Numerics II

Yoann Le Hénaff, Yanyan Shi

Summer semester, 2025

Forefront

This document is a translation of the notes taken by Markus Klein in 2009-2010, of the lecture given in German by Prof. Dr. Christian Lubich.

Any mistake is most likely due to YS or YLH.

Contents

Chapter 1

Fast Fourier Transform

1.1 Fourier Series

Definition 1.1 – Fourier Transform

Let $(c_n)_{n\in\mathbb{Z}}$ be an absolutely summable sequence of complex numbers, i.e., $\sum_{n\in\mathbb{Z}} |c_n| < \infty$. The Fourier transform of $(c_n)_{n\in\mathbb{Z}}$ is given by:

$$\hat{c}(t) = \sum_{n=-\infty}^{\infty} c_n e^{int}, \quad t \in \mathbb{R}.$$

Proposition 1.1 – Properties of the Fourier Transform

The Fourier transform $\hat{c}(t)$ satisfies:

1. $\hat{c}$ is 2π-periodic.
2. $\hat{c}$ is continuous.
3. It satisfies the orthogonality relation:
$$\frac{1}{2\pi}\int_0^{2\pi} e^{-int}e^{imt}dt = \begin{cases} 1, & m = n, \\ 0, & m \neq n. \end{cases}$$
4. The inverse formula holds:
$$c_n = \frac{1}{2\pi}\int_0^{2\pi} e^{-int}\hat{c}(t)dt.$$
5. The Parseval equation holds:
$$\sum_{n=-\infty}^{\infty} |c_n|^2 = \frac{1}{2\pi}\int_0^{2\pi} |\hat{c}(t)|^2 dt.$$
6. Let (c_n), (d_n) be absolutely summable sequences. Then, for the convolution, we have:
$$(c * d)_n := \sum_{j=-\infty}^{\infty} c_{n-j}d_j$$
defined as the convolution, and the convolution theorem holds:
$$\widehat{c * d}(t) = \hat{c}(t) \cdot \hat{d}(t)$$

Proof. 2. $\sum_{n=-N}^{N} c_n e^{int} \to \hat{c}(t)$ is uniformly convergent for $N \to \infty$ because

$$\left| \hat{c}(t) - \sum_{n=-N}^{N} c_n e^{int} \right| \leq \sum_{|n|>N} |c_n| \to 0$$

and this convergence is independent of t, implying that $\hat{c}$ is continuous.

□

Definition 1.2 – Fourier Coefficients

Let f be a 2π-periodic continuous function. Then for $n \in \mathbb{Z}$,

$$c_n = \frac{1}{2\pi} \int_0^{2\pi} f(t) e^{-int} \, dt$$

is the n-th Fourier coefficient of f, and we denote $c_n = \hat{f}(n)$.

Theorem 1.1 – Calculation of Fourier Coefficients and Estimation

Let f be 2π-periodic and p-times differentiable, with $f^{(p)}$ absolutely integrable (it suffices that $f \in W^{p,1}$). Then the following hold:

1. The n-th Fourier coefficient of $f^{(p)}$ is $(in)^p \cdot c_n$.
2. $c_n = O(|n|^{-p})$, i.e., $|c_n| \leq M \cdot |n|^{-p}$ with

$$M = \frac{1}{2\pi} \int_0^{2\pi} |f^{(p)}(t)| \, dt$$

Proof. 1. We obtain by (multiple) integration by parts:

$$\frac{1}{2\pi} \int_0^{2\pi} f^{(p)}(t) e^{-int} \, dt = -\frac{1}{2\pi} \int_0^{2\pi} f^{(p-1)}(t)(-in) e^{-int} \, dt = (in)^p \frac{1}{2\pi} \int_0^{2\pi} f(t) e^{-int} \, dt$$

where the boundary terms vanish due to periodicity.

2. From the above, we have:

$$|c_n| \cdot |n|^p = \left| \frac{1}{2\pi} \int_0^{2\pi} f^{(p)}(t) e^{-int} \, dt \right| \leq \frac{1}{2\pi} \int_0^{2\pi} |f^{(p)}(t)| \, dt$$

□

Remark 1.1

In particular, (c_n) is absolutely summable if $f \in C^2$ (or $f \in W^{2,1}$). With greater effort, it can also be shown that $f \in C^1$ would suffice.

Theorem 1.2 – Convolution Theorem

Let f, g be continuous and 2π-periodic. Then the convolution of f and g, defined by

$$(f * g)(t) = \frac{1}{2\pi} \int_0^{2\pi} f(t - \tau) g(\tau) \, d\tau$$

is again 2π-periodic and continuous. For the Fourier coefficients of the convolution, we have:

$$\widehat{f * g}(n) = \hat{f}(n) \cdot \hat{g}(n), \quad n \in \mathbb{Z}$$

Proof. The periodicity and continuity are clear from Analysis II. Now we calculate:

$$\begin{aligned}\widehat{f * g}(n) &= \frac{1}{2\pi}\int_0^{2\pi}\left(\frac{1}{2\pi}\int_0^{2\pi} f(t-\tau)g(\tau)\,d\tau\right)e^{-int}\,dt \\ &= \left(\frac{1}{2\pi}\right)^2\int_0^{2\pi}\int_0^{2\pi} f(t-\tau)e^{-in(t-\tau)}\cdot g(\tau)e^{-in\tau}\,dtd\tau\end{aligned}$$

Set $s := t - \tau$, then we obtain:

$$\begin{aligned}&= \frac{1}{2\pi}\int_0^{2\pi} f(s)e^{-ins}ds \cdot \frac{1}{2\pi}\int_0^{2\pi} g(\tau)e^{-in\tau}d\tau \\ &= \hat{f}(n)\cdot\hat{g}(n)\end{aligned}$$

Thus, the claim follows. □

Remark 1.2 – Generalization

The Fourier transform is also defined for $n \in \mathbb{R}$, and the convolution theorem holds for this case as well.

Remark 1.3 – Outlook

Can a continuous function f be recovered from its Fourier coefficients? That is, does

$$f(t) \stackrel{?}{=} \sum_{n=-\infty}^{\infty} c_n e^{int}$$

We will see that without additional conditions on f, the sequence (c_n) is not absolutely summable. Even if (c_n) is absolutely summable, does $f(t) = \hat{c}(t)$? We will later see that this is indeed the case.

Theorem 1.3 – Fejér's Theorem (Fejér, 1904)

Let $f : \mathbb{R} \to \mathbb{C}$ be continuous and 2π-periodic with Fourier coefficients

$$(c_n)_{n\in\mathbb{Z}} = \frac{1}{2\pi}\int_0^{2\pi} e^{-int}f(t)\,dt$$

Then the following holds:

$$\sum_{k=-n}^{n}\left(1 - \frac{|k|}{n+1}\right)c_k e^{ikt} \to f(t) \quad \text{uniformly in } t$$

Proof. 1. We consider the convolution with the convolution theorem:

$$(e^{in\tau} * f)(t) = \frac{1}{2\pi}\int_0^{2\pi} e^{in(t-\tau)}f(\tau)\,d\tau = e^{int}c_n$$

Thus, by linearity:

$$\sum_{j=-n}^{n}\left(1 - \frac{|j|}{n+1}\right)e^{ijt}c_j := (K_n * f)(t)$$

with the so-called Fejér kernel K_n, for which we have:

$$K_n(t) = \sum_{j=-n}^{n}\left(1 - \frac{|j|}{n+1}\right)e^{ijt}$$

2. We show that the reduction formula holds:

$$K_n(t) = \frac{1}{n+1}\left(\frac{\sin\left(\frac{n+1}{2}t\right)}{\sin\left(\frac{t}{2}\right)}\right)^2$$

This is shown using the definition of the sine function, where we use the identity:

$$\sin^2\left(\frac{t}{2}\right) = \frac{1}{2}(1-\cos t) = -\frac{1}{4}e^{-it} + \frac{1}{2} - \frac{1}{4}e^{it}$$

Direct computation then gives the identity:

$$\left(-\frac{1}{4}e^{it} + \frac{1}{2} - \frac{1}{4}e^{it}\right)\sum_{j=-n}^{n}\left(1-\frac{|j|}{n+1}\right)e^{ijt} = \frac{1}{n+1}\left(-\frac{1}{4}e^{-i(n+t)t} + \frac{1}{2} - \frac{1}{4}e^{i(n+1)t}\right)$$

This simplifies to:

$$= \frac{1}{n+1}\sin^2\left(\frac{(n+1)t}{2}\right)$$

3. We now consider some properties:

   (a) Due to the orthogonality relation:

$$\frac{1}{2\pi}\int_0^{2\pi} K_n(t)\,dt = \sum_{j=-n}^{n}\left(1-\frac{|j|}{n+1}\right)\frac{1}{2\pi}\underbrace{\int_0^{2\pi} e^{ijt}\,dt}_{=\delta_{j0}} = 1$$

   (b) $K_n(t) \geq 0\ \forall t$, n due to the reduction formula.

   (c) $\forall \delta \in (0,\pi)$, we have:

$$\lim_{n\to\infty}\int_\delta^{2\pi-\delta} K_n(t)\,dt = 0$$

   because

$$K_n(t) \leq \frac{1}{n+1}\frac{1}{\sin^2\left(\frac{\delta}{2}\right)} \to 0 \quad \text{as} \quad n\to\infty$$

4. It follows that:

$$(K_n * f)(t) - f(t) = \frac{1}{2\pi}\int_0^{2\pi} K_n(\tau)\left(f(t-\tau)-f(t)\right)d\tau$$

and thus:

$$\int_0^{2\pi} K_n(\tau)\left(f(t-\tau)-f(t)\right)d\tau = I_1 + I_2$$

where

$$I_1 = \int_{-\delta}^{\delta}\left(f(t-\tau)-f(t)\right)K_n(\tau)d\tau \quad \text{and} \quad I_2 = \int_\delta^{2\pi-\delta}\left(f(t-\tau)-f(t)\right)K_n(\tau)d\tau$$

We estimate:

$$I_1 \leq \max_{|\tau|\leq\delta}|f(t-\tau)-f(t)| \cdot \frac{1}{2\pi}\int_{-\delta}^{\delta} K_n(\tau)d\tau \leq \frac{1}{2\pi}\int_{-\pi}^{\pi} K_n(\tau)d\tau = 1$$

We estimate the second part:

$$I_2 \leq \frac{1}{2}\cdot 2\max_{0\leq\theta\leq 2\pi}|f(\theta)| \cdot \underbrace{\int_\delta^{2\pi-\delta}|K_n(\tau)|d\tau}_{\to 0}$$

Thus, for $n \to \infty$, we have:

$$\limsup_{n\to\infty} |(K_n * f)(t) - f(t)| \leq \max_{|\tau|\leq\delta} |f(t-\tau) - f(t)| \quad \text{for} \quad \forall \delta \in (0, \pi)$$

As $\delta \to 0$, the term tends to 0 due to continuity. Since f is uniformly continuous, the convergence is uniform. Thus, $K_n * f(t) \to f(t)$ uniformly in t.

□

Remark 1.4

It should be noted that

$$\sum_{k=-n}^{n} \left(1 - \frac{|k|}{n+1}\right) c_k e^{int} = \frac{1}{n+1} \sum_{m=0}^{n} \left(\sum_{k=-m}^{m} c_k e^{ikt} \right)$$

is the arithmetic mean of all partial sums, where the partial sums need not necessarily converge.

Theorem 1.4 – Uniqueness Theorem

Let f and g be 2π-periodic and continuous functions with the same Fourier coefficients. Then $f = g$.

Proof. We have

$$f(t) = \lim_{n\to\infty} \sum_{j=-n}^{n} \left(1 - \frac{|j|}{n+1}\right) c_j e^{ijt} = g(t) \quad \forall t.$$

□

Theorem 1.5 – Representation via Fourier Coefficients

Let f be a 2π-periodic and continuous function. If the Fourier coefficients are absolutely summable, then:

$$f(t) = \sum_{n=-\infty}^{\infty} c_n e^{int} \quad \forall t.$$

Proof. The function f and the series have the same Fourier coefficients, and by the uniqueness theorem, both functions must coincide. □

Remark 1.5

This is particularly true if f is once continuously differentiable (i.e., $f \in W^{1,1}$).

Theorem 1.6 – Interpretation of Fejér's Theorem

Every continuous 2π-periodic function can be uniformly approximated by trigonometric polynomials.

Proof. This follows directly from the statement of Fejér's theorem. □

Theorem 1.7 – Weierstrass Approximation Theorem

Every continuous function on a compact interval $g : [a, b] \to \mathbb{R}$ can be uniformly approximated by polynomials, i.e., $\forall \epsilon > 0$, there exists a polynomial p such that

$$\max_{x\in[a,b]} |p(x) - g(x)| < \epsilon.$$

Proof. Without loss of generality, assume that $[a, b] = [-1, 1]$. Set $f(t) = g(x)$ for $x = \cos t$, or more precisely $t = \arccos x \in [0, \pi]$. Next, extend f as an even function, i.e., $f(-t) = f(t)$, and note that f is continuous.

For even functions, we have $c_{-n} = c_n$, because:

$$c_{-n} = \frac{1}{2\pi} \int_0^{2\pi} e^{int} f(t)\, dt = \frac{1}{2\pi} \int_{-2\pi}^{0} e^{-in\tau} f(-\tau)\, d\tau = \frac{1}{2\pi} \int_0^{2\pi} e^{-in\tau} \underbrace{f(-\tau)}_{f(\tau)}\, d\tau = c_n$$

due to the transformation of variables and the 2π-periodicity.

Furthermore, by Fejér's theorem, we know:

$$\sum_{j=-n}^{n} \left(1 - \frac{|j|}{n+1}\right) c_j e^{ijt} = c_0 + 2 \sum_{k=1}^{n} \left(1 - \frac{k}{n+1}\right) c_k \cos(kt).$$

We also know that the left-hand side converges uniformly to $f(t)$. Since $\cos(k \arccos x) = T_k(x)$, the k-th Chebyshev polynomial, we obtain:

$$c_0 + 2 \sum_{k=1}^{n} \left(1 - \frac{k}{n+1}\right) c_k T_k(x) \to g(x).$$

□

Remark 1.6

The convergence in the last theorem can be arbitrarily slow. It will be faster if the coefficients decay rapidly, which is particularly the case when the function is frequently differentiable.

1.2 Discrete Fourier Transform

We consider finite sequences $x = (x_0, \ldots, x_{N-1}) \in \mathbb{C}^N$ periodically extended to arbitrary integer indices (if needed). That is, we set $x_k = x_\ell$ if $k \equiv \ell \mod N$.

Definition 1.3 – Discrete Fourier Transform

The mapping $\mathcal{F}_N : \mathbb{C}^N \to \mathbb{C}^N$ is defined by $\mathcal{F}_N x = \hat{x}$ with

$$\hat{x}_k = \sum_{j=0}^{N-1} w_N^{k \cdot j} x_j,$$

where $w_N = e^{i\frac{2\pi}{N}}$ is the primitive N-th complex root of unity, i.e., $w_N^N = 1$.

Remark 1.7

If N is clear from context, we simply write w for the root of unity.

Remark 1.8 – Computational Complexity

The direct computation of the discrete Fourier transform requires about N^2 operations (multiplications and additions). We will later see that the Fast Fourier Transform (FFT) requires about $N \log_2 N$ operations if $N = 2^L$.

Lemma 1.1 – Orthogonality Relation of the Discrete Fourier Transform

It holds that:

$$\sum_{k=0}^{N-1} w_N^{k\ell} \bar{w}_N^{km} = \begin{cases} N, & \ell \equiv m \mod N \\ 0, & \text{otherwise} \end{cases}$$

Proof. Let $\bar{w} = w^{-1}$. For $\ell \equiv m$,

$$\sum_{k=0}^{N-1} 1 = N.$$

Otherwise, it follows that

$$\sum_{k=0}^{N-1} w_N^{k\ell} w_N^{-km} = \sum_{k=0}^{N-1} w_N^{k(\ell-m)} = \frac{1-(w_N^{\ell-m})^N}{1-w_N^{\ell-m}} = 0.$$

□

Theorem 1.8 – Parseval's Equation

Let $\mathbb{C}^N$ be equipped with the Euclidean norm. Then we have:

$$\frac{1}{\sqrt{N}} \|\mathcal{F}_N x\| = \|x\|, \quad \forall x \in \mathbb{C}^N,$$

i.e., this transformation is an isometry/unitary mapping. Explicitly, this means:

$$\frac{1}{N} \sum_{k=0}^{N-1} |\hat{x}_k|^2 = \sum_{j=0}^{N-1} |x_j|^2.$$

Proof. We compute:

$$\begin{aligned} \|\hat{x}\|^2 &= \sum_{k=0}^{N-1} \hat{x}_k \bar{\hat{x}}_k \\ &= \sum_{k=0}^{N-1} \sum_{\ell=0}^{N-1} w_N^{k\ell} x_\ell \sum_{m=0}^{N-1} \bar{w}_N^{km} \bar{x}_m \\ &= \sum_{\ell} \sum_{m} x_\ell \bar{x}_m \underbrace{\sum_{k} w_N^{k\ell} \bar{w}_N^{km}}_{=N\delta_{lm}} \\ &= N \sum_{\ell=0}^{N-1} x_\ell \bar{x}_\ell = N\|x\|^2, \end{aligned}$$

which proves the equality. □

Notation 1.1 – Fourier Transform Matrix

The mapping $\mathcal{F}_N : \mathbb{C}^N \to \mathbb{C}^N$ is linear and represented by the matrix $(w_N^{kj})_{k,j=0}^{N-1}$. Similarly, we define:

$$\bar{\mathcal{F}}_N : \mathbb{C}^N \to \mathbb{C}^N, \quad \text{with matrix } (\bar{w}_N^{kj})_{k,j=0}^{N-1} = (w_N^{-kj}).$$

Theorem 1.9 – Inverse Discrete Fourier Transform

We have:

$$\mathcal{F}_N^{-1} = \frac{1}{N} \bar{\mathcal{F}}_N,$$

or explicitly,

$$x_j = \frac{1}{N} \sum_{k=0}^{N-1} \bar{w}_N^{kj} \hat{x}_k, \quad \text{for } j = 0, \ldots, N-1.$$

Proof. From the orthogonality lemma, we know that $\mathcal{F}_N \cdot \bar{\mathcal{F}}_N = NI_N$. Explicitly,

$$\sum_{k=0}^{N-1} \bar{w}_N^{kj} \sum_{\ell=0}^{N-1} w_N^{k\ell} x_\ell = \sum_{\ell=0}^{N-1} x_\ell \sum_{k=0}^{N-1} \bar{w}_N^{kj} w_N^{k\ell} = N x_j.$$

Thus, dividing by N gives the result. □

Definition 1.4 – Pointwise Product

For sequences $x, y \in \mathbb{C}^N$, the pointwise multiplication is defined as:

$$(x \cdot y)_k = x_k y_k.$$

Definition 1.5 – Convolution Multiplication

For N-periodic sequences $x, y \in \mathbb{C}^N$, we define the convolution product $x * y \in \mathbb{C}^N$ as:

$$(x * y)_k = \sum_{j=0}^{N-1} x_{k-j} y_j.$$

Theorem 1.10 – Convolution Theorem

The Fourier transform converts convolution into pointwise multiplication:

$$\mathcal{F}_N(x * y) = (\mathcal{F}_N x) \cdot (\mathcal{F}_N y).$$

Proof. We consider the m-th component of the left-hand side:

$$(\mathcal{F}_N(x * y))_m = \sum_{k=0}^{N-1} w_N^{mk} \sum_{j=0}^{N-1} x_{k-j} y_j.$$

Using index shift $k - j = \ell$:

$$\sum_{j=0}^{N-1} \sum_{\ell=0}^{N-1} w_N^{m(\ell+j)} x_\ell y_j.$$

Since sequences are N-periodic,

$$\sum_{j=0}^{N-1} w_N^{mj} y_j \sum_{\ell=0}^{N-1} w_N^{m\ell} x_\ell = \hat{y}_m \hat{x}_m = (\mathcal{F}_N y) \cdot (\mathcal{F}_N x),$$

proving the claim. □

Corollary 1.1 – Properties of Convolution

Since pointwise multiplication is commutative and associative, convolution is also commutative and associative.

Corollary 1.2 – Direct Computation of Convolution

We have:

$$x * y = \frac{1}{N}\bar{\mathcal{F}}_N(\mathcal{F}_N x \cdot \mathcal{F}_N y),$$

which provides an efficient way to compute convolution using the Fourier transform.

Remark 1.9 – Computational Effort for Convolution

A direct computation of the convolution requires approximately N^2 multiplications and additions. With the FFT, we need $N \log_2 N$ operations for the transformation, only N operations for the pointwise multiplication, and another $N \log_2 N$ operations for the inverse FFT. Thus, using the FFT, we require only $3N \log_2 N + 2N$ operations.

1.3 Fast Fourier Transform (FFT)

Given a vector $x = (x_0, \ldots, x_{N-1}) \in \mathbb{C}^N$, we aim to compute $\hat{x} = \mathcal{F}_N x$.

Theorem 1.11 – Reduction Formula

We split the vector x into two vectors u and v, where u contains the even indices and v the odd indices:

$$x = (u_0, v_0, u_1, v_1, \ldots, u_{N/2-1}, v_{N/2-1}) \in \mathbb{C}^N.$$

Then, for $k = 0, \ldots, N/2 - 1$, we have:

$$\begin{aligned}(\mathcal{F}_N x)_k &= (F_{N/2}u)_k + w_N^k (F_{N/2}v)_k,\\ (\mathcal{F}_N x)_{k+N/2} &= (F_{N/2}u)_k - w_N^k (F_{N/2}v)_k.\end{aligned}$$

Proof. We compute the k-th entry of $\mathcal{F}_N x$:

$$(\mathcal{F}_N x)_k = \sum_{\ell=0}^{N-1} w_N^{k\ell} x_\ell = \sum_{j=0}^{N/2-1} w_N^{k(2j)} u_j + \sum_{j=0}^{N/2-1} w_N^{k(2j+1)} v_j.$$

Since $w_N^{2j} = w_{N/2}^j$, this simplifies to:

$$(\mathcal{F}_N x)_k = \sum_{j=0}^{N/2-1} w_{N/2}^{kj} u_j + w_N^k \sum_{j=0}^{N/2-1} w_{N/2}^{kj} v_j = (F_{N/2}u)_k + w_N^k (F_{N/2}v)_k.$$

Due to periodicity, we obtain the second equation using $w_N^{k+N/2} = w_N^k w_N^{N/2} = -w_N^k$. □

Remark 1.10

If $F_{N/2}u$ and $F_{N/2}v$ are known, we require $N/2$ multiplications and N additions to compute $\mathcal{F}_N x$.

Remark 1.11 – Historical Note

This formula dates back to Cooley-Tukey (1965). Similar ideas were found by Danilson and Lanzos (1942), Runge (1925), Gauss, and even Caesar's "divide et impera."

Remark 1.12 – Algorithm Complexity

If $N = 2^L$, we can recursively divide the vector L times until we obtain vectors of length 1. This results in a computational cost of $L \cdot N/2$ multiplications, where $L = \log_2 N$.

Theorem 1.12 – Computational Complexity of FFT

For $N = 2^L$, computing $\mathcal{F}_N x$ requires:

1. $\frac{1}{2} N \log_2 N$ complex multiplications,
2. $N \log_2 N$ complex additions.

Remark 1.13 – Order of Elements

During the execution of the FFT algorithm, the order of elements is permuted. Using binary representation, we obtain the order by reversing the binary digits:

Input (Decimal)	Input (Binary)	Output (Binary)	Output (Decimal)
0	000	000	0
1	001	100	4
2	010	010	2
3	011	110	6
4	100	001	1
5	101	101	5
6	110	011	3
7	111	111	7

Thus, the order is obtained by mirroring the binary digits.

Remark 1.14 – Comparison with Direct Computation

We summarize key values that highlight the advantage of FFT:

N	N^2	$N \log_2 N$	Quotient
$2^5 = 32$	10^3	160	6.4
$2^{10} \approx 10^3$	10^6	10^4	100
$2^{20} \approx 10^6$	10^{12}	$2 \cdot 10^7$	50,000

1.4 Approximation of Fourier coefficients, trigonometric interpolation

We would like to clarify the relationship between the discrete and continuous Fourier transforms. The Fourier coefficients for a 2π-periodic, continuous function f are given by:

$$\hat{f}(n) = c_n = \frac{1}{2\pi} \int_0^{2\pi} e^{-int} f(t)\, dt.$$

We approximate the integral using the trapezoidal rule with step size $h = \frac{2\pi}{N}$, obtaining:

$$\hat{f}_N(n) = \frac{1}{2\pi} \left(\frac{h}{2} e^{-in0} f(0) + h e^{-inh} f(h) + \cdots + h e^{-in(N-1)h} f((N-1)h) + \frac{h}{2} e^{-inNh} f(Nh) \right).$$

Due to the periodicity, we have $h e^{-in0} f(0) = h e^{-inNh} f(Nh)$, so the trapezoidal rule effectively becomes the rectangular rule. Thus, we obtain:

$$\hat{f}_N(n) = \frac{1}{N} \sum_{j=0}^{N-1} e^{-inj\frac{2\pi}{N}} f(t_j) = \frac{1}{N} \sum_{j=0}^{N-1} w_N^{-nj} f(t_j),$$

where $t_j = jh = j\frac{2\pi}{N}$. Therefore, $\hat{f}_N(n)$ is N-periodic as a vector. This leads to the following representation:

$$\hat{f}_N = \mathcal{F}_N^{-1}(f(t_j))_{j=0}^{N-1},$$

which can be computed using the Fast Fourier Transform (FFT).

Theorem 1.13 – Aliasing Formula

Let $(\hat{f}(n))_{n\in\mathbb{Z}}$ be absolutely summable. Then,

$$\hat{f}_N(n) - \hat{f}(n) = \sum_{0\neq\ell\in\mathbb{Z}} \hat{f}(n+\ell N).$$

Proof. From the previous chapter, we know that if the Fourier coefficients are absolutely summable, we have:

$$f(t) = \sum_{k=-\infty}^{\infty} \hat{f}(k)e^{ikt}.$$

Now, consider:

$$\hat{f}_N(n) = \frac{1}{N}\sum_{j=0}^{N-1} w_N^{-nj} \sum_{k=-\infty}^{\infty} \hat{f}(k)e^{ikj\frac{2\pi}{N}} = \sum_{k=-\infty}^{\infty} \hat{f}(k)\frac{1}{N}\sum_{j=0}^{N-1} w_N^{-nj}w_N^{kj},$$

where $w_N = e^{i\frac{2\pi}{N}}$. The inner sum can be simplified as follows:

$$\frac{1}{N}\sum_{j=0}^{N-1} w^{-nj}w^{kj} = \begin{cases} 1 & \text{if } k = n \mod N, \\ 0 & \text{otherwise.} \end{cases}$$

Thus, we obtain:

$$\hat{f}_N(n) = \sum_{\ell=-\infty}^{\infty} \hat{f}(n+\ell N),$$

and for $\ell = 0$, this reduces to $\hat{f}(n)$, which confirms the statement. □

Corollary 1.3 – Error Estimation for the Numerical Fourier Coefficients

Let f be p-times continuously differentiable (with $p \geq 2$) and 2π-periodic. Then, for $|n| \leq \frac{N}{2}$, we have:

$$|\hat{f}_N(n) - \hat{f}(n)| \leq C \cdot N^{-p}.$$

Proof. From the previous section, we know that:

$$|\hat{f}(n)| \leq C_0|n|^{-p},$$

with

$$C_0 = \frac{1}{2\pi}\int_0^{2\pi} |f^{(p)}(t)|\,dt.$$

On the other hand, using the aliasing formula, we have:

$$|\hat{f}_N(n) - \hat{f}(n)| = \sum_{\ell\neq 0} |\hat{f}(n+\ell N)| \leq \sum_{\ell\neq 0} C_0 \cdot |n+\ell N|^{-p}.$$

For $|n| \leq \frac{N}{2}$, we have $|n+\ell N| \geq \frac{N}{2}$. So $|n+\ell N| < N$ for only one $\ell \neq 0$. This is

$$l = \begin{cases} -1 & n > 0, \\ 1 & n < 0. \end{cases}$$

For every interval $I = [N, 2N), [2N, 3N), \cdots$ and $(-2N, -N], (-3N, -2N], \cdots$ there is exactly one ℓ such that $n + \ell N \in I$. This gives us the following in total:

$$\sum_{\ell\neq 0} |n+\ell N|^{-p} \leq (N/2)^{-p} + 2\sum_{m=1}^{\infty} (mN)^{-p} \leq c_p \cdot N^{-p},$$

where c_p is a constant derived from the converging sums. Hence, $C = C_0 c_p$ and the result follows. □

Remark 1.15

For the special case $n = 0$, we obtain for the grid spacing $h = \frac{2\pi}{N}$:

$$\frac{h}{2\pi} \sum_{j=0}^{N-1} f(t_j) - \frac{1}{2\pi} \int_0^{2\pi} f(t)\, dt = O(h^p).$$

Thus, the accuracy of the trapezoidal rule depends on the smoothness of the function, i.e., the trapezoidal rule is very accurate for smooth and periodic integrands.

Theorem 1.14 – Trigonometric Interpolation

The trigonometric polynomial

$$f_N(t) = \sum_{n=-\frac{N}{2}}^{\frac{N}{2}}{}' \hat{f}_N(n) e^{int}$$

interpolates f at the points $t_j = j\frac{2\pi}{N}$ for $j = 0, 1, \ldots, N-1$.

Proof. Due to the periodicity, we have (in particular $e^{int_j} = w_N^{nj}$):

$$f_N(t_j) = \sum_{n=0}^{N-1} \hat{f}_N(n) w_N^{nj}.$$

We have used that the left and right boundary terms are equal, so the $\sum'$ term vanishes. This formula means that

$$f_N(t_j))_{j=0}^{N-1} = \mathcal{F}_N \left(\hat{f}_N(n)\right)_{n=0}^{N-1} = \mathcal{F}_N \mathcal{F}_N^{-1} (f(t_j))_{j=0}^{N-1} = (f(t_j))_{j=0}^{N-1}.$$

Thus, $f_N(t)$ already interpolates the grid points t_j. □

Notation $\sum'$ means that the first and the last term are each taken with the factor $\frac{1}{2}$, i.e., for our sum we obtain:

$$\sum_{n=-\frac{N}{2}}^{\frac{N}{2}}{}' \hat{f}_N(n) := \frac{1}{2}\hat{f}_N\left(-\frac{N}{2}\right) + \hat{f}_N\left(-\frac{N}{2}+1\right) + \cdots + \hat{f}_N\left(\frac{N}{2}-1\right) + \frac{1}{2}\hat{f}_N\left(\frac{N}{2}\right)$$

This is the interpolation at the nodes $t_j = j\frac{2\pi}{N}$.
It would also be permissible to form one of the following sums instead:

$$\sum_{n=-\frac{N}{2}+1}^{\frac{N}{2}}, \quad \sum_{n=-\frac{N}{2}}^{\frac{N}{2}-1}, \quad \sum_{n=-\frac{N}{2}+k}^{\frac{N}{2}-1+k}$$

Using this method, we could also prove the theorem. However, we will see that our variant is the least oscillatory due to symmetry reasons.

Algorithm 1.1 – Trigonometric Interpolation

We assume that f is 2π-periodic. Then we proceed as follows:

1. Compute $f(t_j)$ for $j = 0, \ldots, N-1$ and $t_j = j \cdot \frac{2\pi}{N}$.
2. Compute $\hat{f}_N = \mathcal{F}_N^{-1}(f(t_j))_{j=0}^{N-1}$ using the FFT with a computational complexity of $O(N \log N)$ operations.
3. Obtain the trigonometric interpolation polynomial as in the previous theorem.

Theorem 1.15 – Error Estimate for Trigonometric Interpolation

If $(\hat{f}(n))_n$ is absolutely summable, then:

$$|f_N(t) - f(t)| \leq 2 \sum_{|n| \geq \frac{N}{2}} \prime |\hat{f}(n)|$$

Proof. We use the aliasing formula and obtain:

$$\begin{aligned} |f_N(t) - f(t)| &= \left| \sum_{n=-\frac{N}{2}}^{\frac{N}{2}} \prime \hat{f}_N(n) e^{int} - \sum_{n=-\infty}^{\infty} \hat{f}(n) e^{int} \right| \\ &= \left| \sum_{n=-\frac{N}{2}}^{\frac{N}{2}} \prime (\hat{f}_N(n) - \hat{f}(n)) e^{int} - \sum_{|n| \geq \frac{N}{2}} \prime \hat{f}(n) e^{int} \right| \end{aligned}$$

Using the aliasing formula:

$$\begin{aligned} &= \left| \sum_{n=-\frac{N}{2}}^{\frac{N}{2}} \prime \sum_{\ell \neq 0} \hat{f}(n + \ell N) e^{int} - \sum_{|n| \geq \frac{N}{2}} \prime \hat{f}(n) e^{int} \right| \\ &\leq \sum_{|m| \geq \frac{N}{2}} \prime |\hat{f}(m)| + \sum_{|n| \geq \frac{N}{2}} \prime |\hat{f}(n)| \\ &\leq 2 \sum_{|n| \geq \frac{N}{2}} \prime |\hat{f}(n)| \end{aligned}$$

which proves the theorem. □

Remark 1.16

If $f \in C^p$ with $p \geq 2$, then it is well known that $\hat{f}(n) = O(|n|^{-p})$. Thus, we obtain: $|f_N(t) - f(t)| = O(N^{-p+1})$.

1.5 Inverse Convolution Problem, Regularization, Filtering

Motivation (Problem Statement)
An input signal $u = u(x)$ for $x \in \mathbb{R}$ enters a device, and a signal b is measured, which no longer corresponds to the input signal.

We make the following assumptions:

1. *$u \mapsto b$ is linear.*
2. *$u \mapsto b$ is shift-invariant.*

We know that such mappings are convolutions.

Remark 1.17

This problem arises in many areas, such as image or signal processing.

Model Construction (Device Model)
We consider the following model:

$$\int_{-\infty}^{\infty} a(x-y)u(y)\,dy + \varepsilon(x) = b(x)$$

where $u(y)$ is the unknown input signal, $b(x)$ is the observed signal, and the term $a(x-y)$ represents the device function. The term $\varepsilon(x)$ represents the noise or errors in the model, including

measurement errors, rounding errors, and model errors. This noise is typically unknown, but we may know an upper bound for it, so that $|\varepsilon(x)| \leq M$ pointwise or in the quadratic mean, i.e. $\int_{-\infty}^{\infty} |\varepsilon(x)|^2 \, dx \leq M$ or similar. We aim to reconstruct the input signal u from the observed signal.

Remark 1.18 – Reduction of the problem

In general, a, u, and b have compact support, so that $\operatorname{supp}(a) = \overline{\{x : a(x) \neq 0\}}$. After a variable transformation, we can assume without loss of generality that:

$$\operatorname{supp}(a) \subseteq \left[-\frac{\pi}{2}, \frac{\pi}{2}\right], \quad \operatorname{supp}(u) \subseteq \left[-\frac{\pi}{2}, \frac{\pi}{2}\right].$$

This implies:

$$\operatorname{supp}(b - \varepsilon) \subseteq [-\pi, \pi]$$

since if $b(x) \neq \varepsilon(x)$, then $\exists y$ such that $a(x-y) \cdot u(y) \neq 0$, and thus $x - y \in \left[-\frac{\pi}{2}, \frac{\pi}{2}\right]$, and therefore $x \in [-\pi, \pi]$.

We now extend a, u, ε, and b to be 2π-periodic on $\mathbb{R}$. This reduces the problem to:

$$\frac{1}{2\pi} \int_{-\pi}^{\pi} a(x-y)u(y) \, dy + \varepsilon(x) = b(x) \quad \text{for } x \in [-\pi, \pi],$$

which we can express briefly as:

$$a * u + \varepsilon = b$$

or for the linear operator $Au = a * u$, we have:

$$Au + \varepsilon = b$$

Remark 1.19

The "usual" solution method is to solve the linear system $Au + \varepsilon = b$ in $\mathbb{R}^n$ by neglecting the disturbance ε and solving $Av = b$ using standard methods. Then the error $v - u = A^{-1}\varepsilon$ holds, provided $\|A^{-1}\varepsilon\| \leq \|v\|$, which is not necessarily true if the matrix is ill-conditioned.
Unfortunately, this is the case here.

Reminder (Convolution Theorem) It is known that: $\widehat{a * u}(n) = \hat{a}(n)\hat{u}(n)$

Remark 1.20

For the Fourier coefficients, we have:

$$\hat{a}(n)\hat{u}(n) + \hat{\varepsilon}(n) = \hat{b}(n) \quad \text{for } n \in \mathbb{Z}$$

Thus,

$$\hat{u}(n) = \frac{\hat{b}(n)}{\hat{a}(n)} - \frac{\hat{\varepsilon}(n)}{\hat{a}(n)} = \hat{v}(n) - \frac{\hat{\varepsilon}(n)}{\hat{a}(n)}$$

where $\hat{v}(n)$ solves $\hat{a}(n)\hat{u}(n) = \hat{b}(n)$.
However, since the Fourier coefficients of ε remain approximately constant, while the Fourier coefficients of a decay rapidly (if a is smooth), the expression at the back will dominate the term, leading to the result that

$$u(x) = \sum_{n=-\infty}^{\infty} \hat{u}(n)e^{inx} = \sum_{n=-\infty}^{\infty} \hat{v}(n)e^{inx} - \sum_{n=-\infty}^{\infty} \frac{\hat{\varepsilon}(n)}{\hat{a}(n)} e^{inx}$$

For large n, we observe catastrophic error amplification, meaning the problem is ill-conditioned. This is also referred to as a poorly posed problem, or an ill-posed problem.

