NLASP

1. Math

$\pi \approx 3.141\,59 \quad e \approx 2.718\,28 \quad \sqrt{2} \approx 1.414 \quad \sqrt{3} \approx 1.732$

Binome, Trinome

$(a \pm b)^2 = a^2 \pm 2ab + b^2 \qquad a^2 - b^2 = (a-b)(a+b)$

$(a \pm b)^3 = a^3 \pm 3a^2b + 3ab^2 \pm b^3$

$(a+b+c)^2 = a^2 + b^2 + c^2 + 2ab + 2ac + 2bc$

Folgen und Reihen

$\sum_{k=1}^{n} k = \frac{n(n+1)}{2}$ Aritmetrische Summenformel

$\sum_{k=0}^{n} q^k = \frac{1-q^{n+1}}{1-q}$ Geometrische Summenformel

$\sum_{n=0}^{\infty} \frac{\boldsymbol{z}^n}{n!} = e^{\boldsymbol{z}}$ Exponentialreihe

Mittelwerte $(\sum$ von i bis $N)$ (Median: Mitte einer geordneten Liste)

$\overline{x}_{\text{ar}} = \frac{1}{N}\sum x_i \quad \geq \quad \overline{x}_{\text{geo}} = \sqrt[N]{\prod x_i} \quad \geq \quad \overline{x}_{\text{hm}} = \frac{N}{\sum \frac{1}{x_i}}$

Arithmetisches — Geometrisches Mittel — Harmonisches

Ungleichungen: Bernoulli-Ungleichung: $(1+x)^n \geq 1 + nx$

$||x| - |y|| \leq |x \pm y| \leq |x| + |y|$ Dreiecksungleichung

$\left|\boldsymbol{x}^\top \cdot \boldsymbol{y}\right| \leq \|\boldsymbol{x}\| \cdot \|\boldsymbol{y}\|$ Cauchy-Schwarz-Ungleichung

Mengen: De Morgan: $\overline{A \cap B} = \overline{A} \cup \overline{B} \qquad \overline{A \cup B} = \overline{A} \cap \overline{B}$

1.1. Exp. und Log.

$e^x := \lim_{n\to\infty}\left(1 + \frac{x}{n}\right)^n \qquad e \approx 2,71828$

$a^x = e^{x \ln a} \qquad \log_a x = \frac{\ln x}{\ln a} \qquad \ln x \leq x - 1$

$\ln(x^a) = a\ln(x) \qquad \ln(\frac{x}{a}) = \ln x - \ln a \qquad \log(1) = 0$

1.2. Matrizen $\underset{\sim}{A} \in \mathbb{K}^{m\times n}$

$\underset{\sim}{A} = (a_{ij}) \in \mathbb{K}^{m\times n}$ hat m Zeilen (Index i) und n Spalten (Index j)

$(\underset{\sim}{A} + \underset{\sim}{B})^\top = \underset{\sim}{A}^\top + \underset{\sim}{B}^\top \qquad (\underset{\sim}{A} \cdot \underset{\sim}{B})^\top = \underset{\sim}{B}^\top \cdot \underset{\sim}{A}^\top$

$(\underset{\sim}{A}^\top)^{-1} = (\underset{\sim}{A}^{-1})^\top \qquad (\underset{\sim}{A} \cdot \underset{\sim}{B})^{-1} = \underset{\sim}{B}^{-1}\underset{\sim}{A}^{-1}$

$\dim \mathbb{K} = n = \operatorname{rang} \underset{\sim}{A} + \dim \ker \underset{\sim}{A} \qquad \operatorname{rang} \underset{\sim}{A} = \operatorname{rang} \underset{\sim}{A}^\top$

1.2.1. Quadratische Matrizen $A \in \mathbb{K}^{n\times n}$

regulär/invertierbar/nicht-singulär $\Leftrightarrow \det(\underset{\sim}{A}) \neq 0 \Leftrightarrow \operatorname{rang} \underset{\sim}{A} = n$

singulär/nicht-invertierbar $\Leftrightarrow \det(\underset{\sim}{A}) = 0 \Leftrightarrow \operatorname{rang} \underset{\sim}{A} \neq n$

orthogonal $\Leftrightarrow \underset{\sim}{A}^\top = \underset{\sim}{A}^{-1} \Rightarrow \det(\underset{\sim}{A}) = \pm 1$

symmetrisch: $\underset{\sim}{A} = \underset{\sim}{A}^\top$ schiefsymmetrisch: $\underset{\sim}{A} = -\underset{\sim}{A}^\top$

1.2.2. Determinante von $\underset{\sim}{A} \in \mathbb{K}^{n\times n}$: $\det(\underset{\sim}{A}) = |\underset{\sim}{A}|$

$\det\begin{bmatrix} \underset{\sim}{A} & \underset{\sim}{0} \\ \underset{\sim}{C} & \underset{\sim}{D} \end{bmatrix} = \det\begin{bmatrix} \underset{\sim}{A} & \underset{\sim}{B} \\ \underset{\sim}{0} & \underset{\sim}{D} \end{bmatrix} = \det(\underset{\sim}{A})\det(\underset{\sim}{D})$

$\det(\underset{\sim}{A}) = \det(\underset{\sim}{A}^T) \qquad \det(\underset{\sim}{A}^{-1}) = \det(\underset{\sim}{A})^{-1}$

$\det(\underset{\sim}{A}\underset{\sim}{B}) = \det(\underset{\sim}{A})\det(\underset{\sim}{B}) = \det(\underset{\sim}{B})\det(\underset{\sim}{A}) = \det(\underset{\sim}{B}\underset{\sim}{A})$

Hat $\underset{\sim}{A}$ 2 linear abhäng. Zeilen/Spalten $\Rightarrow |\underset{\sim}{A}| = 0$

1.2.3. Eigenwerte (EW) λ und Eigenvektoren (EV) $\underset{\sim}{v}$

$\underset{\sim}{A}\underset{\sim}{v} = \lambda\underset{\sim}{v} \qquad \det \underset{\sim}{A} = \prod \lambda_i \qquad \operatorname{Sp} \underset{\sim}{A} = \sum a_{ii} = \sum \lambda_i$

Eigenwerte: $\det(\underset{\sim}{A} - \lambda\underset{\sim}{1}) = 0$ Eigenvektoren: $\ker(\underset{\sim}{A} - \lambda_i\underset{\sim}{1}) = \underset{\sim}{v}_i$

EW von Dreieck/Diagonal Matrizen sind die Elem. der Hauptdiagonale.

1.2.4. Spezialfall 2×2 Matrix A

$\det(\underset{\sim}{A}) = ad - bc \qquad \begin{bmatrix} a & b \\ c & d \end{bmatrix}^{-1} = \frac{1}{\det \underset{\sim}{A}}\begin{bmatrix} d & -b \\ -c & a \end{bmatrix}$

$\operatorname{Sp}(\underset{\sim}{A}) = a + d$

$\lambda_{1/2} = \frac{\operatorname{Sp}\underset{\sim}{A}}{2} \pm \sqrt{\left(\frac{\operatorname{sp}\underset{\sim}{A}}{2}\right)^2 - \det \underset{\sim}{A}}$

1.2.5. Differentiation

$\frac{\partial \underset{\sim}{x}^\top \underset{\sim}{y}}{\partial \underset{\sim}{x}} = \frac{\partial \underset{\sim}{y}^\top \underset{\sim}{x}}{\partial \underset{\sim}{x}} = \underset{\sim}{y} \qquad \frac{\partial \underset{\sim}{x}^\top \underset{\sim}{A}\underset{\sim}{x}}{\partial \underset{\sim}{x}} = (\underset{\sim}{A} + \underset{\sim}{A}^\top)\underset{\sim}{x}$

$\frac{\partial \underset{\sim}{x}^\top \underset{\sim}{A}\underset{\sim}{y}}{\partial \underset{\sim}{A}} = \underset{\sim}{x}\underset{\sim}{y}^\top \qquad \frac{\partial \det(\underset{\sim}{B}\underset{\sim}{A}\underset{\sim}{C})}{\partial \underset{\sim}{A}} = \det(\underset{\sim}{B}\underset{\sim}{A}\underset{\sim}{C})\left(\underset{\sim}{A}^{-1}\right)^\top$

1.2.6. Ableitungsregeln $(\forall \lambda, \mu \in \mathbb{R})$

Linearität: $(\lambda f + \mu g)'(x) = \lambda f'(x) + \mu g'(x_0)$

Produkt: $(f \cdot g)'(x) = f'(x)g(x) + f(x)g'(x)$

Quotient: $\left(\frac{f}{g}\right)'(x) = \frac{g(x)f'(x) - f(x)g'(x)}{g(x)^2}$ $\left(\frac{\text{NAZ}-\text{ZAN}}{\text{N}^2}\right)$

