Lecture 1: Review of Linear Algebra, 1. (February 2)

1.1 Matrices and Vectors. (Ch. 1)

Most of this class is focused on the topic of *matrices* and manipulations thereof. In these first few lectures, we review the bases of linear algebra.

1.1.1 Notation.

- We will denote $\mathbb{R}$ the field of real numbers, and $\mathbb{C}$ the field of complex numbers. $\mathbb{F}$ stands in for either $\mathbb{R}$ or $\mathbb{C}$.
- $\mathbb{F}^{m\times n}$ is the ring of matrices with m columns and n rows.
- $\mathbb{F}^n \equiv \mathbb{F}^{n\times 1}$ is the vector space of (column) vectors with n components and dimension n.
- $\mathbb{F}^{1\times n}$ is the vector space of row vectors with n components, which also has dimension n.

Vector spaces Basic notions should be reviewed by reading paragraph 1.1 in the textbook: linear combinations, independence, basis.

1.1.2 Matrix notation.

Row vectors:

$$r = \begin{bmatrix} r_1 & \dots & r_n \end{bmatrix}, \qquad r_i \in \mathbb{F}.$$

Column vectors:

$$c = \begin{bmatrix} c_1 \\ \vdots \\ c_n \end{bmatrix}, \qquad c_i \in \mathbb{F}.$$

Rectangular matrix:

$$A = \begin{bmatrix} a_{11} & a_{12} & \dots & a_{1n} \\ a_{21} & a_{22} & \dots & a_{2n} \\ \vdots & \vdots & & \vdots \\ a_{m1} & a_{m2} & \dots & a_{mn} \end{bmatrix} = \big(a_{ij}\big)_{m\times n} = \big(a_{ij}\big)$$

The i-th row of the matrix A is the row vector of size n: $\begin{bmatrix} a_{i1} & \dots & a_{in} \end{bmatrix}$

The j-th column of the matrix A is the column vector of size m: $\begin{bmatrix} a_{1j} \\ \vdots \\ a_{mj} \end{bmatrix}$

When $m = n$ the matrix is called *square*:

$$A = \begin{bmatrix} a_{11} & \dots & a_{1n} \\ \vdots & & \vdots \\ a_{n1} & \dots & a_{nn} \end{bmatrix}$$

and its *main diagonal* is the vector $\big(a_{11} \quad \dots \quad a_{nn}\big)$.

1.1.3 A few special vectors and matrices.

- Fix $n \geqslant 0$, for each $j \in \{1, \ldots, n\}$ the vector

$$e_j = \left(\delta_{ij}\right)_n = \begin{bmatrix} 0 \\ \vdots \\ 0 \\ 1 \\ 0 \\ \vdots \\ 0 \end{bmatrix} \leftarrow j\text{-th row}$$

  is the j-th standard unit basis vector.

- $0_{m\times n}$ is the $m \times n$ zero matrix.

- 0_n is zero column vector of dimension n.

- I_n is the $n \times n$ identity matrix:

$$I_n = \begin{bmatrix} 1 & 0 & \ldots & 0 \\ 0 & 1 & \ddots & \vdots \\ \vdots & \ddots & \ddots & 0 \\ 0 & \ldots & 0 & 1 \end{bmatrix}$$

  Its $j-th$ column (or row) is the basis vector e_j.

- Trapezoidal or triangular matrices:

	$m > n$	$m = n$	$m < n$
$R \in \mathbb{F}^{m\times n}$ Upper triangular: $r_{ij} = 0$ for $i > j$	$\begin{bmatrix} r_{11} & \ldots & r_{1n} \\ & \ddots & \vdots \\ 0 & & r_{nn} \\ & 0 & \end{bmatrix}$	$\begin{bmatrix} r_{11} & \ldots & r_{1n} \\ & \ddots & \vdots \\ 0 & & r_{nn} \end{bmatrix}$	$\begin{bmatrix} r_{11} & \ldots & r_{1m} & \ldots & r_{1n} \\ & \ddots & \vdots & \vdots & \vdots \\ 0 & & r_{mm} & \ldots & r_{mn} \end{bmatrix}$
$D \in \mathbb{F}^{m\times n}$ Diagonal : $d_{ij} = 0$ for $i \neq j$	$\begin{bmatrix} d_{11} & & 0 \\ & \ddots & \\ 0 & & d_{nn} \\ & 0 & \end{bmatrix}$	$\begin{bmatrix} d_{11} & & 0 \\ & \ddots & \\ 0 & & d_{nn} \end{bmatrix}$	$\begin{bmatrix} d_{11} & & 0 & \\ & \ddots & & 0 \\ 0 & & d_{mm} & \end{bmatrix}$
$L \in \mathbb{F}^{m\times n}$ Lower triangular: $l_{ij} = 0$ for $i < j$	$\begin{bmatrix} l_{11} & & 0 \\ \vdots & \ddots & \\ l_{n1} & \ldots & l_{nn} \\ \vdots & \vdots & \vdots \\ l_{m1} & \ldots & l_{mn} \end{bmatrix}$	$\begin{bmatrix} l_{11} & & 0 \\ \vdots & \ddots & \\ l_{n1} & \ldots & l_{nn} \end{bmatrix}$	$\begin{bmatrix} l_{11} & & 0 & \\ \vdots & \ddots & & 0 \\ l_{m1} & \ldots & l_{mm} & \end{bmatrix}$

1.1.4 Block form

A matrix can often be subdivided in blocks, which may for example have special structure. In such cases, we may use the following *block format*:

$$A = \begin{array}{c} \begin{matrix} n_1 & \dots & n_q \end{matrix} \\ \begin{bmatrix} A_{11} & \dots & A_{1q} \\ A_{21} & \dots & A_{2q} \\ \vdots & & \vdots \\ A_{p1} & \dots & A_{pq} \end{bmatrix} \end{array} \begin{matrix} m_1 \\ m_2 \\ \vdots \\ m_p \end{matrix}$$

where each block $A_{ij} \in \mathbb{F}^{m_i \times n_j}$ is an m_i by n_j matrix, and A is an $\mathbb{F}^{m \times n}$ with $m = m_1 + \cdots + m_p$ and $n = n_1 + \cdots + n_q$.

Remark 1.1. *Annotating with the number of columns or rows per block may be omitted if the context makes clear what they should be.*

Examples. The big matrix:

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

can be decomposed into a Sudoku using a 3×3 block array of 3×3 blocks:

$$A = \left[\begin{array}{ccc|ccc|ccc} 4 & 1 & 2 & 3 & 8 & 9 & 5 & 7 & 6 \\ 8 & 7 & 5 & 6 & 2 & 1 & 3 & 4 & 9 \\ 9 & 6 & 3 & 4 & 7 & 5 & 2 & 1 & 8 \\ \hline 3 & 9 & 1 & 5 & 4 & 8 & 7 & 6 & 2 \\ 6 & 2 & 4 & 7 & 9 & 3 & 1 & 8 & 5 \\ 5 & 8 & 7 & 2 & 1 & 6 & 9 & 3 & 4 \\ \hline 7 & 3 & 6 & 8 & 5 & 2 & 4 & 9 & 1 \\ 1 & 5 & 8 & 9 & 3 & 4 & 6 & 2 & 7 \\ 2 & 4 & 9 & 1 & 6 & 7 & 8 & 5 & 3 \end{array}\right] = \big(A_{ij}\big)_{3\times 3}$$

where $A_{ij} \in \mathbb{R}^{3\times 3}$. Alternatively, we can decompose A into it column form:

$$A = \left[\begin{array}{c|c|c|c|c|c|c|c|c} 4 & 1 & 2 & 3 & 8 & 9 & 5 & 7 & 6 \\ 8 & 7 & 5 & 6 & 2 & 1 & 3 & 4 & 9 \\ 9 & 6 & 3 & 4 & 7 & 5 & 2 & 1 & 8 \\ 3 & 9 & 1 & 5 & 4 & 8 & 7 & 6 & 2 \\ 6 & 2 & 4 & 7 & 9 & 3 & 1 & 8 & 5 \\ 5 & 8 & 7 & 2 & 1 & 6 & 9 & 3 & 4 \\ 7 & 3 & 6 & 8 & 5 & 2 & 4 & 9 & 1 \\ 1 & 5 & 8 & 9 & 3 & 4 & 6 & 2 & 7 \\ 2 & 4 & 9 & 1 & 6 & 7 & 8 & 5 & 3 \end{array}\right] = \big(C_{ij}\big)_{1\times 9}$$

or its row form:

$$A = \left[\begin{array}{ccccccccc} 4 & 1 & 2 & 3 & 8 & 9 & 5 & 7 & 6 \\ \hline 8 & 7 & 5 & 6 & 2 & 1 & 3 & 4 & 9 \\ \hline 9 & 6 & 3 & 4 & 7 & 5 & 2 & 1 & 8 \\ \hline 3 & 9 & 1 & 5 & 4 & 8 & 7 & 6 & 2 \\ \hline 6 & 2 & 4 & 7 & 9 & 3 & 1 & 8 & 5 \\ \hline 5 & 8 & 7 & 2 & 1 & 6 & 9 & 3 & 4 \\ \hline 7 & 3 & 6 & 8 & 5 & 2 & 4 & 9 & 1 \\ \hline 1 & 5 & 8 & 9 & 3 & 4 & 6 & 2 & 7 \\ \hline 2 & 4 & 9 & 1 & 6 & 7 & 8 & 5 & 3 \end{array}\right] = \left(R_{ij}\right)_{9\times 1}.$$

1.1.5 Matrix operations

Let us fix three matrices

$$A = \left(a_{ij}\right)_{m\times n}, \qquad B = \left(b_{ij}\right)_{m\times n}, \qquad C = \left(c_{ij}\right)_{n\times p}$$

We recall now the following operations:

- Transpose: $A^T = \left(a_{ji}\right)_{n\times m}$,

  Remark 1.2. *If $x = \begin{pmatrix} x_1 & \dots & x_n \end{pmatrix}$ is a row vector, then $x^T = \begin{pmatrix} x_1 & \dots & x_n \end{pmatrix}^T$ is a column vector, and vice versa.*

- Conjugate: $\overline{A} = \left(\overline{a_{ij}}\right)_{m\times n}$,
- Adjoint: A^* or $A^H = \left(\overline{a_{ji}}\right)_{n\times m}$,
- Multiplication by a scalar:
$$\alpha A = \left(\alpha a_{ij}\right)_{m\times n},$$
- Addition of matrices of the same size:
$$A + B = \left(a_{ij} + b_{ij}\right),$$
- Linear combinations:
$$\alpha A + \beta B = \left(\alpha a_{ij} + \beta b_{ij}\right)_{m\times n},$$
- Matrix-matrix multiplication:
$$AC = \left(\sum_{i=1}^{n} a_{ik}c_{kj}\right)_{m\times p}$$

Remark 1.3. *The matrix product is only defined when A has as many columns as C has rows!*

Remark 1.4. *The matrix product is non-commutative, such that in general $AC \neq CA$ even if both products make sense.*

Property 1.5. *If $A \in \mathbb{F}^{m\times n}$ and $C \in \mathbb{F}^{n\times p}$, then*

$$(AC)^T = C^T A^T \qquad \text{and} \qquad (AC)^* = C^* A^*.$$

Examples Take $A = \begin{bmatrix} 1 & 1 & 1 \end{bmatrix}$ and

$$B = \begin{bmatrix} 1 & 0 & 1 \\ 0 & 1 & 0 \\ 0 & 0 & 1 \end{bmatrix}$$

We compute then

$$AB = \begin{bmatrix} 1 & 1 & 2 \end{bmatrix},$$

while BA is undefined.
Next, we take

$$A = \begin{bmatrix} 0 & 1 \\ 0 & 0 \end{bmatrix}, \qquad B = \begin{bmatrix} 0 & 0 \\ 1 & 0 \end{bmatrix}.$$

Then we have

$$AB = \begin{bmatrix} 1 & 0 \\ 0 & 0 \end{bmatrix}, \qquad BA = \begin{bmatrix} 0 & 0 \\ 0 & 1 \end{bmatrix}.$$

This shows in particular that $AB \neq BA$.

1.1.6 Block operations

When the matrices have a (compatible) block structure, the operations above can be written using that block structure. In particular, assume we have

$$A = \begin{array}{c} \begin{matrix} n_1 & \dots & n_q \end{matrix} \\ \begin{bmatrix} A_{11} & \dots & A_{1q} \\ \vdots & & \vdots \\ A_{p1} & \dots & A_{pq} \end{bmatrix} \end{array} \begin{matrix} m_1 \\ \vdots \\ m_p \end{matrix} \qquad \text{and} \qquad B = \begin{array}{c} \begin{matrix} n_1 & \dots & n_q \end{matrix} \\ \begin{bmatrix} B_{11} & \dots & B_{1q} \\ \vdots & & \vdots \\ B_{p1} & \dots & B_{pq} \end{bmatrix} \end{array} \begin{matrix} m_1 \\ \vdots \\ m_p \end{matrix}.$$

Then we can write:

- the transpose and adjoint:

$$A^T = \begin{array}{c} \begin{matrix} m_1 & \dots & m_p \end{matrix} \\ \begin{bmatrix} A_{11}^T & \dots & A_{p1}^T \\ \vdots & & \vdots \\ A_{1q}^T & \dots & A_{pq}^T \end{bmatrix} \end{array} \begin{matrix} n_1 \\ \vdots \\ n_q \end{matrix}, \qquad A^* = \begin{array}{c} \begin{matrix} m_1 & \dots & m_p \end{matrix} \\ \begin{bmatrix} A_{11}^* & \dots & A_{p1}^* \\ \vdots & & \vdots \\ A_{1q}^* & \dots & A_{pq}^* \end{bmatrix} \end{array} \begin{matrix} n_1 \\ \vdots \\ n_q \end{matrix}$$

- Linear combinations:

$$\alpha A + \beta B = \begin{array}{c} \begin{matrix} n_1 & \dots & n_q \end{matrix} \\ \begin{bmatrix} \alpha A_{11} + \beta B_{11} & \dots & \alpha A_{1q} + \beta B_{1q} \\ \vdots & & \vdots \\ \alpha A_{p1} + \beta B_{p1} & \dots & \alpha A_{pq} + \beta B_{pq} \end{bmatrix} \end{array} \begin{matrix} m_1 \\ \vdots \\ m_p \end{matrix}$$

Next, if we also have a matrix C with block structure:

$$C = \begin{array}{c} \begin{matrix} k_1 & \dots & k_s \end{matrix} \\ \begin{bmatrix} C_{11} & \dots & C_{1s} \\ \vdots & & \vdots \\ C_{q1} & \dots & C_{qs} \end{bmatrix} \end{array} \begin{matrix} n_1 \\ \vdots \\ n_q \end{matrix}$$

then the matrix-vector product takes the block form

$$AC = \begin{matrix} \\ m_1 \\ \vdots \\ m_p \end{matrix} \begin{matrix} \begin{matrix} n_1 & \dots & n_q \end{matrix} \\ \begin{bmatrix} \sum_{j=1}^{q} A_{1j}C_{j1} & \dots & \sum_{j=1}^{q} A_{1j}C_{js} \\ \vdots & & \vdots \\ \sum_{j=1}^{q} A_{pj}C_{j1} & \dots & \sum_{j=1}^{q} A_{pj}C_{js} \end{bmatrix} \end{matrix}$$

Lecture 2: Review of Linear Algebra, 2. (February 4)

2.1 Matrix-vector product

When computing the matrix-vector product Ax where $A \in \mathbb{F}^{m\times n}$ and $x \in \mathbb{F}^n$, we may consider it under different viewpoints.

- First, recall that A can be identified with a linear map $A : \mathbb{F}^n \to \mathbb{F}^m$, $x \mapsto b = Ax \in \mathbb{F}^m$.
- Next, if we use the block row form of A:

$$A = \begin{bmatrix} r_1^T \\ \vdots \\ r_m^T, \end{bmatrix} \qquad r_i \in \mathbb{F}^n,$$

  then we obtain the row-oriented or inner-product version of the matrix-vector product:

$$b = Ax = \begin{bmatrix} r_1^T \\ \vdots \\ r_m^T, \end{bmatrix} x = \begin{bmatrix} r_1^T x \\ \vdots \\ r_m^T x, \end{bmatrix} \in \mathbb{F}^m,$$

  whereby the entries of the result are the obtained (in the real case) as the Euclidean inner/scalar product between the corresponding row of A and the vector x.

- Third, if we use the block column form of A:

$$A = \begin{bmatrix} c_1 & \dots & c_n, \end{bmatrix} \qquad c_j \in \mathbb{F}^m,$$

  then we obtain the column-oriented or linear combination version of the matrix-vector product:

$$b = Ax = \begin{bmatrix} c_1 & \dots & c_n, \end{bmatrix} \begin{bmatrix} x_1 \\ \vdots \\ x_n \end{bmatrix} = \sum_{j=1}^{n} x_j c_j.$$

  Here the result is framed as a linear combination of the columns of the matrix A with the entries of x as coefficients.

  We usually focus on the column-oriented version in algorithm, as it proves more convenient.

2.2 Range, nullspace and rank

Let $A \in \mathbb{F}^{m\times n}$ be a rectangular matrix. The action of A as a linear map $\mathbb{F}^n \to \mathbb{F}^m$ motivate the following definitions:

- Range / Column space of A:

$$\operatorname{Ran}(A) = \{Ax, \quad x \in \mathbb{F}^n\} \subseteq \mathbb{F}^m.$$

  If we write the block column form of $A = \begin{bmatrix} c_1 & \dots c_n \end{bmatrix}$ where $c_j \in \mathbb{F}^m$, then we may also identify

$$\operatorname{Ran}(A) = \operatorname{Span}\{c_1, \dots, c_n\}.$$

- Range of A^T / Row space of A:

$$\begin{aligned} \operatorname{Ran}\left(A^T\right) &= \{A^T x, \quad x \in \mathbb{F}^m\} \subseteq \mathbb{F}^n \\ &\equiv \{(xA)^T, \quad x \in \mathbb{F}^{1\times m}\} \subseteq \mathbb{F}^{1\times n} \end{aligned}$$

  This subspace of $\mathbb{F}^n$ is spanned by the rows of A.

- Kernel or nullspace of A:

$$\operatorname{Ker}(A) = \operatorname{null}(A) = \{x \in \mathbb{F}^n, \quad Ax = 0\} \subseteq \mathbb{F}^n.$$

- Rank of A:

$$\operatorname{rank}(A) = \dim \operatorname{Ran}(A)$$

The following results are fundamental to linear algebra.

Proposition 2.1. *Let $A \in \mathbb{F}^{m\times n}$ be a matrix, then*

$$\operatorname{rank}(A) + \dim \operatorname{Ker}(A) = n.$$

Proposition 2.2. *Let $A \in \mathbb{F}^{m\times n}$ be a matrix, then*

$$\operatorname{rank}(A) = \operatorname{rank}\left(A^T\right) = \operatorname{rank}\left(A^*\right).$$

Definition 2.3. *A matrix $A \in \mathbb{F}^{m\times n}$ has full rank if* $\operatorname{rank}(A) = \min\{m, n\}$.
It is rank-deficient if $\operatorname{rank}(A) < \min\{m, n\}$.

Theorem 2.4. *Let $A \in \mathbb{F}^{m\times n}$ with $m \geqslant n$. Then the following statements are equivalent:*

1. *A has full rank:* $\operatorname{rank}(A) = n$,
2. *A is one-to-one as a linear map $\mathbb{F}^n \to \mathbb{F}^m$, i.e.*

$$Au = Av \quad \Leftrightarrow \quad u = v,$$

3. $\operatorname{Ker}(A) = \{0\}$,
4. *the columns of A are linearly independent and form a basis for* $\operatorname{Ran}(A)$.

Proposition 2.5. *A matrix $A \in \mathbb{F}m \times n$ had rank 1 if and only if there exists nonzero vectors $u \in \mathbb{F}^m$, $v \in \mathbb{F}^n$ such that*

$$A = uv^*.$$

2.3 Trace and determinant

Definition 2.6. *For a square matrix $A \in \mathbb{F}^{n\times n}$, we define its trace*

$$\mathbb{F} \ni \operatorname{Tr} A = \sum_{i=1}^{n} a_{ii},$$

and its determinant

$$\mathbb{F} \ni \det A = \sum_{\pi\in P} sgn(\pi) a_{1\pi_1} \dots a_{n\pi_n},$$

where P is the set of $n!$ permutations of $\{1, \dots, n\}$.