Remark 1.21 – Alternative Approach (Minimization Problem)

We do not want to solve $Au = b$, but rather we want to minimize $\|Au-b\| \leq \|\varepsilon\|$, i.e., if $\|\varepsilon\| \approx \delta$, we require that $\|Au - b\| \leq \|\delta\|$.
In general, there are infinitely many such u that satisfy this. We wish to choose u such that $\|u''\|$ is minimal (e.g., using cubic splines). Alternatively, we might minimize $\|u\|$. Thus, we are looking for the smoothest or smallest possible u, which leads to a minimization problem.

Remark 1.22

We assume that, unless stated otherwise, we consider the L^2-norm, i.e.

$$\|f\| = \|f\|_{L^2} := \left(\frac{1}{2\pi}\int_{-\pi}^{\pi} |f(x)|^2\,dx\right)^{1/2}.$$

Theorem 1.16 – General Form of the Minimization Problem

We want to minimize $\|Lu\|$ and assume that L is linear, e.g., $Lu = u''$. Additionally, we require the constraint $\|Au - b\| \leq \delta$.
The minimum is achieved for $\|Au - b\| = \delta$.

Proof. Assume this is not the case, i.e., $\|Lu\|$ is minimal for some u with $\|Au - b\| < \delta$. Consider $\tilde{u} = (1 - \rho)u$ for some $\rho > 0$. Since L is linear, we have:

$$\|L\tilde{u}\| = (1 - \rho)\|Lu\| < \|Lu\|$$

and for $\tilde{u}$, we get:

$$\|A\tilde{u} - b\| = \|(1 - \rho)(Au - b) - \rho b\| \leq (1 - \rho)\|Au - b\| + \rho\|b\| \leq \delta$$

This leads to a contradiction for sufficiently small ρ, so the minimum must occur for $\|Au - b\| = \delta$. □

Remark 1.23

In practice, there are often additional constraints, such as considering only solutions that are monotonic or positive.

Remark 1.24

It is a reasonable assumption that $\|b\| > \delta$, otherwise, the observed signal would be weaker than the noise. This directly implies that $u \neq 0$, and hence the problem cannot be trivially solved.

Definition 1.6 – Tikhonov Regularization

We consider a fixed $\alpha > 0$ as a regularization parameter. We solve the minimization problem without the constraint:

$$\|Au - b\|^2 + \alpha\|Lu\|^2 = \min$$

As $\alpha \to 0$, we get $\|Au - b\| = 0$, but $\|Lu\|$ becomes arbitrarily large. As $\alpha \to \infty$, we get $\|Lu\| = 0$, but $\|Au - b\|$ becomes arbitrarily large. The optimal α should be chosen so that $\|Au - b\| \approx \|\varepsilon\|$, if ε is known. Otherwise, it must be determined empirically until the result "looks good".

Remark 1.25

We now clarify the connection between the minimization and regularization problems: We restrict ourselves to a finite-dimensional problem, i.e., $b \in \mathbb{R}^n$ and $u \in \mathbb{R}^n$. Then A, L are matrices,

and $\|\cdot\|$ denotes the Euclidean norm. We have:

$$\|Lu\|_2^2 = u^T L^T Lu, \quad \|Au-b\|_2^2 = (Au-b)^T(Au-b) = u^T A^T Au - 2u^T A^T b + b^T b.$$

The general solution $f(u) = \min$ and $g(u) = 0$ satisfies:

$$f'(u) + g'(u)^T \lambda = 0, \quad g(u) = 0,$$

where λ is a Lagrange multiplier. We then obtain the following two conditions:

$$2L^T Lu + (2A^T Au - 2A^T b)^T \lambda = 0, \quad \lambda \in \mathbb{R}, \quad \|Au-b\|_2^2 - \delta^2 = 0.$$

Thus, we have $(L^T L + \lambda A^T A)u = \lambda A^T b$. For a fixed λ, this is the solution to a minimization problem without constraints:

$$\|Lu\|_2^2 + \lambda\|Au-b\|_2^2 = \min, \quad \text{so for } \alpha = \frac{1}{\lambda}, \quad \text{we have : } \|Au-b\|_2^2 + \alpha\|Lu\|_2^2 = \min.$$

Theorem 1.17 – Tikhonov Regularization

Let $Au = a * u$ and $Lu = u^{(p)}$. The solution to the minimization problem for $\alpha > 0$ as a given regularization parameter is:

$$\|a * u - b\|_2^2 + \alpha\|u^{(p)}\|_2^2 = \min,$$

for u a 2π-periodic function with a square-integrable p-th derivative, and is given by the Fourier coefficients:

$$\hat{u}(n) = \begin{cases} \frac{|\hat{a}(n)|^2}{|\hat{a}(n)|^2 + \alpha n^{2p}} \cdot \frac{\hat{b}(n)}{\hat{a}(n)} & \text{if } \hat{a}(n) \neq 0, \\ 0 & \text{if } \hat{a}(n) = 0. \end{cases}$$

We define the regularization filter $\Phi_\alpha(n)$:

$$\Phi_\alpha(n) := \frac{|\hat{a}(n)|^2}{|\hat{a}(n)|^2 + \alpha n^{2p}}.$$

Proof. By the Parseval formula, the regularization problem is equivalent to:

$$\sum_{n=-\infty}^{\infty} \underbrace{\left(\left| \hat{a}(n)\hat{u}(n) - \hat{b}(n) \right|^2 + \alpha n^{2p} |\hat{u}(n)|^2 \right)}_{=:\Lambda_n} = \min,$$

where we have used the convolution theorem and the fact that $\widehat{u^{(p)}}(n) = (in)^p \hat{u}(n)$. The expression becomes minimal if each individual summand is minimized.

For the n-th summand, which we define as Λ_n, we compute:

$$\begin{aligned} \Lambda_n &= |\hat{a}(n)|^2 |\hat{u}(n)|^2 - \hat{a}(n)\hat{u}(n)\overline{\hat{b}(n)} - \overline{\hat{a}(n)\hat{u}(n)}\hat{b}(n) + |\hat{b}(n)|^2 + \alpha n^{2p}|\hat{u}(n)|^2 \\ &= \underbrace{\left(|\hat{a}(n)|^2 + \alpha n^{2p}\right)}_{=:r} \cdot \underbrace{|\hat{u}(n)|^2}_{=:z^2} - 2\mathrm{Re}\left(\underbrace{\overline{\hat{u}(n)}}_{=:\bar{z}} \underbrace{\overline{\hat{a}(n)} \cdot \hat{b}(n)}_{=:s} \right) + |\hat{b}(n)|^2. \end{aligned}$$

Thus, we must have:

$$r|z|^2 - 2\mathrm{Re}(\bar{z}s) = \min, \quad \text{or for} \quad q = \frac{s}{r} \quad \text{then applies} \quad |z|^2 - 2\mathrm{Re}(\bar{z}q) = \min$$

but due to quadratic completion it applies:

$$|z|^2 - 2\mathrm{Re}(\bar{z}q) \geq |z|^2 - 2|z| \cdot |q| + |q|^2 - |q|^2 \geq -|q|^2$$

i.e.

$$\hat{u}(n) = \frac{|\hat{a}(n)|^2}{|\hat{a}(n)|^2 + \alpha n^{2p}} \frac{\hat{b}(n)}{\hat{a}(n)}.$$

□

1.6 Numerical Deconvolution, Smoothing of Measured Data

Remark 1.26 – Motivation (Problem Statement)

We have the same assumptions as in the previous section and the problem $a * u + \epsilon = b$ is given, where a, u, and b are 2π-periodic and $\epsilon \in L^2$. We are given the conditions:

$$\|u^{(p)}\|_{L^2} = \min, \quad \|a \cdot u - b\|_{L^2} \leq \delta.$$

Now, b is measured at discrete points $x_j = \frac{2\pi}{N} j$, so we replace b with the trigonometric interpolation polynomial b_N, leading to the following conditions:

$$\|u_N^{(p)}\|_{L^2} = \min, \quad \|a * u_N - b_N\|_{L^2} \leq \delta.$$

Definition 1.7 – Regularization

We choose $\alpha > 0$ and obtain the regularization problem:

$$\|a * u_N - b_N\|_{L^2}^2 + \alpha \|u_N^{(p)}\|_{L^2}^2 = \min$$

among all trigonometric polynomials $u_N(x)$:

$$u_N(x) = \sum_{n=-N/2}^{N/2} {}' \, \hat{u}_N(n) e^{inx}.$$

From the theorem in Section 5, we know that:

$$\hat{u}_N(n) = \Phi_\alpha(n) \frac{\hat{b}_N(n)}{\hat{a}(n)}, \quad n = -\frac{N}{2}, \ldots, \frac{N}{2} - 1.$$

Algorithm 1.2 – Practical Computation

We are given $b(x_j)$ for $j = 0, \ldots, N-1$, as well as $a(x)$ or $\hat{a}(n)$.

1. Compute with the FFT, as in Section 4:

$$\left(\hat{b}_N(n)\right)_{n=-\frac{N}{2}}^{\frac{N}{2}-1} = \frac{1}{N} \overline{\mathcal{F}_N}(b(x_j))_{j=0}^{N-1}$$

   with a total of $N \log_2 N$ operations.

2. The Fourier coefficients of the apparatus function $\hat{a}(n)$ are either given (often only $\hat{a}(n)$ is provided, not a), or we approximate $\hat{a}_M(n)$ with $M \geq N$, potentially even $M \gg N$.

3. Then calculate:

$$\hat{u}_N(n) = \frac{\overline{\hat{a}(n)} \hat{b}_N(n)}{|\hat{a}(n)|^2 + \alpha n^{2p}}, \quad n = -\frac{N}{2}, \ldots, \frac{N}{2} - 1.$$

4. Next, compute the discrete Fourier transform using FFT:

$$(u_N(x_j))_{j=0}^{N-1} = \mathcal{F}_N(\hat{u}_N(n))_{n=-\frac{N}{2}}^{\frac{N}{2}-1}.$$

   with $N \log_2 N$ operations.

Remark 1.27 – Choice of Regularization Parameter

We want to know how to determine or approximate the regularization parameter α. If the estimate (this term is also called the variance):

$$\delta \approx \left(\frac{1}{N} \sum_{j=0}^{N-1} |\varepsilon(x_j)|^2 \right)^{\frac{1}{2}} = \|\varepsilon\|_{L^2}$$

is known, we start with some α and compute (using the Parseval formula):

$$d_\alpha^2 = \|a * u_N - b_N\|_{L^2}^2 = \sum_{n=-\frac{N}{2}}^{\frac{N}{2}} \prime |\hat{a}(n)\hat{u}_N(n) - \hat{b}_N(n)|^2 = \sum_{n=-\frac{N}{2}}^{\frac{N}{2}} \prime (1 - \Phi_\alpha(n))^2 |\hat{b}_N(n)|^2.$$

We then choose α such that $d_\alpha \approx \delta$. Note that $\alpha \mapsto d_\alpha$ is monotonically increasing, so this process can be iterated relatively easily. At the optimal α, we compute $\hat{u}_N(n)$, and then, using FFT, obtain $u_N(x_j)$.

Remark 1.28

If the variance is unknown, the procedure makes no sense. However, statistical methods can determine an optimal regularization parameter λ. This can be found in the literature under the term "generalized cross-validation."

Remark 1.29 – Smoothing of Data

We have measured values $b(x_j)$ and a variance of the measurement error that is approximately δ. We are looking for a trigonometric polynomial u_N with:

$$\|u_N - b_N\|_{L^2}^2 = \frac{1}{N} \sum_{j=0}^{N-1} |u_N(x_j) - b(x_j)|^2 \leq \delta^2 = \min, \quad \|u_N^{(p)}\|_{L^2} = \min .$$

Using our formula, we can compute this directly. For the special case $\hat{a}(n) = 1$, we have $a * u = u$, which corresponds to convolution with the Dirac delta function. For $p = 2$, we obtain:

$$\hat{u}_N(n) = \frac{1}{1 + \alpha n^4} \hat{b}_N(n).$$

We note that the high-frequency components of $\hat{b}_N(n)$ (for large n) are filtered out by this formula, which results in smoothing. By the inverse Fourier transform, we eventually obtain the desired u_N.

Remark 1.30 – Approach for Non-Periodic Data

If the data are not periodic, we subtract a fitting line such that it fluctuates around a level and then extend it periodically.

Remark 1.31

An alternative approach would be to smooth the data using splines instead of Fourier transforms.

Remark 1.32 – Differentiation of noisy data

We want to find the derivative of the data, i.e., $u = b'$. Thus, we have:

$$\int_0^x u(t)\,dt = b(x) - b(0).$$

We have $\hat{u}(n) = in\hat{b}(n)$, so:

$$\underbrace{\frac{1}{in}}_{=:\hat{a}(n)}\ \hat{u}(n) = \hat{b}(n)$$

This leads to a new minimization problem:

$$\sum_{n=-\frac{N}{2}}^{\frac{N}{2}-1} \left| \frac{1}{in}\hat{u}_N(n) - \hat{b}_N(n) \right|^2 \leq \delta^2, \quad \|{u_N}''\|_{L^2}^2 = \min.$$

For the special case $\hat{a}(n) = \frac{1}{in}$, we get:

$$\hat{u}_N(n) = \frac{n^{-2}}{n^{-2} + \alpha n^4} in\hat{b}_N(n) = \frac{in}{1 + \alpha n^6}\hat{b}_N(n).$$

Chapter 2

Eigenvalue Problems

In this chapter, n denotes a positive integer.

Review – Left/right eigenvectors

Let $A \in \mathbb{C}^{n\times n}$. If there is $(\lambda, v) \in \mathbb{C} \times \mathbb{C}^n$ such that $Av = \lambda v$, then λ is called an *eigenvalue of* A and v an *eigenvector of* A *associated to* λ. If there is $(\kappa, u) \in \mathbb{C} \times \mathbb{C}^n$ such that $u^*A = \kappa u^*$, then κ is also called an eigenvalue of A and u an *eigenvector of* A *associated to* κ. When it is not specified, an eigenvector usually refers to a right eigenvector.

Question
Why don't we need to distinguish left and right eigenvalues?

Answer
Due to the fact that the determinant of a matrix B is equal to the determinant of its transpose, by taking $B = A - \lambda I$ we get $\chi_A(\lambda) = \det(A - \lambda I) = \det(A^T - \lambda I) = \chi_{A^T}(\lambda)$, where χ_A is the characteristic polynomial of A. Notice that $u^*A = \kappa u^* \iff A^T\bar{u} = \kappa\bar{u}$, hence κ is an eigenvalue of A^T. In other words, κ is also an eigenvalue of A. Hence, all left eigenvalues are right eigenvalues, and there is no need to distinguish them.

Note
The eigenvectors are conventionally taken to be of unit ℓ^2 norm, since for any $c \in \mathbb{C}$,

$$Av = \lambda v \iff A(cv) = \lambda(cv).$$

Taking $c = 1/\|v\|_2$, the eigenvector $u = cv$ is of unit ℓ^2 norm.

In this chapter, we will generally assume that eigenvectors are normalized.

Review – Diagonalization

A matrix $A \in \mathbb{C}^{n\times n}$ is said to be *diagonalizable* if there is an invertible matrix $P \in \mathbb{C}^{n\times n}$ and a diagonal matrix $\Lambda \in \mathbb{C}^{n\times n}$ such that

$$P^{-1}AP = \Lambda.$$

2.1 Fundamentals

Motivation
There are numerous applications where eigenvalues are required:

1. *In mechanics (physics), for example, one is interested in the natural vibrations of membranes. If $u(x)$ represents the deflection on a domain Ω, then we require that*

$$-\Delta u = \lambda u \quad in \quad \Omega, \quad u = 0 \quad on \quad \partial\Omega.$$

For a grid over Ω, we obtain the discretization

$$Av = \lambda v.$$

2. *In biology, the Lotka-Volterra predatorprey model is frequently used to describe the dynamics of two interacting species:*

$$\begin{cases} \dfrac{dx}{dt} = \alpha x - \beta xy \\ \dfrac{dy}{dt} = -\gamma y + \delta xy, \end{cases}$$

where x (resp. y) is the population density of prey (resp. predator). The system's equilibrium is obtained when $\frac{dx}{dt} = \frac{dy}{dt} = 0$, which yields two points:

$$\{x = 0,\, y = 0\} \qquad or \qquad \left\{x = \frac{\gamma}{\delta},\, y = \frac{\alpha}{\beta}\right\}.$$

The stability of fixed points is studied by looking at eigenvalues of a certain matrix.

- $\{x = 0,\, y = 0\}$*: there are always one positive and one negative eigenvalues, hence the equilibrium is unstable*
- $\left\{x = \frac{\gamma}{\delta},\, y = \frac{\alpha}{\beta}\right\}$*: there are always two complex conjuguate eigenvalues, hence the population oscillates with time around that point, and the system is stable in a certain sense.*

3. *Search engines: "The 25,000,000,000 Dollar Eigenvalue Problem", which we will discuss later in the context of Google's mechanism.*

2.1.1 Review (Characteristic Polynomial)

Review – Characteristic polynomial

For $A \in \mathbb{C}^{n\times n}$, the *characteristic polynomial* χ_A of A is defined by $\chi_A(\lambda) := \det(A - \lambda I)$.

The condition $Av = \lambda v$ is equivalent to $\det(A - \lambda I) = \chi_A(\lambda) = 0$. One might consider first computing the characteristic polynomial and then finding its roots to obtain the eigenvalues.

Note
The equivalence is shown as follows:

$$(A - \lambda I)v = 0 \text{ for some } v \neq 0 \iff (A - \lambda I) \text{ is singular} \iff \det(A - \lambda I) = 0.$$

Example 2.1 – Poor Conditioning

Let $A = \mathrm{diag}(10, 11, \ldots, 16)$ be a 7×7 matrix. Here, it is clear what the eigenvalues are, and the characteristic polynomial is:

$$\chi_A(\lambda) = (10 - \lambda)(11 - \lambda)\cdots(16 - \lambda) \quad = -\lambda^7 + 91\lambda^6 - 3535\lambda^5 + \ldots - 31813200\lambda + 57657600.$$

If one computes the roots of χ_A in single precision (i.e. $\texttt{eps} = 10^{-8}$), the result[a] is:

$$9.97,\ 11.31 - 0.30i,\ 11.37 + 0.33i,\ 13.47 - 0.76i,\ 13.57 + 0.76i,\ 15.51 - 0.09i,\ 15.80 + 0.06i.$$

This does not correspond to the actual eigenvalues. The problem is that computing the roots of a polynomial from its coefficients is a poorly conditioned problem.

[a] Results obtained via the Julia programming language with Float32 precision (giving between 6 and 9 digits of precision)

Remark 2.1 – How to explain poor conditioning of the root-finding problem?

Let

$$p(\lambda) = \sum_{k=0}^{n} a_k \lambda^k,$$

and we suppose its roots are simple (i.e. they are all distinct). The coefficients $\{a_k\}$ are the "real" ones, but numerically they are only known up to a certain precision η: the computer only sees coefficients $\{a_k(1+\varepsilon_k)\}$ for some $|\varepsilon_k| \leq \eta$, and the polynomial seen by the computer is

$$p(\lambda, \eta) = \sum_{k=0}^{n} (a_k + a_k \varepsilon_k)\lambda^k = \sum_{k=0}^{n} \left(a_k + \underbrace{a_k \frac{\varepsilon_k}{\eta}}_{=:b_k} \eta \right) \lambda^k = p(\lambda) + q(\lambda)\eta$$

with $q(\lambda) = \sum_{k=0}^{n} b_k \lambda^k$ and $|b_k| \leq |a_k|$. We study the roots $\lambda(\eta)$ of $p(\lambda, \eta) = p(\lambda) + \eta q(\lambda)$ as a function of η. Let $\lambda(0) = \lambda^*$ be a simple root of p. We consider the differentiable function $\eta \mapsto \lambda(\eta)$ defined by $p(\lambda(\eta), \eta) = 0$ for all η with $|\eta| \leq \eta_0$. It exists and is unique by the implicit function theorem, hence:

$$\underbrace{\frac{\partial p}{\partial \lambda}(\lambda(\eta), \eta)}_{p'(\lambda(\eta))\lambda'(\eta) + \mathcal{O}(\eta)} + \underbrace{\frac{\partial p}{\partial \eta}(\lambda(\eta), \eta)}_{q(\lambda(\eta))} = 0.$$

Thus,

$$\lambda'(\eta) \approx -\frac{q(\lambda(\eta))}{p'(\lambda(\eta))}, \quad \lambda(\eta) \approx \lambda^* + \eta \lambda'(0),$$

and the relative error is:

$$\frac{|\lambda(\eta) - \lambda^*|}{|\lambda^*|} \approx \frac{|\eta \lambda'(0)|}{|\lambda^*|} \approx |\eta| \cdot \left| \frac{q(\lambda^*)}{\lambda^* p'(\lambda^*)} \right|.$$

The term $\left| \frac{q(\lambda^*)}{\lambda^* p'(\lambda^*)} \right|$ can become very large. Coming back to Example 2.1, we have

$$q(\lambda) = \frac{1}{\eta} \left(-\varepsilon_0 \lambda^7 + 91\varepsilon_1 \lambda^6 + \cdots - 31813200\varepsilon_6 \lambda + 57657600\varepsilon_7 \right),$$

and thus for $\lambda^* = 10$ we obtain

$$\begin{aligned} |q(\lambda^*)| &\leq \frac{1}{\eta} \left(|\varepsilon_0 (\lambda^*)^7| + |91\varepsilon_1 (\lambda^*)^6| + \cdots + |31813200\varepsilon_6 \lambda^*| + |57657600\varepsilon_7| \right) \\ &\leq |(\lambda^*)^7| + |91(\lambda^*)^6| + \cdots + |31813200\lambda^*| + |57657600| \\ &\leq 10^7 + 91 \cdot 10^7 + \cdots + 31.8132 \cdot 10^7 + 5,76576 \cdot 10^7 \\ &\approx 10^8, \end{aligned}$$

and $|p'(\lambda^*)| = 720 \approx 10^3$. So the relative error is $\eta \cdot 10^4$, which means that we lose about four significant digits in the decimal expansion of the roots. Therefore, it is numerically unreasonable to compute the coefficients of the characteristic polynomial with a coarse precision.

Proposition 2.1 – Obtaining eigenvectors by similarity transformations

Let $A \in \mathbb{C}^{n \times n}$, and $T \in \mathbb{C}^{n \times n}$ invertible. If $B = T^{-1}AT$, then

$$Av = \lambda v \iff TBT^{-1}v = \lambda v \iff B(T^{-1}v) = \lambda (T^{-1}v).$$

Thus, B and A have the same eigenvalues, and v is an eigenvector of A if and only if $T^{-1}v$ is an eigenvector of B.

Definition 2.1 – Unitary/orthogonal matrix

The matrix $U \in \mathbb{C}^{n\times n}$ is *unitary* if $U^*U = UU^* = I$, where $U^* = \overline{U^T}$. In other words, $U^{-1} = U^*$. If U is real, then $U^* = U^T$ and U is said to be *orthogonal*.

Exercise
Let $U_1, \dots, U_k \in \mathbb{C}^{n\times n}$ unitary matrices. Show that $V = U_1 \cdots U_k$ is a unitary matrix.

Answer
We have $V^* = U_k^* \cdots U_1^*$ hence, owing to the unitary character of each matrix U_ℓ,

$$V^*V = U_k^* \cdots U_1^* U_1 \cdots U_k = U_k^* \cdots U_2^* U_2 \cdots U_k = \cdots = I,$$

and

$$VV^* = U_1 \cdots U_k U_k^* \cdots U_1^* = U_1 \cdots U_{k-1} U_{k-1}^* \cdots U_1^* = \cdots = I.$$

Theorem 2.1 – Schur's normal form (1909)

Let $A \in \mathbb{C}^{n\times n}$, there is a unitary matrix U such that

$$U^*AU = \begin{pmatrix} \lambda_1 & \star & \dots & \star \\ 0 & \lambda_2 & \ddots & \vdots \\ \vdots & \ddots & \ddots & \star \\ 0 & \dots & 0 & \lambda_n \end{pmatrix}$$

is an upper triangular matrix. This form is called Schur's normal form.

Proof. The characteristic polynomial $\chi_A(\lambda)$ has a root $\lambda_1 \in \mathbb{C}$, which is an eigenvalue of A. Thus, there exists an eigenvector $0 \neq v_1 \in \mathbb{C}^n$ such that $Av_1 = \lambda_1 v_1$. We can assume without loss of generality that $\|v_1\|_2 = 1$. We now construct (using Gram-Schmidt) a matrix $V_1 = (v_1, v_2, \dots, v_n)$ with $v_2, \dots, v_n$ chosen so that $v_1, v_2, \dots, v_n$ form an orthonormal basis (ONB) of $\mathbb{C}^n$. In other words, V_1 is a unitary matrix.

We now consider

$$AV_1 = (Av_1, Av_2, \dots, Av_n) = V_1 \begin{pmatrix} \lambda_1 & \star \\ 0 & \hat{A} \end{pmatrix}$$

We proceed in the same way with the matrix $\hat{A} \in \mathbb{C}^{(n-1)\times(n-1)}$: there exists an eigenvalue $\lambda_2 \in \mathbb{C}$ of $\hat{A}$ and an associated unit eigenvector $\hat{v}_2$, so one can construct an ONB of $\mathbb{C}^{n-1}$ by the Gram-Schmidt procedure. This yields a unitary matrix $V_2 \in \mathbb{C}^{(n-1)\times(n-1)}$ such that

$$\hat{A}V_2 = V_2 \begin{pmatrix} \lambda_2 & \star \\ 0 & \hat{\hat{A}} \end{pmatrix} \iff \hat{A} = V_2 \begin{pmatrix} \lambda_2 & \star \\ 0 & \hat{\hat{A}} \end{pmatrix} V_2^*.$$

Hence,

$$AV_1 = V_1 \begin{pmatrix} \lambda_1 & & \star \\ 0 & V_2 \begin{pmatrix} \lambda_2 & \star \\ 0 & \hat{\hat{A}} \end{pmatrix} V_2^* \end{pmatrix} = V_1 \begin{pmatrix} 1 & 0 \\ 0 & V_2 \end{pmatrix} \begin{pmatrix} \lambda_1 & \star & \star \\ 0 & \lambda_2 & \star \\ 0 & 0 & \hat{\hat{A}} \end{pmatrix} \begin{pmatrix} 1 & 0 \\ 0 & V_2^* \end{pmatrix}.$$

Finally, since $\begin{pmatrix} 1 & 0 \\ 0 & V_2^* \end{pmatrix}$ is a unitary matrix, we get

$$AV_1 \begin{pmatrix} 1 & 0 \\ 0 & V_2 \end{pmatrix} = V_1 \begin{pmatrix} 1 & 0 \\ 0 & V_2 \end{pmatrix} \begin{pmatrix} \lambda_1 & \star & \star \\ 0 & \lambda_2 & \star \\ 0 & 0 & \hat{\hat{A}} \end{pmatrix}.$$

By repeating this process, we get a matrix $U \in \mathbb{C}^{(n-1)\times(n-1)}$ of the form

$$U = V_1 \begin{pmatrix} 1 & 0 \\ 0 & V_2 \end{pmatrix} \begin{pmatrix} 1 & 0 & 0 \\ 0 & 1 & 0 \\ 0 & 0 & V_3 \end{pmatrix} \dots \begin{pmatrix} 1 & 0 & \dots & 0 & 0 \\ 0 & 1 & \ddots & \vdots & \vdots \\ \vdots & \ddots & \ddots & 0 & \vdots \\ 0 & \dots & 0 & 1 & 0 \\ 0 & \dots & \dots & 0 & V_n \end{pmatrix},$$

where each matrix $V_k \in \mathbb{C}^{(n-k+1)\times(n-k+1)}$ is unitary. The matrix U is unitary as a product of unitary matrices, and it satisfies

$$AU = U \begin{pmatrix} \lambda_1 & \star & \dots & \star \\ 0 & \lambda_2 & \ddots & \vdots \\ \vdots & \ddots & \ddots & \star \\ 0 & \dots & 0 & \lambda_n \end{pmatrix}$$

The claimed result is obtained after left-multiplying by U^*. □

Review 2.1 – Hermitian and symmetric matrices

A matrix $A \in \mathbb{C}^{n\times n}$ is said to be *Hermitian* if $A^* = A$ and *skew-Hermitian* if $A^* = -A$, where $A^* = \bar{A}^T$. A matrix $A \in \mathbb{R}^{n\times n}$ is said to be *symmetric* if $A^T = A$, and *skew-symmetric* if $A^T = -A$.

Note

In some contexts, an Hermitian/symmetric matrix can also be called self-adjoint *when seen as a linear operator.*

Definition 2.2 – Normal matrix

A matrix A is *normal* if $AA^* = A^*A$ (when $A \in \mathbb{C}^{n\times n}$) or if $AA^T = A^TA$ (when $A \in \mathbb{R}^{n\times n}$).

Lemma 2.1

A normal upper triangular matrix is diagonal.

Proof. Let $R \in \mathbb{C}^{n\times n}$ a normal upper triangular matrix

$$R = \begin{pmatrix} r_{11} & \dots & r_{1,n} \\ 0 & & \\ \vdots & & R_1 \\ 0 & & \end{pmatrix},$$

with $R_1 \in \mathbb{C}^{(n-1)\times(n-1)}$ an upper triangular matrix. Since R is normal, $R^*R = RR^*$ and we get

$$\begin{pmatrix} \overline{r_{1,1}} & 0 & \dots & 0 \\ \vdots & & R_1^* & \\ \overline{r_{1,n}} & & & \end{pmatrix} \begin{pmatrix} r_{1,1} & \dots & r_{1,n} \\ 0 & & \\ \vdots & & R_1 \\ 0 & & \end{pmatrix} = \begin{pmatrix} r_{1,1} & \dots & r_{1,n} \\ 0 & & \\ \vdots & & R_1 \\ 0 & & \end{pmatrix} \begin{pmatrix} \overline{r_{1,1}} & 0 & \dots & 0 \\ \vdots & & R_1^* & \\ \overline{r_{1,n}} & & & \end{pmatrix},$$

i.e.

$$\begin{pmatrix} |r_{1,1}|^2 & \overline{r_{1,1}}r_{1,2} & \dots & \overline{r_{1,1}}r_{1,n} \\ \overline{r_{1,2}}r_{1,1} & & & \\ \vdots & & (\overline{r_{1,i+1}}r_{1,j+1})_{i,j=1}^{n-1} + R_1^*R_1 & \\ \overline{r_{1,n}}r_{1,1} & & & \end{pmatrix} = \begin{pmatrix} \sum_{j=1}^n |r_{1,j}|^2 & \begin{pmatrix} r_{1,2} & \dots & r_{1,n} \end{pmatrix} R_1^* \\ R_1 \begin{pmatrix} \overline{r_{1,2}} \\ \vdots \\ \overline{r_{1,n}} \end{pmatrix} & R_1R_1^* \end{pmatrix}$$

The component at index (1, 1) yields $|r_{1,1}|^2 = \sum_{j=1}^{n} |r_{1,j}|^2$, i.e. $r_{1,j} = 0$ for $j = 2, \ldots, n$. Hence,

$$\begin{pmatrix} |r_{1,1}|^2 & 0 & \ldots & 0 \\ 0 & & & \\ \vdots & & R_1^* R_1 & \\ 0 & & & \end{pmatrix} = \begin{pmatrix} |r_{1,1}|^2 & 0 & \ldots & 0 \\ 0 & & & \\ \vdots & & R_1 R_1^* & \\ 0 & & & \end{pmatrix},$$

i.e. the matrix R_1 is normal. In other words, the first row of a normal upper triangular matrix has all its off-diagonal terms equal to zero. Since R_1 is also a normal upper triangular matrix, by induction we obtain that R is a diagonal matrix. □

Theorem 2.2 – Spectral theorem

A matrix $A \in \mathbb{C}^{n\times n}$ is normal if and only if there is a unitary $U \in \mathbb{C}^{n\times n}$ such that

$$U^* A U = \begin{pmatrix} \lambda_1 & & 0 \\ & \ddots & \\ 0 & & \lambda_n \end{pmatrix}.$$

Proof. [⇒] The Schur normal form of A yields a unitary matrix U and an upper triangular matrix R such that $U^*AU =: R$. Since A is normal,

$$R^*R = U^*A^*UU^*AU = U^*A^*AU = U^*AA^*U = U^*AUU^*A^*U = RR^*.$$

The matrix R is normal and upper triangular, hence diagonal.
[⇐] One has $A = U \operatorname{diag}(\lambda_1, \ldots, \lambda_n)U^*$, thus

$$A^*A = U \operatorname{diag}(\overline{\lambda_1}, \ldots, \overline{\lambda_n})U^*U \operatorname{diag}(\lambda_1, \ldots, \lambda_n)U^* = U \operatorname{diag}(|\lambda_1|^2, \ldots, |\lambda_n|^2)U^*,$$

and

$$AA^* = U \operatorname{diag}(\lambda_1, \ldots, \lambda_n)U^*U \operatorname{diag}(\overline{\lambda_1}, \ldots, \overline{\lambda_n})U^* = U \operatorname{diag}(|\lambda_1|^2, \ldots, |\lambda_n|^2)U^*.$$

The matrix A is then normal. □

Question
Show that the λ_j in the spectral theorem are the eigenvalues of A.

Answer
Let κ an eigenvalue of A, it is a root of the characteristic polynomial χ_A: $\chi_A(\kappa) = 0$. On the one hand

$$\det(U^*AU - \kappa I) = \det(U^*AU - \kappa U^*U) = \det(U^*)\chi_A(\kappa)\det(U) = \chi_A(\kappa),$$

where the last equality is due to U being unitary hence $\det(U^*)\det(U) = \det(U^{-1})\det(U) = 1$. On the other hand,

$$\det(U^*AU - \kappa I) = \det(\operatorname{diag}\{\lambda_1 - \kappa, \ldots, \lambda_n - \kappa\}).$$

Thus, κ is a root of χ_A if and only if $\det(\operatorname{diag}\{\lambda_1 - \kappa, \ldots, \lambda_n - \kappa\}) = 0$. The determinant of a diagonal matrix is equal to the product of its diagonal elements, hence it is zero if and only if at least one element is zero, i.e. if there is a j such that $\kappa = \lambda_j$. This shows $\operatorname{sp}(A) \subset \{\lambda_1, \ldots, \lambda_n\}$. It is clear that $\{\lambda_1, \ldots, \lambda_n\} \subset \operatorname{sp}(A)$, since $\chi_A(\lambda_j) = \det(U^*AU - \lambda_j I) = 0$.

Theorem 2.3 – Jordan normal form

For every $A \in \mathbb{C}^{n\times n}$, there is an invertible matrix T such that

$$T^{-1}AT = J = \begin{pmatrix} J_{n_1}(\lambda_1) & 0 & \dots & 0 \\ 0 & \ddots & \ddots & \vdots \\ \vdots & \ddots & \ddots & 0 \\ 0 & \dots & 0 & J_{n_k}(\lambda_k) \end{pmatrix},$$

with λ_k the eigenvalues of A and $n_k \in \mathbb{N}^*$. The Jordan blocks are given by

$$J_m(\lambda) = \begin{pmatrix} \lambda & 0 & \dots & \dots & 0 \\ 1 & \lambda & \ddots & & \vdots \\ 0 & 1 & \ddots & \ddots & \vdots \\ \vdots & \ddots & \ddots & \ddots & 0 \\ 0 & \dots & 0 & 1 & \lambda \end{pmatrix} \in \mathbb{C}^{m\times m}.$$

The number of Jordan blocks associated to a given λ is given by $\dim\ker(A-\lambda I)$, and the sum of the sizes of all Jordan blocks associated to λ is given by its algebraic multiplicity (i.e. the multiplicity of λ as a root of χ_A).

Proof. Linear Algebra II. □

Remark 2.2

On the sub-diagonal of a Jordan block, there can be entries $\varepsilon \neq 0$ instead of ones. Sometimes, the ones of the sub-diagonal are instead on the sup-diagonal.

2.2 Conditioning of the Eigenvalue Problem

Motivation

Let $A = (a_{ij})$ be a given matrix. Due to rounding errors, we can only work with a perturbed matrix $\tilde{A} = (\tilde{a}_{ij})$ such that $\tilde{a}_{ij} = a_{ij}(1+\varepsilon_{ij})$ with $|\varepsilon_{ij}| \leq \eta$, where $\eta > 0$ is the numerical precision. Thus, we can write:

$$\tilde{A} = A + \eta \cdot C, \quad c_{ij} = a_{ij}\frac{\varepsilon_{ij}}{\eta}, \ |c_{ij}| \leq |a_{ij}|.$$

We consider $A(\eta) = A + \eta C$ for small η. We want to find an estimate for the eigenvalues like:

$$|\lambda(\eta) - \lambda(0)| \leq \text{const.} \cdot \eta.$$

Is this even possible, and if so, with which constant? We will see with the following theorem that this depends on a constant that depends on A.

