Outline

Polynomial interpolation

Given:

$$(x_i, f_i), \quad i = 0, \ldots, n.$$

To compute: Polynomial $p(x)$ of degree smaller than or equal to n with

$$p(x_i) = f_i, \quad i = 0, \ldots, n.$$

Interpolation condition: The nodes $x_0, \ldots, x_n$ have to be pairwise different.

Theorem 6.1

For pairwise different grid points the interpolation polynomial is unique.

A simple approach to compute the polynomial:

$$p(x) = a_0 + a_1 x + a_2 x^2 + \ldots + a_n x^n.$$

System of linear equations (Vandermonde's matrix):

$$\begin{pmatrix} 1 & x_0 & x_0^2 & \ldots & x_0^n \\ 1 & x_1 & x_1^2 & \ldots & x_1^n \\ \vdots & \vdots & \vdots & \ddots & \vdots \\ 1 & x_n & x_n^2 & \ldots & x_n^n \end{pmatrix} \begin{pmatrix} a_0 \\ a_1 \\ \vdots \\ a_n \end{pmatrix} = \begin{pmatrix} y_0 \\ y_1 \\ \vdots \\ y_n \end{pmatrix}. \tag{6}$$

with

$$\det(V_{n+1}) = \prod_{1 \le i < j \le n} (x_i - x_j).$$

But: The problem is ill-conditioned. Use other approaches.

Outline

Definition 6.2

According to $x_0, \ldots, x_n$ the Lagrange basis functions are

$$l_i(x) = \prod_{j=0, j\neq i}^{n} \frac{x - x_j}{x_i - x_j}.$$

Obviously holds

$$l_i(x_\ell) = \prod_{j=0, j\neq i}^{n} \frac{x_\ell - x_j}{x_i - x_j} = \begin{cases} 1 & \text{for } \ell = i \\ 0 & \text{otherwise.} \end{cases}$$

Definition 6.3

The construction

$$p(x) = \sum_{i=0}^{n} f_i l_i(x) = \sum_{i=0}^{n} f_i \prod_{j=0, j\neq i}^{n} \frac{x - x_j}{x_i - x_j}$$

is called Lagrange polynomial.

Outline

Newton interpolation (1)

- successive structure, start with one interpolation point, then 2, 3, etc.
- degree of the polynomial increases by one per step,
- initialisation $p_0(x) = y_0$
- first iteration step

$$p_1(x) = p_0(x) + c_1(x - x_0), \qquad \text{with} \qquad c_1 = \frac{y_1 - y_0}{x_1 - x_0}.$$

Newton interpolation (2)

- further iteration from $p_{m-1}(x)$ to $p_m(x)$ by

$$p_m(x) = p_{m-1}(x) + c_m \prod_{i=0}^{m-1}(x - x_i),$$

- to have $p_m(x_m) = y_m$ next to $p_m(x_j) = y_j$ for $j = 0, \ldots, m-1$ use

$$c_m = \frac{y_m - p_{m-1}(x_m)}{\prod_{i=0}^{m-1}(x_m - x_i)}.$$

Definition 6.4

Using $f[x_0] = y_0$ the divided difference are

$$f[x_m, \ldots, x_0] := c_m = \frac{y_m - p_{m-1}(x_m)}{\prod_{i=0}^{m-1}(x_m - x_i)}, \qquad m = 1, 2, \ldots, n.$$

The representation of $p_m(x)$ in Newton's form is

$$p_m(x) = \sum_{i=0}^{m} f[x_i, \ldots, x_0] \prod_{j=0}^{i-1}(x - x_j).$$

Theorem 6.5

Let $x_0, \ldots, x_m$ be pairwise different and $f[x_0] = y_0, \ldots, f[x_m] = y_m$. We can apply the recursion

$$f[x_m, \ldots, x_0] = \frac{f[x_m, \ldots, x_1] - f[x_{m-1}, \ldots, x_0]}{x_m - x_0}.$$

Computation of coefficients using Newton's scheme:

- initialisation $f[x_i] = y_i$ for $i = 0, \ldots, n$,
- triangle scheme

$$
\begin{array}{c|cccc}
x_0 & f[x_0] & & & \\
x_1 & f[x_1] & \frac{f[x_1]-f[x_0]}{x_1-x_0} =: f[x_1,x_0] & & \\
x_2 & f[x_2] & \frac{f[x_2]-f[x_1]}{x_2-x_1} =: f[x_2,x_1] & \frac{f[x_2,x_1]-f[x_1,x_0]}{x_2-x_0} =: f[x_2,x_1,x_0] & \\
x_3 & f[x_3] & \frac{f[x_3]-f[x_2]}{x_3-x_2} =: f[x_3,x_2] & \frac{f[x_3,x_2]-f[x_2,x_1]}{x_3-x_1} =: f[x_3,x_2,x_1] & f[x_3,x_2,x_1,x_0] \\
\vdots & \vdots & \vdots & \vdots & \vdots
\end{array}
$$

- the interpolation polynomial reads

$$p(x) = f[x_0] + f[x_1,x_0](x-x_0) + f[x_2,x_1,x_0](x-x_0)(x-x_1) + \ldots$$

Example:

- interpolation of

$$(-1,5),\ (0,4),\ (1,-3),\ (2,3),$$

- polynomial of degree ≤ 3,
- Newton's scheme by divided differences

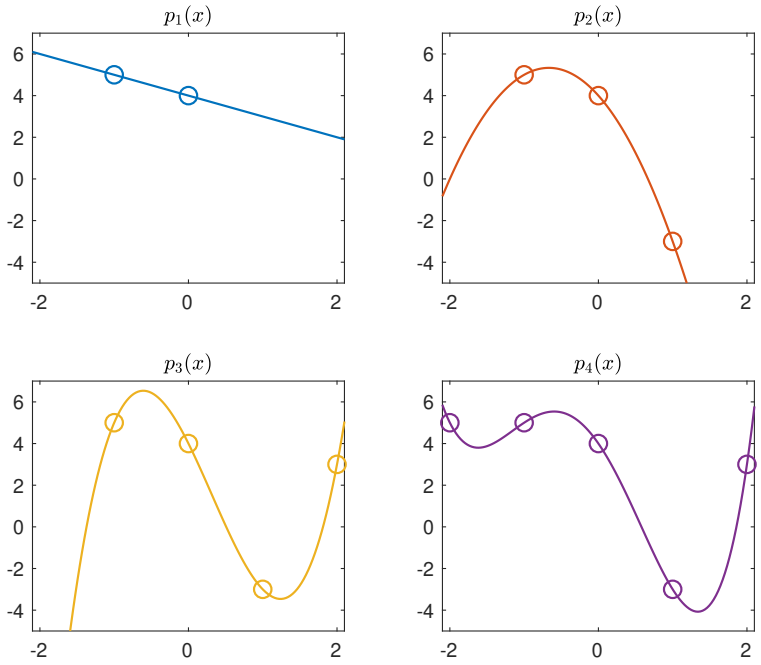
$$\begin{array}{c|cccc} -1 & 5 & & & \\ 0 & 4 & -1 & & \\ 1 & -3 & -7 & -3 & \\ 2 & 3 & 6 & \frac{13}{2} & \frac{19}{6} \end{array}$$

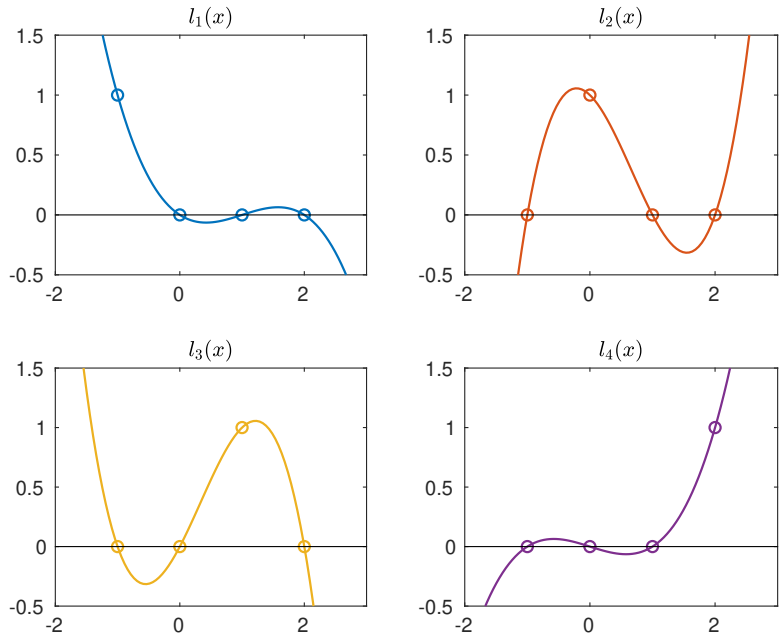
- interpolating polynomial is

$$\begin{aligned} p(x) &= 5 - 1(x+1) - 3(x+1)x + \frac{19}{6}(x+1)x(x-1) \\ &= 4 - \frac{43}{6}x - 3x^2 + \frac{19}{6}x^3. \end{aligned}$$

Newton and Lagrange interpolation

The single Newton steps (left) as well as the single Lagrange basis functions (right)





Both approaches result in the same polynomial

$$
\begin{aligned}
p(x) &= 5 - 1(x+1) - 3(x+1)x + \frac{19}{6}(x+1)x(x-1) \\
&= 5\frac{x(x-1)(x-2)}{-6} + 4\frac{(x+1)(x-1)(x-2)}{2} + (-3)\frac{(x+1)x(x-2)}{-2} + 3\frac{(x+1)x(x-1)}{6} \\
&= 4 - \frac{43}{6}x - 3x^2 + \frac{19}{6}x^3.
\end{aligned}
$$

Extension:

- consider that we add $(-2, 5)$ as additional point to the previous example,
- the extended scheme is

-1	5				
0	4	-1			
1	-3	-7	-3		
2	3	6	$\frac{13}{2}$	$\frac{19}{6}$	
-2	5	$-\frac{1}{2}$	$\frac{13}{6}$	$\frac{13}{6}$	1

- the new polynomial is

$$\tilde{p}(x) = 5 - 1(x+1) - 3(x+1)x + \frac{19}{6}(x+1)x(x-1) + (x+1)x(x-1)(x-2).$$

Outline

How about $\max_{x\in[a,b]} |f(x) - p(x)|$ for a function $f(x)$?

Theorem 6.6

Let $f(x)$ be $(n+1)$ times continously differentiable on the interval $[a,b]$. Further let $a \leq x_0 < x_1 < \cdots < x_n \leq b$ and $p(x)$ be the interpolating polynomial for $(x_i, f(x_i))$, $i = 0 \ldots, n$. Then it holds

$$|f(\tilde{x}) - p(\tilde{x})| \leq \frac{|w(\tilde{x})|}{(n+1)!} \max_{\xi\in[a,b]} \left|f^{(n+1)}(\xi)\right|$$

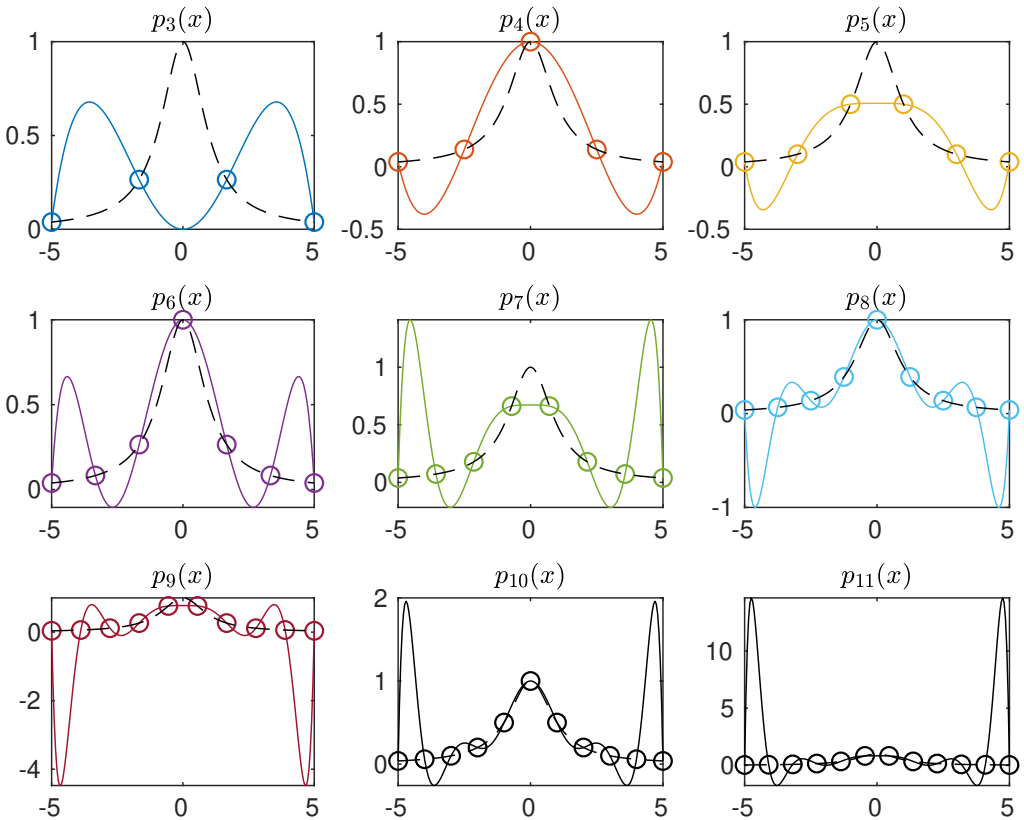
for

$$w(x) = \prod_{j=0}^{n} (x - x_j).$$

Limits of polynomial interpolation

Runge, 1901:

