Numerical Methods

ETH Zurich

Lan Zhang

FS25

Contents

Chapter 1

Numerics and Error Analysis

Definition 1.1 *For $\tilde{x} \in \mathbb{K}$ an approximation of $x \in \mathbb{K}$, we define the **absolute error** and **relative error** as*

$$\varepsilon_{abs} = |x - \tilde{x}|, \quad \varepsilon_{rel} = \frac{|x - \tilde{x}|}{|x|}.$$

Approximation $\tilde{x}$ of x has $l \in \mathbb{N}_0$ correct digits if $\varepsilon_{rel} \leq 10^{-l}$. Machine precision is the maximal relative error of rounding

$$EPS = \max_{x \in \mathbb{R}} \frac{|rd(x) - x|}{|x|}.$$

Cost of basic operations

operation	mult/div	add/sub	asympt. complexity
dot product	n	$n-1$	$\mathcal{O}(n)$
tensor product	nm	0	$\mathcal{O}(mn)$
matrix product	mnk	$mk(n-1)$	$\mathcal{O}(mnk)$

Example 1.2 (Polynomial Evaluation) *To calculate the value of an n-degree polynomial at a certain x, at least n multiplications are required, which can be achieved using Horner's rule.*

Example 1.3 (LGS) *To solve a system of n linear equations with a unique solution using **Cramer's Rule**, we compute:*

$$x_k = \frac{D_k}{D}, \quad k = 1, 2, \ldots, n \quad D = \begin{vmatrix} a_{11} & a_{12} & \cdots & a_{1n} \\ a_{21} & a_{22} & \cdots & a_{2n} \\ \vdots & \vdots & \ddots & \vdots \\ a_{n1} & a_{n2} & \cdots & a_{nn} \end{vmatrix} \quad \mathbf{b} = \begin{pmatrix} b_1 \\ b_2 \\ \vdots \\ b_n \end{pmatrix}$$

where D is the determinant of the coefficient matrix and D_k is obtained by replacing the k-th column of D with b. The total number of operations is $\approx (n+1) \cdot n! \cdot (n-1) + n$.

Definition 1.4 *Cancellation occurs when subtracting two almost equal numbers, which can lead to an amplification of the relative error.*

Example 1.5 (Roots of quadratic polynomial) *We want to stably compute the roots of a polynomial $p(\xi) = \xi^2 + \alpha\xi + \beta$.*

- *Step 1: Compute first root $\xi_1 = -\frac{\alpha}{2} - \frac{1}{2}\sqrt{\alpha^2 - 4\beta}$ (stable).*

- *Step 2: Use Vieta's formula to compute the second root* $\xi_2 = \frac{\beta}{\xi_1}$ *(stable).*

Definition 1.6 *The* ***condition number*** *of an algorithm* $F : X \to Y$ *is defined as*

$$c_F(x) = \sup_{\Delta x} \left(\frac{\|F(x+\Delta x) - F(x)\| / \|F(x)\|}{\|\Delta x\| / \|x\|} \right).$$

The condition number for a matrix A *is defined as*

$$c_A = \|A\|\|A^{-1}\| = \frac{\sigma_{max}}{\sigma_{min}}.$$

A problem is called ***well-conditioned*** *if its condition number is small, otherwise it is called* ***ill-conditioned****.*

1.1 Error Types

Definition 1.7 (Roundoff error) *caused by included terms*

Definition 1.8 (Truncation error) *caused by excluded terms*

Definition 1.9 (Significant digits) *If the error limit of the approximated value* x *is half of the unit* n *digits away from the first non-zero digit of* x*, then* x *has* n *significant digits.*

$$x^* = \pm 0.a_1 a_2 \dots a_n \cdot 10^m \to |x - x^*| \leq \frac{1}{2} 10^{m-n}$$

$$\varepsilon_r \leq \frac{1}{2a_1} \cdot 10^{-n+1}$$

Definition 1.10 (Error Estimation for Functions) *Let the error in the input* x *be* $e(x)$*, and the corresponding error in the output* $y = f(x)$ *be* $e(y)$*. Then approximately:*

$$|e(y)| \approx |f'(x)| \cdot |e(x)|$$

An error in the input x *is* ***amplified by a factor of*** $|f'(x)|$ *after being transformed by the function* f*. If* $|f'(x)| < 1$*, the error is reduced. Therefore, we call* $|f'(x)|$ *the* ***amplification factor*** *(also called the absolute condition number).*

The relative condition number at point x *is given by:*

$$e_r(y) = \left| \frac{x f'(x)}{f(x)} \right| e_r(x)$$

This expresses how much the ***relative error in the input*** *is amplified in the output. The function* f *is* ***well-conditioned/ill-conditioned*** *at that point, if the relative condition number is small/big at* x^**.*

1.1.1 Stability of numerical computation

An unstable algorithm is one where initial errors accumulate rapidly and increase progressively.

Chapter 2

Data Interpolation

2.1 Abstract Interpolation

Given a set of data points $(t_i, y_i) \in \mathbb{R}^2,\ i = 0, \ldots, n,\ t_i \in I \subset \mathbb{R}$. Find an interpolant, i.e. a function $f : I \to \mathbb{R}$, such that $f \in C^0(I)$ and $f(t_i) = y_i,\ \forall i = 0, \ldots, n$. There are infinitely many such functions, therefore we need additional assumptions on f such as smoothness properties. We typically search for $f \in S \subset C^0(I)$, where S is a $(n+1)$-dimensional subspace.

Basis representation $f(t) = \sum_{j=0}^{n} \alpha_j b_j(t)$ with interpolating condition $f(t_i) = \sum_{j=0}^{n} \alpha_j b_j(t_i) = y_i$ written as $(n+1) \times (m+1)$ linear system of equations

$$\mathbf{Ac} = \begin{pmatrix} b_0(t_0) & \ldots & b_m(t_0) \\ \vdots & \ddots & \vdots \\ b_0(t_n) & \ldots & b_m(t_n) \end{pmatrix} \begin{pmatrix} \alpha_0 \\ \vdots \\ \alpha_m \end{pmatrix} = \begin{pmatrix} y_0 \\ \vdots \\ y_n \end{pmatrix} = \mathbf{y},$$

where **A** is called the Vandermonde matrix.

Theorem 2.1 (Existence and Uniqueness) *of interpolant depends on regularity of* **A**, *we therefore require* $m = n$. *Then, invertibility of* **A** *depends on nodes* t_i *and space* S, *but not on the choice of basis* $\{b_i\}_{j=0}^{n}$. *Therefore, the interpolants also yield the same residual. Note that a cardinal basis yields* $\mathbf{A} = \mathbf{I}$.

Interpolation assumes sufficiently accurate measurements, otherwise we would use data fitting. Extrapolation estimates values outside the range of existing data (interpolation points).

Note: If we do not restrict the degree of the polynomial to exactly n, then the interpolation polynomial is not unique. For example, one can add any polynomial of degree greater than n that vanishes at the interpolation nodes, and the resulting function would still interpolate the same data points.

2.2 Global Polynomial Interpolation

Space of Polynomials of degree $\leq k$ P_k can be represented by a monomial basis, such that each polynomial can be written as a linear combination of monomials. This space is $k+1$ dimensional, polynomials of degree k are determined by $k+1$ points.

the Vandermonde matrix is defined as:

$$V(x_0, x_1, \ldots, x_n) = \begin{pmatrix} 1 & x_0 & x_0^2 & \cdots & x_0^n \\ 1 & x_1 & x_1^2 & \cdots & x_1^n \\ \vdots & \vdots & \vdots & \ddots & \vdots \\ 1 & x_n & x_n^2 & \cdots & x_n^n \end{pmatrix}$$

This matrix determines the existence and uniqueness of a solution to the interpolation problem.

Theorem: If the nodes $x_0, x_1, \ldots, x_n$ are distinct (i.e., $x_i \neq x_j$ for $i \neq j$), then:

$$\det V(x_0, x_1, \ldots, x_n) = \prod_{0 \leq j < i \leq n} (x_i - x_j) = \prod_{i=1}^{n} \prod_{j=0}^{i-1} (x_i - x_j) \neq 0$$

Hence, the interpolation system has a unique solution. (Note: A n-fold zero x^n is given by $f^{(k)}(x) = 0 \quad \forall k \in [0, n-1]$)

Advantages of using polynomials are

- differentiation and integration is easy to compute,
- approximation property of polynomials,
- efficient evaluation through Horner scheme with complexity $\mathcal{O}(k)$

$$t(\ldots(t(t(\alpha_k t + \alpha_{k-1}) + \alpha_{k-2}) + \cdots + \alpha_1) + \alpha_0.$$

2.2.1 Lagrange Polynomial

Definition 2.2 (Lagrange Basis) *We define the Lagrange polynomials as*

$$L_i(t) = \prod_{\substack{j=0 \\ j \neq i}}^{n} \frac{t - t_j}{t_i - t_j}.$$

Basic properties are $L_i \in P_n,\ L_i(t_j) = \delta_{i,j}$, linearly independent and form a cardinal basis.

Lagrange Interpolation

$$p(t) = \sum_{i=0}^{n} y_i L_i(t) \quad \sum_i L_i(t) = 1$$

Lemma 2.3 (Existence and uniqueness) *The general Lagrange polynomial interpolation problem admits a unique solution $p \in P_n$.*

Corollary 2.4 *The polynomial interpolation in the nodes $\mathcal{T} = \{t_j\}_{j=0}^n$ defines a linear operator*

$$I_{\mathcal{T}} : \mathbb{R}^{n+1} \to \mathcal{P}_n : (y_0, \ldots, y_n)^T \mapsto \textit{interpolating polynomial } p$$
$$\mathcal{T}_n = \{t_j^{(n)} = a + (b-a)\frac{j}{n}, j = 0, \ldots, n\} \subset I.$$

Complexity *Evaluating L_i is $\mathcal{O}(\mathbf{n})$ using the Horner scheme, evaluating $p(x)$ is $\mathcal{O}(\mathbf{n^2})$. Evaluating N different data value sets using a Lagrange basis approach is $\mathcal{O}(\mathbf{n^2 N})$, while evaluating N different data value sets using a monomial basis approach is $\mathcal{O}(\mathbf{n^3 N})$.*

Remark 2.5 *Lagrange interpolation is ill-conditioned for evenly spaced points mostly due to Runge's phenomenon: small changes in the data may cause huge changes in the interpolant.*

Interpolationsfehler Sei $f : I = [a, b] \to \mathbb{R}$ und $p\left[f \mid x_0, \ldots, x_n\right](x) \in \mathbb{P}_n$ das IP, welches die IB $p\left[f \mid x_0, \ldots, x_n\right]\left(x_j\right) = f\left(x_j\right)$ erfüllt. Für $f(n+1)$-mal stetig diffbar ($\Leftrightarrow f \in C^{n+1}[I]$): $\forall x \in I = [x_0, x_n] \quad \exists \xi = \xi(x) \in I$ mit der Fehlerfunktion $e(x)$, sodass:

$$e(x) = f(x) - p\left[f \mid x_0, \ldots, x_n\right](x) = \frac{f^{(n+1)}(\xi)}{(n+1)!} \prod_{j=0}^{n} (x - x_j)$$

The interpolation polynomial is exact for polynomials of degree $\leq n$. Oft interessieren wir uns nur für den grössten Fehler (anstelle des Fehlers vom IP an jeder Stelle x)

$$\begin{aligned} \|e\|_\infty = \max_{x \in I} |e(x)| &= \max_{x \in I} \left| \frac{f^{(n+1)}(\xi(x))}{(n+1)!} \cdot \prod_{j=0}^{n} (x - x_j) \right| \leqslant \max_{x \in I} \left| \frac{f^{(n+1)}(\xi(x))}{(n+1)!} \right| \cdot \max_{x \in I} \left| \prod_{j=0}^{n} (x - x_j) \right| \\ &= \frac{\left\| f^{(n+1)} \right\|_\infty}{(n+1)!} \cdot \max_{x \in I} \left| \prod_{j=0}^{n} (x - x_j) \right| \leqslant \frac{\left\| f^{(n+1)}(\xi(x)) \right\|_\infty}{(n+1)!} \cdot (b-a)^{n+1} \end{aligned}$$

Runge's phenomenon

Interpolating with high degree polynomials leads to oscillation/artifacts at the endpoints on equidistant nodes (due to the non-local non-zero property of lagrange polynomials). Runge's phenomenon may be avoided through more densely distributed points to the endpoints (Chebychev nodes) or piecewise polynomial interpolation.

2.2.2 Newton Polynomial

Construct an interpolant sequentially such that each time a new node is added, only one additional term needs to be appended:

$$N_n(x) = c_0 + c_1(x - x_0) + c_2(x - x_0)(x - x_1) + \cdots + c_n(x - x_0)(x - x_1) \cdots (x - x_{n-1})$$

The Newton interpolation polynomial takes the form:

$$P_n(x) = f[x_0] + f[x_0, x_1](x - x_0) + f[x_0, x_1, x_2](x - x_0)(x - x_1) + \cdots + f[x_0, \ldots, x_n](x - x_0) \cdots (x - x_{n-1})$$

Where each coefficient is a divided difference:

Definition 2.6 (Divided Differences) *The first-order divided difference is:*

$$f[x_i, x_j] = \frac{f(x_j) - f(x_i)}{x_j - x_i}, \quad x_i \neq x_j \quad f[x_i, x_j, x_k] = \frac{f[x_j, x_k] - f[x_i, x_j]}{x_k - x_i}$$

The general k-th order divided difference is:

$$f[x_0, x_1, \ldots, x_k] = \frac{f[x_1, \ldots, x_k] - f[x_0, \ldots, x_{k-1}]}{x_k - x_0}$$

Note:

- A k-th order divided difference must be formed from $k+1$ distinct nodes.
- Zero-order divided difference: $f[x_i] = f(x_i)$
- Divided differences are independent of the order of nodes and symmetric with respect to their arguments.
- **Connection to Derivatives:** If $f \in C^n[a,b]$, and $x_0, x_1, \ldots, x_n \in [a,b]$, then there exists $\xi \in [a,b]$ such that:
$$f[x_0, x_1, \ldots, x_n] = \frac{f^{(n)}(\xi)}{n!}$$
- **Divided Difference of a Polynomial:** If $f(x)$ is a polynomial of degree $\leq n$, then:
  - The k-th divided difference is a polynomial of degree $n-k$
  - The $(n+1)$-th order divided difference is identically zero

Divided Differences for Repeated Nodes

$$\lim_{x_1 \to x_0} f[x_0, x_1] = f'(x_0)$$
$$\lim_{x_2 \to x_0} f[x_0, x_1, x_2] = \frac{f''(x_0)}{2!}$$
$$\lim_{x_n \to x_0} f[x_0, x_1, \ldots, x_n] = \frac{f^{(n)}(x_0)}{n!}$$

In general, the n-th order divided difference for repeated nodes is:

$$f[x_0, x_0, \ldots, x_0] = \frac{f^{(n)}(x_0)}{n!}$$

Equally Spaced Nodes: Finite Difference Formulas

1. Forward Difference
$$\Delta f_i = f_{i+1} - f_i$$
2. Backward Difference
$$\nabla f_i = f_i - f_{i-1}$$
3. Central Difference
$$\delta f_i = \frac{f_{i+1} - f_{i-1}}{2}$$

2.2.3 Hermite Interpolation

Hermite interpolation not only matches function values at the nodes but also matches a number of derivatives.

We construct a polynomial $\varphi(x)$ such that:

$$\varphi(x_i) = f(x_i), \quad \varphi'(x_i) = f'(x_i), \quad \varphi^{(m_i)}(x_i) = f^{(m_i)}(x_i)$$

This determines a polynomial of degree $N-1$, where $N = \sum(m_i + 1)$.

