Numerical Methods for Computational Science and Engineering
Summary HS 2009

Pascal Spörri
pascal@moeeeep.com

February 2, 2010

Contents

Part I.
Theory

1. Vector norms and matrix norms

Definition: Norm

X= vector space over field $\mathbb{K}$, $\mathbb{K} = \mathbb{R}$, $\mathbb{C}$. A map $||\cdot|| : X \mapsto \mathbb{R}_0^+$ is a *norm* on X, if it satisfies

1. $\forall x \in X : x \neq 0 \quad \iff \quad ||x|| > 0$
2. $||\lambda x|| = |\lambda| \; ||x|| \;\; \forall x \in X, \;\; \lambda \in \mathbb{K}$
3. $||x + y|| \leq ||x|| + ||y|| \; \forall x, y \in X$

Name	Definition	Matlab function
Euclidean norm	$\|\|\vec{x}\|\|_2 = \sqrt{\|x_1\|^2 + \ldots \|x_n\|^2}$	`norm(x)`
1-Norm	$\|\|\vec{x}\|\|_1 = \|x_1\| + \ldots \|x_n\|$	`norm(x,1)`
∞-norm, max-norm	$\|\|\vec{x}\|\|_\infty = \max\{\|x_1\|, \ldots, \|x_n\|\}$	`norm(x,inf)`

Definition: Matrix norm

Given a vector norm $||\cdot||$ on $\mathbb{R}^n$, the associated *matrix norm* is defined by

$$\mathbf{M} \in \mathbb{R}^{m,n} : \quad ||\mathbf{M}|| := \sup_{x \in \mathbb{R}^n \setminus \{0\}} \frac{\mathbf{M}x}{||x||}$$

Definition: Condition (number) of a matrix

Condition of a matrix $\mathbf{A} \in \mathbb{R}^{n,n}$:

$$cond(\mathbf{A}) := ||\mathbf{A}^{-1}|| \; ||\mathbf{A}||$$

$$cond(\mathbf{A}) \gg 1 \qquad \iff \qquad \text{columns/rows of } \mathbf{A} \text{ "almost linearly dependent"}$$

Definition: Symmetric positive definite (s.p.d.) matrices

$\mathbf{M} \in \mathbb{K}^{n,n}$ is *symmetric (Hermitian) positive definite* if

$$\mathbf{M} = \mathbf{M}^H \; \wedge \; x^H \mathbf{M} x > 0 \quad \iff \quad x \neq 0$$

If $x^H \mathbf{M} x \geq 0$ for all $x \in \mathbb{K}^n \implies \mathbf{M}$ positiv semi-definite.

Lemma: Necessary conditions for s.p.d. matrices

For a symmetric/hermitian positiv definite matrix $\mathbf{M} = \mathbf{M}^H$ holds true:

1. $m_{ii} > 0$, $i = 1, \ldots, n$
2. $m_{ii} m_{jj} - |m_{ij}|^2 > 0$ ¡– steht so im skript ist wahrscheinlich aber falsch
3. All eigenvalues of $\mathbf{M}$ are positive.

Definition: Diagonally dominant matrix

$\mathbf{A} \in \mathbb{K}^{n,n}$ is *diagonally dominant*, if

$$\forall k \in \{1, \dots, n\}: \quad \sum_{j \neq k} |a_{kj}| \leq |a_{kk}|$$

The matrix $\mathbf{A}$ is called *strictly diagonally dominant* id

$$\forall k \in \{1, \dots, n\}: \quad \sum_{j \neq k} |a_{kj}| < |a_{kk}|$$

Lemma: Lemma

A diagonally dominant Hermitian/symmetric matrix with non-negative diagonal entries is positive semi-definit.

Definition: Positiv Semidefinite

A *diagonally dominant* Hermitian/symmetric matrix with *non-negative diagonal entries* is positive semi-definite.

Theorem: Gaussian elimination for s.p.d. matrices

Every symmetric/Hermitian positive definite matrix possesses an LU-decomposition.

Lemma: Cholesky decomposition for s.p.d. matrices

For any s.p.d $\mathbf{A} \in \mathbb{K}^{n,n}$ there is a unique upper triangular Matrix $\mathbf{R} \in \mathbb{K}^{n,n}$ with $r_{ii} > 0$ $i = 1, \dots n$ such that $\mathbf{A} = \mathbf{R}^H \mathbf{R}$

Definition: Unitäry und orthogonal matrices

$\mathbf{Q} \in \mathbb{K}^{n,n}$ is *unitary*, if $\mathbf{Q}^{-1} = \mathbf{Q}^H$

$\mathbf{Q} \in \mathbb{R}^{n,n}$ is *orthogonal*, if $\mathbf{Q}^{-1} = \mathbf{Q}^T$

Theorem: Criteria for Unitarity

$$\mathbf{Q} \in \mathbb{C}^{n,n} \quad \text{unitary} \qquad \Longleftrightarrow \qquad ||\mathbf{Q}x||_2 = ||x||_2 \ \forall x \in \mathbb{K}^n$$

Properties of an unitary/orthogonal matrix

If $\mathbf{Q} \in K^{n,n}$ is unitary, then

- $\mathbf{Q}^T\mathbf{Q} = \mathbf{Q}\mathbf{Q}^T = \mathbf{I}$
- $cond(\mathbf{Q}) = 1$
- all rows/columns (regardes as vectors $\in \mathbb{K}^n$ have Euclidean norm$= 1$
- all rows are pairwise orthogonal
- $|det\mathbf{Q}| = 1$ and all eigenvalues $\in \{z \in \mathbb{K} : |z| = 1\}$
- $||\mathbf{QA}||_2 = ||\mathbf{A}||_2$ for any matrix $\mathbf{A} \in \mathbb{K}^{n,m}$

2. Givens Rotations

Let $\mathbf{A}$ be a matrix in $\mathbb{R}^{n,n}$ (im not sure if $\mathbb{K}$ is allowed here). The idea is to rotate the columns of $\mathbf{A}$, in such a way that they stand orthogonal to each other.

Idea Given $(a,b)^T \in \mathbb{R}^2 \setminus \{0\}$. Find $c,\ s \in \mathbb{R}$ with

$$\underbrace{\begin{pmatrix} c & s \\ -s & c \end{pmatrix}}_{\mathbf{C}} \begin{pmatrix} a \\ b \end{pmatrix} = \begin{pmatrix} r \\ 0 \end{pmatrix}$$

and $c^2 + s^2 = 1$. Apparently $\mathbf{C}$ is orthogonal.
Because of the condition $c^2 + s^2 = 1$ it is apparent to represent

$$c = \cos\varphi \qquad s = \sin\varphi$$

Since a rotation doesn't change the length of a vector follows:

$$|r| = ||(r,0)^T||_2 = ||(a,b)^T||_2 = \sqrt(a^2 + b^2)$$

It's now easy to get the solution for the problem above:

$$\begin{aligned} r &= \pm\sqrt(a^2 + b^2) \\ c &= \frac{a}{r} \\ s &= \frac{b}{r} \end{aligned}$$

The givens rotation matrix can now be represented through

$$\mathbf{G}_{i,k} = \begin{array}{c} \\ \\ \\ i \rightarrow \\ \\ \\ \\ k \rightarrow \\ \\ \\ \\ \end{array} \begin{pmatrix} 1 & & & i\downarrow & & & & k\downarrow & & & \\ & \ddots & & & & & & & & & \\ & & 1 & & & & & & & & \\ & & & c & 0 & \cdots & 0 & s & & & \\ & & & 0 & 1 & & & 0 & & & \\ & & & \vdots & & \ddots & & \vdots & & & \\ & & & 0 & & & 1 & 0 & & & \\ & & & -s & 0 & \cdots & 0 & c & & & \\ & & & & & & & & 1 & & \\ & & & & & & & & & \ddots & \\ & & & & & & & & & & 1 \end{pmatrix}$$

$$\mathbf{G}_{i,k} \begin{pmatrix} x_1 \\ x_2 \\ \vdots \\ x_m \end{pmatrix} = \begin{pmatrix} x_1 \\ \vdots \\ x_{i-1} \\ r \\ x_{i+1} \\ \vdots \\ x_{k-1} \\ 0 \\ x_{k+1} \\ \vdots \\ x_m \end{pmatrix}$$

3. Eigenvalues

Definition: Eigenvalues und Eigenvectors

Eigenvalue $\lambda \in \mathbb{C}$ of $\mathbf{A} \in \mathbb{K}^{n,n} \quad :\Leftrightarrow \quad \det(\lambda\mathbf{I} - \mathbf{A}) = 0$

Spectrum of $\mathbf{A} \in \mathbb{K}^{n,n} : \sigma(\mathbf{A}) := \{\lambda \in \mathbb{C} : \text{eigenvalue of } \mathbf{A}\}$ (= Menge aller Eigenwerte)