Lemma 2.2

Let $A \in \mathbb{C}^{n\times n}$ and $\lambda \in \mathbb{C}$ a simple root of χ_A. Let $u, v \in \mathbb{C}^n$ be respectively left and right eigenvectors of A associated to λ. Then

$$u^*v \neq 0.$$

Proof. (Done in Exercise). Since λ is a simple root of χ_A, its algebraic multiplicity is one. With respect to the Jordan form, it means that the sum of the sizes of all Jordan blocks associated to λ is one. Hence, the Jordan form of A is

$$T^{-1}AT = \begin{pmatrix} \lambda & 0 \\ 0 & J' \end{pmatrix},$$

where $J' \in \mathbb{C}^{(n-1)\times(n-1)}$ is a matrix in Jordan form and $T \in \mathbb{C}^{n\times n}$ is an invertible matrix. Since

$$AT = T\begin{pmatrix}\lambda & 0\\ 0 & J'\end{pmatrix} \quad \text{and} \quad T^{-1}A = \begin{pmatrix}\lambda & 0\\ 0 & J'\end{pmatrix}T^{-1},$$

we deduce that the first column of T is a multiple of v, and that the first row of T^{-1} is a multiple of u^*. Thus, there are nonzero complex coefficients α, β such that

$$v = \alpha T_{\cdot,1} \quad \text{and} \quad u = \beta(T^{-*})_{\cdot,1}.$$

We can write $T = (v/\alpha, T_1)$ and $T^{-*} = (u/\beta, Z_1)$ for some matrices $T_1, Z_1 \in \mathbb{C}^{n\times(n-1)}$. By definition,

$$I = T^{-1}T = (T^{-*})^*T = \begin{pmatrix}u^*/\bar{\beta}\\ Z_1^*\end{pmatrix}\begin{pmatrix}v/\alpha & T_1\end{pmatrix} = \begin{pmatrix}u^*v/(\alpha\bar{\beta}) & u^*T_1/\bar{\beta}\\ Z_1^*v/\alpha & Z_1^*T_1\end{pmatrix}.$$

Looking at the component at index (1, 1), we get $u^*v = \alpha\bar{\beta} \neq 0$. □

Definition 2.3 – Condition number of an eigenvalue

Let $A \in \mathbb{C}^{n\times n}$ and $\lambda \in \mathbb{C}$ an eigenvalue. Let u (resp. v) denote a left (resp. right) eigenvector associated to λ. The quantity $1/|u^*v|$ is called the *condition number of the eigenvalue* λ.

A "well-conditioned" matrix means that small perturbations only result in small changes. For a given matrix $A \in \mathbb{C}^{n\times n}$, if the condition number of an eigenvalue λ is large it means that small perturbations of A may result in large perturbations of λ. The condition number of an eigenvalue can be understood as a measure of the "continuity" of this eigenvalue depending on the size of the perturbation.

Theorem 2.4 – Error estimate for a simple eigenvalue

Let $A \in \mathbb{C}^{n\times n}$ and λ a simple root of the characteristic polynomial χ_A. For small ε, an eigenvalue of $A(\varepsilon) = A + \varepsilon C$ satisfies the following relation:

$$\lambda(\varepsilon) = \lambda + \varepsilon\frac{u^*Cv}{u^*v} + \mathcal{O}(\varepsilon^2)$$

where v is an eigenvector of A associated to λ and u is a left eigenvector of A associated to λ.

Proof. Using Lemma 2.2, we know that $u^*v \neq 0$. We start by showing that there is a differentiable eigenvalue $\lambda(\varepsilon)$ and an associated normalized eigenvector $v(\varepsilon)$ depending on ε such that $\lambda(0) = \lambda$ and $v(0) = v$. This is done using the implicit function theorem.

For that, we consider the function

$$F(\varepsilon, \kappa, w) := \begin{pmatrix}(A(\varepsilon) - \kappa I)w\\ w^*w - 1\end{pmatrix}.$$

It is a C^1 function of the three variables, and $F(0, \lambda, v) = 0$ by definition of λ and v. We compute the Jacobian of F with respect to (κ, w), evaluated at the point $(\varepsilon = 0, \kappa = \lambda, w = v)$:

$$\left(D_{(\kappa,w)}F\right)(0, \lambda, v) = \left(D_\kappa F, D_{w_1}F, \ldots, D_{w_n}F\right)(0, \lambda, v) = \begin{pmatrix}-v & A - \lambda I\\ 0 & v^*\end{pmatrix}.$$

For the derivatives with respect to a complex variable $z = x + iy \in \mathbb{C}$, we have used the Wirtinger derivative: $\frac{\partial}{\partial z} = \frac{1}{2}\left(\frac{\partial}{\partial x} - \frac{\partial}{\partial y}\right)$. We now show that this matrix is nonsingular, i.e. that its kernel is the zero element. Assume the matrix is singular, then one can find $(x, y) \in (\mathbb{C}\times\mathbb{C}^n)\setminus\{0\}$ such that

$$\begin{pmatrix}-v & A - \lambda I\\ 0 & v^*\end{pmatrix}\begin{pmatrix}x\\ y\end{pmatrix} = 0 \quad \Longleftrightarrow \quad \begin{cases}-vx + (A - \lambda I)y = 0\\ v^*y = 0\end{cases}$$

Left-multiply the first equation by u^* to obtain

$$-xu^*v + u^*(A - \lambda I)y = 0.$$

However, since $u^*(A - \lambda I) = 0$ and $u^*v \neq 0$, we get $x = 0$. The equation $(A - \lambda I)y = 0$ means $y = \ker(A - \lambda I)$. But since λ is an eigenvalue of algebraic multiplicity one, it is also of geometric multiplicity one which means that $\dim \ker(A - \lambda I) = 1$. Since the kernel is a $\mathbb{C}$-linear space and we know $v \in \ker(A - \lambda I)$, we obtain $\ker(A - \lambda I) = \operatorname{span}\{v\} = \{\alpha v : \alpha \in \mathbb{C}\}$. Therefore there is $c \in \mathbb{C}$ such that $y = cv$. The equation $v^*y = 0$ yields $c = 0$ since $v^*v = 1$. Hence, there is no $(x, y) \in (\mathbb{C} \times \mathbb{C}^n) \setminus \{0\}$ such that

$$\begin{pmatrix} -v & A - \lambda I \\ 0 & v^* \end{pmatrix} \begin{pmatrix} x \\ y \end{pmatrix} = 0.$$

In other words, the matrix $\left(D_{(\kappa,w)}F\right)(0, \lambda, v)$ is invertible, i.e. its determinant is nonzero. The implicit function theorem now applies, and we get the existence of a unique differentiable function $\varphi(\varepsilon) = (\lambda(\varepsilon), v(\varepsilon)) \in \mathbb{C} \times \mathbb{C}^n$ such that $F(\varepsilon, \lambda(\varepsilon), v(\varepsilon)) = 0$ for $|\varepsilon|$ small enough, i.e.

$$(A(\varepsilon) - \lambda(\varepsilon)I)\, v(\varepsilon) = 0,\ \|v(\varepsilon)\|_2 = 1,\ v(0) = v,\ \lambda(0) = \lambda.$$

We have $A(\varepsilon)v(\varepsilon) = \lambda(\varepsilon)v(\varepsilon)$, i.e.

$$(A + \varepsilon C)(v + \varepsilon v'(0) + \mathcal{O}(\varepsilon^2)) = (\lambda + \varepsilon\lambda'(0) + \mathcal{O}(\varepsilon^2))(v + \varepsilon v'(0) + \mathcal{O}(\varepsilon^2)).$$

We multiply this out and compare the powers of ε:

$$\varepsilon^0 : Av = \lambda v, \quad \varepsilon^1 : Cv + Av'(0) = \lambda v'(0) + \lambda'(0)v.$$

We obtain: $(A - \lambda I)v'(0) = -Cv + \lambda'(0)v$. By left multiplying with u^* and using $u^*(A - \lambda I) = 0$, we obtain:

$$0 = -u^*Cv + \lambda' u^*v \Rightarrow \lambda' = \frac{u^*Cv}{u^*v}.$$

□

Remark 2.3

The error estimate for the (right) eigenvectors can be shown to be

$$v_j(\varepsilon) = v_j + \varepsilon \sum_{i=1, i\neq j}^{n} \frac{1}{\lambda_j - \lambda_i} \frac{u_i^* C v_j}{u_i^* v_i} v_i + \mathcal{O}(\varepsilon^2).$$

This is done in an Exercise.

Example 2.2

1. If A is normal, then the left and right eigenvectors are identical, because there is a U unitary (by the spectral theorem) such that:

   $$U^*AU = \operatorname{diag}(\lambda_1, \ldots, \lambda_n).$$

   After multiplying by U^* on the right, we get $U^*A = \operatorname{diag}(\lambda_1, \ldots, \lambda_n)U^*$, i.e. the left eigenvectors are in the matrix U. If we multiply instead by U on the left, we get $AU = U \operatorname{diag}(\lambda_1, \ldots, \lambda_n)$, i.e. the right eigenvectors are in the matrix U. Hence, left and right eigenvectors agree. When the eigenvalues are all simple roots of the characteristic polynomial, we get

   $$\frac{1}{|u^*v|} = 1$$

   for all eigenvalues, if u and v are normalized. In this case, the problem is well-conditioned.

We could even show here that not only

$$|\lambda(\varepsilon) - \lambda| \leq |\varepsilon| \cdot |v^* C v|$$

holds, but also

$$|\lambda(\varepsilon) - \lambda| \leq |\varepsilon| \|C\|_2.$$

2. If A is not normal, $u^* v$ can be arbitrarily small, e.g.,

$$A = \begin{pmatrix} 1 & \alpha \\ 0 & 2 \end{pmatrix}, \quad \lambda = 1, \quad v = \begin{pmatrix} 1 \\ 0 \end{pmatrix}, \quad u^* = \frac{1}{\sqrt{1+\alpha^2}} \begin{pmatrix} 1 & -\alpha \end{pmatrix}.$$

For a very large α, u is almost the second unit vector, and then:

$$u^* v = \frac{1}{\sqrt{1+\alpha^2}} \to 0.$$

Here, the problem would be very poorly conditioned.

Remark 2.4 – Consideration for multiple eigenvalues

(Details given in the tutorials). We cannot transfer the proof because the conditions for the implicit function theorem are not guaranteed. We consider here as an example the matrix with multiple Jordan blocks:

$$A = \begin{pmatrix} \lambda & 1 & 0 & \dots & 0 \\ 0 & \lambda & \ddots & \ddots & \vdots \\ \vdots & \ddots & \ddots & \ddots & 0 \\ \vdots & \dots & \ddots & \lambda & 1 \\ 0 & \dots & \dots & 0 & \lambda \end{pmatrix}$$

which represents a Jordan block of size $n \times n$. We consider the characteristic polynomial of $A + \varepsilon C$, where

$$C = \begin{pmatrix} 0 & & & \\ \vdots & & 0 & \\ 0 & & & \\ c & 0 & \dots & 0 \end{pmatrix}.$$

We have

$$\chi_{A+\varepsilon C}(\lambda)(x) = (\lambda - x)^n + \varepsilon(-1)^{n+1} c.$$

If $\lambda(\varepsilon)$ is an eigenvalue of $A + \varepsilon C$, then:

$$(\lambda - \lambda(\varepsilon))^n = \varepsilon(-1)^n c \implies \lambda(\varepsilon) = \lambda + \varepsilon^{\frac{1}{n}} c^{\frac{1}{n}},$$

where the n-th root is considered as a function $\mathbb{C} \to \mathbb{C}$. The problem is therefore not well-conditioned, since the $O(\varepsilon)$ estimate does not hold. Here, it is difficult to compute eigenvalues since they are all close to each other.

Algorithm 2.1 – QR Algorithm, simple version

We want to compute all eigenvalues of a matrix A. For this, we consider the simple iteration:

1. $A_0 := A$
2. We then compute for all $k = 0, 1, 2, \dots$: $A_k = Q_k R_k$ with the QR decomposition and set $A_{k+1} := R_k Q_k$. Essentially, we then have that $A_k \to R$ converges, where R is a right upper

triangular matrix in Schur's normal form, since:

$$A = Q^* R Q \quad \text{with} \quad Q = Q_0 Q_1 Q_2 \dots$$

Remark 2.5

We will see in Page 43 that we can prove convergence.

2.3 Power Method

Algorithm 2.2 – Power method

To compute individual eigenvalues and eigenvectors of a matrix $A \in \mathbb{C}^{n\times n}$, we consider the following procedure: for $y_0 \in \mathbb{C}^n$ arbitrary, set $y_{k+1} := Ay_k$ for $k = 1, 2, \dots$, i.e. $y_k = A^k y_0$.

If one has an approximation to a desired eigenvalue, the inverse power method can be more efficient.

Algorithm 2.3 – Inverse power method (Wielandt iteration)

Let an approximation μ to a desired eigenvalue λ_1 be known, which does not necessarily have to be the largest eigenvalue. Assume

$$|\mu - \lambda_1| \ll |\mu - \lambda_j| \quad \forall j = 2, \dots, n,$$

we have also:

$$\frac{1}{|\mu - \lambda_1|} \gg \frac{1}{|\mu - \lambda_j|}.$$

Since $\frac{1}{\mu-\lambda_j}$ are the eigenvalues of the matrix $(\mu I - A)^{-1}$, we apply the power method to $(\mu I - A)^{-1}$. This can be done without computing the inverse matrix, by only solving the associateed linear system: let y_0 be a starting vector, we solve in the k-th step:

$$(\mu I - A) y_{k+1} = y_k \quad k = 0, 1, 2, \dots$$

We need only one LR decomposition for all iteration steps (since the matrix is the same for each step).

Remark 2.6 – Convergence speed

The convergence speed of the inverse power method may be much better than that of the normal power method. We can, for example, get a rough estimate of the eigenvalue and eigenvector with the normal power method, and then use the inverse power method to obtain precise estimates.

Definition 2.4 – Rayleigh quotient

Let $A \in \mathbb{C}^{n\times n}$ and y_k as in the power method algorithm. The *Rayleigh quotient* is defined by:

$$\frac{y_k^* A y_k}{y_k^* y_k}.$$

Theorem 2.5 – Convergence of the power method

Let $A \in \mathbb{C}^{n\times n}$ be diagonalizable, with $T^{-1}AT = \mathrm{diag}(\lambda_1, \dots, \lambda_n)$ and $T = (v_1 | \dots | v_n)$, where $Av_i = \lambda v_i$. Assume that $|\lambda_1| > |\lambda_2| \geq |\lambda_3| \geq \dots \geq |\lambda_n|$. If $y_0 = \alpha_1 v_1 + \alpha_2 v_2 + \dots + \alpha_n v_n$ with $\alpha_1 \neq 0$, then for $y_{k+1} = Ay_k$:

1. We have:
$$y_k = \lambda_1^k \left(\alpha_1 v_1 + \mathcal{O}\left(\left| \frac{\lambda_2}{\lambda_1} \right|^k \right) \right).$$
We note that $\frac{1}{\lambda_1^k} y_k$ converges to an eigenvector $\alpha_1 v_1$ corresponding to the largest eigenvalue.

2. For the Rayleigh quotient, we have
$$\frac{y_k^* A y_k}{y_k^* y_k} = \lambda_1 + \mathcal{O}\left(\left| \frac{\lambda_2}{\lambda_1} \right|^k \right).$$
If A is normal, then:
$$\frac{y_k^* A y_k}{y_k^* y_k} = \lambda_1 + \mathcal{O}\left(\left| \frac{\lambda_2}{\lambda_1} \right|^{2k} \right).$$

Proof.

1. Let $y_0 = \alpha_1 v_1 + \alpha_2 v_2 + \ldots + \alpha_n v_n$ with $\alpha_1 \neq 0$. We assume without loss of generality that $\|v_j\|_2 = 1$ for all j. Then
$$y_1 = A y_0 = \alpha_1 \lambda_1 v_1 + \alpha_2 \lambda_2 v_2 + \ldots + \alpha_n \lambda_n v_n,$$
and iteratively we obtain:
$$\begin{aligned} y_k = A^k y_0 &= \alpha_1 \lambda_1^k v_1 + \alpha_2 \lambda_2^k v_2 + \ldots + \alpha_n \lambda_n^k v_n \\ &= \lambda_1^k \left(\alpha_1 v_1 + \alpha_2 \left(\frac{\lambda_2}{\lambda_1} \right)^k v_2 + \ldots + \alpha_n \left(\frac{\lambda_n}{\lambda_1} \right)^k v_n \right). \end{aligned}$$

2. We have:
$$\begin{aligned} y_k^* y_k &= \sum_{i=1}^n \sum_{j=1}^n \overline{\alpha_i} \overline{\lambda_i}^k \alpha_j \lambda_j^k v_i^* v_j = \sum_{i=1}^n |\alpha_i|^2 |\lambda_i|^{2k} \underbrace{v_i^* v_i}_{=1} + \sum_{i=1}^n \sum_{j \neq i}^n \overline{\alpha_i} \overline{\lambda_i}^k \alpha_j \lambda_j^k v_i^* v_j \\ &= |\lambda_1|^{2k} \left(|\alpha_1|^2 + \sum_{i=1}^n |\alpha_i|^2 \left| \frac{\lambda_i}{\lambda_1} \right|^{2k} \right) + |\lambda_1|^{2k} \sum_{i=1}^n \sum_{j \neq i}^n \overline{\alpha_i} \alpha_j \frac{\overline{\lambda_i}^k \lambda_j^k}{|\lambda_1|^{2k}} v_i^* v_j \\ &= |\alpha_1|^2 |\lambda_1|^{2k} \left(1 + \mathcal{O}\left(\left| \frac{\lambda_2}{\lambda_1} \right|^k \right) \right). \end{aligned}$$
Then
$$\begin{aligned} y_k^* A y_k = y_k^* y_{k+1} &= \sum_{i=1}^n |\alpha_i|^2 |\lambda_i|^{2k} \lambda_i v_i^* v_i + \sum_{i=1}^n \sum_{j \neq i}^n \overline{\alpha_i} \overline{\lambda_i}^k \alpha_j \lambda_j^{k+1} v_i^* v_j \\ &= |\alpha_1|^2 |\lambda_1|^{2k} \lambda_1 \left(1 + \mathcal{O}\left(\left| \frac{\lambda_2}{\lambda_1} \right|^k \right) \right). \end{aligned}$$
Then
$$\frac{y_k^* A y_k}{y_k^* y_k} = \frac{|\alpha_1|^2 |\lambda_1|^{2k} \lambda_1 \left(1 + \mathcal{O}\left(\left| \frac{\lambda_2}{\lambda_1} \right|^k \right) \right)}{|\alpha_1|^2 |\lambda_1|^{2k} \left(1 + \mathcal{O}\left(\left| \frac{\lambda_2}{\lambda_1} \right|^k \right) \right)} = \lambda_1 \left(1 + \mathcal{O}\left(\left| \frac{\lambda_2}{\lambda_1} \right|^k \right) \right).$$
For normal matrices, the terms $v_i^* v_j = 0$ vanish for $i \neq j$ since the eigenvectors can be chosen to be orthogonal, and the claim follows for these matrices as well.

□

Example 2.3 – Application of the power method

Let

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

$\lambda_1 = 2 + \sqrt{2} \approx 3.4142$ is the largest eigenvalue. We get for a starting vector y_0 the following iteration:

$$y_0 = \begin{pmatrix} 1 \\ 1 \\ 1 \end{pmatrix}, \quad y_1 = \begin{pmatrix} 3 \\ 4 \\ 3 \end{pmatrix}, \quad y_2 = \begin{pmatrix} 10 \\ 14 \\ 10 \end{pmatrix}, \ldots$$

and the Rayleigh quotient is then:

$$\frac{y_1^* A y_1}{y_1^* y_1} = \frac{y_1^* y_2}{y_1^* y_1} = \frac{116}{34} \approx 3.4117.$$

Exercise

Show that the rate of convergence for this example is ≈ 0.59.

Example 2.4 – Application of the inverse power method

Let

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

Let $\mu = 3.41$ and we choose $y_0 = \begin{pmatrix} 1 \\ 1.4 \\ 1 \end{pmatrix}$. We obtain:

$$\frac{y_1^* (\mu I - A)^{-1} y_1}{y_1^* y_1} = \frac{y_1^* y_2}{y_1^* y_1} = -237.3288707 \approx \frac{1}{\mu - \lambda_1}$$

with which we then obtain $\lambda_1 \approx 3.414213562$ and all given digits agree with the largest eigenvalue, $2 + \sqrt{2}$.

Remark 2.7

To prevent overflow (the numbers becoming larger than the computer's largest representable number), we can set:

$$z_{k+1} := A y_k, \quad y_{k+1} := \frac{1}{\|z_{k+1}\|_\infty} z_{k+1},$$

where $\|z_{k+1}\|_\infty$ is the largest component of z_{k+1} in terms of magnitude. Another option is to normalize y_k to be of unit norm at every step, or every few steps. In this case,

$$z_{k+1} := A y_k, \quad y_{k+1} := \frac{1}{\|z_{k+1}\|_2} z_{k+1}.$$

Remark 2.8 – Application of the power method (Google)

What makes (or made) Google unique is an algorithm that provides a suitable order of results, the so-called PageRank algorithm, which is about characterizing the importance of web pages. Google determines the rank $r(P)$ of a page P by:

$$r(P) = \sum_{Q \in B_P} \frac{r(Q)}{|Q|}$$

where $B_P = \{\text{all pages that link to } P\}$, and where $|Q|$ is the number of links from Q (to any page!). This is a recursive definition, but one can see that the vector $y = (r(P_1), \ldots, r(P_N))$ is an eigenvector of a matrix A associated to eigenvalue 1. The matrix $A = (a_{ij})$ is given by

$$a_{ij} = \begin{cases} \frac{1}{|P_j|} & \text{if } P_j \text{ links to } P_i, \\ 0 & \text{otherwise.} \end{cases}$$

Since the column sum of A is 1, 1 is the largest eigenvalue in magnitude (see the following exercise). Thus, the power method $y_{k+1} = Ay_k$ yields the desired eigenvector y.
For this problem, there is an article "The 25,000,000,000 Dollar Eigenvalue Problem: The Linear Algebra Behind Google" by Bryan and Leise[a] (2005) and Langville and Meyer[b] (2006) in SIAM Review.

[a] https://epubs.siam.org/doi/10.1137/050623280
[b] https://www.jstor.org/stable/j.ctt7t8z9

Exercise
Show that, if a matrix A with non-negative components has a column sum $\leq c$ (i.e. $\sum_{i=1}^{n} A_{i,j} \leq c$, $\forall j$), then $\lambda_j \leq c$ for all j. If the column sum is $= c$, then c is the largest eigenvalue.

Answer
Note that A and A^T have the same characteristic polynomial, hence the roots of χ_A and χ_{A^T} are the same and the eigenvalues of A^T and A are equal. Apply the Gershgorin circle theorem on A^T:

$$\mathrm{sp}(A) = \mathrm{sp}(A^T) \subset \bigcup_{j=1}^{n} \left\{ \mu \in \mathbb{C} : |\mu - a_{j,j}| \leq \sum_{k \neq j} |a_{k,j}| \right\}.$$

Since A has non-negative components, $\sum_{k\neq j} |a_{k,j}| = \sum_{k\neq j} a_{k,j}$, hence $|\mu - a_{j,j}| \leq c - a_{j,j}$. Moreover, the reverse triangle inequality yields

$$|\mu - a_{j,j}| \geq ||\mu| - |a_{j,j}||.$$

To show this, apply the triangle inequality to $|\mu| = |\mu \pm a_{j,j}|$ and $|a_{j,j}| = |a_{j,j} \pm \mu|$. Thus,

$$||\mu| - |a_{j,j}|| \leq c - a_{j,j},$$

and

$$\begin{aligned} |\mu| - a_{j,j} &\leq c - a_{j,j}, \quad \text{if } |\mu| \geq a_{j,j} \\ -|\mu| + a_{j,j} &\leq c - a_{j,j}, \quad \text{if } |\mu| \leq a_{j,j}. \end{aligned}$$

The first line yields $|\mu| \leq c$, and the second line is for $|\mu| \leq a_{j,j} \leq c$. In both cases, we obtain $|\mu| \leq c$, i.e. all eigenvalues of A are smaller or equal to c.

Now, if the column sum is equal to one we have $\mathbf{1}^T A = c\mathbf{1}^T$, where $\mathbf{1} = (1, \ldots, 1)$, which means that c is indeed an eigenvalue of A, since it is an eigenvalue of A^T.

2.4 Simultaneous Iteration and QR Algorithm

Motivation
In the following, let A be a real matrix whose eigenvalues satisfy

$$|\lambda_1| > |\lambda_2| > |\lambda_3| > \ldots > |\lambda_n|$$

i.e. in particular, A is diagonalizable. We want to compute not only the first but also the second, third, etc., eigenvalues.

Review – Power method

Let y_0 be arbitrary and $y_{k+1} = Ay_k$. We know that:

$$y_k = \lambda_1^k \left(\alpha_1 \widetilde{v_1} + \mathcal{O}\left(\left| \frac{\lambda_2}{\lambda_1} \right|^k \right) \right) = \lambda_1^k \left(|\alpha_1| \underbrace{\operatorname{sgn}(\alpha_1) \widetilde{v_1}}_{=:v_1} + \mathcal{O}\left(\left| \frac{\lambda_2}{\lambda_1} \right|^k \right) \right),$$

where we assume that $\alpha_1 \neq 0$ and $\widetilde{v_1}$ is an eigenvector with $\|\widetilde{v_1}\|_2 = 1$. The vector v_1 is also an eigenvector of A with $\|v_1\|_2 = 1$. Thus, $\frac{y_k}{\|y_k\|} \to v_1$ if $\lambda_1 > 0$ and $(-1)^k \frac{y_k}{\|y_k\|} \to v_1$ if $\lambda_1 < 0$. In general, $(\operatorname{sgn} \lambda_1)^k \frac{y_k}{\|y_k\|} \to v_1$.

Algorithm 2.4 – Extension of the power method

Let q_0 be arbitrary with $\|q_0\|_2 = 1$. We consider the iteration $Aq_k = \lambda_1^{(k+1)} q_{k+1}$ with $\|q_{k+1}\|_2 = 1$ and $\operatorname{sgn} \lambda_1^{(k+1)} = \operatorname{sgn}(q_k^T A q_k)$. If k is large, this is $\operatorname{sgn} \lambda_1$. We know that $q_k \to v_1$ converges and $\lambda_1^{(k)} \to \lambda_1$ with the convergence rate $\left| \frac{\lambda_2}{\lambda_1} \right|$.

Idea (Computation of the next eigenvalue)
Now let λ_1, v_1 be known and we want to compute λ_2, v_2. We consider the orthogonal complement to $\mathbb{R}v_1$:

$$V = \{u \in \mathbb{R}^n \mid v_1^T u = 0\}.$$

We know $\dim V = n - 1$. *Furthermore, we consider the following mapping:*

$$L_1 := P \circ (A_{|V}) : V \to \mathbb{R}^n \to V$$

where P is an orthogonal projection: $\mathbb{R}^n \to V$. We have $q = \alpha v_1 + u \in \mathbb{R}^n$, where $u \in V$. Then $v_1^T q = \alpha$, with which we obtain α, noting that $\|v_1\| = 1$. Then $Pq = u = q - \alpha v_1 = (I - v_1 v_1^T)q$. Thus, for $u \in V$:

$$L_1(u) = (I - v_1 v_1^T) A u.$$

With the next theorem, we obtain that L_1 has precisely the remaining eigenvalues of A.

Theorem 2.6 – Eigenvalues of L_1

The eigenvalues of L_1 are $\lambda_2, \ldots, \lambda_n$.

Proof. By Schur's normal form, we know that for the matrix A, we have:

$$U^* A U = \begin{pmatrix} \lambda_1 & \star & \star \\ & \ddots & \star \\ 0 & & \lambda_n \end{pmatrix} = R$$

where $U = (u_1 | \ldots | u_n)$ is a unitary matrix with $u_1 = v_1$. The vectors $(u_2, \ldots, u_n)$ form an ONB of V, in particular $v_1^T u_j = 0$ for $j \geq 2$. Then

$$\begin{aligned} L_1(u_i) &= (I - v_1 v_1^T) A u_i = (I - v_1 v_1^T)(AU)_{\cdot,i} = (I - v_1 v_1^T)(UR)_{\cdot,i} \\ &= (I - v_1 v_1^T)(v_1 r_{1,i} + \ldots + u_{i-1} r_{i-1,i} + u_i \lambda_i) \\ &= u_2 r_{2,i} + \ldots + u_{i-1} r_{i-1,i} + u_i \lambda_i. \end{aligned}$$

From this formula, we can deduce the representation matrix of L_1 with respect to the basis $(u_2, \ldots, u_n)$, which looks as follows:

$$\begin{pmatrix} \lambda_2 & \star & \star \\ 0 & \ddots & \star \\ 0 & 0 & \lambda_n \end{pmatrix}$$

And this matrix has precisely the eigenvalues $\lambda_2, \ldots, \lambda_n$. □

Remark 2.9

It may be sometimes convenient to have L_1 in the $(u_1, \dots, u_n)$ basis. In this case, 0 is another eigenvalue of L_1 and its matrix in the $(u_1, \dots, u_n)$ basis reads

$$\begin{pmatrix} 0 & 0 & 0 & 0 \\ 0 & \lambda_2 & \star & \star \\ 0 & 0 & \ddots & \star \\ 0 & 0 & 0 & \lambda_n \end{pmatrix}$$

The idea of orthogonalizing Au with respect to the previously obtained eigenvectors is called *deflation*.

Algorithm 2.5 – Computation of the second eigenvalue

To compute λ_2, we apply the power method to L_1: let $p_0 \in V$ be arbitrary with $\|p_0\|_2 = 1$. Then by the power method:

$$L_1(p_k) = (I - v_1 v_1^T) A p_k = \lambda_2^{(k+1)} p_{k+1} \quad \text{with} \quad \|p_{k+1}\|_2 = 1, \quad \operatorname{sgn} \lambda_2^{(k+1)} = \operatorname{sgn} p_k^T L_1(p_k).$$

Thus, $\lambda_2^{(k)} \to \lambda_2$ and $p_k \to u_2$ etc., and we obtain Schur's normal form.

Algorithm 2.6 – Modification of the deflated power method

We do not compute first λ_1 and v_1, but we use simultaneous iteration for λ_1 and λ_2. This gives us the following algorithm: let q_0, p_0 be arbitrary with $\|q_0\|_2 = \|p_0\|_2 = 1$ and $q_0 \perp p_0$. Then we compute:

$$A q_k = \lambda_1^{(k+1)} q_{k+1}, \quad \|q_{k+1}\|_2 = 1, \quad \operatorname{sgn} \lambda_1^{(k+1)} = \operatorname{sgn} q_k^T A q_k,$$

and

$$(I - q_{k+1} q_{k+1}^T) A p_k = \lambda_2^{(k+1)} p_{k+1}, \quad \|p_{k+1}\|_2 = 1, \quad \operatorname{sgn} \lambda_2^{(k+1)} = \operatorname{sgn} p_k^T A p_k.$$

The orthogonality $p_{k+1} \perp q_{k+1}$ holds.

Algorithm 2.7 – Alternative notation for simultaneous iteration

We write alternatively, but equivalently to Algorithm 2.6,

$$A(q_k, p_k) = (q_{k+1}, p_{k+1}) \begin{pmatrix} \lambda^{(k+1)} & \alpha_{k+1} \\ 0 & \lambda_2^{(k+1)} \end{pmatrix}, \quad \alpha_{k+1} = q_{k+1}^T A p_k.$$

The right-hand side can be obtained by applying the QR decomposition to the $n \times 2$ matrix $A(q_k, p_k)$.

Algorithm 2.8 – Generalization of the algorithm

We choose for U_0 an arbitrary orthogonal matrix (for example $U_0 = I$). We consider the iteration:

$$A U_k = U_{k+1} R_{k+1},$$

which is precisely the QR algorithm. Then

$$R_k \to \begin{pmatrix} \lambda_1 & \star & \star \\ 0 & \ddots & \star \\ 0 & 0 & \lambda_n \end{pmatrix}$$

converges and $U_k \to (u_1, \dots, u_n)$ converges. By doing this, we obtain Schur's normal form.

Algorithm 2.9 – QR algorithm

We set as above: $Q_k := U_{k-1}^T U_k$. Then

$$Q_{k+1}R_{k+1} = U_k^T \underbrace{U_{k+1}R_{k+1}}_{=AU_k} = U_k^T AU_k = U_k^T \underbrace{AU_{k-1}}_{U_kR_k} Q_k = R_kQ_k.$$

Thus, we obtain the following algorithm:

1. $A_0 = A = Q_0R_0$ (QR decomposition in the last step)
2. $A_1 = R_0Q_0 = Q_1R_1$ (QR decomposition)
3. $A_2 = R_1Q_1 = Q_2R_2$ (QR decomposition)
4. ...

This justifies Algorithm 2.1.

Since the QR algorithm is a just a power method applied to n orthonormal eigenvectors, with eigenvalues all distinct, it converges.

Remark 2.10 – Historical note on the QR algorithm

The algorithm goes back to Rutishauser, 1958, who, however, had used the LR decomposition. The QR algorithm that we consider here goes back to Francis, 1961, and Kublanovskaya, 1961.

Motivation (Outlook)

1. *The QR decomposition of an arbitrary matrix is performed with $\mathcal{O}(n^3)$ operations, which is too expensive. We therefore first transform A into Hessenberg form, i.e. $Q^TAQ = H$ (which is almost triangular form and has a diagonal below the main diagonal), which is about as expensive as a QR decomposition. If A is symmetric, then we obtain for H a tridiagonal matrix. For the QR decomposition of a Hessenberg matrix, we need only $\mathcal{O}(n^2)$ operations, or $\mathcal{O}(n)$ operations if A is symmetric. Then all A_k are again Hessenberg matrices, as we will see in the next section.*

2. *The convergence speed is very slow, about $\left|\frac{\lambda_2}{\lambda_1}\right|$. We consider the idea of shifting the matrix A, i.e. we consider $A_k - \mu_k I$, where we choose the parameter μ_k such that the convergence is accelerated. We will consider this procedure in the sixth section.*

3. *We have not yet captured complex eigenvalues. We cannot converge a real matrix A_k to a real upper triangular matrix if there are complex eigenvalues. In the case of a complex eigenvalue, A converges to a matrix in (almost) Schur's normal form with a 2×2 block on the main diagonal:*

$$\begin{pmatrix} a_{r,r}^{(k)} & a_{r,r+1}^{(k)} \\ a_{r+1,r}^{(k)} & a_{r+1,r+1}^{(k)} \end{pmatrix}$$

This converges to complex conjugate eigenvalues $\alpha \pm i\beta$, as we will see in Section 2.7.

2.5 Transformation to Hessenberg Form

Idea

We will use Householder transformations, defined for a given $v \in \mathbb{R}^m$ by $I - 2vv^T$. They can be understood as the linear operator performing a mirror symmetry with respect to the hyperplane orthogonal to v. In other words, if $x = \alpha v + \beta w$ with $w \perp v$, then $(I - 2vv^T)x = -\alpha v + \beta w$.

Lemma 2.3 – Householder transformations

Let $x \in \mathbb{R}^m$, there is $v \in \mathbb{R}^m$ with $\|v\|_2 = 1$ such that

$$(I - 2vv^*)x = (\star, 0, \dots, 0).$$

Proof. If $x_1 = 0$, define $v = (\|x\|_2, x_2, \dots, x_m)$. Then

$$2\frac{vv^*}{\|v\|_2^2}x = \frac{2}{\|v\|_2^2}\begin{pmatrix}\|x\|_2\\ x_2\\ \vdots\\ x_m\end{pmatrix}\sum_{j=2}^{m}|x_j|^2 = \frac{2}{2\sum_{j=2}^{m}|x_j|^2}\begin{pmatrix}\|x\|_2\\ x_2\\ \vdots\\ x_m\end{pmatrix}\sum_{j=2}^{m}|x_j|^2 = \begin{pmatrix}\|x\|_2\\ x_2\\ \vdots\\ x_m\end{pmatrix}.$$

If $x_1 \neq 0$, define $v = (\|x\|_2 + \sigma x_1, \sigma x_2, \dots, \sigma x_m)$, where $\sigma = \frac{\overline{x_1}}{|x_1|}$ ($\iff$ $x_1 = |x_1|\overline{\sigma}$). Then

$$\begin{aligned}2\frac{vv^*}{\|v\|_2^2}x &= \frac{2}{\|v\|_2^2}\begin{pmatrix}\|x\|_2 + \sigma x_1\\ \sigma x_2\\ \vdots\\ \sigma x_m\end{pmatrix}\left(\|x\|_2 x_1 + \overline{\sigma}\sum_{j=1}^{m}|x_j|^2\right)\\ &= \frac{2}{2\|x\|_2^2 + 2\|x\|_2|x_1|}\begin{pmatrix}\|x\|_2 + \sigma x_1\\ \sigma x_2\\ \vdots\\ \sigma x_m\end{pmatrix}\overline{\sigma}\left(\|x\|_2|x_1| + \|x\|_2^2\right) = \begin{pmatrix}\overline{\sigma}\|x\|_2 + x_1\\ x_2\\ \vdots\\ x_m\end{pmatrix}.\end{aligned}$$

In both cases, we obtain

$$(I - 2vv^*)x = (\star, 0, \dots, 0). \tag{2.1}$$

□

Theorem 2.7 – Transformation to Hessenberg form

Let $A \in \mathbb{R}^{n\times n}$. This matrix can be transformed to Hessenberg form by $(n-2)$ Householder transformations:

$$Q^T AQ = H = \begin{pmatrix}* & * & * & *\\ * & * & * & *\\ 0 & \ddots & \ddots & \vdots\\ \vdots & & & \\ 0 & \cdots & 0 & * & *\end{pmatrix},$$

where $Q := Q_{n-2} \cdot \ldots \cdot Q_1$, and $Q_i := \begin{pmatrix}I_i & 0\\ 0 & I_{n-i} - 2w_i w_i^T\end{pmatrix}$ is an Householder transformation. The matrix H is an upper triangular matrix with entries on the sub-diagonal. If A is symmetric, H is tridiagonal.

