Numerical Methods for CSE

ETH Zurich

Janik Schuettler

HS17

Contents

Chapter 1

Computing with matrices and vectors

1.1 Numerics and Error analysis

Computers can't compute in $\mathbb{R}$ or $\mathbb{C}$, instead in $\mathbb{M}$, which is not closed under arithmetic operations. Given a mapping $op : \mathbb{M} \times \mathbb{M} \to \mathbb{R}$, the implementation on a computer is as follows $\tilde{op} : \mathbb{M} \times \mathbb{M} \to \mathbb{M} : \tilde{op} = rd \circ op$. In $\mathbb{M}$, two variables a, b are called equal iff $|a - b| < \varepsilon$, where ε is the machine precision.

Fixed point representation

Pro: Easy way of storing numbers, e.g. $a + b = (a \cdot 10^k + b \cdot 10^k)10^{-k}$.

Con: Precision issues

Floating point representation (default)

Definition 1.1 *Given a basis $B \in \mathbb{N} \setminus \{1\}$, an exponent range $\{e_{min}, ..., e_{max}\} \subset \mathbb{Z}$, and number $m \in \mathbb{N}$ of digits, the corresponding set of machine numbers is*

$$\mathbb{M} = \{d \cdot B^E | d = i \cdot B^{-m},\ i = B^{m-1}, \dots, B^m - 1,\ E \in \{e_{\min}, \dots, e_{\max}\}\}.$$

There are 5 basic formats for floating point representation.

- 3 binary formats: binary32 (simple), binary64 (double), binary128 (quadruple),
- 2 decimal formats: decimal64 (double), decimal128 (quadruple).

Machine numbers are not evenly spread, gaps increase for larger numbers.

Definition 1.2 *For $\in \mathbb{K}$ 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

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

Definition 1.3 (Axiom of roundoff analysis) *There is a small positive number EPS, the machine precision, such that for the elementary arithmetic operations $* \in \{+, -, \cdot, /\}$ and "hard-wired" functions $f \in \{\exp, \sin, \cos, \log, ...\}$ holds*

$$x \tilde{*} y = (x * y)(1 + \delta),$$
$$\tilde{f}(x) = f(x)(1 + \delta),$$

for all $x, y \in \mathbb{M}$ *with* $|\delta| < EPS$. *Alternatively, EPS is the smallest possible positive number, such that* $1 \tilde{+} EPS \neq 1$.

Forward and backward error

Relative and absolute errors are in general not computable without knowing the exact solution. One possible computation is the worst-case estimate. Suppose we want to solve $Ax_{ex} = b$ for x, which gives the approximation x_{app}, such that $Ax_{app} = b_{app}$.

- The **forward error** is defined as $|x_{ex} - x_{app}|$,
- the **backward error** is defined as $|b - b_{app}|$.

In practice we stop an approximation if the backward error is small. However, a small backward error does not imply small forward error.

Dense Matrix Storage Formats

Matrices are stored as either row major (Python) or column major (MATLAB, Eigen) arrays, indexing starts at 0. In Eigen it is therefore faster to access columns.

1.2 Computational effort and Cancellation

Computational effort is not runtime.

Definition 1.4 (Asymptotic complexity) *The asymptotic complexity of an algorithm characterises the worst-case dependence of its computational effort on one or more problem size parameter(s) when these tend to infinity.*

Implicit assumption: sharpness of $\mathcal{O}$-bound, where sharpness means valid and provable. Asymptotic complexity does not predict runtime, but the dependence of runtime on size of the problem.

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	$mk(n-1)$	$\mathcal{O}(mnk)$

Some tricks to reduce complexity

Try to avoid operations of higher complexity, e.g. use vector products instead of matrix products if possible. Examples: matrix multiplication, hidden summation, Kronecker product

1.3 Cancellation

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).*

1.4 Numerical stability

Given a problem $F : X \to Y$, we want to find an algorithm $\tilde{F} : X \to \tilde{Y} \subset \mathbb{M}$ to approximate F.

Definition 1.6 (Stability) *An algorithm $\tilde{F}$ for solving a problem $F : X \to Y$ is called **numerically (backward) stable** if for all $x \in X$ its results $\tilde{F}(x)$ is the exact result for "slightly perturbed" data,*

$$\exists C \approx 1 : \forall x \in X, \exists \tilde{x} \in X : \|x - \tilde{x}\|_X \leq Cw(x)EPS\|x\|_X \wedge \tilde{F}(x) = F(\tilde{x}),$$

*where $w(x)$ is the computational effort of the problem. $\tilde{F}$ is called **mixed stable** if an $\tilde{x}$ with $\frac{||x-\tilde{x}||}{||x||} \leq \mathcal{O}(EPS)$ exists such that*

$$\frac{\|\tilde{F}(x) - F(\tilde{x})\|}{\|F(\tilde{x})\|} \leq \mathcal{O}(EPS).$$

Backward stability implies mixed stability.

Definition 1.7 *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**.*

If the problem is well-conditioned, backward stability guarantees accurate results.

Chapter 2

Direct Methods for Solving LSE

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.

2.1 Solving LSE

Gauss elimination

The idea is to use $Ax = b \iff TA = Tb$ for $T \in \mathbb{K}^{n,n}$ regular.

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

LU decomposition

The idea is to decompose A such that $LU = PA$, with L lower and U upper triangular and P a permutation matrix. Then solve $Ax = LUx = Lz = b$ for z and $Ux = z$ for x.

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* $\mathcal{O}(\mathbf{n^3 + Nn^2})$ *over* $\mathcal{O}(Nn^3)$ *using Gauss elimination.*

2.2 Exploiting structure when solving LSE

Block elimination

The idea is to rewrite

$$\begin{pmatrix} A_{11} & A_{12} \\ A_{21} & A_{22} \end{pmatrix} \begin{pmatrix} x_1 \\ x_2 \end{pmatrix} = \begin{pmatrix} b_1 \\ b_2 \end{pmatrix} \iff \begin{pmatrix} Id & 0 \\ 0 & Id \end{pmatrix} \begin{pmatrix} x_1 \\ x_2 \end{pmatrix} = \begin{pmatrix} A_{11}^{-1}(b_1 - A_{12}x_2) \\ S^{-1}b_s \end{pmatrix},$$

with the **Schur complement** $S = -A_{22} - A_{21}A_{11}^{-1}A_{12}$ and $b_s = b_2 - A_{21}A_{11}^{-1}b_1$.

Good algorithm if A_{11}^{-1} can be easily computed. However, block elimination can suffer from numerical instability. As a rule of thumb, block elimination is numerically stable for s.p.d. matrices and for diagonally dominant matrices.

Block LU decomposition

Example 2.1 *We can decompose a block matrix A into matrices L, U as*

$$A = \begin{pmatrix} R & v \\ u^T & 0 \end{pmatrix} = \begin{pmatrix} Id & 0 \\ u^T R^{-1} & 1 \end{pmatrix} \begin{pmatrix} R & v \\ 0 & -u^T R^{-1} \end{pmatrix} = LU.$$

Solving $Lz = b$ gives

$$Lz = \begin{pmatrix} Id & 0 \\ u^T R^{-1} & 1 \end{pmatrix} \begin{pmatrix} z_a \\ z_b \end{pmatrix} = \begin{pmatrix} z_0 \\ u^T R^{-1} z_a + z_b \end{pmatrix} = \begin{pmatrix} b_a \\ b_b \end{pmatrix} = b.$$

Therefore $z_a = b_a$ and $z_b = b_b - u^T R^{-1} z_a$.
Solving $Ux = z$ gives

$$Ux = \begin{pmatrix} R & v \\ 0 & -u^T R^{-1} v \end{pmatrix} \begin{pmatrix} x_a \\ x_b \end{pmatrix} = \begin{pmatrix} Rx_a + vx_b \\ -u^T R^{-1} v z_b \end{pmatrix} = \begin{pmatrix} z_a \\ z_b \end{pmatrix} = z.$$

If A is regular, then its Schur complement is non-zero, i.e. $u^T R^{-1} \neq 0$, and therefore $x_b = -\frac{x_b}{u^T R^{-1} v}$. Subsequently, x_a can be obtained by solving the LSE $Rx_a = z_a - vx_b$.

Low-rank modification of an LSE

Having solved $Ax = b$, then solve $\tilde{A}x = \tilde{b}$ with $\text{rank}(A - \tilde{A})$ small. Rank-1 modifications can be decomposed into vectors $\tilde{A} = A + uv^T$. The problem can be rewritten into the block partitioned system

$$\begin{pmatrix} A & u \\ v^T & -1 \end{pmatrix} \begin{pmatrix} \tilde{x} \\ \xi \end{pmatrix} = \begin{pmatrix} b \\ 0 \end{pmatrix}.$$

By using block elimination, we get $\tilde{x} = A^{-1}(b - u\xi)$ and $\xi = \frac{v^T A^{-1} b}{v^T A^{-1} u + 1}$ and hence

$$\tilde{x} = A^{-1}b - A^{-1}u \frac{v^T A^{-1} b}{1 + v^T A^{-1} u},$$

where we have to solve for $A^{-1}b$ and $A^{-1}u$.

Complexity *Knowing the LU decomposition of A, solving for $\tilde{x}$ has complexity $\mathcal{O}(\mathbf{n^2})$.*

The following lemma generalizes this for rank-k perturbations.

Lemma 2.2 (Sherman-Morrison-Woodbury formula) *For regular $A \in \mathbb{K}^{n,n}$ and $U, V \in \mathbb{K}^{n,k}$, $k \leq n$ holds*

$$(A + UV^H)^{-1} = A^{-1} - A^{-1}U(Id + V^H A^{-1} U)^{-1} V^H A^{-1},$$

if $Id + V^H A^{-1} U$ is regular.

Note: System $Id + V^H A^{-1} U$ is $k \times k$ and therefore small if k is small. If $c(Id + V^H A^{-1} U) \leq c(A) \cdot (A + UV^H)$ and the original and perturbed system are well-conditioned, then also the $k \times k$ system is.

Complexity *The factorization of UV^H is of complexity $\mathcal{O}(\mathbf{nk^2})$ if $\tilde{A} - A$ has only a few nonzero columns (or rows).*

2.3 Sparse linear Systems

An $m \times n$-matrix A is called sparse if the number of non-zero entries $\text{nnz}(A) \ll mn$.

2.3.1 Sparse Matrix storage formats

Our goal is to reduce the required memory to around the order of $\text{nnz}(A)$.

Triplet matrices

Matrix stored as array of triples containing position i, j and a value. Format allows multiple elements in array at same position in matrix. The convention is that the values of these entries are added up.

CRS/CCS format

Matrix A stored as three contiguous arrays.

- val: stores values
- col_ind: stores the column indices
- row_ptr: stores the row pointer

Complexity *The cost of inserting a new element into matrix A stored in CRS/CCS format is* $\mathcal{O}(\mathbf{nnz}(\mathbf{A}))$.

Ways to efficiently initialize a sparsematrix:

- Use triplet format for initialization and then change to CRS/CCS format
- "Reverse" enough space in each row for nonzero entries.

Complexity *The cost of efficient sparse initialization is* $\mathcal{O}(\mathbf{n})$ *if* $\mathbf{nnz}(A) = \mathcal{O}(n)$.