Eigenspace associated with eigenvalue $\lambda \in \sigma(\mathbf{A})$

$$Eig_{\mathbf{A}}(\lambda) := Ker(\lambda\mathbf{I} - \mathbf{A}$$

Eigenvetor $x \in Eig_{\mathbf{A}}(\lambda)\ \{0\}$

Lemma: Gershgorin circle theorem

For any $\mathbf{A}\mathbb{K}^{n,n}$

$$\sigma(\mathbf{A}) \subset \bigcup_{j=1}^{n} \left\{ z \in \mathbb{C} : |z - a_{jj}| \leq \sum_{i \neq j} |a_{ji}| \right\}$$

Lemma: Similarity and spectrum

The spectrum of a matrix is invariant with respect to *similarity transformations*

$$\forall \mathbf{A} \in \mathbb{K}^{n,n} : \sigma(\mathbf{S}^{-1}\mathbf{A}\mathbf{S}) = \sigma(\mathbf{A}) \forall \text{ regular } \mathbf{S} \in \mathbb{K}^{n,n}$$

Theorem: Schur normal form

$$\forall \mathbf{A} : \exists \mathbf{U} \in \mathbb{C}^{n,n} \text{ unitary } : \quad \mathbf{U}^H\mathbf{A}\mathbf{A} = T \qquad \text{with} \mathbf{T} \in \mathbb{C}^{n,n} \text{ upper triangular}$$

A matrix $\mathbf{A} \in \mathbb{K}^{n,n}$ with $\mathbf{A}\mathbf{A}^H = \mathbf{A}^H\mathbf{A}$ is called *normal*.

Part II.
Computing with Matrices and Vectors

4. Vectors

$$\text{Column vector} = \begin{pmatrix} x_1 \\ \dots \\ x_n \end{pmatrix} \in \mathbb{K}^n$$

$$\text{Row Vector} = \begin{pmatrix} x_1 & \dots & x_n \end{pmatrix}$$

Initialization of vectors in matlab

```
column_vector = [1;2;3];
row_vector    = [1,2,3];
```

5. Matrices

A $n \times m$ Matrix:

$$\mathbf{A} = \begin{pmatrix} a_{11} & \dots & a_{1m} \\ \dots & & \dots \\ a_{n1} & \dots & a_{nm} \end{pmatrix} \in \mathbb{K}^{n,m}$$

Accessing matrix and sub-matrices

Single entry $(\mathbf{A})_{i,j} = a_{i,j}$,

i: Row (Zeile)

j: Column (Spalte)

i-th row $(\mathbf{A})_{i,:}$

j-th column $(\mathbf{A})_{:,j}$

Types There are two different matrix storage formats used in matlab

normal data is placed in a one-dimensional array using the row major format.

sparse Compressed row-storage (CRS) format. Space: $O(n+m)$, Access time: $O(n)$

6. Elementary operations

dot product $x \cdot y = x^H y = \sum_{i=1}^n \overline{x}_i y_i \in K$

tensor product $xy^H = (x_i \overline{y}_j)_{i=1,\dots,m \ j=1,\dots,n} \in \mathbb{K}^{m,n}$

row scaling multiplication with a diagonal matrix from left

$$\begin{pmatrix} d_1 & 0 & \dots & 0 \\ 0 & d_2 & & 0 \\ \vdots & & & \vdots \\ 0 & 0 & & d_n \end{pmatrix} \begin{pmatrix} a_{11} & a_{12} & \dots & a_{1m} \\ a_{21} & a_{22} & & a_{2m} \\ \vdots & & & \\ a_{n1} & a_{n2} & \dots & a_{nm} \end{pmatrix} = \begin{pmatrix} d_1 a_{11} & d_1 a_{12} & \dots & d_1 a_{1m} \\ d_2 a_{21} & d_2 a_{22} & & d_2 a_{2m} \\ \vdots & & & \vdots \\ d_n a_{n1} & d_n a_{n2} & \dots & d_n a_{nm} \end{pmatrix}$$

column scaling multiplication with diagonal matrix from right

$$\begin{pmatrix} a_{11} & a_{12} & \dots & a_{1m} \\ a_{21} & a_{22} & & a_{2m} \\ \vdots & & & \\ a_{n1} & a_{n2} & \dots & a_{nm} \end{pmatrix} \begin{pmatrix} d_1 & 0 & \dots & 0 \\ 0 & d_2 & & 0 \\ \vdots & & & \vdots \\ 0 & 0 & & d_m \end{pmatrix} = \begin{pmatrix} d_1 a_{11} & d_2 a_{12} & \dots & d_m a_{1m} \\ d_1 a_{21} & d_2 a_{22} & & d_m a_{2m} \\ \vdots & & & \vdots \\ d_1 a_{n1} & d_2 a_{n2} & \dots & d_m a_{nm} \end{pmatrix}$$

6.1. Matrix multiplication rules

The matrix product is

associative $(AB)C = A(BC)$

bi-linear $(\alpha A + \beta B)C = \alpha(AC) + \beta(BC)$, $C(\alpha A + \beta B) = \alpha(CA) + \beta(CB)$

non-commutative $AB \neq BA$

7. Complexity

operation	description	# mult/div	# add/sub	complexity
dot product	$(x, y \in \mathbb{K}^n) \mapsto x^H y$	n	$n-1$	$O(n)$
tensor product	$(x \in \mathbb{K}^m, y \in K^n) \mapsto xy^H$	nm	0	$O(nm)$
matrix product	$(A \in \mathbb{K}^{n,m}, B \in \mathbb{K}^{n,k}) \mapsto AB$	nmk	$mk(n-1)$	$O(nmk)$

7.1. Reading the complexity from a plot

Plot the time measurements for different $t_i = time(f(n_i))$ for different $n_1, n_2, \ldots, n_k$, $n_i \in \mathbb{N}$

plot	function	complexity
loglog	straight line	$O(n^\alpha)$ for some α
semilog	straight line	$O(\alpha^n)$ form some α

8. Numerical stability

Definition: Stable algorithm

An Algorithm G for solving a problem $F : X \mapsto Y$ is *numerically stable*, if for all $x \in X$ its result $G(x)$ is the exact result for “slightly perturbed” data:

$$\exists C \approx 1 : \ \forall x \in X : \ \exists \hat{x} \in X : \ ||x - \hat{x}|| \leq Ceps||x|| \wedge G(x) = F(\hat{x})$$

Part III.
Direct Methods for Linear Systems of Equations

Given matrix $A \in K^{n,n}$, vector $b \in K^n$

Sought solution vector $x \in \mathbb{K}^n$

9. Gaussian Elimination

Asymptotic complexity: $O(n^3)$. (Backsubstitution: $O(n^2)$)

$$\mathbf{A} = \begin{pmatrix} \alpha & \mathbf{c}^T \\ d & \mathbf{C} \end{pmatrix} \rightarrow \mathbf{A}' = \begin{pmatrix} \alpha & \mathbf{c}^T \\ 0 & \mathbf{C}' = \mathbf{C} - \frac{\mathbf{d}\mathbf{c}^T}{\alpha} \end{pmatrix}$$

9.1. Stability

Lemma: Equivalence of gaussian elimination and LU-factorization
The following algorithms for solving the LSE $\mathbf{A}x = b$ are *numerically equivalent*

1. Gauss elimination without pivoting
2. LU-factorization followed by forward and backward substitution

10. Sparse Matrices

Initialization:

```
A = sparse(m,n);
A = spalloc(m,n,nnz)
A = sparse(i,j,s,m,n);
A = spdiags(B,d,m,n);
A = speye(n);
A = spones(S);
```

Theorem: LU-Decomposition Fill-in on sparse matrices
If $\mathbf{A} \in \mathbb{K}^{n.n}$ is *regular* with LU-decomposition $\mathbf{A} = \mathbf{LU}$, then the fill-in is confined to the *envelope* of $\mathbf{A}$

Solving Sparse Band Matrices
Alter the LSE with Gauss and try to remove the fill-in (see: set 3, problem 3)

11. QR-Factorization / QR-Decomposition

A QR decomposition (also called a QR factorization) of a matrix is a decomposition of the matrix into an orthogonal and a right triangular matrix.

QR-Decomposition with Householder reflections advantageous for fully populated target columns (dense matrices).