Proof.

1. We choose $\widetilde{Q}_1 = I - 2w_1 w_1^T$ (with $w_1^T w_1 = 1$) as an $(n-1)\times(n-1)$ Householder matrix, such that

$$\widetilde{Q}_1\begin{pmatrix}a_{2,1}\\ \vdots\\ a_{n,1}\end{pmatrix} = \begin{pmatrix}\star\\ 0\\ \vdots\\ 0\end{pmatrix}$$

Note that $\widetilde{Q}_1$ is symmetric. We obtain:

$$Q_1 = \begin{pmatrix} 1 & 0 \\ 0 & \widetilde{Q}_1 \end{pmatrix} \quad \text{and} \quad A^{(1)} = Q_1 A_1 Q_1^T = \left(\begin{array}{c|c} a_{1,1} & \\ \star & \\ 0 & \star \\ \vdots & \\ 0 & \end{array}\right) \begin{pmatrix} 1 & 0 \\ 0 & \widetilde{Q}_1 \end{pmatrix} = \left(\begin{array}{c|c} a_{1,1} & \\ \star & \\ 0 & \star \\ \vdots & \\ 0 & \end{array}\right).$$

2. We choose $\widetilde{Q}_2 := I - 2w_2 w_2^T$ an $(n-2) \times (n-2)$ Householder matrix such that

$$\widetilde{Q}_2 \begin{pmatrix} a_{32}^{(1)} \\ \vdots \\ a_{n2}^{(1)} \end{pmatrix} = \begin{pmatrix} \star \\ 0 \\ \vdots \\ 0 \end{pmatrix}, \quad \text{and let } Q_2 = \begin{pmatrix} I_2 & 0 \\ 0 & \widetilde{Q}_2 \end{pmatrix} \Rightarrow A^{(2)} = Q_2 A^{(1)} Q_2^T = \left(\begin{array}{cc|c} \star & \star & \\ \star & \star & \\ 0 & \star & \\ \vdots & 0 & \star \\ \vdots & \vdots & \\ 0 & 0 & \end{array}\right).$$

3. We proceed inductively in this manner and obtain:

$$Q^T A Q = H$$

with $Q := Q_{n-2} \cdot \ldots \cdot Q_2 \cdot Q_1$ and H an upper triangular matrix with a nonzero subdiagonal.

4. For symmetric matrices, we have

$$H^T = Q^T A^T Q = Q^T A Q = H$$

Due to H having the all diagonals below the subdiagonal equal to zero, all diagonals of $H^T = H$ above the supdiagonal are zero. Therefore, H is tridiagonal.

□

Remark 2.11 – Consideration of computational effort

It requires about $\frac{5}{3}n^3$ operations for a general A and for symmetric matrices about $\frac{2}{3}n^3$ operations.

Theorem 2.8 – Inheritance of Hessenberg form

Let $H = QR$ be a QR decomposition. We set $\tilde{H} := RQ$. If H is a Hessenberg matrix, then $\tilde{H}$ is also a Hessenberg matrix. If H is tridiagonal and symmetric, then $\tilde{H}$ is also tridiagonal and symmetric.

Proof. By a Householder transformation, we obtain (for the QR decomposition):

$$Q_1 H = \left(\begin{array}{c|c} \star & \\ 0 & \\ \vdots & \star \\ 0 & \end{array}\right) \quad \text{and} \quad Q_1 := I - 2w_1 w_1^T \quad \text{with} \quad w_1 = \begin{pmatrix} \star \\ \star \\ 0 \\ \vdots \\ 0 \end{pmatrix}.$$

The fact that H is an Hessenberg matrix is used to have only two nonzero components in w_1, since $w_1 = (\star, \sigma H_{2,1}, \ldots, \sigma H_{n,1})$ for some $\sigma \in \mathbb{C}$, $|\sigma| = 1$. See the proof of Lemma 2.3 for more details. Then

$$RQ_1 = \begin{pmatrix} * & * & * \\ 0 & * & * \\ 0 & 0 & * \end{pmatrix} \begin{pmatrix} * & * & 0 \\ * & * & 0 \\ 0 & 0 & I_{n-2} \end{pmatrix} = \begin{pmatrix} * & * & * \\ * & * & * \\ 0 & 0 & * \end{pmatrix}$$

i.e. we obtain an upper triangular matrix with an entry on the subdiagonal. More generally, one has $\widetilde{Q_k} = I_{n-k+1} - 2w_k w_k^T$ where $w_k = (\star, \star, 0, \ldots, 0) \in \mathbb{R}^{n-k+1}$, and define

$$Q_k = \begin{pmatrix} I_{k-1} & 0 \\ 0 & \widetilde{Q_k} \end{pmatrix} = \begin{pmatrix} I_{k-1} & 0 & 0 & 0 \\ 0 & \star & \star & 0 \\ 0 & \star & \star & 0 \\ 0 & 0 & 0 & I_{n-k-1} \end{pmatrix}.$$

Letting $Q = Q_1 \ldots Q_n$, one obtains the claim. When H (not necessarily Hessenberg) is tridiagonal, one can show that Q is in Hessenberg form, and so is RQ. This means that $\tilde{H} = RQ$ is in Hessenberg form. Note that $\tilde{H} = RQ = Q^T QRQ = Q^T HQ$, and the symmetry of H yields the symmetry of $\tilde{H}$. A symmetric Hessenberg matrix is tridiagonal, so finally $\tilde{H}$ is symmetric and tridiagonal. □

Algorithm 2.10 – Modification of the QR algorithm

1. We perform a pre-transformation, i.e. we bring $Q^T AQ = H_0$ to Hessenberg form. This requires $\mathcal{O}(n^3)$ steps.
2. Apply the classical QR algorithm to H_0, i.e. $H_k = Q_k R_k$. This requires generally $\mathcal{O}(n^2)$ operations, and only $\mathcal{O}(n)$ operations in the symmetric case.
3. $H_{k+1} := R_k Q_k$, which also requires generally $\mathcal{O}(n^2)$ operations, and only $\mathcal{O}(n)$ operations in the symmetric case.

2.6 QR Algorithm with Shift

Motivation
We will now always assume that A has only real eigenvalues that are pairwise distinct, i.e. $|\lambda_1| > |\lambda_2| > \ldots > |\lambda_n|$. Moreover, we work on the Hessenberg matrix $H = Q^T AQ$. We want to improve the convergence speed with a shift idea.

Idea (Shift)
We expect after the fourth section that we have the convergence rate

$$\left| h_{i+1,i}^{(k)} \right| = \mathcal{O}\left(\left| \frac{\lambda_{i+1}}{\lambda_i} \right|^k \right).$$

This can be very slow under certain circumstances. We consider the shifted matrix $\tilde{H} = H - \mu I$, which has the eigenvalues $\lambda_i - \mu$. Applying the QR algorithm here, we obtain:

$$\left| \tilde{h}_{i+1,i}^{(k)} \right| = \mathcal{O}\left(\left| \frac{\lambda_{i+1} - \mu}{\lambda_i - \mu} \right|^k \right).$$

This converges very quickly if $\mu \approx \lambda_{i+1}$.

Convention
We can assume without loss of generality that H is a non-reduced Hessenberg matrix, i.e. for all i, $h_{i+1,i} \neq 0$. If an element were zero, then the matrix would have the form:

$$H = \begin{pmatrix} H_1 & \star \\ 0 & H_2 \end{pmatrix}$$

and then the eigenvalues of H are the eigenvalues of H_1 and H_2, so the QR algorithm would be used separately on H_1 and H_2.

Theorem 2.9

Let H be a non-reduced Hessenberg matrix and μ an eigenvalue of H. Let $H - \mu I = QR$ be the QR decomposition. We consider $\tilde{H} := RQ + \mu I$. Then $\tilde{h}_{n,n} = \mu$ and $\tilde{h}_{n,n-1} = 0$, i.e. the matrix decomposes after one QR step.

Proof. We consider $H - \mu I$ with $h_{i+1,i} \neq 0$ for all i. This implies that the first $n-1$ columns of this matrix are linearly independent (because column j has a component at index $j+1$ which does not appear in all previous columns). We can write:

$$Q^T(H - \mu I) = \begin{pmatrix} R_{n-1} & * \\ 0 & r_{n,n} \end{pmatrix} =: R$$

This is the QR decomposition, which always exists, and R_{n-1} must be invertible because it is a square matrix of linearly independent columns. Because $H - \mu I$ is singular, we must have $r_{n,n} = 0$, so the last row of RQ must also be zero. Thus, we have:

$$\tilde{H} = RQ + \mu I = \begin{pmatrix} * & * & * & * \\ \ddots & * & * & * \\ & \ddots & * & * \\ & & 0 & \mu \end{pmatrix}$$

□

If one does not know the eigenvalue μ, one can use the iterative values $h^{(k)}_{n,n}$ as approximations.

Algorithm 2.11 – QR algorithm with shift (for real eigenvalues)

Without loss of generality, let H_0 be a non-reduced Hessenberg matrix. We consider $H_k - h^{(k)}_{n,n} I = Q_k R_k$ and $H_{k+1} := R_k Q_k + h^{(k)}_{n,n} I$ for $k = 1, 2, \ldots$ until

$$\left| h^{(k)}_{n,n-1} \right| \leq \texttt{eps} \left(\left| h^{(k)}_{n,n} \right| + \left| h^{(k)}_{n-1,n-1} \right| \right)$$

where `eps` is the machine precision. Then we accept $h^{(k)}_{n,n}$ as an eigenvalue. We start again with the submatrix $(h^{(k)}_{ij})^{n-1}_{i,j=1}$ until we finally reach a 1×1 matrix.

Remark 2.12 – Convergence speed

Let

$$H - h_{n,n} I = \begin{pmatrix} * & * & * & * & * \\ * & * & * & * & * \\ & \ddots & * & * & * \\ & & \ddots & * & b \\ & & & \varepsilon & 0 \end{pmatrix} \Rightarrow Q_{n-2} \cdot Q_{n-3} \cdot \ldots \cdot Q_1 \cdot (H - h_{n,n} I) = \begin{pmatrix} * & * & * & * & * \\ 0 & * & * & * & * \\ & \ddots & * & * & * \\ & & 0 & a & b \\ & & & \varepsilon & 0 \end{pmatrix}$$

where the Q_i are defined as in Theorem 2.8, i.e. we have an identity matrix and once a 2×2 block on the main diagonal. Moreover, ε and b remain unchanged: the matrix Q_i is constructed to only act on the i-th column of $H - h_{n,n} I$. In the symmetric case, $b = \varepsilon$. We still need to compute the QR decomposition of $\begin{pmatrix} a & b \\ \varepsilon & 0 \end{pmatrix}$. It is:

$$\begin{pmatrix} a & b \\ \varepsilon & 0 \end{pmatrix} = \underbrace{\frac{1}{\sqrt{a^2 + \varepsilon^2}} \begin{pmatrix} a & -\varepsilon \\ \varepsilon & a \end{pmatrix}}_{=:\tilde{Q}} \underbrace{\begin{pmatrix} \sqrt{a^2 + \varepsilon^2} & \frac{ba}{\sqrt{a^2+\varepsilon^2}} \\ 0 & -\frac{b\varepsilon}{\sqrt{a^2+\varepsilon^2}} \end{pmatrix}}_{=:\tilde{R}}.$$

Then

$$\tilde{R}\tilde{Q} = \begin{pmatrix} * & * \\ -\frac{b\varepsilon^2}{a^2+\varepsilon^2} & * \end{pmatrix}$$

Let $Q_{n-1} := \begin{pmatrix} I_{n-2} & 0 \\ 0 & \tilde{Q} \end{pmatrix}$ and $R = \begin{pmatrix} * & * & * & * \\ 0 & * & * & * \\ & \ddots & * & * \\ & & 0 & \tilde{R} \\ & & 0 & \end{pmatrix}$, we investigate:

$$H - h_{n,n}I = \underbrace{Q_1 \cdot \ldots \cdot Q_{n-1}}_{=:Q} R,$$

and it is

$$\tilde{H} = RQ + h_{n,n}I = \begin{pmatrix} * & * & * & * & * \\ \ddots & * & * & * & * \\ & \ddots & * & * & * \\ & & \ddots & * & * \\ & & & -\frac{b\varepsilon^2}{a^2+\varepsilon^2} & * \end{pmatrix}.$$

We see that the left element becomes smaller at every iteration, and as soon as this is below the machine precision, we accept $h_{n,n}$ as an eigenvalue. If $\varepsilon^2 \ll a^2$, we obtain quadratic convergence, i.e. from $h_{n,n-1} = \mathcal{O}(\varepsilon)$ it follows that $\tilde{h}_{n,n-1} = \mathcal{O}(\varepsilon^2)$. For symmetric matrices, we even obtain cubic convergence (because $b = \varepsilon$), i.e. from $h_{n,n-1} = \mathcal{O}(\varepsilon)$ it follows that $\tilde{h}_{n,n-1} = \mathcal{O}(\varepsilon^3)$.

Remark 2.13 – Stability consideration of the QR algorithm

We finally obtain that $\hat{Q}^T A \hat{Q} = \hat{R}$ is Schur's normal form. The QR algorithm is stable in the sense of backward analysis, i.e. if $\hat{R} = \hat{Q}^T \hat{A} \hat{Q}$ is Schur's normal form of a perturbed matrix, then:

$$\|A - \hat{A}\|_2 \leq C \cdot \texttt{eps} \cdot \|A\|_2, \quad \hat{Q}^T\hat{Q} = I + F \quad \text{with} \quad \|F\|_2 \leq c \cdot \texttt{eps}.$$

Proof. Wilkinson, The Algebraic Eigenvalue Problem, 1965. □

2.7 Computation of Complex Eigenvalues

Motivation

We now investigate complex, non-real eigenvalues of real matrices $A \in \mathbb{R}^{n\times n}$, which occur in pairs of conjugate complex eigenvalues. Here, we will iteratively compute the real Schur normal form.

Theorem 2.10 – Real Schur normal form

For $A \in \mathbb{R}^{n\times n}$, there exists an orthogonal matrix $Q \in \mathbb{R}^{n\times n}$ such that

$$Q^T A Q = \begin{pmatrix} R_{11} & R_{12} & \ldots & R_{1m} \\ & R_{22} & & \vdots \\ & & \ddots & \vdots \\ & & & R_{mm} \end{pmatrix}$$

where each R_{ii} is either a real number or a real 2×2 matrix with conjugate complex eigenvalues.

Proof. We proceed by induction on the number of pairs of complex conjugate eigenvalues, k. If $k = 0$, then A has only real eigenvalues, and we can choose $Q = U$ real in Schur's normal form, as

in the earlier proof. For the induction step: A has a pair of complex conjugate eigenvalues, i.e. we have an eigenvalue $\lambda = \alpha + i\beta$ and $\bar{\lambda} = \alpha - i\beta$ with $\beta \neq 0$. Then there are linearly independent eigenvectors $v = x + iy$ and $\bar{v} = x - iy$ (since from $Av = \lambda v$ it follows that $\bar{A}\bar{v} = \bar{\lambda}\bar{v}$). It is $Av = \lambda v = (\alpha + i\beta)(x + iy) = (\alpha x - \beta y) + i(\alpha y + \beta x)$. On the other hand, $Av = A(x + iy)$. By coefficient comparison, we obtain:

$$Ax = (\alpha x - \beta y), \quad Ay = \alpha y + \beta x, \quad \text{i.e.} \quad A(x, y) = (x, y) \cdot \begin{pmatrix} \alpha & \beta \\ -\beta & \alpha \end{pmatrix}.$$

Since $v, \bar{v}$ are linearly independent, x, y must also be linearly independent, because:

$$(v, \bar{v}) = (x, y) \cdot \begin{pmatrix} 1 & 1 \\ i & -i \end{pmatrix}, \quad \det \begin{pmatrix} 1 & 1 \\ i & -i \end{pmatrix} = -2i \neq 0.$$

The vectors x and y span a two-dimensional subspace of $\mathbb{R}^n$, which A maps into itself. Let (u_1, u_2) be an ONB of this subspace, which we extend to an ONB $(u_1, \ldots, u_n)$ of $\mathbb{R}^n$. We set $U := (u_1, \ldots, u_n) \in \mathbb{R}^{n \times n}$, which is an orthogonal matrix. It holds that:

$$AU = U \begin{pmatrix} R_{11} & \star \\ 0 & \tilde{A} \end{pmatrix}$$

where R_{11} is a 2×2 matrix with a pair of complex conjugate eigenvalues. By the induction hypothesis, there exists $\tilde{Q}$ such that $\tilde{Q}^T \tilde{A} \tilde{Q}$ has the block triangular form, from which the claim follows immediately. We finally set:

$$Q := \begin{pmatrix} I_2 & 0 \\ 0 & \tilde{Q} \end{pmatrix} U.$$

□

Idea (QR algorithm for complex eigenvalues)
We apply the shifted QR algorithm, with the additional knowledge that there are two complex conjugate eigenvalues. We consider the following iteration for a complex μ_k:

1. $H_k - \mu_k I = Q_k R_k$
2. $H_{k+1} := R_k Q_k + \mu_k I$
3. $H_{k+1} - \bar{\mu}_k I = Q_{k+1} R_{k+1}$
4. $H_{k+2} := R_{k+1} Q_{k+1} + \bar{\mu}_k$

Here, Q_k is of course not orthogonal but unitary.

Theorem 2.11

If H_k is real, then we can choose the QR decomposition such that H_{k+2} is real again.

Proof. We know that:

$$H_{k+1} = R_k Q_k + \mu_k I = Q_k^*(H_k - \mu_k I)Q_k + \mu_k I = Q_k^* H_k Q_k$$

and analogously we obtain:

$$H_{k+2} = Q_{k+1}^* H_{k+1} Q_{k+1} = (Q_k Q_{k+1})^* H_k (Q_k Q_{k+1})$$

It suffices to show that $Q_k Q_{k+1}$ is real. We compute with the help of the above calculations:

$$\begin{aligned} Q_k Q_{k+1} R_{k+1} R_k &= Q_k (H_{k+1} - \bar{\mu}_k I) R_k = Q_k (R_k Q_k + \mu_k I - \bar{\mu}_k I) R_k \\ &= (Q_k R_k)^2 + (\mu_k - \bar{\mu}_k) Q_k R_k \\ &= (H_k - \mu_k I)^2 + (\mu_k - \bar{\mu}_k)(H_k - \mu_k I) \\ &= (H_k - \mu_k I)(H_k - \mu_k I + \mu_k I - \bar{\mu}_k I) \\ &= H_k^2 - 2\mathrm{Re}(\mu_k) H_k + |\mu_k|^2 I =: M_k \end{aligned}$$

and we see that M_k is real. Since $R_{k+1}R_k$ is a triangular matrix and Q_kQ_{k+1} is a unitary matrix, we have computed the QR decomposition of M_k. We can choose the QR decomposition such that the diagonal elements of R_k, R_{k+1} are real, and the upper triangular property of $R_{k+1}R_k$ shows that Q_kQ_{k+1} is also real. □

Remark 2.14 – Uniqueness of the QR decomposition

The QR decomposition is unique up to multiplication by a diagonal matrix.

Remark 2.15 – Computational effort

We want to compute H_{k+2} from H_k only with real operations: The computation of the matrix M_k requires $\mathcal{O}(n^3)$ operations. We show now that we can compute H_{k+2} from H_k in $\mathcal{O}(n^2)$ real operations.

Theorem 2.12

Let $A \in \mathbb{R}^{n\times n}$ and $H = Q^TAQ$ be a Hessenberg matrix with $h_{i+1,i} \neq 0$ for $i = 1, \dots, n-1$. Then Q and H can be determined from the first column of Q.

Proof. Let $Q = (q_1, \dots, q_n)$. We have $AQ = QH$, i.e. $Aq_i = \sum_{j=1}^{i+1} q_j h_{ji}$. On the other hand, $Q^TAQ = H$, i.e. $q_j^TAq_i = h_{ji}$. We take q_1 as given. Then we know that $h_{11} = q_1^TAq_1$. Then q_2 is a multiple of $Aq_1 - h_{11}q_1$. Thus, we can determine q_2 up to the sign uniquely (because $\|q_2\|_2 = 1$). Thus, we also obtain h_{12}, h_{21}, h_{22}. By induction, the claim follows. □

Algorithm 2.12 – Francis' QR step

1. We compute the first column of M_k: $M_ke_1 = H_k(H_ke_1) - 2\mathrm{Re}(\mu_k)(H_ke_1) + |\mu_k|^2e_1$, which takes $\mathcal{O}(n^2)$ operations.
2. We compute the Householder matrix Q_0 with $Q_0(M_ke_1) = \alpha e_1$, which is a reflection, and it is:
$$Q_0 = \begin{pmatrix} \star & \star & \star & \\ \star & \star & \star & 0 \\ \star & \star & \star & \\ & 0 & & I \end{pmatrix}.$$
The 3×3 block comes from $H_ke_1 = \alpha_1e_1 + \alpha_2e_2$ (because H is Hessenberg), and thus $H_k(H_ke_1) = \alpha_1H_ke_1 + \alpha_2H_ke_2 = \beta_1e_1 + \beta_2e_2 + \beta_3e_3$. So the Householder vector for M_ke_1 only involves three nonzero components.
3. We transform $Q_0^TH_kQ_0$ into Hessenberg form $\tilde{H}$ in $\mathcal{O}(n^2)$ operations with Householder matrices $\tilde{Q}_1, \dots, \tilde{Q}_{n-3}$. Then we compute $Q^TH_kQ = \tilde{H}$ where $Q := Q_0\tilde{Q}_1 \cdot \ldots \cdot \tilde{Q}_{n-3}$. It holds $\tilde{H} = H_{k+2}$.

Proof. When computing the Hessenberg matrix $\tilde{H}$ of H_k, we know that the matrices $\tilde{H}$ and Q are fully determined from the first column of Q. We focus now on obtaining that first column. Write $H_k = (h_1|\dots|h_n)$, we have:

$$\begin{aligned} Q_0^TH_kQ_0 &= \begin{pmatrix} * & * & * & \\ * & * & * & 0 \\ * & * & * & \\ & 0 & & I \end{pmatrix} \begin{pmatrix} h_1 & h_2 & h_3 & \dots & h_n \end{pmatrix} \begin{pmatrix} * & * & * & \\ * & * & * & 0 \\ * & * & * & \\ & 0 & & I \end{pmatrix} \\ &= \begin{pmatrix} * & * & * & \\ * & * & * & 0 \\ * & * & * & \\ & 0 & & I \end{pmatrix} \begin{pmatrix} LC(h_1,h_2,h_3) & LC(h_1,h_2,h_3) & LC(h_1,h_2,h_3) & h_4 & \dots & h_n \end{pmatrix}, \end{aligned}$$

where $LC(v_1, \ldots, v_k)$ denotes a linear combination of vectors $v_1, \ldots, v_k$, and the coefficients of the linear combination may be different between all occurences of the notations "LC". Thus,

$$Q_0^T H_k Q_0 = \begin{pmatrix} * & * & * & \\ * & * & * & 0 \\ * & * & * & \\ & 0 & & I \end{pmatrix} \begin{pmatrix} * & * & * & & & & \\ * & * & * & & & & \\ * & * & * & & & & * \\ * & * & * & \ddots & & & \\ 0 & 0 & 0 & \ddots & \ddots & & \\ \vdots & \vdots & \vdots & \ddots & \ddots & \ddots & \\ 0 & 0 & 0 & \ldots & 0 & * & * \end{pmatrix} = \begin{pmatrix} * & * & * & & & & \\ * & * & * & & & & \\ \star & * & * & & & & * \\ \star & \star & * & \ddots & & & \\ 0 & 0 & 0 & \ddots & \ddots & & \\ \vdots & \vdots & \vdots & \ddots & \ddots & \ddots & \\ 0 & 0 & 0 & \ldots & 0 & * & * \end{pmatrix}$$

We want to eliminate the $\star$-entries in the first column, which we can do using Householder matrices of the form

$$\tilde{Q}_j = \begin{pmatrix} I_j & & 0 & & 0 \\ & \star & \star & \star & \\ 0 & \star & \star & \star & 0 \\ & \star & \star & \star & \\ 0 & & 0 & & I_{n-j-3} \end{pmatrix}, \quad j = 1, \ldots, n-3.$$

Then $\tilde{Q}_i e_1 = e_1$ for $i = 1, \ldots, n-3$, so $Qe_1 = Q_0 e_1$. Since $Q_0 (M_k e_1) = \alpha e_1$ and $Q_0^{-1} = Q_0^T = Q_0$, $Q_0 e_1$ is a multiple of $M_k e_1$. On the other hand, we knew that $M_k e_1 = (Q_k Q_{k+1})(R_k R_{k+1}) e_1 = (Q_k Q_{k+1}) \beta e_1$. Then $Q_k Q_{k+1} e_1$ is also a multiple of $M_k e_1$. We deduce that $Q_k Q_{k+1} e_1$ is a multiple of $Q_0 e_1 = Qe_1$, and since the columns of $Q_k Q_{k+1}$ and Q are all normalized we get $Q_k Q_{k+1} e_1 = \pm Q_0 e_1 = Qe_1$. With a suitable choice of signs, $Q_k Q_{k+1} e_1 = Qe_1$.

Now, we have

$$\tilde{H} = Q^T H_k Q \text{ and } H_{k+2} = (Q_k Q_{k+1})^T H_k (Q_k Q_{k+1}).$$

We know that H_k is a Hessenberg matrix, so H_{k+1} and *a fortiori* H_{k+2} as well. Hence, the two equations above are two reductions of H_k to Hessenberg form, and the first column of Q and $Q_k Q_{k+1}$ agree. Because the Hessenberg reduction is completely determined by the first column of the orthogonal matrix, both reductions are the same, hence

$$Q = Q_k Q_{k+1}, \quad \tilde{H} = Q^T H_k Q = H_{k+2}.$$

□

Remark 2.16 – Termination of iteration

We terminate the iteration if for $\ell = n$ (real eigenvalues) or $\ell = n-1$ (complex eigenvalues) it holds:

$$\left| h_{\ell,\ell-1}^{(k)} \right| \leq \text{eps} \left(\left| h_{\ell-1,\ell-1}^{(k)} \right| + \left| h_{\ell,\ell}^{(k)} \right| \right)$$

1. If $\ell = n$, we accept $h_{n,n}^{(k)}$ as the eigenvalue and restart with $\left(h_{ij}^{(k)} \right)_{i,j=1}^{n-1}$.
2. If $\ell = n-1$, we accept the eigenvalues of the lower right 2×2 block of H_k as eigenvalues of A and restart with $\left(h_{ij}^{(k)} \right)_{i,j=1}^{n-2}$.

2.8 Computation of Singular Values

Theorem 2.13 – About singular values

For $A \in \mathbb{R}^{m \times n}$, there exist orthogonal matrices $U \in \mathbb{R}^{m \times m}$ and $V \in \mathbb{R}^{n \times n}$ such that

$$A = U \Sigma V^T$$

with $\Sigma = \text{diag}(\sigma_1, \ldots, \sigma_p) \in \mathbb{R}^{m \times n}$ and $\sigma_1 \geq \sigma_2 \geq \ldots \geq \sigma_p \geq 0$, where $p = \min\{m, n\}$. The σ_j are called singular values of A, which are uniquely determined.

Proof. Let $x \in \mathbb{R}^n$, $y \in \mathbb{R}^m$ with $\|x\|_2 = \|y\|_2 = 1$. Moreover, let $Ax = \sigma y$ with

$$\sigma = \|A\|_2 = \max_{\|v\|=1} \|Av\|_2.$$

Thus, x can be chosen such that $\|Ax\| = \|A\|$. Let V_1 be an orthogonal $n \times n$ matrix with x in the first column (i.e. we extend x to an ONB of $\mathbb{R}^n$) and U_1 an orthogonal $m \times m$ matrix with y in the first column. Then

$$A_1 := U_1^T A V_1 = \left(\begin{array}{c|c} \sigma & \\ 0 & \\ \vdots & \star \\ 0 & \end{array}\right) = \begin{pmatrix} \sigma & w^T \\ 0 & \tilde{A}_1 \end{pmatrix},$$

where $w \in \mathbb{R}^{n-1}$ is a suitable vector and $\tilde{A}_1 \in \mathbb{R}^{(m-1)\times(n-1)}$ is a suitable matrix. We consider

$$\left\| A_1 \begin{pmatrix} \sigma \\ w \end{pmatrix} \right\|_2 = \left\| \begin{pmatrix} \sigma^2 + w^T w \\ \star \end{pmatrix} \right\|_2 \geq \sqrt{\sigma^2 + w^T w}.$$

On the other hand,

$$\|A_1\|_2 = \max_{\|v\|_2=1} \|A_1 v\|_2 = \max_{\|v\|_2=1} \|U_1^T A V_1 v\|_2 = \max_{\|u\|_2=1} \|U_1^T A u\|_2 = \max_{\|z\|_2=1} \|Az\|_2 = \|A\|_2 = \sigma.$$

We deduce $w = 0$, and thus $A_1 = \begin{pmatrix} \sigma & 0 \\ 0 & \tilde{A}_1 \end{pmatrix}$. By induction on $\tilde{A}_1$, the claim follows. □

Remark 2.17

Assume the same notation as in the above theorem and let $U = (u_1, \ldots, u_m)$ and $V = (v_1, \ldots, v_n)$. Furthermore, let $\sigma_1 \geq \ldots \geq \sigma_r > \sigma_{r+1} = \ldots = \sigma_p = 0$. Then

1. $\operatorname{rank} A = r$
2. $\operatorname{Ker} A = \operatorname{Span}\{v_{r+1}, \ldots, v_n\}$ (and $v_{r+1}, \ldots, v_n$ is an ONB of $\operatorname{Ker} A$)
3. $\operatorname{Image} A = \operatorname{Span}\{u_1, \ldots, u_r\}$ (and $u_1, \ldots, u_r$ is an ONB of $\operatorname{Image} A$)
4. $\|A\|_2 = \sigma_1$
5. For the Frobenius norm $\|\cdot\|_F$, we have:

$$\|A\|_F^2 := \sum_{i,j} a_{ij}^2 = \sum_{k=1}^{r} \sigma_k^2.$$

Remark 2.18

It holds that

$$A = U\Sigma V^T = \sum_{i=1}^{r} \sigma_i \underbrace{u_i v_i^T}_{\text{rank}=1}.$$

If the rank of A is small, then much of the information about it can be stored in the vectors u_i and v_i. The best approximation of A of rank $k \leq r$ is:

$$A \approx \sum_{i=1}^{k} \sigma_i u_i v_i^T.$$

This is called the "low-rank approximation of A", which is useful to approximate, e.g., large $N \times N$ matrices by means of r vectors of size N. This information is needed, for example, in data compression, physics, natural sciences (e.g. geoscience, astrophysics)...

Remark 2.19 – Further application

In search engines, one often uses the method of "latent semantic indexing," which is stored in a so-called term-document matrix, where for each search term and each document, it is recorded how often the term appears there. Thus, one replaces the term-document matrix with a low-rank approximation.

Remark 2.20 – Observation on the computation of singular values

One could apply the QR algorithm directly to A^TA (for $n \leq m$) or to AA^T (for $m \leq n$). The product formation is computationally expensive and rounding errors occur. Therefore, we now consider a better algorithm: If P and Q are orthogonal, then A and PAQ have the same singular values, since:

$$A = U\Sigma V^T \iff PAQ = \underbrace{PU}_{\tilde{P}}\Sigma\underbrace{V^TQ}_{\tilde{Q}}.$$

Note that $\tilde{P}$ and $\tilde{Q}$ are orthogonal matrices as products of orthogonal matrices. The following auxiliary theorem shows that we can transform any matrix A with such transformations P and Q into bidiagonal form, and thus the problem reduces to computing the singular values of a bidiagonal matrix.

Theorem 2.14

For $A \in \mathbb{R}^{m\times n}$ (with $m \geq n$ without loss of generality), there exist orthogonal matrices P, Q with

$$PAQ = \begin{pmatrix} B \\ 0 \end{pmatrix}, \quad B = \begin{pmatrix} \ddots & \ddots & 0 \\ 0 & \ddots & \ddots \\ 0 & 0 & \ddots \end{pmatrix}$$

i.e. B is an $n \times n$ bidiagonal matrix.

Proof. We use P_i and Q_i as Householder transformations. By multiplying from the left with P_1, we obtain:

$$P_1A = \left(\begin{array}{c|c} \star & \\ 0 & \\ \vdots & \star \\ 0 & \end{array}\right)$$

Multiplying from the right with $Q_1 = \begin{pmatrix} 1 & 0 \\ 0 & \tilde{Q}_1 \end{pmatrix}$, where $\tilde{Q}_1 \in \mathbb{R}^{(n-1)\times(n-1)}$ is a Householder matrix, gives:

$$P_1AQ_1 = \left(\begin{array}{c|ccc} \star & \star & 0 & \dots & 0 \\ \hline 0 & & & & \\ \vdots & & & \hat{A} & \\ 0 & & & & \end{array}\right)$$

By induction on a smaller matrix $\hat{A}$, the claim follows. □

Remark 2.21

We chase the "bad" elements on and below the subdiagonal until they "fall" out. This method is called "chasing" in the literature.

Remark 2.22 – Computation of Singular Values of a Tridiagonal Matrix

We have a matrix $B^TB =: H$ which is tridiagonal with real nonnegative eigenvalues, and consider a step of the QR algorithm for $\mu := h_{n,n}$:

$$M := H - \mu I = QR.$$

Then $\tilde{H} = RQ + \mu I = Q^T(H - \mu I)Q + \mu I = Q^THQ$. We want to compute the eigenvalues of H without explicitly computing the matrix H or the matrix M. According to Theorem 2.12, $\tilde{H}$ and Q are uniquely determined by the first column of Q. Then $M = QR$ and $Q^T = Q_{n-1} \cdot \ldots \cdot Q_2 \cdot Q_1$ as a product of Householder transformations with

$$Q_j = \begin{pmatrix} I_{j-1} & & & \\ & * & * & \\ & * & * & \\ & & & I_{n-j-1} \end{pmatrix}, \quad j = 1, \ldots, n-1.$$

The first column of Q is $Qe_1 = Q_1e_1$, hence $Me_1 = QRe_1 = \alpha Qe_1 = \alpha Q_1e_1$. This means that Q_1e_1 is a multiple of

$$Me_1 = \begin{pmatrix} h_{1,1} - \mu \\ h_{21} \\ 0 \\ \vdots \\ 0 \end{pmatrix}.$$

Then

$$Q_1e_1 = \pm\frac{Me_1}{\|Me_1\|_2} = \begin{pmatrix} c \\ s \\ 0 \\ \vdots \\ 0 \end{pmatrix} \quad \text{with} \quad c^2 + s^2 = 1, \quad \text{thus:} \quad Q_1 = \left(\begin{array}{cc|c} c & s & 0 \\ s & -c & \\ \hline 0 & & I_{n-2} \end{array}\right).$$

We now transform $Q_1^THQ_1$ to tridiagonal form using Theorem 2.12: since H is symmetric, $Q_1^THQ_1$ is also symmetric, and we are looking for $\tilde{Q}$ such that

$$\tilde{H} = \tilde{Q}^T(Q_1^THQ_1)\tilde{Q} = (Q_1\tilde{Q})^TH(Q_1\tilde{Q}), \quad \tilde{Q} = \begin{pmatrix} I_2 & 0 \\ 0 & \check{Q} \end{pmatrix}, \quad \check{Q}^T\check{Q} = \check{Q}\check{Q}^T = I,$$

where $\tilde{H}$ is Hessenberg symmetric, hence tridiagonal. We have $Q_1\tilde{Q}e_1 = Q_1e_1 = Qe_1$. Thus, $Q_1\tilde{Q} = Q$, since the first column matches. Since $H = B^TB$, $\tilde{H} = Q^TB^TBQ = Q^TB^TP^TPBQ$ is tridiagonal if PBQ is bidiagonal. We transform BQ_1 to bidiagonal form using Theorem 2.14.

Algorithm 2.13 – Golub, Kahan, 1965

Let $A \in \mathbb{R}^{m \times n}$ with $m \geq n$. Then we perform the following algorithm:

1. We transform A with Householder matrices to bidiagonal form:

$$PAQ = \begin{pmatrix} B \\ 0 \end{pmatrix}, \quad B = \begin{pmatrix} \ddots & \ddots & 0 \\ 0 & \ddots & \ddots \\ 0 & 0 & \ddots \end{pmatrix}, \quad \beta = \|B\|_F := \sqrt{\sum_{i,j=1}^{n} b_{ij}^2}.$$

2. We use Remark 2.22 to find the singular values of B, which are the square root of the

eigenvalues of $H = B^T B$. We compute

$$Q_1 = \begin{pmatrix} \star & \star & 0 \\ \star & \star & \\ 0 & & I \end{pmatrix}$$

as in Remark 2.22. We then transform BQ_1 by "chasing" to bidiagonal form $\tilde{B}$. We repeat (with $\tilde{B}$ instead of B) until $|b_{n-1,n}| \leq \beta \cdot \text{eps}$, .i.e. until the off-diagonal term on row $n-1$ is "small enough to be considered zero". When this is done, $h_{n-1,n} \approx 0$, hence $h_{n,n}$ is a good approximation to the eigenvalue of H, which is the square of a singular value of B (which is itself a singular value of A).

