## Programming, numerics and optimization Lecture B-3: Linear systems I: Direct and iterative methods

Łukasz Jankowski ljank@ippt.pan.pl

Institute of Fundamental Technological Research Room 4.32, Phone +22.8261281 ext. 428

April 13, 2021<sup>1</sup>

<sup>1</sup>Current version is available at http://info.ippt.pan.pl/~ljank.

Outline •	Basics 000000000	Existence & uniqueness	Direct methods	Iterative methods	HW7 00			
Outline								



2 Existence and uniqueness of solution

#### Oirect methods

Iterative methods

#### 5 Homework 7



#### Basic notions

- Types of problems
- Methods of solution



• A is an  $m \times n$  matrix: *m* rows, *n* columns

• If 
$$m = n$$
, then **A** is square.

- A is a row vector, if m = 1, and a column vector, if n = 1.
- In algebraic terms A defines a linear mapping A: ℝ<sup>n</sup> → ℝ<sup>m</sup>, such that x → y = Ax.
- $\mathbf{A}^{\mathsf{T}}$  is the *transpose* of  $\mathbf{A}$  ( $\mathbf{A}$  flipped about its main diagonal, with rows turned into columns and vice versa):  $a_{ij} \rightarrow a_{ji}$ . If  $\mathbf{A}^{\mathsf{T}} = \mathbf{A}$ , then  $\mathbf{A}$  is *symmetric* (and obviously square).
- For complex matrices  $\mathbf{A}^{H}$  is the *conjugate transpose*:  $a_{ij} \rightarrow \bar{a}_{ji}$ . If  $\mathbf{A}^{H} = \mathbf{A}$ , then  $\mathbf{A}$  is *Hermitian*.

## Basic matrix operations

Existence & uniqueness

Outline

Basics

000000000

• *Matrix sum*:  $\mathbf{A}, \mathbf{B} \in \mathbb{R}^{m \times n}$ , numerical cost: O(mn) operations

Direct methods

$$\mathbf{C} = \mathbf{A} + \mathbf{B}, \quad c_{ij} = a_{ij} + b_{ij}$$

• Matrix-vector product:  $\mathbf{A} \in \mathbb{R}^{m \times n}, \mathbf{x} \in \mathbb{R}^{n}$ , numerical cost: O(mn) operations

$$\mathbf{y} = \mathbf{A}\mathbf{x}, \quad y_i = \sum_{j=1}^n a_{ij} x_j$$

 $\begin{bmatrix} a_{11} & a_{12} & \cdots & a_{1n} \\ a_{21} & a_{22} & \cdots & a_{2n} \\ \vdots & \vdots & \ddots & \vdots \\ a_{m1} & a_{m2} & \cdots & a_{mn} \end{bmatrix} \begin{bmatrix} x_1 \\ x_2 \\ \vdots \\ x_n \end{bmatrix} = \begin{bmatrix} \sum_{j=1}^n a_{1j} b_j \\ \sum_{j=1}^n a_{2j} b_j \\ \vdots \\ \sum_{j=1}^n a_{mj} b_j \end{bmatrix} = \begin{bmatrix} y_1 \\ y_2 \\ \vdots \\ y_m \end{bmatrix}$ 

Iterative methods

#### Basic matrix operations

Basics

000000000

Existence & uniqueness

• *Matrix-matrix product*:  $\mathbf{A} \in \mathbb{R}^{m \times n}$ ,  $\mathbf{B} \in \mathbb{R}^{n \times p}$ , numerical cost: O(mnp) operations

Direct methods

$$\mathbf{C} = \mathbf{A}\mathbf{B}, \quad c_{ij} = \sum_{k=1}^{n} a_{ik} b_{kj}$$

$$\begin{bmatrix} a_{11} & a_{12} & \cdots & a_{1n} \\ a_{21} & a_{22} & \cdots & a_{2n} \\ \vdots & \vdots & \ddots & \vdots \\ a_{m1} & a_{m2} & \cdots & a_{mn} \end{bmatrix} \begin{bmatrix} b_{11} & b_{12} & \cdots & b_{1p} \\ b_{21} & b_{22} & \cdots & b_{2p} \\ \vdots & \vdots & \ddots & \vdots \\ b_{n1} & b_{n2} & \cdots & b_{np} \end{bmatrix}$$
$$= \begin{bmatrix} c_{11} & c_{12} & \cdots & c_{1n} \\ c_{21} & c_{22} & \cdots & a_{2n} \\ \vdots & \vdots & \ddots & \vdots \\ c_{m1} & c_{m2} & \cdots & c_{mp} \end{bmatrix}$$

6/62

Iterative methods

#### Basic matrix operations

Matrix sum

- commutative:  $\mathbf{A} + \mathbf{B} = \mathbf{B} + \mathbf{A}$
- associative:  $\mathbf{A} + (\mathbf{B} + \mathbf{C}) = (\mathbf{A} + \mathbf{B}) + \mathbf{C}$

Matrix product

- in general, not commutative:
  - if either **A** or **B** is non-square, both multiplications may not be possible (incompatible dimensions)
  - even if both matrices are square of the same dimensions, usually  $\textbf{AB} \neq \textbf{BA}$
- distributive: A(B + C) = AB + AC
- associative: A(BC) = (AB)C (but the numerical cost can be very different)

#### Basic matrix operations

Matrix product is associative, but the numerical cost can be much different. Consider a matrix-matrix-vector product<sup>2</sup>:

$$\label{eq:alpha} \begin{split} \textbf{A}, \textbf{B} \in \mathbb{R}^{N \times N} \text{ (matrices)}, \quad \textbf{C} \in \mathbb{R}^{N} \text{ (a vector)} \\ \textbf{A}(\textbf{BC}) = (\textbf{AB})\textbf{C} \end{split}$$

#### Numerical cost of A(BC)

$$\cot \mathbf{BC} = O(N^2)$$
$$\cot \mathbf{A(BC)} = O(N^2) + O(N^2) = O(N^2)$$

#### Numerical cost of (AB)C

$$\cot \mathbf{AB} = O(N^3)$$
$$\cot (\mathbf{AB})\mathbf{C} = O(N^3) + O(N^2) = O(N^3)$$

<sup>2</sup>In C/C++, operator \* is left-associative, that is  $\mathbf{A} * \mathbf{B} * \mathbf{C} = (\mathbf{A} * \mathbf{B}) * \mathbf{C}$ .

8/62



Problems related to finite-dimensional linear systems can be roughly classified into three groups:

Find-a-solution problems:

given **A** and **y**, solve Ax = y

2 Least-square problems:

given **A** and **y**, minimize  $\|\mathbf{y} - \mathbf{A}\mathbf{x}\|^2$ 

S Eigenvalue problems: in the narrow sense,

given **A**, find vector-scalar pairs  $(\mathbf{x}, \lambda)$  such that  $\mathbf{A}\mathbf{x} = \lambda \mathbf{x}$ . In a broader sense, the term "eigenvalue problems" can be used for all related problems, like the problems of finding singular values, null-spaces, etc.

## Methods of solution

00000

Outline

Basics

Methods used to solve problems involving linear system can be classified into two broad groups

- Direct methods,
- Iterative methods.

#### The direct methods

- Compute the solution in a finite number of steps, which is known in advance.
- In the exact arithmetic, the computed solution would be exact.
- Stability in finite precision arithmetics is usually well-analyzed.
- Require direct access to the elements of the system matrix **A** (which has thus to be known explicitly).