QR-Decomposition with Givens Rotations more efficient, if target column sparsely populated

Lemma: Uniqueness of QR-factorization
The "econimcal QR-factorization of $\mathbf{A} \in \mathbb{K}^{m,n}$, $m \geq n$ with $rank(\mathbf{A}) = n$ is unique, if we demand $r_{ii} > 0$

Stability of the QR-Decomposiztion

- Computing the generalized QR-decomposition $\mathbf{A} = \mathbf{QR}$ by means of Householder reflections or Givens rotations is (numerically) stable for any $\mathbf{A} \in \mathbb{C}^{m,n}$
- For *any* regular systems matrix ans LSE can be solved by means of

$$\text{QR-Decomposition} + \text{orthogonal transformation} + \text{backward substitution}$$

12. Modification Techniques

Lemma: Sherman Morrison Woodbury formula
For regular $\mathbf{A} \in \mathbb{K}^{n,n}$, and $\mathbf{U}$, $\mathbf{V} \in \mathbb{K}^{n,k}$, $l \leq n$, holds

$$(\mathbf{A} + \mathbf{U}\mathbf{V}^H)^{-1} = \mathbf{A}^{-1} - \mathbf{A}^{-1}\mathbf{U}(\mathbf{I} + \mathbf{V}^H\mathbf{A}^{-1}\mathbf{U})^{-1}\mathbf{V}^H\mathbf{A}^{-1}$$

If $\mathbf{I} + \mathbf{V}^H\mathbf{A}^{-1}\mathbf{U}$ regular.

12.1. Rank-1 modifications

with LU-Decomposition Let $\tilde{\mathbf{A}} := \mathbf{A} + uv^H$, $u, v \in \mathbb{K}^n$. uv^H is a general *rank 1 matrix*. For solving $\tilde{\mathbf{A}}x = b$ when $\mathbf{A} = \mathbf{LU}$ is already known. Apply the Sherman Morrison Woodbury formula:

$$x = \left(\mathbf{I} - \frac{\mathbf{A}^{-1}uv^H}{1 + v^H\mathbf{A}^{-1}u}\right)\mathbf{A}^{-1}b$$

with QR-Decomposition Matlab Command

```
[Q1,R1] = qrupdate(Q,R,u,v)
```

with Cholesky factorization Matlab Command

```
R = cholupdate(R,v)
```

Part IV. Iterative Methods for Non-Linear Systems of Equations

13. Iterative Methods

An *iterative method* for (approximately solving) the non-linear equation $F(\vec{x}) = 0$ is an algorithm generating a sequence $\left(x^{(k)}\right)_{k\in\mathbb{N}}$ of *approximate solutions.*

Definition: Convergence of iterative methods

An iterative methods *converges*

$$x^{(k)} \to x^* \text{ and } F(x^*) = 0$$

Definition: Consistency of iterative methods

An iterative method is *consistent* with $F(x) = 0$

$$:\Longleftrightarrow \qquad \Phi_F(x^*, x^*, \dots, x^*) = x^* \Leftrightarrow F(x^*) = 0$$

13.1. Speed of convergence

Definition: Linear convergence

A sequence $x^{(k)},\ k = 0, 1, 2, \dots$ in $\mathbb{R}^n$ *converges linearly* to $x^* \in \mathbb{R}^n$ if

$$\exists L < 1 \quad \left|\left|x^{(k+1)} - x^*\right|\right| \leq C \left|\left|x^{(k)} - x^*\right|\right| \quad \forall k \in \mathbb{N}_0$$

$\Longrightarrow$ straight line in *lin-log plot.*

Definition: Order of convergence

A *convergent* sequence $x^{(k)}$, $k = 0, 1, 2, \dots$ in $\mathbb{R}^n$ converges with *order* $\mathbf{p}$ to $x^* \in \mathbb{R}^n$ if

$$\exists C > 0 : \quad \left|\left|x^{(k+1)} - x^*\right|\right| \leq C \left|\left|x^{(k)} - x^*\right|\right|^p \qquad \forall k \in \mathbb{N}_0$$

with $C < 1$. (For $p = 1$: linear convergence)

Guessing the order of convergence

Abbreviate $\varepsilon_k := \left|\left|x^{(k)} - x^*\right|\right|$

$$\varepsilon_{k+1} \approx C\varepsilon_k^p \quad \Rightarrow \log \varepsilon_{k+1} \approx \log C + p \log \varepsilon_k \quad \to \quad p \approx \frac{\log \varepsilon_{k+1} - \log \varepsilon_k}{\log \varepsilon_k - \log \varepsilon_{k-1}}$$

13.2. Termination criteria

Usually the iteration will never arricew at an/the exact solution x^* after finitely many steps. Thus we can only hope to compute an approximate solution by accepting an $x^{(k)}$ as a result.

A priori termination stop iteration after a fixed number of steps.
Problem: *Hardly ever possible*

A posteriori termination criteria use already computed iterates to decide when to stop. Reliable termination: stop iteration

$$\left|\left|x^{(k)} - x^*\right|\right| \leq \tau \qquad \tau \equiv \text{ prescribed } \textit{tolerance}$$

Problem: x^* *not known*

Stationary iteration use that the finite numbers are finite: Wait until (convergent) iteration becomes stationary.

$$\textit{wait until}: x^{(k)} = x^{(k+1)}$$

Problem: *Very inefficent*

Residual based termination Stop convergent iteration when

$$\left|\left|F\left((x^{(k)}\right)\right|\right| \leq \tau \qquad \tau \equiv \text{ prescribed } \textit{tolerance}$$

Problem: *No guaranteed accuracy since* $\left|\left|F\left((x^{(k)}\right)\right|\right|$ small $\not\Rightarrow |x^{(k)} - x^*|$ small.

14. Fixed Point Iterations

F is a non-linear system of equations with $F : D \subset \mathbb{R}^n \mapsto \mathbb{R}^n$
A *fixed point iteration* is defined by an *iteration function* $\Phi : U \subset \mathbb{R}^n \mapsto \mathbb{R}^n$ and an initial guess $x^{(0)} \in U$.

$$x^{(k+1)} := \Phi\left(x^{(k)}\right)$$

Definition: Consistency of fixed point iterations
A fixed point iteration $x^{(k+1)} := \Phi\left(x^{(k)}\right)$ is *consistent* with $F(x) = 0$ if

$$F(x) = 0 \qquad \text{and} \qquad x \in U \cap D \quad \Longleftrightarrow \quad \Phi(x) = x$$

If Φ continous and the fixed point iteratin is (locally) convergent to x^* then x^* ist the fixed point of the iteration function Φ.

Definition: Contractive mapping
$\Phi : U \subset \mathbb{R}^n \mapsto \mathbb{R}^n$ is *contractive*, if

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

Theorem: Banach's fixed point theorem
If $D \subset \mathbb{K}^n$ $(\mathbb{K} = \mathbb{R}, \mathbb{C})$ closed and $\Phi : D \mapsto D$ satisfies

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

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

$$
\begin{aligned}
-1 < \Phi'(x^*) < 1 &\quad \text{convergence} \\
\Phi'(x^*) < -1 &\quad \text{divergence} \\
\Phi'(x^*) > 1 &\quad \text{divergence}
\end{aligned}
$$

Lemma: Sufficient condition for local linear convergence of fixed point iterations
If $\Phi : U \subset \mathbb{R}^n \mapsto \mathbb{R}^n$, $\Phi(x^*) = x^*$, Φ differentiable in x^* and $||D\Phi(x^*)|| < 1$, then the fixed point iteration converges locally and at least *linearly*.

If $0 < ||D\Phi(x)|| < 1$, then the *asymptotic rate* of linear convergence is $L = ||D\Phi(x)||$ (where as L: lipschitz constant).

Lemma: Higher order local convergence of fixed point iterations
If $\Phi : U \subset \mathbb{R} \mapsto \mathbb{R}$ is $m+1$ times *continuously differentiable*, $\Phi(x^*) = x^*$ for some x^* in the interrior uf U and $\Phi^{(l)}(x^*) = 0$ for $l = 1, \ldots, m$, $m \geq 1$, then the fixed point iteration converges locally to x^* with

$$
\mathit{order} \geq m + 1
$$

15. Zero Finding

$F : I \subset \mathbb{R} \mapsto \mathbb{R}$ *continous*, I interval. *Sought:* $x^* \in I : F(x^*) = 0$.

15.1. Bisection

Use of ordering of real numbers and itermediate value theorem. *Input:* $a, b \in I$ such that $F(a)F(b) < 0$ (different signs).

```
function x = bisect(F,a,b,tol)
   fa = F(a);
   fb = F(b);
   if (fa*fb > 0)
       error('f(a) and f(b) have the same sign');
   end

   v = 1;
   if (fa > 0)
       v = -1;
   end
   x = 0.5 * (a+b);
   while ((b-a > tol) & (a<x) & (x<b))
       if (v*F(x) >0)
           b = x;
       else
           a = x;
       end
       x = 0.5*(a+b)
   end
end
```

Advantages foolproof, requires only F evaluations

Drawbacks Merely linear convergence $|x^{(k)} - x^*| \leq 2^{-k}|b - a|$

$\Longrightarrow$ `fzero` uses this approach.

15.2. Model Function Methods

Model function Methods is a class of iterative methods for finding zeroes of F:

Idea Given: approximate zeroes $x^{(k)}$, $x^{(k-1)}$, $,\dots,x^{(k-m)}$

1. replace F with a *model function* $\tilde{F}$ (using function values/derivative values in $x^{(k)}$, $x^{(k-1)}$, $,\dots,x^{(k-m)}$)
2. $x^{(k+1)} :=$ zero of $\tilde{F}$

Distinguish between one-point methods and multipoint methods.