3. We reduce the dimension by one and repeat with $(b_{ij})_{i,j=1}^{n-1}$, until we finally reach (almost-) diagonal form.

Remark 2.23

Due to equivalence to the QR algorithm for $H = B^T B$, we obtain cubic convergence, since this matrix is symmetric and tridiagonal.

Chapter 3

Conjugate Gradient Methods

Motivation
The conjugate gradient method can be applied to minimize functions and solve large linear systems (derived from symmetric positive definite matrices).

- *Minimization*
  Let $f : \mathbb{R}^n \supset D \to \mathbb{R}$. We want to determine the minimum of f, i.e., $f'(x) = 0$. This can be computed via Newton-type methods (cf. Numerics I). However, these require knowledge of the Hessian matrix, which is computationally expensive. We seek a method that does not require the second derivative. The conjugate gradient method fulfills this requirement.

- *Solution of a Linear System*
  Let f be quadratic, i.e., $f(x) = \frac{1}{2}x^\top Ax - x^\top b$ with $A \in \mathbb{R}^{n\times n}$ symmetric positive definite and $b \in \mathbb{R}^n$. Then: $f'(x) = 0 \iff Ax - b = 0 \iff Ax = b$.
  The conjugate gradient method is iterative, requires no matrix decomposition, not even the matrix A itself, but only the mapping $x \mapsto Ax$.
  In many applications (such as solving discretizations of finite element methods), very large and sparsely populated matrices often arise. For these, it is not suitable to solve the system of linear equations using decomposition, as the computational effort would be too high and nonzero elements would be filled in.

3.1 One-Dimensional Minimization

Algorithm 3.1 – Golden Section Search (Jack Kiefer, 1953)

Let $f : \mathbb{R} \to \mathbb{R}$. We seek the minimum in $[a, b]$. Choose $v, w \in (a, b)$ such that:

$$\frac{v-a}{w-a} = \frac{w-a}{b-a}, \quad \frac{b-w}{b-v} = \frac{b-v}{b-a}$$

By solving the equations, we obtain:

$$v = a + \frac{3-\sqrt{5}}{2}(b-a), \quad w = a + \frac{\sqrt{5}-1}{2}(b-a).$$

This ratio is precisely the golden ratio. We now proceed:

1. If $f(v) < f(w)$, then the new search interval is $[a, w]$. We can compute that v should replace w, and we only need one additional function evaluation.
2. If $f(v) > f(w)$, then the new search interval is $[v, b]$, and w takes the role of v.

We repeat this interval reduction until $b - a \leq \text{tol}$, where a and b are the interval boundaries in the k-th step.

Algorithm 3.2 – Quadratic Interpolation

We fit a parabola through three points $(x_0, f(x_0))$, $(x_1, f(x_1))$, and $(x_2, f(x_2))$ with $x_0 < x_1 < x_2$ and determine its minimum x^*. We then choose the new interval as either $[x_0, x_1]$ or $[x_1, x_2]$, specifically the one that contains x^*. For the new interval, we proceed iteratively in the same manner and repeat the iteration until the interval length is smaller than a given tolerance tol.

3.2 Steepest Descent Method

Algorithm 3.3 – Gradient Method

e have a function $f : \mathbb{R}^n \to \mathbb{R}$ with $f(x^*) = \min$, and we replace the n-dimensional minimization problem with a sequence of one-dimensional minimization problems.
Starting from an initial point $x_0 \in \mathbb{R}^n$, we search for a direction that leads us towards x^*. We choose the search direction $0 \neq d \in \mathbb{R}^n$ such that it locally represents the steepest descent. We then minimize $f(x_0 + \alpha d)$ with respect to α (which can be done using one-dimensional methods) and set $x_1 := x_0 + \alpha^* d$. We iterate this process and obtain convergence $x_k \to x^*$.

Example 3.1

Let $f(x) = \frac{1}{2}x^\top Ax - x^\top b$ with A symmetric and positive definite. The minimum of $f(x + \alpha d)$ is attained at

$$\alpha = -\frac{d^\top (Ax - b)}{d^\top Ad},$$

which can be easily shown. We note that the gradient of $f(x)$ is given by $\nabla f(x) = Ax - b$.

Remark 3.1 – Choice of Search Direction

Using the Taylor expansion, we obtain:

$$f(x + \alpha d) = f(x) + \alpha f'(x) \cdot d + O(\alpha^2).$$

Locally, the steepest descent direction is given by the negative gradient of f, i.e.,

$$d = -\nabla f(x).$$

Algorithm 1 Steepest Descent Method

Require: Symmetric positive definite matrix $A \in \mathbb{R}^{n \times n}$, right-hand side $b \in \mathbb{R}^n$, initial guess x_0, tolerance $\varepsilon > 0$

Compute $r_0 = b - Ax_0$
for $k = 0, 1, 2, \ldots$ until convergence **do**
  $\alpha_k = \frac{r_k^\top r_k}{r_k^\top A r_k}$
  $x_{k+1} = x_k + \alpha_k r_k$
  $r_{k+1} = r_k - \alpha_k A r_k$
  if $\|r_{k+1}\| < \varepsilon$ **then**
    break
  end if
end for

Remark 3.2

Locally, this method is naturally optimal. However, globally, many iterations are needed to reach the solution. Therefore, we consider another method.

Remark 3.3 – Conjugate Gradients

Let $r := b - Ax$. We consider the vector space: $V_k := \text{span}\{r, Ar, \ldots, A^{k-1}r\}$ and minimize at step k, starting from an initial vector x_0: $f(x_k) = \min_{d \in V_k} f(x_0 + d)$.

Outlook We will see that the iterates x_k can be computed recursively via one-dimensional minimizations.

3.3 Ritz-Galerkin Method

Motivation (Problem Statement)
Let $f : V \to \mathbb{R}$ be a function on a real (possibly infinite-dimensional) vector space V, given by:

$$f(v) = \frac{1}{2}a(v, v) - b(v),$$

where $a : V \times V \to \mathbb{R}$ is a symmetric, positive definite bilinear form, and $b : V \to \mathbb{R}$ is a linear form. We aim to determine the minimum of f.

Remark 3.4

If $V = \mathbb{R}^n$, then $a(v, v) = \frac{1}{2}v^\top Av$, $b(v) = v^\top b$, with A symmetric and positive definite, and $b \in \mathbb{R}^n$. In this case, we know that minimizing $f(v) = \min \iff Av = b$.

Theorem 3.1 – Characterization of Minimization

For $u \in V$, the following holds:

$$f(u) = \min_{v \in V} f(v) \quad \iff \quad a(u, v) = b(v) \quad \forall v \in V.$$

Proof. We set $v := u + \lambda w$ for $\lambda \in \mathbb{R}$ and $w \in V$. Then:

$$f(u + \lambda w) - f(u) = \frac{1}{2}\left(a(u + \lambda w, u + \lambda w) - a(u, u)\right) - \left(b(u + \lambda w) - b(u)\right).$$

Using the bilinearity of a and linearity of b, we have:

$$\begin{aligned} f(u + \lambda w) - f(u) &= \frac{1}{2}\left(a(u, u) + 2\lambda a(u, w) + \lambda^2 a(w, w) - a(u, u)\right) - \left(b(u) + \lambda b(w) - b(u)\right) \\ &= \lambda\left(a(u, w) - b(w)\right) + \frac{1}{2}\lambda^2 a(w, w). \end{aligned}$$

Since $f(u) \leq f(u + \lambda w)$ for all $\lambda \in \mathbb{R}$ and $w \in V$, this is equivalent to $a(u, w) = b(w) \quad \forall w \in V$. □

Idea (Approximate Minimization)
We choose a k-dimensional subspace $V_k \subset V$ and determine the minimum of f in this subspace, i.e., find $u_k \in V_k$ such that:

$$f(u_k) = \min_{v_k \in V_k} f(v_k).$$

This approach is known as the Ritz method.

By the previous theorem, this is equivalent to finding $u_k \in V_k$ satisfying:

$$a(u_k, v_k) = b(v_k) \quad \forall v_k \in V_k.$$

This is called the Galerkin method.

Remark 3.5 – Applications

These approaches are very important in the following examples:

1. To solve $Ax = b$ for A a symmetric, positive definite matrix and A being finite-dimensional and sparsely populated. For the approximation space, we often take $V_k := \langle g, Ag, \dots, A^{k-1}g \rangle$. We will discuss this space in the next section when we talk about the conjugate gradient method. This approach works particularly well when g is an eigenvector of A.
2. Finite elements, which is an important method for solving PDEs. This is discussed in numerical methods for partial differential equations.

Theorem 3.2 – Existence and Uniqueness of the Minimization in Subspace

There exists exactly one solution $u_k \in V_k$ to the minimization problem

$$f(u_k) = \min_{v_k \in V_k} f(v_k)$$

Proof. Let $(\varphi_1, \dots, \varphi_k)$ be a basis of V_k. We now seek

$$u_k = \sum_{i=1}^{k} \mu_i \varphi_i \in V_k$$

such that $a(u_k, v_k) = b(v_k)$ for all v_k that can be represented as $v_k = \sum_{j=1}^{k} \nu_j \varphi_j$, i.e.

$$\sum_{i=1}^{k} \sum_{j=1}^{k} \mu_i \nu_j a(\varphi_i, \varphi_j) = \sum_{j=1}^{k} \nu_j b(\varphi_j) \quad \forall \nu := (\nu_j) \in \mathbb{R}^k.$$

If we set $A := a(\varphi_i, \varphi_j)$, $\mu := (\mu_i)$, and $\ell := b(\varphi_j)$, then in this notation, we must solve the following problem:

$$\nu^\top A \mu = \nu^\top \ell \quad \forall \nu \in \mathbb{R}^k.$$

Since this must hold for all ν, it must particularly hold for $\nu = (e_i)$, and we obtain the new system of linear equations $A\mu = \ell$.

We now compute that A is positive definite:

$$\nu^\top A \nu = \sum_{i,j=1}^{k} \nu_i \nu_j a(\varphi_i, \varphi_j) = a\left(\sum_{i=1}^{k} \nu_i \varphi_i, \sum_{j=1}^{k} \nu_j \varphi_j \right) > 0$$

if $\sum_{i=1}^{k} \nu_i \varphi_i \neq 0$, i.e. $\nu = (\nu_i)_{i=1}^{k} \neq 0$, since φ is a basis.

Thus, A is positive definite (and clearly symmetric) and therefore invertible. This proves the claim.

□

Theorem 3.3 – Cea's Lemma

If $u_* \in V$ is a solution to the minimization problem in V and $u_k \in V_k$ is a solution in the subspace V_k, then it holds in the energy norm $\|v_k\|_a := \sqrt{a(v_k, v_k)}$ that

$$\|u_k - u_*\|_a = \min_{v_k \in V_k} \|v_k - u_*\|_a,$$

i.e., u_k is the best possible approximation to u in this sense.

Remark 3.6

The Ritz–Galerkin approximation thus has, among all elements of the approximation space V_k, the smallest distance (with respect to the energy norm) to the exact solution.

Proof. Since V_k is a subspace of V, it follows for all $v_k \in V$ that $a(u_*, v_k) = b(v_k)$ and $a(u_k, v_k) = b(v_k)$. Subtracting these two equations from each other and using bilinearity, we get:

$$a(u_k - u_*, v_k) = 0 \quad \forall v_k \in V_k.$$

Since we are allowed to replace v_k by $u_k - v_k \in V_k$ and insert a zero, it follows that:

$$a(u_k - u_*, u_k - u_* + u_* - v_k) = 0 \quad \Longleftrightarrow \quad a(u_k - u_*, u_k - u_*) = a(u_k - u_*, v_k - u_*) \quad \forall v_k \in V_k.$$

Using the definition of the energy norm, the above equation, and applying the Cauchy-Schwarz inequality, we obtain:

$$\|u_k - u_*\|_a^2 = a(u_k - u_*, u_k - u_*) \leq \|u_k - u_*\|_a \|v_k - u_*\|_a \quad \forall v_k \in V_k.$$

Thus, it holds that:

$$\|u_k - u_*\|_a \leq \|v_k - u_*\|_a \quad \forall v_k \in V_k.$$

□

3.4 The Conjugate Gradient Method

Motivation
We assume in this chapter that we want to solve a system of linear equations $Ax = b$, where $A \in \mathbb{R}^{n \times n}$ is symmetric and positive definite, and $b \in \mathbb{R}^n$. Typically, n is very large, and A is sparse, as is the case, for example, when solving partial differential equations.
We already know that this is equivalent to minimizing $f(x) = \frac{1}{2} x^\top Ax - x^\top b$.
The conjugate gradient method is often referred to as the cg-method (conjugate gradient method) in the literature.

Historical Note. This method was introduced by Hestenes and Stiefel in 1952.

Definition 3.1 – Krylov Subspace

The vector space $\mathcal{K}_k(A, r)$

$$\mathcal{K}_k(A, r) := \operatorname{span}\{r, Ar, \ldots, A^{k-1} r\}$$

is called the k-th Krylov subspace.

Remark 3.7

From the gradient method (see Remark 3.2), we already know that the vector space V_k

$$V_k := \operatorname{span}\{r_0, Ar_0, \ldots, A^{k-1} r_0\} \quad \text{with} \quad r_0 = -g_0 := b - Ax_0 = -\nabla f(x_0)$$

for a starting vector x_0 is a k-th Krylov subspace.

Idea (Ritz-Galerkin Approach)
We now apply the Ritz-Galerkin approach to the approximation space V_k, i.e., we seek an $x_k \in x_0 + V_k$ such that

$$f(x_k) = \min_{v_k \in V_k} f(x_0 + v_k).$$

Remark 3.8

Since this space is generally k-dimensional, it is difficult to calculate such minima. However, we will show in the following that it is sufficient to solve k one-dimensional problems recursively in order to reach the minimum in V_k.
For the derivation, we assume, without loss of generality, that $x_0 = 0$ (otherwise, consider $y = x - x_0$; then the problem would be to solve $Ay = b - Ax_0$ for the starting value $y_0 = 0$).

Definition 3.2 – Inner Product Induced by A

We define the inner product induced by A as $\langle u, v\rangle_A := u^\top Av = \langle Au, v\rangle$, where $\langle \cdot, \cdot \rangle$ denotes the Euclidean inner product.

Idea (Decomposition of the Krylov Subspace)
We decompose the Krylov subspace in the following manner:

$$V_{k+1} = V_k \oplus V_k^\perp$$

with $V_k^\perp = \{w \in V_{k+1} \mid w^\top Av = 0 \ \forall v \in V_k\}$, *which is often referred to in the literature as the A-orthogonal complement of V_k in V_{k+1}, and by construction, it is orthogonal with respect to the inner product induced by A.*

Construction (Derivation of the Algorithm)
It is clear that $\dim V_k^\perp \leq 1$.

Thus, we can uniquely write $x_{k+1} = u + w$ with $u \in V_k$ and $w \in V_k^\perp$. Therefore, x_{k+1} is determined by the conditions that

$$x_{k+1} \in V_{k+1}, \quad f(x_{k+1}) = \min_{v_{k+1} \in V_{k+1}} f(v_{k+1}),$$

which is equivalent to finding a $v_{k+1} \in V_{k+1}$ such that

$$\langle Ax_{k+1}, v_{k+1}\rangle = \langle b, v_{k+1}\rangle \quad \forall v_{k+1} \in V_{k+1}.$$

Since $V_k \subseteq V_{k+1}$, it follows that

$$\langle Ax_{k+1}, v_k\rangle = \langle b, v_k\rangle \quad \forall v_k \in V_k.$$

However, the left-hand side simplifies due to orthogonality:

$$\langle Ax_{k+1}, v_k\rangle = \langle Au, v_k\rangle + \underbrace{\langle Aw, v_k\rangle}_{=0}.$$

On the other hand, $x_k \in V_k$ is uniquely determined by

$$\langle Ax_k, v_k\rangle = \langle b, v_k\rangle \quad \forall v_k \in V_k.$$

Thus, $u = x_k$, and we obtain that $x_{k+1} = x_k + w$ with $w \in V_k^\perp$, or equivalently, for d_k (e.g., a basis of $V_k^\perp$) with $V_k^\perp = \mathbb{R}d_k$, we have for $\alpha_k \in \mathbb{R}$

$$x_{k+1} = x_k + \alpha_k d_k.$$

Thus, x_{k+1} is the solution to a one-dimensional minimization problem:

$$f(x_{k+1}) = \min_{\alpha_k \in \mathbb{R}} f(x_k + \alpha_k d_k).$$

We need to find the minimum of a quadratic function, for which we already know the optimal α_k from Chapter 2, namely

$$\alpha_k = \frac{\langle d_k, b - Ax_k\rangle}{\langle Ad_k, d_k\rangle} = \frac{\langle d_k, r_k\rangle}{\langle Ad_k, d_k\rangle}.$$

Construction (Determination of the Search Direction)
We want to determine the search direction d_k as a basis vector of $V_k^\perp$. To do this, we consider different cases:

1. *If $V_{k+1} = V_k$, then $V_k^\perp = 0$, and thus $d_k = 0$. In this case, for $x_k \in V_k$, we also have $Ax_k \in V_{k+1} = V_k$. But it holds that:*

$$\langle Ax_k, v_k\rangle = \langle b, v_k\rangle \quad \forall v_k \in V_k, \quad \langle Ax_k - b, v_k\rangle = 0 \quad \forall v_k \in V_k.$$

Since $Ax_k - b \in V_k$, the only solution to the zero vector in the first component of the scalar product is $Ax_k = b$. Therefore, x_k is the solution to the linear system.

2. *Let $\dim V_k^\perp = 1$. Then $r_k = b - Ax_k \neq 0$, so $r_k \notin V_k$, but $r_k \in V_{k+1}$. We now decompose $r_k = c_k + d_k$, where $c_k \in V_k$ and $d_k \in V_k^\perp$, and in particular, $d_k \neq 0$. Thus, c_k is the A-orthogonal projection of r_k onto V_k with respect to the inner product induced by A.*

Let $\{d_0, d_1, \ldots, d_{k-1}\}$ be an orthonormal basis (with respect to A) of V_k, i.e., $\langle Ad_i, d_j\rangle = 0$ for $i \neq j$. By the Gram-Schmidt process, we obtain for c_k:

$$c_k = \sum_{i=0}^{k-1} \frac{\langle Ad_i, r_k\rangle}{\langle Ad_i, d_i\rangle} d_i.$$

Since $Ad_j \in V_k$ for $j \leq k-2$, due to the definition of x_k:

$$\langle r_k, Ad_j\rangle = \langle b - Ax_k, Ad_j\rangle = 0.$$

This greatly simplifies the sum, and we ultimately obtain:

$$d_k = r_k - c_k = r_k - \frac{\langle Ad_{k-1}, r_k\rangle}{\langle Ad_{k-1}, d_{k-1}\rangle} d_{k-1}.$$

Remark 3.9 – Computational Cost of the Iteration

We want to compute $r_{k+1} = Ax_{k+1} - b$. However, the cost of matrix multiplication is too high. Based on the definition of x_{k+1}, we have:

$$r_{k+1} = b - A(x_k + \alpha_k d_k) = (b - Ax_k) - \alpha_k Ad_k = r_k - \alpha_k Ad_k.$$

The product Ad_k is needed anyway in the next step for calculating the new search direction d_{k+1}, so this calculation is quite optimal in terms of computational cost.

Algorithm 3.4 – Conjugate Gradient Method

Let $x_0 \in \mathbb{R}^n$ be an arbitrary initial vector. Then, $d_0 = r_0 = b - Ax_0$. For $k = 0, 1, 2, \ldots$ (until $\|b - Ax_k\| \leq \text{tol} \cdot \|b\|$), the following recursion is performed:

$$\begin{aligned} x_{k+1} &= x_k + \alpha_k d_k, \\ r_{k+1} &= r_k - \alpha_k Ad_k, \\ d_{k+1} &= r_{k+1} + \beta_k d_k, \end{aligned}$$

where the coefficients are given by:

$$\begin{aligned} \alpha_k &= \frac{\langle d_k, r_k\rangle}{\langle Ad_k, d_k\rangle}, \\ \beta_k &= -\frac{\langle Ad_k, r_{k+1}\rangle}{\langle Ad_k, d_k\rangle}. \end{aligned}$$

By construction, we have:

$$f(x_k) = \min_{v_k \in V_k} f(x_0 + v_k).$$

Remark 3.10 – Computational Cost

We require only a single matrix multiplication and three scalar products per step.

Remark 3.11

We will show in an exercise that we can write:

$$\alpha_k = \frac{\langle r_k, r_k \rangle}{\langle Ad_k, d_k \rangle}, \quad \beta_k = \frac{\langle r_{k+1}, r_{k+1} \rangle}{\langle r_k, r_k \rangle}.$$

Thus, we only need two distinct scalar products per step.

Algorithm 2 Conjugate Gradient Method

Require: $A \in \mathbb{R}^{n \times n}$ SPD, $b \in \mathbb{R}^n$, initial guess x_0, tolerance $\varepsilon > 0$

$r_0 = b - Ax_0$
$d_0 = r_0$
for $k = 0, 1, 2, \ldots$ **do**
  $\alpha_k = \frac{r_k^\top r_k}{d_k^\top A d_k}$
  $x_{k+1} = x_k + \alpha_k d_k$
  $r_{k+1} = r_k - \alpha_k A d_k$
  if $\|r_{k+1}\| < \varepsilon$ **then**
    break
  end if
  $\beta_k = \frac{r_{k+1}^\top r_{k+1}}{r_k^\top r_k}$
  $d_{k+1} = r_{k+1} + \beta_k d_k$
end for

3.5 Error Analysis of the CG Method

Theorem 3.4 – Finite Termination of CG Method

After at most n steps of the CG method, we obtain the exact solution to the linear system

$$Ax^* = b, \quad \text{that is,} \quad \exists k \leq n \text{ such that } x_k = x^*.$$

Proof. It is $V_1 \subseteq V_2 \subseteq \cdots \subseteq \mathbb{R}^n$, so there exists a $k \leq n$ with $V_{k+1} = V_k$. Hence, x_k is exact. □

Remark 3.12

For practical purposes, this theorem is uninteresting since we usually use this method for very large n. Moreover, we only require a very good approximation to the solution, for example, $\|x_k - x^*\| \leq \text{tol} \cdot \|x_0 - x^*\|$ for $k \leq n$. Furthermore, rounding errors can worsen the result at each step.

Theorem 3.5 – Error Estimate for the CG Method

For all polynomials q_k of degree $\deg q_k \leq k$ with $q_k(0) = 1$, it holds that:

$$\|x_k - x^*\|_A \leq \max_{\lambda \in \text{EV}(A)} |q_k(\lambda)| \cdot \|x_0 - x^*\|_A,$$

where x^* is the exact solution.

Remark 3.13 – Observation

Let A be a matrix of any dimension with exactly k distinct eigenvalues. Then, the CG method will provide the exact solution after at most k steps. We consider the polynomial that has the eigenvalues as roots. Moreover, the CG method converges quickly when the eigenvalues are clustered together.

Proof. Without loss of generality, assume $x_0 = 0$ (otherwise, we consider $y = x - x_0$, and for $Ay = b - Ax_0$, we would have $y_0 = 0$).

1. From the Cea's theorem, we know that

$$\|x_k - x^*\|_A = \min_{v_k \in V_k} \|v_k - x^*\|_A.$$

For a $V_k \ni v_k = p_{k-1}(A)b$ and $\deg p_{k-1} \leq k-1$, it follows from $Ax^* = b$:

$$x^* - v_k = \underbrace{(I - p_{k-1}(A)A)}_{=:q_k(A)} x^*,$$

where q_k is a polynomial of degree $\deg q_k \leq k$ with $q_k(0) = 1$. Thus, we get:

$$\|x_k - x^*\|_A = \min_{\deg q_k \leq k, q_k(0)=1} \|q_k(A)x^*\|_A.$$

2. Since A is symmetric and positive definite, we can diagonalize A using an orthogonal matrix, $A = Q\Lambda Q^\top$, where Q is orthogonal and $\Lambda = \operatorname{diag}(\lambda_1, \ldots, \lambda_n)$. Due to positive definiteness, $\lambda_i > 0$. Then, we compute:

$$\begin{aligned}
\|q_k(A)x^*\|_A^2 &= \langle Aq_k(A)x^*, q_k(A)x^* \rangle = \langle Q\Lambda Q^\top Q q_k(\Lambda) Q^\top x^*, Q q_k(\Lambda) Q^\top x^* \rangle, \\
&= \langle \Lambda q_k(\Lambda) Q^\top x^*, q_k(\Lambda) Q^\top x^* \rangle = \langle \Lambda^{\frac{1}{2}} \Lambda^{\frac{1}{2}} q_k(\Lambda) Q^\top x^*, q_k(\Lambda) Q^\top x^* \rangle \\
&= \langle q_k(\Lambda) \Lambda^{\frac{1}{2}} Q^\top x^*, q_k(\Lambda) \Lambda^{\frac{1}{2}} Q^\top x^* \rangle = \|q_k(\Lambda) \Lambda^{\frac{1}{2}} Q^\top x^*\|_2^2 \\
&\leq \underbrace{\|q_k(\Lambda)\|_2^2}_{=(\max_{i=1\cdots,n} |q_k(\lambda_i)|)^2} \|\Lambda^{\frac{1}{2}} Q^\top x^*\|_2^2
\end{aligned}$$

We now need to find a clever expression for $\|\Lambda^{\frac{1}{2}} Q^\top x^*\|_2^2$:

$$\|\Lambda^{\frac{1}{2}} Q^\top x^*\|_2^2 = \left\langle \Lambda^{\frac{1}{2}} Q^\top x^*, \Lambda^{\frac{1}{2}} Q^\top x^* \right\rangle = \langle \Lambda Q^\top x^*, Q^\top x^* \rangle = \langle Q\Lambda Q^\top x^*, x^* \rangle = \langle Ax^*, x^* \rangle = \|x^*\|_A^2$$

Thus, we obtain:

$$\|x_k - x^*\|_A \leq \|q_k(A)x^*\|_A \leq \max_{i=1,\cdots,n} |q_k(\lambda_i)| \|x^*\|_A.$$

□

Remark 3.14 – Recap

1. For the condition number with respect to the Euclidean norm for a symmetric positive matrix, we have: $\operatorname{cond}_2(A) = \frac{\lambda_{\max}(A)}{\lambda_{\min}(A)}$.
2. Among all polynomials p_k of degree at most k satisfying $p_k(t_0) = 1$, the Chebyshev polynomial $T_k(t)/T_k(t_0)$ attains the smallest possible maximum absolute value on the interval $[-1, 1]$ (see Exercise 7).

Theorem 3.6 – Condition Number Dependent Error Estimate

For the condition number $\kappa = \text{cond}_2(A)$, it holds that:

$$\|x_k - x^*\|_A \leq 2 \cdot \left(\frac{\sqrt{\kappa}-1}{\sqrt{\kappa}+1}\right)^k \cdot \|x_0 - x^*\|_A.$$

Proof. Let all eigenvalues of A be in $[\lambda_{\min}, \lambda_{\max}]$ with $0 < \lambda_{\min} < \lambda_{\max}$ and then $\kappa = \frac{\lambda_{\max}}{\lambda_{\min}}$. From the previous theorem, for all polynomials q_k of degree $\deg q_k \leq k$ and $q_k(0) = 1$, we have:

$$\|x_k - x\|_A \leq \max_{\lambda \in \text{EV}(A)} |q_k(\lambda)| \cdot \|x_0 - x\|_A.$$

We now transform $[\lambda_{\min}, \lambda_{\max}]$ to $[-1, 1]$ by the mapping:

$$\lambda \mapsto t = \frac{2\lambda - \lambda_{\min} - \lambda_{\max}}{\lambda_{\max} - \lambda_{\min}}, \quad \text{in particular} \quad 0 \mapsto -\frac{\lambda_{\max} + \lambda_{\min}}{\lambda_{\max} - \lambda_{\min}} = -\frac{\kappa+1}{\kappa-1} =: t_0 < -1.$$

We then get the following estimate:

$$\|x_k - x\|_A \leq \max_{t \in [-1,1]} |p_k(t)| \cdot \|x_0 - x\|_A,$$

for all polynomials $p_k(t) := q_k(\lambda)$ with $\deg p_k \leq k$ and $p_k(t_0) = 1$. From Numerics I, we know that the right-hand side of the estimate is minimized for the Chebyshev polynomial $T_k(t)/T_k(t_0)$, and thus we have:

$$\|x_k - x\|_A \leq \frac{1}{|T_k(t_0)|} \cdot \|x_0 - x\|_A.$$

We now use that for a Chebyshev polynomial, the following holds:

$$T_k(t) = \frac{1}{2}\left(\left(t - \sqrt{t^2-1}\right)^k + \left(t + \sqrt{t^2-1}\right)^k\right) \Rightarrow |T_k(t_0)| = T_k(|t_0|) \geq \frac{1}{2}\left(|t_0| + \sqrt{t_0^2-1}\right)^k.$$

We calculate as follows:

$$\begin{aligned} |t_0| + \sqrt{t_0^2-1} &= \frac{\kappa+1}{\kappa-1} + \sqrt{\frac{\kappa^2+2\kappa+1-(\kappa^2-2\kappa+1)}{(\kappa-1)^2}} = \frac{\kappa+1+2\sqrt{\kappa}}{\kappa-1} \\ &= \frac{(\sqrt{\kappa}+1)^2}{(\sqrt{\kappa}+1)(\sqrt{\kappa}-1)} = \frac{\sqrt{\kappa}+1}{\sqrt{\kappa}-1} \end{aligned}$$

Finally, we get:

$$|T_k(t_0)| \geq \frac{1}{2}\left(\frac{\sqrt{\kappa}+1}{\sqrt{\kappa}-1}\right)^k, \quad \text{and thus} \quad \|x_k - x\|_A \leq 2\left(\frac{\sqrt{\kappa}+1}{\sqrt{\kappa}-1}\right)^k \|x_0 - x\|_A.$$

□

Remark 3.15

1. For the steepest descent method, the corresponding statement is:

$$\|x_k - x\|_A \leq \left(\frac{\kappa-1}{\kappa+1}\right)^k \|x_0 - x\|_A.$$

   For a large condition number, this expression is much closer to one than the first expression, meaning that we have very slow convergence.

2. Rapid error reduction in the CG method occurs if:
   a) the condition is small, or
   b) the eigenvalues of A are clustered in a few groups.

3.6 Preconditioned CG Method

Idea
We now want to solve $Ax = b$ for a symmetric and positive definite matrix A. Typically, A is very large and sparsely populated.

If B is symmetric and positive definite, we set $B \approx A^{-1}$ such that $x \mapsto Bx$ is easy to compute. This gives us an equivalent linear system $BAx = Bb$. We now require that $cond_2(BA) \ll cond_2(A)$ so that the CG method converges faster.

Remark 3.16 – Choice of Preconditioner

We have the problem of finding B that satisfies these requirements. Therefore, we perform an incomplete Cholesky decomposition. For $k = 1, \dots, n$, we compute:

$$\tilde{\ell}_{kk} = \sqrt{a_{kk} - \sum_{j=1}^{k-1} \tilde{\ell}_{kj}^2}$$

and for $i = k+1, \dots, n$, we compute:

$$\tilde{\ell}_{ik} = \begin{cases} \frac{1}{\tilde{\ell}_{kk}} \left(a_{ik} - \sum_{j=1}^{i-1} \ell_{ij}\ell_{kj} \right), & a_{ik} \neq 0, \\ 0 & a_{ik} = 0 \end{cases}$$

This ensures that only non-zero terms of a_{ij} are summed, preserving the sparsity of A. We have $A = \tilde{L}\tilde{L}^\top + R$, where R is the remainder matrix. $\tilde{L}\tilde{L}^\top$ is symmetric and positive definite, making it a candidate for B. We do not fill in the non-zero entries, which generally causes the eigenvalues of BA to cluster, leading to faster convergence of the CG method.
In particular, the mapping $x \mapsto Bx$ is relatively easy to compute, as only sparse triangular systems need to be solved.

Remark 3.17

The difficulty is that BA is not symmetric, even if A and B are both symmetric and positive definite. We seek a workaround: If we can write $B = CC^\top$ (e.g., through the full Cholesky decomposition), then:

$$CC^\top Ax = CC^\top b \iff C^\top ACC^{-1}x = C^\top b \iff \tilde{A}\tilde{x} = \tilde{b}$$

where $\tilde{x} = C^{-1}x$, $\tilde{b} = C^\top b$, and $\tilde{A} := C^\top AC$ is again symmetric and positive definite, allowing us to apply the CG method.

Construction (Formal Application to the CG Method)
We start with $\tilde{x}_0 = C^{-1}x_0$ in the algorithm and apply the procedure to all quantities with tildes. We show that we can do this without knowing C. We compute:

$$\tilde{x}_{k+1} = \tilde{x}_k + \alpha_k \tilde{d}_k \iff C^{-1}x_{k+1} = C^{-1}x_k + \alpha_k C^{-1}d_k$$

where $d_k := C\tilde{d}_k$. Multiplying by C gives:

$$x_{k+1} = x_k + \tilde{\alpha}_k d_k$$

Now, we express $\tilde{r}_k$ as:

$$\tilde{r}_k = \tilde{A}\tilde{x}_k - \tilde{b} = C^\top ACC^{-1}x_k - C^\top b = C^\top (Ax_k - b) = C^\top r_k$$

Thus, with the CG algorithm:

$$\tilde{r}_{k+1} = \tilde{r}_k - \alpha_k \tilde{A}\tilde{d}_k \iff C^\top r_{k+1} = C^\top r_k - \alpha_k C^\top ACC^{-1}d_k \iff r_{k+1} = r_k - \alpha_k Ad_k$$

So, the determination of r_k remains unchanged. For the search directions:

$$\tilde{d}_{k+1} = \tilde{r}_{k+1} + \beta_k \tilde{d}_k \iff C^{-1}d_{k+1} = C^\top r_{k+1} + \beta_k C^{-1} d_k \iff d_{k+1} = Br_{k+1} + \beta_k d_k$$

Finally, we show that no C appears in the coefficients:

$$\alpha_k = \frac{\langle \tilde{r}_k, \tilde{r}_k \rangle}{\langle \tilde{A}\tilde{d}_k, \tilde{d}_k \rangle} = \frac{\langle C^\top r_k, C^\top r_k \rangle}{\langle C^\top ACC^{-1}d_k, C^{-1}d_k \rangle} = \frac{\langle r_k, CC^\top r_k \rangle}{\langle Ad_k, d_k \rangle} = \frac{\langle r_k, Br_k \rangle}{\langle Ad_k, d_k \rangle},$$
$$\beta_k = \frac{\langle \tilde{r}_{k+1}, \tilde{r}_{k+1} \rangle}{\langle \tilde{r}_k, \tilde{r}_k \rangle} = \frac{\langle r_{k+1}, Br_{k+1} \rangle}{\langle r_k, Br_k \rangle}.$$

Algorithm 3.5 – Preconditioned CG Method

We want to solve $Ax = b$ with A symmetric and positive definite, and use B symmetric and positive definite with $B \approx A^{-1}$.
We choose $x_0 \in \mathbb{R}^n$ arbitrarily and $r_0 = b - Ax_0$, $d_0 = Br_0$ $\rho_0 = \langle Br_0, r_0 \rangle$, and then iterate for $k = 0, 1, 2, \ldots$ (until $\sqrt{\rho_k} \leq \text{tol} \cdot \|b\|_2$):

$$x_{k+1} = x_k + \alpha_k d_k$$
$$r_{k+1} = r_k - \alpha_k A d_k$$
$$d_{k+1} = Br_{k+1} + \beta_k d_k$$

with the coefficients:

$$\alpha_k = \frac{\rho_k}{\langle Ad_k, d_k \rangle}, \quad \beta_k = \frac{\rho_{k+1}}{\rho_k}, \quad \rho_{k+1} = \langle Br_{k+1}, r_{k+1} \rangle.$$

Algorithm 3 Preconditioned Conjugate Gradient Method

Require: $A \in \mathbb{R}^{n \times n}$ SPD, $b \in \mathbb{R}^n$, preconditioner $B \approx A^{-1}$ (SPD), initial guess x_0, tolerance $\varepsilon > 0$

$r_0 = b - Ax_0$
$d_0 = Br_0$
for $k = 0, 1, 2, \ldots$ until convergence **do**
  $\alpha_k = \frac{r_k^\top Br_k}{d_k^\top Ad_k}$
  $x_{k+1} = x_k + \alpha_k d_k$
  $r_{k+1} = r_k - \alpha_k Ad_k$
  if $\|r_{k+1}\| < \varepsilon$ **then**
    break
  end if
  $\beta_k = \frac{r_{k+1}^\top Br_{k+1}}{r_k^\top Br_k}$
  $d_{k+1} = Br_{k+1} + \beta_k d_k$
end for

Remark 3.18 – Implementation

For the implementation of the method, we store only 5 vectors: (x, r, Br, d, Ad). Furthermore, we note that we do not need the matrices A and B themselves, but only the mappings $u \mapsto Au$ and $v \mapsto Bv$.