Property 2.7. *Let $A \in \mathbb{F}^{n\times m}$, $B \in \mathbb{F}^{m\times n}$, then*

$$\operatorname{Tr} AB = \operatorname{Tr} BA.$$

Let $A, B \in \mathbb{F}^{n\times n}$, then

$$\det A = \det A^T \qquad \det AB = \det A \det B.$$

2.4 Matrix inverse.

Definition 2.8. *A square matrix $A \in \mathbb{F}^{n\times n}$ is said to be **invertible** or **non-singular** if there exists $B \in \mathbb{F}^{n\times n}$ such that*

$$AB = BA = I_n.$$

Such a matrix is unique and we denote then $A^{-1} = B$.
If there exists no such matrix B then we say that A is singular.

Property 2.9.

- *A is invertible if and only if A has full rank or* $\det(A) \neq 0$.
- *If A is invertible, then so are A^T and A^* and*

$$(A^{-1})^T = (A^T)^{-1}, \qquad (A^{-1})^* = (A^*)^{-1}.$$

- *If A and B are invertible then AB is invertible and*

$$(AB)^{-1} = B^{-1}A^{-1}.$$

Practical remarks

- There exists an explicit formula for computing the elements of A^{-1} in terms of the determinant of A and its minors, but it is very impractical in practice as its cost is $O(n!)$ floating-point operations.
- Never compute A^{-1} explicitely if it is not necessary, as even the best algorithms have a cost scaling as $O(n^3)$.

2.5 Special matrices

Definition 2.10. *A square matrix $A \in \mathbb{F}^{n\times n}$ is said to be...*

- *Symmetric if $A = A^T$,*
- *Orthogonal if $A^{-1} = A^T$,*
- *Hermitian or self-adjoint if $A = A^*$,*
- *Unitary if $A^{-1} = A^*$,*
- *Normal if $AA^* = A^*A$.*

2.6 Scalar (or inner) product in finite dimension

Definition 2.11. *A scalar product on a vector space V over $\mathbb{F} = \mathbb{R}$ or $\mathbb{C}$ is a map $(\cdot,\cdot)$: $V \times V \to \mathbb{F}$ such that:*

- *$(\cdot,\cdot)$ is linear in its second variable:*

$$(x, \alpha y_1 + \beta y_2) = \alpha(x, y_1) + \beta(y, y_2),$$

- *$(\cdot,\cdot)$ is hermitian:*

$$(x, y) = \overline{(y, x)},$$

- *$(\cdot,\cdot)$ is positive definite:*

$$(x, x) \geqslant 0 \qquad \text{and} \qquad (x, x) = 0 \implies x = 0.$$

Example: *the standard inner product on $\mathbb{F}^n$ is*

$$(x, y) = x^* y = \sum_{i=1}^{n} \overline{x_i} y_i.$$

Remark 2.12. *We adopt here the "physics" convention where, in the complex case, the scalar product is linear in its second variable and anti-linear in its first variable. Often, the mathematical literature adopts the different convention (linear in the first variable, anti-linear in the second variable), leading to a different definition of the standard inner product, $(x, y) = y^* x$.*

Here are a couple properties of the standard Euclidean inner product:

Property 2.13.

- *Given a matrix $A \in \mathbf{F}^{n\times n}$,*

$$(x, Ay) = (A^* x, y), \qquad \forall x, y \in \mathbb{F}^n.$$

- *Unitary matrices conserve the scalar product, in the sense that*

$$(Qx, Qy) = (x, y), \qquad \forall x, y \in \mathbb{F}^n.$$

The structure associated with scalar products is extremely useful. The concept of *orthogonality* in particular has a wide range of uses.

Definition 2.14.

- *Two vectors $x, y \in \mathbb{F}^n$ are orthogonal if*

$$(x, y) = 0.$$

- *Two sets of vectors $X, Y \subset \mathbb{F}^n$ are orthogonal if*

$$(x, y) = 0, \qquad \forall x \in X,\ y \in Y.$$

- *A set of vectors $\{\mathbf{v}_1, \ldots, \mathbf{v}_m\}$ is orthogonal if*

$$(v_i, v_j) = 0, \qquad \forall i \neq j \in \{1, \ldots, m\}.$$

It is orthonormal if, in addition,

$$(v_i, v_i) = 1, \qquad \forall i \in \{1, \ldots, m\}.$$

Proposition 2.15. *If $\{v_1, \ldots, v_m\} \subset \mathbb{F}^n$ is orthogonal and the v_i are all nonzero, then $\{v_1, v_m\}$ is linearly independent and $m \leqslant n$.*

Proof. Let $\{v_1, \ldots, v_m\}$ an orthogonal subset of $\mathbb{F}^n \backslash \{0\}$ as in the theorem. Suppose a linear combination $\sum_{j=1}^m c_j v_j$ vanishes, then for all $i = 1, \ldots, m$ we have

$$\left(v_i,\ \sum_{j=1}^m c_j v_j \right) = \sum_{j=1}^m c_j (v_i, v_j) = c_i (v_i, v_i) = 0.$$

Now since $v_i \neq 0$, we have $(v_i, v_i) \neq 0$ and hence $c_i = 0$. By definition, this shows that the set $\{v_1, \ldots, v_m\}$ is linearly independent. Thus m must be lesser or equal than the dimension of the vector space, which is n. □

Corollary 2.16. *If $\{v_1, \ldots, v_n\}$ is as in Proposition 2.15 with $m = n$, then it is a basis for $\mathbb{F}^n$.*

Lecture 3: Gram-Schmidt Process; Vector Norms. (February 9)

3.1 Orthogonal Projection.

Given a set of orthonormal vectors $\{q_1, \ldots, q_m\} \in \mathbb{F}^n$ spanning a subspace $S \subset \mathbb{F}^n$, we define the linear map

$$\mathbb{F}^n \ni v \quad \mapsto \quad Pv = \sum_{i=1}^{m} (q_i, v) q_i \qquad\qquad = \sum_{i=1}^{m} q_i (q_i^* v) \in S.$$

Note that P corresponds to the matrix, also denoted P,

$$P = \sum_{i=1}^{m} q_i q_i^*.$$

Then for any $q \in S$, we have the property

$$(q, v) = (q, Pv) \qquad \text{or} \qquad (q, v - Pv) = 0,$$

meaning that the difference $v - Pv$ is orthogonal to the subspace S.

Definition 3.1. *The matrix $P = \sum_{i=1}^{m} q_i q_i^*$ is called the **orthogonal projector** onto S.*

This matrix has the classical property of projectors, $P^2 = P$, and also it is Hermitian, $P = P^*$.

3.2 Gram-Schmidt Process.

Let $\{v_1, \ldots, v_m\} \subset \mathbb{F}^m$ be a set of independent vectors, $m \leqslant n$. The following procedure yields an orthonormal set of vectors $q_1, \ldots, q_m$ spanning the same subspace of $\mathbb{F}^n$:

1. Set
$$w_1 = v_1 \qquad\qquad \text{then} \qquad\qquad q_1 = \frac{1}{\sqrt{(w_1, w_1)}} w_1,$$
such that $(q_1, q_1) = 1$.

2. Next, set
$$w_2 = v_2 - (q_1, v_2) q_1 \qquad\qquad \text{and} \qquad\qquad q_2 = \frac{1}{\sqrt{(w_2, w_2)}} w_2,$$
such that $(q_1, w_2) = (q_1, v_2) - (q_1, v_2)(q_1, q_1) = 0$ and hence $(q_2, q_1) = 0$, $(q_2, q_2) = 1$.

...

Step j. Continue the process with

$$w_j = v_j - \underbrace{\left(\sum_{i=1}^{j-1} q_i q_i^* \right)}_{\text{orthogonal projector onto } \mathrm{Span}(q_1, \ldots, q_{j-1})} v_j \qquad\qquad \text{and} \qquad\qquad q_j = \frac{1}{\sqrt{(w_j, w_j)}} w_j,$$

such that $(q_i, w_j) = (q_i, v_j) - (q_i, P_{j-1} v_j) = 0$ for $i < j$ where P_{j-1} is the orthogonal projector onto the subspace spanned by $q_1, \ldots, q_{j-1}$, and hence the vectors $q_1, \ldots, q_j$ form an orthonormal family.

3.3 Vector Norms.

We define now a central object for the study of vectors, and in particular for the introduction of a topology and notions such as limits and continuity.

Definition 3.2. *Let V be a vector space over $\mathbb{F}$. A map $\|\cdot\| : V \to \mathbb{R}_+$ is a **norm** on V if*

1. *$\|\mathbf{v}\| = 0$ implies $v = 0$,*
2. *$\|\alpha v\| = |\alpha|\|v\|$ for any scalar $\alpha \in \mathbb{F}$ and vector $v \in V$ (homogeneity),*
3. *$\|u + v\| \leqslant \|u\| + \|v\|$ for any two vectors $u, v \in V$ (triangular inequality).*

Examples. for $x \in \mathbb{F}^n$, we define the classical definitions:

- The Euclidean norm:
$$\|x\|_2 = \sqrt{(x, x)} = (x^* x)^{1/2} = \left(\sum_{i=1}^{n} |x_j|^2 \right)^{1/2}$$
- The Hölder p-norm: for $p \geqslant 1$, we set
$$\|x\|_p = \left(\sum_{i=1}^{n} |x_j|^p \right)^{1/p}$$
(Note that the Euclidean norm is a special case obtained for $p = 2$.)
- The maximum or infinity norm:
$$\|x\|_\infty = \max_{1 \leqslant i \leqslant n} |x_i|.$$
- Given a full-rank matrix $A \in \mathbb{F}^{m \times n}$ with $m \geqslant n$ and $p \in [1, \infty]$, we define
$$\|x\|_{A,p} = \|Ax\|_p.$$

One can check that each of these examples satisfies properties 1-3 above, and define proper *norms* on $V = \mathbb{F}^n$. Norms bear a strong relations to scalar products, in particular the Euclidean or 2-norm. For example, we have the following classical result:

Proposition 3.3. *Let x, y be two vectors in $\mathbb{F}^n$ equipped with the scalar product $(x, y) = x^* y$.*

- *Cauchy-Schwartz inequality:*
$$|(x, y)| \leqslant \|x\|_2 \|y\|_2.$$
- *The above is a special case of the Hölder inequality:*
$$|(x, y)| \leqslant \|x\|_p \|y\|_q, \qquad \text{where } p, q \in [1, \infty] \text{ and } \frac{1}{p} + \frac{1}{q} = 1.$$

Since there are many possible norms, it is important to understand to which extent these norms are fundamentally different - in particular, are they comparable? In finite dimension, the following definition and theorem answer this question.

Definition 3.4 (Equivalence of norms)**.** *Two norms $\|\cdot\|$ and $|||\cdot|||$ on V are* ***equivalent*** *if there exists two constants $c, C > 0$ such that*

$$c|||x||| \leqslant \|x\| \leqslant C|||x|||, \qquad \text{for any } v \in V.$$

Theorem 3.5. *Let V be a finite-dimensional vector space. Then all norms on V are equivalent.*

Corollary 3.6. *Let $\|\cdot\|$ be any norm on $\mathbb{F}^n$ and a sequence $(x^{(k)})_{k\geqslant 0}$ of elements of $\mathbb{F}^n$. Then we say that $(x^{(k)})$ converges to $x \in \mathbb{F}^n$, or*

$$\lim_{k\to\infty} x^{(k)} = x,$$

if and only if $\lim_{k\to\infty} x_i^{(k)}$ as number sequences for all $i = 1, \ldots, n$, and this is equivalent to

$$\lim_{k\to\infty} \|x^{(k)} - x\| = 0.$$

3.4 Matrix Norms

Similarly to vector spaces, spaces of matrices can be equipped with norms, defined in the same way.

Definition 3.7. *A function $\|\cdot\|\colon \mathbb{F}^{m\times n} \to \mathbb{R}_+$ is a norm if*

1. *$\|A\| \geqslant 0$, and $\|A\| = 0$ iff $A = 0$, for all $A \in \mathbb{F}^{m\times n}$.*
2. *$\|\alpha A\| = |\alpha|\|A\|$, , for all $A \in \mathbb{F}^{m\times n}$ and $\alpha \in \mathbb{F}$.*
3. *$\|A + B\| \leqslant \|A\| + \|B\|$, for all $A, B \in \mathbb{F}^{m\times n}$.*

Because matrices can be multiplied, the following notion is useful.

Definition 3.8. *We say that a matrix norm $\|\cdot\|$ is* ***sub-multiplicative*** *if*

$$\|AB\| \leqslant \|A\|\|B\|, \qquad \forall A \in \mathbb{F}^{m\times n} \text{ and } B \in \mathbb{F}^{n\times p}.$$

In some cases, matrix norms may be related to norms on the vectors spaces on which they act as linear transformations: $A : \mathbb{F}^n \to \mathbb{F}^m$.

Definition 3.9. *A matrix norm $\|\cdot\|$ on $\mathbb{F}^{m\times n}$ is called compatible or consistent with vector norms also denoted $\|\cdot\|$ on $\mathbb{F}^m$ and $\mathbb{F}^n$ if*

$$\|Ax\|_{\mathbb{F}^m} \leqslant \|A\|\|x\|_{\mathbb{F}^n}.$$

Note that not all matrix norms are sub-multiplicative or consistent. For example, the function $\|A\|_\Delta = \max_{i,j} \|a_{ij}\|$ is a norm on the space of 2×2 matrices, yet

$$A = B = \begin{bmatrix} 1 & 1 \\ 1 & 1 \end{bmatrix} \text{ have norm } \|A\|_\Delta \|B\|_\Delta = 1,$$

but

$$AB = \begin{bmatrix} 2 & 2 \\ 2 & 2 \end{bmatrix} \text{ has norm } \|AB\|_\Delta = 2,$$

so $\|\cdot\|_\Delta$ is not submultiplicative.
On the other hand, define the Frobenius norm:

$$\|A\|_F = \left(\sum_{i,j=1}^{n} |a_{i,j}|^2 \right)^{1/2} = \sqrt{\operatorname{Tr} A^* A}.$$

Property 3.10. *The Frobenius norm is sub-multiplicative, compatible with the Euclidean norm $\|\cdot\|_2$, and*

$$\|A\|_F = \|A^T\|_F = \|A^*\|_F.$$

Lecture 4: Induced Norms, Eigen-problems. (February 11)

4.1 Induced Norms

Definition 4.1. *Let $\|\cdot\|$ be a vector norm. The function:*

$$\|A\| = \sup_{x\neq 0} \frac{\|Ax\|}{\|x\|} = \sup_{\|x\|=1} \|Ax\| \tag{4.1}$$

is a matrix norm, called the induced norm or natural matrix norm associated with the vector norm $\|\cdot\|$.

Property 4.2. *If $\|A\|$ is induced by the vector norm $\|x\|$, then*

- $\|Ax\| \leqslant \|A\|\|x\|$ *for any matrix $A \in \mathbb{F}^{m\times n}$ and vector $x \in \mathbb{F}^n$,*
- $\|AB\| \leqslant \|A\|\|B\|$ *for any matrices $A \in \mathbb{F}^{m\times n}$, $B \in \mathbb{F}^{n\times p}$*
  (all induced norms are sub-multiplicative),
- $\|I_n\| = 1$.

Before proving that (4.1) indeed defines a *norm* on matrices, let us propose and compute a few such induced norms. Given $1 \leqslant p \leqslant \infty$, the Hölder p-norm on vectors defines an induced matrix norm by

$$\|A\|_p = \sup_{\|x\|=1} \|Ax\|_p.$$

- For $p = 1$, we can compute explicitely this quantity. Let $A \in \mathbb{F}^{m\times n}$ with columns $c_1, \ldots, c_n \in \mathbb{F}^m$, for any vector $x \in \mathbb{F}^n$,

$$\|Ax\|_1 = \left\| \sum_{j=1}^n x_j c_j \right\|_1 \leqslant \sum_{j=1}^n |x_j| \|c_j\|_1 \leqslant (\max_{1\leqslant j\leqslant n} \|c_j\|_1) \sum_{j=1}^n |x_j|,$$

  where we have used the triangular inequality for the 1-norm. Since the last quantity on the right is $\|x\|_1$, we have that for any vector x,

$$\frac{\|Ax\|_1}{\|x\|_1} \leqslant \max_{j=1,\ldots,n} \|c_j\|_1 = \max_{j=1,\ldots,n} \left(\sum_{i=1}^n |a_{ij}| \right).$$

  Taking the maximum over the left-hand side of the inequality shows that

$$\|A\|_1 \leqslant \max_{j=1,\ldots,n} \left(\sum_{i=1}^n |a_{ij}| \right).$$

  Taking $x = e_j$ where j is chosen such that $\|c_j\|_1 = \max_{k=1,\ldots,n} \|c_k\|_1$ yields $\|x\|_1 = 1$, $\|Ax\|_1 = \|c_j\|_1 = \max_{j=1,\ldots,n} (\sum_{i=1}^n |a_{ij}|)$, which shows that in fact this inequality is an equality:

$$\|A\|_1 = \max_{j=1,\ldots,n} \left(\sum_{i=1}^n |a_{ij}| \right).$$

  The induced 1-norm on matrices is dubbed the "column sum norm".

- For $p = \infty$, one computes similarly,

$$\|A\|_\infty = \max_{i=1,\dots,n} \left(\sum_{j=1}^{n} |a_{ij}| \right).$$

The induced 1-norm on matrices is dubbed the "row sum norm".

The expressions above lead to the following properties of the induced 1- and ∞-norms, since transposition exchanges the role of rows and columns:

Proposition 4.3. *Let $A \in \mathbb{F}^{m\times n}$, then*

- $\|A\|_1 = \|A^T\|_\infty = \|A^*\|_\infty$
- $\|A\|_\infty = \|A^T\|_1 = \|A^*\|_1$

- Finally, the induced 2-norm or spectral norm does not have a simple expression in terms of the entries of A, and is rather more difficult to compute. Thanks to the sub-multiplicativity of the Frobenius norm, we have however

$$\|A\|_2 = \sum_{x\neq 0} \frac{\|Ax\|_2}{\|x\|_2} \leqslant \|A\|_F.$$

Let us now prove that these objects are indeed norms on matrices!

Proof. Let $\|\cdot\|$ be a norm on $\mathbb{F}^m$ and $\mathbb{F}^n$, and define the quantity

$$\|A\| = \sup_{x\neq 0} \frac{\|Ax\|}{\|x\|}.$$

- Clearly, $\|A\|$ is a positive quantity. If $A \neq 0$, there exists at least one vector $x \notin \mathrm{Ker}\,(A)$, hence such that $\|Ax\| > 0$. As a result,

$$\|A\| \geqslant \frac{\|Ax\|}{\|x\|} > 0.$$

- Next, let A, B two matrices and take $x \in \mathbb{F}^n$ with norm 1, then using the properties of vector norms,

$$\|(A+B)x\| = \|Ax + Bx\| \leqslant \|Ax\| + \|Bx\| \leqslant \|A\| + \|B\|.$$

Taking the supremum on the left-hand side, we have thus the triangle inequality,

$$\|A + B\| \leqslant \|A\| + \|B\|.$$

- Finally, for $\alpha \in \mathbb{F}$, $\|\alpha Ax\| = |\alpha|\|Ax\|$, so by taking the maximum on both sides, we have clearly homogeneity: $\|\alpha A\| = |\alpha|\|A\|$.

Hence the induced norm $\|A\|$ is indeed a norm on matrices.

□

4.2 Eigenvalues and Eigenvectors.

Definition 4.4. *Given a square matrix $A \in \mathbb{F}^{n\times n}$, we say that $\lambda \in \mathbb{C}$ is an **eigenvalue** of A if there exists a vector $v \in \mathbb{F}^n \backslash \{0\}$ such that*

$$Ax = \lambda x.$$

*The vector x is called a (right) **eigenvector** of A, and the set of all eigenvalues is called the spectrum of A and denoted $\sigma(A)$.*
*A **left eigenvector** of A is a vector y satisfying*

$$y^*A = \lambda y^*.$$

Definition 4.5. *We further define ...*

- ***the Rayleigh quotient:*** $\dfrac{x^*Ax}{x^*x}$, *which equals an eigenvalue λ if x is the corresponding eigenvector,*
- ***the spectral radius:*** $\rho(A) = \max_{\lambda\in\sigma(A)} |\lambda|$,
- ***the characteristic polynomial:*** $p_A(\lambda) = \det A - \lambda I$, *which is a polynomial of degree n. Note that the spectrum $\sigma(A)$ is the set of roots of $p_A(\lambda)$.*

Property 4.6.
- *The spectrum of A coincides with the spectrum of its transpose: $\sigma(A) = \sigma(A^T)$.*
- *The spectrum of A coincides with the complex conjugate of the spectrum of its adjoint: $\sigma(A^*) = \sigma(\overline{A}) = \overline{\sigma(A)}$.*

Definition 4.7. *For a given eigenvalue $\lambda \in \sigma(A)$, we define*

- *the **algebraic multiplicity**, which is the mupltiplicity of λ as a root of the characteristic polynomial p_A,*
- *the **geometric multiplicity**, which is the dimension of* $\operatorname{Ker}(A - \lambda I_n)$.