15.2.1. Newton Method

Assume: $F : I \to \mathbb{R}$ continuously differentiable. Model function := tangent af F in $x^{(k)}$.

$$\tilde{F}(x) := F\left(x^{(k)}\right) + F'\left(x^{(k)}\right)\left(x - x^{(k)}\right)$$

with $x^{(k+1)} :=$ zero of the tangent.

We obtain the *Newton Iteration*

$$x^{(k+1)} := x^{(k)} - \frac{F\left(x^{(k)}\right)}{F'\left(x^{(k)}\right)} \qquad \text{with } F'\left(x^{(k)}\right) \neq 0$$

15.2.2. Multi Point Methods

Replace F with an *interpolation polynomial* producing interpolatory model function methods.

Secant Method

$$\begin{aligned} x^{(k+1)} &= \text{ zero of secant} \\ s(x) &= x^{(k)} - \frac{F(x^{(k)}) - F(x^{(k-1)})}{F(x^{(k)}) - F(x^{(k-1)})} \cdot (x - x^{(k)}) \\ \implies \qquad x^{(k+1)} &= x^{(k)} - \frac{F(x^{(k)}) \cdot (x^{(k)} - x^{(k-1)})}{F(x^{(k)}) - F(x^{(k-1)})} \end{aligned}$$

- Only one function evaluation per step
- no derivatives required

15.3. Efficiency

Efficiency of an iterative method $\leftrightarrow$ *computational effort* to reach prescribed number of significant digits in result.

Computational effort/step

$$W \approx \frac{\#\{\text{evaluations of } F\}}{\text{step}} + n \cdot \frac{\#\{\text{evaluations of } F'\}}{\text{step}} \dots$$

Definition: Efficiency

$$\begin{aligned} \text{Efficiency} &= \frac{\#\text{ of digits gained}}{\text{total work required}} = \frac{|\log p|}{k(p) \cdot W} \\ k(p) &= \text{ number of steps to achieve relative reduction of error} \\ |\log p| &= \text{ number of significant digits of } x^{(k)} \end{aligned}$$

16. Newton's Method

16.1. The Newton Iteration

Definition: Newton Iteration

$$x^{(k+1)} := x^{(k)} - DF\left(x^{(k)}\right)^{-1} F\left(x^{(k)}\right)$$

```
function x = newton(x,F,DF,tol)
    for i = 1:MAXIT
        s = DF(x) \ F(x);
        x = x - s;
        if (norm(s) < tol*norm(x))
            return;
        end
    end
end
```

If $DF(x)$ is not available use

$$\frac{\delta F_i}{\delta x_j}(x) \approx \frac{F_i(x + h\vec{e}_j) - F_i(x)}{h}$$

to approximate $DF(x)$. Warning: Impact of roundoff errors for small h.

The Newton Method has *Local quadratic convergence* if $DF(x^*)$ is regular.

A posteriori termination criterion

Quit as soon as

$$\left|\left|DF\left(x^{(k)}\right)^{-1} F(x^{(k)}\right|\right| < \tau \left|\left|x^{(k)}\right|\right|$$

Since we expect that $DF\left(x^{(k-1)}\right) \approx DF\left(x^{(k)}\right)$, when the Newton Iteration has converged

The Newton Method

- converges *asymptotically* very fast: doubling of number of significant digits in each step
- often a *very small region of convergence*, which requires an initial guess rather close to the solution

16.2. Damped Newton Method

We observe an "overshooting" of the Newton correction.

Idea: Use a damping factor for the Newton correction:

$$x^{(k+1)} := x^{(k)} - \lambda^{(k)} DF\left(x^{(k)}\right)^{-1} F\left(x^{(k)}\right) \qquad \text{with: } \lambda^{(k)} > 0$$

Choice of damping factor: Use maximal $\lambda^{(k)} > 0 : \left|\left|\Delta\overline{x}\left(\lambda^{(k)}\right)\right|\right| \leq \left(1 - \frac{\lambda^{(k)}}{2}\right)\left|\left|\Delta x^{(k)}\right|\right|$ where

$$\Delta x^{(k)} = DF\left(x^{(k)}\right)^{-1} F\left(x^{(k)}\right)$$
$$\Delta\overline{x}\left(\lambda^{(k)}\right) = DF\left(x^{(k)}\right)^{-1} F\left(x^{(k)} + \lambda^{(k)}\Delta x^{(k)}\right)$$

Policy

Reduce damping factor by a factor $q \in]0,1[$ (usually $q = \frac{1}{2}$) until the affine invariant natural monotonicity test passed.

16.3. Quasi-Newton Method (Broyden Method)

Use when $DF(x)$ is not available and numerical differentiation is too expensive.
Worthwhile for dimensions $n \gg 1$ and low accuracy requirements.

Part V. Krylov Methods for Linear Systems of Equations

A class of *iterative methods* for approximate solutions of large linear systems of equations.

17. Descent Methods

Definition: Energy norm

A s.p.d matrix $\mathbf{A} \in \mathbb{R}^{n,n}$ induces a *energy norm*

$$||x||_{\mathbf{A}} := \left(x^T \mathbf{A} x\right)^{1/2} \quad x \in \mathbb{R}^n$$

Lemma: S.p.d LSE and quadratic minimization problem

An LSE with $\mathbf{A} \in \mathbb{R}^{n,n}$ s.p.d is equivalent to a minimization problem:

$$\mathbf{A}x = b \quad \Longleftrightarrow \quad x = \arg\min_{y \in \mathbb{R}^n} J(y), \quad J(y) = \frac{1}{2} y^T \mathbf{A} y - b^T y$$

17.1. Abstract steepest descent

Given continuously differentiable $F : D \subset \mathbb{R}^n \mapsto \mathbb{R}$

Find minimizer $x^* \in D : x^* = \arg\min_{x \in D} F(x)$

$x^{(0)} \in D$
$k = 0$
while $\left|\left|x^{(k)} - x^{(k-1)}\right|\right| \leq \tau \left|\left|x^{(k)}\right|\right|$ **do**
  $d_k = -\mathbf{grad} F(x^{(k)})$
  $t^* = \arg\min_{t \in \mathbb{R}} F(x^{(k)} + t d_k)$
  $x^{(k+1)} = x^{(k)} + t^* d_k$
  $k = k + 1$
end while

17.2. Gradient Method for s.p.d linear systems of equations

Adapt the steepest descent algorithm for the quadratic minimization problem.

$$F(x) = J(x) = \frac{1}{2} x^T \mathbf{A} x - b^T x \quad \Rightarrow \quad \mathbf{grad} J(x) = \mathbf{A}x - b$$

$x^{(0)} \in \mathbb{R}^n$
$k = 0$
$r_0 = b - \mathbf{A}x^{(0)}$
while $\left|\left|x^{(k)} - x^{(k-1)}\right|\right| \leq \tau \left|\left|x^{(k)}\right|\right|$ **do**
  $t^* = \frac{r_k^T r_k}{r_k^T \mathbf{A} r_k}$
  $x^{(k+1)} = x^{(k)} + t^* r_k$
  $r_{k+1} = r_k - t^* \mathbf{A} r_k$
  $k = k + 1$
end while

17.3. Convergence

The steepest descent and the gradient method posess at least linear convergence

Theorem: Convergence of gradient/steepest descent method
The iterates of the gradient method satisfy

$$\left\|x^{(k+1)} - x^*\right\|_A \leq L \left\|x^{(k)} - x^*\right\|_A \qquad L = \frac{cond_2(\mathbf{A}) - 1}{cond_2(\mathbf{A}) + 1}$$

that is, the iteration converges at least linearly

18. Conjugate gradient method

Again, we consider a linear system of equations $\mathbf{A}x = b$ with s.p.d system matrix $\mathbf{A} \in \mathbb{R}^{n,n}$ and given $b \in \mathbb{R}^n$

Idea Replace linear search with *subspace correction*

Given Initial gues $x^{(0)}$ and *nested* subspaces $U_1 \subset U_2 \subset \ldots \subset U_n = \mathbb{R}^n$, $\dim U_k = k$

$$U_{k+1} = Span\{U_k, r_k\}$$

18.1. Krylov Spaces

Definition: Krylov Space
For $\mathbf{A} \in \mathbb{R}^{n,n}$, $z \in \mathbb{R}^n$, $z \neq 0$, the l-th *Krylov space* is defined as

$$\mathcal{K}(\mathbf{A}, z) = Span\{z, \mathbf{A}z, \ldots, A^{l-1}z\}$$

Lemma:
The subspaces $U_k \subset \mathbb{R}^n$, $k \geq 1$ defined above satisfy

$$U_k = Span\{r_0, \mathbf{A}r_0, \ldots, A^{l-1}r_0\} = \mathcal{K}(\mathbf{A}, z)$$

where $r_0 = b - \mathbf{A}x^{(0)}$ is the initial residual.