Theorem 3.7

Let $0 < \gamma \langle v, B^{-1}v \rangle \leq \|v\|_A \leq \Gamma \langle v, B^{-1}v \rangle$ for all $v \in \mathbb{R}^n$. Then:

$$\|x_k - x^*\|_A \leq 2 \left(\frac{\sqrt{\tilde{\kappa}} - 1}{\sqrt{\tilde{\kappa}} + 1} \right)^k \|x_0 - x^*\|_A$$

with $\tilde{\kappa} = \text{cond}_2(BA) \leq \frac{\Gamma}{\gamma}$.

Proof. 1. First, we compute using $\tilde{A} = C^\top AC$:

$$\begin{aligned}\|x_k - x^*\|_A^2 &= \langle x_k - x^*, A(x_k - x^*)\rangle = \langle C(\tilde{x}_k - \tilde{x}^*), AC(\tilde{x}_k - \tilde{x}^*)\rangle \\ &= \langle \tilde{x}_k - \tilde{x}^*, \tilde{A}(\tilde{x}_k - \tilde{x}^*)\rangle = \|\tilde{x}_k - \tilde{x}^*\|_{\tilde{A}}^2.\end{aligned}$$

Using the theorem on condition-dependent error estimation for the tilde-system, we immediately obtain the estimate for our theorem with $\tilde{\kappa} = \text{cond}_2(\tilde{A})$.

2. By the definition of the condition number:

$$\text{cond}_2(\tilde{A}) = \sqrt{\frac{\lambda_{\max}(\tilde{A}^\top \tilde{A})}{\lambda_{\min}(\tilde{A}^\top \tilde{A})}}, \quad \text{cond}_2(BA) = \sqrt{\frac{\lambda_{\max}((BA)^\top (BA))}{\lambda_{\min}((BA)^\top (BA))}}.$$

To compute the condition number of BA, we first compute:

$$\begin{aligned}(BA)^\top BA = ABBA = ACC^\top C^\top AC = (C^\top)^{-1} C^\top ACC^\top CC^\top ACC^{-1} \\ = (C^\top)^{-1} \tilde{A} C^\top \cdot C\tilde{A}C^{-1} = (C\tilde{A}C^{-1})^\top (C\tilde{A}C^{-1})\end{aligned}$$

But this has exactly the same eigenvalues as $\tilde{A}$ due to the change of basis. Thus, we have $\text{cond}_2(BA) = \text{cond}_2(\tilde{A})$.

3. Clearly, it holds that:

$$\lambda_{\min}(\tilde{A})\langle v, B^{-1}v\rangle \leq \langle v, Av\rangle \leq \lambda_{\max}(\tilde{A})\langle v, B^{-1}v\rangle, \quad \forall v \in \mathbb{R}^n,$$

and therefore, $\gamma \leq \lambda_{\min}(\tilde{A})$, $\Gamma \geq \lambda_{\max}(\tilde{A})$.

□

3.7 Conjugate Gradient Method for Minimizing Non-Quadratic Functions

Motivation
We now consider a function $f : \mathbb{R}^n \to \mathbb{R}$ and seek a (local) minimum of f, where f is generally not quadratic.

Review From undergraduate analysis, we know that if $x^* \in \mathbb{R}^n$ is a local minimum of f (with f twice continuously differentiable), then $g(x^*) = \nabla f(x^*) = 0$ and the Hessian matrix $H(x^*) = \nabla^2 f(x^*)$ is symmetric and positive semi-definite.

Conversely, if $\nabla f(x^*) = 0$ and the Hessian matrix is positive definite, then x^* is a local minimum.

Idea
We choose an initial point x near x^*. Then, using the Taylor expansion:

$$f(x) = f(x^*) + \underbrace{\nabla f(x^*)(x - x^*)}_{=0} + \frac{1}{2}(x - x^*)^\top \nabla^2 f(x^*)(x - x^*) + O(\|x - x^*\|^3)$$

locally behaves like a quadratic function. We thus modify methods that solve quadratic minimization problems well, which leads to the conjugate gradient (CG) method. If x is far from x^*, at least we move in the descent direction of the CG method.

Algorithm 3.6 – Modification of the CG Method

We start with $x_0 \in \mathbb{R}^n$ arbitrary and choose $d_0 = -g_0 = -\nabla f(x_0)$, and iterate for $k = 0, 1, 2, \ldots$, until, for example, the stopping criterion $\|g_k\| \leq \text{tol} \cdot \|g_0\|$ is satisfied: We select the minimum of f in the search direction d_k, i.e., we find α_k for

$$x_{k+1} := x_k + \alpha_k d_k \quad \text{with} \quad f(x_{k+1}) = \min_{\alpha_k} f(x_k + \alpha_k d_k),$$

which is a one-dimensional minimization problem that we can also approximate, for example, using the criterion:

$$|\langle \nabla f(x_k + \alpha_k d_k), d_k \rangle| \leq \sigma \, |\langle \nabla f(x_k), d_k \rangle|$$

for some $\sigma \in \mathbb{R}$. We then set $g_{k+1} := \nabla f(x_{k+1})$.
Next, we update the search direction as $d_{k+1} = -g_{k+1} + \beta_k d_k$.

Algorithm 4 Nonlinear Conjugate Gradient Method (Polak–Ribière)

Require: Objective function $f : \mathbb{R}^n \to \mathbb{R}$, gradient ∇f, initial guess x_0, tolerance $\varepsilon > 0$
$g_0 = \nabla f(x_0)$
$d_0 = -g_0$
for $k = 0, 1, 2, \ldots$ until convergence **do**
  Find $\alpha_k > 0$ satisfying sufficient decrease condition (e.g., Armijo/Wolfe) for $f(x_k + \alpha_k d_k)$
  $x_{k+1} = x_k + \alpha_k d_k$
  $g_{k+1} = \nabla f(x_{k+1})$
  if $\|g_{k+1}\| < \varepsilon \cdot \|g_0\|$ **then**
    break
  end if
  $\beta_k = \frac{(g_{k+1} - g_k)^\top g_{k+1}}{g_k^\top g_k}$ ▷ Polak–Ribière
  $d_{k+1} = -g_{k+1} + \beta_k d_k$
end for

Remark 3.19

There are several different approaches in the literature for the coefficient β_k in the update of the search direction $d_{k+1} = -g_{k+1} + \beta_k d_k$:

1. **Fletcher–Reeves:** $\beta_k^{\text{FR}} = \dfrac{\langle g_{k+1}, g_{k+1} \rangle}{\langle g_k, g_k \rangle}$
2. **Polak–Ribière:** $\beta_k^{\text{PR}} = \dfrac{\langle g_{k+1} - g_k, g_{k+1} \rangle}{\langle g_k, g_k \rangle}$
3. **Hestenes–Stiefel:** $\beta_k^{\text{HS}} = \dfrac{\langle g_{k+1} - g_k, g_{k+1} \rangle}{\langle g_{k+1} - g_k, d_k \rangle}$
4. **Dai–Yuan:** $\beta_k^{\text{DY}} = \dfrac{\langle g_{k+1}, g_{k+1} \rangle}{\langle g_{k+1} - g_k, d_k \rangle}$

In the quadratic case, all these formulas are equivalent to $\beta_k = \frac{\langle r_{k+1}, r_{k+1} \rangle}{\langle r_k, r_k \rangle}$ ($r_k = -g_k$), due to the property that $\langle r_{k+1} r_k \rangle = 0$ and $\langle r_k, r_k \rangle = \langle r_k, d_k \rangle$
In the non-quadratic case, the different choices of β_k lead to different convergence behaviors:

- The Polak–Ribière formula is often preferred in practice, because if the algorithm "stalls," i.e., $x_{k+1} \approx x_k$ and $g_{k+1} \approx g_k$, then for Polak-Ribiere, $\beta_k = 0$, and there is a new search direction, meaning the method starts over with $d_{k+1} = -g_{k+1}$. On the other hand, for Fletcher–Reeves, $\beta_k = 1$, and the descent is not guaranteed.

- The Hestenes–Stiefel and Dai–Yuan formulas offer alternative trade-offs between descent guarantees and practical performance. Dai–Yuan is designed to improve global convergence and ensures the direction remains a descent direction under mild conditions.

In practice, hybrid strategies (e.g., combining Polak–Ribière and Dai–Yuan, or using a safeguarded $\max(0, \beta_k)$) are also commonly used to balance convergence speed and stability.

Algorithm 3.7 – Preconditioned CG Method

We can view the preconditioned CG method as an acceleration of the simplified Newton method. It is

$$\nabla f(x) = 0, \quad x_{k+1} = x_k + \alpha_k \, \Delta x_k$$

with

$$\Delta x_k = -\left(\nabla^2 f(x_0)\right)^{-1} \underbrace{\nabla f(x_k)}_{=g(x_k)}.$$

Here, we use the Hessian matrix $H(x_0) \approx L_0 L_0^\top$ as possibly an incomplete Cholesky decomposition. Then $H(x_0)^{-1} \approx B := L_0^{-T} L_0^{-1} \approx C C^\top$, where B is a preconditioner. It follows that $g(x) = 0 \iff \tilde{g}(\tilde{x}) = C^\top \cdot g(C \underbrace{C^{-1} x}_{\tilde{x}}) = 0$. Furthermore, $\tilde{H} = C^\top H C$.

We can now proceed as in Chapter 6 and obtain the new search direction:

$$d_{k+1} = B r_{k+1} + \beta_k d_k,$$

with

$$\beta_k = \frac{\langle g_{k+1} - g_k, B g_{k+1} \rangle}{\langle g_k, B g_k \rangle}.$$

Algorithm 5 Preconditioned Conjugate Gradient Method for Nonlinear Optimization

Require: Objective function $f : \mathbb{R}^n \to \mathbb{R}$, gradient ∇f, preconditioner $B \approx H(x_0)^{-1}$ (SPD), initial guess x_0, tolerance $\varepsilon > 0$

$g_0 = \nabla f(x_0)$
$d_0 = -B g_0$
for $k = 0, 1, 2, \ldots$ until convergence **do**
  Find $\alpha_k > 0$ satisfying sufficient decrease (e.g. Wolfe/Armijo) for $f(x_k + \alpha_k d_k)$
  $x_{k+1} = x_k + \alpha_k d_k$
  $g_{k+1} = \nabla f(x_{k+1})$
  if $\|g_{k+1}\| < \varepsilon \cdot \|g_0\|$ **then**
    break
  end if
  $\beta_k = \frac{(g_{k+1} - g_k)^\top B g_{k+1}}{g_k^\top B g_k}$ ▷ Polak–Ribière
  $d_{k+1} = -B g_{k+1} + \beta_k d_k$
end for

Remark 3.20

We could also show that the CG method converges for strictly convex functions, i.e., the Hessian matrix is positive definite at every point. However, for non-convex functions, convergence is not guaranteed.
In the next chapter, we will see how to construct a method if A is not symmetric and positive definite.

Chapter 4

Iterative Methods for Large Linear Systems

Motivation
We consider the linear system $Ax = b$ with $A \in \mathbb{R}^{n\times n}$ invertible. Typically, n is very large, and A is sparse. We aim to construct a method that requires $O(n)$ operations. In Numerical Analysis I, we know of direct methods for this, such as the LU decomposition. However, this approach has the problem of "fill-ins," i.e., zero elements become filled, causing the cost to increase to $O(n^3)$.

An alternative method with better computational complexity is an iterative indirect method. From the previous section, we know that for a symmetric and positive definite A, such a method exists in the Krylov space. Even though our A is no longer symmetric and positive definite, we still seek an iterative method in a Krylov space with the approximation:

$$\mathcal{K}_k(A, r_0) := span\{r_0, Ar_0, \dots, A^{k-1}r_0\}, \quad r_0 = b - Ax_0$$

since constructing this space requires only multiplications with A, which is computationally inexpensive when A is sparse. (In the following, without loss of generality, we assume $x_0 = 0$ and thus $r_0 = b$.)

Remark 4.1 – Problem with the Basis Selection

If $V_k = (v_1, \dots, v_k) \in \mathbb{R}^{n\times k}$ is the basis of $\mathcal{K}_k$, an element $x_k \in \mathcal{K}_k$ can be written as:

$$x_k = \sum_{i=1}^{k} y_{k,i} v_i = V_k y_k.$$

However, there are issues:

1. How do we choose the basis? It is not a good idea to simply take $A^k r$, because in the power method, large eigenvalues dominate the matrix, leading to nearly linearly dependent vectors and an unmanageable basis.
2. We do not know how to select the coefficients y_k.

We will now discuss two methods based on the Gram-Schmidt procedure: the Arnoldi method and the Lanczos method, which do not calculate the Krylov iterations directly. Additionally, we will see how to cleverly choose the coefficient vector y_k.

4.1 Arnoldi Method

Idea
We construct an orthonormal basis (ONB) for $\mathcal{K}_k(A, b)$ using a modification of the Gram-Schmidt

procedure. We use a modification because the original Gram-Schmidt process is numerically unstable, and rounding errors cause vectors that are not orthogonal. We compute recursively: Let v_1 be an ONB of $\mathcal{K}_1$. Then we set:

$$v_1 := \frac{b}{\|b\|_2}.$$

For an ONB $v_1, \dots, v_k$ of $\mathcal{K}_k$ (with respect to the 2-norm), we choose for $\mathcal{K}_{k+1}$:

$$\tilde{v}_{k+1} := Av_k - \sum_{j=1}^{k} h_{jk} v_j \in \mathcal{K}_{k+1},$$

with coefficients h_{jk} chosen so that $\tilde{v}_{k+1}$ is orthogonal to the others. We then set:

$$v_{k+1} := \frac{\tilde{v}_{k+1}}{\|\tilde{v}_{k+1}\|_2}.$$

We now want to determine the coefficients h_{jk}. For $i = 1, \dots, k$, we must have:

$$0 = \langle v_i, \tilde{v}_{k+1} \rangle = \langle v_i, Av_k \rangle - \sum_{j=1}^{k} h_{jk} \underbrace{\langle v_i, v_j \rangle}_{\delta_{ij}} = \langle v_i, Av_k \rangle - h_{ik}.$$

Thus, $h_{ik} = \langle v_i, Av_k \rangle$.

Algorithm 4.1 – Arnoldi, 1951

We begin by setting $\beta := \|b\|$ and $v_1 := \frac{b}{\beta}$. For $k \geq 1$, we compute:

$$\tilde{v}_{k+1}^{(1)} = Av_k,$$

and then, for $j = 1, \dots, k$, we compute:

$$h_{jk} := \langle v_j, \tilde{v}_{k+1}^{(j)} \rangle, \quad \tilde{v}_{k+1}^{(j+1)} := \tilde{v}_{k+1}^{(j)} - h_{jk} v_j,$$

and finally:

$$h_{k+1,k} := \|\tilde{v}_{k+1}^{(k+1)}\|, \quad v_{k+1} := \frac{\tilde{v}_{k+1}^{(k+1)}}{h_{k+1,k}}.$$

Remark 4.2 – Termination Behavior

The Arnoldi method terminates when $h_{k+1,k} = 0$. In this case:

$$AV_k = V_k H_k \quad \Longrightarrow \quad \mathcal{K}_{k+1} = \mathcal{K}_k.$$

Remark 4.3 – Simplifications in Matrix Notation

We set $H_k = (h_{ij})_{i,j=1}^{k}$ and note that $h_{ij} = 0$ for $i > j + 1$, so this is a Hessenberg matrix. Then, $\tilde{H}_k = (h_{ij})_{i,j=1}^{k+1}$ is a Hessenberg matrix with an additional row. Using this notation and the algorithm, we can write, for each k (with $V_k := (v_1, \dots, v_k)$):

$$Av_k = h_{k+1,k} v_{k+1} + \sum_{j=1}^{k} h_{jk} v_j, \quad \text{so}: \; AV_k = V_{k+1} \tilde{H}_k = V_k H_k + h_{k+1,k} v_{k+1} e_k^T.$$

The orthogonality $\langle v_i, v_j \rangle = \delta_{ij}$ then gives: $V_k^T V_k = I$, and we obtain:

$$V_k^T A V_k = V_k^T V_{k+1} \tilde{H}_k = \begin{pmatrix} I_k & 0 \end{pmatrix} \tilde{H}_k = H_k, \quad \text{so} \quad H_k = V_k^T A V_k.$$

Remark 4.4 – Special Case of a Symmetric Matrix

If A is symmetric, some of the recursions simplify significantly. In this case, H_k is symmetric, hence tridiagonal. We then obtain the following three-term recursion:

$$h_{k+1,k}v_{k+1} = \tilde{v}_{k+1} = Av_k - h_{kk}v_k - h_{k-1,k}v_{k-1}.$$

Thus, we have $O(k)$ operations to compute $v_1, \dots, v_k$. In the general (non-symmetric) case, this requires $O(k^2)$ operations.

Theorem 4.1

If $h_{k+1,k} = 0$ and $h_{j+1,j} \neq 0$ for $j \leq k-1$, then:

1. Every eigenvalue of H_k is an eigenvalue of A. This holds even if A is not invertible.
2. There exists a vector $y_k \in \mathbb{R}^k$ such that $Ax = b$ holds with $x = V_k y_k$.

Proof.

1. Let λ be an eigenvalue of H_k with eigenvector $y \neq 0$. Then: $H_k y = \lambda y \iff V_k H_k y = \lambda V_k y$, but $V_k H_k = AV_k$ due to $h_{k+1,k} = 0$. Hence, $A(V_k y) = \lambda(V_k y)$. Therefore, λ is an eigenvalue of A with eigenvector $V_k y$.
2. Since A is invertible, 0 is not an eigenvalue of A. By item 1, 0 is not an eigenvalue of H_k. Hence, H_k is invertible. We want to find $x = V_k y_k$ such that $Ax = b$. Using the Arnoldi relation $AV_k = V_k H_k$ (which holds since $h_{k+1,k} = 0$), this is equivalent to $AV_k y_k = V_k H_k y_k = b$. Recall that $b = \beta v_1$, where $\beta = \|b\|$ and v_1 is the first column of V_k. Thus, $b = \beta v_1 = V_k \beta e_1$. Since H_k is invertible, the linear system $H_k y_k = \beta e_1$ has a unique solution y_k. Then, $x = V_k y_k$ satisfies $Ax = b$.

□

4.2 FOM and GMRES: Galerkin and Minimization of the Residuum

We now want to determine the coefficient vector y_k. There are two approaches:

Idea (Galerkin Approach)
We require that the residual $Ax_k - b$ is orthogonal to the Krylov subspace, i.e.,

$$\langle Ax_k - b, v\rangle = 0 \quad \forall v \in \mathcal{K}_k.$$

If A is symmetric and positive definite, we would get the Conjugate Gradient (CG) method. We ignore the definiteness of A and continue analogously.

It suffices that this is orthogonal to all basis vectors, i.e. $V_k^T(Ax_k - b) = 0$. Then, from $x_k = V_k y_k$, we have $Ax_k = AV_k y_k$ and $b = \beta v_1 = \beta V_k e_1$. Using $V_k^T V_k = I_k$ and $V_k^T v_{k+1} = 0$, we get:

$$V_k^T(Ax_k - b) = V_k^T Ax_k - V_k^T b = V_k^T V_{k+1}\tilde{H}_k y_k - \beta V_k^T V_k e_1 = H_k y_k - \beta e_1.$$

We require this to be zero, i.e., we obtain the equation $H_k y_k = \beta e_1$, which is uniquely solvable if H_k is invertible.

Remark 4.5

In the CG method, A is symmetric and positive definite, so is H_k.

Algorithm 4.2 – FOM

We iteratively solve $x_k = V_k y_k$ with $H_k y_k = \beta e_1$.

Historical Note. FOM stands for "Full Orthogonalisation Method" and was introduced by Saad in 1981.

Idea (Minimization of the Residuum)
We now minimize the residual, i.e., we seek $x_k \in \mathcal{K}_k(A,b)$ *such that* $\|Ax_k - b\|_2 = \min$. *Since* V_{k+1} *is orthogonal and norms are invariant under orthogonal matrices, we have:*

$$\|Ax_k - b\|_2 = \|AV_k y_k - \beta v_1\|_2 = \|V_{k+1}(\tilde{H}_k y_k - \beta e_1)\|_2 = \|\tilde{H}_k y_k - \beta e_1\|_2.$$

This gives us a linear least squares problem: Find $y_k \in \mathbb{R}^k$ *such that*

$$\|\tilde{H}_k y_k - \beta e_1\|_2 = \min$$

If $h_{i+1,i} \neq 0$ *for* $i = 1, \ldots, k$, *then the least squares problem is uniquely solvable.*

Algorithm 4.3 – GMRES

We compute $x_k := V_k y_k$ with $\|\tilde{H}_k y_k - \beta e_1\|_2 = \min$.

Historical Note. GMRES stands for "Generalized Minimum Residual Method" and was developed by Saad and Schultz in 1986.

Proposition 4.1 – Properties of FOM and GMRES

1. If k is the first index with $h_{k+1,k} = 0$ (then $AV_k = V_k H_k$), both FOM and GMRES in the k-th step provide the exact solution to $Ax = b$.
2. For the GMRES residual, we have:
$$\|Ax_k - b\|_2 = \min_{\substack{\deg p_k = k \\ p_k(0)=1}} \|p_k(A)b\|_2.$$
3. Let $A = T\Lambda T^{-1}$ with $\Lambda = \operatorname{diag}(\lambda_j)$. Then,
$$\|Ax_k - b\| \leq \operatorname{cond}(T) \min_{\substack{\deg p_k = k \\ p_k(0)=1}} \max_{j=1,\ldots,n} |p_k(\lambda_j)| \|b\|.$$
4. If H_k is not invertible (i.e., x_k^{FOM} not exists), then $x_{k+1}^{\mathrm{GMRES}} = x_k^{\mathrm{GMRES}}$ stagnates.

Proof. 1. For FOM, as H_k is nonsingular (since its eigenvalues are inherited from A, and A is invertible), the equation $H_k y_k = \beta e_1$ has a unique solution y_k, and hence $x_k = V_k y_k$ satisfies:

$$Ax_k = AV_k y_k = V_k H_k y_k = V_k \beta e_1 = \beta v_1 = b.$$

For GMRES, we compute:

$$\|Ax_k^{\mathrm{GMRES}} - b\|_2 \leq \|Ax_k^{\mathrm{FOM}} - b\|_2 = 0.$$

2. We have $x_k = q_{k-1}(A)b$ with $\deg q_{k-1} = k-1$, and we compute:

$$b - Ax_k = (I - Aq_{k-1}(A))b = p_k(A)b \quad \text{with} \quad p_k(\lambda) = 1 - \lambda q_{k-1}(\lambda).$$

From the minimization condition for the residual, the statement follows.

3. From the second part, we have:

$$\|Ax_k - b\| = \min_{\substack{\deg p_k = k \\ p_k(0)=1}} \|p_k(A)b\| = \min_{\substack{\deg p_k = k \\ p_k(0)=1}} \|Tp_k(\Lambda)T^{-1}b\| \leq \operatorname{cond}(T) \min_{\substack{\deg p_k = k \\ p_k(0)=1}} \max_{j=1,\ldots,n} |p_k(\lambda_j)| \|b\|.$$

4. Omitted proof.

□

Remark 4.6 – Implementation of GMRES

We perform a QR decomposition of $\tilde{H}_k$, i.e., there exists a Q_k such that:

$$\tilde{H}_k = Q_k \begin{pmatrix} R_k \\ 0 \end{pmatrix},$$

and thus we have:

$$\|\tilde{H}_k y_k - \beta e_1\|_2 = \left\| \begin{pmatrix} R_k y_k \\ 0 \end{pmatrix} - \beta Q_k^\top e_1 \right\|_2.$$

Remark 4.7 – Computational Cost of GMRES

We now want to examine the computational cost for the non-symmetric case $A \neq A^T$:

1. We must compute v_{k+1} using the Arnoldi method, which involves k multiplications of A with a vector. This requires $O(k^2 n)$ operations.
2. Computing y_k requires $O(k^2)$ operations.
3. Computing $x_k = V_k y_k$ requires $O(kn)$ operations.

A major issue is storage: we need $(k+1)n$ memory slots, meaning that the GMRES method can only be performed if k is not too large, provided n is large. This can be addressed by performing only a fixed number of iterations and restarting the method. The downside is worse convergence.

Remark 4.8 – Stopping Criterion

A typical stopping condition is that we require $\|Ax_k - b\| \leq \text{tol}$. This is computationally expensive. However, we know that $\|Ax_k - b\| = \|\tilde{H}_k y_k - \beta e_1\|$, which is easier to compute. Therefore, we stop the method when:

$$\|\tilde{H}_k y_k - \beta e_1\| \leq \text{tol}.$$

4.3 Lanczos Algorithm

Motivation
The difficulty with the Arnoldi method is that we need the vectors $v_1, \ldots, v_k$ to compute v_{k+1}, which leads to a high computational and especially memory cost. Our goal is to obtain a basis of $\mathcal{K}_k(A, b)$ with a short recursion (not only for symmetric A).

Idea (Lanczos) *1. We abandon the orthonormal basis (ONB).*

2. *We compute two bases $v_1, \ldots, v_k$ for $\mathcal{K}_k(A, b)$ and $w_1, \ldots, w_k$ for $\mathcal{K}_k(A^T, c)$.*
3. *We require bi-orthogonality, i.e.,*

$$\langle w_i, v_j \rangle = 0 \quad \text{for} \quad i \neq j.$$

Construction (Derivation of the Algorithm)
We make the following ansatz:

$$v_{k+1} = Av_k - \sum_{j=1}^{k} h_{jk} v_j, \quad w_{k+1} = A^T w_k - \sum_{j=1}^{k} \ell_{jk} w_j.$$

Due to bi-orthogonality, we now require for $i = 1, \ldots, k$:

$$0 = \langle w_i, v_{k+1} \rangle = \langle w_i, Av_k \rangle - \langle w_i, v_i \rangle h_{ik}, \quad 0 = \langle w_{k+1}, v_i \rangle = \langle A^T w_k, v_i \rangle - \langle w_i, v_i \rangle \ell_{ik}.$$

We now compute using the basis representation:

$$\langle w_i, Av_k \rangle = \langle A^T w_i, v_k \rangle = \left\langle w_{i+1} + \sum_{j=1}^{i} \ell_{ji} w_j, v_k \right\rangle = 0 \quad \text{for} \quad i + 1 < k.$$

If additionally $\langle w_i, v_i\rangle \neq 0$, it follows that $h_{ik} = 0$ for $i+1 < k$, and similarly $\ell_{ik} = 0$ for $i+1 < k$.

We continue the computation:

$$h_{kk} = \frac{\langle w_k, Av_k\rangle}{\langle w_k, v_k\rangle} = \frac{\langle A^T w_k, v_k\rangle}{\langle w_k, v_k\rangle} = \ell_{kk}.$$

Next, we compute using the recurrence relation:

$$h_{k-1,k} = \frac{\langle w_{k-1}, Av_k\rangle}{\langle w_{k-1}, v_{k-1}\rangle} = \frac{\langle A^T w_{k-1}, v_k\rangle}{\langle w_{k-1}, v_{k-1}\rangle} = \frac{\langle w_k + \sum \ldots, v_k\rangle}{\langle w_{k-1}, v_{k-1}\rangle} = \frac{\langle w_k, v_k\rangle}{\langle w_{k-1}, v_{k-1}\rangle} = \ell_{k-1,k}.$$

Algorithm 4.4 – Lanczos, 1952

1. We choose $b, c \neq 0$ and set $v_1 := b, w_1 := c$, and $v_0 := w_0 := 0$.
2. For $k = 1, 2, 3, \ldots$ (as long as $\langle w_k, v_k\rangle \neq 0$), we set:

$$\alpha_k := \frac{\langle w_k, Av_k\rangle}{\langle w_k, v_k\rangle}, \quad \beta_k := \frac{\langle w_k, v_k\rangle}{\langle w_{k-1}, v_{k-1}\rangle},$$

where β_k exists only for $k \geq 2$. Finally, we set:

$$v_{k+1} := Av_k - \alpha_k v_k - \beta_k v_{k-1}, \quad w_{k+1} := A^T w_k - \alpha_k w_k - \beta_k w_{k-1}.$$

Remark 4.9

For $A = A^T$, the Arnoldi and Lanczos methods coincide.

Notation – Simplification in Matrix Notation

We denote the basis matrices as:

$$V_k := (v_1, \ldots, v_k), \quad W_k := (w_1, \ldots, w_k),$$

and the matrix T_k as:

$$T_k := \begin{pmatrix} \alpha_1 & \beta_2 & 0 & \cdots & 0 \\ 1 & \alpha_2 & \beta_3 & \cdots & \vdots \\ 0 & \ddots & \ddots & \ddots & 0 \\ \vdots & \ddots & \ddots & \ddots & \beta_k \\ 0 & \cdots & 0 & 1 & \alpha_k \end{pmatrix}.$$

We also define:

$$\tilde{T}_k := \begin{pmatrix} & T_k & & \\ 0 & \cdots & 0 & 1 \end{pmatrix} \in \mathbb{R}^{(k+1)\times k}.$$

Remark 4.10 – Lanczos Method in Matrix Notation

In the above notation, we obtain:

$$AV_k = V_{k+1}\tilde{T}_k, \quad A^T W_k = W_{k+1}\tilde{T}_k.$$

In particular, we have:

$$\mathcal{K}_k(A, b) = \mathrm{Span}\{v_1, \ldots, v_k\}, \quad \mathcal{K}_k(A^T, c) = \mathrm{Span}\{w_1, \ldots, w_k\},$$

and with bi-orthogonality:

$$W_k^T V_k = \mathrm{diag}(w_1^T v_1, \ldots, w_k^T v_k) =: D_k.$$

It follows that: $W_k^T A V_k = D_k T_k$, $T_k = D_k^{-1} W_k^T A V_k$, since we have:

$$W_k^T A V_k = W_k^T V_{k+1} \tilde{T}_k = \begin{pmatrix} D_k & 0 \end{pmatrix} \tilde{T}_k = \begin{pmatrix} D_k & 0 \end{pmatrix} \begin{pmatrix} T_k \\ * \end{pmatrix} = D_k T_k.$$

Remark 4.11

1. In practice, the vectors v_i and w_i are often normalized.
2. We choose the stopping criterion $\langle w_{k+1}, v_{k+1} \rangle = 0$. This can be divided into two different types of stopping criteria:
   - A regular stop occurs if $v_{k+1} = 0$, in which case $AV_k = V_k T_k$ and $\mathcal{K}_{k+1}(A, b) = \mathcal{K}_k(A, b)$. Due to the symmetry of v and w, we analogously obtain for $w_{k+1} = 0$ that $A^T W_k = W_k T_k$ and $\mathcal{K}_{k+1}(A^T, c) = \mathcal{K}_{\mathcal{K}}(A^T, c)$.
   - A serious stop occurs if $v_{k+1} \neq 0$ and $w_{k+1} \neq 0$, but it could happen that $\langle v_{k+1}, w_{k+1} \rangle \approx 0$, due to rounding errors or numerical instability during inversion due to poor conditioning.

Remark 4.12 – Look-ahead Lanczos

We want to avoid a serious stop as much as possible. To achieve this, we modify the Lanczos method by relaxing the bi-orthogonality to block orthogonality, i.e., we have:

$$W_k^T V_k = \begin{pmatrix} B_1 & 0 & \cdots & 0 \\ 0 & B_2 & \cdots & 0 \\ \vdots & \vdots & \vdots & \vdots \\ 0 & 0 & \cdots & B_\ell \end{pmatrix},$$

where B_i is a square matrix block, and $\|B_i^{-1}\|$ is not too large. This results in a stable algorithm with a short recursion.

4.4 BiCG and QMR

Motivation
We are still working with the linear system $Ax = b$ and seek an approximation $x_k = V_k y_k$, where $V_k = \begin{pmatrix} v_1 & \dots & v_k \end{pmatrix}$ is a Lanczos basis. We consider various approaches for this.

Remark 4.13 – Galerkin Approach

We require that $\langle w, Ax_k - b \rangle = 0$ for all $w \in \mathcal{K}_k(A^T, c)$. We test this for $w = w_i$, resulting in $W_k^T(Ax_k - b) = 0$. Similar to the Arnoldi method, we obtain for $b = \beta v_1 = V_k \beta e_1$:

$$W_k^T A x_k - W_k^T V_k e_1 = 0.$$

Since $x_k = V_k y_k$, this becomes: $W_k^T A V_k y_k - W_k^T V_k \beta e_1 = 0$. But since $W_k^T A V_k = D_k T_k$ and $W_k^T V_k = D_k$, we get:

$$D_k T_k y_k = D_k \beta e_1.$$

If D_k is invertible, we obtain the following linear system with the tridiagonal matrix T_k:

$$T_k y_k = \beta e_1.$$

Algorithm – Bi-Conjugate Gradient Method (BiCG)

We solve $x_k = V_k y_k$ and $T_k y_k = \beta e_1$.

Remark 4.14 – Computational Effort

We solve the linear system via an LU decomposition $T_k = L_k R_k$, i.e.

$$x_k = V_k y_k = \underbrace{V_k R_k^{-1}}_{=:P_k} \underbrace{L_k^{-1} \beta e_1}_{=:q_k} \quad \text{with} \quad q_k = \begin{pmatrix} \rho_1 \\ \vdots \\ \rho_k \end{pmatrix} = \begin{pmatrix} q_{k-1} \\ \rho_k \end{pmatrix}.$$

Thus, $L_k q_k = \beta_k e_1$, implying $\rho_k = -\ell_{k,k-1}\rho_{k-1}$, and we can compute this in a short recursion. With the definition of P_k, we have $P_k R_k = V_k = (v_1 \cdots v_k)$, so $P_k = \begin{pmatrix} p_1 & \dots & p_k \end{pmatrix}$, and from $V_k = P_k R_k$, we obtain:

$$v_1 = p_1 r_{11}, \quad v_2 = p_1 r_{12} + p_2 r_{22}, \quad \dots, \quad v_k = p_{k-1} r_{k-1,k} + p_k r_{kk}.$$

These are short recursions for p_k, and thus for x_k:

$$x_k = P_k q_k = P_{k-1} q_{k-1} + p_k \rho_k = x_{k-1} + p_k \rho_k.$$

Thus, we have short recursions for x_k, and we can iteratively compute (x_k, p_k, ρ_k) to obtain $(x_{k+1}, p_{k+1}, \rho_{k+1})$.

Historical Note. The BiCG method was proposed by Lanczos in 1952 and by Fletcher in 1976.

Construction (Minimization Approach)

We now consider another approach to solving the problem and require:

$$\beta := \|b\|, \quad \|v_j\| = \|w_j\| = 1 \quad \forall j$$

which yields:

$$\|Ax_k - b\| = \|AV_k y_k - \beta v_1\| = \|V_{k+1}\tilde{T}_k y_k - V_{k+1}\beta e_1\| \leq \|V_{k+1}\| \cdot \|\tilde{T}_k y_k - \beta e_1\|$$

If $\|v_j\| = 1$ for all j, then $\|V_{k+1}\| \leq \sqrt{k+1}$. The idea is now to minimize only the norm of the coefficient vector $\|\tilde{T}_k y_k - \beta e_1\|$.

Algorithm 4.5 – Quasi-Minimization of Residual (QMR)

We compute $x_k = V_k y_k$ with $\|\tilde{T}_k y_k - \beta e_1\| = \min$.

Remark 4.15 – Computational Effort

The implementation of the QMR method is similar to the GMRES method and utilizes ideas from the BiCG method. We have $\tilde{T}_k = Q_k \begin{pmatrix} R_k \\ 0 \end{pmatrix}$ and

$$Q_k^T(\beta e_1) = \begin{pmatrix} q_k \\ \rho_{k+1} \end{pmatrix}, \quad \text{with} \quad q_k = \begin{pmatrix} \rho_1 \\ \vdots \\ \rho_k \end{pmatrix} = \begin{pmatrix} q_{k-1} \\ \rho_k \end{pmatrix}.$$

Since the solution to the least squares problem is equivalent to $R_k y_k = q_k$, we have $x_k = V_k y_k = V_k R_k^{-1} q_k$, with $V_k R_k^{-1} = P_k = (p_1 \cdots p_k)$, so:

$$x_k = \underbrace{P_{k-1} q_{k-1}}_{=x_{k-1}} + p_k \rho_k.$$

Thus, we obtain the same short recursions for BiCG, allowing us to compute (x_k, P_k, ρ_k) and then $(x_{k+1}, P_{k+1}, \rho_{k+1})$.

Historical Note. The QMR method was proposed by Freund and Nachtigal in 1991.

Remark 4.16

The QMR method converges much more smoothly than the BiCG method, but it is not necessarily monotonically decreasing (as in the GMRES method).

Summary We considered several methods:

	Galerkin	Residual minimization
Arnoldi	FOM	GMRES
Lanczos	BiCG	QMR

We compare the Arnoldi and Lanczos methods:

Arnoldi	Lanczos
Orthonormal basis (ONB)	Biorthogonal bases
Stable (in general)	Stabilization may require look-ahead
Long recurrence	Short recurrence

We can also compare Galerkin and residual minimization:

Galerkin	Residual Minimization
Linear system (LGS)	Linear least squares problem
	Residual norms vary more smoothly
	(in GMRES: monotonically decreasing)

In practice, preconditioning is very important. However, this often falls under the motto: *"more art than science."* Instead of solving the system $Ax = b$, we solve the preconditioned system

$$M_1 A M_2 u = M_1 b, \quad \text{with} \quad x = M_2 u,$$

for example, using an incomplete LU decomposition.