*If the two are different, we say that the eigenvalue (and the matrix) is **defective**.*

Note that the geometric multiplicity is always smaller or equal than the algebraic multiplicity.

Example: the matrix $A = \begin{bmatrix} 0 & 1 \\ 0 & 0 \end{bmatrix}$, such that $p_A(\lambda) = \lambda^2$, has a unique eigenvalue $\lambda = 0$ with algebraic multiplicity 2 and geometric multiplicity 1.

4.3 Similarities

Definition 4.8. *Two matrices $A, B \in \mathbb{F}^{n\times n}$ are called **similar** if there exists an invertible matrix C such that*

$$B = C^{-1}AC.$$

*They are **unitarily similar** if the matrix C is unitary.*

Property 4.9. *If two matrices A, B are similar, their characteristic polynomials and spectra are identical:*

$$p_A = p_B, \qquad \sigma(A) = \sigma(B).$$

Proof.

$$p_B = \det C^{-1}AC - \lambda C^{-1}C = \det C^{-1}(A - \lambda I_n)C = \det C \det C^{-1} \det A - \lambda I = p_A.$$

□

A very important result in the theory of square matrices is that any matrix is unitarily similar to a triangular matrix.

Theorem 4.10 (Schur decomposition)**.** *Given $A \in \mathbb{F}^{n\times n}$, there exists a unitary matrix U such that*

$$U^{-1}AU = U^*AU = \begin{bmatrix} \lambda_1 & b_{12} & \dots & b_{1n} \\ & \lambda_2 & \ddots & \vdots \\ & & \ddots & b_{n-1n} \\ 0 & & & \lambda_n \end{bmatrix} = T,$$

i.e. A is unitarily similar with an upper triangular matrix T with the eigenvalues of A (counting algebraic multiplicity) on the diagonal.

We omit the proof of this result.

Corollaries.

- The determinant of a matrix is the product of its eigenvalues, counting algebraic multiplicity:

$$\det A = \prod_{i=1}^{n} \lambda_i.$$

- If a matrix A is Hermitian: $A = A^*$, then $T^* = (U^*AU)^* = U^*AU = T$, which implies that T is diagonal with real entries.

- If A is a normal matrix, then T is also normal, which can be shown to imply that T is a diagonal matrix (but may have complex entries).

  Hence, any normal matrix A (including Hermitian matrices) is *diagonalizable*, it is not defective (the algebraic and geometric multiplicities coincide for each eigenvalue), and the columns of U form an orthonormal basis of eigenvectors of A for $\mathbb{F}^n$:

$$A = \underbrace{U}_{\text{unitary}} \underbrace{\Lambda}_{\text{diagonal}} U^* = \sum_{i=1}^{n} \lambda_i \overbrace{u_i u_i^*}^{\text{rank 1 term}} .$$

Finally, by relaxing the conditions on C one may find a matrix which is similar to A, but with further structure.

Theorem 4.11 (Jordan Canonical Form)**.** *Let $A \in \mathbb{F}^{n\times n}$. There exists an invertible matrix X such that*

$$X^{-1}AX = \begin{matrix} m_1 & \dots & m_p \\ \end{matrix}\begin{bmatrix} J_1 & & 0 \\ & \ddots & \\ 0 & & J_p \end{bmatrix}\begin{matrix} m_1 \\ \vdots \\ m_p \end{matrix} = J,$$

where the ***Jordan blocks*** *J_k are $m_k \times m_k$ blocks with the form $J_k = [\lambda_k]$ if $m_k = 1$, or*

$$J_k = \begin{bmatrix} \lambda_k & 1 & & 0 \\ & \ddots & \ddots & \\ & & \ddots & 1 \\ 0 & & & \lambda_k \end{bmatrix} \qquad \text{if } m_k > 1,$$

where λ_k is an eigenvalue of A, and $m_1 + \cdots + m_p = n$.

Lecture 5: Singular Values and Singular Value Decomposition (SVD) (February 18)

5.1 Singular Values

Definition 5.1. *Let $A \in \mathbb{F}^{m\times n}$, and let $\sigma \geqslant 0$, $u \in \mathbb{C}^m$, $v \in \mathbb{C}^n$ two vectors with $\|u\|_2 = \|v\|_2 = 1$, such that*

$$Av = \sigma u \qquad \text{and} \qquad u^*A = \sigma v^*.$$

*Then we say that σ is a **singular value** of A and u, v are respectively the left and right singular vectors associated with σ.*

Remark 5.2. *Note that if σ is a singular value of A,*

- $A^*Av = \sigma A^*u = \sigma(u^*A)^* = \sigma(\sigma v^*)^* = \sigma^2 v$, *and*
- $u^*AA^* = \sigma A^*u = \sigma(u^*A)^* = \sigma^2 u^*$.

Hence σ^2 is an eigenvalue of both AA^* and A^*A.

Proposition 5.3. *Given a matrix $A \in \mathbb{F}^{m\times n}$, the matrix $H = A^*A \in \mathbb{F}^{n\times n}$ is Hermitian positive semi-definite, meaning:*

- $H^* = H$, *and* $x^*Hx \geqslant 0 \forall x \in \mathbb{F}^n$.
- $\operatorname{rank}(H) = \operatorname{rank}(A)$.
- *H is positive definite if an only if* $\operatorname{rank}(A) = n$, *i.e.* $x^*Hx = 0 \implies x = 0$.

Proof. First,

$$H^* = (A^*A)^* = A^*(A^*)^* = A^*A = H.$$

Hence $H = A^*A$ is Hermitian. Next, we compute

$$x^*Hx = (Ax)^*Ax = \|Ax\|_2^2 \geqslant 0.$$

Furthermore, this shows that if $x \in \operatorname{Ker}(H)$, then $x^*Hx = \|Ax\|_2^2 = 0$ and hence $Ax = 0$, that is $x \in \operatorname{Ker}(A)$. Thus $\operatorname{Ker}(A^*A) \subset \operatorname{Ker}(A)$. Since the reverse inclusion is obvious, this proves that $\operatorname{Ker}(A^*A) = \operatorname{Ker}(A)$. Then, using the fundamental theorem of linear algebra,

$$\operatorname{rank}(A) = n - \dim \operatorname{Ker}(A) = n - \dim \operatorname{Ker}(A^*A) = \operatorname{rank}(A^*A).$$

If $\operatorname{rank}(A) = n$, then $\operatorname{Ker}(A) = \{0\}$, thus $x^*Hx = 0$ which implies $x \in \operatorname{Ker}(A)$ now implies $x = 0$. □

5.2 Singular Value Decomposition

Theorem 5.4. *Suppose $A \in \mathbb{F}^{m\times n}$. Then,*

- *There exists unitary matrices $U \in \mathbb{F}^{m\times m}$ and $V \in \mathbb{F}^{n\times n}$ such that*

$$A = U\Sigma V^*,$$

where $\Sigma = diag(\sigma_1, \ldots, \sigma_p)$ is an $m \times n$ diagonal matrix, with $p = \min(m, n)$ and $\sigma_1 \geqslant \ldots, \sigma_p \geqslant 0$ are positive real numbers in decreasing order.

- *The σ_i, $i = 1 \dots p$ are the singular values of A and are uniquely determined.*
- *The corresponding columns u_i, v_i of U and V are respectively the left and right singular vectors of A for $i = 1, \dots, p$ and form an orthonormal basis of $\mathbb{F}^m$ and $\mathbb{F}^n$ respectively.*
- *If $A \in \mathbb{R}^{m \times n}$, then U and V can be chosen as real orthogonal matrices.*

The Singular Value Decomposition, or SVD, has countless applications in scientific computing, data science, engineering, etc. such as Principal Component Analysis (PCA), Proper Orthogonal Decomposition (POD), data fitting, low rank approximation based on the equivalent expression

$$A = \sum_{i=1}^{p} \sigma_i u_i v_i^*.$$

Proof. Let A be rectangular, $m \times n$ matrix with real or complex entries. As we have seen, $A^*A \in \mathbb{F}^{n \times n}$ is Hermitian and positive definite, so it is unitarily diagonalizable with positive eigenvalues: let V be a unitary $n \times n$ matrix such that

$$V^*(A^*A)V = \begin{bmatrix} \lambda_1 & & & 0 \\ & \ddots & & \\ & & \lambda_r & \\ 0 & & & 0_{n-r} \end{bmatrix},$$

where $\lambda_1 \geqslant \cdots \geqslant \lambda_r > \lambda_{r+1} = \cdots = \lambda_n = 0$ are the eigenvalues of A^*A arranged in decreasing order, where $r = \operatorname{rank}(A) = \operatorname{rank}(A^*A)$. Let us write now the block decomposition

$$V = \begin{bmatrix} V_1 & V_2 \end{bmatrix},$$

with $V_1 \in \mathbb{F}^{n \times r}$ and $V_2 \in \mathbb{F}^{n \times (n-r)}$. We can rewrite the expression

$$V^*A^*AV = \begin{bmatrix} V_1^*A^* \\ V_2^*A^* \end{bmatrix} \begin{bmatrix} AV_1 & AV_2 \end{bmatrix} = \begin{bmatrix} V_1^*A^*AV_1 & V_1^*A^*AV_2 \\ V_2^*A^*AV_1 & V_2^*A^*AV_2 \end{bmatrix} = \begin{bmatrix} \lambda_1 & & & 0 \\ & \ddots & & \\ & & \lambda_r & \\ 0 & & & 0_{n-r} \end{bmatrix},$$

so identifying the blocks of the matrix leads to

$$V_1^*A^*AV_1 = \begin{bmatrix} \lambda_1 & & 0 \\ & \ddots & \\ 0 & & \lambda_r \end{bmatrix} \qquad \text{and} \qquad V_2^*A^*AV_2 = 0_{n-r}.$$

Let us first define, since $\lambda_1 \geqslant \cdots \geqslant \lambda_r > 0$,

$$\Sigma_r = \begin{bmatrix} \sqrt{\lambda_1} & & 0 \\ & \ddots & \\ 0 & & \sqrt{\lambda_r} \end{bmatrix} \qquad \text{so that} \qquad \Sigma_r^*\Sigma_r = \begin{bmatrix} \lambda_1 & & 0 \\ & \ddots & \\ 0 & & \lambda_r \end{bmatrix} = V_1^*A^*AV_1,$$

and thus

$$(\Sigma_r^{-1})^* V_1^* A^* A V_1 \Sigma_r^{-1} = (\Sigma_r^{-1})^* \Sigma_r^* \Sigma_r \Sigma_r^{-1} = I_r,$$

or in other terms,

$$(AV_1\Sigma_r^{-1})^*(AV_1\Sigma_r^{-1}) = I_r.$$

This means that $U_1 = AV_1\Sigma_r^{-1}$ is an $m \times r$ matrix with r orthonormal columns. Using, for example, the Gram-Schmidt process, we can find $U_2 \in \mathbb{F}^{m\times(m-r)}$ such that $U = \begin{bmatrix} U_1 & U_2 \end{bmatrix}$ is a unitary matrix (i.e. its columns form an orthonormal basis for $\mathbb{F}^m$).
Next, since $\operatorname{Tr}(AV_2)^*AV_2 = 0 = \|AV_2\|_F^2 = 0$, it follows that $AV_2 = 0$.
Finally, we obtain:

$$AV = \begin{bmatrix} AV_1 & AV_2 \end{bmatrix} = \begin{bmatrix} U_1\Sigma_r & 0_{m(n-r)} \end{bmatrix} = \begin{bmatrix} U_1 & U_2 \end{bmatrix} \begin{bmatrix} \Sigma_r & 0_{r\times(n-r)} \\ 0_{(m-r)\times r} & 0_{(m-r)\times(n-r)} \end{bmatrix} = U\Sigma,$$

where Σ is as in the theorem:

$$A = U\Sigma V^*, \quad \text{with } \Sigma = \operatorname{diag}(\underbrace{\sigma_1, \ldots, \sigma_r}_{r \text{ non-zero square roots of the eiganvalues of } A^*A}, \overbrace{0, \ldots, 0}^{(p-r) \text{ zeros}}).$$

□

Remark 5.5.

- *The singular values of A are the square roots of the largest $p = \min(m, n)$ eigenvalues of A^*A, or equivalently of AA^*.*
- *If A is **square and normal** and λ is an eigenvalue of A, then $|\lambda|$ is a singular value of A. Proof: if A is normal, it is unitarily diagonalizable: there is U unitary such that*

  $$A = U\Lambda U^* = \sum_{i=1}^{n} \lambda_i u_i u_i^* = \sum_{i=1}^{n} \sigma_i u_i (e^{-i\theta_i} u_i)^* = U\Sigma V^*,$$

  where $\lambda_i = |\lambda_i| e^{i\theta_i}$ are the eigenvalues of A with phase θ_i, $\sigma_i = |\lambda_i|$ are the singular values, and $v_i = e^{-i\theta_i} u_i$ are orthonormal vectors forming the columns of a unitary matrix V.

Note that in general, eigenvalues and singular values are not directly related. For example, the matrix

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

has eigenvalues 1 and 0, but the eigenvalues of $AA^* = \begin{bmatrix} 2 & 0 \\ 0 & 0 \end{bmatrix}$ are 2 and 0, hence the singular values of A are $\sqrt{2}$ and 0.
The proof also shows that the full SVD expression can be compactified:

Corollary 5.6 (Compact SVD.)**.** *For any $A \in \mathbb{F}^{m\times n}$ with rank r, there exists $U_1 \in \mathbb{F}^{m\times r}$ and $V_1 \in \mathbb{F}^{n\times r}$ with orthonormal columns:*

$$U_1^*U_1 = I_r, \qquad V_1^*V_1 = I_r,$$

such that

$$A = U_1\Sigma_r V_1^* = \sum_{i=1}^{r} \sigma_i u_i v_i^*,$$

where $\sigma_r = \begin{bmatrix} \sigma_1 & & 0 \\ & \ddots & \\ 0 & & \sigma_r \end{bmatrix}$ with the singular values $\sigma_1 \geqslant \cdots \geqslant \sigma_r > 0$. Note that the columns of U form an orthonormal basis of the range of A and the columns of V form an orthonormal basis of the range of A^*, which is the orthogonal subspace to $\operatorname{Ker}(A)$.

5.3 Examples and Applications.

Examples. Consider the diagonal matrix:

$$D = \begin{bmatrix} 1 & 0 & 0 & 0 \\ 0 & -1 & 0 & 0 \\ 0 & 0 & -3 & 0 \end{bmatrix}.$$

Since $DD^* = \begin{bmatrix} 1 & 0 & 0 \\ 0 & 1 & 0 \\ 0 & 0 & 9 \end{bmatrix}$, the $p = \min(3, 4) = 3$ largest eigenvalues of DD^* are $9, 1, 1$ and the singular values of D are $3, 1, 1$.
Consider the deficient matrix:

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

Now $AA^* = \begin{bmatrix} 2 & 1 \\ 1 & 1 \end{bmatrix}$ has eigenvalues $\frac{1}{2}(3 \pm \sqrt{5})$, hence the singular values of A are $\sqrt{\frac{3\pm\sqrt{5}}{2}}$.

Lecture 6: Applications of the SVD. (February 23)

6.1 Moore Pseudo-Inverse

Given a matrix A with SVD $U\Sigma V^*$, we can define a matrix

$$A^\dagger = V_1 \Sigma_r^{-1} U_1^*,$$

which we call the generalized inverse of A. This object coincides with A^{-1} if A is invertible, and has many more interesting properties (see the homework problems).

6.2 Low-rank Approximation.

The singular value decomposition $A = U\Sigma V^*$ can be recast as the expression

$$A = \sum_{i=1}^{\operatorname{rank}(A)} \sigma_i u_i v_i^*,$$

where the matrix $u_i v_i^*$ is a rank one matrix formed as the outer product of the i-th column of U and i-th column of V, hence each term in the sum $\sigma_i u_i v_i^*$ can be stored using $m + n + 1$ real or complex values. If the rank of A is small, this allows to store A with a reduced amount of storage compared to the mn total entries of A. If the rank of A is not small but the singular values of A decay rapidly, this will allow to *approximate* A using a smaller amount of storage - we will quantify this later on.

6.3 Spectral norm / induced 2-norm

Recall the induced 2-norm on matrices:

$$\|A\|_2 = \max_{\|x\|=1} \|Ax\|_2.$$

Theorem 6.1. *Consider a matrix $A \in \mathbb{F}^{m\times n}$.*

- *Let $\sigma_1(A)$ be the largest singular value of A. Then*

$$\|A\|_2 = \sqrt{\rho(AA^*)} = \sqrt{\rho(A^*A)} = \sigma_1(A).$$

- *If A is Hermitian, then $\|A\|_2 = \rho(A)$.*
- *If A is unitary, then $\|A\|_2 = 1$.*

Proof. Set $A \in \mathbb{F}^{m\times n}$, since A^*A is Hermitian hence diagonalizable with unitary U:

$$U^*A^*AU = \operatorname{diag}(\mu_1, \dots, \mu_n)$$

where $\mu_1 \geqslant \cdots \geqslant \mu_n \geqslant 0$ are the eigenvalues of A.
Let $x \in \mathbb{F}^n$, $\|x\|_2 = 1$, and set $y = U^*x$ such that $\|y\|_2 = 1$ (since the Euclidean norm is unitary invariant). Then

$$\|Ax\|_2^2 = (Ax)^*Ax = x^*A^*Ax = x^*UU^*A^*AUU^*x = y^*(U^*A^*AU)y = \sum_{i=1}^n \overline{y_i}\mu_i y_i = \sum_{i=1}^n \mu_i |y_i|^2.$$

Since $\mu_i \leqslant \mu_1$, we deduce that $\|Ax\|_2^2 \leqslant \mu_1 \sum_{i=1}^n |y_i|^2 = \mu_1$: this holds for any vector x with $\|x\|_2 = 1$. Furthermore, equality is achieved if we choose $y = e_1$, or $x = Ue_1 = u_1$. Hence, by definition

$$\|A\|_2 = \sqrt{\mu_1} = \sqrt{\rho(A^*A)}.$$

The same reasoning can be followed by looking at AA^* to prove that $\|A\|_2 = \sqrt{\rho(AA^*)}$. □

Corollary 6.2. *If $A \in \mathbb{F}^{n\times n}$ is invertible, then*

$$\|A\|_2 = \sigma_1(A), \qquad \text{and} \qquad \|A^{-1}\|_2 = \frac{1}{\sigma_n(A)}.$$

Using the previous characterization of the induced $\|\cdot\|_2$ norm on matrices, we obtain that for $A = U\Sigma V^* = \sum_{i=1}^r \sigma_i u_i v_i^*$,

$$\left\| A - \underbrace{\sum_{i=1}^{k} \sigma_i u_i v_i^*}_{\text{Rank } k \text{ matrix.}} \right\|_2 = \left\| \sum_{i=k+1}^{r} \sigma_i u_i v_i^* \right\|_2 = \sigma_{k+1}(A).$$

Hence, the difference between A and the rank k matrix $\sum_{i=1}^k \sigma_i u_i v_i^*$ is directly related to the singular value $\sigma_{k+1}(A)$, and in particular decreases as k increases.

Theorem 6.3. *For $A \in \mathbb{F}^{m\times n}$ and $1 \leqslant k < p = \min(m,n)$, let*

$$A_k = \sum_{i=1}^{k} \sigma_i u_i v_i^*.$$

Then we have the best-approximation results in the 2- and Frobenius norms:

- $\|A - A_k\|_2 = \sigma_{k+1}(A) = \min_{\text{rank}(()B)\leqslant k} \|A - B\|_2$, *and*
- $\|A - A_k\|_{Fro} = \sigma_{k+1}(A) = \min_{\text{rank}(()B)\leqslant k} \|A - B\|_{Fro}$.