18.2. Implementation of CG

Left out

```
x = pcg(A,b,tol,maxit,[],[],x0);
x = pcg(Afun,b,tol,maxit,[],[],x0);
```

CG is used for lager n as iterative solver $x^{(k)}$ for some $k \ll n$ is expected to provide good approximation for x^*

19. Preconditioning

CG has a slow convergence rate in case $\mathcal{K}(\mathbf{A}) \gg 1$

Idea Apply CG Method to transformed linear systems

$$\tilde{\mathbf{A}}\tilde{x} = \tilde{b}$$
$$\tilde{\mathbf{A}} = \mathbf{B}^{-1/2}\mathbf{A}\mathbf{B}^{-1/2}$$
$$\tilde{x} = \mathbf{B}^{1/2}x$$
$$\tilde{b} = \mathbf{B}^{-1/2}b$$

where as $\mathbf{B}^{1/2} = \mathbf{Q}^T \mathbf{D}^{1/2} \mathbf{Q}$ and $\mathbf{Q} = \mathbf{Q}^T$.

Preconditioner

A s.p.d matrix $\mathbf{B} \in \mathbb{R}^{n,n}$ is called a *preconditioner* for the s.p.d matrix $\mathbf{A} \in \mathbb{R}^{n,n}$ if

1. $\mathcal{K}(\mathbf{A}^{-1/2}\mathbf{A}\mathbf{B}^{-1/2})$ is "small"
2. the evaluation of $\mathbf{B}^{-1}x$ is about as expensive as the matrix vector multiplication $\mathbf{A}x$, $x \in \mathbb{R}^n$

20. Survey of Krylov Subspace Methods

20.1. Minimal residual function

Replace inner Euclidean product in CG with $\mathbf{A}$-inner product.

$$\left|\left|x^{(l)} - x\right|\right|_A \qquad \text{replaced with} \qquad \left|\left|\mathbf{A}(x^{(l)} - x)\right|\right|_2$$

`minres` $\Longrightarrow$ Iterative solver for *symmetric* Matrices $\mathbf{A}$

`gmres` $\Longrightarrow$ Iterative solver for *general* Matrices $\mathbf{A}$

Part VI.
Eigenvalues

21. "Direct" Eigensolvers

All "direct" eigensolvers are iterative methods

```
function d= eigqr(A,tol)
   n = size(A,1);
   while (norm(tril(A,-1))> tol*norm(A))
       shift = A(n,n);
       [Q,R] = qr(A-shift*eye(n));
       A = Q'*A*Q;
       tril(A,-1)
   end
   d = diag(A);
end
```

22. Power Methods

22.1. Direct Power Method

Initial Guess $z^{(0)}$ "arbitrary"

Next Iterate $w = Az^{(k-1)}$, $z^{(k)} = \frac{w}{||w||_2}$

Computes the eigenvector for λ_{max}. Get eigenvalue through raleigh quotient.

Definition: Raleigh Quotient

$$p_{\mathbf{A}}(u) = \frac{u^H \mathbf{A} u}{u^H u}$$

If $\lambda \in \sigma(\mathbf{A})$ and $z \in Eig_\lambda(\mathbf{A})$ then $p_{\mathbf{A}}(z) = \lambda$.

22.1.1. Normalized Cut

Pixel set $\mathcal{V} : \{1, \dots, nm\}$

Indexing (since all pixels are saved in a row)

$$k = index(pixel_{i,j}) = (i-1)n + j$$

Notation

$$p_k = (\mathbf{P})_{ij} \qquad k = 1, \dots, nm$$

Local similarity matrix $\mathbf{W} \in \mathbb{R}^{N,N}$ where as $N = nm$.

$$(\mathbf{W})_{i,j} = \begin{cases} 0 & \text{if pixels } i,\ j \text{ not adjacent} \\ 0 & \text{if } i = j \\ \sigma(p_i, p_j) & \text{if pixels } i \text{ and } j \text{ adjacent} \end{cases}$$

σ is a *similarity function*

$$\sigma(x, y) = e^{-\alpha(x-y)^2} \qquad \alpha > 0$$

Definition: Normalized Cut
For $\mathcal{X} \subset \mathcal{V}$ we define the normalized cut as

$$Ncut(\mathcal{X}) = \frac{cut(\mathcal{X})}{weight(\mathcal{X})} + \frac{cut(\mathcal{X})}{weight(\mathcal{V} \setminus \mathcal{X})}$$

with

$$cut(\mathcal{X}) = \sum_{i \in \mathcal{X},\ j \notin \mathcal{X}} w_{ij}, \qquad weight(\mathcal{X}) = \sum_{i \in \mathcal{X},\ j \in \mathcal{V}} w_{ij}$$

Segmentation problem find

$$\mathcal{X}^* \subset \mathcal{V} : \mathcal{X}^* = \arg\min_{\mathcal{X} \subset \mathcal{V}} Ncut(\mathcal{X})$$

Reformulate the problem

$$\textbf{Indicator function :} \qquad z : \{1, \dots N\} \mapsto \{-1, 1\},\ z_i := z(i) = \begin{cases} 1 & \text{if } i \in \mathcal{X} \\ -1 & \text{if } i \notin \mathcal{X} \end{cases}$$

Lemma: Ncut and Rayleigh quotient
With $z \in \{-1, 1\}^N$ (indicator function) there holds

$$Ncut(\mathcal{X}) = \frac{y^T \mathbf{A} y}{y^T \mathbf{D} y}, \qquad y = (1 + z) - \beta(1 - z), \quad \beta = \frac{\sum_{z_i > 0} d_i}{\sum_{z_i < 0} d_i}$$

22.2. Inverse Iteration

If $\mathbf{A} \in \mathbb{K}^{n,n}$ regular:

$$\text{Smallest (in modulus) EV of } \mathbf{A} = \frac{1}{(\text{Largest (in modulus) EV of } \mathbf{A}^{-1})}$$

22.3. Preconditioned Inverse Iteration

Given $\mathbf{A} \in \mathbb{K}^{n,n}$ find *smallest* Eigenvalue of regular $\mathbf{A}$. Instead of solving $\mathbf{A}w = z^{(k-1)}$ compute $w = \mathbf{B}^{-1} z^{(k-1)}$ with "inexpensive s.p.d. *approximate inverse* $\mathbf{B}^{-1} \approx \mathbf{A}^{-1}$ ($\mathbf{B}$: preconditioner).

Initial Guess $z^{(0)}$

Next Iterate

$$\begin{aligned} w &= z^{(k-1)} - \mathbf{B}^{-1}(\mathbf{A} z^{(k-1)}) - p_{\mathbf{A}}(z^{(k-1)}) z^{(k-1)} \\ z^k &= \frac{w}{||w||_2} \end{aligned}$$

- Linear convergence
- fast convergence if spectral condition number $\mathcal{K}(\mathbf{B}^{-1}\mathbf{A})$ small.

22.4. Subspace Iterations

Task Compute m, $m \ll n$ of the largest/smallest eigenvalues of $\mathbf{A} = \mathbf{A}^H$ and associated eigenvectors.

Use orthogonality of the Eigenvectors.

Part VII.
Least Squares

Given $\mathbf{A} \in \mathbb{K}^{m,n},\ m,\ n \in \mathbb{N},\ b \in \mathbb{K}^m$

Find $x \in \mathbb{K}^n$ such that

1. $||\mathbf{A}x - b|| = \inf\{||\mathbf{A}y - b||_2 : y \in \mathbb{K}^\}$
2. $||x||$ is minimal

Lemma: Existence & Uniqueness of Solutions of the Least squares problem
The least squares problem for $\mathbf{A} \in \mathbb{K}^{m,n}$, $\mathbf{A} \neq 0$ has a unique solution for every $b \in \mathbb{K}^m$

23. Normal Equations

$$\mathbf{A}^H\mathbf{A}x = \mathbf{A}^H b$$

Numerically unstable

$$cond_2(\mathbf{A}^H\mathbf{A}) = cond_2(\mathbf{A})^2$$

Part VIII.
Filtering Algorithms

24. Discrete Convolutions

Definition: Discrete Convolution
Given $x = (x_0, \dots, x_{n-1})^T \in \mathbb{K}^n$, $h = (h_0, \dots, h_{n-1})^T \in \mathbb{K}^n$ their *discrete convolution* is the vector $y \in \mathbb{K}^{2n-1}$ with components

$$y_k = \sum_{j=0}^{n-1} h_{k-j} x_j, \qquad k = 0, \dots, 2n-2$$

Definition: Discrete Periodic Convolution
The *discrete periodic convolution* of two n-periodic sequences $(x_k)_{k\in\mathbb{Z}}$, $(y_k)_{k\in\mathbb{Z}}$ yields the n-periodic sequence

$$(z_k) = (x_k) *_n (y_k)$$

$$z_k = \sum_{j=0}^{n-1} x_{k-j} y_j = \sum_{j=0}^{n-1} \qquad k \in \mathbb{Z}$$