Kettenregel $(f(g(x)))' = f'(g(x))g'(x)$

1.3. Integrale $\int e^x \mathrm{d}x = e^x = (e^x)'$

Partielle Integration: $\int uw' = uw - \int u'w$

Substitution: $\int f(g(x))g'(x)\,\mathrm{d}x = \int f(t)\,\mathrm{d}t$

$F(x) - C$	$f(x)$	$f'(x)$
$\frac{1}{q+1}x^{q+1}$	x^q	qx^{q-1}
$\frac{2\sqrt{ax^3}}{3}$	$\sqrt{ax}$	$\frac{a}{2\sqrt{ax}}$
$x\ln(ax) - x$	$\ln(ax)$	$\frac{1}{x}$
$\frac{1}{a^2}e^{ax}(ax-1)$	$x \cdot e^{ax}$	$e^{ax}(ax+1)$
$\frac{a^x}{\ln(a)}$	a^x	$a^x\ln(a)$
$-\cos(x)$	$\sin(x)$	$\cos(x)$
$\cosh(x)$	$\sinh(x)$	$\cosh(x)$
$-\ln\lvert\cos(x)\rvert$	$\tan(x)$	$\frac{1}{\cos^2(x)}$

$\int e^{at}\sin(bt)\,\mathrm{d}t = e^{at}\frac{a\sin(bt)+b\cos(bt)}{a^2+b^2}$

$\int \frac{\mathrm{d}t}{\sqrt{at+b}} = \frac{2\sqrt{at+b}}{a} \qquad \int t^2e^{at}\,\mathrm{d}t = \frac{(ax-1)^2+1}{a^3}e^{at}$

$\int te^{at}\,\mathrm{d}t = \frac{at-1}{a^2}e^{at} \qquad \int xe^{ax^2}\,\mathrm{d}x = \frac{1}{2a}e^{ax^2}$

1.3.1. Volumen und Oberfläche von Rotationskörpern um x-Achse

$V = \pi\int_a^b f(x)^2\mathrm{d}x \qquad O = 2\pi\int_a^b f(x)\sqrt{1+f'(x)^2}\mathrm{d}x$

2. Floating-Point Arithmetics

2.1. IEEE 754

Representation of real number $x \in \mathbb{R}$:

$x = (-1)^s(1 + m2^{-t})2^{e-\beta}$

Single Precision	1.8.23
Double Precision	1.11.52
Largest finite number	$(2 - 2^{-t})2^{1-\frac{e_{max}}{2}}$
Smallest non zero Number	$2^{1-\frac{e_{max}-1}{2}}$

Accuracy:

$\frac{|x_2 - x_1|}{|x_1|} \leq 2^{-t}$

Machine Epsilon ϵ_M

$fl(x) = x(1+\epsilon)$ $\qquad fl(x) = \frac{x}{1+\epsilon}$

with different ϵ.

Maximum error is half the precision given by Machine Epsilon: $\epsilon_M = 2^{-t-1}$.

Worst case error

$\rho = r(1+\theta_n)$ with $|\theta_n| \leq \frac{n\epsilon_M}{1-\frac{n}{2}\epsilon_M}$.

The number of operations is given by n.

A more general formulation is:

$\rho = r\Pi_{i=1}^n(1+\epsilon_i)^{\eta_i}$ with $\eta_i \in \{-1;+1\}$ this results in:

$\rho = r(1+\theta_n)$ with $|\theta_n| \leq \gamma_n$ and $\gamma_n = \frac{n\epsilon_M}{1-\frac{n}{2}\epsilon_M}$

2.2. Complex Floating Point Arithmetic

The error in complex floating point operations can also be represented with:

$x \odot y = (x \circ y)(1+\epsilon)$

with $|\epsilon| \leq \alpha\epsilon_M$. The alpha coefficients are given in the table below:

complex floating-point operation	$\oplus$	$\ominus$	$\otimes$	$\oslash$
α	1	1	$2^{3/2}$	6

Table 1.1. Factors for precision of complex floating-point operations

3. Linear Algebra

3.1. SVD

A matrix is unitary if: $\mathbf{U}^H\mathbf{U} = \mathbf{U}\mathbf{U}^H = \mathbf{I}$

The singular decomposition (SVD) of the matrix $\mathbf{A} \in \mathbb{C}^{mxn}$ reads as:

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

$\mathbf{U}$ and $\mathbf{V}$ are unitary and $\mathbf{\Sigma} \in \mathbb{C}^{mxn}$ is diagonal.

$\mathbf{\Sigma}_0 = \operatorname{diag}(\sigma_1, \sigma_2, ..., \sigma_{\min(m,n)})$

$$\mathbf{\Sigma} = \begin{cases} \mathbf{\Sigma}_0, \text{for } m = n \\ [\mathbf{\Sigma}_0, \mathbf{0}], \text{for } m < n \text{ (wide matrix)} \\ \begin{bmatrix} \mathbf{\Sigma}_0, \\ \mathbf{0} \end{bmatrix}, \text{for } m > n \text{ (tall matrix)} \end{cases}$$

The **SVD** can also be represented in the following form:

$\mathbf{A} = \sum_{i=1}^{\min(m,n)} \sigma_i\mathbf{u}_i\mathbf{v}_i^H$

with

$\mathbf{U} = [\mathbf{u}_1, \mathbf{u}_2, ...\mathbf{u}_m] \in \mathbb{C}^{mxm}$

$\mathbf{V} = [\mathbf{v}_1, \mathbf{v}_2, ...\mathbf{v}_n] \in \mathbb{C}^{nxn}$

The rank of $r = rank(\mathbf{A})$ is given by the number of **non zero** singular values $\sigma_i > 0$

$range(\mathbf{A}) = span(\mathbf{u}_1, ...\mathbf{u}_r)$

$null(\mathbf{A}) = span(\mathbf{v}_{r+1}, ...\mathbf{v}_n)$

Removing the singular values that are zero $\sigma_{r+1} = ... = \sigma_n = 0$ the reduced SVD can be expressed with:

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

with

$\mathbf{\Sigma}_{red} = \operatorname{diag}(\sigma_1, ...\sigma_r)$, $\mathbf{U}_{red} = [\mathbf{u}_1, ...\mathbf{u}_r]$ and

$\mathbf{V}_{red} = [\mathbf{v}_1, ...\mathbf{v}_r]$

Gram Matrix

$\mathbf{B} = \mathbf{A}\mathbf{A}^H$ is Hermitian by definition. It's SVD reads as:

$\mathbf{B} = \mathbf{U}_{red}\mathbf{\Sigma}_{red}\mathbf{U}_{red}^H$

A similar discussion is possible for $\mathbf{C} = \mathbf{A}^H\mathbf{A}$:

$\mathbf{C} = \mathbf{A}^H\mathbf{A} = \mathbf{V}_{red}\mathbf{\Sigma}_{red}\mathbf{V}_{red}^H$

Both decompositions are like the (reduced) EVD.

When computing the Gram matrix $\mathbf{A}^H\mathbf{A}$, every element is equal to the inner product of the two corresponding columns of $\mathbf{A}$. Let $\mathbf{a}_i$ denote the i-th column of $\mathbf{A}$. Then, the i,j-element of the Gram matrix can be written as

$$[\mathbf{A}^H\mathbf{A}]_{ij} = \mathbf{a}_i^H\mathbf{a}_j$$

Only the upper triangular part of $\mathbf{A}^H\mathbf{A}$ must be computed due to the Hermitian structure of $\mathbf{A}^H\mathbf{A}$. The lower triangular are simply the complex conjugate of the corresponding elements of the upper triangular.

Operation Count Gram Matrix: #FLOPS $= \mathcal{O}(mn^2)$

3.2. Projectors

Projectors are idempotent matrices. $\mathbf{P} \in \mathbb{C}^{mxm}$ is a projector if:

$$\mathbf{P}^2 = \mathbf{P}$$

Orthogonal projectors are additionally Hermitian. Therefore:

$$\mathbf{P}^2 = \mathbf{P} \text{ and } \mathbf{P}^H = \mathbf{P}$$

is a orthogonal projector.

The orthogonal complement of a projector $\mathbf{P}$ is defined as:

$\mathcal{S}_P^\perp = \{\mathbf{v} : (\mathbf{I} - \mathbf{P})\mathbf{v} = \mathbf{v}\}$

The eigenvalues of a projector are out of the set $\lambda_i \in \{1; 0\}$. And therefore:

$$\mathbf{P} = \mathbf{U}\text{diag}(1, ..., 1, 0, ...0)\mathbf{U}^H$$

with reads in the reduced form:

$$\mathbf{P} = \mathbf{U}_{red}\mathbf{U}_{red}^H = \sum_{i=1}^{r}\mathbf{u}_i\mathbf{u}_i^H$$

with $\mathbf{U}_{red} = [\mathbf{u}_1, ..., \mathbf{u}_r]$. Furthermore it can be concluded that:

$$rank(\mathbf{P}) = tr(\mathbf{P}) = r$$

.

