix}$$

<u>Observation:</u>

$$P_l M P_l^{-1} = \begin{pmatrix}
1 & 0 & \cdots & \cdots & \cdots & \cdots & \cdots & \cdots & \cdots & \cdots & 0 \\
0 & \ddots & \ddots & & & & & & & & \vdots \\
\vdots & \ddots & \ddots & \ddots & & & & & & & \vdots \\
\vdots & & 0 & 1 & \ddots & & & & & & \vdots \\
\vdots & & \vdots & m_{k+1,k} & \ddots & \ddots & & & & & \vdots \\
\vdots & & \vdots & \vdots & 0 & \ddots & \ddots & & & & \vdots \\
\vdots & & \vdots & m_{i_l,k} & \vdots & \ddots & \ddots & \ddots & & & \vdots \\
\vdots & & \vdots & \vdots & \vdots & & \ddots & \ddots & \ddots & & \vdots \\
\vdots & & \vdots & m_{l,k} & \vdots & & & \ddots & \ddots & \ddots & \vdots \\
\vdots & & \vdots & \vdots & \vdots & & & & \ddots & \ddots & 0 \\
0 & \cdots & 0 & m_{n,k} & 0 & \cdots & \cdots & \cdots & \cdots & 0 & 1
\end{pmatrix}$$

As a consequence, the matrix

$$L'_k := P_{n-1} \cdots P_{k+1} \tilde{L}_k P_{k+1}^{-1} \cdots P_{n-1}^{-1}$$

has the same structure as $\tilde{L}_k$ but just the entries in column k below the diagonal are permuted. Since

$$L'_k P_{n-1} \cdots P_{k+1} = P_{n-1} \cdots P_{k+1} \tilde{L}_k$$

we obtain from (4.1) that

$$U = \underbrace{L'_{n-1} \cdots L'_1}_{=:L^{-1}} \underbrace{P_{n-1} \cdot P_1}_{=:P} A \quad \Rightarrow LU = PA$$

which is a factorisation of the desired form.

Algorithm 3 LUPP

input: $A = (a_{ij})_{i,j=1}^n \in \mathbb{C}^{n\times n}$ regular.
output: $L \in \mathbb{C}^{n\times n}$ unit lower triangular, $U \in \mathbb{C}^{n\times n}$ regular upper triangular, $P \in \mathbb{C}^{n\times n}$ permutation matrix with $PA = LU$.

```
1: $U = A$, $L = I$, $P = I$.
2: for $k = 1$ to $n-1$ do
3:   choose $i \in \{k,\dots,n\}$ such that $|u_{i,k}|$ is maximal
4:   exchange $(u_{k,k},\dots,u_{k,n})$ with $(u_{i,k},\dots,u_{i,n})$
5:   exchange $(l_{k,1},\dots,l_{k,k-1})$ with $(l_{i,1},\dots,l_{i,k-1})$
6:   exchange $(p_{k,1},\dots,p_{k,n})$ with $(p_{i,1},\dots,p_{i,n})$
7:   for $j = k+1$ to $n$ do
8:     $l_{j,k} := u_{j,k}/u_{k,k}$
9:     $u_{j,k} := 0$
10:    for $i = k+1$ to $n$ do
11:      $u_{j,i} := u_{j,i} - l_{j,k}u_{k,i}$
12:    end for
13:  end for
14: end for
```

A more elegant variant of the algorithms stores the permutation in a vector of length n initialised with the numbers $(1,\dots,n)$ that finally contains the permutation π associated with P, i.e., the i^{th} entry of that vector contains $\pi(i)$. See example below.

Algorithm 4 GEPP (Gaussian elimination with partial pivoting)

input: $A \in \mathbb{C}^{n\times n}$ regular, $b \in \mathbb{C}^n$.
output: $x \in \mathbb{C}^n$ with $Ax = b$.

```
1: find $PA = LU$ with algorithm LUPP
2: solve $Ly = Pb$ with FS
3: solve $Ux = y$ with BS
```

Example: Consider

$$A = \begin{pmatrix} -2 & 2 & 0 & 0 \\ 2 & -4 & 1 & 1 \\ 0 & 4 & -2 & 0 \\ 1 & 1 & 0 & 1 \end{pmatrix}, \quad b = \begin{pmatrix} 0 \\ 0 \\ 2 \\ 3 \end{pmatrix}$$

and execute **GEPP**.
Step 1, computation of the LU factorisation with **LUPP**.
The last vector will contain the permutation π:

$$L^{(0)} = \begin{pmatrix} 1 & 0 & 0 & 0 \\ 0 & 1 & 0 & 0 \\ 0 & 0 & 1 & 0 \\ 0 & 0 & 0 & 1 \end{pmatrix} \quad U^{(0)} = \begin{pmatrix} -2 & 2 & 0 & 0 \\ 2 & -4 & 1 & 1 \\ 0 & 4 & -2 & 0 \\ 1 & 1 & 0 & 1 \end{pmatrix}, \quad \pi^{(0)} = \begin{pmatrix} 1 \\ 2 \\ 3 \\ 4 \end{pmatrix}$$

Apparently, $2 = |U_{1,1}^{(0)}| \geq \max_{j\geq 1} |U_{j,1}^{(0)}|$, hence we need no permutation. One elimination step leads to

$$L^{(1)} = \begin{pmatrix} 1 & 0 & 0 & 0 \\ -1 & 1 & 0 & 0 \\ 0 & 0 & 1 & 0 \\ -\frac{1}{2} & 0 & 0 & 1 \end{pmatrix} \quad U^{(1)} = \begin{pmatrix} -2 & 2 & 0 & 0 \\ 0 & -2 & 1 & 1 \\ 0 & 4 & -2 & 0 \\ 0 & 2 & 0 & 1 \end{pmatrix}, \quad \pi^{(1)} = \begin{pmatrix} 1 \\ 2 \\ 3 \\ 4 \end{pmatrix}$$

Now, $4 = |U^{(1)}_{3,2}| \geq \max_{j\geq 2} |U^{(1)}_{j,2}|$, hence we permute rows 2 and 3:

$$(L^{(1)})' = \begin{pmatrix} 1 & 0 & 0 & 0 \\ 0 & 1 & 0 & 0 \\ -1 & 0 & 1 & 0 \\ -\frac{1}{2} & 0 & 0 & 1 \end{pmatrix} \quad (U^{(1)})' = \begin{pmatrix} -2 & 2 & 0 & 0 \\ 0 & 4 & -2 & 0 \\ 0 & -2 & 1 & 1 \\ 0 & 2 & 0 & 1 \end{pmatrix}, \quad \pi^{(2)} = \begin{pmatrix} 1 \\ 3 \\ 2 \\ 4 \end{pmatrix}$$

The next elimination step yields

$$L^{(2)} = \begin{pmatrix} 1 & 0 & 0 & 0 \\ 0 & 1 & 0 & 0 \\ -1 & -\frac{1}{2} & 1 & 0 \\ -\frac{1}{2} & \frac{1}{2} & 0 & 1 \end{pmatrix} \quad U^{(2)} = \begin{pmatrix} -2 & 2 & 0 & 0 \\ 0 & 4 & -2 & 0 \\ 0 & 0 & 0 & 1 \\ 0 & 0 & 1 & 1 \end{pmatrix}, \quad \pi^{(2)} = \begin{pmatrix} 1 \\ 3 \\ 2 \\ 4 \end{pmatrix}$$

We have that $1 = |U^{(1)}_{4,3}| \geq \max_{j\geq 3} |U^{(1)}_{j,3}|$, hence we permute rows 3 and 4:

$$(L^{(2)})' = \begin{pmatrix} 1 & 0 & 0 & 0 \\ 0 & 1 & 0 & 0 \\ -\frac{1}{2} & \frac{1}{2} & 1 & 0 \\ -1 & -\frac{1}{2} & 0 & 1 \end{pmatrix} \quad (U^{(2)})' = \begin{pmatrix} -2 & 2 & 0 & 0 \\ 0 & 4 & -2 & 0 \\ 0 & 0 & 1 & 1 \\ 0 & 0 & 0 & 1 \end{pmatrix}, \quad \pi^{(3)} = \begin{pmatrix} 1 \\ 3 \\ 4 \\ 2 \end{pmatrix}$$

No further elimination step is required as $(U^{(2)})'$ already is upper triangular. Hence

$$L^{(3)} = \begin{pmatrix} 1 & 0 & 0 & 0 \\ 0 & 1 & 0 & 0 \\ -\frac{1}{2} & \frac{1}{2} & 1 & 0 \\ -1 & -\frac{1}{2} & 0 & 1 \end{pmatrix} \quad U^{(3)} = \begin{pmatrix} -2 & 2 & 0 & 0 \\ 0 & 4 & -2 & 0 \\ 0 & 0 & 1 & 1 \\ 0 & 0 & 0 & 1 \end{pmatrix}, \quad \pi^{(3)} = \begin{pmatrix} 1 \\ 3 \\ 4 \\ 2 \end{pmatrix}$$

One may now check that $LU = PA$ with $U = U^{(3)}$, $L = L^{(3)}$, and

$$P = \begin{pmatrix} 1 & 0 & 0 & 0 \\ 0 & 0 & 1 & 0 \\ 0 & 0 & 0 & 1 \\ 0 & 1 & 0 & 0 \end{pmatrix}$$

which is the permutation matrix associated with $\pi = \pi^{(3)}$.
<u>Step2</u>, solving $Ly = Pb$. We have that

$$Pb = \begin{pmatrix} e_{\pi(1)} \cdot b \\ \vdots \\ e_{\pi(n)} \cdot b \end{pmatrix} = \begin{pmatrix} b_{\pi(1)} \\ \vdots \\ b_{\pi(n)} \end{pmatrix}$$

which indicates how the action of the permutation matrix P on a vector can be computed given the associated permutation π. In our example, the solution is $y = (0, 2, 2, 1)^T$.
<u>Step3</u>, solving $Ux = y$.
One can check that $x = (1, 1, 1, 1)^T$ which indeed is the solution to $Ax = b$.

Lecture 5

Matrix Norms, Part I

Some remarks on norms and inner products first.

A <u>vector norm</u> on $\mathbb{C}^n$ is a map $\|\cdot\| : \mathbb{C}^n \to \mathbb{R}$ with

1. $\|x\| \geq 0$ for all $x \in \mathbb{C}^n$ and $\|x\| = 0$ if and only it $x = 0$,
2. $\|\alpha x\| = |\alpha|\|x\|$ for all $\alpha \in \mathbb{C}$, $x \in \mathbb{C}^n$,
3. $\|x + y\| \leq \|x\| + \|y\|$ for all $x, y \in \mathbb{C}^n$.

Theorem 5.1. *All norms on $\mathbb{C}^n$ are equivalent, i.e., given norms $\|\cdot\|_a$ and $\|\cdot\|_b$, there are constants $0 < c_1 \leq c_2 < \infty$ such that*

$$c_1\|x\|_a \leq \|x\|_b \leq c_2\|x\|_a \quad \forall x \in \mathbb{C}^n.$$

An <u>inner product</u> on $\mathbb{C}^n$ is a map $\langle\cdot,\cdot\rangle : \mathbb{C}^n \times \mathbb{C}^n \to \mathbb{C}$ with

1. $\langle x, x\rangle \in \mathbb{R}^+$ for all $x \in \mathbb{C}^n$, and $\langle x, x\rangle = 0$ if and only if $x = 0$,
2. $\langle x, y\rangle = \overline{\langle y, x\rangle}$ for all $x, y \in \mathbb{C}^n$,
3. $\langle x, \alpha y\rangle = \alpha\langle x, y\rangle$ for all $\alpha \in \mathbb{C}$ and $x, y \in \mathbb{C}^n$,
4. $\langle x, y + z\rangle = \langle x, y\rangle + \langle x, z\rangle$ for all $x, y, z \in \mathbb{C}^n$.

Examples:

- p-norm: $\|x\|_p := (\sum_i |x_i|^p)^{1/p}$ for $p \in [1, \infty)$,
- maximum-norm: $\|x\|_\infty := \max_i |x_i|$,
- standard inner product: $\langle x, y\rangle := \sum_i \overline{x}_i y_i$.

Lemma 5.2. [1.5] *(Cauchy-Schwarz) We have that*

$$|\langle x, y\rangle|^2 \leq \langle x, x\rangle\langle y, y\rangle \quad \forall x, y \in \mathbb{C}^n$$

with equality if and only if $x = \alpha y$ or $y = \alpha x$ for some $\alpha \in \mathbb{C}$.

Lemma 5.3. [1.6] *Given an inner product $\langle\cdot,\cdot,\rangle$,*

$$x \mapsto \sqrt{\langle x, x\rangle}$$

is a norm on $\mathbb{C}^n$.

Given $A = (a_{k,l})_{k,l=1}^{m,n} \in \mathbb{C}^{m\times n}$, the adjoint $A^* \in \mathbb{C}^{n\times m}$ is the matrix with entries $(A^*)_{i,j} = \overline{a}_{j,i}$. A vector $x \in \mathbb{C}^n$ can be considered as an $n \times 1$ matrix allowing for the notation $x^* \in \mathbb{C}^{1\times n}$. Then

$$x^*y = \begin{pmatrix} \overline{x}_1 & \dots & \overline{x}_n \end{pmatrix} \begin{pmatrix} y_1 \\ \vdots \\ y_n \end{pmatrix} = \sum_{i=1}^{n} \overline{x}_i y_i = \langle x, y\rangle.$$

We also write

$$xy^* = (x_i \overline{y}_j)_{i,j=1}^{n,n} = x \otimes \overline{y} \in \mathbb{C}^{n\times n}.$$

moreover,

$$\langle Ax, y\rangle = (Ax)^* y = x^* A^* y = \langle x, A^* y\rangle$$

for all $A \in \mathbb{C}^{m\times n}$, $x \in \mathbb{C}^n$ and $y \in \mathbb{C}^m$.
Some further definitions:

1. $Q \in \mathbb{C}^{m\times n}$ is unitary if $Q^*Q = I_n \in \mathbb{C}^{n\times n}$,
   $Q \in \mathbb{R}^{m\times n}$ is orthogonal if $Q^TQ = I_n \in \mathbb{R}^{n\times n}$.
2. $A \in \mathbb{C}^{n\times n}$ is Hermitian if $A^* = A$,
   $A \in \mathbb{R}^{n\times n}$ is symmetric if $A^T = A$.
3. A Hermitian matrix is positive (semi-)definite if

$$\langle x, Ax\rangle = x^*Ax > 0 \quad (\geq 0) \qquad \forall x \in \mathbb{C}^n \backslash \{0\}.$$

Q being unitary means that the columns are orthonormal. In the case $m = n$ we have that $Q^{-1} = Q^*$ and also $QQ^* = I$.

Theorem 5.4. *For any unitary $Q \in \mathbb{C}^{m\times n}$*

$$\langle Qx, Qy\rangle = \langle x, y\rangle, \quad \|Qx\|_2 = \|x\|_2, \qquad \forall x, y \in \mathbb{C}^n.$$

Proof. **Exercise.** □

One can also show that if $A \in \mathbb{C}^{n\times n}$ is positive definite then

$$\langle x, y\rangle_A := \langle x, Ay\rangle \text{ is an inner product,} \tag{1.4}$$
$$\|x\|_A := \sqrt{\langle x, x\rangle_A} \text{ is a vector norm.} \tag{1.5}$$

We can consider $\mathbb{C}^{m\times n}$ as a $\mathbb{C}$ vector space of dimension mn; possible norms:

$$\|A\|_{\max} := \max_{i,j} |a_{i,j}| \qquad \text{maximum norm,}$$

$$\|A\|_F := \left(\sum_{i,j} |a_{i,j}|^2\right)^{1/2} \qquad \text{Frobenius norm.}$$

Another way of assigning a norm to a matrix is the operator norm: Given norms $\|\cdot\|_{\hat{m}}$ on $\mathbb{C}^m$ and $\|\cdot\|_{\hat{n}}$ on $\mathbb{C}^n$ define

$$\|A\|_{(\hat{m},\hat{n})} := \max_{x\in\mathbb{C}^n\backslash\{0\}} \frac{\|Ax\|_{\hat{m}}}{\|x\|_{\hat{n}}} = \max_{\|x\|_{\hat{n}}=1} \|Ax\|_{\hat{m}}.$$

Definition 5.5. [1.25] *A matrix norm on $\mathbb{C}^{n\times n}$ is a mapping $\|\cdot\| : \mathbb{C}^{n\times n} \to \mathbb{R}$ with the properties*

1. $\|A\| \geq 0$ *for all* $A \in \mathbb{C}^{n\times n}$*, and* $\|A\| = 0$ *if and only if* $A = 0$*,*
2. $\|\alpha A\| = |\alpha|\|A\|$ *for all* $\alpha \in \mathbb{C}, A \in \mathbb{C}^{n\times n}$*,*
3. $\|A + B\| \leq \|A\| + \|B\|$ *for all* $A, B \in \mathbb{C}^{n\times n}$*,*
4. $\|AB\| \leq \|A\|\|B\|$ *for all* $A, B \in \mathbb{C}^{n\times n}$*.*

The last condition makes the difference to a vector norm. It means that the norm is compatible with the matrix-matrix product.

Definition 5.6. [1.26] *Given a vector norm $\|\cdot\|_v$ on $\mathbb{C}^n$, we define the induced (operator) norm $\|\cdot\|_m$ on $\mathbb{C}^{n\times n}$ by*

$$\|A\|_m := \max_{x\in\mathbb{C}^n\backslash\{0\}} \frac{\|Ax\|_v}{\|x\|_v} = \max_{\|x\|_v=1} \|Ax\|_v.$$

Theorem 5.7 (1.27)**.** *The induced norm $\|\cdot\|_m$ of a vector norm $\|\cdot\|_v$ is a matrix norm with $\|I_n\|_m = 1$ and*

$$\|Ax\|_v \leq \|A\|_m\|x\|_v \quad \forall A \in \mathbb{C}^{n\times n}, x \in \mathbb{C}.$$

Proof. Clearly $\|A\|_m \in \mathbb{R}$ and ≥ 0. We have

$$\|A\|_m = 0 \ \Leftrightarrow\ \frac{\|Ax\|_v}{\|x\|_v} = 0\,\forall x \in \mathbb{C}^n\backslash\{0\} \ \Leftrightarrow\ \|Ax\|_v = 0\,\forall x \in \mathbb{C}^n \ \Leftrightarrow\ A = 0$$

which shows the first point. For the second point we observe that

$$\|\alpha A\|_m = \max_{\|x\|_v=1} \|\alpha Ax\|_v = \max_{\|x\|_v=1} |\alpha|\|Ax\|_v = |\alpha| \max_{\|x\|_v=1} \|Ax\|_v = |\alpha|\|A\|_m,$$

and similarly the third property can be deduced from the corresponding property of the vector norm:

$$\begin{aligned}\|A + B\|_m = \max_{\|x\|_v=1} \|(A + B)x\|_v &\leq \max_{\|x\|_v=1} (\|Ax\|_v + \|Bx\|_v)\\ &\leq \max_{\|x\|_v=1} \|Ax\|_v + \max_{\|x\|_v=1} \|Bx\|_v = \|A\|_m + \|B\|_m.\end{aligned}$$

Clearly $\|I_n\|_m = \max_{\|x\|_v=1} \|I_nx\|_v = 1$, and for any $y \in \mathbb{C}^n\backslash\{0\}$

$$\|A\|_m = \max_{x\neq 0} \frac{\|Ax\|_v}{\|x\|_v} \geq \frac{\|Ay\|_v}{\|y\|_v} \quad \Rightarrow \quad \|Ay\|_v \leq \|A\|_m\|y\|_v.$$

Using this we can show the submultiplicativity, the fourth property of matrix norms:

$$\|AB\|_m = \max_{\|x\|_v=1} \|ABx\|_v \leq \max_{\|x\|_v=1} \|A\|_m\|Bx\|_v = \|A\|_m \max_{\|x\|_v=1} \|Bx\|_v = \|A\|_m\|B\|_m$$

□

Some remarks:

- We will use the same symbol / subscript for a vector norm and its induced norm, e.g. $\|x\|_2$ and $\|A\|_2$.

- Not every matrix norm is an induced norm.
  Exercise: Find an example. Hint: $\|I_n\| = 1$ does not have to be fulfilled.
- $\|x\|' := \|Ax\|$ is a vector norm if A is invertible. Theorem [1.27] yields

$$\frac{1}{\|A^{-1}\|}\|x\| \le \|x\|' \le \|A\|\|x\|.$$

  Exercise: Show these inequalities.

Theorem 5.8. [1.2] *The matrix norm induced by the infinity norm is the* <u>*maximum row sum*</u>*,*

$$\|A\|_\infty = \max_{1\le i\neq n} \sum_{j=1}^{n} |a_{i,j}|, \quad A \in \mathbb{C}^{n\times n}$$

Proof. For any $x \in \mathbb{C}^n$

$$\|Ax\|_\infty = \max_i \Big|\sum_j a_{i,j}x_j\Big| \le \max_i \sum_j |a_{i,j}||x_j| \le \max_i \sum_j |a_{i,j}|\|x\|_\infty.$$

To show the other inequality let $k \in \{1,\dots,n\}$ be the row index with maximal sum,

$$\max_i \sum_j |a_{i,j}| = \sum_j |a_{k,j}|.$$

Define $x \in \mathbb{C}^n$ by

$$x_j := \begin{cases} \frac{\overline{a_{k,j}}}{|a_{k,j}|} & \text{if } a_{k,j} \neq 0, \\ 0 & \text{else.} \end{cases}$$

Then $\|x\|_\infty = 1$ and

$$\|A\|_\infty \;\ge\; \|Ax\|_\infty \;=\; \max_i \Big|\sum_j a_{i,j}\frac{\overline{a_{k,j}}}{|a_{k,j}|}\Big| \;\ge\; \Big|\sum_j a_{k,j}\frac{\overline{a_{k,j}}}{|a_{k,j}|}\Big| \;=\; \sum_j |a_{k,j}| \;=\; \max_i \sum_j |a_{i,j}|.$$

□

Theorem 5.9. [1.29] *The matrix norm induced by the 1-norm is the* <u>*maximum column sum*</u>*,*

$$\|A\|_1 = \|A^*\|_\infty = \max_{1\le j\le n} \sum_{i=1}^{n} |a_{i,j}|. \quad A \in \mathbb{C}^{n\times n}.$$

Lecture 6

Eigenvalues and Eigenvectors

Given $A \in \mathbb{C}^{n\times n}$, λ is an eigenvalue of A if there is a $x \in \mathbb{C}^n \backslash \{0\}$ such that $Ax = x\lambda$.
Characteristic polynomial: $\rho_A(z) = \det(A - zI)$.
λ is eigenvalue if and only if $\rho_A(\lambda) = 0$.
Algebraic multiplicity of an eigenvalue λ: largest integer q such that $(z-\lambda)^q$ is a factor of $\rho_A(z)$.
Geometric multiplicity of an eigenvalue λ: dimension r of the kernel of $A - \lambda I$.
Simple eigenvalue: $r = q = 1$.
Example:

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

has $\rho_A(z) = (2-z)(1-z)^3$. Eigenvalues $\lambda = 1$: Algebraic multiplicity $q = 3$, geometric multiplicity $r = 2$ since the kernel of

$$A - \lambda I_n = \begin{pmatrix} 1 & 0 & 0 & 0 \\ 0 & 0 & 0 & 0 \\ 0 & 0 & 0 & 1 \\ 0 & 0 & 0 & 0 \end{pmatrix}$$

is two-dimensional.

Theorem 6.1. *$r \le q$.*

Definition 6.2. *The spectral radius of $A \in \mathbb{C}^{n\times n}$ is*

$$\rho(A) = \max\big\{|\lambda| \,\big|\, \lambda \textit{ eigenvalue of } A\big\}.$$

Similarity transformation: $B = XAX^{-1}$ with regular $X \in \mathbb{C}^{n\times n}$.

