Numerical Programming 1 (CSE @ TUM) cheat sheet

Felix Köhler

May 24, 2021

Contents

1 General **4**
1.1 Taylor Series . . . 4
1.2 Euler's Formula . . . 4
1.3 Trigonometric Identities . . . 4
1.4 L'Hospital's Rule . . . 4
1.5 Total Derivative . . . 4
1.6 Matrix Norms . . . 5
1.7 Implicit Function Theorem . . . 5
1.8 Gershgorin Disks . . . 5
1.9 Matrix Properties . . . 6
1.10 Determinant of a square matrix . . . 6
1.11 Determinant of a block matrix by its blocks . . . 6
1.12 Taylor Expansion of Exponential Function . . . 6
1.13 Sum series . . . 6

2 Polynomials **6**
2.1 Monomial Basis . . . 6
2.2 Lagrange Polynomials . . . 7
2.3 Chebychev Polynomials . . . 7
2.4 Hermite Polynomials . . . 7
2.5 Legendre Polynomials . . . 7
2.6 Bernoulli Polynomials . . . 7

3 Error Analysis **8**
3.1 Condition Number . . . 8
3.2 Loss of significant digits . . . 9
3.3 Rounding errors in floating point numbers . . . 9
3.4 Machine Precision . . . 9
3.5 Unique solution to a perturbed linear system . . . 9
3.6 Forward stability . . . 9

3.7 Backward Stability 10

4 Matrix Factorization **10**
4.1 LU Decomposition with Gauss-Jordan/Dolittle Algorithm . . 10
4.2 QR Decompostion with Householder transformations 10
4.3 QR Decomposition with Givens Rotations 10
4.4 Cost of Decompositions 11

5 Polynomial Interpolation **11**
5.1 Lagrangian Form 11
5.2 Direct - Barycentric Lagrange Interpolation 12
5.3 Recursive - Aitken-Neville Algorithm 12
5.4 Recursive - Newton Algorithm 13
5.5 Horner's scheme for efficient evaluation 13
5.6 Error in Polynomial Interpolation 13
5.7 Chebychev nodal support for converging interpolation 14
5.8 Cubic Splines 14
5.9 Cost of interpolation 16

6 Trigonometric Interpolation **16**
6.1 Continous Fourier Transform 16
6.2 Discrete Fourier Transform (DFT) 17
6.3 Fast Fourier Transform (FFT) 17

7 Quadrature - Numerical Integration **18**
7.1 Newton-Cotes-Methods 18
7.2 Adaptive Composite Rules 18
7.3 Romberg's Extrapolation Method 18
7.4 Gauss-Legendre Quadrature 19

8 Eigenvalues **19**
8.1 Power Iteration 19
8.2 Inverse Power Iteration 20
8.3 QR Algorithm 20
8.4 Shifts 21
8.5 Rayleigh Shift 21
8.6 Wilkinson Shift 21

9 Nonlinear Equations **21**
9.1 General 21
9.2 Chord Method 21
9.3 Secant Method 21
9.4 Regula Falsi - False Positive 22
9.5 Anderson-Björck 22
9.6 Newton's Method 22

9.7 Simplified Newton 22
9.8 Bisection Method 22
9.9 Fixed-Point Methods 22
9.10 Order of Convergence for Contractive Fixed-Point Methods . 23
9.11 Fixed Point Theorem 23

1 General

1.1 Taylor Series

One-dimensional Taylor-Series Expansion for a n+1 continously differentiable function on the the interval a to b with a $\xi \in [x, x+h]$

$$f(x+h) = \sum_{k}^{n} \frac{1}{k!} f^{(k)}(x) h^k + \frac{1}{(n+1)!} f^{(n+1)}(\xi) h^{n+1} \tag{1}$$

1.2 Euler's Formula

$$e^{ix} = \cos(x) + i\sin(x) \tag{2}$$

1.3 Trigonometric Identities

$$\sin(u \pm v) = \sin(u)\cos(v) \pm \cos(u)\sin(v) \tag{3}$$

$$\cos(u \pm v) = \cos(u)\cos(v) \mp \sin(u)\sin(v) \tag{4}$$

1.4 L'Hospital's Rule

Technique to evaluate the limits of indertminate forms.

If $\lim_{x\to c} f(x) = \lim_{x\to c} g(x) = 0$ or $\pm\infty$ and $g'(x) \neq 0 \forall x \neq c$ then

$$\lim_{x\to c} \frac{f(x)}{g(x)} = \lim_{x\to c} \frac{f'(x)}{g'(x)} \tag{5}$$

Can be extended to higher order derivatives.

1.5 Total Derivative

First order

$$\mathrm{d}f(x_1, x_2, ..., x_n) = \sum_{i=1}^{n} \frac{\partial f(\mathbf{x})}{\partial x_i} \mathrm{d}x_i \tag{6}$$

Second Order

$$\mathrm{d}^2 f = \mathrm{d}(\mathrm{d}f) = ... = \frac{\partial^2 f}{\partial x^2}\mathrm{d}x^2 + 2\frac{\partial^2 f}{\partial x \partial y}\mathrm{d}x\mathrm{d}y + \frac{\partial^2 f}{\partial y^2}\mathrm{d}y^2 \tag{7}$$

