

Plan général

Partie 1: Généralités

Partie 2: Méthodes directes

Partie 3: Méthodes itératives

Partie 4: Problèmes aux valeurs et vecteurs propres

Partie 5: Systèmes linéaires mal conditionnés

General considerations

- $Ax = b$ uniquely solvable if (i) $\mathcal{N}(A) = \{0\}$ and (ii) $b \in \mathcal{R}(A)$
True for any b when $m = n$ and A is invertible.

- Solution given by general theoretical formulas, e.g.:

$$x_i = \det(A_i) / \det(A), \quad \text{with } A_i := [a_1, \dots, a_{i-1}, b, a_{i+1}, \dots, a_n] \quad (\text{Cramer}).$$

However, totally impractical except for very small matrices

Direct vs. iterative methods for $Ax = b$:

- Direct methods: compute exact solution within finitely many operations. (assuming idealized conditions of exact arithmetic). Example: forward / backward substitution (soon).
- Iterative methods: compute **sequences** of successive approximations (limiting process with infinitely many operations). Examples: later.

Direct methods: usually exploit **multiplicative decomposition of A** into “simple” factors (diagonal, unitary, triangular).

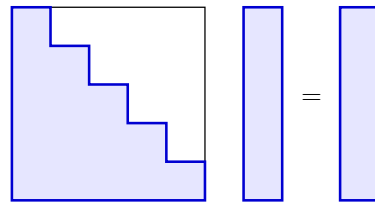
- This course: LU, Cholesky, QR factorizations.

Triangular systems

Systems with triangular matrices: Easy to solve numerically!

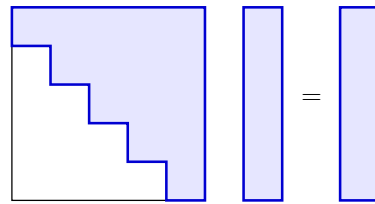
- A lower triangular: **forward substitution**

$$\begin{aligned} x_1 &= b_1/a_{11}, \\ x_2 &= (b_2 - a_{21}x_1)/a_{22}, \\ &\vdots \\ x_i &= \left(b_i - \sum_{j=1}^{i-1} a_{ij}x_j \right) / a_{ii} \quad (i \leq n) \end{aligned}$$



- A upper triangular: **backward substitution**

$$\begin{aligned} x_n &= b_n/a_{nn}, \\ x_{n-1} &= (b_{n-1} - a_{n-1,n}x_n)/a_{n-1,n-1}, \\ &\vdots \\ x_i &= \left(b_i - \sum_{j=i+1}^n a_{ij}x_j \right) / a_{ii} \quad (i \geq 1) \end{aligned}$$



- Forward/backward substitution: simple but essential components for many solution methods.

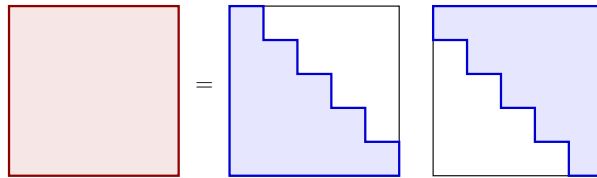
Backward stability of forward/backward substitution

Let $A \in \mathbb{K}^{n \times n}$ triangular. Let \tilde{x} : solve $Ax = b$ by finite-precision (forward/backward) substitution. Then: $(A + E)\tilde{x} = b$ where E is triangular and $|E| \leq n\varepsilon_{\text{mach}}|A|$ (notation: $|X| = [|x_{ij}|]$).

\tilde{x} exactly solves nearby linear system $(A + E)\tilde{x} = b$. Backward stability if A well-conditioned.

LU factorization

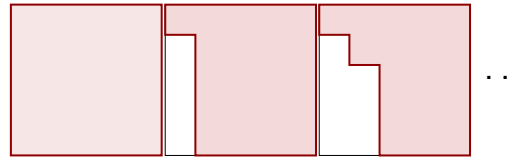
For $A \in \mathbb{K}^{n \times n}$ invertible: $A = LU$ (L, U : lower / upper triangular – may need pivoting, see later)



Solution method:

compute $A = LU$, then, (i): solve $Ly = b$; (ii): solve $Ux = y$.

Gaussian elimination:



Algebraically described using *elimination matrices* ($z \in \mathbb{K}^{n-1,1}$):

$$G_1(z) := \begin{bmatrix} 1 & 0 \\ z & I_{n-1} \end{bmatrix} = \begin{bmatrix} 1 & & & \\ z_2 & 1 & & \\ \vdots & & \ddots & \\ z_n & & & 1 \end{bmatrix}$$

Then (zeroing-out first column below diagonal):

$$A = \begin{bmatrix} a_{11} & u_1 \\ \ell_1 & A_2 \end{bmatrix}, \quad G_1(-\ell_1/a_{11})A = \begin{bmatrix} 1 & 0 \\ -\ell_1/a_{11} & I_{n-1} \end{bmatrix} \begin{bmatrix} a_{11} & u_1 \\ \ell_1 & A_2 \end{bmatrix} = \begin{bmatrix} a_{11} & u_1 \\ 0 & A'_2 \end{bmatrix} \cdot \begin{bmatrix} \square \\ \square \end{bmatrix}$$

Computation of LU factorization (continued)

- Zeroing-out first column below diagonal (in compact notation):

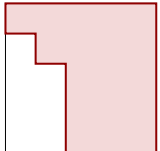
$$A = \begin{bmatrix} a_{11} & u_1 \\ \ell_1 & A_2 \end{bmatrix}, \quad G_1(-\ell_1/a_{11})A = \begin{bmatrix} 1 & 0 \\ -\ell_1/a_{11} & I_{n-1} \end{bmatrix} \begin{bmatrix} a_{11} & u_1 \\ \ell_1 & A_2 \end{bmatrix} = \begin{bmatrix} a_{11} & u_1 \\ 0 & A'_2 \end{bmatrix}.$$