Example 2.7 *Taylor Polynomial When interpolating at a single point x_0:*

$$\varphi(x) = f(x_0) + f'(x_0)(x - x_0) + \cdots + \frac{f^{(m_0)}(x_0)}{m_0!}(x - x_0)^{m_0}$$

The remainder term is:

$$R(x) = f(x) - \varphi(x) = \frac{f^{(m_0+1)}(\xi)}{(m_0+1)!}(x - x_0)^{m_0+1}, \quad \xi \in [a, b]$$

Definition 2.8 (Hermite Polynomial) *Given:*

$$x_0, x_1, \ldots, x_n \in [a, b], \quad f(x_i) = y_i, \quad f'(x_i) = y_i'$$

Construct a polynomial $H_{2n+1}(x)$ such that:

$$H_{2n+1}(x_i) = y_i, \quad H_{2n+1}'(x_i) = y_i'$$

Assume:

$$H_{2n+1}(x) = \sum_{i=0}^{n} \left[\alpha_i(x) y_i + \beta_i(x) y_i'\right]$$

Where:

$$\alpha_i(x_j) = \delta_{ij}, \quad \alpha_i'(x_j) = 0$$
$$\beta_i(x_j) = 0, \quad \beta_i'(x_j) = \delta_{ij}$$

To ensure this, let:

$$\alpha_i(x) = A_i(x)(1 - 2(x - x_i)l_i'(x_i))(l_i(x))^2 \beta_i(x) = B_i(x)(x - x_i)(l_i(x))^2$$

Definition 2.9 (Error Term) *Let $f \in C^{2n+2}[a, b]$, and $x_0 < x_1 < \cdots < x_n \in [a, b]$*

Then:

$$R(x) = f(x) - H_{2n+1}(x) = \frac{f^{(2n+2)}(\xi)}{(2n+2)!} \prod_{i=0}^{n} (x - x_i)^2$$

2.3 Piecewise Interpolation

2.3.1 Linear Interpolation

Connect data points (t_i, y_i) by line segments. Here is

$$S = \{f \in C^0(I) : f(t) = \beta_i t + \gamma_i \text{ on } [t_{i-1}, t_i],\ \beta_i, \gamma_i \in \mathbb{R},\ i = 0, \ldots, n\}$$

with $\dim(S) = n + 1$. *A basis for S are the hat functions b_i*

$$b_0 = \begin{cases} 1 - \frac{t - t_0}{t_1 - t_0} & \text{for } t_0 \le t < t_1, \\ 0 & \text{for } t \ge t_1. \end{cases}$$

$$b_j = \begin{cases} 1 - \frac{t_j - t}{t_j - t_{j-1}} & \text{for } t_{j-1} \le t < t_j, \\ 1 - \frac{t - t_j}{t_{j+1} - t_j} & \text{for } t_j \le t < t_{j+1}, \\ 0 & \text{for } t \ge t_1. \end{cases}$$

$$b_n = \begin{cases} 1 - \frac{t_n - t}{t_n - t_{n-1}} & \text{for } t_{n-1} \le t < t_n, \\ 0 & \text{for } t < t_{n-1}. \end{cases}$$

Therefore, $b_i(t_j) = \delta_{i,j}$. This basis is unique. Such a basis is called cardinal basis. Note that S and $\{b_j\}_{j=0}^n$ depend on points t_i. The interpolant then is given by

$$f(t) = \sum_{j=0}^{n} y_j b_j(t).$$

Given nodes $x_0 < x_1 < \cdots < x_n$, the function $f(x)$ is approximated on each subinterval $[x_k, x_{k+1}]$ by the linear interpolation polynomial:

$$I_h(x) = \frac{x - x_k}{h} f(x_{k+1}) + \frac{x_{k+1} - x}{h} f(x_k), \quad x \in [x_k, x_{k+1}]$$

where $h = x_{k+1} - x_k$ (equal spacing assumed).

Properties:

- The interpolated function $I_h(x)$ is continuous, but its derivative is generally discontinuous at the nodes. - If $f \in C^2[a, b]$, the error satisfies:

$$\max_{x \in [a,b]} |f(x) - I_h(x)| \leq \frac{h^2}{8} \max_{x \in [a,b]} |f''(x)|$$

Cubic Hermite Interpolation

If both function values $f(x_k)$ and derivative values $f'(x_k)$ are known at each node, we can construct a cubic Hermite interpolation polynomial $I_h(x)$ on each subinterval $[x_k, x_{k+1}]$, which ensures C^1 continuity (i.e., continuous function and first derivative).

$$I_h(x) = \sum_{k=0}^{n-1} \left[f(x_k)\alpha_k(x) + f'(x_k)\beta_k(x) + f(x_{k+1})\alpha_{k+1}(x) + f'(x_{k+1})\beta_{k+1}(x) \right]$$

Local Basis Functions: *Define a normalized variable on $[x_k, x_{k+1}]$:*

$$t = \frac{x - x_k}{h}, \quad 0 \leq t \leq 1$$

Then the basis functions are:

$$\begin{aligned} \alpha_k(x) &= 2t^3 - 3t^2 + 1 \\ \beta_k(x) &= (t^3 - 2t^2 + t)h \\ \alpha_{k+1}(x) &= -2t^3 + 3t^2 \\ \beta_{k+1}(x) &= (t^3 - t^2)h \end{aligned}$$

Error Estimate: *If $f \in C^4[a, b]$, then the interpolation error satisfies:*

$$\max_{x \in [a,b]} |f(x) - I_h(x)| \leq Ch^4 \max_{x \in [a,b]} |f^{(4)}(x)|$$

where C is a constant.

Local Support of Interpolation Basis Functions

Each basis function $\alpha_k(x), \beta_k(x)$ is nonzero only on its corresponding subinterval $[x_k, x_{k+1}]$. This locality of support is important for numerical stability.

Chapter 3

Function Approximation

Given a function f, find a "simple" approximation $\tilde{f}$, where simple means that $\tilde{f}$ is

- *encoded by small amount of information,*
- *easy to evaluate.*

Definition 3.1 (Weierstrass Approximation Theorem) *Let $f(x)$ be continuous on $[a,b]$. Then for any $\varepsilon > 0$, there exists a polynomial $P_m(x)$ such that:*

$$|f(x) - P_m(x)| < \varepsilon \quad \forall x \in [a,b]$$

Definition 3.2 (Inner Product Space) *Let $\rho(x) \geq 0$ on $[a,b]$. Define the inner product:*

$$\langle f, g \rangle = \int_a^b \rho(x) f(x) g(x)\, dx \quad \|f\| = \left(\int_a^b \rho(x) f^2(x)\, dx \right)^{1/2}$$

If $f(x), g(x)$ are square-integrable with respect to $\rho(x)$, then they belong to an inner product space over $[a,b]$.

Definition 3.3 (Inequalities in Inner Product Spaces)

- *Cauchy–Schwarz Inequality:* $|\langle f, g \rangle| \leq \|f\| \cdot \|g\|$
- *Triangle Inequality:* $\|f + g\| \leq \|f\| + \|g\|$
- *Parallelogram Law:* $\|f+g\|^2 + \|f-g\|^2 = 2\|f\|^2 + 2\|g\|^2$

Definition 3.4 (Linear Independence) *Let $\{\varphi_0(x), \varphi_1(x), \ldots, \varphi_n(x)\}$ be a set of functions.*

Define the Gram determinant:

$$G = \begin{vmatrix} \langle \varphi_0, \varphi_0 \rangle & \langle \varphi_0, \varphi_1 \rangle & \cdots & \langle \varphi_0, \varphi_n \rangle \\ \langle \varphi_1, \varphi_0 \rangle & \langle \varphi_1, \varphi_1 \rangle & \cdots & \langle \varphi_1, \varphi_n \rangle \\ \vdots & \vdots & \ddots & \vdots \\ \langle \varphi_n, \varphi_0 \rangle & \langle \varphi_n, \varphi_1 \rangle & \cdots & \langle \varphi_n, \varphi_n \rangle \end{vmatrix}$$

The functions are linearly independent on $[a,b]$ if and only if $G \neq 0$.

3.1 Least-Squares Approximation

Given a function $f(x)$, we seek an approximation $s(x) \in \Phi$, where $\Phi = span\{\varphi_0(x), \varphi_1(x), \ldots, \varphi_n(x)\}$, such that

$$\|f(x) - s(x)\|_2 = \min_{\varphi \in \Phi} \|f(x) - \varphi(x)\|_2$$

We assume the weight function $\rho(x) > 0$ on the interval $[a, b]$, and the norm is defined by:

$$\|f(x) - s(x)\|_2 = \left(\int_a^b \rho(x)[f(x) - s(x)]^2 dx \right)^{1/2}$$

Let the approximation function be:

$$s(x) = \sum_{k=0}^{n} a_k \varphi_k(x)$$

Normal Equations

To find the best approximation, we minimize:

$$\int_a^b \rho(x) \left[f(x) - \sum_{k=0}^{n} a_k \varphi_k(x) \right]^2 dx$$

Taking derivatives with respect to a_j, we obtain the normal equations:

$$\sum_{k=0}^{n} a_k \langle \varphi_k, \varphi_j \rangle = \langle f, \varphi_j \rangle \quad \text{for } j = 0, 1, \ldots, n$$

$$||\delta||^2 = ||f||_2^2 - \sum_{k=0}^{n} (a_k^* \varphi_k, f)$$

Matrix Form of Normal Equations

In matrix form, this system becomes:

$$\begin{bmatrix} \langle \varphi_0, \varphi_0 \rangle & \langle \varphi_0, \varphi_1 \rangle & \cdots & \langle \varphi_0, \varphi_n \rangle \\ \langle \varphi_1, \varphi_0 \rangle & \langle \varphi_1, \varphi_1 \rangle & \cdots & \langle \varphi_1, \varphi_n \rangle \\ \vdots & \vdots & \ddots & \vdots \\ \langle \varphi_n, \varphi_0 \rangle & \langle \varphi_n, \varphi_1 \rangle & \cdots & \langle \varphi_n, \varphi_n \rangle \end{bmatrix} \begin{bmatrix} a_0 \\ a_1 \\ \vdots \\ a_n \end{bmatrix} = \begin{bmatrix} \langle f, \varphi_0 \rangle \\ \langle f, \varphi_1 \rangle \\ \vdots \\ \langle f, \varphi_n \rangle \end{bmatrix}$$

This is a linear system with a unique solution if the basis functions $\{\varphi_k\}$ are linearly independent, because the Gram matrix on the left-hand side is then invertible.

The system of normal equations is only dependent upon the chosen basis functions and nodes. Observe the data points to determine the order of the approximation function. Higher order approximation generally leads to instability. If the basis is orthogonal, the system can be reduced to a diagonal matrix.

$$||\delta||^2 = ||f||_2^2 - \sum_{k=0}^{n} a_k^{*2}$$

Interpolation is done by first sampling the given function on some nodes t_i and then interpolating the resulting data set. The error is measured in L^p-norm as $\|f - \tilde{f}\|_p$.

$$f : I \subset \mathbb{R} \to \mathbb{K} \overset{sampling}{\longrightarrow} (t_i, y_i = f(t_i))_{i=0}^{m} \overset{interpolation}{\longrightarrow} \tilde{f} = I_{\mathcal{T}} y \ (\tilde{f}(t_i) = y_i)$$

3.2 Taylor Approximation

Taylor polynomials yield local approximation of sufficiently smooth functions. Any function $f \in C^k(I)$ can be approximated by a Taylor polynomial. Given $t_0 \in I$, there exists a function $h_k : \mathbb{R} \to \mathbb{R}$ such that

$$f(t) = \underbrace{\sum_{j=0}^{k} \frac{f^{(j)}(t_0)}{j!}(t-t_0)^j}_{=T_k(t)} + h_k(t)(t-t_0)^k$$

and $h_k(t) \to 0$ for $t \to t_0$. T_k approximates f in a (possibly small) neighbourhood $J \subset I$ of t_0. If $f \in C^{k+1}(I)$, we can quantify the error as

$$f(t) - T_k(t) = \frac{f^{(k+1)}(\xi)}{(k+1)!}(t-t_0)^{k+1}$$

for some point $\xi \in (\min(t,t_0), \max(t,t_0))$.

Taylor approximation is easy and direct, but inefficient (same accuracy often reached with lower degree polynomials). Access to higher order derivatives is required, which can be hard to obtain.

3.3 Lagrange Approximation

We first consider Lagrange interpolation with equidistant nodes.

Definition 3.5 *Given an interval $I \subset \mathbb{R}, n \in \mathbb{N}$, a node set $\mathcal{T} = \{t_0, \dots, t_n\}$, the **Lagrangian (interpolation polynomial) approximation scheme** $L_\mathcal{T} : C^0(I) \to P_n$ is defined by*

$$L_\mathcal{T}(f) = I_\mathcal{T}(y) \in P_n \quad \textit{with} \quad y = (f(t_0), \dots, f(f_n))^T \in \mathbb{K}^{n+1}.$$

Definition 3.6 *We define two types of convergence, namely*

***algebraic convergence** if $\|f - I_\mathcal{T} f\| = \mathcal{O}(n^{-p})$ for $n \to \infty$,*

***exponential convergence** if $\|f - I_\mathcal{T} f\| = \mathcal{O}(q^n)$ for $n \to \infty$.*

Theorem 3.7 (Representation of interpolation error) *We consider $f \in C^{n+1}(I)$ and the Lagrangian interpolation approximation scheme for a node set $\mathcal{T} = \{t_0, \dots, t_n\} \subset I$. Then, for every $t \in I$ there exists a $\mathcal{T}_t \in (min\{t, t_0, \dots, t_n\}, \max\{t, t_0, \dots, t_n\})$ such that*

$$f(t) - L_\mathcal{T}(f)(t) = \frac{f^{(n+1)}(\mathcal{T}_t)}{(n+1)!} \prod_{j=0}^{n} (t - t_j).$$

For $f \in C^{n+1}(I)$ and equidistant nodes $\mathcal{T} = \{t_0, \dots, t_n\} \subset I$ using Lagrangian interpolation, this results in the following estimates

$$\|f - L_\mathcal{T} f\|_{L^\infty(I)} \leq \frac{\|f^{(n+1)}\|_{L^\infty(I)}}{(n+1)!} \max_{t \in I} |(t-t_0)\dots(t-t_n)| \tag{3.1}$$

3.4 Piecewise Polynomial Lagrange Interpolation

General local Lagrange interpolation on a mesh $\mathcal{M} = \{a = x_0 < x_1 < \ldots < x_m = b\}$.

1. *Choose local degree $n_j \in \mathbb{N}_0$ for each cell of the mesh $j = 1, \ldots, m$.*
2. *Choose set of local interpolation nodes*

$$T^j = \{t_0^j, \ldots, t_{n_j}^j\} \subset I_j = [x_{j-1}, x_j], \quad j = 1, \ldots, m$$

   for each mesh cell/grid interval I_j. The size of every node set is $n_j + 1$.
3. *Define piecewise polynomial interpolant $s : [x_0, x_m] \to \mathbb{K}$*

$$s_j = s|_{I_j} \in \mathcal{P}_{n_j} \quad \textit{and} \quad s_j(t_i^j) = f(t_i^j) \quad i = 0, \ldots, n_j,\ j = 1, \ldots, m. \tag{3.2}$$

Corollary 3.8 (Continuous local Lagrange Interpolants) *If the local degrees n_j are at least 1 and the interpolation nodes t_k^j, $j = 1, \ldots, m$, $k = 0, \ldots, n_j$, for local Lagrange interpolation satisfy*

$$t_{n_j}^j = t_0^{j+1}\ \forall j = 1, \ldots, m-1 \implies s \in C^0([a, b]),$$

then the piecewise polynomial Lagrange interpolant according to 3.2 is continuous on $[a, b]$, i.e. $s \in C^0([a, b])$.

Error estimate

Derivation of error estimate by decreasing the mesh width $h_{\mathcal{M}}$ (h-convergence) considering the special case of fixed number of nodes per grid $n_j = n$. Number of nodes $\leq \frac{|b-a|}{h_{\mathcal{M}}} \leq \frac{|b-a|(n-1)}{h_{\mathcal{M}}}$.

Applying the old estimate 3.1 on each subinterval gives the overall estimate with algebraic convergence of rate $n + 1$

$$\|f - s\|_{L^\infty([x_0, x_m])} \leq h_{\mathcal{M}}^{n+1} \frac{1}{(n+1)!} \|f^{(n+1)}\|_{L^\infty([x_0, x_m])}$$

3.5 Orthogonal Polynomials

Let $\rho(x)$ be a weight function defined on $[a, b]$, and let $\{g_0(x), g_1(x), \ldots, g_n(x)\}$ be a sequence of polynomials such that:

$$\int_a^b g_i(x) g_j(x) \rho(x)\, dx = \begin{cases} A_k > 0, & \textit{if } i = j = k \\ 0, & \textit{if } i \neq j \end{cases}$$

The necessary and sufficient condition for a degree-n polynomial $P_n(x)$ to be orthogonal to all polynomials of degree less than n is:

$$\int_a^b g_i(x) P_n(x) \rho(x)\, dx = 0, \quad i = 0, 1, \ldots, n-1$$

Properties of Orthogonal Polynomials

Let $\{g_k(x)\}$ be a sequence of orthogonal polynomials on $[a, b]$ with respect to $\rho(x)$. Then:

- *Property 1: The polynomials $\{g_k(x)\}$ are linearly independent.*

- *Property 2: Each $g_k(x)$ has k real, distinct (simple) roots inside (a, b).*
- *Property 3: The orthogonal polynomials with leading coefficient equal to 1 satisfy a three-term recurrence relation:*

$$g_{k+1}(x) = (x - a_{k+1})g_k(x) - b_k g_{k-1}(x) \quad a_{k+1} = \frac{(xg_k, g_k)}{(g_k, g_k)}, b_0 = 0, b_k = \frac{(g_k, g_k)}{(g_{k-1}, g_{k-1})}$$

Expansion in Terms of Orthogonal Polynomials

Any polynomial $Q_n(x)$ of degree n can be expanded in terms of orthogonal polynomials $\{\tilde{P}_k(x)\}$:

$$Q_n(x) = \sum_{k=0}^{n} \alpha_k \tilde{P}_k(x),$$

where coefficients α_k can be determined by inner product projection:

$$\alpha_k = \frac{\langle Q_n(x), \tilde{P}_k(x)\rangle}{\langle \tilde{P}_k(x), \tilde{P}_k(x)\rangle} = \frac{2k+1}{2} \int_{-1}^{1} Q(x)\tilde{P}_k(x)dx.$$

Then the least-square approximation error is given by

$$\|\delta_n\|_2^2 = \int_{-1}^{1} f(x)^2 dx - \sum_{k=0}^{n} \frac{2}{2k+1}\alpha_k^2$$

3.5.1 Legendre Polynomials

When $[a, b] = [-1, 1]$ and $\rho(x) \equiv 1$, the orthogonal polynomials obtained from orthogonalizing $\{1, x, x^2, \ldots, x^n\}$ are called Legendre polynomials, denoted by $P_n(x)$.

$$P_0(x) = 1, \quad P_1(x) = x,$$

$$P_n(x) = \frac{1}{2^n n!} \frac{d^n}{dx^n}[(x^2 - 1)^n],$$

which ensures $P_n(x)$ is of degree n.