Definition: Circulant Matrix
A matrix $\mathbf{C} = (c_{ij})_{i,j=1}^n \in \mathbb{K}^{n,n}$ is *circulant*

$$:\Leftrightarrow \quad \exists (u_k)_{k\in\mathbb{Z}}\ n-\text{periodic sequence:}\ \ c_{ij} = u_{i-j},\ 1 \le i, j \le n$$

25. Discrete Fourier Transform (DFT)

Fourier-Matrix

$$F_n = \begin{pmatrix} w_n^0 & w_n^0 & \dots & w_n^0 \\ w_n^0 & w_n^1 & \dots & w_n^{n-1} \\ w_n^0 & w_n^2 & \dots & w_n^{2n-2} \\ \vdots & \vdots & & \vdots \\ w_n^0 & w_n^{n-1} & \dots & w_n^{(n-1)^2} \end{pmatrix}$$

$w_n^k = e^{2\pi\ i\ k/n}$

Lemma: Properties of Fourier Matrix
The scaled Fourier Matrix $\frac{1}{\sqrt{n}}\mathbf{F}_n$ is unitary:

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

Lemma: Diagonalization of Circulant Matrices
For any circulant matrix $\mathbf{C} \in \mathbb{K}^{n,n}$, $c_{ij} = u_{i-j}$, $(u_k)_{k\in\mathbb{Z}}$ n-periodic sequence, holds true

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

Conclusion:

$$\mathbf{C} = \mathbf{F}_n^{-1} diag(d_1, \ldots, d_n) \mathbf{F}_n$$

Definition: Discrete Fourier Transform (DFT)

The linear map $\mathcal{F}_n : \mathbb{C}^n \mapsto \mathbb{C}^n$, $\mathcal{F}_n(y) := \mathbf{F}_n y$, $y \in C^n$ is called *discrete Fourier transform*

```
%% DFT
c = fft(y)
%% Inverse DFT
y = ifft(c);
```

25.1. Two-Dimensional DFT

```
fft2(Y) = fft(fft(Y).').'
```

26. Fast Fourier Transform (FFT)

Complexity of FFT algorithm: $n = 2^L$

$$O(L2^L) = O(n \log_2 n)$$

Part IX.
Polynomial Interpolation

27. Polynomials

$$\mathcal{P}_k := \{t \mapsto a_k t^k + a_{k-1}t^{k-1} + \cdots + a_0,\ a_j \in \mathbb{K}$$

Theorem: Dimension of Space of Polynomials

$$\dim \mathcal{P}_j = k+1 \qquad \text{and} \qquad \mathcal{P}_k \subset C^\infty(\mathbb{R})$$

Matlab:

$$a_k t^k + a_{k-1}t^{k-1} + \cdots + a_0 \quad \text{(use horner schema to calculate)}$$

```
polyval(p,x);
```

28. Polynomial Interpolation: Theory

Given: Simple nodes $t_0, \ldots, t_n$, $n \in \mathbb{N}$, $-\infty < t_0 < t_1 < \cdots < t_n < \infty$ and the values $y_0, \ldots, y_n \in \mathbb{K}$ compute $p \in \mathcal{P}_n$ such that

$$p(t_j) = y_j \qquad \text{for } j = 0, \ldots n$$

28.1. Lagrange Polynomials

For *nodes* $t_0 < t_1 < \cdots < t_n$ consider

$$\textbf{Lagrange Polynomials:}\quad L_i(t) = \prod_{j=0\ \wedge\ j\neq i}^{n} \frac{t-t_j}{t_i - t_j}$$

29. Chebychev Interpolation

Definition: Chebychev Polynomial

The n^{th} *Chebychev Polynomial* is

$$T_n(t) = \cos(n \arccos(t)) \qquad -1 \le t \le 1$$
$$\text{Zeros of } T_n: \qquad t_k = \cos\left(\frac{2k-1}{2n}\pi\right), \quad k = 1, \ldots, n$$

Scaling argument

$$[-1,1] \xrightarrow{\hat{t} \mapsto t := a + \frac{1}{2}(\hat{t}+1)(b-a)} [a,b] \qquad \hat{f}(\hat{t}) := f(t)$$

29.1. Computational Aspects

Theorem: Orthogonality of Chebychef polynomials

The Chebychef polynomials are orthogonal with respect to the scalar product

$$\langle f, g\rangle = \int_{-1}^{1} f(x)g(x)\frac{1}{\sqrt{1-x^2}}dx$$

Theorem: Representation formula

The interpolation polynomial p of f in the Chebychev nodes $x_0, \dots, x_n$ (the zeros of T_{n+1} is given by:

$$p(x) = \frac{1}{2}c_0 + c_1 T_1(x) + \dots c_n T_n(x)$$

with

$$c_k = \frac{2}{n+1} \sum_{t=0}^{n} f\left(\cos\left(\frac{2l+1}{n+1} \cdot \frac{\pi}{2}\right)\right) \cdot \cos\left(k \frac{2l+1}{n+1} \cdot \frac{\pi}{2}\right)$$

Theorem: Clenshah algorithm

Let $p \in \mathcal{P}_n$ be an arbitrary polynomial

$$p(x) = \frac{1}{2}c_0 + c_1 T_1(x) + \dots + c_n T_n(x)$$

Set

$$\begin{aligned} d_{n+2} &= d_{n+1} = 0 \\ d_k &= c_k + (2x) \cdot d_{k+1} - d_{k+2} \qquad \text{for } k = n, n-1, \dots, 0 \end{aligned}$$

Then

$$p(x) = \frac{1}{2}(d_0 - d_2)$$

The Clenshaw algorithm is numerically stable

Part X.
Piecewise Polynomials

Perspective Data Interpolation

Problem Model a functional relation: $f : I \subset \mathbb{R} \mapsto \mathbb{R}$ from the (exact) measurments (t_i, y_i), $i = 0, \ldots, n$.
Interpolation constraint $f(t_i) = y_i \ \forall i$

Goal Shape preserving interpolation

$$\begin{aligned} \text{positive data} \quad &\rightarrow \quad \text{positive interpolant } f \\ \text{monotonic data} \quad &\rightarrow \quad \text{monotonic interpolant } f \\ \text{convex data} \quad &\rightarrow \quad \text{convex interpolant } f \end{aligned}$$

30. Piecewise Lagrange Interpolation

30.1. Piecewise Linear Interpolation

Data $(t_i, y_i) \in \mathbb{R}^2,\ i = 0, \ldots, n,\ n \in \mathbb{N},\ t_0 < t_1 < \cdots < t_n$

Piecewise linear interpolant connect the dots with direct lines.

$$s(x) = \frac{(t_{i+1} - t)y_i + (t - t_i)y_{i+1}}{t_{i+1} - t_i} \qquad t \in [t_i, t_{i+1}]$$

30.2. Piecewise Polynomial Interpolation

Use a polynom instead of a direct line.

31. Cubic Hermite Interpolation

Given Mesh points $(t_i, y_i) \in \mathbb{R}^2,\ i = 0, \ldots, n,\ t_0 < t_1 < \ldots < t_n$

Goal Function $f \in \mathbb{C}^1([t_0, t_n]), f(t_i) = y_i,\ i = 0, \ldots, n$

$$s(t) = y_{i-1}H_1(t) + y_i H_2(t) + c_{i-1}H_3(t) + c_i H_4(t), \qquad t \in [t_{i-1}, t_i]$$

$$\begin{aligned} H_1(t) &= \phi\left(\frac{t_i - t}{h_i}\right) \\ H_2(t) &= \phi\left(\frac{t - t_{i-1}}{h_i}\right) \\ H_3(t) &= -h_i\theta\left(\frac{t_i - t}{h_i}\right) \\ H_4(t) &= h_i\theta\left(\frac{t - t_{i-1}}{h_i}\right) \\ h_i &= t_i - t_{i-1} \\ \phi(\tau) &= 3\tau^2 - 2\tau^3 \\ \theta(\tau) &= \tau^3 - \tau^2 \end{aligned}$$

Choose slopes c_i according to specification. For example:

$$c_i = \begin{cases} \Delta_1 & \text{for } i = 0 \\ \Delta_n & \text{for } i = n \\ \frac{t_{i+1}-t_i}{t_{i+1}-t_{i-1}}\Delta_i + \frac{t_i - t_{i-1}}{t_{i+1}-t_{i-1}}\Delta_{i+1} & \text{if } i \leq i < n \end{cases}$$

$$\Delta_j = \frac{y_j - y_{j-1}}{t_j - t_{j-1}}$$

31.1. Shape Preserving Hermite Interpolation

Hermite interpolation does not preserve monotonicity. Choose a different formula for the slopes:

$$c_i = \begin{cases} 0 & & \text{if } sgn(\Delta_i) \neq sgn(\Delta_{i+1}) \\ \frac{1}{\frac{w_a}{\Delta_i}+\frac{w_b}{\Delta_{i+1}}} & \text{(weighted average)} & \text{otherwise} \quad (w_a + w_b = 1) \end{cases}$$