Needs nonzero pivot: $a_{11} \neq 0$

- To continue process: more elimination matrices

$$G_k(z) = \begin{bmatrix} 1 & & & & & \\ & \ddots & & & & \\ & & 1 & & & \\ & & z_{k+1} & 1 & & \\ & & \vdots & & \ddots & \\ & & z_n & & & 1 \end{bmatrix}$$

- Then (zeroing-out second column below diagonal):

$$A'_2 = \begin{bmatrix} a'_{22} & u'_2 \\ \ell'_2 & A'_3 \end{bmatrix}, \quad G_2(-\ell'_2/a'_{22})G_1(-\ell_1/a_{11})A = \begin{bmatrix} a_{11} & u_1 \\ 0 & a'_{22} & u'_2 \\ 0 & 0 & A''_3 \end{bmatrix}$$


- Still continuing (produces upper triangular U , needs nonzero pivots $a_{11}, \dots, a_{n-1,n-1}$):

$$G_{n-1}(-\ell_{n-1}/a_{n-1,n-1}) \dots G_2(-\ell'_2/a'_{22}) G_1(-\ell_1/a_{11})A = U.$$

- Finally, invert using $G_k^{-1}(z) = G_k(-z)$ to obtain

$$A = G_1(\ell_1/a_{11}) G_2(\ell'_2/a'_{22}) \dots G_{n-1}(\ell_{n-1}/a_{n-1,n-1})U, \quad \text{i.e. } A = LU.$$

Example (with no roundoff errors)

$$A = \begin{bmatrix} 2 & 1 & 1 & 0 \\ 4 & 3 & 3 & 1 \\ 8 & 7 & 9 & 5 \\ 6 & 7 & 9 & 8 \end{bmatrix} = \begin{bmatrix} a_{11} & u_1 \\ l_1 & A_2 \end{bmatrix} \quad l_1 = \begin{bmatrix} 4 \\ 8 \\ 6 \end{bmatrix}, \quad u_1 = [1 \ 1 \ 0], \quad A_2 = \begin{bmatrix} 3 & 3 & 1 \\ 7 & 9 & 5 \\ 7 & 9 & 8 \end{bmatrix}$$

Introduce required zeros in three steps:

$$G_1 A = \begin{bmatrix} 1 & 0 & 0 & 0 \\ -2 & 1 & 0 & 0 \\ -4 & 0 & 1 & 0 \\ -3 & 0 & 0 & 1 \end{bmatrix} \begin{bmatrix} 2 & 1 & 1 & 0 \\ 4 & 3 & 3 & 1 \\ 8 & 7 & 9 & 5 \\ 6 & 7 & 9 & 8 \end{bmatrix} = \begin{bmatrix} 2 & 1 & 1 & 0 \\ 0 & 1 & 1 & 1 \\ 0 & 3 & 5 & 5 \\ 0 & 4 & 6 & 8 \end{bmatrix}$$

$$G_2 G_1 A = \begin{bmatrix} 1 & 0 & 0 & 0 \\ 0 & 1 & 0 & 0 \\ 0 & -3 & 1 & 0 \\ 0 & -4 & 0 & 1 \end{bmatrix} \begin{bmatrix} 2 & 1 & 1 & 0 \\ 0 & 1 & 1 & 1 \\ 0 & 3 & 5 & 5 \\ 0 & 4 & 6 & 8 \end{bmatrix} = \begin{bmatrix} 2 & 1 & 1 & 0 \\ 0 & 1 & 1 & 1 \\ 0 & 0 & 2 & 2 \\ 0 & 0 & 2 & 4 \end{bmatrix}$$

$$G_3 G_2 G_1 A = \begin{bmatrix} 1 & 0 & 0 & 0 \\ 0 & 1 & 0 & 0 \\ 0 & 0 & 1 & 0 \\ 0 & 0 & -1 & 1 \end{bmatrix} \begin{bmatrix} 2 & 1 & 1 & 0 \\ 0 & 1 & 1 & 1 \\ 0 & 0 & 2 & 2 \\ 0 & 0 & 2 & 4 \end{bmatrix} = \begin{bmatrix} 2 & 1 & 1 & 0 \\ 0 & 1 & 1 & 1 \\ 0 & 0 & 2 & 2 \\ 0 & 0 & 0 & 2 \end{bmatrix} = U$$

Then: $(G_3 G_2 G_1)^{-1} = \begin{bmatrix} 1 & 0 & 0 & 0 \\ 2 & 1 & 0 & 0 \\ 4 & 3 & 1 & 0 \\ 3 & 4 & 1 & 1 \end{bmatrix} = L \quad \text{and} \quad A = (G_3 G_2 G_1)^{-1} (G_3 G_2 G_1 A) = LU$

Basic LU factorization algorithm (not recommended, see next)

Algorithm 1 Basic LU factorization

```
 $A \in \mathbb{K}^{n \times n}$  (data)
for  $k = 1$  to  $n - 1$  do
  for  $j = k + 1$  to  $n$  do
     $a_{jk} = a_{jk} / a_{kk}$  (compute  $jk$ -th entry of  $L$ )
     $a_{j,(k+1):n} = a_{j,(k+1):n} - a_{jk} a_{k,(k+1):n}$  (update entries  $U_{j,(k+1):n}$  of  $U$ )
  end for
end for
```

Algorithm breaks down if top left entry of any of A_1, A'_2, A''_3, \dots (i.e. $a_{11}, a'_{22}, a''_{33} \dots$) is zero.