1.6 Matrix Norms

Adhere to the regular norm properties (greater than zero, homogenity, triangular inequality)

$$||\mathbf{A}|| > 0, \quad \forall \mathbf{A} \neq \mathbf{0}, \quad (||\mathbf{A}|| = 0 <-> \mathbf{A} = \mathbf{0}) \tag{8}$$

$$||\alpha \mathbf{A}|| = |\alpha| ||\mathbf{A}|| \tag{9}$$

$$||\mathbf{A} + \mathbf{B}|| \leq ||\mathbf{A}|| + ||\mathbf{B}|| \tag{10}$$

Additionaly it can be *sub-multiplicative*, if

$$||\mathbf{CD}|| \leq ||\mathbf{C}|| \cdot ||\mathbf{D}|| \tag{11}$$

And *compatible* or *consistent* with a vector norm

$$||\mathbf{A}\vec{x}|| \leq ||\mathbf{A}|| ||\vec{x}|| \tag{12}$$

1.7 Implicit Function Theorem

States whether an implicitly given function $F(x, y) = 0$ can be expressed by an explicit relation between its arguments $y = y(x)$.

In higher dimensions: Let $\vec{F} : U \times V \to \mathbb{R}^n, \quad U \subseteq \mathbb{R}^m, \quad V \subseteq \mathbb{R}^n$.

In one dimensions with a two-dimensional input: Let $g(x, y) = 0$ be an implicitly given function. Since g is zero on the entire input set its (total) derivative at every function point must also be zero. This can then be rearranged to get the info of the derivative of the explicit functions with respect to the other variable

$$y^{'}(x, y) = -\frac{\frac{\partial g(x,y)}{\partial x}}{\frac{\partial g(x,y)}{\partial y}}, \quad x^{'}(x, y) = -\frac{\frac{\partial g(x,y)}{\partial y}}{\frac{\partial g(x,y)}{\partial x}} \tag{13}$$

So at a given point (x_0, y_0) that satisfies the implicit equation $g(x_0, y_0)$ (use e.g. root finding methods to get such a value pair) we can also easily get the values of the derivatives of x and y. This can then be useful to approximately rebuild the explicit function by the means of a Taylor expansion.

1.8 Gershgorin Disks

Gershgorin disks give an estimate to where eigenvalues of a matrix lie in the complex plain. Assume $A \in \mathbb{C}^{nxn}$

$$R_i := \left\{ z \in \mathbb{C} || z - a_{ii}| \leq \sum_{\substack{j=1 \\ j \neq i}}^{n} |a_{ij}| \right\} \tag{14}$$

The the set of z (=R) describe the Gershgorin disks in which the eigenvalue can be found

1.9 Matrix Properties

- Normal Matrices commute with their hermitian $\mathbf{A}\mathbf{A}^T = \mathbf{A}^T\mathbf{A}$, every hermitian and every unitary matrix is normal
- Positive definite Matrices follow $\vec{x}^T\mathbf{A}\vec{x} >= 0, \forall\vec{x} \in \mathbb{R}^n$
- Positive definite Matrix if hermitian follows Sylvester's Criterion: All the principal minors and the matrix itself must have a positive determinant

1.10 Determinant of a square matrix

$$det(\mathbf{A}) := \sum_{j=1}^{n}(-1)^{j+1}a_{j1}det(\mathbf{A}_{j1}) \tag{15}$$

1.11 Determinant of a block matrix by its blocks

todo

1.12 Taylor Expansion of Exponential Function

Also valid matrix exponential

$$e^z = \sum_{k=0}^{\infty}\frac{z^k}{k!} \tag{16}$$

1.13 Sum series

Interrupted and to infinity geometric series
Leibniz-series error bound cut-off
Adam-Riese series

2 Polynomials

2.1 Monomial Basis

$$span\{1, x, x^2, x^3, ..., x^n\} = \Pi_n \tag{17}$$

2.2 Lagrange Polynomials

On a set of points t_i these polynomials are zero at $k \neq j$ and 1 at $k = i$

$$L_k(t) := \prod_{\substack{i=0 \\ i \neq j}}^{n} \frac{t - t_i}{t_k - t_i} \in \Pi_n \tag{18}$$

2.3 Chebychev Polynomials

The Chebychev polynomials are directly defined by

$$T_n(x) = \cos(n \arccos(x)) \tag{19}$$

Or by a three term recursion (it shares the first two polynomials with the common monomial basis)

$$T_n(x) = 2xT_{n-1}(x) - T_{n-2}(x), \quad T_0(x) = 1, \quad T_1(x) = x \tag{20}$$

MinMax Property: Chebychev polynomials minize the ∞-norm (max-norm) on the interval $x \in [-1, 1]$.

2.4 Hermite Polynomials

Require position and slope at each point. Can be used for Hermite cubic spline interpolation, however then only C^1 over the nodes

2.5 Legendre Polynomials

The Legendre Polynomials form a basis of the space of polynomials and are directly defined by

$$P_n(x) = \frac{1}{2^n n!} \frac{\mathrm{d}^n}{\mathrm{d}x^n}[(x^2 - 1)^n] \tag{21}$$

The recursive definition is given by

$$(n+1)P_{n+1}(x) = (2n+1)xP_n(x) - nP_{n-1}(x), \quad P_0 = 1, \quad P_1 = x \tag{22}$$

2.6 Bernoulli Polynomials

Form a basis of Π_n. With $b_0(x) := 1$ recursively defined as

$$b_k'(x) = b_{k-1}(x) \quad \text{and} \quad \int_0^1 b_k(x)\,\mathrm{d}x = 0 \tag{23}$$

The first three read

$$b_0(x) = 1, \quad b_1(x) = x - 1/2, \quad x^2/2 - x/2 + 1/12 \tag{24}$$

The integral fixes the integration constant.