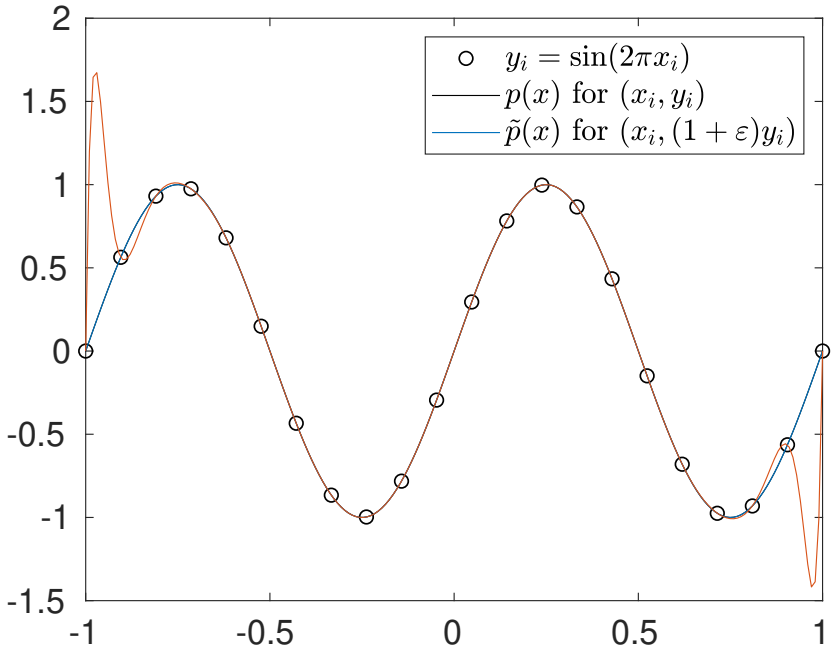
- interpol. of $(1+x^2)^{-1}$, polynomials of nth degree, equidistant nodes in $[-5,5]$,
- very bad approimations for large n, oscillations at the boundary.



Limits of polynomial interpolation

Sesitivity for noise:

- interpolation of $f(x) = \sin(2\pi x)$ using 22 equidistant knots on $[-1, 1]$,
- comparison of $p(x)$ for noisy and noise free data $\tilde{y}_i = (1 + \delta) f(x_i)$ with $\varepsilon = 10^{-4}$,
- large differences for the polynomial and high oscillations at the boundaries.



Outline

Modified problem:

- interpolation also for values of derivates of $f(x)$,
- nodes $x_0 < x_1 < \cdots < x_m$ and values $f_i^{(j)} = f^{(j)}(x_i)$ for $j = 0, \ldots, n_i - 1$,
- compute an interpolation polynomial $p(x)$ with

$$\deg(p(x)) \leq n, \quad n+1 = \sum_{i=0}^{m} n_i$$

such that

$$p^{(j)}(x_i) = f_i^{(j)}, \qquad j = 0, \ldots, n_i - 1, \ i = 0, \ldots, m. \tag{7}$$

- the interpolating polynomial is unique as $n+1$ degrees of freedom equals the number of conditions.

Scheme for a block to one node:

$$
\begin{array}{c|cccc}
x_0 & f[x_0] & & & \\
x_0 & f[x_0] & \frac{f'(x_0)}{1!} & & \\
x_0 & f[x_0] & \frac{f'(x_0)}{1!} & \frac{f''(x_0)}{2!} & \\
\vdots & \vdots & & & \ddots \\
x_0 & f[x_0] & \frac{f'(x_0)}{1!} & \cdots & \frac{f^{(n)}(x_0)}{n!}
\end{array}
$$

Interpolating polynomial

$$
p(x) = f(x_0) + \frac{f'(x_0)}{1!}(x - x_0) + \cdots + \frac{f^{(n)}(x_0}{n!}(x - x_0)^n
$$

Hermite Interpolation

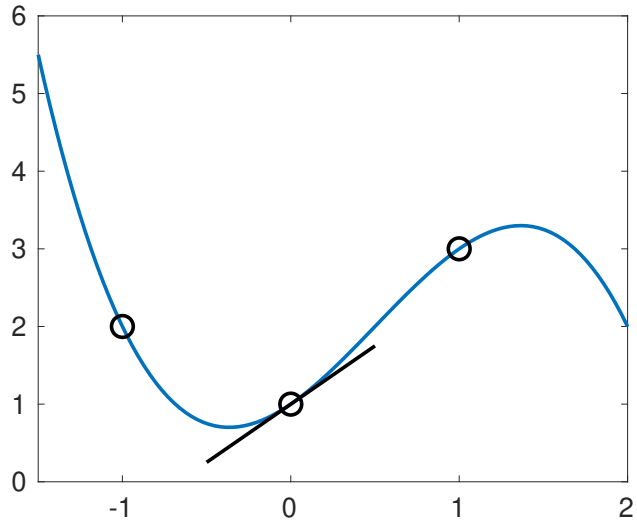
Example:

- interpolation of $(-1,2)$, $(0,1)$, $(1,3)$ with the additional condition $p'(0) = 1.5$,
- Newton's scheme

$$\begin{array}{c|cccc} -1 & 2 & & & \\ 0 & 1 & -1 & & \\ 0 & 1 & 1.5 = \frac{1.5}{1!} & 2.5 & \\ 1 & 3 & 2 & 0.5 & -1 \end{array}$$

- interpolation polynomial

$$p(x) = 2 - (x+1) + 2.5(x+1)x - 1(x+1)x^2.$$



Example:

- interpolation polynomial for $f(x) = 1/x$ using $x_0 = 1, x_1 = 2$,
- conditions

$$f(x_0) = 1,\ f'(x_0) = -1,\ f''(x_0) = 2,\ f'''(x_0) = -6$$
$$f(x_1) = 0.5,\ f'(x_1) = -0.25$$

- Newton's scheme

$$\begin{array}{c|cccccc} 1 & 1 & & & & & \\ 1 & 1 & -1 = \frac{-1}{1!} & & & & \\ 1 & 1 & -1 & 1 = \frac{2}{2!} & & & \\ 1 & 1 & -1 & 1 & -1 = -\frac{6}{3!} & & \\ 2 & 1/2 & -\frac{1}{2} & \frac{1}{2} & -\frac{1}{2} & \frac{1}{2} & \\ 2 & 1/2 & -\frac{1}{4} & \frac{1}{4} & -\frac{1}{4} & \frac{1}{4} & -\frac{1}{4} \end{array}$$

- interpolation polynomial

$$p(x) = 1 - (x-1) + (x-1)^2 - (x-1)^3 + \frac{1}{2}(x-1)^4 - \frac{1}{4}(x-1)^4(x-2).$$

Outline

Splines

Splines:

- many interpolating points,
- local interpolations for a few neighboring points instead of a global polynomial,
- piecewise composed polynomial using smoothness requirements at transitions.

Definition 6.7

The cubic splines are defined as follows: For the pairs (x_i, y_i), $i = 0, \ldots, n$, compute a spline S that interpolates these points such that

$$S_{|[x_i,\, x_{i+1}]}(x) := p_i(x) \qquad \text{is a polynomial of degree } 3$$

and to make sure that the function is twice continously differentiable at the transition points. This means S has to fulfill the four conditions

$$p_i(x_i) = y_i, \quad p_i(x_{i+1}) = y_{i+1}, \quad p_i'(x_i) = p_{i+1}'(x_i), \quad p_i''(x_i) = p_{i+1}''(x_i).$$

For the first and the last polynomial there are no boundary conditions. The selection

$$S''(a) = S''(b) = 0$$

results in the natural splines.

Computation of the splines:

- a linear system of equations (moment equations) to calculate the cubic splines,
- quite technical, see the following theorem.

Theorem 6.8

If the twice continuously differentiable cubic spline S satisfies the interpolation condition

$$S(x_j) = f_j, \quad j = 0, \ldots, n,$$

then on $[x_j, x_{j+1}]$ its form reads

$$\begin{aligned} S(x) = {} & f_j \frac{x_{j+1} - x}{x_{j+1} - x_j} + f_{j+1} \frac{x - x_j}{x_{j+1} - x_j} \\ & - \frac{1}{6} S''(x_j) \frac{(x_{j+1} - x)(x - x_j)}{x_{j+1} - x_j} ((x_{j+1} - x) + (x_{j+1} - x_j)) \\ & - \frac{1}{6} S''(x_{j+1}) \frac{(x_{j+1} - x)(x - x_j)}{x_{j+1} - x_j} ((x - x_j) + (x_{j+1} - x_j)). \end{aligned}$$

Theorem 6.9

Under the conditions of Thm. 6.8 it holds for the moments $S''(x_j)$ for $j = 1, \ldots, n-1$ that

$$\begin{aligned}
&\frac{x_{j+1} - x_j}{x_{j+1} - x_{j-1}} S''(x_{j+1}) + 2S''(x_j) + \frac{x_j - x_{j-1}}{x_{j+1} - x_{j-1}} S''(x_{j-1}) \\
&= \frac{6}{x_{j+1} - x_{j-1}} \left(\frac{f_{j+1} - f_j}{x_{j+1} - x_j} - \frac{f_j - f_{j-1}}{x_j - x_{j-1}} \right) \\
&= 6f[x_{j-1}, x_j, x_{j+1}].
\end{aligned}$$

Theorem 6.10

Together with the two conditions, e.g. $S''(x_0) = S''(x_n) = 0$ for the natural splines, the system of linear equations has a unique solutions. The matrix of the system has a condition not larger then 3.

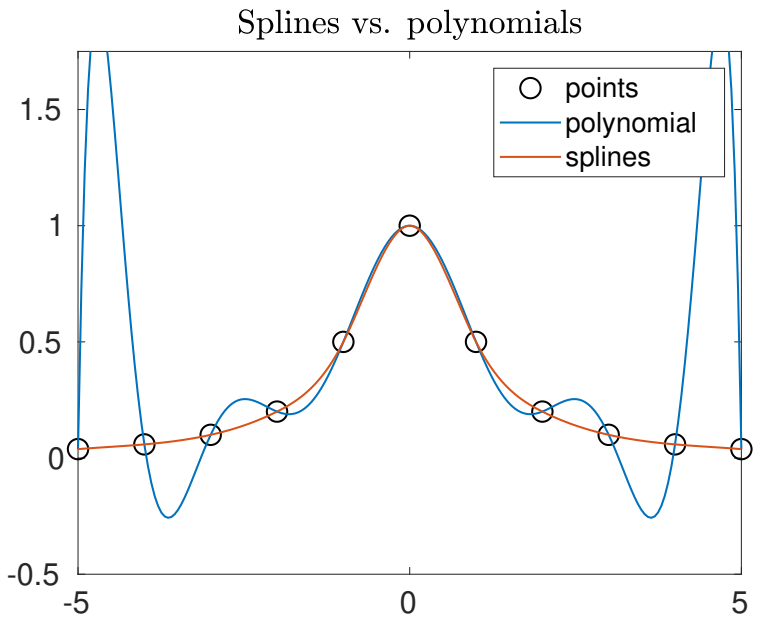
Example:

- consider once again the example of Runge

$$f(x) = \frac{1}{1+x^2},$$

- cubic splines for equidistant nodes and $[-5, 5]$ using MatLab,
- much better results than for polynomials, maximal error

$$\max_{x\in[-5,5]} \left|S(x) - f(x)\right| \approx 0.022.$$



Splines in MatLab:

```
f = @(x) 1./(1+x.^2);
X = -5:5;
Y = f(X);
xx = linspace(-5,5,401);
yy = spline(X,Y,xx);
plot(X,Y,'o',xx,yy)
max(abs(yy-f(xx)))
```

Outline

Definition 6.11 (Finite differences)

The approximations

$$D^+ f(x) = \frac{f(x+h) - f(x)}{h},$$
$$D^- f(x) = \frac{f(x) - f(x-h)}{h},$$
$$D^0 f(x) = \frac{f(x+h) - f(x-h)}{2h}$$

for $f'(x)$ are called forward difference, backward difference and central difference.