Computational complexity (counting arithmetic operations in algorithm):

$$\text{operation count of LU} = \sum_{k=1}^{n-1} (n-k)(2n-2k+1) = \frac{2}{3}n(n^2-1) \sim \frac{2}{3}n^3$$

Problems with basic LU factorization

- **LU factorization may fail on “nice” matrices.** Examples:

$$A = \begin{bmatrix} 0 & 1 \\ 1 & 1 \end{bmatrix}, \quad B = \begin{bmatrix} 1 & 2 & 1 & 1 \\ 1 & 3 & 2 & 2 \\ 1 & 2 & 1 & 2 \\ 2 & 5 & 4 & 5 \end{bmatrix} \quad (\text{zero pivot will appear at } b_{33})$$

A has basic LU factorization if $\det(A_{1:k,1:k}) \neq 0$ for $1 \leq k \leq n-1$ (may fail for invertible A)

- **LU factorization may produce ill-conditioned L, U .**
- **LU factorization may produce too-large L, U .** Example:

$$A_\varepsilon = \begin{bmatrix} \varepsilon & 1 \\ 1 & 1 \end{bmatrix} = L_\varepsilon U_\varepsilon \quad \text{with} \quad L_\varepsilon = \begin{bmatrix} 1 & 0 \\ -\varepsilon^{-1} & 1 \end{bmatrix}, \quad U_\varepsilon = \begin{bmatrix} \varepsilon & 1 \\ 0 & 1 - \varepsilon^{-1} \end{bmatrix}$$

$$\kappa_2(A_\varepsilon) = (3 + \sqrt{5})/2 + O(\varepsilon), \quad \kappa_2(L_\varepsilon) = 1 + O(\varepsilon) \quad \text{but} \quad \kappa_2(U_\varepsilon) = O(\varepsilon^{-2})$$

$|L_\varepsilon|, |U_\varepsilon| = O(\varepsilon^{-1})$ (large L, U factors)

roundoff errors from LU factorization

Assume A (i) made of floating-point numbers, (ii) no zero pivot found in LU factorization. Computed factors \tilde{L}, \tilde{U} then satisfy $\tilde{L}\tilde{U} = A + E$, $|E| \leq 3(n-1)\varepsilon_{\text{mach}}(|A| + |L||U|)$

- **Why these problems?** Factorization may encounter small pivots a_{kk} .

Pivoting

General idea: permute row and/or columns to move larger entries of A to diagonal.

Pivoting methods:

- Full (row and column) pivoting, step k : find largest entry in A'_k . Quite expensive for large matrices.
- Row pivoting, step k : seek largest entry in $\begin{bmatrix} a'_{kk} \\ l'_k \end{bmatrix}$
- Rook pivoting: alternate row/column searches until finding pivot largest in row and column.

LU factorization with row pivoting

Let $A \in \mathbb{K}^{n \times n}$ invertible. There exists (i) a permutation matrix Π , (ii) lower / upper triangular factors L, U such that $\Pi A = LU$.

LU -factorized form of system $Ax = b$ is

$$LUx = \Pi b \quad (\text{apply permutation to } b, \text{ then solve as before}).$$

LU factorization with row pivoting: proof

Proof on small matrix $A \in \mathbb{K}^{4 \times 4}$:

- assume row pivoting at each LU step:

$$G_3 \Pi_3 G_2 \Pi_2 G_1 \Pi_1 A = U$$

i.e. $G_3 \underbrace{(\Pi_3 G_2 \Pi_3^T)}_{\hat{G}_2} \underbrace{(\Pi_3 \Pi_2 G_1 \Pi_2^T \Pi_3^T)}_{\hat{G}_1} \underbrace{\Pi_3 \Pi_2 \Pi_1}_{\Pi} A = U$

i.e. $G_3 \hat{G}_2 \hat{G}_1 \Pi A = U$

- Process generalizes to $A \in \mathbb{K}^{n \times n}$, and

$$G_{n-1} \hat{G}_{n-2} \hat{G}_{n-3} \dots \hat{G}_1 \Pi A = U$$

i.e. $\Pi A = LU, \quad L = \hat{G}_1^{-1} \hat{G}_1^{-2} \dots \hat{G}_{n-2}^{-1} G_{n-1}^{-1}$

SPD matrices: LDL^T and Cholesky factorizations

- $A \in \mathbb{K}^{n \times n}$ Hermitian with $x^H A x > 0$ for all $x \neq 0$ is called *symmetric positive definite* (SPD).
- Classical examples: stiffness or mass matrix in finite element method, covariance matrices
- LU factorization applicable but **sub-optimal**

Idea: find **symmetric** factorization of A SPD via **symmetric** elimination:

$$A = \begin{bmatrix} a_{11} & \ell_1^H \\ \ell_1 & A_2 \end{bmatrix}, \quad G_1(z) := \begin{bmatrix} 1 & 0 \\ z & I_{n-1} \end{bmatrix} = \begin{bmatrix} 1 & & & \\ z_2 & 1 & & \\ \vdots & & \ddots & \\ z_n & & & 1 \end{bmatrix} \quad (1)$$

$$\Rightarrow G_1(-\ell_1/a_{11}) A G_1^H(-\ell_1/a_{11}) = \begin{bmatrix} a_{11} & 0 \\ 0 & A'_2 \end{bmatrix}, \quad A'_2 := A_2 - \ell_1 \ell_1^H / a_{11} \quad (2)$$

(A SPD and G invertible) $\Rightarrow G A G^H$ SPD (exercise), hence A'_2 is SPD.

Continuing elimination yields the **LDL^T factorization** of A :

$$A = L D L^H \quad \text{with } D \text{ diagonal SPD, i.e. } D = \text{diag}(d_{11}, \dots, d_{nn}) > 0$$

$$L = G_1(\ell_1/a_{11}) G_2(\ell'_2/a'_{22}) \dots G_{n-1}(\ell'_{n-1}/a'_{n-1,n-1}) \quad \text{lower-triangular.}$$

(L lower-triangular: see derivation of LU factorization).

