2 Intermezzo on (Numerical) Linear Algebra

Let A be an $n \times n$-matrix.

Note that the result of the multiplication of the matrix A with a vector x is a linear combination of the columns of A:

$$Ax = \sum_{i=1}^{n} x_i A_{:,i}$$

Nonsingular	Singular
A is invertible	A is not invertible
The columns are independent	The columns are dependent
The rows are independent	The rows are dependent
det $A \neq 0$	det $A = 0$
$Ax = 0$ has one solution $x = 0$	$Ax = 0$ has infinitely many solutions
$Ax = b$ has one solution $x = A^{-1}b$	$Ax = b$ has no solution or infinetly many
A has full rank	A has rank $r < n$
A has n nonzero pivots	A has $r < n$ pivots
span$\{A_{:,1}, \ldots, A_{:,n}\}$ has dimension n	span$\{A_{:,1}, \ldots, A_{:,n}\}$ has dimension $r < n$
span$\{A_{1,:}, \ldots, A_{n,:}\}$ has dimension n	span$\{A_{1,:}, \ldots, A_{n,:}\}$ has dimension $r < n$
All eigenvalues of A are nonzero	0 is eigenvalue of A
$0 \notin \sigma(A) =$ Spectrum of A	$0 \in \sigma(A)$
$A^H A$ is symmetric positive definite	$A^H A$ is only semidefinite
A has n (positive) singular values	A has $r < n$ (positive) singular values

Essential Decompositions

(1) Gaussian-elimination is nothing than LU-decomposition: $A = LU$ with lower triangular matrix L having ones on the main diagonal

Example 2.0.1 (Gaussian elimination and LU-factorization).

consider (forward) Gaussian elimination:

$$\begin{pmatrix} 1 & 1 & 0 \\ 2 & 1 & -1 \\ 3 & -1 & -1 \end{pmatrix} \begin{pmatrix} x_1 \\ x_2 \\ x_3 \end{pmatrix} = \begin{pmatrix} 4 \\ 1 \\ -3 \end{pmatrix} \quad \longleftrightarrow \quad \begin{array}{rcrcrcr} x_1 & + & x_2 & & & = & 4 \\ 2x_1 & + & x_2 & - & x_3 & = & 1 \\ 3x_1 & - & x_2 & - & x_3 & = & -3 \end{array} .$$

$$\begin{pmatrix} 1 & & \\ & 1 & \\ & & 1 \end{pmatrix} \begin{pmatrix} 1 & 1 & 0 \\ 2 & 1 & -1 \\ 3 & -1 & -1 \end{pmatrix} \begin{pmatrix} 4 \\ 1 \\ -3 \end{pmatrix} \Longrightarrow \begin{pmatrix} 1 & & \\ 2 & 1 & \\ & 0 & 1 \end{pmatrix} \begin{pmatrix} \mathbf{1} & 1 & 0 \\ 0 & -1 & -1 \\ 3 & -1 & -1 \end{pmatrix} \begin{pmatrix} 4 \\ -7 \\ -3 \end{pmatrix}$$

$$\Longrightarrow \begin{pmatrix} 1 & & \\ 2 & 1 & \\ 3 & 0 & 1 \end{pmatrix} \begin{pmatrix} \mathbf{1} & 1 & 0 \\ 0 & -1 & -1 \\ 0 & -4 & -1 \end{pmatrix} \begin{pmatrix} 4 \\ -7 \\ -15 \end{pmatrix} \Longrightarrow \underbrace{\begin{pmatrix} 1 & & \\ 2 & 1 & \\ 3 & 4 & 1 \end{pmatrix}}_{=L} \underbrace{\begin{pmatrix} 1 & 1 & 0 \\ 0 & \mathbf{-1} & -1 \\ 0 & 0 & 3 \end{pmatrix}}_{=U} \begin{pmatrix} 4 \\ -7 \\ 13 \end{pmatrix}$$

= pivot row, pivot element **bold**, *negative* multipliers red

◇

Perspective: link Gaussian elimination to matrix factorization
(row transformation = multiplication with elimination matrix)

$$a_1 \neq 0 \quad \blacktriangleright \quad \begin{pmatrix} 1 & 0 & \cdots & \cdots & 0 \\ -\frac{a_2}{a_1} & 1 & & & 0 \\ -\frac{a_3}{a_1} & & \ddots & & \\ \vdots & & & \ddots & \\ -\frac{a_n}{a_1} & 0 & & & 1 \end{pmatrix} \begin{pmatrix} a_1 \\ a_2 \\ a_3 \\ \vdots \\ a_n \end{pmatrix} = \begin{pmatrix} a_1 \\ 0 \\ 0 \\ \vdots \\ 0 \end{pmatrix} .$$

$n-1$ steps of Gaussian elimination: $\Longrightarrow$ matrix factorization

(non-zero pivot elements assumed)

$\mathbf{A} = \mathbf{L}_1 \cdot \cdots \cdot \mathbf{L}_{n-1}\mathbf{U}$ with elimination matrices $\mathbf{L}_i$, $i = 1, \ldots, n-1$,
upper triangular matrix $\mathbf{U} \in \mathbb{R}^{n,n}$.

$$\begin{pmatrix} 1 & 0 & \cdots & \cdots & 0 \\ l_2 & 1 & & & 0 \\ l_3 & & \ddots & & \\ \vdots & & & \ddots & \\ l_n & 0 & & & 1 \end{pmatrix} \begin{pmatrix} 1 & 0 & \cdots & \cdots & 0 \\ 0 & 1 & & & 0 \\ 0 & h_3 & 1 & & \\ \vdots & \vdots & & \ddots & \\ 0 & h_n & 0 & & 1 \end{pmatrix} = \begin{pmatrix} 1 & 0 & \cdots & \cdots & 0 \\ l_2 & 1 & & & 0 \\ l_3 & h_3 & 1 & & \\ \vdots & \vdots & & \ddots & \\ l_n & h_n & 0 & & 1 \end{pmatrix}$$

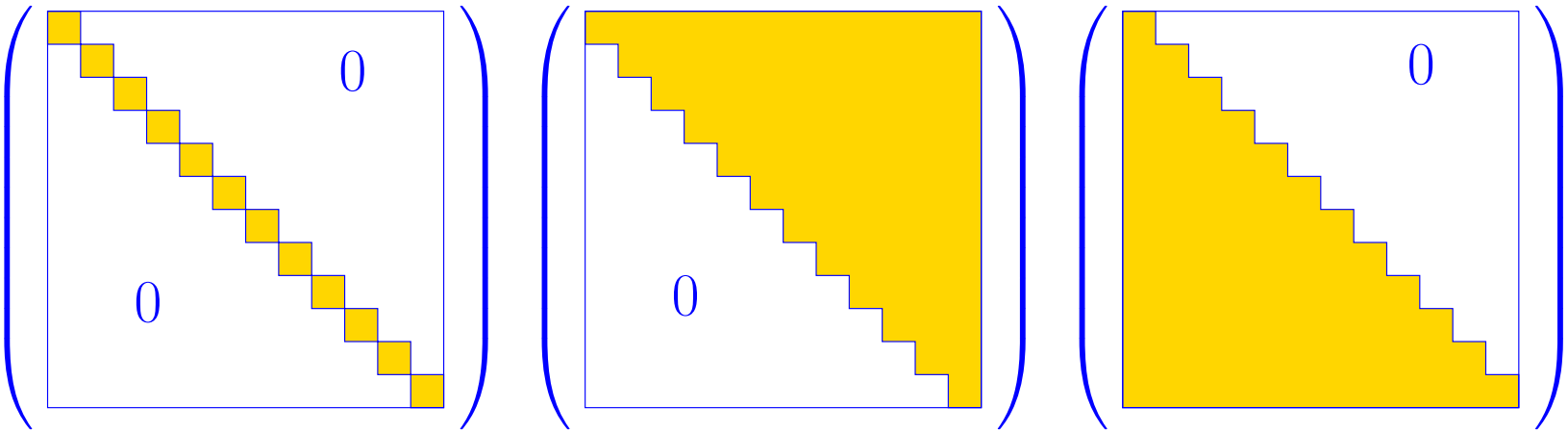
$\mathbf{L}_1 \cdot \cdots \cdot \mathbf{L}_{n-1}$ are normalized lower triangular matrices

(entries = multipliers $-\frac{a_{ik}}{a_{kk}}$)

Definition 2.0.1 (Types of matrices)**.**
A matrix $\mathbf{A} = (a_{ij}) \in \mathbb{R}^{m,n}$ *is*

- *diagonal matrix*, *if* $a_{ij} = 0$ *for* $i \neq j$,
- *upper triangular matrix* *if* $a_{ij} = 0$ *for* $i > j$,
- *lower triangular matrix* *if* $a_{ij} = 0$ *for* $i < j$.

A triangular matrix is normalized, *if* $a_{ii} = 1$, $i = 1, \ldots, \min\{m, n\}$.



diagonal matrix upper triangular lower triangular

The (forward) Gaussian elimination (without pivoting), for $\mathbf{Ax} = \mathbf{b}$, $\mathbf{A} \in \mathbb{R}^{n,n}$, if possible, is algebraically equivalent to an LU-factorization/LU-decomposition $\mathbf{A} = \mathbf{LU}$ of $\mathbf{A}$ into a normalized lower triangular matrix $\mathbf{L}$ and an upper triangular matrix $\mathbf{U}$.

Solving a linear system of equations by LU-factorization:

Algorithm 2.0.2 (Using LU-factorization to solve a linear system of equations).

① LU-decomposition $\mathbf{A} = \mathbf{LU}$, #elementary operations $\frac{1}{3}n(n-1)(n+1)$

$\mathbf{Ax} = \mathbf{b}$: ② forward substitution, solve $\mathbf{Lz} = \mathbf{b}$, #elementary operations $\frac{1}{2}n(n-1)$

③ backward substitution, solve $\mathbf{Ux} = \mathbf{z}$, #elementary operations $\frac{1}{2}n(n+1)$

Stability needs pivoting:

Example 2.0.3.

$$\mathbf{A} = \begin{pmatrix} 1 & 2 & 2 \\ 2 & -7 & 2 \\ 1 & 24 & 0 \end{pmatrix} \xrightarrow{❶} \begin{pmatrix} 2 & -7 & 2 \\ 1 & 2 & 2 \\ 1 & 24 & 0 \end{pmatrix} \xrightarrow{❷} \begin{pmatrix} 2 & -7 & 2 \\ 0 & 5.5 & 1 \\ 0 & 27.5 & -1 \end{pmatrix} \xrightarrow{❸} \begin{pmatrix} 2 & -7 & 2 \\ 0 & 27.5 & -1 \\ 0 & 5.5 & 1 \end{pmatrix} \xrightarrow{❹} \begin{pmatrix} 2 & -7 & 2 \\ 0 & 27.5 & -1 \\ 0 & 0 & 1.2 \end{pmatrix}$$

$$\mathbf{U} = \begin{pmatrix} 2 & -7 & 2 \\ 0 & 27.5 & -1 \\ 0 & 0 & 1.2 \end{pmatrix}, \quad \mathbf{L} = \begin{pmatrix} 1 & 0 & 0 \\ 0.5 & 1 & 0 \\ 0.5 & 0.2 & 1 \end{pmatrix}, \quad \mathbf{P} = \begin{pmatrix} 0 & 1 & 0 \\ 0 & 0 & 1 \\ 1 & 0 & 0 \end{pmatrix}.$$

◇

Lemma 2.0.2 (Existence of LU-factorization with pivoting)**.**
For any regular $\mathbf{A} \in \mathbb{K}^{n,n}$ there is a permutation matrix $\mathbf{P} \in \mathbb{K}^{n,n}$, a normalized lower triangular matrix $\mathbf{L} \in \mathbb{K}^{n,n}$, and a regular upper triangular matrix $\mathbf{U} \in \mathbb{K}^{n,n}$, such that $\mathbf{PA} = \mathbf{LU}$.

python-function: `LU, piv = scipy.linalg.lu_factor(A)`

`LU` = Matrix containing U in its upper triangle, and L in its lower triangle;

`piv` = pivot indices representing the permutation matrix P: row i of matrix was interchanged with row piv[i]

Round-off errors can be dangerous.