2.3.2 Direct solutions of sparse LSE

Solvers like SparseLU exploit the sparsity of matrices.

Complexity *The cost of sparse solvers roughly between* $\mathcal{O}(\mathbf{nnz}(\mathbf{A})^{\mathbf{3/2}})$ *and* $\mathcal{O}(\mathbf{nnz}(\mathbf{A})^{\mathbf{5/2}})$.

Chapter 3

Direct Methods for solving Least Square Problems

We want to estimate parameter. 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.

3.1 Least square solutions

Definition 3.1 *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 3.2 *For any $A \in \mathbb{K}^{m,n}$, $b \in \mathbb{K}^m$ a least squares solution of $Ax = b$ exists.*

Lemma 3.3 *For any matrix $A \in \mathbb{K}^{m,n}$ holds*

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

Theorem 3.4 (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 $x \in \text{lsq}(A,b) \iff Ax$ is closest element in $\text{Im}(A)$ to b $\iff Ax - b \in \text{Im}(A)^\perp = \ker(A^H) \iff A^T(Ax-b) = 0$ □

Theorem 3.5 *For $Ax = b$, $A \in \mathbb{K}^{m,n}$, $m \geq n$, holds*

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

Corollary 3.6 *If $m \geq n$ and $\ker(A) = \{0\}$, 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*

$$x = (A^TA)^{-1}A^Tb$$

that can be obtained by solving the normal equations.

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

3.1.1 General solution and Moore-Penrose generalized inverse

Definition 3.7 *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*

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

Theorem 3.8 *The generalized solution $x^\dagger$ is unique.*

Theorem 3.9 *Given $A \in \mathbb{K}^{m,n}$, $b \in \mathbb{K}^m$, the generalized solution $x^\dagger$ is defined by*

$$x^\dagger = V(V^TA^TAV)^{-1}(V^TA^Tb),$$

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

3.2 Normal equation methods

Algorithm using normal equation to solve full-rank least squares problem $Ax = b$

1. Compute regular matrix $C = A^TA$ $\mathcal{O}(mn^2)$
2. Compute RHS $c = A^Tb$ $\mathcal{O}(nm)$
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^TA, i.e. $\text{c}_{A^TA} = \text{c}_a^2$. It can happen that A^TA is not regular in $\mathbb{M}$ even if A is regular in $\mathbb{M}$. Also, if A is sparse, A^TA is not necessarily sparse. Therefore, be careful when using normal equations.

Extended normal equation

Extended normal equations maintain sparsity and we can insert a regularization term for better conditioning.

$$A^HAx = A^Hb \iff B\begin{pmatrix} r \\ x \end{pmatrix} = \begin{pmatrix} -Id & A \\ A^H & 0 \end{pmatrix}\begin{pmatrix} r \\ x \end{pmatrix} = \begin{pmatrix} b \\ 0 \end{pmatrix}$$

If A is sparse, then B is also sparse, but conditioning is not improved.

More generally, we set $r = \alpha^{-1}(Ax - b)$ for some choice of parameter $\alpha > 0$.

$$A^H Ax = A^H b \iff B_\alpha \begin{pmatrix} r \\ x \end{pmatrix} = \begin{pmatrix} -\alpha Id & A \\ A^H & 0 \end{pmatrix} \begin{pmatrix} r \\ x \end{pmatrix} = \begin{pmatrix} b \\ 0 \end{pmatrix}$$

Good choice of α can give better $c_{B_{ff}}$, hopefully $c_{B_{ff}} \approx c_A$.

3.3 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 $\text{lsq}(\tilde{A}, \tilde{b}) = \text{lsq}(A, b)$.

Theorem 3.10 *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 $c_R = c_A$.

3.3.1 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 $A = QR$.

Theorem 3.11 (QR-decomposition) *For any matrix $A \in \mathbb{K}^{n,k}$ with $\text{rank}(A) = k$ 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.

Corollary 3.12 *The economical QR-factorization of $A \in \mathbb{K}^{m,n}$, $m \geq n$, with $\text{rank}(A) = n$ is unique if we demand $(R_0)_{i,i} > 0$.*

Gram-Schmidt orthogonalization suffers from numerical instabilities due to possible cancellation in subtraction and dividing by ≈ 0.

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.

Normal equations vs orthogonal tranfsformation method

For least squares problems

use	orthogonal methods	expanded normal equations
if	$A \in \mathbb{R}^{m,n}$ dense and n small	$A \in \mathbb{R}^{m,n}$ dense and m, n big
because	Superior numerical stability of orthogonal transformations methods	SVD/QR-factorization cannot exploit sparsity

3.4 Singular Value Decomposition

Theorem 3.13 (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 3.14 *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 3.15 *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*

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

Definition 3.16 *The "**numerical rank**" is computed as*

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

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

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

JacobiSVD is numerically stable.

3.4.1 Generalized solutions by SVD

Theorem 3.17 *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 $A^\dagger = V_1 \Sigma_r^{-1} U_1^H$.*

3.4.2 SVD-based optimization & approximation

Norm constrained extrema

Given $A \in \mathbb{K}^{m,n}, m \geq n$, find $x \in \mathbb{K}^n, \|x\|_2 = 1$, such that $\|Ax\|_2 \to \min$.

Minimum for $x = Ve_n = V_{:,n}$.

Best low-rank approximation

Definition 3.18 *The **Frobenius norm** (or matrix norm) $\|\cdot\|_F$ of $A \in \mathbb{K}^{m,n}$ is defined as*

$$\|A\|_F^2 = \sum_{i=1}^{m} \sum_{j=1}^{n} |a_{i,j}|^2 = \sum_{j=1}^{r} \sigma_j^2,$$

with $\sigma_i, 1 \leq i \leq r, r = rank(A)$, A's singular values.

Given $A \in \mathbb{K}^{m,n}$, find a matrix $\tilde{A} \in \mathbb{K}^{m,n}$, $\operatorname{rank}(\tilde{A}) \leq k$, such that $\|A - \tilde{A}\|_{2/F} \to \min$ over rank-k matrices.

Theorem 3.19 (Best k-rank approximation) *Let $A = U\Sigma V^H$ be the SVD of $A \in \mathbb{K}^{m,n}$. For $1 \leq k \leq \operatorname{rank}(A)$ set $U_k = (u_{:,1} \ldots u_{:,k}) \in \mathbb{K}^{m,k}$, $V_k = (v_{:,1} \ldots v_{:,k}) \in \mathbb{K}^{n,k}$, $\Sigma_k = \operatorname{diag}(\sigma_1, \ldots, \sigma_k) \in \mathbb{K}^{k,k}$. Then for $\|\cdot\| = \|\cdot\|_F, \|\cdot\|_2$ holds true*

$$\|A - U_k \Sigma_k V_k^H\| \leq \|A - F\| \quad \forall F \in R_k(m,n),$$

i.e. $U_k \Sigma_k V_k^H$ is the best k-rank approximation to A.

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. TODO: what does that mean mathematically

Find first p singular values that are way larger than the final ones.

3.5 Total least squares

For least squares problems $Ax = b$, we now also allow A to be perturbed.

Given $A \in \mathbb{K}^{m,n}, m > n,\ \text{rank}(A) = n,\ b \in \mathbb{K}^m$, find $\hat{A} \in \mathbb{K}^{m,n},\ \hat{b} \in \mathbb{K}^m$ with

$$\|(A \quad b) - (\hat{A} \quad \hat{b})\|_F \to \min \text{ and } \hat{b} \in \text{Im}(\hat{A}),$$

which means the perturbed system should be solvable. If $(A\ b)$'s SVD is given by

$$\begin{pmatrix} A & b \end{pmatrix} = U\Sigma V^T = \sum_{j=1}^{n+1} \sigma_j (U)_{:,j} (V)_{:,j}^T,$$

then a solution to the total least squares problem is given by

$$x = \hat{A}^{-1}\hat{b} = -\frac{1}{(V)_{n+1,n+1}} (V)_{1:n,n+1}$$

if $(V)_{n+1,n+1} \neq 0$, otherwise no small perturbation can make $\hat{b} \in \text{Im}(\hat{A})$, i.e. the system solvable.

3.6 Constrained least squares

Given

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

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

Solution via Lagrange multiplier (a saddle point problem)

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

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.

Chapter 4

Filtering Algorithms

Signal processing: time-discrete signals as vectors/sequences $x_j = X(j\Delta t)$ through sampling of time-continuous signal $X(t)$, $t \in [0,T]$, where $\mathbf{x} = (x_0,\dots,x_{n-1})^T \in \mathbb{R}^n$ and $n\,\Delta t \leq T$. More generally $l^\infty(\mathbb{Z})$ as vector space of bounded signals.

4.1 Discrete Convolutions

We consider finite, linear, time-invariant causal filters. Mathematically, filters are mappings $\mathcal{F} : l^\infty(\mathbb{Z}) \to l^\infty(\mathbb{Z}) : F((x_j)_{j\in\mathbb{Z}}) = (y_i)_{i\in\mathbb{Z}}$. $\mathcal{F}$ is

- finite: every finite-length signal $(x_j)_{j\in\mathbb{Z}}$ produces finite-length output $\mathcal{F}((x_j)_{j\in\mathbb{Z}})$,
- time-invariant: $\mathcal{F}$ commutes with shift operator, i.e. time-shifted input + filter $\hat{=}$ apply filters + time-shifted output. The time shift operator $S_m : l^\infty(\mathbb{Z}) \to l^\infty(\mathbb{Z}) : S_m((x_j)_{j\in\mathbb{Z}}) = (x_{j-m})_{j\in\mathbb{Z}}$, $m \in \mathbb{Z}$. Mathematically, time-invariance is $\mathcal{F}(S_m((x_j)_{j\in\mathbb{Z}})) = S_m(\mathcal{F}((x_j)_{j\in\mathbb{Z}}))$,
- linear: for all $(x_j)_{j\in\mathbb{Z}}$, $(y_j)_{j\in\mathbb{Z}} \in l^\infty(\mathbb{Z})$, $\alpha,\beta \in \mathbb{R}$ it holds true that $\mathcal{F}(\alpha(x_j)_{j\in\mathbb{Z}} + \beta(y_j)_{j\in\mathbb{Z}}) = \alpha\mathcal{F}((x_j)_{j\in\mathbb{Z}}) + \beta\mathcal{F}((y_j)_{j\in\mathbb{Z}})$,
- causal: output only depends on past and present inputs, not on the future. If $x_j = 0\ \forall j \leq M \implies \mathcal{F}((x_j)_{j\in\mathbb{Z}})_k = 0\ \forall k \leq M$.