Theorem 6.12 (Approximation orders)

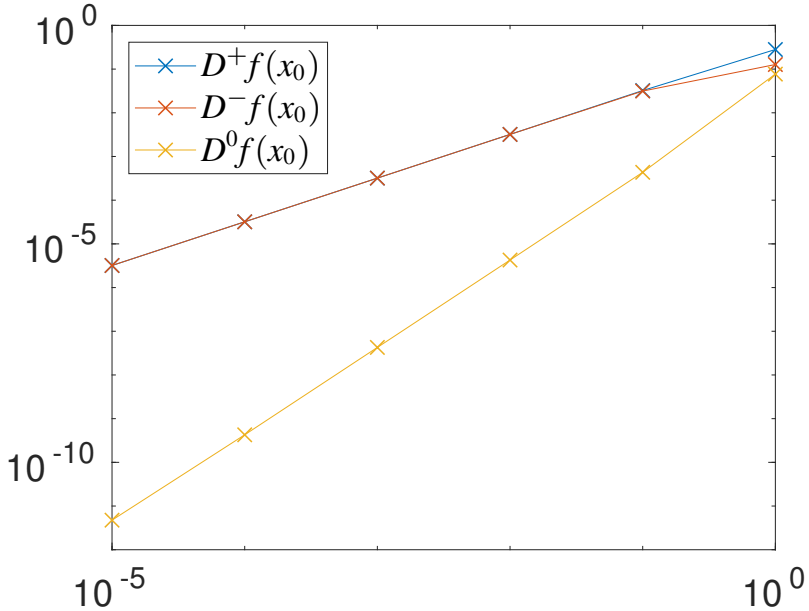
For the finite differences holds

$$f'(x) - D^+ f(x) = \mathcal{O}(h), \qquad \text{(order 1)}$$
$$f'(x) - D^- f(x) = \mathcal{O}(h), \qquad \text{(order 1)}$$
$$f'(x) - D^0 f(x) = \mathcal{O}(h^2), \qquad \text{(order 2).}$$

Example:

- approximation of $f'(x_0)$ for $f(x) = \arctan(x)$ and $x_0 = 0.5$
- approximations errors for different h

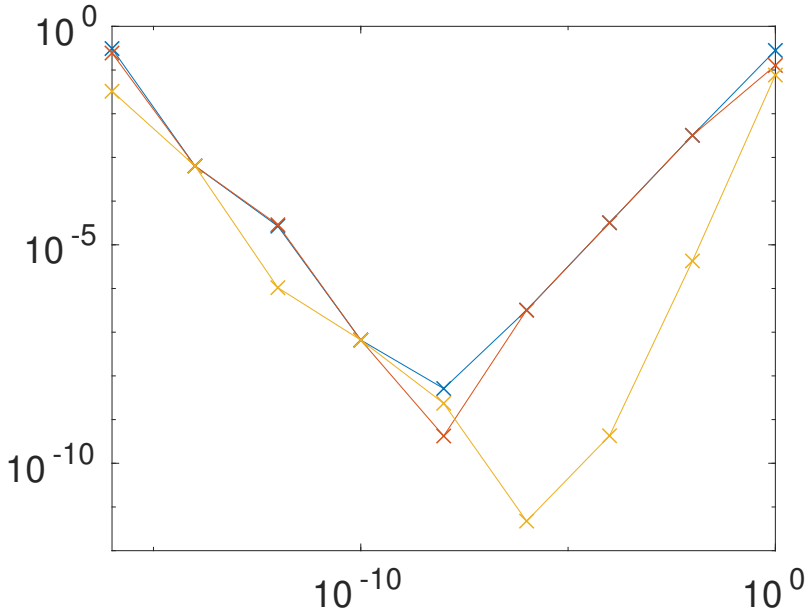
h	$\lvert D^+ f(x_0) - f'(x_0)\rvert$	$\lvert D^- f(x_0) - f'(x_0)\rvert$	$\lvert D^0 f(x_0) - f'(x_0)\rvert$
0.1	$3.2 \cdot 10^{-2}$	$3.1 \cdot 10^{-2}$	$4.3 \cdot 10^{-4}$
0.01	$3.2 \cdot 10^{-3}$	$3.2 \cdot 10^{-3}$	$4.3 \cdot 10^{-6}$
10^{-5}	$3.2 \cdot 10^{-6}$	$3.2 \cdot 10^{-6}$	$4.8 \cdot 10^{-12}$.



Example:

- approximation of $f'(x_0)$ for $f(x) = \arctan(x)$ and $x_0 = 0.5$
- but we cannot reduce h arbitrarily, as

h	$\lvert D^+ f(x_0) - f'(x_0)\rvert$	$\lvert D^- f(x_0) - f'(x_0)\rvert$	$\lvert D^0 f(x_0) - f'(x_0)\rvert$
10^{-12}	$2.7 \cdot 10^{-5}$	$2.9 \cdot 10^{-5}$	$1.0 \cdot 10^{-6}$
10^{-14}	$6.4 \cdot 10^{-4}$	$6.4 \cdot 10^{-4}$	$6.4 \cdot 10^{-4}$.



Definition 6.13 (Finite differences)

The approximation

$$D^2 f(x) = \frac{f(x-h) - 2f(x) + f(x+h)}{h^2}$$

for $f''(x)$ is called central difference of second order.

The error of the second order central difference is in $\mathcal{O}(h^2)$, so $D^2 f$ is an approximation second order, thus

$$f''(x) = D^2 f(x) + \mathcal{O}(h^2).$$

Partial derivatives of higher order in 2D:

- mixed partial derivatives for $h_x = h_y = h$

$$u_{xy} \approx \frac{u(x+h,y+h) - u(x+h,y-h) - u(x-h,y+h) + u(x-h,y-h)}{4h^2},$$

- central difference of second order for $h_x = h$

$$u_{xx} \approx \frac{u(x+h,y) - 2u(x,y) + u(x-h,y)}{h^2}$$

- for the Laplace operator or Laplacian for $h_x = h_y = h$ we get

$$\begin{aligned}\Delta u &= u_{xx} + u_{yy} \\ &\approx \frac{u(x+h,y) + u(x-h,y) + u(x,y+h) + u(x,y-h) - 4u(x,y)}{h^2}.\end{aligned}$$

Outline

Goal:

$$f : [a, b] \to \mathbb{R}, \qquad I(f) = \int_a^b f(x)\,\mathrm{d}x = ?$$

Approach:

1. segmentation
$$a = x_0 < x_1 < \ldots < x_{n-1} < x_n = b,$$
2. approximation of $f(x)$ on $[x_{k-1}, x_k]$ by a function (polynomial $p_k(x)$) which is easy to interate and use
$$\begin{aligned}\int_a^b f(x)\,\mathrm{d}x &= \sum_{k=1}^{n} \int_{x_{k-1}}^{x_k} f(x)\,\mathrm{d}x \\ &\approx \sum_{k=1}^{n} \int_{x_{k-1}}^{x_k} p_k(x)\,\mathrm{d}x.\end{aligned}$$

Definition 7.1

In general an approximation formula of the form

$$Q_m(f) = \sum_{k=0}^{m} a_k f(x_k) \approx \int_a^b f(x)\,\mathrm{d}x$$

is called an quadrature formula.

Remark 3

A quadrature formula is a weighted sum of the function values, as

$$\text{for 1 segment:} \quad Q_m(f) = \int_a^b p_m(x)\,\mathrm{d}x = \sum_{k=0}^{m} f(x_k) \underbrace{\int_a^b l_k(x)\,\mathrm{d}x}_{q_k \in \mathbb{R},\ \text{weights}}.$$

Definition 7.2

The summation over all segements results in the composite rectangle rule

$$R_n = h\big(f_0 + f_1 + \ldots + f_{n-1}\big),$$

with $f_i = f(t_i)$, *the composite midpoint rule*

$$M_n = h\big(f(a + 0.5h) + f(t_1 + 0.5h) + \ldots + f(b - 0.5h)\big),$$

and the composite trapezion rule

$$T_n = h\big(0.5f_0 + f_1 + \ldots + f_{n-1} + 0.5f_n\big).$$

Outline

Standard integration problem

Standardization:

$$\text{general domain } x \in [a,b] \quad \rightarrow \quad t \in [-1,1].$$

Definition 7.3

In the following we consider the standard integration problem

$$\int_{-1}^{1} f(t)\,\mathrm{d}t.$$

The transformation to a general interval is

$$[-1,1] \rightarrow [a,b], \quad \varphi(t) = \frac{b+a}{2} + t\frac{b-a}{2}.$$

In

$$Q_n(f) = \sum_{k=0}^{n} a_k f(t_k)$$

we call $t_k \in [-1,1]$ *the nodes and* a_k *the weights.*

Outline

Idea:

- generalization of the compsite interation rules,
- divide the interval $[-1,1]$ into m equal parts by

$$t_k = -1 + kh,\ k = 0,\dots,m, \quad \text{with} \quad h = \frac{2}{m},$$

- take all $m+1$ nodes and replace the function $f(t)$ by its interpolation polynomial $p_m(t)$ of degree $\leq m$,
- for simplification we focus not on subintervals but on the whole $[-1,1]$ and derive formulas

$$p_m(t) = \sum_{i=0}^{m} f(t_i) l_i(t), \qquad l_i(t) = \prod_{j=0, j\neq i}^{m} \frac{t - t_j}{t_i - t_j}.$$

Definition 7.4

The degree of exactness of an quadrature formula $Q_m(f)$ is the maximal integer $q \geq 0$, for which holds

$$I(p(t)) = Q_m(p(t))$$

for all polynomials $p(t)$ of degree q.

Theorem 7.5

Let $t_0, \ldots, t_m \in [-1, 1]$ be $m+1$ pairwise different nodes. The quadrature formula

$$I(f) = \int_{-1}^{1} g(t)\,\mathrm{d}t \to Q_m(f) = \sum_{i=0}^{m} a_i g(t_i)$$

is exact for all polynomials of degree $\leq m \Leftrightarrow$ the weights are given by

$$a_i = \int_{-1}^{1} \prod_{j=0, j\neq i}^{m} \frac{t - t_j}{t_i - t_j}\,\mathrm{d}t.$$

(Integral of the Lagrange basis functions.)

Computation of the weights

Conditions:

- quadrature formula

$$\int_{-1}^{1} g(t)\,\mathrm{d}t \to \sum_{i=0}^{m} a_i g(t_i),$$

- exactness for all polynomials of degree $\leq m \Leftrightarrow$ it is exact for $t^0,\ t^1,\ t^2, \ldots, t^m$,
- thus we demand

$$\int_{-1}^{1} t^j\,\mathrm{d}t = \sum_{i=0}^{m} a_i (t_i)^j, \quad j = 0, \ldots, m,$$

- the weights a_i satisfy the linear system of equations

$$\sum_{i=0}^{m} t_i^j a_i = \begin{cases} \frac{2}{j+1} & \text{for even } j \\ 0 & \text{for odd } j \end{cases}$$

for all $j = 0, \ldots, m$.