Definition 2.0.3 (Condition (number) of a matrix)**.**
Condition (number) of a matrix $\mathbf{A} \in \mathbb{R}^{n,n}$*:* $\operatorname{cond}(\mathbf{A}) := \left\|\mathbf{A}^{-1}\right\| \|\mathbf{A}\|$

Note:

$\operatorname{cond}(\mathbf{A})$ depends on $\|\cdot\|$!

- If $\operatorname{cond}(\mathbf{A}) \gg 1$, *small perturbations* in $\mathbf{A}$ can lead to *large relative errors* in the solution of the LSE.
- If $\operatorname{cond}(\mathbf{A}) \gg 1$, an algorithm can produce solutions with large relative error !

(1') If A is symmetric, then $A = LDL^H = R^H R$ with diagonal matrix D containing the pivots (Choleski-decomposition) and $R = \sqrt{D}L^H$. No pivoting is necessary.

python-function: `scipy.linalg.cho_factor(A)`

(2) Orthogonalisation: $A = QR$ with the matrix Q having orthonormal columns: $Q^H Q = QQ^H = I$ (see below)

(3) Singular value decomposition (SVD):

$$\mathbf{A} = \mathbf{U}\mathbf{\Sigma}\mathbf{V}^H$$

where each of the matrices U and V have orthonormal columns, $\mathbf{\Sigma} = \mathrm{diag}(\sigma_1, \ldots, \sigma_r, 0,]ldots, 0)$, $r = \text{rank}(A)$, $\sigma_1 \geq \sigma_2 \geq \sigma_r > 0$. Actually, $\sigma_1^2, \ldots, \sigma_r^2$ are the eigenvalues of $A^H A$ (see below).

(4) Schur-decomposition:

$$\forall \mathbf{A} \in \mathbb{K}^{n,n}: \quad \exists \mathbf{U} \in \mathbb{C}^{n,n} \text{ unitary: } \quad \mathbf{U}^H \mathbf{A} \mathbf{U} = \mathbf{T} \quad \text{with } \mathbf{T} \in \mathbb{C}^{n,n} \text{ upper triangular .}$$

Remark 2.0.4. All presented python-functions are in fact wrappers to LAPACK Fortran- or C-routines.

△

Remark 2.0.5. Not discussed in this lecture, but of essential importance in applications are the **sparse matrices** (i.e. having the number of non-zero elements much smaller than n). Special storing schemes and algorithms can sometimes keep the factors L and U sparse, but in general this is difficult or impossible. For such cases, **iterative methods** for LSE (as e.g. preconditioned conjugate gradient) are the methods of choice.

△

2.1 QR-Factorization/QR-decomposition

Recall from linear algebra:

Definition 2.1.1 (Unitary and orthogonal matrices)**.**

- $\mathbf{Q} \in \mathbb{K}^{n,n}$, $n \in \mathbb{N}$, *is unitary*, *if* $\mathbf{Q}^{-1} = \mathbf{Q}^H$.
- $\mathbf{Q} \in \mathbb{R}^{n,n}$, $n \in \mathbb{N}$, *is orthogonal*, *if* $\mathbf{Q}^{-1} = \mathbf{Q}^T$.

Theorem 2.1.2 (Criteria for Unitarity)**.**

$$\mathbf{Q} \in \mathbb{C}^{n,n} \quad \textit{unitary} \quad \Leftrightarrow \quad \|\mathbf{Q}\mathbf{x}\|_2 = \|\mathbf{x}\|_2 \quad \forall \mathbf{x} \in \mathbb{K}^n \ .$$

$\boxed{\mathbf{Q} \text{ unitary} \quad \Rightarrow \quad \mathrm{cond}(\mathbf{Q}) = 1}$ $\overset{(??)}{\Rightarrow}$ unitary transformations enhance (numerical) stability

If $\mathbf{Q} \in \mathbb{K}^{n,n}$ unitary, then

- all rows/columns (regarded as vectors $\in \mathbb{K}^n$) have Euclidean norm $= 1$,
- all rows/columns are pairwise orthogonal (w.r.t. Euclidean inner product),
- $|\det \mathbf{Q}| = 1$, and all eigenvalues $\in \{z \in \mathbb{Z}: |z| = 1\}$.
- $\|\mathbf{QA}\|_2 = \|\mathbf{A}\|_2$ for any matrix $\mathbf{A} \in \mathbb{K}^{n,m}$

Drawbacks of LU-factorization:

☹ often pivoting required (➞ destroys structure, Ex. **??**, leads to fill-in)

☹ Possible (theoretical) instability of partial pivoting $\longrightarrow$ Ex. **??**

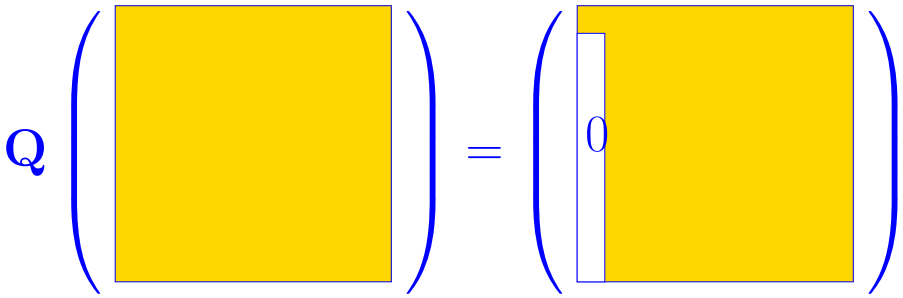
Stability problems of Gaussian elimination without pivoting are due to the fact that row transformations can convert well-conditioned matrices to ill-conditioned matrices, *cf.* Ex. **??**

Which bijective row transformations preserve the Euclidean condition number of a matrix ?

➢ transformations hat preserve the Euclidean norm of a vector **!**

▶ Investigate algorithms that use orthogonal/unitary row transformations to convert a matrix to upper triangular form.

Goal: find unitary row transformation rendering certain matrix elements zero.



with $\mathbf{Q}^H = \mathbf{Q}^{-1}$.

This "annihilation of column entries" is the key operation in Gaussian forward elimination, where it is achieved by means of non-unitary row transformations, see Sect. **??**. Now we want to find a counterpart of Gaussian elimination based on unitary row transformations on behalf of numerical stability.

In 2D: two possible orthogonal transformations make 2nd component of $\mathbf{a} \in \mathbb{R}^2$ vanish:

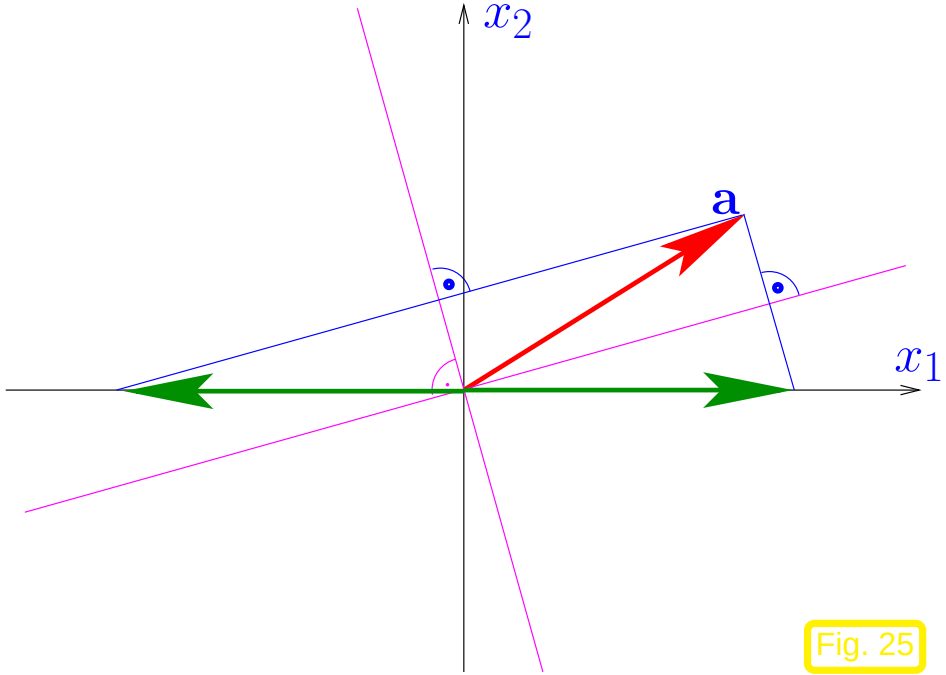


Fig. 25

reflection at angle bisector,

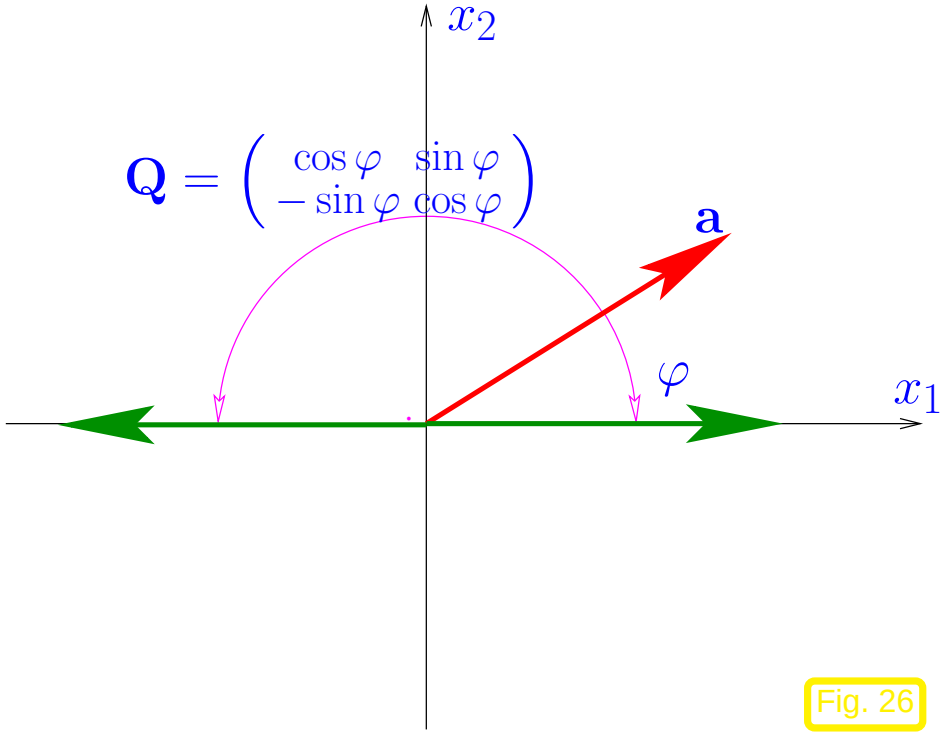


Fig. 26

rotation turning $\mathbf{a}$ onto x_1-axis.

➢ Note: two possible reflections/rotations

In nD: given $\mathbf{a} \in \mathbb{R}^n$ find orthogonal matrix $\mathbf{Q} \in \mathbb{R}^{n,n}$ such that $\mathbf{Qa} = \|\mathbf{a}\|_2\,\mathbf{e}_1$, $\mathbf{e}_1 \mathrel{\hat{=}}$ 1st unit vector.

Choice 1: Householder reflections

$$\mathbf{Q} = \mathbf{H}(\mathbf{v}) := \mathbf{I} - 2\frac{\mathbf{v}\mathbf{v}^H}{\mathbf{v}^H\mathbf{v}} \quad \text{with} \quad \mathbf{v} = \tfrac{1}{2}(\mathbf{a} \pm \|\mathbf{a}\|_2\,\mathbf{e}_1)\ . \tag{2.1.1}$$

"Geometric derivation" of Householder reflection, see Figure 25

Given $\mathbf{a}, \mathbf{b} \in \mathbb{R}^n$ with $\|\mathbf{a}\| = \|\mathbf{b}\|$, the difference vector $\mathbf{v} = \mathbf{b} - \mathbf{a}$ is orthogonal to the bisector.