3.2.1. Projectors of vectors

We can construct a projector of a vector $\mathbf{a}$ according to:

$$\mathbf{P}_a = \frac{\mathbf{a}\mathbf{a}^H}{\|\,\mathbf{a}\,\|_2^2}$$

The projector onto the orthogonal complement is given by:

$$\mathbf{P}_\mathbf{a}^\perp = \mathbf{I} - \mathbf{P}_a = \mathbf{I} - \frac{\mathbf{a}\mathbf{a}^H}{\|\,\mathbf{a}\,\|_2^2}$$

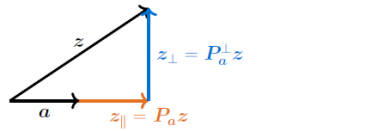

Fig. 2.1. Splitting a vector into a part along a and perpendicular to a

Given orthogonal vectors $\mathbf{a}_1, ...\mathbf{a}_n$ we can perform multiple projections onto the orthogonal complement via:

$$\prod_{i=1}^{n}\mathbf{P}_{\mathbf{a}_i}^\perp = \mathbf{I} - \sum_{i=1}^{n}\frac{\mathbf{a}_i\mathbf{a}_i^H}{\|\,\mathbf{a}_i\,\|_2^2}$$

3.3. Matrix and Vector norms

Vector norms:

p-norm:	$\|\,\mathbf{x}\,\|_p = \sqrt[p]{\sum_{i=1}^n \lvert x_i\rvert^p}$
1-norm:	$\|\,\mathbf{x}\,\|_1 = \sum_{i=1}^n \lvert x_i\rvert$
2-norm:	$\|\,\mathbf{x}\,\|_2 = \sqrt{\sum_{i=1}^n \lvert x_i\rvert^2}$
∞-norm	$\|\,\mathbf{x}\,\|_\infty = \max_{i\in\{1,...,n\}} \lvert x_i\rvert$

Matrix norms:

$\|\,\mathbf{A}\,\| \leq \|\,\mathbf{A}\,\|\|\,\mathbf{B}\,\|$ The columnwise vectorization of $\mathbf{A}$ reads as:

$\text{vec}(\mathbf{A}) = [\mathbf{a}_1^T, ..., \mathbf{a}_n^T]^T \in \mathbb{C}^{mxn}$

$$\|\,\mathbf{A}\,\|_F = \|\,vec(A)\,\|_2 = \sqrt{\sum_{i=1}^{m}\sum_{j=1}^{n}\lvert a_{ij}\rvert^2}$$

3.4. Householder Reflection

With the Householder Reflection we want remove all elements of a vector except the first one. Give a vector $\mathbf{x}$:

$$\mathbf{x} = \begin{bmatrix} x_1 \\ \mathbf{x}_2 \end{bmatrix} \in \mathbb{C}^m$$

With the unitary matrix $\mathbf{F} \in \mathbb{C}^{mxm}$ $(\mathbf{F}^H\mathbf{F} = \mathbf{F}\mathbf{F}^H = \mathbf{I})$we want to achieve the following:

$$\mathbf{F}\mathbf{x} = \begin{bmatrix} \alpha \\ 0 \\ . \\ . \\ . \\ 0 \end{bmatrix} = \alpha\mathbf{e}_1$$

It follows: $\| \mathbf{F}\mathbf{x} \|_2^2 = \mathbf{x}^H\mathbf{F}^H\mathbf{F}\mathbf{x} = \| \mathbf{x} \|_2^2$ and therefore:
$|\alpha| = \| \mathbf{x} \|_2^2$
The overall goal is to find the matrix $\mathbf{F}$ that achieves:

$$\mathbf{F}\mathbf{x} = \begin{bmatrix} \| \mathbf{x} \|_2 \, e^{j\phi} \\ 0 \\ . \\ . \\ . \\ 0 \end{bmatrix} = \| \mathbf{x} \|_2 \, e^{j\phi}\mathbf{e}_1$$

We want to find the vector

$$\mathbf{v} = \| \mathbf{x} \|_2 \, e^{j\phi}\mathbf{e}_1 - \mathbf{x}$$

The vectors $\mathbf{x}$, $\mathbf{v}$ and $\| \mathbf{x} \|_2 \, e^{j\phi}\mathbf{e}_1$ form an isosceles triangle. The height vector is given by the projection onto the orthogonal complement of $\mathbf{v}$:

$$\mathbf{h} = \mathbf{P}_\mathbf{v}^\perp\mathbf{x} = (\mathbf{I} - \mathbf{P}_\mathbf{v})\mathbf{x} = \mathbf{x} - \frac{\mathbf{v}\mathbf{v}^H}{\|\mathbf{v}\|_2^2}\mathbf{x}$$

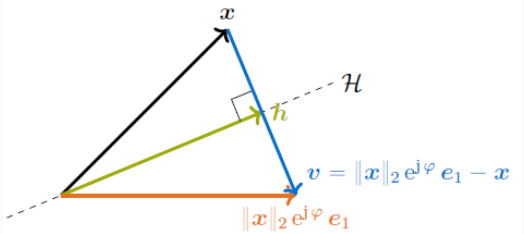

Fig. 2.2. Householder reflection

The matrix $\mathbf{F}$ is given by:

$$\mathbf{F} = \mathbf{I} - 2\frac{\mathbf{v}\mathbf{v}^H}{\| \mathbf{v} \|_2^2}$$

Decomposing the Householder reflector into it's EVD leads to:

$$[\mathbf{w}, \mathbf{W}_\perp]\text{diag}(-1, 1, ..., 1)[\mathbf{w}, \mathbf{W}_\perp]^H$$

where $\mathbf{w} = \frac{\mathbf{v}}{\|\mathbf{v}\|_2}$ and $\mathbf{P}_\mathbf{v}^\perp = \mathbf{W}_perp\mathbf{W}_\perp^H$
Only $\lambda_1 = -1$ of $\mathbf{F}$ all other eigenvalues are $\lambda_i = 1$.

3.4.1. Choice of Phase after Reflection

A possible choice is: $\phi = 0$ such that α is positive-real. An other goal is to make $\mathbf{v}$ as long as possible therefore:

$$\phi_{opt} = \text{argmax}_\phi = \| \mathbf{v} \|_2^2$$

with $\mathbf{x} = [x_1, \mathbf{x}_2^T]^T$ and
$\mathbf{v} = \| \mathbf{x} \|_2 \, e^{j\phi}\mathbf{e}_1 - \mathbf{x}$ the objective function can be expressed as

$$\| \mathbf{v} \|_2^2 = |e^{j\phi} \| \mathbf{x} \|_2 - x_1|^2 + \| \mathbf{x}_2 \|_2^2$$

As the second term is independent of ϕ, the first term must be maximized. To this end, it is necessary to choose $\phi_{opt} = \psi + \pi$ for $x_1 = |x_1|e^{j\psi}$, resulting in

$$\mathbf{v} = \begin{bmatrix} -(\| \mathbf{x} \|_2 + |x_1|)e^{j\psi} \\ -\mathbf{x}_2 \end{bmatrix}$$

The maximum objective function is:

$$\| \mathbf{v}_{opt} \|_2^2 = 2 \| \mathbf{x}_2 \|_2^2 + 2|x_1| \| \mathbf{x} \|_2$$

4. QR Decomposition

For a full-rank and tall matrix $\mathbf{A} = [\mathbf{a}_1, ..., \mathbf{a}_n] \in \mathbb{C}^{mxn}$ with $m \gg n$ the QR decomposition reads as:

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

where $\mathbf{Q} \in \mathbb{C}^{mxm}$ is unitary and

$$\mathbf{R} = \begin{bmatrix} \mathbf{R}_{red} \\ \mathbf{0} \end{bmatrix} \in \mathbb{C}^{mxn}$$

with

$$\mathbf{R}_{red} = \begin{bmatrix} r_{11} & r_{12} & \cdots & r_{1n} \\ 0 & r_{22} & \cdots & r_{2n} \\ & & \ddots & \vdots \\ 0 & 0 & & \\ 0 & \cdots & r_{nn} & \end{bmatrix} \in \mathbb{C}^{nxn}$$

with $(\nu \leq n)$:
$span(\mathbf{a}_1, ..., \mathbf{a}_\nu) = span(\mathbf{q}_1, ..., \mathbf{q}_\nu)$
which results in:
$range(\mathbf{A}) = span(\mathbf{a}_1, ..., \mathbf{a}_n) = span(\mathbf{q}_1, ..., \mathbf{q}_n) = range(\mathbf{Q}_{red})$
$\mathbf{Q}_{red}$ is an orthonormal basis for the range space of $\mathbf{A}$.

