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.:

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

  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).

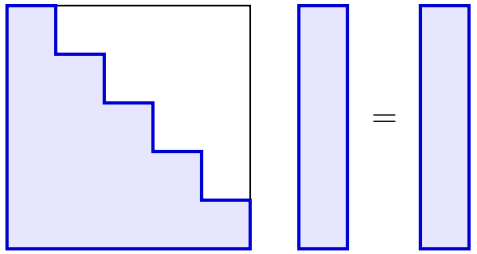
- This course: LU, Cholesky, QR factorizations.

Triangular systems

Systems with triangular matrices: Easy to solve numerically!

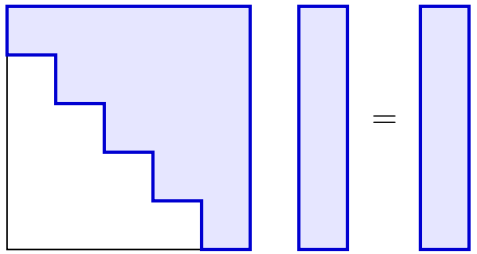
- A lower triangular: forward substitution

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



- A upper triangular: backward substitution

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



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

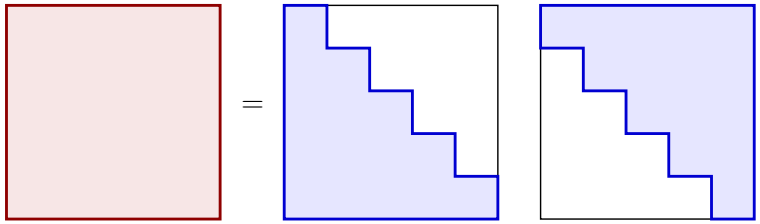
Backward stability of forward/backward substitution

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

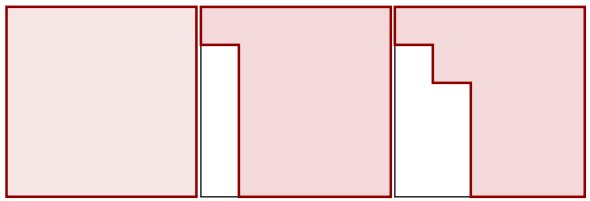
For $A \in \mathbb{K}^{n\times n}$ invertible: $A = LU$ (L, U: lower / upper triangular – may need pivoting, see later)



Solution method:

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

Gaussian elimination:



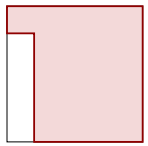
...

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):

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



Computation of LU factorization (continued)

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

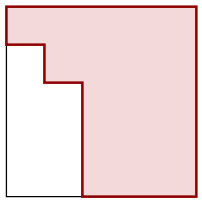
$$A = \begin{bmatrix} a_{11} & u_1 \\ \ell_1 & A_2 \end{bmatrix}, \qquad G_1(-\ell_1/a_{11})A = \begin{bmatrix} 1 & 0 \\ -\ell_1/a_{11} & I_{n-1} \end{bmatrix} \begin{bmatrix} a_{11} & u_1 \\ \ell_1 & A_2 \end{bmatrix} = \begin{bmatrix} a_{11} & u_1 \\ 0 & A_2' \end{bmatrix}.$$

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

- 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):

$$A_2' = \begin{bmatrix} a_{22}' & u_2' \\ \ell_2' & A_3' \end{bmatrix}, \qquad G_2(-\ell_2'/a_{22}')G_1(-\ell_1/a_{11})A = \begin{bmatrix} a_{11} & u_1 & \\ 0 & a_{22}' & u_2' \\ 0 & 0 & A_3'' \end{bmatrix}$$



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

$$G_{n-1}(-\ell_{n-1}/a_{n-1,n-1}) \ldots G_2(-\ell_2'/a_{22}')\, G_1(-\ell_1/a_{11})A = U.$$

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

$$\boxed{A = G_1(\ell_1/a_{11})\, G_2(\ell_2'/a_{22}') \ldots G_{n-1}(\ell_{n-1}/a_{n-1,n-1})U, \quad \text{i.e.} \quad A = LU.}$$

Example (with no roundoff errors)

$$A = \begin{bmatrix} 2 & 1 & 1 & 0 \\ 4 & 3 & 3 & 1 \\ 8 & 7 & 9 & 5 \\ 6 & 7 & 9 & 8 \end{bmatrix} = \begin{bmatrix} a_{11} & u_1 \\ \ell_1 & A_2 \end{bmatrix} \qquad \ell_1 = \begin{bmatrix} 4 \\ 8 \\ 6 \end{bmatrix}, \quad u_1 = \begin{bmatrix} 1 & 1 & 0 \end{bmatrix}, \quad A_2 = \begin{bmatrix} 3 & 3 & 1 \\ 7 & 9 & 5 \\ 7 & 9 & 8 \end{bmatrix}$$

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

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 $A_1, A_2', A_3'', \ldots$ (i.e. $a_{11}, a_{22}', a_{33}'' \ldots$) is zero.