Theorem 6.3. *If $B \in \mathbb{C}^{n\times n}$ is similar to $A \in \mathbb{C}^{n\times n}$ then A and B have the same eigenvalues with the same multiplicities.*

Definition 6.4. *A matrix $A \in \mathbb{C}^{n\times n}$ is normal if it has n orthogonal eigenvectors.*

Normal matrices can be diagonalised: Let $Q := (q_1, \dots, q_n)$ with q_i orthonormal eigenvectors, hence Q is unitary. Then $AQ = Q\Lambda$ with $\Lambda = \operatorname{diag}(\lambda_1, \dots, \lambda_n)$ where the λ_i are the eigenvalues corresponding to the q_i. Hence

$$A = Q\Lambda Q^{-1} = Q\Lambda Q^*$$

i.e., A is similar to a diagonal matrix.
Further consequence:

$$A^*A = Q\Lambda^*Q^*Q\Lambda Q^* = Q\Lambda^*\Lambda Q^* = Q\Lambda\Lambda^*Q^* = Q\Lambda Q^*Q\Lambda^*Q^* = AA^*.$$

We will see below that normal matrices indeed can be characterised by this relation! To see this, we need

Theorem 6.5. [2.2] *Given $A \in \mathbb{C}^{n\times n}$, there is a unitary $Q \in \mathbb{C}^{n\times n}$ and an upper triangular $T \in \mathbb{C}^{n\times n}$ such that $A = QTQ^*$.*

Proof. By induction on n. In the case $n = 1$ we clearly may choose $Q = 1$ and $T = A$.
Let $n \geq 2$. By the fundamental theorem of algebra, the characteristic polynomial of A has a zero λ and, hence, an eigenvalue. Let y_1 denote a corresponding eigenvector with $\|y_1\|_2 = 1$ and extend it by $\{y_2, \dots, y_n\}$ to an orthonormal basis of $\mathbb{C}^n$.
Setting $U := (y_1, \dots, y_n) \in \mathbb{C}^{n\times n}$, the identity $Ay_1 = \lambda y_1$ leads to

$$AU = U\begin{pmatrix}\lambda & r^* \\ 0 & \tilde{A}\end{pmatrix} \quad \text{with some } r \in \mathbb{C}^{n-1},\ \tilde{A} \in \mathbb{C}^{(n-1)\times(n-1)}.$$

Using the induction assumption, there is a factorisation $\tilde{A} = V\tilde{T}V^*$ with upper triangular $\tilde{T} \in \mathbb{C}^{(n-1)\times(n-1)}$ and unitary $V \in \mathbb{C}^{(n-1)\times(n-1)}$. Therefore,

$$A = U\begin{pmatrix}\lambda & r^* \\ 0 & V\tilde{T}V^*\end{pmatrix}U^* = \underbrace{U\begin{pmatrix}1 & 0 \\ 0 & V\end{pmatrix}}_{=:Q}\underbrace{\begin{pmatrix}\lambda & r^*V \\ 0 & \tilde{T}\end{pmatrix}}_{=:T}\begin{pmatrix}1 & 0 \\ 0 & V^*\end{pmatrix}U^*.$$

Observing that Q is unitary since U and V are, this factorisation is of the desired structure. □

Theorem 6.6. [2.3] *If $A \in \mathbb{C}^{n\times n}$ satisfies $A^*A = AA^*$ then there is a unitary $Q \in \mathbb{C}^{n\times n}$ and a diagonal $D \in \mathbb{C}^{n\times n}$ such that $A = QDQ^*$.*

Proof. We have from Th. [2.2] that $A = QTQ^*$ with T upper triangular. We will show that T is diagonal.
Since $QT^*TQ^* = QT^*Q^*QTQ^* = A^*A = AA^* = QTQ^*QT^*Q^* = QTT^*Q^*$ we obtain $T^*T = TT^*$. Therefore

$$(T^*T)_{i,i} = \sum_{k=1}^{n}(T^*)_{i,k}T_{k,i} = \sum_{k=1}^{n}\overline{T}_{k,i}T_{k,i} = \sum_{k=1}^{i}|T_{k,i}|^2 \qquad (\star_1)$$

where we used that T is triangular in the last identity. Similarly,

$$(TT^*)_{i,i} = \sum_{k=i}^{n}|T_{i,k}|^2. \qquad (\star_2)$$

We now show by induction on i that the off-diagonal entries of T vanish.
For $i = 1$ we get from ($\star_1$) and ($\star_2$) that $|T_{1,1}|^2 = \sum_{k=1}^{n}|T_{1,k}|^2$ from which we conclude that $T_{1,k} = 0$ for $k = 2, \dots, n$.
Let now $i > 1$ and assume that $T_{k,j} = 0$ for $1 \leq k \leq i-1$ and all $j > k$. We need to show that $T_{i,k} = 0$ for $k = i+1, \dots, n$. Since, in particular, $T_{k,i} = 0$ for $k = 1, \dots, i-1$ we obtain from ($\star_1$) and ($\star_2$) that $|T_{i,i}|^2 = \sum_{k=i}^{n}|T_{i,k}|^2$ from which we can conclude that indeed $T_{i,k} = 0$ for $k = i+1, \dots, n$. □

The proofs of the following two statements are left as **exercises**:

Theorem 6.7. [2.4] *If $A \in \mathbb{C}^{n\times n}$ is Hermitian then there is a unitary Q and a* **real** $\Lambda \in \mathbb{R}^{n\times n}$ *such that $A = Q\Lambda Q^*$.*

Lemma 6.8. [2.5] *$A \in \mathbb{C}^{n\times n}$ positive definite. There is a positive definite $A^{1/2} \in \mathbb{C}^{n\times n}$ (the square root of A) such that $A = A^{1/2}A^{1/2}$.*

Lecture 7

Matrix Norms, Part II [1.4]

Theorem 7.1. [1.30] *For any matrix norm $\|\cdot\|$, $A \in \mathbb{C}^{n\times n}$, and $k \in \mathbb{N}$*

$$\rho(A)^k \le \rho(A^k) \le \|A^k\| \le \|A\|^k$$

Proof. Let $B := A^k$. If λ is an eigenvalue of A then λ^k is an eigenvalue of B which implies the first inequality. The third inequality is a consequence of matrix norms being sub-multiplicative. To prove the second inequality, let μ be the eigenvalue of B such that $\rho(B) = |\mu|$, let $x \in \mathbb{C}^n \setminus \{0\}$ be a corresponding eigenvector and set $X := (x, \dots, x) \in \mathbb{C}^{n\times n}$. Then

$$\|B\|\|X\| \ge \|BX\| = \|\mu X\| = |\mu|\|X\| = \rho(B)\|X\|$$

from which we get the second inequality after dividing by $\|X\|$. □

The inequalities in the above theorem become equalities if the matrix is normal and we use the matrix norm induced by the Euclidian norm:

Theorem 7.2. [1.31] *If $A \in \mathbb{C}^{n\times n}$ is normal then $\rho(A)^l = \|A\|_2^l$ for all $l \in \mathbb{N}$.*

Proof. Let $x_1, \dots, x_n$ be an orthonormal basis of eigenvectors of A corresponding to the eigenvalues $\lambda_1, \dots, \lambda_n$ where w.l.o.g. $\rho(A) = |\lambda_1|$. For any $x \in \mathbb{C}^n$ we can write

$$x = \sum_{j=1}^n \alpha_j x_j \text{ with } \alpha_j = \langle x_j, x\rangle_2 \quad \Rightarrow \quad \|x\|_2^2 = \sum_{j=1}^n |\alpha_j|^2.$$

We have as well that

$$Ax = \sum_{j=1}^n \alpha_j \lambda_j x_j \quad \Rightarrow \quad \|Ax\|_2^2 = \sum_{j=1}^n |\lambda_j \alpha_j|^2.$$

Therefore

$$\frac{\|Ax\|_2^2}{\|x\|_2^2} = \frac{\sum_j |\alpha_j|^2 |\lambda_j|^2}{\sum_j |\alpha_j|^2} \le \frac{\sum_j |\alpha_j|^2 |\lambda_1|^2}{\sum_j |\alpha_j|^2} = |\lambda_1|^2$$

from which we see that $\|A\|_2 \le |\lambda_1|$. Together with Theorem 7.1 the assertion follows. □

A result for non-square matrices:

Theorem 7.3. [1.32] *For all $A \in \mathbb{C}^{m\times n}$ the equality $\|A\|_2^2 = \rho(A^*A)$ holds true.*

Proof. The matrix A^*A is Hermitian, hence normal, and by Theorem 7.2

$$\rho(A^*A) = \|A^*A\|_2 = \max_{\|x\|_2=1} \|A^*Ax\|_2 = \max_{\|x\|_2=1} \big(\max_{\|y\|_2=1} \big|\langle y, A^*Ax\rangle\big|\big) \qquad (\star)$$

where we used a duality argument for the last identity. On the one hand,

$$(\star) \ \geq \max_{\|x\|_2=1} \big|\langle x, A^*Ax\rangle\big| = \max_{\|x\|_2=1} \big|\langle Ax, Ax\rangle\big| = \max_{\|x\|_2=1} \|Ax\|_2^2 = \|A\|_2^2.$$

On the other hand, using Cauchy-Schwarz

$$\begin{aligned}(\star) \ \leq \max_{\|x\|_2=1} \big(\max_{\|y\|_2=1} \big|\langle Ay, Ax\rangle\big|\big) &\leq \max_{\|x\|_2=1} \big(\max_{\|y\|_2=1} \|Ay\|_2\|Ax\|_2\big) \\ &= \big(\max_{\|x\|_2=1} \|Ax\|_2\big)\big(\max_{\|y\|_2=1} \|Ay\|_2\big) = \|A\|_2^2.\end{aligned}$$

□

Exercise: Use the above theorem to show that $\|UA\|_2 = \|A\|_2 = \|AV\|_2$ for all unitary matrices $U \in \mathbb{C}^{m\times m}$ and $V \in \mathbb{C}^{n\times n}$.

δ-Jordan canonical form

For any $A \in \mathbb{C}^{n\times n}$ there is an invertible $S \in \mathbb{C}^{n\times n}$ and a $J \in \mathbb{C}^{n\times n}$ such that $A = SJS^{-1}$ where J is a Jordan matrix, i.e., denoting by $\lambda_1, \dots, \lambda_k$ the eigenvalues of A, J is of the form

$$J = \begin{pmatrix} J_{n_1}(\lambda_1) & 0 & \cdots & 0 \\ 0 & J_{n_2}(\lambda_2) & \ddots & \vdots \\ \vdots & \ddots & \ddots & 0 \\ 0 & \cdots & 0 & J_{n_k}(\lambda_k) \end{pmatrix}$$

with δ-Jordan blocks

$$J_{n_l}(\lambda_l) = \begin{pmatrix} \lambda_l & \delta & 0 & \cdots & 0 \\ 0 & \ddots & \ddots & \ddots & \vdots \\ \vdots & \ddots & \ddots & \ddots & 0 \\ \vdots & \ddots & \ddots & \ddots & \delta \\ 0 & \cdots & \cdots & 0 & \lambda_l \end{pmatrix}$$

This corresponds to the standard Jordan canonical form except that the 1s in the first off-diagonal are replaced by δs. But recall that the off-diagonal elements serve to characterise the dimensions of the eigenspaces and stand for discrete information whence we may represent this information by a number different from 1. In fact, one can derive the above δ-Jordan canonical form from the usual one by an appropriate similarity transformation (Exercise).
Later on, we will pick an arbitrary small δ and use the following lemma which says that the spectral radius is no matrix norm but can be approximated by matrix norms.

Lemma 7.4. *For any $A \in \mathbb{C}^{n\times n}$ and $\delta > 0$ there is a vector norm $\|\cdot\|_\delta$ on $\mathbb{C}^n$ such that the induced norm fulfils $\rho(A) \leq \|A\|_\delta \leq \rho(A) + \delta$.*

Proof. The first inequality is true by Theorem 7.1.
Let $A = S_\delta J_\delta S_\delta^{-1}$ be the factorisation such that J_δ is the δ-Jordan canonical form and define the norm $\|\cdot\|_\delta$ by

$$\|x\|_\delta := \|S_\delta^{-1}x\|_\infty, \quad x \in \mathbb{C}^n$$

Then

$$\begin{aligned}
\|A\|_\delta &= \max_{x\neq 0} \frac{\|Ax\|_\delta}{\|x\|_\delta} \\
&= \max_{x\neq 0} \frac{\|S_\delta^{-1}Ax\|_\infty}{\|S_\delta^{-1}x\|_\infty}
\end{aligned}$$

and inserting $y = S_\delta^{-1}x$ this is

$$\begin{aligned}
&= \max_{y,\, S_\delta y\neq 0} \frac{\|S_\delta^{-1}AS_\delta y\|_\infty}{\|y\|_\infty} \\
&= \max_{y\neq 0} \frac{\|J_\delta y\|_\infty}{\|y\|_\infty} \\
&= \|J_\delta\|_\infty
\end{aligned}$$

and recalling that $\|\cdot\|_\infty$ is the maximum row sum we obtain that this is

$$\begin{aligned}
&= \max_i \sum_j |(J_\delta)_{i,j}| \\
&\leq \max_i |\lambda_i| + \delta \\
&= \rho(A) + \delta
\end{aligned}$$

where we used the special structure of J_δ for establishing the inequality. $\square$

Lecture 8

Error Analysis

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

$$\begin{tikzcd}\boxed{\texttt{Input}} \arrow[r] & \boxed{\texttt{Algorithm}} \arrow[r] & \boxed{\texttt{Result}}\end{tikzcd}$$

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

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

$$\begin{tikzcd}\boxed{\begin{array}{l}\texttt{Input}\\ \texttt{Errors}\end{array}} \arrow[r] & \boxed{\begin{array}{l}\texttt{Algorithm}\\ \texttt{Errors}\end{array}} \arrow[r] & \boxed{\begin{array}{l}\texttt{Output}\\ \texttt{Errors}\end{array}}\end{tikzcd}$$

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

<u>Input Errors</u>: We have two sources in mind,

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

<u>Algorithm Errors</u>: Again, there a mainly two sources,

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

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

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

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

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

Assumption 8.1. [3.19]

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

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

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

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

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

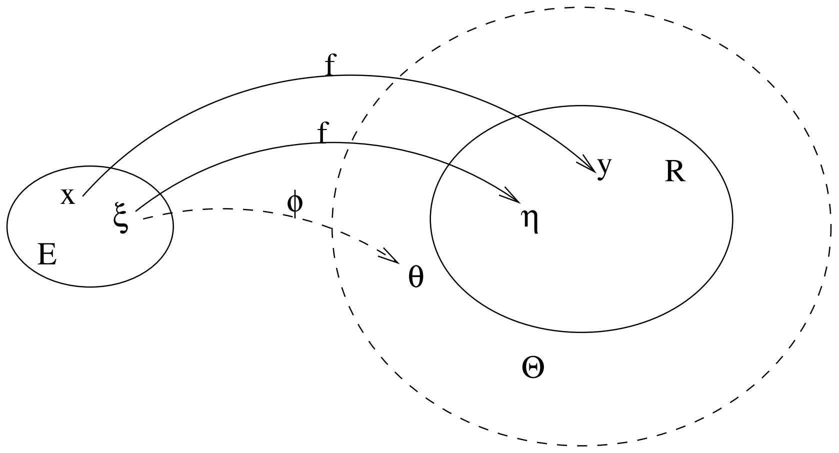
In particular, overflow and underflow are neglected.

Forward Error Analysis

Subject is the analysis of $\Theta = \phi(E)$ where ϕ is an algorithm for f. The set E represents input data that cannot be distinguished from x,

$$E := \{\xi \mid \|\xi - x\| \le \delta \|x\|\} \text{ with a given accuracy / tolerance } \delta. \tag{8.1}$$

The set R will be defined later on.



Definition 8.2. [3.17] *The forward error is defined by:*

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

An algorithm is forward stable if

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

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

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

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

Hence we obtain for the relative forward error that

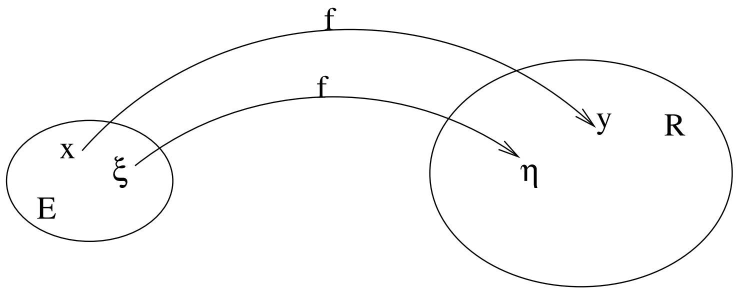
$$
\frac{|\tilde{f}(x)-f(x)|}{|f(x)|} = |\varepsilon_1+\varepsilon_2-\varepsilon_3+\delta_1+\delta_2+O(\varepsilon_m^2)| \leq 5\varepsilon_m + O(\varepsilon_m^2) = O(\varepsilon_m) \quad \text{as } \varepsilon_m \to 0.
$$

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

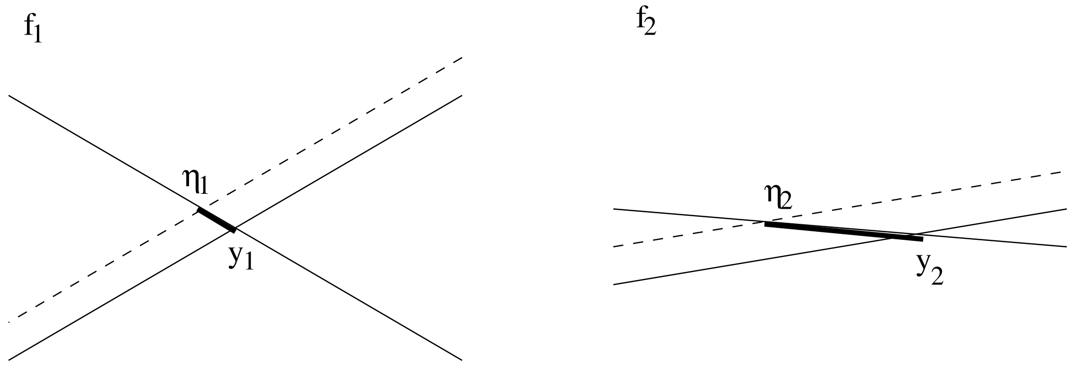
Lecture 9

Conditioning

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



The result set is $R = f(E)$. The Conditioning depends on the size of R. The problem f is called well-conditioned if R small and ill-conditioned if R is rather big.
Example: Consider a 2×2 linear system of equations or, as an equivalent geometric problem, the cut of two lines.



Errors in the input data correspond to changes of the lines. Problem f_1 is better conditioned than problem f_2 since the change of the intersection point is less drastic when shifting one of the lines by about the same amount. Appearently, here a small angle between the lines in unfavourable.
The general goal is to find a significant number to measure how well a problem is conditioned.

Consider a simple scalar problem $f : \mathbb{R} \to \mathbb{R}$. Using the Taylor expansions we have to first order $f(x+h) - f(x) \approx f'(x)h$ from which we obtain

$$\frac{f(x+h) - f(x)}{f(x)} \approx \left[\frac{xf'(x)}{f(x)}\right] \times \frac{h}{x}$$

which relates the relative output error (on the left hand side) to the relative input error h/x. A meaningful definition of a condition number (of problem f in a point x) therefore is

$$\kappa_f(x) := \Big\| \frac{xf'(x)}{f(x)} \Big\|.$$

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

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

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

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

Conditioning of (SLE)

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

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

We deduce that

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

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

Definition 9.1. [3.2] *The condition number of a square matrix $A \in \mathbb{C}^{n\times n}$ is*

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

One can show the following:

Proposition 9.2. [3.3] *Let A regular, $Ax = b \neq 0$ and $A(x + \Delta x) = b + \Delta b$. Then*

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

In order to get an estimate with respect to perturbations of the matrix A we need the following lemma:

Lemma 9.3. [3.4] *Assume that $B \in \mathbb{C}^{n\times n}$ satisfies $\|B\| < 1$. Then $I + B$ is invertible and*

$$\|(I + B)^{-1}\| \leq (1 - \|B\|)^{-1}.$$

Proposition 9.4. [3.5] *Let A be regular, $Ax = b \neq 0$ and $(A + \Delta A)(x + \Delta x) = b$. If $\kappa(A)\|\Delta A\| < \|A\|$ then*

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

Proof. Clearly

$$\|A^{-1}\Delta A\| \leq \|A^{-1}\|\|\Delta A\| = \kappa(A)\frac{\|\Delta A\|}{\|A\|}. \tag{+}$$

From the assumptions $(A+\Delta A)\Delta x = -\Delta A x$, hence

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

By Lemma 9.3 and (+) $(I + A^{-1}\Delta A)^{-1}$ exists so that

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

Using (+) again yields the assertion:

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

□

Putting the results of the propositions 9.2 and 9.4 together one can show

Theorem 9.5. *Let $A, \Delta A \in \mathbb{C}^{n\times n}$ regular such that $\kappa(A)\|\Delta A\| \leq \|A\|$ and let $b \in \mathbb{C}^n\backslash\{0\}$, $\Delta b \in \mathbb{C}^n$. Assume that $x \in \mathbb{C}^n$ solves $Ax = b$ and $\hat{x}$ solves $(A + \Delta A)\hat{x} = b + \Delta b$. Then*

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

Lecture 10

Backward Error Analysis

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

Definition 10.1. *An algorithm is a process consisting of a set of elementary components (called steps) to derive specific output data from specific input data where the following conditions have to be fulfilled:*

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

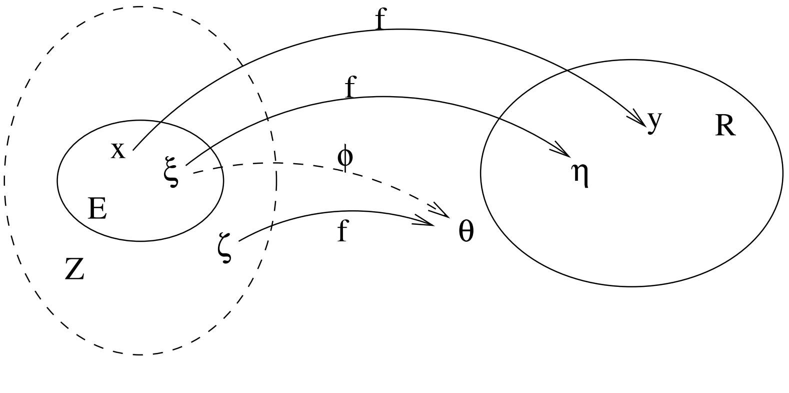
Elementary executable operations are the operations $\{+, -, \times, /, \sqrt{\cdot}\}$. An algorithm for a map f is a decomposition

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

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

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

Backward Error Analysis: The basic idea is to assume that the computed value $\theta = \phi(\xi)$ is the exact result for perturbed input data ζ of ξ, i.e. $\theta = f(\zeta)$. If ζ does not exist the algorithm is termed not backward stable. If there are multiple choices for ζ (f not injective) we choose ζ minimising $\|\cdot\|$.



Definition 10.2. [3.17],[3.20] *The backward error is defined by:*

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

An algorithm is backward stable if

$$\frac{\|\zeta - x\|}{\|x\|} = O(\varepsilon_m) \quad \textit{as } \varepsilon_m \searrow 0. \tag{10.1}$$

This notion of backward stability is useful in the following sense: Obtaining the solution with an algorithm involving approximations instead of performing exact computation is converted to exactly solving the problem with perturbed initial data. But then the conditioning of the problem can tell us how far the computed solution may deviate from the exact one.
Recall that the condition number was intended to yield an estimate of the form

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

But this means that

$$\text{relative forward error} \le \kappa_f \text{ relative backward error.}$$

Recall that we finally are interested in estimating the forward error. The above outlined backward error analysis typically yields sharper estimates than the standard forward error analysis and, in addition, splits into a problem intrinsic part (the conditioning) and an algorithm intrinsic part (the backward error). For a deeper discussion on error analysis, see [5] (Higham, 2002).

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

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

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

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

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

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

is a backward stable algorithm in the sense that

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

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

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

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

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

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

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

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

as well as

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

which yields the asserted estimate of the relative backward error. $\square$

Lecture 11

Error Analysis of the Gaussian Elimination

Some notation: If $A \in \mathbb{C}^{m\times n}$ then

$$|A| := (|a_{i,j}|)_{i,j=1}^{m,n} \in \mathbb{C}^{m\times n},$$

and inequalities like $|B| \le |A|$ for matrices have to be understood as valid for each element.

Theorem 11.1. [5.5] **FS** *and* **BS** *are backward stable.*
The computed solution $\hat{x}$ satisfies $(T + \Delta T)\hat{x} = b$ with some triangular matrix ΔT satisfying

$$|\Delta T| \le (n\varepsilon_m + o(\varepsilon_m))|T| \quad \textit{as } \varepsilon_m \to 0. \tag{11.1}$$

Proof. We consider the forward substitution for a unit lower triangular T only. The algorithm can be recursively defined by

$$x_k = b_k - \langle l^{k-1}, x^{k-1}\rangle$$

for $k = 1, \dots, n$ where

$$x^{k-1} = (x_1, \dots, x_{k-1})^T, \quad l^{k-1} = (\bar{l}_{k,1}, \dots, \bar{l}_{k,k-1})^T$$

and the $l_{i,j}$ are the entries of T. Using the result for the scalar product, a realisation in floating point arithmetic yields

$$\hat{x}_k = (b_k - \phi_{k-1}(l^{k-1}, x^{k-1}))(1 + \varepsilon^{(k)}) = (b_k - \langle \hat{l}^{k-1}, x^{k-1}\rangle)(1 + \varepsilon^{(k)})$$

with a vector $\hat{l}^{k-1} \in \mathbb{C}^{k-1}$ such that

$$|\hat{l}_i^{k-1} - l_i^{k-1}| \le ((k-1)\varepsilon_m + o(\varepsilon_m))|l_i^{k-1}| \le (n\varepsilon_m + o(\varepsilon_m))|l_{k,i}|.$$

Setting $(\Delta T)_{k,i} := \hat{l}_i^{k-1} - l_i^{k-1}$, $i = 1, \dots, k-1$, and $k = 2, \dots, n$, and $(\Delta T)_{k,k} := -\varepsilon^{(1)}$ then we indeed have that $(T + \Delta T)\hat{x} = b$, and (11.1) holds true, too. □

Remark: The results (11.1) can be improved to

$$|\Delta T| \le \frac{n\varepsilon_m}{1 - n\varepsilon_m}|T|.$$

With respect to the Gaussian elimination, the following result is proved in [5] (Higham, 2002).

Theorem 11.2. [5.6] *Assume that the LU factorisation of a matrix $A \in \mathbb{C}^{n\times n}$ exists and denote by $\hat{L}$, $\hat{U}$ the LU factors computed by* **LU**. *Then $\hat{L}\hat{U} = A + \Delta A$ where*

$$|\Delta A| \le \frac{n\varepsilon_m}{1-n\varepsilon_m}|\hat{L}||\hat{U}| \le (n\varepsilon_m + o(\varepsilon_m))|\hat{L}||\hat{U}|.$$

As a consequence of Theorems 11.1 and 11.2 we obtain

Theorem 11.3. [5.7] *Assume that the LU factorisation of a matrix $A \in \mathbb{C}^{n\times n}$ exists and denote by $\hat{L}$, $\hat{U}$ the LU factors computed by* **LU**. *Then the solution $\hat{x} \in \mathbb{C}^n$ for $Ax = b$ computed by* **GE** *satisfies*

$$(A + \Delta A)\hat{x} = b$$

with a matrix $\Delta A \in \mathbb{C}^{n\times n}$ satisfying

$$|\Delta A| \le \frac{3n\varepsilon_m}{1-3n\varepsilon_m}|\hat{L}||\hat{U}| \le (3n\varepsilon_m + o(\varepsilon_m))|\hat{L}||\hat{U}|.$$

In order to obtain an estimate for the relative backward error (something like $\|\Delta A\|/\|A\|$) it apparently would be sufficient to estimate $\||\hat{L}||\hat{U}|\|$ in terms of $\|A\|$. To see that this can be problematic consider the matrix

$$A = \begin{pmatrix} \delta & 1 \\ 1 & 1 \end{pmatrix} \quad \Rightarrow \quad L = \begin{pmatrix} 1 & 0 \\ \frac{1}{\delta} & 1 \end{pmatrix}, U = \begin{pmatrix} \delta & 1 \\ 0 & 1-\frac{1}{\delta} \end{pmatrix}$$

where $0 < \delta \ll 1$ is small. The condition number is $\kappa(A) = 4 + O(\delta)$ so the problem of solving $Ax = b$ is well-conditioned. But

$$|L||U| = \begin{pmatrix} \delta & 1 \\ 1 & \frac{2}{\delta}-1 \end{pmatrix} \quad \Rightarrow \quad \||L||U|\|_\infty = O\left(\frac{1}{\delta}\right) \text{ as } \delta \to 0.$$

So we have no chance to estimate $\||\hat{L}||\hat{U}|\|_\infty$ in terms of $\|A\|_\infty = 2$. But

$$PA = \begin{pmatrix} 1 & 1 \\ \delta & 1 \end{pmatrix} \quad \Rightarrow \quad L = \begin{pmatrix} 1 & 0 \\ \delta & 1 \end{pmatrix}, U = \begin{pmatrix} 1 & 1 \\ 0 & 1-\delta \end{pmatrix}$$

with P being the permutation matrix exchanging rows 1 and 2. And then

$$|L||U| = \begin{pmatrix} 1 & 1 \\ \delta & 1 \end{pmatrix} = PA,$$

so pivoting can cure this problem.