Proof. Take $B \in \mathbb{F}^{m\times n}$ with rank $\text{rank}\,(B) \leqslant k$. Then, $\dim \text{Ker}\,(B) \geqslant n - k$. In particular, since $v_1, \ldots, v_{k+1}$ are independent vectors, the intersection between $\text{Span}(v_1, \ldots, v_{k+1})$ and $\text{Ker}\,(B)$ has dimension at least 1. Let $x \in \mathbb{F}^n$ such that

- $x = \sum_{i=1}^{k+1} y_i v_i$,
- $Bx = 0$,
- $\|x\|_2 = 1 = \|y\|_2$,

such that in particular,

$$Ax = \sum_{i=1}^{p} \sigma_i u_i v_i^* \left(\sum_{j=1}^{k+1} y_j v_j \right) = \sum_{j=1}^{k+1} \sigma_j y_j u_j \qquad \text{since } v_i^* v_j = \delta_{ij}.$$

Thus, since the u_j form an orthonormal basis and $\sigma_1 \geqslant \cdots \geqslant \sigma_k \geqslant \sigma_{k+1}$:

$$\|Ax - Bx\|_2 = \|Ax\|_2 = \left(\sum_{j=1}^{k+1} |\sigma_j y_j|^2 \right)^{1/2} \geqslant \sigma_{k+1} \left(\sum_{j=1}^{k+1} |y_j|^2 \right)^{1/2} = \sigma_{k+1}.$$

Bye definition of the induced norm, this ensures the estimate:

$$\|A - B\|_2 \geqslant \|(A - B)x\|_2 \geqslant \sigma_{k+1}.$$

Since $\|A - A_k\|_2 = \sigma_{k+1}$ as proved above, we have that the best approximation for A by a rank k matrix B, measured in the 2-norm, is A_k. We skip the similar proof for the Frobenius norm. □

6.4 Examples of Numerical Linear Algebra questions.

- Algorithms to solve linear systems:
  Find $x_1, \ldots, x_n$ such that $\sum_{j=1}^{n} a_{ij} x_j = b_i$, or in matrix form
  $$Ax = b.$$
- Algorithms to solve eigenvalue problems:
  $$\text{Find } x, \lambda \text{ such that } Ax = \lambda x.$$
- Algorithms to compute the singular value decomposition, an eigenvalue decomposition, etc.
- Assert stability and efficiency of the above algorithms.

6.5 Solution of linear systems.

As a first question, we are interested in understanding, and ultimately developing algorithms such that, given a square matrix $A \in \mathbb{F}^{n \times n}$, a right-hand side $b \in \mathbb{F}^n$,

$$\text{Solve the linear system} \qquad Ax = b. \tag{6.1}$$

Before developing these algorithms, we need to understand this problem well: in particular, whether it is well-posed, that is if it has a unique solution, and if this solution depends continuously (read: in a stable manner) on the data. The following statements are known to be equivalent:

1. Problem (6.1) has a unique solution x,
2. A is invertible,
3. $\operatorname{rank}(A) = n$,
4. $Ax = 0 \qquad \Leftrightarrow \qquad x = 0.$

There is an explicit formula for the entries of the solution, given by Cramer's rule:

$$x_j = \frac{\Delta_j}{\det A},$$

where Δ_j is the determinant of the matrix obtained by substituting the j-th column of A by b. In practice however, the numerical cost in the order of $(n+1)!$ flops (floating-point operations) to evaluate directly this formula is unacceptable.

Numerical Approaches. There are two broad categories of algorithms for the solution of linear systems:

- **Direct solvers:** for example, Gaussian elimination. These algorithms see to obtain an "exact" answer (modulo rounding errors due to floating-point operations) in a finite number of steps.
- **Iterative solvers:** with this approach, one seeks to reduce the error at each step, but possibly convergence happens only after an infinite number of total steps.

The right choice of approach depends on the matrix, in particular its size, but also its properties (symmetry, etc.)

Lecture 7: Conditioning of Linear Systems (February 25)

7.1 Stability Analysis.

Consider the linear system:

$$Ax = b.$$

How sensitive is the solution x to perturbations in the data A, b?

Definition 7.1. *The condition number of a matrix $A \in \mathbb{F}^{n\times n}$ is defined as the number*

$$K(A) = \|A\|\|A^{-1}\|,$$

where $\|\cdot\|$ is an induced norm.
If A is not invertible, we set $K(A) = +\infty$.

Application

- Matrix-vector product: given an exact computation $x = Ab$, an inexact one writes:

  $$x + \delta x = A(b + \delta b),$$

  where δx is the perturbation in the computed result as a consequence of the perturbation in the data δb. In this case, we have by linearity $\delta x = A\delta b$

  $$\|\delta x\| \leqslant \|A\|\|\delta b\|,$$

  using the properties of the induced norm $\|\cdot\|$. On the other hand, $b = A^{-1}x$ and hence

  $$\|b\| \leqslant \|A^{-1}\|\|x\|,$$

  so after some manipulations we find the estimate

  $$\frac{\|\delta x\|}{\|x\|} \leqslant \frac{\|A\|\|\delta b\|}{\|b\|/\|A^{-1}\|} = K(A)\frac{\|\delta b\|}{\|b\|}.$$

  Hence, the condition number of $\|A\|$ is also the condition number of the matrix-vector product operation with the matrix A.

- Solution of a linear system: given an exact solution x to the problem $Ax = b$, an inexact one writes:

  $$A(x + \delta x) = b + \delta b,$$

  where δx is the perturbation in the computed result as a consequence of the perturbation in the data δb. In this case, we have by linearity $\delta x = A^{-1}\delta b$ and thus

  $$\delta x\| \leqslant \|A^{-1}\|\|\delta b\|,$$

  using the properties of the induced norm $\|\cdot\|$. On the other hand, $b = x$ and hence

  $$\|b\| \leqslant \|A\|\|x\|,$$

  so after some manipulations we find the estimate

  $$\frac{\|\delta x\|}{\|x\|} \leqslant \frac{\|A^{-1}\|\|\delta b\|}{\|b\|/\|A\|} = K(A)\frac{\|\delta b\|}{\|b\|}.$$

  Hence, the condition number of $\|A\|$ is also the condition number of the linear system with the matrix A.

Remark 7.2. • *In general, the condition number depends on the norm: for example, we note*

$$K_\infty(A) = \|A\|_\infty \|A^{-1}\|_\infty.$$

- *The condition number is always greater than one:*

$$K(A) \geqslant 1 \qquad \textit{since} \quad \|A\|\|A^{-1}\| \geqslant \|AA^{-1}\| = \|I\| = 1,$$

  using the properties of induced norms.

- $K(A) = K(A^{-1})$.

Special case: Spectral condition number. When $p = 2$, we compute explicitely

$$\|A\|_2 = \sigma_1(A) \qquad \text{and} \qquad \|A^{-1}\|_2 = 1/\sigma_n(A),$$

where $\sigma_1(A)$ and $\sigma_n(A)$ are respectively the largest and smallest singular values of A. Hence the condition number of A is the ratio:

$$K_2(A) = \frac{\sigma_1(A)}{\sigma_n(A)}.$$

In particular, when A is symmetric positive definite, its singular values are its eigenvalues and hence

$$K_2(A) = \frac{\lambda_{max}}{\lambda_{min}} = \rho(A)\rho(A^{-1}).$$

Because of these properties, we call $K_2(A)$ the spectral condition number of A.

7.2 A priori / Forward Analysis

In this section, we aim to find the result of a perturbation of the matrix A and the right-hand side b on the solution of the linear system $Ax = b$.

Theorem 7.3. *Let $A, \delta A \in \mathbb{F}^{n\times n}$ such that*

$$\|A^{-1}\|\|\delta A\| < 1,$$

and $x, \delta x, b, \delta b \in \mathbb{F}^n$ such that $b \neq 0$,

$$Ax = b \qquad \textit{and} \qquad (A - \delta A)(x + \delta x) = b + \delta b.$$

Then,

$$\frac{\|\delta x\|}{\|x\|} \leqslant \frac{K(A)}{1 - K(A)\frac{\|\delta A\|}{\|A\|}} \left(\frac{\|\delta b\|}{\|b\|} + \frac{\|\delta A\|}{\|A\|} \right)$$

Proof. First, let us prove that $A - \delta A$ is non singular and compute its inverse. To this effect, we define S, the limit of the series

$$S = I + A^{-1}\delta A + (A^{-1}\delta A)^2 + \cdots = \sum_{k=0}^{\infty} (A^{-1}\delta A)^k.$$

This sum converges absolutely because $\|A^{-1}\delta A\| \leqslant \|A^{-1}\|\|\delta A\| < 1$, and by the triangular inequality,

$$\|S\| \quad \leqslant \quad \sum_{k=0}^{\infty} (\|A^{-1}\|\|\delta A\|)^k = \frac{1}{1 - \|A^{-1}\|\|\delta A\|}. \tag{7.1}$$

Furthermore, we observe that

$$A^{-1}\delta_A = \sum_{k=0}^{\infty}(A^{-1}\delta A)^{k+1} = \sum_{k=1}^{\infty}(A^{-1}\delta A)^k = S - I,$$

which leads to the identity

$$(I - A^{-1}\delta A)S = I \qquad \text{or} \qquad S = (I - A^{-1}\delta A)^{-1}.$$

In particular, $(A - \delta A)^{-1} = SA^{-1}$, and $A - \delta A$ is an invertible matrix.
Now, we have by linearity

$$(A - \delta A)x + (A - \delta A)\delta x = b + \delta b,$$

and since $Ax = b$,

$$(A - \delta A)\delta x = \delta b + \delta A x \quad \text{or} \quad \delta x = SA^{-1}(b + \delta A x).$$

Using the properties of the induced norms and the triangular inequality, this leads to

$$\|\delta x\| \leqslant \|S\|\|A^{-1}\| \left(\|b\| + \|\delta A\|\|x\|\right),$$

and using (7.1) for $\|S\|$,

$$\frac{\|\delta x\|}{\|x\|} \leqslant \frac{\|A^{-1}\|}{1 - \|A^{-1}\|\|\delta A\|}\|A\| \left(\frac{\|\delta b\|}{\|A\|\|x\|} + \frac{\|\delta A\|}{\|A\|}\right).$$

To conclude, we note that $\|b\| \leqslant \|A\|x\|$, $\|A\|\|A^{-1}\| = K(A)$ and $\|A^{-1}\|\|\delta A\| = K(A)\frac{\|\delta A\|}{\|A\|}$. □

Corollary 7.4. *If $\delta A = 0$ above (perturbation on the RHS only), then*

$$\frac{1}{K(A)}\frac{\|\delta b\|}{\|b\|} \leqslant \frac{\|\delta x\|}{\|x\|} \leqslant K(A)\frac{\|\delta b\|}{\|b\|}.$$

Indeed, to obtain the left inequality we observe that b is the solution of the linear system $A^{-1}b = x$ and $K(A) = K(A^{-1})$. More particularly, we may be interested in the case where the perturbations are very small, perhaps caused by rounding errors:

$$\frac{\|\delta A\|}{\|A\|} = O(u), \qquad \frac{\|\delta b\|}{\|b\|} = O(u),$$

where $u = \beta^{1-t}$ is the machine precision - on the order of 10^{-16} for double precision (64-bit floating point numbers).

Theorem 7.5. *Assume $\|\delta A\| \leqslant \varepsilon\|A\|$, $\|\delta b\| \leqslant \varepsilon\|b\|$, where $\varepsilon > 0$ and $A, \delta A \in \mathbb{F}^{n\times n}$, $b, \delta b \in \mathbb{F}^n$. Then, if $\epsilon K(A) < 1$, we have*

1. $$\frac{\|x + \delta x\|}{\|x\|} \leqslant \frac{1 + \epsilon K(A)}{1 - \epsilon K(A)},$$

2. $$\frac{\|\delta x\|}{\|x\|} \leqslant \frac{2K(A)}{1 - \epsilon K(A)}\epsilon.$$

Proof. The second estimate follows directly from Theorem 7.3. To get the first one, using the notations from the proof of that theorem we note

$$x + \delta x = SA^{-1}(b + \delta b) = S(x + A^{-1}\delta b),$$

hence

$$\|x + \delta x\| \leqslant \frac{1}{1 - \|A^{-1}\|\|\delta A\|} \left(\|x\| + \|A^{-1}\|\|\delta b\|\right) = \frac{\|x\|}{1 - \varepsilon K(A)} \left(1 + K(A)\frac{\varepsilon\|b\|}{\|A\|\|x\|}\right)$$

Now since $\|b\| \leqslant \|A\|\|x\|$,

$$\frac{\|x + \delta x\|}{\|x\|} \leqslant \frac{1}{1 - \varepsilon K(A)} \left(1 + \varepsilon K(A)\right)$$

□

7.3 Backward Analysis

Importantly, in most cases the chosen numerical algorithm is itself the source of errors (in particular, rounding errors), which are not predetermined by a perturbation at the data level. As such, it is often very useful to observe that the numerical algorithm is producing an *exact* solution $\widehat{x}$ to an *approximate* problem:

$$\widehat{x} = Cb, \qquad \text{where} \quad C \approx A^{-1}.$$

In such cases, the following proposition may prove useful:

Proposition 7.6. *Let $R = AC - I$. If $\|R\| < 1$, then A and C are both invertible and*

$$\|C^{-1}\| \leqslant \frac{\|A\|}{1 - \|R\|}, \quad \|A^{-1}\| \leqslant \frac{\|C\|}{1 - \|R\|}, \quad \textit{and} \quad \frac{\|R\|}{\|A\|} \leqslant \|C - A^{-1}\| \leqslant \frac{\|C\|\|R\|}{1 - \|R\|}. \tag{7.2}$$

If the frame of *backward analysis*, we see C as the exact inverse or solution operator to a modified problem with perturbed matrix $A + \delta A$, such that

$$\delta A = C^{-1} - A = (-AC - I)C^{-1} = -RC^{-1}$$

is small. Indeed, we have the estimate, provided $\|R\| < 1$:

$$\|\delta A\| \leqslant \frac{\|R\|\|A\|}{1 - \|R\|}.$$

7.4 A posteriori Analysis

Finally, given an approximate solution $y \approx x = A^{-1}b$, one may seek to estimate the error $e = y - x$ from known quantities, which at this point include the approximate solution y (which is absent from the original estimate in Theorem 7.3). A good starting point to such analysis is the *residual vector*:

$$r = b - Ay,$$

which measures how y fails to solve the linear system. In particular, since the error writes

$$e = A^{-1}(Ay - b) = -A^{-1}r,$$

we obtain from (7.2):

$$\|e\| \leqslant \frac{\|C\|}{1 - \|R\|}\|r\|,$$

meaning that the norm of r is indeed related to the error. Another estimate may be derived from interpreting

$$Ay = b + r, \qquad \text{i.e.} r = \delta b,$$

which results (using the forward estimate) in the bound

$$\frac{\|e\|}{\|x\|} \leqslant K(A)\frac{\|r\|}{\|b\|}.$$

Variations of these formulae, including the additional effects of rounding errors, have been established for use in modern linear algebra libraries (notably LAPACK implementations).

Lecture 8: Gaussian elimination.

8.1 Solution of triangular systems. March 2

First, let us consider solving upper or lower triangular systems such as:

$$\begin{bmatrix} l_{11} & 0 & 0 \\ l_{21} & l_{22} & 0 \\ l_{31} & l_{32} & l_{33} \end{bmatrix} \begin{bmatrix} x_1 \\ x_2 \\ x_3 \end{bmatrix} = \begin{bmatrix} b_1 \\ b_2 \\ b_3 \end{bmatrix} \qquad \text{or} \qquad \begin{bmatrix} u_{11} & u_{12} & u_{13} \\ 0 & u_{22} & u_{23} \\ 0 & 0 & u_{33} \end{bmatrix} \begin{bmatrix} x_1 \\ x_2 \\ x_3 \end{bmatrix} = \begin{bmatrix} b_1 \\ b_2 \\ b_3 \end{bmatrix}.$$

In such cases, a simple criterion for the system to be nonsingular is that all diagonal elements should be different from zero: $l_{ii} \neq 0$ or $u_{ii} \neq 0$ for $i = 1, 2, 3$.

Forward Substitution. Linear systems $Lx = b$ with a lower triangular coefficient matrix L:

$$\begin{cases} l_{11}x_1 & = b_1, \\ l_{21}x_1 + l_{22}x_2 & = b_2, \\ l_{31}x_1 + l_{32}x_2 + l_{33}x_3 & = b_3, \\ \dots & \\ l_{n1}x_1 + l_{n2}x_2 + \ \dots \ + l_{nn}x_n & = b_n, \end{cases}$$

can be solved without inverting L by the following elimination procedure:

1. $x_1 = b_1/l_{11}$, then
2. $x_2 = (b_2 - l_{21}x_1)/l_{22}$, then
3. $x_3 = (b_3 - l_{31}x_1 - l_{32}x_2)/l_{33}$,
4. ...
5. $x_n = (b_n - l_{n1}x_1 - \cdots - l_{n,n-1}x_{n-1})/l_{nn}$.

This procedure clearly produces a solution, with a cost of order $2 \times \frac{n(n-1)}{2} \approx n^2$ floating-point operations, using the general formula

$$x_1 = \frac{b_1}{l_{11}} \quad \text{then} \quad x_i = \frac{1}{l_{ii}} \left(b_i - \sum_{j=1}^{i-1} l_{ij}x_j \right), \quad \text{for } i = 2, \dots, n.$$

Backward Substitution. Similarly, linear systems $Ux = b$ with an *upper* triangular coefficient matrix L can be solved without inverting U by the similar following procedure:

$$x_n = \frac{b_n}{u_{nn}} \quad \text{then} \quad x_i = \frac{1}{u_{ii}} \left(b_i - \sum_{j=i+1}^{n} u_{ij}x_j \right), \quad \text{for } i = n-1, \dots, 1.$$

The computational cost in floating-point operations is again of the order n^2.

8.2 Solution of general linear systems.

As we shall see, the process of Gaussian elimination converts the solution of one linear system (a difficult task) into the solution of two triangular systems (an easy task, using the substitution methods from the previous paragraph.)
This is a template for algorithms in Numerical Linear Algebra:

Step 1. Transform the problem / matrix, converting the original problem into a series of easier systems in condensed form.

Step 2. Solve the transformed system using the special structures of the condensed form.

Step 3. Recover the solution of the original problem using the solution of the transformed systems.

8.2.1 Elementary matrices

Consider a nonsingular matrix $A \in \mathbb{R}^{n\times n}$, where $a_{11} \neq 0$. We denote $A^{(1)} := A$ and $b^{(1)} := b$, and introduce the multipliers

$$m_{i1} = \frac{a_{i1}^{(1)}}{a_{11}^{(1)}}, \qquad\qquad i = 2, \ldots, n,$$

which allow to eliminate the unknown x_1 from a row (equation) other than the first one:

$$a_{i1}^{(1)} x_1 + a_{i2}^{(1)} x_2 + \cdots + a_{in}^{(1)} x_n = b_i^{(1)}$$

by substracting from it m_{i1} times the first row:

$$\underbrace{m_{i1} a_{11}^{(1)}}_{=a_{i1}^{(1)}} x_1 + m_{21} a_{12}^{(1)} x_2 + \cdots + m_{i1} a_{1n}^{(1)} x_n = m_{i1} b_1^{(1)}$$

yielding the new equation:

$$0 + \underbrace{(a_{i2}^{(1)} - m_{i1} a_{12}^{(1)})}_{a_{i2}^{(2)}} x_2 + \cdots + \underbrace{(a_{in}^{(1)} - m_{i1} a_{1n}^{(1)})}_{a_{in}^{(2)}} x_n = \underbrace{b_i^{(1)} - m_{i1} b_1^{(1)}}_{b_i^{(2)}}.$$

In matrix form, the new set of equations forms a new system, which has the same solution as the first one:

$$\begin{bmatrix} a_{11}^{(1)} & a_{12}^{(1)} & \ldots & a_{1n}^{(1)} \\ 0 & a_{12}^{(2)} & \ldots & a_{1n}^{(2)} \\ \vdots & \vdots & & \vdots \\ 0 & a_{n2}^{(2)} & \ldots & a_{nn}^{(2)} \end{bmatrix} \begin{bmatrix} x_1 \\ x_2 \\ \vdots \\ x_n \end{bmatrix} = \begin{bmatrix} b_1^{(1)} \\ b_2^{(2)} \\ \vdots \\ b_n^{(2)}, \end{bmatrix}$$

where the coefficients of the coefficient matrix $A^{(2)}$ are given by

$$a_{ij}^{(2)} = a_{ij}^{(1)} - m_{i1} a_{1j}^{(1)}$$
$$b_i^{(2)} = b_i^{(1)} - m_{i1} b_i^{(1)}, \qquad 1 < i \leqslant n,\ 1 \leqslant j \leqslant n.$$

It is notable that this coefficient transformation can be recast as a matrix-vector product: introducing the column vector of multipliers

$$m_1 = [0, m_{21}, \ldots, m_{n1}]^T$$

then we find that

$$A^{(2)} = A^{(1)} - m_1 A^{(1)}(1,:) = A^{(1)} - m_1 e_1^T A^{(1)} = (I - m_1 e_1^T) A^{(1)},$$

where $A^{(1)}(1,:) = e_1^T A^{(1)}$ is the first row of the matrix $A^{(1)}$. Hence, if we introduce the matrix

$$M_1 = I - m_1 e_1^T = \begin{bmatrix} 1 & & & 0 \\ -m_{21} & 1 & & \\ \vdots & & \ddots & \\ -m_{n1} & 0 & & 1 \end{bmatrix},$$

then $A^{(2)} = M_1 A^{(1)}$ and $b^{(2)} = M_1 b^{(1)}$.
This motivates the introduction of the following *elementary matrices*:

Definition 8.1. *An elementary lower triangular matrix of order n has the form:*

$$M_k = I_n - m_k e_k^T = \begin{bmatrix} 1 & & & & & \\ & \ddots & & 0 & & \\ & & 1 & & & \\ & & -m_{k+1} & \ddots & & \\ & 0 & \vdots & & \ddots & \\ & & -m_n & 0 & & 1 \end{bmatrix},$$

where $m_k = [0, \ldots, 0, m_{k+1}, \ldots m_n]^T$ is a column vector where the first k entries vanish.