Computational complexity (counting arithmetic operations in algorithm):

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

Problems with basic LU factorization

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

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

A has basic LU factorization if $\det\big(A_{1:k,1:k} \neq 0\big)$ 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:

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

$\kappa_2(A_\varepsilon) = (3+\sqrt{5})/2 + O(\varepsilon)$, $\kappa_2(L_\varepsilon) = 1 + O(\varepsilon)$ but $\kappa_2(U_\varepsilon) = O(\varepsilon^{-2})$
$|L_\varepsilon|$, $|U_\varepsilon| = O(\varepsilon^{-1})$ (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 $\widetilde{L}, \widetilde{U}$ then satisfy $\widetilde{L}\widetilde{U} = A + E$, $|E| \leq 3(n-1)\varepsilon_{\text{mach}}\big(|A| + |L||U|\big)$

- **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 $\begin{bmatrix} a'_{kk} \\ \ell'_k \end{bmatrix}$
- 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

$LUx = \Pi b$ (apply permutation to b, 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. $$\Pi A = LU, \qquad L = \hat{G}_1^{-1} \hat{G}_1^{-2} \dots \hat{G}_{n-2}^{-1} G_{n-1}^{-1}$$

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:

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

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

(A SPD and G invertible) $\Longrightarrow$ GAG^H SPD (exercise), hence $\boxed{A_2' \text{ is SPD}}$.

Continuing elimination yields the LDL^T factorization of A:

$A = LDL^H$ with D diagonal SPD, i.e. $D = \mathrm{diag}(d_{11}, \ldots, d_{nn}) > 0$

$L = G_1(\ell_1/a_{11})\, G_2(\ell_2'/a_{22}') \ldots G_{n-1}(\ell_{n-1}'/a_{n-1,n-1}')$ lower-triangular.

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

SPD matrices: LDL^T and Cholesky factorizations

- Variant of LDL^T: the Cholesky factorization

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

Backward stability of Cholesky factorization

Let $A \in \mathbb{K}^{m\times n}$ with $m \geq n$. Let $\widetilde{G}$ 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

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

- Solution method:

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

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: $\quad a_{kk} = [a_{kk} - (a_{k,1:k-1})^{\mathrm{H}} a_{k,1:k-1}]^{1/2}$ (Compute diagonal entry of Cholesky factor G)
4: $\quad$ **for** $j = k+1$ to n **do**
5: $\quad\quad a_{jk} = a_{jk} - [(a_{j,1:k-1})^{\mathrm{H}} a_{k,1:k-1}]/a_{kk}$ (update j-th column of Cholesky factor G)
6: $\quad$ **end for**
7: **end for**

Computational complexity of Cholesky factorization:

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

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 $m < n$)

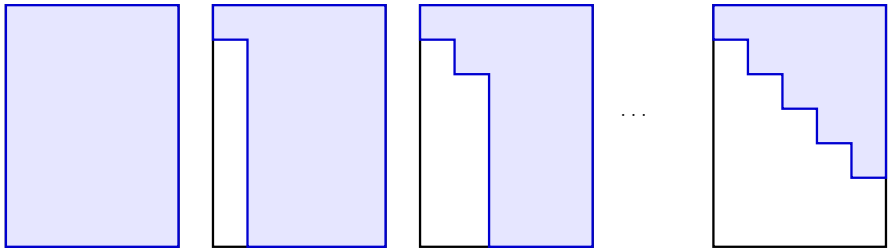
  Data analysis, interpretation of experiments with models...
- Solution then (i) may not exist, (ii) is multiple if it exists.
  Common approach: seek least-squares (LS) solutions:

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

- 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.

$$A = \begin{bmatrix} a_{11} & u_1 \\ \ell_1 & A_2 \end{bmatrix} \rightarrow FA = \begin{bmatrix} a'_{11} & u_1 \\ 0 & A'_2 \end{bmatrix} \dots \qquad (F^H F = I)$$

  Will prove very suitable for least squares.