For a matrix $A \in \mathbb{C}^{n\times n}$ let L, U denote the LU factors computed by **GEPP** if all calculations are carried out exactly. Assume that the permutation matrix $\hat{P}$ computed by **GEPP** coincides with the correct one P. Let $b \in \mathbb{C}^n$.

Definition 11.4. [5.9] *The growth factor of A is defined by*

$$g_n(A) := \frac{\|U\|_{\max}}{\|A\|_{\max}}.$$

With this definition we obtain that

$$\|U\|_\infty \le n g_n(A)\|A\|_\infty.$$

Since all the matrix entries in L are ≤ 1 thanks to the pivoting we also have that

$$\|L\|_\infty \le n.$$

Altogether, replacing A by $PA = \hat{P}A$ and using that $\|P\|_\infty = 1$ we obtain a backward estimate:

Theorem 11.5. *The algorithm* **GEPP** *computes a vector* $\hat{x} \in \mathbb{C}^n$ *solving*

$$(A + \Delta A)\hat{x} = b$$

where $\Delta A \in \mathbb{C}^{n \times n}$ *satisfies*

$$\|\Delta A\|_\infty \leq C n^3 g_n(A) \varepsilon_m \|A\|_\infty \tag{11.2}$$

with a constant $C \geq 3$ *independent of* n*.*

To conclude, we need an answer on how big the growth factor $g_n(A)$ can become.

Lemma 11.6. [5.10] $g_n(A) \leq 2^{n-1}$ *for all* $A \in \mathbb{C}^{n \times n}$*, and this estimate is sharp.*

Proof. See [1] (Stuart & Voss notes). □

Remark: In view of the exponential growth of $g_n(A)$ in n the stability estimate (11.2) is rather weak. A better estimate for the growth factor is obtained when applying complete pivoting **GECP**. However, the upper bound in Lemma 11.6 is a worst case and, in practice, the additional effort due to the higher cost for the pivot search in **GECP** does not count off. Other, more costly algorithms for solving (SLE) with better stability properties will be discussed later on.

Lecture 12

Computational Cost

Recall Definition 10.1: An algorithm for a map f is a decomposition

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

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

Definition 12.1. [4.1] *The cost of an algorithm is*

$$C(n) = \textit{ number of elementary executable operations } [= l]$$

where n is a representative number for the size of the input data.

Example: Consider the following algorithm for the standard matrix-vector product.

Algorithm 5 MatVecStd (standard matrix-vector product)

```
input:  $A = (a_{i,j})_{i,j=1}^{m,n} \in \mathbb{C}^{m \times n}$, $b = (b_i)_{i=1}^{n} \in \mathbb{C}^n$.
output:  $x = Ab \in \mathbb{C}^m$.
1: for $i = 1$ to $m$ do
2:     $x_i := 0$
3:     for $j = 1$ to $n$ do
4:         $x_i := x_i + a_{i,j} b_j$
5:     end for
6: end for
```

Line 4 involves 1 addition and 1 multiplication $\rightsquigarrow$ 2op. With the loop in line 3 we obtain $n \times 2\text{op} = 2n\text{op}$. The assignment in line 2 does not count as an operation but from the loop in line 1 we get $m \times 2n\text{op}$, whence Algorithm 5 has computational cost

$$C(m, n) = 2mn\text{op}.$$

Any algorithm for the matrix-vector product will have at least $\Theta(m)$ operations as $m \to \infty$ which can be seen from the fact that m values have to be computed.

Some remarks:

- Estimating the computation time is **not** the aim as this depends too much on the computer architecture. We rather want to get an idea of the complexity of the algorithm for which the number of operations is a good measure.

- In older books there is a distinction between addition/subtraction (cheapest), multiplication, division, and the square root (most expensive) motivated by the way how the operations were carried out by the processors. Nowadays, the processors contain look-up tables so that the time to execute each of those operations basically is the same and this distinction is obsolete.
- High performance computers are parallel computers containing multiple processing units. Issues here are not only the number of operations but also the data exchange and the balancing of the work load. An algorithm is said to scale optimally if doubling the number of processors halves the computation time.

Let us turn our attention to the computational cost of solving systems of linear equations with Gaussian elimination, [5.1], [5.2].

Lemma 12.2. [5.2] **LU** *has cost* $C_{\mathbf{LU}}(n) = \frac{2}{3}n^3 + O(n^2)$ *as* $n \to \infty$.

Proof. Recall Algorithm 2.
Line 6: 1 addition and 1 multiplication, 2op.
Line 5: loop over i, $\sum_{i=k+1}^{n}(2\text{op}) = 2(n-k)\text{op}$.
Line 4: 1 division, $1 + 2(n-k)\text{op}$.
Line 3: loop over j, $\sum_{j=k+1}^{n}(2(n-k)+1)\text{op} = (n-k)(2(n-k)+1)\text{op}$.
Line 2: loop over k, as $n \to \infty$

$$\begin{aligned}\sum_{k=1}^{n-1}(n-k)(2(n-k)+1) &= 2\sum_{k=1}^{n-1}(n-k)^2 + \sum_{k=1}^{n-1}(n-k)\\ &= 2\sum_{l=1}^{n-1} l^2 + O(n^2)\\ &= \frac{2}{6}(n-1)n(2(n-1)+1) + O(n^2)\\ &= \frac{2}{3}n^3 + O(n^2).\end{aligned}$$

□

It is relatively straightforward to show that the cost of **FS** or **BS** are $O(n^2)$ which leads to the following result:

Theorem 12.3. *The computational cost of* **GE** *is*

$$C_{\mathbf{GE}}(n) \sim \frac{2}{3}n^3 \quad \text{as } n \to \infty.$$

The cost of **GEPP** is the same since the pivoting does not affect the computational cost as exchanging rows does not contribute to the cost according to our definition. But it should be remarked that exchanging data is a big issue on parallel computers and has a huge influence on the overall performance.
The algorithms seen so far have been relatively easy to analyse. Things are getting more interesting when considering divide & conquer algorithms.

Lecture 13

Divide & Conquer Algorithms

Divide & Conquer is an important design paradigm for algorithms. The idea is to break down a problem into sub-problems which are recursively solved until the problem become small enough to be solved directly. After, the sub-solutions are combined in merging steps to yield the solution to the originating problem.

As an example, we will look at a method to compute the matrix-matrix product which goes back to Strassen (1969). Recall that the standard matrix-matrix product $C = AB$ of to matrices $A, B \in \mathbb{C}^{n\times n}$ computed via

$$c_{i,j} = \sum_{k=1}^{n} a_{i,k} b_{k,j}, \quad i, j = 1, \dots, n$$

has a computational cost of $C_{\mathbf{MMStd}}(n) = \Theta(n^3)$ as $n \to \infty$. Since n^2 entries are to be computed any algorithm has cost $\Omega(n^2)$ as $n \to \infty$.

Assume, just for convenience, that $n = 2^k$ with some $k \in \mathbb{N}$ and write

$$A = \begin{pmatrix} A_{11} & A_{12} \\ A_{21} & A_{22} \end{pmatrix} \in \mathbb{C}^{2^k \times 2^k}, \quad A_{ij} \in \mathbb{C}^{2^{k-1}\times 2^{k-1}}$$

and analogously for B, C. $C = AB$ then means that

$$\begin{aligned} C_{11} &= A_{11}B_{11} + A_{12}B_{21}, \\ C_{12} &= A_{11}B_{12} + A_{12}B_{22}, \\ C_{21} &= A_{21}B_{11} + A_{22}B_{21}, \\ C_{22} &= A_{21}B_{12} + A_{22}B_{22}. \end{aligned}$$

Computing C this way involves eight multiplications of $2^{k-1} \times 2^{k-1} = \frac{n}{2} \times \frac{n}{2}$-matrices. Using recursion, the cost of an algorithms based on this splitting satisfies

$$C_{\mathbf{MMSplit}}(n) = 8C_{\mathbf{MMSplit}}(\tfrac{n}{2}) + n^2$$

where the n^2 contribution comes from the 4 additions of $\frac{n}{2} \times \frac{n}{2}$-matrices. One can show that this leads to a cost of $\Theta(n^3)$ as $n \to \infty$, and the algorithm is not better than the standard one.

However, defining the **seven** $2^{k-1} \times 2^{k-1}$-matrices

$$\begin{aligned}
P_1 &:= (A_{11} + A_{22})(B_{11} + B_{22}),\\
P_2 &:= (A_{21} + A_{22})B_{11},\\
P_3 &:= A_{11}(B_{12} - B_{22}),\\
P_4 &:= A_{22}(B_{21} - B_{11}),\\
P_5 &:= (A_{11} + A_{12})B_{22},\\
P_6 &:= (A_{21} - A_{11})(B_{11} - B_{12}),\\
P_7 &:= (A_{12} - A_{22})(B_{21} + B_{22}) \qquad (13.1)
\end{aligned}$$

one can show that

$$\begin{aligned}
C_{11} &= P_1 + P_4 - P_5 + P_7,\\
C_{12} &= P_3 + P_5,\\
C_{21} &= P_2 + P_4,\\
C_{22} &= P_1 + P_3 - P_2 + P_6. \qquad (13.2)
\end{aligned}$$

Using this recursively as in Algorithm 6 we obtain that the cost satisfies

$$C_{\textbf{MMStrassen}}(n) = 7^{k+1} - 6 \cdot 4^k = 7n^{\log_2(7)} - 6n^2, \quad n = 2^k,\ k \in \mathbb{N}, \qquad (13.3)$$

and since $\log_2(7) \approx 2.807 < 3$ we have got an algorithm that asymptotically is of lower cost $\Theta(n^{\log_2(7)})$ as $n \to \infty$ than the previously presented algorithms. Let us briefly prove (13.3):

Proof. The recursive definition of the Strassen multiplication invites for an induction proof. For $k = 0$ there is only one multiplication of scalars, and formula (13.3) indeed yields one.
Assume now that (13.3) is true for $k - 1$. As there are 18 additions of $\frac{n}{2} \times \frac{n}{2}$-matrices in (13.1) and (13.2) we obtain the formula

$$C_{\textbf{MMStrassen}}(2^k) = 7C_{\textbf{MMStrassen}}(2^{k-1}) + 18(2^{k-1})^2,$$

which yields, using the induction hypothesiss

$$\begin{aligned}
&= 7(7^k - 6 \times 4^{k-1}) + 18 \times 4^{k-1}\\
&= 7^{k+1} - 24 \times 4^{k-1}\\
&= 7^{k+1} - 6 \times 4^k.
\end{aligned}$$

□

To deal with the case that n is no power of 2 one may pick $k \in \mathbb{N}$ such that $2^k \geq n > 2^{k-1}$ and then define

$$\tilde{A} = \begin{pmatrix} A & 0 \\ 0 & 0 \end{pmatrix} \in \mathbb{C}^{2^k \times 2^k}$$

by adding some blocks containing zeros, similarly with $\tilde{B}$. It turns out that the upper left $n \times n$ block of $\tilde{C} := \tilde{A}\tilde{B}$ then contains the desired product $C = AB$. Applying the Strassen multiplication to $\tilde{A}$ and $\tilde{B}$ involves a cost that is $\Theta((2^k)^{\log_2(7)})$ as $k \to \infty$, but since $2^k \leq 2n$ by the choice of k in dependence of n this is $\Theta(n^{\log_2(7)})$ as $n \to \infty$. We may summarise the findings in

Theorem 13.1. *Using the Strassen multiplication one can construct an algorithm for computing the product of $n \times n$-matrices, $n \in \mathbb{N}$, with computational cost*

$$C(n) = \Theta(n^{\log_2(7)}) \quad \textit{as } n \to \infty.$$

An algorithm developed by Coppersmith and Winograd (1990) scales even better as it has a cost of about $\Theta(n^{2.376})$ but it is not (yet?) practical as its advantage only becomes perceptible for such big n that the corresponding matrices are just too big for even the most modern supercomputers. Any algorithm will have a cost $\Omega(n^2)$ as n^2 is the number of elements to be computed.

What is this studying of the exponent useful for? Well, if one can multiply $n\times n$-matrices with an asymptotic cost of $O(n^\alpha)$ as $n \to \infty$, $\alpha \geq 2$, then it is also possible to invert regular $n\times n$ matrices with cost $O(n^\alpha)$. For the proof one may use the Schur complement $S = D_{22} - D_{12}^* D_{11}^{-1} D_{12}$ of a Hermitian matrix

$$D = \begin{pmatrix} D_{11} & D_{12} \\ D_{12}^* & D_{22} \end{pmatrix}$$

and proceed recursively. The assertion on the cost then follows similarly as for the Strassen multiplication which is why the proof is omitted.

Algorithm 6 MMStrassen (Strassen matrix-matrix multiplication)

```
input:  $A, B \in \mathbb{C}^{n\times n}$ with $n = 2^k$ for some $k \in \mathbb{N}$.
output: $C = AB \in \mathbb{C}^{n\times n}$.
1: if $n = 1$ then
2:     return $C := AB$
3: else
4:     compute $P_1, \dots, P_7$ as defined in (13.1) using recursion for the matrix-matrix products
5:     compute $C_{11}, C_{12}, C_{21}, C_{22}$ according to (13.2)
6:     return C
7: end if
```

Lecture 14

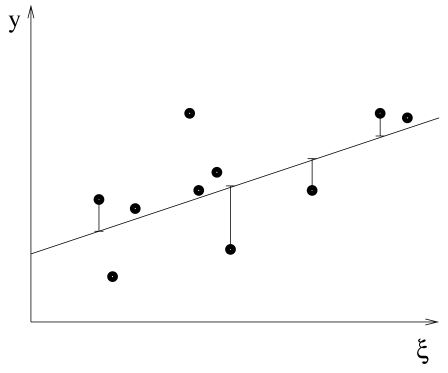
Least Squares Problems

Definition 14.1. *Given a matrix $A \in \mathbb{R}^{m\times n}$ and a vector $b \in \mathbb{R}^m$, the* <u>*least squares problem*</u> LSQ *consists of minimising the function*

$$g : \mathbb{R}^n \to \mathbb{R}, \quad g(x) = \frac{1}{2}\|Ax - b\|_2^2.$$

Example: Recall the linear regression problem. Given points $(\xi_i, y_i)_{i=1}^m$ find a linear function $\xi \to x_1 + x_2\xi$ such that

$$g(x) = \frac{1}{2}\sum_{i=1}^{m} \big(x_1 + x_2\xi_i - y_i\big)^2 \quad \text{is minimal.}$$



In this case,

$$A = \begin{pmatrix} 1 & \xi_1 \\ \vdots & \vdots \\ 1 & \xi_m \end{pmatrix}, \quad b = \begin{pmatrix} y_1 \\ \vdots \\ y_m \end{pmatrix}.$$

Theorem 14.2. [7.1] *$x \in \mathbb{R}^n$ solves the least squares problem if and only if $Ax - b \perp \text{range}(A)$ which is the case if and only if the* <u>*normal equation*</u>

$$A^T A x = A^T b \tag{7.1}$$

is satisfied.

Proof. If x minimises g then for all $y \in \mathbb{R}^n$

$$\begin{aligned}
0 &= \frac{d}{d\varepsilon} g(x+\varepsilon y)\Big|_{\varepsilon=0} \\
&= \frac{d}{d\varepsilon}\Big(\frac{1}{2}\langle Ax+\varepsilon Ay-b, Ax+\varepsilon Ay-b\rangle\Big)\Big|_{\varepsilon=0} \\
&= \frac{d}{d\varepsilon}\Big(\frac{1}{2}\langle Ax-b, Ax-b\rangle + \frac{1}{2}\varepsilon\big(\langle Ay, Ax-b\rangle + \langle Ax-b, Ay\rangle\big) + \frac{1}{2}\varepsilon^2\langle Ay, Ay\rangle\Big)\Big|_{\varepsilon=0} \\
&= \frac{d}{d\varepsilon}\Big(\frac{1}{2}\|Ax-b\|_2 + \varepsilon\langle Ay, Ax-b\rangle + \frac{1}{2}\varepsilon^2\|Ay\|_2\Big)\Big|_{\varepsilon=0} \\
&= \langle Ay, Ax-b\rangle
\end{aligned}$$

which means that $Ax-b \perp \text{range}(A)$. The other way round, if $Ax-b\perp\text{range}(A)$ then $\langle Ax-b, Ay-Ax\rangle = 0$ for all $y\in\mathbb{R}^n$, hence with Pythagoras

$$2g(y) = \|Ay-b\|_2^2 = \|Ay-Ax\|_2^2 + \|Ax-b\|_2^2 \geq \|Ax-b\|_2^2 = 2g(x).$$

For the second assertion we use that $Ax-b\perp\text{range}(A) \Leftrightarrow Ax-b\perp a_i$ where the a_i, $i=1,\dots,m$, are the column vectors of A. But this is equivalent to

$$\big(\langle a_i, Ax\rangle\big)_{i=1}^m = \big(\langle a_i, b\rangle\big)_{i=1}^m \quad \Leftrightarrow \quad (7.1)$$

□

Example: In the linear regression problem the normal equation is a 2×2 system where

$$A^TA = \begin{pmatrix} m & \sum_{i=1}^m \xi_i \\ \sum_{i=1}^m \xi_i & \sum_{i=1}^m \xi_i^2 \end{pmatrix}, \quad A^Tb = \begin{pmatrix} \sum_{i=1}^m y_i \\ \sum_{i=1}^m \xi_i y_i \end{pmatrix}.$$

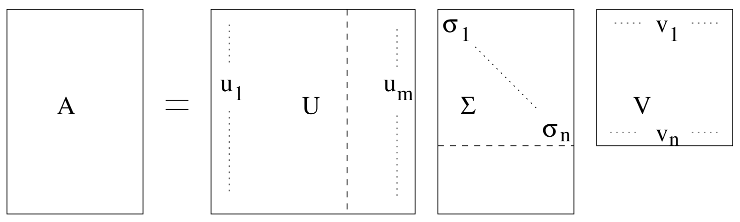
It is certainly possible to use (7.1) to solve LSQ, for example one could use Cholesky since A^TA is positive definite provided that A has full rank. But, as we will see later on, we have

$$\kappa_2(A^TA) = (\kappa_2(A))^2$$

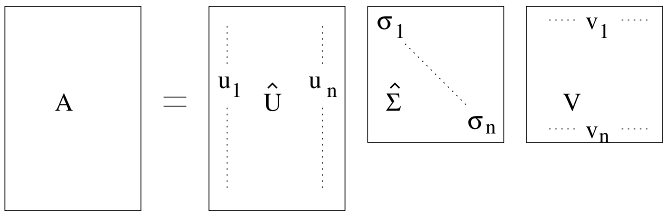
for the condition number, and even the condition number $\kappa_2(A)$ (which is not yet defined for $m\neq n$) can be big in practical applications. There are better approaches, based on the QR factorisation (later) or on the singular value decomposition that we consider next.

Singular Value Decomposition (SVD) [2.3]

This is a factorisation of the form $A = U\Sigma V^T$ where $U\in\mathbb{R}^{m\times m}$ and $V\in\mathbb{R}^{n\times n}$ are orthogonal matrices and $\Sigma\in\mathbb{R}^{m\times n}$ is a diagonal matrix with entries $\sigma_1\geq\sigma_2\geq\cdots\geq\sigma_p\geq 0$ where $p=\min(m,n)$. Those values σ_i are called singular values.



Denoting by u_i, $i = 1 = 1, \dots, m$, the column vectors of U and by v_i, $i = 1, \dots, n$, the column vectors of V the SVD says that $Av_i = \sigma_i u_i$, $i = 1, \dots, p$.
In the case $m \geq n(= p)$ which is of most interest to us the reduced singular value decomposition is defined by dropping the last $m - n$ columns of U and the last $m - n$ rows of Σ. Defining $\hat{U} := (u_1, \dots, u_n) \in \mathbb{R}^{m \times n}$ and $\hat{\Sigma} := \mathrm{diag}(\sigma_1, \dots, \sigma_n) \in \mathbb{R}^{n \times n}$ we then have that $A = \hat{U}\hat{\Sigma}V^T$.



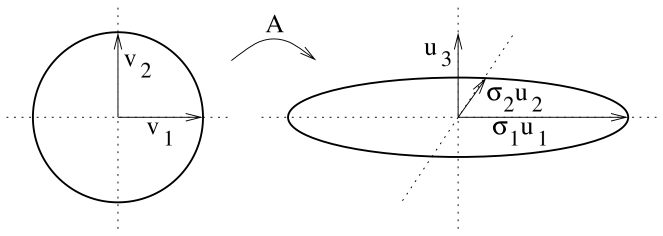
The SVD is a very powerful decomposition for analytical purposes as it provides much insight into the properties of the linear map associated with the matrix. The rank is the number of non-vanishing singular values, i.e., the biggest integer $r \leq p$ such that $\sigma_r > 0$ but $\sigma_{r+1} = 0$ (if $r + 1 \leq p$). Moreover, the image of A is spanned by the first r left singular vectors,

$$\mathrm{range}(A) = \mathrm{span}\{u_1, \dots, u_r\}$$

and the kernel is

$$\mathrm{kernel}(A) = \mathrm{span}\{v_{r+1}, \dots, v_p\}.$$

In addition to this algebraic information the SVD also reveals some geometrical information. Recalling the identities $Av_i = u_i\sigma_i$, $i = 1, \dots, p$, we see that the unit sphere in $\mathbb{R}^p$ containing the vectors v_i is mapped to an ellipsoid which has semi-axes in the directions of the u_i that have the extensions σ_i.



Example: Images can be compressed using the SVD. Try the following `matlab` code (it's probably best to put it into an m-file):

```
load clown.mat;
figure;
image(X);
colormap('gray');
pause
[U,S,V] = svd(X);
figure;
for k=1:1:60
    image(U(:,1:k)*S(1:k,1:k)*V(:,1:k)');
    colormap('gray');
    disp(k)
    disp('Reduction:  ')
    disp(521*k / 64000)
    pause
end
```

Lecture 15

More on the Singular Value Decomposition

Theorem 15.1. [2.12] *Every matrix $A \in \mathbb{R}^{m\times n}$ has a SVD, and the singular values are uniquely determined.*

Proof. We first show the existence by induction on $p = \min(m, n)$.
For $p = 1$ we may choose $u = 1$, and if $a_{11} \neq 0$ we may set $v = a_{11}/|a_{11}|$ and $\sigma_1 = |a_{11}|$ whilst in the case $a_{11} = 0$ we choose $v = 1$ and $\sigma_1 = 0$.
Let now $p > 1$ and assume without loss of generality that $A \neq 0$ (since the SVD is trivial otherwise with U and V being the identity and $\Sigma = 0$). The map $x \mapsto \|Ax\|_2$ on $\mathbb{R}^n$ is continuous. When restricted to the compact unit sphere $S^{n-1} = \{\|x\|_2 = 1 \,|\, x \in \mathbb{R}^n\}$ it attains a maximum which we denote by v_1. Further, we define

$$\sigma_1 := \|Av_1\|_2 = \max_{x\in S^{n-1}} \|Ax\|_2 = \|A\|_2.$$

Since $A \neq 0$ we have that $\sigma_1 > 0$ and we can define

$$u_1 := \frac{1}{\sigma_1} Av_1.$$

Let us now extend v_1 to an orthonormal basis (ONB) $\{v_1, \ldots, v_n\}$ of $\mathbb{R}^n$, and u_1 to an ONB $\{u_1, \ldots, u_m\}$ of $\mathbb{R}^m$. The matrices $U_1 := (u_1, \ldots, u_m) \in \mathbb{R}^{m\times m}$ and $V_1 := (v_1, \ldots, v_n) \in \mathbb{R}^{n\times n}$ then are orthogonal. Let

$$C := U_1^T AV_1 =: \begin{pmatrix} \sigma_1 & w^T \\ 0 & B \end{pmatrix}$$

with $w \in \mathbb{R}^{n-1}$ and $B \in \mathbb{R}^{m-1,n-1}$ where the zeros in the first column below the diagonal arise from the fact that $Av_1 = \sigma_1 u_1$ is orthogonal to the columns $u_2, \ldots, u_m$ of U_1. Then

$$\|C\|_2 = \max_{\|x\|_2=1} \|U_1^T AV_1 x\|_2 = \max_{\|V_1 x\|_2=1} \|AV_1 x\|_2 = \|A\|_2 = \sigma_1.$$

Since

$$\|C\|_2 \Big\| \begin{pmatrix} \sigma_1 \\ w \end{pmatrix} \Big\|_2 \geq \Big\| C \begin{pmatrix} \sigma_1 \\ w \end{pmatrix} \Big\|_2 = \Big\| \begin{pmatrix} \sigma_1^2 + \|w\|_2^2 \\ Bw \end{pmatrix} \Big\|_2 \geq \sigma_1^2 + \|w\|_2^2 \geq \sqrt{\sigma_1^2 + \|w\|_2^2} \Big\| \begin{pmatrix} \sigma_1 \\ w \end{pmatrix} \Big\|_2$$

we conclude that

$$\sigma_1 = \|C\|_2 \geq \sqrt{\sigma_1^2 + \|w\|_2^2} \quad \Rightarrow \quad w = 0.$$

By the induction hypothesis there is a singular value decomposition $B = U_2\Sigma_2 V_2^T$ of the $(m-1)\times(n-1)$ matrix B. Writing $\Sigma_2 = \operatorname{diag}(\sigma_2,\dots,\sigma_p)$ we observe that

$$\sigma_1 = \max_{\|x\|_2=1}\|Cx\|_2 \geq \max_{\|y\|_2=1,\, y\in\mathbb{R}^{n-1}}\left\|C\begin{pmatrix}0\\ y\end{pmatrix}\right\|_2 = \|By\|_2 = \sigma_2.$$

Therefore

$$A = U_1 C V_1^T = \underbrace{U_1\begin{pmatrix}1 & 0\\ 0 & U_2\end{pmatrix}}_{=:U}\underbrace{\begin{pmatrix}\sigma_1 & 0\\ 0 & \Sigma_2\end{pmatrix}}_{=:\Sigma}\underbrace{\begin{pmatrix}1 & 0\\ 0 & V_2^T\end{pmatrix}V_1^T}_{=:V^T}$$

is a SVD of A.
Coming to the second claim, we remark that for any SVD $A = U\Sigma V^T$

$$\|A\|_2 = \max_{\|x\|_2=1}\|U\Sigma V^T x\|_2 = \max_{\|V^Tx\|_2=1}\|\Sigma V^T x\|_2 = \max_{\|y\|_2=1} = \|\Sigma y\|_2 = \sigma_1.$$

Using an induction argument again one can show the uniqueness of the singular values. □

Corollary 15.2. $\|A\|_2 = \sigma_1$.