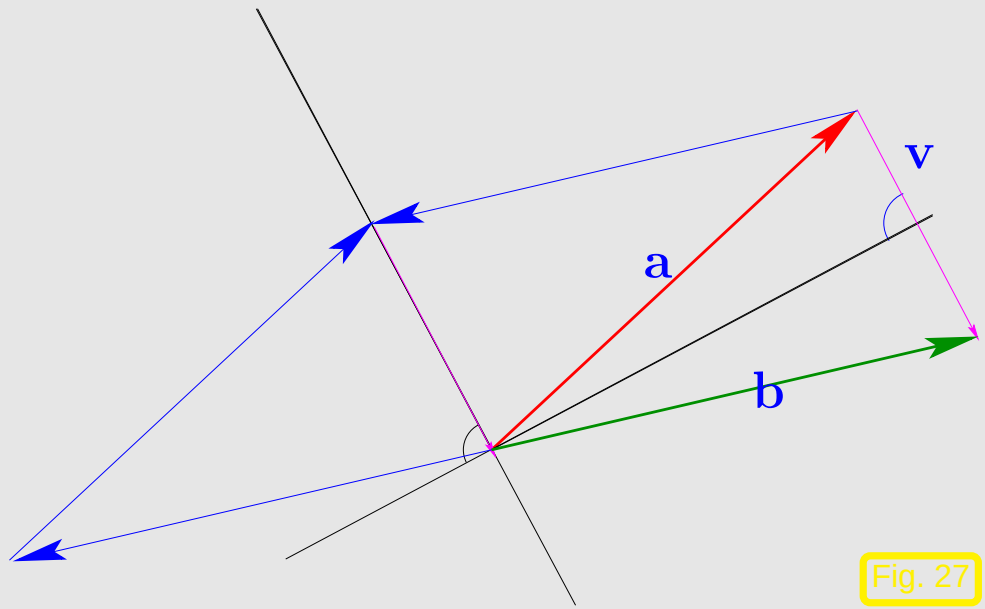


Fig. 27

$$\mathbf{b} = \mathbf{a} - (\mathbf{a} - \mathbf{b}) = \mathbf{a} - \mathbf{v}\frac{\mathbf{v}^T\mathbf{v}}{\mathbf{v}^T\mathbf{v}} = \mathbf{a} - 2\mathbf{v}\frac{\mathbf{v}^T\mathbf{a}}{\mathbf{v}^T\mathbf{v}} = \mathbf{a} - 2\frac{\mathbf{v}\mathbf{v}^T}{\mathbf{v}^T\mathbf{v}}\mathbf{a} = \mathbf{H}(\mathbf{v})\mathbf{a}\ ,$$

because, due to orthogonality $(\mathbf{a} - \mathbf{b}) \perp (\mathbf{a} + \mathbf{b})$

$$(\mathbf{a} - \mathbf{b})^T(\mathbf{a} - \mathbf{b}) = (\mathbf{a} - \mathbf{b})^T(\mathbf{a} - \mathbf{b} + \mathbf{a} + \mathbf{b}) = 2(\mathbf{a} - \mathbf{b})^T\mathbf{a}\ .$$

Remark 2.1.1 (Details of Householder reflections).

- Practice: for the sake of numerical stability (in order to avoid so-called *cancellation*) choose

$$\mathbf{v} = \begin{cases} \frac{1}{2}(\mathbf{a} + \|\mathbf{a}\|_2\,\mathbf{e}_1) & \text{, if } a_1 > 0\ , \\ \frac{1}{2}(\mathbf{a} - \|\mathbf{a}\|_2\,\mathbf{e}_1) & \text{, if } a_1 \leq 0\ . \end{cases}$$

However, this is not really needed [**?**, Sect. 19.1] !

- If $\mathbb{K} = \mathbb{C}$ and $a_1 = |a_1|\exp(i\varphi)$, $\varphi \in [0, 2\pi[$, then choose

$$\mathbf{v} = \frac{1}{2}(\mathbf{a} \pm \|\mathbf{a}\|_2\,\mathbf{e}_1 \exp(-i\varphi)) \quad \text{in (2.1.1).}$$

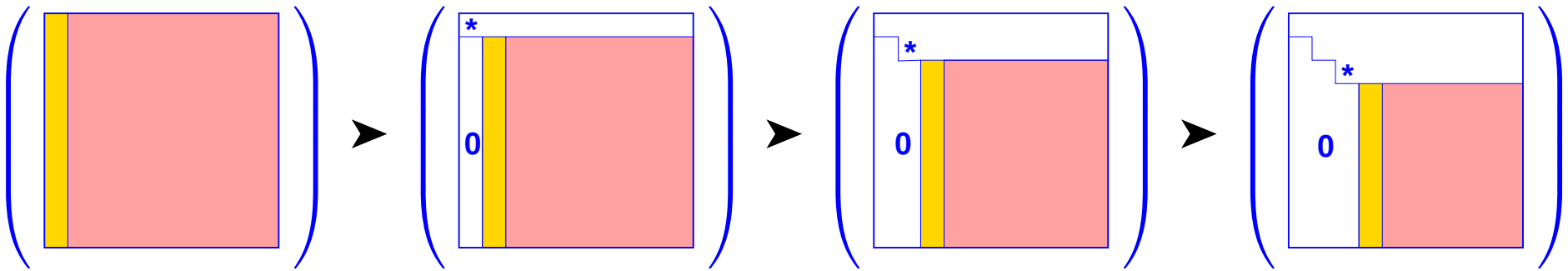
- efficient storage of Householder matrices $\longrightarrow$ [**?**]

$\triangle$

Choice 2: successive Givens rotations ($\longrightarrow$ 2D case)

$$\mathbf{G}_{1k}(a_1, a_k)\mathbf{A} := \begin{pmatrix} \bar{\gamma} & \cdots & \bar{\sigma} & \cdots & 0 \\ \vdots & \ddots & \vdots & & \vdots \\ -\sigma & \cdots & \gamma & \cdots & 0 \\ \vdots & & \vdots & \ddots & \vdots \\ 0 & \cdots & 0 & \cdots & 1 \end{pmatrix} \begin{pmatrix} a_1 \\ \vdots \\ a_k \\ \vdots \\ a_n \end{pmatrix} = \begin{pmatrix} a_1^{(1)} \\ \vdots \\ 0 \\ \vdots \\ a_n \end{pmatrix} \quad \text{, if} \quad \begin{aligned} \gamma &= \frac{a_1}{\sqrt{|a_1|^2+|a_k|^2}} \,, \\ \sigma &= \frac{a_k}{\sqrt{|a_1|^2+|a_k|^2}} \,. \end{aligned} \qquad (2.1.2)$$

Transformation to *upper triangular form* by successive unitary transformations:



= "target column $\mathbf{a}$" (determines unitary transformation),

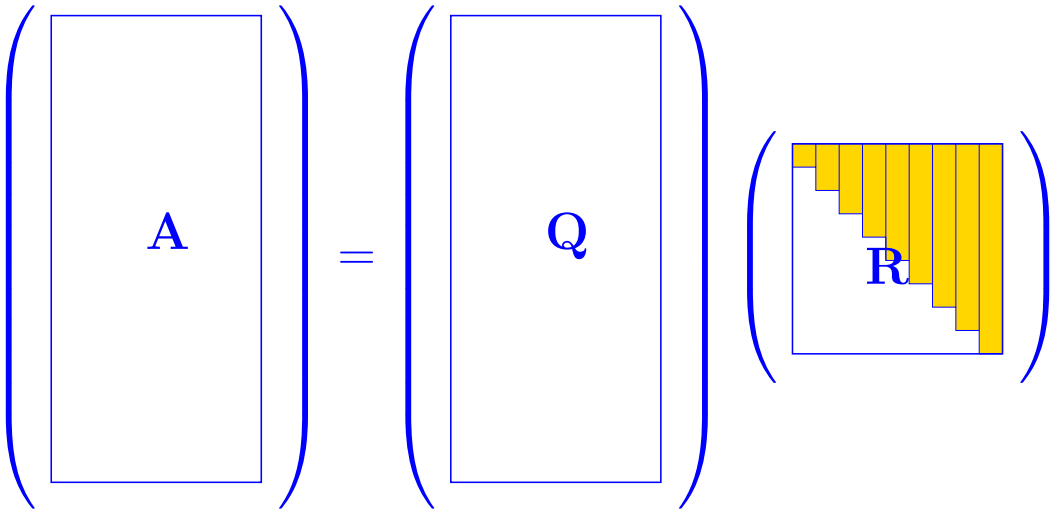
= modified in course of transformations.

$$\mathbf{Q}_{n-1}\mathbf{Q}_{n-2} \cdot \cdots \cdot \mathbf{Q}_1\mathbf{A} = \mathbf{R} \,,$$

QR-factorization (QR-decomposition) of $\mathbf{A} \in \mathbb{C}^{n,n}$: $\mathbf{A} = \mathbf{Q}\mathbf{R}$, $\mathbf{Q} := \mathbf{Q}_1^H \cdot \cdots \cdot \mathbf{Q}_{n-1}^H$ unitary matrix , $\mathbf{R}$ upper triangular matrix .

Generalization to $\mathbf{A} \in \mathbb{K}^{m,n}$:

$m > n$:



, $\mathbf{A} = \mathbf{Q}\mathbf{R}$, $\mathbf{Q} \in \mathbb{K}^{m,n}$, $\mathbf{R} \in \mathbb{K}^{n,n}$,

(2.1.3)

where $\mathbf{Q}^H\mathbf{Q} = \mathbf{I}$ (orthonormal columns), $\mathbf{R}$ upper triangular matrix.

Lemma 2.1.3 (Uniqueness of QR-factorization)**.**
The "economical" QR-factorization (2.1.3) *of* $\mathbf{A} \in \mathbb{K}^{m,n}$, $m \geq n$, *with* $\operatorname{rank}(\mathbf{A}) = n$ *is unique, if we demand* $r_{ii} > 0$.

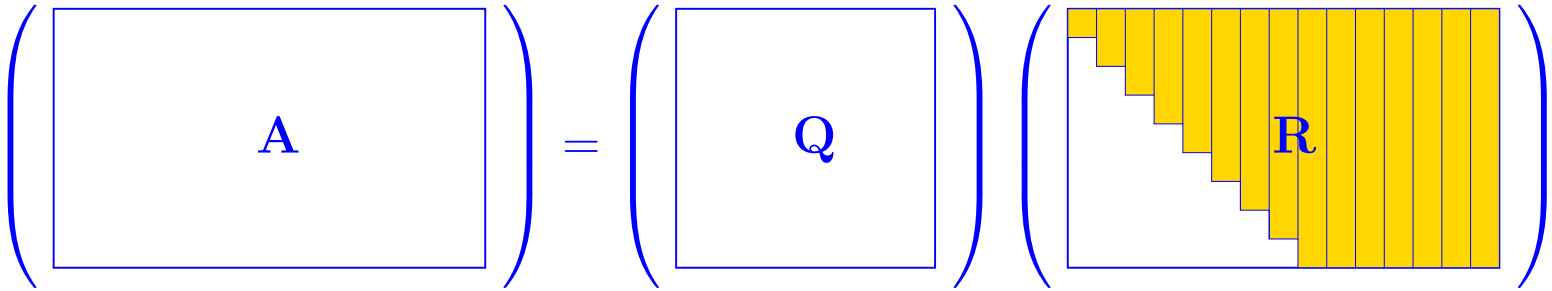
Proof. we observe that $\mathbf{R}$ is regular, if $\mathbf{A}$ has full rank n. Since the regular upper triangular matrices form a group under multiplication:

$$\mathbf{Q}_1\mathbf{R}_1 = \mathbf{Q}_2\mathbf{R}_2 \quad \Rightarrow \quad \mathbf{Q}_1 = \mathbf{Q}_2\mathbf{R} \quad \text{with upper triangular } \mathbf{R} := \mathbf{R}_2\mathbf{R}_1^{-1} \ .$$

$$\blacktriangleright \qquad \mathbf{I} = \mathbf{Q}_1^H\mathbf{Q}_1 = \mathbf{R}^H \underbrace{\mathbf{Q}_2^H\mathbf{Q}_2}_{=\mathbf{I}}\mathbf{R} = \mathbf{R}^H\mathbf{R} \ .$$

The assertion follows by uniqueness of Cholesky decomposition, Lemma **??**. □

$m < n$:



$$\mathbf{A} = \mathbf{Q}\mathbf{R} \quad , \quad \mathbf{Q} \in \mathbb{K}^{m,m} \ , \quad \mathbf{R} \in \mathbb{K}^{m,n} \ ,$$

where $\mathbf{Q}$ unitary, $\mathbf{R}$ upper triangular matrix.