Concrete choice of weights:

$$w_a = \frac{2h_{i+1} + h_i}{3(h_{i+1} + h_i)} \qquad w_b = \frac{h_{i+1} + 2h_i}{3(h_{i+1} + h_i)} \qquad \text{Matlab Function: pchip}$$

32. Splines

Definition: Spline Space

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

$$\mathcal{S}_{d,\Updownarrow} := \left\{ s \in C^{d-1}(I) : \; s_j = s_{|[t_{j-1},t_j]} \in \mathcal{P}_d \; \forall j = 1, \ldots, n \right\}$$

$d = 0$: $\mathcal{M}$-piecewise constant *discontinuous* functions

$d = 1$: $\mathcal{M}$-piecewise linear *continuous* functions

$d = 2$: continously differentiable $\mathcal{M}$-piecewise quadratic functions

Dimension of Spline Space

Dimension of spline space by *counting argument*

$$dim \mathcal{S}_{d,\mathcal{M}} = n \cdot dim \mathcal{P}_d - \#\{C^{d-1}\text{continuity constraints}\} = n \cdot (d + 1) - (n - 1) \cdot d = n + d$$

32.1. Cubic Spline Interpolation

Special case of Spline interpolation. Since C^2-functions are perceived as smooth. Choose $d = 3$.

Reuse representation through cubic Hermite basis polynomials:

$$s_{[t_{j-1},t_j]}(t) = s(t_{j-1}) \cdot (1 - 3\tau^2 + 2\tau^3) + s(t_j) \cdot (3\tau^2 - 2\tau^3) + h_j s'(t_{j-1}) \cdot (\tau - 2\tau^2 + \tau^3) + h_j s'(t_j) \cdot (-\tau^2 + \tau^3)$$

with $h_j = t_j - t_{j-1}$ and $\tau = (t - t_{j-1})/h_j$

Produces Linear $\mathbf{n-1}$ *linear* equations for n slopes

$$\frac{1}{h_j}c_{j-1} + \left(\frac{2}{h_j} + \frac{2}{h_{j+1}}\right)c_j + \frac{1}{h_{j+1}}c_{j+1} = 3\left(\frac{y_j - y_{j-1}}{h_j^2} + \frac{y_{j+1} - y_j}{h_{j+1}^2}\right) \qquad c_j = s'(t_j)$$

$$\begin{pmatrix} b_0 & a_1 & b_1 & 0 & \cdots & \cdots & \cdots & 0 \\ 0 & b_1 & a_2 & b_2 & 0 & & & \vdots \\ \vdots & 0 & \ddots & \ddots & \ddots & \ddots & & \vdots \\ \vdots & & \ddots & \ddots & \ddots & \ddots & \ddots & \vdots \\ \vdots & & & 0 & b_{n-3} & a_{n-1} & b_{n-2} & 0 \\ 0 & \cdots & \cdots & \cdots & 0 & b_{n-2} & a_0 & b_{n-1} \end{pmatrix} \begin{pmatrix} c_0 \\ \vdots \\ \vdots \\ \vdots \\ c_n \end{pmatrix} = \begin{pmatrix} 3\left(\frac{y_1-y_0}{h_1^2} + \frac{y_2-y_1}{h_2^2}\right) \\ \vdots \\ \vdots \\ \vdots \\ \vdots \\ 3\left(\frac{y_{n-1}-y_{n-2}}{h_{n-1}^2} + \frac{y_n-y_{n-1}}{h_n^2}\right) \end{pmatrix} \qquad \Rightarrow \begin{cases} a_j = \frac{1}{h_j} \\ b_j = \frac{2}{h_j} + \frac{2}{h_{j+1}} \end{cases}$$

Two additional constraints are required, three different choices are possible (put them into the LSE above):

Complete cubic spline interpolation

$$s'(t_0) = c_0$$
$$s'(t_n) = c_n$$

Natural cubic spline interpolation

$$s''(t_0) = s''(t_n) = c_n \qquad \Rightarrow \begin{cases} \frac{2}{h_1}c_0 + \frac{1}{h_1}c_1 = 3\frac{y_1-y_0}{h_1^2} \\ \frac{1}{h_n}c_{n-1} + \frac{2}{h_n}c_n = 3\frac{y_n-y_{n-1}}{h_n^2} \end{cases}$$

Periodic cubic spline interpolation

$$s'(t_0) = s'(t_n)$$
$$s''(t_0) = s''(t_n)$$

produces an $n \times n$-linear system with s.p.d. coefficient matrix: TODO: REDO MATRIX

$$\begin{pmatrix} a_1 & b_1 & 0 & \cdots & 0 & b_0 \\ b_1 & a_2 & b_2 & \ddots & & 0 \\ 0 & b_2 & \ddots & \ddots & \ddots & \vdots \\ \vdots & \ddots & \ddots & \ddots & b_{n-2} & 0 \\ 0 & & \ddots & b_{n-2} & a_{n-1} & b_{n-1} \\ b_0 & 0 & \cdots & 0 & b_{n-1} & a_0 \end{pmatrix}$$

Matlab function: v = spline(t,y,x)

32.2. Shape Preserving Spline Interpolation

Since the cubic spline interpolant is *not* monotonicty or curvature-preserving. We fix the slopes c_i in the nodes using the harmonic mean of data slopes Δ_j, the final interpolant will be tangents of these segments in the points $(t_i, y_i$. If (t_i, y_i) is a local maximum or minimum of the data, c_j is set toi zero.

$$c_i = \begin{cases} \frac{2}{\Delta_i^{-1} + \Delta_{i+1}^{-1}} & \text{if } sign(\Delta_i) = sign(\Delta_{i+1}) \\ 0 & \text{otherwise} \end{cases}$$
$$c_0 = 2\Delta_1 - c_1$$
$$c_n = 2\Delta_n - c_{n-1}$$
$$\Delta_j = \frac{y_j - y_{j-1}}{t_j - t_{j-1}}$$

Part XI.
Numerical Quadrature

Approximate evaluation of $\int_\Omega f(x)dx$, integration domain $\Omega \subset \mathbb{R}^d$. Continous function $f : \Omega \subset \mathbb{R}^d \mapsto \mathbb{R}$ only available as `function y=f(x)` (point evaluation).

33. Quadrature Formulas

n-point *quadrature formula* on

$$\begin{aligned}[a,b]: &\quad \int_a^b f(t)dt \approx Q_n(f) = \sum_{j=1}^{n} w_j^n f(\xi_j^n) \\ w_j^n: &\quad \text{Quadrature weights } \in \mathbb{R} \\ \xi_j^n: &\quad \text{Quadrature nodes } \in [a,b]\end{aligned}$$

Given Quadrature formula $(\hat{\xi}_j, \hat{w}_j)_{j=1}^n$ on *reference interval* $[-1,1]$

Idea Transformation formula for integrals

$$\int_a^b f(t)dt = \frac{1}{2}(b-a)\int_{-1}^{1} \hat{f}(\tau)d\tau = \frac{1}{2}(b-a)\int_{-1}^{1} \hat{f}\left(\frac{1}{2}(1-\tau)a + \frac{1}{2}(\tau+1)b\right)d\tau$$

34. Polynomial Quadrature Formulas

Idea replace integrand f with $p_{n-1} \in \mathcal{P}_{n-1}$ = polynomial interpolant of f for given interpolation nodes $\{t_0, \dots, t_{n-1}\} \subset [a,b]$.

$$\int_a^b f(t)dt \approx Q_n(f) := \int_a^b p_{n-1}(t)dt$$

Newton Cotes Formulas

$n = 1$: **Trapezoidal rule** (order 2)

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

$n = 2$: **Simpson rule** (order 4)

$$\int_a^b f(t)dt \approx \frac{b-a}{6}\left(f(a) + 4f\left(\frac{a+b}{2}\right) + f(b)\right)$$

$n \geq 8$: **Quadrature formulas with negative weights**

$$\int_a^b f(t)dt \approx \frac{b-a}{28350}\left(989f(a) + 5888f\left(\frac{b-a}{8}\right) - 928f\left(\frac{b-a}{4}\right) + 10496f\left(3\frac{b-a}{8}\right) - \right.$$
$$\left. 4540f\left(\frac{b-a}{2}\right) + 10496f\left(5\frac{b-a}{8}\right) - 928f\left(3\frac{b-a}{4}\right) + 5888f\left(7\frac{b-a}{8}\right) + 989f(b)\right)$$