SPD matrices: LDL^T and Cholesky factorizations

- Variant of LDL^T : the **Cholesky factorization**

$$A = LDL^H \implies \boxed{A = GG^H \text{ with } G := LD^{1/2}} \quad (G \text{ lower-triangular})$$

Backward stability of Cholesky factorization

Let $A \in \mathbb{K}^{m \times n}$ with $m \geq n$. Let \tilde{G} be its Cholesky factor computed (in finite precision) using Algorithm 13. Then there exists a matrix $E \in \mathbb{K}^{m \times n}$ such that

$$A + E = \tilde{G}\tilde{G}^H \quad \text{with} \quad \|E\|/\|A\| = O(\varepsilon_{\text{mach}})$$

- Solution method:

compute $A = GG^H$, then: (i): solve $Gy = b$; (ii): solve $G^Hx = y$

Cholesky factorization

Algorithm 2 Cholesky factorization (derived from equations $a_{ij} = \sum_{k=i}^j \bar{g}_{ki} g_{kj}$)

```
1:  $A \in \mathbb{K}^{n \times n}$  (data – only lower triangular part need be stored)
2: for  $k = 1$  to  $n$  do
3:    $a_{kk} = [a_{kk} - (a_{k,1:k-1})^H a_{k,1:k-1}]^{1/2}$  (Compute diagonal entry of Cholesky factor  $G$ )
4:   for  $j = k+1$  to  $n$  do
5:      $a_{jk} = a_{jk} - [(a_{j,1:k-1})^H a_{k,1:k-1}] / a_{kk}$  (update  $j$ -th column of Cholesky factor  $G$ )
6:   end for
7: end for
```

Computational complexity of Cholesky factorization:

$$\text{operation count} = \sum_{k=1}^n \left\{ 2(k-1) + \sum_{j=k+1}^n (2k-1)(n-k-1) \right\} \sim \frac{1}{3} n^3$$

Outlook

- So far, **square, uniquely solvable** systems $Ax = b$
- Two quite classical, “workhorse” algorithms:
 - LU factorization (square invertible), needs $\sim 2n^3/3$ operations if A dense
 - LDL^T /Cholesky factorization (SPD), needs $\sim n^3/3$ operations if A dense
- Not covered: symmetric indefinite (**not** SPD).
- For example, Cholesky a classical choice in finite element methods.

What if system $Ax = b$ **not square**?

Least-squares problems

- So far, **square, uniquely solvable** systems $Ax = b$
- However, more annoying systems often come up:
 - $A \in \mathbb{K}^{n \times n}$ square but not invertible
 - $A \in \mathbb{K}^{m \times n}$ not square (often $m > n$, occasionally $m < n$)

Data analysis, interpretation of experiments with models...

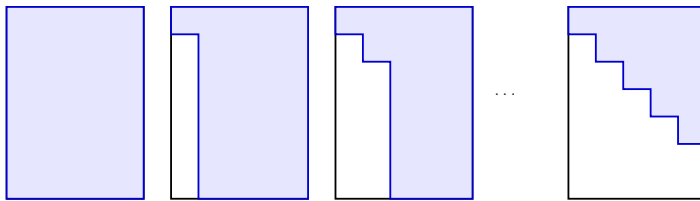
- Solution then (i) may not exist, (ii) is multiple if it exists.
Common approach: seek **least-squares (LS) solutions**:

$$\text{find } x \in \mathbb{K}^n, \|Ax - b\|^2 \rightarrow \min$$

- Solution methods for LS problems provide many of the **general-purpose tools** of computational linear algebra. In particular (see later), very useful components of
 - Eigenvalue algorithms
 - Data compression, image processing
- Introduction of two new fundamental tools of numerical linear algebra:
 - QR factorization
 - Singular value decomposition

The QR factorization

- LU (square invertible), LDL^T (square SPD)...
Factorization for (arbitrary, rectangular) $A \in \mathbb{K}^{m \times n}$?
- General approach: **zeroing out** columns under diagonal (as with LU):



- However, Gaussian elimination (LU style) not a great approach, as A is arbitrary:
 - Could meet zero pivots
 - Potential conditioning issues
- Instead, seek to zero out columns using **unitary matrices**, e.g.

$$A = \begin{bmatrix} a_{11} & u_1 \\ l_1 & A_2 \end{bmatrix} \rightarrow FA = \begin{bmatrix} a'_{11} & u_1 \\ 0 & A'_2 \end{bmatrix} \dots \quad (F^H F = I)$$

Will prove **very suitable** for least squares.

Householder reflections

- Householder reflection:

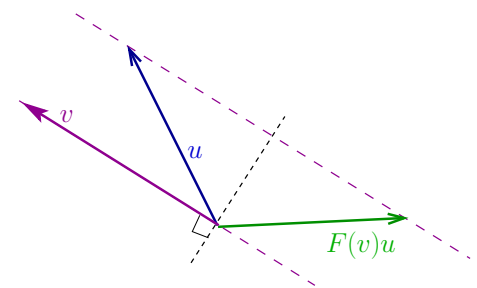
$$F(v) = I_m - 2 \frac{v v^H}{\|v\|^2}$$

$$(v \in \mathbb{K}^m, F(v) \in \mathbb{K}^{m \times m})$$

$$F(v)^H F(v) = I_m$$

vv^H is a **rank-one** matrix: $(vv^H)v = (v^Hv)v$ and $(vv^H)v_\perp = 0$

$$F(v)v = -v, \quad F(v)v_\perp = v_\perp$$

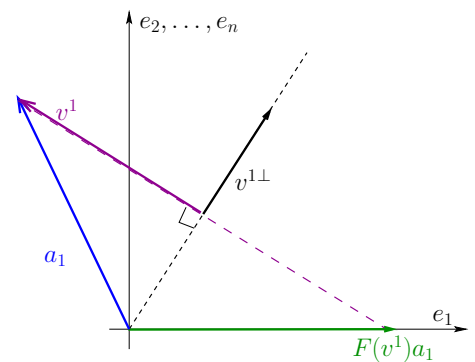


- First column of $A \in \mathbb{K}^{m \times n}$, $m \geq n$: choosing

$$v^1 = \begin{Bmatrix} a_{11} + \text{sign}(a_{11}) \|a_1\| \\ \ell_1 \end{Bmatrix} \quad (\text{sign}(x) := x/|x|)$$

we find

$$A = \begin{bmatrix} a_{11} & u_1 \\ \ell_1 & A_2 \end{bmatrix} \rightarrow F(v^1)A = \begin{bmatrix} -\text{sign}(a_{11})a_{11} & u'_1 \\ 0 & A'_2 \end{bmatrix}$$