Remark 2.1.2 (Choice of unitary/orthogonal transformation).

When to use which unitary/orthogonal transformation for QR-factorization **?**

- Householder reflections advantageous for fully populated target columns (dense matrices).
- Givens rotations more efficient ($\leftarrow$ more selective), if target column sparsely populated.

△

functions:

`Q,R = qr(A)` $\mathbf{Q} \in \mathbb{K}^{m,m}$, $\mathbf{R} \in \mathbb{K}^{m,n}$ for $\mathbf{A} \in \mathbb{K}^{m,n}$

`Q,R = qr(A,econ=True)` $\mathbf{Q} \in \mathbb{K}^{m,n}$, $\mathbf{R} \in \mathbb{K}^{n,n}$ for $\mathbf{A} \in \mathbb{K}^{m,n}$, $m > n$

(*economical* QR-factorization)

Computational effort for Householder QR-factorization of $\mathbf{A} \in \mathbb{K}^{m,n}$, $m > n$:

`Q,R = qr(A)` ➔ Costs: $O(m^2 n)$

`Q,R = qr(A,econ=True)` ➔ Costs: $O(mn^2)$

Example 2.1.3 (Complexity of Householder QR-factorization).

Code 2.1.4: timing QR-factorizations

```
from numpy import r_, mat, vstack, eye, ones, zeros
from scipy.linalg import qr
import timeit

def qr_full():
    global A
    Q, R = qr(A)

def qr_econ():
    global A
    Q, R = qr(A, econ=True)

def qr_ovecon():
    global A
    Q, R = qr(A, econ=True, overwrite_a=True)

def qr_r():
    global A
    R = qr(A, mode='r')

def qr_recon():
    global A
    R = qr(A, mode='r', econ=True)
```

```
nrEXP = 4
sizes = 2**r_[2:7]
qrtimes = zeros((5,sizes.shape[0]))
k = 0
for n in sizes:
    print 'n=', n
    m = n**2#4*n
    A = mat(1.*r_[1:m+1]).T*mat(1.*r_[1:n+1])
    A += vstack((eye(n), ones((m-n,n)) ))

    t = timeit.Timer('qr_full()','from __main__ import qr_full')
    avqr = t.timeit(number=nrEXP)/nrEXP
    print avqr
    qrtimes[0,k] = avqr

    t = timeit.Timer('qr_econ()','from __main__ import qr_econ')
    avqr = t.timeit(number=nrEXP)/nrEXP
    print avqr
    qrtimes[1,k] = avqr

    t = timeit.Timer('qr_ovecon()','from __main__ import qr_ovecon')
    avqr = t.timeit(number=nrEXP)/nrEXP
    print avqr
```

```
    qrtimes[2,k] = avqr

    t = timeit.Timer('qr_r()','from __main__ import qr_r')
    avqr = t.timeit(number=nrEXP)/nrEXP
    print avqr
    qrtimes[3,k] = avqr

    t = timeit.Timer('qr_recon()','from __main__ import qr_recon')
    avqr = t.timeit(number=nrEXP)/nrEXP
    print avqr
    qrtimes[4,k] = avqr

    k += 1

#print qrtimes[3]
import matplotlib.pyplot as plt
plt.loglog(sizes, qrtimes[0], 's', label='qr')
plt.loglog(sizes, qrtimes[1], '*', label="qr(econ=True)")
plt.loglog(sizes, qrtimes[2], '.', label="qr(econ=True, overwrite_a=True)")
plt.loglog(sizes, qrtimes[3], 'o', label="qr(mode='r')")
plt.loglog(sizes, qrtimes[4], '+', label="qr(mode='r', econ=True)")
v4 =  qrtimes[1,1]* (sizes/sizes[1])**4
v6 =  qrtimes[0,1]* (sizes/sizes[1])**6
```

```
plt.loglog(sizes, v4, label='O(n^4)')
plt.loglog(sizes, v6, '--',label='O(n^6)')
plt.legend(loc=2)
plt.xlabel('n')
plt.ylabel('time [s]')
plt.savefig('qrtiming.eps')
plt.show()
```

timing of different variants of QR-factorization

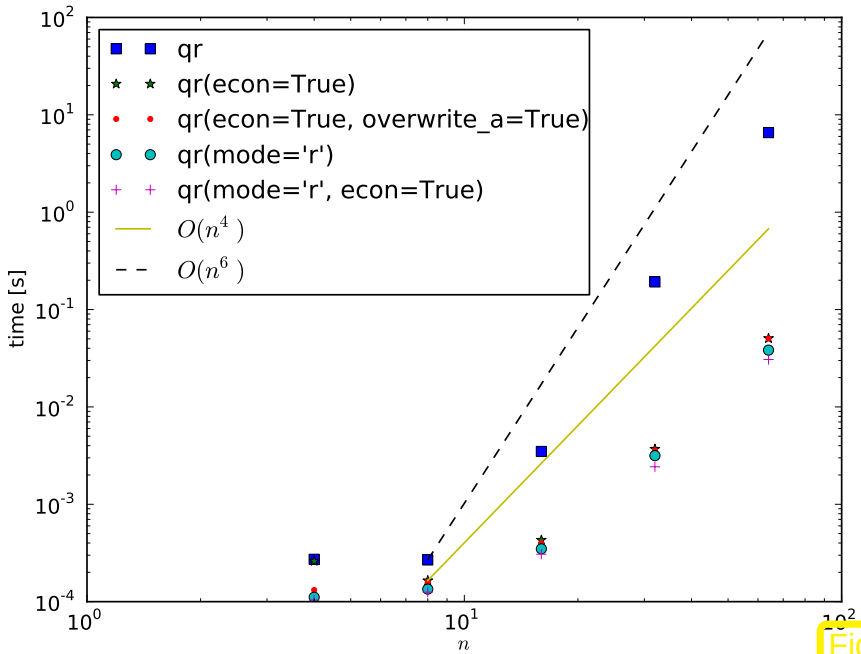
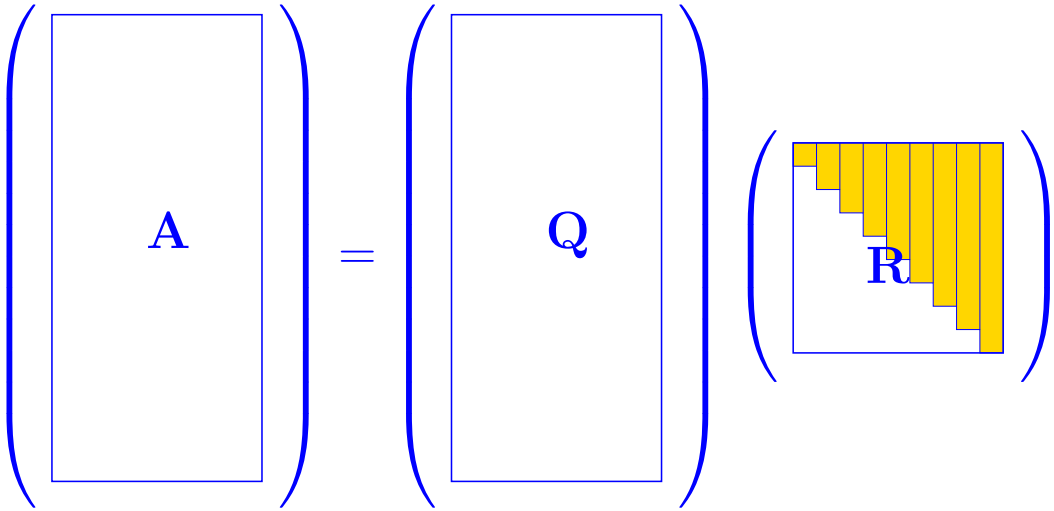


Fig. 28 ◇

Remark 2.1.5 (QR-orthogonalization).



$$, \quad \mathbf{A}, \mathbf{Q} \in \mathbb{K}^{m,n}, \mathbf{R} \in \mathbb{K}^{n,n} .$$

If $m > n$, $\operatorname{rank}(\mathbf{R}) = \operatorname{rank}(\mathbf{A}) = n$ (full rank)

➤ $\{\mathbf{q}_{\cdot,1}, \dots, \mathbf{q}_{\cdot,n}\}$ is orthonormal basis of $\operatorname{Im}(\mathbf{A})$ with $\operatorname{Span}\{\mathbf{q}_{\cdot,1}, \dots, \mathbf{q}_{\cdot,k}\} = \operatorname{Span}\{\mathbf{a}_{\cdot,1}, \dots, \mathbf{a}_{\cdot,k}\}$,$1 \le k \le n$.

△

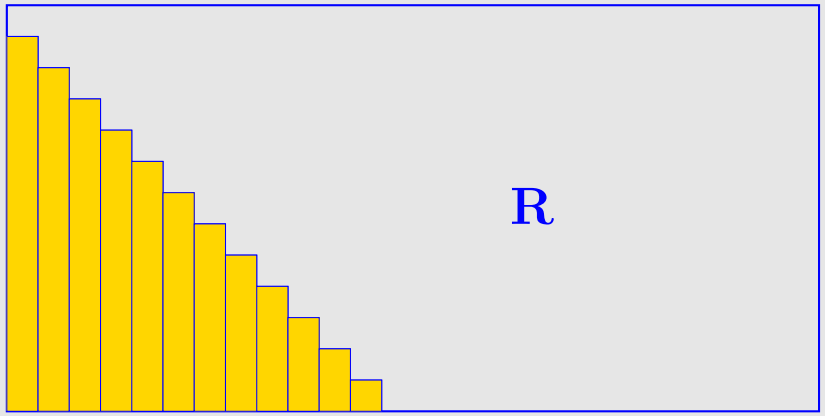
Remark 2.1.6 (Keeping track of unitary transformations).

How to store $\mathbf{G}_{i_1 j_1}(a_1, b_1) \cdot \dots \cdot \mathbf{G}_{i_k j_k}(a_k, b_k)$, $\mathbf{H}(\mathbf{v}_1) \cdot \dots \cdot \mathbf{H}(\mathbf{v}_k)$?