Property 8.2. *Let $M_k = I_n - m_k e_k^T$ as above, then*

- $M_k^{-1} = I_n + m_k e_k^T$,
- $M_{k_1}^{-1} \ldots M_{k_p}^{-1} = I_n + m_{k_1} e_{k_1}^T + \cdots + m_{k_p} e_{k_p}^T$, *when* $k_1 < \cdots < k_p$.

Proof. We compute directly:

$$M_k(I_n + m_k e_k^T) = (I_n - m_k e_k^T)(I_n + m_k e_k^T) = I_n - m_k e_k^T + m_k e_k^T - m_k (e_k^T m_k) e_k^T.$$

Now the k-th entry of m_k is zero, hence $e_k^T m_k = 0$. Therefore

$$M_k(I_n + m_k e_k^T) = I_n,$$

which proves that $M_k^{-1} = I_n + m_k e_k^T$. To prove the second formula, one uses induction on p; let us show that the case $p = 2$, with $k_1 = k < k_2 = l$:

$$M_k^{-1} M_l^{-1} = (I_n + m_k e_k^T)(I_n + m_l e_l^T) = I_n + m_k e_k^T + m_l e_l^T + m_k (e_k^T m_l) e_l^T = I_n + m_k e_k^T + m_l e_l^T,$$

because the k-th entry of m_l is zero since $k < l$. □

This elementary Gauss transformation matrices can be used to create zeros in vectors! Indeed, let $a_k = \begin{bmatrix} a_{1k} & \ldots & a_{kk} & \ldots & a_{nk} \end{bmatrix}^T$ with $a_{kk} \neq 0$, and define

$$m_k = \begin{bmatrix} 0, & \ldots, & 0, & a_{k+1,k}/a_{kk}, & \ldots, & a_{nk}/a_{kk} \end{bmatrix}^T, \qquad M_k = I_n - m_k e_k^T.$$

Then, we have

$$M_k a_k = \begin{bmatrix} a_{1k}, & \ldots, & a_{kk}, & 0, & \ldots, & 0 \end{bmatrix}^T,$$

and we have eliminated all entries below the k-th one.

8.2.2 Continuing the GEM

We can now continue the elimination method started earlier with the first step $A^{(1)} \to A^{(2)}$. Starting from any step k where $A^{(k)}$ has the form

$$A^{(k)} = \begin{bmatrix} a_{11}^{(1)} & a_{12}^{(1)} & & \cdots & & a_{1n}^{(1)} \\ 0 & a_{22}^{(2)} & & \cdots & & a_{2n}^{(2)} \\ & \ddots & \ddots & & & \vdots \\ \vdots & & 0 & a_{kk}^{(k)} & \cdots & a_{kn}^{(k)} \\ & & & \vdots & & \vdots \\ 0 & \cdots & 0 & a_{nk}^{(k)} & \cdots & a_{nn}^{(n)} \end{bmatrix}$$

Note that we have eliminated all entries below the diagonal in columns $1 \dots k-1$, and are looking to continue the process. Using the elementary matrices introduced in the previous section, this can be achieved by forming the matrix

$$M_k = I_n - m_k e_k^T, \qquad \text{where } m_k = \begin{bmatrix} 0 \\ \vdots \\ 0 \\ a_{k+1,k}^{(k)}/a_{kk}^{(k)} \\ \vdots \\ a_{nk}^{(k)}/a_{kk}^{(k)} \end{bmatrix},$$

and then proceed to the next step via

$$A^{(k+1)} = M_k A^{(k)}.$$

Indeed, the resulting matrix $A^{(k+1)}$ has the right form with zeros under the diagonal in columns $1, \dots, k$. At the end of this process, we obtain a matrix

$$A^{(n)} = M_{n-1} \dots M_1 A,$$

which has zeros below the diagonal throughout and is an upper triangular matrix. Let us define

$$U := A^{(n)} = M_{n-1} \dots M_1 A, \qquad b^{(n)} = M_{n-1} \dots M_1 b,$$

then we can solve the transformed system $Ux = b^{(n)}$ using backward substitution. The total cost for this operation is around $2(n-1)n(n+1)/3 + O(n^2)$ floating-point operations (around $2n^3/3$ flops).

Note that the Gaussian Elimination Method terminates safely if and only if $a_{kk}^{(k)} \neq 0$ at every step, for $k = 1, \dots, n-1$. These entries are called the *pivot*.

8.3 Gaussian Elimination as a factorization method

We have seen that Gaussian Elimination transforms a given matrix into an upper triangular one via a series of elementary transformations,

$$U = A^{(n)} = M_{n-1} \cdots M_1 A,$$

which leads to the alternative form

$$A = (M_{n-1} \cdots M_1)^{-1} U = M^{-1} \cdots M_{n-1}^{-1} U.$$

Now $M^{-1} \cdots M_{n-1}^{-1}$ is the product of lower triangular matrices, hence it is a lower triangular matrix which has the specific form according to Prop. 8.2:

$$L = M^{-1} \cdots M_{n-1}^{-1} = I_n + m_1 e_1^T + \cdots + m_{n-1} e_{n-1}^T = \begin{bmatrix} 1 & 0 & \cdots & 0 \\ m_{21} & 1 & \ddots & \vdots \\ \vdots & \ddots & \ddots & 0 \\ m_{n1} & \cdots & m_{n-1,n} & 1 \end{bmatrix}.$$

Note that the multipliers of the Gaussian elimination process appear below the diagonal, which is filled with elements equal to 1.

Definition 8.3 (LU Factorization from the Gaussian Elimination Method)**.**

$$A = LU$$

is the LU factorization of A, where $L = M^{-1} \cdots M_{n-1}^{-1}$ is a unit lower triangular matrix with diagonal elements equal to 1, and $U = M_{n-1} \cdots M_1 A$ is an upper triangular matrix.

LU factorization algorithm via Gaussian elimination. Given a matrix $A \in \mathbb{R}^{n \times n}$:

```
function LUfact(A)
    Set U = A, L = I.
    for k = 1, ..., n − 1 do
        for i = k + 1, ..., n do
            L(i, k) = U(i, k)/U(k, k);
            U(i, k) = 0;
            for h = k + 1, ..., n do
                U(i, j) = U(i, j) − L(i, k)U(k, j);
            end for
        end for
    end for
    return L, U
end function
```

Existence and Uniqueness.

Definition 8.4. *The k-th leading principal minor of a matrix $A \in \mathbb{R}^{n \times n}$ for $1 \leqslant k \leqslant n$ is the submatrix*

$$A_k = \begin{bmatrix} a_{11} & \cdots & a_{1k} \\ \vdots & & \vdots \\ a_{k1} & \cdots & a_{kk} \end{bmatrix},$$

i.e. the first k rows and columns of A.

Theorem 8.5. *An $n \times n$ matrix has a unique LU factorization if and only if its leading principal minors $A_k, k = 1, \ldots, n-1$ are all non-singular.*

The proof of this result is worked out in Homework 5.

Lecture 9: LU factorization with pivoting. March 4

9.1 Difficulties of Gaussian elimination without pivoting.

- The Gaussian elimination algorithm fails if any of the pivots vanishes, $a_{kk}^{(k)} = 0$.
- One obtains an even worse situation if $a_{kk}^{(k)} \approx 0$: then the method may finish, but (due to rounding errors and catastrophic cancellation) the algorithm may produce the wrong results!

Example by Forsythe and Moler (1967). Let us consider in 3-digit decimal arithmetic the matrix

$$A = \begin{bmatrix} 10^{-4} & 1 \\ 1 & 1 \end{bmatrix}.$$

We compute the first multiplier $m_{21} = 1/10^{-4} = 10^4$ and

$$M_1 = \begin{bmatrix} 1 & 0 \\ -10^4 & 1 \end{bmatrix} \qquad \text{and} \qquad U = A^{(2)} = M_1 A = \begin{bmatrix} 10^{-4} & 1 \\ 0 & \mathrm{fl}(1 - 10^4) \end{bmatrix} = \begin{bmatrix} 10^{-4} & 1 \\ 0 & -10^4 \end{bmatrix},$$

where we note that the bottom right entry in U, whose exact value is -0.9999×10^4 has been rounded down to -0.100×10^5 since we are operating in 3-digit floating-point arithmetic. Hence, the computed factors $\widehat{L}$, $\widehat{U}$ read

$$\widehat{L} = M_1^{-1} = \begin{bmatrix} 1 & 0 \\ -10^4 & 1 \end{bmatrix}, \qquad \widehat{U} = \begin{bmatrix} 10^{-4} & 1 \\ 0 & -10^4 \end{bmatrix}.$$

We compute from this the product

$$\widehat{L}\widehat{U} = \begin{bmatrix} 10^{-4} & 1 \\ 1 & 0 \end{bmatrix} \quad \text{and} \quad A - \widehat{L}\widehat{U} = \begin{bmatrix} 0 & 0 \\ 1 & 0 \end{bmatrix}.$$

This is a really large error: $\|A - \widehat{L}\widehat{U}\|_{Fro} = 1$, even though the conditioning of A is very good: $\mathrm{cond}(A) = 2.6\ldots$! The issue is that the pivot is very small: 10^{-4}, resulting in a huge multiplier m_{21}. The problem can be entirely avoided if we exchange the two rows:

$$A' = \begin{bmatrix} 1 & 1 \\ 10^{-4} & 1 \end{bmatrix}, \qquad M_1' = \begin{bmatrix} 1 & 0 \\ -10^{-4} & 1 \end{bmatrix}, \qquad U' = M_1' A' = \begin{bmatrix} 1 & 1 \\ 0 & \mathrm{fl}(1 - 10^{-4}) \end{bmatrix},$$

such that

$$\widehat{L}' = \begin{bmatrix} 1 & 0 \\ -10^{-4} & 1 \end{bmatrix}, \qquad \widehat{U}' = \begin{bmatrix} 1 & 1 \\ 0 & 1 \end{bmatrix}.$$

This time, rounding happened again in the computation of $\widehat{U}$ yet

$$\widehat{L}'\widehat{U}' = \begin{bmatrix} 1 & 1 \\ 10^{-4} & \mathrm{fl}(1 + 10^{-4}) \end{bmatrix} = \begin{bmatrix} 1 & 1 \\ 10^{-4} & 1 \end{bmatrix} = A.$$

9.2 Permutation Matrices

In order to enact the row exchanges used in the previous example, we introduce:

Definition 9.1. *A square matrix is called a permutation matrix if there is exactly one non-zero entry in each row and in each column, which equals one.*
If $(\alpha_1, \ldots, \alpha_n)$ is a permutation of $(1, \ldots, n)$, then the associated permuation matrix is

$$P = \begin{bmatrix} e_{\alpha_1} & \cdots & e_{\alpha_n} \cdot \end{bmatrix}^T$$

Similarly, $P^{-1} = P^T = \begin{bmatrix} e_{\alpha_1} & \cdots & e_{\alpha_n} \cdot \end{bmatrix}$ is also a permutation matrix.

Effect of multiplying a matrix by a permutatioin matrix. Let $P_1 = \begin{bmatrix} e_{\alpha_1} & \cdots & e_{\alpha_n}. \end{bmatrix}^T$ be the permutation matrix associated with the permutation $(\alpha_1, \ldots, \alpha_n)$, then we compute the entries of $P_1 A$:

$$(P_1 A)_{ij} = \sum_{k=1}^{n} P_{1,ik} A_{kj} = \sum_{k=1}^{n} \delta_{k,\alpha_i} Akj = A_{\alpha_i, j},$$

Therefore

$$P_1 A = \begin{bmatrix} \alpha_1\text{-th row of } A \\ \vdots \\ \alpha_n\text{-th row of } A \end{bmatrix}$$

is the matrix obtained by permuting the *rows* of A in the order $\alpha_1, \ldots, \alpha_n$.
Similarly, if $P_2 = P_1^T = P_1^{-1} = \begin{bmatrix} e_{\alpha_1} & \cdots & e_{\alpha_n} \end{bmatrix}$ then AP_2 is the matrix obtained by permuting the *columns* of A in the order $\alpha_1, \ldots, \alpha_n$. To sum things up: multiplying a matrix by a permutation matrix...

- from the left: permutes the rows of A,
- from the right: permutes the columns of A.

In the earlier example, the permutation matrix associated with the permutation $(2, 1)$ leads to the transformation

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

9.3 Gaussian Elimination with Partial Pivoting.

As a rule of thumb, disaster in the GEM may be avoided by choosing a good pivot (with large magnitude). There are many strategies to achieve this. We can look for a good pivot:

- in the k-th column of $A^{(k)}$ **(partial pivoting)**, or
- in a submatrix of $A^{(k)}$ **(complete pivoting)**.

Algorithm: Gaussian Elimination with partial pivoting. Set $A^{(1)} = A$, then at step k, $k = 1, \ldots, n-1$:

1. Identify the largest element (by magnitude) in column k below the diagonal. Let it be
$$a_{r_k,k}^{(k)} \qquad \text{(Note that } r_k \geqslant k.\text{)}$$
2. Exchange the rows r_k and k bringing $a_{r_k,k}^{(k)}$ on the diagonal.
3. Apply the normal k-th step of the Gaussian elimination method.

In terms of matrix multiplications, this algorithm yields the following sequence:

$$\begin{aligned} A^{(1)} &= A, \\ A^{(2)} &= M_2 P_2 A^{(2)}, \\ &\vdots \\ A^{(n)} &= M_{n-1} P_{n-1} A^{(n-1)} \end{aligned}$$

Hence we obtain the upper triangular matrix after $n-1$ steps:

$$U = A^{(n)} = M_{n-1} P_{n-1} M_{n-2} P_{n-2} \ldots M_1 P_1 A.$$

LU factorization from GEPP. Let us now show that the above process yields a factorization $PA = LU$, where P is a permutation matrix and LU is a factorization of the matrix A with permuted rows.

- First, by construction:
$$U = M_{n-1}P_{n-1}M_{n-2}P_{n-2}\dots M_1P_1A.$$
is an upper triangular matrix.

- Second, let $M = M_{-1}P_{n-1}\cdots M_1P_1$. How to extract L, P from M? We note that since P_i represents the exchange of two rows, the action of P_iP_i returns the matrix to its original state, i.e. $P_i^2 = I_n$. Let us define:
$$\begin{aligned} M'_{n-1} &= M_{n-1}, \\ M'_{n-2} &= P_{n-1}M_{n-2}P_{n-1}, \\ M'_{n-3} &= P_{n-1}P_{n-2}M_{n-3}P_{n-2}P_{n-1}, \\ &\vdots \\ M'_1 &= P_{n-1}\dots P_2M_1P_2\dots P_{n-1}, \end{aligned}$$
Then we observe that
$$\begin{aligned} M &= M_{n-1}P_{n-1}M_{n-2}P_{n-2}\dots M_1 \\ &= M_{n-1}(P_{n-1}M_{n-2}P_{n-1})(P_{n-1}P_{n-2}M_{n-2}P_{n-2}P_{n-1})P_{n-1}P_{n-2}P_{n-3}M_{n-4}\dots M_1 \\ &= M'_{n-1}M'_{n-2}\cdots M'_1P_{n-1}\cdots P_1. \end{aligned}$$
Denoting $P = P_{n-1}\cdots P_1$ and $L = (M'_{n-1}M'_{n-2}\cdots M'_1)^{-1}$, it follows that
$$PA = LU.$$
Furthermore, construction of the M'_k from M_k involves multiplying M_k from the left and right by permutation matrices P_j, $j \geqslant l$, exchanging its rows and columns j and r_j. Because $r_j \geqslant j$ for all j and M_k has the form
$$M_k = I_n - m_ke_k^T = \begin{bmatrix} 1 & & & & & \\ & \ddots & & 0 & & \\ & & 1 & & & \\ & & -m_{k+1} & \ddots & & \\ & 0 & \vdots & & \ddots & \\ & & -m_n & 0 & & 1 \end{bmatrix},$$
these operations only permute the multipliers $-m_{k+1}, \dots, -m_n$ but keeps the structure intact. We can see this also by direct computation:
$$\begin{aligned} M'_k &= P_{n-1}\cdots P_{k+1}(I_n - m_ke_k^T)P_{k+1}\cdots P_{n-1} \\ &= P_{n-1}\cdots P_{k+1}P_{k+1}\cdots P_{n-1} + P_{n-1}\cdots P_{k+1}m_k(P_{k+1}\cdots P_{n-1}e_k)^T \quad = I_n + m'_ke_k^T, \end{aligned}$$
where $m'_k = P_{n-1}\cdots P_{k+1}m_k$ is a vector of permuted multipliers (still with entries $m'_1, \dots, m'_k$ equal to zero), because e_k has entries all zero for indices $j, r_j > k$. Hence, as before we obtain
$$L = (M'_1)^{-1}\cdots(M'_{n-1})^{-1} = I_n + m'_1e_1^T + \cdots + m'_{n-1}e_{n-1}^T.$$

Lecture 10: Complete pivoting for LU and other factorizations. March 9

10.1 Complete Pivoting.

In this paragraph, we investigate a strategy of *complete pivoting* for the LU factorization algorithm, meaning that at step $k = 1, \ldots, n-1$, given the partial factorization of the type

$$A^{(k)} = \begin{bmatrix} a_{11}^{(1)} & a_{12}^{(1)} & & \cdots & & a_{1n}^{(1)} \\ 0 & a_{22}^{(2)} & & \cdots & & a_{2n}^{(2)} \\ & \ddots & \ddots & & & \vdots \\ \vdots & & 0 & a_{kk}^{(k)} & \cdots & a_{kn}^{(k)} \\ & & & \vdots & & \vdots \\ 0 & \cdots & 0 & a_{nk}^{(k)} & \cdots & a_{nn}^{(n)} \end{bmatrix} :$$

- Identify the largest element **in magnitude** among all elements of the submatrix of $A^{(k)}$ with rows and column indices larger than k, aka the block $\begin{bmatrix} a_{kk}^{(k)} & \cdots & a_{kn}^{(k)} \\ \vdots & & \vdots \\ a_{nk}^{(k)} & \cdots & a_{nn}^{(n)} \end{bmatrix}$; let it be $a_{r_k,s_k}^{(k)}$ such that $r_k, s_k \geqslant k$.
- Exchange the rows k and r_k and columns k and s_k of $A^{(k)}$, bringing the pivot $a_{r_k,s_k}^{(k)}$ to the diagonal in position (k,k).
- Apply the k-th step of Gaussian elimination:

  $$A^{(k+1)} = M_k P_k A^{(k)} Q_k,$$

  where P_k, Q_k are permutation matrices exchanging respectively the rows k, r_k and the columns k, s_k of $A^{(k)}$.

At the end of this procedure, we obtain an upper triangular matrix $U = A^{(n)}$, with the full process being summarized as

$$U = M^{n-1} P_{n-1} M_{n-2} P_{n-2} \cdots M_1 P_1 A Q_1 \cdots Q_{n-1}.$$

Let us define

$$\begin{cases} & Q = Q_1 Q_2 \cdots Q_{n-1}, \\ & P = P_{n-1} P_{n-2} \cdots P_1, \\ L = (M_1')^{-1} \cdots (M_{n-1}')^{-1} & \text{as for the GEPP previously.} \end{cases}$$

Then we obtain the fully pivoted LU factorization:

$$PAQ = LU.$$

LU factorization algorithm via Gaussian elimination with complete pivoting.
Given a matrix $A \in \mathbb{R}^{n\times n}$:

function LUCPFACT(A)
  Set $U = A$, $L = I$, $p = [1:n]$, $q = [1:n]$.
  for $k = 1, \ldots, n-1$ **do**
    Determine r_k, s_k such that $|u_{r_k,s_k}| = \max_{k\leqslant i,j\leqslant n} |u_{ij}|$.
    $U(:,k) \leftrightarrow U(:,s_k)$; ▷ Exchange the columns k, s_k of U
    $q(k) \leftrightarrow q(s_k)$; ▷ Update column permutation vector q

    Exchange the rows k, r_k of L, U:
    $U(k,:) \leftrightarrow U(r_k,:)$; ▷ Exchange the rows k, r_k of U
    $L(k, 1:k-1) \leftrightarrow L(r_k, 1:k-1)$; ▷ Permute multipliers stored in L
    $p(k) \leftrightarrow p(r_k)$; ▷ Update column permutation vector p

    $L(k+1:n, k) = U(k+1:n, k)/U(k,k)$;
    $U(k+1:n, k) = 0$;
    $U(k+1:n, k+1:n) = U(k+1:n, k+1:n) - L(k+1:n, k)U(k, k+1:n)$;
  end for
  return L, U, p, q ▷ Now $A(p,q) = LU$.
end function

10.2 Other types of factorizations.

10.2.1 The LDM^T factorization.

Given an LU factorization of U, let us introduce the diagonal matrix:

$$D = \begin{bmatrix} u_{11} & & 0 \\ & \ddots & \\ 0 & & u_{nn} \end{bmatrix}, \qquad \text{and} M = (D^{-1}U)^T.$$

The matrix M is then **unit** lower triangular (i.e., it has all ones on its diagonal).