3 Error Analysis

3.1 Condition Number

General Relative condition nuumber for $\mathbf{f} : \mathbb{R}^n -> \mathbb{R}^m$ if the peturbed input $\tilde{\mathbf{x}}$ approaches the non-perturbed

$$\frac{||\mathbf{f}(\mathbf{x}) - \mathbf{f}(\tilde{\mathbf{x}})||}{||\mathbf{f}(\mathbf{x})||} \leq \kappa_{rel}(\mathbf{f})(\mathbf{x}) \cdot \frac{||\mathbf{x} - \tilde{\mathbf{x}}||}{||\mathbf{x}||} \tag{25}$$

Additionally, one has to outfit both multidimensional quantitities with a corresponding norm (idealls, it is the same type of norm for both). If the function is continously differentiable we get the first order approximation to

$$\kappa_{rel}(\mathbf{f})(\mathbf{x}) \doteq \frac{||\mathbf{x}||}{||\mathbf{f}(\mathbf{x})||} ||\mathbf{f}'(\mathbf{x})|| \tag{26}$$

Where the derivative of the function is its Jacobian. Therefore, one has to define an appropriate matrix norm.

Name	Symbol	Condition number
Addition / Subtraction	$x + a$	$\left\vert \frac{x}{x+a} \right\vert$
Scalar Multiplication	ax	1
Division	$1/x$	1
Polynomial	x^n	$\vert n \vert$
Exponential function	e^x	$\vert x \vert$
Natural logarithm function	$\ln(x)$	$\left\vert \frac{1}{\ln(x)} \right\vert$
Sine function	$\sin(x)$	$\vert x \cot(x) \vert$
Cosine function	$\cos(x)$	$\vert x \tan(x) \vert$
Tangent function	$\tan(x)$	$\vert x(\tan(x) + \cot(x)) \vert$
Inverse sine function	$\arcsin(x)$	$\frac{x}{\sqrt{1-x^2}\arcsin(x)}$
Inverse cosine function	$\arccos(x)$	$\frac{\vert x \vert}{\sqrt{1-x^2}\arccos(x)}$
Inverse tangent function	$\arctan(x)$	$\frac{x}{(1+x^2)\arctan(x)}$

Figure 1: Overview of relative condition number of basic operations.

Relative condition number of a multi-variant problem with one-dimensional output

$$\kappa_{rel}(f, \vec{x}) = \frac{\sum_{i=1}^{n} |f_{x_i}(\vec{x})| \cdot |x_i|}{|f(\vec{x})|} \tag{27}$$

Condition number of a matrix (e.g. choose the infinity matrix norm - row sum norm)

$$\kappa_{rel}(A) = ||A|| \cdot ||A^{-1}|| \tag{28}$$

Condition number of a single component of $\vec{f}(\vec{x}) : \mathbb{R}^n \to \mathbb{R}^m$

$$\kappa_{ij} = \frac{x_j}{f_i(\vec{x})} \frac{\partial f_i(\vec{x})}{\partial x_j} \tag{29}$$

3.2 Loss of significant digits

$$\text{loss} = log_{10}(\kappa_{rel}) \tag{30}$$

3.3 Rounding errors in floating point numbers

Absolute rounding error

$$|fl(r) - r| \leq 0.5 \cdot B^E \tag{31}$$

Relative rounding error

$$\left| \frac{fl(r) - r}{r} \right| \leq \frac{1}{2|M|} \leq \frac{B}{2} B^{-t}, \quad \text{because} |M| \geq B^{t-1} \tag{32}$$

3.4 Machine Precision

Defined $\epsilon_0 := \epsilon_t := B^{1-t}$

Then we get an estimate on the rounded value

$$fl(r) = r \cdot (1 + \epsilon), \quad |\epsilon| \leq \frac{\epsilon_0}{2} \tag{33}$$

For the correct rounding we need $t + 2$ digits on the computer, if the result has to be accurate to t digits.

3.5 Unique solution to a perturbed linear system

Only guaranteed if the absolute error is bounded

$$||\delta \mathbf{A}|| < \frac{1}{||\mathbf{A}^{-1}||} \tag{34}$$

3.6 Forward stability

Algorithm is forward stable if it introduces errors that are in a similar order of magnitude as the unavoidable error due to the condition number. If the algorithm consists of sub-algorithms the following estimate holds:

$$\sigma_f \kappa_{rel}(f, x) \leq \sigma_g \kappa_{rel}(g, h(x)) + \sigma_h \kappa_{rel}(g, h(x)) \kappa_{rel}(h, x) \tag{35}$$

3.7 Backward Stability

Set the disturbed output of our imperfect algorithm subject to unperturbed input to be the output of a perfect algorithm of perturbed input data.

$$\tilde{f}(x) = \hat{y} \quad \rightarrow \quad f(\hat{x}) = y \tag{36}$$

If one can find an expression for $\hat{x}$ (i.e. how a perfect algorithm would map this perturbed data to the correct output), this can be uses as a stability estimator. If the backward error is bounded in a similar order of magnitude as the condition the algorithm is called backward stable.

Backwards stability is a stronger attribute but more difficult to analyze. If an algorithm is backward stable, it is also forward stable. The reverse is not necessarily true.