Definition 4.1 (Impulse response) *The **impulse response** of channel/filter is the output for a single unit pulse at $t = 0$ as input, i.e. the input signal is $x_j = \delta_{j,0}$.*

We write FIR filters for Finite impulse response and LT-FIR for finite time-invariant linear causal filters. An impulse response has n-th order for $(\dots,0,h_0,h_1,\dots,h_{n-1},0,\dots)$, $n \in \mathbb{N}$. Any finite signal $(x_j)_{j\in\mathbb{Z}}$ is a linear combination of shifted pulses.

$$\mathcal{F}((x_j)_{j\in\mathbb{Z}}) = \mathcal{F}(\sum_{k=0}^{m-1} x_k S_k((\delta_{j,0})_{j\in\mathbb{Z}})) = \sum_{k=0}^{m-1} x_k S_k(\mathcal{F}(\delta_{j,0})_{j\in\mathbb{Z}})) = (y_j)_{j\in\mathbb{Z}}$$

$$\implies y_k = \mathcal{F}((x_j)_{j\in\mathbb{Z}}) = \sum_{j=0}^{m-1} x_j h_{k-j}$$

where $k = 0,\dots,m+n-2$, $h_j = 0$ for $j < 0, j \geq n$. The maximal duration of output is $(m+n-2)\Delta t$ (length of filter + length of signal).

Finite length signals $(\dots,0,x_0,\dots,x_{m-1},0,\dots)$ can be represented by a vector $(x_0,\dots,x_{m-1})^T$ and a filter by a linear mapping $\mathcal{F} : \mathbb{R}^m \to \mathbb{R}^{m+n+1}$, i.e. a matrix.

Definition 4.2 (Discrete convolution) *For two sequences $\mathbf{f},\mathbf{g} \in l^\infty(\mathbb{Z})$ their **discrete convolution** $\mathbf{u} = \mathbf{f} * \mathbf{g} \in l^\infty$ is defined by*

$$u_k = \sum_{j\in\mathbb{Z}} f_j\, g_{k-j} = \sum_{j\in\mathbb{Z}} f_{k-j}\, g_j.$$

The discrete convolution for vectors, i.e. finite length sequences, is defined equivalently. Take $\mathbf{y} = \mathcal{F}((x_j)_{j\in\mathbb{Z}}) = \mathbf{x} * \mathbf{h} \in l^\infty$, then for $\mathbf{y} \in \mathbb{R}^{m+n-1}$ as a vector $y_k = \sum_{j=0}^{m-1} x_j h_{k-j}$, where $h_j = 0$ for $j < 0$ and $j \geq n$.

Filtering an m-periodic signal $x_j = x_{j+m}$ for all $j \in \mathbb{Z}$ with $p_j = \sum_{j\in\mathbb{Z}} h_{j+lm}$ motivates the following definition as a special case of discrete convolution.

Definition 4.3 (Discrete periodic convolution) *The **discrete periodic convolution** of two n-periodic sequences $(p_j)_{j\in\mathbb{Z}}, (x_j)_{j\in\mathbb{Z}}$ yields the n-periodic sequence*

$$(y_k) = (p_k) *_n (x_k), \quad y_k = \sum_{j=0}^{n-1} p_{k-j} x_j = \sum_{j=0}^{n-1} x_{k-j} p_j, \quad k \in \mathbb{Z}.$$

Definition 4.4 (Circulant matrix) *A matrix $\mathbf{C} \in \mathbb{K}^{n,n}$ is **circulant** $\iff$ $\exists (p_k)_{k\in\mathbb{Z}}$ n-periodic sequence with $C_{i,j} = p_{j-i}$, $1 \leq i, j \leq n$.*

A circulant matrix is represented by a vector $\mathbf{p} = (p_0, \ldots, p_{n-1})^T$, discrete periodic convolution by multiplication with a circulant matrix

$$\begin{pmatrix} y_0 \\ \vdots \\ y_{m-1} \end{pmatrix} = \begin{pmatrix} p_0 & p_{n-1} & \cdots & p_1 \\ p_1 & p_0 & & \vdots \\ \vdots & \vdots & \ddots & p_{n-1} \\ p_{n-1} & p_{n-2} & \cdots & p_1 \end{pmatrix} \begin{pmatrix} x_0 \\ \vdots \\ x_{m-1} \end{pmatrix}.$$

Discrete convolution can be reduced to discrete periodic convolution by zero-padding $\mathbf{x}$ to $\mathbf{x}^L$ with length $L = m + n - 1$. Then for $\mathbf{y}^L = \mathbf{x}^L *_L \mathbf{h}^L$ it follows that $y_k^L = y_k$ for all $0 \leq k \leq L$. For $\mathbf{x} \in \mathbb{R}^m$, $\mathbf{h} \in \mathbb{R}^n$, the output vector will be $\mathbf{y} = \mathbf{x} * \mathbf{h} \in \mathbb{R}^L$, which can be represented as $\mathbf{A}\mathbf{x} = \mathbf{y}$, where $\mathbf{A} \in \mathbb{R}^{L,m}$ and

$$\mathbf{x} * \mathbf{h} = \mathbf{x}^L *_L \mathbf{h}^L = \mathbf{y} = \begin{pmatrix} y_0 \\ y_1 \\ \\ \vdots \\ \\ y_{n+m-2} \end{pmatrix} = \begin{pmatrix} h_0 & 0 & \cdots & 0 \\ \vdots & h_0 & & \vdots \\ h_{n-1} & \vdots & & 0 \\ 0 & h_{n-1} & \ddots & h_0 \\ \vdots & \vdots & \ddots & \vdots \\ 0 & 0 & \cdots & h_{n-1} \end{pmatrix} \begin{pmatrix} x_0 \\ \vdots \\ x_{m-1} \end{pmatrix} = \mathbf{A}\mathbf{x}.$$

4.2 Discrete Fourier Transform (DFT)

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

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

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

Lemma 4.6 (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 4.7 (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 4.8 (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

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

4.2.1 Discrete Convolution via DFT

Theorem 4.9 (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.*

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

4.2.2 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* $\mathcal{O}(\mathbf{n}\log\mathbf{n})$.

4.2.3 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.

4.2.4 2D DFT

Given a matrix $\mathbf{Y} \in \mathbb{C}^{m,n}$, its 2D DFT is defined as 2 nested 1D DFTs, i.e.

$$(\mathbf{C})_{k_1,k_2} = \sum_{j_1=0}^{m-1} \sum_{j_2=0}^{n-1} y_{j_1,j_2} \omega_m^{j_1 k_1} \omega_n^{j_2 k_2} = \sum_{j_1=0}^{m-1} \omega_m^{j_1 k_1} \left(\sum_{j_2=0}^{n-1} \omega_n^{j_2 k_2} y_{j_1,j_2} \right), \quad 0 \le k_1 < m, 0 \le k_2 < n.$$

Theorem 4.10 (2D DFT) *The 2D DFT and 2D inverse DFT are given by*

$$\mathbf{C} = \mathbf{F}_m (\mathbf{F}_n \mathbf{Y}^T)^T = \mathbf{F}_m \mathbf{Y} \mathbf{F}_n,$$
$$\mathbf{Y} = \mathbf{F}_m^{-1} \mathbf{C} \mathbf{F}_n^{-1} = \frac{1}{mn} \overline{\mathbf{F}}_m \mathbf{C} \overline{\mathbf{F}}_n.$$

Filtering with 2D DFT is analogous to 1D filtering. 2D discrete convolution is reducible to 2D discrete periodic (circular) convolution.

Theorem 4.11 (2D Convolution Theorem) *Let $\mathbf{U}, \mathbf{X} \in \mathbb{C}^{m,n}$ and let the 2D discrete convolution $\mathbf{U} *_{m,n} \mathbf{X}$ be defined by*

$$(\mathbf{U} *_{m,n} \mathbf{X})_{k,l} = \sum_{i=0}^{m-1} \sum_{j=0}^{n-1} (U)_{i,j} (X)_{i',j'},$$

where $i' = (k-i) \mod m$, $j' = (l-j) \mod n$. Then

$$\mathbf{U} *_{m,n} \mathbf{X} = \frac{1}{mn} \overline{\mathbf{F}}_m \left((\mathbf{F}_m \mathbf{U} \mathbf{F}_n)_{i,j} \cdot (\mathbf{F}_m \mathbf{X} \mathbf{F}_n)_{i,j} \right)_{i=0,\ldots,m-1, j=0,\ldots,n-1} \overline{\mathbf{F}}_n.$$

The theorem states that

$$\mathbf{U} *_{m,n} \mathbf{X} = \mathrm{DFT2} \left((\mathrm{DFT2}(\mathbf{U}))_{i,j} \cdot (\mathrm{DFT2}(\mathbf{X})_{i,j}) \right)_{i=0,\ldots,m-1, j=0,\ldots,n-1}.$$

Chapter 5

Data Interpolation in 1D

5.1 Abstract Interpolation

Given a set of data points $(t_i, y_i) \in \mathbb{R}^2,\ i = 0, \dots, n,\ t_i \in I \subset \mathbb{R}$. Find an interpolant, i.e. a function $f : I \to \mathbb{R}$, such that $f \in C^0(I)$ and $f(t_i) = y_i,\ \forall i = 0, \dots, n$. 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 $(m+1)$-dimensional subspace. The function f is then represented as $f(t) = \sum_{i=0}^{m-1} \alpha_i b_j(t)$ with the basis vectors $b_0, \dots, b_{m-1}$ and coefficients $\alpha_0, \dots, \alpha_{m-1}$.

Interpolation assumes sufficiently accurate measurements, otherwise we would use data fitting.

5.2 Piecewise linear Interpolation

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

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

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 $\{b_i\}_{j=0}^n$ depend on points t_i. The interpolant then is given by

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

More general setting

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 $(m+1) \times (n+1)$ linear system of equations

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

where $\mathbf{A}$ is called the Vandermande matrix.

Existenz and uniqueness of interpolant depends on regularity of $\mathbf{A}$, we therefore require $m = n$. Then, invertibility of $\mathbf{A}$ depends on nodes t_i and space S, but not on the choice of basis $\{b_i\}_{j=0}^n$.

Note that a cardinal basis yields $\mathbf{A} = \mathbf{I}$.

5.3 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.

Advantages of using polynomials are

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

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

5.3.1 Lagrange Polynomials

Definition 5.1 (Lagrange Polynomials) *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 $L_i \in P_n$, $L_i(t_j) = \delta_{i,j}$, linear independent and form a cardinal basis.

Lagrange interpolation

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

Lemma 5.2 (Existence and uniqueness) *The general Lagrange polynomial interpolation problem admits a unique solution $p \in P_n$.*

Corollary 5.3 *The polynomial interpolation in the nodes $\mathcal{T} = \{t_j\}_{j=0}^n$ defines a linear operator*

$$I_{\mathcal{T}} : \mathbb{R}^{n+1} \to \mathcal{P}_n : (y_0, \dots, y_n)^T \mapsto \textit{interpolating polynomial } p$$

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 5.4 *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.*

5.3.2 Barycentric interpolation approach

Introduce a factor λ_i, $i = 0, \dots, n$ such that

$$p(t) = \sum_{i=0}^{n} y_i \frac{\lambda_i}{t - t_i} \prod_{j=0}^{n} (t - t_j), \quad \lambda_i = \frac{1}{(t_i - t_0)(t_i - t_1) \dots (t_i - t_{i-1})(t_i - t_{i+1}) \dots (t_i - t_n)}.$$

Setting $p_1(t) = 1$, we can solve for $p(t)$ to get the

Barycentric interpolation

$$p(t) = \frac{\sum_{i=0}^{n} y_i \frac{\lambda_i}{t - t_i}}{\sum_{i=0}^{n} \frac{\lambda_i}{t - t_i}}.$$

Complexity *Evaluating N different data value sets using a Barycentric interpolation formula comprises the following computational costs:*

- *Computing* $\lambda_0, \dots, \lambda_n$ *is* $\mathcal{O}(\mathbf{n^2})$
- *Evaluating* $p(x_k)$ *for each k is* $\mathcal{O}(\mathbf{n})$

The total computational complexity is $\mathcal{O}(\mathbf{n^2 + nN})$.