Theorem 10.1. *If all leading principal minors A_k, $k = 1, \ldots, n$ of A are non-singular, then there exists a unique diagonal matrix D and unit lower triangular matrices L, M such that*

$$A = LDM^T.$$

10.2.2 The LDL^T factorization.

If furthermore, $A = A^T$ is symmetric, then the factorization further simplifies:

$$A^T = (LDM^T)^T = MDL^T = LDM^T = A,$$

so by uniqueness of the factorization, we have $M = L$.

Theorem 10.2. *If all leading principal minors A_k, $k = 1, \ldots, n$ of A are non-singular, then there exists a unique diagonal matrix D and unit lower triangular matrix L such that*

$$A = LDL^T.$$

The advantage of such a factorization is that the computational cost and memory necessary to compute it is halved compared to the usual LU factorization.

10.2.3 The Cholesky factorization.

If, in addition, A is a symmetric positive definite matrix: $A = A^T$ and

$$x^T A x > 0 \quad \Leftrightarrow \quad x \neq 0,$$

then all of its leading principal minors are also symmetric positive definite matrix: indeed, for $x_k \in \mathbb{R}^k \backslash \{0\}$,

$$x_k^T A_k x_k = \begin{bmatrix} x_k \\ 0 \end{bmatrix}^T A \begin{bmatrix} x_k \\ 0 \end{bmatrix} > 0.$$

In such a case, the elements $u_{11}, \ldots, u_{nn}$ forming the diagonal of the matrix D must all be positive, since the determinants $\det(A_k) = u_{11} \cdots u_{kk}$ are strictly positive for $k = 1, \ldots, n$. Then we can compute their square roots:

$$S = \begin{bmatrix} \sqrt{u_{11}} & & 0 \\ & \ddots & \\ 0 & & \sqrt{u_{nn}} \end{bmatrix} \qquad \text{such that} \qquad D = S^2,$$

leading to the new factorization $A = LS^2L^T = LS(LS)^T = HH^T$ where H is a lower triangular matrix.

Theorem 10.3. *Let $A \in \mathbb{R}^{n \times n}$ be a symmetric positive definite matrix. Then, there exists a lower triangular matrix $H \in \mathbb{R}^{n \times n}$ with strictly positive diagonal entries such that*

$$A = HH^T.$$

This is the Cholesky factorization. The entries of H are given by the algorithm:

---

function CHOLESKY(A)
  Set $H(1,1) = \sqrt{A(1,1)}$.
  for $i = 2, \ldots, n$ **do**
    for $j = 1, \ldots, n-1$ **do**
      $H(i,j) = \left(A(i,j) - \sum_{k=1}^{j-1} H(j,k)H(i,k) \right) / H(j,j)$
    end for
    $H(i,i) = \left(A(i,i) - \sum_{k=1}^{i-1} H(i,k)^2 \right)^{1/2}$
  end for
  return H ▷ Now $A = HH^T$.
end function

Proof. We proceed by induction on n, without relying on the LU factorization.

- The result is trivial for $n = 1$: $A = [a_{11}] = [\sqrt{a_{11}}][\sqrt{a_{11}}]^T$.
- If $n > 1$, write the block decomposition

$$A = \begin{bmatrix} A_{n-1} & v \\ v^T & a_{nn} \end{bmatrix},$$

where A_{n-1} is the $n-1$-th leading principal minor of A and $v = [a_{n1}, \ldots, a_{n,n-1}]^T \in \mathbb{R}^{n-1}$. By induction, assume there exists a matrix H_{n-1} given by the above algorithm such that

$$A_{n-1} = H_{n-1} H_{n-1}^T.$$

Then, we seek a matrix H of the form

$$H = \begin{bmatrix} H_{n-1} & 0 \\ h^T & h_{nn} \end{bmatrix} \qquad \text{such that} \qquad HH^T = \begin{bmatrix} H_{n-1}H_{n-1}^T & H_{n-1}h \\ (H_{n-1}h)^T & h_{nn}^2 + h^T h \end{bmatrix} = \begin{bmatrix} A_{n-1} & v \\ v^T & a_{nn} \end{bmatrix} = A.$$

The vector h has entries $[h_{n1}, \ldots, h_{n,n-1}]^T$. Hence, we want $H_{n-1}h = v$. Since H_{n-1} is lower triangular, this system can be uniquely solved by forward substitution, with the formula:

$$h_{nj} = \left(a_{nj} - \sum_{k=1}^{j-1} h_{jk} h_{nk} \right) / h_{jj}, \qquad \text{for } j = 1, \ldots, n-1.$$

Compare this to Line 5 of the above algorithm, with $i = n$. In addition, we want $a_{nn} = h_{nn}^2 + h^T h$. Since $h \in \mathbb{R}^{n-1}$ is determined by the algorithm above, this gives as a unique solution

$$h_{nn} = \left(a_{nn} - \sum_{j=1}^{n-1} h_{nj}^2 \right)^{1/2}.$$

Compare this to Line 7 of the algorithm, again with $i = n$. The resulting number h_{nn} is real and positive, because H_{n-1} is real (by induction) and

$$h_{nn}^2 \det(H_{n-1})^2 = \det(A) > 0,$$

hence h_{nn}^2 and also h_{nn} is strictly positive, and H has all real entries with strictly positive diagonal entries.

□

Computational complexity. The total cost of this procedure is about i^2 at each step for $i = 2, \ldots, n$ (solution of the system by forward substitution), adding up to a total $n^3/3$ floating-point operations over the loop, with an additional $2i$ operations and square root computation to compute the element h_{ii}, adding up to n^2 flops. The total, $n^3/3 + n^2$ flops and n square root computations is about half of the cost of the LU factorization.

Stability property. Let $A \in \mathbb{R}^{n \times n}$, symmetric positive definite. The above algorithm yields an approximate Cholesky factor $\widehat{H}$ in floating-point arithmetic. It can be shown (Demmel, 1989) that this result satisfies the relation

$$A + E = \widehat{H}\widehat{H}^T,$$

where the perturbation $E = [e_{ij}]$ can be bounded componentwise:

$$|e_{ij}| \leqslant \frac{(n+1)u}{1-(n+1)u} \sqrt{a_{ii} a_{jj}},$$

where u is the round-off unit ($\approx 10^{-16}$ in double precision). This result shows that the Cholesky factorization is always backwards stable (which is not necessarily the case for the LU factorization, as we will see.)

Lecture 11: Solving linear systems. Stability and Accuracy. (March 11)

11.1 A practical method for solving linear systems.

Any factorization of A suggests a method for solving the linear system

$$Ax = b,$$

for one or more right-hand sides b. We will focus on the LU factorizations, possibly with pivoting:

1. $A = LU$ (no pivoting),
2. $PA = LU$ (partial pivoting),
3. $PAQ = LU$ (complete pivoting).

In all cases, the linear system can be solved by solving two triangular systems with coefficient matrices L and U, possibly with permutations of the resulting vectors:

1. $Ax = b \quad \Leftrightarrow \quad LUx = b$, suggesting the method
$$\begin{cases} Ly = b & \text{solved using forward substitution,} \\ Ux = y & \text{solved using backward substitution.} \end{cases}$$
2. $Ax = b \quad \Leftrightarrow \quad PAx = Pb \quad \Leftrightarrow \quad LUx = Pb$:
$$\begin{cases} Ly = Pb & \text{solved using forward substitution,} \\ Ux = y & \text{solved using backward substitution.} \end{cases}$$
3. $Ax = b \quad \Leftrightarrow \quad PAQQ^T x = Pb \quad \Leftrightarrow \quad LU(Q^T x) = Pb$:
$$\begin{cases} Lz = Pb & \text{solved using forward substitution,} \\ Uy = z & \text{solved using backward substitution,} \\ x = Qy & \text{permutation of the vector } y. \end{cases}$$

Computational Complexity:

- LU decomposition, $\sim \frac{2}{3}n^3 + O(n^2)$;
- Two triangular solves: $\sim 2n^2$;
- Permutations: no operations, just data movement.

The total cost is thus on the order of $\frac{2}{3}n^3 + O(n^2)$ floating-point operations, with the actual solves contributing relatively little the the overall cost.
Pivoting strategies add to this cost a total of $O(n^2)$ comparisons in the case of the partial pivoting strategy, and $O(n^3)$ in the case of the complete pivoting strategy - improving the stability of the approach at the expense of greatly increasing the computational cost of the solution.

In particular, computing explicitely the inverse of A can now be achieved by solving n linear systems:

$$A^{-1} = \begin{bmatrix} A^{-1}e_1 & \cdots A^{-1}e_n \end{bmatrix}$$

for a total cost around $\frac{2}{3}n^3 + 2n^3 + O(n^2)$ floating-point operations.

Numerical stability. Floating-point operations in most computers obey the following IEEE error model: if $\cdot$ is one of the operations $+$, $-$, $\times$ or $\div$, and no overflow occurs then

$$x \odot y = (x \cdot y)(1+\delta), \qquad \text{for } x, y \in \mathbb{F}(\beta, t, L, U), \quad |\delta| \leqslant u,$$

where $u = \frac{1}{2}\beta^{1-t}$ is the round-off unit. In base 2 with $t = 53$ binary digits (double-precision accuracy), we have $u = 2^{-53} \approx 10^{-16}$.

11.1.1 Error analysis for back-substitution.

Let $Ux = b$ be a linear system with upper triangular coefficient matrix U and denote by $\widehat{x}_j \approx x_j$ the computed solution, using the back-substitution algorithm but taking into account floating-point rounding errors. Denoting by $\oplus$, $\ominus$, $\otimes$, $\oslash$ the floating-point operations approximating the exact operations $+$, $-$, $\times$ or $\div$:
Recall the process:

- $\widehat{x}_n = b_n \oslash u_{nn}$,
- $\widehat{x}_{n-1} = (b_{n-1} \ominus u_{n-1,n} \otimes \widehat{x}_n) \oslash u_{n-1,n-1}$,
- $\cdots$,
- $\widehat{x}_1 = (b_1 \ominus u_{12} \otimes \widehat{x}_2 \ominus \cdots \ominus u_{1n} \otimes \widehat{x}_n) \oslash u_{11}$.

Using the error model above with $\alpha_{.,.}$, $\beta_{.,.}$ and $\delta_.$ indicating errors smaller than u, the round-off unit, occuring through floating-point computations:

- First step:

$$\widehat{x}_n = (b_n/u_{nn})(1+\delta_n) \qquad = \frac{\overbrace{1+\delta_n}^{1/\widehat{u}_{nn}}}{u_{nn}} b_n =$$

  which can thus be rewritten:

$$\widehat{x}_n = \frac{1}{\widehat{u}_{nn}} b_n,$$

- Second step:

$$\widehat{x}_{n-1} = \left[b_{n-1} - u_{n-1,n}\widehat{x}_n \overbrace{(1+\alpha_{n-1,n})}^{\text{from } \otimes} \right] \overbrace{(1+\beta_{n-1,n})}^{\text{from } \ominus} / u_{n-1,n-1} \overbrace{(1+\delta_n)}^{\text{from } \oslash}$$

$$= \left[\overbrace{(1+\beta_{n-1,n})b_{n-1}}^{\widehat{b}_{n-1}} - \overbrace{(1+\beta_{n-1,n})(1+\alpha_{n-1,n})u_{n-1,n}}^{\widehat{u}_{n-1,n}} \widehat{x}_n \right] \frac{\overbrace{1+\delta_n}^{1/\widehat{u}_{n-1,n-1}}}{u_{n-1,n-1}}$$

  or

$$\widehat{x}_{n-1} == \frac{1}{\widehat{u}_{n-1,n-1}} \left(\widehat{b}_{n-1} - \widehat{u}_{n-1,n}\widehat{x}_n \right),$$

- $\cdots$,

and we see that, without going through all the details, the process becomes equivalent to the *exact* solution of a *perturbed* triangular system

$$\widehat{U}\widehat{x} = \widehat{b}.$$

Explicit inspection of the coefficients $\widehat{u}_{ij}$ and $\widehat{b}_i$ shows directly that

$$\frac{|\widehat{u}_{ij} - u_{ij}|}{|u_{ij}|} \leqslant nu + O(u^2), \qquad \frac{|\widehat{b}_i - b_i|}{|b_i|} \leqslant (n-1)u + O(u^2).$$

Such estimates lead to the following stability result:

Theorem 11.1 (Backwards stability of back-substitution)**.** *Let $\widehat{x}$ be the solution of $Ux = b$ computed in finite precision with floating-point round-off unit u using the back-substitution algorithm. Then, $\widehat{x}$ is the exact solution of a system $\widehat{U}\widehat{x} = \widehat{b}$, where $\widehat{U} = U + \delta U$, $\widehat{b} = b + \delta b$ with $U, \delta U$ upper triangular,*

$$\frac{|\delta u_{ij}|}{|u_{ij}|} \leqslant nu + O(u^2), \qquad \frac{|\delta b_i|}{|b_i|} \leqslant (n-1)u + O(u^2).$$

In particular, in any norm $\|\cdot\|$ on vectors and matrices we have

$$\frac{\|\delta U\|}{\|U\|} = O(u), \qquad \frac{\|\delta b\|}{\|b\|},$$

and by the stability theorem of linear systems, we have the forward stability estimate

$$\frac{\|\widehat{x} - x\|}{\|x\|} = K(U)O(u).$$

11.1.2 Error estimates for Gaussian elimination.

When rounding errors are taken into account, the Gaussian elimination algorithms described in previous lectures produce *approximate* factors $\widehat{L}$, $\widehat{U}$ such that

$$\widehat{L}\widehat{U} = A + \delta A,$$

where δA is a perturbation resulting from the imperfect computation. One can estimate, if $nu < 1$ then the following entrywise estimate holds:

$$|\delta A| \leqslant \frac{nu}{1 - nu} |\widehat{L}||\widehat{U}|,$$

where we have adopted the notation $|A| = [|a_{ij}|]$. However, we do not control directly the entries of the factors $\widehat{L}$, $\widehat{U}$, so the goal is in general to find a stability bound of the form:

$$|\delta A| \leqslant g(u)\|A\|.$$

In general, such an estimate can only be obtained if one controls the growth of elements during the Gaussian elimination process, i.e. the maximum magnitude of the entries in the sequence of matrices

$$A^{(1)},\ A^{(2)},\ \ldots, A^{(n)} = \ U.$$

Definition 11.2. *The **growth factor** $\rho(A)$ is the quantity*

$$\rho = \frac{\max \alpha_1, \ldots, \alpha_n}{\alpha},$$

where $\alpha = \max_{i,j} |a_{ij}$ and $\alpha_k = \max_{i,j} |a_{ij}^{(k)}$.

In particular, whenever pivoting (whether complete or partial) is used, it is easy to check that the elements of the matrix L are bounded:

$$|l_{ik}| = \frac{|a_{ik}^{(k)}|}{|a_{rk,k}^{(k)}|} \leqslant 1,$$

since the pivot $a_{rk,k}^{(k)}$ is the largest entry by magnitude in the column k below the diagonal, or even larger in the complete pivoting case. Furthermore, we have

$$|u_{ij}| \leqslant \rho \max_{ij} |a_{ij}|.$$

Theorem 11.3 (Round-off property for GEPP method)**.** *The matrices $\widehat{L}$ and $\widehat{U}$ computed via GEPP satisfy*

$$\widehat{L}\widehat{U} = A + \delta A,$$

where $\|\delta A\|_\infty \leqslant 8n^3 \rho \|A\|_\infty u + O(u^2)$.

Proof: omitted.
To claim backwards stability, the question is then: how large can the growth factor $\rho(A)$ get?

Growth factor for GEPP The following result is known:

Using GEPP, $\rho(A)$ can be as big as 2^{n-1}.

This worst-case scenario can be attained, for example for matrices of the type:

$$A = \begin{bmatrix} 1 & 0 & \cdots & 0 & 1 \\ -1 & 1 & \ddots & \vdots & \vdots \\ -1 & -1 & \ddots & 0 & 1 \\ \vdots & \vdots & \ddots & 1 & 1 \\ -1 & -1 & \cdots & -1 & 1 \end{bmatrix}.$$

However, in most cases, this extreme behavior does not happen! For example:

- For symmetric positive definite matrices: $\rho(A) = 1$,
- For tri-diagonal matrices: $\rho(A) \leqslant 2$,
- ...

Growth factor for GECP Complete pivoting yields a better upper range for the growth factor:

$$\rho(A) \leqslant \sqrt{n} \left(2 3^{1/2} 4^{1/3} \cdots n^{1/n-1}\right)^{1/2}$$

In most cases, the additional stability gained is not worth the additional computational complexity due to the complete pivoting.

Without pivoting, the growth factor cannot be bounded. This shows that Gaussian elimination without pivoting is a completely unstable algorithm!

11.1.3 Iterative Refinement.

Finally, let us sketch a simple method allowing to increase the accuracy of the solution obtained using one of the LU factorizations above. Let $\widehat{x}$ be the approximate computed solution to the system $Ax = b \approx A\widehat{x}$.
Usually, the residual vector $r = b - A\widehat{x}$ is different from zero. To improve on the computed solution, one may solve the system again with r as the new right-hand side:

$$A\widehat{c} \approx r,$$

using the factorization at hand. Then, because $\|r\|$ is much smaller than $\|b\|$, the relative error on $\widehat{c}$ is much smaller than the relative error on $\widehat{x}$, and

$$\widehat{y} = \widehat{x} + \widehat{c}$$

is usually a better approximation of the exact solution x. This process can be repeated to increase progressively the accuracy of the solution, usually until the residual is small enough:

Iterative refinement.
Given A, b and an initial approximation to the solution $\widehat{x}^{(0)}$:

1: **function** ITERATIVEREFINEMENT(A, b, $\widehat{x}^{(0)}$)
2: **for** $i = 0, 1, \ldots$ until convergence **do**
3: Compute $r^{(i)} = b - Ax^{(i)}$,
4: Solve $Az = r^{(i)}$,
5: Update $x^{(i+1)} = x^{(i)} + z$,
6: If $\|z\|/\|x^{(i+1)}\| < tolerance$, terminate.
7: **end for**
8: **end function**

Analysis: if $\||A^{-1}||\widehat{L}||\widehat{U}|\|_\infty$ is small enough, this iterative process will reduced the error by a fixed factor at each step.

Lecture 12: Householder Triangularization. (March 16)

New problem of interest: we turn now to the solution of least-squares problems of the type

$$\text{Find } x \text{ minimizing the functional } \|Ax - b\|_2,$$

where $A \in \mathbb{R}^{m\times n}$ is in general a rectangular matrix, b is a vector of length m and x is a vector of length n.

- When A is square and invertible, the solution is simply the solution of the linear system $Ax = b$: $x = A^{-1}b$,
- If the SVD of A is available, then the solution may be found using the pseudo-inverse: $x = A^{\dagger}b$,
- If $A \in \mathbb{R}^{m\times n}$ with $m \geqslant n$ and $\text{rank}\,(()\,A) = n$, the tool of choice is the QR factorization:

  $$A = QR,$$

  where Q is a matrix with orthogonal columns, either square of size $m \times m$ (full factorization) or of size $m \times n$ (reduced factorization), and R is an upper trapezoidal / triangular matrix of size $m \times n$ (full factorization) or of size $n \times n$ (reduced factorization).

The QR factorization may be computed several ways:

- Using Householder reflectors;
- Using Givens rotations;
- Using the Classical or Modified Gram-Schmidt algorithms.

We will present the first method in this lecture, the Gram-Schmidt approach in the next one, and the Givens rotations will be introduced if time permits in later discussions.