# Methods of solution

Outline

Basics

#### The iterative methods

- Compute an *approximate* solution in an iterative way.
- In general, the number of iterations to obtain a good approximation is unknown in advance.
- The convergence properties of the methods are often hard to analyze, especially in finite precision arithmetics. In some cases, finite-precision iterations may not converge at all.
- Many of the methods require access only to a matrix-vector multiplication procedure (Ax, sometimes also A<sup>T</sup>x), so that A may be given only implicitly (e.g. Ax can be the result of an experiment or a simulation).
- They are most useful for solving large sparse or structured systems, for which any factorization (necessary in direct methods) would take too much time, destroy the sparsity or the matrix structure, or be too inaccurate.

Outline 0	Basics 000000000	Existence & uniqueness	Direct methods	Iterative methods	HW7 00
Out	line				



#### Range and null-space of a matrix

Outline

Basics

In algebraic terms, a matrix  $\mathbf{A} \in \mathbb{R}^{n \times m}$  is a *linear mapping*  $\mathbf{A} : \mathbb{R}^m \to \mathbb{R}^n$ , defined by  $\mathbf{x} \mapsto \mathbf{y} = \mathbf{A}\mathbf{x}$ .

• The range of **A** is defined as

range  $\mathbf{A} = {\mathbf{A}\mathbf{x} \in \mathbb{R}^n | \mathbf{x} \in \mathbb{R}^m} \subseteq \mathbb{R}^n$ . If range  $\mathbf{A} = \mathbb{R}^n$ , the mapping  $\mathbf{A}$  is a *surjection*. *(existence)*  $\mathbf{A}\mathbf{x} = \mathbf{y}$  is solvable iff  $\mathbf{y} \in \text{range } \mathbf{A}$ .

• The null-space (or kernel) of A is defined as

$$\begin{split} & \ker \mathbf{A} = \{\mathbf{x} \in \mathbb{R}^m | \mathbf{A}\mathbf{x} = \mathbf{0}\}.\\ & \text{If } \ker \mathbf{A} = \{\mathbf{0}\}, \text{ then the mapping } \mathbf{A} \text{ is an injection.}\\ & (\textit{uniqueness}) \text{ If } \mathbf{x} \text{ is a solution to } \mathbf{A}\mathbf{x} = \mathbf{y},\\ & \text{then } \mathbf{x} + \tilde{\mathbf{x}} \text{ for each } \tilde{\mathbf{x}} \in \ker \mathbf{A} \text{ is also a solution.} \end{split}$$

## Matrix rank

The surjectivity and injectivity of the matrix  $\mathbf{A}$  can be conveniently expressed in terms of its *rank*, which can be defined as

```
rank \mathbf{A} = \dim \operatorname{range} \mathbf{A} \leq \min(m, n).
```

- A is called full-rank, if it has the largest possible rank. Otherwise it is said to be singular or rank-deficient (which in the exact arithmetic are synonyms).
- The rank of the matrix **A** equals the number of its linearly independent columns (or, equivalently, rows).

The dimensionality m of the domain of **A** is split into the dimensionality of its range and of its null-space,

 $m = \operatorname{rank} \mathbf{A} + \dim \ker \mathbf{A}.$ 

- A is surjective, iff  $n = \operatorname{rank} \mathbf{A}$  (requires  $n \le m$ ).
- A is injective, iff  $m = \operatorname{rank} A$  (requires  $m \le n$ ).
- A is bijective, iff  $m = n = \operatorname{rank} A$  (full-rank square A).

#### Existence and uniqueness of solution

Outline

Consider a linear equation Ax = y, where  $A \in \mathbb{R}^{n \times m}$ . Depending on the surjectivity and injectivity of A, four general cases are possible:

- A is a bijection  $(n = m = \text{rank } \mathbf{A}$ , full-rank square A). The equation has a unique solution.
- A is surjective, but not injective (n = rank A < m). For each y there are infinitely many solutions, which can be expressed as x<sub>p</sub> + x<sub>0</sub>, where x<sub>p</sub> is a particular solution and x<sub>0</sub> ∈ ker A ≠ {0}.
- A is injective, but not surjective (m = rank A < n). Depending on y, there is either a unique solution (if y ∈ range A) or no solution at all.
- A is neither injective, nor surjective (rank A < min(m, n)). Depending on y, there are either infinitely many solutions (if y ∈ range A) or no solutions at all.

#### Outline



#### O Direct methods

- Special matrices
- Factorizations and decompositions
- Gaussian elimination

## Direct methods

Direct methods

- Compute the solution in a finite and known in advance time (number of steps).
- In the exact arithmetic, the computed solution would be exact.
- Stability in finite precision arithmetics are usually well-analyzed.
- Require direct access to the elements of the system matrix **A** (which has thus to be explicitly given).

#### Direct methods

Direct methods solve Ax = y in two general steps:

- Factorize A into a product of two or more matrices (e.g. A = Q<sub>1</sub>Q<sub>2</sub>Q<sub>3</sub>), such that Q<sub>i</sub>v = u are all *easily solvable*. The original equation is then expressed as e.g. Q<sub>1</sub>(Q<sub>2</sub>(Q<sub>3</sub>x)) = y.
- Compute the solution x by successively solving the resulting equations, e.g.

 $\begin{array}{l} {\bf Q}_1 {\bf x}_1 = {\bf y} \\ {\bf Q}_2 {\bf x}_2 = {\bf x}_1 \\ {\bf Q}_3 {\bf x}_3 = {\bf x}_2, \end{array}$ 

so that finally  $\mathbf{x} = \mathbf{x}_3$ .

The numerical cost of the first step (for a square  $n \times n$  matrix) is usually  $O(n^3)$  and much larger then that of the second step  $O(n^2)$ . If several equations with the same **A** and different right-hand sides **y** have to be solved, **A** is factorized only once. In the first step, a direct method of solving Ax = y factorizes A into a product of two or more *special matrices*  $Q_1, Q_2, \ldots, Q_N$ .

The matrices are called special, since  $\mathbf{Q}_i \mathbf{v} = \mathbf{u}$  have all to be *easily solvable*. They are usually:

- diagonal,
- unitary or orthonormal,
- permutation matrices,
- lower or upper triangular.

#### Square diagonal matrices

$$\mathbf{D} = \operatorname{diag}(d_1, d_2, \dots, d_n) = \begin{bmatrix} d_1 & & \mathbf{0} \\ & d_2 & \\ & & \ddots & \\ \mathbf{0} & & & d_n \end{bmatrix}$$

- Elements of a diagonal matrix are all zero except the diagonal.
- If d<sub>i</sub> ≠ 0 for all i, then D is full-rank and bijective. The system Dx = y is then uniquely solvable for all y.
- If  $d_i = 0$  for some *i*, then **D** is singular and neither surjective nor bijective. Depending on **y**, the system **Dx** = **y** has either infinitely many solutions or no solutions at all.

HW7

#### Square diagonal matrices

Existence & uniqueness

Outline

Basics

The linear system  $\mathbf{D}\mathbf{x} = \mathbf{y}$ , where  $\mathbf{D}$  is an  $n \times n$  diagonal matrix, is a system of *n* decoupled equations:

Direct methods

$$d_i x_i = y_i, \qquad i = 1, 2, \ldots, n,$$

which can be solved in O(n) operations:

- If all  $d_i \neq 0$ , then the unique solution is given by  $x_i = y_i/d_i$ .
- Otherwise there is *i* such that d<sub>i</sub> = 0. The matrix D (and thus also A) is singular and neither surjective nor injective. Existence of solution depends on y:
  - If y<sub>i</sub> = 0 for all i such that d<sub>i</sub> = 0, then there are *infinitely* many solutions, since equation 0x<sub>i</sub> = 0 is satisfied by any x<sub>i</sub>.
  - If there exists *i* such that  $d_i = 0$  and  $y_i \neq 0$ , then there are *no* solutions, since no  $x_i$  can satisfy  $0x_i = y_i$ , where  $y_i \neq 0$ .