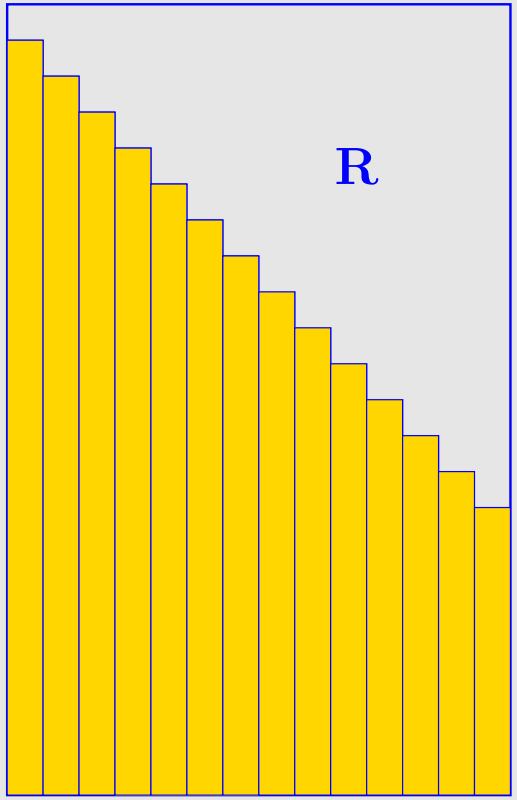
☛ For Householder reflections

$$\mathbf{H}(\mathbf{v}_1) \cdot \dots \cdot \mathbf{H}(\mathbf{v}_k): \quad \text{store } \mathbf{v}_1, \dots, \mathbf{v}_k$$

For in place QR-factorization of $\mathbf{A} \in \mathbb{K}^{m,n}$: store "“Householder vectors” $\mathbf{v}_j$ (decreasing size !) in lower triangle of $\mathbf{A}$



↑ Case $m < n$



= Householder vectors

Case $m > n$ →

☛ Convention for Givens rotations ($\mathbb{K} = \mathbb{R}$)

$$\mathbf{G} = \begin{pmatrix} \bar{\gamma} & \bar{\sigma} \\ -\sigma & \gamma \end{pmatrix} \quad \Rightarrow \quad \text{store} \quad \rho := \begin{cases} 1 & \text{, if } \gamma = 0 \,, \\ \frac{1}{2}\operatorname{sign}(\gamma)\sigma & \text{, if } |\sigma| < |\gamma| \,, \\ 2\operatorname{sign}(\sigma)/\gamma & \text{, if } |\sigma| \geq |\gamma| \,. \end{cases}$$

$$\blacktriangleright \quad \begin{cases} \rho = 1 & \Rightarrow \quad \gamma = 0 \,, \quad \sigma = 1 \\ |\rho| < 1 & \Rightarrow \quad \sigma = 2\rho \,, \quad \gamma = \sqrt{1-\sigma^2} \\ |\rho| > 1 & \Rightarrow \quad \gamma = 2/\rho \,, \quad \sigma = \sqrt{1-\gamma^2} \,. \end{cases}$$

Store $\mathbf{G}_{ij}(a,b)$ as triple (i,j,ρ)

Storing orthogonal transformation matrices is usually inefficient !

Algorithm 2.1.7 (Solving linear system of equations by means of QR-decomposition).

$\mathbf{Ax} = \mathbf{b}$:

① QR-decomposition $\mathbf{A} = \mathbf{QR}$, computational costs $\frac{2}{3}n^3 + O(n^2)$ (about twice as expensive as LU-decomposition without pivoting)

② orthogonal transformation $\mathbf{z} = \mathbf{Q}^H\mathbf{b}$, computational costs $4n^2 + O(n)$ (in the case of *compact storage* of reflections/rotations)

③ Backward substitution, solve $\mathbf{Rx} = \mathbf{z}$, computational costs $\frac{1}{2}n(n+1)$

- Computing the generalized QR-decomposition $\mathbf{A} = \mathbf{QR}$ by means of Householder reflections or Givens rotations is (numerically stable) for any $\mathbf{A} \in \mathbb{C}^{m,n}$.
- For *any* regular system matrix an LSE can be solved by means of

  QR-decomposition **+** orthogonal transformation **+** backward substitution

  in a stable manner.

Example 2.1.8 (Stable solution of LSE by means of QR-decomposition). → Ex. **??**

Code 2.1.9: R-fac. ↔ Gaussian elimination

```
from numpy import tril, vstack, hstack, eye, zeros, ones, dot, r_
from numpy.linalg import norm, solve, qr
sizes = r_[10:1001:10]
errlu = zeros(sizes.shape)
errqr = zeros(sizes.shape)
k = 0
for n in sizes:
    A = -tril(ones((n,n-1))) + 2*vstack((eye(n-1),zeros(n-1)))
    A = hstack((A, ones((n,1))))
    x = (-1.)**r_[1:n+1]
    b = dot(A,x)
    Q, R = qr(A)
    errlu[k] = norm(solve(A,b)-x)/norm(x)
    errqr[k] = norm(solve(R,dot(Q.T.conj(),b))-x)/norm(x)
    k += 1

import matplotlib.pyplot as plt
plt.semilogy(sizes, errlu, '*', label='LU')
plt.semilogy(sizes, errqr, 'ro', label='QR')
plt.legend(loc='center right')
plt.savefig('wilksolerr.eps')
plt.show()
```

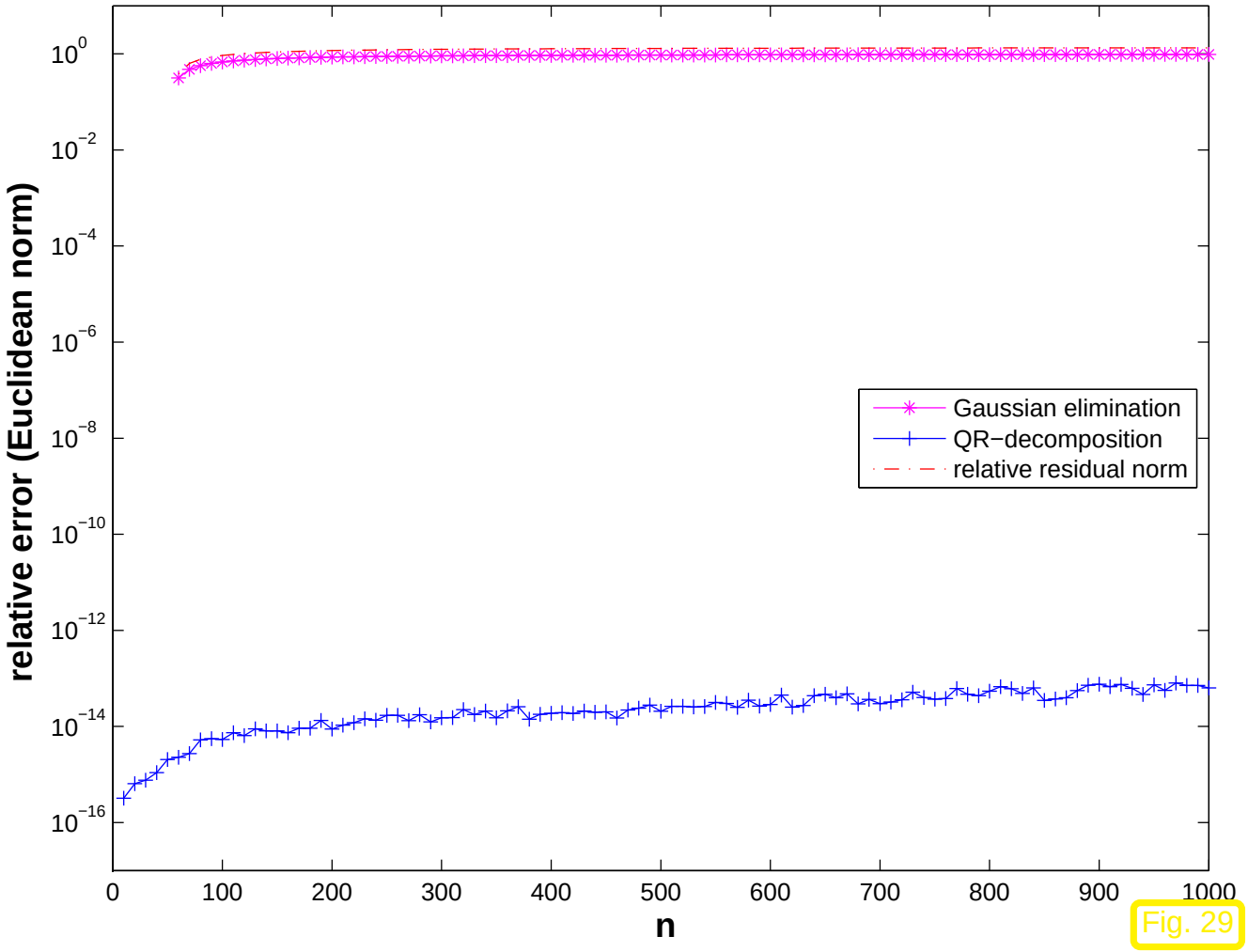


Fig. 29

◁ superior stability of QR-decomposition ! ◇

Fill-in for QR-decomposition **?**

bandwidth

$\mathbf{A} \in \mathbb{C}^{n,n}$ with QR-decomposition $\mathbf{A} = \mathbf{QR}$ $\Rightarrow$ $m(\mathbf{R}) \leq m(\mathbf{A})$ (→ Def. **??**)

Example 2.1.10 (QR-based solution of tridiagonal LSE).

Elimination of Sub-diagonals by $n-1$ successive Givens rotations:

$$\begin{pmatrix} * & * & 0 & 0 & 0 & 0 & 0 & 0 \\ * & * & * & 0 & 0 & 0 & 0 & 0 \\ 0 & * & * & * & 0 & 0 & 0 & 0 \\ 0 & 0 & * & * & * & 0 & 0 & 0 \\ 0 & 0 & 0 & * & * & * & 0 & 0 \\ 0 & 0 & 0 & 0 & * & * & * & 0 \\ 0 & 0 & 0 & 0 & 0 & * & * & * \\ 0 & 0 & 0 & 0 & 0 & 0 & * & * \end{pmatrix} \xrightarrow{\mathbf{G}_{12}} \begin{pmatrix} * & * & * & 0 & 0 & 0 & 0 & 0 \\ 0 & * & * & 0 & 0 & 0 & 0 & 0 \\ 0 & * & * & * & 0 & 0 & 0 & 0 \\ 0 & 0 & * & * & * & 0 & 0 & 0 \\ 0 & 0 & 0 & * & * & * & 0 & 0 \\ 0 & 0 & 0 & 0 & * & * & * & 0 \\ 0 & 0 & 0 & 0 & 0 & * & * & * \\ 0 & 0 & 0 & 0 & 0 & 0 & * & * \end{pmatrix} \xrightarrow{\mathbf{G}_{23}} \cdots \xrightarrow{\mathbf{G}_{n-1,n}} \begin{pmatrix} * & * & * & 0 & 0 & 0 & 0 & 0 \\ 0 & * & * & * & 0 & 0 & 0 & 0 \\ 0 & 0 & * & * & * & 0 & 0 & 0 \\ 0 & 0 & 0 & * & * & * & 0 & 0 \\ 0 & 0 & 0 & 0 & * & * & * & 0 \\ 0 & 0 & 0 & 0 & 0 & * & * & * \\ 0 & 0 & 0 & 0 & 0 & 0 & * & * \\ 0 & 0 & 0 & 0 & 0 & 0 & 0 & * \end{pmatrix}$$

$\diamond$

2.2 Singular Value Decomposition

Remark 2.2.1 (Principal component analysis (PCA)).