12.1 Householder matrices.

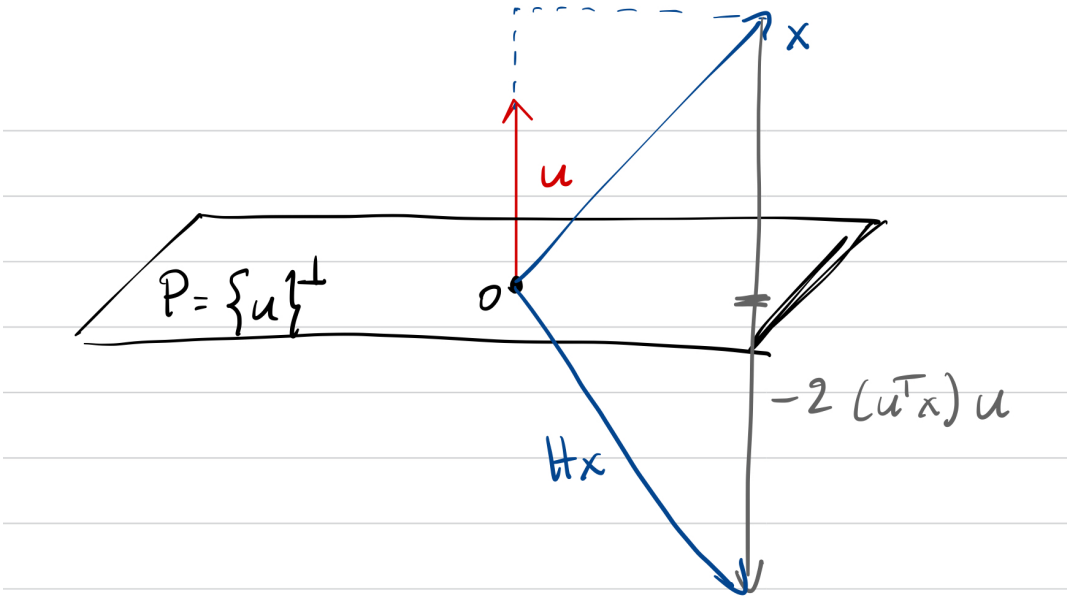
Definition 12.1. *A matrix of the form*

$$H = I_m - 2\frac{uu^T}{u^Tu},$$

where $u \in \mathbb{R}^m$ is a nonzero vector (called the Householder vector), is called a Householder matrix, or Householder reflector, or Householder transformation.

Remark 12.2. *These transformations bear the name of Alston S. Householder, an American numerical analyst (1904-1993) who invented them and the Householder method, among many other contributions to numerical linear algebra and numerical analysis.*

These transformations have a simple geometric interpretation: taking $\|u\| = 1$ for simplicity, we draw the following diagram, upon which one observes that the result Hx is simply the reflection of a vector x across the hyperplane of vectors orthogonal to u:



As a result, these matrices have the following properties:

Proposition 12.3 (Properties of Householder reflectors)**.** *Let $H = I_m - 2\frac{uu^T}{u^Tu}$, where $\mathbb{R}^m \ni u \neq 0$ is the Householder vector. Then*

1. *H is symmetric: $H^T = H$,*
2. *H is orthogonal: $H^T = H^{-1}$,*
3. *$H^2 = I_m$,*
4. *$Hu = -u$,*
5. *$Hv = v$ iff $v^Tu = 0$,*
6. *If u is a nonzero multiple of $x - y$, with x, y two distinct vectors in $\mathbb{R}^m$ such that $\|x\|_2 = \|y\|_2$, then*

$$Hx = y.$$

Proof. Define $\beta = 2/u^Tu$. To prove 1/, we compute directly

$$H^T = (I_m - \beta uu^T)^T = I_m - \beta(u^T)^T u^T = I_m - \beta uu^T = H.$$

Next, 2/ follows from the computation

$$H^TH = (I_m - \beta uu^T)(I_m - \beta uu^T) = I - \beta uu^T - \beta uu^T + \beta^2 uu^T uu^T = I + (\beta^2(u^Tu) - 2\beta)uu^T,$$

since we notice that $\beta^2 u^Tu = 4/(u^Tu)^2 u^Tu = 4/u^Tu = 2\beta$.
Now 3/ is a consequence of 1/ and 2/, 4/ and 5/ are left as an exercise (and can be observed directly on the diagram above.)
Finally, we show 6/: taking x, y as in the proposition, we define $u = \alpha(x - y)$ and note

$$x = \frac{1}{2}(x + y) + \frac{1}{2}(x - y) \quad \text{and} \quad Hx = \frac{1}{2}H(x + y) + \frac{1}{2}H(x - y).$$

Now, since $x + y$ is a multiple of u it follows that $H(x - y) = -(x - y)$, and in addition

$$(x + y)^T u = \alpha(x + y)^T(x - y) = \alpha(x^T x - x^T y + y^T x - y^T y) = 0,$$

because $x^T y = y^T x$ (symmetry of the scalar product) and $x^T x = \|x\|_2^2 = \|y\|_2^2 = y^T y$. Therefore, we have $H(x + y) = x + y$, so

$$Hx = \frac{1}{2}(x + y) - \frac{1}{2}(x - y) = y.$$

□

12.2 Computing matrix-vector and matrix-matrix products involving Householder transformations.

When using Householder transformations, one should avoid forming explicit the matrix H to apply it to a vector or a matrix. Instead, the special structure allows to compute the result with a much lower complexity:

- Matrix-vector product:
$$Hx = (I - \beta uu^T)x = x - \beta(u^T x)u.$$
The scalar product $u^T x$ may be computed in $2m$ flops, then computing the difference $x - (\beta u^T x)u$ takes an additional $2m$ flops.
In total, the computation of Hx takes $4m$ floating-point operations.

- Matrix-matrix product: for $A \in \mathbb{R}^{n\times m}$,
$$HA^T = (I - \beta uu^T)A^T = A^T - u(\beta Au)^T \quad \text{or} \quad AH = A(I - \beta uu^T) = A - (\beta Au)u^T$$
The matrix-vector product βAu may be computed in $2mn$ flops, then computing the outer product $\beta u(Au)^T$ takes an additional mn flops, for a total of $4mn$ floating-point operations including the matrix difference.

- Product of Householder reflectors:
$$Q = H_1 \cdots H_r \qquad \text{with} \quad H_i = I - \beta_i u_i u_i^T.$$
This product may be computed in $4(m^2 r - mr^2 + r^3/3)$ floating-point operations.

- Finally, the product $Q^T A$, where Q is as above and $A \in \mathbb{R}^{m\times n}$, can be computed two ways:
  - if the matrix Q is computed explicitely, in $2m^2 n$ floating-point operations,
  - if computing the sequence of matrix products with the Householder reflectors $H_r \cdots H_1 A$ and $r \leqslant m$, then the cost is only $2nr(2m - r)$.

Numerical stability. It can be shown that, when $\widehat{H}$ is a Householder reflector computed in floating-point arithmetic:

$$\begin{aligned} \|\widehat{H} - H\| &= O(u), \\ \mathrm{fl}(\widehat{H}A) &= H(A + E), && \text{where } \|E\|_2 = O(u\|A\|_2), \\ \mathrm{fl}(A\widehat{H}) &= (A + E)H, && \text{where } \|E\|_2 = O(u\|A\|_2). \end{aligned}$$

Hence, all computations involving Householder reflectors are inherently backward stable.

12.3 Creating zeros in vectors.

Theorem 12.4. *Given $x \neq \pm\|x\|_2 e_1$, let H be the Householder reflector defined by the Householder vector $u = x \mp \|x\|_2 e_1$. Then*

$$Hx = \pm\|x\|_2 e_1.$$

Proof. From Prop. 12.3 with $y = \pm\|x\|_2 e_1$, the result follows. □

Hence, there is an easily computed Householder reflection allowing us to transform a given vector x into a multiple of e_1:

$$x = \begin{bmatrix} x_1 \\ \vdots \\ x_m \end{bmatrix} \qquad \xrightarrow{H} \qquad Hx = \|x\|_2 e_1 = \begin{bmatrix} \|x\|_2 \\ 0 \\ \vdots \\ 0 \end{bmatrix}.$$

Notes.

- Typically, to avoid cancellations we choose the sign that makes the first entry of u as large as possible in magnitude, i.e.:
$$u = x + \operatorname{sign}(x_1)\|x\|_2 e_1.$$
- u only differs from x in the first component.
- For additional stability, one may choose to scale the Householder vector, e.g.:
$$u = \frac{1}{\mu}\left(x + \operatorname{sign}(x_1)\|x\|_2 e_1\right), \qquad \text{where } \mu = \max_i |x_i|.$$

12.4 Householder triangularization.

Let us now apply a sequence of Householder transformations with the goal of obtaining an upper triangular or trapezoidal matrix R:

$$H_n \cdots H_1 A = R.$$

We begin by setting $A^{(1)} = A$, an $m \times n$ real matrix with $m \geqslant n$.

Step 1. Let a_1 be the first column of $A^{(1)} = A$. Using a Householder reflector H_1 with vector $u_1 = a_1 + \operatorname{sign}(a_{11})\|a_1\|_2 e_1$, by the previous theorem we have $H_1 a_1 = [\|a_1\|_2, 0, \ldots, 0]^T$ and hence

$$A^{(2)} := H_1 A^{(1)} = \begin{bmatrix} \times & \cdots & \cdots & \times \\ 0 & \times & \cdots & \times \\ \vdots & \vdots & & \vdots \\ 0 & \times & \cdots & \times \end{bmatrix},$$

i.e. we have eliminated all entries below the diagonal in the 1st column of A.

Step 2. Now, let $\widehat{A}^{(2)}$ be the submatrix formed by the last $m-1$ rows and $n-1$ columns of $A^{(2)}$: in block form,

$$A^{(2)} = \begin{bmatrix} \times & \times \\ 0 & \widehat{A}^{(2)} \end{bmatrix}$$

Let $\widehat{a}_2 = [a_{22}^{(2)}, \dots, a_{m2}^{(2)}]^T$ be the first column of $\widehat{A}^{(2)}$, we can form a Householder reflector $\widehat{H}_2$ of size $m-1 \times m-1$ with vector $\widehat{u}_2 = \widehat{a}_2 + \operatorname{sign}(a_{22}^{(2)})\|\widehat{a}_2\|_2 \widehat{e}_1$, such that by the previous theorem we have $\widehat{H}_2\widehat{a}_2 = [\|\widehat{a}_2\|_2, 0, \dots, 0]^T$. To apply $\widehat{H}_2$ to the submatrix $\widehat{A}^{(2)}$ of $A^{(2)}$, we can form the Householder reflector

$$H_2 = \begin{bmatrix} I_1 & 0 \\ 0 & \widehat{H}_2 \end{bmatrix} = I - 2u_2u_2^T/u_2^Tu_2 \qquad \text{with} \quad u_2 = \begin{bmatrix} 0 \\ \widehat{u}_2 \end{bmatrix},$$

such that

$$A^{(3)} := H_2A^{(2)} = \begin{bmatrix} \times & \times \\ 0 & \widehat{H}_2\widehat{A}^{(2)} \end{bmatrix} \qquad = \begin{bmatrix} \times & \cdots & \cdots & \cdots & \times \\ 0 & \times & \cdots & \cdots & \times \\ \vdots & 0 & \times & \cdots & \times \\ \vdots & \vdots & \vdots & & \vdots \\ 0 & 0 & \times & \cdots & \times \end{bmatrix},$$

i.e. we have eliminated all entries below the diagonal in the first two columns of A.

Step k. The process is now clear. Assuming that, after $k-1$ steps we have obtained a matrix of the form

$$A^{(k)} = \begin{bmatrix} \times & \cdots & \cdots & \cdots & \cdots & \times \\ 0 & \ddots & & & & \vdots \\ \vdots & \ddots & \times & \cdots & \cdots & \times \\ \vdots & & 0 & \times & \cdots & \times \\ \vdots & & \vdots & \vdots & & \vdots \\ 0 & \cdots & 0 & \times & \cdots & \times \end{bmatrix},$$

we denote by $\widehat{A}^{(k)}$ the $(m-k+1) \times (n-k+1)$ block of entries on the bottom right of the matrix, we denote by $\widehat{a}_k = [a_{kk}^{(k)}, \dots, a_{mk}^{(k)}]^T$ its first column and form a Householder reflector with vector $\widehat{u}_k = \widehat{a}_k + \operatorname{sign}(a_k^{(k)})\|\widehat{a}_k\|_2\widehat{e}_1$ that will eliminate all entries in the first column on $\widehat{A}^{(k)}$ below the diagonal. We form the Householder reflector

$$H_k = \begin{bmatrix} I_{k-1} & 0 \\ 0 & \widehat{H}_k \end{bmatrix} = I - 2u_ku_k^T/u_k^Tu_k \qquad \text{with} \quad u_k = \begin{bmatrix} 0_{k-1} \\ \widehat{u}_k \end{bmatrix},$$

then we obtain $A^{(k+1)} = H_kA^{(k)}$ with zeros below the diagonal in columns $1, \dots, k$, and continue onto step $k+1$.

Conclusion. After n steps, we obtain $R := A^{(n+1)}$ an upper triangular matrix, which is formed by the sequence of products

$$H_n \cdots H_1A = R.$$

Hence, if we form $Q = H_1 \cdots H_n$ then, since $H_i = H_i^{-1} = H_i^T$:

$$A = QR, \qquad \text{with } Q^TQ = I_m, \quad R \text{ upper triangular.}$$

The full algorithm is detailed as follows.

Householder QR algorithm.
Input: an $m \times n$ matrix A, with $m \geqslant n$.
Output: (i) the Householder vectors $\mathbf{u}_1, \ldots, \mathbf{u}_n$,
(ii) an upper triangular matrix R of size $m \times n$.
Storage:

- The matrix R is stored in the upper triangular part of A,
- Householder vector components: $u_{k+1,k}, \ldots, u_{m,k}$ in the (strictly) lower triangular part of A,
- $u_{11}, \ldots, u_{kk}$ in a separate array $\mathbf{v}$.

1: **function** HOUSEHOLDERQR(A)
2: **for** $k = 1, \ldots, n$ **do**
3: Form the Householder vector $\widehat{\mathbf{u}}_k = [u_{kk}, \ldots, u_{mk}]^T$ with

$$\begin{cases} u_{kk} = a_{kk} + \operatorname{sign}(a_{kk})\|\widehat{\mathbf{a}}_k\|_2, \\ u_{k+1,k}, \ldots, u_{m,k} = a_{k+1,k}, \ldots, a_{m,k} \end{cases},$$

such that $\widehat{H}_k \widehat{\mathbf{a}}_k = \widehat{H}_k \begin{bmatrix} a_{kk} \\ a_{k+1,k} \\ \vdots \\ a_{mk} \end{bmatrix} = \begin{bmatrix} r_{kk} \\ 0 \\ \vdots \\ 0 \end{bmatrix}$, where $r_{kk} = \|\widehat{a}_k\|_2$.

4: $v_k = u_{kk}$.
5: $a_{kk} = r_{kk}$.
6: $\beta = 2/\widehat{\mathbf{u}}_k^T \widehat{\mathbf{u}}_k$
7: $A(k:m, k+1:n) = A(k:m, k+1:n) - \beta \widehat{\mathbf{u}}_k \left(A(k:m, k+1:n)^T \widehat{\mathbf{u}}_k \right)^T$
8: **end for**
9: **return** A, **v**
10: **end function**

Remark 12.5. *Q is not formed explicitely; rather, the Householder vectors $\widehat{u}_1, \ldots, \widehat{u}_k$ are stored for later application of the corresponding reflectors when multiplication by Q or Q^T is needed.*

Computational complexity. For each step k: when $m = n$, we need

- about $4(n-k)$ floating-point operations to construct $\widehat{\mathbf{u}}_k$,
- about $4(n-k)^2$ flops to update $\widehat{A}^{(k)}$.

These costs add up to about $4\sum_{k=1}^{n}(n-k) + (n-k)^2 \sim \dfrac{4n^3}{3} + O(n^2)$ flops.
When $m \geqslant n$ in general, the total cost is about $2n^2(m - n/3)$ floating-point operations.

Computational stability. The result of the Householder QR factorization method in floating-point arithmetic yields

$$\widehat{Q}\widehat{R} = A + E,$$

where $\widehat{Q}$, $\widehat{R}$ are the computed factors, and $\|E\|_F \leqslant g(n)\|A\|_F u$, where $g(n)$ is a slowly increasing function of n and u the round-off precision.

Lecture 13: QR Factorization, Modified Gram Schmidt. (March 18)

13.1 Full and reduced QR factorizations.

The Householder algorithm presented in the previous lecture yields the so-called **full QR factorization:**

- a matrix $Q \in \mathbb{F}^{m\times m}$, orthogonal ($Q^TQ = I$) in the real case, and unitary ($Q^*Q = I$) in the complex case),
- and a matrix $R \in \mathbb{F}^{m\times n}$ upper trapezoidal ($r_{ij} = 0$ for $i > j$), such that

$$A = QR.$$

Now we note that R has all zeros under the diagonal, and if $m \geqslant n$ we can reduce this factorization:

$$A = QR = \begin{bmatrix} \widehat{Q} & \widehat{Q} \end{bmatrix} \begin{bmatrix} \widehat{R} \\ 0 \end{bmatrix} = \widehat{Q}\widehat{R},$$

where $\widehat{Q}$ is composed of the first n columns of Q and $\widehat{R}$ of the first n rows of R.
The resulting smaller factors $\widehat{Q} \in \mathbb{F}^{m\times n}$, a matrix with orthogonal columns such that $\widehat{Q}^*\widehat{Q} = I_n$ and $\widehat{R} \in \mathbb{F}^{n\times n}$, an upper triangular square matrix, form the **reduced QR factorization** (also called thin or economy-sized factorization):

$$A = \widehat{Q}\widehat{R}.$$

Remark 13.1. *Note that $A^*A = (\widehat{Q}\widehat{R}) * (\widehat{Q}\widehat{R}) = \widehat{R}^*\widehat{R}$, i.e. $\widehat{R}^*$ is the Cholesky factor of the symmetric positive definite matrix A^*A.*

Property 13.2. *The columns of $\widehat{Q}$ form an orthonormal basis for the range of A.*

This property shows that computing the (reduced) QR factorization of A is equivalent to generating an orthonormal basis out of a set of vectors, such as the columns of A.

Proposition 13.3.

- *Every matrix $A \in \mathbb{F}^{m\times n}$ with $m \geqslant n$ has a full QR factorization as well as a reduced QR factorization.*
- *If A has full rank n, there exists a unique reduced QR factorization with all positive entries on the diagonal of R.*
- *If $m < n$, then the QR factorization takes the form $A = Q\begin{bmatrix} R_1 & R_2 \end{bmatrix}$ where $R_1 \in \mathbb{F}^{m\times m}$ is upper triangular, and $R_2 \in \mathbb{F}^{m\times n-m}$ is rectangular.*

Proof. Existence results from the Householder algorithm presented earlier. The new result is the uniqueness for full rank A. Assume that

$$\widehat{Q}_1\widehat{R}_1 = \widehat{Q}_2\widehat{R}_2$$

or, since $\widehat{Q}_1^*\widehat{Q}_1 = I_n$ and R_2 is invertible,

$$\widehat{Q}_1^*\widehat{Q}_1\widehat{R}_1\widehat{R}_2^{-1} = \widehat{Q}_1^*\widehat{Q}_2 \qquad \text{or} \qquad \widehat{R}_1\widehat{R}_2^{-1} = \widehat{Q}_1^{-1}\widehat{Q}_2.$$

Since the left-hand side is an upper triangular matrix with positive diagonal entries and the right-hand side a unitary one, they must be in fact the identity, which yields

$$\widetilde{R}_1 = \widetilde{R}_2, \qquad \text{and} \qquad \widetilde{Q}_1 = \widetilde{Q}_2.$$

□

13.2 Gram-Schmidt algorithm for the QR factorization

Since the columns of $\widehat{Q}$ form an orthonormal basis for the columns of A, an alternative to the Householder approach is to use the Gram-Schmidt algorithm to construct an orthonormal set of vectors out of the columns of A. Suppose

$$A = \begin{bmatrix} a_1 & \cdots & a_n \end{bmatrix}, \qquad a_i \in \mathbb{F}^m.$$

Step 1. Compute

$$q_1 = \frac{1}{r_{11}} a_1, \qquad \text{where } r_{11} = \|a_1\|_2,$$

Step 2. Next,

$$\begin{aligned} \widehat{q}_2 &= a_2 - r_{12} q_1, && \text{where } r_{12} = q_1^* a_2, \\ q_2 &= \frac{1}{r_{22}} \widehat{q}_2 && \text{where } r_{22} = \|\widehat{q}_2\|_2, \end{aligned}$$

More generally: **Step k.**

$$\begin{aligned} \widehat{q}_k &= a_k - \sum_{i=1}^{k-1} r_{ik} q_i, && \text{where } r_{ik} = q_i^* a_k, \\ q_k &= \frac{1}{r_{kk}} \widehat{q}_k && \text{where } r_{kk} = \|\widehat{q}_k\|_2, \end{aligned}$$

until we reach $k = n$. Note that, in exact arithmetic, we have the resulting relation for each column of the matrix A,

$$a_k = q_1 r_{1k} + \cdots + q_{k-1} r_{k-1,k} + q_k r_{kk},$$

which is summed up by the expression

$$A = \widehat{Q}\widehat{R}$$

where $\widehat{Q} = \begin{bmatrix} q_1 & \cdots & q_n \end{bmatrix}$ and $\widehat{R}$ is the upper triangular matrix with entries r_{ij} computed in the course of the algorithm above, which is called the **classical Gram-Schmidt algorithm**.