Computation of the weights

Example:

- set up a formula of second order using the nodes $t_0 = -1$, $t_1 = 0$ and $t_2 = 1$,
- formula has the form

$$\int_{-1}^{1} g(x)\,\mathrm{d}x \to a_0 g(t_0) + a_1 g(t_1) + a_2 g(t_2)$$

- system of linear equations

$$\begin{aligned}
g(t) = t^0 = 1: \quad & a_0 & + a_1 & & + a_2 & & = 2 \\
g(t) = t^1: \quad & -a_0 & & & + a_2 & & = 0 \\
g(t) = t^2: \quad & a_0 & & & + a_2 & & = \frac{2}{3}
\end{aligned}$$

- solution is $a_0 = \frac{1}{3}$, $a_1 = \frac{4}{3}$, $a_2 = \frac{1}{3}$, (Simpson's rule)

$$\int_{-1}^{1} g(t)\,\mathrm{d}t \approx \frac{1}{3} g(-1) + \frac{4}{3} g(0) + \frac{1}{3} g(1).$$

Simpson's rule for an arbitrary interval:

$$\int_a^b f(x)\,\mathrm{d}x \approx \frac{b-a}{6}\left(f(a)+4f(\frac{a+b}{2})+f(b)\right).$$

Remark 4

In general the degree of exactness of Newton-Cotes formulas using $m+1$ nodes is

$$\begin{aligned} &m \text{ for odd } m,\\ &m+1 \text{ for even } m. \end{aligned}$$

Kepler

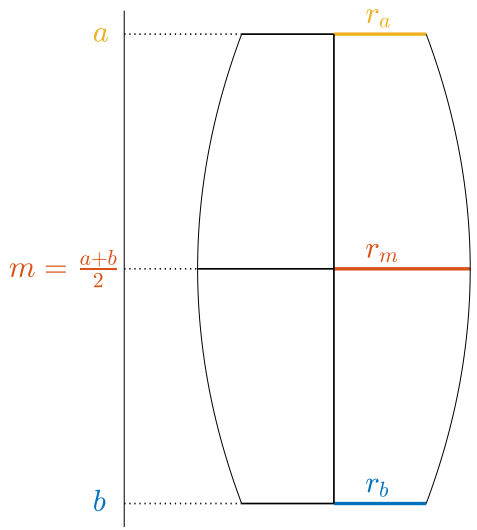
Application from 1615:

- already Kepler used Simpson's method without knowing it,
- computation of the volume of a barrel, thus approximation of

$$V = \pi \int_a^b (f(x))^2 \, \mathrm{d}x,$$

- Kepler used

$$V \approx \pi \frac{h}{6} \left(r_a^2 + 4 r_m^2 + r_b^2 \right).$$



Weights:

- for the standard integration problem on $[-1, 1]$,
- using one single subinterval and the nodes

$$t_k = -1 + k\frac{2}{m}, \quad k = 0, \ldots, m.$$

m	weights (a_k)	degree of exactness	error	name
0	2	0	$\frac{h^2}{2} f'(\xi)$	rectangle rule
1	$1,\ 1$	1	$\frac{h^3}{12} f''(\xi)$	trapezion rule
2	$\frac{1}{3},\ \frac{4}{3},\ \frac{1}{3}$	3	$\frac{h^5}{90} f^{(4)}(\xi)$	Simpson's rule
3	$\frac{1}{4},\ \frac{3}{4},\ \frac{3}{4},\ \frac{1}{4}$	3	$\frac{3h^5}{80} f^{(4)}(\xi)$	3/8 rule
4	$\frac{7}{45},\ \frac{32}{45},\ \frac{12}{45},\ \frac{12}{45},\ \frac{32}{45},\ \frac{7}{45}$	5	$\frac{8h^7}{945} f^{(4)}(\xi)$	Milne's rule
$\vdots$	$\vdots$			

Remarks:

1. Don't use Newton-Cotes formulas of higher order, because some weights will become negative, which will cause cancellation of significant digits.
2. In general, the sequence

$$Q_m(f), \quad m = 1, 2, 3, \ldots$$

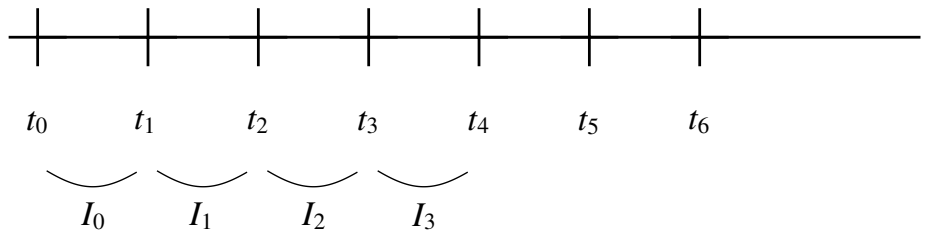
does not converge to $\int_{-1}^{1} f(t)\,\mathrm{d}t$.

Outline

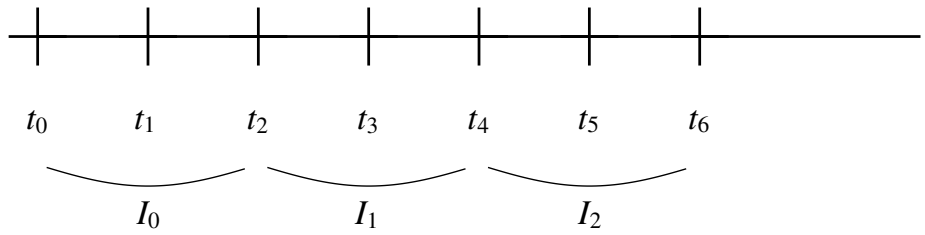
Composite rules for the standard interval

Composite formula:

- use a discretization of the complete interval,
- apply the previous quadrature formulas for the single segements
- $m = 0$: constant polynomial, rectangle rule,
- $m = 1$: polynomial of degree 1, trapezion rule,



- $m = 2$: polynomial of degree 2, Simpson's rule,



Composite rules for the standard interval

Examples for the standard interval $[-1,1]$:

1. composite rectangle rule

$$h\big(f(-1)+f(-1+h)+f(-1+2h)+\ldots+f(1-h)\big)$$

2. composite trapezion rule

$$\frac{h}{2}\big(f(-1)+2f(-1+h)+2f(-1+2h)+\ldots+2f(1-h)+f(1)\big)$$

3. composite Simpson's rule for an even n

$$\frac{h}{6}\big(f(-1)+4f(-1+h)+2f(-1+2h)+4f(-1+3h)+\ldots+4f(1-h)+f(1)\big).$$

Composite rules for a arbitrary interval

Procedure for an arbitrary interval:

- select a number n of nodes $x_i = a + ih$ with stepsize $h = \frac{b-a}{n}$,
- compute the function values $f_i = f(x_i)$
- select an integration rule

The composite rules are:

$$R_n = h\left(f_0 + f_1 + f_2 + \ldots + f_{n-1}\right),$$

$$T_n = \frac{h}{2}\left(f_0 + 2f_1 + 2f_2 + \ldots + 2f_{n-1} + f_n\right),$$

$$S_n = \frac{h}{3}\left(f_0 + 4f_1 + 2f_2 + 4f_3 + 2f_4 + \ldots + 2f_{n-2} + 4f_{n-1} + f_n\right).$$

Example:

- consider the approximation of

$$I = \frac{1}{\ln(2)} \int_0^{\pi/2} \frac{x}{1+\sin(x)} \, \mathrm{d}x = 1.$$

- using different composite formulas for different discretizations

$$x_i = \frac{\pi}{2n} i, \quad i = 0, \ldots, n.$$

n	R_n	M_n	T_n	S_n
5	0.8162	1.0029	0.9942	
10	0.9095	1.0007	0.9985	0.999975
50	0.9821	1.0000	0.9999	1.0
100	0.9911	1.0000	1.0000	1.0
200	0.9955	1.0000	1.0000	1.0

Composite rules for a arbitrary interval

Error terms for an arbitrary interval:

$$\left| \int_a^b f(x)\,\mathrm{d}x - R_n \right| \leq \frac{(b-a)h}{2} \max_{x\in[a,b]} |f'(x)|,$$

$$\left| \int_a^b f(x)\,\mathrm{d}x - T_n \right| \leq \frac{h^2}{12}(b-a) \max_{x\in[a,b]} |f''(x)|,$$

$$\left| \int_a^b f(x)\,\mathrm{d}x - S_n \right| \leq \frac{h^4}{180}(b-a) \max_{x\in[a,b]} |f^{(4)}(x)|.$$

Remark 5

1. *The rectangle rule is exact only for constant polynomials, the trapezion rule is exact for polynomials of degree 1 and Simpson's rule is exact for polynomials of degree 3.*
2. *The integration errors can be used to determine a proper number of nodes n so that a given error level is not exceeded.*

Outline

Romberg's method

Idea:

- multiple calculation with a quadrature formula with halved step size each time,
- use of previously calculated values for composite trapezoidal rule,

$$I(h) = T_{2n} = \frac{1}{2}T_n + \frac{h}{2}\sum_{i=1}^{n} f(a + \frac{2i-1}{2}h)$$

- extrapolation to improve the approximation

$$\lim_{h\to 0} I(h) = \int_a^b f(x)\,\mathrm{d}x.$$

Romberg's method

Procedure:

- basic step size $\bar{h} = (b-a)/m$,
- initialization for $i = 0, 1, 2, \ldots$,

$$I_{i,0} = T_{2^i m} \qquad \text{(comp. trapezion rule for } h = \bar{h}/2^i),$$

- extrapolation to $h = 0$ using $I_{i,0}$ yields Romberg's scheme

$$I_{i,j} = \frac{4^j I_{i,j-1} - I_{i-1,j-1}}{4^j - 1}, \qquad j = 1, \ldots, i,$$

- scheme for basic step size $\bar{h}$

$$\begin{array}{lccccccc}
\bar{h}: & I_{0,0} & & & & & & \\
 & & \searrow & & & & & \\
\frac{\bar{h}}{2}: & I_{1,0} & \to & I_{1,1} & & & & \\
 & & \searrow & & \searrow & & & \\
\frac{\bar{h}}{4}: & I_{2,0} & \to & I_{2,1} & \to & I_{2,2} & & \\
 & & \searrow & & \searrow & & \searrow & \\
\frac{\bar{h}}{8}: & I_{3,0} & \to & I_{3,1} & \to & I_{3,2} & \to & I_{3,3}.
\end{array}$$

Romberg's method

Example:

- approximation of

$$\int_1^{10} \ln(x)\,\mathrm{d}x,$$

- using trapezion rule and Romberg' scheme for extrapolation,
- run the iteration until

$$|T_{n,n-1} - T_{n,n}| \leq \varepsilon,$$

- the scheme (using 5 digits) reads

10.362							
12.852	13.682						
13.685	13.963	13.982					
13.934	14.017	14.021	14.022				
14.002	14.025	14.026	14.026	14.026			
14.020	14.026	14.026	14.026	14.026	14.026		
14.024	14.026	14.026	14.026	14.026	14.026	14.026	
14.025	14.026	14.026	14.026	14.026	14.026	14.026	14.026

- error for $\varepsilon = 10^{-5}$ is $6.3753 \cdot 10^{-6}$,
- error for $\varepsilon = 10^{-10}$ is $-3.8345 \cdot 10^{-10}$.

Outline

Example:

- consider a quadrature rule for $[-1,1]$ with two nodes $Q(f) = a_0 f(t_0) + a_1 f(t_1)$,
- try to reach maximal degree of exactness (3 since 4 degrees of freedom), thus

$$Q(t^i) = I(t^i), \quad i = 0, \dots, 3,$$

- equations

$$Q(1) = a_0 + a_1 = 2, \qquad Q(t) = a_0 t_0 + a_1 t_1 = 0,$$
$$Q(t^2) = a_0 t_0^2 + a_1 t_1^2 = \frac{2}{3}, \qquad Q(t^3) = a_0 t_0^3 + a_1 t_1^3 = 0,$$

- solution of the nonlinear system of equations is

$$t_{0,1} = \pm\sqrt{1/3}, \qquad a_{0,1} = 1,$$

- quadrature rule (Gaussian quadrature for two nodes) reads

$$Q_1(f) = f(-\sqrt{1/3}) + f(\sqrt{1/3}).$$

Gaussian quadrature

General approach:

$$Q_m(f) = \sum_{i=0}^{m} a_i g(t_i) \tag{8}$$

- $2m+2$ degrees of freedom, thus exactness for x^i for $i = 0, \ldots, 2m+1$.