The leading coefficient of $P_n(x)$ is:

$$\frac{(2n)!}{2^n (n!)^2} \quad \tilde{P}_n(x) = \frac{n!}{(2n)!} P_n(x)$$

Properties of Legendre Polynomials

- ***Orthogonality:***

$$\int_{-1}^{1} P_n(x)P_m(x)\, dx = \begin{cases} 0, & n \neq m, \\ \frac{2}{2n+1}, & n = m. \end{cases}$$

- ***Parity:***

$$P_n(-x) = (-1)^n P_n(x).$$

- ***Least-Squares Property:*** *Among all monic polynomials of degree n, $P_n(x)$ minimizes the weighted L^2-norm:*

$$\int_{-1}^{1} [f(x) - P_n(x)]^2\, dx.$$

- ***Recurrence Relation:***

$$(n+1)P_{n+1}(x) = (2n+1)xP_n(x) - nP_{n-1}(x), \quad \textit{for } n \geq 1.$$

- ***Special Values:***

$$P_n(1) = 1, \quad P_n(-1) = (-1)^n, \quad P_n'(1) = \frac{n(n+1)}{2}.$$

Examples:

$$P_2(x) = \frac{1}{2}(3x^2 - 1), \quad P_3(x) = \frac{1}{2}(5x^3 - 3x), \quad P_4(x) = \frac{1}{8}(35x^4 - 30x^2 + 3).$$

3.5.2 Chebyshev Polynomials

On $[-1, 1]$ *with weight function* $\rho(x) = \frac{1}{\sqrt{1-x^2}}$*, the orthogonal polynomials obtained are Chebyshev polynomials:*

$$T_n(x) = \cos(n \arccos x), \quad |x| \leq 1$$
$$x = \cos(\theta) \rightarrow T_n(x) = \cos(n\theta), 0 \leq \theta \leq \pi$$

- *Orthogonality:*

$$\int_{-1}^{1} \frac{T_n(x)T_m(x)}{\sqrt{1-x^2}} dx = \begin{cases} 0, & n \neq m \\ \pi, & n = m = 0 \\ \frac{\pi}{2}, & n = m \neq 0 \end{cases}$$

- *Recurrence:*

$$T_0(x) = 1, \quad T_1(x) = x, \quad T_{n+1}(x) = 2xT_n(x) - T_{n-1}(x)$$

- *Even (odd) Chebyshev polynomials contain only even (odd) powers of x.*
- *The n zeros of* $T_n(x)$ *on* $[-1, 1]$ *are at:* $x_k = \cos \frac{2k+1}{2n}\pi \quad k = 0, \ldots, n-1$

Examples:

$$T_2(x) = 2x^2 - 1$$
$$T_3(x) = 4x^3 - 3x$$
$$T_4(x) = 8x^4 - 8x^2 + 1$$
$$T_5(x) = 16x^5 - 20x^3 + 5x$$

To approximate a function $f(x)$ *using a polynomial* $P(x)$ *such that the error is within a specified tolerance TOL, using orthogonal polynomials and the least squares method.*

3.5.3 Orthogonal Polynomials Approximation

1. *Set* $\varphi_0(x) \equiv 1$*, compute*

$$a_0 = \frac{(\varphi_0, y)}{(\varphi_0, \varphi_0)}, \quad P(x) = a_0\varphi_0(x), \quad err = (y, y) - a_0(\varphi_0, y)$$

2. *Set*

$$\alpha_1 = \frac{(x\varphi_0, \varphi_0)}{(\varphi_0, \varphi_0)}, \quad \varphi_1(x) = (x - \alpha_1)\varphi_0(x)$$

$$a_1 = \frac{(\varphi_1, y)}{(\varphi_1, \varphi_1)}, \quad P(x) \mathrel{+}= a_1\varphi_1(x), \quad err \mathrel{-}= a_1(\varphi_1, y)$$

3. *Set* $k = 1$
4. *While* $k < Max\_n$ *and* $|err| \geq TOL$, *do:*
   a) *Increment* k
   b) *Compute:*

$$\alpha_k = \frac{(x\varphi_{k-1}, \varphi_{k-1})}{(\varphi_{k-1}, \varphi_{k-1})}, \quad \beta_{k-1} = \frac{(\varphi_{k-1}, \varphi_{k-1})}{(\varphi_{k-2}, \varphi_{k-2})}$$

$$\varphi_k(x) = (x - \alpha_k)\varphi_{k-1}(x) - \beta_{k-1}\varphi_{k-2}(x)$$

$$a_k = \frac{(\varphi_k, y)}{(\varphi_k, \varphi_k)}, \quad P(x) \mathrel{+}= a_k\varphi_k(x), \quad err \mathrel{-}= a_k(\varphi_k, y)$$

5. *Output coefficients* $a_0, a_1, \ldots, a_k$ *and stop.*

Error Expression

$$err = \|y - P\|^2 = \sum_{i=1}^{m} w_i \left(y_i - P(x_i)\right)^2 = (y, y) - \sum_{k=0}^{n} a_k(\varphi_k, y)$$

3.5.4 Discrete Least Squares Approximation

Given data points (x_i, y_i), $i = 1, 2, \ldots, N$, *where* y_i *are noisy observations of* $f(x_i)$, *the goal is to find a polynomial* $P(x)$ *such that:*

$$\sum_{i=1}^{N} [P(x_i) - y_i]^2 \quad \textit{is minimized}$$

Choose basis functions $\{\varphi_0(x), \varphi_1(x), \ldots, \varphi_m(x)\}$ *that are linearly independent and span the approximation space. Let:*

$$P^*(x) = \sum_{k=0}^{m} a_k\varphi_k(x)$$

Minimize:

$$I(a_0, \ldots, a_m) = \sum_{i=1}^{N} \omega(x_i)[f(x_i) - P^*(x_i)]^2$$

This leads to solving the normal equations:

$$\sum_{i=1}^{N} \omega(x_i)\varphi_j(x_i) \left(f(x_i) - \sum_{k=0}^{m} a_k\varphi_k(x_i) \right) = 0, \quad j = 0, 1, \ldots, m$$

or

$$\alpha_0(\varphi_j, \varphi_0) + \alpha_1(\varphi_j, \varphi_1) + \ldots + \alpha_m(\varphi_j, \varphi_m) = (\varphi_j, y), \quad j = 0, 1, \ldots, m$$

Matrix form:

$$A^T W A\alpha = A^T W Y \quad A \in \mathbb{R}^{N\times(m+1)}$$

Where:

$$A = \begin{bmatrix} \varphi_0(x_1) & \varphi_1(x_1) & \cdots & \varphi_m(x_1) \\ \vdots & \vdots & & \vdots \\ \varphi_0(x_N) & \varphi_1(x_N) & \cdots & \varphi_m(x_N) \end{bmatrix}, \quad Y = \begin{bmatrix} y_1 \\ \vdots \\ y_N \end{bmatrix}$$

Then, the coefficients $\alpha = [a_0, a_1, \ldots, a_m]^T$ are the regression coeffcients to the least squares problem, and:

$$P^*(x) = \sum_{k=0}^{m} a_k \varphi_k(x)$$

The least squares error is:

$$\delta^2 = \|Y - A\alpha\|^2 = Y^T(Y - A\alpha) = \|y\|_2^2 - \sum_{j=0}^{m} \alpha_j^*(\varphi_j, y)$$

Linear Approximation ?

Assume $P(x) = \frac{x}{ax+b}$. Minimize:

$$\sum_{i=1}^{m} (y_i - (ax_i + b))^2$$

Linearize: Let $X = x$, $Y = y$, solve $Y \approx aX + b$

Exponential Approximation

Assume $P(x) = ae^{-b/x}$ where $a > 0, b > 0$

Linearize by taking logarithms:

$$\ln y \approx \ln a - \frac{b}{x} \Rightarrow Y = \ln y, X = \frac{1}{x}, A = \ln a, B = -b \Rightarrow Y \approx A + BX$$

3.5.5 Uniform Approximation (Minimax)

To minimize:

$$\|f - P_n\|_\infty = \max_{x\in[a,b]} |f(x) - P_n(x)|$$

The optimal uniform approximating polynomial (OUAP) exists and is unique if there are $n + 2$ alternating extreme deviation points:

$$|f(t_k) - P_n(t_k)| = \pm\|f - P_n\|_\infty, \quad a \le t_1 < \cdots < t_{n+2} \le b$$

Polynomial Degree Reduction (Economization)

Given $f(x) \approx P_n(x)$, drop the highest-degree term to form $P_{n-1}(x)$, aiming to minimize the increase in approximation error:

$$\|f - P_{n-1}\|_\infty \le \|f - P_n\|_\infty + \max_{x\in[-1,1]} |P_n(x) - P_{n-1}(x)|$$

Let a_n be the coefficient of the highest degree term in P_n, and choose it to minimize the loss of accuracy.

Chapter 4

Numerical Quadrature

Approximate $I = \int_a^b f(t)\mathrm{d}t$ *using only point evaluations of* f.

Definition 4.1 (Newton-Leibnitz) $\int_a^b f(x)dx = F(b) - F(a)$

4.1 Quadrature Formulas

Definition 4.2 *An n-point* ***quadrature formula/rule*** *on* $[a,b]$ *provides an approximation of the value of an integral through a weighted sum of point values of the integrand*

$$\int_b^a f(x)\mathrm{d}x \approx Q_n(f,a,b) = \sum_{i=1}^n w_i^n f(c_i^n) \qquad \sum_i \frac{w_i}{(b-a)} = 1 \quad c_i \in [a,b] \to 2n \text{ Freiheitsgrade}$$

Cost of evaluation of $Q_n(f)$*: n point evaluations of f and n multiplications and additions.*

Reference Intervals

$g : [a,b] \to [c,d]$ $g(t) = \alpha t + \beta$ $g(a) = c$ $g(b) = d$ $g(t) = c + \frac{(d-c)}{(b-a)}(t-a)$

$I_{ref} = [-1,1] \to I = [a,b] \quad \int_b^a f(x)\mathrm{d}x = \frac{1}{2}(b-a)\int_{-1}^1 \hat{f}(x)\mathrm{d}x \quad \hat{f}(x) = f\left(\frac{1}{2}(a+b) + \frac{1}{2}(b-a)x\right)$

$I_{ref} = [0,1] \to I = [a,b] \quad \int_b^a f(x)\mathrm{d}x = (b-a)\int_0^1 \hat{f}(x)\mathrm{d}x \quad \hat{f}(x) = f\left(a + (b-a)x\right)$

where $\hat{f}$ *is the affine pullback* $\Phi^* f$ *of f to the reference interval.*

Quadrature by Approximation Schemes

For given linear interpolation scheme $I_{\mathcal{T}}$ *with node set* $\mathcal{T} = \{c_1, \ldots, c_n\} \subset [a,b]$*, we can approximate*

$$\int_a^b f(t)\mathrm{d}t \approx \int_a^b I_{\mathcal{T}}(f(c_1), \ldots, f(c_n))^T(x)\mathrm{d}t = \sum_{j=1}^n f(c_j)\omega_j^n$$

with $\omega_j^n = \int_a^b I_{\mathcal{T}}(e_j)(x)\mathrm{d}x$.

We define the quadrature error as $E_n(f) = |\int_a^b f(x)\mathrm{d}x - Q_n(f)|$*,* $E(n) \to 0$ *for* $n \to \infty$*. We get*

$$E_n(f) \leq |b-a| \underbrace{\|f - I_{\mathcal{T}}(f(c_1), \ldots, f(c_n))^T\|_{L^\infty([a,b])}}_{\text{interpolation error}} = \int_a^b \frac{f^{(n)}(\xi_x)}{n!}\prod_{j=1}^n (x - c_j)dx$$

Quality measure for QF

Definition 4.3 *The order of a quadrature rule $Q_n : [a,b] \to \mathbb{R}$ is defined as*

$$\text{order}(\text{Q}_\text{n}) = \max\{m \in \mathbb{N}_0 : Q_n(p) = \int_a^b p(t)\text{d}t \quad \forall p \in P_m\} + 1,$$

that is the maximum degree $+1$ of polynomials for which the quadrature rule is guaranteed to be exact.

Note that the order is invariant under affine transformations. Polynomial QF with n points is exact for $p \in P_{n-1}$, which is of order $\leq n$.

Theorem 4.4 *An n-point quadrature rule on $[a,b]$ $Q_n(f) = \sum_{j=1}^n \omega_j f(c_j)$, $f \in C^0([a,b])$ with nodes $c_j \in [a,b]$ and weights $\omega_j \in \mathbb{R}, j = 1,\dots,n$ is of order $\geq n$ if and only if*

$$\omega_j = \int_a^b L_j(t)\text{d}t, \quad j = 1,\dots,n \quad L_0^0 = 1$$

with L_j the j-th Lagrange polynomial associated with the ordered node set $\{c_1,\dots,c_n\}$

This means for QF to have order $\leq n$, weights ω_j only depend on node set $\mathcal{T} = \{c_1,\dots,c_n\}$.

4.2 Polynomial Quadrature Formulas

QF induced by Lagrange interpolation scheme $I_\mathcal{T}$.

For given Lagrange interpolation scheme $I_\mathcal{T}$ with node set $\mathcal{T} = \{t_0,\dots,t_n\} \subset [a,b]$, we can approximate the integral Q_{n+1} by approximating f with a polynomial p_n.

$$\int_a^b f(t)\text{d}t \approx \int_a^b p_n(t)\text{d}t = \sum_{j=1}^n f(t_j)\omega_j^n$$

with $\omega_j^n = \int_a^b L_j(t)\text{d}t$.

$$\left|\int_a^b f(x)dx - \int_a^b p_n(x)dx\right| \leq \frac{1}{n!}\left(\frac{b^{n+1}-a^{n+1}}{n+1} - \sum_{j=1}^n w_j x_j^n\right) \approx \frac{1}{n!}(b-a)^{n+1}\max\left|f^{(n)}(z)\right|$$

A quadrature formula has order n+1 if it integrates polynomials of degree n exactly, i.e. $I(x^n) = Q(x^n)$. A quadrature formula on [-1,1] with nodes $c_1,...,c_n$ and weights $w_1,...,w_n$ is symmetrical if: $w_i = w_{n+1-i}$ and $c_i = -c_{n+1-i}$. The order of a symmetrical quadrature formula is even.

Example 4.5 (Newton-Cotes formulas) *Lagrange interpolation with equidistant nodes $t_j = a + \frac{b-a}{n-1}j$ with $j = 0,\dots,n-1$. Letting the substitution $x = a + th$, the Cotes coefficients are given by:*

$$A_k = h\int_0^n \prod_{\substack{j=0\\ j\neq k}}^n \frac{t-j}{k-j}\,dt = h\cdot\underbrace{\frac{(-1)^{n-k}}{k!(n-k)!}\int_0^n \prod_{\substack{j=0\\ j\neq k}}^n (t-j)\,dt}_{C_k^{(n)}}.$$

- $n = 1$ *(Midpoint rule)* $\int_a^b f(t)\text{d}t \approx (b-a)f(\frac{1}{2}(a+b)) \qquad E = \frac{1}{24}(b-a)^3|f(\xi)^{(2)}|$
- $n = 2$ *(Trapezoidal rule)* $\int_a^b f(t)\text{d}t \approx \frac{(b-a)}{2}(f(a)+f(b)) \qquad E = \frac{1}{12}(b-a)^3|f(\xi)^{(2)}|$
- $n = 3$ *(Simpson rule)* $\int_a^b f(t)\text{d}t \approx \frac{(b-a)}{6}(f(a)+4f(\frac{a+b}{2})+f(b)) \qquad E = \frac{1}{90}(\frac{b-a}{2})^5|f(\xi)^{(4)}|$

The order is at least as large as the number of interpolation points (even: n, odd: n+1). For $n > 6$ negative weights occur, which makes the method unstable.