Warning: *Negative weights compromise numerical stability.*

Quadrature Error

Quadrature error estimates directly from L^∞-interpolation error for Lagrangian interpolation with polynomial degree $n-1$

$$f \in C^n([a,b]) \;\Rightarrow\; \left|\int_a^b f(t)dt - Q_n(f)\right| \le \frac{1}{n!}(b-a)^{n+1}\left|\left|f^{(n)}\right|\right|_{L^\infty([a,b])}$$

35. Composite Quadrature

With $a = x_0 < x_1 < \ldots < x_{m-1} < x_m = b$

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

- Partition integration domain $[a,b]$ by *mesh*
- Apply the quadrature formulas from above on the sub-intervals

Theorem: Convergence of composite quadrature formulas

For a composite quadrature formula Q based on a local quadrature formula of order $p \in \mathbb{N}$ holds:

$$\exists C > 0 : \left|\int_I f(t)dt - Q(f)\right| \le Ch^p \left|\left|f^{(p)}\right|\right|_{L^\infty(I)} \qquad h: \text{max Mesh width}$$

Lemma: Bound for order of quadrature formula

There is no n-point quadrature of order $2n+1$

36. Gauss Quadrature

Necessary & Sufficient conditions of order 4

$$Q_n(p) = \int_a^b p(t)dt \;\forall p \in \mathcal{P}_3 \quad\Leftrightarrow\quad Q_n(t^q) = \frac{1}{q+1}(b^{q+1} - a^{q+1}), \qquad q = 0,1,2,3$$

This gives us 4 equations:

$$\begin{aligned}
\int_{-1}^{1} 1dt &= 2 = 1w_1 + 1w_2 \\
\int_{-1}^{1} tdt &= 0 = \xi_1 w_1 + \xi_2 w_2 \\
\int_{-1}^{1} t^2 dt &= \frac{2}{3} = \xi_1^2 w_1 + \xi_2^2 w_2 \\
\int_{-1}^{1} t^3 dt &= 0 = x_1^3 w_1 + \xi_2^3 w_2
\end{aligned}$$

$$\begin{aligned}
\Longrightarrow \quad w_1 &= 1 \\
w_2 &= 1 \\
\xi_1 &= \frac{1}{3} \\
\xi_2 &= \frac{-1}{3}
\end{aligned}$$

Theorem: Existence of n-point quadrature formulas of order $2n$

Let $\{\overline{P}_n\}_{n\in\mathbb{N}_0}$ be a family of non-zero polynomials that satisfies

- $\overline{P}_n \in \mathcal{P}_n$
- $\int_{-1}^{1} q(t)\overline{P}_n(t)dt = 0$ for all $q \in \mathcal{P}_{n-1}$
- The set $\{\xi_j^n\}_{j=1}^m w_j^n f(\xi_j^n)$ of real zeros of $\overline{\P}_n$ is contained in $[-1, 1]$

$$\text{then} \quad Q_n(f) = \sum_{j=1}^{m} w_j^n f(\xi_j^n)$$

Lemma: Zeros of Legendre Polynomials

P_n has n *distinct* zeros in $]-1, 1[$.

Part XII.
Integration of Ordinary Differential Equations: Single Step Methods

37. Initial Value Problems (IVP) for ODEs

Initial value problem (**IVP**) for *first-order ordinayr differential equation* (**ODE**)

$$\dot{y} = f(t, y)$$
$$y(t_0) = y_0$$

- $f : I \times D \mapsto \mathbb{R}^d$ (= right hand side) given in procedural form: `function v = f(t,y)`
- $I \subset \mathbb{R}$ (= time interval)
- $D \subset \mathbb{R}^d$ (= state space / phase space)
- $\Omega = I \times D$ (= extended state space)
- t_0 (= initial time)
- y_0 (= initial value)

For $d > 1$: $\dot{y} = f(t, y)$ can be viewed as a *system of ordinary differential equations.*

$$\dot{y} = f(y) \quad \Longleftrightarrow \quad \begin{pmatrix} y_1 \\ \dots \\ y_n \end{pmatrix} = \begin{pmatrix} f_1(t, y_1, \dots, y_n) \\ \dots \\ f_n(t, y_1, \dots, y_n) \end{pmatrix}$$

$$\text{Autonomous IVP } H : \quad \begin{cases} \dot{y} & = f(y) \\ y(0) & = y_0 \end{cases}$$

Assumption: Global Solutions
All solutions of H are global. $J(y_0) = \mathbb{R}$ for all $y_0 \in D$

Definition: Evolution Operator
Under the Assumption above the mapping:

$$\Phi : \begin{cases} \mathbb{R} \times D & \mapsto D \\ (t, y_0) & \mapsto \Phi'(y_0) = y(t) \end{cases}$$

where $t \mapsto y(t) \in C^1(\mathbb{R}, \mathbb{R}^d)$ is the unique (global) solution of the IVP $\dot{y} = f(y)$. $y(0) = y_0$ is the *evolution operator* for the autonomous ODE $\dot{y} = f(y)$

38. Euler Methods

Idea *timestepping*: successive approximation of evolution on small intervals $[t_{k-1}, t_k]$
Approximation of solution on $[t_{k-1}, t_k]$ by *tangent* curve to current global condition.

Explicit Euler Method
Explicit euler method generates a sequence by the recursion:

$$y_{k+1} = y_k + h_k f(t_k, y_k) \qquad k = 0, \dots, N-1$$

with local *timestep* $h_k = t_{k+1} - t_k$

Implicit Euler Method

$$y_{k+1} = y_k + h_k f(t_{k+1}, y_{k+1})$$

39. Convergence of Single Step Methods

$$e_{k+1} = \Psi^{h_k} y_k - \Psi^{h_k} y(t_k) = \underbrace{\Psi^{h_k} y_k - \Psi^{h_k} y(t_k)}_{\text{propagated error}} + \underbrace{\Psi^{h_k} y(t_k) - \Psi^{h_k} y(t_k)}_{\text{one-step erro}}$$

40. Runge-Kutta Methods

$$\left.\begin{aligned} \dot{y}(t) &= f(t, y(t)) \\ y(t_0) &= y_0 \end{aligned}\right\} \quad \Rightarrow \quad y(t_1) = y_0 + \int_{t_0}^{t_1} f(\tau, y(\tau)) d\tau$$

Idea Approximate integral by means of s-point quadratur formula defined on reference interval $[0, 1]$ with nodes $c_1, \dots, c_s$ and weights $b_1, \dots, b_s$

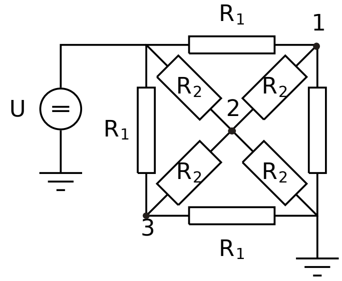
$$y(t_1) \approx y_1 = y_0 + h \sum_{i=1}^{s} b_i f\big(t_0 + c_i h, y(t_0 + c_i h)\big) \qquad h = t_1 - t_0$$

Get $y(t_0 + c_i h)$ by *bootstrapping*.

Part XIII.
Applications

41. Electrical circuits

Consider the following linear circuit ($U_{input} = U,\ R_1,\ R_2$ given). Since the circuit is grounded: $U_{ground} = 0$



To derive the linear system of equations. One has to look at the specific nodes and create an LSE:

Node 1

$$\frac{1}{R_1} \cdot (U_1 - U) + \frac{1}{R_1} \cdot (U_1 - 0) + \frac{1}{R_2} \cdot (U_1 - U_2) = 0$$
$$\implies \qquad \left(2\frac{1}{R_1} + \frac{1}{R_2}\right) \cdot U_1 - \frac{1}{R_2} \cdot U_2 = \frac{1}{R_1} \cdot U$$

Node 2

$$-\frac{1}{R_2} \cdot U_1 + 4\frac{1}{R_2} \cdot U_2 - \frac{1}{R_2} \cdot U_3 = \frac{1}{R_2} \cdot U$$

Node 3

$$-\frac{1}{R_2} \cdot U_2 + \left(2\frac{1}{R_1} + \frac{1}{R_2}\right) \cdot U_3 = \frac{1}{R_1} \cdot U$$

Now we are able to create the LSE:

$$\begin{pmatrix} 2\frac{1}{R_1} + \frac{1}{R_2} & -\frac{1}{R_2} & 0 \\ -\frac{1}{R_2} & 4\frac{1}{R_2} & -\frac{1}{R_2} \\ 0 & -\frac{1}{R_2} & 2\frac{1}{R_1} + \frac{1}{R_2} \end{pmatrix} \begin{pmatrix} U_1 \\ U_2 \\ U_3 \end{pmatrix} = \begin{pmatrix} \frac{1}{R_1}U \\ \frac{1}{R_2}U \\ \frac{1}{R_1}U \end{pmatrix}$$