Definition 7.6

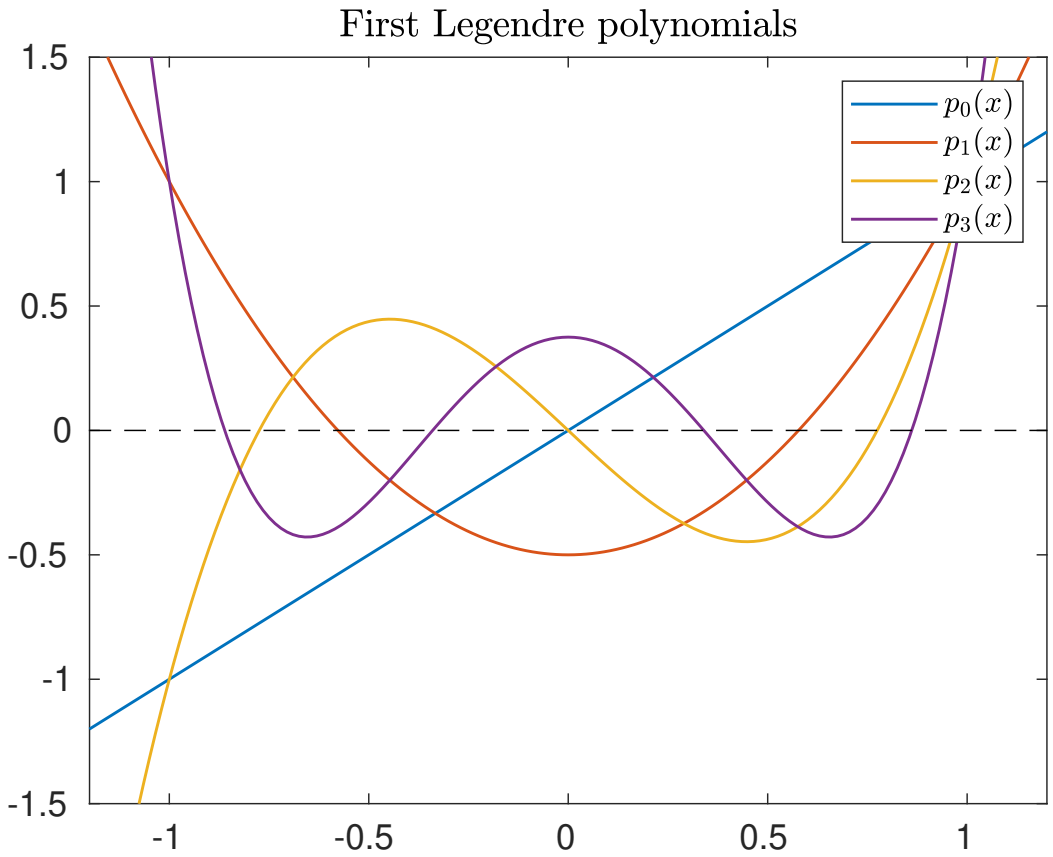
With the initialization $p_0(x) = 1$ and $p_1(x) = x$ the Legendre polynomials are defined as

$$(n+1)p_{n+1}(x) = (2n+1)xp_n(x) - np_{n-1}(x).$$

Eamples:

$$p_2(x) = \frac{1}{2}(3x^2 - 1),$$
$$p_3(x) = \frac{1}{3}(\frac{5}{2}x(3x^2 - 1) - 2x) = \frac{1}{2}(5x^2 - 3)x,$$
$$p_4(x) = \frac{1}{8}(35x^4 - 30x^2 + 3).$$

Legendre polynomials for $[-1, 1]$



Theorem 7.7

1. *The nodes of the quadrature formula (8) are the zeros of the Legendre polynomial of degree* $m+1$.
2. *All zeros are simple and in* $[-1,1]$.
3. *The degree of exactness of the Gaussian quadrature rules are* $2m+1$.

Gaussian rule for $m=2$ for $[-1,1]$

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

Nodes and weights for $m=3$

$$t_{0,3} = \pm 0.861136\ldots, \qquad t_{1,2} = \pm 0.339981\ldots,$$
$$a_{0,3} = \pm 0.347854\ldots, \qquad a_{1,2} = \pm 0.652145\ldots.$$

Arbitrary integration domain:

- for the complete domain with $h = b - a$,
- degree of exactness 3

$$Q_1(f) = \frac{h}{2}\left(f(\frac{a+b}{2} - \sqrt{\frac{1}{3}}\frac{b-a}{2}) + f(\frac{a+b}{2} + \sqrt{\frac{1}{3}}\frac{b-a}{2})\right),$$

- degree of exactness 5

$$Q_2(f) = \frac{h}{2}\left(\frac{5}{9}f(\frac{a+b}{2} - \sqrt{\frac{3}{5}}\frac{b-a}{2}) + \frac{8}{9}f(\frac{a+b}{2}) + \frac{5}{9}f(\frac{a+b}{2} + \sqrt{\frac{3}{5}}\frac{b-a}{2})\right).$$

Improper integral

Example:

- consider (with the true value $I = 1$)

$$I = \frac{1}{\sqrt{\pi}} \int_{-\infty}^{\infty} \exp(-x^2)\, \mathrm{d}x,$$

- use the coordinate transformation

$$x = \tan(t), \quad \frac{\mathrm{d}x}{\mathrm{d}t} = \frac{1}{\cos(t)^2} \quad \mathrm{d}x = \frac{1}{\cos(t)^2}\, \mathrm{d}t,$$

- approximation of

$$\frac{1}{\sqrt{\pi}} \int_{-\frac{\pi}{2}}^{\frac{\pi}{2}} \exp(-\tan(t)^2) \frac{1}{\cos(t)^2}\, \mathrm{d}t,$$

- using composite Gaussian quadrature formulas with $n \in \{5, 10\}$ subintervals

n	$Q_2(f)$	$Q_3(f)$
5	0.99942	1.00221
10	1.00088	0.99995

Outline

Two-dimensional integrals

Problem:

$$u : S \to \mathbb{R},\ S \subset \mathbb{R}^2, \qquad \iint_S u(x,y)\,\mathrm{d}\sigma = ?$$

Applications:

- finite elements method to solve elliptic pde's numerically.

Different situations:

- quadrangles or triangles with (4 resp. 2) edges being parallel to the axes
  $\to$ simple
- arbitrary triangle/quadrangle
  $\to$ more complex, use a coordinate transformation and cubature formulas,
- general domain S (curved boundary)
  $\to$ most complex.

Definition 7.8

The triangle with the vertices $(0,0)$, $(1,0)$ *and* $(0,1)$ *is called reference triangle* T_0.

Approximation of

$$I = \iint_{T_0} f(x,y)\,\mathrm{d}y\,\mathrm{d}x = \int_0^1 \int_0^{1-x} f(x,y)\,\mathrm{d}y\,\mathrm{d}x.$$

Example

1. *The following cubature formula is exact for all polynomials up to degree 1*

$$\frac{1}{6}\big(f(0,0)+f(1,0)+f(0,1)\big) \approx I.$$

2. *The following cubature formula is exact for all polynomials up to degree 2*

$$\frac{1}{6}\big(f(0.5,0)+f(0,0.5)+f(0.5,0.5)\big) \approx I.$$

Outline

Definition 8.1

An ordinary differential equation of first order (ode) has the form

$$y'(x) = f(x, y(x)), \qquad a \le x \le b,$$

with $f : [a,b] \times \mathbb{R}^n \to \mathbb{R}^n$ *(this includes also systems with* $n > 1$*). Typically* f *is called the ride-hand side of the ode.*

Definition 8.2

A initial value problem or Cauchy problem consists of an ordinary differential equation of first order and an initial value

$$\begin{aligned} y'(x) &= f(x, y(x)), \qquad a \le x \le b, \\ y(a) &= y_0 \end{aligned}$$

with $f : [a,b] \times \mathbb{R}^n \to \mathbb{R}^n$.

Wanted: $y : [a,b] \to \mathbb{R}^n$.

Outline

General procedure:

- discretisation of the interval $[a, b]$

$$a = x_0 < x_1 < x_2 < \ldots < x_N = b,$$

- step sizes $h_i = x_{i+1} - x_i$,
- approximations $Y_i \approx y(x_i)$,
- initialization and iteration

$$\begin{aligned} Y_0 &= y(a) = y_0, \\ Y_{i+1} &= Y_i + h_i \Phi(x_i, Y_i, h_i), \qquad i = 0, \ldots, N-1. \end{aligned}$$

Euler's method (only to explain the basic idea):

$$Y_{i+1} = Y_i + hf(x_i, Y_i), \qquad i = 0, 1, 2, \ldots.$$

Crank-Nicolson (Trapezion rule):

$$Y_{i+1} = Y_i + \frac{h_i}{2}\big(f(x_i, Y_i) + f(x_{i+1}, Y_{i+1}))\big).$$

Midpoint-rule (Gaussian integration):

$$Y_{i+1} = Y_i + h_i f\big(x_i + h_i/2, (Y_i + Y_{i+1})/2)\big).$$

Euler-Heun method:

$$Y_{i+1} = Y_i + \frac{h_i}{2}\big(f(x_i, Y_i) + f(x_i + h, Y_i + hf(x_i, Y_i))\big).$$

Outline

Runge-Kutta methods

Idea:

- simple iterations

$$Y_{i+1} = Y_i + h_i \Phi(x_i, Y_i, h_i)$$

  without using further iterations directly,
- using s stages between x_i and x_{i+1} for high accuracy,
- evaluation of K_j using results of previous stages $K_1, \ldots, K_{j-1}$,
- linear combination of evaluations to define the next iterate,
- simple analysis of the error and the order.

Definition 8.3 (Kutta 1901)

Let s be an integer (the number of stages) and $a_{ij}, b_i, c_l \in \mathbb{R}$. Then the method

$$\begin{aligned}
K_1 &= f(x_\ell, y_\ell),\\
K_2 &= f(x_\ell + c_2 h, y_\ell + h a_{21} K_1),\\
K_3 &= f(x_\ell + c_3 h, y_\ell + h(a_{31} K_1 + a_{32} K_2)),\\
&\vdots\\
K_s &= f(x_\ell + c_s h, y_\ell + h(a_{s1} K_1 + \ldots + a_{s,s-1} K_{s-1}))
\end{aligned}$$

with

$$Y_{\ell+1} = Y_\ell + h(b_1 K_1 + \ldots + b_s K_s)$$

is called an s-stage explicit Runge-Kutta method. Further, $\Phi(x_\ell, Y_\ell, h)$ with

$$Y_{\ell+1} = Y_\ell + h\Phi(x_\ell, Y_\ell, h)$$

is called process function.

Butcher tableau of a Runge-Kutta method:

$$\begin{array}{c|ccccc} 0 & & & & & \\ c_2 & a_{21} & & & & \\ c_3 & a_{31} & a_{32} & & & \\ \vdots & \vdots & & \ddots & & \\ c_s & a_{s1} & & & a_{s,s-1} & \\ \hline & b_1 & b_2 & \cdots & b_{s-1} & b_s \end{array}$$

Butcher-tableau for Euler's method

$$\begin{array}{c|c} 0 & \\ \hline & 1 \end{array}$$

Butcher-tableau for Euler-Heun

$$\begin{array}{c|cc} 0 & & \\ 1 & 1 & \\ \hline & 1/2 & 1/2 \end{array}$$

Definition 8.4

The Butcher-tableau

$$\begin{array}{c|cccc} 0 & & & & \\ \frac{1}{2} & \frac{1}{2} & & & \\ \frac{1}{2} & 0 & \frac{1}{2} & & \\ 1 & 0 & 0 & 1 & \\ \hline & 1/6 & 1/3 & 1/3 & 1/6 \end{array}$$

with

$$\begin{aligned} K_1 &= f(x_\ell, Y_\ell), \\ K_2 &= f(x_\ell + \frac{1}{2}h, Y_\ell + \frac{h}{2}K_1), \\ K_3 &= f(x_\ell + \frac{1}{2}h, Y_\ell + \frac{h}{2}K_2), \\ K_4 &= f(x_\ell + h, Y_\ell + hK_3), \\ Y_{\ell+1} &= Y_\ell + \frac{h}{6}(K_1 + 2K_2 + 2K_3 + K_4) \end{aligned}$$

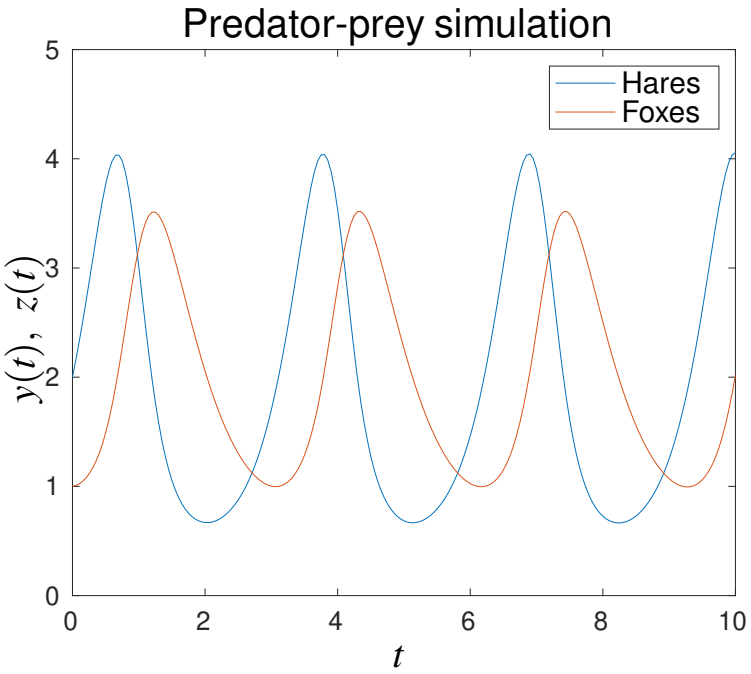
is called "the" Runge-Kutta method or RK 4.