The case of a non-square diagonal matrix can be treated in a similar way.

Iterative methods

Outline

#### Diagonal matrices — examples

#### Square diagonal matrix

$$\begin{bmatrix} 1 & 0 & 0 & 0 \\ 0 & 2 & 0 & 0 \\ 0 & 0 & 0 & 0 \\ 0 & 0 & 0 & 4 \end{bmatrix} \begin{bmatrix} x_1 \\ x_2 \\ x_3 \\ x_4 \end{bmatrix} = \begin{bmatrix} 1 \\ 1 \\ y \\ 1 \end{bmatrix}$$

If y == 0, then the equation has infinitely many solutions:

$$\begin{bmatrix} x_1 \\ x_2 \\ x_3 \\ x_4 \end{bmatrix} = \begin{bmatrix} 1 \\ 1/2 \\ 0 \\ 1/4 \end{bmatrix} + \begin{bmatrix} 0 \\ 0 \\ c \\ 0 \end{bmatrix}, \qquad c \in \mathbb{R},$$

where  $\begin{bmatrix} 1 & 1/2 & 0 & 1/3 \end{bmatrix}^T$  is a particular solution and  $\begin{bmatrix} 0 & 0 & c & 0 \end{bmatrix}^T$  belongs to the null space of the system matrix. Otherwise (if  $y \neq 0$ ), the equation nas no solutions.

Outline

#### Diagonal matrices — examples

#### Non-square diagonal matrix

The equation

$$\begin{bmatrix} 1 & 0 & 0 & 0 \\ 0 & 2 & 0 & 0 \\ 0 & 0 & 3 & 0 \end{bmatrix} \begin{bmatrix} x_1 \\ x_2 \\ x_3 \\ x_4 \end{bmatrix} = \begin{bmatrix} 1 \\ 1 \\ 1 \end{bmatrix}$$

has infinitely many solutions,

$$\begin{bmatrix} x_1 \\ x_2 \\ x_3 \\ x_4 \end{bmatrix} = \begin{bmatrix} 1 \\ 1/2 \\ 1/3 \\ 0 \end{bmatrix} + \begin{bmatrix} 0 \\ 0 \\ 0 \\ c \end{bmatrix}, \qquad c \in \mathbb{R},$$

where  $\begin{bmatrix} 1 & 1/2 & 1/3 & 0 \end{bmatrix}^T$  is a particular solution and  $\begin{bmatrix} 0 & 0 & 0 & c \end{bmatrix}^T$  belongs to the null space of the system matrix.

#### Diagonal matrices — examples

#### Non-square diagonal matrix

$$\begin{bmatrix} 1 & 0 & 0 \\ 0 & 2 & 0 \\ 0 & 0 & 3 \\ 0 & 0 & 0 \end{bmatrix} \begin{bmatrix} x_1 \\ x_2 \\ x_3 \end{bmatrix} = \begin{bmatrix} 1 \\ 1 \\ 1 \\ y \end{bmatrix}$$

If y == 0, the equation has a unique solution

$$\begin{bmatrix} x_1 \\ x_2 \\ x_3 \end{bmatrix} = \begin{bmatrix} 1 \\ 1/2 \\ 1/3 \end{bmatrix}$$

Otherwise, the equation has no solutions.

## Unitary matrices

A real square matrix  $\mathbf{Q}$  is called unitary, if  $\mathbf{Q}^{-1} = \mathbf{Q}^{\mathsf{T}}$ .

If a matrix  $\mathbf{Q} \in \mathbb{R}^{n imes n}$  is unitary, then

• Q has orthonormal rows and columns, i.e.

$$\mathbf{Q}^{\mathsf{T}}\mathbf{Q}=\mathbf{Q}\mathbf{Q}^{\mathsf{T}}=\mathbf{I},$$

where I is the *n* by *n* diagonal matrix.

• Columns  $\mathbf{q}_1, \mathbf{q}_2, \dots, \mathbf{q}_n$  (and rows) of  $\mathbf{Q}$  form an orthonormal basis in  $\mathbf{R}^n$ , that is

$$\mathbf{q}_i^{\mathsf{T}}\mathbf{q}_j = \delta_{ij},$$

where  $\delta_{ij}$  is Kronecker's delta.

• Unitary matrices are thus always full-rank.

#### Unitary and orthonormal matrices

Since unitary matrices are always full-rank and easily invertible  $(\mathbf{Q}^{-1} = \mathbf{Q}^{\mathsf{T}})$ , a linear system  $\mathbf{Q}\mathbf{x} = \mathbf{y}$  with a unitary matrix  $\mathbf{Q} \in \mathbb{R}^{n \times n}$  has a unique solution for all  $\mathbf{y}$  and can be solved in  $O(n^2)$  operations,

 $\mathbf{x} = \mathbf{Q}^{-1} \mathbf{y} = \mathbf{Q}^{\mathsf{T}} \mathbf{y}.$ 

A unitary matrix with a part of rows (or columns) removed is called an orthonormal matrix. The removed (or, more often, not computed at all) vectors usually form a basis of the null-space of the considered system matrix. They can be thus disregarded, if only a particular solution is sought for instead of the full solution space. The particular solution obtained this way is usually the minimum-norm solution.

#### Permutation matrices

A square  $n \times n$  matrix  $\Pi$  is said to be a permutation matrix, if it is obtained from an  $n \times n$  identity matrix by permuting its rows.

- Every row and every column of a permutation matrix has exactly single 1 and everywhere else 0s.
- There are *n*! different permutations of an *n*-element sequence. So, there are exactly *n*! permutation matrices of the dimensions *n* × *n*.
- A permutation matrix Π satisfies Π<sup>T</sup>Π = I, therefore it is a special case of a unitary matrix.
- When applied to an  $n \times n$  matrix **A**:
  - $\Pi A$  is the matrix A with permuted rows.
  - $A\Pi$  is the matrix A with permuted columns.

#### Lower and upper triangular matrices

Outline

A square matrix **L** is called a lower triangular matrix, if all its elements above the main diagonal are zero:  $l_{ij} = 0$  for i < j.

A square matrix **U** is called an upper triangular matrix, if all its elements below the main diagonal are zero:  $u_{ij} = 0$  for i > j.

$$\mathbf{L} = \begin{bmatrix} l_{11} & & \mathbf{0} \\ l_{21} & l_{22} & & \\ \vdots & \vdots & \ddots & \\ l_{n1} & l_{n2} & \cdots & l_{nn} \end{bmatrix}$$

$$\mathbf{U} = \begin{bmatrix} u_{11} & u_{12} & \cdots & u_{1n} \\ & u_{22} & \cdots & u_{2n} \\ & & \ddots & \vdots \\ \mathbf{0} & & & u_{nn} \end{bmatrix}$$

HW7

#### Lower triangular systems — forward-substitution

An  $n \times n$  lower triangular system  $\mathbf{L}\mathbf{x} = \mathbf{y}$ ,

$$\begin{bmatrix} I_{11} & \mathbf{0} \\ I_{21} & I_{22} \\ \vdots & \vdots & \ddots \\ I_{n1} & I_{n2} & \cdots & I_{nn} \end{bmatrix} \begin{bmatrix} x_1 \\ x_2 \\ \vdots \\ x_n \end{bmatrix} = \begin{bmatrix} y_1 \\ y_2 \\ \vdots \\ y_n \end{bmatrix},$$

can be solved with  $O(n^2)$  operations by forward-substitution.