4 Matrix Factorization

QR decomposition is unique aside from a sign. LU decomposition is not unique and hardly depends on the pivoting strategy.

4.1 LU Decomposition with Gauss-Jordan/Dolittle Algorithm

Influence of Pivoting:

4.2 QR Decompostion with Householder transformations

Choose a vector by

$$\vec{u} = A[i:, i] + sign(A[i, i]) \cdot norm(A[i:, i]) \cdot \vec{e}_1 \tag{37}$$

Householder transformations are unitary and hermitian.

$$\mathbf{T}^{-1} = \mathbf{T}^H = \mathbf{T} = I_n - 2\frac{\vec{u}\vec{u}^H}{\vec{u}^H\vec{u}} \in \mathbb{C}^{nxn} \tag{38}$$

4.3 QR Decomposition with Givens Rotations

Givens rotations are orthogonal and independent on the sign of the rotation $\mathbf{R}(i, j, \phi)^{-1} = \mathbf{R}(i, j, -\phi) = \mathbf{R}(i, j, \phi)^T$. They differ only in 4 elements from the identity matrix $\mathbf{I}_n$. Assume you want to rotate the a_{ij} element of matrix $\mathbf{A}$ to zero. Then:

$$r = sgn(a_{jj})\sqrt{a_{jj}^2 + a_{ij}^2} \tag{39}$$

Whereas a_{jj} is the diagonal element in the column where we want to rotate the element and a_{ij} is the element we want to rotate to zero. Then

we get the values of cosine and sine (They never have to be calculated explicitly!)

$$c = \frac{a_{jj}}{r} \tag{40}$$

$$s = \frac{a_{ij}}{r} \tag{41}$$

Then the Givens rotation matrix is defined as follows:

$$\mathbf{G}(i,j) = \mathbf{I}_n, \text{with} \quad G_{jj} = G_{ii} = c, \quad G_{ij} = -s = -G_{ji} \tag{42}$$

To remember: The negative sine is always on a similar position as the element we want to rotate.

If applied to get a **QR** decomposition eleminate the non-zero entries below the main-diagonal by going columnwise from left to right and within the columns from bottom to top.

4.4 Cost of Decompositions

- **LU** $\propto 2\frac{n^3}{3}$
- Householder **QR** $\propto 4\frac{n^3}{3}$
- Givens **QR** $\propto 6\frac{n^3}{3}$

5 Polynomial Interpolation

Polynomial Interpolation is unique because the the determinant of the Vandermonde matrix

$$\det \begin{pmatrix} 1 & \dots & t_0^n \\ \dots & & \dots \\ 1 & \dots t_n^n & \end{pmatrix} = \prod_{i=0}^{n} \prod j = i + 1^n (t_j - t_i) \tag{43}$$

is non-zero if nodes are distinct. If not additional information on the derivative has to be given.

5.1 Lagrangian Form

Calculate the Lagrange Polynomials on the interpolation nodes and then the interpolating polynomial is given by

$$P_n(t) = \sum_{k=0}^{n} y_k L_k(t) \tag{44}$$

5.2 Direct - Barycentric Lagrange Interpolation

Rewrite the Lagrange Polynomial

$$L_k(t) = \omega_{n+1}(t)\frac{w_k}{t - t_k} \tag{45}$$

With the nodal polynomial of degree $n + 1$

$$\omega_{n+1}(t) := \prod_{i=0}^{n}(t - t_i) \tag{46}$$

Then the weights can be precomputed

$$w_k := \frac{1}{\prod_{k \neq j}(t_k - t_j} \tag{47}$$

And the interpolating polynomial is given by

$$P(t) = \omega_{n+1}(t) \sum_{k=0}^{n} \frac{w_k}{t - t_k} y_k \tag{48}$$

Better stability properties than regular Lagrangian interpolation.

5.3 Recursive - Aitken-Neville Algorithm

Polynomial interpolation with Aitken-Neville of a set of $n + 1$ data points to interpolate a polynomial of degree n in the form $P(x) = p_{0,n}(x)$. Start by setting the initial values in the recursive scheme to $p_{i,i}(x) = y_i$. Then calculate the other values by:

$$p_{i,j}(x) = \frac{(x - x_j)p_{i,j-1}(x) - (x - x_i)p_{i+1,j}(x)}{x_i - x_j} \tag{49}$$

More efficient evaluation of the Aitken-Polynomial at one certain point

$$seematlab \tag{50}$$

Aitken's theorem states that if two polynomial functions $M(t)$ and $N(t)$ interpolate $n - 2$ common points, and $M(t)$ is additionally interpolating a point before all the others and $N(t)$ is addtionally interpolating a point at the end of the interval, then there is a unique polynomial $P(t)$ that interpolates all value pairs as follows

$$P(t) = N(t) + \frac{t - t_k}{t_k - t_i}(N(t) - M(t)) \tag{51}$$

5.4 Recursive - Newton Algorithm

Newton's Method provides a nice evalutation of a polynomial interpolation at many points since it easily gives you the functional form of the interpolating polynomial. Define the divided differences

$$f[t_i, t_{i+1}, ..., t_k] = \frac{f[t_{i+1}, ..., t_k] - f[t_i, ..., t_{k-1}]}{t_k - t_i}, \quad f[t_i] := f(t_i) = y_i \tag{52}$$

Then the polynomial is defined as

$$P(t) = \sum_{k=0}^{n} f[t_0, ..., t_k] \prod_{i=0}^{k-1} (t - t_i) \tag{53}$$

If a derivative is given instead of a point, or an abscissa appears double and it therefore has to be given the first derivative. The Newton's divided difference becomes accurate to

$$f[x_0, \ldots, x_k] = \frac{f^{(k)}(x^*)}{k!}, \quad \text{if} x^* := x_0 = \cdots = x_k \tag{54}$$

Mind the $k!$ in the denominator (relevant for $k \geq 2$).