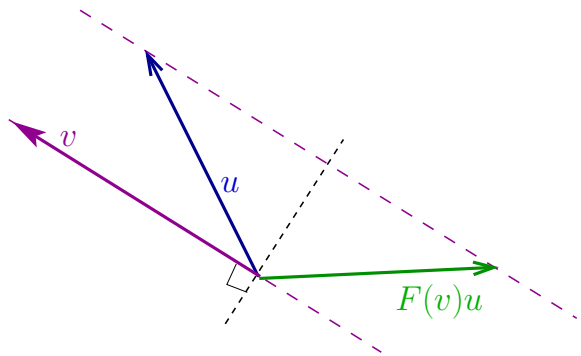
Householder reflections

- Householder reflection:

$$F(v) = I_m - 2\frac{v\,v^{\mathsf{H}}}{\|v\|^2} \qquad (v \in \mathbb{K}^m,\ F(v) \in \mathbb{K}^{m\times m}) \qquad F(v)^{\mathsf{H}}F(v) = I_m$$

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

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

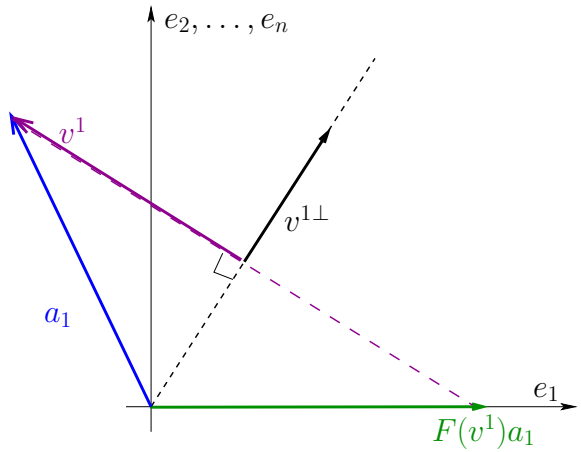


- First column of $A \in \mathbb{K}^{m\times n}$, $m \geq n$: choosing

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

we find

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



Householder reflections

- Continuing the process:

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

we reach

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

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

$$\boxed{A = QR, \qquad Q := F(v^1)^{\mathsf{H}}\,F(v^2)^{\mathsf{H}}\,\dots F(v^{n-1})^{\mathsf{H}}\,F(v^n)^{\mathsf{H}}},$$

$Q \in \mathbb{K}^{m\times m}$ unitary

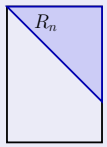
$R \in \mathbb{K}^{m\times n}$ upper triangular

QR factorization

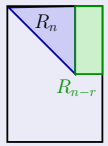
Let $A \in \mathbb{K}^{m\times n}$, $m \geq n$. There exist $Q \in \mathbb{K}^{m\times m}$ unitary, $R \in \mathbb{K}^{m\times n}$ upper triangular, and a permutation $\Pi \in \mathbb{R}^{n\times n}$ such that

$$A\Pi = QR.$$

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

(a) If $r = n$: $A = QR \quad , \quad R = \begin{bmatrix} R_n \\ 0 \end{bmatrix}$, 

$R_n \in \mathbb{K}^{n\times n}$ upper triangular invertible.

(b) If $r < n$: $A = QR\Pi^{\mathsf{T}}, \quad R = \begin{bmatrix} R_r & R_{n-r} \\ 0 & 0 \end{bmatrix}$, 

$R_r \in \mathbb{K}^{r\times r}$ upper triangular invertible,
$R_{n-r} \in \mathbb{K}^{r\times(n-r)}$.

Algorithm 3 QR factorization using Householder reflections

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

Backward stability of QR factorization

Let $A \in \mathbb{K}^{m\times n}$ with $m \geq n$. Let $\widetilde{Q}\widetilde{R}$ 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

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

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 $r \leq n$):

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

(a) If $r = n$:

$$\begin{bmatrix} R_n \\ 0 \end{bmatrix} \{x'\} = \begin{Bmatrix} (Q^{\mathsf{H}}b)_{1:n} \\ (Q^{\mathsf{H}}b)_{n+1:m} \end{Bmatrix}$$

If $(Q^{\mathsf{H}}b)_{n+1:m} = 0$ unique solution $\boxed{R_n x' = (Q^{\mathsf{H}}b)_{1:n}}$