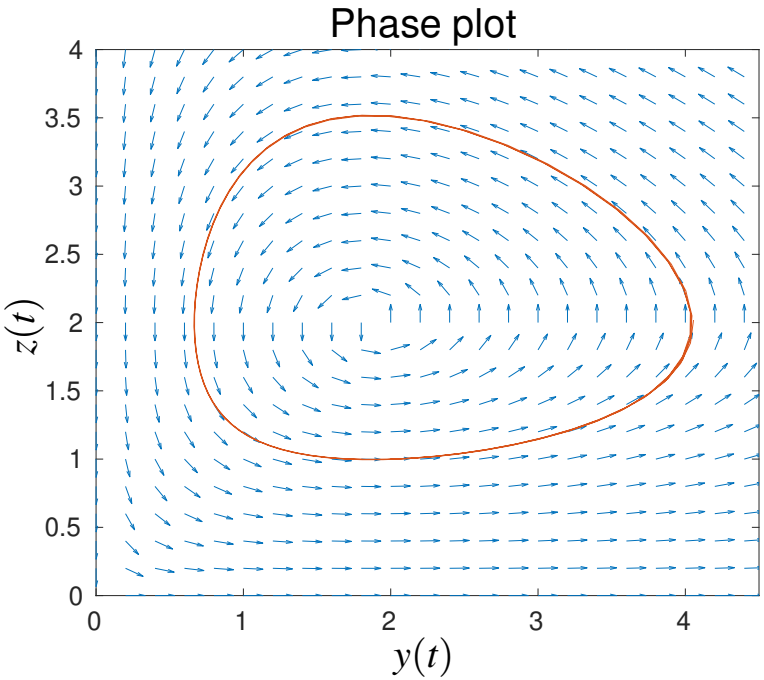
Predator-prey simulation

Consider the Predator-prey equation

$$\dot{y}(t) = ay(t) - by(t)z(t),$$
$$\dot{z}(t) = cy(t)z(t) - dz(t).$$

Simulation for $t \in [0, 10]$ as well as the initial values $y(0) = 2$, $z(0) = 1$ and the parameters $a = 3$, $b = 1.5$, $c = 0.8$ and $d = 1.5$.





Outline

Local truncation error

Evaluation of the local error in a single step.

Definition 8.5

Let $\Phi(x, y, h)$ be a process function. We call the term

$$\tau(x, h) = \frac{y(x+h) - y(x)}{h} - \Phi(x, y, h)$$

local truncation error of the Runge-Kutta method. If

$$\|\tau(x, h)\| = \mathcal{O}(h^p) \quad \text{for } h \to 0,$$

then we call p the order of the Runge-Kutta method.

Local truncation error

Local truncation error for Euler's method ($\Phi(x,y,h) = f(x,y)$):

$$\begin{aligned}\tau(x,h) &= \frac{y(x+h)-y(x)}{h} - \Phi(x,y,h) \\ &= \frac{y(x)+hy'(x)+\mathcal{O}(h^2)-y(x)}{h} - f(x,y) \\ &= y'(x) + \mathcal{O}(h) - \underbrace{f(x,y)}_{y'(x)} = \mathcal{O}(h).\end{aligned}$$

$\rightarrow$ order $p = 1$.

Local truncation error

Taylor-series 1:

- the term

$$\frac{y(x+h)-y(x)}{h} = y(x) + hy'(x) + \frac{h^2}{2}y''(x) + \dots$$

  is always the necessary,
- therefore we use

$$\begin{aligned} y'(x) &= f(x,y), \\ y''(x) &= (y'(x))' = \frac{\mathrm{d}f(x,y(x))}{\mathrm{d}x} = f_x + f_y y' = f_x + f_y f, \\ y'''(x) &= f_{xx} + 2f_{xy}f + f_{yy}f^2 + f_y f_x + f_y^2 f, \end{aligned}$$

- so the transformed term using a Taylor-series reads

$$\frac{y(x+h)-y(x)}{h} = f + \frac{h}{2}(f_x + f_y f) + \frac{h^2}{6}(f_{xx} + 2f_{xy}f + f_{yy}f^2 + f_y f_x + f_y^2 f) + \dots,$$

- the other Taylor-series depends on the method.

Local truncation error

For Euler-Heun:

- the iteration is

$$Y_{i+1} = Y_i + \frac{h_i}{2}\big(f(x_i, Y_i) + f(x_i + h, Y_i + hf(x_i, Y_i))\big),$$

- Taylor-series no. 1 from the last slide

$$\frac{y(x+h) - y(x)}{h} = f + \frac{h}{2}(f_x + f_y f) + \dots,$$

- Taylor-series no. 2

$$\Phi(x, y, h) = \frac{1}{2}\big(f(x, y) + f(x + h, y + hf(x, y)) + \dots\big) = \frac{1}{2}\big(f + f + hf_x + f_y hf + \dots\big),$$

- local truncation error

$$\begin{aligned}\tau(x, h) &= \frac{y(x+h) - y(x)}{h} - \Phi(x, y, h) \\ &= \big(f + \frac{h}{2}(f_x + f_y y')\big) - \frac{1}{2}\big(f + f + hf_x + f_y hf\big) + \mathcal{O}(h^2) = \mathcal{O}(h^2),\end{aligned}$$

- thus the method is of order 2 (at least).

Order of Runge-Kutta methods

Some orders of Runge-Kutta methods:

- Euler's method has order 1,
- Euler-Heun has order 2,
- Crank-Nicolson has order 2 (implicit),
- midpoint rule has order 2 (implicit),
- the Runge-Kutta method from definition 8.4 has order 4.

Remark 6

The order of an explicit Runge-Kutta method is bounded by the number of stages as follows

s	1	2	3	4	5	6	7	8	9	$s \geq 9$
q	1	2	3	4	4	5	6	6	7	$q \leq s-2$

The number of equations that we have to respect depend on the order as follows

s	1	2	3	4	5	6	7	8
q	1	2	4	8	17	37	85	200

Order of Runge-Kutta methods

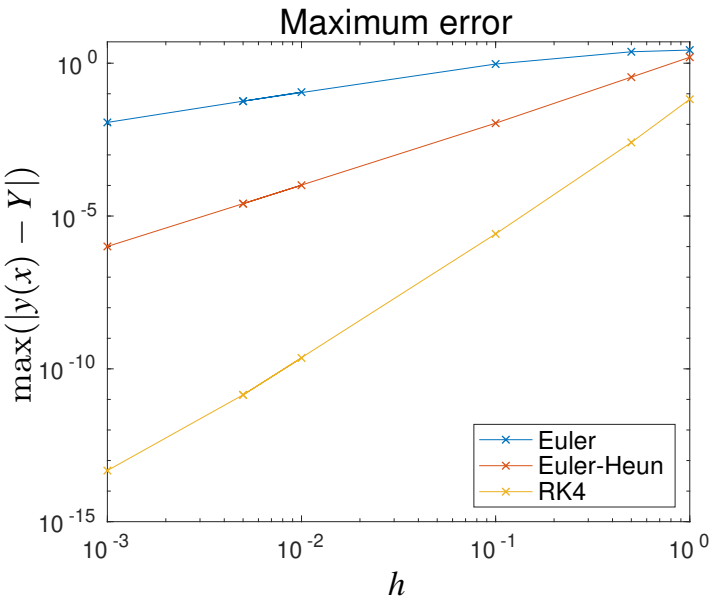
Example:

- consider the initial value problem

$$y'(x) = -\sin(x)y(x), \qquad x \in [-10,\ 10],$$
$$y(-10) = \mathbf{e}^{\cos(-10)}$$

- the analytical solution reads $y(x) = \mathbf{e}^{\cos(x)}$,
- application of different numerical methods and different step sizes,
- calculation of the maximal error

$$\max_{x\in[-10,10]} \left|y(x) - Y_h(x)\right|.$$



Outline

Adaptive Runge-Kutta methods and step size control

Accuracy and effort:

- accuracy of the approximations & required computation time depend on the method, the step size and the ode,
- small step sizes provide good approximations, but cause a high computational effort.

Adaptive Runge-Kutta methods:

- adjust the step size to the current behavior of the solution of the ode,
- estimate the suitable step size using a second method and compare the two approximations.

Procedure for a desired accuracy ε:

1. Compute approximations for the current iteration

$$Y = Y_i + h\Phi(x_i, Y_i, h_i),$$
$$\tilde{Y} = Y_i + h\tilde{\Phi}(x_i, Y_i, h_i)$$

   using the process functions Φ and $\tilde{\Phi}$ with the orders p and $p+1$,

2. Compute the value

$$\vartheta = \left(\frac{\varepsilon}{\|Y - \tilde{Y}\|} \right)^{\frac{1}{p+1}}.$$

3. Case differentiation

   $\vartheta < 1$: Discard the step, set $h := \max(\vartheta, 0.5)h$, execute step 1 again.

   $\vartheta \geq 1$: Accept the approximation and increase the step size for the next step

$$x_{i+1} = x_i + h, \quad Y_{i+1} = \tilde{Y}, \quad h := \min(2, \vartheta)h.$$

Embedded Runge-Kutta method DOPRI(4)5:

0							
$\frac{1}{5}$	$\frac{1}{5}$						
$\frac{3}{10}$	$\frac{3}{40}$	$\frac{9}{40}$					
$\frac{4}{5}$	$\frac{44}{45}$	$-\frac{56}{15}$	$\frac{32}{9}$				
$\frac{8}{9}$	$\frac{19372}{6561}$	$-\frac{25360}{2187}$	$\frac{64448}{6561}$	$-\frac{212}{729}$			
1	$\frac{9017}{3168}$	$-\frac{355}{33}$	$\frac{46732}{5247}$	$\frac{49}{176}$	$-\frac{5103}{18656}$		
1	$\frac{35}{384}$	0	$\frac{500}{1113}$	$\frac{125}{192}$	$-\frac{2187}{6784}$	$\frac{11}{84}$	
Y	$\frac{35}{384}$	0	$\frac{500}{1113}$	$\frac{125}{192}$	$-\frac{2187}{6784}$	$\frac{11}{84}$	
$\tilde{Y}$	$\frac{5179}{57600}$	0	$\frac{7571}{16695}$	$\frac{393}{640}$	$-\frac{92097}{339200}$	$\frac{187}{2100}$	$\frac{1}{40}$

Outline

Idea of multistep methods:

- use previous iterates directly to compute the next approximation,
- iteration of the form

$$Y_{k+1} = Y_k + h_k \Phi(x_{k+1}, x_k, \ldots, x_{k+1-m}, Y_{k+1}, Y_k, \ldots, Y_{k+1-m}),$$

- m-step linear multistep method (with $a_0 \neq 0$)

$$\sum_{j=0}^{m} a_j Y_{k+1-j} = h \sum_{j=0(1)}^{m} b_j f\left(x_{k+1-j}, Y_{k+1-j}\right).$$

Remarks:

- more degrees of freedom allow (theoretically) a higher accuracy,
- stability of multistep methods is not for sure
  $\rightarrow$ use only well-known methods as Adams, Nyström or BDF.

BDF-methods (most popular multistep methods

BDF-formulas for equidistant h of order 1 to 5:

$$
\begin{aligned}
Y_{k+1} &= Y_k + hf\left(x_{k+1}, Y_{k+1}\right) \\
\frac{3}{2}Y_{k+1} &= 2Y_k - \frac{1}{2}Y_{k-1} + hf\left(x_{k+1}, Y_{k+1}\right) \\
\frac{11}{6}Y_{k+1} &= 3Y_k - \frac{3}{2}Y_{k-1} + \frac{1}{3}Y_{k-2} + hf\left(x_{k+1}, Y_{k+1}\right) \\
\frac{25}{12}Y_{k+1} &= 4Y_k - 3Y_{k-1} + \frac{4}{3}Y_{k-2} - \frac{1}{4}Y_{k-3} + hf\left(x_{k+1}, Y_{k+1}\right) \\
\frac{137}{60}Y_{k+1} &= 5Y_k - 5Y_{k-1} + \frac{10}{3}Y_{k-2} - \frac{5}{4}Y_{k-3} + \frac{1}{5}Y_{k-4} + hf\left(x_{k+1}, Y_{k+1}\right).
\end{aligned}
$$

Remarks:

- BDF formulas define implicit methods,
- up to $m = 6$ they are very effective for stiff problems.