#### Upper triangular systems — back-substitution

Similarly, an  $n \times n$  upper triangular system  $\mathbf{U}\mathbf{x} = \mathbf{y}$ ,

$$\begin{bmatrix} u_{11} & u_{12} & \cdots & u_{1n} \\ & u_{22} & \cdots & u_{2n} \\ & & \ddots & \vdots \\ 0 & & & u_{nn} \end{bmatrix} \begin{bmatrix} x_1 \\ x_2 \\ \vdots \\ x_n \end{bmatrix} = \begin{bmatrix} y_1 \\ y_2 \\ \vdots \\ y_n \end{bmatrix},$$

can be solved with  $O(n^2)$  operations by back-substitution.

Back substitution  $x_n = \frac{y_n}{u_{11}},$   $x_i = \frac{y_i - \sum_{j=i+1}^n u_{ij}x_j}{u_{ii}}.$ 

#### Factorizations and decompositions

Solution of the system Ax = y by direct methods requires a factorization of A into a product of two or more special matrices that make the system easier to solve, e.g.

$$\mathbf{A} = \mathbf{Q}_1 \mathbf{Q}_2 \qquad \text{or} \qquad \mathbf{A} = \mathbf{Q}_1 \mathbf{Q}_2 \mathbf{Q}_3.$$

Direct methods can be broadly classified into two groups

- Decomposition methods use a factorization with unitary matrices and (usually) a diagonal matrix, which directly provide important information about the matrix and the related mapping (dimensionality and basis of the null-space, eigen- or singular values, etc.). The factorization (decomposition) often amounts to solving the related eigenproblem.
- The other direct methods are known under the general name of factorization methods.

#### Factorizations

The probably most commonly used factorizations are

- LU factorization: **A** = LU, where L and U are respectively lower and upper triangular matrices. It exists for any square nonsingular matrix<sup>3</sup>.
- QR factorization: A = QR, where Q is a unitary (or orthogonal) matrix and R is an upper triangular matrix. Similarly, there exist QL, RQ and LQ factorizations.
- LDL factorization: A = LDL<sup>T</sup>, where L is a lower triangular matrix and D is a diagonal matrix with positive elements. LDL factorization exists for symmetric positive definite matrices (for other matrices it may not exist).
- Cholesky factorization: A = LL<sup>T</sup>, where L is a lower triangular matrix. Cholesky factorization exists only for symmetric positive definite matrices.

<sup>3</sup>Sometimes a pre-multiplication by a permutation matrix  $\Pi$  is necessary, so that  $\Pi A = LU$ .

#### Decompositions — eigen decomposition Eigenvalues and eigenvectors

A fundamental notion in linear algebra is that of an eigenvalue and the corresponding eigenvectors of a square matrix.

Let **A** be a square  $n \times n$  matrix. A number  $\lambda$  is called an eigenvalue of **A** and a vector **v** is called a corresponding eigenvector if and only if  $\mathbf{Av} = \lambda \mathbf{v}$ .

As the above condition yields  $(\mathbf{A} - \lambda \mathbf{I}) \mathbf{v} = \mathbf{0}$ , the eigenvalues of  $\mathbf{A}$  are the roots of its characteristic polynomial,

$$f_{\mathbf{A}}(\lambda) = \det (\mathbf{A} - \lambda \mathbf{I}),$$

which always has *n* complex roots. Thus, every square  $n \times n$  matrix has always *n* complex eigenvalues (some or all of which can be real). Every eigenvalue has a multiplicity, which is defined as the multiplicity of the corresponding root of  $f_A(\lambda)$ .

#### Decompositions — eigen decomposition Symmetric (Hermitian) matrices

Outline

The probably most known decomposition is the eigen decomposition of a symmetric (Hermitian) matrix:

Every square  $n \times n$  symmetric (Hermitian) matrix **A** has n linearly independent eigenvectors and can be expressed as

#### $\mathbf{A} = \mathbf{P}\mathbf{D}\mathbf{P}^{-1},$

where **D** is a diagonal matrix with the eigenvalues of **A** on the diagonal,  $\mathbf{D} = \text{diag}(\lambda_1, \lambda_2, \dots, \lambda_n)$ , and **P** is the matrix with the corresponding eigenvectors as columns. The eigenvalues are all real. The eigenvectors are orthogonal, and if they are scaled to be orthonormal, then  $\mathbf{P}^{-1} = \mathbf{P}^T$ and

$$\mathbf{A} = \mathbf{P}\mathbf{D}\mathbf{P}^{\mathsf{T}}.$$

#### Decompositions — eigen decomposition

If **A** is non-symmetric (non-Hermitian), the existence of the eigen decomposition depends on the number of the eigenvectors:

If a square  $n \times n$  matrix **A** has n linearly independent eigenvectors, than it can be expressed as

#### $\mathbf{A}=\mathbf{P}\mathbf{D}\mathbf{P}^{-1},$

where **P** collects all the eigenvectors as columns, and **D** is a diagonal matrix with the corresponding eigenvalues on the diagonal,  $\mathbf{D} = \text{diag}(\lambda_1, \lambda_2, \dots, \lambda_n)$ .

If A

has an eigen decomposition, it is called *diagonalizable*<sup>4</sup>. A non-symmetric (non-Hermitian) matrix may have complex eigenvalues.

<sup>&</sup>lt;sup>4</sup>It is diagonal in the coordinates defined by the columns of **P**.

#### Decompositions — eigen decomposition — examples

#### Symmetric matrix

The matrix

$$\mathbf{A} = \begin{bmatrix} 1 & -1 \\ -1 & 1 \end{bmatrix}, \qquad f_{\mathbf{A}}(\lambda) = \lambda^2 - 2\lambda,$$

is symmetric, thus it has two linearly independent orthogonal eigenvectors with two corresponding real eigenvalues,

$$\mathbf{v}_1 = [-1 \ 1]^{\mathsf{T}}$$
  $\mathbf{v}_2 = [1 \ 1]^{\mathsf{T}}$ ,  
 $\lambda_1 = 2$   $\lambda_2 = 0.$ 

One of the eigenvalues is zero, so **A** is singular.

$$\mathbf{A} = \begin{bmatrix} 1 & -1 \\ -1 & 1 \end{bmatrix} = \begin{bmatrix} -1 & 1 \\ 1 & 1 \end{bmatrix} \begin{bmatrix} 2 & 0 \\ 0 & 0 \end{bmatrix} \begin{bmatrix} -0.5 & 0.5 \\ 0.5 & 0.5 \end{bmatrix}$$

4/62

#### Decompositions — eigen decomposition — examples

#### Non-symmetric diagonalizable matrix

The non-symmetric matrix

$$\mathbf{A} = \begin{bmatrix} 1 & 1 \\ -1 & 1 \end{bmatrix}, \qquad f_{\mathbf{A}}(\lambda) = \lambda^2 - 2\lambda + 2,$$

is diagonalizable, since it has two linearly independent eigenvectors,

$$\mathbf{v}_1 = \begin{bmatrix} -i \ 1 \end{bmatrix}^\mathsf{T}$$
  $\mathbf{v}_2 = \begin{bmatrix} i \ 1 \end{bmatrix}^\mathsf{T}$ .