Examples:

1. A symmetric matrix $A\in\mathbb{R}^{n\times n}$ can be diagonalised in the form $A = Q\Lambda Q^T$ with $Q\in\mathbb{R}^{n\times n}$ orthogonal and $\Lambda = \operatorname{diag}(\lambda_1,\dots,\lambda_n)\in\mathbb{R}^{n\times n}$. Without loss of generality we may assume that $|\lambda_1|\geq\dots\geq|\lambda_n|$. Otherwise perform a similarity transformation of Λ with an appropriate permutation matrix which then is absorbed into Q. Denote the columns of Q by $q_1,\dots,q_n$. A SVD of A is obtained by setting $U := (u_1,\dots,u_n)$ where $u_i = \operatorname{sign}(\lambda_i)q_i$, $\Sigma = \operatorname{diag}(|\lambda_1|,\dots,|\lambda_n|)$, and $V := Q$.

2. Let $A\in\mathbb{R}^{m\times n}$ with $m\geq n$ (so that $p = \min(m,n) = n$). The matrix $A^TA\in\mathbb{R}^{n\times n}$ has eigenvalues σ_i^2 with corresponding eigenvectors v_i, $i = 1,\dots,p$. The matrix $AA^T\in\mathbb{R}^{m\times m}$ has eigenvectors $\{u_1,\dots,u_m\}$ with corresponding eigenvalues $\{\sigma_1^2,\dots,\sigma_p^2,0,\dots,0\}$ (with $m-n$ zeros).
   To see this, let us exemplary consider the latter case. We have that
   $$AA^T = U\Sigma V^TV\Sigma^TU^T \quad\Rightarrow\quad AA^TU = U\Sigma\Sigma^T,$$
   but $\Sigma\Sigma^T = \operatorname{diag}(\sigma_1^2,\dots,\sigma_p^2,0,\dots,0)$ with exactly $m-n$ zeros.

3. Assume now that $A\in\mathbb{R}^{n\times n}$ is regular. Then the matrix
   $$H := \begin{pmatrix}0 & A^T\\ A & 0\end{pmatrix}$$
   has $2n$ eigenvalues $\{\sigma_1,-\sigma_1,\sigma_2,-\sigma_2,\dots,\sigma_n,-\sigma_n\}$
   with corresponding eigenvectors $\{\binom{v_1}{u_1},\binom{v_1}{-u_1},\binom{v_2}{u_2},\binom{v_2}{-u_2},\dots,\binom{v_n}{u_n},\binom{v_n}{-u_n}\}$.
   To show this, assume that $Hx = \lambda x$ for some $\lambda\in\mathbb{R}$ and some $x\in\mathbb{R}^{2n}\backslash\{0\}$. Writing $x = \binom{y}{z}$ with $y,z\in\mathbb{R}^n$ this means that
   $$\begin{array}{c} A^Tz = \lambda y,\\ Ay = \lambda z, \end{array} \quad\Rightarrow\quad \begin{array}{c} AA^Tz = \lambda Ay = \lambda^2 z,\\ A^TAy = \lambda A^Tz = \lambda^2 y. \end{array}$$
   From the previous example we know that the eigenvalues of A^TA and AA^T are $\{\sigma_1^2,\dots,\sigma_n^2\}$ where $\sigma_n > 0$ by the regularity of A. Hence $\lambda = \pm\sigma_i$ for some $i\in\{1,\dots,n\}$. That $\binom{v_i}{\pm u_i}$ is a corresponding eigenvector is easy to show.

Remark 15.3. *The above results hold true analogously for complex matrices if the transposed matrices are replaced by the adjoint matrices.*

Computation of the SVD (remarks only, [7.3])

The idea is to exploit the last example which yields a relation between the eigenspaces of the matrix $H = \left(\begin{smallmatrix} 0 & A^T \\ A & 0 \end{smallmatrix}\right)$ and the SVD of A. Later on in this course we will see methods for computing eigenvalues and eigenvectors. These will be iterative methods requiring a matrix-vector multiplication per step as most costly ingredient ($O(n^2)$ as $n \to \infty$). The matrix A therefore is preprocessed. By applying some similarity transformations (that preserve the eigenvalues!) it is transformed into bidiagonal form so that the matrix-vector multiplication with H has cost $\leq 6n$. In [4] the total cost for computing a SVD of an $m \times n$ matrix is stated to be

$$C_{\mathbf{SVD}}(m,n) = 2mn^2 + 11n^3 + \Theta(mn + n^2) \quad \text{as } m, n \to \infty.$$

Lecture 16

Conditioning of LSQ

Recall the normal equation (7.1) $A^T Ax = A^T b$ for a solution to LSQ.
Introducing the pseudo-inverse or Moore-Penrose inverse

$$A^\dagger := (A^T A)^{-1} A^T$$

the solution is just $x = A^\dagger b$, provided $A^\dagger$ exists. This is the case if and only if A has full rank which is equivalent to $A^T A$ being regular.

Definition 16.1. [3.7] *The condition number of a matrix $A \in \mathbb{C}^{m\times n}$ with respect to a norm $\|\cdot\|$ is*

$$\kappa(A) = \begin{cases} \|A\| \, \|A^\dagger\| & \text{if } A \text{ has full rank,} \\ \infty & \text{otherwise.} \end{cases}$$

We remark that $A^\dagger = A^{-1}$ if $m = n$ so that the above definition is consistent with the previous one for $n \times n$ matrices.

Lemma 16.2. *If A has full rank: $\kappa_2(A) = \sigma_1/\sigma_n$ where σ_1 and σ_n are the biggest and smallest singular value, respectively.*

Proof. From Corollary 15.2 we know that $\|A\|_2 = \sigma_1$. Writing $A = U\Sigma V^T$ for a SVD, a short calculation results in a SVD

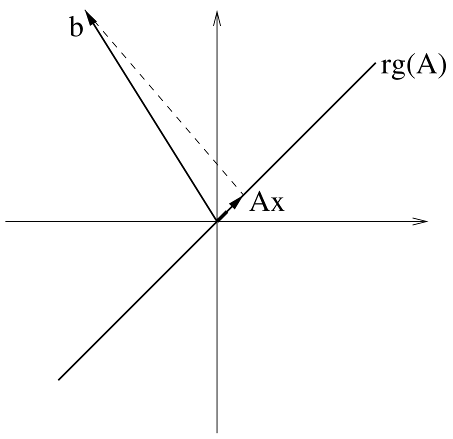
$$A^\dagger = V\tilde{\Sigma}\tilde{U}^T \tag{16.1}$$

with $\tilde{\Sigma} = \operatorname{diag}(\frac{1}{\sigma_n}, \dots, \frac{1}{\sigma_1})$ so that $\|A^\dagger\|_2 = \frac{1}{\sigma_n}$. □

Moreover $\|b\|_2^2 = \|b - Ax\|_2^2 + \|Ax\|_2^2 \geq \|Ax\|_2^2$ which motivates to introduce the angle $\theta \in [0, \frac{\pi}{2}]$ via

$$\cos(\theta) = \frac{\|Ax\|_2}{\|b\|_2}.$$

To provide a geometric interpretation, this is the angle between the b and the range of A:



Theorem 16.3. *Assume that $x \neq 0$ solves* LSQ *for data (A, b) and $x + \Delta x$ for $(A, b + \Delta b)$. Then*

$$\frac{\|\Delta x\|_2^2}{\|x\|_2^2} \le \frac{\kappa_2(A)}{\eta \cos(\theta)} \frac{\|\Delta b\|_2}{\|b\|_2}$$

where $\eta := \|A\|_2 \|x\|_2 / \|Ax\|_2 \ge 1$.

Proof. Assume that A has full rank (otherwise the assertion is trivial). From the assumptions we furthermore have that $\Delta x = A^\dagger \Delta b$. Therefore

$$\frac{\|\Delta x\|_2}{\|x\|_2} \le \frac{\|A^\dagger\|_2 \|\Delta b\|_2}{\|x\|_2} = \frac{\kappa_2(A)\|\Delta b\|_2}{\|A\|_2\|x\|_2} \le \frac{\kappa_2(A)\|\Delta b\|_2}{\eta\|Ax\|_2} \le \frac{\kappa_2(A)}{\eta\cos(\theta)}\frac{\|\Delta b\|_2}{\|b\|_2}.$$

□

Imagine now that b is (almost) orthogonal to range(A) which means that the solution is close to zero, $\|x\|$ is small. A small error in the data b then may to a small absolute deviation in the solution but can also lead to a big relative error in the solution. Example:

$$A = \begin{pmatrix} 1 \\ 1 \end{pmatrix}, \quad b = \begin{pmatrix} -1+\delta \\ 1 \end{pmatrix} \quad \rightsquigarrow \quad x = \frac{\delta}{2}.$$

Now, think of a small error in the data b, $\Delta b = (\epsilon, 0) \rightsquigarrow \Delta x = \epsilon/2$. Hence, $\|\Delta x\|_2/\|x\|_2 = \epsilon/\delta$ may be large.

Solving LSQ using the SVD [7.3]

Algorithm 7 LSQ_SVD (solving LSQ using a SVD)

```
input:  $A \in \mathbb{R}^{m\times n}$ with $m \ge n = \mathrm{rank}(A)$, $b \in \mathbb{R}^m$.
output: $x \in \mathbb{R}^n$ minimiser of $g(x) = \frac{1}{2}\|Ax - b\|_2^2$.
1: compute the reduced SVD $A = \hat{U}\hat{\Sigma}V^T$
2: compute $y = \hat{U}^T b$
3: solve $\hat{\Sigma}z = y$
4: return $x = Vz$
```

The thus computed x satisfies

$$A^T A x = V\hat{\Sigma}^T \underbrace{\hat{U}^T\hat{U}}_{=I} \hat{\Sigma} V^T x = V\hat{\Sigma}^T\hat{\Sigma} z = V\hat{\Sigma}^T y = V\hat{\Sigma}^T\hat{U}^T b = A^T b$$

which is the normal equation (7.1) and, hence, x indeed solves LSQ.
The cost of **LSQ_SVD** is dominated by computing a reduced SVD of A as the subsequent steps essentially are matrix-vector multiplications only. According to [4] we have

$$C_{\mathbf{LSQ\_SVD}}(m,n) = 2mn^2 + 11n^3 + \Theta(mn + n^2) \quad \text{as } m,n \to \infty.$$

This is more that for solving the normal equation with, for instance, GEPP (or Cholesky, a special version for positive definite matrices). The benefit is better stability. Example:

$$A = \begin{pmatrix} 1 & 1 \\ \delta & 0 \\ 0 & \delta \end{pmatrix}, \quad b = \begin{pmatrix} 2 \\ \delta \\ \delta \end{pmatrix} \quad \rightsquigarrow \quad x = \begin{pmatrix} 1 \\ 1 \end{pmatrix}.$$

When using the normal equation we encounter problems for much larger values of δ than when employing the method based on the SVD.

We will next learn about a method in between the two presented methods with respect to stability and cost.

Lecture 17

QR factorisation

Let $A \in \mathbb{R}^{m\times n}$ with $m \geq n$ and denote the column vectors by a_i, $i = 1, \dots, n$.
To orthonormalise the columns of A one may proceed as follows:

$$
\begin{aligned}
&\hat{q}_1 := a_1 && q_1 := \frac{\hat{q}_1}{r_{1,1}} \quad \text{with } r_{1,1} := \|\hat{q}_1\|_2,\\
&\hat{q}_2 := a_2 - \underbrace{\langle q_1, a_2\rangle}_{=:r_{1,2}} q_1 && q_2 := \frac{\hat{q}_2}{r_{2,2}} \quad \text{with } r_{2,2} := \|\hat{q}_2\|_2,\\
&\hat{q}_3 := a_3 - \underbrace{\langle q_1, a_3\rangle}_{=:r_{1,3}} q_1 - \underbrace{\langle q_2, a_3\rangle}_{=:r_{2,3}} q_2 && q_3 := \frac{\hat{q}_3}{r_{3,3}} \quad \text{with } r_{3,3} := \|\hat{q}_3\|_2,\\
&\quad\vdots && \quad\vdots\\
&\hat{q}_n := a_n - \sum_{j=1}^{n-1} \underbrace{\langle q_j, a_n\rangle}_{=:r_{j,n}} q_j && q_n := \frac{\hat{q}_n}{r_{n,n}} \quad \text{with } r_{n,n} := \|\hat{q}_n\|_2,
\end{aligned}
$$

This is the so-called Gram-Schmidt orthonormalisation. Equivalently to the above formulas, we can write

$$
a_i = \sum_{k=1}^{i} q_k r_{ik}, \quad i = 1, \dots, n, \qquad \Leftrightarrow \qquad A = \hat{Q}\hat{R} \tag{17.1}
$$

where

$$
\hat{Q} = (q_1, \dots, q_n) \in \mathbb{R}^{m\times n} \quad \text{and} \quad \hat{R} = \begin{pmatrix} r_{1,1} & r_{1,2} & \cdots & r_{1,n} \\ 0 & r_{2,2} & \ddots & \vdots \\ \vdots & \ddots & \ddots & r_{n-1,n} \\ 0 & \cdots & 0 & r_{n,n} \end{pmatrix} \in \mathbb{R}^{n\times n}.
$$

A factorisation of the form (17.1) with $\hat{Q}$ having orthonormal column vectors and $\hat{R}$ upper triangular is called reduced QR factorisation. Extending $\{q_1, \dots, q_n\}$ by vectors $\{q_{n+1}, \dots, q_m\}$ to an orthonormal basis of $\mathbb{R}^m$, defining $Q = (q_1, \dots, q_m) \in \mathbb{R}^{m\times m}$ and extending $\hat{R}$ with an $(m-n)\times n$ block of zeros to

$$
R := \begin{pmatrix} \hat{R} \\ 0 \end{pmatrix} \in \mathbb{R}^{m\times n}
$$

we obtain a

$$\underline{\text{QR factorisation:}} \quad A = QR. \tag{17.2}$$

Theorem 17.1. *Every matrix $A \in \mathbb{R}^{m\times n}$ with $m \geq n$ can be factorised in the form $A = QR$ with $Q \in \mathbb{R}^{m\times m}$ orthogonal and $R \in \mathbb{R}^{m\times n}$ upper triangular.*

Solving LSQ using the QR factorisation [7.2]

Algorithm 8 LSQ_QR (solving LSQ using a QR factorisation)

```
input:  $A \in \mathbb{R}^{m\times n}$ with $m \geq n = \mathrm{rank}(A)$, $b \in \mathbb{R}^m$.
output: $x \in \mathbb{R}^n$ minimiser of $g(x) = \frac{1}{2}\|Ax - b\|_2$.
1: compute a reduced QR factorisation $A = \hat{Q}\hat{R}$
2: compute $y = \hat{Q}^T b \in \mathbb{R}^n$
3: solve $\hat{R}x = y$ with BS
```

The thus computed x satisfies

$$A^T A x = \hat{R}^T \underbrace{\hat{Q}^T \hat{Q}}_{=I} \hat{R} x = \hat{R}^T y = \hat{R}^T \hat{Q}^T b = A^T b$$

which is the normal equation (7.1) and, hence, x indeed solves LSQ.
However, in practice, Gram-Schmidt is not used for computing the (reduced) QR factorisation, mainly for stability reasons (though there are stabilised variants) but methods employ reflections or rotations.

Householder reflections

A QR factorisation may be computed by transforming the matrix to upper triangular form by successively multiplying with appropriate orthogonal matrices as follows:

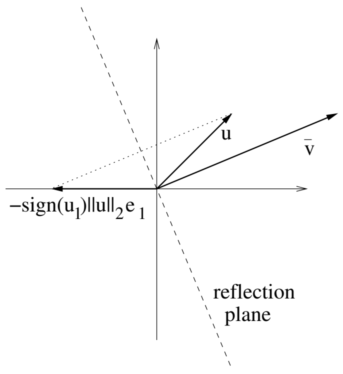
$$A \xrightarrow{Q_1\cdot()} \underbrace{\begin{pmatrix} * & \cdots & \cdots & \cdots & * \\ 0 & * & \cdots & \cdots & * \\ \vdots & \vdots & & & \vdots \\ \vdots & \vdots & & & \vdots \\ \vdots & \vdots & & & \vdots \\ \vdots & \vdots & & & \vdots \\ \vdots & \vdots & & & \vdots \\ 0 & * & \cdots & \cdots & * \end{pmatrix}}_{=:R^{(1)}} \xrightarrow{Q_2\cdot()} \underbrace{\begin{pmatrix} * & \cdots & \cdots & \cdots & * \\ 0 & * & \cdots & \cdots & * \\ \vdots & 0 & * & \cdots & * \\ \vdots & \vdots & \vdots & & \vdots \\ \vdots & \vdots & \vdots & & \vdots \\ \vdots & \vdots & \vdots & & \vdots \\ \vdots & \vdots & \vdots & & \vdots \\ 0 & 0 & * & \cdots & * \end{pmatrix}}_{=:R^{(2)}} \to \cdots \to \underbrace{\begin{pmatrix} * & \cdots & \cdots & \cdots & * \\ 0 & * & \cdots & \cdots & * \\ \vdots & \ddots & \ddots & & \vdots \\ \vdots & & \ddots & \ddots & \vdots \\ \vdots & & & \ddots & * \\ \vdots & & & & 0 \\ \vdots & & & & \vdots \\ 0 & \cdots & \cdots & \cdots & 0 \end{pmatrix}}_{=:R^{(n)}} \tag{17.3}$$

so that $R = Q_n \cdots Q_1 A \Leftrightarrow A = QR$ with $Q = Q_1^T \cdots Q_n^T$. For the orthogonal matrices, reflections may be used. Consider the vector $u^{(k)} := (r_{k,k}^{(k-1)}, \ldots, r_{m,k}^{(k-1)})^T \in \mathbb{R}^{m-k+1}$. In step k, we want to reflect it to $-\mathrm{sign}(u_1^{(k)})\|u^{(k)}\|_2 e_1$ where e_1 denotes the first standard basis vector in $\mathbb{R}^{m-k+1}$. An appropriate reflection matrix is given by

$$H_k := I_{m-k+1} - 2v^{(k)}(v^{(k)})^T \in \mathbb{R}^{(m-k+1)\times(m-k+1)}$$

$$\text{where } v^{(k)} := \frac{\hat{v}^{(k)}}{\|\hat{v}^{(k)}\|_2} \text{ with } \hat{v}^{(k)} := \mathrm{sign}(u_1^{(k)})\|u^{(k)}\|_2 e_1 + u^{(k)}$$

where I_{m-k+1} denotes the identity matrix in $\mathbb{R}^{(m-k+1)\times(m-k+1)}$.



Indeed, using that

$$\begin{aligned}\|\hat{v}^{(k)}\|_2^2 &= (\text{sign}(u_1^{(k)})\|u^{(k)}\|_2 e_1 + u^{(k)})^T(\text{sign}(u_1^{(k)})\|u^{(k)}\|_2 e_1 + u^{(k)})\\ &= \|u^{(k)}\|_2^2 + 2\text{sign}(u_1^{(k)})u_1^{(k)}\|u^{(k)}\|_2 + \|u^{(k)}\|_2^2 = 2(\text{sign}(u_1^{(k)})u_1^{(k)}\|u^{(k)}\|_2 + \|u^{(k)}\|_2^2)\end{aligned}$$

we obtain that

$$\begin{aligned}H_k u^{(k)} &= u^{(k)} - 2\frac{\hat{v}^{(k)}}{\|\hat{v}^{(k)}\|_2}\frac{(\hat{v}^{(k)})^T u^{(k)}}{\|\hat{v}^{(k)}\|_2}\\ &= u^{(k)} - 2\frac{\hat{v}^{(k)}}{\|\hat{v}^{(k)}\|_2^2}(\text{sign}(u_1^{(k)})u_1^{(k)}\|u^{(k)}\|_2 + \|u^{(k)}\|_2^2) = u^{(k)} - \hat{v}^{(k)} = -\text{sign}(u_1^{(k)})\|u^{(k)}\|_2 e_1.\end{aligned}$$

Moreover, $(H_k)^{-1} = (H_k)^T = H_k$ is orthogonal so that

$$Q_k := \begin{pmatrix} I_{k-1} & 0 \\ 0 & H_k \end{pmatrix} \in \mathbb{R}^{m\times m}$$

is orthogonal, too, and this is an appropriate matrix to be used in (17.3).

Algorithm 9 QR_HH (computing a QR factorisation with Householder reflections)

input: $A = (a_{ij})_{i,j=1}^{m,n} \in \mathbb{R}^{m\times n}$ with $m \geq n$ and full rank.

output: $Q \in \mathbb{R}^{m\times m}$ orthogonal, $R \in \mathbb{R}^{m\times n}$ upper triangular with $A = QR$.

```
 1: $R^{(0)} := A$, $Q^{(0)} := I_m$.
 2: for $k = 1$ to $n-1$ (to $n$ if $m > n$) do
 3:     $u^{(k)} := (r_{k,k}^{(k-1)}, \dots, r_{m,k}^{(k-1)})^T \in \mathbb{R}^{m-k+1}$
 4:     $\hat{v}^{(k)} := \text{sign}(u_1^{(k)})\|u^{(k)}\|_2 e_1 + u^{(k)} \in \mathbb{R}^{m-k+1}$
 5:     $v^{(k)} := \hat{v}^{(k)}/\|\hat{v}^{(k)}\|_2$
 6:     $H_k := I_{m-k+1} - 2v^{(k)}(v^{(k)})^T \in \mathbb{R}^{(m-k+1)\times(m-k+1)}$
 7:     $Q_k := \text{diag}(I_{k-1}, H_k) \in \mathbb{R}^{m\times m}$
 8:     $R^{(k)} := Q_k R^{(k-1)}$
 9:     $Q^{(k)} := Q_k Q^{(k-1)}$
10: end for
11: return $Q^{(n-1)}$ and $R^{(n-1)}$ ($Q^{(n)}$ and $R^{(n)}$ if $m > n$)
```

A reduced QR factorisation is obtained by dropping the last $m-n$ columns of Q and the last $m-n$ rows of R (which contain zeros only).