4.3 Gauss Quadrature

For an interpolatory quadrature formula with $n+1$ nodes, the algebraic degree of precision is at least n, and at most $2n+1$.
Non-equidistant interpolation points. Choose n weights and n nodes such that E(n) = 0 for polynomials of degree 2n - 1 (order 2n). Increase accuracy without interval subdivision.

Theorem 4.6 *The maximal order of an n-point quadrature is $2n$.*

If we can find a family Q_n of QFs such that Q_n is n point and of order $2n$, we must have

- $\bar{P}_n = (x - c_1^n) \dots (x - c_n^n)$, $\bar{P}_n \in P_n$
- $\bar{P}_n \perp P_{n-1}$ *in* $L^2([-1,1])$
- $\bar{P}_n$ *is unique*
- *For* $\bar{P}_n = x^n + \alpha_{n-1}x^{n-1} + \ldots + \alpha_1 x + \alpha_0$, *we have* $\sum_{j=0}^{n-1} \alpha_j \int_{-1}^{1} x^l x^j \mathrm{d}x = -\int_{-1}^{1} x^l x^n \mathrm{d}t \iff A\alpha = b$, *where A is symmetric and positive definite*
- α *exists and is unique*

Theorem 4.7 *Let $\{\bar{P}_n\}_{n\in\mathbb{N}_0}$ be a family of non-zero orthogonal polynomials that satisfies*

- $\bar{P}_n \in P_n$
- $\int_{-1}^{1} q(t)\bar{P}_n(t)\mathrm{d}t = 0$ *for all* $q \in P_{n-1}$ *(L^2-orthogonality)*
- *The set* $\{c_j^n\}_{j=1}^m, m \le n$, *of real zeros of* $\bar{P}_n$ *contained in* $[-1,1]$

Then the quadrature rule $Q_n(f) = \sum_{j=1}^m \omega_j^n f(c_j^n)$ with weights $\omega_j = \int_a^b L_j(t)\mathrm{d}t, \quad j = 1, \dots, n$, provides a quadrature formula of order $2n$ on $[-1,1]$. If not open-ended and 1/2 interval ends are included, the order is 2n-1/2 (Radau/Lobatto).

Quadrature formulas with n points of order $2n$ are unique.

Quadratur	Intervall	Gewichtsfunktion	Polynom	Not.	`scpy.special.`
Gauss	$(-1,1)$	1	Legendre	P_k	`roots_legendre`
Chebyschev I	$(-1,1)$	$\frac{1}{\sqrt{1-x^2}}$	Chebyschev I	T_k	`roots_chebyt`
Chebyschev II	$(-1,1)$	$\sqrt{1-x^2}$	Chebyschev II	U_k	`roots_chebyu`
Jacobi $\alpha, \beta > 1$	$(-1,1)$	$(1-x)^\alpha(1+x)^\beta$	Jacobi	$P_k^{(\alpha,\beta)}$	`roots_jacobi`
Hermite	$\mathbb{R}$	e^{-x^2}	Hermite	H_k	`roots_hermite`
Laguerre	$(0,\infty)$	$x^\alpha e^{-x}$	Laguerre	L_k	`roots_genlaguerre`

Figure 4.1: Weight functions for quadratures

Definition 4.8 *The n-point Quadrature formulas whose nodes, the **Gaussian points**, are given by the zeros of the Legendre polynomial, and whose weights are chosen according to Theorem 4.4, are called **Gauss-Legendre quadrature formulas**.*

Lemma 4.9 *The weights of the Gauss-Legrendre quadrature formulas are positive.*

Recursive formula for Legrendre polynomials

$$P_0(t) = 1, \quad P_1(t) = t, \quad P_{n+1}(t) = \frac{2n+1}{n+1}tP_n(t) - \frac{n}{n+1}P_{n-1}(t) \quad w_j = \frac{2 - \left(1 - x_j^2\right)}{\left((n+1)\cdot P_n\left(x_j\right)\right)^2}$$

Examples on $[-1,1]$:

$$Q_2 = f\left(\frac{1}{\sqrt{3}}\right) + f\left(-\frac{1}{\sqrt{3}}\right) \quad Q_3 = \frac{5}{9}f\left(-\frac{\sqrt{15}}{5}\right) + \frac{8}{9}f(0) + \frac{5}{9}f\left(\frac{\sqrt{15}}{5}\right)$$

Definition 4.10 *Lobatto quadrature (on [0, 1], symmetrical): Gauss quadrature with interval end points as nodes (n nodes, n weights 2n-2 order)*

Definition 4.11 (Gauss-Chebyschev)

$$\int_{-1}^{1} f(x)\,dx \approx \sum_{k=1}^{n} f\left(\cos\left(\frac{2k-1}{2n}\pi\right)\right) \cdot w_k$$

Quadrature error & best approximation error

Theorem 4.12 *For every n-point quadrature rule* $Q_n = \sum_{j=1}^{n} w_j^n f(c_j^n)$ *of order* $q \in \mathbb{N}$ *with weights* $\omega_i \geq 0$*, the quadrature error satisfies*

$$E_n(f) \leq 2|b-a| \underbrace{\inf_{p \in P_{q-1}} \|f-p\|_{L^\infty([a,b])}}_{\text{best approximation error}} \quad \forall f \in C^0([a,b]).$$

Lemma 4.13 *For every n-point quadrature rule* $Q_n = \sum_{j=1}^{n} w_j^n f(c_j^n)$ *of order* $q \in \mathbb{N}$ *with weights* $\omega_i \geq 0$*, the quadrature error* $E_n(f)$ *for integrand* $f \in C^r([a,b])$*,* $r \in \mathbb{N}_0$ *satisfies*

- *if* $q \geq r$*:* $E_n(f) \leq Cq^{-r}|b-a|^{r+1}\|f^{(r)}\|_{L^\infty([a,b])}$ *(algebraic convergence),*
- *if* $q < r$*:* $E_n(f) \leq \frac{|b-a|^{q+1}}{q!}\|f^{(q)}\|_{L^\infty([a,b])}$ *(exponential convergence),*

with a constant $C > 0$ *independent of* n, f *and* $[a,b]$*.*

$f \in C^r([a,b])$*: algebraic convergence with rate r (log-log plot)*
$E_n = O(\frac{1}{n^r})$ *with* $r > 0$ $E = cn^{-r} \to \log(E) = \log(c) - r\log(n)$
$f \in C^\infty([a,b])$*: exponential convergence (lin-log plot))*
$E_n = O(\lambda^n)$ *with* $0 \leq \lambda < 1$ $E = c\lambda^n \to \log(E) = \log(c) + n\log(\lambda)$

Substitution might change smoothness of function on $[a,b]$*.*

We can approximate sharp algebraic convergence as $E_n(f) = \Theta(n^{-r}) \approx C_1 n^{-r}$*, and sharp exponential convergence as* $E_n(f) = \Theta(\lambda^n) \approx C_2\lambda^n$*, for some constant* $C_1, C_2 > 0$ *independent of* n*.*

To reduce quadrature error of algebraic convergence by factor $\rho > 1$*, we must increase the number of nodes by*

$$n_{new} = n_{old}\,\rho^{1/r}.$$

For higher r, we need less additional nodes to increase accuracy.

To reduce quadrature error of exponential convergence by factor $\rho > 1$*, we must increase the number of nodes by*

$$n_{new} = n_{old} + \left\lceil \left| \frac{\log(\rho)}{\log(\lambda)} \right| \right\rceil,$$

so the additional number of nodes is independent of the already added nodes.

4.4 Composite Quadrature

High-order interpolation has the Runge phenomenon, so piecewise low-order interpolation is used.

Divide interval with a mesh $M = \{a = x_0 < x_1 < \ldots < x_m = b\}$ *and apply QF on each cell* $I_j = [x_{j-1}, x_j]$ *to get QFs* $Q^j_{n_j}$

$$\int_a^b f(t)\mathrm{d}t = \sum_{j=1}^{m} \int_{x_{j-1}}^{x_j} f(t)\mathrm{d}t \quad x_j = a + jh \quad h = \frac{(b-a)}{N}$$

The error estimate for composite quadrature is

$$E(n) = \left| \sum_{j=1}^{m} \int_{x_{j-1}}^{x_j} f(t)\mathrm{d}t - Q^j_{n_j}(f) \right| \leq C\, h_M^s |b-a| \max_{j=1,\ldots,m} \|f^{(s)}\|_{L^\infty(I_j)} \leq C(b-a)\frac{h^q}{q!}.$$

for $s = \min\{r, q\}$. *Algebraic convergence in the mesh width* h_M *"h-convergence". For large r of rate q, for small r of rate r.*

Example 4.14 (Composite QF)

- *CMR:* $I(f,a,b) \approx \sum_{i=0}^{N-1} hf(\frac{x_i+x_{i+1}}{2})$
- *CTR:* $I(f,a,b) \approx \sum_{i=0}^{N-1} \frac{h}{2}(f(x_i)+f(x_{i+1})) = \frac{h}{2}(f(a) + 2\sum_{i=1}^{N-1} f(x_i) + f(b))$
- *CSR:* $I(f,a,b) \approx \sum_{i=0}^{N-1} \frac{h}{6}(f(x_i) + 4f(\frac{x_i+x_{i+1}}{2}) + f(x_{i+1})) = \frac{h}{6}(f(a) + 2\sum_{i=1}^{N-1} f(x_i) + 4\sum_{i=1}^{N} f(\frac{x_i+x_{i-1}}{2}) + f(b))$

For $f \in C^r$

- *composite QF is* $\mathcal{O}(n^{-\min\{q,r\}})$,
- *Gauss QF is* $\mathcal{O}(n^{-r})$.

Therefore, Gauss is at least as good as composite QF.

For $f \in C^\infty$

- *composite QF is* $\mathcal{O}(n^{-q})$ *(algebraic convergence),*
- *Gauss QF is* $\mathcal{O}(\lambda^n)$ *(exponential convergence).*

Therefore, use composite QF only if function is not overally smooth.

4.4.1 Romberg Algorithm

Divide the integration interval $[a,b]$ *into n equal parts, resulting in* $n+1$ *nodes. An approximate value is calculated using the trapezoidal rule.*

If the integration interval is subdivided again, the number of points increases to $2n+1$, *of which* $n+1$ *are old points and n are new points.*

To avoid redundancy in computation, the formula is modified as follows:

$$T_{2n} = \frac{1}{2}T_n + \frac{b-a}{2n}\sum_{k=1}^{n} f\left(a + \frac{2k-1}{2n}(b-a)\right)$$

Note that the new nodes are given by:

$$x_k = a + \frac{2k-1}{2n}(b-a), \quad k = 1, 2, \ldots, n$$

Recursive Richardson extrapolation is used:

$$S_n = T_{2n} + \frac{1}{4}(T_{2n} - T_n)$$

$$C_n = S_{2n} + \frac{1}{16}(S_{2n} - S_n)$$

$$R_n = C_{2n} + \frac{1}{64}(C_{2n} - C_n)$$

4.4.2 Richardson Extrapolation

Using low-order formulas to obtain high-accuracy results is known as Richardson extrapolation acceleration.

The general Richardson extrapolation formula is:

$$T_m^{(1)}(h) = \frac{2^m T(h/2) - T(h)}{2^m - 1}$$

After m steps of acceleration, the remainder takes the form:

$$T(h) = I + \delta_1 h^2 + \delta_2 h^4 + \cdots + \delta_m h^{2m}$$

Let $T_k^{(0)}$ denote the result of the trapezoidal rule after k halving steps. Let $T_k^{(m)}$ denote the result after m steps of Richardson extrapolation on $T_k^{(0)}$. Then, the recursive formula becomes:

$$T_k^{(m)} = \frac{4^m T_{k+1}^{(m-1)} - T_k^{(m-1)}}{4^m - 1}, \quad m = 1, 2, \ldots$$

This is also called the Romberg integration algorithm.

4.5 Adaptive Quadrature

Adaptive quadrature aims to achieve such a subdivision of the integration area s.t. the errors on the individual sub-intervals contribute equally to the overall error. Then we further subdivide only those subintervals that have a large error. Procedure:

1. *Evaluate quadratures (either higher degree or composite method) over interval*
2. *Calculate local error, decide to split intervals or not depending on tolerance*
3. *Continue recursively until error over entire interval is below tolerance*

4.6 Quadrature in $\mathbb{R}^d$

The number of function evaluations increases with n^d. The convergence rate $K = O(N^{-r/d})$, where r is the convergence rate of the one-dimensional quadrature formula.

4.7 Convergence and Stability

In a quadrature formula, if

$$\lim_{n\to\infty,\ h\to 0} \sum_{k=0}^{n} A_k f(x_k) = \int_a^b f(x)\,dx,$$

where

$$h = \max_{1\le i\le n} |x_i - x_{i-1}|,$$

then the quadrature formula

$$\int_a^b f(x)\,dx \approx \sum_{k=0}^{n} A_k f(x_k)$$

is said to be ***convergent****.*

If

$$\left| \sum_{k=0}^{n} A_k \left[\tilde{f}(x_k) - f(x_k) \right] \right| \le \varepsilon, \quad \textit{whenever} \quad |\tilde{f}(x_k) - f(x_k)| < \delta,$$

then the quadrature formula is said to be ***stable****.*

If the coefficients

$$A_k > 0 \quad \textit{for } k = 0, 1, \ldots, n,$$

then the quadrature formula is ***stable****.*

However, when $n \ge 8$, the ***Cotes coefficients*** *have both positive and negative values, in which case* ***stability cannot be guaranteed****.*

Chapter 5

Trigonometric Interpolation

Sei $f \in L^2(a,b)$.

5.0.1 Fourier-Reihe

Jede auf dem Intervall $(0,1)$ im Quadrat integrierbare Funktion f ist der Grenzwert der Fourierreihe

$$f(t) = \sum_{k=-\infty}^{\infty} \hat{f}(k) e^{2\pi i k t}$$

wobei der Fourier-Koeffizient definiert ist als:

$$\hat{f}(k) = \int_0^1 f(t) e^{-2\pi i k t}\, dt$$

5.0.2 Reelle Fourier-Reihe

Ist f insbesondere reellwertig, so kann man auch schreiben:

$$f(t) = \frac{a_0}{2} + \sum_{j=1}^{\infty} \left(a_j \cos(2\pi j t) + b_j \sin(2\pi j t) \right)$$

mit den Koeffizienten:

$$a_j = 2 \int_0^1 f(t) \cos(2\pi j t)\, dt, \quad b_j = 2 \int_0^1 f(t) \sin(2\pi j t)\, dt$$

5.0.3 Skalierte Fourier-Matrix

Die skalierte Fourier-Matrix F_n ist unitär:

$$F_n^{-1} = \frac{1}{n} F_n^H = \frac{1}{n} \overline{F_n}^T$$

5.0.4 L^2-Norm über (a,b)

$$\|g\|_{L^2(a,b)} = \left(\sum_{j=-N/2}^{N/2} |g_j|^2 (b-a) \right)^{1/2}$$

5.1 Discrete Fourier Transform (DFT)

Given points $x_j = \frac{2\pi j}{n}$ and their function values $y_j = y(x_j)$, we want to interpolate the function with a trigonometric series, that fits our sampled function values at all points. This is equivalent to solving the LGLS $\overline{F_n}c = y$. Since $\overline{F_n}^{-1} = \frac{1}{n}F_n \to c = \frac{1}{n}F_n y =$ `1/n * fft.fft(y) = fft.ifft(y)` and $F_n^{-1} = n\overline{F_n} \to y = nF_n^{-1}c =$ `n * fft.ifft(c) = fft.fft(c)`.

$$c = [\ \underbrace{c_0, c_1, \cdots, c_{\frac{n}{2}-1}}_{\text{coefficients of positive freqs}}\ ,\ \underbrace{0, \cdots, 0}_{\text{zero padding}},\ \underbrace{c_{\frac{n}{2}}, \cdots, c_{n-1}}_{\text{coefficients of negative freqs}}\ ]$$

```
freqs = np.fft.fftshift(np.arange(-maxfreq, maxfreq)
```

All circulant matrices of the same dimensions have the same set of eigenvectors $\mathbf{v}_k$, the eigenvalues λ_k differ depending on the sequence $\mathbf{u}$ defining $\mathbf{C}$. Let $C_{i,j} = u_{i-j}$ for an n-periodic $\mathbf{u} \in l^\infty(\mathbb{Z}), u_i \in \mathbb{C}$. $\mathbf{C}$'s eigenvalues and eigenvectors are then given by

$$\lambda_k = \sum_{l=0}^{n-1} u_l \omega_n^{-lk} \quad \textit{and} \quad \mathbf{v}_k = (\omega_n^{jk})_{j=0}^{n-1} \in \mathbb{C}^n,\ \omega_n = e^{-2\pi i/n}$$

for $k \in \{0, \ldots, n-1\}$. The set $\{\mathbf{v}_0, \ldots, \mathbf{v}_{n-1}\}$ forms a trigonometric basis of $\mathbb{C}^n$, i.e. $\mathbf{v}_m^H \mathbf{v}_m = n$ and $\mathbf{v}_k^H \mathbf{v}_m = 0$ for $k \neq m$.