Given: n data points $\mathbf{a}_j \in \mathbb{R}^m$, $j = 1, \ldots, n$, in m-dimensional (feature) space

Conjectured: "linear dependence": $\mathbf{a}_j \in V$, $V \subset \mathbb{R}^m$ p-dimensional subspace,

$p < \min\{m, n\}$ *unknown*

($\rightarrow$ possibility of dimensional reduction)

Task (PCA): determine (minimal) p and (orthonormal basis of) V

Perspective of linear algebra:

$$\text{Conjecture} \quad \Leftrightarrow \quad \operatorname{rank}(\mathbf{A}) = p \text{ for } \mathbf{A} := (\mathbf{a}_1, \ldots, \mathbf{a}_n) \in \mathbb{R}^{m,n}, \quad \operatorname{Im}(\mathbf{A}) = V$$

Extension: Data affected by measurement errors

(but conjecture upheld for unperturbed data)

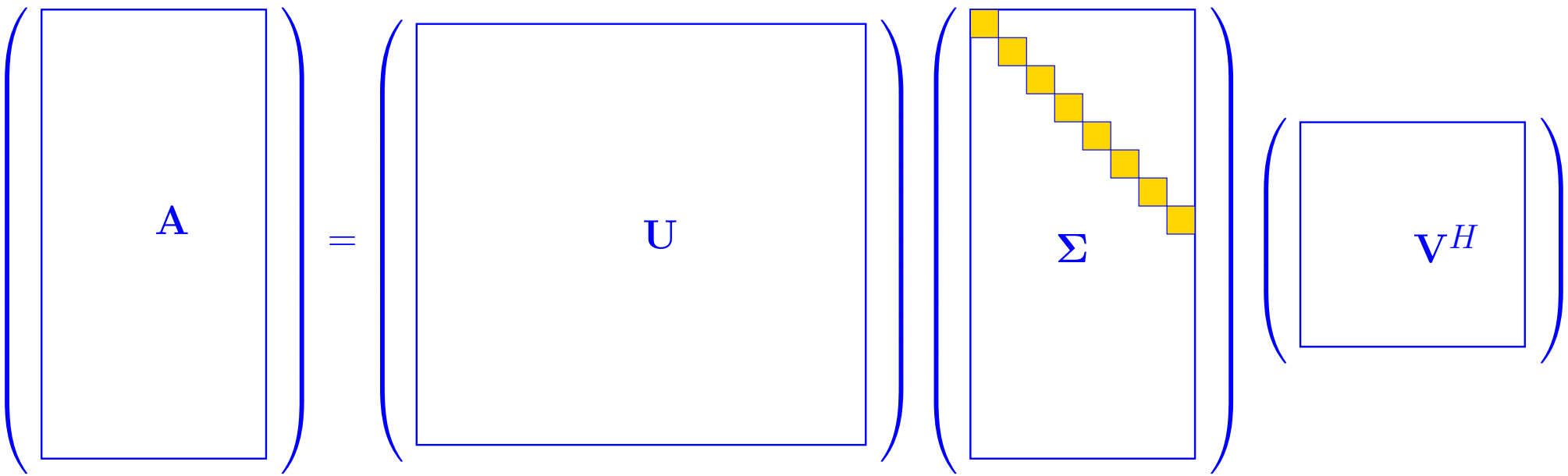
Application: ▶ Chemometrics (multivariate calibration methods for the analysis of chemical mixtures)

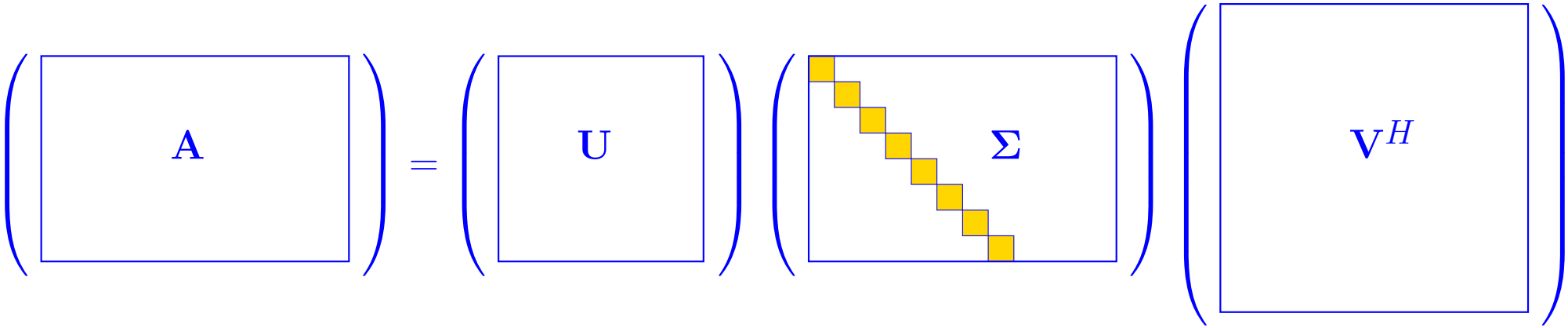
△

Theorem 2.2.1. *For any* $\mathbf{A} \in \mathbb{K}^{m,n}$ *there are unitary matrices* $\mathbf{U} \in \mathbb{K}^{m,m}$, $\mathbf{V} \in \mathbb{K}^{n,n}$ *and a (generalized) diagonal* $^{(*)}$ *matrix* $\mathbf{\Sigma} = \mathrm{diag}(\sigma_1, \dots, \sigma_p) \in \mathbb{R}^{m,n}$, $p := \min\{m, n\}$, $\sigma_1 \geq \sigma_2 \geq \sigma_p \geq 0$ *such that*

$$\mathbf{A} = \mathbf{U}\mathbf{\Sigma}\mathbf{V}^H \ .$$

$(*)$: $\mathbf{\Sigma}$ (generalized) diagonal matrix $:\Leftrightarrow$ $(\mathbf{\Sigma})_{i,j} = 0$, if $i \neq j$, $1 \leq i \leq m$, $1 \leq j \leq n$.





Proof. (of Thm. 2.2.1, by induction)

[**?**, Thm. 4.2.3]: Continuous functions attain extremal values on compact sets (here the unit ball $\{\mathbf{x} \in \mathbb{R}^n\colon \|\mathbf{x}\|_2 \leq 1\}$)

$$\blacktriangleright \quad \exists \mathbf{x} \in \mathbb{K}^n, \mathbf{y} \in \mathbb{K}^m \ , \quad \|\mathbf{x}\| = \|\mathbf{y}\|_2 = 1 : \quad \mathbf{A}\mathbf{x} = \sigma \mathbf{y} \ , \quad \sigma = \|\mathbf{A}\|_2 \ ,$$

where we used the definition of the matrix 2-norm, see Def. 1.1.12. By Gram-Schmidt orthogonalization: $\exists \widetilde{\mathbf{V}} \in \mathbb{K}^{n,n-1}$, $\widetilde{\mathbf{U}} \in \mathbb{K}^{m,m-1}$ such that

$$\mathbf{V} = (\mathbf{x}\ \widetilde{\mathbf{V}}) \in \mathbb{K}^{n,n} \quad , \quad \mathbf{U} = (\mathbf{y}\ \widetilde{\mathbf{U}}) \in \mathbb{K}^{m,m} \quad \text{are unitary.}$$

$$\blacktriangleright \quad \mathbf{U}^H \mathbf{A} \mathbf{V} = (\mathbf{y}\ \widetilde{\mathbf{U}})^H \mathbf{A} (\mathbf{x}\ \widetilde{\mathbf{V}}) = \left(\begin{array}{c|c} \mathbf{y}^H \mathbf{A} \mathbf{x} & \mathbf{y}^H \mathbf{A} \widetilde{\mathbf{V}} \\ \hline \widetilde{\mathbf{U}}^H \mathbf{A} \mathbf{x} & \widetilde{\mathbf{U}}^H \mathbf{A} \widetilde{\mathbf{V}} \end{array} \right) = \left(\begin{array}{c|c} \sigma & \mathbf{w}^H \\ \hline 0 & \mathbf{B} \end{array} \right) =: \mathbf{A}_1 \ .$$

$$\left\|\mathbf{A}_1\begin{pmatrix}\sigma\\ \mathbf{w}\end{pmatrix}\right\|_2^2 = \left\|\begin{pmatrix}\sigma^2+\mathbf{w}^H\mathbf{w}\\ \mathbf{B}\mathbf{w}\end{pmatrix}\right\|_2^2 = (\sigma^2+\mathbf{w}^H\mathbf{w})^2 + \|\mathbf{B}\mathbf{w}\|_2^2 \geq (\sigma^2+\mathbf{w}^H\mathbf{w})^2 \;,$$

$$\|\mathbf{A}_1\|_2^2 = \sup_{0\neq\mathbf{x}\in\mathbb{K}^n}\frac{\|\mathbf{A}_1\mathbf{x}\|_2^2}{\|\mathbf{x}\|_2^2} \geq \frac{\left\|\mathbf{A}_1\binom{\sigma}{\mathbf{w}}\right\|_2^2}{\left\|\binom{\sigma}{\mathbf{w}}\right\|_2^2} \geq \frac{(\sigma^2+\mathbf{w}^H\mathbf{w})^2}{\sigma^2+\mathbf{w}^H\mathbf{w}} = \sigma^2+\mathbf{w}^H\mathbf{w}\;. \tag{2.2.1}$$

$$\sigma^2 = \|\mathbf{A}\|_2^2 = \left\|\mathbf{U}^H\mathbf{A}\mathbf{V}\right\|_2^2 = \|\mathbf{A}_1\|_2^2 \overset{(2.2.1)}{\Longrightarrow} \|\mathbf{A}_1\|_2^2 = \|\mathbf{A}_1\|_2^2 + \|\mathbf{w}\|_2^2 \quad\Rightarrow\quad \mathbf{w}=0\;.$$

$$\blacktriangleright \qquad \mathbf{A}_1 = \left(\begin{array}{c|c}\sigma & 0\\ \hline 0 & \mathbf{B}\end{array}\right)\;.$$

Then apply induction argument to $\mathbf{B}$ $\square$.

Definition 2.2.2 (Singular value decomposition (SVD))**.**
The decomposition $\mathbf{A}=\mathbf{U}\boldsymbol{\Sigma}\mathbf{V}^H$ *of Thm. 2.2.1 is called singular value decomposition (SVD) of* $\mathbf{A}$*. The diagonal entries* σ_i *of* $\boldsymbol{\Sigma}$ *are the singular values of* $\mathbf{A}$*.*

Lemma 2.2.3. *The squares* σ_i^2 *of the non-zero singular values of* $\mathbf{A}$ *are the non-zero eigenvalues of* $\mathbf{A}^H\mathbf{A}$, $\mathbf{A}\mathbf{A}^H$ *with associated eigenvectors* $(\mathbf{V})_{:,1},\dots,(\mathbf{V})_{:,p}$, $(\mathbf{U})_{:,1},\dots,(\mathbf{U})_{:,p}$, *respectively.*

Proof. $\mathbf{A}\mathbf{A}^H$ and $\mathbf{A}^H\mathbf{A}$ are similar ($\longrightarrow$ Lemma 4.1.4) to diagonal matrices with non-zero diagonal entries σ_i^2 ($\sigma_i \neq 0$), e.g.,

$$\mathbf{A}\mathbf{A}^H = \mathbf{U}\mathbf{\Sigma}\mathbf{H}^H\mathbf{V}\mathbf{\Sigma}^H\mathbf{U}^H = \mathbf{U}\underbrace{\mathbf{\Sigma}\mathbf{\Sigma}^H}_{\text{diagonal matrix}}\mathbf{U}^H . \qquad \square$$

Remark 2.2.2 (SVD and additive rank-1 decomposition).

Recall from linear algebra: rank-1 matrices are tensor products of vectors

$$\mathbf{A} \in \mathbb{K}^{m,n} \quad \text{and} \quad \operatorname{rank}(\mathbf{A}) = 1 \quad \Leftrightarrow \quad \exists \mathbf{u} \in \mathbb{K}^m, \mathbf{v} \in \mathbb{K}^n\colon \quad \mathbf{A} = \mathbf{u}\mathbf{v}^H , \tag{2.2.2}$$

because $\operatorname{rank}(\mathbf{A}) = 1$ means that $\mathbf{a}\mathbf{x} = \mu(\mathbf{x})\mathbf{u}$ for some $\mathbf{u} \in \mathbb{K}^m$ and linear form $\mathbf{x} \mapsto \mu(\mathbf{x})$. By the Riesz representation theorem the latter can be written as $\mu(\mathbf{x}) = \mathbf{v}^H\mathbf{x}$.

▶ Singular value decomposition provides additive decomposition into rank-1 matrices:

$$\mathbf{A} = \mathbf{U}\mathbf{\Sigma}\mathbf{V}^H = \sum_{j=1}^{p} \sigma_j \, (\mathbf{U})_{:,j}(\mathbf{V})_{:,j}^H . \tag{2.2.3}$$

Remark 2.2.3 (Uniqueness of SVD).