Lecture 18

QR factorisation with Householder reflections - continued

We will have a closer look at algorithm 17 discussing the computational complexity and error analysis. But first, consider a concrete example with the following data:

$$A = \begin{pmatrix} 3 & 3 & 2 \\ 4 & 4 & 1 \\ 0 & 6 & 2 \end{pmatrix} = R^{(0)}.$$

$k = 1:$

$$u^{(1)} = \begin{pmatrix} 3 \\ 4 \\ 0 \end{pmatrix}, \quad \|u^{(1)}\|_2 = 5,$$

$$\hat{v}^{(1)} = \begin{pmatrix} 5 \\ 0 \\ 0 \end{pmatrix} + \begin{pmatrix} 3 \\ 4 \\ 0 \end{pmatrix} = \begin{pmatrix} 8 \\ 4 \\ 0 \end{pmatrix}, \quad \|\hat{v}^{(1)}\|_2 = \sqrt{80},$$

$$v^{(1)} = \frac{1}{\sqrt{80}} \begin{pmatrix} 8 \\ 4 \\ 0 \end{pmatrix}$$

$$Q_1 = H_1 = I_3 - 2v^{(1)}(v^{(1)})^T = I_3 - \frac{1}{40}\begin{pmatrix} 64 & 32 & 0 \\ 32 & 16 & 0 \\ 0 & 0 & 0 \end{pmatrix} = I_3 - \frac{1}{5}\begin{pmatrix} 8 & 4 & 0 \\ 4 & 2 & 0 \\ 0 & 0 & 0 \end{pmatrix},$$

$$R^{(1)} = Q_1 R^{(0)} = \begin{pmatrix} 3 & 3 & 2 \\ 4 & 4 & 1 \\ 0 & 6 & 2 \end{pmatrix} - \frac{1}{5}\begin{pmatrix} 40 & 40 & 20 \\ 20 & 20 & 10 \\ 0 & 0 & 0 \end{pmatrix} = \begin{pmatrix} -5 & -5 & -2 \\ 0 & 0 & -1 \\ 0 & 6 & 2 \end{pmatrix}.$$

$k = 2:$

$$\begin{aligned}
u^{(2)} &= \begin{pmatrix} 0 \\ 6 \end{pmatrix}, \quad \|u^{(2)}\|_2 = 6, \\
\hat{v}^{(2)} &= \begin{pmatrix} 0 \\ 6 \end{pmatrix} + \begin{pmatrix} 6 \\ 0 \end{pmatrix} = \begin{pmatrix} 6 \\ 6 \end{pmatrix}, \quad \|\hat{v}^{(2)}\|_2 = \sqrt{72}, \\
v^{(2)} &= \frac{1}{\sqrt{72}} \begin{pmatrix} 6 \\ 6 \end{pmatrix}, \\
H_2 &= I_2 - \frac{1}{36} \begin{pmatrix} 36 & 36 \\ 36 & 36 \end{pmatrix} = \begin{pmatrix} 0 & -1 \\ -1 & 0 \end{pmatrix}, \\
Q_2 &= \begin{pmatrix} 1 & 0 \\ 0 & H_2 \end{pmatrix} = \begin{pmatrix} 1 & 0 & 0 \\ 0 & 0 & -1 \\ 0 & -1 & 0 \end{pmatrix}, \\
R^{(2)} &= Q_2 R^{(1)} = \begin{pmatrix} -5 & -5 & -2 \\ 0 & -6 & -2 \\ 0 & 0 & 1 \end{pmatrix} = R.
\end{aligned}$$

We end up with

$$R = R^{(2)}, \quad Q = Q_1 Q_2 = \begin{pmatrix} -\frac{3}{5} & 0 & \frac{4}{5} \\ -\frac{4}{5} & 0 & -\frac{3}{5} \\ 0 & -1 & 0 \end{pmatrix}$$

Computational Complexity

Lemma 18.1. [5.11] *The cost for QR_HH except line 9 is*

$$C_{\mathbf{QR\_HH}}(m, n) \sim 2mn^2 - \frac{2}{3}n^3 \quad \text{as } m, n \to \infty.$$

Proof. Let us first consider the cost for one k-step. Computing $\|u^{(k)}\|_2$ involves $m-k$ additions, $m-k+1$ multiplications, and one square root, hence constructing $\hat{v}^{(k)}$ requires $2(m-k+1)+1$ operations (we here assume that obtaining $\operatorname{sign}(u_1)$ and changing the sign does not involve any cost).
Line 8 essentially requires multiplying the lower right $(m-k+1) \times (n-k+1)$ block $\tilde{R}^{(k-1)}$ of $R^{(k-1)}$ with $H_k = I_{m-k+1} - \frac{2}{\|\hat{v}^{(k)}\|_2^2} \hat{v}^{(k)} (\hat{v}^{(k)})^T$ from the left:

- The computation of $(s^{(k)})^T := (\hat{v}^{(k)})^T \tilde{R}^{(k-1)}$ (these are $n-k+1$ standard inner products of vectors of length $m-k+1$) involves $m-k$ additions and $m-k+1$ multiplications for each column of $\tilde{R}^{(k-1)}$, hence $(n-k+1)(2(m-k)+1)$ operations.
- Computing $c^{(k)} := \|\hat{v}^{(k)}\|_2^2/2$ requires $2(m-k+1)$ operations.
- For computing $t^{(k)} := \frac{1}{c^{(k)}} \hat{v}^{(k)}$, $m-k+1$ divisions are needed.
- To get the $(m-k+1) \times (n-k+1)$ matrix $t^{(k)}(s^{(k)})^T (= \frac{2}{\|\hat{v}^{(k)}\|_2^2} \hat{v}^{(k)} (\hat{v}^{(k)})^T \tilde{R}^{(k-1)})$ requires $(m-k+1)(n-k+1)$ multiplications.
- To finalise the computation of $H_k \tilde{R}^{(k-1)} = \tilde{R}^{(k-1)} - \frac{2}{\|\hat{v}^{(k)}\|_2^2} \hat{v}^{(k)} (\hat{v}^{(k)})^T \tilde{R}^{(k-1)}$ we have to perform $(m-k+1)(n-k+1)$ subtractions.

Summing up the costs for each k-step we obtain

$$\begin{aligned}
C_{\mathbf{QR\_HH}}(m,n) &= \sum_{k=1}^{n} \Big(2(m-k+1)+1+(n-k+1)(2(m-k)+1) \\
&\qquad + 3(m-k+1)+2(m-k+1)(n-k+1)\Big) \\
&= \sum_{k=1}^{n} 5(m-k+1)+1+(n-k+1)(4(m-k+1)-1) \\
&= 4\sum_{k=1}^{n}(nm-mk-nk+k^2) + \text{ l.o.t. (lower order terms)} \\
&= 4\Big(mn^2 - m\frac{n(n+1)}{2} - n\frac{n(n+1)}{2} + \frac{1}{6}n(n+1)(2n+1)\Big) + \text{ l.o.t.} \\
&= 2mn^2 - \frac{2}{3}n^3 + \text{ l.o.t.}
\end{aligned}$$

□

Error Analysis

The following result is restated from [5] (Higham, 2002):

Theorem 18.2. *Let x denote the solution to* LSQ *with data (A,b) and $\hat{x}$ the solution computed via* **LSQ_QR**. *Then $\hat{x}$ minimises $\hat{g}(y) = \frac{1}{2}\|(A+\Delta A)y-(b+\Delta b)\|_2^2$ with*

$$\begin{aligned}
\|\Delta a_i\|_2 &\leq \frac{Cmn\varepsilon_m}{1-Cmn\varepsilon_m}\|a_i\|_2, \quad i=1,\dots,n, \\
\|\Delta b\|_2 &\leq \frac{Cmn\varepsilon_m}{1-Cmn\varepsilon_m}\|b\|_2
\end{aligned}$$

where a_i and Δa_i denote the column vectors of A and ΔA, respectively.

In the case $m=n$ we may use the QR factorisation to solve (SLE). Recalling that for **GEPP** we had an error bound of the form

$$\|\Delta A\|_\infty \leq Cn^3 g_n(A)\varepsilon_m\|A\|_\infty$$

we see from the previous theorem that **LSQ_QR** has much better stability properties.
However, the cost for **LSQ_QR** are $C_{\mathbf{LSQ\_QR}}(n) \sim \frac{4}{3}n^3$ as $n\to\infty$ while for **GEPP** we only had $C_{\mathbf{GEPP}}(n) \sim \frac{2}{3}n^3$.
In practice, **GEPP** is preferred to solve (SLE). The exponential growth of the growth factor $g_n(A)$ is only a worst case estimate and, in application, matrices do not behave that badly.

Lecture 19

Linear Iterative Methods for SLE

Iterative methods aim for constructing a sequence $\{x^{(k)}\}_{k\in\mathbb{N}} \subset \mathbb{C}^n$ such that $x^{(k)} \to x$. Within this context, the two (typically competing) criteria are:

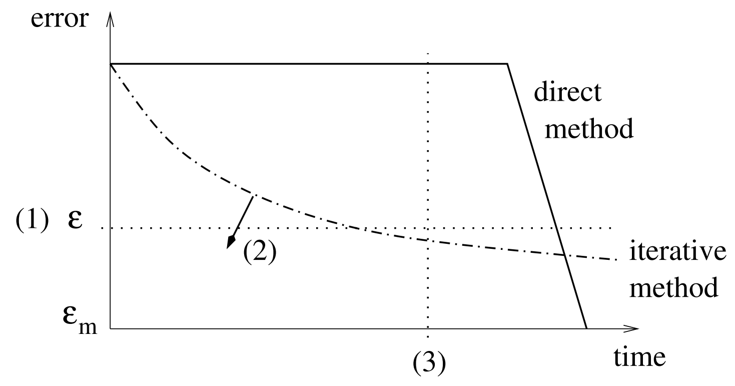
- the computation of $x^{(k)}$ from the data and previous iterates should be <u>inexpensive</u> (relative to a direct SLE solve),
- the convergence to the exact solution x is <u>fast</u>.

Such methods can become advantageous if:

(1) an error $O(\varepsilon)$ with ε much bigger than ε_m is acceptable (e.g. discretisation of differential equation),

(2) the matrix is sparse so that the matrix-vector product, usually needed by iterative methods, is cheap to compute (e.g. banded matrices),

(3) the computational resources are limited since from the actual iterate one may learn at least something about the solution (e.g. real-time control).



The basic idea of linear iterative methods is to split $A = M + N$ and, given some initial guess $x^{(0)} \in \mathbb{C}^n$ for the solution, to solve

$$Mx^{(k)} = b - Nx^{(k-1)}$$

at every step. If $x^{(k)} \to x$ then indeed

$$Mx \leftarrow Mx^{(k)} = -Nx^{(k-1)} + b \to -Nx + b \quad \Rightarrow \quad Ax = b.$$

Assuming convergence we will need a criterion to stop the iteration. Let us define the

$$\underline{\text{error}} \quad e^{(k)} := x - x^{(k)}.$$

It would be best if we could ensure that $\|e^{(k)}\|$ is small enough, say, smaller than a given tolerance. But since x is not available one has to estimate the error. This may be done using the

$$\underline{\text{residual (vector)}} \quad r^{(k)} := b - Ax^{(k)}.$$

Observe that $r^{(k)} = Ae^{(k)}$ from which we deduce that

$$\|e^{(k)}\| = \|A^{-1}r^{(k)}\| \le \|A^{-1}\|\|r^{(k)}\|.$$

Similarly, $\|b\| = \|Ax\| \le \|A\|\|x\|$, and provided that we are not in the trivial case $x = b = 0$ we conclude that

$$\frac{1}{\|x\|} \le \frac{\|A\|}{\|b\|}.$$

Putting both inequalities together we obtain (similarly as Proposition 9.2) that

$$\frac{\|e^{(k)}\|}{\|x\|} \le \frac{\|A\|}{\|b\|}\|A^{-1}\|\|r^{(k)}\| = \kappa(A)\frac{\|r^{(k)}\|}{\|b\|}. \tag{19.1}$$

Error Analysis

Error analysis for iterative methods means analysing the convergence of algorithms (rather than their stability).

Lemma 19.1. [6.1] *Assume that $e^{(k)} = Re^{(k-1)} (= R^k e^{(0)})$ with some matrix $R \in \mathbb{C}^{n\times n}$. Then $e^{(k)} \to 0$ for all $e^{(0)}$ if an only if $\rho(R) < 1$.*

Proof. Assume that $\rho(R) < 1$. For every $\delta > 0$ there is a matrix norm $\|\cdot\|_\delta$ with $\|R\|_\delta \le \rho(R)+\delta$. Choosing δ small enough such that $\rho(R) + \delta < 1$ we obtain for all $e^{(0)}$ that

$$\|e^{(k)}\|_\delta = \|R^k e^{(0)}\|_\delta \le \|R\|_\delta^k \|e^{(0)}\|_\delta \to 0 \quad \text{as } k \to \infty.$$

In turn, if $\rho(R) \ge 1$ then $Re^{(0)} = \lambda e^{(0)}$ for some $e^{(0)} \ne 0$ and $|\lambda| \ge 1$. Hence $\|e^{(k)}\| = |\lambda|^k \|e^{(0)}\|$ does not converge to zero. $\square$

In our case we have $x^{(k)} = -M^{-1}(Nx^{(k-1)} - b)$ and, as a consequence of $Ax = b$, we obtain that $x = -M^{-1}Nx + M^{-1}b$. Subtracting those two equations we see that $e^{(k)} = Re^{(k-1)}$ with $R = -M^{-1}N$. Since $\rho(\cdot) \le \|\cdot\|$ for every matrix norm it is sufficient to show that $\|-M^{-1}N\| < 1$ to ensure convergence.

Remark 19.2. *In the context of iterative methods, rounding errors are not discussed for the following reason. We require that we have stable operations in every iteration step so that the rounding errors are $O(\varepsilon_m)$ in every step. This error should be small in comparison with*

$$\frac{\|e^{(k-1)}\| - \|e^{(k)}\|}{\|e^{(k-1)}\|} \quad \text{or} \quad \frac{\|r^{(k-1)}\| - \|r^{(k)}\|}{\|r^{(k-1)}\|}.$$

If this assumption is not fulfilled then the iteration converges too slow anyway and one should abandon it. A good criterion is to require that in every step

$$\frac{\|r^{(k-1)}\| - \|r^{(k)}\|}{\|r^{(k-1)}\|} \ge \delta \quad \Leftrightarrow \quad \frac{\|r^{(k)}\|}{\|r^{(k-1)}\|} \le 1-\delta \qquad \text{with some } 1 > \delta \gg \varepsilon_m.$$

Lecture 20

The Jacobi Method

Let us split our matrix $A \in \mathbb{C}^{n\times n}$ in the form $A = D + L + U$ where $D = \operatorname{diag}(a_{1,1}, \dots, a_{n,n})$ is the diagonal part and L and U are the lower and upper triangular parts given by

$$l_{i,j} := \begin{cases} a_{i,j} & \text{if } i > j, \\ 0 & \text{else,} \end{cases} \qquad u_{i,j} := \begin{cases} a_{i,j} & \text{if } i < j, \\ 0 & \text{else.} \end{cases}$$

The Jacobi method is the linear iterative method that consists in choosing $M = D$ and $N = L + U$, whence

$$x^{(k)} = D^{-1}\big(b - (L+U)x^{(k-1)}\big).$$

Example: For

$$A = \begin{pmatrix} 2 & -1 & 0 & \cdots & 0 \\ -1 & 2 & \ddots & \ddots & \vdots \\ 0 & \ddots & \ddots & \ddots & 0 \\ \vdots & \ddots & \ddots & \ddots & -1 \\ 0 & \cdots & 0 & -1 & 2 \end{pmatrix}$$

we have

$$D = \begin{pmatrix} 2 & 0 & \cdots & \cdots & 0 \\ 0 & \ddots & \ddots & & \vdots \\ \vdots & \ddots & \ddots & \ddots & \vdots \\ \vdots & & \ddots & \ddots & 0 \\ 0 & \cdots & \cdots & 0 & 2 \end{pmatrix}, \; L = \begin{pmatrix} 0 & \cdots & \cdots & \cdots & 0 \\ -1 & \ddots & & & \vdots \\ 0 & \ddots & \ddots & & \vdots \\ \vdots & \ddots & \ddots & \ddots & \vdots \\ 0 & \cdots & 0 & -1 & 0 \end{pmatrix}, \; U = \begin{pmatrix} 0 & -1 & 0 & \cdots & 0 \\ \vdots & \ddots & \ddots & \ddots & \vdots \\ \vdots & & \ddots & \ddots & 0 \\ \vdots & & & \ddots & -1 \\ 0 & \cdots & \cdots & \cdots & 0 \end{pmatrix}$$

so that

$$x^{(k)} = \begin{pmatrix} \frac{1}{2} & 0 & \cdots & \cdots & 0 \\ 0 & \ddots & \ddots & & \vdots \\ \vdots & \ddots & \ddots & \ddots & \vdots \\ \vdots & & \ddots & \ddots & 0 \\ 0 & \cdots & \cdots & 0 & \frac{1}{2} \end{pmatrix} \left(b - \begin{pmatrix} 0 & -1 & 0 & \cdots & 0 \\ -1 & \ddots & \ddots & \ddots & \vdots \\ 0 & \ddots & \ddots & \ddots & 0 \\ \vdots & \ddots & \ddots & \ddots & -1 \\ 0 & \cdots & 0 & -1 & 0 \end{pmatrix} x^{(k-1)} \right).$$

Writing $x^{(k)} = (x_1^{(k)}, \dots, x_n^{(k)})^T$, this means that

$$\begin{aligned} x_i^{(k)} &= \frac{1}{2}(b_i + x_{i-1}^{(k-1)} + x_{i+1}^{(k-1)}), \quad 2 \le i \le n-1, \\ x_1^{(k)} &= \frac{1}{2}(b_1 + x_2^{(k-1)}), \\ x_n^{(k)} &= \frac{1}{2}(b_n + x_{n-1}^{(k-1)}). \end{aligned}$$

For convergence it is sufficient if the spectral radius of the iteration matrix $R = -M^{-1}N = -D^{-1}(L+U)$ is smaller than one.

Theorem 20.1. *The Jacobi method is convergent if A satisfies*

(1) $|a_{i,i}| > \sum_{j \neq i} |a_{i,j}|$ for all i (strong row sum criterion), **or**

(2) $|a_{j,j}| > \sum_{i \neq j} |a_{i,j}|$ for all j (strong column sum criterion).

Proof. In the first case we have that the entries of R satisfy $r_{i,j} = -a_{i,j}/a_{ii}$ if $i \neq j$ and $r_{i,i} = 0$ for all i. Therefore $\rho(R) \le \|R\|_\infty = \max_i \sum_j |r_{i,j}| = \max_i \frac{1}{|a_{i,i}|} \sum_{j \neq i} |a_{i,j}| < 1$. The other case can be proved similarly. $\square$

One can weaken this criterion, just a little bit so that it becomes much more useful for quite a few applications. We need the notion of irreducibility for this purpose. A matrix $A \in \mathbb{C}^{n \times n}$ is irreducible if there is no permutation matrix P such that

$$P^T A P = \begin{pmatrix} \tilde{A}_{1,1} & \tilde{A}_{1,2} \\ 0 & \tilde{A}_{2,2} \end{pmatrix}$$

where $\tilde{A}_{1,1}$ and $\tilde{A}_{2,2}$ both are square blocks of size $p \times p$ and $(n-p) \times (n-p)$ with some $1 \le p \le n-1$.

Graph of a matrix

It can be very tedious to check this irreducibility criterion but, fortunately, there is another way based on studying the oriented graph $G(A)$ of the matrix A. It consists of the vertices $1, \dots, n$, and there is an (oriented) edge from vertex i to vertex j (denoted by $i \to j$) if $a_{i,j} \neq 0$.

Example:

$$A_1 = \begin{pmatrix} 2 & -1 & 0 \\ 0 & 2 & -1 \\ 0 & 0 & 2 \end{pmatrix}, \quad A_2 = \begin{pmatrix} 2 & -1 & 0 \\ -1 & 2 & -1 \\ 0 & -1 & 2 \end{pmatrix}$$

$$\begin{tikzcd} \boxed{1} \arrow[loop above] \arrow[d, bend left] & \boxed{1} \arrow[loop above] \arrow[d, bend left] \\ \boxed{2} \arrow[loop right] \arrow[d, bend left] & \boxed{2} \arrow[loop right] \arrow[u, bend left] \arrow[d, bend left] \\ \boxed{3} \arrow[loop below] & \boxed{3} \arrow[loop below] \arrow[u, bend left] \end{tikzcd}$$

We say that two vertices i, j are connected if there is a chain of connecting edges (or direct connections) $i = i_0 \to i_1 \to \cdots \to i_k = j$ with some $k \in \mathbb{N}$. The graph $G(A)$ then is called connected if any two vertices i, j of it are connected. Irreducibility of the matrix A now may be checked using the following lemma.

Lemma 20.2. *A is irreducible if and only if $G(A)$ is connected.*

Proof. Exercise. □

Back to the question of convergence of the Jacobi method:

Theorem 20.3. *If A is irreducible and satisfies the weak row sum criterion*

(1) $|a_{i,i}| \geq \sum_{j\neq i} |a_{i,j}|$ for all $i = 1, \dots, n$, **and**

(2) $|a_{k,k}| > \sum_{j\neq k} |a_{k,j}|$ for at least one index $k \in \{1, \dots, n\}$

then the Jacobi method converges.

Proof. Recall that we need to prove that $\rho(R) < 1$. Let $e := (1, \dots, 1)^T \in \mathbb{C}^n$ and $|R| = (|r_{i,j}|)_{i,j}$. Then thanks to the first condition

$$0 \leq \big(|R|e\big)_i = \sum_{j=1}^{n} |r_{i,j}| = \sum_{j\neq i} \frac{|a_{i,j}|}{|a_{i,i}|} \leq 1 = e_i$$

so that $e \geq |R|e \geq |R|^2 e \geq \dots$ where the inequality for vectors here and in the following has to be understood component-wise.
Let $t^{(l)} := e - |R|^l e \geq 0$, $l \in \mathbb{N}$. Assume now that there is a positive number of non-vanishing components of $t^{(l)}$ that become stationary. We may assume that these are the first m entries where $m > 0$ thanks to the second condition, i.e.,

$$t^{(l)} = \begin{pmatrix} b^{(l)} \\ 0 \end{pmatrix}, \quad t^{(l+1)} = \begin{pmatrix} b^{(l+1)} \\ 0 \end{pmatrix}$$

where $b^{(l)}, b^{(l+1)} \in \mathbb{R}^m$ have positive entries, $b^{(l)} > 0, b^{(l+1)} > 0$.
Suppose that $m < n$. Then

$$\begin{pmatrix} b^{(l+1)} \\ 0 \end{pmatrix} = e - |R|^{l+1}e \geq |R|e - |R|^{l+1}e = |R|(e - |R|^l e) = |R| \begin{pmatrix} b^{(l)} \\ 0 \end{pmatrix} = \begin{pmatrix} |R_{1,1}| & |R_{1,2}| \\ |R_{2,1}| & |R_{2,2}| \end{pmatrix} \begin{pmatrix} b^{(l)} \\ 0 \end{pmatrix}$$

with $R_{1,1} \in \mathbb{R}^{m\times m}$ and the other blocks accordingly. Since $b^{(l)} > 0$ necessarily $|R_{2,1}| = 0$. Therefore R is not irreducible. And since $r_{i,j} = a_{i,j}/a_{i,i}$ if $i \neq j$ we obtain that A is not irreducible in contradiction to the assumption. Hence $m = n$.
Conwequently, $t^{(l)} > 0$ as long as l is big enough (using the above contradiction argument again we see that $l > n$ is sufficient). This means that $e > |R|^l e$, whence

$$\rho(R)^l \leq \rho(R^l) \leq \|R^l\|_\infty \leq \||R|^l\|_\infty = \max_i \big(|R|^l e\big)_i < \max_i e_i = 1$$

so that $\rho(R) < 1$ as desired. □

Lecture 21

Computational Complexity of Linear Iterative Methods

In some applications, the goal will be to decrease the relative forward error $\|e^{(k)}\|/\|x\|$ below a given threshold while in others it is sufficient to decrease the relative backward error $\|r^{(k)}\|/\|b\|$. We will concentrate on the latter goal but recall that by (19.1) the two goals are related. Of course, the knowledge of the condition number is required to deduce an estimate for the forward error from the backward error.
So our goal is

$$\|r^{(k)}\| \leq \varepsilon_r \|b\| \tag{21.1}$$

where $\varepsilon_r > 0$ is a given tolerance. Using $r^{(k)} = Ae^{(k)} = AR^k e^{(0)}$ it is sufficient to achieve that

$$\|A\|\|R\|^k \|e^{(0)}\| \leq \varepsilon_r \|b\| \quad \Leftrightarrow \quad \Big(\underbrace{\frac{1}{\|R\|}}_{>1}\Big)^k \geq \frac{\|A\|\|e^{(0)}\|}{\|b\|\varepsilon_r}$$

which is the case if an only if

$$k \geq \frac{\log(\|A\|) + \log(\|e^{(0)}\|) - \log(\|b\|) - \log(\varepsilon_r)}{\log(\|R\|^{-1})} =: k^\sharp(n, \varepsilon_r). \tag{21.2}$$

In practice, the iteration matrix R often depends on n in a very unfavourable way while the other data A, b and $e^{(0)}$ do not affect the number of steps that much.

Assumption 21.1.

1. *The calculation of Rx involves a cost of $\Theta(n^\alpha)$ as $n \to \infty$ with some $\alpha > 0$.*
2. *$\|R\| = 1 - h(n)$ with a positive function h such that $h(n) = \Theta(n^{-\beta})$ as $n \to \infty$ with some $\beta > 0$.*
3. *$\|A\|$, $\|b\|$, and $\|e^{(0)}\|$ are uniformly bounded in n.*

Theorem 21.2. *Under Assumption 21.1, the computational cost to achieve (21.1) is bounded by a function $C(n, \varepsilon_r)$ satisfying*

$$C(n, \varepsilon_r) = \Theta(n^{\alpha+\beta} \log(\varepsilon_r^{-1})) \quad \textit{as } (n, \varepsilon_r) \to (\infty, 0).$$

Proof. Recall that $\log(1/(1-x)) = x + x^2/2 + x^3/3 + \dots$. Thus, by Assumption 21.1.2

$$\log(\|R\|^{-1}) = \log(1/(1-h(n))) = h(n) + \text{ higher order terms } = \Theta(n^{-\beta})$$

so that $(\log(\|R\|^{-1}))^{-1} = \Theta(n^{\beta})$ as $n \to \infty$. From (21.2) and Assumption 21.1.3 we get

$$k^{\sharp}(n, \varepsilon_r) = \Theta(n^{\beta} \log(\varepsilon_r^{-1})) \quad \text{as } (n, \varepsilon_r) \to (\infty, 0)$$

for the number of steps. Taking the cost per step into account (Assumption 21.1.1), the total cost is

$$k\sharp(n, \varepsilon_r) C_{\text{one\_step}}(n) = \Theta(n^{\alpha+\beta} \log(\varepsilon_r^{-1})).$$

$\square$

Assuming a polynomial dependence in Assumption 21.1.3 one would obtain additional $\log(n)$ terms in the cost estimate.

Computational complexity of the Jacobi method

In each iteration step:

- computing $(L+U)x^{(k-1)}$ involves at most $O(n^2)$ operations,
- computing $b - \dots$ is $O(n)$,
- computing $D^{-1}(\dots)$ is $O(n)$, too.

So the essential cost is coming from the first step. If the matrix is sparse, i.e., the number of non-vanishing entries in each row is $\sim n^{\eta}$ with some $\eta < 1$ then the number of operations is $O(n^{1+\eta})$. For instance, $\eta = 0$ if A is tridiagonal as the number of non-vanishing entries in each row is bounded by a constant (namely 3). In any case, with $\alpha \in [1, 2]$ the general result on the computational cost in Theorem 21.2 is applicable.

Variants of the Jacobi method

The successive over relaxation (SOR) method generalises the Jacobi method by setting

$$M := L + \omega D, \quad N := U + (1-\omega)D, \qquad \omega \in \mathbb{R}.$$

For $\omega = 1$ we obtain the Gauss-Seidel method.
Results for convergence criteria and computations cost read similarly. Yet the relaxation parameter ω can have a massive influence on the spectral properties of the iteration matrix R and speed up the convergence a lot. The notion *over-relaxation* refers to choosing $\omega > 1$, hence bigger than in the Gauss-Seidel method.