If nodes are close to each other, numerical instabilities might worsen the computation of λ_i and Lagrange polynomials.

5.3.3 Newton basis

Definition 5.5 (Newton Polynomials) *We define the Newton polynomials as*

$$N_0(t) = 1, \quad N_i(t) = \prod_{j=0}^{i-1} (t - t_j).$$

Note that $\{N_0, \dots, N_n\}$ are linearly independent. Since $N_i(t_l) = 0$ for all $l < i$, Newton interpolation can be represented by a lower triangular Vandermande matrix $\mathbf{A}$

$$\begin{pmatrix} 1 & 0 & \dots & \dots & 0 \\ 1 & N_1(t_1) & 0 & \dots & 0 \\ 1 & N_1(t_2) & N_2(t_2) & \ddots & \vdots \\ \vdots & \vdots & \vdots & \ddots & 0 \\ 1 & N_1(t_n) & N_2(t_n) & \dots & N_n(t_n) \end{pmatrix} \begin{pmatrix} a_0 \\ \vdots \\ a_n \end{pmatrix} = \begin{pmatrix} y_0 \\ \vdots \\ y_n \end{pmatrix}.$$

Lagrange interpolation

$$p(t) = \sum_{i=0}^{n} a_i L_i(t)$$

Complexity *The effort for building the system Vandermande matrix for a Newton basis is* $\mathcal{O}(\mathbf{n^2})$. *Solving* $\boldsymbol{A\alpha} = \boldsymbol{y}$ *for* $\boldsymbol{\alpha}$ *is of complexity* $\mathcal{O}(\mathbf{n^2})$ *using forward substitution. For N evaluations, this results in an overall complexity of* $\mathcal{O}(\mathbf{n^2 N})$.

Note that algebraically, Vandermande, Lagrange, and Newton interpolation are equivalent and therefore the respective interpolants unique. Higher degree does not result in a better approximation.

Runge's phenomenon

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

5.3.4 Update friendly schemes

Aitken-Neville scheme

We define partial interpolating polynomials $p_{k,l}$ as the unique interpolating polynomial of degree $l-k$ through $(t_k, y_k), \ldots, (t_l, y_l)$, where $p_{k,k}(x) = y_k$ and

$$\begin{aligned} p_{k,l}(x) =& \frac{(x-t_k)p_{k+1,l}(x) - (x-t_l)p_{k,l-1}(x)}{t_l - t_k} \\ =& p_{k+1,l}(x) + \frac{x-t_l}{t_l-t_k}(p_{k+1,l}(x) - p_{k,l-1}(x)), \quad 0 \le k \le l \le n. \end{aligned}$$

The Aitken-Neville scheme then computes $p_{0,n}$ recursively as visualized here

n	0	1	2	3
t_0	$y_0 = p_{0,0}(x)$	$p_{0,1}(x)$	$p_{0,2}(x)$	$p_{0,3}(x)$
t_1	$y_1 = p_{1,1}(x)$	$p_{1,2}(x)$	$p_{1,3}(x)$	
t_2	$y_2 = p_{2,2}(x)$	$p_{2,3}(x)$		
t_3	$y_3 = p_{3,3}(x)$			

This is update friendly as a new data value (t_{n+1}, y_{n+1}) can be simply added to the above scheme and the updated interpolant can be computed partially from already computed results.

Complexity *The effort for evaluation of $p_{0,n}$ at point x using the Aitken-Neville scheme is $\mathcal{O}(\mathbf{n^2})$.*

Divided differences

Update-friendly version using the Newton basis. We set

$$y[t_i] = y_i, \quad y[t_i, \ldots, t_{i+k}] = \frac{y[t_{i+1}, \ldots, t_{i+k}] - y[t_i, \ldots, t_{i+k-1}]}{t_{i+k} - t_i},$$

which visually gives the following recursive calculation

$$\begin{array}{llll} t_0 & y[t_0] & & \\ & & y[t_0,t_1] & \\ t_1 & y[t_1] & & y[t_0,t_1,t_2] \\ & & y[t_1,t_2] & & y[t_0,t_1,t_2,t_3] \\ t_2 & y[t_2] & & y[t_1,t_2,t_3] \\ & & y[t_2,t_3] & \\ t_3 & y[t_3] & & \end{array}$$

The interpolant is then given by

$$p(t) = a_0 + a_1(t-t_0) + a_2(t-t_0)(t-t_1) + \ldots + a_n \prod_{j=0}^{n-1}(t-t_j),$$

where $a_0 = y[t_0]$, $a_1 = y[t_0, t_1]$, $a_2 = y[t_0, t_1, t_2]$, ... This can be evaluated in a Horner scheme

$$(t - t_0)(\dots((t - t_{n-2})((t - t_{n-1})a_n + a_{n-1}) + a_{n-2}) + \dots + a_1) + a_0.$$

Complexity *Using divided differences, the effort to update the scheme after adding a new data point is $\mathcal{O}(\mathbf{n})$, evaluating $p(t)$ at given point x using Horner scheme is $\mathcal{O}(\mathbf{n})$.*

5.4 Splines

Piecwise polynomial interpolation

- on each subinterval $[t_{i-1}, t_i]$ polynomial of degree d
- matching of first $d-1$ derivatives at nodes t_i

Definition 5.6 (Spline space) *Given an interval $I = [a, b] \subset \mathbb{R}$ and a mesh $\mathcal{M} = \{a = t_0 < t_1 < \dots < t_{n-1} < t_n\}$, the vector space $\mathcal{S}_{d,\mathcal{M}}$ of the spline functions of degree d (i.e. order $d+1$) is defined by*

$$\mathcal{S}_{d,\mathcal{M}} = \{s \in C^{d-1}(I) : s_j = s|_{[t_{j-1}, t_j]} \in \mathcal{P}_d \ \forall j = 1, \dots, n\}.$$

Note that for $s \in \mathcal{S}_{d,\mathcal{M}}$, it is $s' \in \mathcal{S}_{d-1,\mathcal{M}}$ and $\int_a^b s(t)\mathrm{d}t \in s \in \mathcal{S}$. We have $d+1$ degrees of freedom on n intervals with d constraints on each interior point, in total we get $\dim(\mathcal{S}_{d,\mathcal{M}}) = n(d+1) - (n-1)d = n + d$.

5.4.1 Cubic spline interpolation

We now consider $\mathcal{S}_{3,\mathcal{M}}$, i.e. $s_j(t) = a_j + b_j t + c_j t^2 + d_j t^3$. The $4n$ coefficients are determined by

- $2n$ interpolating conditions $s_j(t_{j-1}) = y_{j-1}$ and $s_j(t_j) = y_j$,
- $n-1$ smoothness conditions for first derivatives $s_j'(t_j) = s_{j+1}'(t_j)$,
- $n-1$ smoothness conditions for second derivatives $s_j''(t_j) = s_{j+1}''(t_j)$,
- 2 more conditions, for example natural cubic spline interpolation $s''(t_0) = s''(t_n) = 0$

Economical implementation

Let $s_j(t) = a_j + b_j(t - t_{j-1}) + c_j(t - t_{j-1})^2 + d_j(t - t_{j-1})^3$ for all $j = 1, \dots, n$. Then the coefficients are given by

$$a_j = y_{j-1}, \quad b_j = \frac{y_j - y_{j-1}}{h_j} - \frac{h_j(2\sigma_{j-1} + \sigma_j)}{6}, \quad c_j = \frac{\sigma_{j-1}}{2}, \quad d_j = \frac{\sigma_j - \sigma_{j-1}}{6h_j},$$

where $h_j = t_j - t_{j-1}$ and $\sigma_1, \dots, \sigma_{n-1}$ are given by the $(n-1) \times (n-1)$ linear system of equations with right hand side $r_j = \frac{y_{j+1} - y_j}{h_{j+1}} - \frac{y_j - y_{j-1}}{h_j}$

$$\begin{pmatrix} \frac{h_1+h_2}{3} & \frac{h_2}{6} & 0 & \dots & 0 \\ \frac{h_2}{6} & \frac{h_2+k_3}{3} & \frac{h_3}{6} & \dots & 0 \\ \vdots & \ddots & \ddots & & \vdots \\ \vdots & & \ddots & \ddots & \frac{h_{n-1}}{6} \\ 0 & \dots & 0 & \frac{h_{n-1}}{6} & \frac{h_{n-1}+h_n}{3} \end{pmatrix} \begin{pmatrix} \sigma_1 \\ \vdots \\ \vdots \\ \vdots \\ \sigma_{n-1} \end{pmatrix} = \begin{pmatrix} r_1 \\ \vdots \\ \vdots \\ \vdots \\ r_{n-1} \end{pmatrix}.$$

Chapter 6

Approximation of Functions in 1D

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.

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 $\|f - \tilde{f}\|_p$.

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

6.1 Taylor Approximation

Any function $f \in C^k(I)$ can be approximated by 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.

6.2 Bernstein Approximation

Taylor polynomials yield local approximation of sufficiently smooth functions. However, we would like to have a uniform approximation on I without smoothness requirements (functions are merely continuous).

Definition 6.1 *We define the **Bernstein polynomials** as*

$$B_j^n(t) = \binom{n}{j} t^j (1 - t)^{n-j}.$$

Theorem 6.2 (Uniform approximation by polynomial) *For $f \in C^0([0,1])$ we define the **n-th Bernstein approximant** as*

$$p_n(t) = \sum_{j=0}^{n} f(j/n) B_j^n(t) = \sum_{j=0}^{n} f(j/n) \binom{n}{j} t^j (1-t)^{n-j} \quad p_n \in P_n.$$

It satisfies $\|f - p_n\|_\infty \to 0$ for $n \to \infty$. If $f \in C^m([0,1])$, then $\|f^{(k)} - p_n^{(k)}\|_\infty \to 0$ for $n \to \infty$ and $0 \le k \le m$.

B_j^n satisfy $\sum_{j=0}^n B_j^n(t) \equiv 1$, $0 \le B_j^n(t) \le 1$ for all $0 \le t \le 1$.

This approximation in $\|\cdot\|_\infty$ is uniform. However, convergence is slow (see proof in the lecture notes).

6.3 Global polynomial Approximation Theory

We first require a notion of a best approximation error.

Definition 6.3 *Let $\|\cdot\|$ be a (semi-)norm on a space X of functions $I \to \mathbb{K}$, $I \subset \mathbb{R}$ an interval. The (size of the) **best approximation error** of $f \in X$ in the space P_k of polynomials of degree $\le k$ with respect to $\|\cdot\|$ is*

$$\operatorname{dist}_{\|\cdot\|}(f, P_k) = \inf_{p \in P_k} \|f - p\|.$$

The best possible L^∞-approximation is given by

Theorem 6.4 (Jackson's theorem) *If $f \in C^r([-1,1])$, $r \in \mathbb{N}$, then for any polynomial degree $n \le r$*

$$\inf_{p \in P_n} \|f - p\|_{L^\infty([-1,1])} \le \left(1 + \frac{\pi^2}{2}\right)^r \frac{(n-r)!}{n!} \|f^{(r)}\|_{L^\infty([-1,1])} = \mathcal{O}(n^{-r}).$$

The norm $\inf_{p \in P_n} \|f - p\|_{L^\infty([-1,1])}$ is the norm of the best approximation error, it always exists (P_n is finite-dimensional) and is a uniform approximation. Note that this estimate depends on smoothness of f.