SVD of Def. 2.2.2 is not (necessarily) unique, but the singular values are.

Assume that $\mathbf{A}$ has two singular value decompositions

$$\mathbf{A} = \mathbf{U}_1\mathbf{\Sigma}_1\mathbf{V}_1^H = \mathbf{U}_2\mathbf{\Sigma}_2\mathbf{V}_2^H \quad \Rightarrow \quad \mathbf{U}_1 \underbrace{\mathbf{\Sigma}_1\mathbf{\Sigma}_1^H}_{=\mathrm{diag}(s_1^1,\dots,s_m^1)} \mathbf{U}_1^H = \mathbf{A}\mathbf{A}^H = \mathbf{U}_2 \underbrace{\mathbf{\Sigma}_2\mathbf{\Sigma}_2^H}_{=\mathrm{diag}(s_1^2,\dots,s_m^2)} \mathbf{U}_2^H \,.$$

Two similar diagonal matrices are equal ! □

△

Python-function: `scipy.linalg.svd`
`scipy.sparse.linalg.svds` in scipy 0.10
SVD on a large sparse matrix: package `divisi`

python-functions (for algorithms see [**?**, Sect. 8.3]):

`s = svd(A)` : computes singular values of matrix $\mathbf{A}$

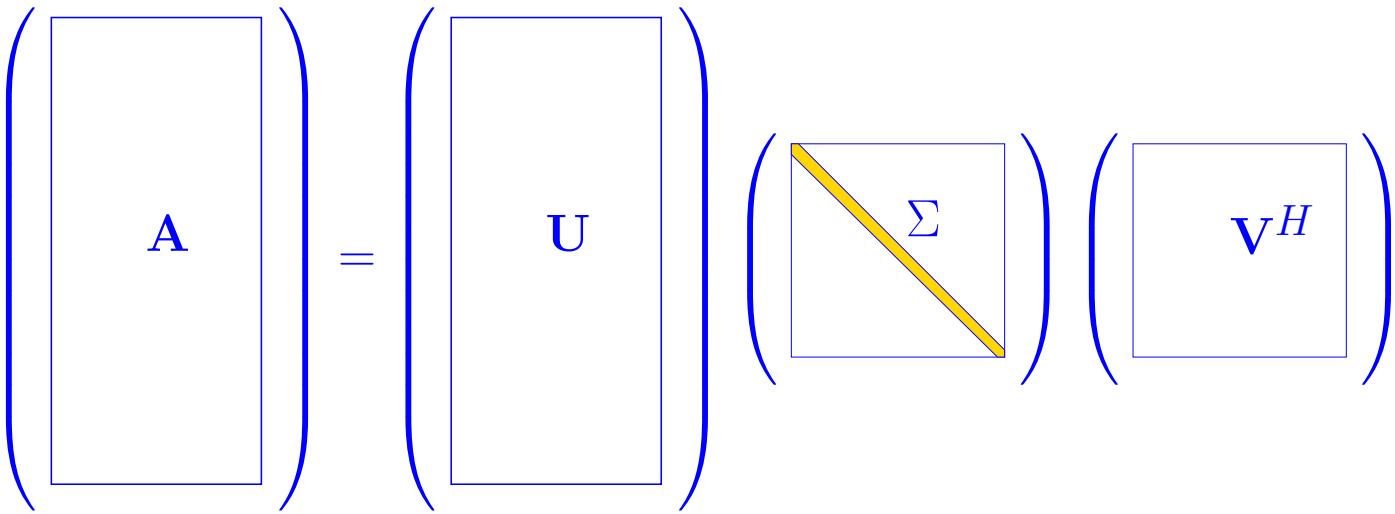
`[U,S,V] = svd(A)` : computes singular value decomposition according to Thm. 2.2.1

`[U,S,V] = svd(A,0)` : "economical" singular value decomposition for $m > n$: : $\mathbf{U} \in \mathbb{K}^{m,n}$, $\Sigma \in \mathbb{R}^{n,n}$, $\mathbf{V} \in \mathbb{K}^{n,n}$

`s = svds(A,k)` : k largest singular values (important for sparse $\mathbf{A} \longrightarrow$ Def. **??**)

`[U,S,V] = svds(A,k)`: partial singular value decomposition: $\mathbf{U} \in \mathbb{K}^{m,k}$, $\mathbf{V} \in \mathbb{K}^{n,k}$, $\Sigma \in \mathbb{R}^{k,k}$ diagonal with k largest singular values of $\mathbf{A}$.

Explanation: "economical" singular value decomposition:



(python) algorithm for computing SVD is (numerically) stable

Complexity:

$2mn^2 + 2n^3 + O(n^2) + O(mn)$ for `s = svd(A)`,

$4m^2n + 22n^3 + O(mn) + O(n^2)$ for `[U,S,V] = svd(A)`,

$O(mn^2) + O(n^3)$ for `[U,S,V]=svd(A,0)`, $m \gg n$.

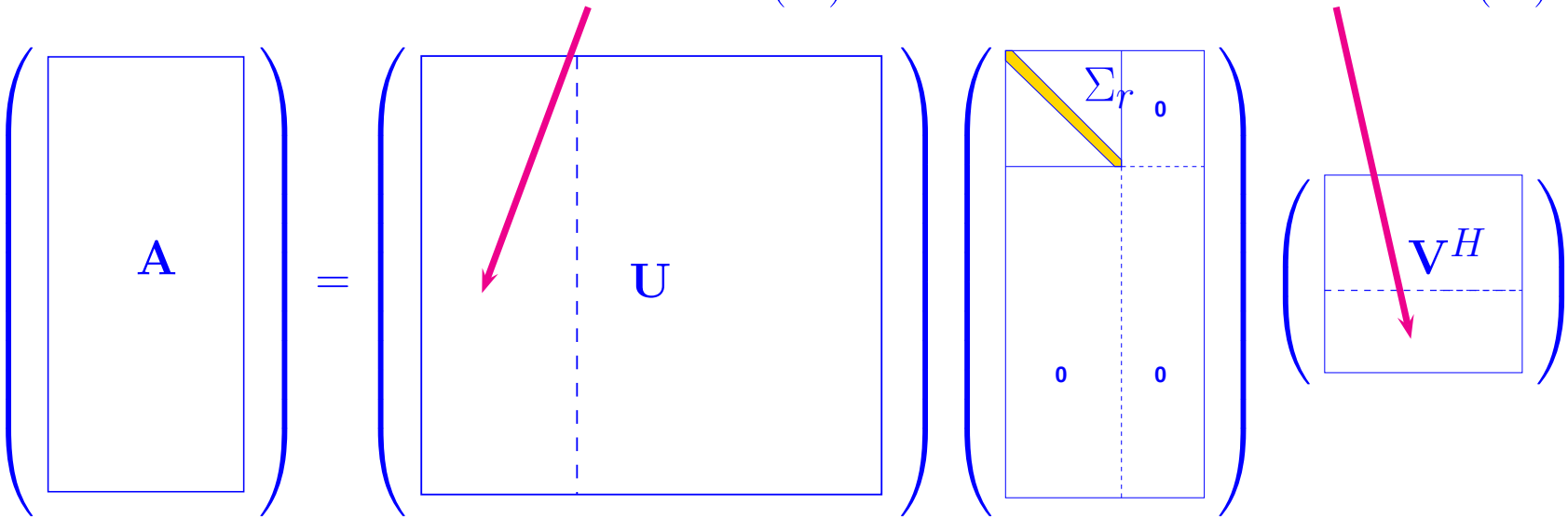
- Application of SVD: computation of rank , kernel and range of a matrix

Lemma 2.2.4 (SVD and rank of a matrix)**.**

If the singular values of $\mathbf{A}$ *satisfy* $\sigma_1 \geq \cdots \geq \sigma_r > \sigma_{r+1} = \cdots \sigma_p = 0$*, then*

- $\operatorname{rank}(\mathbf{A}) = r$,
- $\operatorname{Ker}(\mathbf{A}) = \operatorname{Span}\left\{(\mathbf{V})_{:,r+1}, \ldots, (\mathbf{V})_{:,n}\right\}$,
- $\operatorname{Im}(\mathbf{A}) = \operatorname{Span}\left\{(\mathbf{U})_{:,1}, \ldots, (\mathbf{U})_{:,r}\right\}$.

Illustration: columns = ONB of $\mathrm{Im}(\mathbf{A})$ rows = ONB of $\mathrm{Ker}(\mathbf{A})$



(2.2.4)

Remark: python function r=rank(A) relies on svd(A)

Lemma 2.2.4 ▶ PCA by SVD

❶ no perturbations:

SVD: $\mathbf{A} = \mathbf{U}\boldsymbol{\Sigma}\mathbf{V}^H$ satisfies $\sigma_1 \geq \sigma_2 \geq \ldots \sigma_p > \sigma_{p+1} = \cdots = \sigma_{\min\{m,n\}} = 0$,

$$V = \mathrm{Span}\underbrace{\{(\mathbf{U})_{:,1}, \ldots, (\mathbf{U})_{:,p}\}}_{\text{ONB of } V} .$$

❷ with perturbations:

SVD: $\mathbf{A} = \mathbf{U}\mathbf{\Sigma}\mathbf{V}^H$ satisfies $\sigma_1 \geq \sigma_2 \geq \ldots \sigma_p \gg \sigma_{p+1} \approx \cdots \approx \sigma_{\min\{m,n\}} \approx 0$,

$$V = \mathrm{Span}\,\underbrace{\{(\mathbf{U})_{:,1}, \ldots, (\mathbf{U})_{:,p}\}}_{\text{ONB of } V} .$$

If there is a pronounced gap in distribution of the singular values, which separates p large from $\min\{m,n\} - p$ relatively small singular values, this hints that $\mathrm{Im}(\mathbf{A})$ has essentially dimension p. It depends on the application what one accepts as a "pronounced gap".

Example 2.2.4 (Principal component analysis for data analysis).

$\mathbf{A} \in \mathbb{R}^{m,n}$, $m \gg n$:

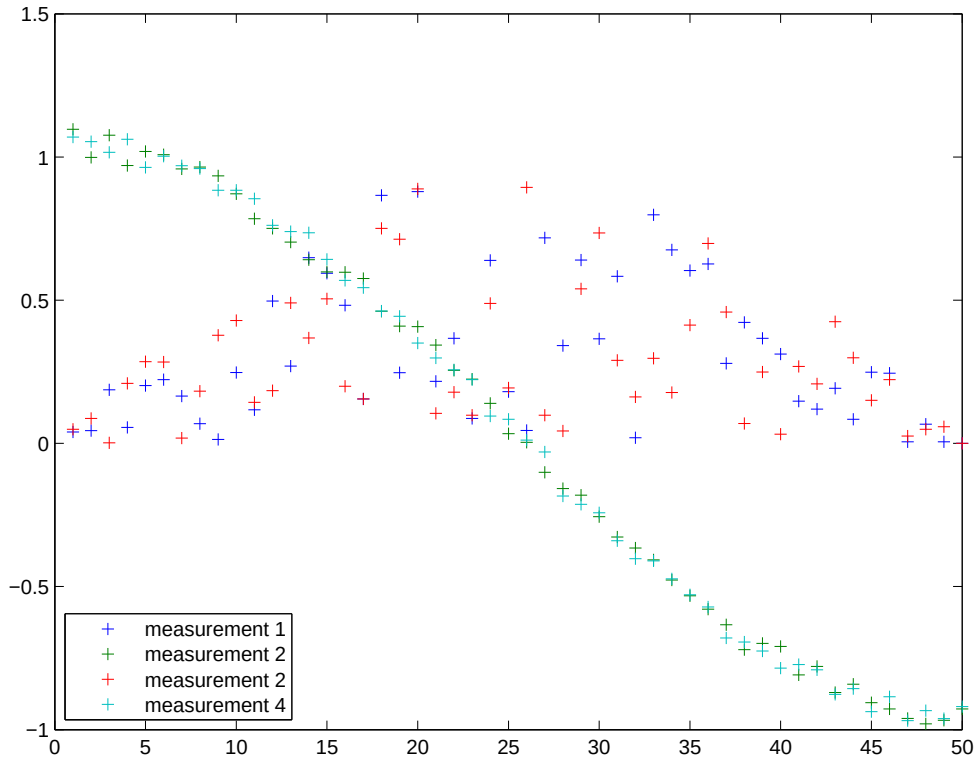
Columns $\mathbf{A}$	$\longrightarrow$	series of measurements at different times/locations etc.
Rows of $\mathbf{A}$	$\longrightarrow$	measured values corresponding to one time/location etc.