Lecture 22

Nonlinear Iterative Methods, Steepest Decent

We restrict our attention now to positive definite (symmetric) matrices $A \in \mathbb{R}^{n\times n}$.
Recall the notation $\langle x, y\rangle_A := \langle x, Ay\rangle$ and $\|x\|_A := \sqrt{\langle x, x\rangle_A}$.
Clearly, $x \in \mathbb{R}^n$ solves $Ax = b$ if and only if x is minimiser of

$$g : \mathbb{R}^n \to \mathbb{R}, \quad g(y) = \frac{1}{2}\|Ay - b\|^2_{A^{-1}}. \tag{22.1}$$

Recalling that $e^{(k)} = x - x^{(k)}$, $r^{(k)} = b - Ax^{(k)} = Ae^{(k)}$ one can easily show that

$$g(x^{(k)}) = \frac{1}{2}\|r^{(k)}\|^2_{A^{-1}} = \frac{1}{2}\|e^{(k)}\|^2_A. \tag{22.2}$$

Hence, minimising g means

- minimising the residual in the $\|\cdot\|_{A^{-1}}$ norm or
- minimising the error in the $\|\cdot\|_A$ norm (energy norm).

We consider nonlinear iterative methods for solving SLEs that have the form

$$x^{(k)} = x^{(k-1)} + \alpha^{(k-1)} d^{(k-1)} \tag{22.3}$$

where $d^{(k-1)} \in \mathbb{R}^n$ is the search direction and $\alpha^{(k-1)} \in \mathbb{R}$ is the step length. The step length is chosen such that

$$f(\alpha) := g(x^{(k-1)} + \alpha d^{(k-1)})$$

is minimal. This uniquely determines $\alpha^{(k-1)}$ since f is convex and tends to infinity as $|\alpha| \to \infty$. This also allows to derive an explicit formula for the step length:

$$\begin{aligned}
f(\alpha) &= \frac{1}{2}\langle \alpha A d^{(k-1)} + \underbrace{Ax^{(k-1)} - b}_{-r^{(k-1)}}, A^{-1}(\alpha A d^{(k-1)} + \underbrace{Ax^{(k-1)} - b}_{-r^{(k-1)}})\rangle \\
&= \frac{1}{2}\alpha^2 \langle Ad^{(k-1)}, d^{(k-1)}\rangle + \frac{1}{2}\alpha\langle Ad^{(k-1)}, -A^{-1}r^{(k-1)}\rangle \\
&\quad + \frac{1}{2}\alpha\langle -r^{(k-1)}, Ad^{(k-1)}\rangle + \frac{1}{2}\langle -r^{(k-1)}, -A^{-1}r^{(k-1)}\rangle \\
&= \frac{1}{2}\alpha^2\|d^{(k-1)}\|^2_A - \alpha\langle d^{(k-1)}, r^{(k-1)}\rangle + \frac{1}{2}\|r^{(k-1)}\|^2_{A^{-1}} \qquad (22.4) \\
\Rightarrow \quad f'(\alpha) &= \alpha\|d^{(k-1)}\|^2_A - \langle d^{(k-1)}, r^{(k-1)}\rangle
\end{aligned}$$

As the minimiser fulfils $f'(\alpha^{(k-1)}) = 0$ we obtain

$$\alpha^{(k-1)} = \frac{\langle r^{(k-1)}, d^{(k-1)} \rangle}{\|d^{(k-1)}\|_A^2}. \tag{22.5}$$

Before we start looking at possible search direction we make two observations:

1. The residual is subject to the iterative formula
$$r^{(k)} = b - Ax^{(k)} = b - Ax^{(k-1)} - \alpha^{(k-1)} Ad^{(k-1)} = r^{(k-1)} - \alpha^{(k-1)} Ad^{(k-1)}. \tag{22.6}$$

2. A computation similar to the above on shows that
$$\partial_{x_i} g(x^{(k-1)}) = \frac{d}{d\alpha} g(x^{(k-1)} + \alpha e_i)\Big|_{\alpha=0} = -\langle e_i, r^{(k-1)} \rangle$$
from which we conclude that $\nabla g(x^{(k-1)}) = -r^{(k-1)}$.

Steepest Decent Method

The idea of this method is to choose $d^{(k-1)} = r^{(k-1)} = -\nabla g(x^{(k-1)})$. This choice is motivated from the fact that the gradient points in the direction of the fastest increase, hence a sufficiently small step in direction $-\nabla g(x^{(k-1)})$ will decrease the value of our target function g that is to be minimised. According to (22.5) the optimal step length then is $\alpha^{(k-1)} = \|r^{(k-1)}\|_2^2 / \|r^{(k-1)}\|_A^2$.

Algorithm 10 SD (steepest decent method)

```
input:  $A = (a_{ij})_{i,j=1}^n \in \mathbb{R}^{n\times n}$ positive definite, $b, x^{(0)} \in \mathbb{R}^n$, $\varepsilon_r > 0$.
output: $x \in \mathbb{R}^n$ with $\|Ax - b\|_2 \le \varepsilon_r$.
1: for $k = 1, 2, \dots$ do
2:   $r^{(k-1)} := b - Ax^{(k-1)}$
3:   if $\|r^{(k-1)}\|_2 \le \varepsilon_r$ then
4:     return $x^{(k-1)}$
5:   else
6:     $\alpha^{(k-1)} := \|r^{(k-1)}\|_2^2 / \|r^{(k-1)}\|_A^2$
7:     $x^{(k)} := x^{(k-1)} + \alpha^{(k-1)} r^{(k-1)}$
8:   end if
9: end for
```

Introducing a help variable $h^{(k-1)} = Ar^{(k-1)}$ it is possible to formulate the algorithm such that only one matrix-vector multiplication per iteration step is required (exercise).

One observes that subsequent search directions are orthogonal with respect to the standard scalar product: Thanks to (22.6) and $d^{(k-1)} = r^{(k-1)}$

$$\begin{aligned}\langle r^{(k-1)}, r^{(k)} \rangle &= \langle r^{(k-1)}, r^{(k-1)} - \alpha^{(k-1)} Ar^{(k-1)} \rangle \\ &= \langle r^{(k-1)}, r^{(k-1)} \rangle - \frac{\|r^{(k-1)}\|_2^2}{\|r^{(k-2)}\|_A^2} \langle r^{(k-1)}, Ar^{(k-1)} \rangle = 0.\end{aligned}$$

The effect is a zig-zag path of the iterates when approaching the minimum of g as illustrated in Figure 22.1. Since g is a quadratic function the level sets of g are ellipsoids. As an observation, the longer the ellipsoids are stretched, the longer it takes for the algorithms to obtain an

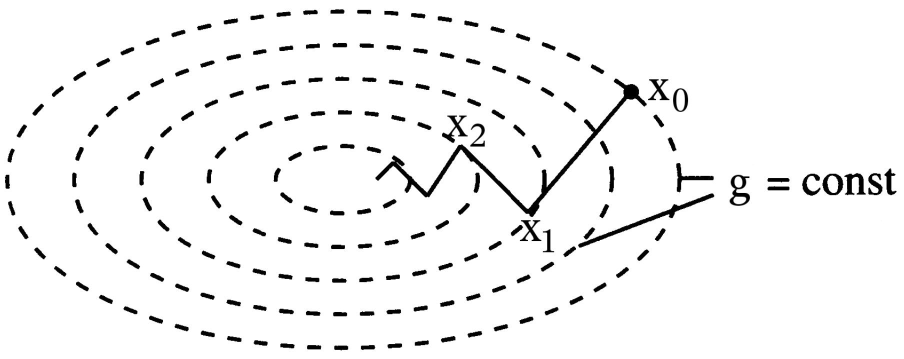


Figure 22.1: Behaviour of **SD**, zigzag path to the minimum due to orthogonal (w.r.t. the Euclidian scalar product) search directions.

iterate close to the minimum. It turns out that the main axes of the level set ellipsoids are the eigenspaces, and the stretching of the ellipsoids depends on the ratio of the eigenvalues, most prominently $\lambda_{\max}/\lambda_{\min}$. Recalling that for positive definite matrices $\|A\|_2 = \lambda_{\max}$ and $\|A^{-1}\|_2 = 1/\lambda_{\min}$ we see that it is exactly the condition number $\kappa_2(A) = \|A\|_2\|A^{-1}\|_2 = \lambda_{\max}/\lambda_{\min}$ that can serve as a measure how elongated the level sets are. We will see later on how the condition number influences the convergence of **SD**.

Lecture 23

Conjugate Gradient Method

As in the previous lecture, $A \in \mathbb{R}^{n\times n}$ is positive definite throughout.
In SD, subsequent search directions were orthogonal with respect to the standard scalar product. Now, we consider $\underline{A\text{-orthogonal}}$ (or $\underline{\text{conjugate}}$) search directions $d^{(k)}$, i.e., $\langle d^{(i)}, d^{(j)}\rangle_A = 0$ if $i \neq j$. This has the following advantage:

Lemma 23.1. *Assume that the search directions $d^{(0)}, \dots, d^{(l-1)}$ in the iteration (22.3) form an A-orthogonal set. Then $x^{(l)}$ minimises g over the set $x^{(0)} + \operatorname{span}\{d^{(0)}, \dots, d^{(l-1)}\}$.*

Proof. Consider the map

$$h : \mathbb{R}^l \to \mathbb{R}, \quad h(\gamma_0, \dots, \gamma_{l-1}) = g\Big(x^{(0)} + \sum_{i=0}^{l-1} \gamma_i d^{(i)}\Big).$$

Since h is convex and tends to infinity as $\|\gamma_0, \dots, \gamma_{l-1}\| \to \infty$ it has a unique minimum $\hat{\gamma}$. Recalling that $\nabla g(x) = Ax - b = -r$ and using the A-orthogonality of the search directions we obtain for all $m = 0, \dots, l-1$ that

$$0 = \frac{\partial}{\partial \gamma_m} h(\hat{\gamma}) = \Big\langle \nabla g\Big(x^{(0)} + \sum_{i=0}^{l-1} \hat{\gamma}_i d^{(i)}\Big), d^{(m)}\Big\rangle$$
$$= \Big\langle Ax^{(0)} - b + \sum_{i=0}^{l-1} \hat{\gamma}_i A d^{(i)}, d^{(m)}\Big\rangle = -\langle r^{(0)}, d^{(m)}\rangle + \sum_{i=0}^{l-1} \hat{\gamma}_i \underbrace{\langle Ad^{(i)}, d^{(m)}\rangle}_{=1 \text{ if } i=m,\ =0 \text{ else}}$$

so that $\hat{\gamma}_m = \langle r^{(0)}, d^{(m)}\rangle / \|d^{(m)}\|_A^2$.
On the other hand, the optimal step length is $\alpha^{(m)} = \langle d^{(m)}, r^{(m)}\rangle / \|d^{(m)}\|_A^2$. But using the iterative formula for the residual (22.6)

$$\langle d^{(m)}, r^{(m)}\rangle = \langle d^{(m)}, r^{(m-1)}\rangle - \alpha^{(m-1)} \underbrace{\langle d^{(m)}, Ad^{(m-1)}\rangle}_{=0} = \dots = \langle d^{(m)}, r^{(0)}\rangle$$

whence $\hat{\gamma}_m = \alpha^{(m)}$ so that $(\alpha^{(0)}, \dots, \alpha^{(l-1)})$ is the minimiser of h.
Consequently, $x^{(l)} = x^{(0)} + \sum_{i=0}^{l-1} \alpha^{(i)} d^{(i)}$ computed by the nonlinear iteration is the minimum of g on $x^{(0)} + \operatorname{span}\{d^{(0)}, \dots, d^{(l-1)}\}$ as asserted. $\square$

For $l = n$ and on assuming that $d^{(k)} \neq 0$ for all $k = 0, \dots, n-1$ we have that $x^{(0)} + \operatorname{span}\{d^{(0)}, \dots, d^{(n-1)}\} = \mathbb{R}^n$, so the above lemma then means that $x^{(n)}$ is the global minimiser

of g and the desired solution to $Ax = b$. Going back to the case $n = 2$ and Figure 22.1, choosing $d^{(0)} = r^{(0)}$ as first search direction this means that the second search direction would ensure jumping from $x^{(1)}$ immediately to the minimum and we would avoid the zigzag path. So the big questions is: How can we obtain A-orthogonal search directions?
More precisely, given $d^{(0)}, \dots, d^{(k-1)}$ (and the $x^{(i)}$ and $r^{(i)}$), how can an appropriate $d^{(k)}$ A-orthogonal to all the previous search directions be obtained? The following ideas go back to Hestenes and Stiefel:

1. Given any $v \notin \operatorname{span}\{d^{(0)}, \dots, d^{(k-1)}\}$, such a vector can be computed via the Gram-Schmidt orthogonalisation method (applied with the A-scalar product, of course):

$$\tilde{d}^{(k)}(v) := v - \frac{\langle d^{(k-1)}, v\rangle_A}{\|d^{(k-1)}\|_A^2} d^{(k-1)} - \frac{\langle d^{(k-2)}, v\rangle_A}{\|d^{(k-2)}\|_A^2} d^{(k-2)} - \dots - \frac{\langle d^{(0)}, v\rangle_A}{\|d^{(0)}\|_A^2} d^{(0)}.$$

   Apart from stability issues, this becomes very expensive when k becomes big.

2. The choice $v = r^{(k)} = -\nabla g(x^{(k)})$ is quite reasonable: If $r^{(k)} \in \operatorname{span}\{d^{(0)}, \dots, d^{(k-1)}\}$ then $r^{(k)} = 0$ because $x^{(k)} = x$ already is the minimum by the preceding result in Lemma 23.1 and we would stop the iteration anyway. Moreover, since $r^{(k)}$ points in the direction of the steepest decent it gives a good idea into which direction roughly to proceed next. So set $d^{(k)} := \tilde{d}^{(k)}(r^{(k)})$.

3. Set $d^{(0)} = r^{(0)}$. Consequently, the first step of the iteration is the same as for **SD**.

The most amazing point with the above choices is that $\langle d^{(i)}, r^{(k)}\rangle_A = 0$ for $i = 0, \dots, k-2$ as we shall prove later on. This means that

$$d^{(k)} = r^{(k)} \underbrace{- \frac{\langle d^{(k-1)}, r^{(k)}\rangle_A}{\|d^{(k-1)}\|_A^2}}_{=:\beta^{(k)}} d^{(k-1)} - \sum_{i=0}^{k-2} \underbrace{\frac{\langle d^{(i)}, r^{(k)}\rangle_A}{\|d^{(k-2)}\|_A^2}}_{=0} d^{(i)} = r^{(k)} + \beta^{(k)} d^{(k-1)},$$

hence *the update of the search direction in fact is cheap!*
In the algorithm below the scalars $\alpha^{(k-1)}$ and $\beta^{(k)}$ are computed with slightly different formulas that will turn out to be algebraically equivalent (see Lemma 24.1) but in practice turned out to perform somewhat better.

Algorithm 11 CG (conjugate gradient method)

```
input: $A = (a_{ij})_{i,j=1}^n \in \mathbb{R}^{n\times n}$ positive definite, $b, x^{(0)} \in \mathbb{R}^n$, $\varepsilon_r > 0$.
output: $x \in \mathbb{R}^n$ with $\|Ax - b\|_2 \leq \varepsilon_r$.
 1: $d^{(0)} = r^{(0)} = b - Ax^{(0)}$
 2: if $\|r^{(0)}\|_2 \leq \varepsilon_r$ then
 3:     return $x^{(0)}$
 4: else
 5:     for $k = 1, 2, \dots$ do
 6:         $\alpha^{(k-1)} := \|r^{(k-1)}\|_2^2 / \|d^{(k-1)}\|_A^2$
 7:         $x^{(k)} := x^{(k-1)} + \alpha^{(k-1)} d^{(k-1)}$
 8:         $r^{(k)} := r^{(k-1)} - \alpha^{(k-1)} A d^{(k-1)}$
 9:         if $\|r^{(k)}\|_2 \leq \varepsilon_r$ then
10:             return $x^{(k)}$
11:         end if
12:         $\beta^{(k)} := \|r^{(k)}\|_2^2 / \|r^{(k-1)}\|_2^2$
13:         $d^{(k)} := r^{(k)} + \beta^{(k)} d^{(k-1)}$
14:     end for
15: end if
```

It is possible to formulate the algorithm such that only one matrix-vector multiplication per iteration step is required (exercise).
Example: Consider the data

$$A = \begin{pmatrix} 2 & -1 \\ -1 & 2 \end{pmatrix}, \quad b = \begin{pmatrix} 1 \\ 0 \end{pmatrix}, \quad x^{(0)} = \begin{pmatrix} 0 \\ 0 \end{pmatrix}.$$

We then have $d^{(0)} = r^{(0)} = b$.
Step $k = 1$:

$$\begin{aligned}
h^{(0)} &:= Ad^{(0)} = \begin{pmatrix} 2 \\ -1 \end{pmatrix} \\
\alpha^{(0)} &:= \frac{\|r^{(0)}\|_2^2}{\|d^{(0)}\|_A^2} = \frac{1}{\langle d^{(0)}, h^{(0)} \rangle} = \frac{1}{2} \\
x^{(1)} &:= x^{(0)} + \alpha^{(0)} d^{(0)} = \begin{pmatrix} 1/2 \\ 0 \end{pmatrix} \\
r^{(1)} &:= r^{(0)} - \alpha^{(0)} h^{(0)} = \begin{pmatrix} 1 \\ 0 \end{pmatrix} - \frac{1}{2}\begin{pmatrix} 2 \\ -1 \end{pmatrix} = \begin{pmatrix} 0 \\ 1/2 \end{pmatrix} \\
\beta^{(1)} &:= \frac{\|r^{(1)}\|_2^2}{\|r^{(0)}\|_2^2} = \frac{1}{4} \\
d^{(1)} &:= r^{(1)} + \beta^{(1)} d^{(0)} = \begin{pmatrix} 0 \\ 1/2 \end{pmatrix} + \frac{1}{4}\begin{pmatrix} 1 \\ 0 \end{pmatrix} = \begin{pmatrix} 1/4 \\ 1/2 \end{pmatrix}
\end{aligned}$$

Step $k = 2$:

$$
\begin{aligned}
h^{(1)} &:= Ad^{(1)} = \begin{pmatrix} 0 \\ 3/4 \end{pmatrix} \\
\alpha^{(1)} &:= \frac{\|r^{(1)}\|_2^2}{\langle d^{(1)}, h^{(1)} \rangle} = \frac{1/4}{3/8} = \frac{2}{3} \\
x^{(2)} &:= x^{(1)} + \alpha^{(1)} d^{(1)} = \begin{pmatrix} 1/2 \\ 0 \end{pmatrix} + \frac{2}{3} \begin{pmatrix} 1/4 \\ 1/2 \end{pmatrix} = \frac{1}{3} \begin{pmatrix} 2 \\ 1 \end{pmatrix} \\
r^{(2)} &:= r^{(1)} - \alpha^{(1)} h^{(1)} = \begin{pmatrix} 0 \\ 1/2 \end{pmatrix} - \frac{2}{3} \begin{pmatrix} 0 \\ 3/4 \end{pmatrix} = \begin{pmatrix} 0 \\ 0 \end{pmatrix}
\end{aligned}
$$

Lecture 24

More on CG

We have motivated to use conjugate or A-orthogonal search directions in order to improve the convergence in comparison with the straight forward steepest decent method. An open issue has been the claim that the update of the search direction in each step, which is based on a Gram-Schmidt orthogonalisation, is cheap as most of the terms drop out.

Lemma 24.1. *Let $x^{(1)},\dots,x^{(k)}$ be the iterates computed by* **CG** *and assume that $r^{(0)},\dots,r^{(k)},d^{(0)},\dots,d^{(k-1)}\neq 0$. Then*

(1) $\|d^{(k)}\|_2 \geq \|r^{(k)}\|_2 > 0$,

(2) $\langle d^{(k-1)}, r^{(k)}\rangle = 0$,

(3) $\langle r^{(k-1)}, r^{(k)}\rangle = 0$,

(4) $\alpha^{(k-1)} = \|r^{(k-1)}\|_2^2/\|d^{(k-1)}\|_A^2 > 0$,

(5) $\beta^{(k)} = \|r^{(k)}\|_2^2/\|r^{(k-1)}\|_2^2 > 0$.

Proof. Using the update formulas $r^{(k)} = r^{(k-1)} - \alpha^{(k-1)}Ad^{(k-1)}$ and the one for $\alpha^{(k-1)}$ we obtain that

$$\langle d^{(k-1)}, r^{(k)}\rangle = \langle d^{(k-1)}, r^{(k-1)}\rangle - \frac{\langle d^{(k-1)}, r^{(k-1)}\rangle}{\|d^{(k-1)}\|_A^2}\langle d^{(k-1)}, Ad^{(k-1)}\rangle = 0 \tag{24.1}$$

which proves (2). A consequence of this Euclidean orthogonality is that

$$\|d^{(k)}\|_2^2 = \|r^{(k)} + \beta^{(k)}d^{(k-1)}\|_2^2 = \|r^{(k)}\|_2^2 + |\beta^{(k)}|^2\|d^{(k-1)}\|_2^2 \geq \|r^{(k)}\|_2^2 > 0$$

which is assertion (1). A further consequence of (24.1) is that for $k>1$

$$\langle d^{(k-1)}, r^{(k-1)}\rangle = \langle r^{(k-1)} + \beta^{(k-1)}d^{(k-2)}, r^{(k-1)}\rangle = \|r^{(k-1)}\|_2^2,$$

and thanks to the choice $d^{(0)} = r^{(0)}$ this is also true for $k=1$. This implies assertion (4),

$$\alpha^{(k-1)} = \frac{\langle d^{k-1)}, r^{(k-1)}\rangle}{\|d^{(k-1)}\|_A^2} = \frac{\|r^{(k-1)}\|_2^2}{\|d^{(k-1)}\|_A^2} > 0.$$

To show (3) we first observe that for $k>1$ thanks to the A orthogonality of the $d^{(i)}$

$$\langle r^{(k-1)}, d^{(k-1)}\rangle_A = \langle d^{(k-1)}, d^{(k-1)}\rangle_A - \beta^{(k)}\underbrace{\langle d^{(k-2)}, d^{(k-1)}\rangle_A}_{=0} = \|d^{(k-1)}\|_A^2,$$

and by the choice $d^{(0)} = r^{(0)}$ this is also true for the case $k = 1$. Therefore, using the already proved identity (4)

$$\begin{aligned}\langle r^{(k-1)}, r^{(k)}\rangle &= \langle r^{(k-1)}, r^{(k-1)}\rangle - \alpha^{(k-1)}\langle r^{(k-1)}, Ad^{(k-1)}\rangle \\ &= \langle r^{(k-1)}, r^{(k-1)}\rangle - \frac{\|r^{(k-1)}\|_2^2}{\|d^{(k-1)}\|_A^2}\|d^{(k-1)}\|_A^2 = 0.\end{aligned}$$

Finally we prove the update formula (5). Since $\alpha^{(k-1)} > 0$ we can write $-Ad^{(k-1)} = \frac{1}{\alpha^{(k-1)}}(r^{(k)} - r^{(k-1)})$, Using this and the already shown identities (3) and (4) we get

$$\beta^{(k)} = -\frac{\langle d^{(k-1)}, r^{(k)}\rangle_A}{\|d^{(k-1)}\|_A^2} = \frac{\langle r^{(k)} - r^{(k-1)}, r^{(k)}\rangle}{\alpha^{(k-1)}\|d^{(k-1)}\|_A^2} = \frac{\|r^{(k)}\|_2^2}{\|r^{(k-1)}\|_2^2} > 0.$$

□

The following lemma is central to **CG**:

Lemma 24.2. *The vectors $d^{(0)}, \ldots, d^{(k)}$ are A-orthogonal. Moreover*

$$\operatorname{span}\{r^{(0)}, \ldots, r^{(l)}\} = \operatorname{span}\{d^{(0)}, \ldots, d^{(l)}\} = \operatorname{span}\{r^{(0)}, Ar^{(0)}, \ldots, A^l r^{(0)}\} \tag{24.2}$$

for $l = 0, \ldots, k-1$.

Proof. We start with proving the second assertion by induction where the case $l = 0$ is clear thanks to the choice $d^{(0)} = r^{(0)}$. So let $l > 0$ and assume that (24.2) is true for $l - 1$.
Since $\alpha^{(l-1)} \neq 0$ we have that $Ad^{(l-1)} = \frac{1}{\alpha^{(l-1)}}(r^{(l-1)} - r^{(l)}) \in \operatorname{span}\{r^{(l-1)}, r^{(l)}\}$. Therefore, using the induction hypothesis,

$$A^l r^{(0)} = A(A^{l-1}r^{(0)}) \in \operatorname{span}\{Ad^{(0)}, \ldots, Ad^{(l-1)}\} \subseteq \operatorname{span}\{r^{(0)}, \ldots, r^{(l)}\}$$

which implies that

$$\operatorname{span}\{r^{(0)}, Ar^{(0)}, \ldots, A^l r^{(0)}\} \subseteq \operatorname{span}\{r^{(0)}, \ldots, r^{(l)}\}.$$

Now, $r^{(l)} = d^{(l)} - \beta^{(l)}d^{(l-1)} \in \operatorname{span}\{d^{(l-1)}, d^{(l)}\}$ which, with the induction hypothesis, yields that

$$\operatorname{span}\{r^{(0)}, \ldots, r^{(l)}\} \subseteq \operatorname{span}\{d^{(0)}, \ldots, d^{(l)}\}.$$

Finally, since $Ad^{(l-1)} \in \operatorname{span}\{r^{(0)}, \ldots, A^l r^{(0)}\}$ and using the induction hypothesis again,

$$d^{(l)} = r^{(l)} + \beta^{(l)}d^{(l-1)} = r^{(l-1)} - \alpha^{(l-1)}Ad^{(l-1)} + \beta^{(l)}d^{(l-1)} \in \operatorname{span}\{r^{(0)}, \ldots, A^l r^{(0)}\}$$

so that

$$\operatorname{span}\{d^{(0)}, \ldots, d^{(l)}\} \subseteq \operatorname{span}\{r^{(0)}, \ldots, A^l r^{(0)}\}.$$

Let us now come to the assertion on the A-orthogonality. We show this by induction, too, where the case $k = 1$ trivially is fulfilled. So let $k > 1$ and assume that $d^{(0)}, \ldots, d^{(k-1)}$ are A-orthogonal.
Consider an index $i < k - 1$. Then for all $j = i + 2, \ldots, k$

$$\langle d^{(i)}, r^{(j)}\rangle = \langle d^{(i)}, r^{(j-1)}\rangle - \alpha^{(j-1)}\underbrace{\langle d^{(i)}, Ad^{(j)}\rangle}_{=0} = \langle d^{(i)}, r^{(j-1)}\rangle$$

where the second term vanishes thanks to the induction hypothesis. But since $\langle d^{(i)}, r^{(i+1)}\rangle = 0$ by lemma 24.1 we conclude that

$$\langle d^{(i)}, r^{(j)}\rangle = 0, \quad j = i+1, \dots, k \tag{24.3}$$

where $i = 0, \dots, k-2$.
Let $l < k-1$. Then thanks to (24.2), $Ad^{(l)} \in \operatorname{span}\{r^{(l)}, r^{(l+1)}\} \subset \operatorname{span}\{d^{(0)}, \dots, d^{(l+1)}\}$, so that, using (24.3), $\langle d^{(l)}, r^{(k)}\rangle_A = \langle Ad^{(l)}, r^{(k)}\rangle = 0$. Consequently,

$$\langle d^{(l)}, d^{(k)}\rangle_A = \underbrace{\langle d^{(l)}, r^{(k)}\rangle_A}_{=0} + \beta^{(k)} \underbrace{\langle d^{(l)}, d^{(k-1)}\rangle_A}_{=0} = 0,$$

where we used the induction hypothesis for the second term.
In the only remaining case $l = k-1$ it is the choice of $\beta^{(k)}$ which ensures A-orthogonality:

$$\begin{aligned}\langle d^{(k-1)}, d^{(k)}\rangle_A &= \langle d^{(k-1)}, r^{(k)}\rangle_A + \beta^{(k)}\langle d^{(k-1)}, d^{(k-1)}\rangle_A \\ &= \langle d^{(k-1)}, r^{(k)}\rangle_A - \frac{\langle d^{(k-1)}, r^{(k)}\rangle_A}{\|d^{(k-1)}\|_A^2}\|d^{(k-1)}\|_A^2 = 0.\end{aligned}$$

□

The spaces in (24.2), denoted by

$$\mathcal{K}_k(r^{(0}, A) := \operatorname{span}\{r^{(0)}, \dots, A^{k-1}r^{(0)}\}$$

are called Krylov subspaces and play a prominent role in other iterative methods such as GMRES and BiCGstab which, indeed, are even termed Krylov (sub)space methods. A characteristic property of these methods is that the increment lies in the actual Krylov subspace:

$$x^{(k)} - x^{(k-1)} \in \mathcal{K}_k(r^{(0)}, A).$$

A consequence of the previous results is

Theorem 24.3. *The* **CG** *algorithm reaches the exact solution to SLEs in at most n steps for any $x^{(0)}$.*

So in fact **CG** is a direct method. But in practice it is considered as an iterative method because ε_r usually is much bigger than the machine precision ε_m so that the iteration terminates with an approximation $x^{(k)}$ where k is much smaller than n.