The two corresponding eigenvalues are complex:

$$\lambda_1 = 1 + i \qquad \qquad \lambda_2 = 1 - i.$$

$$\mathbf{A} = \begin{bmatrix} 1 & 1 \\ -1 & 1 \end{bmatrix} = \begin{bmatrix} -\mathbf{i} & \mathbf{i} \\ 1 & 1 \end{bmatrix} \begin{bmatrix} 1+\mathbf{i} & 0 \\ 0 & 1-\mathbf{i} \end{bmatrix} \begin{bmatrix} 0.5\mathbf{i} & 0.5 \\ -0.5\mathbf{i} & 0.5 \end{bmatrix}$$

4/62

#### Decompositions — eigen decomposition — examples

Non-symmetric non-diagonalizable matrix

The non-symmetric matrix

$$\mathbf{A} = \left[ egin{array}{cc} 1 & 0 \ 1 & 1 \end{array} 
ight], \qquad \qquad f_{\mathbf{A}}(\lambda) = \lambda^2 - 2\lambda + 1,$$

is non-diagonalizable, since it has only one linearly independent eigenvector,

$$\mathbf{v}_1 = \begin{bmatrix} 0 & 1 \end{bmatrix}^\mathsf{T},$$

even if its single eigenvalue has the multiplicity of two:

$$\lambda_1 = \lambda_2 = 1.$$

The matrix **A** has no eigen decomposition and is non-singular.

#### Decompositions — singular value decomposition

The probably most important decomposition is the singular value decomposition (SVD):

Every rectangular matrix  $\mathbf{A} \in \mathbb{R}^{n \times m}$  can be expressed as

$$\mathbf{A} = \mathbf{U} \left[ \begin{array}{cc} \boldsymbol{\Sigma}_0 & \mathbf{0} \\ \mathbf{0} & \mathbf{0} \end{array} \right] \mathbf{V}^{\mathcal{T}} = \mathbf{U} \boldsymbol{\Sigma} \mathbf{V}^{\mathcal{T}},$$

where **U** and **V** are unitary matrices, and  $\Sigma$  is an  $n \times m$  diagonal matrix with nonnegative diagonal elements ordered in a nonincreasing way.

#### Decompositions — singular value decomposition

Every rectangular matrix  $\mathbf{A} \in \mathbb{R}^{n \times m}$  can be expressed as

$$\mathbf{A} = \mathbf{U} \begin{bmatrix} \mathbf{\Sigma}_0 & \mathbf{0} \\ \mathbf{0} & \mathbf{0} \end{bmatrix} \mathbf{V}^\mathsf{T} = \mathbf{U} \mathbf{\Sigma} \mathbf{V}^\mathsf{T},$$

where  $\boldsymbol{U}$  and  $\boldsymbol{V}$  are unitary, and  $\boldsymbol{\Sigma}$  is a diagonal matrix.



#### Decompositions — singular value decomposition

Every rectangular matrix  $\mathbf{A} \in \mathbb{R}^{n \times m}$  can be expressed as

$$\mathbf{A} = \mathbf{U} \begin{bmatrix} \mathbf{\Sigma}_0 & \mathbf{0} \\ \mathbf{0} & \mathbf{0} \end{bmatrix} \mathbf{V}^\mathsf{T} = \mathbf{U} \mathbf{\Sigma} \mathbf{V}^\mathsf{T},$$

where  $\boldsymbol{U}$  and  $\boldsymbol{V}$  are unitary, and  $\boldsymbol{\Sigma}$  is a diagonal matrix.

- Diagonal elements  $\sigma_i$  of  $\Sigma$  are called the singular values of **A**.
- The number r of positive singular values equals to rank A. The matrix Σ<sub>0</sub> is thus rank A × rank A.
- The SVD is unique up to the ordering of the singular vectors (columns of **U** and **V**) corresponding to equal singular values.
- The columns of **V**, which correspond to the vanishing singular values, form a basis for the null-space of **A**.
- $\mathbf{A}^{\mathsf{T}}\mathbf{A} = \mathbf{V}\mathbf{\Sigma}^{\mathsf{T}}\mathbf{U}^{\mathsf{T}}\mathbf{U}\mathbf{\Sigma}\mathbf{V}^{\mathsf{T}} = \mathbf{V}\mathbf{\Sigma}^{\mathsf{T}}\mathbf{\Sigma}\mathbf{V}^{\mathsf{T}}$  is the eigen decomposition of  $\mathbf{A}^{\mathsf{T}}\mathbf{A}$ . Therefore,  $\sigma_i^2(\mathbf{A}) = \lambda_i(\mathbf{A}^{\mathsf{T}}\mathbf{A})$ .
- The singular values provide full information about conditioning of **A** (see Lecture B-4).

# Decompositions — singular value decomposition Examples

#### Non-symmetric non-diagonalizable square matrix

The non-symmetric and non-diagonalizable matrix

$$\mathbf{A} = \begin{bmatrix} 2 & 0 \\ 3 & 2 \end{bmatrix}$$

has the following singular value decomposition:

$$\mathbf{A} = \begin{bmatrix} \frac{1}{\sqrt{5}} & -\frac{2}{\sqrt{5}} \\ \frac{2}{\sqrt{5}} & \frac{1}{\sqrt{5}} \end{bmatrix} \begin{bmatrix} 4 & 0 \\ 0 & 1 \end{bmatrix} \begin{bmatrix} \frac{2}{\sqrt{5}} & -\frac{1}{\sqrt{5}} \\ \frac{1}{\sqrt{5}} & \frac{2}{\sqrt{5}} \end{bmatrix}^{\mathsf{T}}$$

# Decompositions — singular value decomposition Examples

#### Non-square matrix

The non-square matrix

$$\mathbf{A} = \left[ \begin{array}{rrr} 2 & 0 & 0 \\ 3 & 2 & 0 \end{array} \right]$$

has the following singular value decomposition:

$$\mathbf{A} = \begin{bmatrix} \frac{1}{\sqrt{5}} & -\frac{2}{\sqrt{5}} \\ \frac{2}{\sqrt{5}} & \frac{1}{\sqrt{5}} \end{bmatrix} \begin{bmatrix} 4 & 0 & 0 \\ 0 & 1 & 0 \end{bmatrix} \begin{bmatrix} \frac{2}{\sqrt{5}} & -\frac{1}{\sqrt{5}} & 0 \\ \frac{1}{\sqrt{5}} & \frac{2}{\sqrt{5}} & 0 \\ 0 & 0 & 1 \end{bmatrix}^{T}$$

#### Decompositions — rank-revealing QR factorization

Another decomposition is called the rank-revealing QR factorization:

Let the rank of  $\mathbf{A} \in \mathbb{R}^{n \times m}$  be  $r = \operatorname{rank} \mathbf{A} \leq \min(n, m)$ . There exists a rank-revealing QR factorization,

$$\mathbf{A}\mathbf{\Pi} = \left[ \begin{array}{cc} \mathbf{Q}_1 & \mathbf{Q}_2 \end{array} \right] \left[ \begin{array}{cc} \mathbf{R}_1 & \mathbf{R}_2 \\ \mathbf{0} & \mathbf{0} \end{array} \right].$$

where  $\mathbf{R}_1$  is an  $r \times r$  upper triangular matrix with positive diagonal elements,  $\mathbf{R}_2$  is an  $r \times (m - r)$  matrix and  $\Pi$  is a permutation matrix such that the first r columns of  $\mathbf{A}\Pi$ are linearly independent. The matrix  $[\mathbf{Q}_1\mathbf{Q}_2]$  is unitary. The

matrices  $\mathbf{Q}_1$  and  $\mathbf{Q}_2$  are (column) orthonormal:

- The r columns of  $\mathbf{Q}_1$  form a basis for range  $\mathbf{A}\mathbf{\Pi}$ .
- The n r columns of  $\mathbf{Q}_2$  form a basis for ker  $(\mathbf{A}\mathbf{\Pi})^{\mathsf{T}}$ .