$\mathbf{Q}_{red} = [\mathbf{q}_1, ..., \mathbf{q}_n] \in \mathbb{C}^{mxn}$
$\mathbf{Q}_{red}^\perp = [\mathbf{q}_{n+1}, ..., \mathbf{q}_m] \in \mathbb{C}^{mxm-n}$
It follows from $\mathbf{Q} = [\mathbf{Q}_{red}, \mathbf{Q}_{red}^\perp]$ that

$$\mathbf{I} = \mathbf{Q}^H\mathbf{Q} = \begin{bmatrix} \mathbf{Q}_{red}^H \\ \mathbf{Q}_{red}^{\perp,H} \end{bmatrix} \cdot [\mathbf{Q}_{red}, \mathbf{Q}_{red}^\perp] = \begin{bmatrix} \mathbf{Q}_{red}^H\mathbf{Q}_{red} & \mathbf{Q}_{red}^H\mathbf{Q}_{red}^\perp \\ \mathbf{Q}_{red}^{\perp,H}\mathbf{Q}_{red} & \mathbf{Q}_{red}^{\perp,H}\mathbf{Q}_{red}^\perp \end{bmatrix}$$

Therefor: $\mathbf{Q}_{red}^H\mathbf{Q}_{red}^\perp = 0$. Consequently the space spanned by the columns of $\mathbf{Q}_{red}^\perp$ is perpendicular to $range(\mathbf{Q}_{red})$. In other words:

$$null(\mathbf{A}^H) = range(\mathbf{Q}_{red}^\perp)$$

that is, the matrix $\mathbf{Q}_{red}^\perp$ is a basis for the left nullspace of $\mathbf{A}$.

4.1. Operation Count of basic operations

f	Mult.+Sum.	#FLOPS
$\mathbf{a}^H\mathbf{b} = \sum_{i=1}^m a_i^* b_i$	$m+m-1$	$2m-1$
$\| \mathbf{a} \|_2 = \sqrt{\sum_{i=1}^m a_i^* a_i}$	$m+m-1+1$	$2m$
$\gamma\mathbf{a}$	m	m
$\mathbf{a} \pm \mathbf{b}$	m	m

4.2. Classical Gram-Schmidt Orthogonalization

Algorithm 1 Classical Gram-Schmidt Orthogonalization

```
function CGS(A)                                   ▷ with A = [a_1, ..., a_n] ∈ C^{m×n} with m ⩾ n
    for j = 1 to n do
        v_j ← a_j                                 ▷ orthogonalize the j-th column
        for i = 1 to j − 1 do
            r_ij ← q_i^H a_j                      ▷ coefficient of projection onto q_i
            v_j ← v_j − r_ij q_i                  ▷ remove component along q_i
        end for
        r_jj ← ‖v_j‖_2
        q_j ← v_j / r_jj                          ▷ normalization
    end for
    return Q_red, R_red
end function
```

4.3. Modified Gram-Schmidt Orthogonalization

Algorithm 2 Modified Gram-Schmidt Orthogonalization

```
function MGS1(A)                                  ▷ with A = [a_1, ..., a_n] ∈ C^{m×n} with m ⩾ n
    for j = 1 to n do
        v_j ← a_j
        for i = 1 to j − 1 do
            r_ij ← q_i^H v_j                      ▷ from CGS to MGS: change a_j to v_j
            v_j ← v_j − r_ij q_i
        end for
        r_jj ← ‖v_j‖_2
        q_j ← v_j / r_jj
    end for
    return Q_red, R_red
end function
```

Algorithm 3 Modified Gram-Schmidt Orthogonalization

```
function MGS2(A)                                  ▷ A = [a_1, ..., a_n]
    for j = 1 to n do
        v_j ← a_j                                 ▷ store columns of A in v_1, ..., v_n
    end for
    for i = 1 to n do
        r_ii ← ‖v_i‖_2
        q_i ← v_i / r_ii                          ▷ normalization
        for j = i + 1 to n do
            r_ij ← q_i^H v_j                      ▷ coefficient of projection onto q_i
            v_j ← v_j − r_ij q_i                  ▷ subtraction of component along q_i
        end for
    end for
    return Q_red, R_red
end function
```

Operation Count: $\#\text{FLOPS} = \mathcal{O}(2mn^2)$

4.4. Householder QR Decomposition

With the Householder reflector $\mathbf{Q}_1 = \mathbf{F}_1$ the first column of matrix $\mathbf{A}$ is transformed as follows

$$\mathbf{Q}_1\mathbf{A} = \begin{bmatrix} \times & \times & \times \\ 0 & \times & \times \\ 0 & \times & \times \\ 0 & \times & \times \\ 0 & \times & \times \end{bmatrix}$$

In the next step we apply the matrix $\mathbf{Q}_2$ which inherits a Householder reflector $\mathbf{F}_2 \in \mathbb{C}^{m-1 \times m-1}$

$$\mathbf{Q}_2 = \begin{bmatrix} 1 & \mathbf{0}^T \\ \mathbf{0} & \mathbf{F}_2 \end{bmatrix} \in \mathbb{C}^{m \times m}$$

After that the matrix $\mathbf{Q}_2\mathbf{Q}_1\mathbf{A}$ looks like this

$$\mathbf{Q}_2\mathbf{Q}_1\mathbf{A} = \begin{bmatrix} \times & \times & \times \\ 0 & \times & \times \\ 0 & \times & \times \\ 0 & \times & \times \\ 0 & \times & \times \end{bmatrix}$$

The unitary rotation matrix $\mathbf{Q}$ at step $i \in [1, ..., n]$ is given by

$$\mathbf{Q}_i = \begin{bmatrix} \mathbf{I}_{i-1} & \mathbf{0} \\ \mathbf{0} & \mathbf{F}_i \end{bmatrix} \in \mathbb{C}^{m \times m}$$

with $\mathbf{F}_i = \mathbf{I} - 2\frac{\mathbf{v_i}\mathbf{v_i}^H}{\|\mathbf{v}\|_2^2} \in \mathbb{C}^{m-i+1 \times m-i+1}$

After rotating all n columns of $\mathbf{A}$ we generate the QR decomposition with
$\mathbf{A} = \mathbf{Q}_1^H\mathbf{Q}_2^H \cdots \mathbf{Q}_n^H\mathbf{R} = \mathbf{Q}\mathbf{R}$

$\mathbf{R} \in \mathbb{C}^{mxn}$
Furthermore it holds
$\mathbf{Q} = \mathbf{Q}_1^H\mathbf{Q}_2^H \cdots \mathbf{Q}_n^H = \mathbf{Q}_1\mathbf{Q}_2 \cdots \mathbf{Q}_n$ since $\mathbf{Q}_i$ is Hermitian.

Algorithm 4 Householder QR Decomposition

```
function HouseholderQR(A)                         ▷ A = [a_1, ..., a_n] ∈ C^{m×n} with m ⩾ n
    for i = 1 to n do
        x ← [A]_{i:m,i}                           ▷ take entries i to m of the i-th column
        [A]_{i:m,i} ← exp(j(arg(x_1) + π))‖x‖_2 e_1   ▷ result of reflection for i-th column
        v_i ← [A]_{i:m,i} − x
        v_i ← v_i / ‖v_i‖_2                       ▷ normalization
        [A]_{i:m,i+1:n} ← [A]_{i:m,i+1:n} − 2v_i(v_i^H [A]_{i:m,i+1:n})   ▷ no reflection for i = n
    end for
    return A, v_1, ..., v_n
end function
```

Operation Count Algorithm 4: $\#\text{FLOPS} = \mathcal{O}(2mn^2 - \frac{2}{3}n^3)$

Algorithm 5 Householder QR Decomposition: Computation of Q_{red}

```
function HouseholderQ(v_1, ..., v_n)              ▷ v_i defines the rotation Q_i
    Q ← [I_n; 0]                                  ▷ use first n columns of I as initialization
    for i = n downto 1 do
        [Q]_{i:m,i:n} ← [Q]_{i:m,i:n} − 2v_i(v_i^H [Q]_{i:m,i:n})   ▷ reflection based on v_i
    end for
    return Q                                      ▷ return Q_red
end function
```

Algorithm 6 Householder QR Decomposition: Computation of $Q^H b$

```
function HouseholderQb(v_1, ..., v_n, b)          ▷ v_i defines the rotation Q_i
    for i = 1 to n do
        [b]_{i:m} ← [b]_{i:m} − 2v_i(v_i^H [b]_{i:m})   ▷ reflection with normalized v_i
    end for
    return b
end function
```

Operation Count Algorithm 5: $\#\text{FLOPS} = \mathcal{O}(2mn^2 - \frac{2}{3}n^3)$

Operation Count Algorithm 6: $\#\text{FLOPS} = \mathcal{O}(4mn - 2n^2)$

Algorithm 7 Householder QR Decomposition: Computation of $Q_{\text{red}}^\perp$

```
function HouseholderQperp(v_1, ..., v_n)          ▷ v_i defines the rotation Q_i
    Q ← [0; I_{m−n}]                              ▷ use last m − n columns of I as initialization
    for i = n downto 1 do
        [Q]_{i:m,1:m−n} ← [Q]_{i:m,1:m−n} − 2v_i(v_i^H [Q]_{i:m,1:m−n})   ▷ reflection based on v_i
    end for
    return Q                                      ▷ return Q_red^⊥
end function
```

Operation Count Algorithm 7: $\#\text{FLOPS} = \mathcal{O}(4m^2n + 2n^3 - 6mn^2)$
If $m = n$ the number of FLOPS for Algorithm 7 is zero, because the partition $\mathbf{Q}_red^\perp$ does not exist in this case.