If the distance between points is equal (e.g. on an equidistant measurement grid) we get

$$f[x_0, \ldots, x_k] = todo, seewikipedia \tag{55}$$

5.5 Horner's scheme for efficient evaluation

Using Horner's scheme, the polynomial can be evaluated more efficiently

```
P_n := f[t_0, t_1, ..., t_n]
k=n-1:-1:0:
    P_k := P_{k+1} * (t - t_k) + f[t_0, t_1, ..., t_k]
```

5.6 Error in Polynomial Interpolation

Compare the maximum componentwise error in approximation

$$||P_1 - P_2||_\infty = \left|\left| \sum_{k=0}^{n} y_k L_k - \sum_{k_0}^{n} \tilde{y}_k K_k \right|\right| =\leq \max_{k=0...n} |y_k - \tilde{y}_k| \cdot \max_{t \in [t_0, t_n]} \left| \sum_{k=0}^{n} L_k(t) \right| \tag{56}$$

The one can identify the Lebesguq constant for the lower bound of the error on equidistant grids

$$\Lambda(n) := \max_{t\in[t_0,t_n]} \sum_{k=0}^{n} L_k(t) \sim \frac{2^{n+1}}{n\ln(n)} \tag{57}$$

This can lead to a definition of the conditioning for interpolation tasks on equidistant grids.

For Chebychev grids these values increase way slower.

That means the choice of node placements has an important influence on the accuracy of our interpolation. Assume Q_n is the best interpolating polynomial (best choice of node placement) and P_n is our interpolating polynomial (e.g. equidistant nodes). Then

$$||f - P_n||_\infty \leq (1 + \Lambda(n)) \cdot ||f - Q_n||_\infty \tag{58}$$

5.7 Chebychev nodal support for converging interpolation

Chebychev nodes on an interval from (a, b). n nodes with index k. They are the roots of the Chebychev polynomial of degree n.

$$x_k = \frac{1}{2}(a+b) + \frac{1}{2}(b-a)\cos\left(\frac{2k-1}{2n}\pi\right) \tag{59}$$

5.8 Cubic Splines

For efficient evaluation of cubic splines, use a different basis instead of the standard monomial one (where we would have to solve for all 4 coefficients on each sub-interval) - This basis has no special name

$$\Pi_3 = \text{span}\left\{1-\xi, \xi, -\frac{\xi}{3} + \frac{\xi^2}{2} - \frac{\xi^3}{6}, -\frac{\xi}{6} + \frac{\xi^3}{6}\right\}, \quad \xi = \frac{t - t_i}{t_{i+1} - t_i} \tag{60}$$

Then the interpolating polynomial is given by

$$s_3(t) = y_i(1-\xi) + y_{i+1}\xi + h_i^2 y_i^{''}\left(-\frac{\xi}{3} + \frac{\xi^2}{2} - \frac{\xi^3}{6}\right) + h_i^2 y_{i+1}^{''}\left(-\frac{\xi}{6} + \frac{\xi^3}{6}\right) \tag{61}$$

This is beneficial since y_i is already available due to the inerpolating condition. The second derivatives at each node can be efficiently computed. Introduce the following abbreviations ($\mu_i + \lambda_i = 1$)

$$\mu_i := \frac{h_{i-1}}{h_{i-1} + h_i} = \frac{t_i - t_{i-i}}{t_{i+1} - t_{i-1}} > 0 \tag{62}$$

$$\lambda_i := \frac{h_i}{h_{i-1} + h_i} = \frac{t_{i+1} - t_i}{t_{i+1} - t_{i-1}} > 0 \tag{63}$$

Then the following triagonal (non-square since we have 2 DoF) system has to be solved

$$\mu_i y_{i-1}^{''} + 2y_i^{''} + \lambda_i y_{i+1}^{''} = 6y_{i-1,i,i+1} \tag{64}$$

This creates $(n-2)xn$ system for the $n-2$ interior nodes. It uses the second devided difference (refer to Newton's interpolation scheme) defined by

$$y_{i-1,i,i+1} = \frac{y_{i,i+1} - y_{i-1,i}}{h_{i-1} + h_i} \tag{65}$$

With the regular divided differenc given by

$$y_{i,i+1} = \frac{y_{i+1} - y_i}{h_i} \tag{66}$$

This system is underconstrained. We have to prescribe boundary conditions. These can be of following type

- Hermite splines $s_3^{'}(t_0) = y_0^{'}$ and $s_3^{'}(t_n) = y_n^{'}$ are prescribed
- $s_3^{''}(t_0) = y_0^{''}$ and $s_3^{''}(t_n) = y_n^{''}$ are prescribed. "Natural Spline" if both are set to zero.
- Periodic Splines $s_3^{'}(t_0) = s_3^{'}(t_n)$ and $s_3^{''}(t_0) = s_3^{''}(t_n)$ are prescribed

By the help of the Gershgorin Disks we can proof that under a given bounary condition the cubic spline is unique.