Householder reflections

- Continuing the process:

$$v^2 = \begin{Bmatrix} 0 \\ a'_{22} + \text{sign}(a'_{22}) \|a'_2\| \\ \ell'_2 \end{Bmatrix}, \quad v^3 = \begin{Bmatrix} 0 \\ 0 \\ a''_{33} + \text{sign}(a''_{33}) \|a''_3\| \\ \ell'_3 \end{Bmatrix}, \quad \dots$$

we reach

$$F(v^n) F(v^{n-1}) \dots F(v^2) F(v^1) A = R, \quad R = \begin{bmatrix} r_{11} & \dots & r_{1n} \\ 0 & \ddots & \vdots \\ \vdots & & r_{nn} \\ 0 & \dots & 0 \end{bmatrix}$$

Moreover, $F(v^n) F(v^{n-1}) \dots F(v^2) F(v^1)$ is **unitary** (as product of unitary matrices), so:

$$\boxed{A = QR, \quad Q := F(v^1)^H F(v^2)^H \dots F(v^{n-1})^H F(v^n)^H}, \quad \begin{array}{l} Q \in \mathbb{K}^{m \times m} \text{ unitary} \\ R \in \mathbb{K}^{m \times n} \text{ upper triangular} \end{array}$$

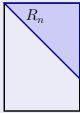
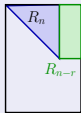
QR factorization ($A \in \mathbb{K}^{m \times n}$, $m \geq n$)

QR factorization

Let $A \in \mathbb{K}^{m \times n}$, $m \geq n$. There exist $Q \in \mathbb{K}^{m \times m}$ unitary, $R \in \mathbb{K}^{m \times n}$ upper triangular, and a permutation $\Pi \in \mathbb{R}^{n \times n}$ such that

$$A\Pi = QR.$$

More precisely, with $r \leq \min(m, n)$ the rank of A :

- (a) If $r = n$: $A = QR$, $R = \begin{bmatrix} R_n \\ 0 \end{bmatrix}$,  $R_n \in \mathbb{K}^{n \times n}$ upper triangular invertible.
- (b) If $r < n$: $A = QR\Pi^T$, $R = \begin{bmatrix} R_r & R_{n-r} \\ 0 & 0 \end{bmatrix}$,  $R_r \in \mathbb{K}^{r \times r}$ upper triangular invertible,
 $R_{n-r} \in \mathbb{K}^{r \times (n-r)}$

QR factorization ($A \in \mathbb{K}^{m \times n}$, $m \geq n$)

Algorithm 3 QR factorization using Householder reflections

```
1:  $A \in \mathbb{K}^{m \times n}$  (data)
2: for  $k = 1$  to  $n$  do
3:    $x = A_{k:m,k}$  (part of  $k$ -th column below diagonal of  $A$  (included))
4:    $\gamma = \text{sign}(x_1)\|x\|$ 
5:    $v = (x + \gamma e_1)/(x_1 + \gamma)$  (Householder vector  $v^k$ , normalized so that  $v_1^k = 1$ )
6:    $\beta_k = 1/(x_1 + \gamma)\gamma$  (normalization factor  $2/\|v\|^2$  for  $F(v^k)$ )
7:    $a_{kk} = -\gamma$  (Apply  $F(v^k)$  to column  $k$  of  $A$ , only computing leading entry)
8:    $A_{k:m,(k+1):n} = A_{k:m,(k+1):n} - \beta_k v(v^H A_{k:m,(k+1):n})$ 
   (Apply  $F(v^k)$  to block of  $A$  hanging on the right of  $a_{kk}$ )
9:    $A_{(k+1):m,k} = v_{2:(m-k+1)}$  (store  $v^k$  minus leading entry in column  $k$  of  $A$ )
10: end for
```

Backward stability of QR factorization

Let $A \in \mathbb{K}^{m \times n}$ with $m \geq n$. Let $\tilde{Q}\tilde{R}$ be the QR factorization of A computed (in finite precision) using Algorithm 3. Then there exists $E \in \mathbb{K}^{m \times n}$ such that

$$A + E = \tilde{Q}\tilde{R} \quad \text{with} \quad \|E\|/\|A\| = O(\varepsilon_{\text{mach}})$$

Computational complexity of QR factorization: operation count $\sim \frac{2}{3}n^2(3m - n)$.

QR factorization ($A \in \mathbb{K}^{m \times n}$, $m \geq n$)

- QR factorization applicable to any matrix.
- Computing cost of QR factorization: $\sim \frac{4}{3}n^3$ ops. for square A , i.e. twice the LU cost.
- LU factorization for rectangular A may or may not work
- QR factorization is rank-revealing: rank r of A is size of R_r .