4.4.1. Householder for null($\mathbf{A}^H$)

In the case, that a basis for the nullspace $\mathbf{A}^H$, i.e., the orthogonal complement of $range(A)$ must be found Algorithm 7 can be employed.
null($\mathbf{A}^H$) =range($\mathbf{Q}_{red}^H$)

$$\mathbf{Q} = [\mathbf{Q}_{red}, \mathbf{Q}_{red}^\perp]$$

5. Back Substitution

Solving an equation system with an upper triangular $\mathbf{R}$
$\mathbf{Rx} = \mathbf{b}$
can be solved with Algorithm 8.

$$\mathbf{R} = \begin{bmatrix} r_{11} & r_{12} & \cdots & r_{1n} \\ 0 & r_{22} & & \\ & \ddots & & \vdots \\ & & & r_{nn} \end{bmatrix} \in \mathbb{C}^{n \times n}$$

Algorithm 8 Back Substitution: Computation of the solution to $Rx = b$

```
function BackSubstitution(R, b)                    ▷ R ∈ C^{n×n} is upper triangular
    for i = n downto 1 do
        x_i ← (1/r_ii) (b_i − Σ_{j=i+1}^{n} r_ij x_j)        ▷ for i = n, the summation is zero
    end for
    return x
end function
```

Operation Count Algorithm 8: #FLOPS $= \mathcal{O}(n^2)$

Algorithm 9 Back Substitution: Computation of R^{-1}

```
function BackSubstitutionInverse(R)                ▷ R ∈ C^{n×n} is upper triangular
    P ← I                                          ▷ apply back substitution to columns of the identity matrix
    for j = 1 to n do
        for i = j downto 1 do
            p_ij ← (1/r_ii) (p_ij − Σ_{l=i+1}^{j} r_il p_lj)        ▷ for i = n, the summation is zero
        end for
    end for
    return P
end function
```

Operation Count Algorithm 9: #FLOPS $= \mathcal{O}(\frac{1}{3}n^3)$

#FLOPS$= \sum_{j=1}^{n}\sum_{i=1}^{j}(2j - 2i + 1)$
$= \sum_{j=1}^{n}[2j^2 - j(j+1) + j] = \frac{(2n+1)(n+1)n}{6}$

6. Least Squares

6.1. Least Square for Full-Rank Matrix

$\mathbf{Ax} = \mathbf{b}$ with $\mathbf{A} \in \mathbb{C}^{m \times n}$ and $m \geq n$. For tall $\mathbf{A} \in \mathbb{C}^{m \times n}$ with $m >$ no solution exists in general, except $b \in$range$(\mathbf{A})$.
For a square and invertible $\mathbf{A}$ the solution
The Least Square solution is given by the normal equation:

$$\mathbf{A}^H\mathbf{A}\mathbf{x}_{\mathsf{LS}} = \mathbf{A}^H\mathbf{b}$$

$$\mathbf{x}_{\mathsf{LS}} = (\mathbf{A}^H\mathbf{A})^{-1}\mathbf{A}^H\mathbf{b}$$

The pseudoinverse of $\mathbf{A}$: $\mathbf{A}^\dagger = (\mathbf{A}^H\mathbf{A})^{-1}\mathbf{A}^H\mathbf{b}$

6.2. Least Square with Cholesky

We need to compute the Gram matrix: $\mathbf{A}^\mathsf{H}\mathbf{A} = \mathbf{LDL}^\mathsf{H}$.

$$\mathbf{LDL}^\mathsf{H}\mathbf{x}_{\mathsf{LS}} = \mathbf{A}^\mathsf{H}\mathbf{b}$$

Due to $\mathbf{L}$ being lower triangular we need two times backsubstitution.
The condition number with Cholesky increases because if we compute the Gram matrix it holds:

$$\kappa(\mathbf{A}^H\mathbf{A}) = \kappa^2(\mathbf{A})$$

The condition number of the Gram matrix of $\mathbf{A}$ is the square of the original condition number κ.

6.3. Least Square with QR

Using the decomposition $\mathbf{A} = \mathbf{Q}_{\mathsf{red}}\mathbf{R}_{\mathsf{red}}$ We obtain the solution

$$\mathbf{A}^\dagger = \mathbf{R}_{\mathsf{red}}^{-1}\mathbf{Q}_{\mathsf{red}}^\mathsf{H}$$

$$\mathbf{x}_{\mathsf{LS}} = \mathbf{A}^\dagger\mathbf{b}$$

Where $\mathbf{R}_{\mathsf{red}}^{-1}$ describes a backsubstitution. Therefore we need to compute $\mathbf{Q}^\mathsf{H}\mathbf{b}$ and then employ a backsubstitution on this result. QR decomposition and Cholesky have the same order of complexity if $m = n$.

Algorithm 10 Least Squares via Cholesky Factorization

```
function LSCholesky(A, b)                          ▷ constituting matrix A ∈ C^{m×n} and right-hand side b
    B ← A^H A                                      ▷ compute Gram of A
    c ← A^H b                                      ▷ right-hand side of normal equations
    L, D ← Cholesky factorization of B
    v ← L^{-1} c                                   ▷ with back substitution
    w ← D^{-1} v                                   ▷ scaling every element
    x_LS ← L^{H,-1} w                              ▷ with back substitution
end function
```

Operation Count Algorithm 10: #FLOPS $= \mathcal{O}(mn^2 + \frac{1}{3}n^3)$
Computing the Gram matrix and Cholesky factorization is dominant. In total we have: $\mathcal{O}(mn^2 + 2mn + \frac{1}{3}n^3 + 2n^2 + n)$

Algorithm 11 Least Squares via QR Decomposition

```
function LSQR(A, b)                                ▷ constituting matrix A ∈ C^{m×n} and right-hand side b
    Q, R ← QR decomposition of A                   ▷ preferably Householder QR
    c ← Q_red^H b
    x_LS ← R_red^{-1} c                            ▷ with back substitution
end function
```

Operation Count Algorithm 11: #FLOPS $= \mathcal{O}(2mn^2 - \frac{2}{3}n^3)$
The complexity analysis assumes that we used Householder QR decomposition.

6.4. Least Squares with SVD

We decompose $\mathbf{A} \in \mathbb{C}^{m \times n}$ as follows

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

with sub-unitary $\mathbf{U}_{\mathsf{red}} \in \mathbb{C}^{m \times n}$, diagonal $\mathbf{\Sigma}_{\mathsf{red}} \in \mathbb{C}^{n \times n}$ and unitary $\mathbf{V} \in \mathbb{C}^{n \times n}$. The pseudoinverse of $\mathbf{A}$ can be written as

$$\mathbf{A}^\dagger = \mathbf{V}\mathbf{\Sigma}_{\mathsf{red}}^{-1}\mathbf{U}_{\mathsf{red}}^\mathsf{H}$$

Thus the solution can be expressed as

$$\mathbf{x}_{\mathsf{LS}} = \mathbf{V}\mathbf{\Sigma}_{\mathsf{red}}^{-1}\mathbf{U}_{\mathsf{red}}^\mathsf{H}\mathbf{b}$$

Algorithm 12 Least Squares via SVD

```
function LSSVD(A, b)                               ▷ constituting matrix A ∈ C^{m×n} and right-hand side b
    U_red, Σ_red, V ← singular value decomposition of A
    c ← U_red^H b
    w ← Σ_red^{-1} c
    x_LS ← V w
end function
```

Operation Count Algorithm 12: #FLOPS=$\mathcal{O}(2mn^2 + 11n^3)$

6.5. Comparsion of Different Approaches

$$\#\mathsf{FLOPS}_{\mathsf{Cholesky}} \leq \#\mathsf{FLOPS}_{\mathsf{QR\text{-}Householder}} \leq \#\mathsf{FLOPS}_{\mathsf{SVD}}$$

The left inequality becomes and equality for $m = n$ and the right inequality is approximately an equality for $m \gg n$.
For numerical stability, the order is the same

$$\mathsf{stability}_{\mathsf{Cholesky}} \leq \mathsf{stability}_{\mathsf{QR\text{-}Householder}} \leq \mathsf{stability}_{\mathsf{SVD}}$$

Due to the **explicit computation of the Gram matrix** $\mathbf{A}^\mathsf{H}\mathbf{A}$ in the Cholesky factorization based approach the condition of the problem is deteriorated considerably.
QR decomposition does no perform this step. However, the **back substitution** is disadvantageous compared to the **simple scaling** and the multiplication by the unitary $\mathbf{V}$ in the SVD approach.