Chapter 5

Linear Optimization

5.1 Examples (from Economics)

5.1.1 Introductory examples

A factory produces two products p_1 and p_2 from three raw materials q_1, q_2, q_3. Producing one unit of p_1 requires 1 unit of q_1, 2 units of q_2, and 4 units of q_3. For the production of p_2, 6 units of q_1, 2 units of q_2, and 1 unit of q_3 are needed.

We have 30 units of q_1, 15 units of q_2, and 24 units of q_3 available. The sales profit for p_1 is two euros per unit and for p_2 one euro per unit.

We want to find the optimal production quantities for p_1 and p_2, i.e. we seek $x = (x_1, x_2)$ such that $2x_1 + 1x_2 = \max!$. The constraints are $x_1 \geq 0$ and $x_2 \geq 0$, because we can't sell a negative number of products. Due to the limited raw materials, we get additional constraints:

$$\begin{aligned} 1x_1 + 6x_2 &\leq 30 \quad \text{(available } q_1 \text{ material)} \\ 2x_1 + 2x_2 &\leq 15 \quad \text{(available } q_2 \text{ material)} \\ 4x_1 + 1x_2 &\leq 24 \quad \text{(available } q_3 \text{ material).} \end{aligned}$$

We can solve this graphically by plotting the constraints and identifying the feasible region as a polygon. We then look for equipotential line, i.e. lines defined by $2x_1 + x_2 = c$ for any c. The reason is that we want to find the largest value of c such that (x_1, x_2) is in the feasible region. See Figure 5.1 for the graphical solution.

The solution occurs at a corner, i.e. we can try all corners and obtain the solution by comparing a finite number of values. Here, the solution is unique, namely $(x_1, x_2) = \left(\frac{11}{2}, 2\right)$. The profit is: $2 \cdot \frac{11}{2} + 1 \cdot 2 = 13$.

Remark 5.1

1. For $2x_1 + 2x_2 = \max$, this would not be unique but would yield the same maximum profit along an edge (the red edge on Figure 5.1).

2. The solution $x_1 = \frac{11}{2}$ assumes the divisibility of product p_1; otherwise, we could require $(x_1, x_2) \in \mathbb{Z} \times \mathbb{Z}$, leading to integer optimization, which is much more complex and not covered here (but still very interesting and very much used in practice!).

5.1.2 Consumer Problem

We have four food items characterized by the following table:

We want to buy 12 nutritional units and 7 vitamin units at the minimum price. We seek $x = (x_1, x_2, x_3, x_4)$ with $x_i \geq 0$ and the constraints:

$$\begin{aligned} 2x_1 + 1x_2 + 0x_3 + 1x_4 &\geq 12 \quad \text{(Nutritional units)} \\ 3x_1 + 4x_2 + 3x_3 + 5x_4 &\geq 7 \quad \text{(Vitamin units),} \end{aligned}$$

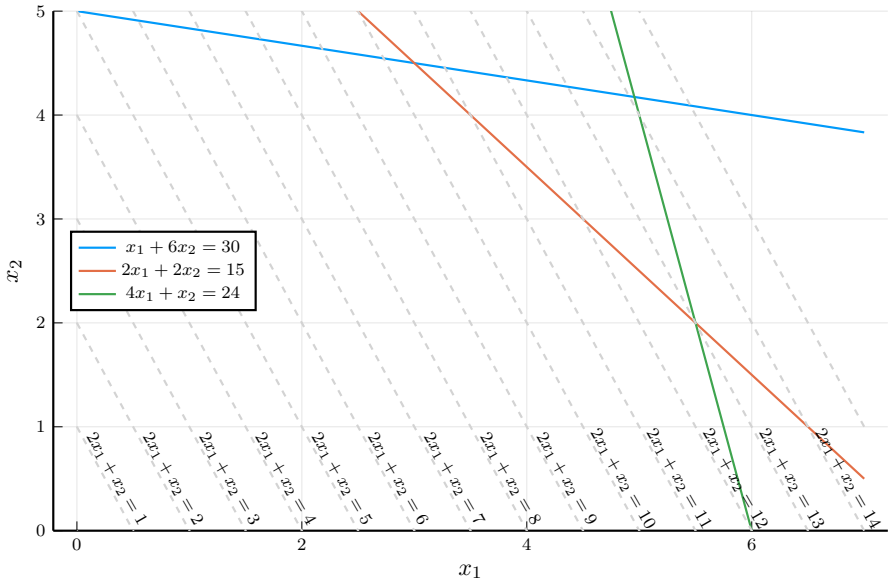

Figure 5.1: Graphical solution for the problem of Section 5.1.1

Food	A	B	C	D
Nutritional Value	2	1	0	1
Vitamins	3	4	3	5
Price	2	2	1	8

Table 5.1: Data for the Consumer Problem

and the minimization problem:

$$2x_1 + 2x_2 + 1x_3 + 8x_4 = \min!$$

The variables x_j here count the number of food types we pay for (x_1 for food type A, x_2 for food type B, ...).

Notation 5.1

The notation $f(x) = \min!$ means that we are looking for x such that $f(x) = \min_y f(y)$.

5.1.3 Extension to the Competitor Problem

A competitor introduces two new food items with the following characteristics:

	I	II
Nutritional Value	1	0
Vitamins	0	1
Price	y_1	y_2

We want to determine the prices y_1 and y_2 such that the total sales price is maximized, i.e. we have the problem $12y_1 + 7y_2 = \max!$, as consumers will buy only what is necessary. We must find the same amount of nutritional value and vitamins but at a lower cost than with products A, B, C, D. Under $y_1, y_2 \geq 0$, the constraints are:

$$\begin{aligned} 2y_1 + 3y_2 &\leq 2 \quad \text{(to be better than food A)} \\ 1y_1 + 4y_2 &\leq 2 \quad \text{(to be better than food B)} \\ 0y_1 + 3y_2 &\leq 1 \quad \text{(to be better than food C)} \\ 1y_1 + 5y_2 &\leq 8 \quad \text{(to be better than food D).} \end{aligned}$$

Remark 5.2

The two problems are *dual* to each other in the following sense:

- The consumer problem can be written as: $c^T x = \min!$ with $x \in \mathbb{R}^4_+$ and a matrix A such that $Ax \geq b$.
- In the competitor problem, we have $b^T y = \max!$ with $y \in \mathbb{R}^2_+$ and $A^T y \leq c$.

Just to be clear, we have

$$A = \begin{pmatrix} 2 & 1 & 0 & 1 \\ 3 & 4 & 3 & 5 \end{pmatrix}, b = \begin{pmatrix} 12 \\ 7 \end{pmatrix}, c = \begin{pmatrix} 2 \\ 2 \\ 1 \\ 8 \end{pmatrix}.$$

In both cases, we consider the inequalities component-wise.

5.1.4 Transportation Problem

We have m producers and n markets. The transportation cost for one unit from i to j is given by C_{ij}. The supply is $a_1, \ldots, a_m$ and the demand is $b_1, \ldots, b_n$.

The problem is to find the most cost-effective transportation plan to send the products to the markets. Our unknowns are the delivery quantities from i to j, called $x_{ij} \geq 0$, with $i = 1, \ldots, m$ and $j = 1, \ldots, n$. The total cost is calculated as:

$$z = \sum_{i=1}^{m} \sum_{j=1}^{n} C_{ij} x_{ij}$$

We assume that the total supply equals the total demand, i.e. $\sum a_i = \sum b_j$. We have:

- the total shipment from producer i is $\sum_{j=1}^{n} x_{ij} = a_i$ (a producer sends everything it has),
- the total intake of market j is $\sum_{i=1}^{m} x_{ij} = b_j$ (the demand of a market is fully supplied).

We have mn unknowns $x_{ij} \geq 0$, under $m+n$ constraints of which only $m+n-1$ are independent.

5.1.5 Dual Problem to the Transportation Problem

A package service offers free transportation on all routes but charges a fee u_i for sending a unit from i and v_j for delivering a unit to j.

The earnings of this service can be computed as follows:

- a producer ships everything it has, i.e. its a_i units. They have to be picked by the transportation company, hence they earn $u_i a_i$ for producer i
- Each market is fully supplied, hence its demand of b_j units is fulfilled. The transportation company needs to deliver b_j units to market j, hence they earn $v_j b_j$ for market j.

By summing over all producers and all markets, the total earnings of the transportation company are

$$z' = \sum_{i=1}^{m} u_i a_i + \sum_{j=1}^{n} v_j b_j.$$

This cost[1] z' should be optimized but as a constraint, it must still be cheaper than traditional post, i.e.

$$u_i + v_j \leq C_{ij} \quad \forall i, j.$$

For the optima, it always holds that $z' \leq z$, because:

$$z' = \sum_{i=1}^{m} u_i a_i + \sum_{j=1}^{n} v_j b_j = \sum_i u_i \sum_j x_{ij} + \sum_j v_j \sum_i x_{ij} = \sum_{i,j} (u_i + v_j) x_{ij} \leq \sum_{i,j} C_{ij} x_{ij} = z.$$

We now calculate when the equality holds: since all the summands are nonnegative,

$$z' = z \iff (u_i + v_j - C_{ij}) x_{ij} = 0 \quad \forall i, j \iff u_i + v_j = C_{ij} \text{ or } x_{ij} = 0 \quad \forall i, j.$$

If this holds, z' is already maximal and z is minimal.

Remark 5.3

The converse of the last statement also holds, i.e. if z' is maximal and z is minimal, then $z = z'$, which we will see in the section on duality.

5.2 Linear Programs (Optimization problems)

Definition 5.1 – Mathematical problem formulation

We are given $A \in \mathbb{R}^{m \times n}$, $b \in \mathbb{R}^m$, and $c \in \mathbb{R}^n$. We write this in a tableau:

A	b
c^T	

This leads to various problem formulations:

(a) We seek $x \in \mathbb{R}^n$ with:

$$Ax \leq b, \quad c^T x = \min!$$

(b) We seek $x \in \mathbb{R}^n$ with:

$$Ax \leq b, \quad x \geq 0, \quad c^T x = \min!$$

(c) We seek $x \in \mathbb{R}^n$ with:

$$Ax = b, \quad x \geq 0, \quad c^T x = \min!$$

In each case, we consider the inequalities component-wise. The function $c^T x$ is also called the *objective* or *cost* function.

Remark 5.4

We have written various problem formulations as minimizations as it is commonly done. This is done without loss of generality, since a maximization problem can be reformulated as a minimization problem:

$$\max(f(x)) = -\min(-f(x)).$$

Proposition 5.1 – Reformulation of standard forms

All three problems from Definition 5.1 can be equivalently transformed into each other.

[1] The term "cost" commonly refers to something that is paid/lost, but in optimization is it usual to call "cost" the function to optimize (i.e. maximize or minimize). It can also be called "objective function".

Proof. (a) ⇒ (b): We set $x := x_+ - x_-$ with $x_+, x_- \geq 0$. Then:

$$\begin{aligned}(a) &\implies A(x_+ - x_-) \leq b, \quad x_+, x_- \geq 0, \quad c^T(x_+ - x_-) = \min! \\ &\implies (A, -A)\begin{pmatrix} x_+ \\ x_- \end{pmatrix} \leq b, \quad \begin{pmatrix} x_+ \\ x_- \end{pmatrix} \geq 0, \quad (c^T, -c^T)\begin{pmatrix} x_+ \\ x_- \end{pmatrix} = \min!\end{aligned}$$

(b) ⇒ (c): We introduce so-called *slack variables*: let $z \in \mathbb{R}^m$ with $z \geq 0$. Then we get:

$$\begin{aligned}(b) &\implies Ax + z = b, \quad x \geq 0, \quad z \geq 0, \quad c^T x = \min! \\ &\implies (A, I)\begin{pmatrix} x \\ z \end{pmatrix} = b, \quad \begin{pmatrix} x \\ z \end{pmatrix} \geq 0, \quad (c^T, 0)\begin{pmatrix} x \\ z \end{pmatrix} = \min!\end{aligned}$$

(c) ⇒ (a):

$$\begin{aligned}(c) &\implies Ax \leq b, \quad Ax \geq b, \quad x \geq 0, \quad c^T x = \min! \\ &\implies Ax \leq b, \quad -Ax \leq -b, \quad -x \leq 0, \quad c^T x = \min! \\ &\implies \begin{pmatrix} A \\ -A \\ -I \end{pmatrix} x \leq \begin{pmatrix} b \\ -b \\ 0 \end{pmatrix}, \quad c^T x = \min!\end{aligned}$$

□

Remark 5.5

We note that the transitions each lead to increases in dimension.

Assumption 5.2.1
From now on, we will always consider formulations of the type (c), and we will assume $\operatorname{rank}(A) = m$.

The feasible region of the solution is $\{x \in \mathbb{R}^n, Ax = b, x \geq 0\}$. Here, n is the dimension of the solution and m is the number of equality constraints. It is also sensible to assume that $n \geq m$.

Example 5.1

1. $n = 2, m = 1$. Then the feasible region is a part of a line that runs in the first quadrant of a coordinate system. This can be bounded, unbounded, or even empty.
2. $n = 3, m = 1$. Then the feasible region is a plane that is a side face of a polyhedron (also called a simplex). The feasible region can, however, also be unbounded or empty.
3. $n = 3, m = 2$. This is the intersection of two planes, which is generally a line.

We note that the corners (or endpoints) in our examples have at most m non-zero components. This motivates the general definition of what we understand as a corner in higher dimensions.

Remark 5.6

Note that the assumption $\operatorname{rank}(A) = m$ is done without loss of generality: assume it is not, then at least one row (labelled r) of A is linearly dependent from the others, i.e. there are coefficients α_j such that $A_{r,\bullet} = \sum_{j=1, j\neq r}^{m} \alpha_j A_{j,\bullet}$. Two situations can occur:

- $b_r \neq \sum_{j=1, j\neq r}^{m} \alpha_j b_j$: then the linear system has no solution, and the feasible region is empty.
- $b_r = \sum_{j=1, j\neq r}^{m} \alpha_j b_j$, in which case the linear system $Ax = b$ is the same if the row r is removed.

We can then examine all redundant rows in A, and either find one that shows the feasible region is empty, or delete all of them until A has only linearly independent rows. We then let m the number of independent rows.

An interesting property when considering optimization problems is the convexity of the feasible set: if it is, some things are sometimes easier to show.

Definition 5.2 – Convex set

A set $\mathcal{C}$ is *convex* if, for any $x_0, x_1 \in \mathcal{C}$, $x_\lambda = (1-\lambda)x_0 + \lambda x_1 \in \mathcal{C}$.

Proposition 5.2 – Convexity of the solution set

The set of feasible solutions to a linear programming problem is convex.

Proof. Let x, y two feasible solutions for the linear program, i.e. they satisfy $Ax = b, x \geq 0$ and $Ay = b, y \geq 0$. Consider $z = (1-\lambda)x + \lambda y$ for $0 \leq \lambda \leq 1$, then $Az = (1-\lambda)Ax + \lambda Ay = b$, and $z \geq 0$. □

Generally, in optimization problems, one must usually be careful about the optimum: is it only a local or a global optimum? The linear property of the linear problems tells us that any local optimum is a global optimum.

Lemma 5.1 – Local minima are global minima ([1, Lemma 1.4])

Any solution to a linear programming problem that is a local minimum solution is also a global minimum solution.

Proof. Let $p = (p_1, \ldots, p_n, z_p)$ a local minimum solution, where $(p_1, \ldots, p_n)$ is the vector that attains the local minima z_p. Assume that it is not a global minimum solution, then there is another solution $q = (q_1, \ldots, q_n, z_q)$ with $z_q < z_p$. By convexity of the feasible set, any point $x = (x_1, \ldots, x_n, z_x) = (1-\lambda)p + \lambda q$, $0 \leq \lambda \leq 1$ is feasible. Moreover, $z_x < z_p$ for all $0 < \lambda < 1$, which contradicts the fact that p is a local minimum: if it was the case, all points in a neighborhood of p would have a larger objective cost. Hence, z_p can only be a global minimum. □

Definition 5.3 – Corner (Vertex)

The point $x \in \mathbb{R}^n$ is called a *corner* if the following properties are fulfilled:

- x is feasible, i.e. $Ax = b$ and $x \geq 0$.
- x has at most m non-zero components.
- The columns in A corresponding to non-zero components in x are linearly independent.

A corner is often called a "basic feasible solution" in the literature since the columns in A corresponding to the nonzero components form a basis.

Let $\mathcal{I}$ the set of indices such that $x_\mathcal{I} \neq 0$. By definition of a corner, $A_{\bullet,\mathcal{I}}$ has linearly independent columns but this matrix might not be invertible if $|\mathcal{I}| < m$. However, one can always add to $\mathcal{I}$ some indices for which $x_\mathcal{I} = 0$ and such that the new matrix $A_{\bullet,\mathcal{I}}$ is now invertible. It is possible because $\text{rank}(A) = m$, which means there are exactly m columns in A that are linearly independent.

Therefore, up to a permutation of the columns in A, we have for any corner x,

$$A = (B, N), \quad B \in \mathbb{R}^{m \times m} \text{ invertible}, \qquad x = \begin{pmatrix} x_B \\ 0 \end{pmatrix}, x_B \in \mathbb{R}^m.$$

5.3 Simplex Method

Idea

We start at a corner and look for a neighboring corner with a smaller objective function. Then we start over, continuing until we are stuck in a minimum.

Remark 5.7 – Historical note

The algorithm goes back to Dantzig who developed it in 1947, during his time working for the US Air Force in project SCOOP (Scientific Computation Of Optimum Programs).

5.3.1 Derivation of the procedure

Let x be a corner, i.e. $Ax = b$ and $x \geq 0$, with at least $n - m$ zero components. We have

$$b = Ax = \begin{pmatrix} B & N \end{pmatrix} \begin{pmatrix} x_B \\ 0 \end{pmatrix} = Bx_B \quad \text{with } B \in \mathbb{R}^{m \times m} \text{ invertible.}$$

The cost function at the corner is $c^T x = \begin{pmatrix} c_B^T & c_N^T \end{pmatrix} \begin{pmatrix} x_B \\ 0 \end{pmatrix} = c_B^T x_B$ with $c_B \in \mathbb{R}^m$. We now seek another corner with lower costs. Instead of the x defined above, we consider the costs at another feasible point $x' = (x'_B, x'_N)$. Because $Ax' = b \iff Bx'_B + Nx'_N = b \iff x'_B = B^{-1}b - B^{-1}Nx'_N$ and $B^{-1}b = x_B$, it follows that:

$$x' = \begin{pmatrix} x'_B \\ x'_N \end{pmatrix} = \begin{pmatrix} x_B - B^{-1}Nx'_N \\ x'_N \end{pmatrix}.$$

The difference in the cost function is then:

$$c^T x' - c^T x = c_B^T(x'_B - x_B) + c_N^T x'_N = -c_B^T B^{-1} N x'_N + c_N^T x'_N = \underbrace{\left(c_N^T - c_B^T B^{-1} N\right)}_{=:r^T \in \mathbb{R}^{1,n-m}} x'_N = \sum_{k=1}^{n-m} r_k \cdot \underbrace{x'_{N,k}}_{\geq 0}.$$

If $r \geq 0$, then the costs do not decrease for any choice of $x'_N \geq 0$, i.e. the corner x was already optimal. Conversely, if the k-th component $r_k < 0$, then the cost decreases if the k-th component of x'_N increases. We choose x'_N in the direction of the component with the strongest decrease. If $r_i = \min_j\{r_j < 0\}$, then we choose $x'_N = \xi e_i$, i.e. $c^T x' - c^T x = r_i \xi < 0$.

We choose ξ as large as possible because the larger ξ is, the stronger the cost reduction. However, the constraints $Ax' = b, x' \geq 0$ must still be satisfied. It holds:

$$0 \leq x'_B = x_B - B^{-1}Nx'_N = x_B - \xi \underbrace{B^{-1}Ne_i}_{=:v \in \mathbb{R}^m}.$$

There are two possible situations:

- $v \leq 0$: Then $x'_B \geq 0$ for any choice of $\xi > 0$ is fulfilled. We then let $\xi \to \infty$, thus the cost function is unbounded, so there is no solution and we are done.
- $\exists k : v_k > 0$: Then we choose

  $$\xi := \min_{k:v_k>0} \left\{ \frac{(x_B)_k}{v_k} \right\} \quad \Rightarrow \quad x'_B = x_B - v\xi \geq 0,$$

  and (at least) one component of x'_B is zero. Then it holds:

  $$x' = \left(x'_1, \ldots, x'_{k-1}, 0, x'_{k+1}, \ldots, x'_m, 0, \ldots, 0, \xi, 0, \ldots, 0\right)^T,$$

  where the ξ component is at the $(m+i)$-th index, because we have chosen $x'_N = \xi e_i$. We have thus found a new point with lower costs. We then swap the zero value at position k with ξ, and also swap the k-th and $(m+i)$-th columns of A and c^T. We denote by $B' \in \mathbb{R}^{m \times m}$ the matrix formed of the m first columns of A after swapping the columns. Since B' is invertible (shown right after in Lemma 5.2), x' is a corner and one can start again the iteration.

Lemma 5.2

The new "basis matrix" B' corresponding to x' has linearly independent columns.

Proof. (Taken from [2, Lemma 5.6.1]) Consider the matrix $M := B^{-1}B'$. It is sufficient to show that M has linearly independent columns to show that B' has linearly independent columns (because M invertible $\iff$ B' invertible). Note that the matrix B' has the same columns as B except for the column k, therefore $B^{-1}B'e_\ell = e_\ell$ for $\ell \neq k$. We deduce that M has linearly independent columns if and only if $M_{k,k} \neq 0$.

However, by construction of B', its column k is the i-th column of N: $Me_k = B^{-1}B'e_k = B^{-1}Ne_i = -v$. The index k has been chosen so that $v_k < 0$, thus $M_{k,k} = e_k^T Me_k = -v_k > 0$. □

Lemma 5.2 shows that we indeed have a corner after each iteration of the simplex algorithm.

Simplex step (Phase II) Let $x = (x_B, 0)^T$ with $x_B \in \mathbb{R}^m$, $A = \begin{pmatrix} B & N \end{pmatrix}$ with $B \in \mathbb{R}^{m\times m}$ invertible, and $c^T = \begin{pmatrix} c_B^T & c_N^T \end{pmatrix}$. Then we compute:

1. Set $r^T := c_N^T - c_B^T B^{-1}N$.
2. If $r \geq 0$ element-wise, then x is already optimal. Otherwise, look for $r_i := \min_k\{r_k\}$.
3. Set $v := B^{-1}Ne_i$, where Ne_i is the i-th column of N.
4. If $v \leq 0$ element-wise, then the cost function is unbounded, so there is no solution. Otherwise, we determine j such that $\frac{x_j}{v_j} = \min_{k:v_k>0}\left\{\frac{x_k}{v_k}\right\} =: \xi \geq 0$.
5. We replace x_k by $x_k - \xi v_k$ for $k = 1, \ldots, m$ (except $k = j$, where we set $x_j := \xi$: this corresponds to swapping the j-th and $(m+i)$-th coordinates x). Then we swap the j-th column with the $(m+i)$-th column in the matrix A and in the vector c^T and start again at the first step.

Remark 5.8

Degenerate corners may occur, i.e. corners with more than $n - m$ zero components. Then we could obtain $\xi = 0$ (which indeed happens if $x_j = 0$ and $v_j > 0$). In this case, we would swap a zero value with another zero, and the new corner would be the old one. We would be then stuck. However, this generally does not occur in practice due to rounding errors.

Remark 5.9 – Numerical Linear Algebra of the Simplex step

1. We compute $c_B^T B^{-1}N$ by solving the system of equations $B^T u = c_B$. This yields $u^T := c_B^T B^{-1}$, then we compute $r := c_N - N^T u$. Similarly, we obtain $v = B^{-1}Ne_i$ by solving $Bv = Ne_i$.
2. In the first step, we compute an LR[a] decomposition of B with $PB = LR$. Then we replace the j-th column of B with the i-th column of N, i.e. $B' = B\Gamma$ with

$$\Gamma e_\ell = e_\ell,\ \forall 1 \leq \ell \leq m, \ell \neq j, \quad \Gamma e_j = v = B^{-1}Ne_i \in \mathbb{R}^m.$$

   We have $v_j > 0$ by definition of j, which makes the columns of Γ all linearly independent. Thus, Γ is invertible, and thus so is B' (because B is invertible). We store the LR decomposition of B for further steps. If there are more than m simplex steps, perform again a LR decomposition on $B\Gamma_1 \cdots \Gamma_m = \tilde{B}$.
3. We perform a column swap (possibly via a pointer field to avoid reconstructing a matrix at every iteration), i.e.

$$\left(\begin{array}{c|c} B & N \end{array}\right) = \left(\begin{array}{ccc|ccc} A_{i_1} & \ldots & A_{i_m} & A_{i_{m+1}} & \ldots & A_{i_n} \end{array}\right)$$

   and i_k is the original position of the current column k. For the decomposition in one step, we need (if $PB = LR$ is already known) $m^2 + (m-j)m \leq 2m^2$ operations.

[a] Also called an LU decomposition.

So far, we have described how to move from one corner to the next. It is still necessary to clarify how to determine an initial corner.

Idea (Initial Corner search)
We introduce m new variables: $x_{n+1}, \dots, x_{n+m}$ and consider the vector $\bar{x} = (x_1, \dots, x_{n+m})$. Furthermore, we set $\bar{c}^T := \begin{pmatrix} 0 & \mathbf{1}^T \end{pmatrix}$ with $\mathbf{1}^T = (1, \dots, 1) \in \mathbb{R}^m$ and $\bar{A} := \begin{pmatrix} A & I_m \end{pmatrix}$. We assume without loss of generality that $b \geq 0$: if this is not the case, we multiply the bad component and the associated line in A by (-1).

Remark 5.10 – Auxiliary Problem (Phase I)

We now want to solve the auxiliary problem $\bar{A}\bar{x} = b$ with $\bar{x} \geq 0$ under $\bar{c}^T\bar{x} = \min!$. A corner for the auxiliary problem is $\bar{x} = (0, b)$: it solves $\bar{A}\bar{x} = b$, it is nonnegative since we assumed $b \geq 0$, it has at most m nonzero components, and the identity matrix obviously has linearly independent columns.
If we assume the initial problem to be feasible, one can find an optimal solution among the corners of the auxiliary problem by using Phase II, and for such optimum we have $x_{n+1} = \dots = x_{n+m} = 0$. Thus,

- $\bar{A}\bar{x} = Ax = b$
- $\bar{x} \geq 0 \implies x \geq 0$.

Moreover, since this optimum is a corner for the auxiliary problem, $\bar{x}$ has at most m nonzero components and the columns in $\bar{A}$ corresponding to those nonzero components are linearly independent. But owing to $x_{n+1} = \dots = x_{n+m} = 0$, we deduce (up to a permutation) $x = (x_B, 0)$ with $x_B \in \mathbb{R}^m$ and the columns in A corresponding to x_B are linearly independent.
Finally, x is a corner for the initial problem.

With this, we have finished the description of the Simplex Algorithm. We now only need to show that the minima actually occur at the corners, which we need for the construction of the initial corner. We will prove this with the algorithm just constructed, but without creating a circular reference with the auxiliary problem.

Theorem 5.1 – Minima at corners

Assume the linear optimization problem $Ax = b, x \geq 0$ and $c^T x = \min!$ has a solution. Then at least one corner gives the solution.

Proof. (Adapted from [2, Theorem 4.2.3]). Let $x = (x_B, x_N)$ in the feasible region, i.e. $Ax = b$ and $x \geq 0$. Consider all feasible solutions x' such that $c^T x' \leq c^T x$, and among them choose one that has the maximal number of zero components. To show that x' is a corner, we still need to show that the columns in A associated to nonzero components of x' are linearly independent.

Let $J = \{j = 1, \dots, n : x'_j > 0\}$, and suppose the columns of A_J are linearly dependent. Then, there is a nonzero vector v such that $A_J v = 0$. Let w the zero-padded vector such that $w_J = \pm v$, we have $Aw = \pm A_J v = 0$. The sign can be determined such that w satisfies

(i) $c^T w \leq 0$.

(ii) There is $j \in J$ with $w_j < 0$.

Indeed, if $c^T w = 0$, (i) holds and (ii) can be obtained by eventually changing the sign of w (note that we don't have the restriction $w \geq 0$ since it is not required to be a feasible vector). Assume now that $c^T w \neq 0$, and again by an eventual change of sign we get (i). But (ii) may fail in this case, which means that $w \geq 0$. Let $y(t) = x' + tw$ for $t \geq 0$, all such $y(t)$ are feasible and the objective function tends to $-\infty$ as $t \to \infty$, i.e. the problem is unbounded. So, by assuming the linear problem has a solution, we can indeed choose the sign in w such that (i) and (ii) hold.

Let us go back to showing that the columns of A_J are linearly independent, by assuming they are not. We have $Ay(t) = b$ for all $t \geq 0$. For $t = 0$, the vector $y(0) = x'$ has all components from

J positive and all others zero (by definition of J). For any $j = 1, \dots, n$, we have $y_j(t) = x'_j + tw_j$. For j such that $x'_j = 0$, w_j is defined such that $w_j = 0$ and thus $y_j(t) = 0$ for all t. For $j \in J$ such that $w_j > 0$, $y_j(t) \geq 0$ for all j. For $j \in J$ such that $w_j < 0$, $y_j(t)$ is a decreasing function of t. Therefore, there is a t^* such that $y_j(t^*) = 0$ for some $j \in J$ such that $w_j < 0$. Morevoer, $c^T y(t^*) = c^T x' + tc^T w \leq c^T x$. This vector $y(t^*)$ is feasible and has more zero components than x'. This contradicts our assumption that x' has the largest number of zero components among all feasible solutions with better cost than x. Finally, the columns in A associated to nonzero components of x' are linearly independent. □

Note that Theorem 5.1 does not say there is a unique solution that is a corner, simply that among all solutions in the feasible set at least one is a corner.

Remark 5.11

If a solution exists, then the simplex method finds it after a finite number of steps, because there are also only a finite number of corners. The expected value is $\frac{3m}{2}$ steps; in the worst case, all corners must be traversed, i.e. $\mathcal{O}(2^{m-1})$ steps are necessary.

5.4 Duality

As seen in the second section, there is a linear program that we would like to call the "primal problem" in the future. Here we had the tableau:

$$\begin{array}{c|c} A & b \\ \hline c^T & \end{array},$$

which corresponds to various problems (called the *Primal problems*):

(a) $Ax \leq b$ and $c^T x = \min!$

(b) $Ax \leq b, x \geq 0$ and $c^T x = \min!$

(c) $Ax = b, x \geq 0$ and $c^T x = \min!$

Idea
We would like to obtain a priori *bounds on the optimal cost of the primal problem, without having to solve it first.*

Let us focus on the problem type (b) for this explanation. One way of obtaining a lower bound is to note that for $d \leq c$, $d^T x \leq c^T x$ because $x \geq 0$. For any $d \in \mathbb{R}^n$ such that $d \leq c$, $d^T x$ is a "lower bound" for the objective of the primal problem. However, this "lower bound" still depends on x, hence the quotes on "lower bound". In order to obtain a lower bound on $d^T x$, we can use the condition of the primal problem: for any $y \geq 0$,

$$y^T Ax \leq y^T b = b^T y.$$

If we let $d = A^T y$, we get $d^T x \leq b^T y$. The good thing is that the bound does not depend on x anymore, the bad thing is that the bound is an upper bound instead of a lower bound. To fix this, it suffices to consider $y \leq 0$ instead of $y \geq 0$, and we obtain $d^T x \geq b^T y$ for any $y \leq 0$. Finally, we have $c^T x \geq b^T y$, for any y such that $y \leq 0$ and $d = A^T y \leq c$. We can maximize $b^T y$ under the same conditions and get a lower bound for the primal problem. This leads to the so-called "Dual problem":

$$(\text{b}^*) \left| \begin{array}{ll} \max & b^T y \\ s.t. & A^T y \leq c \\ & y \leq 0. \end{array} \right.$$

For problem type (a), the same holds except that we don't know the sign of x so we can only look for a lower bound to $c^T x$ under the form $d^T x$ with $d = c$. Another way of saying this is that

we are looking for a lower bound that holds for both $x_j \geq 0$ (i.e. $d_j \leq c_j$) and $x_j \leq 0$ (i.e. $d_j \geq c_j$), because the sign of x_j is yet unknown. We obtain

$$(\text{a}^*) \left| \begin{array}{ll} \max & b^T y \\ s.t. & A^T y = c \\ & y \leq 0. \end{array} \right.$$

For problem type (c), the lower bound is of the form $d^T x \leq c^T x$ for $d \leq c$, and $y^T A x = b^T y$ for all $y \in \mathbb{R}^m$. By maximizing this lower bound with respect to y, we get

$$(\text{c}^*) \left| \begin{array}{ll} \max & b^T y \\ s.t. & A^T y \leq c \\ & y \in \mathbb{R}^m. \end{array} \right.$$

Note that we wrote $y \in \mathbb{R}^m$ here to emphasize the fact that there is no sign condition on y, we will generally not write this line.

Another way of obtaining the three dual problems is to obtain one of them (e.g. (b*)), and then to use the equivalence between (a) or (c) to transform them into (b). Using the dual problem of (b), namely (b*), we can obtain the dual problems (a*) and (c*).

Definition 5.4 – Dual Problem

We now consider the dual problem:

$$\begin{array}{c|c} A^T & c \\ \hline b^T & \end{array}$$

Then we have the various problems:

(a*) $A^T y = c$, $y \leq 0$ and $b^T y = \max!$

(b*) $A^T y \leq c$, $y \leq 0$ and $b^T y = \max!$

(c*) $A^T y \leq c$, $b^T y = \max!$

We see that, for example, (a*) is equivalent to (c), and of course (a*), (b*) and (c*) are equivalent to each other. We will work with problem (c*) in the future, because we were concerned with problem type (c) for the primal problem.

Definition 5.5 – Feasible region

We say $x \in \mathbb{R}^n$ is feasible (for the primal problem (c)) if $Ax = b$ and $x \geq 0$, and $y \in \mathbb{R}^m$ is feasible (for the dual problem (c*)) if $A^T y \leq c$.

The statement of the following theorem is very clear if one understands the above construction of the dual problem:

Theorem 5.2 – Weak duality

For all feasible x and y, it holds that:

$$c^T x \geq y^T b.$$

Proof. Since x is feasible, $Ax = b$ and $x \geq 0$, and since y is feasible, $A^T y \leq c$. Thus, it holds that:

$$y^T b = y^T (Ax) = (y^T A)x \leq c^T x.$$

□

Corollary 5.1 – Optimal solutions

If $c^T x = y^T b$ for feasible x and y, then x and y are already optimal.

Proof. For all feasible $\eta \in \mathbb{R}^m$ (for the dual problem), it holds that $\eta^T b \leq c^T x = y^T b$. Then $y^T b = \max!$, i.e. y is optimal for the dual problem. Analogously, we obtain for all feasible $\zeta \in \mathbb{R}^n$ (for the primal problem) that $c^T \zeta \geq y^T b = c^T x$, which means x is optimal. □

Corollary 5.2 – Unbounded $\implies$ *-unfeasible

If the infimum in the primal problem is $-\infty$, then the feasible region of the dual problem is empty. Conversely, if the supremum of the dual problem is $+\infty$, then the feasible region of the primal problem is empty.

Proof. According to weak duality, it holds that $-\infty = \inf\{c^T x : x \text{ feasible}\} \geq y^T b$, $\forall y$ feasible. But such a y cannot exist. Analogously, we obtain $+\infty \leq c^T x$, $\forall x$ feasible. Thus, there is no feasible x. □

Theorem 5.3 – Duality theorem

Let $x \in \mathbb{R}^n$ and $y \in \mathbb{R}^m$ be feasible. Then it holds that:

$$x, y \text{ optimal} \iff c^T x = y^T b.$$

Proof. The direction "⇐" has already been shown. We must now show under the assumption that x is optimal that there exists a feasible y with $c^T x = y^T b$. It would work as well if one assumed y optimal and looked for a feasible x s.t. $c^T x = y^T b$. Since x is optimal, we can assume it is a corner by Theorem 5.1 We use the simplex method for $A = (B, N)$, $c^T = (c_B^T, c_N^T)$, $x = (x_B, 0)$, and $b = Ax = Bx_B$. Then it holds that $c^T x = c_B^T x_B = c_B^T B^{-1} b = y^T b$ for $y^T := c_B^T B^{-1} \in \mathbb{R}^m$. We must now show that this defined y is feasible. Note that

$$x = \begin{pmatrix} x_B \\ 0 \end{pmatrix} \text{ optimal} \iff r^T := c_N^T - c_B^T B^{-1} N \geq 0,$$

hence

$$y^T A = c_B^T B^{-1} A = c_B^T B^{-1} \begin{pmatrix} B & N \end{pmatrix} = \begin{pmatrix} c_B^T & c_B^T B^{-1} N \end{pmatrix}.$$

Since $r^T \geq 0$, we finally obtain:

$$y^T A \leq \begin{pmatrix} c_B^T & c_N^T \end{pmatrix} = c^T,$$

thus y is feasible. □

Corollary 5.3

If the primal problem has a solution, then the dual problem also has a solution. In particular, the minimum of the primal problem is the maximum of the dual problem.

Remark 5.12

The simplex method always computes the solution of the dual problem automatically since $B^T y = c_B$ must be solved anyway, see Remark 5.9.