Solvability of linear systems $Ax = b$ ($A \in \mathbb{K}^{m \times n}$, $m \geq n$, rank $r \leq n$):

$$Ax = b \implies QR\Pi^T x = b \implies Rx' = Q^H b \quad (\text{with } x' := \Pi^T x),$$

(a) If $r = n$:

$$\begin{bmatrix} R_n \\ 0 \end{bmatrix} \begin{Bmatrix} x' \end{Bmatrix} = \begin{Bmatrix} (Q^H b)_{1:n} \\ (Q^H b)_{n+1:m} \end{Bmatrix} \quad \begin{array}{l} \text{If } (Q^H b)_{n+1:m} = 0 \quad \text{unique solution} \\ \text{If } (Q^H b)_{n+1:m} \neq 0 \quad \text{no solution} \end{array} \quad \boxed{R_n x' = (Q^H b)_{1:n}}$$

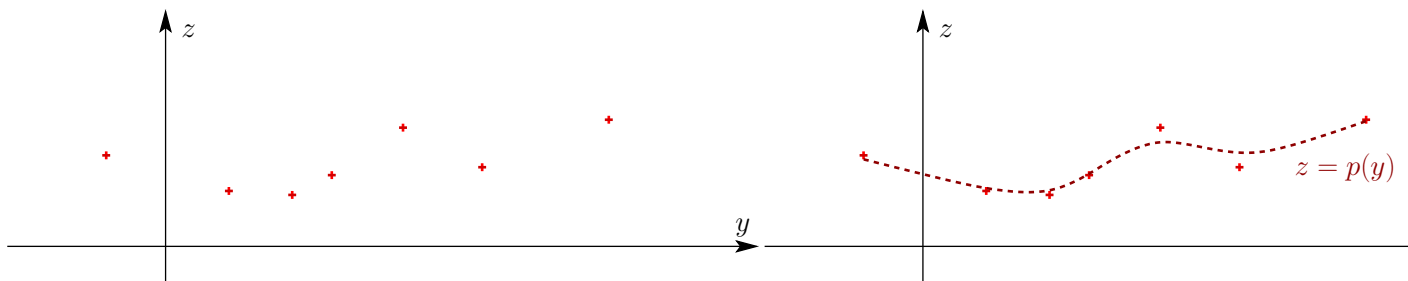
(b) If $r < n$:

$$\begin{bmatrix} R_r & R_{n-r} \\ 0 & 0 \end{bmatrix} \begin{Bmatrix} x'_{1:r} \\ x'_{r+1:n} \end{Bmatrix} = \begin{Bmatrix} (Q^H b)_{1:r} \\ (Q^H b)_{r+1:m} \end{Bmatrix}$$

$$\begin{array}{l} \text{If } (Q^H b)_{r+1:m} = 0 \quad \text{multiple solutions} \\ \text{If } (Q^H b)_{r+1:m} \neq 0 \quad \text{no solution} \end{array} \quad \boxed{R_r x'_{1:r} = (Q^H b)_{1:r} - R_{n-r} x'_{r+1:n}}$$

Least squares problems

- **Data fitting example:** find best degree- n polynomial $p(y)$ to fit $m \geq n$ data points $(y_i, z_i)_{1 \leq i \leq m}$:



Find $x := \{a_0, a_1, \dots, a_n\}^T$, $J(x) := \sum_{i=1}^m |a_0 + a_1 y_i \dots + a_n y_i^n - z_i|^2 \rightarrow \min$

Matrix form:

$$J(x) := \|Ax - b\|^2 \rightarrow \min, \quad A = \begin{bmatrix} 1 & y_1 & \dots & y_1^n \\ \vdots & & & \vdots \\ 1 & y_m & \dots & y_m^n \end{bmatrix}, \quad x = \begin{Bmatrix} a_0 \\ a_1 \\ \vdots \\ a_n \end{Bmatrix}, \quad b = \begin{Bmatrix} z_1 \\ \vdots \\ z_m \end{Bmatrix}$$

1st-order (necessary) optimality condition: $\nabla_x J = 0$, i.e. $A^H A x = A^H b$ ("normal equations")

- **General linear least squares problem:** find a best solution in the 2-norm sense:

$$\min_{x \in \mathbb{K}^n} \|Ax - b\|^2.$$

- LS solution(s) always exist (since we work in finite dimension!)
- If $\min_{x \in \mathbb{K}^n} \|Ax - b\|^2 = 0$, x solves $Ax = b$ (obviously).

Least squares problems

- Solvability of $\min_{x \in \mathbb{K}^n} \|Ax - b\|^2$? Use QR factorization $A\Pi = QR$, set $x' = \Pi^T x = \Pi^{-1}x$:

$$\begin{aligned} \|Ax - b\|^2 &= \|QRx' - b\|^2 = \|Q(Rx' - Q^H b)\|^2 = \|Rx' - Q^H b\|^2 \\ &= \underbrace{\|R_r x'_{1:r} + R_{n-r} x'_{r+1:n} - (Q^H b)_{1:r}\|^2}_{\text{minimize w.r.t. } x} + \underbrace{\|(Q^H b)_{r+1:n}\|^2}_{\text{fixed}}, \end{aligned}$$

- Set to zero the x -dependent part of the residual:

(a) If $r = n$:

$$R_n x' = (Q^H b)_{1:n}$$

unique solution $R_n x' = (Q^H b)_{1:n}$

(b) If $r < n$:

$$R_r x'_{1:r} + R_{n-r} x'_{r+1:n} = (Q^H b)_{1:r}$$

multiple solutions $R_r x'_{1:r} = (Q^H b)_{1:r} - R_{n-r} x'_{r+1:n}$

- Least squares residual:

$$\min_{x \in \mathbb{K}^n} \|Ax - b\|^2 = \|(Q^H b)_{r+1:n}\|^2$$

If residual is zero, $Ax = b$ has a solution.

Singular value decomposition (SVD)

- Diagonalization of square matrices: $A = X\Lambda X^{-1}$ for some invertible X and diagonal Λ
 - A normal if and only if A diagonalizable with X unitary. Includes all Hermitian matrices.
 - There are (non-normal) non-diagonalizable matrices, e.g. $A = \begin{bmatrix} a & b \\ 0 & a \end{bmatrix}$

Singular value decomposition generalizes diagonalization to any (even rectangular) matrix.

- Let $A \in \mathbb{K}^{m \times n}$, then $A^H A \in \mathbb{K}^{n \times n}$ and $AA^H \in \mathbb{K}^{m \times m}$ are square Hermitian.