6.6. Rank-Deficient Least Square

The SVD of rank-deficient A is given by

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

with $\mathbf{U}_{\mathsf{red}} \in \mathbb{C}^{m \times r}$, $\mathbf{\Sigma}_{\mathsf{red}} \in \mathbb{C}^{r \times r}$ which is diagonal with positive diagonal elements, sub unitary $\mathbf{V}_{\mathsf{red}} \in \mathbb{C}^{n \times r}$ and $r =$rank$(\mathbf{A})$.

$$\mathbf{\Sigma}_{\mathsf{red}}^2\mathbf{V}_{\mathsf{red}}^\mathsf{H}\mathbf{x} = \mathbf{\Sigma}_{\mathsf{red}}\mathbf{U}_{\mathsf{red}}^\mathsf{H}$$

The equation is underdetermined with more unknowns n than equations r. There fore infinitely many solutions for $\mathbf{x}$ exist. We rewrite the equation with

$$\mathbf{x}_{\mathsf{LS}} = \mathrm{argmin}_{\mathbf{x}} \parallel \mathbf{x} \parallel_2^2 \quad \text{s.t.} \mathbf{\Sigma}_{\mathsf{red}}\mathbf{V}_{\mathsf{red}}^H\mathbf{x} = \mathbf{U}_{\mathsf{red}}^\mathsf{H}\mathbf{b}$$

For rank-deficient LS we obtain the solution:

$$\mathbf{x}_{\mathsf{LS}} = \mathbf{V}_{\mathsf{red}}\mathbf{\Sigma}_{\mathsf{red}}^{-1}\mathbf{U}_{\mathsf{red}}\mathbf{b}$$

The residual is given by

$$\mathbf{r} = (\mathbf{I} - \mathbf{U}_{\mathsf{red}}\mathbf{U}_{\mathsf{red}}^\mathsf{H})\mathbf{b}$$

that is, the residual of the rank-deficient LS problem is equal to the part of the right-hand side $\mathbf{b}$ lying in the orthogonal complment of range$(\mathbf{U}_{\mathsf{red}})$ = range$(\mathbf{A})$.

6.7. Moore-Penrose Pseudoinverse

The generalization for all LS problems is given by the Moore-Penrose pseudoinverse with $\mathbf{\Sigma}^\dagger = [\Sigma_1^\dagger, \mathbf{0}]$ we can rewrite $\mathbf{A}^\dagger$ as

$$\mathbf{A}^\dagger = \mathbf{V}[\mathrm{diag}(\frac{1}{\sigma_1}, \cdots, \frac{1}{\sigma_r}, 0, \cdots, 0), \mathbf{0}]\mathbf{U}^\mathsf{H}$$

When $\mathbf{A}$ is full-rank and square $\mathbf{A}^\dagger = \mathbf{A}^{-1}$
For full-rank and tall $\mathbf{A} \in \mathbb{C}^{m \times n}$, $m > n$: $\mathbf{A}^\dagger = (\mathbf{A}^\mathsf{H}\mathbf{A})^{-1}\mathbf{A}^\mathsf{H}$
For full-rank and wide matrix $\mathbf{A} \in \mathbb{C}^{m \times n}$, $m < n$:
$\mathbf{A}^\dagger = \mathbf{A}^H(\mathbf{A}\mathbf{A}^\mathsf{H})^{-1}$

6.8. Rank-Revealing QR Decomposition

The application of the rank-revealing QR decomposition of $\mathbf{A} \in \mathbb{C}^{m \times n}$ with $m \geq n$ gives

$$\mathbf{A}\mathbf{\Pi} = \mathbf{Q}_{\mathsf{red}}\mathbf{R}_{\mathsf{red}}$$

where $\mathbf{Q}_{\mathsf{red}} \in \mathbb{C}^{m \times n}$ is sub unitary and

$$\mathbf{R}_{\mathsf{red}} = \begin{bmatrix} \mathbf{R}_{11} & \mathbf{R}_{12} \\ \mathbf{0} & \mathbf{0} \end{bmatrix} \in \mathbb{C}^{n \times n}$$

The partition $\mathbf{R}_{11} \in \mathbb{C}^{r \times r}$ is upper-triangular and full-rank with r=rank$(\mathbf{A})$
In every step the **column** whose lower part, that has not yet been brought to triangular form and **exhibits maximum norm** in the current step is **permuted** to the first column of the part that has not yet been transformed to triangular form.

7. Condition of a Problem

7.1. Norm-Wise Condition Number

In abstract form, when solving a problem e.g., solving an equation system, the following map must be applied

$$\mathbf{f} : \mathbb{C}^m \to \mathbb{C}^n; \mathbf{x} \to \mathbf{y} = \mathbf{f}(\mathbf{x})$$

A prominent example for such a problem is solving an equation system of the form

$$\mathbf{Av} = \mathbf{b}$$

Interpreting the matrix $\mathbf{A}$ constituting the equation system as parameter of the problem, the problem to be solved is:

$$\mathbf{f} : \mathbb{C}^m \to \mathbb{C}^n; \mathbf{b} \to \mathbf{v} = \mathbf{f}(\mathbf{b})$$

Note that a different $\mathbf{A}_2$ leads to a different problem $\mathbf{f}_2$.

The norm-wise condition number is given by:

$$\kappa_{\mathsf{N}}(\mathbf{f}, \mathbf{x}) = \frac{\parallel \mathbf{J}(\mathbf{x}) \parallel \cdot \parallel \mathbf{x} \parallel}{\parallel \mathbf{f}(\mathbf{x}) \parallel}$$

where

$$\mathbf{J}(\mathbf{x}) = \frac{\partial \mathbf{f}(\mathbf{x})}{\partial \mathbf{x}^T}$$

The perturbed outcome of the problem $\mathbf{f}$ can be bouned as follows

$$\frac{\parallel \mathbf{f}(\mathbf{x} + \Delta\mathbf{x}) - \mathbf{f}(\mathbf{x}) \parallel}{\parallel \mathbf{f}(\mathbf{x}) \parallel} \leq \kappa_{\mathsf{N}}(\mathbf{f}, \mathbf{x}) \frac{\parallel \Delta\mathbf{x} \parallel}{\parallel \mathbf{x} \parallel} + o(\parallel \Delta\mathbf{x} \parallel)$$

- **well-conditioned**: If $\kappa_{\mathsf{N}}(\mathbf{f}, \mathbf{x}) \approx 1$, (e.g., 1,100). For a well-conditioned problem, it is possible fo find algorithms with small errors.
- **ill-conditioned**: If $\kappa_{\mathsf{N}}(\mathbf{f}, \mathbf{x}) \gg 1$, (e.g., 10^6, 10^10). When the problem is ill-conditioned, large errors must be expected from all algorithms.
- **ill-posed**: If $\kappa_{\mathsf{N}}(\mathbf{f}, \mathbf{x}) = \infty$. In this case, the solution $\mathbf{f}(\mathbf{x})$ either does not exist or is not unique. It can be expected that no algorithm is able so give a reasonable result.

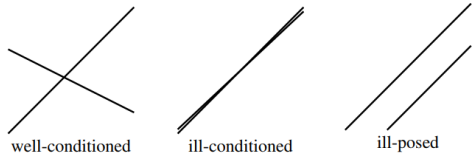

Fig. 6.3. Condition of intersecting two straight lines

$f(x)$	$J(x)$	κ_{N}
$f(x) = \sqrt{x}$	$J(x) = \frac{d\sqrt{x}}{dx} = \frac{1}{2}\frac{1}{\sqrt{x}}$	$\kappa_{\mathsf{N}} = \frac{1}{2}$
$f(x) = ax$	$J(x) = a$	$\kappa_{\mathsf{N}} = \frac{\lvert a \rvert \lvert x \rvert}{\lvert ax \rvert} = 1$
$f(\begin{bmatrix} x_1 \\ x_2 \end{bmatrix}) = x_1 \cdot x_2$	$J(x) = [x_2, x_1]$	$\kappa_{\mathsf{N}} = \lvert \frac{x_1}{x_2} \rvert + \lvert \frac{x_2}{x_1} \rvert$

7.2. Component-wise Condition Number

The component-wise condition number is defined by

$$\kappa_{\mathsf{C}}(f, \mathbf{x}) = \frac{\parallel \mathbf{J}(\mathbf{x}) \odot \mathbf{x}^\mathsf{T} \parallel_\infty}{\lvert f(\mathbf{x}) \rvert}$$

$$= \frac{\lvert \mathbf{J}^\mathsf{T}(\mathbf{x}) \rvert \lvert \mathbf{x} \rvert}{\lvert f(\mathbf{x}) \rvert}$$

the product $\lvert \mathbf{J}^\mathsf{T}(\mathbf{x}) \rvert \lvert \mathbf{x} \rvert$ is performed element wise.