#### Gaussian elimination

#### Gaussian elimination is a method of

- solving a full-rank  $\mathbf{A}\mathbf{x} = \mathbf{y}$  by
- performing the LU factorization of **A**.

Gaussian elimination uses two elementary operations

- adding a multiple of the *i*th row to the *j*th row and
- interchanging two rows/equations (or columns/unknowns), called *pivoting*

to eliminate the unknowns  $x_i$  in order to obtain an equivalent upper triangular system  $\mathbf{U}\mathbf{x} = \hat{\mathbf{y}}$ , which can be solved by back-substitution.

#### Gaussian elimination

$$\mathbf{A}\mathbf{x} = \begin{bmatrix} a_{11} & a_{12} & \cdots & a_{1n} \\ a_{21} & a_{22} & \cdots & a_{2n} \\ \vdots & \vdots & \ddots & \vdots \\ a_{n1} & a_{n2} & \cdots & a_{nn} \end{bmatrix} \begin{bmatrix} x_1 \\ x_2 \\ \vdots \\ x_n \end{bmatrix} = \begin{bmatrix} y_1 \\ y_2 \\ \vdots \\ y_n \end{bmatrix} = \mathbf{y}$$

First,  $x_1$  is eliminated from the other equations by subtracting the multiple  $l_{i1} = a_{i1}/a_{11}$  of the first equation from the *i*th equation. This way a reduced system of n-1 equations with n-1 unknowns  $x_2, \ldots, x_n$  is obtained,

$$\mathbf{A}^{(2)}\mathbf{x} = \begin{bmatrix} a_{11} & a_{12} & \cdots & a_{1n} \\ 0 & a_{22}^{(2)} & \cdots & a_{2n}^{(2)} \\ \vdots & \vdots & \ddots & \vdots \\ 0 & a_{n2}^{(2)} & \cdots & a_{nn}^{(2)} \end{bmatrix} \begin{bmatrix} x_1 \\ x_2 \\ \vdots \\ x_n \end{bmatrix} = \begin{bmatrix} y_1 \\ y_2^{(2)} \\ \vdots \\ y_n^{(2)} \end{bmatrix} = \mathbf{y}^{(2)},$$
  
where  $a_{ij}^{(2)} = a_{ij} - l_{i1}a_{i1}$  and  $y_i^{(2)} = b_i - l_{i1}b_1.$ 

41/62

#### Gaussian elimination

w

The procedure is repeated n-1 times:  $x_k$ , k = 2, ..., n-1 is eliminated from the rest i = k + 1, ..., n equations using the multiplier  $l_{ik} = a_{ik}^{(k)}/a_{kk}^{(k)}$ . This yields an upper triangular system, which can be solved by back-substitution:

$$\mathbf{A}^{(n)}\mathbf{x} = \begin{bmatrix} a_{11} & a_{12} & \cdots & a_{1n} \\ & a_{22}^{(2)} & \cdots & a_{2n}^{(2)} \\ & & \ddots & \vdots \\ \mathbf{0} & & & a_{nn}^{(n)} \end{bmatrix} \begin{bmatrix} x_1 \\ x_2 \\ \vdots \\ x_n \end{bmatrix} = \begin{bmatrix} y_1 \\ y_2^{(2)} \\ \vdots \\ y_n^{(n)} \end{bmatrix} = \mathbf{y}^{(n)},$$
  
here  $a_{ii}^{(k+1)} = a_{ii}^{(k)} - l_{ik}a_{ik}^{(k)}$  and  $y_i^{(k+1)} = b_i^{(k)} - l_{ik}b_k^{(k)}.$ 

#### Gaussian elimination with pivoting

Gaussian elimination uses in the *k*th step the multiplier  $I_{ik} = a_{ik}^{(k)} / a_{kk}^{(k)}$  to eliminate the unknown  $x_k$ . This works seamlessly, unless the diagonal element  $a_{kk}^{(k)} \approx 0$ .

- If  $a_{kk}^{(k)} = 0$ , elimination is not possible.
- If  $a_{kk}^{(k)}$  is very small, elimination can be numerically unstable.

The second elementary operation (row pivoting) can be then used. The kth row is interchanged with one of the next rows, so that the diagonal element is maximized. Some procedures pivot also columns (interchange the unknowns). Before performing the Gaussian elimination with pivoting, the matrix should be rescaled

- The maximum magnitude of the elements in each row to one.
- The sum of the magnitudes of the row elements to one.

There is no need for pivoting, if **A** is symmetric positive definite.

#### Gaussian elimination with pivoting — example

Example (Dahlquist and Björck)

$$\mathbf{A}\mathbf{x} = \left[ \begin{array}{cc} 10^{-6} & 1 \\ 1 & 1 \end{array} \right] \left[ \begin{array}{c} x_1 \\ x_2 \end{array} \right] = \left[ \begin{array}{c} 1 \\ 0 \end{array} \right] = \mathbf{y}.$$

A is nonsingular and well-conditioned. The exact solution is

$$\mathbf{x} = rac{1}{1-10^{-6}} \left[ egin{array}{c} -1 \ 1 \end{array} 
ight] pprox \left[ egin{array}{c} -1 \ 1 \end{array} 
ight]$$

However, the solution computed by Gaussian elimination *without pivoting* in the precision of 6 decimal digits yields

$$\mathbf{x} = \left[ egin{array}{c} 0 \ 1 \end{array} 
ight]$$

٠

#### Gaussian elimination — numerical costs

After all steps an upper triangular system is produced,

$$\mathbf{A}^{(n)}\mathbf{x} = \begin{bmatrix} a_{11} & a_{12} & \cdots & a_{1n} \\ & a_{22}^{(2)} & \cdots & a_{2n}^{(2)} \\ & & \ddots & \vdots \\ \mathbf{0} & & & a_{nn}^{(n)} \end{bmatrix} \begin{bmatrix} x_1 \\ x_2 \\ \vdots \\ x_n \end{bmatrix} = \begin{bmatrix} y_1 \\ y_2^{(2)} \\ \vdots \\ y_n^{(n)} \end{bmatrix} = \mathbf{y}^{(n)}.$$

The number of operations in Gaussian elimination is  $\sim n^3/3$ . This is substantially more than  $\sim n^2/2$  necessary to solve the resulting upper triangular system.

If the multipliers  $l_{ik}$  are stored (together with the information about row interchange and scaling, if necessary), then  $\mathbf{y}^{(n)}$  for different right-hand side vectors  $\mathbf{y}$  can be computed at later times at the cost of  $\sim n^2/2$  only. Hence the total cost of each subsequent computation would be  $\sim n^2$  only.