Example

Cyclic example:

- three-body problem,
- two bodies of masses $1-\mu$ (Earth) and μ (Moon) move circularly in a plane,
- a third body (satellite), of negligible mass, also moves in the plane
- equations of motion for this body are

$$\ddot{y}_1 = y_1 + 2\dot{y}_2 - \mu' \frac{y_1+\mu}{D_1} - \mu \frac{y_1-\mu'}{D_2}$$
$$\ddot{y}_2 = y_2 - 2\dot{y}_1 - \mu' \frac{y_2}{D_1} - \mu \frac{y_2}{D_2}$$
$$D_1 = ((y_1+\mu)^2 + y_2^2)^{3/2}, \quad D_2 = ((y_1-\mu')^2 + y_2^2)^{3/2}$$
$$\mu = 0.012277472, \quad \mu' = 1-\mu$$

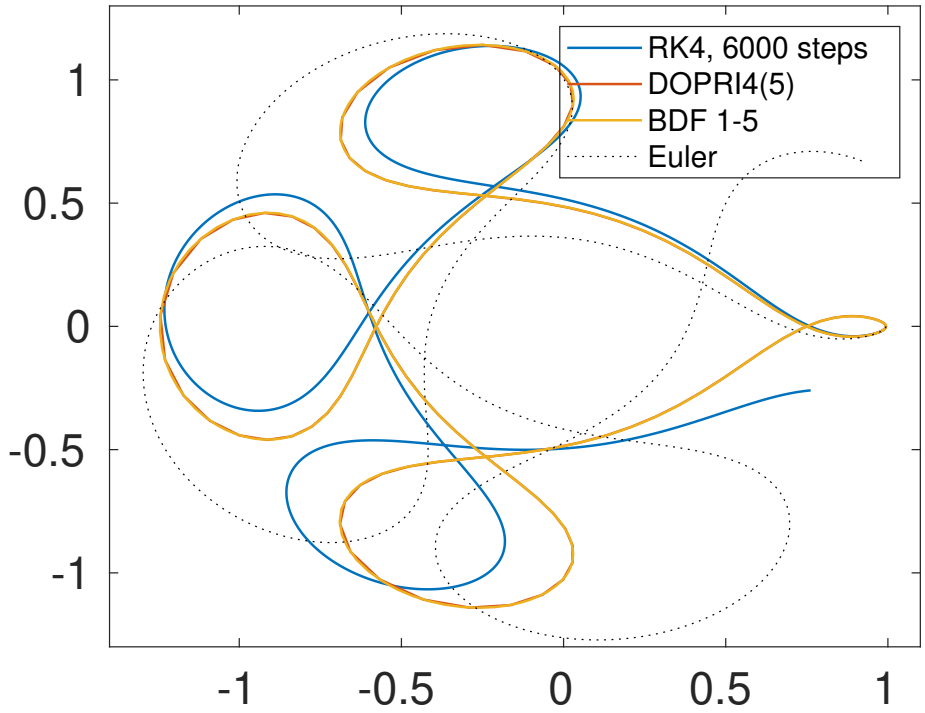
- interval $[t_0, t_1] = [0, 17.065216560157963]$, parameters

$$y_1(0) = 0.994,\ \dot{y}_1(0) = y_2(0) = 0,\ \dot{y}_2(0) = -2.001585106379082 5.$$

Example

Applied methods:

- Euler's method using $100\,000$ steps,
- Runge-Kutta using $6\,000$ steps,
- DOPRI4(5) with step size control (finally uses 212 steps),
- BDF-1 and BDF-5 with step size control (ode15s in matlab,finally uses 468 steps).



Outline

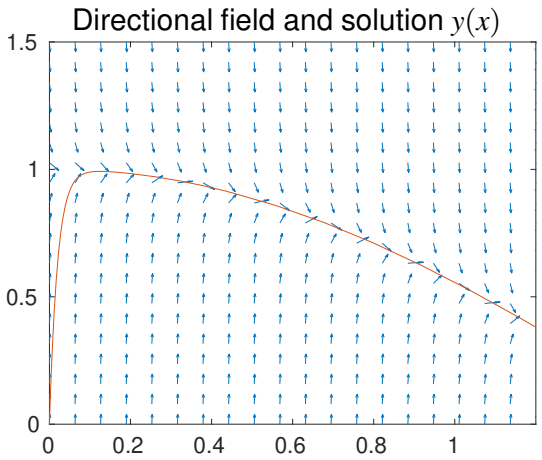
Stiff problems

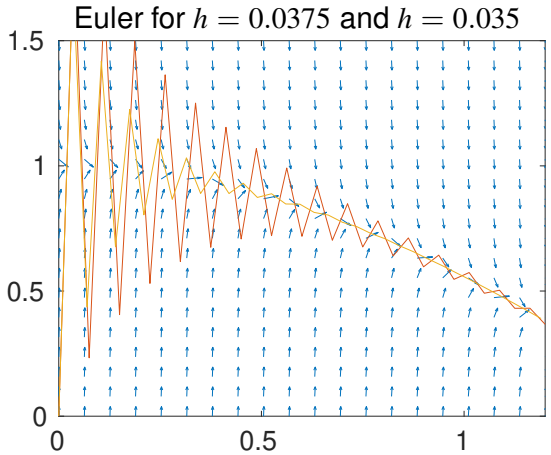
Curtiss, Hirschfelder 1952:

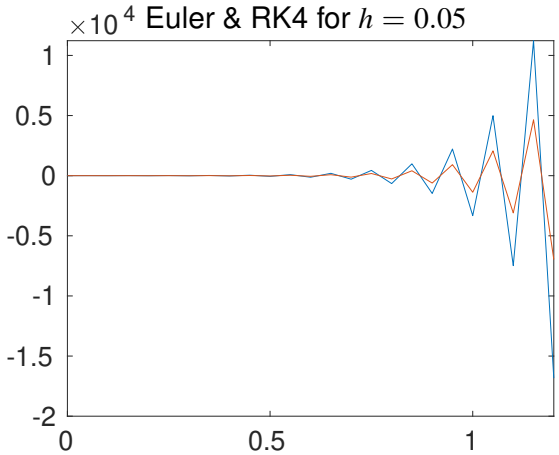
- initial value problem

$$y'(x) = -50(y(x) - \cos(x)), \qquad y(0) = 0,$$

- solution by Euler's method and RK 4 using different step sizes.







$\rightsquigarrow$ Analysis of the behavior.

Model problem:

$$y' = \lambda y, \quad y(0) = y_0$$

with the solution

$$y(x) = y_0 \mathbf{e}^{\lambda x}.$$

Boundedness:

$$\mathrm{Re}(\lambda) \leq 0 \quad \Rightarrow \quad |y(x)| = \Big| y_0 \big(\underbrace{\mathbf{e}^{(\mathrm{Re}(\lambda))x}}_{\leq 1} (\cos(x\mathrm{Im}(\lambda)) + \mathbf{i} \sin(x\mathrm{Im}(\lambda)) \big) \Big| \leq |y_0|.$$

Definition 8.6

A numerical method is called A-stable if its application to the model problem with $\mathrm{Re}(\lambda) \leq 0$ *for any step size* $h > 0$ *always yields a sequence of iterates* Y_k, $k = 1, 2, \ldots$ *which are bounded by* $|y_0|$.

Remark

Any explicit s-step Kutta method applied to the model problem can be written in the form

$$Y_{i+1} = p(h\lambda)Y_i(*)$$

with a polynomial $p(\mu)$ of degree s.

Definition 8.7

The polynomial $p(\mu)$ from $(*)$ is called stability function.

Stability function for Euler's methods:

$$p(\mu) = 1 + \mu.$$

Definition 8.8

Let $p(\mu)$ be the stability function of the Runge-Kutta method. For constant step size h is

$$S = \{\mu \in \mathbb{C} \mid |p(\mu)| \leq 1\}$$

the stability domain of the Runge-Kutta method.

Conclusion:

1. For constant h the solutions remain bounded if and only if $h\lambda \in S$.
2. If the stability domain contains the left half plane

$$\{\mu \in \mathbb{C} \mid \mathrm{Re}(\mu) \leq 0\},$$

then the method is according to definition 8.6 A-stable.

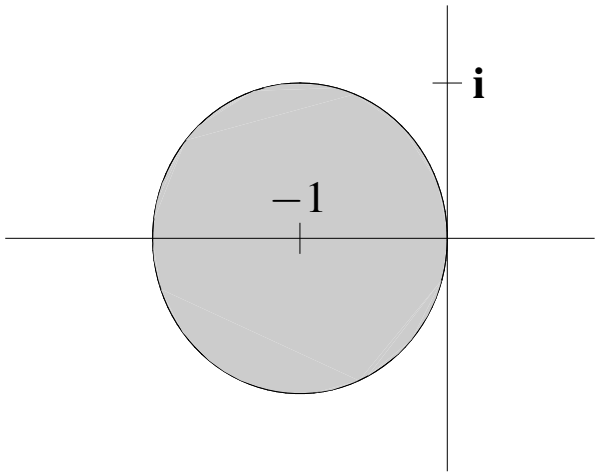
Stability domain Euler's method

Euler (explicit):

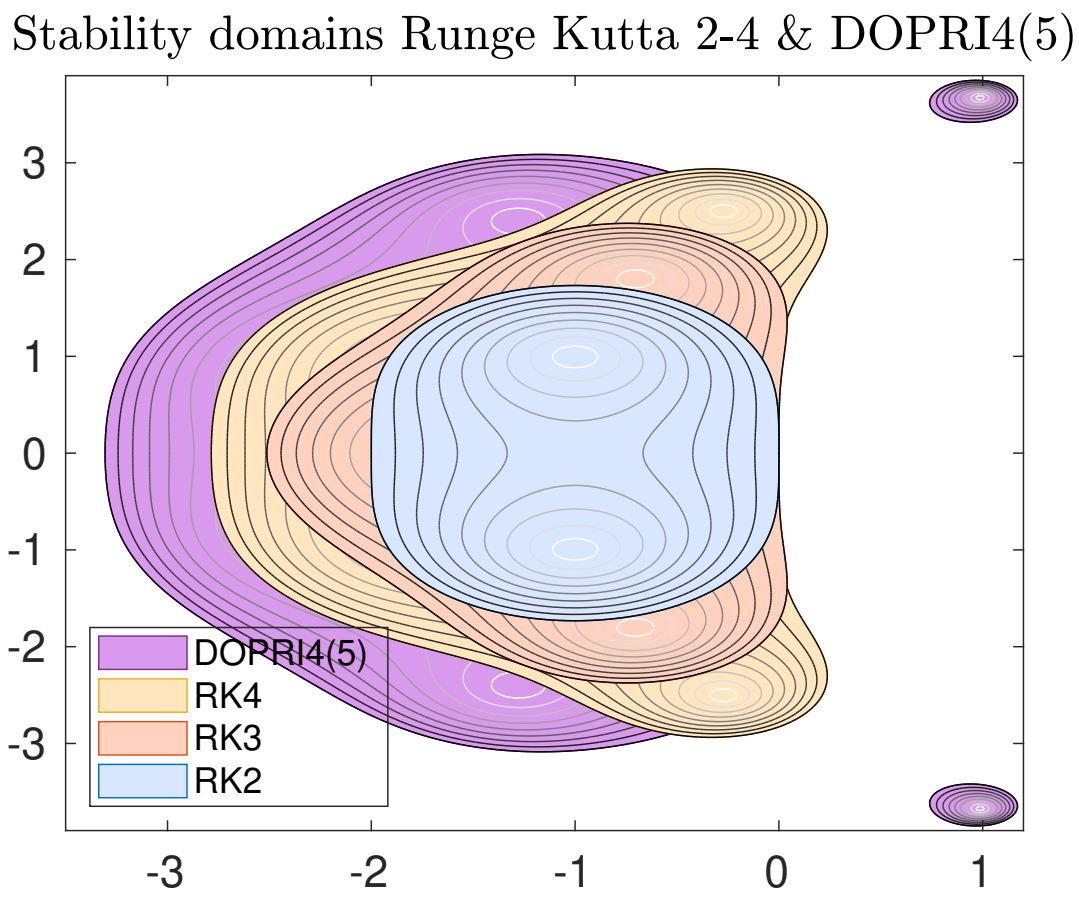
- the stability function is $p(\mu) = 1 + \mu$,
- the stability domain is

$$S = \{\mu \in \mathbb{C} \mid |p(\mu)| \leq 1\}$$

and thus a circle with radius 1 and center -1.