Since we require the function to be interpolated to be $f \in C^4$ we can define two upper bounds.

$$L := \max_{t \in [a,b]} |f''''(t)| \tag{67}$$

$$H := \max_{0 \leq i \leq n-1} h_i \tag{68}$$

Then upper bounds for the error on each subinterval can defined to

$$|f - s_3| \leq \frac{7}{64} L \cdot H^2 \cdot h_i^2 \tag{69}$$

$$|f' - s_3'| \leq \frac{7}{16} L \cdot H^2 \cdot h_i \tag{70}$$

$$|f'' - s_3''| \leq \frac{7}{8} L \cdot H^2 \tag{71}$$

$$|f''' - s_3'''| \leq 2 \cdot L \cdot H^2 / h_i \tag{72}$$

By this we can conclude that that the splines converge against the function they are interpolating for increasingly finer subintervals. For Polynomials this is not given (only on Chebychev nodes).

5.9 Cost of interpolation

- Lagrangian interpolation, each evaluation $O(n^2)$ additions and multiplications, adding new pairs requires completely new calculation. Computation can becom numerically unstable
- Barycentric Lagrange: Computing weights $\propto n^2$, adding value $\propto n$, evaluating polynomial $\propto n$
- Aitken-Neville : Storage only needs $n+1$ dimensional vector
- Newton: todo
- Cubic Splines in alternative form: Calculate the second divided differences, calculate parameters λ_i and μ_i, then solve tridiagonal system (can be accomplished by Thomas Algorithm $\sim O(n)$)

6 Trigonometric Interpolation

Instead of Polynomial interpolation trigonometric interpolation uses periodic functions (e.g. $\sin(x)$ and $\cos(x)$ or e^{ix} in general) to interpolate a function on an interal $[a, b]$. The interpolation conditions holds and for the actual task one has not to provide a function value for b (i.e. $f(b)$) because the function is assumed to be periodic on this interval, so $f(a) = f(b)$.

A special case is the trigonometric interpolation with an equidistant spacing between the interpolating points on the basic periodic interval $[0, 2\pi]$:

$$x_k = \frac{k \cdot 2 \cdot \pi}{n}, \quad \text{with } k = 0, 1, \ldots, n-1 \tag{73}$$

Every other interval can be transformed to this interval by a linear transformation.

The trigonometric functions make up the basis of the L_2 space, i.e.

$$L_2 = span\{e^{jx}\}, \quad \text{with } j \in \mathbb{Z} \tag{74}$$

One is basically free to choose the n functions to interpolate the values with. However, similar to polynomial interpolation choosing only the lowest degree ones (in terms of trigonometric interpolation the ones with the lowest frequencies) one minimizes the interpolating error. Additionally the interpolating coefficients coincide with the values of the discrete Fourier Transform.

6.1 Continous Fourier Transform

Find the coefficients by

$$c_k := \frac{1}{2\pi} \int_0^{2\pi} f(x) e^{-ikx} \, \mathrm{d}x, \quad k \in \mathbb{Z} \tag{75}$$

Get the Fourier series of f by

$$S_f(x) = f(x) = \sum_{-\infty}^{\infty} c_k e^{ikx} \tag{76}$$

6.2 Discrete Fourier Transform (DFT)

Discrete Function values are given on equidistant abscissae on $(0, 2\pi)$ exluding the right boundary of the interval.

The coeffients of the interpolation (these are approximations of the Fourier coefficients of the n lowest frequencies) are found by solving the system

$$\sum_{j=\lceil -n/2 \rceil}^{\lceil n/2 \rceil} \gamma_j e^{i \cdot j \cdot x_k} = y_k \tag{77}$$

Or in matrix vector notation

$$\mathbf{F}\vec{\gamma} = \vec{y}, \quad \text{with } F_{kj} = e^{i \cdot \frac{2\pi}{n} \cdot jk} \tag{78}$$

The matrix $\mathbf{F}$ is perfectly conditioned and almost unitary $\mathbf{F}^{-1} = \frac{1}{n}\mathbf{F}^H$. Therefore, in this very special case the system can be solved by using the inverse.

The one can assemble the approximate Fourier Series to

$$f(x) \approx T_n(x) = \sum_{j=\lceil -n/2 \rceil}^{\lceil n/2 \rceil} \gamma_j e^{i \cdot j} \tag{79}$$

Remarks:

- When interpolating real values, it holds $\gamma_k = \gamma_{-k}^-$
- When interpolating real values, the interpolating polynomial is also real (transform to it via Euler's Identity)

In case of n being even (this means an uneven distribution of the lowest frequncies around 0), it is arbitrary whether if one chooses $\lceil -n/2 \rceil$ or $\lceil n/2 \rceil$. Since this is not allowed to have an influence on the solution, the contribution by this frequency may only be real.

6.3 Fast Fourier Transform (FFT)

Performing the DFT takes $O(n^2)$ operations. This can be reduced by exploiting the shape of the matrix. By the help of permuations one can subdivide the matrix $\mathbf{F}$ into matrices of half sice, reducing the effort to $O(n \log(n))$ with the fastest result if n is a power of 2.