The component-wise condition number is smaller than the norm-wise condition number if the infinity norm is employed. For $f : \mathbb{R}^m \to \mathbb{R}$ and when applying the infinity norm, the norm wise condition number is given by

$$\kappa_{\mathsf{N}}(\mathbf{f}, \mathbf{x}) = \frac{\parallel \mathbf{J}(\mathbf{x}) \parallel_\infty \cdot \parallel \mathbf{x} \parallel_\infty}{\lvert \mathbf{f}(\mathbf{x}) \rvert}$$

as $f(\mathbf{x})$ is assumed to be scalar. Because $\mathbf{J}^\mathsf{T}(\mathbf{x}) \in \mathbb{R}^{1 \times m}$ is a row vector for scalar $f(\mathbf{x})$, $\parallel \mathbf{J}^\mathsf{T}(\mathbf{x}) \parallel_\infty = \parallel \mathbf{J}(\mathbf{x}) \parallel_1 = \sum_{i=1}^{m} \lvert [\mathbf{J}(\mathbf{x})]_i \rvert$ holds, and due to $\parallel \mathbf{x} \parallel_\infty = \max_{i \in \{1, \cdots, m\}} \lvert x_i \rvert$

$$\kappa_{\mathsf{N}}(f, \mathbf{x}) \geq \frac{\lvert \mathbf{J}^\mathsf{T}(\mathbf{x}) \rvert \cdot \lvert \mathbf{x} \rvert}{\lvert f(\mathbf{x}) \rvert} = \kappa_{\mathsf{C}}(f, \mathbf{x})$$

f	$x_1 + x_2$	$x_1 - x_2$	$x_1 x_2$	$\frac{x_1}{x_2}$	$\frac{1}{x_1}$	$\sqrt{x_1}$
$\kappa_C(f;\boldsymbol{x})$	$\frac{\lvert x_1\rvert+\lvert x_2\rvert}{\lvert x_1+x_2\rvert}$	$\frac{\lvert x_1\rvert+\lvert x_2\rvert}{\lvert x_1-x_2\rvert}$	2	2	1	$\frac{1}{2}$

Table 6.1. Element-wise condition number of basic arithmetic operations

7.3. Condition of a Matrix

We can express the condition number of matrix $\mathbf{A} \in \mathbb{C}^{m\times n}$ with the help of the pseudoinverse $\mathbf{A}^\dagger$ as

$$\kappa(\mathbf{A}) = \| \mathbf{A} \| \cdot \| \mathbf{A}^\dagger \|$$

When we employ the 2-norm we get the result

$$\kappa_2(\mathbf{A}) = \| \mathbf{A} \| \cdot \| \mathbf{A}^\dagger \| = \frac{\sigma_1}{\sigma_n}$$

Wide Matrix For a wide matrix we divide the vector $\mathbf{x}$ into a component $\mathbf{x}_\mathbf{A}^\perp$ in the nullspace of $\mathbf{A}$ and a component $\mathbf{x}_\mathbf{A}$ perpendicular to the nullspace of $\mathbf{A}$ by employing the project $\mathbf{P}_\mathbf{A}$ onto the range of $\mathbf{A}^\mathsf{H}$, that is, $\mathbf{x}_\mathbf{A}^\perp = \mathbf{P}_\mathbf{A}^\perp \mathbf{x}$ and $\mathbf{x}_\mathbf{A} = \mathbf{P}_\mathbf{A}\mathbf{x}$, respectively, such that $\mathbf{x}_\mathbf{A}^\mathsf{H}\mathbf{x}_\mathbf{A} = 0$. Substituting into κ_N results in

$$\kappa_\mathsf{N}(\mathbf{A}\mathbf{x}, \mathbf{x}) = \frac{\| \mathbf{A} \| \cdot \| \mathbf{x}_\mathbf{A} + \mathbf{x}_\mathbf{A}^\perp \|}{\| \mathbf{A}\mathbf{x}_\mathbf{A} \|}$$

7.3.1. Condition of Unitary Matrix

$$\kappa(\mathbf{U}) = \| \mathbf{U} \| \cdot \| \mathbf{U}^\mathsf{H} \| = 1$$

8. Stability of an Algorithm

The algorithm $\tilde{\mathbf{f}}(\mathbf{x})$ to solve a problem $\mathbf{f}(\mathbf{x})$ is denoted as

$$\tilde{\mathbf{f}} : \mathbb{C}^m \to \mathbb{C}^n; \mathbf{x} \to \tilde{\mathbf{y}} = \tilde{\mathbf{f}}$$

8.1. Accuracy

For any possible value for the input $\mathbf{x}$, the relative error of the outcome $\tilde{\mathbf{f}}(\mathbf{x})$ of the algorithm is in the order of ϵ_M, which is the least error to be expected.

$$\forall \mathbf{x} \in \mathbb{C}^m : \frac{\| \tilde{\mathbf{f}}(\mathbf{x}) - \mathbf{f}(\mathbf{x}) \|}{\| \mathbf{f}(\mathbf{x}) \|} = \mathcal{O}(\epsilon_\mathsf{M})$$

8.2. Stability

An algorithm $\tilde{\mathbf{f}}$ is stable, if

$$\forall \mathbf{x} \in \mathbb{C}^m : \exists \tilde{\mathbf{x}} \text{ with } \frac{\| \tilde{\mathbf{x}} - \mathbf{x} \|}{\| \mathbf{x} \|} = \mathcal{O}(\epsilon_\mathsf{M})$$

such that

$$\frac{\| \tilde{\mathbf{f}}(\mathbf{x}) - \mathbf{f}(\tilde{\mathbf{x}}) \|}{\| \mathbf{f}(\tilde{\mathbf{x}}) \|} = \mathcal{O}(\epsilon_\mathsf{M})$$

It is not possible to infer the accuracy of an algorithm from the observation that it is stable.

8.3. Backward Stability

An algorithm $\tilde{\mathbf{f}}$ is backward stable, if

$$\forall \mathbf{x} \in \mathbb{C}^m : \exists \tilde{\mathbf{x}} \text{ with } \frac{\| \tilde{\mathbf{x}} - \mathbf{x} \|}{\| \mathbf{x} \|} = \mathcal{O}(\epsilon_\mathsf{M})$$

such that

$$\tilde{\mathbf{f}}(\mathbf{x}) = \mathbf{f}(\tilde{\mathbf{x}})$$

This definition of backward stability is stronger than the definition of stability before, since the relative error between the output of the algorithm for the true input $\mathbf{x}$, i.e., $\tilde{\mathbf{f}}(\mathbf{x})$, and the output of the underlying problem $\mathbf{f}(\tilde{\mathbf{x}})$ for the perturbed input $\tilde{\mathbf{x}}$ most be zero. **Backward stability implies stability**.

$$\frac{\| \tilde{\mathbf{f}}(\mathbf{x}) - \mathbf{f}(\mathbf{x}) \|}{\| \mathbf{f}(\mathbf{x}) \|} = \mathcal{O}(\kappa_\mathsf{N}(\mathbf{f}, \mathbf{x})\epsilon_\mathsf{M})$$

9. System of Equations

$$\mathbf{A} = \mathbf{L}\mathbf{U}$$

with the unit lower-triangular $\mathbf{L} \in \mathbb{C}^{m\times m}$ and the upper-triangular $\mathbf{U} \in \mathbb{C}^{m\times m}$ The coefficients for $\mathbf{L}$ are given as $l_{i,j} = \frac{a_{ij}}{a_{jj}}$

$$\mathbf{L} = \mathbf{I} + \mathbf{l}_1\mathbf{e}_1^\mathsf{T} + \cdots \mathbf{l}_{m-1}e_{m-1}^\mathsf{T} = \begin{bmatrix} 1 & & & \\ l_{21} & 1 & & \\ \vdots & \ddots & \ddots & \\ l_{m,1} & \cdots & l_{m,m-1} & 1 \end{bmatrix}$$

Algorithm 13 Gaussian Elimination without Pivoting

```
function GaussianEliminationNoPivoting(A)                ▷ square full-rank A ∈ C^{m×m}
    U ← A
    L ← I                                                ▷ initialize diagonal of L
    for k = 1 to m − 1 do
        [L]_{k+1:m,k} ← (1/u_kk) [U]_{k+1:m,k}           ▷ coefficients for Gaussian elimination step k
        [U]_{k+1:m,k} ← 0                                ▷ elimination for column k
        [U]_{k+1:m,k+1:m} ← [U]_{k+1:m,k+1:m} − [L]_{k+1:m,k}[U]_{k,k+1:m}   ▷ elimination for rest
    end for
    return L, U
end function
```

Operation Count Algorithm 13: #FLOPS=$\mathcal{O}(\frac{2m^3}{3})$

Algorithm 14 Solving Equation System with Gaussian Elimination

```
function SolvingEquationSystemLU(A, b)
    L, U ← lu(A)                                         ▷ LU factorization of A
    w ← L^{-1} b                                         ▷ back substitution
    x ← U^{-1} w                                         ▷ back substitution
    return x
end function
```

Operation Count Algorithm 14: #FLOPS=$\mathcal{O}(\frac{2m^3}{3} + 2n^2)$
Worst case error for Gauss with LU is:
$\mathcal{O}(\rho_{\text{worst-case}} \cdot \epsilon_\mathsf{M})$ with $\rho_{\text{worst-case}} = 2^{m-1}$. Solving an equation system with QR does not have such problems.