Definition 5.1 *The **Fourier matrix** is defined as $\mathbf{F}_n = (\omega_n^{lj})_{l,j=0}^{n-1} \in \mathbb{C}^{n,n}$.*

Lemma 5.2 (Diagonalization of circulant matrices) *For any matrix $\mathbf{C} \in \mathbb{K}^{n,n}, C_{i,j} = u_{i-j}, (u_k)_{k\in\mathbb{Z}}$ n-periodic sequence, holds true*

$$\mathbf{C}\mathbf{F}_n = \overline{\mathbf{F}}_n\, diag(d_1, \ldots, d_n), \quad \mathbf{d} = \mathbf{F}_n(u_0, \ldots, u_{n-1})^T.$$

*The mapping $\mathcal{F} : \mathbf{y} \mapsto \mathbf{F}_n\mathbf{y}$ is called **DFT**.*

Definition 5.3 (DFT) *The linear map $\mathcal{F}_n : \mathbb{C}^n \to \mathbb{C}^n, \mathcal{F}_n(\mathbf{y}) = \mathbf{F}_n\mathbf{y},\ \mathbf{F} \in \mathbb{C}^n$, is called **discrete Fourier transform** (DFT), i.e. for $\mathbf{c} = \mathcal{F}_n(\mathbf{y})$*

$$c_k = \sum_{j=0}^{n-1} y_j \omega_n^{kj}, \quad k = 0, ..., n-1.$$

Lemma 5.4 (Inverse Fouriermatrix) *The scaled Fourier-matrix $\frac{1}{\sqrt{n}}\mathbf{F}_n$ is unitary, its inverse is therefor given by*

$$\mathbf{F}_n^{-1} = \frac{1}{n}\mathbf{F}_n^H = \frac{1}{n}\overline{\mathbf{F}}_n.$$

The inverse DFT therefore satisfies

$$c_k = \sum_{j=0}^{n-1} y_j \omega_n^{kj} \iff y_k = \sum_{j=0}^{n-1} c_j \omega_n^{-kj}.$$

5.1.1 Discrete Convolution via DFT

Theorem 5.5 (Convolution Theorem) *The discrete periodic convolution $*_n$ between n-dimensional vectors* $\mathbf{u}$ *and* $\mathbf{x}$ *is equal to the inverse DFT of the component-wise product between the DFTs of* $\mathbf{u}$ *and* $\mathbf{x}$*, i.e.*

$$\mathbf{u} *_n \mathbf{x} = \sum_{j=0}^{n-1} u_{k-j} x_j = \mathbf{F}_n^{-1}((\mathbf{F}_n \mathbf{u})_j (\mathbf{F}_n \mathbf{x})_j)_{j=1}^n.$$

5.1.2 Fast Fourier Transform

Let $c_k = (\mathbf{F}_n \mathbf{y})_k$. By splitting $(\mathbf{F}_n \mathbf{y})_k = c_k = (c^1)_k + \omega_n^k (c^2)_k$, where c^1 m-DFT of y^1, c^2 m-DFT of y^2 and $n = 2m$. This is the basis for a divide and conquer approach by further splitting c^1, c^2 in the next step.

Complexity *The overall complexity for FFT is $\mathcal{O}(\mathbf{n} \log \mathbf{n})$.*

5.1.3 Frequency Filtering via DFT

Given signal $\mathbf{x}$*, the Fourier transform is* $\mathbf{c} = \mathbf{F}_n \mathbf{x}$*. Vector $|c_k|, |c_{n-k}|$ measures how much oscillation with frequency k is present in signal* $\mathbf{x}$*, $k = 0, \ldots, \lfloor \frac{n}{2} \rfloor$. Note that $c_{n-k} = \overline{c_k}$ and $\omega_n^{-(n-k)j} = \overline{\omega}_n^{-kj}$.*

Idea for denoising a signal:

1. *transform signal to frequency domain,*
2. *apply a low-pass filter to cut off high frequency content,*
3. *transform back to time/space domain.*

Chapter 6

Differential Equations

Ein Fixpunkt einer Differentialgleichung ist ein Wert y^*, bei dem sich die Lösung nicht mehr ändert: $y'(t) = 0$ wenn $y(t) = y^*$.The idea is to discretize the domain and the differential equation, such that approximate values of the solution $y(x)$ at a series of nodes can be computed:

$$a = x_0 < x_1 < \cdots < x_N = b, \quad x_n = a + nh, \quad h = \frac{b-a}{N}$$

with corresponding approximations:

$$y(x_n) \approx y_n, \quad f(x_n, y_n) \approx y'(x_n) \approx f_n \quad n = 0, 1, \ldots, N.$$

- ***Finite Difference Approximation:*** *Use finite differences to approximate derivatives.*
- ***Numerical Integration:*** *Transform the ODE into an integral equation:*

  $$y(x_m) - y(x_n) = \int_{x_n}^{x_m} f(x, y(x))\, dx, \quad y(x_0) = y_0,$$

  then apply numerical quadrature to approximate the integral.
- ***Taylor Series Expansion:*** *Expand $y(x)$ using a Taylor series to construct numerical schemes and analyze truncation error to achieve desired accuracy.*

6.1 ODE - Ordinary Differential Equations

Equation containing derivatives of an unknown function dependent on one variable. General form:

$$\vec{\dot{y}} = \vec{f}(t, y(t))$$

$$y(t_0) = y_0$$

An order n diff. equation can be reduced to a system of n order 1 diff. equations, which requires n initial conditions:

$$y^{(n)}(t) = f(t, \dot{y}, \cdots, y^{(n-1)})$$

An ODE with no explicit time dependence is called autonomous. Any ODE can be autonomized by defining a new state vector with t as the last component.

Definition 6.1 (Lipschitz-Continuity) *If $f(x, y)$ is continuous on $[a, b] \times \mathbb{R}^d$, and there exists a constant $L > 0$, independent of x and y, such that for any functions $y_1(x)$ and $y_2(x)$ defined on $[a, b]$, the following condition holds:*

$$|f(x, y_1) - f(x, y_2)| \leq L|y_1 - y_2|.$$

Then the initial-value problem has a unique solution.

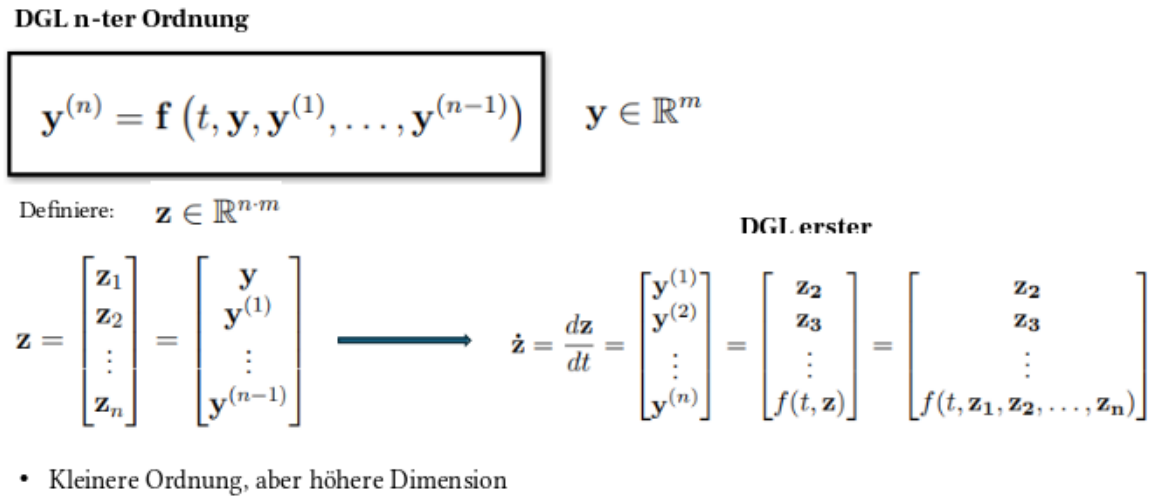

Figure 6.1: Conversion to 1 order DGL

6.1.1 Methods for solving 1st order ODEs:

The implicit methods require solving nonlinear equations to calculate y at previously unknown points. Explicit methods require less computing but have lower stability and accuracy.

- *Explicit Euler (eE)* $O(h)$: $y_{n+1} = y_n + hf(y_n, t_n) \quad h = t_{n-1} - t_n$
- *Implicit Euler (iE)*: $y_{n+1} = y_n + hf(y_{n+1}, t_{n+1}) \quad h = t_{n-1} - t_n$
- *Explicit Middle-Point (eMP)* $O(h^2)$: $y_{n+1} = y_n + hf(y_{n+\frac{1}{2}}, t_{n+\frac{1}{2}}) \quad y_{n+\frac{1}{2}} = y_n + \frac{h}{2}f(y_n, t_n)$
- *Implicit Middle-Point (iMP)*: $y_{n+1} = y_n + hf(t_n + \frac{h}{2}, \frac{1}{2}(y_{n+1} + y_n))$
- *Explicit Trapezoidal Rule (eTR)* $O(h^2)$: $y_{n+1} = y_n + \frac{h}{2}(f(y_n, t_n) + f(y_n + hf(y_n, t_n), t_{n+1}))$
- *Implicit Trapezoidal Rule (iTR)*: $y_{n+1} = y_n + \frac{h}{2}(f(y_n, t_n) + f(y_{n+1}, t_{n+1}))$

Iterative Scheme for Implicit Methods

$$y_{n+1}^{(0)} = y_n + hf(x_n, y_n)$$
$$y_{n+1}^{(k+1)} = y_n + hf(x_{n+1}, y_{n+1}^{(k)}), \quad k = 0, 1, 2, \ldots$$

Converges to solution if iteration is stable.

Multi-Step Methods

A linear multistep method approximates $y(x_{n+1})$ *using a linear combination of the values of y and* y' *at several previous nodes.*

The general form is:

$$y_{n+1} = a_0 y_n + \cdots + a_k y_{n-k} + h\left(b_{-1} f_{n+1} + b_0 f_n + \cdots + b_k f_{n-k}\right)$$

To derive a specific multistep method, apply a Taylor series expansion of all terms y_j *and* f_j *around the point* x_n*, and compare with the Taylor expansion of the exact solution* $y(x_{n+1})$ *at the same point.*

By matching the coefficients of like powers of h, a system of equations is obtained. Solving this system allows the determination of the coefficients a_i and b_i. A k-step method requires $k-1$ initial values, which must be computed using a higher-order single-step method to match the overall accuracy.

6.1.2 Methods for solving 2nd order ODEs:

- *Strömer-Verlet (St-V) ($O(h^2)$, explicit, 2-step method):* $y_{n+1} = 2y_n - y_{n-1} + a(y_n)h^2$
- *Leap-Frog (1-step method):*

  $$v_{n+\frac{1}{2}} = v_{n-\frac{1}{2}} + ha(y_k) \quad y_{n+1} = y_n + hv_{n+\frac{1}{2}}$$

  alternately calculates the y and v at different points in time
- *Velocity-Verlet (VV) ($O(h^2)$, explicit, more stable than St-V, same time grid, energy conservation):*

  $$v_{n+1} = v_n + \frac{h}{2}(f(t_n, y_n) + f(t_{n+1}, y_{n+1})) \quad y_{n+1} = y_n + hv_n + \frac{h^2}{2}f(t_n, y_n)$$

[Clenshaw-Curtis quadratur, Vergleich mit der Linearisierung]

6.1.3 Error Analysis

The global truncation error consists of the initial error and the accumulation of local truncation errors. Its order is one less than the order of the local truncation error.

Definition 6.2 *Local Truncation Error (LTE)*
We assume that the numerical solution is exact up to step n, i.e., $y_n = y(x_n)$.

$$T_{n+1} = y(x_{n+1}) - y_{n+1}, \quad \textit{assuming } y_n = y(x_n)$$

*If $T_{n+1} = \mathcal{O}(h^{p+1})$, the method has **order** p.*

Definition 6.3 *Global Truncation Error (GTE)*

$$e_n = y(x_n) - y_n.$$

It measures the cumulative effect of all local truncation errors and their propagation from x_0 to x_n. If $E = \max_i(y(x_i) - y_i) = O(h^p)$, then the ESV is convergent with order p.

Definition 6.4 (Consistency Error) *The consistency error at x_n ist defined as : $\tau_n = \frac{e_n}{h} = \frac{y(x_n)-y(x_{n-1})}{h} - \varphi(x_{n-1}, y(x_{n-1}), h) \approx y'(x_{n-1}) - y'_{n-1}$. A Single-Step method is consistent of order p, if holds:*

$$\tau = \max_i |\tau_i| = O(h^p) \quad \textit{for h small enough}, p \geq 1$$

Theorem 6.5 (Convergence & Consistency) *If the step function $\varphi(x, y, h)$ is Lipschitz-continuous in y, then the following global error estimation holds:*

$$E = \max_i ||y(x_i) - y_i|| \leq (||y(x_0) - y_0|| + \sum_{i=1}^{N} ||e_i||)e^{L(x_n - x_0)}$$

, where L ist die Lipschitz constant of phi. For f smooth enough: consistency of order $p \rightarrow$ convergence of order p.

6.2 Runge-Kutta-Verfahren

The idea is to construct a high-order single-step recursive method. The highest achievable accuracy with Euler's method and its variations is second order. To approximate the derivative in the mean value theorem, one may use a weighted average of derivatives at different points.

$$\frac{y(x_{n+1}) - y(x_n)}{h} = y'(\xi) \quad \xi \in [x_n, x_{n+1}]$$

A general explicit Runge-Kutta (RK) method with s stages:

$$k_i = f\left(x_n + c_i h, y_n + h \sum_{j=1}^{s} a_{ij} k_j\right), \quad y_{n+1} = y_n + h \sum_{i=1}^{s} b_i k_i$$

Where $\sum_{i=1}^{s} c_i = 1, c_i \leq 1, \sum_{j=1}^{s} a_{ij=1}$ *For an explicit method,* $k_1 = f(x_n, y_n)$*, i.e.* $c_1 = 0$*. For an implicit method, one has to solve a nonlinear equation system of order s. If the number of degrees of freedom (free parameters) exceeds the number of order conditions, a family of methods can be generated.*

Definition 6.6 (Autonomization Invariance) *A one-step method is automization invariant, if:*

$$\sum_{i=1}^{s} b_i = 1 \quad c_i = \sum_{i=1}^{s} a_{ij} \quad \forall i \in \{0, \ldots, s\}$$

For these methods, one obtains the same result with the autonomous or non-autonomous system.

Butcher-Tableau

For explicit RK-ESV the maximal order of consistency is

$$p = \begin{cases} s & \text{for } s \leq 4 \\ s-1 & \text{for } 4 < s \leq 7 \\ s-2 & \text{for } 8 \leq s \end{cases}$$

Since the derivation of Runge-Kutta methods is based on Taylor expansion, the accuracy primarily depends on the smoothness of the solution function. Therefore, for problems where the solution function is not very smooth, it is often better to use a low-order method with a smaller step size h, rather than a high-order method.

6.2.1 Convergence and Stability

Definition 6.7 (Convergence) *An algorithm is said to be* ***convergent*** *if, for any fixed* $x = x_n = x_0 + nh$*, as* $h \to 0$ *(and thus* $n \to \infty$*), the numerical solution satisfies* $y_n \to y(x_n)$ *where* $y(x_n)$ *is the exact solution at* x_n*.*

Theorem 6.8 *An explicit one-step method of order p is convergent if the following conditions are satisfied:*

1. *The increment function* $\varphi(x, y, h)$ *satisfies the Lipschitz condition in y, i.e., there exists a constant* $L_\varphi > 0$ *such that for all* $x, y_1, y_2 \in \mathbb{R}$*,*

$$|\varphi(x, y_1, h) - \varphi(x, y_2, h)| \leq L_\varphi |y_1 - y_2|.$$

2. *The initial value is exact:* $y_0 = y(x_0)$.

Theorem 6.9 (Kollokationsverfahren)

Gauss–Legendre Runge–Kutta (GLRK) methods are based on collocation at Gauss–Legendre nodes, which are the roots of Legendre polynomials on the interval $[0, 1]$. *The collocation points* c_i *(nodes) are the roots of the Legendre polynomial of degree* s, b_i *are the weights of the Gauss–Legendre quadrature rule and* a_{ij} *are computed from the Lagrange basis polynomials evaluated at the collocation points.*

6.2.2 Stabilität der RK-Verfahren

Die numerische Stabilität gibt an, wie gut die numerische Methode mit kleinen Störungen in den Daten umgeht, ohne dass die Fehler zu sehr wachsen. Es gibt Differentialgleichungen, bei welchen die Approximation der Lösung sehr stark mit der Schrittweite h variiert.