Lemma 6.5 *The Stirling approximating states that $n! \sim \sqrt{2\pi n} \left(\frac{n}{e}\right)^n$.*

Sterling's formula reduces $\frac{(n-r)!}{n!} \le C(r) n^{-r}$, which is algebraic convergence.

Lemma 6.6 *If $\Phi^* : C^0([a,b]) \to C^0([-1,1]) : \Phi^*(f)(\hat{t}) = f(\Phi(\hat{t}))$, is an affine pullback based on*

$$\Phi : [-1,1] \to [a,b] \ , \ \Phi(\hat{t}) = a + \frac{1}{2}(\hat{t}+1)(b-a), \quad -1 \le \hat{t} \le 1,$$

then $\Phi^* : \mathcal{P}_n \to \mathcal{P}_n$ is a bijective linear mapping for any $n \in \mathbb{N}_0$.

Lemma 6.7 (Transformation of norms) *For every $f \in C^0([a,b])$ we have*

$$\|f\|_{L^\infty([a,b])} = \|\Phi^* f\|_{L^\infty([-1,1])}, \qquad \|f\|_{L^2([a,b])} = \sqrt{\frac{|b-a|}{2}} \|\Phi^* f\|_{L^2([-1,1])}.$$

Proof Use the affine pullback in the L^p integral. □

If A is approximation scheme for $f \in C^0([a,b])$

$$\|f - A\|_{L^\infty([a,b])} = \|\Phi^* f - \hat{A}(\Phi^* f)\|_{L^\infty([-1,1])},$$
$$\|f - A\|_{L^2([a,b])} = \sqrt{\frac{|b-a|}{2}} \|\Phi^* f - \hat{A}(\Phi^* f)\|_{L^2([-1,1])}.$$

Using this approximation scheme and the chain rule for $\|f^{(r)}\|_{L^\infty([a,b])}$, we can express Jackson's theorem for arbitrary intervals, i.e. the best polynomial approximation on $C^0([a,b])$, as

$$\inf_{p \in P_n} \|f - p\|_{L^\infty([a,b])} \leq \left(1 + \frac{\pi^2}{2}\right)^r \frac{(n-r)!}{n!} \left(\frac{b-a}{2}\right)^r \|f^{(r)}\|_{L^\infty([a,b])}.$$

For a family of polynomial approximation schemes $\{A_n\}_{n \in \mathbb{N}}$, we would like to address the question of whether we can bound the interpolation error as the number of nodes is increased, i.e. $\|f - A_n f\| \leq T(n)$ for $n \to \infty$, for different families $\mathcal{T}_n = \{t_0^{(n)}, \ldots, t_n^{(n)}\}$ of nodes

$$\mathcal{T}_n = \{t_j^{(n)} = a + (b-a)\frac{j}{n}, j = 0, \ldots, n\} \subset I.$$

6.4 Lagrange Approximation

We first consider Lagrange interpolation with equidistant nodes.

Definition 6.8 *Given an interval $I \subset \mathbb{R}, n \in \mathbb{N}$, a node set $\mathcal{T} = \{t_0, \ldots, t_n\}$, the **Lagrangian (interpolation polynomial) approximation scheme** $L_\mathcal{T} : C^0(I) \to P_n$ is defined by*

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

Definition 6.9 *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 6.10 (Representation of interpolation error) *We consider $f \in C^{n+1}(I)$ and the Lagrangian interpolation approximation scheme for a node set $\mathcal{T} = \{t_0, \ldots, t_n\} \subset I$. Then, for every $t \in I$ there exists a $\mathcal{T}_t \in (min\{t, t_0, \ldots, t_n\}, \max\{t, t_0, \ldots, t_n\})$ 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 $\mathcal{T} = \{t_0, \ldots, t_n\} \subset I$ using Lagrangian interpolation, this results in the following estimates

$$\|f - L_\mathcal{T} f\|_{L^\infty(I)} \leq \frac{\|f^{(n+1)}\|_{L^\infty(I)}}{(n+1)!} \max_{t \in I} |(t - t_0) \ldots (t - t_n)|, \tag{6.1}$$

$$\|f - L_\mathcal{T} f\|_{L^2(I)} \leq \frac{2^{(n-1)/4} |I|^{n+1}}{\sqrt{n!(n+!)!}} \|f^{(n+1)}\|_{L^2(I)}. \tag{6.2}$$

Example 6.11 *Using this new global estimate, we are able to understand the following examples as they demonstrate quite different error behavior.*

- $g(t) = \sin(t) \in C^\infty$ *has exponential convergence* $\mathcal{O}(q^n)$*, simple Lagrange interpolation is doing much better than predicted by Jackson's theorem. Using the global estimate we get* $\|g - L_\mathcal{T} f\|_{L^\infty(I)} \leq \frac{1}{n+1} \left(\frac{\pi}{n}\right)^{n+1}$*, which is indeed exponential.*
- $f(t) = \frac{1}{1+t^2}$ *shows bad behavior on edges, i.e. Runge's phenomenon. Using the global estimate we get* $\|f - L_\mathcal{T} f\|_{L^\infty(I)} \leq n! \left(\frac{20}{n}\right)^n \frac{10}{n}$*, which by Stirling is exponential growing and therefore does not guarantee to convergence.*

6.5 Chebychev Approximation

Since it is not possible to control the error of $\|f^{(n+1)}\|_{L^\infty}$, we would like to find nodes $t_0, \ldots, t_n$ such that $\|\omega\|_{L^\infty(I)}$ is minimal. Because $\omega(t) = \prod_{j=0}^n (t - t_j) \in \mathcal{P}_{n+1}$, it is equivalent to searching for $q \in \mathcal{P}_{n+1}$ with minimal $\|q\|_{L^\infty(I)}$, where q has $n+1$ zeros in I.

Definition 6.12 (Chebychev Polynomials) *The n-th* ***Chebychev polynomial*** *is*

$$T_n(t) = \cos(n \arccos(t)) \quad \text{for} -1 \leq t \leq 1,\ n \in \mathbb{N}.$$

The zeros of T_n *are called the* ***Chebychev nodes*** *and given by* $t_j = \cos\left(\frac{2j+1}{2n}\pi\right)$ *for* $j = 1, \ldots, n$ *and the leading coefficient of* T_n *is* 2^{n-1}*.*

Theorem 6.13 *Chebychev polynomials satisfy the 3-term recursion for all* $n \in \mathbb{N}$

$$T_{n+1}(t) = 2tT_n(t) - T_{n-1}(t), \quad T_1(t) = t, \quad T_0(t) = 1.$$

Theorem 6.14 *The Chebychev polynomials minimize the supremum norm in the following sense*

$$\|T_n\|_{L^\infty([-1,1])} = \inf\{\|p\|_{L^\infty([-1,1])} : p \in \mathcal{P}_n,\ p(t) = 2^{n-1}t^n + \ldots\}, \quad \forall n \in \mathbb{N}.$$

The nodal polynomial $\omega(t) = \prod_{j=0}^n (t - t_j)$ with minimal $\|\omega\|_{L^\infty(I)}$ is given by

$$\omega(t) = 2^{-n} T_{n+1}(t),$$

where t_j are the Chebychev nodes of T_{n+1}. Chebychev nodes are equidistant on circle, but projected onto the interval $[-1, 1]$ more dense at edges. The interpolation error with this choice of $\omega(t)$, $\|\omega\|_{L^\infty([-1,1])} = 2^{-n}$, is

$$\|f - I_\mathcal{T}(f)\|_{L^\infty([-1,1])} \leq \frac{2^{-n}}{(n+1)!} \|f^{(n+1)}\|_{L^\infty([-1,1])}$$

Arbitrary interval $[a, b]$

The Chebychev nodes in the interval $[a, b]$ are

$$t_k = a + \frac{1}{2}(b-a)\left(\cos\left(\frac{2k+1}{2(n+1)}\pi\right) + 1\right), \quad k = 0, \ldots, n,$$

the interpolation error for interval $[a, b]$ is

$$\|f - I_\mathcal{T}(f)\|_{L^\infty([a,b])} \leq \frac{2^{-2n-1}}{(n+1)!} |I|^{n+1} \|f^{(n+1)}\|_{L^\infty([a,b])}.$$

Note that this is a much better estimate than for equidistant nodes. However, if $\|f^{(n)}\|_{L^\infty([-1,1])}$ grows too fast it is still possible for the right hand side to diverge.

Example 6.15 *For* $\frac{1}{1+t^2}$*, which has* $\|f^{(n+1)}\|_{L^\infty([-5,5])} \sim 2^{n+1}(n+1)!$*, the right hand side becomes* 105^n*.*

Lebesgue constant

We would like to have an estimate such that $\|f^{(n)}\|_{L^\infty([-1,1])}$ is independent of the number of nodes as in Jackson's theorem.

Definition 6.16 (Lebesgue constant) *Given a node set $\mathcal{T} = \{t_0, \dots, t_n\}$ and approximation scheme $I_\mathcal{T} : \mathbb{R}^{n+1} \to \mathcal{P}_n : I_\mathcal{T}(f(t_0), \dots, f(t_n)) = L_\mathcal{T} f$, the Lebesgue constant is defined as*

$$\lambda_\mathcal{T} = \sup_{y \in \mathbb{R}^{n+1}} \frac{\|I_\mathcal{T} y\|_\infty}{\|y\|_\infty}.$$

The Lebesgue constant is a quality measure for polynomial interpolation scheme, i.e. the norm of the operator $I_\mathcal{T}$ in L^∞ sense. For $L_\mathcal{T} : C^0(I) \to \mathcal{P}_n : f \mapsto L_\mathcal{T} f = I_\mathcal{T}(f(t_0), \dots, f(t_n))$ it follows

$$\|L_\mathcal{T} f\|_{L^\infty(I)} \leq \lambda_\mathcal{T} \|f\|_{L^\infty(I)}.$$

For all $f \in C^0(I)$ and $\lambda_\mathcal{T} = \max_{t \in I} \sum_{j=0}^n |L_j(t)|$, it follows that the interpolation error is at most $(1 + \lambda_\mathcal{T})$ worse than the best approximation error for all f

$$\|f - L_\mathcal{T} f\|_{L^\infty(I)} \leq (1 + \lambda_\mathcal{T}) \inf_{p \in \mathcal{P}_n} \|f - p\|_{L^\infty(I)}.$$

This links the best approximation error with the interpolation error from Jackson's theorem 6.4. In approximation theory, this result is a special case of

Theorem 6.17 (Lebesgue's lemma) *Given a normed vector space $(X, \|\cdot\|)$, $U \subset X$ subspace, and $P : X \to U$ a linear projection onto U. Then*

$$\|x - Px\| \leq (1 + \|P\|) \inf_{u \in X} \|x - u\|, \quad \forall x \in X.$$

The Lebesgue constant for different type of nodes can be estimated as

- Equidistant nodes: $\lambda_\mathcal{T} \geq C e^{n/2}$ (at least exponential growth),
- Chebychev nodes: $\lambda_\mathcal{T} \leq \frac{2}{\pi} \log(n+1) + 1$ (at most log growth).