9.1. Stability of Gaussian Elimination

Without pivoting it holds that

$$\tilde{\mathbf{L}}\tilde{\mathbf{U}} = \mathbf{A} + \Delta\mathbf{A} \quad \text{with} \quad \frac{\| \Delta\mathbf{A} \|}{\| \mathbf{L} \| \| \mathbf{U} \|} = \mathcal{O}(\epsilon_\mathsf{M})$$

Gaussian elimination without pivoting is not backward stable!

9.2. Complete Pivoting

Leads to smallest coefficients for back substitution procedure. We have to compare $(m-k+1)^2$ elements in the k-th step. All entries in $\mathbf{L}$ below the diagonal are smaller than one.

Operation Count: #FLOPS=$\mathcal{O}(\frac{m^3}{3})$

Full pivoting ensures that all off-diagonal elements of the resulting $\mathbf{U}$ are smaller than or equal to the diagonal entry in the same row. Thus complete pivoting also improves the properties of the second back substitution.

Algorithm 15 Gaussian Elimination with Partial Pivoting

```
function GaussianEliminationPivoting(A)
    U ← A, L ← I, Π ← I                                  ▷ initialization
    for k = 1 to m − 1 do
        i_max ← max_{k≤i≤m} |u_ik|                       ▷ find maximum pivot
        [U]_{k,k:m} ↔ [U]_{i_max,k:m}                    ▷ exchange rows i_max and k of U
        [L]_{k,1:k−1} ↔ [L]_{i_max,1:k−1}                ▷ exchange rows of lower non-zero part of L
        [Π]_{k,:} ↔ [Π]_{i_max,:}                        ▷ include permutation in Π
        [L]_{k+1:m,k} ← (1/u_kk) [U]_{k+1:m,k}           ▷ coefficients for Gaussian elimination step k
        [U]_{k+1:m,k} ← 0                                ▷ elimination for column k
        [U]_{k+1:m,k+1:m} ← [U]_{k+1:m,k+1:m} − [L]_{k+1:m,k}[U]_{k,k+1:m}   ▷ elimination of rest
    end for
    return L, U, Π
end function
```

Operation Count Algorithm 15: #FLOPS=$\mathcal{O}(\frac{2m^3}{3} + \frac{m^2}{2})$

9.3. Partial Pivoting

With partial pivoting we have

$$\mathbf{\Pi}\mathbf{A} = \mathbf{L}\mathbf{U}$$

For partial pivoting $m-k+1$ elements must be compared in step k

Operation Count: #FLOPS=$\mathcal{O}(\frac{m^2}{2})$

The computational costs of partial pivoting compared to the LU factorization ($\mathcal{O}(\frac{2m^3}{3})$) are neglectable.

Gaussian elimination with partial pivoting is potentially backward stable. With growth factor $\rho = \frac{\max_{i,j}(|u_{i,j}|)}{\max_{i,j}|a_{i,j}|}$

With pivoting it holds: $\tilde{\mathbf{L}}\tilde{\mathbf{U}} = \mathbf{A} + \Delta\mathbf{A}$ with $\frac{\| \Delta\mathbf{A} \|}{\| \mathbf{L} \| \| \mathbf{U} \|} = \mathcal{O}(\rho\epsilon_\mathsf{M})$

9.4. Cholesky Factorization

A positive definite matrix $\mathbf{A}$ is Hermitian i.e. $\mathbf{A}^\mathsf{H} = \mathbf{A}$ All eigenvalues of a Hermitian $\mathbf{A}$ are real-valued. If $\mathbf{A}$ is not only Hermitian but positive definite, all eigenvalues are not only real-valued but also positive!

$$\forall \mathbf{x} \in \mathbb{C}^m : \mathbf{x}^\mathsf{H}\mathbf{A}\mathbf{x} > 0$$

$$\mathbf{A} \succ 0$$

For Cholesky the matrix $\mathbf{A}$ must be positive definite!
$\mathbf{A} \succ 0$

All diagonal elements of a positive definite matrix are greater than zero!

$$\mathbf{e}_k^\mathsf{T}\mathbf{A}\mathbf{e}_k^\mathsf{T} = [\mathbf{A}]_{k,k} > 0$$

With the Cholesky factorization the root of matrix can be defined as

$$\mathbf{A}^{1/2} = \mathbf{L}\mathbf{D}^{1/2}$$

$$\mathbf{A} = \mathbf{A}^{1/2}\mathbf{A}^{\mathsf{H}/2}$$

Algorithm 16 Cholesky Factorization

```
function CholeskyFactorization(A)
    L ← A
    for k = 1 to m − 1 do
        for j = k + 1 to m do
            L_{j:m,j} ← L_{j:m,j} − (l*_jk / l_kk) L_{j:m,k}   ▷ elimination only of lower triangular part
        end for
    end for
    D ← diag(diag(L))
    L ← tril(LD^{-1})                                    ▷ copy the lower triangular part, set rest to zero
    return L, D
end function
```

Operation Count Algorithm 16: #FLOPS=$\mathcal{O}(\frac{1}{3}m^3)$

$\#\text{FLOP}_k = \sum_{j=k+1}^{m} 2m - 2j + 3 = \sum_{j'=1}^{m-k} 2j' + 1$
$= (m-k)(m-k+1) + m - k \approx m^2 - 2mk + k^2$

$\#\text{FLOPS} = \sum_{k=1}^{m-1} \#\text{FLOPS}_k = \sum_{k=1}^{m-1} m^2 - 2mk + k^2$
$= (m-1)m^2 - m^2(m-1) + \frac{(2m-1)m(m-1)}{6}$

9.5. Stability of Cholesky Factorization

Contrary to Gaussian Elimination, the Cholesky factorization can be used without pivoting. If, however, the stability of the Cholesky factorization must be improved, pivoting can be employed.

$$\mathbf{\Pi}\mathbf{A}\mathbf{\Pi}^\mathsf{T} = \mathbf{L}\mathbf{D}\mathbf{L}^\mathsf{H}$$

Cholesky factorization, even without pivoting, can be shown to be backward stable

9.6. Hessenberg-Form

A matrix whose elements below the first lower sub-diagonal are all zero is called **Hessenberg** matrix $\mathbf{H}$

$$\mathbf{Q}^\mathsf{H}\mathbf{A}\mathbf{Q} = \begin{bmatrix} \times & \times & \cdots & \times & \times \\ \times & \times & \cdots & \times & \times \\ 0 & \times & \cdots & \times & \times \\ \vdots & \ddots & \ddots & \vdots & \vdots \\ 0 & \cdots & 0 & \times & \times \end{bmatrix}$$

The Hessenberg-Form can be achieved for a matrix $\mathbf{A} \in \mathbb{C}^{m\times m}$ if we employ unitary reflection matrices $\mathbf{Q}_i$ with the resulting reflection matrix $\mathbf{Q} = \mathbf{Q}_1 \cdots \mathbf{Q}_{m-2}$.

$$\mathbf{Q} = \begin{bmatrix} \mathbf{I}_i & \mathbf{0}^\mathsf{T} \\ \mathbf{0} & \mathbf{F}_i \end{bmatrix} \in \mathbb{C}^{m\times m}$$

With the Householder reflectors $\mathbf{F}_i \in \mathbb{C}^{m-i+1\times m-i+1}$ and $I_i \in \mathbb{C}^{i-1\times i-1}$

For Hermitian $\mathbf{A}$, i.e. $\mathbf{A}^\mathsf{H} = \mathbf{A}$ the result of the transformations leading to a Hessenberg matrix is a Hermitian Hessenberg matrix, that is, a tri-diagonal matrix.

$$\mathbf{H} = \mathbf{Q}^\mathsf{H}\mathbf{A}\mathbf{Q} = \begin{bmatrix} \times & \times & 0 & 0 & 0 & \cdots & 0 \\ \times & \times & \times & 0 & 0 & \cdots & 0 \\ 0 & \times & \times & \times & 0 & \cdots & 0 \\ \vdots & \ddots & \ddots & \ddots & \ddots & \ddots & \vdots \\ 0 & \cdots & 0 & \times & \times & \times & 0 \\ 0 & \cdots & 0 & 0 & \times & \times & \times \\ 0 & \cdots & 0 & 0 & 0 & \times & \times \end{bmatrix} = \mathbf{H}^\mathsf{H}$$