7 Quadrature - Numerical Integration

7.1 Newton-Cotes-Methods

Newton-Cotes type quadrature interpolates the integrand by the help of polynomials on an equidistant grid. Composite rules define their sub-interval length to $\frac{b-a}{n}$ with n sub-intervals.

name	definition	remainder
Trapezoid	$\frac{b-a}{2}(f(a)+f(b))$	$-\frac{f''(\eta)}{12}(b-a)^3$
Kepler	$\frac{b-a}{6}\left(f(a)+4f(\frac{a+b}{2})+f(b)\right)$	$-\frac{f^{(4)}(\eta)}{90}\left(\frac{b-a}{2}\right)^5$
Comp. Trapez.	$\frac{h}{2}\left(f(a)+2\sum_{j=1}^{n-1}f(x_j)+f(b)\right)$	$-\frac{b-a}{12}h^2f''(\eta)$
Simpson	$\frac{h}{3}\sum_{j=1}^{n/2}[f(x_{2j-2})+4f(x_{2j-1})+f(x_{2j})]$	$-\frac{h^4}{180}(b-a)f^{(4)}(\eta)$

7.2 Adaptive Composite Rules

- Start with course sub-intervals h_1
- Half the sub-interval-lengths everywhere $h_2 = h_1/2$
- For each interval calculate the error $err_j = |S_{\text{coarse}} - S_{\text{fine}}|$
- Weight each error with the sub-interval length $err_j \frac{h_j}{b-a}$
- Refine the sub-intervals further (recursively) if error is above a set tolerance

7.3 Romberg's Extrapolation Method

Let $T(h)$ denote the composite trapezoidal approximation of an integral with n subintervals each of length h. Then the error according to *Euler-Maclaurin* reads

$$R(h) = \sum_{k=1}^{n+1} b_{2k}(0)h^{2k}\cdot f^{(2k-1)}(x)\Big|_{x=a}^{x=b} - h^{2n+2}\int_a^b b_{2n+2}(\frac{x-x_i}{h})f^{(2n+2)}(x)\,\mathrm{d}x \tag{80}$$

According to this the error expands in h^2.

$$R(h) = c_1h^2 + c_2h^4 + c_3h^6 + \cdots + c_mh^{2m} + \alpha_{m+1}(h)h^{2m+2} \tag{81}$$

We know that we have the true solution of the integral for an infinitely small sub-interval width h. Therefore, we can expand the function $T(h)$ in terms of h^2 and make an extrapolation at $h = 0$.

Then, with the help of the Aitken-Algorithmi ($R_{i,i} = T(h_i)$), Romberg's method reads

$$R_{i,k} = \frac{h_k^2}{h_k^2 - h_i^2} R_{i,k-1} - \frac{h_i^2}{h_k^2 - h_i^2} r_{i+1,k} \tag{82}$$

In the case of the Romberg sequence of sub-interval length (halfing each time), we get

$$R_{i,k} = \frac{4^{-k}}{4^{-k} - 4^{-i}} R_{i,k-1} - \frac{4^{-i}}{4^{-k} - 4^{-i}} R_{i+1,k} \tag{83}$$

The error estimator is given as

$$R_n = O(\prod_{i=0}^{n-1} h_i^2) \tag{84}$$

One can derive the Trapezoidal-Rule and the 3/8 Rule with the help of Romberg but with a worse error estimator.

Mind that Richardson's idea is not applicable to periodic functions integrated over one period, since the error expansion term vanish. Outcome: split up quadrature interval into two !unequally long intervals! and apply Romberg on them.

7.4 Gauss-Legendre Quadrature

8 Eigenvalues

Eigenvalues should numerically never be determined by the roots of the characteristic polynomial. Instead use other algorithms like (inverse) power iteration or the QR algorithm. Then the condition number of finding an eigenvalue is given as

$$\kappa(\lambda) = ||\mathbf{V}|| \cdot ||\mathbf{V}^{-1}|| \tag{85}$$

With the eigenvector matrix $\mathbf{V}$. If the matrix $\mathbf{A}$ is normal, i.e. $\mathbf{A}\mathbf{A}^H = \mathbf{A}^H\mathbf{A}$, then $\mathbf{V}$ is unitary and the problem is perfectly conditioned.

8.1 Power Iteration

Choose an arbitrary vector $\vec{q_0}$. Then perform one iteration

$$\vec{q_{i+1}} = \vec{q_i}^H \mathbf{A} \vec{q_i} \tag{86}$$

Normalization ensures that no overflow occurs. The approximation to the biggest eigenvalue is given by the Rayleigh Quotient

$$\mu_i = \frac{\vec{q}_i^H \mathbf{A} \vec{q}_i}{\vec{q}_i^H \vec{q}_i} \tag{87}$$

Converges against the greatest eigenvalue. $\vec{q}$ is a bad approximation of the corresponding eigenvector.

Converging if, greatest eigenvalue is distinct or appears multiple times. If greatest eigenvalue also has its negative or is the complex conjugate, no convergence. Convergence is faster, the smaller the ratio $|\lambda_i/\lambda_1|$.

Define residual

$$\vec{r}_i = \mathbf{A}\vec{q}_i - \mu_i \vec{q}_i \tag{88}$$

Stopping criterion

$$\frac{||\vec{r}_i||}{|\mu_i| \cdot ||\vec{q}_i||} \leq \sigma \epsilon_0 \tag{89}$$

8.2 Inverse Power Iteration

Choose arbitrary vector $\vec{q}_0$ and initial guess σ_0 (for example by one step of a regualr power iteration.