Other stability domains



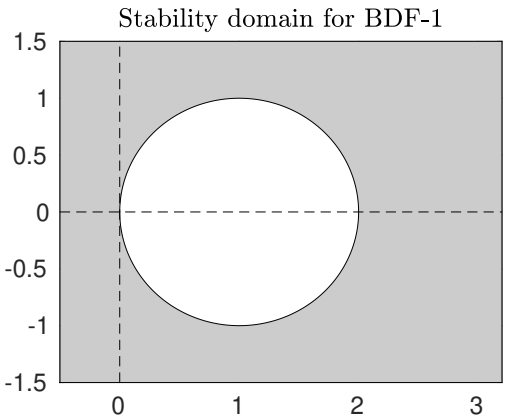
Theorem 8.9

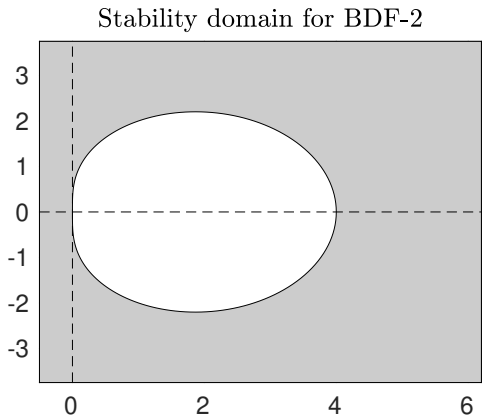
No explicit Runge-Kutta procedure is A-stable.

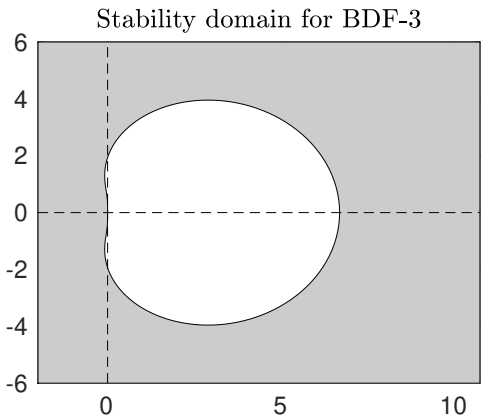
Stability domains for BDF

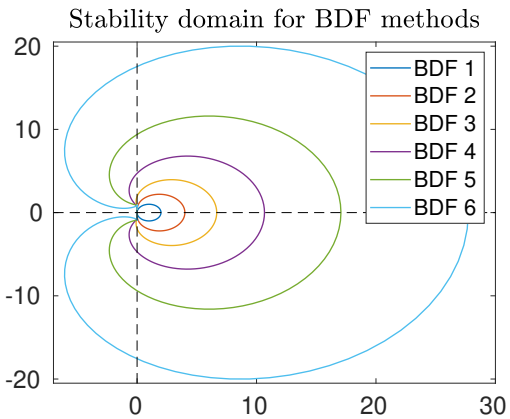
Remarks:

- calculation of the stability domains for multistep methods is quite complex,
- only BDF-1 and BDF-2 are A-stable, but BDF-3 to BDF-6 have large and unbounded(!) stability domains, well suited for solving stiff problems.









Outline

Example (Implicit Euler method)

- *iteration*

$$Y_{i+1} = Y_i + hf(x_{i+1}, Y_{i+1}),$$

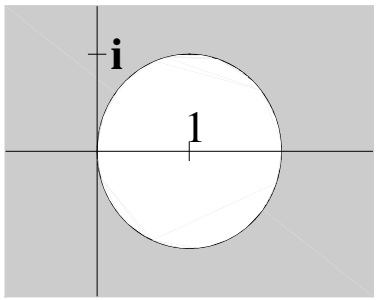
- *application for the model problem* $y' = \lambda y$

$$Y_{i+1} = Y_i + h\lambda Y_{i+1} \quad \Leftrightarrow \quad (1 - h\lambda)Y_{i+1} = Y_i \quad \Leftrightarrow \quad Y_{i+1} = \frac{1}{1 - h\lambda} Y_i,$$

- *boundedness of the iterates*

$$\left| \frac{1}{1 - h\lambda} \right| \leq 1 \quad \Leftrightarrow \quad 1 \leq |1 - h\lambda|,$$

- *satisfied with* $h > 0$ *as well as* $\mathrm{Re}(\lambda) \leq 0 \quad \Rightarrow \quad A$*-stable.*



Definition 8.10

Let $a_{jl}, b_j \in \mathbb{R}$ for $j, l = 1, \ldots, s$ and $c_j = \sum_{l=1}^{s} a_{jl}$. Let the quantities $y_1, \ldots, y_s$ be solutions of the (i. a. non-linear) system of equations

$$y_j = Y_i + h \sum_{l=1}^{s} a_{jl} f(x + c_l h, y_l)$$

for $j = 1, \ldots, s$.
Then the new iterate using the s-step implicit Runge-Kutta method reads

$$Y_{i+1} = Y_i + h_i \sum_{j=1}^{s} b_j f(x + c_j h, y_j).$$

Implicit Runge-Kutta methods

Butcher tableau:

$$\begin{array}{c|ccc} c_1 & a_{11} & \cdots & a_{1s} \\ \vdots & \vdots & \ddots & \vdots \\ c_s & a_{s1} & \cdots & a_{ss} \\ \hline & b_1 & \cdots & b_s \end{array}$$

Remarks:

- Many, but not all, implicit Runge-Kutta methods are A-stable.
- This advantage comes with the price of solving a nonlinear system of equations.

Some A-stable methods:

1. The implicit Euler method has the Butcher-tableau

$$\begin{array}{c|c} 1 & 1 \\ \hline & 1 \end{array}.$$

2. The Butcher-tableau

$$\begin{array}{c|cc} 0 & 0 & 0 \\ 1 & \frac{1}{2} & \frac{1}{2} \\ \hline & \frac{1}{2} & \frac{1}{2} \end{array}$$

defines the Crank-Nicolson-Verfahren as

$$Y_{i+1} = Y_i + \frac{h}{2}\big(f(x_i, Y_i) + f(x_{i+1}, Y_{i+1})\big).$$

More A-stable methods:

3. The Radau-IA method with $s = 2$ stages has the Butcher-tableau

$$\begin{array}{c|cc} 0 & \frac{1}{4} & -\frac{1}{4} \\ \frac{2}{3} & \frac{1}{4} & \frac{5}{12} \\ \hline & \frac{1}{4} & \frac{3}{4} \end{array}$$

   and is of order 3.

4. The Radau-IIA method with $s = 2$ stages has the Butcher-tableau

$$\begin{array}{c|cc} \frac{1}{3} & \frac{5}{12} & -\frac{1}{12} \\ 1 & \frac{3}{4} & \frac{1}{4} \\ \hline & \frac{3}{4} & \frac{1}{4} \end{array}$$

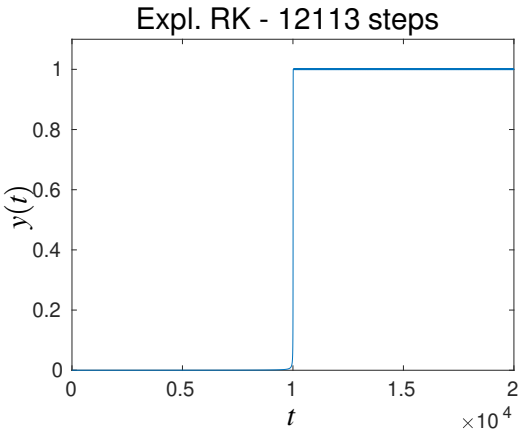
   and is also of order 3.

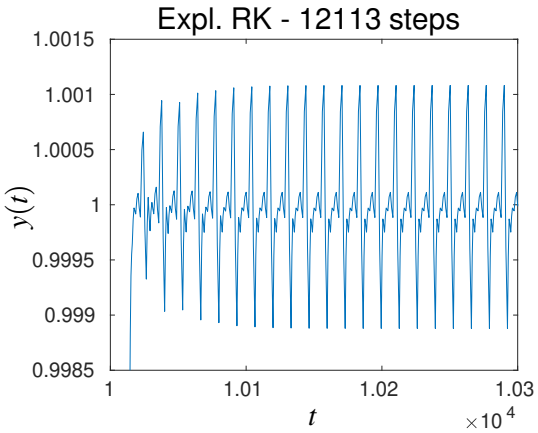
Example:

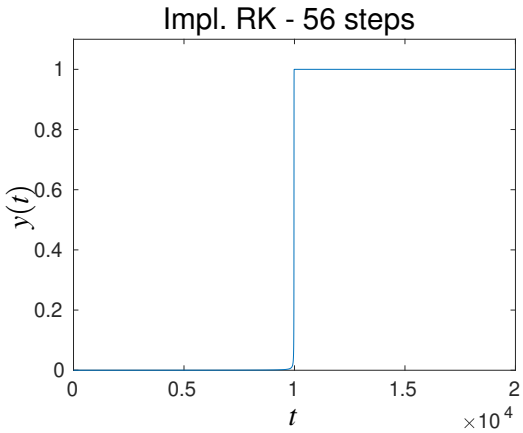
- simplified simulation of flame propagation in gases,
- initial value problem with $\varepsilon > 0$

$$y'(x) = y(x)^2 - y(x)^3, \qquad 0 \le x \le \frac{2}{\varepsilon},$$
$$y(0) = \varepsilon.$$

- numerical results for $\varepsilon = 10^{-4}$







Outline

Boundary value problem of second order

Scalar linear boundary value problem of second order: (for $\lambda \geq 0$)

$$\begin{aligned} -y''(x) + \lambda y(x) &= f(x_i), \quad a \leq x \leq b, \\ y(a) &= \alpha, \qquad y(b) = \beta \end{aligned}$$

Segmentation:

$$a = x_0 < x_1 < \ldots < x_N < x_{N+1} = b, \qquad x_j = a + jh, \qquad h = \frac{b-a}{m+1}.$$

Boundary points: (x_0, α) and (x_{N+1}, β).

Inner knots: $x_1, \ldots, x_N$, to these knots we compute approximations $Y_i \approx y(x_i)$.

Finite differences:

$$-y''(x_i) \approx \frac{-Y_{i-1} + 2Y_i - Y_{i+1}}{h^2}.$$

Equations:

$$\begin{aligned}
i = 1: &\quad \frac{-\alpha + 2Y_1 - Y_2}{h^2} + \lambda Y_1 = f(x_1), \\
2 \le i \le N-1: &\quad \frac{-Y_{i-1} + 2Y_i - Y_{i+1}}{h^2} + \lambda Y_i = f(x_i), \\
i = N: &\quad \frac{-Y_{N-1} + 2Y_N - \beta}{h^2} + \lambda Y_N = f(x_N).
\end{aligned}$$

Boundary value problem of second order

System of linear equations:

$$\frac{1}{h^2}\begin{pmatrix} 2+\lambda h^2 & -1 & & & \\ -1 & 2+\lambda h^2 & -1 & & \\ & \ddots & \ddots & \ddots & \\ & & -1 & 2+\lambda h^2 & -1 \\ & & & -1 & 2+\lambda h^2 \end{pmatrix} \begin{pmatrix} Y_1 \\ Y_2 \\ \vdots \\ Y_N \end{pmatrix} = \begin{pmatrix} \alpha/h^2 + f(x_1) \\ f(x_2) \\ \vdots \\ f(x_{N-1}) \\ \beta/h^2 + f(x_N) \end{pmatrix}.$$

Connections:

- Sec. 2: solution of this system of equations for $\alpha^2 + \beta^2 > 0$, the matrix is for $\lambda > 0$ strictly diagonally dominant and iterative methods converge, see theorem 2.11,
- Sec. 4 and some problems from the tutorials: Calculation/approximation of the Eigenvalues and vectors for $\alpha = \beta = 0$ and $u'' = -\lambda u$.

Outline

Literature


Atkinson, K.: Elementary Numerical Analysis, John Wiley & Sons, 1993.


Burden, R., Faries, J. D.: Numerical Analysis, Brooks Cole Publishing Company, 1997.


Friedmann, M., Kandel, A.: Fundamentals of Computer Numerical Analysis, CRC Press, 1993.


Golub, G. H., Ortega, J. M.: Scientific Computing and Differential Equations: An Introduction to Numerical Methods, Academic Press, 1992.


Kharab, A., Guenther, R. B.: An Introduction to Numerical Methods: A Matlab Approach, Chapman & Hall / CRC, 2002.


Quarteroni, A., Sacco, R., Saleri, F.: Numerical Mathematics, Springer, 2007.


Stoer, J., Bulirsch, R.: Introduction to Numerical Analysis, Springer, 2002.


Süli, E., Mayers, D.: An Introduction to Numerical Analysis, Cambridge University Press, 2003.