#### Gaussian elimination — numerical costs

After all steps an upper triangular system is produced,

$$\mathbf{A}^{(n)}\mathbf{x} = \begin{bmatrix} a_{11} & a_{12} & \cdots & a_{1n} \\ & a_{22}^{(2)} & \cdots & a_{2n}^{(2)} \\ & & \ddots & \vdots \\ \mathbf{0} & & & a_{nn}^{(n)} \end{bmatrix} \begin{bmatrix} x_1 \\ x_2 \\ \vdots \\ x_n \end{bmatrix} = \begin{bmatrix} y_1 \\ y_2^{(2)} \\ \vdots \\ y_n^{(n)} \end{bmatrix} = \mathbf{y}^{(n)}.$$

To save the memory, the multipliers  $I_{ik}$  can be stored for later use in the lower part of the matrix  $\mathbf{A}^{(k)}$  (in the place of the zeroed elements),

$$\begin{bmatrix} a_{11} & a_{12} & \cdots & a_{1n} \\ l_{21} & a_{22}^{(2)} & \cdots & a_{2n}^{(2)} \\ \vdots & \vdots & \ddots & \vdots \\ l_{n1} & l_{n2} & \cdots & a_{nn}^{(n)} \end{bmatrix}$$

#### Gaussian elimination — LU factorization

Gaussian elimination is an algorithm for LU factorization of the system matrix,  ${\bf A}={\bf L}{\bf U},$  where:

• 
$$\mathbf{L} = [I_{ik}]$$
, where  $I_{kk} = 1$  and  $I_{ik} = 0$  for  $i < k$ .  
•  $\mathbf{U} = [u_{kj}]$ , where  $u_{kj} = a_{kj}^{(k)}$  for  $j \ge k$  and  $u_{kj} = 0$  otherwise

$$\mathbf{A}^{(n)} \rightarrow \begin{bmatrix} a_{11} & a_{12} & \cdots & a_{1n} \\ l_{21} & a_{22}^{(2)} & \cdots & a_{2n}^{(2)} \\ \vdots & \vdots & \ddots & \vdots \\ l_{n1} & l_{n2} & \cdots & a_{nn}^{(n)} \end{bmatrix} \\ \rightarrow \begin{bmatrix} 1 & & \mathbf{0} \\ l_{21} & 1 & & \\ \vdots & \vdots & \ddots & \\ l_{n1} & l_{n2} & \cdots & 1 \end{bmatrix} \begin{bmatrix} a_{11} & a_{12} & \cdots & a_{1n} \\ & a_{22}^{(2)} & \cdots & a_{2n}^{(2)} \\ & & \ddots & \vdots \\ \mathbf{0} & & & a_{nn}^{(n)} \end{bmatrix}$$

#### Gaussian elimination — LU factorization

Gaussian elimination yields LU factorization of the system matrix. Therefore, it is one of the direct methods, as it computes the solution of a linear system Ax = y in two steps

- Single LU factorization ΠA = LU, where Π is the permutation and row rescaling matrix (if necessary). This step requires ~ n<sup>3</sup>/3 operations.
- Solution of  $(\Pi^{-1}LU) \mathbf{x} = \mathbf{y}$  via the solution of the equivalent systems  $\mathbf{L}\mathbf{x}_1 = \Pi \mathbf{y}$  and  $\mathbf{U}\mathbf{x} = \mathbf{x}_1$ , where  $\Pi \mathbf{y}$  is the rescaled and pivoted  $\mathbf{y}$ .

The second step costs only  $\sim n^2$  operations and can be repeated several times for different right-hand side vectors **y**.





- Stationary methods
- Krylov subspace methods

#### Iterative methods

Iterative methods

- Compute an approximate solution in an iterative way.
- In general, the number of iterations to obtain a good approximation is unknown in advance.
- The convergence properties of the methods are often hard to analyze, especially in finite precision arithmetics. In some cases, finite-precision iterations may not converge at all.
- Many of the methods require access only to a matrix-vector multiplication procedure (Ax, sometimes also A<sup>T</sup>x), so that A may be given only implicitly (e.g. Ax can be the result of an experiment or a simulation).
- They are most useful for solving large sparse or structured systems, for which any factorization (necessary in direct methods) would take too much time, destroy the sparsity or the matrix structure, or be too inaccurate.

#### Iterative methods

There are two important groups of iterative methods

- Stationary methods: Jacobi, Gauss-Seidel, successive over-relaxation (SOR), etc.
- Krylov subspace methods, out of which the conjugate gradient method seems to be the most important.

#### Stationary methods for Ax = y

• Split the system matrix **A** into

$$\mathbf{A} = \mathbf{M} - \mathbf{N},$$

such that  ${\bf M}$  is a nonsingular special matrix (diagonal, lower triangular, etc.) Direct access to  ${\bf A}$  is thus required.

• Compute iteratively the solution using an easily solvable<sup>5</sup>

$$\mathbf{M}\mathbf{x}_{k+1} = \mathbf{N}\mathbf{x}_k + \mathbf{y}.$$

• Convergence properties are usually well-analyzed, e.g.

If  $\mathbf{A} \in \mathbb{R}^{n \times n}$  is nonsingular and the spectral radius<sup>6</sup> of  $\mathbf{M}^{-1}\mathbf{N}$  satisfies  $\rho(\mathbf{M}^{-1}\mathbf{N}) = < 1$ , then

- $\mathbf{x}_k$  converge to  $\mathbf{x} = \mathbf{A}^{-1}\mathbf{y}$  for any starting  $\mathbf{x}_0$ ,
- the error  $\|\mathbf{x} \mathbf{x}_k\|$  tends to zero like  $\rho(\mathbf{M}^{-1}\mathbf{N})^k$ .

<sup>5</sup>The subscript k in  $\mathbf{x}_k$  denotes the iteration number, and not the kth component  $x_k$  of the (kth iterate) vector  $\mathbf{x}_k$ .

<sup>6</sup>Magnitude of the maximum-magnitude eigenvalue.



#### Stationary methods — Jacobi and Gauss-Seidel

Stationary methods for Ax = y are defined by

$$\mathbf{M}\mathbf{x}_{k+1} = \mathbf{N}\mathbf{x}_k + \mathbf{y}, \quad \text{where } \mathbf{A} = \mathbf{M} - \mathbf{N}.$$

Split the system matrix  ${\bf A}$  into its strictly lower triangular  ${\bf L},$  diagonal  ${\bf D}$  and strictly upper triangular  ${\bf U}$  parts,

 $\mathbf{A} = \mathbf{L} + \mathbf{D} + \mathbf{U}.$ 

• The Jacobi method is defined by

 $\mathbf{M} = \mathbf{D},$  $\mathbf{N} = -\mathbf{L} - \mathbf{U}.$ 

• The Gauss-Seidel method is defined by

$$\mathbf{M} = \mathbf{D} + \mathbf{L},$$
$$\mathbf{N} = -\mathbf{U}.$$

Both method are convergent if **A** is symmetric positive-definite.

#### Stationary methods — Jacobi and Gauss-Seidel

Both Jacobi and Gauss-Seidel formulas have a simple explanation. For example, consider a linear system with three unknowns:

> $a_{11}x_1 + a_{12}x_2 + a_{13}x_3 = y_1$  $a_{21}x_1 + a_{22}x_2 + a_{23}x_3 = y_2$  $a_{31}x_1 + a_{32}x_2 + a_{33}x_3 = y_3$

It can be transformed to an equivalent form

 $a_{11}x_1 = y_1 - a_{12}x_2 - a_{13}x_3$  $a_{22}x_2 = y_2 - a_{11}x_1 - a_{13}x_3$  $a_{33}x_3 = y_3 - a_{11}x_1 - a_{12}x_2$ 

which in the vector notation takes the form

$$\mathbf{D}\mathbf{x} = \mathbf{y} - (\mathbf{L} + \mathbf{U})\mathbf{x},$$

where D, L and U are respectively the diagonal, strictly lower and strictly upper triangular parts of A.