Then perform one iteration

$$(\mathbf{A} - \mu \mathbf{I})\vec{q}_i = \vec{q}_{i-1} \tag{90}$$

Approximation to the eigenvalue is given by the Rayleigh Quotient. Converges against eigenvalue closest to guess μ. $\vec{q}$ converges quite well against the corresponding eigenvector.

Costs $O(n^2)$ flop per step when LU decomposition of shifted matrix is saved.

8.3 QR Algorithm

First preconditioning, transform general matrix into upper Hessenberg form or Hermitian matrix into real! tridiagonal form by QR modifications:

1. Choose a first column and get an Householder so that all entries below the subdiagonal would be zero
2. Apply $A \to T^{-1}AT$

Then iterator over the following algorithm

1. $Q_k R_k = R_{k-1}$ Decompose into QR of a previous
2. $T_k := R_k Q_k$ Reverse order of multiplication

8.4 Shifts

For QR-Algorithm.

1. $Q_k R_k = R_{k-1} - \mu I$ Decompose into QR of a previous matrix minus shift
2. $T_k := R_k Q_k + \mu I$ Reverse order of multiplication and add the shift

8.5 Rayleigh Shift

Use bottom right element as shift. Once element in subdiagonal in last row is close to zero, deflate the rest of the matrix.

Does not work for complex EWs.

8.6 Wilkinson Shift

Consider the matrix 2x2 block in the bottom right. Analytically calculate the eigenvalues and use this eigenvalue as a shift that is the closest to the bottom right element

9 Nonlinear Equations

9.1 General

Derivation: Take Taylor Series up to constant or linear term and approximate the derivatives.

General form:

$$x^{[n+1]} = x^{[n]} - \frac{f(x^{[n]}}{q^{[n]}} \tag{91}$$

Whereas the basic methods only differ in their choice or approximation of $q^{[n]}$

9.2 Chord Method

$$q_k = \frac{f(b) - f(a)}{b - a} = const \tag{92}$$

Onstep method, converges with $p = 1$

9.3 Secant Method

$$q_k = \frac{f(x_k) - f(x_{k-1})}{x_k - x_{k-1}} \tag{93}$$

Two-Step Method, converges locally with $p \approx 1.63$.

9.4 Regula Falsi - False Positive

Choose the two pairs of the values so that at least one zero of the function is between the secant.

Same computational effort as Secant Method but globally convergent with $p = 1$

9.5 Anderson-Björck

Modification of Regula Falsi, provides an "adaptive" mix between it and secant method

In case of simple roots, superlinear and globally converging with $p \approx 1.7$

Method of Choice in 1D cases.

9.6 Newton's Method

$$q_k = f'(x_k) \tag{94}$$

Converges locally with $p = 2$ for single roots and with $p = 2$ for multiple roots.

9.7 Simplified Newton

Derivative is calculated once ($p = 1$) or recalculated after n iteration steps.

9.8 Bisection Method

Consecutively half the search interval (divide and conquer approach) and choose that interval in which the sign of the function changes (indicates the root is in that interval).

Simple, extremely stable and globlly convergent (worse than linear)

Method of choice for multple roots, e.g. rough estimate by Newton or Anderson-Björck and then further refinement with Bisection.

9.9 Fixed-Point Methods

The presented iterative methods for root-finding form a subset of Fixed-Point Methods (ODE Methods, Eigenvalue methods etc. can also be categorized in this regard, but difficult to analyze).

Fixed point defined

$$\lim_{k \to \infty} x_k = x^*, \quad \Phi(x^*) = x^* \tag{95}$$

The Method is called contractive if it reduces the distance between two points in the subset it operates on

$$|\Phi(x) - \Phi(y)| \leq L \cdot |x - y|, \quad \forall x, y \in [a, b] \tag{96}$$

With a Lipschitz-Constant that has to be $L < 1$ for convergence.

If the the iteration $\Phi(x)$ is differentiable (!!! Important, no analysis for non-differential functions possible, errors not bounded), then

$$L \geq \left| \frac{\Phi(x) - \Phi(y)}{x - y} \right| = \Phi'(z), \quad \forall z \in [a, b] \tag{97}$$

9.10 Order of Convergence for Contractive Fixed-Point Methods

Φ converges (locally) to x^* with order $p \geq 1$, if for $\epsilon > 0$ properly chosen

$$\exists C > 0 : |\Phi(x) - x^*| \leq C|x - x^*|^p, \quad \forall x \in [x^* - \epsilon, x^* + \epsilon] \tag{98}$$

and $C < 1$ in case of linear ($q = 1$).

In other words: The distance to the fixed-point has to be reduced for in every iteration

!!! The order of convergence is equal to the order of the first non-vanishing derivative of Φ

9.11 Fixed Point Theorem

If (a) a fixed point method is confined to one interval and (b) contractive

$$x_k \in [a, b], \quad \forall k = 0, 1, 2, \ldots \tag{99}$$

$$|\Phi(x) - \Phi(y)| \leq L \cdot |x - y|, \quad \forall x, y \in [a, b] \quad \text{with} L < 1 \tag{100}$$

Then the following holds:

1. The fixed point exists in the limit in the interval $[a, b]$
2. it is the only fixed point in the interval
3. the fixed point method converges at least linearly

Additionally some insights on the distance to the fixed point,

$$|x_k - x*| \leq L^k |x_0 - x*| \tag{101}$$

$$|x_k - x*| \leq \frac{L^k}{1 - L} |x_0 - x_1| \tag{102}$$