Well-defined (symmetric positive) eigenvalue problems: $A^H A v = \lambda v$ and $AA^H u = \mu u$

(λ, v) eigenpair of $A^H A \implies (\lambda, Av)$ eigenpair of AA^H

(μ, u) eigenpair of $AA^H \implies (\mu, A^H u)$ eigenpair of $A^H A$

(equal multiplicities if $\lambda = \mu > 0$, unequal multiplicity in general for $\lambda = \mu = 0$).

Singular value decomposition

Any $A \in \mathbb{K}^{m \times n}$ has a SVD $A = USV^H$, where:

- $U = [u_1, \dots, u_m] \in \mathbb{K}^{m \times m}$, $V = [v_1, \dots, v_n] \in \mathbb{K}^{n \times n}$: unitary square matrices,
- $S \in \mathbb{R}^{m \times n}$ "diagonal", with $S_{ij} = \sigma_i$, $S_{ij} = 0$ if $i \neq j$.
- **Singular values** σ_i are real positive; conventionally $\sigma_1 \geq \sigma_2 \geq \dots \geq \sigma_{\min(m,n)} \geq 0$.

- Available operators: $[U, S, V] = \text{svd}(A)$ (MATLAB), $F = \text{svdfact}(A)$ (Julia; F contains U, S, V)

Some properties

- $\lambda_i = \mu_i = \sigma_i^2$ ($1 \leq i \leq r$).
- $A = USV^H = \sum_{i=1}^{\min(m,n)} \sigma_i u_i v_i^H$ (SVD is weighted sum of rank-one matrices).
- **Reduced SVD.** For $A \in \mathbb{K}^{m \times n}$, $\text{rank}(A) = r \leq \min(m, n)$:

$$A = USV^H = U_r S_r V_r^H \quad U_r = [u_1, \dots, u_r], \quad S_r = \text{diag}(\sigma_1, \dots, \sigma_r), \quad V_r = [v_1, \dots, v_r]$$

(vectors u_{r+1}, \dots, u_m and v_{r+1}, \dots, v_n inactive, generate $\mathcal{N}(A)$ and $\mathcal{R}(A)^\perp$).

- SVD is **rank-revealing**: $\text{rank}(A)$ equal to number of nonzero singular values.
- Matrix 2-norm: we have $\|A\|_2 = \sigma_1$, since
$$\|Ax\|_2 = \|U_r S_r V_r x\|_2 = \|S_r V_r x\|_2 \leq \|S_r\|_2 \|V_r x\|_2 \leq \|S_r\|_2 \|x\|_2 = \sigma_1 \|x\|_2$$
- Computing a SVD
 - requires solving an eigenvalue problem (see later),
 - takes $O(m^2 n)$ operations for $A \in \mathbb{K}^{m \times n}$.

Singular value decomposition (SVD): example

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

Full SVD:

$$\boxed{A = USV^H}, \quad U = \begin{bmatrix} 1 & 0 & 0 \\ 0 & 0 & 1 \\ 0 & 1 & 0 \end{bmatrix}, \quad S = \begin{bmatrix} 2 & 0 & 0 & 0 \\ 0 & 1 & 0 & 0 \\ 0 & 0 & 0 & 0 \end{bmatrix}, \quad V = \begin{bmatrix} 0 & 0 & 1 & 0 \\ 1 & 0 & 0 & 0 \\ 0 & 1 & 0 & 0 \\ 0 & 0 & 0 & 1 \end{bmatrix}$$

- $\text{rank}(A) = 2$
- $\sigma_1 = 2, \sigma_2 = 1, \sigma_3 = 0$

Reduced SVD:

$$\boxed{A = U_r S_r V_r^H}, \quad U_r = \begin{bmatrix} 1 & 0 \\ 0 & 0 \\ 0 & 1 \end{bmatrix}, \quad S_r = \begin{bmatrix} 2 & 0 \\ 0 & 1 \end{bmatrix}, \quad V_r = \begin{bmatrix} 0 & 0 \\ 1 & 0 \\ 0 & 1 \\ 0 & 0 \end{bmatrix}$$

SVD applied to arbitrary linear systems

Solvability of $Ax = b$ (again):

$$Ax = b \implies USV^H x = b \implies S(V_r^H x) = U^H b$$

$$\text{i.e. } \begin{cases} \sigma_i(v_i^H x) = u_i^H b & (1 \leq i \leq r) \\ 0 = u_i^H b & (r+1 \leq i \leq m) \end{cases}$$

- Solvability condition: $u_i^H b = 0$ ($r+1 \leq i \leq m$), expresses $b \in \mathcal{R}(A)$.

Then, $\sigma_i(v_i^H x) = u_i^H b$ ($1 \leq i \leq r$) determine projections $v_i^H x$ uniquely

Remaining projections $v_i^H x$ ($r+1 \leq i \leq m$) **arbitrary**.

- General solution (if it exists):

$$x = \sum_{i=1}^r \frac{u_i^H b}{\sigma_i} v_i + \sum_{i=r+1}^n x_i v_i \quad (x_{r+1}, \dots, x_n) \in \mathbb{K}^{n-r} \text{ arbitrary}$$

Setting $x_{r+1} = \dots = x_n = 0$ gives **minimum-norm solution**

- Uniqueness condition: $r = n$ (implies $m \geq n$).