If $(Q^{\mathsf{H}}b)_{n+1:m} \neq 0$ no solution

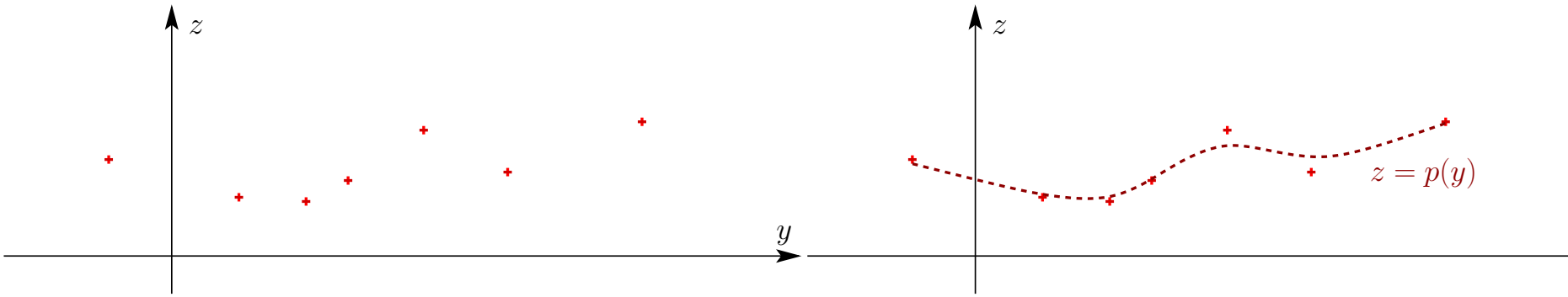
(b) If $r < n$:

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

If $(Q^{\mathsf{H}}b)_{r+1:m} = 0$ multiple solutions $\boxed{R_r x'_{1:r} = (Q^{\mathsf{H}}b)_{1:r} - R_{n-r}x'_{r+1:n}}$

If $(Q^{\mathsf{H}}b)_{r+1:m} \neq 0$ no solution

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

Matrix form:

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

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:

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

$\rightarrow$ LS solution(s) always exist (since we work in finite dimension!)

$\rightarrow$ If $\min_{x \in \mathbb{K}^n} \|Ax - b\|^2 = 0$, x solves $Ax = b$ (obviously).

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

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

- Set to zero the x-dependent part of the residual:
  (a) If $r = n$:

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

  (b) If $r < n$:

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

- Least squares residual:

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

  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 $A^{\mathsf{H}}A \in \mathbb{K}^{n\times n}$ and $AA^{\mathsf{H}} \in \mathbb{K}^{m\times m}$ are square Hermitian.
  Well-defined (symmetric positive) eigenvalue problems: $\boxed{A^{\mathsf{H}}Av = \lambda v}$ and $\boxed{AA^{\mathsf{H}}u = \mu v}$

  (λ, v) eigenpair of $A^{\mathsf{H}}A \implies (\lambda, Av)$ eigenpair of AA^{H}
  (μ, u) eigenpair of $AA^{\mathsf{H}} \implies (\mu, A^{\mathsf{H}}u)$ eigenpair of $A^{\mathsf{H}}A$

  (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 $A = USV^{\mathsf{H}}$, where:

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

- Available operators: `[U,S,V]=svd(A)` (MATLAB), `F=svdfact(A)` (Julia; `F` contains U, S, V)

Some properties

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

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

  (vectors $u_{r+1}, \ldots, u_m$ and $v_{r+1}, \ldots, v_n$ inactive, generate $\mathcal{N}(A)$ and $\mathcal{R}(A)^{\perp}$).
- SVD is rank-revealing: $\operatorname{rank}(A)$ 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:

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

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

Reduced SVD:

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

SVD applied to arbitrary linear systems

Solvability of $Ax = b$ (again):

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

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

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

  Then, $\boxed{\sigma_i(v_i^{\mathrm{H}}x) = u_i^{\mathrm{H}}b}$ $(1 \le i \le r)$ determine projections $v_i^{\mathrm{H}}x$ uniquely

  Remaining projections $v_i^{\mathrm{H}}x$ $(r+1 \le i \le m)$ arbitrary.
- General solution (if it exists):

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

  Setting $x_{r+1} = \ldots = x_n = 0$ gives minimum-norm solution
- Uniqueness condition: $\boxed{r = n}$ (implies $m \ge n$).