Lecture 25

Error Analysis - Comparison of SD and CG

Error analysis for iterative methods is convergence analysis, rounding errors are not taken into account. The concepts of analysing SD and CG are similar which is why they are presented together. We start with a helpful lemma relating energy and Euclidean norm.

Lemma 25.1. *Assume that $\|e^{(k)}\|_A \leq cq^k \|e^{(0)}\|_A$ for all $k \in \mathbb{N}$ and some constants $c, q > 0$. Then*

$$\|e^{(k)}\|_2 \leq \sqrt{\kappa_2(A)} cq^k \|e^{(0)}\|_2 \quad \forall k \in \mathbb{N}.$$

Proof. Denoting the minimal and maximal eigenvalue of A by $\lambda_{\min}$ and $\lambda_{\max}$, respectively, and recalling that $\kappa_2(A) = \lambda_{\max}/\lambda_{\min}$:

$$\|e^{(k)}\|_2^2 \leq \frac{1}{\lambda_{\min}} \|e^{(k)}\|_A^2 \leq \frac{1}{\lambda_{\min}} c^2 q^{2k} \|e^{(0)}\|_A^2 \leq \frac{\lambda_{\max}}{\lambda_{\min}} (cq^k)^2 \|e^{(0)}\|_2^2.$$

□

The first results concerns **SD**.

Theorem 25.2. *The convergence rate of* **SD** *is*

$$\|e^{(k)}\|_A \leq \left(\sqrt{1 - \frac{1}{\kappa_2(A)}} \right)^k \|e^{(0)}\|_A.$$

Proof. Recalling (22.4) which with $d^{(k)} = r^{(k)}$ reads

$$g(x^{(k-1)} + \alpha r^{(k-1)}) = \frac{1}{2}\alpha^2 \|r^{(k-1)}\|_A^2 - \alpha \langle r^{(k-1)}, r^{(k-1)} \rangle + \frac{1}{2} \|r^{(k-1)}\|_{A^{-1}}^2,$$

we first observe that

$$\begin{aligned} g(x^{(k)}) &= g(x^{(k-1)} + \alpha^{(k-1)} d^{(k-1)}) \\ &= \frac{\|r^{(k-1)}\|_2^4}{2\|r^{(k-1)}\|_A^4} \|r^{(k-1)}\|_A^2 - \frac{\|r^{(k-1)}\|_2^2}{\|r^{(k-1)}\|_A^2} \|r^{(k-1)}\|_2^2 + \frac{1}{2} \|r^{(k-1)}\|_{A^{-1}}^2 \\ &= \frac{1}{2} \|r^{(k-1)}\|_{A^{-1}}^2 - \frac{\|r^{(k-1)}\|_2^4}{2\|r^{(k-1)}\|_A^2} \\ &= \left(1 - \frac{\|r^{(k-1)}\|_2^4}{\|r^{(k-1)}\|_A^2 \|r^{(k-1)}\|_{A^{-1}}^2} \right) \underbrace{\frac{1}{2} \|r^{(k-1)}\|_{A^{-1}}^2}_{= g(x^{(k-1)})}, \end{aligned}$$

Using the estimates $\|v\|_A^2 \leq \lambda_{\max}\|v\|_2^2$ and $\|v\|_{A^{-1}}^2 \leq \frac{1}{\lambda_{\min}}\|v\|_2^2$ we obtain that this is

$$\begin{aligned}
&\leq \left(1 - \frac{\|r^{(k-1)}\|_2^4}{\lambda_{\max}\|r^{(k-1)}\|_2^2 \frac{1}{\lambda_{\min}}\|r^{(k-1)}\|_2^2}\right) g(x^{(k-1)}) \\
&= \left(1 - \frac{1}{\kappa_2(A)}\right) g(x^{(k-1)}).
\end{aligned}$$

Therefore

$$g(x^{(k)}) \leq \left(1 - \frac{1}{\kappa_2(A)}\right)^k g(x^{(0)})$$

Using that $g(x^{(l)}) = \frac{1}{2}\|e^{(l)}\|_A^2$ for $l = 0, k$ (see (22.2)) yields the assertion. □

For **CG** we have the following result where $\mathcal{P}^k$ denotes the set of polynomials p of degree $\leq k$ with $p(0) = 1$ and $\Lambda(A)$ is the set of eigenvalues of A:

Theorem 25.3. *If* **CG** *has not yet converged after step k then*

$$\|e^{(k)}\|_A = \inf_{p\in\mathcal{P}^k} \|p(A)e^{(0)}\|_A \leq \inf_{p\in\mathcal{P}^k} \max_{\lambda\in\Lambda(A)} |p(\lambda)|\|e^{(0)}\|_A.$$

Proof. We only prove the first equality. The second one is an exercise.
As $x^{(0)} - x^{(k)}$ is an element of the Krylov space $\mathcal{K}_k(A, r^{(0)}) = \text{span}\{r^{(0)}, \dots, A^{k-1}r^{(0)}\}$ we can write

$$e^{(k)} = x - x^{(k)} = x - x^{(0)} + x^{(0)} - x^{(k)} = e^{(0)} + \sum_{j=0}^{k-1} \eta_{j+1} A^j \underbrace{r^{(0)}}_{=Ae^{(0)}} = \Big(1 + \sum_{j=1}^{k} \eta_j A^j\Big) e^{(0)}$$

with appropriate $\eta_1, \dots, \eta_k \in \mathbb{R}$. Let now $q(\lambda) := 1 + \sum_{j=1}^k \eta_j \lambda^j \in \mathcal{P}^k$. Then $e^{(k)} = q(A)e^{(0)}$ so that

$$\|x - x^{(k)}\|_A = \|e^{(k)}\|_A = \|q(A)e^{(0)}\|_A. \tag{25.1}$$

For any polynomial $p \in \mathcal{P}^k$, $p(\lambda) = 1 + \sum_{j=1}^k \gamma_j\lambda^j$, we have

$$p(A)e^{(0)} = e^{(0)} + \sum_{j=1}^{k} \gamma_j A^j e^{(0)} = x - x^{(0)} + \underbrace{\sum_{j=0}^{k-1} \gamma_{j+1} A^j r^{(0)}}_{\in\mathcal{K}_k(A, r^{(0)})}.$$

But as we have seen in Lemma 23.1, the iterate $x^{(k)}$ minimises $y \mapsto \|y - x\|_A = \sqrt{2g(y)}$ on $x^{(0)} + \mathcal{K}_k(A, r^{(0)})$. Therefore

$$\|e^{(k)}\|_A = \|x - x^{(k)}\|_A \leq \left\| x - x^{(0)} + \sum_{j=0}^{k-1} \gamma_{j+1} A^j r^{(0)} \right\|_A = \|p(A)e^{(0)}\|_A.$$

Together with (25.1) this proves the first equality. □

Chebyshev polynomials

These polynomials are defined by

$$T_n(x) := \frac{1}{2}\big((x+\sqrt{x^2-1})^n + (x-\sqrt{x^2-1})^n\big), \quad x \in [-1,1]$$

and fulfil the recursive formula

$$T_0(x) = 1, \quad T_1(x) = x, \qquad T_{n+1}(x) = 2xT_n(x) - T_{n-1}(x),\ n = 1,2,\dots$$

A further definition is

$$T_n(x) = \cos(n \arccos(x)), \quad x \in (-1,1).$$

We see that $\max_{|x|\le 1} |T_n(x)| \le 1$, and indeed the Chebyshev polynomials play an important role when optimising with respect to the norm $\|\cdot\|_\infty$.
Let $\lambda_{\max}$ and $\lambda_{\min}$ denote the maximal and minimal eigenvalue of A and consider the rescaled Chebyshev polynomial

$$p(x) := T_n\Big(\gamma - \frac{2x}{\lambda_{\max}-\lambda_{\min}}\Big)\Big/T_n(\gamma)$$

where

$$\gamma = \frac{\lambda_{\max}+\lambda_{\min}}{\lambda_{\max}-\lambda_{\min}} = \frac{\lambda_{\max}/\lambda_{\min}+1}{\lambda_{\max}/\lambda_{\min}-1} = \frac{\kappa_2(A)+1}{\kappa_2(A)-1}, \tag{25.2}$$

where we recall that $\|A\|_2 = \lambda_{\max}$ and $\|A^{-1}\|_2 = \lambda_{\min}^{-1}$ so that $\kappa_2(A) = \lambda_{\max}/\lambda_{\min}$. For $x \in [\lambda_{\min}, \lambda_{\max}]$ we have that $\gamma - 2x/(\lambda_{\max}-\lambda_{\min}) \in [-1,1]$, and since then $|T_n(x)| \le 1$ we arrive at

$$|p(x)| \le 1/T_n(\gamma), \quad x \in [\lambda_{\min}, \lambda_{\max}]. \tag{25.3}$$

Writing $\kappa := \kappa_2(A)$ we first observe that

$$\begin{aligned}\frac{\kappa+1}{\kappa-1} \pm \sqrt{\frac{(\kappa+1)^2}{(\kappa-1)^2}-1} &= \frac{\kappa+1}{\kappa-1} \pm \sqrt{\frac{(\kappa+1)^2-(\kappa-1)^2}{(\kappa-1)^2}} \\ &= \frac{\kappa+1\pm\sqrt{4\kappa}}{\kappa-1} = \frac{(\sqrt{\kappa}\pm 1)^2}{(\sqrt{\kappa}+1)(\sqrt{\kappa}-1)} = \Big(\frac{\sqrt{\kappa}+1}{\sqrt{\kappa}-1}\Big)^{\pm 1}.\end{aligned}$$

Thanks to (25.2) and (25.3)

$$\begin{aligned}T_n(\gamma) &= \frac{1}{2}\left(\left(\frac{\kappa+1}{\kappa-1} + \sqrt{\frac{(\kappa+1)^2}{(\kappa-1)^2}-1}\right)^n + \left(\frac{\kappa+1}{\kappa-1} - \sqrt{\frac{(\kappa+1)^2}{(\kappa-1)^2}-1}\right)^n\right) \\ &= \frac{1}{2}\left(\Big(\frac{\sqrt{\kappa}+1}{\sqrt{\kappa}-1}\Big)^n + \Big(\frac{\sqrt{\kappa}-1}{\sqrt{\kappa}+1}\Big)^n\right).\end{aligned}$$

Using (25.3) we see that for all $x \in [\lambda_{\min}, \lambda_{\max}]$

$$|p(x)| \le 2\left(\Big(\frac{\sqrt{\kappa}+1}{\sqrt{\kappa}-1}\Big)^n + \Big(\frac{\sqrt{\kappa}-1}{\sqrt{\kappa}+1}\Big)^n\right)^{-1} \tag{25.4}$$

Since $p \in \mathcal{P}^k$ for $k = n$ and since estimate (25.4) holds true for all $x \in \Lambda(A) \subset [\lambda_{\min}, \lambda_{\max}]$ we deduce from Theorem [6.19] the following result:

Theorem 25.4. [6.20] *If* **CG** *has not yet converged after step k then*

$$\|e^{(k)}\|_A \le 2\left(\Big(\frac{\sqrt{\kappa_2(A)}+1}{\sqrt{\kappa_2(A)}-1}\Big)^k + \Big(\frac{\sqrt{\kappa_2(A)}+1}{\sqrt{\kappa_2(A)}-1}\Big)^{-k}\right)^{-1} \|e^{(0)}\|_A.$$

Lecture 26

Computational Complexity and Preconditioning

Computational Complexity

For the computational complexity we proceed analogously as in the context of the linear iterative methods: Get an estimate for the required number of steps in terms of the system size n and the tolerance ε_r, and then multiply with the cost per step.

Assumption 26.1.

1. *Computing Ax involves a cost of $\Theta(n^\alpha)$ as $n \to \infty$ with some $\alpha \in [1, 2]$.*
2. *$\kappa_2(A) = \Theta(n^\beta)$ as $n \to \infty$ with some $\beta \geq 0$.*
3. *$\|e^{(0)}\|_A$ is uniformly bounded in n.*

Theorem 26.2. *Under Assumption 26.1, the cost to achieve $\|e^{(k)}\|_A \leq \varepsilon_r$ with* **SD** *is bounded by a function $C(n, \varepsilon_r)$ satisfying*

$$C(n, \varepsilon_r) = \Theta(n^{\alpha+\beta} \log(\varepsilon_r^{-1})) \quad \textit{as } (n, \varepsilon_r) \to (\infty, 0).$$

Proof. By Theorem 25.2, $\|e^{(k)}\|_A \leq \left(\sqrt{1 - \frac{1}{\kappa_2(A)}}\right)^k \|e^{(0)}\|_A$, hence it is sufficient to achieve that

$$\frac{\|e^{(0)}\|_A}{\varepsilon_r} \leq \left(\frac{1}{\sqrt{1 - \frac{1}{\kappa_2(A)}}}\right)^k \quad \Leftrightarrow \quad k \geq \frac{\log(\|e^{(0)}\|_A) + \log(\varepsilon_r^{-1})}{\log\left(1/\sqrt{1 - \frac{1}{\kappa_2(A)}}\right)} =: k^\sharp(n, \varepsilon_r).$$

Using the Taylor expansion we see that

$$\log\left(\frac{1}{\sqrt{1-x}}\right) = \frac{1}{2}x + O\left(x^2\right) \text{ as } x \to 0,$$

and with $x = 1/\kappa_2(A) = \Theta(n^{-\beta})$ one can proceed as in the proof of Theorem 21.2 to show the assertion. $\square$

Theorem 26.3. *Under Assumption 26.1, the cost to achieve $\|e^{(k)}\|_A \leq \varepsilon_r$ with* **CG** *is bounded by a function $C(n, \varepsilon_r)$ satisfying*

$$C(n, \varepsilon_r) = \Theta(n^{\alpha+\frac{1}{2}\beta} \log(\varepsilon_r^{-1})) \quad \textit{as } (n, \varepsilon_r) \to (\infty, 0).$$

(Spot the difference to the previous theorem!)

Proof. As for the previous theorem but based on the estimate in Theorem 25.4. The fact that there only $\sqrt{\kappa_2(A)}$ appears rather than $\kappa_2(A)$ as in the estimate for **SD** (see Theorem 25.2) leads to the prefactor $\frac{1}{2}$ in front of β. □

Preconditioning

Apparently, the lower the condition number $\kappa_2(A)$ the better the convergence of **SD** and **CG**, and this holds true for Krylov space methods in general. Since for every regular $B \in \mathbb{R}^{n\times n}$

$$b = Ax = ABB^{-1}x = (AB)(B^{-1}x) =: AB\tilde{x},$$

an appropriate choice of B may yield a system with $\kappa(AB) < \kappa(A)$. But a problem emerges: Even if $B \in \mathbb{R}^{n\times n}$ is positive definite (which we assume from now on) there is no guarantee that AB is positive definite. To solve this problem let us recall what we effectively need for **CG**. The symmetry of A is equivalent to

$$\langle Ax, y\rangle = \langle x, Ay\rangle \quad \forall x, y \in \mathbb{R}^{n\times n}.$$

This means that A considered as a linear operator on $\mathbb{R}^n$ is self-adjoint with respect to the standard inner product. Consider now the bilinear form

$$\langle \cdot,\cdot\rangle_B : \mathbb{R}^n \times \mathbb{R}^n \to \mathbb{R}, \quad \langle x, y\rangle_B := \langle x, By\rangle$$

which, thanks to the positivity of B, is an inner product (recall the arguments around (1.4) on page 15, and recall also the notation $\|\cdot\|_B$ for the associated norm). It turns out that $\tilde{A} = AB$ is self-adjoint with respect to $\langle\cdot,\cdot\rangle_B$: For all $x, y \in \mathbb{R}^n$

$$\langle ABx, y\rangle_B = \langle ABx, By\rangle = x^TB^TA^TBy \underbrace{=}_{A,B \text{ symmetric}} x^TBABy = \langle x, B(AB)y\rangle = \langle x, ABy\rangle_B.$$

Hence, we may just replace the standard inner product by $\langle\cdot,\cdot\rangle_B$ in algorithm **CG** and thus obtain a method for computing the solution to $b = AB\tilde{x}$. The essential lines read as follows where we inserted the definition of $\langle\cdot,\cdot\rangle_B$:

```
1:  $d^{(0)} := r^{(0)} := b - AB\tilde{x}^{(0)}$
2:  $l^{(0)} := \langle r^{(0)}, Br^{(0)}\rangle$
3:  if $l^{(0)} \le \varepsilon_r$ then
4:    return $\tilde{x}^{(0)}$
5:  else
6:    for $k = 1, 2, \dots$ do
7:      $h^{(k-1)} := ABd^{(k-1)}$
8:      $\alpha^{(k-1)} := l^{(k-1)}/\langle d^{(k-1)}, Bh^{(k-1)}\rangle \left( = \langle r^{(0)}, Br^{(0)}\rangle/\langle Bd^{(k-1)}, ABd^{(k-1)}\rangle\right)$
9:      $\tilde{x}^{(k)} := \tilde{x}^{(k-1)} + \alpha^{(k-1)}d^{(k-1)}$
10:     $r^{(k)} := r^{(k-1)} - \alpha^{(k-1)}h^{(k-1)} \left( = r^{(k-1)} - \alpha^{(k-1)}ABd^{(k-1)}\right)$
11:     $l^{(k)} := \langle r^{(k)}, Br^{(k)}\rangle$
12:     if $l^{(k)} \le \varepsilon_r$ then
13:       return $\tilde{x}^{(k)}$
14:     end if
15:     $\beta^{(k)} := l^{(k)}/l^{(k-1)} \left( = \langle r^{(k)}, Br^{(k)}\rangle/\langle r^{(k-1)}, Br^{(k-1)}\rangle\right)$
```

```
16:         $d^{(k)} := r^{(k)} + \beta^{(k)} d^{(k-1)}$
17:     end for
18: end if
```

As we are interested in x rather than $\tilde{x}$ we may multiply line 9 with B from the left to obtain $x^{(k)} := x^{(k-1)} + \alpha^{(k-1)} B d^{(k-1)}$. Doing the same with line 16 we see that we may introduce the vectors $q^{(k)} := Bd^{(k)}$ to replace the vectors $d^{(k)}$. The preconditioner B only acts on the residual, and it is possible to formulate and implement the algorithm such that only one matrix-vector multiplication with B per step is required. Introducing $s^{(k)} := Br^{(k)}$ we arrive at:

Algorithm 12 PCG (preconditioned conjugate gradient method)

```
input:  $A, B \in \mathbb{R}^{n\times n}$ positive definite, $b, x^{(0)} \in \mathbb{R}^n$, $\varepsilon_r > 0$.
output:  $x \in \mathbb{R}^n$ with $\|Ax - b\|_B \le \varepsilon_r$.
 1: $r^{(0)} := b - Ax^{(0)}$
 2: $q^{(0)} := s^{(0)} := Br^{(0)}$
 3: $l^{(0)} := \langle r^{(0)}, s^{(0)} \rangle$
 4: if $l^{(0)} \le \varepsilon_r$ then
 5:     return $x^{(0)}$
 6: else
 7:     for $k = 1, 2, \dots$ do
 8:         $h^{(k-1)} := Aq^{(k-1)}$
 9:         $\alpha^{(k-1)} := l^{(k-1)} / \langle q^{(k-1)}, h^{(k-1)} \rangle$
10:         $x^{(k)} := \tilde{x}^{(k-1)} + \alpha^{(k-1)} q^{(k-1)}$
11:         $r^{(k)} := r^{(k-1)} - \alpha^{(k-1)} h^{(k-1)}$
12:         $s^{(k)} := Br^{(k)}$
13:         $l^{(k)} := \langle r^{(k)}, s^{(k)} \rangle$
14:         if $l^{(k)} \le \varepsilon_r$ then
15:             return $x^{(k)}$
16:         end if
17:         $\beta^{(k)} := l^{(k)} / l^{(k-1)}$
18:         $q^{(k)} := s^{(k)} + \beta^{(k)} q^{(k-1)}$
19:     end for
20: end if
```

The big question now: How to choose B? One will want that the computation of $s^{(k)} = Br^{(k)}$ is cheap. But a good approximation of A^{-1} or, at least, keeping $\kappa(AB)$ low, is desired as well. To find a good trade-off depends strongly on the actual problem and the used computing hardware so, usually, requires some testing and trying. Some ideas to define preconditioners are stated below. Observe that in algorithm **PCG** we need the action of B on a vector which may be cheaper to compute than building the matrix B and employing the usual matrix-vector product algorithm.

- Very simple but occasionally highly effective: Choose $B := D^{-1}$ where D is the diagonal of A.

- Computing the LU factorisation or, since A is symmetric, the Cholesky factorisation $A = LL^T$ usually leads to a fill-in, i.e., zero-entries in a sparse matrix A become non-zero in the triangular matrices of the factorisation. But often those entries are 'small' compared to the other entries. So one may neglect the fill-in and compute an incomplete factorisation

$A \approx \tilde{A} = \tilde{L}\tilde{L}^T$. The preconditioner then is $B := \left(\tilde{L}\tilde{L}^T\right)^{-1}$.

- The linear solvers can serve as preconditioners as well. For instance, the action of B on a vector may correspond to a few steps of the Jacobi method.
- Modern general preconditioners also include (possibly algebraic) multigrid methods (not discussed in this course).

Lecture 27

Introduction to Eigenvalue Problems

Let $A \in \mathbb{C}^{n\times n}$. Suppose that $Ax = \alpha x$ with some $x \in \mathbb{C}^n\backslash\{0\}$ and $\alpha \in \mathbb{C}$. Then $\langle x, Ax\rangle = \alpha\langle x, x\rangle$. The fraction

$$r_A(x) := \frac{\langle x, Ax\rangle}{\langle x, x\rangle}$$

is called Rayleigh coefficient. And from the preceding calculation we see that $Ax = r_A(x)x$ if x is an eigenvector. Remarkably, this provides a method to compute the eigenvalue corresponding to an eigenvector.
Finding the eigenvalue of A is equivalent to finding the roots of the characteristic polynomial. However, there is the following result of Abel (1824):

Theorem 27.1. *If $n \geq 5$ then there is a polynimial of degree n with rational coefficients that has a real root which cannot be expressed by only using rational numbers, $+$, $-$, $*$, $/$, $(\cdot)^{\frac{1}{k}}$ with $k \in \mathbb{N}$.*

Consequently, algorithms for solving EVPs will be iterative.

Conditioning

The question is what impact a small perturbation ΔA of A has on the eigenvalues.
Let $\lambda(A) \in \mathbb{C}^n$ denote the set of eigenvalues, ordered in decreasing absolute value and repeated according to their algebraic multiplicity. The coefficients of the characteristic polynomial $\rho_A(z)$ of A depend continuously on the matrix entries, whence the roots as well. Therefore, $\lambda : \mathbb{C}^{n\times n} \to \mathbb{C}^n$ is a continuous function.