Theorem 6.10 (Steife Differentialgleichungen) *Ein Problem ist steif, wenn die Stabilität eine stärkere Beschränkung für die Zeitschritte vorgibt, als die Genauigkeit.*

Indikatoren für steife Probleme:

- *Prozesse mit stark unterschiedlichen Zeitskalen in einem Gleichungssystem*
- *Schnelle Übergänge in einem System*

Steifigkeit bei linearen DGL:

$$\dot{y} = Ay + b \quad A \in R^{n \times n}$$

$$\textit{System ist steif} \leftrightarrow Re(\lambda_i) < 0 \quad \forall i \qquad S = \frac{max_i|Re(\lambda_i)|}{min_i|Re(\lambda_i)|} \gg 1 \quad \textit{Steifigkeitsparameter}$$

Steifigkeit bei nichtlinearen DGL: Um die lokale Steifheit zu bestimmen, kann man das System lokal linearisieren.

6.2.3 Stabilitätsgebiet

Theorem 6.11 (Stabilitätsfunktion) *Wenn man ein Einschrittverfahren auf das Modellprolbem anwendet:*

$$y(t) = y(t_0)e^{\lambda t} \rightarrow y_k = S(z)y_{k-1} = S(z)^k y_0 \quad z := h\lambda$$

Für explizite Verfahren ist $S(z)$ *ein Polynom vom Grad höchstens* s.

- *eE:* $S(z) = 1 + z$
- *iE:* $S(z) = \frac{1}{1-z}$
- *eMR:* $S(z) = 1 + z + \frac{1}{2}z^2$
- *RK45:* $S(z) = 1 + z + \frac{1}{2}z^2 + \frac{1}{6}z^3 + \frac{1}{24}z^4$
- *RK-Verfahren:* $S(z) = 1 + z\boldsymbol{b}^T(I - z\boldsymbol{A})^{-1}\boldsymbol{1} = \frac{det(\boldsymbol{I} - z\boldsymbol{A} + z\boldsymbol{1}\boldsymbol{b}^T)}{det(\boldsymbol{I} - z\boldsymbol{A})} \quad 1 = (1, \ldots, 1)^T \in \mathbb{R}^s$

Definition 6.12 (A-Stabilität) *SG enthält die gesamte komplexe LHP. If the error introduced at any step of a numerical method gradually decays in subsequent steps, the method is said to be* ***absolutely stable****.*

Definition 6.13 (L-Stabilität) *A-stabil und* $\lim_{z \to \infty} S(z) = 0$.

*We say method A is **more stable** than method B if the absolute stability region of A is larger than that of B or it poses less constraints on the timestep.*

Theorem 6.14 (Stabilitätsgebiet)

$$SG = \{z \in \mathbb{C}; |S(z)| < 1 \; bzw. \; |S(z)|^2 < 1\}$$

- *eE:* $|1 + h\lambda| < 1 \Leftrightarrow h \in (0, \frac{2}{|\lambda|}) \qquad |1 + z| < 1 \Leftrightarrow z \in (-2, 0)$

Implizite RK-Verfahren stellen keine Bedingung an den Zeitschritt h, jedoch verlangt es für jeden Zeitschritt die Lösung eines nicht-linearen Systems der Dimension $s \cdot d$ (s = Stufenordnung, d = Dimension).

$$\boldsymbol{k}_i = \boldsymbol{f}\left(\boldsymbol{y}_0 + h\sum_{j=1}^{s} a_{ij}\boldsymbol{k}_j\right), i = 1, \ldots, s$$

$$\xrightarrow{Linearisierung} \boldsymbol{k}_i = \boldsymbol{f}(\boldsymbol{y}_0) + h\boldsymbol{D}\boldsymbol{f}(\boldsymbol{y}_0)\left(\sum_{j=1}^{s} a_{ij}\boldsymbol{k}_j\right), i = 1, \ldots, s$$

Problem: Diese Linearisierung führt aber zu einer schlechten Konvergenzordnung.

6.2.4 Linear-implizite und ROW-Methoden

Für diagonalimplizite RK-Verfahren

6.2.5 Lineare Transportgleichung

Theorem 6.15 (Lax Wendroff)

$$u_{n+1} = u(t_{n+1}, x) = u_n - hc(x)$$

6.2.6 Adaptive Schrittweitensteuerung

Sei TOL die globale und tol die lokale Fehlertoleranz.

Toleranzkriterien:

$$TK1 \quad |e_j| \leq tol_j = TOL\frac{h_j}{T - t_0}$$

6.3 Splitting Verfahren

Definition 6.16 (Symplektische Verfahren) *Numerische Verfahren für Erhaltungsgrößen. Time-reversible, stable for oscillatory problems, bounded energy error over time, very stable for conservative systems and can use relatively large time steps without instability. Examples: StV, iMp, PRK, GLRK*

Ein Evolutionsoperator propagiert eine Anfangsbedingung, entsprechend der DGL:

$$\Phi^{t_2,t_1} y(t_1) = y(t_2)$$

Aufteilung in mehrere, einfachere Propagatoren zur Reduktion der Komplexität:

$$\Phi^{t,t_0} = \Phi_1^{t,t_0}\Phi_2^{t,t_0}$$

Time-autonomous propagators are translation-invariant. Discretization to simplify solution.

Chapter 7

Methods for Solving Linear Systems

We want to solve an LSE $Ax = b$ for given $A \in \mathbb{K}^{n,n}, b \in \mathbb{K}^n$ for x. Regularity of A ensures existence and uniqueness of a solution x.

7.1 Direct Methods

Direct Methods compute the exact solution of a system of linear equations in a finite number of arithmetic operations (assuming no rounding errors). They are effective for solving low-order dense linear systems.

7.1.1 Norms of Vectors and Matrices

Vector Norms

For any vector $x = (x_1, x_2, \ldots, x_n) \in \mathbb{R}^n$, the p-norm is defined as:

$$\|x\|_p = \left(\sum_{i=1}^{n} |x_i|^p \right)^{1/p}, \quad 1 \leq p < \infty$$

$$\|x\|_\infty = \max_{1 \leq i \leq n} |x_i|$$

Theorem: *All norms on $\mathbb{R}^n$ are equivalent. That is, they induce the same topology.*

Matrix Norms

A matrix norm $\|A\|$ for $A \in \mathbb{R}^{m \times n}$ must satisfy:

1. ***Positive Definiteness:*** $\|A\| = 0 \iff A = 0$
2. ***Homogeneity:*** $\|\alpha A\| = |\alpha| \cdot \|A\| \quad \forall \alpha \in \mathbb{C}$
3. ***Triangle Inequality:*** $\|A + B\| \leq \|A\| + \|B\|$
4. ***Submultiplicativity (Consistency):*** $\|AB\| \leq \|A\| \cdot \|B\|$ *(if A, B are conformable)*

If $\|AB\|_\gamma \leq \|A\|_\alpha \cdot \|B\|_\beta$, the norms are said to be ***consistent****.*

Frobenius norm:

$$\|A\|_F = \left(\sum_{i=1}^{n} \sum_{j=1}^{n} |a_{ij}|^2 \right)^{1/2}$$

This is a natural extension of the vector 2-norm.

Operator norm (induced by vector norms):

$$\|A\|_p = \max_{x \neq 0} \frac{\|Ax\|_p}{\|x\|_p}$$

Special cases:

- ***Row-sum norm:*** $\|A\|_\infty = \max_{1\le i\le n} \sum_{j=1}^{n} |a_{ij}|$
- ***Column-sum norm:*** $\|A\|_1 = \max_{1\le j\le n} \sum_{i=1}^{n} |a_{ij}|$
- ***Spectral norm:*** $\|A\|_2 = \sqrt{\lambda_{\max}(A^T A)}$

*Note: Frobenius norm is **not** an operator norm, and we typically focus on norms that are consistent. Operator norms are always consistent.*

*The **spectral radius** of a matrix $A \in \mathbb{R}^{n\times n}$ is defined as:*

$$\rho(A) = \max_{1\le i\le n} |\lambda_i|$$

where λ_i are the eigenvalues of A. Even when all elements of A are real, its eigenvalues and eigenvectors may be complex.

The condition number measures how the relative error in the input affects the relative error in the output.

$$\frac{\|\delta \mathbf{x}\|}{\|\mathbf{x}\|} \le \frac{cond(A)}{1 - cond(A)\frac{\|\delta \mathbf{A}\|}{\|\mathbf{A}\|}} \cdot \frac{\|\delta \mathbf{b}\|}{\|\mathbf{b}\|}$$

Notes:

- *$cond(A)$ depends on the matrix norm used.*
- *Common norms and corresponding condition numbers:*

$$\begin{aligned} cond_1(A) &= \|A\|_1 \cdot \|A^{-1}\|_1 \\ cond_\infty(A) &= \|A\|_\infty \cdot \|A^{-1}\|_\infty \\ cond_2(A) &= \|A\|_2 \cdot \|A^{-1}\|_2 = \frac{\lambda_{\max}(A)}{\lambda_{\min}(A)} \quad \textit{(if A is symmetric)} \end{aligned}$$

- *If A is invertible, then $cond(A) < \infty$*
- *If A is orthogonal, then $cond_2(A) = 1$*
- *For any scalar $\alpha \in \mathbb{R}$, $cond(\alpha A) = cond(A)$*
- *If R is orthogonal, then:*

$$cond_2(RA) = cond_2(AR) = cond_2(A)$$

A matrix is considered ill-conditioned if:

- *$cond(A) \to \infty$ as $n \to \infty$*
- *Determinant is too large or too small (rows or columns are nearly linearly dependent)*
- *Matrix elements differ by several orders of magnitude, irregularly*
- *Very small pivot elements during elimination*
- *Eigenvalues differ by several orders of magnitude*

7.1.2 Matrix-Zerlegungen

Gauss elimination

The idea is to use $Ax = b \iff TAx = Ux = Tb$ *for* $T \in \mathbb{K}^{n,n}$ *regular and A non singular.*

Complexity *The complexity of gauss elimination is* $\mathcal{O}(\mathbf{n^3})$*. If the given matrix is triangular, the complexity reduces to* $\mathcal{O}(\mathbf{n^2})$*.*

Das Lösen von LGS mittels Matrizenterlegungen ist besonders nützlich bei wiederholten Lösungen des gleichen Systems.

Theorem 7.1 (LU-Zerlegung) *Sei* $A \in \mathbb{R}^{n\times n}$ *invertierbar, dann existieren* $P, U, L \in \mathbb{R}^{n\times n}$*, sodass PA = LU. L ist eine untere Dreiecksmatrix mit 1 auf der Diagonale, U eine obere Dreiecksmatrix und P eine Permutationsmatrix. Die Gauss-Elimination ergibt die LU-Zerlegung.*

$$Ax = b \Leftrightarrow LUx = Pb \Leftrightarrow Lz = Pb \quad \& \quad Ux = z$$

Vorwärtssubstitution $y_i = \frac{b_i - \sum_{j=1}^{i-1} L_{ij} y_j}{L_{ii}}$ *Rückwärtssubstitution* $x_i = \frac{y_i - \sum_{j=1+1}^{n} U_{ij} x_j}{U_{ii}}$

Complexity *The complexity of solving LSEs using LU decomposition is* $\mathcal{O}(\mathbf{n^3})$*. However, solving for N RHS, we archieve a complexity of* $\mathcal{O}(\mathbf{n^3} + \mathbf{Nn^2})$ *over* $\mathcal{O}(Nn^3)$ *using Gauss elimination.*

Definition 7.2 (Uniqueness) *Let A be an* $n \times n$ *matrix. If all the leading principal minors* $D_i \neq 0$ *for* $i = 1, 2, \ldots, n$*, then the LU decomposition of A is unique, where L is a unit lower triangular matrix.*

Definition 7.3 (Properties of Symmetric Positive Definite Matrices) *If A is symmetric positive definite (SPD), then the following hold:*

- A^{-1} *is also SPD, and* $a_{ii} > 0$
- *All eigenvalues* $\lambda_i > 0$
- *All leading principal minors* $\det(A_k) > 0$*, where* A_k *is the* $k \times k$ *leading principal submatrix*

Theorem 7.4 (Cholesky-Zerlegung) *Ist A symmetrisch und positiv definit, dann existiert eine Zerlegung* $A = LL^T = U^T U$*, wobei L/U eine untere/obere Dreiecksmatrix mit strikt positiven Diagonaleinträgen ist.*

$A = \tilde{L}U = \tilde{L}D\tilde{U} = \tilde{L}\sqrt{D}\sqrt{D}\tilde{L}^T = L^T L \quad D =$ *Diagonaleinträge von* $U \quad \tilde{U} =$ *i-te Zeile durch* d_i *geteilt*

Theorem 7.5 (QR-Zerlegung) ***Reduzierte QR-Zerlegung****: Sei* $A \in \mathbb{R}^{m\times n}$ *mit* $m \geq n$*, dann existiert ein* $\hat{Q} \in \mathbb{R}^{m\times n}, \hat{R} \in \mathbb{R}^{n\times n}$ *sodass* $A = \hat{Q}\hat{R}$*, wobei* $\hat{Q}$ *orthonormale Spalten hat und* $\hat{R}$ *eine obere Dreiecksmatrix ist.* ***Vollständige QR-Zerlegung****:*$A = QR$*, wobei* $Q := \begin{pmatrix}\hat{Q} q_{n+1} \ldots & q_m\end{pmatrix} \in \mathbb{R}^{m\times m}, R := \binom{\mathbb{R}}{0} \in \mathbb{R}^{m\times n}, \text{mit } Q^T Q = \mathbb{I}$ *(orthogonal).*

Householder-Reflexion*: Lineare Transformation, die einen Vektor bzgl. einer Ebene oder Hyperebene spiegelt.* $Q = I - 2\frac{vv^T}{v^T v}$ *für v gleich dem normierten Normalenvektor der Spiegelebene.* $Q = Q_0 \tilde{Q}_1 \ldots \tilde{Q}_n$

Lemma 7.6 *Sei* $x \neq 0 \in \mathbb{R}^m$ *und* $x \notin span(e_1)$*. Definiere v als* $x \pm \frac{e_1}{||x||}$*, dann gilt* $Qx = ||x||e_1$

Theorem 7.7 (Singulärwertzerlegung) *Sei* $A \in \mathbb{R}^{m\times n}$*, dann existiert ein* $U \in \mathbb{R}^{m\times m}, V \in \mathbb{R}^{n\times n}$ *und* $\Sigma \in \mathbb{R}^{m\times n}$*, sodass* $A = U\Sigma V^T$ *wobei V und U orthogonal und* $\Sigma = \text{diag}(\sigma_1, \ldots, \sigma_p)$ *mit* $p = \min\{m, n\}.\sigma_1 \geq \cdots \geq \sigma_p \geq 0$ *sind die Singulärwerte von A.*

7.2 Least Square Solutions

We want to estimate the "most likely" model parameters given a set of data. Suppose we have a model $f(x) = a_1x_1 + ... + a_nx_n$, $f : \mathbb{R}^n \to \mathbb{R}$ *and a series of measurements* $(x^{(k)}, y^{(k)})_{k=1}^n$, $x^{(k)} \in \mathbb{R}^n$, $y^{(k)} \in \mathbb{R}$, *where* $x^{(k)} \mapsto y^{(k)} = f(x^{(k)})$. *Our goal is to estimate the parameters* $a_1, ..., a_n$ *with this series of experiments. We can write the problem in matrix form*

$$\begin{pmatrix} x_1^{(1)} & x_2^{(1)} & \dots & x_n^{(1)} \\ x_1^{(2)} & x_2^{(2)} & \dots & x_n^{(2)} \\ \vdots & \vdots & \ddots & \vdots \\ x_1^{(n)} & x_2^{(n)} & \dots & x_n^{(n)} \end{pmatrix} \begin{pmatrix} a_1 \\ \vdots \\ a_n \end{pmatrix} = \begin{pmatrix} y^{(1)} \\ \vdots \\ y^{(n)} \end{pmatrix}$$

and estimate parameters by solving $X^T a = y$, *which is linear regression. Due to measurement errors, a large number of experiments and errors in our model, a solution to our problem generally does not exist. We therefore want to approximate the solution such that* $Ax \approx b$, *which is equivalent to minimizing the norm of residual* $||Ax - b||_2$. *These solutions are called least squares solutions.*

Definition 7.8 *For given* $A \in \mathbb{K}^{m,n}$, $b \in \mathbb{K}^m$ *the vector* $x \in \mathbb{R}^n$ *is a* ***least squares solution*** *of the linear system of equations* $Ax = b$ *if*

$$x \in \operatorname{argmin}_{y \in \mathbb{K}^n} ||Ay - b||_2 \iff ||Ax - b|| = \sup_{y \in \mathbb{K}^n} ||Ay - b||_2$$

If $x \in \operatorname{lsq}(A, b)$, *then* Ax *is closest to B in* $\operatorname{Im}(A)$, *i.e. projection of b on* $\operatorname{Im}(A)$.

Theorem 7.9 *For any* $A \in \mathbb{K}^{m,n}$, $b \in \mathbb{K}^m$ *a least squares solution of* $Ax = b$ *exists.*

Lemma 7.10 *For any matrix* $A \in \mathbb{K}^{m,n}$ *holds*

$$\ker(A) = \operatorname{Im}(A^H)^\perp,$$
$$\ker(A)^\perp = \operatorname{Im}(A^H).$$

A vector $x \in \mathbb{K}^n$ *lies in the null space of A if and only if it is orthogonal to the row space of A, i.e. the column space of* A^H.