SVD applied to least-squares problems

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

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

Solutions:

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

Residual:

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

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

$$\begin{array}{llll} \text{(a)} & AA^+A = A, & \text{(b)} & (AA^+)^{\mathrm{H}} = AA^+, \\ \text{(c)} & A^+AA^+ = A^+, & \text{(d)} & (A^+A)^{\mathrm{H}} = A^+A. \end{array}$$

(Moore-Penrose conditions)

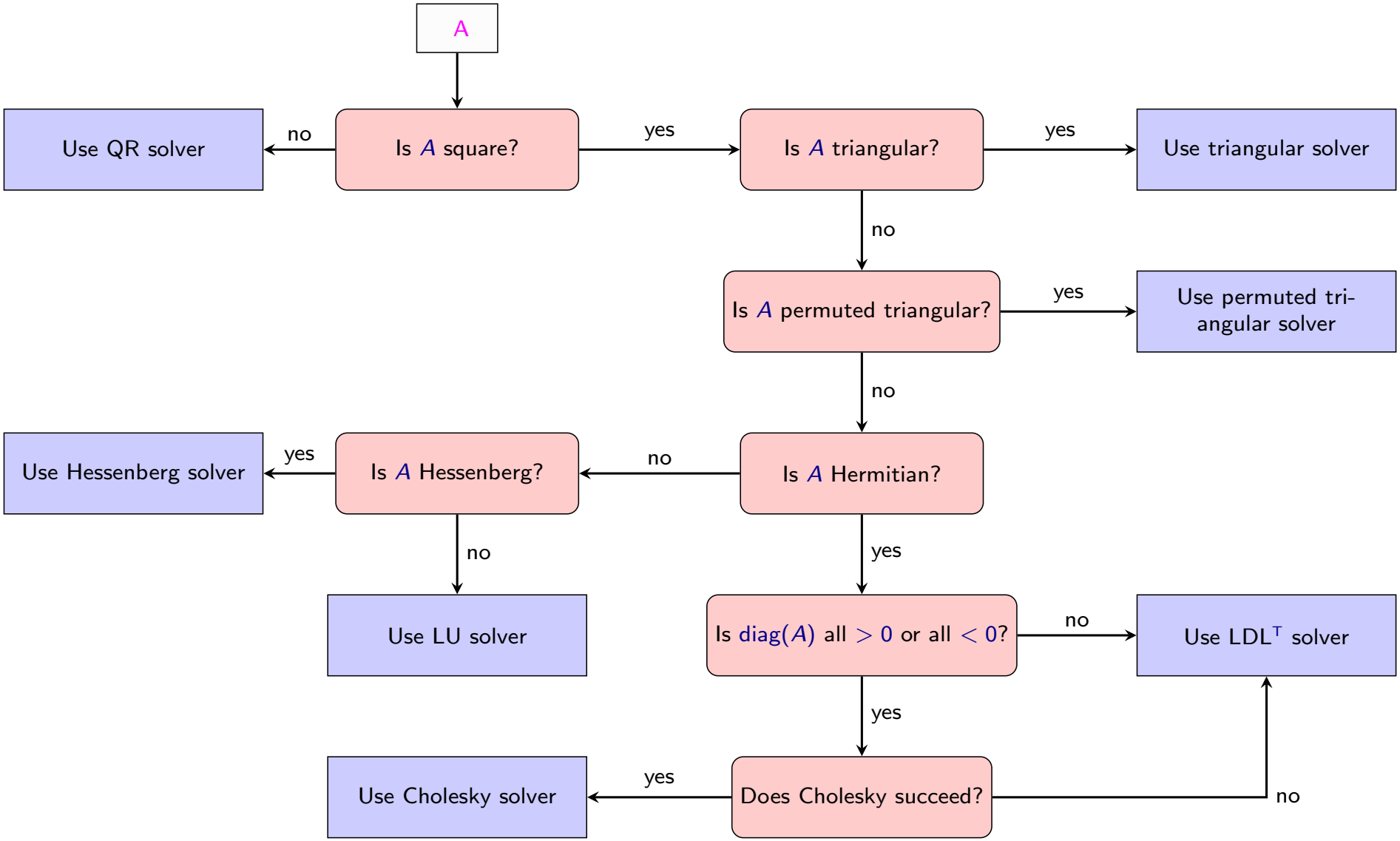
Algebraic properties of A^+

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

- Explicit formula using reduced SVD:
  $A^+ = V_r S_r^{-1} U_r^{\mathrm{H}}$
- General solution least-squares problem:
  $x = A^+b + (I - A^+A)w, \qquad w \in \mathbb{K}^n$ arbitrary
- A^+ does not depend continuously on A. Example ($\mathrm{rank}(A) = r$, $\mathrm{rank}(A_\varepsilon) = r+1$):

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

The backslash operator by way of summary



Flowchart of the backslash operator "\" of MATLAB