Theorem 27.2. *Let λ be a simple eigenvalue of A with associated right and left normalised eigenvectors x, y. Then for all sufficiently small $\Delta A \in \mathbb{C}^{n\times n}$ the matrix $A+\Delta A$ has an eigenvalue $\lambda + \Delta\lambda$ with*

$$\Delta\lambda = \frac{1}{\langle x, y\rangle}\Big(\langle y, \Delta A x\rangle + O(\|\Delta A\|_2^2)\Big) \quad \text{as } \|\Delta A\|_2 \to 0.$$

Definition 27.3. *Given $A \in \mathbb{C}^{n\times n}$ and an eigenvalue $\lambda \in \mathbb{C}$, let $x, y \in \mathbb{C}^n$ denote normalised right and left eigenvector, respectively. Then the eigenvalue condition number is*

$$\kappa_A(\lambda) := \begin{cases} \min_{x,y}(1/|\langle x, y\rangle|) & \text{with } x, y \text{ normalised right and left eigenvector s.t. } \langle x, y\rangle \neq 0 \\ \infty & \text{if no such right and left eigenvectors } x, y \in \mathbb{C}^n \text{ exist.} \end{cases}$$

Corollary 27.4. *Let $\lambda \in \mathbb{C}$ be a simple eigenvalue of $A \in \mathbb{C}^{n\times n}$ with corresponding right and left normalised eienvectors $x \in \mathbb{C}^n$ and $y \in \mathbb{C}^n$. Then for all sufficiently small $\Delta A \in \mathbb{C}^{n\times n}$ the matrix $A + \Delta A$ has an eigenvalue $\lambda + \Delta\lambda$ with*

$$|\Delta\lambda| \leq \kappa_A(\lambda)\big(\|\Delta A\|_2 + O(\|\Delta A\|_2^2)\big) \quad \textit{as } \Delta A \to 0.$$

A proof of the above theorem 27.2 is stated in Stuart & Voss [1] - Theorem 3.15. We look at an example with a non-simple eigenvalue in order to see why things go wrong in that case.

Example: (1) Consider the matrix

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

$\lambda = 1$ is an eigenvalue of A with algebraic multiplicity 2. A right eigenvector if $x = (1,0)^T$ and a left eigenvector is $y = (0,1)$, which means that $\kappa_A(\lambda) = \infty$. In fact, consider

$$\Delta A = \begin{pmatrix} 0 & 0 \\ \delta & 0 \end{pmatrix}, \quad \|\Delta A\|_a = O(|\delta|), \text{ as } \delta \to 0.$$

The matrix $A + \Delta A$ has eigenvalues $1 \pm \sqrt{\delta}$ so that $|\Delta\lambda| = \sqrt{|\delta|}$ but not $O(\|\Delta A\|_2) = O(|\delta|)$. In this example we have $\delta \mapsto \lambda_1(\delta) = 1 + \sqrt{\delta} \in \mathbb{C}$ for the first eigenvalue, and this curve is continuous in $\delta = 0$ but not differentiable.
(2) If A is Hermitian then the left and right eigenspaces coincide so that $\kappa_A(\lambda) = 1$ in this case.

Real Symmetric Case

From now on we will restrict our attention to symmetric real matrices $A \in \mathbb{R}^{n\times n}$, where the eigenvalues are denoted by $|\lambda_1| \geq |\lambda_2| \geq \cdots \geq |\lambda_n| \geq 0$. Associated normalised eigenvectors are denoted by $x_1, \ldots, x_n \in \mathbb{R}^n$, which then also form an orthonormal basis of $\mathbb{R}^n$.
Before proceeding with the first method to compute an eigenvalue let us state on result that will be used later on in the convergence proof.

Theorem 27.5.
(a) *A vector $x \in \mathbb{R}^n\backslash\{0\}$ is an eigenvector of A with eigenvalue λ if and only if $r_A(x) = \lambda$ and $\nabla r_A(x) = 0$.*
(b) *If $x \in \mathbb{R}^n$ is an eigenvector of A then*

$$|r_A(x) - r_A(y)| = O(\|x-y\|_2^2) \quad \textit{as } y \to x.$$

Proof. (a) A short calculation (exercise) shows that

$$\nabla r_A(z) = \frac{2}{\|z\|_2^2}\big(Az - r_A(z)z\big)$$

for any $z \neq 0$. If $Ax = \lambda x$ then $r_A(x) = \langle x, Ax\rangle / \langle x, x\rangle = \lambda$ and $\nabla r_A(x) = 2(\lambda x - r_A(x)x)/\|x\|_2^2 = 0$ as claimed.
Vice versa, if $\nabla r_A(x) = 0$ then $Ax = r_A(x)x$ so that $(x, r_A(x))$ is an eigenpair of A.
(b) This follows from considering the Taylor expansion of r_A around x and using part (a). $\square$

Lecture 28

Power Iteration

Recall that we consider the case of $A \in \mathbb{R}^{n\times n}$ symmetric. Denote the eigenvalues by $|\lambda_1| \geq |\lambda_2| \geq \cdots \geq |\lambda_n|$ with corresponding normalised eigenvectors $x_1, \dots, x_n \in \mathbb{R}^n$.

Idea: iterate $z^{(k)} = Az^{(k-1)}$ and hope that the iterates align with the eigenspace of λ_1.

Example: consider

$$A = \begin{pmatrix} 1 & 0 \\ 0 & \frac{1}{2} \end{pmatrix}, \quad z^{(0)} = \begin{pmatrix} 1 \\ 1 \end{pmatrix}.$$

Then

$$z^{(1)} = \begin{pmatrix} 1 \\ 2^{-1} \end{pmatrix}, z^{(2)} = \begin{pmatrix} 1 \\ 2^{-2} \end{pmatrix}, \dots, z^{(k)} = \begin{pmatrix} 1 \\ 2^{-k} \end{pmatrix} \to \begin{pmatrix} 1 \\ 0 \end{pmatrix}$$

which is an eigenvector to $\lambda_1 = 1$.

Algorithm 13 PI (power iteration)

input: $A \in \mathbb{R}^{n\times n}$ symmetric, $z^{(0)} \in \mathbb{R}^n$ with $\|z^{(0)}\|_2 \neq 0$ and $\langle z^{(0)}, x_1\rangle \neq 0$.
output: $z^{(k)} \in \mathbb{R}^n$ and $\lambda^{(k)} \in \mathbb{R}$ approximation to x_1 and λ_1.

```
1: $k = 1$
2: repeat
3:     $w^{(k)} := Az^{(k-1)}$
4:     $\lambda^{(k-1)} := \langle w^{(k)}, z^{(k-1)}\rangle$
5:     $z^{(k)} := w^{(k)}/\|w^{(k)}\|_2$
6:     $k := k + 1$
7: until Stopping criterion fulfilled
```

Observe that $\lambda^{(k-1)} = r_A(z^{(k-1)})$. When the goal is to compute the eigenvalue (rather than an eigenvector) then a criterion of the form $|\lambda^{(k)} - \lambda^{(k-1)}| \leq \varepsilon_r$ may be used.
For computing an eigenvector, a possible stopping criterion is $\|z^{(k)} - z^{(k-1)}\|_2 \leq \varepsilon_r$ or $\|z^{(k)} + z^{(k-1)}\|_2 \leq \varepsilon_r$ where we have to distinguish the sign because λ_1 may be negative. In fact, for

$$A = \begin{pmatrix} -1 & 0 \\ 0 & \frac{1}{2} \end{pmatrix}, \quad z^{(0)} = \begin{pmatrix} 1 \\ 1 \end{pmatrix}.$$

we obtain the iteratives

$$z^{(1)} = \begin{pmatrix} -1 \\ 2^{-1} \end{pmatrix}, z^{(2)} = \begin{pmatrix} 1 \\ 2^{-2} \end{pmatrix}, z^{(3)} = \begin{pmatrix} -1 \\ 2^{-3} \end{pmatrix} \dots, z^{(k)} = \begin{pmatrix} (-1)^k \\ 2^{-k} \end{pmatrix}$$

which do not converge, but the vectors $(-1)^k z^{(k)} = (1, (-1)^k 2^{-k})^T$ converge to $(1, 0)^T$.

Error Analysis

Theorem 28.1. *Assume that* $|\lambda_1| > |\lambda_2| \geq \dots$ *and that* $\alpha_1 := \langle x_1, z^{(0)}\rangle \neq 0$. *Then the sequences* $\{z^{(k)}\}_k$ *and* $\{\lambda^{(k)}\}_k$ *generated by* **PI** *satisfy*

$$\|z^{(k)} - \sigma^{(k)} x_1\|_2 = O\Big(\Big|\frac{\lambda_2}{\lambda_1}\Big|^k\Big), \quad |\lambda^{(k)} - \lambda_1| = O\Big(\Big|\frac{\lambda_2}{\lambda_1}\Big|^{2k}\Big) \qquad \text{as } k \to \infty$$

where $\sigma^{(k)} = \alpha_1\lambda_1^k/|\alpha_1\lambda_1^k| \in \{\pm 1\}$.

Proof. Let us write $z^{(0)} = \sum_{i=1}^n \alpha_i x_i$. Since $\alpha_1 \neq 0$

$$A^k z^{(0)} = \sum_i \alpha_i \lambda_i^k x_i = \alpha_1 \lambda_1^k \Big(x_1 + \sum_{i=2}^n \frac{\alpha_i}{\alpha_1}\Big(\frac{\lambda_i}{\lambda_1}\Big)^k x_i\Big)$$

and, using the orthonormality of the x_i,

$$\|A^k z^{(0)}\|_2^2 = |\alpha_1\lambda_1^k|^2 \Big(1 + \underbrace{\sum_{i=2}^n \Big|\frac{\alpha_i}{\alpha_1}\Big|^2 \Big|\frac{\lambda_i}{\lambda_1}\Big|^{2k}}_{=:\gamma_k \geq 0}\Big).$$

Therefore

$$z^{(k)} = \frac{A^k z^{(0)}}{\|A^k z^{(0)}\|_2} = \frac{A^k z^{(0)}}{|\alpha_1\lambda_1^k|\sqrt{1+\gamma_k}} = \underbrace{\frac{\alpha_1\lambda_1^k}{|\alpha_1\lambda_1^k|}}_{=\sigma^{(k)}} \Big(x_1 + \sum_{i=2}^n \frac{\alpha_i}{\alpha_1}\Big(\frac{\lambda_i}{\lambda_1}\Big)^k x_i\Big) \frac{1}{\sqrt{1+\gamma_k}}.$$

Thus,

$$\begin{aligned}
\|z^{(k)} - \sigma^{(k)} x_1\|_2 &\leq \Big\|z^{(k)} - \frac{A^k z^{(0)}}{|\alpha_1\lambda_1^k|}\Big\|_2 + \Big\|\frac{Az^{(k)}}{|\alpha_1\lambda_1^k|} - \sigma^{(k)} x_1\Big\|_2 \\
&= \Big|\frac{1}{\sqrt{1+\gamma_k}} - 1\Big| \Big\|\frac{A^k z^{(0)}}{|\alpha_1\lambda_1^k|}\Big\|_2 + \Big\|\sum_{i=2}^n \frac{\alpha_i}{\alpha_1}\Big(\frac{\lambda_i}{\lambda_1}\Big)^k x_i\Big\|_2 \\
&= \Big(1 - \frac{1}{\sqrt{1+\gamma_k}}\Big)\sqrt{1+\gamma_k} + \sqrt{\gamma_k} \\
&= \sqrt{1+\gamma_k} - 1 + \sqrt{\gamma_k} \\
&\leq 2\sqrt{\gamma_k}.
\end{aligned}$$

Now,

$$\gamma_k \leq \frac{1}{|\alpha_1|^2}\Big|\frac{\lambda_2}{\lambda_1}\Big|^{2k} \sum_{i=2}^n |\alpha_i|^2 \underbrace{\Big|\frac{\lambda_i}{\lambda_2}\Big|^{2k}}_{\leq 1} \leq \frac{1}{|\alpha_1|^2}\Big|\frac{\lambda_2}{\lambda_1}\Big|^{2k} \sum_{i=1}^n |\alpha_i|^2 = \frac{1}{|\alpha_1|^2}\Big|\frac{\lambda_2}{\lambda_1}\Big|^{2k} \|z^{(0)}\|_2^2$$

so that

$$\sqrt{\gamma_k} \leq \frac{\|z^{(0)}\|_2}{|\alpha_1|}\Big|\frac{\lambda_2}{\lambda_1}\Big|^k$$

from which we get the first assertion. The second follows with Theorem 27.5 (b):

$$|\lambda^{(k)} - \lambda_1| = |r_A(z^{(k)}) - r_A(\sigma^{(k)} x_1)| = O(\|z^{(k)} - \sigma^{(k)} x_1\|_2^2) = O\Big(\Big|\frac{\lambda_2}{\lambda_1}\Big|^{2k}\Big).$$

□

Remarks on the Implementation

The most expensive part in each step is the matrix-vector multiplication in line 3 (the other lines involve only $O(n)$ operations as $n \to \infty$). To reduce the cost, the idea is to transform the matrix to $B = SAS^{-1}$ with a regular matrix S so that B contains many zeros. Recall that such a similarity transformation does not change the eigenvalues, and eigenvectors corresponding to $\lambda_1, \dots, \lambda_n$ are given by $Sx_1, \dots, Sx_n$.
One possibility is to transform A to <u>upper Hessenberg form</u>,

$$B = \begin{pmatrix} * & \cdots & \cdots & \cdots & * \\ * & \ddots & & & \vdots \\ 0 & \ddots & \ddots & & \vdots \\ \vdots & \ddots & \ddots & \ddots & \vdots \\ 0 & \cdots & 0 & * & * \end{pmatrix}$$

using Householder reflections again. But in contrast to the computation of a QR factorisation only the entries below the first lower diagonal are made to vanish. Let us start with

$$A = \begin{pmatrix} * & u_1 & \cdots & u_{n-1} \\ u_1 & * & \cdots & * \\ \vdots & \vdots & \ddots & \vdots \\ u_{n-1} & * & \cdots & * \end{pmatrix} \quad \rightsquigarrow \quad Q_1 A = \begin{pmatrix} * & u_1 & \cdots & \cdots & u_{n-1} \\ \pm\|u\|_2 & * & \cdots & \cdots & * \\ 0 & \vdots & \ddots & & \vdots \\ \vdots & \vdots & & \ddots & \vdots \\ 0 & * & \cdots & \cdots & * \end{pmatrix}$$

where $Q_1 = \operatorname{diag}(1, H_1)$ with $H_1 = I - 2vv^T \in \mathbb{R}^{(n-1)\times(n-1)}$ is a reflection that leaves the first row of A unchanged. When multiplying with $Q_1^{-1} = Q_1^T = Q_1$ from the right then also the entries above the first upper diagonal in the first row vanish:

$$Q_1 A Q_1^{-1} = \begin{pmatrix} * & \pm\|u\|_2 & 0 & \cdots & 0 \\ \pm\|u\|_2 & * & \cdots & \cdots & * \\ 0 & \vdots & \ddots & & \vdots \\ \vdots & \vdots & & \ddots & \vdots \\ 0 & * & \cdots & \cdots & * \end{pmatrix}.$$

Observe that this is due to the symmetry of A. For an arbitrary matrix we would keep some entries in the first row.
Since $Q_1 A Q_1^{-1} = Q_1 A Q_1^T$ is symmetric, too, we may proceed in a similar fashion to transform A to a tridiagonal matrix

$$Q_{n-1} \cdots Q_1 A Q_1^{-1} \cdots Q_{n-1}^T = \begin{pmatrix} * & * & 0 & \cdots & 0 \\ * & * & \ddots & \ddots & \vdots \\ 0 & \ddots & \ddots & \ddots & 0 \\ \vdots & \ddots & \ddots & \ddots & * \\ 0 & \cdots & 0 & * & * \end{pmatrix} = B.$$

The matrix-vector multiplication with B now has a cost of $O(n)$ as the other operations per step.
It should be remarked that **PI** yields an eigenvector for B which, using $Q_1, \dots, Q_{n-1}$ needs to be transformed back to an eigenvector of A.

Computational Complexity

We proceed in the usual way for iterative methods, estimating the number of required steps and the multiplying with the cost per step.
The goal

$$\|z^{(k)} - \sigma^{(k)} x_1\|_2 \le \varepsilon_r \tag{28.1}$$

is achieved if

$$2\frac{\|z^{(0)}\|_2}{|\alpha_1|}\left|\frac{\lambda_2}{\lambda_1}\right|^k \le \varepsilon_r$$

which requires

$$k^\sharp(n, \varepsilon_r) = \frac{\log(\varepsilon_r^{-1}) + \log(2\|z^{(0)}\|_2) - \log(|\alpha_1|)}{\log(|\lambda_2/\lambda_1|)}$$

steps.
Assume now that

$$\left|\frac{\lambda_1}{\lambda_2}\right| = 1 + \Theta(n^{-\beta}) \text{ for some } \beta > 0 \text{ as } n \to \infty,$$

that $\|z^{(0)}\|_2$ and $|\alpha_1|$ are $\Theta(1)$ as $n \to \infty$, and that the cost per iteration step is $O(n)$ as $n \to \infty$.

Theorem 28.2. *Under the above assumptions, the cost to achieve (28.1) with* **PI** *is bounded by a function $C(n, \varepsilon_r)$ satisfying*

$$C(n, \varepsilon_r) = \Theta(n^{1+\beta}\log(\varepsilon_r^{-1}) + n^3) \quad \textit{as } (n, \varepsilon_r) \to (\infty, 0).$$

The proof is similarly to the proof of Theorem 21.2 which explains the first term $n^{1+\beta}\log(\varepsilon_r^{-1})$. Here, the β arises from the fact that λ_2/λ_1 converges to 1 as $n \to \infty$, and the 1 comes from the cost per iteration step which requires that the matrix-vector multiplication is $O(n)$ as $n \to \infty$. The difference to previous results is the additional term n^3 which arises from transforming the initial matrix to a tridiagonal matrix.

Lecture 29

Simultaneous Iteration and the QR Method

Suppose that λ_1 and x_1 are known. If $z^{(0)} \perp x_1$ then also $Az^{(0)} \perp x_1$, at least modulo numerical errors, which may require to project to $\operatorname{span}\{x_1\}^\perp$. As a consequence: **PI** with such a $z^{(0)}$ will yield λ_2 and x_2 (under appropriate, not too severe assumptions). Afterwards, one could go on picking a $z^{(0)} \perp \operatorname{span}\{x_1, x_2\}$ in order to compute λ_3 and x_3 and so on. But subsequent iterations eventually are disadvantageous.

Idea: Perform **PI** with a set $\{z_1^{(0)}, \dots, z_n^{(0)}\}$ of orthonormal vectors and re-orthonormalise after each multiplication with A. More precisely, defining $Z^{(0)} := (z_1^{(0)}, \dots, z_n^{(0)}) \in \mathbb{C}^{n\times n}$, let us consider the following simultaneous iteration:

```
for $k = 1, 2, \dots$ do
  $W^{(k)} := AZ^{(k-1)}$
  compute a QR factorisation $W^{(k)} = Z^{(k)} R^{(k)}$
  $\Lambda^{(k)} := (Z^{(k-1)})^T W^{(k)} \quad \big( = (Z^{(k-1)})^T A Z^{(k-1)} \big)$
end for
```

We write $w_i^{(k)}$, $i = 1, \dots, n$, for the column vectors of $W^{(k)}$ in the following, and let us for simplicity assume that $\lambda_i > 0$ so that we do not have to discuss oscillations. Clearly, $w_1^{(k)} = Az_1^{(k-1)}$, and from the QR factorisation $w_1^{(k)} = z_1^{(k)} r_{1,1}$. Because $\|z_1^{(k)}\|_2 = 1$ we obtain that $r_{1,1} = \|w_1^{(k)}\|_2$. But this means that

$$z_1^{(k)} = \frac{w_1^{(k)}}{\|w_1^{(k)}\|_2} = \frac{Az_1^{(k-1)}}{\|Az_1^{(k-1)}\|_2},$$

which we recognise as the power iteration. We expect that $z_1^{(k)} \to x_1$, and furthermore

$$\Lambda_{1,1}^{(k)} = (z_1^{(k-1)})^T w_1^{(k)} = (z_1^{(k-1)})^T A z_1^{(k-1)} = r_A(z_1^{(k-1)}) \to \lambda_1 \quad \text{as } k \to \infty.$$

So far so good. Now, consider

$$w_2^{(k)} = Az_2^{(k-1)} = z_1^{(k)} r_{1,2} + z_2^{(k)} r_{2,2}.$$

Taking the scalar product with $z_1^{(k)}$ we see that $r_{1,2} = \langle z_1^{(k)}, w_2^{(k)} \rangle$, whence

$$z_2^{(k)} r_{2,2} = \underbrace{w_2^{(k)} - \langle z_1^{(k)}, w_2^{(k)} \rangle z_1^{(k)}}_{\text{projection of } w_2^{(k)} \text{ onto } \operatorname{span}\{z_1^{(k)}\}^\perp}.$$

Since $r_{2,2} = \|w_2^{(k)} - \langle z_1^{(k)}, w_2^{(k)}\rangle z_1^{(k)}\|_2$ and recalling that $w_2^{(k)} = Az_2^{(k-1)}$ we obtain that

$$z_2^{(k)} = \frac{Az_2^{(k-1)} - \langle z_1^{(k)}, Az_2^{(k-1)}\rangle z_1^{(k)}}{\|Az_2^{(k-1)} - \langle z_1^{(k)}, Az_2^{(k-1)}\rangle z_1^{(k)}\|_2}.$$

Given that $z_1^{(k)} \to x_1$ we expect that $\lim_{k\to\infty} z_2^{(k)} \in \operatorname{span}\{x_1\}^\perp$. Altogether, the iterates $z_2^{(k)}$ form an approximation to the power iteration on $\operatorname{span}\{x_1\}^\perp$ and, as discussed at the beginning, can be expected to converge to x_2. We also see that then

$$\Lambda_{2,2} = (z_2^{(k-1)})^T w_2^{(k)} = (z_2^{(k-1)})^T Az_2^{(k-1)} = r_A(z_2^{(k-1)}) \to \lambda_2 \quad \text{as } k \to \infty.$$

Furthermore, using the orthogonality of x_1 and x_2 we expect that

$$\Lambda_{1,2} = (z_1^{(k-1)})^T w_2^{(k)} \to x_1^T Ax_2 = x_1^T(\lambda_2 x_2) = 0 \quad \text{as } k \to \infty.$$

Arguing in an analogous way for the other $z_i^{(k)}$ and entries of $\Lambda^{(k)}$ we altogether expect that

$$Z^{(k)} \to (x_1, \dots, x_n) \text{ and } \Lambda^{(k)} \to \operatorname{diag}(\lambda_1, \dots, \lambda_n) \quad \text{as } k \to \infty.$$

If only the eigenvalues are required then there is a very elegant way to reformulate the simultaneous iteration. Define

$$Q^{(k)} := (Z^{(k-1)})^T Z^{(k)}.$$

Then

$$Q^{(k)}R^{(k)} = (Z^{(k-1)})^T Z^{(k)}(Z^{(k)})^{-1}W^{(k)} = (Z^{(k-1)})^T W^{(k)} = \Lambda^{(k)}$$

and

$$\begin{aligned}R^{(k)}Q^{(k)} &= (Z^{(k)})^{-1}W^{(k)}(Z^{(k-1)})^T Z^{(k)} \\ &= (Z^{(k)})^T AZ^{(k-1)}(Z^{(k-1)})^{-1}Z^{(k)} = (Z^{(k)})^T AZ^{(k)} = \Lambda^{(k+1)}.\end{aligned}$$

So when we have a QR factorisation of the actual matrix approximating the eigenvalues $\Lambda^{(k)}$ then we only have to interchange the two matrices to compute the next iterate $\Lambda^{(k+1)}$.

Algorithm 14 QRI (QR iteration for eigenvalues)

input: $A \in \mathbb{R}^{n\times n}$ symmetric and tridiagonal.
output: $\Lambda \in \mathbb{R}^{n\times n}$ with diagonal entries approximating the eigenvalues of A.

```
1: $\Lambda^{(0)} := A$
2: for $k = 1, 2, \dots$ do
3:     compute a QR factorisation $\Lambda^{(k-1)} = Q^{(k-1)}R^{(k-1)}$ (with Givens rotations)
4:     $\Lambda^{(k)} := R^{(k-1)}Q^{(k-1)}$
5:     stop iteration if diagonal entries of $\Lambda^{(k)}$ are close to those of $\Lambda^{(k-1)}$ (modulo sign)
6: end for
```

Example: Numerically investigate (and perhaps consider a bit of history):

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

Appendix A

Floating point representation

Any number $x \in \mathbb{R}$ can be represented with respect to a basis $\beta \in \mathbb{N}\backslash\{0,1\}$:

$$x = \sigma \sum_{n \in \mathbb{Z}} a_n \beta^n, \qquad a_n \in \{0,\dots,\beta-1\} \ \forall n \in \mathbb{Z}, \quad \sigma \in \{\pm 1\} \text{ sign.}$$

Computers are based on the dual system, $\beta = 2$. And as they are finite, the idea is to cut the infinite sum and to approximate x by

$$\xi = \sigma 2^e \Big(1 + \sum_{n=1}^{t} a_n 2^{-n}\Big) = \sigma 2^e \times (1.a_1 \dots a_t)_2, \qquad e = 2^{-m} \sum_{i=1}^{s} b_i 2^i.$$

The $(a_1 \dots a_t)$ are called mantissa, here of length t with fraction bits $a_n \in \{0,1\}$, and the $(b_1 \dots b_s)$ represent the exponent of length s with exponent bits $b_i \in \{0,1\}$. The number m is called bias or shift.

Example, IEEE Standard 754 Double Precision:
There are 64 bits to represent a number. The first bit is the sign bit, the next eleven bits are the exponent bits, and the final 52 bits are the fraction bits:

$$(\sigma b_1 \dots b_{11} a_1 \dots a_{52})$$

The bias is fixed at $m = 1023$.
The relative error when approximating a number x by its nearest neighbour ξ is

$$\frac{|\xi - x|}{|x|} \le \varepsilon_m \approx 10^{-16}.$$

ε_m is called machine precision. For positive x, Values for ξ are between $\approx 10^{-320}$ and 10^{308}. If x is bigger (smaller) then we have to deal with an overflow (underflow).

Bibliography

[1] Andrew Stuart and Jochen Voss, *Matrix Analysis and Algorithms*, Lecture notes.

[2] Roger A. Horn and Charles R. Johnson, *Matrix Analysis*, Cambridge University Press, 1985.

[3] Gene H. Golub and Charles F. van Loan, *Matrix Computations*, 3. ed., Johns Hopkins University Press, 1996.

[4] Lloyd Trefethen, David Bau *Numerical Linear Algebra*, SIAM, 1997.

[5] Nicholas Higham, *Accuracy and Stability of Numerical Algorithms*, SIAM, 2002.

[6] David Kincaid and Ward Cheney, *Numerical Analysis*, 3. ed., AMS, 2002.

[7] Thomas S. Shores, *Applied Linear Algebra and Matrix Analysis*, Springer, 2007.