For $f \in C^r([-1,1])$, combining the general estimate with Jackson's theorem gives

$$\|f - L_\mathcal{T} f\|_{L^\infty([-1,1])} \leq \underbrace{\left(\frac{2}{\pi} \log(1+n) + 2\right)}_{1+\lambda_\mathcal{T}} \left(1 + \frac{\pi^2}{2}\right)^r \frac{(n-r)!}{n!} \|f^{(r)}\|_{L^\infty([-1,1])}.$$

Implementation

Instead of sampling a given function on the Chebychev nodes, we write p as a Chebychev expansion $p(t) = \sum_{j=0}^n \alpha_j T_j(t)$ since the Chebychev polynomials form a basis of P_n.

To efficiently evaluate Chebychev polynomials given coefficients α_j, use the 3-term recursion formula 6.13

$$p(x) = \sum_{j=0}^{n-1} \tilde{\alpha}_j T_j(x) \quad \text{with } \tilde{\alpha}_j = \begin{cases} \alpha_j + 2x\alpha_{j+1} & \text{if } j = n-1, \\ \alpha_j - \alpha_{j+2} & \text{if } j = n-2, \\ \alpha_j & \text{else.} \end{cases}$$

To compute the coefficients α_j use the interpolation condition to get

$$p(\cos(2\pi s)) = \sum_{j=-n}^{n} \beta_j \exp(-2\pi i j s), \quad \text{where } \beta_j = \begin{cases} 0 & \text{for } j = n+1, \\ \frac{1}{2}\alpha_j & \text{for } j = 1, \dots, n, \\ \alpha_0 & \text{for } j = 0, \\ \frac{1}{2}\alpha_{n-j} & \text{for } j = -n, \dots, -1. \end{cases}$$

Using some further transformations it follows that

$$\sum_{j=0}^{2n+1} \beta_{j-n} \exp\left(-\frac{\pi i (j-n)}{2(n+1)}\right) \omega_{2(n+1)}^{jk} = \exp\left(-\frac{\pi i n k}{n+1}\right) z_k,$$

$$\text{with } \mathbf{z}_k = \begin{cases} \mathbf{y}_k = f(t_k) & k = 0, \dots, n \\ \mathbf{y}_{2n+1-k} & k = n+1, \dots, 2n+1 \end{cases}.$$

Using

$$\mathbf{c}_k = \beta_{j-n} \exp\left(-\frac{\pi i (k-n)}{2(n+1)}\right), \quad \mathbf{b}_k = \mathbf{z}_k \exp\left(-\frac{\pi i n k}{n+1}\right),$$

the system can be solved via inverse DFT $\mathbf{c} = F_{2(n+1)}^{-1}\mathbf{b}$ to recover β_j from $\mathbf{c}$ and α_j from β_j.

Comlexity *The cost of recursively evaluating a Chebychev expansion is $\mathcal{O}(\mathbf{n})$. The cost of calculating coefficients α_j for a Chebychev expansion is $\mathcal{O}(\mathbf{n} \log \mathbf{n})$.*

6.6 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 $j = 1, \dots, m$.
2. Choose set of local interpolation nodes

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

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

   $$s_j = s|_{I_j} \in \mathcal{P}_{n_j} \quad \text{and} \quad s_j(t_i^j) = f(t_i^j) \quad i = 0, \dots, n_j,\ j = 1, \dots, m. \tag{6.3}$$

Corollary 6.18 (Continuous local Lagrange Interpolants) *If the local degrees n_j are at least 1 and the interpolation nodes t_k^j, $j = 1, \dots, m$, $k = 0, \dots, n_j$, for local Lagrange interpolation satisfy*

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

then the piecewise polynomial Lagrange interpolant according to 6.3 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 6.1 on each subinterval gives the overall estimate with algebraic convergence of rate $n+1$

$$\|f-s\|_{L^\infty([x_0,x_m])} \leq h_{\mathcal{M}}^{n+1} \frac{1}{(n+1)!} \|f^{(n+1)}\|_{L^\infty([x_0,x_m])},$$
$$\|f-s\|_{L^2(I)} \leq h_{\mathcal{M}}^{n+1} \frac{2^{(n-1)/4}}{\sqrt{n!(n+1)!}} \|f^{(n+1)}\|_{L^2([x_0,x_m])}.$$

Remark 6.19 *Note that*

- *n is now fixed (and small), e.g. for piecewise linear $n = 1$ and estimate holds if $f|_{I_j} \in C^2$,*
- *piecewise smoothness of f is sufficient,*
- *since n can be small, convergence result also for f with low regularity,*
- *slow convergence.*

Similar estimate is possible for cubic spline interpolation. For equidistant mesh with mesh width h, we get an error estimate with algebraic convergence in h

$$\|f-s\|_{L^\infty([t_0,t_n])} \leq \frac{5}{384} h^4 \|f^{(4)}\|_{L^\infty([t_0,t_n])}, \quad f \in C^4([t_0,t_n]).$$

6.7 Overview of estimates

Jackson's theorem

If $f \in C^r([-1,1])$, $r \in \mathbb{N}$, then for any polynomial degree $n \leq r$

$$\inf_{p \in P_n} \|f-p\|_{L^\infty([-1,1])} \leq \left(1+\frac{\pi^2}{2}\right)^r \frac{(n-r)!}{n!} \|f^{(r)}\|_{L^\infty([-1,1])} = \mathcal{O}(n^{-r}).$$

$$\inf_{p \in P_n} \|f-p\|_{L^\infty([a,b])} \leq \left(1+\frac{\pi^2}{2}\right)^r \frac{(n-r)!}{n!} \left(\frac{b-a}{2}\right)^r \|f^{(r)}\|_{L^\infty([a,b])}.$$

Lagrange

For $f \in C^{n+1}(I)$ and equidistant node set $\mathcal{T} = \{t_0, \ldots, t_n\} \subset I$

$$\|f - L_{\mathcal{T}} f\|_{L^\infty(I)} \leq \frac{\|f^{(n+1)}\|_{L^\infty(I)}}{(n+1)!} \max_{t \in I} |(t-t_0)\ldots(t-t_n)|,$$
$$\|f - L_{\mathcal{T}} f\|_{L^2(I)} \leq \frac{2^{(n-1)/4}|I|^{n+1}}{\sqrt{n!(n+!)!}} \|f^{(n+1)}\|_{L^2(I)},$$

Chebychev

For $f \in C^{n+1}(I)$, $I = [-1,1], [a,b]$ and node set $\mathcal{T} = \{t_0, \ldots, t_n\} \subset I$

$$\|f - I_{\mathcal{T}}(f)\|_{L^\infty([-1,1])} \leq \frac{2^{-n}}{(n+1)!} \|f^{(n+1)}\|_{L^\infty([-1,1])},$$
$$\|f - I_{\mathcal{T}}(f)\|_{L^\infty([a,b])} \leq \frac{2^{-2n-1}}{(n+1)!} |I|^{n+1} \|f^{(n+1)}\|_{L^\infty([a,b])}.$$

Lebesgue constant

$$\|f - L_{\mathcal{T}} f\|_{L^\infty(I)} \leq (1 + \lambda_{\mathcal{T}}) \inf_{p \in \mathcal{P}_n} \|f - p\|_{L^\infty(I)} \quad \forall f \in C^0(I).$$

Chebychev with Lebesgue constant For $f \in C^r([-1,1])$

$$\|f - L_{\mathcal{T}} f\|_{L^\infty([-1,1])} \leq \underbrace{\left(\frac{2}{\pi} \log(1+n) + 2\right)}_{1+\lambda_{\mathcal{T}}} \left(1 + \frac{\pi^2}{2}\right)^r \frac{(n-r)!}{n!} \|f^{(r)}\|_{L^\infty([-1,1])}.$$

Piecewise Lagrange

$$\|f - s\|_{L^\infty([x_0,x_m])} \leq h_{\mathcal{M}}^{n+1} \frac{1}{(n+1)!} \|f^{(n+1)}\|_{L^\infty([x_0,x_m])},$$
$$\|f - s\|_{L^2(I)} \leq h_{\mathcal{M}}^{n+1} \frac{2^{(n-1)/4}}{\sqrt{n!(n+1)!}} \|f^{(n+1)}\|_{L^2([x_0,x_m])}.$$

Chapter 7

Numerical Quadrature

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

7.1 Quadrature Formulas

Definition 7.1 *An n-point **quadrature formula/quadrature rule** on $[a,b]$ provides an approximation of the value of an integral through a weighted sum of points values of the integrand*

$$\int_b^a f(t)\mathrm{d}t \approx Q_n(f) = \sum_{j=1}^n w_j^n f(c_j^n).$$

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

It is sufficient to consider a reference interval $[-1,1]$, because

$$\int_b^a f(t)\mathrm{d}t = \frac{1}{2}(b-a)\int_{-1}^1 \hat{f}(t)\mathrm{d}t$$
$$\hat{f}(t) = f\left(\frac{1}{2}(1-t)a + \frac{1}{2}(t+1)b\right),$$

where $\hat{f}$ is the affine pullback $\Phi^* f$ of f to $[-1,1]$.

Quadrature by approximation schemes

For given linear interpolation scheme $I_\mathcal{T}$ with node set $\mathcal{T} = \{t_1,\ldots,t_n\} \subset [a,b]$, we can approximate

$$\int_a^b f(t)\mathrm{d}t \approx \int_a^b I_\mathcal{T}(f(t_1),\ldots,f(t_n))^T(t)\mathrm{d}t = \sum_{j=1}^n f(t_j)\omega_j^n$$

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

We define the quadrature error as $E_n(f) = |\int_a^b f(t)\mathrm{d}t - Q_n(f)|$. We get

$$E_n(f) \le |b-a| \underbrace{\|f - I_\mathcal{T}(f(t_1),\ldots,f(t_n))^T\|_{L^\infty([a,b])}}_{\text{interpolation error}}$$

7.2 Polynomial Quadrature Formulas

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

For given Lagrange interpolation scheme $I_{\mathcal{T}}$ with node set $\mathcal{T} = \{t_1, \dots, t_n\} \subset [a, b]$, we can approximate the integral Q_n by approximating f with a polynomial p_{n-1}.

$$\int_a^b f(t)\mathrm{d}t \approx \int_a^b p_{n-1}(t)\mathrm{d}t = \sum_{j=1}^{n} f(t_{j-1})\omega_{j-1}^n$$

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

Example 7.2 (Midpoint rule) *Rule for $n = 1$ and $t_0 = \frac{1}{2}(a + b)$*

$$\int_a^b f(t)\mathrm{d}t \approx (b - a)f(\frac{1}{2}(a + b))$$

Example 7.3 (Newton-Cotes formulas) *Lagrange interpolation with equidistant nodes $t_j = a + \frac{b-a}{n-1}j$ with $j = 0, \dots, n - 1$.*

- *$n = 2$ (Trapezoidal rule)* $\int_a^b f(t)\mathrm{d}t \approx \frac{(b-a)}{2}(f(a) + f(b))$
- *$n = 3$ (Simpson rule)* $\int_a^b f(t)\mathrm{d}t \approx \frac{(b-a)}{6}(f(a) + 4f(\frac{a+b}{2}) + f(b))$

However, Lagrange interpolation is numerically unstable for large n. Therefore, use Chebychev interpolation instead, which is the Clenshaw-Curtis QF.