Corollary 5.4 – Optimality criterion

Let x, y be feasible. Then x and y are optimal if and only if for all $j = 1, \ldots, n$ it holds that $x_j = 0$ or $(A^T y)_j = c_j$.

Proof. ⇐: Let $(y^T A)_j = c_j^T$ for $x_j \neq 0$. Let $J = \{j = 1, \ldots, n : x_j \neq 0\}$. Since x and y are

feasible, by Theorem 5.3 we need to check

$$c^T x = y^T b \iff c^T x = y^T Ax \iff c^T x = \sum_{i=1}^{n} (y^T A)_i x_i \iff c_J^T x_J = \sum_{i \in J} (y^T A)_i x_i.$$

By definition of y, we have $(y^T A)_i = c_i$ for all $i \in J$, hence the last equality holds and x and y are optimal.
$\Rightarrow$: Let x, y be optimal. Then it holds that:

$$y^T b = c^T x \quad \Rightarrow \quad \left(c^T - y^T A\right) x = 0.$$

Since y is feasible, we have $c^T - y^T A \geq 0$. Since x is feasible, we have $x \geq 0$. Hence $\left(c^T - y^T A\right) x$ is a sum of nonnegative terms, and it is zero if and only if all the summands are zero. Thus, it holds for every j that $(c^T - y^T A)_j = 0$ or $x_j = 0$. □

Example 5.2

In a free market economy, it is about equilibrium conditions. To produce the products $j = 1, \ldots, n$, the quantity a_{ij} of raw materials $i = 1, \ldots, m$ is required. The value of product j is c_j and the quantity produced of j is $x_j \geq 0$. The price of raw material i is $y_i \geq 0$ and the quantity of raw material i is b_i. Thus, we obtain as a constraint $Ax \leq b$, i.e.

$$\sum_j a_{ij} x_j = Ax \leq b \quad \forall i.$$

The revenue of the company is

$$\sum_j c_j x_j = c^T x.$$

and the expenses

$$\sum_j \left(\sum_i a_{ij} y_i \right) x_j = y^T Ax.$$

Thus, the profit is then $c^T x - y^T Ax = (c^T - y^T A)x$. This should be maximized. From the perspective of the raw material suppliers, it follows that as long as the company's profit is positive, the prices should be increased (because the company is not making a loss, it will continue to buy). Thus, we obtain as a constraint $y^T A \leq c^T$ with $y \geq 0$ for the dual problem of the raw material suppliers. These two perspectives are reflected in the doctrines of the companies and suppliers:

1. If $(c^T - y^T A)_j < 0$, then set $x_j := 0$. Here, the costs for producing product j are greater than the sales revenue, so production of this product is stopped.

2. If $(Ax - b)_i < 0$, then set $y_i := 0$. Here, the supply for raw material i is greater than the demand, so this raw material is available for free.

With the duality theorem, it holds that: $c^T x = \max$ and $Ax \leq b$ with $x \geq 0$ as well as $b^T y = \min$ and $A^T y \geq c$ with $y \geq 0$. We interpret this as the value creation $c^T x$ being maximized in the free market economy.

Remark 5.13 – Effects of Small Perturbations

We have $c^T x = \min$ and $Ax = b$ with $x \geq 0$ given. We want to know how the minimal costs $c^T x$ change if we change b. We see that if instead of b, we now have $b + \Delta b$ with Δb "sufficiently small", then x becomes $x + \Delta x$. In the dual problem $A^T y \leq c, (b + \Delta b)^T y = \max$, the feasible region does not change, so the solution y of the new dual problem remains in the same corner if Δb is small enough. We now consider the changes in costs and use the duality theorem:

$$c^T (x + \Delta x) - c^T x = (b + \Delta b)^T y - b^T y = (\Delta b)^T y$$

We interpret the dual variable y as the change in costs divided by the change in supply. In terms (where y was the price) of the above example, this results in:

$$\text{Price} = \frac{\text{Change in costs}}{\text{Change in supply}}$$

5.4.1 Preparation for Karmarkar, Projection, Scaling

Motivation
We will now prepare the algorithm of Karmarkar in this and the following sections, which was published in 1984 in the article "A new polynomial-time algorithm for linear programming". In the simplex method, we had an exponential runtime in the worst case, while in this algorithm we always have a polynomial runtime.

In this section, we will set the foundations of the algorithm. We seek x with $Ax = b, x \geq 0$ and $c^T x = \min!$. Let x^0 be given in the interior of the feasible region (i.e. $x_j > 0$ for $j = 1, \ldots, n$). We now seek a feasible $x^1 := x^0 + \Delta x$ with lower costs. It requires two ingredients, the projection and the scaling, which we discuss now.

Idea (Projection)
We want to choose Δx in the direction of the steepest descent of the cost function, i.e. Δx should be a negative multiple of c, because it holds that:

$$c^T \Delta x = \|c^T\| \cdot \|\Delta x\| \cos \angle(c, \Delta x)$$

and it is the smallest when the cosine is -1, i.e. when $\Delta x = -c$. However, we still need the feasibility of x^1, in particular $Ax^1 = b$, i.e. it must hold that $A\Delta x = 0$, so $\Delta x \in \operatorname{Ker}(A)$. Therefore, Δx cannot generally be chosen as above, and it must be restricted to $\operatorname{Ker}(A)$, i.e.:

$$\Delta x := -\xi v$$

where ξ is a (yet to be determined) scaling factor and v is the orthogonal projection of c onto $\operatorname{Ker}(A)$, i.e. $v \in \operatorname{Ker}(A)$ and $v - c \perp \operatorname{Ker}(A)$. In particular, $v - c \in \operatorname{Ker}(A)^\perp = \operatorname{Image}(A^T)$ and there exists $\lambda \in \mathbb{R}^m$ with $v - c = -A^T \lambda$. In matrix form, this means:

$$\begin{pmatrix} I_n & A^T \\ A & 0 \end{pmatrix} \begin{pmatrix} v \\ \lambda \end{pmatrix} = \begin{pmatrix} c \\ 0 \end{pmatrix}$$

Without loss of generality, we can assume that the rows of the matrix A are linearly independent (otherwise, we omit linearly dependent rows that only provide redundant information for our problem, see Remark 5.6). Thus, v and λ are always uniquely determined. We now consider the choice of the scaling factor ξ. We must not go too far here, otherwise we would fall out of the feasible region. We first note that it holds that $c = v + A^T \lambda$. Thus, the change in costs, since $v \in \operatorname{Ker}(A)$, is:

$$c^T \Delta x = \left(v + A^T \lambda\right)^T (-\xi v) = -\xi v^T v - \xi \lambda^T A v = -\xi \|v\|^2$$

We have 3 possibilities for what v can be:

1. *$v = 0$: Then there exists a μ such that $c = A^T \mu$, and $c^T x = \mu^T A x = \mu^T b$ for all feasible x. Thus, all feasible x are optimal and we are done.*
2. *For all ξ, $x^1 = x^0 - \xi v \geq 0$: Then the cost function is unbounded within the feasible region, so there is no minimum and we are done.*
3. *There exists a $\xi^* > 0$ for which $x^0 - \xi^* v$ is on the boundary of the feasible region, i.e. we have $x^0 - \xi^* v \geq 0$ and there exists a j with $(x^0 - \xi^* v)_j = 0$.*

We then choose (analogous to the simplex method)

$$\xi^* := \min_{j : v_j > 0} \frac{x_j^0}{v_j}$$

However, we stop shortly before the boundary in order to remain within the feasible set and be able to apply the same idea iteratively, i.e. for an $\alpha < 1$, we set

$$x^1 := x^0 - \alpha \xi^* v.$$

From the expression of ξ^* one can see that the further away x^0 is from the boundary, the larger the step. One could apply iteratively the projection step only, but then the scaling ξ^* could quickly become very small and not yield a large improvement for the cost. One idea of Karmarkar consists in distorting the feasible set before applying the projection, so that x^1 is closer to the center of the new simplex. Because we are closer to the center, the step can be large and the improvement with respect to the cost is large. Karmarkar's algorithm will proceed in three main steps: get the new simplex so that the current iterate is closer to the center, apply the projection step to get a new iterate, and transform back the new iterate to the original coordinates.

Visual explanation with Figure 5.2: if the lines are in the opposite direction of c, the starting point x^2 can improve just a little bit before reaching the boundary, while a point at the center can improve much more.

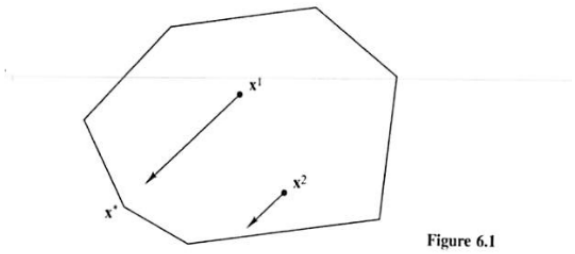

Figure 5.2: Figure taken from `https://ise.ncsu.edu/wp-content/uploads/sites/9/2021/07/LPchapter-6.pdf`

For the sake of presentation, we assume now that the initial iterate x^0 is close to the center of the original simplex. Moreover, the first application of the scaling idea does nothing since we would transform x^0 to x^0. So for the explanation of the scaling idea, assume that we already have an iterate x^1 (obtained via the projection idea).

Idea (Scaling)
We choose a diagonal matrix $D = \operatorname{diag}(d_1, \dots, d_n)$ such that $Dx^1 = x^0$, i.e.

$$d_j := \frac{x_j^0}{x_j^1} > 0.$$

This matrix transforms x^1 into x^0, i.e. brings x^1 back to x^0, the center of the simplex. We now write the initial problem $Ax = b, x \geq 0$ and $c^T x = \min!$ in the new coordinates $\tilde{x} := Dx$:

$$AD^{-1}\tilde{x} = b, \quad \tilde{x} \geq 0, \quad c^T D^{-1}\tilde{x} = \min!.$$

We apply the projection step to the scaled problem with the scaled initial value $\tilde{x}^1 = Dx^1 = x^0$:

$$\begin{pmatrix} I & D^{-1}A^T \\ AD^{-1} & 0 \end{pmatrix} \begin{pmatrix} \tilde{v} \\ \tilde{\lambda} \end{pmatrix} = \begin{pmatrix} D^{-1}c \\ 0 \end{pmatrix}.$$

One can rewrite this system as

$$\begin{pmatrix} D^{-1} & 0 \\ 0 & I \end{pmatrix} \begin{pmatrix} D^2 & A^T \\ A & 0 \end{pmatrix} \begin{pmatrix} D^{-1} & 0 \\ 0 & I \end{pmatrix} \begin{pmatrix} \tilde{v} \\ \tilde{\lambda} \end{pmatrix} = \begin{pmatrix} D^{-1}c \\ 0 \end{pmatrix} \iff \begin{pmatrix} D^2 & A^T \\ A & 0 \end{pmatrix} \begin{pmatrix} v \\ \lambda \end{pmatrix} = \begin{pmatrix} c \\ 0 \end{pmatrix},$$

where $v = D^{-1}\tilde{v}$ and $\tilde{\lambda} = \lambda$.

Another way of understanding this new system is the following: since $\tilde{v}$ corresponds to the scaled coordinates, one can define v (its corresponding vector in the initial coordinates) by $\tilde{v} := Dv$. The vector $\tilde{\lambda}$ is only there to specify that $c - v \in \operatorname{Image}(A^T)$, so its scaling doesn't really matter.

We no longer have an orthogonal projection here, but a weighted projection. More precisely, this is the orthogonal projection of the induced scalar product of D^2, i.e.

$$\langle v, w\rangle = v^T D^2 w = \tilde{v}^T \tilde{w}.$$

Note that the v obtained with this weighted projection is different than the one we would obtain from the projection step alone, underlining the importance of rescaling. Then we set, according to the projection idea,

$$\tilde{x}^2 := \tilde{x}^1 - \alpha\xi^* \tilde{v}, \quad \xi^* := \min_{j:v_j>0} \frac{x_j^1}{v_j}.$$

We recall that the step ξ^* is chosen so that the new iterate obtained from the current one in the original coordinates remains within the inside of the feasible set. After going back to the initial coordinates, we get

$$x^2 = x^1 - \alpha\xi^* v.$$

For the next steps, we choose a diagonal matrix D with $Dx^2 = x^0$, perform the projection, etc.

Scaling algorithm Let x^0 be given in the interior of the feasible region. For $k = 0, 1, \ldots$, we then perform the following steps:

1. For the matrix $M := \mathrm{diag}(d_1^2, \ldots, d_n^2)$ with $d_j := \frac{x_j^0}{x_j^k} > 0$, solve the system of equations

$$\begin{pmatrix} M & A^T \\ A & 0 \end{pmatrix} \begin{pmatrix} v \\ \lambda \end{pmatrix} = \begin{pmatrix} c \\ 0 \end{pmatrix}$$

2. If $v_j \leq 0$ for all j, then there is no minimum and we are done. Otherwise, we choose

$$\xi := \min_{j:v_j>0} \frac{x_j^k}{v_j}.$$

and set $\Delta x := \alpha\xi v$ for a suitable $\alpha < 1$ close to one. Then

$$x^{k+1} := x^k - \Delta x.$$

By construction, the costs decrease at each step.

Remark 5.14 – Linear algebra of the algorithm

For the first step, we have three possibilities at the linear algebra level:

1. We can perform a Gaussian elimination of the matrix $\begin{pmatrix} M & A^T \\ A & 0 \end{pmatrix}$ and obtain as the solution for λ the system of equations:

$$AM^{-1}A^T\lambda = AM^{-1}c$$

and for v we get:

$$Mv = c - A^T\lambda,$$

thus the system of equations is decoupled. For the symmetric and positive definite matrix $AM^{-1}A^T$, we can perform a Cholesky decomposition LL^T.

2. We can also perform a QR decomposition for $AM^{-1}A^T = AD^{-1}D^{-1}A^T$ in the sense that

$$D^{-1}A^T = Q\begin{pmatrix} R \\ 0 \end{pmatrix},$$

where Q is orthogonal and R is an invertible triangular matrix. Then it holds that:

$$AM^{-1}A^T = \begin{pmatrix} R^T & 0 \end{pmatrix} Q^T Q \begin{pmatrix} R \\ 0 \end{pmatrix} = R^T R.$$

Then $R^T R\lambda = AM^{-1}c$ is to be solved.

3. We can use iterative methods, such as the CG method, to compute the first step. Since A is usually sparsely populated (and M or M^{-1} anyway), solving by the CG method is recommended. This way, no non-zero elements are to be stored.

Remark 5.15 – Finding the starting value

We now introduce an additional column in A, i.e.

$$\hat{A} := \begin{pmatrix} A & b - Ae \end{pmatrix} \quad \text{with } e := \begin{pmatrix} 1 \\ \vdots \\ 1 \end{pmatrix} \in \mathbb{R}^n \quad \text{and } \hat{e} := \begin{pmatrix} e \\ 1 \end{pmatrix} \in \mathbb{R}^{n+1}$$

Then it holds that:

$$\hat{A}\hat{e} = Ae + b - Ae = b$$

We now want to reduce this to a problem of the form $\hat{A}\hat{x} = b, \hat{x} \geq 0, \hat{c}^T\hat{x} = \min!$. For this, we choose:

$$\hat{c} := \begin{pmatrix} c \\ c_{n+1} \end{pmatrix} \quad \text{with } c_{n+1} \text{ large enough.}$$

Then the last component of $\hat{x}$ would be zero to compensate for the size of c_{n+1}, with which we can forget the last component again. As the starting value for the modified problem, we choose $x^0 := \hat{e}$ as the starting value, which is in the interior of the feasible region.

Remark 5.16 – Termination criterion

If Δx is "small", it means we are near a corner and we can jump to the nearest corner from the currently chosen x^k. Then we check the corner for optimality with one step of the simplex algorithm.

5.5 Karmarkar's algorithm

If the optimization problem is in a particular form, Karmarkar's algorithm is simpler to use. This special formulation is the following:

1. $Ax = 0$ with $A \in \mathbb{R}^{n \times n}$ and $x \in \mathbb{R}^n$
2. $\sum_{i=1}^n x_i = 1$
3. $x \geq 0$
4. $c^T x = \min$ with $c \in \mathbb{R}^n$, where the minimum is 0.
5. $e := \frac{1}{n} \begin{pmatrix} 1 \\ \vdots \\ 1 \end{pmatrix}$ is feasible.

Remark 5.17 – Achieving the special formulation

We are currently solving the problem $c^T x = \min!$, $Ax = b$, $x \geq 0$. We assume that there is a constant $\sigma > 0$ such that, for x^* optimal, $\sum_{i=1}^n x_i^* < (n+2)\sigma$.
We start by rescaling the problem: let $\tilde{x} = \frac{1}{\sigma}x$, we are now solving $\sigma c^T \tilde{x} = \min!$ under the

constraints $A\tilde{x} = \frac{1}{\sigma}b$, $\tilde{x} \geq 0$.
Introduce a variable y with the condition $y = 1$, then the constraint can be rewritten as $A\tilde{x} - \frac{1}{\sigma}by = 0$, $y = 1$, $\tilde{x}, y \geq 0$.
Finally introduce z, the slack variable for the inequality $\sum_{i=1}^{n} \tilde{x}_i < n + 2$: z must then satisfy $\sum_{i=1}^{n} \tilde{x}_i + z = n + 2$, or written differently, $\sum_{i=1}^{n} \tilde{x}_i + y + z = n + 3$.
The optimization problem is now $\sigma c^T \tilde{x} = \min!$ under the constraints $A\tilde{x} - \frac{1}{\sigma}by = 0$, $y = 1$, $\tilde{x}, y, z \geq 0$, $\sum_{i=1}^{n} \tilde{x}_i + y + z = n + 3$. We can rescale the last equation by $n + 3$.
In order to have $e = \frac{1}{n}(1, \ldots, 1)$ a feasible solution, we introduce an additional artificial variable w, and assign a large cost coefficient M to it. The aim is to have $w = 0$ at optimality, but a feasible vector might have $w \neq 0$. The modified optimization problem is now

$$\begin{aligned} &\text{Minimize} && \sigma c^T \tilde{x} + Mw \\ &\text{under the constraints} && A\tilde{x} - \frac{1}{\sigma}by + \left(\frac{1}{\sigma}b - Ae\right) w = 0 \\ & && \sum_{i=1}^{n} x_i + y + z + w = 1 \\ & && \tilde{x}, y, z, w \geq 0. \end{aligned}$$

Now, $e = \frac{1}{n+3}(1, \ldots, 1) \in \mathbb{R}^{n+3}$ is feasible.
In order to meet the final requirement, namely the objective being zero at optimality, it "suffices" to modify the cost into $\tilde{c} = c - \kappa e$, where κ is the optimal cost. In practice though, one does not know κ but one can use the cost of the dual problem as a lower bound of $c^T x$, and update it at every iteration.

Motivation
In Karmarkar's algorithm, we consider a transformation of the feasible region such that we transform the simplex into another simplex. This is done by a nonlinear mapping (scaling step detailed previously). In the scaling algorithm, the triangle was rotated and distorted. This will no longer happen here.

Notation 5.2

We denote by Δ the feasible region, i.e.

$$\Delta := \left\{ x \in \mathbb{R}^n : \sum_{i=1}^{n} x_i = 1, x_i \geq 0 \right\}.$$

Construction (Projective transformation of the simplex)
Let a be an interior point in Δ, i.e. $\sum a_i = 1$ and $a_i > 0$. We now consider the mapping $T_a : \Delta \to \Delta$ with

$$x \mapsto \frac{Dx}{\sum_{i=1}^{n} \frac{x_i}{a_i}} = T_a(x) \quad \text{with } D = \operatorname{diag}\left(\frac{1}{a_i}\right).$$

The mapping T_a has the following properties:

1. *$T_a(a) = e$ with $e = \frac{1}{n}(1, \ldots, 1)$.*
2. *T_a is well-defined and bijective with the inverse mapping*

$$T_a^{-1}(\tilde{x}) = \frac{D^{-1}\tilde{x}}{\sum_{i=1}^{n} a_i \tilde{x}_i}.$$

3. *$T_a(\partial\Delta) = \partial\Delta$.*
4. *If we write $\tilde{x} = T_a(x)$, then it holds that:*
   (a) *$Ax = 0 \iff AD^{-1}\tilde{x} = 0$,*

(b) $c^T x = 0 \iff c^T D^{-1}\tilde{x} = 0$,

(c) $c^T x > 0 \iff c^T D^{-1}\tilde{x} > 0$.

Construction (Iteration of Karmarkar)
We denote $x^{k+1} := \mathcal{K}(x^k)$, the mapping $\mathcal{K}$ uses the two steps explained previously: scaling and projection. We recall that the scaling is done to put the current iterate closer to the center of the simplex, in this case e. The projection step evolves the current iterate in the modified coordinates along $\mathrm{Proj}_{Ker A}\tilde{c}$. Finally the new iterate in the modified coordinates is brought back to the initial coordinates.

$$\begin{array}{ccc} x^k & \xrightarrow{\quad T_{x^k} \quad} & \tilde{x}^k = e \\ & & \Big\downarrow (*) \\ x^{k+1} & \xleftarrow[\quad T_{x_k}^{-1} \quad]{} & x^{k+1} = e + \Delta\tilde{x} \end{array}$$

In step $(*)$, we solve the problem $c^T D^{-1}\tilde{x} = \min!, AD^{-1}\tilde{x} = 0, \sum \tilde{x}_i = 1$ and $\tilde{x} \geq 0$. We choose $\Delta\tilde{x}$ in the negative direction of the orthogonal projection of $\tilde{c} = D^{-1}c$ onto $\mathrm{Ker}\, B$ with $B := \begin{pmatrix} AD^{-1} \\ \mathbf{1}^T \end{pmatrix}$, and denote this projection by P. The row $\mathbf{1}^T$ is added to take into account the constraint $\sum \tilde{x}_i = 1$. We then solve $P\tilde{c} = \tilde{v}$ with

$$\begin{pmatrix} I & B^T \\ B & 0 \end{pmatrix} \begin{pmatrix} \tilde{v} \\ \lambda \end{pmatrix} = \begin{pmatrix} \tilde{c} \\ 0 \end{pmatrix}$$

We then set

$$\Delta\tilde{x} := -\alpha r \frac{P\tilde{c}}{\|P\tilde{c}\|} \quad \text{so: } \tilde{x}^{k+1} = e - \alpha r \frac{P\tilde{c}}{\|P\tilde{c}\|}$$

with a parameter $0 < \alpha < 1$ and where $r := \frac{1}{\sqrt{n(n-1)}}$ is the radius of the largest circle in the simplex Δ.

Remark

The quantity $P\tilde{c}$ plays the role of v earlier, and $\frac{r}{\|P\tilde{c}\|}$ plays the role of ξ^* earlier.

We choose $x^0 := e$ and for $k = 0, 1, 2, \ldots$, we compute $x^{k+1} = \mathcal{K}(x^k)$ until the termination criterion is met.

5.6 Convergence of Karmarkar's algorithm

Motivation
We now assume the assumptions as in the previous section and will show the convergence of the algorithm, whose effort grows polynomially.

Theorem 5.4 – Convergence of Karmarkar's algorithm

Let $x^0 = e$ and let $x^{k+1} = \mathcal{K}(x^k)$ be the sequence defined by Karmarkar's algorithm (with $\alpha = \frac{1}{2}\sqrt{\frac{n-1}{n}}$). If the problem has a solution (i.e. if there exists a feasible x with $c^T x = 0$), then, for $q \in \mathbb{N}$ large enough, after

$$k \geq \frac{10}{3} n(\log n + 0.7q)$$

iterations, it holds that the reduction of the objective function satisfies:

$$\frac{c^T x^k}{c^T x^0} \leq 2^{-q}.$$

Proof. We need some auxiliary results to prove this. □

Definition 5.6 – Potential function

Let Δ be the simplex of the feasible region. Then we define the potential function:

$$\varphi : \overset{\circ}{\Delta} \to \mathbb{R} : \varphi(x) := \sum_{i=1}^{n} \log \frac{c^T x}{x_i} = n \log c^T x - \sum_{i=1}^{n} \log x_i,$$

where $\overset{\circ}{\Delta}$ denotes the interior of the simplex Δ.

Lemma 5.3

Let $x^0 = e$ and let $x^{k+1} = \mathcal{K}(x^k)$ be the sequence defined by Karmarkar's algorithm (with $\alpha = \frac{1}{2}\sqrt{\frac{n-1}{n}}$). If the problem has a solution, then it holds for all $k \in \mathbb{N}$ that:

$$\varphi(x^{k+1}) \leq \varphi(x^k) - \delta$$

with $\delta > 0.3$.

Proof. We need some properties of the potential function to prove this. □

Proposition 5.3 – Properties of the potential function

1. If x^* is a feasible solution of the problem $c^T x = \min!$, then for a sequence $x \to x^*$ along a line in $\overset{\circ}{\Delta}$, it holds that:
$$\varphi(x) \to -\infty.$$
2. If $\bar{x} \in \partial\Delta$ with $c^T \bar{x} > 0$ (i.e. not a solution), then $\varphi(x) \to +\infty$ for $x \to \bar{x}$ along a line in $\overset{\circ}{\Delta}$.
3. The potential function is invariant under projective transformations T_a for $a \in \overset{\circ}{\Delta}$.

Proof. 1. Let $x = x^* + tu$ with $u_i > 0$ for every i with $x_i^* = 0$. For t small enough, we have $x \in \overset{\circ}{\Delta}$,

$$0 \geq c^T x = t c^T u.$$

The equality only happens in the case where all points in Δ are optimal, which means $\varphi = -\infty$ everywhere on Δ and we're done. Apart from this trivial situation, we know that the optimum must occur on corners, and thus a feasible point in the interior of the simplex has a suboptimal cost $c^T x > 0$. We compute, in the non-trivial case,

$$\varphi(x^* + tu) = \sum_{i=1}^{n} \log \left(\frac{t c^T u}{x_i^* + t u_i} \right).$$

The argument of the logarithm is independent of t if $x_i^* = 0$. If $x_i^* > 0$, the argument of the logarithm goes to zero as $t \searrow 0$. But at least one x_i^* must be positive since x^* is normalized. Thus, the argument goes to zero and the function thus goes to $-\infty$.

2. We write analogously to above $x = \bar{x} + tu$ and set $u_i > 0$ if $\bar{x}_i = 0$. For $t \searrow 0$ small enough,

$x \in \overset{\circ}{\Delta}$, and we compute:

$$\varphi(\bar{x} + tu) = \sum_{i=1}^{n} \log\left(\frac{c^T\bar{x} + tc^Tu}{\bar{x}_i + tu_i}\right).$$

Since $\bar{x} \in \partial\Delta$, there exists an i with $\bar{x}_i = 0$. The numerator is strictly positive since $c^T\bar{x} > 0$, so as $t \searrow 0$ the function $\varphi(\bar{x} + tu) \to +\infty$.

3. For $\tilde{x} = T_a(x)$, i.e.

$$x = T_a^{-1}(\tilde{x}) = \frac{D^{-1}\tilde{x}}{\sum_{i=1}^n a_i\tilde{x}_i}$$

with $D = \operatorname{diag}\left(\frac{1}{a_i}\right)$, we define $\tilde{\varphi}$ by:

$$\tilde{\varphi}(\tilde{x}) = \varphi(x)$$

Thus, we compute:

$$\tilde{\varphi}(\tilde{x}) = n\log\left(c^T\frac{D^{-1}\tilde{x}}{\sum_{i=1}^n a_i\tilde{x}_i}\right) - \sum_{i=1}^{n}\log\left(\frac{a_i\tilde{x}_i}{\sum_{j=1}^n a_j\tilde{x}_j}\right)$$

For $\tilde{c}^T := c^TD^{-1}$, we thus obtain (with the logarithm rules):

$$\tilde{\varphi}(\tilde{x}) = n\log\tilde{c}^T\tilde{x} - \sum_{i=1}^{n}\log\tilde{x}_i - \underbrace{\sum_{i=1}^{n}\log a_i}_{=\text{ const}}.$$

Thus, all properties hold.

□

We will now show with the next lemma that the projection step in Karmarkar's algorithm can be interpreted as minimization over a circle.

Lemma 5.4

It holds that:

$$\min_{\substack{\|x-e\|\le\eta \\ \sum_i x_i = 1 \\ Ax = 0}} c^Tx = c^T\left(e - \eta\frac{Pc}{\|Pc\|}\right)$$

where P is the orthogonal projection onto $\operatorname{Ker}\begin{pmatrix} A \\ \mathbf{1}^T\end{pmatrix}$.

Proof. We find a lower bound for $c^T(x-e)$ and show that it is attained. First of all, since x and e are feasible, we have $x = Px$ and $e = Pe$. Thus,

$$c^T(x-e) = c^TP(x-e) = (Pc)^T(x-e).$$

By Cauchy-Schwarz:

$$(Pc)^T(x-e) \ge -\|Pc\|\|x-e\| \ge -\eta\|Pc\|.$$

Equality holds if $\frac{Pc}{\|Pc\|} = -\frac{x-e}{\|x-e\|}$ and $\|x-e\| = \eta$. This is exactly the case when $x - e = -\eta\frac{Pc}{\|Pc\|}$. Thus, the desired equality already holds. □

Proof of lemma 5.3. 1. We must show that, with $\alpha = \frac{1}{2}\sqrt{(n-1)/n}$, $\varphi(x^{k+1}) \le \varphi(x^k) - \delta$ with $\delta > 0.3$. We already know that

$$\varphi(x^{k+1}) = \tilde{\varphi}(\tilde{x}^{k+1}) = \tilde{\varphi}(e - \alpha r\tilde{d}),$$

with $\tilde{d} = \frac{P\tilde{c}}{\|P\tilde{c}\|}$ and $r = \frac{1}{\sqrt{n(n-1)}}$. The product αr has the role of η previously. It holds that:

$$\varphi(x^k) = \tilde{\varphi}(\tilde{x}^k) = \tilde{\varphi}(e).$$

We know that $\tilde{\varphi}$ has the same form as φ, but with $\tilde{c}$ instead of c. We leave out the tilde in the following calculation to simplify the notation, but we calculate in the tilde (transformed) variables.

2. We must now show that it holds that:

$$\varphi(e - \eta d) \leq \varphi(e) - \delta$$

with $x(\alpha) := e - \eta d, d := \frac{Pc}{\|Pc\|}, \eta = \alpha r$, and P is the orthogonal projection onto $\mathrm{Ker}\begin{pmatrix} A \\ \mathbf{1}^T \end{pmatrix}$. In particular, $Ax(\alpha) = 0$ and $\sum_i x(\alpha)_i = 1$. Now, consider the difference

$$\varphi(x(\alpha)) - \varphi(e) = \psi(\alpha) - \psi(0) + n\frac{c^T(x(\alpha) - e)}{c^T e} = n\frac{c^T(x(\alpha) - e)}{c^T e} + \int_0^\alpha \psi'(\beta)d\beta,$$

where $\psi(\alpha) := \varphi(x(\alpha)) - n\frac{c^T x(\alpha)}{c^T e}$. We estimate $\frac{c^T(x(\alpha)-e)}{c^T e}$ and $\psi(\alpha) - \psi(0)$, and then combine the estimates to obtain the result.

3. First estimate. We have $x(\alpha) - e = \eta d = \alpha r d$, which implies $\|x(\alpha) - e\| = \eta$. By definition, $x(\alpha)$ minimizes the cost $c^T y$ on the set of y such that $\|y - e\| \leq \alpha r = \eta$, $Ay = 0$ and $\sum_i y_i = 1$. Thus, if one considers the line from e to the optimum x^*, it intersects with the circle $\|y - e\| = \eta$ at a point x' which satisfies (by the previous lemma) $c^T x' \geq c^T x(\alpha)$. One can write $x' = e + \eta(x^* - e)$, so $\frac{c^T(x'-e)}{c^T(x^*-e)} = \eta$. Finally, using the optimality of x^* and in particular that $c^T x^* = 0$,

$$\frac{c^T(x(\alpha) - e)}{c^T e} = \frac{c^T(x(\alpha) - e)}{c^T(e - x^*)} \leq \frac{c^T(x' - e)}{c^T(e - x^*)} = -\frac{c^T(x' - e)}{c^T(x^* - e)} = -\eta.$$

4. Second estimate. We first investigate the auxiliary function ψ defined above, for which it holds that:

$$\psi'(\alpha) = \varphi'(x(\alpha))x'(\alpha) - \frac{n}{c^T e}c^T x'(\alpha)$$

Since $x(\alpha) = e - \alpha r d$, it follows that $x'(\alpha) = -rd$ and thus we obtain with the definition of φ:

$$\psi'(\alpha) = nr\underbrace{\left(\frac{1}{c^T e} - \frac{1}{c^T x(\alpha)}\right)}_{\leq 0} c^T d + r\sum_{i=1}^n \frac{d_i}{x_i(\alpha)}.$$

Since $d = \frac{Pc}{\|Pc\|}$ and P is a projection, it holds that $c^T Pc = c^T P^2 c = c^T P^T Pc = (Pc)^T(Pc) \geq 0$, so $c^T d \geq 0$. Furthermore, it holds that:

$$r\sum_{i=1}^n \frac{d_i}{x_i(\alpha)} = r\sum_{i=1}^n \frac{d_i}{\frac{1}{n} - \alpha r d_i} = nr\sum_{i=1}^n \frac{d_i}{1 - \alpha r n d_i}.$$

Since $\sum_i d_i = 0$, it follows that:

$$r\sum_{i=1}^n \frac{d_i}{x_i(\alpha)} = nr\sum_{i=1}^n d_i\left(\frac{1}{1 - \alpha r n d_i} - 1\right) = nr\sum_{i=1}^n d_i\frac{\alpha r n d_i}{1 - \alpha n r d_i} = \alpha(nr)^2\sum_{i=1}^n \frac{d_i^2}{1 - \alpha r n d_i}.$$

Since $\sum_{i=1}^n d_i^2 = \|d\|^2 = 1$, we obtain $|d_i| \leq 1$ for all i and thus $\frac{1}{1-\alpha r n d_i} \leq \frac{1}{1-\alpha r n}$. It finally holds that:

$$r\sum_{i=1}^n \frac{d_i}{x_i(\alpha)} \leq \frac{\alpha(rn)^2}{1 - \alpha r n}.$$

With the above, it finally holds that:

$$\psi(\alpha) - \psi(0) \leq \int_0^\alpha \frac{\beta(rn)^2}{1 - \beta r n}d\beta = -\alpha r n - \log(1 - \alpha r n)$$

5. We now put both estimates together again and obtain in total:

$$\varphi(x(\alpha)) - \varphi(e) \leq -2\alpha rn - \log(1 - \alpha rn).$$

The right side becomes minimal for $\alpha rn = \frac{1}{2}$. Since $rn = \sqrt{\frac{n}{n-1}}$, we must set

$$\alpha := \frac{1}{2}\sqrt{\frac{n-1}{n}},$$

and for this α, it finally holds that:

$$\varphi(x(\alpha)) - \varphi(e) \leq -1 - \log\frac{1}{2} = -1 + \log 2 =: -\delta < -0.3.$$

Thus, the lemma holds.

□

Proof of Theorem 5.4. One has

$$\varphi(x^k) = n\log(c^T x^k) - \sum_{i=1}^{n} \log(x_i^k)$$

$$\varphi(x^0) - k\delta = n\log(c^T x^0) - \sum_{i=1}^{n} \log(x_i^0) - k\delta,$$

where $x_i^0 = \frac{1}{n}$ due to the initial choice $x^0 = e$. With Lemma 5.3, it holds that $\varphi(x^k) \leq \varphi(x^0) - k\delta$, so:

$$n\log\left(\frac{c^T x^k}{c^T x^0}\right) \leq \underbrace{\sum_{i=1}^{n} \log(x_i^k)}_{<0 \text{ because } x \in \mathring{\Delta}} \underbrace{- \sum_{i=1}^{n} \log\frac{1}{n}}_{=-n\log n} - k\delta \leq n\log n - k\delta.$$

Thus, it holds that:

$$\frac{c^T x^k}{c^T x^0} \leq e^{\log n - \frac{k}{n}\delta},$$

and the claim is shown if

$$e^{\log n - \frac{k}{n}\delta} \leq 2^{-q} = e^{q\log 2}.$$

Since $\delta > 0.3$ and $\log 2 \approx 0.69$, the condition

$$k \geq \frac{n}{\delta}(q\log 2 + \log n)$$

is sufficient to ensure $\frac{c^T x^k}{c^T x^0} \leq 2^{-q}$. □

Remark 5.18 – Practical choice of α

In practice, we do not choose our parameter α as given in the theorem, but there we minimize $\varphi(x(\alpha))$ under the constraints $\alpha > 0$ and $e - \alpha rd > 0$. It is sufficient to approximate the minimum. Typically, α is significantly closer to 1 than the α given in the theorem.

Remark 5.19 – Termination criterion

1. If $\varphi(x^k) \leq \varphi(x^0) - \epsilon$ for a given tolerance ϵ, then we terminate the algorithm and search for the nearest corner. With a simplex step, we check this corner for optimality. Otherwise, we may continue with the simplex algorithm, which then (for good q) does not need much more.
2. If $\varphi(x^{k+1}) > \varphi(x^k) - \delta$, then there is no solution (it can even be shown that if no solution exists, there can be such a k).

Bibliography

[1] Dantzig, Thapa, 2003, Linear Programming: 2

[2] Matoušek, Gärtner, 2007, Understanding and Using Linear Programming