Numerical issues. It turns out that, implemented in floating-point arithmetic, the classical Gram-Schmidt suffers from catastrophic floating-point cancellation. As a result, the columns q_k of the computed matrix are often not orthogonal to each other, and the resulting decomposition is, at best, not useful.

Modified Gram-Schmidt algorithm A simple modification is enough to stabilize the computation of the reduced QR factorization by the Gram-Schmidt algorithm, resulting in a much more stable computation.

Modified Gram-Schmidt algorithm.

Input: an $m \times n$ matrix $A = \begin{bmatrix} a_1 & \cdots & a_n \end{bmatrix}$, with $\operatorname{rank}(A) = n$.

Output: Matrices $\widehat{Q}$ and $\widehat{R}$ of the reduced QR factorization.

function ModifiedGramSchmidtQR(A)
    Set R an $n \times n$ matrix with all zero entries.
    Set $Q = A$.
    for $k = 1, \ldots, n$ **do**
        $r_{kk} = \|q_k\|_2$.
        $q_k = q_k / r_{kk}$.
        for $j = k + 1, \ldots, n$ **do**
            $r_{kj} = q_k^* q_j$
            $q_j = q_j - r_{kj} q_k$
        end for
    end for
    return $\widehat{Q}$, $\widehat{R}$.
end function

Lecture 14: Solving least-squares problems with factorizations. (March 23)

14.1 Problem statement.

Given a coefficient matrix $A \in \mathbb{F}^{m\times n}$ and a vector $b \in \mathbb{F}^n$, we seek to determine the vector $x \in \mathbb{F}^m$ which minimizes

$$\|Ax - b\|_2. \tag{14.1}$$

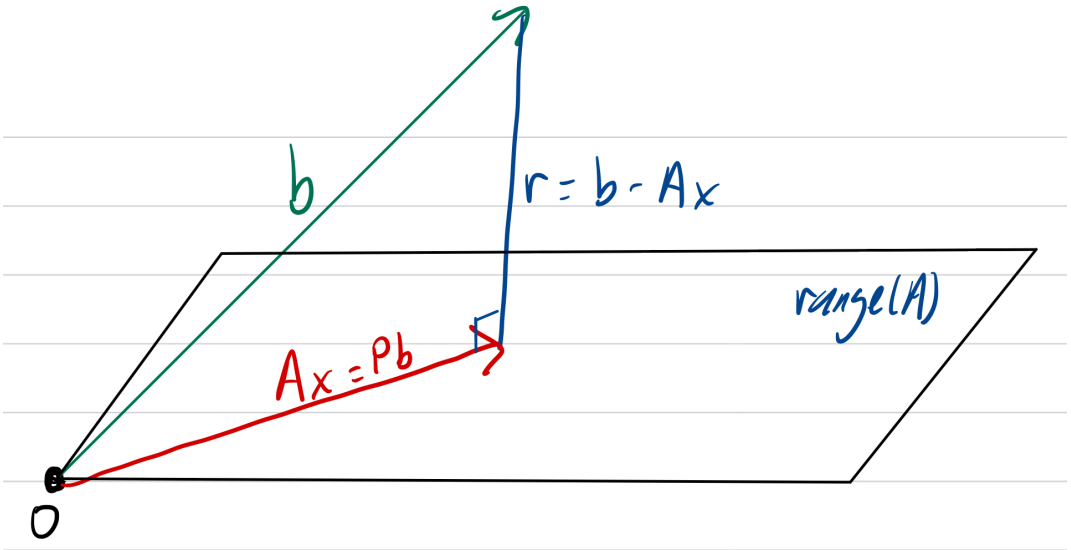
Definition 14.1. *The vector $r = b - Ax$ is called the **residual**.*

Definition 14.2. *If the least-squares problem has more than one solution, the one having minimal Euclidean norm is called the **minimum length** or **minimum norm solution**.*

- If $m < n$ (less equations than unkowns), we say that the system is **underdetermined**, and
- if $m > n$ (more equations than unknowns), the system is **overdetermined**.

14.2 Geometric interpretation.

The solution $x \in \mathbb{F}^n$ is the vector x such that Ax is the orthogonal projection of b onto the range of A:



As a result, there is always at least one solution to the least-squares problem. Furthermore, we note that $\|r\|_2$ is the distance of b to the range of A.
Next, we observe that $r = b - Ax$ is orthogonal to the range of A, and since the columns of A are a generating set of vectors for the range, we deduce that

$$A^*r = 0, \qquad \text{or} \quad A^*Ax = A^*b.$$

Definition 14.3. *Let $A \in \mathbb{F}^{m\times n}$. The linear system*

$$A^*Ax = A^*b$$

is called the normal equations.

14.3 Existence, Uniqueness and Properties

We have seen that the least-squares problem always has a solution, which satisfies the normal equations. In general, we can only expect uniqueness if the system is overdetermined and the matrix has full rank, i.e. $\operatorname{rank}(A) = n$.

Theorem 14.4. *Given $A \in \mathbb{F}^{m\times n}$ with $m \geqslant n$, $b \in \mathbb{F}^n$, define*

$$r(x) = b - Ax \in \mathbb{F}^n.$$

The following statements are equivalent:

(a) x minimises $\|r(x)\|_2$,

(b) x satisfies $A^*r = 0$,

(c) x solves the normal equations $A^*Ax = A^*b$,

(d) x solves $Ax = Pb$, where P is the orthogonal projector onto $\operatorname{Ran}(A)$.

Proof. **First, a geometrical proof.** We start with showing the equivalence (a) $\Leftrightarrow$ (b). We know that

$$b - Pb \;\perp\; Pb - Ax \;\in\; \operatorname{Ran}(A).$$

Hence,

$$\|r(x)\|_2^2 = \|b - Ax\|_2^2 = \|b - Pb + Pb - Ax\|_2^2 = \|b - Pb\|_2^2 + \|Pb - Ax\|_2^2 \geqslant \|b - Pb\|_2.$$

Now $\|b - Pb\|_2$ is the minimum for $\|r(x)\|_2$, attained if and only if $Ax = Pb$. Note that such an x always exists since $Pb \in \operatorname{Ran}(A)$ by definition.
Next, we show (d) $\Leftrightarrow$ (b). Writing again the decomposition

$$r = (b - Pb) + (Pb - Ax),$$

we note that $r \perp \operatorname{Ran}(A)$ if and only if $Pb - Ax = 0$.
Finally, we note that (b) is trivially equivalent to (c).
Another, analytical proof. For simplicity, let us investigate the real case. Here

$$\Phi(y) = \|r(y)\|_2^2 = (b - Ay)^T(b - Ay).$$

We develop this expression:

$$\begin{aligned}\Phi(y) &= b^Tb - b^TAy - y^TA^Tb + y^TA^TAy \\ &= \|b\|_2^2 - 2y^TA^Tb + y^TA^TAy.\end{aligned}$$

The gradient of this expression is:

$$\nabla\Phi(y) = 2(A^TAy - A^Tb).$$

Therefore the normal equations are necessarily satisfied by the minimum x, solution of the least-squares problem, since the gradient must be zero at any critical point. □

14.4 Numerical solution: over-determined case.

14.4.1 Normal equations method

When $m \geqslant n$ and A has full rank: $\operatorname{rank}(A) = n$, then A^*A is Hermitian positive definite.

Proposition 14.5. *If* $\operatorname{rank}(A) = n$ *(A has full rank), then the least-squares problem has a unique solution*

$$x = (A^*A)^{-1}A^*b = A^\dagger b,$$

where $A^\dagger$ *is the pseudo-inverse of the matrix A.*

This proposition then leads to the following numerical method: we can form the matrix A^*A and then solve the normal equations, using the Cholesky factorization of $A^*A = RR^*$ with R upper triangular.

1: **function** NormalEquations LS(A)
2: Compute A^*A and A^*b.
3: Compute the Cholesky factorization

$$A^*A = RR^*.$$

4: Solve $Ry = A^*b$ using back-substitution.
5: Solve $R^*x = y$ using forward substitution.
6: **end function**

Complexity: the computation of A^*A can be achieved in mn^2 floating-point operations, exploiting the symmetry of the result. The Cholesky factorization is then computed in $\frac{1}{3}n^3$ floating-point operations, and the triangular systems solved in $O(n^2)$ operations.
The total cost is hence on the order $mn^2 + \frac{n^3}{3} + O(n^2)$ flops.

Numerical difficulties.

- A^*A may be close to singular, and
- $K_2(A^*A) = K_2(A)^2$ is often much larger than the condition number of A - meaning that the normal equations may have worse conditioning than the least-squares problem itself.

14.4.2 QR factorization approach.

Given $A \in \mathbb{F}^{m\times n}$ with $\operatorname{rank}(A) = n$, write its reduced QR factorization $A = \widehat{Q}\widehat{R}$ where $\widehat{Q} \in \mathbb{F}^{m\times n}$ has orthogonal columns, forming an orthonormal basis of $\operatorname{Ran}(A)$, and $\widehat{R} \in \mathbb{F}^{n\times n}$ is upper triangular.
In particular, the orthogonal projector onto the range is obtained as

$$P = \widehat{Q}\widehat{Q}^* \in \mathbb{F}^{m\times m}.$$

Hence, x solves the least-squares problem if and only if

$$Ax = \widehat{Q}\widehat{R}x = \widehat{Q}\widehat{Q}^*b,$$

or, multiplying from the left by $\widehat{Q}^*$,

$$\widehat{R}x = \widehat{Q}^*b.$$

1: **function** REDUCEDQR LS(A)
2: Compute a reduced QR factorization $A = \widehat{Q}\widehat{R}$,
3: Compute $\widehat{Q}^*b$,
4: Solve $\widehat{R}x = \widehat{Q}^*b$ using back-substitution.
5: **end function**

Complexity: The QR factorization may be computed in different ways, resulting in different costs:

- using the Householder method, the QR factorization is obtained in $2mn^2 - 2n^3/3$ flops. In this case there is no need to store $\widehat{Q}$, rather we apply the Householder reflectors to b as the factor R is computed. Then $\widehat{Q}^*b$ is obtained in $O(n^2)$ flops.
- Using the modified Gram-Schmidt method, the cost of computing the factors is $2mn^2$. In this case, $\widehat{Q}^*b$ should be computed by orthogonalizing b with respect to the columns of $\widehat{Q}$ using the modified Gram-Schmidt, rather than by applying $\widehat{Q}^*$.

In both cases, the final triangular solve has negligible cost $O(n^2)$.

14.5 Numerical approach: rank-deficient or underdetermined case.

When $r = \operatorname{rank}(A) < n$, in particular in the case $m < n$, both previous approaches fail. In this case, one may use the SVD decomposition, from which we also extract a reduced or thin variant:

$$A = U\Sigma V^* = \begin{bmatrix}\widehat{U} & \widetilde{U}\end{bmatrix}\begin{bmatrix}\Sigma_r & \\ 0 & 0\end{bmatrix}\begin{bmatrix}\widehat{V}^* \\ \widetilde{V}^*\end{bmatrix} = \widehat{U}\Sigma_r\widehat{V}^*,$$

where $\widehat{U} \in \mathbb{F}^{m\times r}$ and $\widehat{V} \in \mathbb{F}^{n\times r}$, formed by the first r columns of U and V respectively, have orthogonal columns, and $\Sigma_r = \operatorname{diag}(\sigma_1, \dots, \sigma_r)$ is a diagonal matrix with the singular values of A on the diagonal.

Remark 14.6. ***Numerical rank.*** *In practice, due to numerical (floating-point) errors, the rank is an ill-defined quantity. Given singular values $\sigma_1, \dots, \sigma_p$ where $p = \min(m, n)$ and a tolerance $\delta > 0$, one may set a numerical rank $r \geqslant 0$ such that*

$$\sigma_1 \geqslant \cdots \sigma_r > \delta \geqslant \sigma_{r+1} \geqslant \cdots \geqslant \sigma_p \geqslant 0.$$

Now, the orthogonal projector onto the range is given by the expression $P = \widehat{U}\widehat{U}^*$, so

$$Ax = Pb \quad \Leftrightarrow \quad \Sigma_r\widehat{V}^*x = \widehat{U}^*b \in \mathbb{F}^r.$$

This suggests the following algorithm:

1: **function** SVD LS(A)
2: Compute a reduced SVD factorization $A = \widehat{U}\Sigma_r\widehat{V}^*$,
3: Compute $\widehat{U}^*b$,
4: Solve the diagonal system $\Sigma_r y = \widehat{U}^*b$,
5: Compute $x = \widehat{V}y$.
6: **end function**

The result of this computation is

$$x = \widehat{V}\Sigma_r^{-1}\widehat{U}^*b = V\Sigma^\dagger U^* = A^\dagger b,$$

where $A^\dagger$ is the pseudo-inverse of the matrix A.
Indeed, we have two cases:

- **Case 1:** A has full rank n, then $x = V\Sigma_n^{-1}\widehat{U}^*b$. In this case, the computational complexity is higher than for the other methods which are available, mainly due to the high cost of computing an SVD.
- **Case 2:** if r rank $(A) < n$, then $x = \widehat{V}\Sigma_r^{-1}\widehat{U}^*b$. In this case, Ker $(A) \neq \{0\}$, and there is an infinity of solutions:

$$S = \{x_0 + y, \quad x_0 = A^\dagger b, \quad y \in \text{Ker}\,(A)\}.$$

Proposition 14.7. *$x_0 = A^\dagger b$ is the minimum norm solution to the least-squares problem* min $\|Ax - b\|_2$.

Proof. Consider $y = V^*x = \begin{bmatrix} \widehat{V}^*x \\ \widetilde{V}^*x \end{bmatrix} = \begin{bmatrix} y_1 \\ y_2 \end{bmatrix}$. Then, using the unitary invariance of the 2-norm,

$$\begin{aligned} \|r(x)\|_2^2 &= \|UU^*b - U\Sigma y\|_2 \\ &= \|U^*b - \Sigma y\|_2^2 \\ &= \left\| \begin{bmatrix} \widehat{U}^*b - \Sigma_r y_1 \\ \widetilde{U}^*b \end{bmatrix} \right\|_2^2 \qquad = \|\widehat{U}^*b - \Sigma_r y_1\|_2^2 + \|\widetilde{U}^*b\|_2^2. \end{aligned}$$

Hence, we solve the least squares problem if and only if $y_1 = \Sigma_r^{-1}\widehat{U}^*b$, and furthermore

$$\|y\|_2^2 = \|x\|_2^2 = \|y_1\|_2^2 + \|y_2\|_2^2 = \|\Sigma_r^{-1}\widehat{U}^*b\|_2^2 + \|\widetilde{V}^*x\|_2^2.$$

Moreover, $\|x\|_2$ is minimized if and only if $y_2 = \widetilde{V}^*x = 0$, i.e.

$$x = \widehat{V}y_1 = \widehat{V}\Sigma_r^{-1}\widehat{U}^*b = A^\dagger b.$$

□

This shows that, unlike the other two methods, the SVD approach can solve underdetermined least-squares problems, where rank $(A) < n$.

Lecture 15: Stability for least-squares problems

Let us investigate the sensitivity of the solutions to a least-squares problem:

$$\text{Find the minimum } x \in \mathbb{F}^n \text{ to the functional } \|b - Ax\|_2.$$

Let us start by examining variations with respect to the vector b.

Theorem 15.1. *Suppose x minimizes $\|Ax - b\|_2$, and $x + \delta x$ minimizes $\|A(x + \delta x) - (b + \delta b)\|_2$, where $A \in \mathbb{F}^{m\times n}$ with $\operatorname{rank}(A) = n$.*
Then,

$$\frac{\|\delta x\|}{\|x\|} \leqslant K(A)\frac{\|\delta b_R\|}{\|b_R\|} \qquad \text{if } \|b_R\| \neq 0,$$

where $K(A) = \|A\|\|A^\dagger\|$, $b_R = Pb$ and $\delta b_R = P\delta b$ with P the orthogonal projector onto the range $\operatorname{Ran}(A)$.

Remark 15.2. *If we pertub only the vector b, the condition number of the problem is*

$$K(A) = \|A\|\|A^\dagger\|.$$

Remark 15.3. *If $\|b_R\| \approx 0$ (e.g. if b is almost orthogonal to the range of A), a small change in b might have a large effect.*

Proof. The solution is given by

$$x = A^\dagger b, \qquad x + \delta x = A^\dagger(b + \delta b), \quad \text{so} \quad \delta x = A^\dagger \delta b.$$

Recall now that

$$A^\dagger = (A^*A)^{-1}A^*, \qquad \text{so } A^\dagger \delta b = A^\dagger \delta b_R,$$

hence

$$\|\delta x\| \leqslant \|A^\dagger\|\|\delta b_R\|.$$

Furthermore, $Ax = Pb = b_R$, so $\|x\| \geqslant \|b_R\|/\|A\|$. Hence,

$$\frac{\|\delta x\|}{\|x\|} \leqslant \frac{\|A^\dagger\|\|\delta b_R\|}{\|b_R\|/\|A\|}.$$

□

Next, we present a result on the stability of solutions with respect to the coefficients of A.

Theorem 15.4. *Let $A, E \in \mathbb{F}^{m\times n}$ such that $\operatorname{rank}(A + E) = \operatorname{rank}(A) = n$, and*

$$\|E\|_2 \leqslant \varepsilon\|A\|_2.$$

Then, if x minimizes $\|Ax - b\|_2$, with $r = b - Ax$,
and $y = x + \delta x$ minimizes $\|(A + E)y - b\|_2$, with $s = b - (A + E)y$, then

$$\frac{\|\delta x\|_2}{\|x\|_2} \leqslant \frac{K_2(A)\epsilon}{1 - K_2(A)\varepsilon}\left(2 + (K_2(A) + 1)\frac{\|r\|}{\|A\|_2\|x\|_2}\right)$$

We omit the proof.

Remark 15.5. *Understanding the stability of the least-squares problem is tricker than for the linear systems. In general, the solution seems sensitive as the square of the condition number $K_2(A)$.*
On the other hand, if the residual is small ($r = b - Ax = b - b_R$) then $K_2(A)$ serves as the condition number of the problem.

Application: stability of the Householder QR least-squares solution. As we saw earlier, the Householder QR factorization is backwards stable, in the sense that there exists E such that

$$\widehat{Q}\widehat{R} = A + E, \qquad \text{where } \|E\|_F \leqslant g(n)u\|A\|_F,$$

where $g(n)$ is a slowly increasing function of n and u the machine precision. Furthermore, we know that the application of Householder reflectors and the solution of triangular system by backward substitution are backward stable algorithms. This shows that the computed solution $\widehat{x}$ using the Householder QR minimizes exactly an approximate problem of the form

$$\|(A + E)\widehat{x} - (b + \delta b)\|_2,$$

where

$$\|E\|_F \leqslant cun\|A\|_F + O(u^2), \qquad \text{and} \qquad \|\delta b\|_F \leqslant cu\|b\|_2 + O(u^2),$$

with u the machine precision and $c \approx 6m - 3n + 41$.

15.1 Summary: comparison of approaches.

Case 1: $m \geqslant \operatorname{rank}(A) = n$ **(overdetermined problem).**

(a) Normal equations and Cholesky:
Computational cost: $mn^2 + n^3/3$. This is the fastest approach.
Difficulties: stability of the computation of A^*A; sometimes produces more errors than necessary.

(b) Householder-QR:
Computational cost: $2mn^2 - 2n^3/3$.
Backward stable.

(c) Modified Gram-Schmidt QR:
Computational cost: $2mn^2$.
Almost as stable as the Householder QR.

Case 2: $r = \operatorname{rank}(A) < n$ **(underdetermined problem).**

(a) SVD and pseudo-inverse:
Computational cost: $4mn^2 + 8n^3$. This is the slowest approach.
Stable.

(b) Householder QR with column pivoting (aka, rank-revealing QR):
Computational cost: $2mr^2 - r^2(m + n) + 2r^3/3$.
Forward stable, but not backward stable.