7.3 Gauss Quadrature

Quality measure for QF.

Definition 7.4 *The order of a quadrature rule $Q_n : [a, b] \to \mathbb{R}$ is defined as*

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

that is the maximum degree $+1$ of polynomials for which the quadrature rule is guaranteed to ne 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 7.5 *An n-point quadrature rule on $[a, b]$ $Q_n(f) = \sum_{j=1}^n \omega_j f(t_j)$, $f \in C^0([a, b])$ with nodes $t_j \in [a, b]$ and weights $\omega_j \in \mathbb{R}, j = 1, \dots, n$ is of order $\geq n$ if and only if*

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

with L_j the j-th Lagrange polynomial associated with the ordered node set $\{t_1, \dots, t_n\}$

This means for QF to have order $\leq n$, weights ω_j only depend on node set $\mathcal{T} = \{t_1, \dots, t_n\}$.

Theorem 7.6 *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

- $\bar{P}_n = (t - c_1^n) \dots (t - c_n^n)$, $\bar{P}_n \in P_n$
- $\bar{P}_n \perp P_{n-1}$ in $L^2([-1, 1])$
- $\bar{P}_n$ is unique

- For $\bar{P}_n = t^n + \alpha_{n-1}t^{n-1} + \ldots + \alpha_1 t + \alpha_0$, we have $\sum_{j=0}^{n-1} \alpha_j \int_{-1}^{1} t^l t^j \mathrm{d}t = -\int_{-1}^{1} t^l t^n \mathrm{d}t \iff A\alpha = b$, where A is symmetric and positive definite
- α exists and is unique

Theorem 7.7 *Let $\{\bar{P}_n\}_{n\in\mathbb{N}_0}$ be a family of non-zero 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* $\{c_j^n\}_{j=1}^m, m \leq n$, *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* $\omega_j = \int_a^b L_j(t)\mathrm{d}t, \quad j = 1,\ldots,n,$ *provides a quadrature formula of order* $2n$ *on* $[-1,1]$.

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

Definition 7.8 *The n-th* ***Legendre polynomial*** P_n *is defined by*

- $P_n \in P_n$
- $\int_{-1}^{1} q(t)P_n(t)\mathrm{d}t = 0$ *for all* $q \in P_{n-1}$ *(L^2-orthogonality)*
- $P_n(1) = 1$.

Lemma 7.9 *P_n has n distinct zeros in $(-1,1)$. These zeros are called the Gauss points.*

Definition 7.10 *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 7.5, are called* ***Gauss-Legendre quadrature formulas****.*

Lemma 7.11 *The weights of the Gauss-Legrendre quadrature formulas are positive.*

Recursive formula for Legrendre polynomials

$$P_0(t) = 0, \quad P_1(t) = t, \quad P_{n+1}(t) = \frac{2n+1}{n+1}tP_n(t) - \frac{n}{n+1}P_{n-1}(t)$$

Quadrature error & best approximation error

Theorem 7.12 *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 7.13 *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)
$f \in C^\infty([a,b])$: exponential convergence (lin-log plot))

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

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

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

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

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

7.4 Composite Quadrature

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 $Q^j_{n_j}$

$$\int_a^b f(t)\mathrm{d}t = \sum_{j=1}^{m} \int_{x_{j-1}}^{x_j} f(t)\mathrm{d}t.$$

The error estimate for composite quadrature is

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

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.

Composite trapezoidal has $q = 2$, composite Simpson has $q = 4$. So for smooth functions, error is of $\mathcal{O}(h^2)$ and $\mathcal{O}(h^4)$, respectively.

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.

Chapter 8

Iterative Methods for Non-Linear Systems of Equations

Given a function F, we want to efficiently solve for $f(x) = 0$. Higher order converge faster.

Definition 8.1 (Order of convergence) *A sequence $x^{(k)}$ in $\mathbb{R}^n$ with limit $x^* \in R^n$ converges with order p if*

$$\exists 0 < L : \|x^{(k+1)} - x^*\| \leq L\|x^{(k)} - x^*\|^p, \quad \forall k \in \mathbb{N}.$$

For $p = 1$, we also require $L < 1$. This convergence is called linear.

8.1 1D Iterative Methods

8.1.1 Bisection

Intermediate value theorem yields an easy method for finding the root for continuous functions. Convergence $|e^{(k)}| = |x^{(k)} - x^*| \leq 2^{-k}|a - b|$.

+ robustness, global convergence
- rather slow convergence
- no extension to higher dimensions

Bisection does not converge linear, because $|e^{(k)}| \leq L^k|a - b|$ does not exclude $|e^{(k)}| > L|e^{(k-1)}|$, which is needed for linear convergence.

8.1.2 Fixed Point iterations

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

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

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^*.

8.1.3 Algorithm for root-finding with quadratic convergence

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

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.

Lemma 8.2 *If $\Phi : U \subset \mathbb{R} \to \mathbb{R}$ is $m+1$ times continuously differentiable, $\Phi(x^*) = x^*$ for some x^* in the interior of U, and $\Phi(l)(x^*) = 0$ for $l = 1, ..., m, m \geq 1$, then the fixed point iteration (8.2.2) converges locally to x^* with order greater than $m+1$.*

Remark 8.3 *$x^{(0)} \in I^*$ guarantees $f'(x^{(k)}) \neq 0$ for all the iterates. For quadratic convergence $f \in C^2(I^*)$ is sufficient.*

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

Approximating derivatives

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 8.4 *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.

8.2 Nonlinear systems of equations

Given $F : D \subset \mathbb{R}^n \to \mathbb{R}^n$, find a root x^* such that $F(x^*) = 0$. Aspects of iterative methods

- Convergence: $(x^{(k)})_{k \in \mathbb{N}}$ convergent, $\lim_{k \to \infty} x^{(k)} = x^*$
- Consistency: $\Phi_F(x^*, \ldots, x^*) = x^* \iff F(x^*) = 0$
- Rate and order of convergence $\|x^{(k)} - x^*\| \to 0$

Definition 8.5 *Two norms $\|\cdot\|_a$ and $\|\cdot\|_b$ on a vector space V are **equivalent** if*

$$\exists C_1, C_2 > 0 : C_1 \|v\|_a \leq \|v\|_b \leq C_2 \|v\|_a \quad \forall v \in V$$

This implies that convergence in $\mathbb{R}^n$ (or any finite dimensional vector space) is independent of choice of norm, but in general the convergence rate depends on the chosen norm.

Definition 8.6 (Local and global convergence) *As stationary m-point iterative method **converges locally** to x^* if there is a neighbourhood $U \subset D$ of x^* such that $x^{(0)}, \ldots, x^{(m-1)} \in U$ implies that $x^{(k)}$ is well defined and $\lim_{k \to \infty} x^{(k)} = x^*$, where $(x^{(k)})_{k \in \mathbb{N}_0}$ is the (infinite) sequence of iterates. If $U = D$, the iterative method is **globally convergent**.*

8.2.1 Fixed point iterations in $\mathbb{R}^n$

Definition 8.7 (Consistency of fixed point iterations) *A fixed point iteration $x^{(k+1)} = \Phi(x^{(k)})$ is **consistent** with $F(x) = 0$ if for $x \in U \cap D$*

$$F(x) = 0 \quad \iff \Phi(x) = x.$$

Definition 8.8 (Contractive mapping) *$\Phi : U \in \mathbb{R}^n$ is **contractive** if*

$$\exists L < 1 \quad \|\Phi(x)0\Phi(y)\| \leq L\|x - y\| \quad \forall x, y, \in U$$

Contractivity of Φ implies that if $\Phi(x^*) = x^*$, then a fixed point iteration will converge to x^*. The convergence is at least linear. If Φ is contractive $\implies$ Φ has at most one fix point

Theorem 8.9 (Banach's fixed point theorem) *If $D \subset \mathbb{K}^n$ closed and bounded and $\Phi : D \to D$ satisfies*

$$\exists L < 1 : \quad \|\Phi(x) - \Phi(y)\| \leq L\|x - y\| \quad \forall x, y \in D,$$

then there exists a unique fixed point $x^* \in D, \Phi(x^*) = x^*$, which is the limit of the sequence of iterates $x^{(k+1)} = \Phi(x^{(k)})$ for any $x^{(0)} \in D$.

Convergence criteria for FPI for Φ differentiable and knowing $\Phi(x^*) = x^*$.

Lemma 8.10 *If $\Phi : U \subset \mathbb{R}^n \to \mathbb{R}^n$, $\Phi(x^*) = x^*$, Φ differentiable in x^* and $\|\mathrm{D}\Phi(x^*)\| < 1$, then the fixed point iteration $x^{(k+1)=\Phi(x^{(k)})}$ converges locally and at least linearly.*

Lemma 8.11 *Let U be convex and $\Phi : U \subset \mathbb{R}^n \to \mathbb{R}^n$ be continously differentiable with*

$$L = \sup_{x \in U} \|D\Phi(x)\| < 1.$$

If $\Phi(x^*) = x^*$ for some interior point $x^* \in U$, then the fixed point iteration $x^{(k+1)} = \Phi(x^{(k)})$ converges to x^* at least linearly with rate L.

Locally contractive Φ implies that the iteration converges locally around FP at least linearly.

Termination criterion for contractive FPI

- Residual based: stop when $\|F(x^{(k)})\| \leq \tau$,
- correction based: stop when $\|x^{(k+1)} - x^{(k)}\| \leq \tau$ or $\|x^{(k+1)} - x^{(k)}\| \leq \tau_{\mathrm{rel}}\|x^{(k+1)}\|$.

From discussion about condition number: $\|F(x^{(k)} - F(x^*))\|$ small does not imply that $\|x^{(k)} - x^*\|$ is small.

If iteration is linearly convergent: $\|x^{(k)} - x^*\| \leq (1 - L)\|x^{(k)}\| \leq \|x^{(k+1)} - x^*\|$. This suggests to use

$$\frac{L}{1 - L}\|x^{(k+1)} - x^{(k)}\| \leq \tau$$

as stopping criterion. It guarantees $\|x^{(k+1)} - x^*\| \leq \tau$. However, estimating L can be difficult. But the pessimistic estimate $\tau > L$ is still reliable.

8.3 Newton's method

If $DF(x^{(k)})$ is regular, general Newton's method in $\mathbb{R}^n$ is given by

$$x^{(k+1)} = x^{(k)} - DF(x^{(k)})^{-1}F(x^{(k)}).$$

Theorem 8.4.45 of the lecture notes roughly states: If $F(x^*) = 0$ and $DF(x^*)$ is regular, then it is locally quadratically convergent.