Theorem 7.11 (Normal equation) *The vector* $x \in \mathbb{K}^n$ *is least squares solution to the system* $Ax = b$, $A \in \mathbb{K}^{m,n}$, $b \in \mathbb{K}^m$ *if and only if it solves the* ***normal equations***

$$A^T Ax = A^T b.$$

Proof $x \in \operatorname{lsq}(A, b) \iff Ax$ is closest element in $\operatorname{Im}(A)$ to b $\iff Ax - b \in \operatorname{Im}(A)^\perp = \ker(A^H) \iff A^T(Ax - b) = 0$ □

Theorem 7.12 *For* $Ax = b$, $A \in \mathbb{K}^{m,n}$, $m \geq n$, *holds*

$$\operatorname{Im}(A^T A) = \operatorname{Im}(A^T),$$
$$\ker(A^T A) = \ker(A).$$

Corollary 7.13 *If* $m \geq n$ *and* $\ker(A) = \{0\}$ *(rank(A)=n), then the linear system of equations* $Ax = b$, $A \in \mathbb{K}^{m,n}$, $b \in \mathbb{K}^m$ *has a unique least squares solution*

$$x = (A^T A)^{-1} A^T b$$

that can be obtained by solving the normal equations.

1. *Compute regular matrix* $C = A^T A$ $\mathcal{O}(mn^2)$
2. *Compute RHS* $c = A^T b$ $\mathcal{O}(nm)$
3. *Solve s.p.d. linear system of equations* $Cx = c$ $\mathcal{O}(n^3)$

Complexity *The cost of solving a least squares problem using normal equation is* $\mathcal{O}(\mathbf{n^3 + mn^2})$.

The condition number squares in the matrix multiplication $A^T A$, *i.e.* $\mathrm{c}_{A^T A} = \mathrm{c}_a^2$. *It can happen that* $A^T A$ *is not regular in* $\mathbb{M}$ *even if* A *is regular in* $\mathbb{M}$. *Also, if* A *is sparse,* $A^T A$ *is not necessarily sparse.*

Least squares solutions are only unique if they fulfill the full-rank condition.

7.2.1 General solution and Moore-Penrose generalized inverse

Definition 7.14 *The* ***generalized solution*** $x^\dagger$ *of a linear system of equations* $Ax = b$, $A \in \mathbb{K}^{m,n}$, $b \in \mathbb{K}^m$ *is defined as*

$$x^\dagger = \operatorname{argmin}\{||x||_2 : x \in \operatorname{lsq}(A, b)\}.$$

Theorem 7.15 *The generalized solution* $x^\dagger$ *is unique.*

Theorem 7.16 *Given* $A \in \mathbb{K}^{m,n}$, $b \in \mathbb{K}^m$, *the generalized solution* $x^\dagger$ *is defined by*

$$x^\dagger = V(V^T A^T AV)^{-1}(V^T A^T b),$$

where V *is any matrix whose columns form a basis of* $\operatorname{Im}(A)^\perp$. $V(V^T A^T AV)^{-1}V^T$ *is called the* ***Moore-Peurose pseudoinverse*** $A^\dagger$ *of* A.?

7.2.2 Constrained least squares

Given

$$A \in \mathbb{K}^{m,n}, \operatorname{rank}(A) = n,\ b \in \mathbb{K}^m,$$
$$C \in \mathbb{R}^{p,n}, \operatorname{rank}(C) = p,\ p < n,\ d \in \mathbb{R}^p,$$

find $x \in \mathbb{R}^m$, *such that* $\|Ax - b\|_2 \to \min$ *and* $Cx = d$.

Solution via Lagrange multiplier (a saddle point problem)

$$\mathrm{L}(y, m) = \frac{1}{2}\|Ay - b\|_2^2 + m^t(Cy - d)$$
$$x = \operatorname{argmin}_{y \in \mathbb{R}^n} \max_{m \in \mathbb{R}^p} \mathrm{L}(y, m)$$

Solution via augmented normal equations

$$\begin{pmatrix} A^T A & C^T \\ C & 0 \end{pmatrix} \begin{pmatrix} x \\ m \end{pmatrix} = \begin{pmatrix} A^T b \\ d \end{pmatrix}.$$

Solution via SVD. Let $C = U(\Sigma_p\ 0)(V_1\ V_2)^T$ *and* $x_0 = V_1 \Sigma^{-1} U^T d$ *be a particular solution to* $Cx = d$, *then* $x = x_0 + \ker(C) = x_0 + V_2 y$ *is also a solution. Now solve* $\|Ax - b\|_2 = \|AV_2 y - (b - Ax_0)\|_2 \to \min$ *as a linear least squares problem.*

7.3 Orthogonal Transformation Methods

Consider least squares problem $Ax = b, A \in \mathbb{K}^{m,n}, m \gg n$ with a full-rank A. The idea is instead of solving $Ax = b$ find an easier to solve system $\tilde{A}x = \tilde{b}$ such that $\mathrm{lsq}(\tilde{A}, \tilde{b}) = \mathrm{lsq}(A, b)$.

Theorem 7.17 *A matrix is unitary/ orthogonal if and only if the associated linear mapping preserves the 2-norm*

$$Q \in \mathbb{K}^{n,n}\text{unitary} \iff \|Qx\|_2 = \|x\|_2, \forall x \in \mathbb{K}^n.$$

If we can decompose $A = QR$ with Q unitary/orthogonal and R triangular, the normal equation becomes

$$A^T Ax = A^T b \iff x = R^{-1}Q^T b$$

This system is better conditioned with $\mathrm{c_R} = \mathrm{c_A}$.

7.3.1 QR-Decomposition

As a first approach we use Gram-Schmidt orthogonalization. In practice we multiple A by upper-triangular matrices to obtain an orthogonal matrix $Q = AT_1T_2 \dots T_n = AT$. Since A has full rank and each T_i also does, T is invertible. Let $R = T^{-1}$, then $A = QR$. Gram-Schmidt orthogonalization suffers from numerical instabilities due to possible cancellation in subtraction and dividing by ≈ 0.

Theorem 7.18 (QR-decomposition) *For any matrix $A \in \mathbb{K}^{n,k}$ with* $\mathrm{rank}(A) = k$ *there exists*

- *a unique unitary matrix $Q_0 \in \mathbb{K}^{n,k}$ that satisfies $Q_0^H Q_0 = Q_0 Q_0^H = Id_k$ and a unique upper triangular matrix $R_0 \in \mathbb{K}^{k,k}$ with $R_{i,i} > 0, \forall 1 \leq i \leq k$, such that*

  $$A = Q_0 R_0,$$

  the "economical" QR-decomposition.

- *a unitary matrix $Q \in \mathbb{K}^{n,n}$ and a unique upper triangular matrix $R \in \mathbb{K}^{n,k}$ with $R_{i,i} > 0, \forall 1 \leq i \leq n$, such that*

  $$A = QR,$$

  the full QR-decomposition.

Computation of QR decomposition (Householder reflections)

Thea idea is to find a series of orthogonal transformations, such that applied from the left yield a triangular matrix $Q_n \dots Q_2 Q_1 A = R$, similar to Gauss elimination, but now we use orthogonal transformations. These transformations can only rotate and reflect vectors, i.e. preserve length of and angles between vectors. Householder reflections use only reflection represented as projections

$$\mathrm{H}_v a = -(a - 2\mathrm{proj}_v a) = -\left(I_m - 2\frac{vv^T}{v^T v}\right) a$$

The j-th step then tries to eliminate A's j-th column $a_j = (a_1^j \; a_2^j)^T$ by setting

$$v^j = \begin{pmatrix} 0 \\ a_2^j \end{pmatrix} - c_j e^j, \quad \text{with } c_j = \pm\|a_1^j\|.$$

Then $H_{v^j}a^j = (a_1^j\ 0\ \dots\ 0)^T$ and $H_{v^j}q^k = q^k$, where $q^k = H_{v^k}a^k$ for some $k < j$. Note that c_j must be chosen such that cancellation is avoided, i.e. if $(0\ a_2^j)^T$ is almost parallel to the j-th basis vector e_j, choose c_j such that v^j is not small.

Altogether $H_{v^n} \dots H_{v^1} A = R$, $Q = H_{v^1}^T H_{v^n}^T$, then $A = QR$. Q is stored implicitly by storing vectors $v^1, \dots, v^n$ as lower triangular matrix (compressed format).

Complexity *The computational effort for HouseholderQR() of $A \in \mathbb{K}^{m,n}, m > n$, is $\mathcal{O}(\mathbf{mn^2})$ for $m, n \to \infty$.*

Givens rotations, an alternative QR factorization, use rotations instead of reflections for building Q.

7.3.2 Singular Value Decomposition

Theorem 7.19 (SVD decomposition) *For any matrix $A \in \mathbb{K}^{m,n}$ there exist unitary matrices $Q \in \mathbb{K}^{m,m}$, $V \in \mathbb{K}^{n,n}$, and a (generalized) diagonal matrix $\Sigma = \mathrm{diag}(\sigma_1, \dots, \sigma_p) \in \mathbb{R}^{m,n}$ with $p = \min(m, n)$ and $\sigma_1 \geq \sigma_2 \geq \dots \geq \sigma_p \geq 0$, such that*

$$A = U\Sigma V^H.$$

Lemma 7.20 *The squares σ_i^2 of the non-zero singular values of A are the non-zero eigenvalues of $A^K A$ and AA^H with associated eigenvectors $(V)_{:,1}, \dots, (V)_{:,p}$, $(U)_{:,1}, \dots, (U)_{:,p}$, respectively.*

Lemma 7.21 *If for some $1 \leq r \leq p = \min\{m, n\}$, the singular values of $A \in \mathbb{K}^{m,n}$ satisfy $\sigma_1 \geq \sigma_2, \dots, \sigma_r > \sigma_{r+1} = \dots = \sigma_p = 0$, then*

- $\mathrm{rank}(A) = r$, *(Σ encodes rank)*
- $\ker(A) = \mathrm{span}\{(V)_{:,r+1}, \dots, (V)_{:,n}\}$, *($V$ encodes nullspace)*
- $\mathrm{Im}(A) = \mathrm{span}\{(U)_{:,1}, \dots, (V)_{:,r}\}$. *($U$ encodes range)*

Definition 7.22 *The "**numerical rank**" is computed as*

$$r = \#\{|\sigma_i| \geq tol \cdot \max_j\{|\sigma_j|\}\},$$

for some tolerance tol. By default $tol = EPS$.

Complexity *Cost of thin SVD is $\mathcal{O}(\mathbf{mn^2}), m > n$.*

Theorem 7.23 *If $A \in \mathbb{K}^{m,n}$ has the SVD decomposition $A = U\Sigma V^H$ in block matrix form*

$$A = \begin{pmatrix} U_1 & U_2 \end{pmatrix} \begin{pmatrix} \Sigma_r & 0 \\ 0 & 0 \end{pmatrix} \begin{pmatrix} V_1^H \\ V_2^H \end{pmatrix},$$

*then its **Moore-Penrose pseudoinverse** is given by $A^\dagger = V_1 \Sigma_r^{-1} U_1^H$.*

7.3.3 Principal component analysis (PCA)

PCA is used for dimensionality reduction, trend analysis, and data classification. We try to identify and approximate trends in given data.

Find first r singular values that are way larger than the final ones. SVD decomposes an $n \times n$ matrix into r components with the singular value σ_i demonstrating its significant. This is a way to extract entangled and related properties into fewer principal directions with no correlations and highest variances. If data is highly correlated, many σ_i values should be small and can be ignored.
$\Sigma = \frac{XX^T}{n} \quad S^2 = \frac{AA^T}{n-1}$

Chapter 8

Non-Linear Systems of Equations

8.1 Iterative Methods

Iterative Methods use a limiting process to gradually approximate the exact solution of a non-analytically solvable system. They are particularly useful for large sparse systems, especially those arising from the discretization of differential equations.

Consider solving the linear system: $A\mathbf{x} = \mathbf{b}$ *using an iterative method. Start with an initial guess* $\mathbf{x}^{(0)}$*, and generate a sequence* $\{\mathbf{x}^{(k)}\}$ *using an iterative formula of the form:*

$$\mathbf{x}^{(k+1)} = B\mathbf{x}^{(k)} + \mathbf{f}$$

Assume the iteration function has a unique fixed point, i.e., there exists a unique solution $\mathbf{x}^*$ *satisfying:*

$$\mathbf{x}^* = B\mathbf{x}^* + \mathbf{f}$$

Theorem 8.1 (Convergence) *The iterative method converges to the solution* $\mathbf{x}^*$ *for any initial guess* $\mathbf{x}^{(0)}$ *if and only if the spectral radius satisfies:*

$$\rho(B) \leq \|B\| < 1$$

This means all eigenvalues of B lie within the unit circle in the complex plane.

If there exists a matrix norm such that $\|B\| = q < 1$*, then the iterative method converges and satisfies the following error estimates:*

- $\|\mathbf{x}^* - \mathbf{x}^{(k)}\| \leq \frac{q}{1-q}\|\mathbf{x}^{(k)} - \mathbf{x}^{(k-1)}\|$
- $\|\mathbf{x}^* - \mathbf{x}^{(k)}\| \leq \frac{q^k}{1-q}\|\mathbf{x}^{(1)} - \mathbf{x}^{(0)}\|$

The convergence rate is related to the spectral radius $\rho(B)$*. The smaller* $\rho(B)$*, the faster the convergence. The asymptotic convergence rate can be characterized by:*

$$R(B) = -\ln(\rho(B))$$

Basic Conditions for Convergence

(I) $\varphi(x) \in [a, b]$ *for all* $x \in [a, b]$

(II) There exists $L \in [0, 1)$ *such that:*

$$|\varphi'(x)| \leq L < 1 \quad \forall x \in [a, b]$$

Then, starting from any $x_0 \in [a, b]$*, the iteration*

$$x_{k+1} = \varphi(x_k)$$

converges to the unique fixed point x^* *of* $\varphi(x)$ *on* $[a, b]$*.*

Local Convergence

If $\varphi \in C^1[a,b]$ and $|\varphi'(x^*)| < 1$, then there exists $\delta > 0$ such that for all $x_0 \in R = \{x : |x - x^*| \leq \delta\}$, the sequence x_k converges to x^*. This is called ***local convergence****.*

Error Estimates

Let $e_k = |x_k - x^*|$, then:

1. *General error estimate:*
$$|x_k - x^*| \leq \frac{1}{1-L}|x_{k+1} - x_k|$$
2. *One-step bound:*
$$|x^* - x^k| \leq \frac{L^k}{1-L}|x_1 - x_0|$$

Sufficient Condition for Order-p Convergence

Suppose x^* is a fixed point of $\varphi(x)$, and:

$$\varphi^{(j)}(x^*) = 0 \quad \text{for } j = 1, 2, \ldots, p-1, \quad \text{but } \varphi^{(p)}(x^*) \neq 0$$

Then the iteration converges with order $p \geq 2$.

Proof Sketch: *Using Taylor expansion around x^*:*

$$x_{k+1} = \varphi(x_k) = x^* + \frac{\varphi^{(p)}(x^*)}{p!}(x_k - x^*)^p + \text{higher order terms}$$

which implies:

$$e_{k+1} \approx C e_k^p$$

Theorem 8.2 (Konvergenz) *Eine konvergente Folge x_k mit dem Grenzwert x^* hat die Konvergenzordnung p & die Konvergenzrate $C < \infty$, falls gilt:*

$$e_{k+1} = |x_{k+1} - x^*| \leq C|x_k - x^*|^p \quad \forall k \in \mathbb{N}$$

$$p = \frac{\log(e_{k+1}) - \log(e_k)}{\log(e_k) - \log(e_{k-1})} \quad C = \frac{e_{k+1}}{e_k^p}$$

Für $p = 1$ benötigen wir außerdem $C < 1$. Diese Konvergenz wird als linear bezeichnet. Es gilt: $e_k \approx L e_{k-1} \approx L^k e_0 \Rightarrow log(e_k) = k log(L) + log(e_0)$

Abbruchkriterien

1. *After a fix number of steps*
2. *Residual based: stop when $\|f(x^{(k)})\| \leq \tau$,*
3. *Correction based: stop when $\|x^{(k+1)} - x^{(k)}\| \leq \tau$ or $\|x^{(k+1)} - x^{(k)}\| \leq \tau_{rel}\|x^{(k+1)}\|$.*

8.2 Iterative Verfahren

Iterative Methods start with an initial guess of the solution and then repeatedly improve the solution until the change of the solution is below a threshold.