#### Stationary methods — Jacobi and Gauss-Seidel

Therefore, every linear system  $\boldsymbol{A}\boldsymbol{x}=\boldsymbol{y}$  can be transformed to the equivalent form

$$\mathsf{D}\mathsf{x}=\mathsf{y}-(\mathsf{L}+\mathsf{U})\mathsf{x}.$$

The Jacobi method takes this formula directly and obtains

$$\mathsf{Dx}_{k+1} = \mathsf{y} - (\mathsf{L} + \mathsf{U})\mathsf{x}_k,$$

where all components of  $\mathbf{x}_{k+1}$  are computed using the previous-step iterate  $\mathbf{x}_k$ . However, when the *i*th component of the vector  $\mathbf{x}_{k+1}$  is being computed, then all the components preceding it (no. 1, 2, ..., *i* - 1) are already known and can be used instead of these of the previous-step iterate  $\mathbf{x}_k$ . This yields the Gauss-Seidel method,

$$\begin{aligned} \mathbf{D}\mathbf{x}_{k+1} &= \mathbf{y} - \mathbf{L}\mathbf{x}_{k+1} - \mathbf{U}\mathbf{x}_k, \qquad \text{that is} \\ (\mathbf{D} + \mathbf{U})\mathbf{x}_{k+1} &= \mathbf{y} - \mathbf{U}\mathbf{x}_k. \end{aligned}$$

#### Stationary methods — successive over-relaxation (SOR)

The Jacobi and Gauss-Seidel methods are appealing because of their simplicity. However, when the spectral radius of  $\mathbf{M}^{-1}\mathbf{N}$  is close to unity, their convergence is very slow.

The successive over-relaxation (SOR) method tries to improve the convergence by modifying the original Gauss-Seidel step,

$$\mathbf{D}\mathbf{x}_{k+1} = \mathbf{y} - \mathbf{L}\mathbf{x}_{k+1} - \mathbf{U}\mathbf{x}_k,$$

so that the formula for the iterate  $\mathbf{x}_{k+1}$  is weighted against the previous value  $\mathbf{x}_k$  with such a *relaxation factor*  $\omega$ ,

$$\mathbf{x}_{k+1} = \omega \mathbf{D}^{-1} \left[ \mathbf{y} - \mathbf{L} \mathbf{x}_{k+1} - \mathbf{U} \mathbf{x}_k \right] + (1 - \omega) \mathbf{x}_k,$$

that the corresponding spectral radius is minimized.

#### Stationary methods — successive over-relaxation (SOR)

The SOR method can be compactly represented in the general form of the stationary methods,

$$\mathbf{M}_{\omega}\mathbf{x}_{k+1} = \mathbf{N}_{\omega}\mathbf{x}_k + \omega\mathbf{y},$$

where

$$\begin{split} \mathbf{M}_{\omega} &= \mathbf{D} + \omega \mathbf{L}, \\ \mathbf{N}_{\omega} &= (1 - \omega) \mathbf{D} - \omega \mathbf{U}. \end{split}$$

The relaxation factor  $\omega$  should minimize the spectral radius of  $\mathbf{M}_{\omega}^{-1}\mathbf{N}_{\omega}$ . The choice of its optimum value is not easy. In general

- SOR can be convergent only for  $\omega \in (0, 2)$ .
- If **A** is symmetric positive-definite, then SOR converges for all  $\omega \in (0, 2)$ .

## Krylov subspace methods

Existence & uniqueness

Outline

Basics

Krylov subspace methods find iteratively the best (in a given sense) solution  $\mathbf{x}_k$  of  $\mathbf{A}\mathbf{x} = \mathbf{y}$  in successively larger Krylov subspaces  $\mathcal{K}_k(\mathbf{A}, \mathbf{y}) = \text{span}(\mathbf{y}, \mathbf{A}\mathbf{y}, \mathbf{A}^2\mathbf{y}, \dots, \mathbf{A}^{k-1}\mathbf{y})$ .

Direct methods

- They are often expressed as iterative optimization methods.
- Most of them can be also related to the Lanczos iterative tridiagonalization procedures.
- No access to the full matrix A is required, only a routine for matrix-vector multiplication Ax (and sometimes also A<sup>T</sup>y).

The most important Krylov subspace method seems to be the *conjugate gradient method*.

Iterative methods

0000

#### Krylov subspace methods — conjugate gradient method

The conjugate gradient method (CG) for a linear system Ax = y with a symmetric positive-definite A can be expressed as

• an iterative optimization of the objective function

$$\phi(\mathbf{x}) = rac{1}{2} \mathbf{x}^\mathsf{T} \mathbf{A} \mathbf{x} - \mathbf{y}^\mathsf{T} \mathbf{x}$$

- in successively growing Krylov subspaces  $\mathcal{K}_k(\mathbf{A}, \mathbf{y})$ ,
- which can be shown to be generated by the gradients in the successive approximations x<sub>k</sub>,

$$\nabla \phi(\mathbf{x}_k) = \mathbf{A}\mathbf{x}_k - \mathbf{y} = -\mathbf{r}_k,$$

where  $\mathbf{r}_k$  denotes the residual.

# Outline Basics Existence & uniqueness Direct methods Iterative methods HW7 0 0000

- The matrix **A** has to be symmetric positive-definite.
- Note that the matrix A is required only in the form of a matrix-vector multiplication Ax.
- The algorithm requires additional storage for few vectors and scalars only.

 $\mathbf{p} = \mathbf{r}_0 = \mathbf{y} - \mathbf{A}\mathbf{x}$ while  $\|\mathbf{r}_0\| > \epsilon$  $\mathbf{q} = \mathbf{A}\mathbf{p}$  $\alpha = \frac{\|\mathbf{r}_0\|^2}{\mathbf{p}^{\mathsf{T}}\mathbf{q}}$  $\mathbf{x} = \mathbf{x} + \alpha \mathbf{p}$  $\mathbf{r}_1 = \mathbf{r}_0 - \alpha \mathbf{q}$  $\beta = \|\mathbf{r}_1\|^2 / \|\mathbf{r}_0\|^2$  $\mathbf{p} = \mathbf{r}_1 + \beta \mathbf{p}$  $\mathbf{r}_{0} = \mathbf{r}_{1}$ 

If **A** is not symmetric positive-definite, then the CG method can be applied to the *normal equations*,

 $\mathbf{A}^{\mathsf{T}}\mathbf{A}\mathbf{x}=\mathbf{A}^{\mathsf{T}}\mathbf{y},$ 

which amounts to minimizing

$$\phi(\mathbf{x}) = \frac{1}{2} \|\mathbf{A}\mathbf{x} - \mathbf{y}\|^2.$$

This is the conjugate gradient least-squares (CGLS) method. **A** is required only in the form of the multiplications Ax and  $A^Ty$ .  $\mathbf{r} = \mathbf{y} - \mathbf{A}\mathbf{x}$  $\mathbf{p} = \mathbf{s}_0 = \mathbf{A}^{\mathsf{T}} \mathbf{r}$ while  $\|\mathbf{r}\| > \epsilon$  $\mathbf{q} = \mathbf{A}\mathbf{p}$  $\alpha = \|\mathbf{s}_0\|^2 / \|\mathbf{q}\|^2$  $\mathbf{x} = \mathbf{x} + \alpha \mathbf{p}$  $\mathbf{r} = \mathbf{r} - \alpha \mathbf{q}$  $\mathbf{s}_1 = \mathbf{A}^{\mathsf{T}} \mathbf{r}$  $\beta = \|\mathbf{s}_1\|^2 / \|\mathbf{s}_0\|^2$  $\mathbf{p} = \mathbf{s}_1 + \beta \mathbf{p}$  $s_0 = s_1$ 

Outline 0	Basics 000000000	Existence & uniqueness	Direct methods	Iterative methods	HW7 ●○
Out	ine				





#### Available soon at http://info.ippt.pan.pl/~ljank.

#### E-mail the answer and the source codes to ljank@ippt.pan.pl.