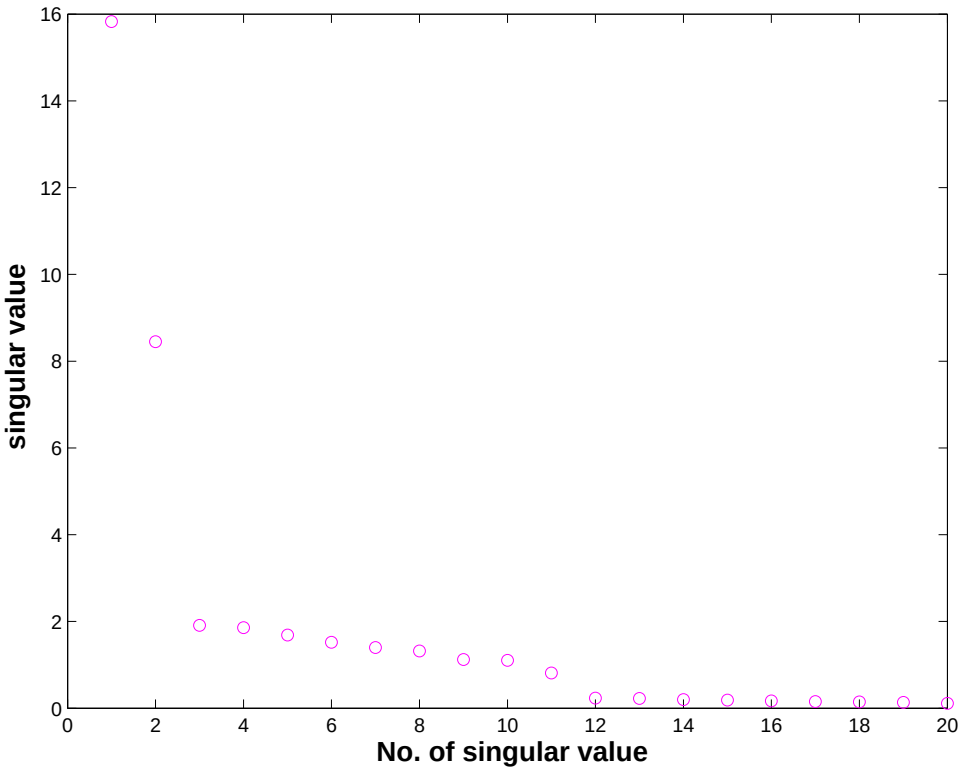
Goal: detect linear correlations

Concrete: two quantities measured over one year at 10 different sites

(Of course, measurements affected by errors/fluctuations)

```
n = 10; m = 50
r = linspace(1,m,m)
x = sin(pi*r/m)
y = cos(pi*r/m)
A = zeros((2*n,m))
for k in xrange(n):
    A[2*k] = x*rand(m)
    A[2*k+1] = y+0.1*rand(m)
```





← distribution of singular values of matrix

two dominant singular values !

measurements display linear correlation with **two** principal components

principal components = $\mathbf{u}_{\cdot,1}$, $\mathbf{u}_{\cdot,2}$ (leftmost columns of $\mathbf{U}$-matrix of SVD)

their relative weights = $\mathbf{v}_{\cdot,1}$, $\mathbf{v}_{\cdot,2}$ (leftmost columns of $\mathbf{V}$-matrix of SVD)

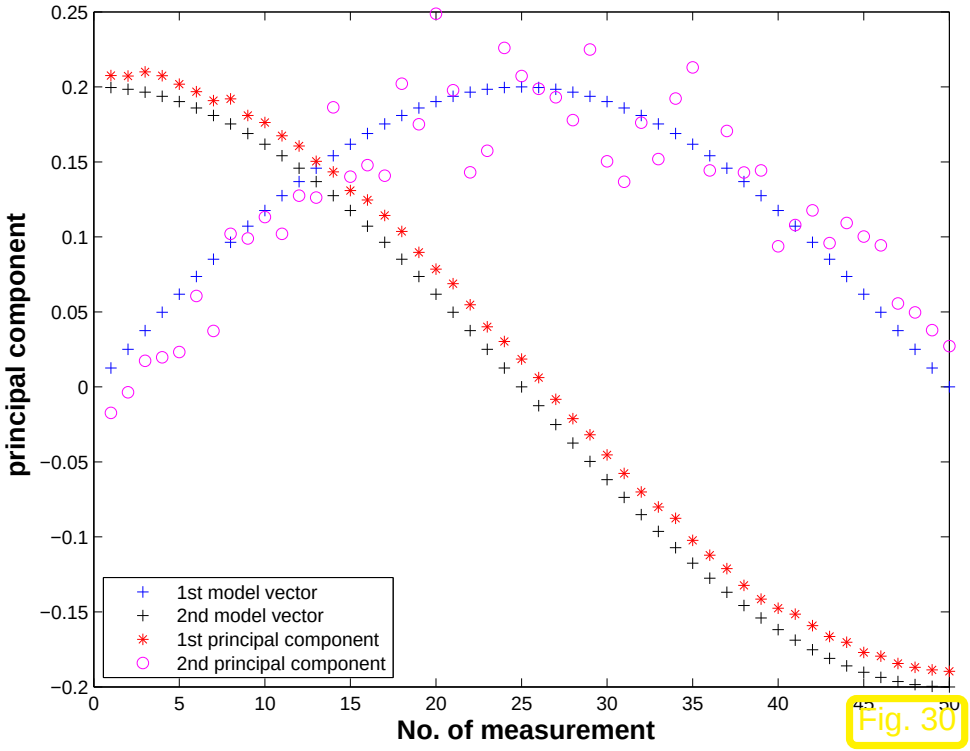


Fig. 30

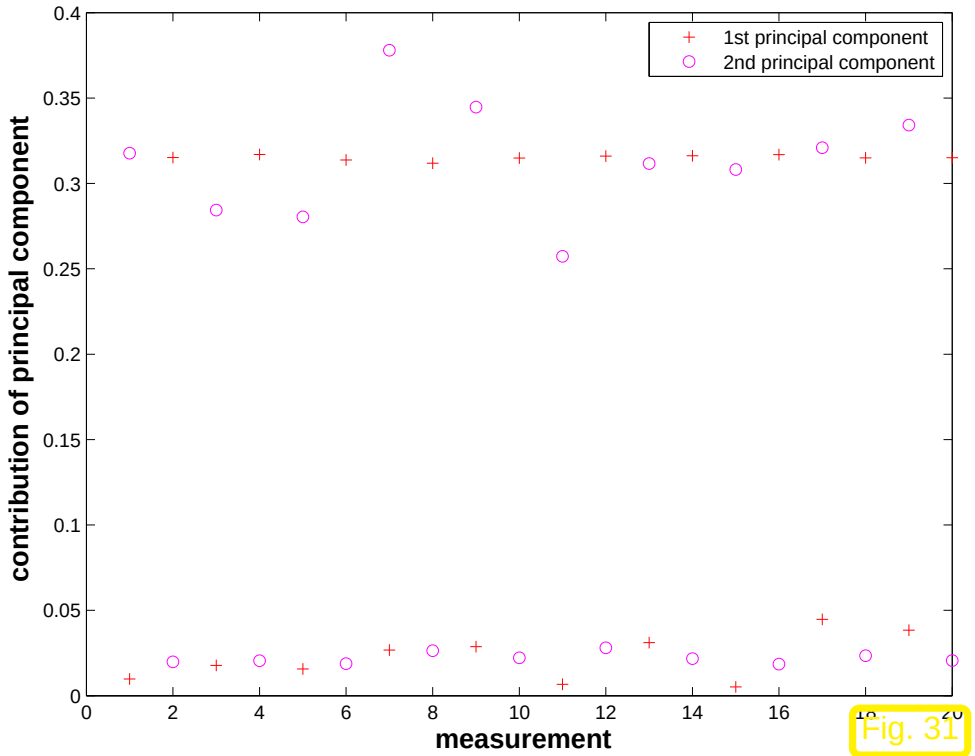


Fig. 31 ◇

- Application of SVD: extrema of quadratic forms on the unit sphere

A minimization problem on the Euclidean unit sphere $\{\mathbf{x} \in \mathbb{K}^n\colon \|\mathbf{x}\|_2 = 1\}$:

$$\text{given} \quad \mathbf{A} \in \mathbb{K}^{m,n}\ ,\ m > n, \quad \text{find } \mathbf{x} \in \mathbb{K}^n,\ \|\mathbf{x}\|_2 = 1\ , \quad \|\mathbf{A}\mathbf{x}\|_2 \to \min\ . \tag{2.2.5}$$

Use that multiplication with unitary matrices preserves the 2-norm and the singular value decomposition $\mathbf{A} = \mathbf{U}\Sigma\mathbf{V}^H$ (→ Def. 2.2.2):

$$\min_{\|\mathbf{x}\|_2=1} \|\mathbf{A}\mathbf{x}\|_2^2 = \min_{\|\mathbf{x}\|_2=1} \left\|\mathbf{U}\Sigma\mathbf{V}^H\mathbf{x}\right\|_2^2 = \min_{\|\mathbf{V}^H\mathbf{x}\|_2=1} \left\|\mathbf{U}\Sigma(\mathbf{V}^H\mathbf{x})\right\|_2^2$$

$$= \min_{\|\mathbf{y}\|_2=1} \|\Sigma\mathbf{y}\|_2^2 = \min_{\|\mathbf{y}\|_2=1} (\sigma_1^2 y_1^2 + \cdots + \sigma_n^2 y_n^2) \geq \sigma_n^2 \ .$$

The minimum σ_n^2 is attained for $\mathbf{y} = \mathbf{e}_n \quad \Rightarrow \quad$ minimizer $\mathbf{x} = \mathbf{V}\mathbf{e}_n = (\mathbf{V})_{:,n}$.

By similar arguments:

$$\sigma_1 = \max_{\|\mathbf{x}\|_2=1} \|\mathbf{A}\mathbf{x}\|_2 \quad , \quad (\mathbf{V})_{:,1} = \operatorname*{argmax}_{\|\mathbf{x}\|_2=1} \|\mathbf{A}\mathbf{x}\|_2 \ . \tag{2.2.6}$$

Recall: 2-norm of the matrix $\mathbf{A}$ is defined as the maximum in (2.2.6). Thus we have proved the following theorem:

Lemma 2.2.5 (SVD and Euclidean matrix norm)**.**

- $\forall \mathbf{A} \in \mathbb{K}^{m,n}$: $\|\mathbf{A}\|_2 = \sigma_1(\mathbf{A})$,
- $\forall \mathbf{A} \in \mathbb{K}^{n,n}$ *regular*: $\operatorname{cond}_2(\mathbf{A}) = \sigma_1/\sigma_n$.

Remark: functions `norm(A)` and `cond(A)` rely on svd(A)

Remark: Enchanced PCA in `matplotlib.mlab.PCA` and in the package MDA (Modular Toolkit for Data Processing)

2.3 Essential Skills Learned in Chapter 2

You should know:

- what is the QR-decomposition and possibilities to get it
- what is the singular value decomposition and how to use it
- applications of the svd: principal component analysis, extrema of quadratic forms on the unit sphere