8.2.1 Bisektionsverfahren

Methode zur Nullstellensuche, welche das Intervallhalbierungsprinzip nutzt. Wir betrachten eine 1-dim. stetige Funktion f auf einem Intervall [a,b], s.d. $f(a)f(b) < 0$. *Rekursion mit der Hälfte, in welcher die Nulstelle liegt. Globale lineare Konvergenz* $|e^{(k)}| = |x^{(k)} - x^*| \leq 2^{-k}|a-b|$.

The first approximation is $x_1 = \frac{a+b}{2}$ *with error:* $|x_1 - x^*| \leq \frac{b-a}{2}$ *After k iterations, the error satisfies:* $|x_k - x^*| \leq \frac{b-a}{2^k}$ *To achieve accuracy ε, the number of iterations k must satisfy:* $\frac{b-a}{2^k} < \varepsilon \implies k > \frac{\ln(b-a)-\ln(\varepsilon)}{\ln 2}$

The advantages include it is simple and robust and requires only continuity of $f(x)$ *(no derivatives needed). The disadvantages are that we cannot find complex or multiple roots and the convergence is slow.*

Note: *When using the bisection method, it is helpful to sketch* $f(x)$ *to locate approximate root intervals. Alternatively, divide the interval* $[a,b]$ *into subintervals and apply bisection where* $f(a_k) \cdot f(b_k) < 0$.

8.2.2 Fixpunktiteration

Jede allgemeine Gleichung kann in ein Nullstellenproblem umgeformt werden und weiter zu Fixpunktproblemen überführt werden.

$$x^* = \Phi(x^*) \quad \Phi : \textit{Fixpunktfunktion} \quad x^* : \textit{Fixpunkt}$$

$x_1 = \Phi(x_0) \ldots x_n = \Phi(x_{n-1})$ *Das Verfahren soll zum Fixpunkt konvergieren.* $\lim_{n\to\infty} x_n = x^*$, *wobei* x_0 *der intiale Schätzwert ist. Das Fixpunktproblem ist konsistent mit dem Nullstellenproblem* $f(x^*) = 0$.

If x^* *is a fixed point (FP) of* $\Phi = f(x) + x$, *i.e.* $\Phi(x^*) = x^*$, *then* $f(x^*) = 0$.

Bisection only needed continuity, now we require Lipschitz continuity for Φ *on* $[a,b]$

$$\exists L < 0 \quad \forall x, y \in [a,b] : \quad |\Phi(x) - \Phi(y)| \leq L|x-y|.$$

Suppose Φ *has a fixed point* x^*. *If* $L < 1$, Φ *is called a contractive mapping. This guarantees convergence of the fixed point iteration* $x^{(k)} = \Phi(x^{(k-1)})$ *for some initial guess* $x^{(0)}$ *to* x^*, *because*

$$|x^{(k)} - x^*| \leq L|x^{(k-1)} - x^*| \to 0 \quad \textit{for } k \to \infty.$$

Fixed Point iterations converge at least linearly. Local contractivity is sufficient, but does not guarante global convergence. Also, the initial value $x^{(0)}$ *must be sufficiently close to* x^*.

8.2.3 Newtonverfahren

Suppose x^* *is a root of* $f(x)$ *of multiplicity* $m \geq 2$, *i.e.,*

$$f(x) = (x - x^*)^m g(x), \quad \textit{where } g(x^*) \neq 0$$

The standard Newton iteration:

$$x_{k+1} = x_k - \frac{f(x_k)}{f'(x_k)}$$

converges slowly (only linearly) for multiple roots. To accelerate convergence, use the modified Newton method:

$$x_{k+1} = x_k - m \cdot \frac{f(x_k)}{f'(x_k)}$$

If $f \in C^1$ and $f'(x_k) \neq 0$ in a neighborhood of x^*, then the modified method regains quadratic convergence even for multiple roots. Newton converges quadratically but can diverge for bad initial values. For example, if the derivative is close to 0, then the Newton step will be very large and lead far away from the root.

Line Search Newton Method (Descent Strategy)

To improve robustness of Newton's method, especially when x_{k+1} does not reduce $|f(x)|$, we can introduce a step-size factor $\lambda \in (0, 1]$, also known as a descent factor, and define:

$$x_{k+1} = x_k - \lambda \cdot \frac{f(x_k)}{f'(x_k)}$$

Choose λ such that:

$$|f(x_{k+1})| = \left| f\left(x_k - \lambda \cdot \frac{f(x_k)}{f'(x_k)} \right) \right| < |f(x_k)|$$

This strategy ensures each iteration strictly reduces the function value, improving global convergence properties.

Let $f \in C^1$. Rearranging its Taylor series $f(x) = f(x^{(k)}) + (x - x^{(k)})f'(x^k) = 0$ gives the

Newton formula $x^{(k+1)} = x^{(k)} - \frac{f(x^{(k)})}{f'(x^{(k)})} \quad Df(x^{(k)})\Delta x = -f(x^{(k)})$.

Newton method as a fixed point iteration is $\Phi(x) = x - \frac{f(x)}{f'(x)}$. If $f \in C^2$ and $f'(x^*)) \neq 0$, then the convergence will be quadratic.

In summary we need a sufficiently small neighbourhood I^* of x^* such that $f'(x) \neq 0$ on I^* and $f \in C^2(I^*)$.

Stop if

$$\|x^{(k+1)} - x^{(k)}\| = \|DF(x^{(k)})^{-1}F(x^{(k)})\| \leq \tau\|x^k\| \text{ or } \|DF(x^{(k-1)})^{-1}F(x^{(k)})\| \leq \tau\|x^k\|.$$

Secant Method

Each Newton step requires the computation of $f'(x^{(k)})$, which can be costly or not accessible at all. The Secant method approximates derivatives $f'(x^{(k)}) \approx \frac{f(x^{(k)}) - f(x^{(k-1)})}{x^{(k)} - x^{(k-1)}}$, a Newton step becomes

$$x^{(k+1)} = x^{(k)} - \frac{f(x^{(k)})(x^{(k)} - x^{(k-1)})}{f(x^{(k)}) - f(x^{(k-1)})}.$$

This method's convergence is again local, it needs $f'(x^*) \neq 0$ (simple root), f locally C^2. Its rate is superlinear $p = \frac{1+\sqrt{5}}{2} \approx 1.61$, but not of quadratic order.

Definition 8.3 *An iterative method is a **stationary m-point method** if $x^{(k)}$ depends on m most recent iterates $x^{(k-1)}, \ldots, x^{(k-m)}$, i.e. $x^{(k)} = \Phi_F(x^{(k-1)}, \ldots, x^{(k-m)})$, for solving $F(x) = 0$.*

Secant method is a 2-point method, Newton method is a 1-point method.

Damped Newton Method

We want a large region of convergence. Check in each iteration whether distance $\|x^{(k+1)} - x^{(k)}\|$ is decreasing. If not, don't take a full Newton step, but instead damp the Newton correction by a factor $0 \leq \lambda^{(k)} \leq 1$

$$x^{(k+1)} = x^{(k)} - \lambda^{(k)} \mathrm{D}F(x^{(k)})^{-1} F(x^{(k)}).$$

Choose $\lambda^{(k)}$ maximal such that the distance between iterates is decreasing. In practice, λ is often chosen by backtracking line search or trial values like $\frac{1}{2}, \frac{1}{4}, \ldots$ until the condition is met.

Broyden's Quasi Newton Method

Broyden's quasi-Newton method for solving $F(x) = 0$ is

$$x^{(k+1)} = x^{(k)} + \Delta x^{(k)}, \quad \Delta x^{(k)} = -J_k^{-1} F(x^{(k)}),$$
$$J_{k+1} - J_k = \frac{F(x^{(k+1)})(\Delta x^{(k)})^T}{\|\Delta x^{(k)}\|_2^2}.$$

$J_k - J_{k-1}$ is a rank 1 matrix. Given an initial J_0, we can obtain J_k by rank-1 updates.

*Note: one can use Sherman-Morrison-Woodbury formula **??** to calculate J_k^{-1} from J_{k-1}^{-1}.*

$$\boldsymbol{k}_i^{(0)} = \sum_{j=1}^{i-1} \frac{d_{ij}}{a_{ii}} \boldsymbol{k}_j \Rightarrow (\boldsymbol{I} - h a_{ii} \boldsymbol{J}) \boldsymbol{k}_i = f(\boldsymbol{y}_0 + h \sum_{j=1}^{i-1} (a_{ij} + d_{ij}) \boldsymbol{k}_j) - h \boldsymbol{J} \sum_{j=1}^{i-1} d_{ij} \boldsymbol{k}_j \quad \boldsymbol{J} = Df(\boldsymbol{y}_0 + h \sum_{j=1}^{i-1} (a_{ij} + d_{ij}) \boldsymbol{k}_j)$$

Vereinfacht: $\boldsymbol{J} := Df\boldsymbol{y}_0$

8.3 Unconstrained Optimization

We consider $f : \mathbb{R}^n \to \mathbb{R}$.

Gradient Descent

$\Delta x = -\nabla F(x)$ is the steepest descent/ gradient descent direction $\nabla F(x)^T \Delta x < 0$. If $\nabla F(x) \leq 0$ and $\alpha > 0$ is sufficiently small, then gradient descent guarantees $F(x - \alpha \nabla F(x)) \leq F(x)$. A gradient descent iteration sets $x^{(k+1)} = x^{(k)} - t^{(k)} \nabla F(x^{(k)})$, where finding the step size $t^{(k)}$ is a 1D problem. In each iteration $F(x^{(k)})$ decreases, and the algorithm terminates when $\nabla F(x^{(k)}) \approx 0$.

```
start with initial guess x^(0)
while stopping criterion not satisfied (e.g. while ‖∇F‖_2 > tol)
  take g^(k)(t) = F(x^(k) − t∇F(x^(k)))
  find step size t* through line search, e.g. t* = argmin_{t≥0} g^(k)(t)
  take x^(k+1) = x^(k) − t*∇F(x^(k))
```

The step size t^* can be found through a

***line search**. Search for exact minimum $t^* = \operatorname{argmin}_{t \geq 0} g^{(k)}(t)$, which is a 1D minimization problem. However, most of the time not worth the effort.*

***backtracking line search**. Estimating the Taylor expansion gives $F(x - t\nabla F(x)) \approx F(x) - t\|\nabla F(x)\|^2 < F(x) - \alpha t \|\nabla F(x)\|^2$ for t small enough and some $\alpha \in (0,1)$. Start with $t = 1$ and fix $\alpha \in (0, \frac{1}{2})$. Now decrease t until $F(x - t\nabla F(x)) < F(x) - \alpha t \|\nabla F(x)\|^2$, iterate until "good decrease" is reached. This guarantees a decrease in F, i.e. $F(x^{(k)}) - F(x^{(k+1)}) > \alpha t \|\nabla F(x^{(k)})\|^2$.*

`Initialize` $t = 1,\ \alpha \in (0, \frac{1}{2}),\ \beta \in (0,1)$
`while` $F(x - t\nabla F(x)) > F(x) - \alpha t \|\nabla F(x)\|^2$
  $t \leftarrow \beta t$

Newton's Method

If F is twice differentiable, differentiating and setting the right-hand size of its Taylor expansion to zero

$$F(x) \approx F(x^{(k)}) + \nabla F(x^{(k)})(x - x^{(k)}) + \frac{1}{2}(x - x^{(k)})^T H_F(x^{(k)})(x - x^{(k)}),$$

i.e. the minimum of quadratic approximation, suggests

$$x^{(k+1)} = x^{(k)} - (H_F(x^{(k)}))^{-1}\nabla F(x^{(k)})$$

Intuitively, Newton's method is faster because it "knows" more about the function, because it approximates up to second order terms. Near minimum, its convergence is quadratic and therefore faster than gradient descent with linear convergence.

Comparison of Newton's Method and Gradient Descent

In each iteration, Gradient descent computes a line search, Newton's method computes H_F and solves an LSE. Newton's method requires fewer iterations to converge, Gradient descent typically converges on a larger region. Both can get stuck at local minima or saddle points.

BFGS method

Instead of computing and solving the Hessian $H_F(x^{(k)})$, approximate by B_k such that B_{k+1} is obtained from simple updates of B_k. This method is quasi Newton.

Newton's method computes $x^{(k+1)} - x^{(k)} = -(H_F(x^{(k)}))^{-1}\nabla F(x^{(k)})$. We approximate $H_F(x^{(k)})$ as B_k using a secant-like condition as for Broyden's method

$$B_{k+1}s^{(k)} = B_{k+1}(x^{(k+1)} - x^{(k)}) = \nabla F(x^{(k+1)}) - \nabla F(x^{(k)}) = y^{(k)}.e$$

However, B_{k+1} needs to be s.p.d (symmetric positive definite). Using a rank 2 update $B_{k+1} = B_k + \alpha uu^T + \beta vv^T$ with

$$u = y^{(k)}, \quad v = B_k s^{(k)}, \quad \alpha = \frac{1}{(y^{(k)})^T s^{(k)}}, \quad \beta = -\frac{1}{(s^{(k)})^T B_k s^{(k)}},$$

the BFGS update and its inverse using the Shermann-Morrison Woodbury formula become

$$B_{k+1} = B_k + \frac{y^{(k)}(y^{(k)})^T}{(y^{(k)})^T s^{(k)}} - \frac{B_k s^{(k)}(s^{(k)})^T B_k^T}{(s^{(k)})^T B_k s^{(k)}},$$
$$B_{k+1}^{-1} = \left(I - \frac{s^{(k)}(y^{(k)})^T}{(y^{(k)})^T s^{(k)}}\right) B_k^{-1} \left(I - \frac{y^{(k)}(s^{(k)})^T}{(s^{(k)})^T s^{(k)}}\right) + \frac{s^{(k)}(s^{(k)})^T}{(y^{(k)})^T s^{(k)}}$$

L-BFGS does not require the storage of dense matrix B_k.

Chapter 9

Eigenwertverfahren

Theorem 9.1 (Schur-Zerlegung)

$$\forall A \in \mathbb{K}^{n,n} : \exists U \in \mathbb{C}^{n,n} \text{ unitär} : U^H A U = T \text{ mit } T \in \mathbb{C}^{n,n} \text{ obere Dreiecksmatrix}$$

Die Diagonaleinträge in T sind die Eigenwerte von A. Die Spalten von U bilden eine orthonormale Basis von Eigenvektoren von A. (A normal → A diagonalisierbar)

Beispiel: QR-Algorithmus $O(n^3)$

```
ew, ev = scipy.linalg.eig(A)
ew, ev = scipy.linalg.eigh(A) // symm. / herm. Matrizen
```

Algorithm 1 Basic QR algorithm

Input: A matrix $A \in \mathbb{C}^{n \times n}$
Output: Matrices U and T such that $A = UTU^*$.
  Set $A_0 := A$ and $U_0 = I$
  for $k = 1, \ldots$ **do**
    Compute QR-factorization: $A_{k-1} =: Q_k R_k$
    Set $A_k := R_k Q_k$
    Set $U_k := U_{k-1} Q_k$
  end for
  Return $T = A_\infty$ and $U = U_\infty$

Figure 9.1: Basic QR

Zu viele Schritte für $k \to \infty$. 2 Phasig: Berechne die Hessenbergmatrix H $A = UHU^*$ (finite Anzahl Schritte), dann QR-Algorithmus auf H anwenden $O(n^2)$.

Theorem 9.2 (Potenzmethoden) *Suche nach einem speziellen Eigenwert einer quadratischen Matrix.*

Algorithmus: Wähle $\vec{z}^{(0)}$ zufällig mit $||\vec{z}^{(0)}|| = 1$
for 1,2, ..., maxit:
$\vec{w} = A\vec{z}^{(k-1)}$
$\vec{z}^{(k)} = \frac{\vec{w}}{||\vec{w}||}$
$\lambda^k = (\vec{z}^{(k)})^* A \vec{z}^{(k)}$

1. *Direkte Potenzmethode: A diagonalisierbar. $\frac{A^k \vec{z}}{||A^k \vec{z}||} \to$ Eigenraum von A für $k \to \infty$, der zum betragsgrössten Eigenwert gehört.*
2. *Inverse Iteration: A regulär, finde den betragskleinsten EW. Ersetze A durch A^{-1}.*

3. *Shifted Inverse Iteration: Finde EW nahe gegebenem* $\alpha \in \mathbb{C}$.

   $Ax = \lambda x \Leftrightarrow (A - \alpha I)x = (\lambda - \alpha)x$

   Idee: Verwende inverse Iteration auf $(A - \alpha I)$ *an, da der nächste EW zu a der betragskleinste EW von* $(A - \alpha I)$ *ist.*

   Beachte: $(A - \alpha I)$ *nur invertierbar, falls* α *kein EW ist.*

Theorem 9.3 (Krylov-Verfahren) *Seien* $A \in \mathbb{C}^{n \times n}, \vec{z} \in \mathbb{C}^n / \{0\}, l \in \mathbb{N}$. *Definiere den l-ten Krylov-Unterraum:*

$$K_l(A, \vec{z}) = span\{\vec{z}\}$$

1. *Baue ONB mit Arnoldi-Verfahren*
2. *Berechne* H_l
3. *QR-Algorithmus auf* H_l