8.3.1 Stopping criterion for Newton's method

Stop if

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

However, if $x^{(k)}$ was a good approximation, we would have computed new Newton correction $DF(x^{(k)})^{-1}F(x^{(k)})$ and not used it in iteration. In practice, one therefore often uses a simplified Newton correction as a cheaper stopping criterion

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

8.3.2 Damped Newton method

Examples of failures of Newton's method

- Local min/max
- Asymptotes for $F(x) = xe^x - 1$
- Overshooting for $F(x) = \arctan(x)$

We want a large region of convergence. Idea for damped Newton method: 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 $0 \leq \lambda^{(k)} \leq 1$

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

Choose $\lambda^{(k)}$ maximal sucht that the distance between iterates is decreasing.

Natural Monotonicity Test NMT *Choose maximal* $0 < \lambda^{(k)} < 1$ *such that*

$$\|DF(x^{(k)})^{-1}F(x^{(k)} + \lambda^{(k)}\Delta x^{(k)})\| \leq \left(1 - \frac{\lambda^{(k)}}{2}\right)\|DF(x^{(k)})^{-1}F(x^{(k)})\|_2.$$

In practice set $\lambda^{(k)} = 1$ and check NMT, repeatedly take $\lambda^{(k)} \leftarrow \frac{\lambda^{(k)}}{2}$ until NMT passes for the first time.

8.3.3 Quasi Newton method

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 2.2 to calculate J_k^{-1} from J_{k-1}^{-1}.

Remark 8.12 *In general, iterative methods for nonlinear systems should have convergence monitor, i.e. a simple check at each iteration whether convergence to be expected or not. Example: NMT for damped Newton. If it fails repeatedly, stop and report error.*

8.4 Unconstrained Optimization

Given $F : \mathbb{R}^n \to \mathbb{R}$, find min/max of F. Optimization problems we have already seen are

- least-squares solution: find $x \in \mathbb{K}^n$ such that $\|Ax - b\|_2 \to \min$,
- generalized solution: find least squares solution x to $Ax = b$ such that $\|x\|_2 \to \min$,
- norm-constrained extrema: given $A \in \mathbb{K}^{m,n}$, $m \geq n$, find $x \in \mathbb{K}^n$, $\|x\|_2 = 1$ such that $\|Ax\|_2 \to \min$,
- best low-rank approximation: given $A \in \mathbb{K}^{m,n}$, find $\tilde{A} \in \mathbb{K}^{m,n}$, $\operatorname{rank}(\tilde{A}) \leq k$ such that $\|A - \tilde{A}\|_{2/F} \to \min$ over rank-k matrices,
- total least squares problem: given $A \in \mathbb{K}^{m,n}$, $m \geq n$, $\operatorname{rank}(A) = n$, $b \in \mathbb{R}^n$, find $\hat{A} \in \mathbb{K}^{m,n}$, $\hat{b} \in \mathbb{R}^m$ such that $\|[Ab] - [\hat{A}\hat{b}]\|_F \to \min$ with $\hat{b} \in \operatorname{Im}(\hat{A})$

We only consider minimization, because maximizing F is equivalent with minimizing $-F$.

Definition 8.13 (Global vs. local minimum) *x^* is a global minimum of $F : \mathbb{R}^n \to \mathbb{R}$ if $F(x^*) \leq F(x)$ $\forall x \in \mathbb{R}^n$, x^* is a local minimum of F if there is $\varepsilon > 0$ such that for all x with $\|x - x^*\| \leq \varepsilon$ $F(x^*) \leq F(x)$.*

An application of optimization techniques to machine learning is the following

Maximum likelihood estimation

Suppose some quantity can be modeled with a probability distribution, we would like to estimate mean μ and variance σ through randomized sample $\{w_1, \ldots, w_n\}$ assuming a normal distribution for f, i.e.

$$f(w, \mu, \sigma) = \frac{1}{\sqrt{2\pi}\sigma} e^{-(w-\mu)^2/(2\sigma^2)},$$

where $f(w_i, \mu, \sigma)$ likelihood to observe w_i for sample i. Value of sample i is independent of value of sample j.

Maximize P

$$P(\{w_1, \ldots, w_n\}, \mu, \sigma) = \prod_{j=1}^{n} f(w_j, \mu, \sigma)$$

as a function in μ, σ, while $\{w_1 \ldots, w_n\}$ is fixed, to estimate μ, σ. In practice, one maximizes $\log P$ instead (same location of max, but better numerical properties).

8.4.1 Optimization with differentiable objective function

$F : \mathbb{R}^n \to \mathbb{R}$ differentiable, ∇F direction of greatest increase, $-\nabla F$ is direction of steepest descent, because locally around $\bar{x}$

$$F(x) \approx F(\bar{x}) + (x - \bar{x})^T \nabla F(\bar{x}) \implies F(\bar{x} + \tau \nabla F(\bar{x})) \approx F(\bar{x}) + \tau \|\nabla F(\bar{x})\|^2,$$

which for $\tau > 0$ increases and for $\tau < 0$ decreases.

A stationary point $\nabla F(x) = 0$ could be a local or global maximum, minimum, or saddle point. If F is twice differentiable, we can check the Hessian matrix at a stationary point

$$H_F(x) = \left(\frac{\partial^2 F}{\partial x_i \partial x_j}(x) \right)_{i,j=1}^{n}$$

Taylor expansion if $\bar{x}$ is stationary point

$$F(x) \approx F(\bar{x}) + \underbrace{\nabla F(\bar{x})^T (x - \bar{x})}_{=0} + \underbrace{\frac{1}{2}(x - \bar{x})^T H_F(\bar{x})(x - \bar{x})}_{\text{increase/decrease/unclear}}$$

Type of extremum in $\bar{x}$ is determined by H_F, i.e. if

- $H_F(\bar{x})$ positive definite, $\bar{x}$ is a local minimum,
- $H_F(\bar{x})$ negative definite, $\bar{x}$ is a local maximum,
- $H_F(\bar{x})$ indefinite, $\bar{x}$ is a saddle point,
- $H_F(\bar{x})$ not invertible, e.g. whole region of saddle points (unlikely).

Positive definiteness can be checked for example by checking whether Cholesky factorization exists (cf. exercises).

8.4.2 Optimization with convex objective function

Definition 8.14 (Convex function) *A function $F : \mathbb{R}^n \to \mathbb{R}$ is called (stricly) convex if for all $x, y \in \mathbb{R}^n$ and all $\alpha \in (0, 1)$*

$$F((1-\alpha)x + \alpha y) \leq (1-\alpha)F(x) = \alpha F(y), \qquad \text{(convex)}$$
$$F((1-\alpha)x + \alpha y) < (1-\alpha)F(x) = \alpha F(y). \qquad \text{(strictly convex)}$$

Lemma 8.15 *If $\bar{x} \in \mathbb{R}^n$ is a local minimum of $F : \mathbb{R}^n \to \mathbb{R}$, then it is a global minimum.*

8.4.3 Methods in 1D

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

Newton's method

Newton's methods (or variants) applied to f' if $f \in C^2$, i.e. iterate

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

Note that Newton's method for minimization is equivalent with approximating a function locally by a parabola and look for its vertex, i.e.

$$f(x) \approx f(x_k) + f'(x_k)(x - x_k) + \frac{1}{2} f''(x_k)(x - x_k)^2,$$

which is a parabola with vertex $x_k - \frac{f'(x_k)}{f''(x_k)}$.

Golden Section Search

Algorithm for non-differentiable unimodal functions. For unimodal functions, a local minimum is a global minimum.

Definition 8.16 *A function $f : [a,b] \to \mathbb{R}$ is called unimodal if there exists $x_u \in [a,b]$ such that f is monotonically decreasing on $[a,x_u]$ and monotonically increasing on $[x_u,b]$.*

Example 8.17 *$f(x) = |x|$, the absolute value function, is unimodal.*

Idea: Suppose for 2 values x_0, x_1 such that $a < x_0 < x_1 < b$ we know $f(x_0) \geq f(x_1)$ and we can discard interval $[a,x_0]$. If instead $f(x_0) \leq f(x_1)$ discard $[x_1,b]$. Then iterate.

Suppose $a = 0$, $b = 1$, $x_0^{(0)} = 1-\lambda$, $x_1^{(0)} = \lambda$, $\lambda \in (\frac{1}{2},1)$. Define λ such that $\lambda^2 = 1-\lambda$. Positive solution $\lambda = \frac{1}{2}(\sqrt{5}-1)$ ($\varphi = \lambda + 1$ is golden ratio).

`initialize` $x_0 = a + (1-\lambda)(b-a)$, $x_1 = a + \lambda(b-a)$, $f_0 = f(x_0)$, $f_1 = f(x_1)$
`while` $|b-a| > tol$
  `if` $f_0 \geq f_1$
    $a \leftarrow x_0$, $x_0 \leftarrow x_1$, $f_0 \leftarrow f_1$
    $x_1 \leftarrow a + \lambda(b-a)$, $f_1 \leftarrow f(x_1)$
  `if` $f_1 > f_0$
    $b \leftarrow x_1, x_1 \leftarrow x_0, f_1 \leftarrow f_0$
    $x_0 \leftarrow a + (1-\lambda)(b-a), f_0 \leftarrow f(x_0)$

If f is unimodal on $[a,b]$, this algorithm converges to the global minimum. In each iteration, the interval size is reduced by factor $0.618 \approx \lambda < 1$, i.e. linear-type convergence as for bisection (root-finding). If f has multiple local minima, golden section search finds some local minimum.

8.4.4 Methods in higher Dimensions

We now 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` $\|\nabla F\|_2 > tol$`)`
  `take` $g^{(k)}(t) = F(x^{(k)} - t\nabla F(x^{(k)}))$
  `find step size` t^* `through line search, e.g.` $t^* = \operatorname{argmin}_{t \geq 0} g^{(k)}(t)$
  `take` $x^{(k+1)} = x^{(k)} - t^* \nabla F(x^{(k)})$

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.

Appendix A

Appendix

A.1 Polynomials

polynomial	definition	recursion	zeros
Lagrange L_i on $\mathbb{R}$?	$L_i(t) = \prod_{\substack{j=0 \\ j \neq i}}^{n} \frac{t-t_j}{t_i-t_j}$		$t_j \; i \neq j$
Newton N_i on $\mathbb{R}$?	$N_0(t) = 1$ $N_i(t) = \prod_{j=0}^{i}(t-t_j)$		$t_j \; i \neq j$
Bernstein B_j on $[0,1]$	$B_j^n(t) = \binom{n}{j} t^j (1-t)^{n-j}$		
Chebychev T_n on $[-1,1]$	$T_n(t) = \arccos(n\cos(t))$	$T_0(t) = 1$, $T_1(t) = t$, $T_{n+1(t)} = 2tT_n(t) - T_{n-1}(t)$	$n, t_j = \cos\left(\frac{2j+1}{2n}\pi\right)$
Legendre P_n on $[-1,1]$	$P_n \in \mathcal{P}_n$, $P_n(1) = 1$, $\int_{-1}^{1} P_n(t)\mathcal{P}_{n-1}(t)\mathrm{d}t = 0$	$P_0(t) = 0$, $P_1(t) = t$, $P_{n+1}(t) = \frac{2n+1}{n+1}tP_n(t) - \frac{n}{n+1}P_{n-1}(t)$	n on $(-1,1)$, Gauss points