SVD applied to least-squares problems

Linear least-squares problem $\min_{x \in \mathbb{K}^n} \|Ax - b\|^2$:

$$\begin{aligned}\|Ax - b\|^2 &= \|USV^H x - b\|^2 = \|U(SV^H x - U^H b)\|^2 = \|S(V^H x) - U^H b\|^2 \\ &= \sum_{i=1}^r |\sigma_i (v_i^H x) - (u_i^H b)|^2 + \sum_{i=r+1}^m |u_i^H b|^2\end{aligned}$$

Solutions:

$$x = \sum_{i=1}^r \frac{u_i^H b}{\sigma_i} v_i + \sum_{i=r+1}^n x_i v_i \quad (x_{r+1}, \dots, x_n) \in \mathbb{K}^{n-r} \text{ arbitrary}$$

Residual:

$$\min_{x \in \mathbb{K}^n} \|Ax - b\|^2 = \sum_{i=r+1}^n |u_i^H b|^2$$

Pseudo-inverse of a matrix

Generalized inverse (i.e. pseudo-inverse) of $A \in \mathbb{K}^{m \times n}$: a matrix $A^+ \in \mathbb{K}^{n \times m}$ verifying

(a) $AA^+A = A,$	(b) $(AA^+)^H = AA^+,$
(c) $A^+AA^+ = A^+,$	(d) $(A^+A)^H = A^+A.$

(Moore-Penrose conditions)

Algebraic properties of A^+

- $(A^+)^+ = A.$
- (Moore-Penrose) Pseudo-inverse A^+ satisfying (a)-(d) exists and is unique
- If A invertible, $A^+ = A^{-1}$
- If $\text{rank}(A) = n$ (full column rank, hence $m \geq n$, $A^H A$ invertible): $A^+ = (A^H A)^{-1} A^H.$
- If $\text{rank}(A) = m$ (full row rank, hence $m \leq n$, AA^H invertible), $A^+ = A^H (AA^H A)^{-1}.$

- Explicit formula using reduced SVD:

$$A^+ = V_r S_r^{-1} U_r^H$$

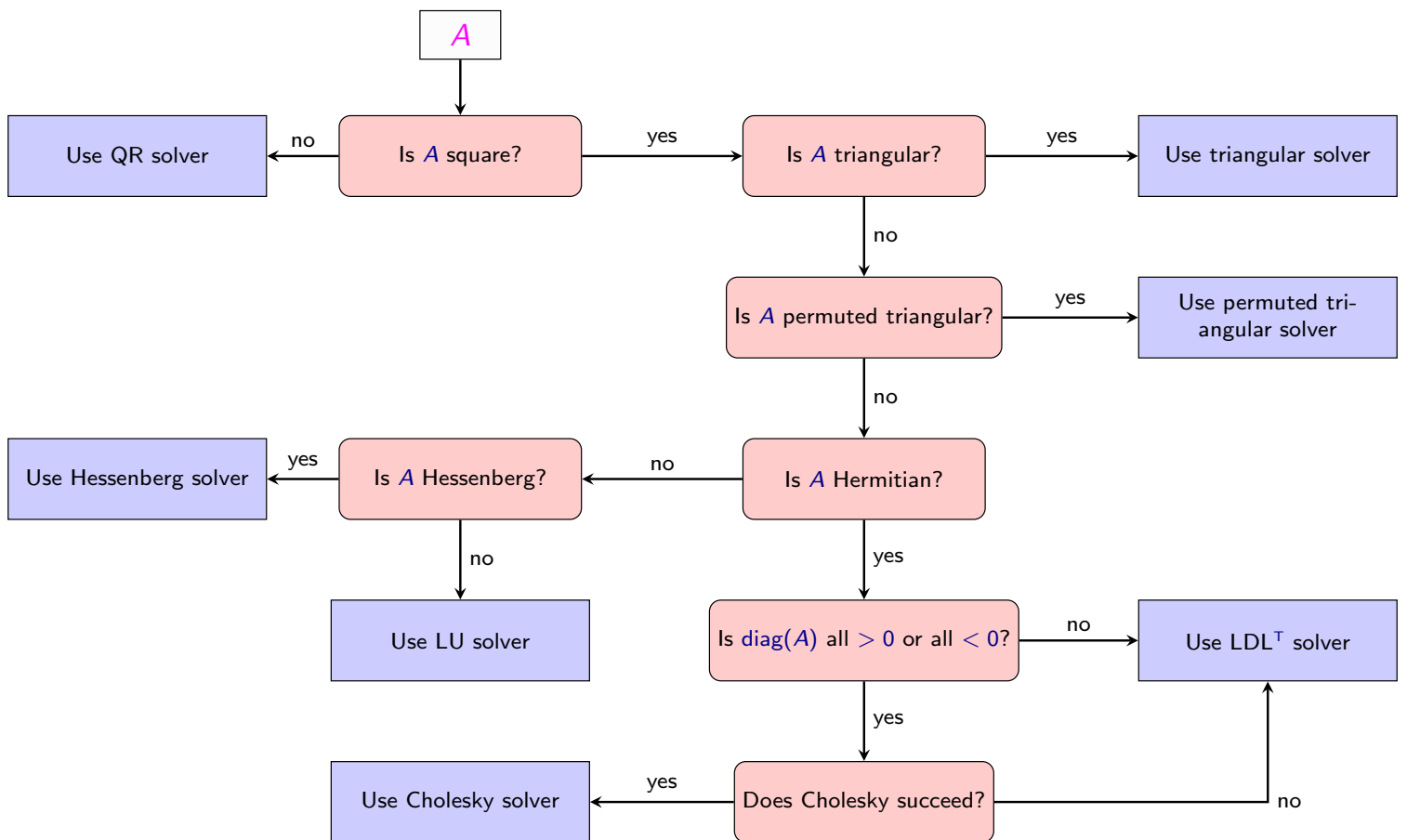
- General solution least-squares problem:

$$x = A^+ b + (I - A^+ A) w, \quad w \in \mathbb{K}^n \text{ arbitrary}$$

- A^+ does not depend continuously on A . Example ($\text{rank}(A) = r$, $\text{rank}(A_\varepsilon) = r+1$):

$$A_\varepsilon = U_r S_r V_r^H, \quad A_\varepsilon = U_r S_r V_r^H + \varepsilon u_{r+1} v_{r+1}^H \implies \frac{\|A_\varepsilon^+ - A^+\|}{\|A^+\|} = \frac{\sigma_r}{\varepsilon} \gg 1$$

The backslash operator by way of summary



Flowchart of the backslash operator “\” of MATLAB